Design and Implementation of Montgomery Multiplication

Vankudoth Venkanna@ Dr. R. P. Singh*

@Research Scholar, Department of Electronics & Communication Engineering, Sri Satya Sai University of Technology and Medical Science, Opposite Oilfed, Bhopal Indore Highway, Pachama, Sehore, Bhopal, Madhya Pradesh, India-466001.

*Vice- Chancellor, Sri Satya Sai University of Technology & Medical Science, Opposite Oilfed, Bhopal Indore Highway, Pachama, Sehore, Bhopal, Madhya Pradesh, India-466001

Abstract

In this paper, Low power consumption and smaller area requirement are prime concern in fabrication of DSP system on FPGA. Modular arithmetic is core operation in cryptosystems since they are efficient when data size is large (1024 bits or greater). In this paper a novel architecture of radix-2, 4,8,16 Montgomery multiplier is presented and implemented on FPGA device. Simulation shows that our design performs faster in terms of clock frequency while it requires lower area.

***Keywords:-* DSP, FPGA,Radix-2,4,8,16, Montgomery Multiplier, RNS and Cryptosystems**

1. INTRODUCTION

With the advancement in communication systems, security is a prime concern which is offered by public key cryptosystems. These systems offer authentication, confidentiality and privacy. Many cryptosystems including RSA, DSA and ECC systems requires modular multiplication for private key generation. [1] P. Montgomery developed an efficient algorithm for the calculation of (A X B) mod M called Montgomery Multiplication algorithm. Montgomery multiplication has been used as a fundamental operation of arithmetic operations in RSA-Algorithm. This paper presents FPGA implementation of scalable architecture for radix-2 Montgomery multiplication algorithm for 1024-bit operand.

2. MONTGOMERY MULTIPLICATION:-

MONTGOMERY MULTIPLICATION In 1985 a method for modular multiplication using Residue Number System (RNS) representation of integers is proposed by Peter L. Montgomery. In this method, the costly division operation usually needed to perform modular reduction is replaced by simple shift operations by transforming the operands into the RNS domain before the operation and re-transforming the result after operation. A radix R is selected to be two to the power of a multiple of the word size and greater than the modulus, i.e. R = 2w> M. For the algorithm to work R and M need to be relatively prime i.e. must not have any common non-trivial divisors. With R a power of two, this requirement is easily satisfied by selecting an odd modulus. This also fits in nicely with the cryptographic algorithms that we are targeting, where the modulus is either a prime always odd with the exception of 2or the product of two primes and therefore odd as well. RNS representations of integers are called M residues and are usually denominated as the integer variable name with a bar above it. An integer a is transformed into its corresponding M-residue $\bar{a}$ by multiplying it by R and reducing modulo M. The back-transformation is done in an equally straight forward manner by dividing the residue by R modulo M.

Thus here are the following equations as transformation rules between the integer and the RNS Domain:

$a = \bar{a}R^{-1}$

Montgomery Multiplication can be defined simply as the product of two M residues divided by the radix modulo M:

$\bar{c} = \bar{a}\bar{b}R^{-1}(mod\ M)$

Division by the Radix is required to make the result again an M-residue.

The key concepts of the Montgomery algorithm are the following:

i) Adding a multiple of M to the intermediate result does not change the value of the final result; because the result is computed modulo M. M is an odd number

ii) After each addition in the inner loop the least significant bit of the intermediate result is inspected. If it is 1, i.e., the intermediate result is odd, we add M to make it even. This even number can be divided by 2 without remainder. This division by 2 reduces the intermediate result to n+1 bits again.

iii) After n steps these divisions add up to one division by 2n.

Radix-2 Montgomery Multiplication Algorithm Let X and Y be two n-bit operands and M be any odd integer which is greater than zero for satisfying radix-2 operation. Montgomery multiplication involves first transformation of operands into Montgomery domain and then after result is re-transformed into Montgomery domain. This conversion process replaces division by several shift operations then Montgomery multiplication process for inputs X, Y, M and output Z is described as follows: Output to be obtained: Z= (X,Y) mod M Where X'=X.2n mod M Y'=Y.2n mod M Z' = MP(X', Y', M) = X Y 2n mod M Hence Z' = Z 2n mod M Hardware reduction of this algorithm is possible by pre-computation. The values to be added to the intermediate result within the loop can be pre-computed. Delay due to carry propagation must be avoided.

