IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Vol. 9, Issue 4, April 2021

DOI 10.17148/IJIREEICE.2021.9424

Performance Analysis of Montgomery Modular Multiplier Using Karatsuba Algorithm

Ashly George[1], Ajeesh S.[2], Sindhu T.V.[3]

P.G Scholar, M.Tech in VLSI Design, IES College of Engineering Chittilappilly[1]

Assistant Professor, ECE, IES College of Engineering Chittilappilly [2,3]

Abstract: Montgomery Modular Multiplication is a method for performing fast modular multiplication. Modular multiplication is important in modern cryptography, as it is the basic operation that needs to be performed in many public key cryptosystems such as RSA. Rivest, Shamir, Adleman (RSA) algorithm can be used in cryptography to send confidential messages in secure manner. RSA cryptosystem is based on modular multiplication to factorize the large integers. Montgomery modular multiplication can be used as modular multiplication because it is one of the efficient and fastest modular multiplication. Montgomery algorithm is based on add and shift operation. Hardware is described for implementing the fast modular multiplication algorithm of P. L. Montgomery. The Karatsuba-Ofman multiplier replaces a multiplication by three ones of half-length operands which are performed in parallel. Large integer multiplication is the critical operation to design a modular multiplier. Karatsuba algorithm(KO algorithm) to split the operands into two parts is generally used to design a large integer multiplier. The most latest approach of Montgomery modular multiplication using KO algorithm, with which hardware resources can be reduced. Compared with other designs, the proposed design shows a better performance in terms of delay, area and power, and there is a significant improvement in latency and trade-off between area and performance.

Keywords: Montgomery modular multiplier, Large integer multiplication, Karatsuba algorithm.

I. INTRODUCTION

Cryptography plays an important role in information security. In public-key cryptographic algorithms, such as RSA, Digital Signature Algorithm(DSA) and Elliptical Curve Cryptography(ECC), it is inevitable to perform modular multiplication. Therefore, a high-performance cryptographic system depends on the design of modular multiplication. There are two algorithms widely used to design a modular multiplier, which are Montgomery modular multiplication[1] and the Barret modular multiplication[2].

Arithmetic operations like addition, multiplication are commonly used in many of the data processing applications. An operation like division is less frequently used but it is still important. There are a few applications that require only the remainder from the division process. The aim of the application might be to find the last two digits of a number after multiplication or exponentiation operation or find if a particular number is divisible by a particular number. For example, the last two digits of 99^{99} or find if 187596 is divisible by 82 and so on. There are also applications like cryptography where the inputs are to be wrapped around a particular input which is the key. In these instances modular arithmetic operations are employed. The other significant applications of modular operations are hashing, generation of random numbers, International Standard Book Number (ISBN) which is a unique numeric book identifier constitutes the group, publication, title, parity check bit and music. Digital information is stored in the form of binary digits and these digits are segmented into blocks. Each of these blocks are appended with parity bit. Modular Multiplication (MM) with large integers is a time consuming operation in many public-key cryptosystems [1]. Therefore, many algorithms have been presented to carry out MM more quickly and Montgomery's algorithm is one of them. Montgomery's algorithm determines the quotient only depending on the least significant digit of operands [2]. It replaces the complicated division in MM with a series of shifting modular additions.

The Montgomery algorithm for modular multiplication is considered to be the fastest algorithm to compute x*y mod n in computers when the values of x, y, and n are large. Among the different ways of performing modular multiplication, Montgomery modular multiplication [4] introduced by Peter Montgomery is the widely used algorithm. It replaces the trial divisions by modulus with a series of additions and divisions by two suitable for VLSI implementations [4-5]. There are several techniques proposed in literature that avoid the carry propagation involved in addition circuitry of Montgomery algorithm which is a key factor in determining the overall performance.

There are two main categories of hardware-based implementation of Montgomery modular multiplier: bit-wise and word-wise. Systolic array, parallel array and multiplex array are three typical bit-wise implementations. They require *O*(N^2) additions to accomplish N-bit modular multiplier instead of N-bit multiplication, where N is the bit-width of operand. The word-wise modular multipliers use the multiple-precision multiplication algorithm instead. The complexity are gradually decreased toward the theoretical minimum value *O(n logn),* where n is the bit-width of multiplicand.