Faster Montgomery Multiplier The motivation behind this optimized algorithm is that of reducing the chip area for practical hardware implementation. This is possible if we can pre-compute four possible values to be added to the intermediate result. These are the four possible scenarios:

i) if the sum of the old values of S and C is an even number, and if the actual bit xi of X is 0, then we add 0 before we perform the reduction of S and C by division by 2.

ii) if the sum of the old values of S and C is an odd number, and if the actual bit xi of X is 0, then we must add M to make the intermediate result even. Afterwards, we divide S and C by 2.

iii) if the sum of the old values of S and C is an even number, and if the actual bit xi of X is 1, but the increment xi*Y is even, too, then we do not need to add M to make the intermediate result even. Thus, in the loop we add Y before we perform the reduction of S and C by division by 2. The same action is necessary if the sum of S and C is odd, and if the actual bit xi of X is 1 and Y is odd as well. In this case, S+C+Y is an even number, too.

iv) if the sum of the old values of S and C is odd, the actual bit xi of X is 1, but the increment xi *Y is even, then we must add Y and M to make the intermediate result even. Thus, in the loop we add Y+M before we perform the reduction of S and C by division by 2. The same action is necessary if the sum of S and C is even, and the actual bit xi of X is 1, and Y is odd. In this case, S+C+Y+M is an even number, too.

v) The computation of Y+M can be done prior to the loop. This saves one of the two additions which are replaced by the choice of the right operand to be added to the old values of S and C. Algorithm in 5.1 is a modification of Montgomery's method which takes advantage of this idea.

Algorithm for Faster Montgomery Multiplier Inputs:

```
Inputs: X, Y, M with 0 ≤X, Y< M
Output: P =(X*Y ( 2n ) -1)mod M
n: number of bits in X;
xi : ith bit of X;
s0: LSB of S,
c0: LSB of C,
y0: LSB of Y;
R: precomputed value of Y+ M;
        S =0 ; C =0 ;
        for (i=0 ; i <n; i++)
        {
        if ((s0= c0) and not xi ) then I =0;
        if ((s0≠ c0) and not xi ) then I= M
        if (not(s0^ c0^ y0 ) and xi ) then I = Y;
        if ((s0^c0^ y0 ) and xi ) then I=R;
        S,C= S+ C+ I; S : S div ;
        C= C div 2 ;
                          }
        P= S+ C;
        if (P≥ M) then P =P-M;
```

The advantage of Algorithm in Montgomery Multiplier in comparison to Algorithm in Radix-2 Montgomery Multiplication can be seen in the implementation of the loop of Algorithm in Montgomery Multiplier. The possible values of I are stored in a lookup-table, which is addressed by the actual values of xi, y0, s0 and c0. The operations in the loop are now reduced to one table lookup and one carry save addition. Both these activities can be performed concurrently. Note that the shift right operations that implement the division by 2 can be done by routing.