II. DESIGN AND IMPLEMENTATION OF MONTGOMERY MODULAR MULTIPLIER

A. *Architecture Of Montgomery Modular Multiplier*

Montgomery modular multiplication is described by P. Montgomery in 1985[2]. The dominance of this algorithm is to division with shift and addition. The concrete description is illustrated in Algorithm1. Compared with division, the hardware implementations of shift and addition are rather easier and less costly.

```
Algorithm 1:Montgoemry modular multiplication
Input    $M$ is odd,$1 < M < 2^N$,$0 \leq A, B \leq 2^N$
         and $rr^{-1} - MM^{-1} = 1$,$r = 2^N$
Output   $R = ABr^{-1} mod \ (M)$
step1    $T = A * B$, $T_L = T mod \ r$
step2    $m = T_L * N^{-1} mod \ r$
step3    $R = (T + m * M)/r$
step4    if$R \geq M$,return $R - M$
         else return$R$
```

B. *Improved Montgomery Modular Multiplication*

From Algorithm 1, we can find there is a final subtraction. C. D. Walter [9] proposes a method to omit it by choosing another set parameters. The improved algorithm is presented in the following figure.

```
Algorithm 2: Improved Montgoemry modular multiplication
Input    $M$ is odd,$1 < M < 2^N$,$0 \leq A, B \leq 2M$
         and $M' = r - M^{-1} mod \ r$,$r = 2^{N+2}$
Output   $R = ABr^{-1} mod \ (M)$
step1    $T = A * B$, $T_L = T mod \ r$
step2    $m = T_L * M' mod \ r$
step3    $R = (T + m * M)/r$
```

C. Design and Implementation of Multiplier

Multipliers are the fundamental and necessary building blocks of many high performance systems. Multiplication is commonly used operation which is now implemented in many processors, attaining security objectives of data integrity and confidentiality for high speed network applications. The speed of the multiplier determines the speed of the system; hence the speed of the multiplier has to be increased. The performance of the system relay upon the multiplier's speed which is optimized by the proposed multiplier. Low power consumption is also a critical issue in multiplier design. The main contributors to the power consumption of digital circuits are adders, multipliers, comparators and shifters. Power reduction is now becoming very crucial for implementation of VLSI designs. With increasing importance of power reduction, it is becoming essential to evaluate different data path architectures from the point of view of both delay and power.

KOA is the fast multiplication algorithm which helps to increase the speed of the multiplier, and is one of the fastest methods to multiply long integers. The classical multiplication algorithm multiplies every digit of a multiplicand by every digit of the multiplier and adds the result to the partial product. KOA has less complexity than classical schoolbook method and thus it multiplies large numbers faster. The Karatsuba Algorithm is more efficient for multiplication of large numbers. It uses Divide and Conquer strategy, divides a multiplication operation into smaller and smaller products recursively, and then computes the result of the multiplication, benefiting from the results of these smaller products. Thus it saves coefficient multiplications at the cost of extra additions as compared to the ordinary multiplication method. The Karatsuba Algorithm is more efficient for multiplication of large numbers and the number of multiplication is reduced by increasing the number of additions, thus reducing the overall cost of the hardware design.

Multiplication can be done by using some smaller multiplication, addition and some shifting operations. Thus speed of the multiplier can be increased by using high speed adder. This work presents the design of a fast multiplier using the Karatsuba Algorithm to multiply two numbers using the technique of integer multiplication.

C. *Karatsuba Algorithm*

The Karatsuba algorithm is a fast multiplication algorithm. It was first implemented by Anatolii Alexeevitch Karatsuba in 1960 and published in 1962[4]. It scales down the multiplication of two n-digit numbers to two single-digit multiplications. Multiplying two polynomials efficiently is a crucial point in different applications such as signal processing, cryptography and coding theory. The Karatsuba Algorithm (KA) reduces coefficient multiplications at the cost of extra additions compared to the schoolbook or ordinary multiplication method [2, 3]. We consider the KA to be efficient if the total cost of using it is less than the cost of the ordinary method.

In KA number of multiplication is reduced by increasing the number of additions, thus reducing the overall cost of the hardware design. Thus, the efficient software implementations of the multiplication operation in the finite fields are desired in cryptographic applications, particularly in the elliptic curve cryptography [9]. Several new methods for basic arithmetic operations in the finite fields, suitable for software implementations have been recently developed [2]. Among these algorithms, the Karatsuba-Ofman Algorithm (KOA) [5,6] has a special place since it is the only practical algorithm which is asymptotically faster than the standard methods for the cryptographic applications in which the numbers in the range 160 to 1024 bits are used. However, most of these implementations report speedup only for higher bit lengths, i.e., bit lengths of 1024 or more. Multiplication can be done by using some smaller multiplication, addition and some shifting operations. Thus speed of the multiplier can be increased by using high speed adder. KOA could be used in recursive mode and applied for any degree m, utilizing the scheme will yield more gate savings with longer delay. It is used for polynomial multiplication. A main advantage of KOA approach could be contributed to its recursive possibility.

According to Ruirui Liu and Shuguo Li, KO algorithm is to split 2N-bit multiplier and multiplicand in half,

$$A = a_1 2^N + a_0 \qquad (1)$$
$$B = b_1 2^N + b_0 \qquad (2)$$

Then the product of AB can be computed as:
$$P = A*B = a_1*b_1 2^{2N} + (a_1*b_0 + a_0*b_1)2^N + a_0*b_0 \qquad (3)$$

Let p_0,p_1 and p_{01} represent a_0*b_0, a_1*b_1 and $(a_1 + a_0)*(b_1+b_0)$ respectively, which can be written as:

$$p_0 = a_0*b_0 \qquad (4)$$
$$p_1 = a_1*b_1 \qquad (5)$$
$$p_{01} = (a_1+a_0)*(b_1+b_0) \qquad (6)$$

Then according to KO algorithm, P can be denoted by p_0,p_1 and p_{01} as the following expression:

$$P = a_1*b_1 2^{2N} + (a_1*b_0+a_0*b_1)2^N + a_0*b_0 = p_1 2^{2N} + (p_{01}-p_0-p_1)2^N + p_0 \qquad (7)$$

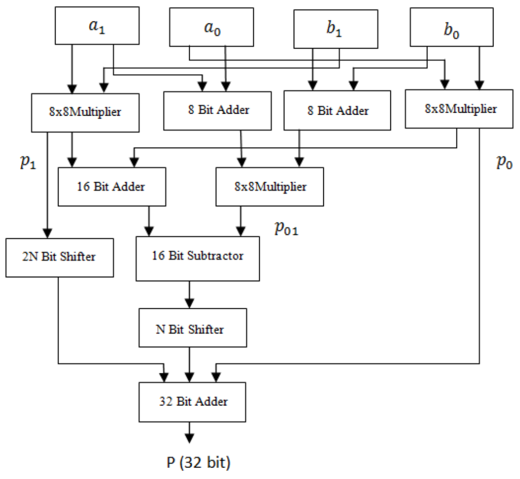
In this way, the original four N-bit multipliers can be reduced to two N- bit and one N+1-bit multipliers only at the cost of adders, which can cut down much hardware cost when N is large.

D. *Architecture of 32 Bit Karatsuba Multiplier for Integer Multiplication*

The building blocks for a 32 bit Karatsuba Multiplier are adder, subtractor, multiplier and N-bit and 2N-bit shifters.

Vol. 9, Issue 4, April 2021

DOI 10.17148/IJIREEICE.2021.9424



III. PROPOSED METHOD

In the recent decades, the mobile electronic devices is exponentially increased which creates an urge to design highly effective VLSI structures. The operations in the devices necessitate to be computed by low - power, area efficient designs which operate at higher speed. Addition is the commonly used arithmetic operation; and adder is the basic arithmetic element of the processor. In Processors adder is an important element. As such, extensive research continues to be focused on improving the power-delay performance of the adder. In VLSI implementations, parallel-prefix adders are known to have the best performance. Reconfigurable Field Programmable Gate Arrays (FPGAs) has been gaining in popularity in recent years because it offers improved performance in terms of speed and power over DSP-based and microprocessor-based solutions for many practical designs and a significant reduction in development time and cost over Application Specific Integrated Circuit (ASIC) designs. Fast and accurate operation of digital system depends on the performance of adders. Hence improving the performance of adder is the main area of research in system design. Arithmetic (such as addition, subtraction, multiplication and division) performed in a program; additions are required to increment the program counter and to calculate the effective address. Show that in a prototypical RISC machine (DLX) 72of the instructions perform additions (or subtractions) in the data path. Over the last decade many different adder architectures were studied and proposed to speed up the binary additions.