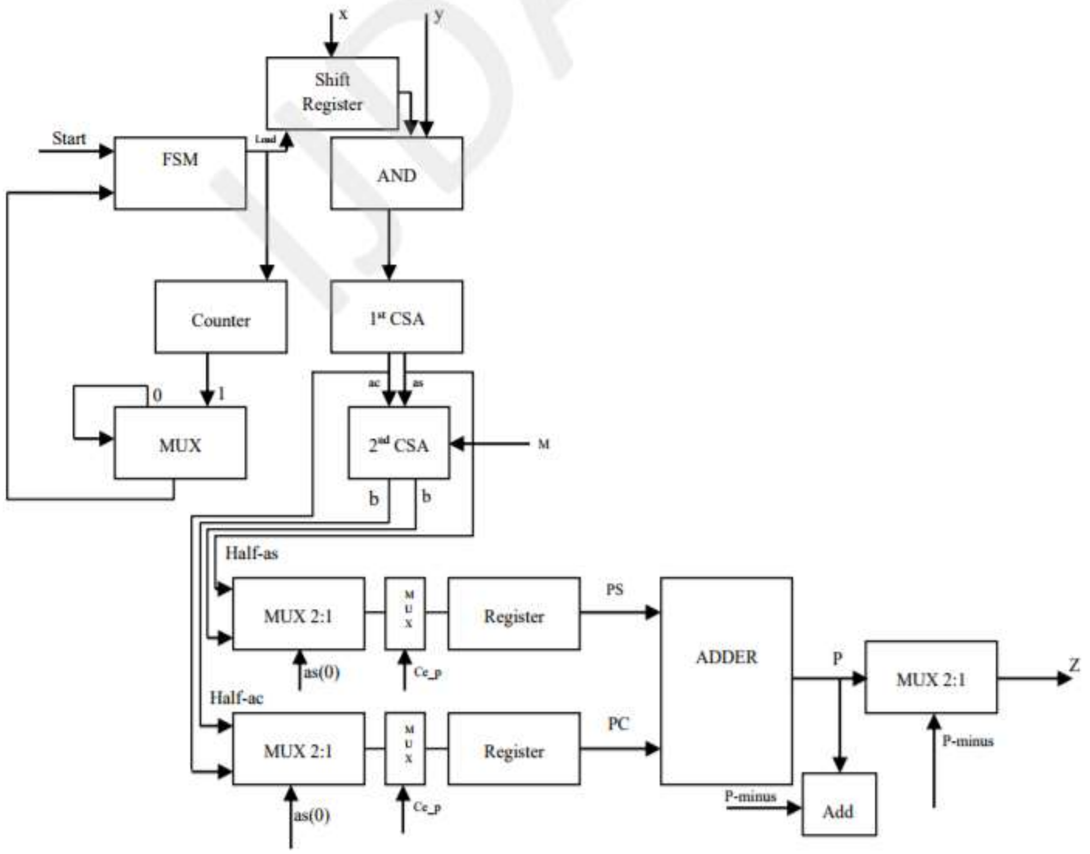


Figure:1 Representing Montgomery Multiplication Block Diagram

3. MONTGOMERY MULTIPLIER ARCHITECTURE

The detailed architecture of the Montgomery multiplier is given in Figure 2.

- The first Multiplexer MUX21 passes 0 or content of register B depending on bit a0. MUX22 passes 0 or contents register M depending on r0
- ADDER1 delivers the sum R + ai × B while ADDER2 gives R +M.
- SHIFT REGISTER1 provides bit ai and is right shifted for each I so that $a_0 = a_i$.
- Controller synchronizes the shifting and loading operations of shift registers.

- A major design concern in multiplication units in cryptography is the large no. of input bits which lead to complex systems. Many implementations of the Montgomery algorithm. But all these were for fixed precision of operands that is; once hardware is designed for n bits it cannot work with more no. of bits. For improved performance many high radix designs were also proposed but due to their increased complexity, low radix designs still remain an attractive choice for hardware implementation of Montgomery Multiplier.