Basically used adder is Ripple Carry Adder. Arithmetic operations like addition, subtraction, multiplication, division are basic operations to be implemented in digital computers using basic gates like AND, OR, NOR, NAND etc. Among all the arithmetic operations if we can implement addition then it is easy to perform multiplication (by repeated addition), subtraction (by negating one operand) or division (repeated subtraction). Half Adders can be used to add two one bit binary numbers and Full adders to add two three bit numbers. The disadvantage of the ripple-carry adder is that it can get very slow when one needs to add many bits.

To overcome the limitations of RCA, a faster and efficient Carry Look - Ahead Adder is used. Proposed method design utilizes the CLA in every phase of operation. The method is implemented and the results are compared with RCA in terms of area, delay and speed. CLA is advantageous and has several benefits like energy efficient, compact size and found to be faster. Carry Look Ahead Adder is an improved version of the ripple carry adder. It generates the carry-in of each full adder simultaneously without causing any delay. The time complexity of carry look ahead adder = (log n).

In parallel adders, carry output of each full adder is given as a carry input to the next higher-order state. Hence, these adders it is not possible to produce carry and sum outputs of any state unless a carry input is available for that state. So, for computation to occur, the circuit has to wait until the carry bit propagated to all states. This induces carry propagation delay in the circuit. In this adder, the propagation delay is reduced. The carry output at any stage is dependent only on the initial carry bit of the beginning stage. Using this adder it is possible to calculate the intermediate results. This adder is the fastest adder used for computation.

In this adder, the propagation delay is reduced. The carry output at any stage is dependent only on the initial carry bit of the beginning stage. Using this adder it is possible to calculate the intermediate results. This adder is the fastest adder used for computation. High-speed Carry Look-ahead Adders are used as implemented as IC’s. Hence, it is easy to embed the adder in circuits. By combining two or more adders calculations of higher bit Boolean functions can be done

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

Vol. 9, Issue 4, April 2021

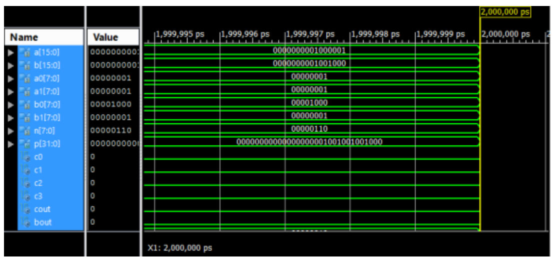
DOI 10.17148/IJIREEICE.2021.9424

easily. Here the increase in the number of gates is also moderate when used for higher bits. For this Adder there is a trade-off between area and speed. When used for higher bit calculations, it provides high speed but the complexity of the circuit is also increased thereby increasing the area occupied by the circuit. This adder is usually implemented as 4-bit modules which are cascaded together when used for higher calculations. This adder is costlier compared to other adders.

IV. EXPERIMENTAL RESULTS

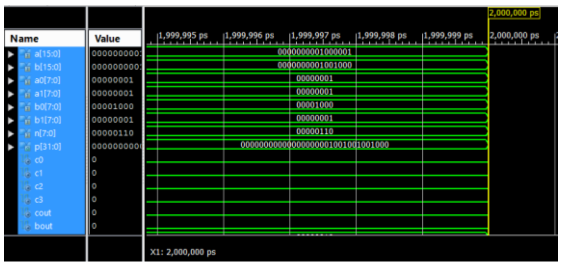
The Montgomery Modular Multiplier is simulated on Xilinx Vivado15.4 Software tool on Virtex-6 FPGA platform. Xilinx Vivado Design Suite is a software suite produced by Xilinx for synthesis and analysis of HDL designs, superseding Xilinx ISE with additional features for system on a chip development and high-level synthesis. Like the later versions of ISE, Vivado includes the in-built logic simulator ISIM. Xilinx Virtex-6 FPGA connectivity kit is a development platform using the Virtex-6 family for high-bandwidth and high-performance applications in multiple market segments. FPGAs are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. In addition to configurable FPGA logic, Virtex FPGAs include fixed-function hardware for multipliers, memories, microprocessor cores, FIFO and ECC logic, DSP blocks, PCI Express controllers, Ethernet MAC blocks, and high-speed serial transceivers.

The output waveforms of all the designs i.e, Montgomery Multiplier using Karatsuba Algorithm with Ripple Carry Adder and Carry Look-Ahead Adder as components which are explained in the design implementation are shown in figures Fig.1 and Fig.2 respectively.



```
Name      Value        2,000,000 ps
                       1,999,995 ps  1,999,996 ps  1,999,997 ps  1,999,998 ps  1,999,999 ps  2,000,000 ps
a[15:0]   000000000:   0000000001000001
b[15:0]   000000000:   0000000001001000
a0[7:0]   00000001     00000001
a1[7:0]   00000001     00000001
b0[7:0]   00001000     00001000
b1[7:0]   00000001     00000001
n[7:0]    00000110     00000110
p[31:0]   0000000000   00000000000000000001001001001000
c0        0
c1        0
c2        0
c3        0
cout      0
bout      0
X1: 2,000,000 ps
```

Fig. 1 Waveform of MM Multiplier using RCA



```
Name      Value        2,000,000 ps
                       1,999,995 ps  1,999,996 ps  1,999,997 ps  1,999,998 ps  1,999,999 ps  2,000,000 ps
a[15:0]   000000000:   0000000001000001
b[15:0]   000000000:   0000000001001000
a0[7:0]   00000001     00000001
a1[7:0]   00000001     00000001
b0[7:0]   00001000     00001000
b1[7:0]   00000001     00000001
n[7:0]    00000110     00000110
p[31:0]   0000000000   00000000000000000001001001001000
c0        0
c1        0
c2        0
c3        0
cout      0
bout      0
X1: 2,000,000 ps
```

Fig. 2 Waveform of MM Multiplier using CLA

TABLE I PERFORMANCE ANALYSIS OF MONTGOMERY MODULAR MULTIPLIER

Component	Power	No. of LUTs	Junction Temperature(C)
RCA	1.293	453	33.3
CLA	0.177	351	25.3

A comparative analysis of these two methods in terms of delay, area and power has been done in Table I and Table II defines the logic level utilization in which the availability and utilizations of all resources (LUT and IO) used in the designing of the multipliers are mentioned.

TABLE II LOGIC LEVEL UTILIZATION OF MONTGOMERY MODULAR MULTIPLIER

Component	Resource	Utilization	Available	Utilization Percentage
RCA	LUT	453	46560	0
	IO	64	240	24
CLA	LUT	351	20400	1
	IO	64	600	10

A comparative analysis is done between these two methods in terms of delay, area and power. Power analysis is done with X power Analyzer Tool which shows a power consumption of 1.293 µW for Montgomery Modular Multiplier

Vol. 9, Issue 4, April 2021

DOI 10.17148/IJIREEICE.2021.9424

using RCA and 0.177 µW for Montgomery Modular Multiplier using CLA. Area analysis is done by comparing the availability and utilization of the resources used while designing the multiplier. It has been found that multiplier CLA is more efficient than the multiplier using RCA in terms of delay, area and power.

TABLE III COMPARISON BETWEEN MODULAR MULTIPLIER USING RCA AND CLA

		MM Multiplier - RCA		MM Multiplier - CLA	
Logic Utilization	Available	Used	Utilization	Used	Utilization
Number of Slices	46560	453	0	351	1%
Number of 4 Input LUTs	46560	453	0	0	0%
Number of Bonded IOBs	240	64	26	24	10%
Total Time Delay		22.481ns		17.683ns	

V. CONCLUSION

In this work, main focus is on designing Montgomery Modular Multiplier using Karatsuba-Ofman Algorithm in order to minimize the delay, area and power. In this algorithm there are multiplication, addition and shift operations, among which multiplication consumes a lot of hardware resources. The multiplier is designed using both Ripple Carry Adder and Carry Look-Ahead Adder as components. The multiplier is compared in terms of delay, power and number of LUTs. It has been found that the Carry Look-Ahead method is more efficient in terms of delay, power and area when compared to Ripple Carry method. The delay for the multiplier when using Ripple Carry method is 22.481ns and when using Carry Look-Ahead method, it is 17.683ns. As far as area is concerned utilization LUTs in Ripple Carry method is 453 and 351 in the case of Carry Look-Ahead method. The power for Ripple Carry method is 1.293µW and it is reduced to 0.177µW in the case of Carry Look-Ahead method.