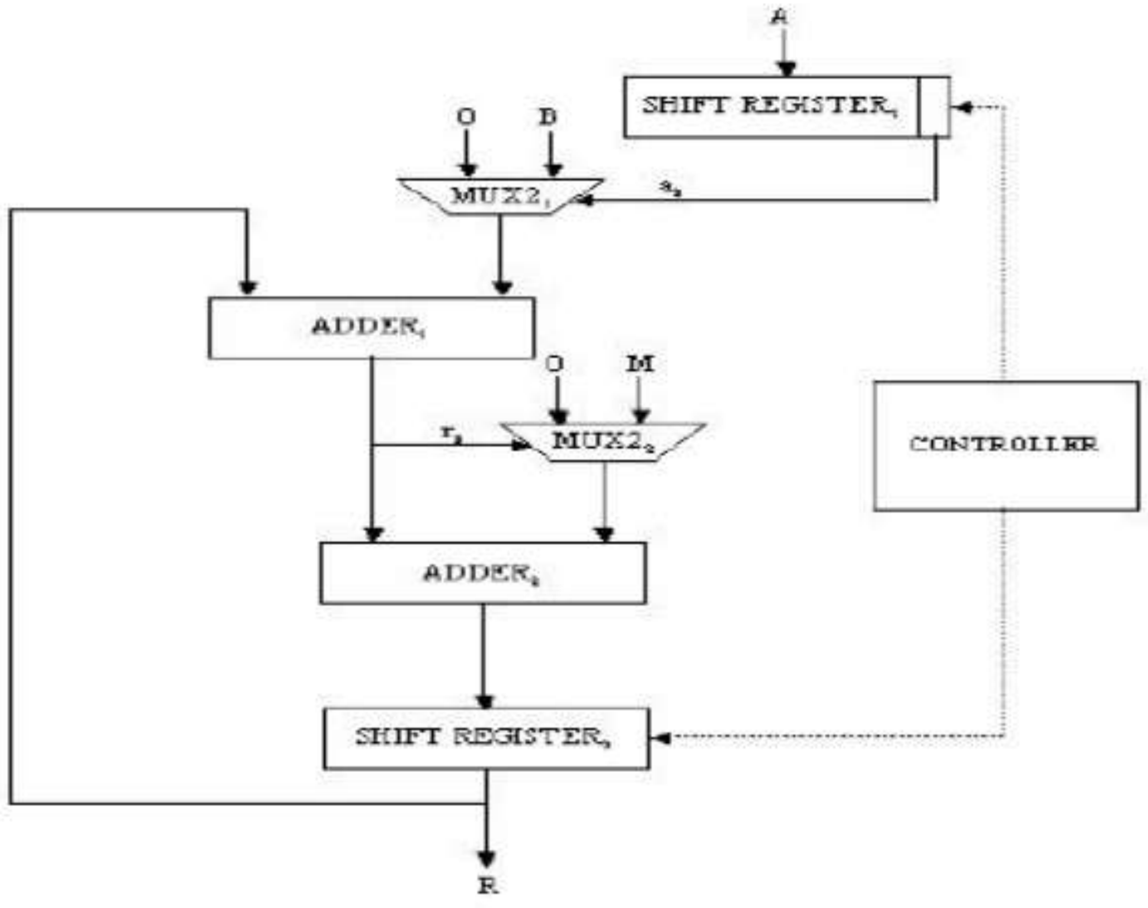


Figure:2 Montgomery Multiplier Architecture

Design Evaluation

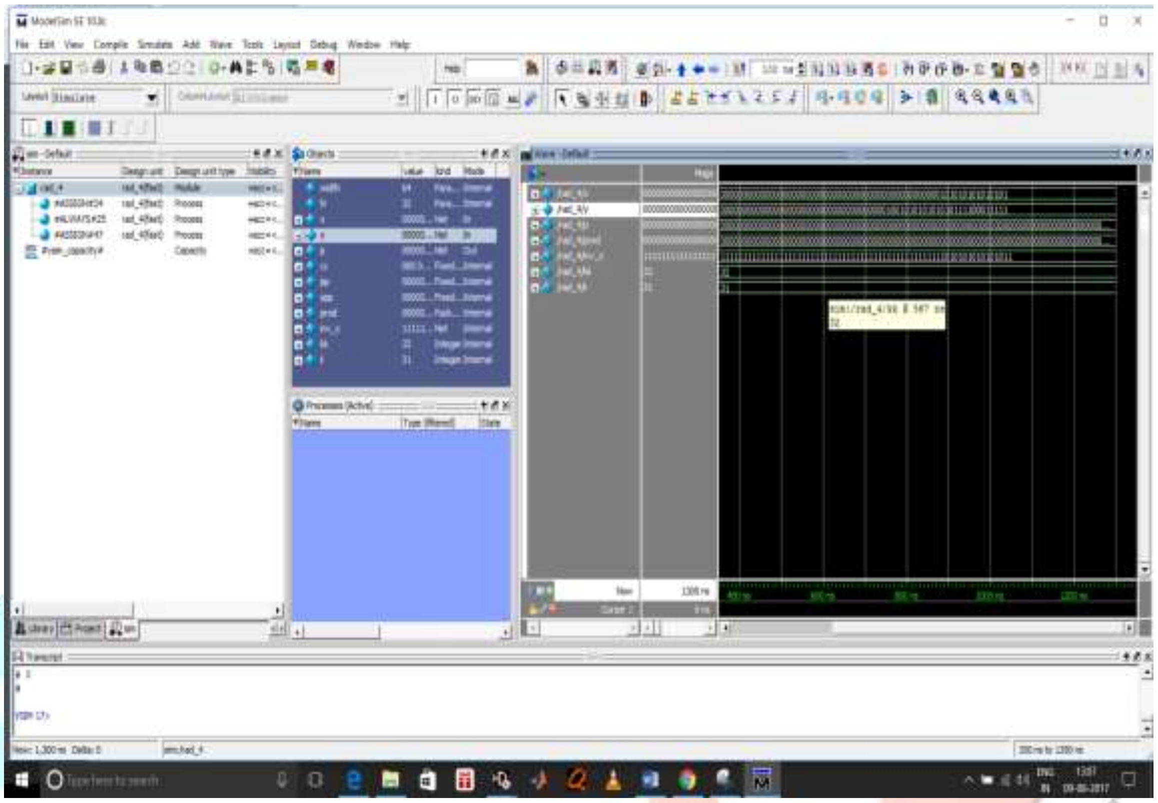
Faster Montgomery consumes less area and also less power. Delay analysis of Montgomery multiplier and faster Montgomery Multiplier using Xilinx 14.2 gives the following result:

Multiplier	Area	Delay
Normal Montgomery Multiplier	Number of slice registers used : 69	41.702ns
	Number of slice LUTs : 270	
	Number of bonded IOBs : 195	
Faster Montgomery Multiplier	Number of slice registers used : 81	32.55ns
	Number of slice LUTs : 227	
	Number of bonded IOBs : 131	

Table 1.Area and Delay Analysis

4. SIMULATION RESULTS:-

Simulation of the veri-log code for a Faster Montgomery Multiplier using Xilinx gave the following results



4.1 Comparison with Previous Techniques :-

Various techniques available for performing modular multiplication have been compared. The time consuming division operation has been replaced. It is seen that a Montgomery multiplier performs faster modular multiplication. Also the power consumption is lower in a Montgomery multiplier. The new method has a simple structure and requires a small amount of precomputation and storage. It reduces the number of necessary additions. The possible values are stored in a lookup-table.

5. CONCLUSION:

The estimated total circuit area and critical path delay of the modular multiplier based algorithm show that it can be implemented in much smaller hardware than that necessary to implement multiplier and divider separately. We conclude that, among the various algorithms proposed in literature for calculating modular multiplication, the Montgomery modular multiplication algorithm seem to be the suitable one to be combined. This paper presented an efficient algorithm to reduce the energy consumption and enhance the throughput of Montgomery modular multipliers simultaneously. The work can be further extended to modular exponentiation and squaring. A reversible architecture can be implemented for lower power consumption

References:-

[1] Tenca, Alexandre F., and Cetin K. Koc. "A scalable architecture for modular multiplication based on Montgomery's algorithm." Computers, IEEE Transactions on 52.9 (2003): 1215-1221.

[2] Kaihara, Marcelo E, and Naofumi Takagi. "A hardware algorithm for modular multiplication/division." Computers, IEEE Transactions on 54.1 (2005): 12-21.

[3] Kuang, Shiann-Rong, et al. "Energy-efficient high-throughput Montgomery modular multipliers for RSA cryptosystems." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 21.11 (2013): 1999-2009.

[4] Knezevic, Miroslav, Frederik Vercauteren, and Ingrid Verbauwhede. "Faster interleaved modular multiplication based on Barrett and 398 Nitha Thampi and Meenu Elizabath Jose / Procedia Technology 25 (2016) 392 – 398 Montgomery reduction methods. "Computers, IEEE Transactions on 59.12 (2010): 1715-1721.

[5] Kong, Yinan, and Braden Phillips. "Comparison of Montgomery and Barrett modular multipliers on FPGAs." (2006).

[6] Chen, Jun Hong, et al. "High-speed modular multiplication design for public-key cryptosystems." Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on. IEEE, 2008.

[7] Cho, Koon-Shik, Je-Hyuk Ryu, and Jun-Dong Cho. "High-speed modular multiplication algorithm for RSA cryptosystem." Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE. Vol. 1. IEEE, 2001.

[8] Shieh, Ming-Der, et al. "A new algorithm for high-speed modular multiplication design." Circuits and Systems I: Regular Papers, IEEE Transactions on 56.9 (2009).

[9] Peter L. Montgomery. Modular multiplication without trial division. Mathematics of Computation, 44(170):519–521, April 1985.
[10] Jean-Claude Bajard, Laurent-Stephane Didier, and Peter Komerup. AnRNS Montgomery modular multiplication algorithm. IEEE Transaction on Computers, 47(7):766–776, July 1998.

ABOUT AUTHOR(S):

Vankudoth Venkanna is a Research Scholar in the Department of ECE at Sri Satya Sai University of Technology and Medical Sciences, Sehore, Madhya Pradesh. He has completed Graduated (B.Tech) in ECE from JNTUH, India and Post Graduated (M.Tech) in VLSI System Design from JNTUH, India. He has 4 years of teaching, research, and administrative experience.

Dr. R. P. Singh has 39 years of teaching, research, and administrative experience as Professor. He has worked as Professor In-charge Academic and Chairman Admission Committee Dean (Academic) & Dean (R/D) at MACT /MANIT, Bhopal. He has published 125 papers in National / International reputed and indexed Journals including SCI. He has worked as Secretary, Chairman, IETE, M.P. and C.G. and Council Member, IETE. He was first Counselor of IEEE student's chapter at MACT, Bhopal. He has been member of Executive Committee, Institution of Engineers (I) M.P. Circle. He was Chairman of Computer Society of India. Bhopal. He was member of Board of Studies, and Research Degree committee of many UniversitiesHe has been Consulting Editor of Journal of Institution of Engineers and reviewer in many International/National Journals. Dr. Singh visited about 70 Institutions as an expert of NBA and about 35 Institutes as AICT/U G.C. expert team for approval. Twenty candidates have completed their Ph.D. under his supervision and another six are registered in the area of Electronics/Communication system and related disciplines.