Carry Look-Ahead adder is a fast adder. It is an improved version of the Ripple Carry adder. In this adder, the propagation delay is reduced. The carry output at any stage is dependent only on the initial carry bit of the beginning stage. Using this adder it is possible to calculate the intermediate results. This adder is the fastest adder used for computation. It improves the speed. It calculates one or more carry bits before the sum, this reduces the wait time to calculate the result of the bits which have a larger value. This is the addition technique that eliminates the problem due to interstage carry delay. The look ahead carry addition will therefore speed up the addition and hence reduce the delay for the multiplier. Hence the performance of Montgomery Modular Multiplier using KO algorithm can be improved in terms of delay, area and power, by replacing the Carry Look-Ahead Adder instead of the Ripple Carry Adder. Thus an improved Montgomery Modular Multiplier is designed and implemented.

REFERENCES

[1]. Ruirui Liu, Shuguo Li , "A Design and Implementation of Montgomery Modular Multiplier" *2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019*
[2]. P. L. Montgomery, “Modular Multiplication Without Trial Divisions,” *Mathematics of Computation, vol. 44, no.170,pp. 519–521, 1985.*
[3]. P. Barrett, “Implementation of The Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor,” *Advances in cryptology CRYPTO’86, pp.311-323, 1986.*
[4]. X. Yan, G. Wu, D. Wu, F. Zheng,and X. Xie, “An Implementation of Montgomery Modular Multiplication on FPGAs,” *2013 International Conference on Information Science and Cloud Computing, Dec2013,pp. 32–38*
[5]. A. Karatsuba and Y.Ofman, “Multiplication of Many-digital Numbers by Automatic Computers,” *Proceedings of Ussr Academy of Sciences, vol.145, no. 2,p. 293,1962.*
[6]. S. A. CookS. O. Aanderaa, “On the minimum computation time of functions,” *Transactions of the American Mathematical Society,Vo. 142, 1969,pp. 91–314.*
[7]. A. L. Toom, “The complexity of a scheme of functional elements realizing the multiplication of integers,” *Soviet Mathematics Doklady, vol. 3,pp. 714–716,1963.*
[8]. A. Zanoni, “Toom-Cook 8-way for Long Integers Multiplication,” *2009 11th International Symposium on Symbolic and Numeric Algorithms for Scientic Computing, Timisoara, 2009, pp. 54-57.*
[9]. C. D. Walter, “Montgomery exponentiation nedds no final substractions.” *Electronics Letters,vo. 35,no. 21,pp. 1831-1832.Oct 1999.*
[10]. J.DingandS.Li, “Broken-Karatsuba multiplication and its application to Montgomery modular multiplication,” *2017 27th International Conference on Field Programmable Logic and Applications (FPL), Ghent, 2017, pp. 1-4.*
[11]. J. C. T. Chow, K.Eguro, W. Luk, and P. Leong, “A Karatsuba-Based Montgomery Multiplier,” *in 2010 International Conference on Field Programmable Logic and Applications, Aug 2010,pp.434–437.*

Vol. 9, Issue 4, April 2021
DOI 10.17148/IJIREEICE.2021.9424

[12]. K. Javeed, X.Wang, and M. Scott, "Serial and parallel interleaved modular multipliers on FPGA platform," *in 2015 25th International Conference on Field Programmable Logic and Applications(FPL), Sept 2015,pp. 1–4.R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.*

[13]. Stephen E. Eldridge and Colin D. Walter "Hardware Implementation of Montgomery's Modular Multiplication Algorithm" *IEEE Transactions On Computers, Vol 42, No 6, June 1993*

[14]. Shiny P. Wilson and Ramesh P. "Karatsuba Multiplier for High Speed Applications" *2015 Global Conference on Communication Technologies (GCCT)*

[15]. Nitha Thampi, Meenu Elizabath Jose "Montgomery Multiplier for Faster Cryptosystems" *Global Colloquium in Recent Advancement and Effectual Researches in Engineering, Science and Technology (RAEREST 2016)*

[16]. Deepti Rajput, "Designing of Look Ahead Carry Adder by using VHDL" *International Journal of Electronics, Electrical and Computational System IJEECS ISSN 2348-117X Volume 5, Issue 12 December 2016*