# Optimized Multiple Word Radix-2 Montgomery Multiplication Algorithm

Harmeet Kaur[1], Charu Madhu[2]

[1]Post graduate (M.Tech) in UIET, Panjab University, Chandigarh, India

[2]Assistant Professor, UIET, Panjab University, Chandigarh, India

***Abstract*- Montgomery multiplication algorithm is used in the implementation of RSA and other cryptosystems based on modular arithmetic. Several improvements have been suggested to increase its suitability for hardware implementation. Radix-2 Montgomery architectures are easier to implement in hardware. In this paper a modified optimized algorithm for radix-2 Montgomery Multiplication is presented which is based on parallelizing the multiplications within each Processing Element and pre-computation of partial results using assumptions regarding the most significant bit of the previous design thereby improving speed. The design has been modeled using VHDL. The VHDL code has been synthesized and simulated using Xilinx ISE 10.1.**

***Keywords* - Montgomery Multiplication, RSA, Modular Multiplication, MWR2MM Algorithm**

## I. INTRODUCTION

With the explosive growth in telecommunication network and Internet popularity, care for information security issues is also increasing. For their security, cryptographic systems are important measures since these offer maximum security, along with high level of flexibility[1]. The RSA (Rivest, Shamir and Adleman) Algorithm is the most widely used public-key cryptosystem and many public key cryptography schemes, including RSA, involve the use of modular exponentiation of large numbers for encryption of data. This method is considered secure since factorization becomes intractable for very large numbers[2]. For large numbers modular exponentiation is a very slow process due to repeated modular multiplications with division involved to calculate the remainder.RSA Algorithm was introduced in 1978 and since then space efficient, high speed architectures for modular multiplication have been a subject of constant interest.

During this period, one of the most useful advances came with the introduction of Montgomery multiplication algorithm due to Montgomery [3]. Montgomery Multipliers are considered useful since the modulus reduction is done by shift operations eliminating the division step, thereby dramatically increasing the speed of encryption system as well as making its hardware implementation easy[4]. So Montgomery Multiplication has since then become an essential step in RSA, ECC and other crypto-systems involving modular exponentiation.

Many architectures for Montgomery Multiplication have been proposed improving various parameters like hardware requirements, speed, area, scalability etc. The most important development came with the introduction of a word-based algorithm and a scalable architecture for Montgomery multiplication called Multiple-Word Radix-2 Montgomery Multiplication (MWR2MM) by Tenca and Koc at CHES 1999 .Several designs based on the MWR2MMalgorithmhave been proposed [5], [6], [7], [8], [19], [10].In [5], Booth encoding technique has been used in a high radix version of word-based Montgomery algorithm (MWR2kMM). Number of scanning steps were reduced in the approach but complexity of system increased to a large extent. In [6] Harris et al. replaced right shifting of S in MWR2MM algorithm by left shifting of Y and M. The design maintained the scalability of original design and processed n-bit Montgomery multiplication in n clock cycles. In [7] and [8], the left-shifting technique was applied on the radix-2 and radix-4 versions of the parallelized Montgomery algorithm [9], respectively. The systolic implementation given in [11] by McIvor et al. improves speed to a large extent but has very large area requirements.

In this paper we focus on the optimization of the radix-2 MWR2MM algorithm to improve the speed of multiplication process. This paper is organized as follows: In section 2 the Montgomery multiplication, MWR2MM algorithm and other improvements are explained. In section 3 new radix-2 algorithm is discussed. Architecture of Montgomery multiplier is discussed in section 4. Results of the implementation are detailed in section 5.Section 6 presents a conclusion to this paper followed by references.

## II. MONTGOMERY MULTIPLICATION

One of the widely used algorithms for efficient modular multiplication is the Montgomery's algorithm [3]. This algorithm computes the product of two integers modulo a third one without performing division by M[4]. The reduced product is yielded using a series of additions. If A, B and M be the multiplicand and multiplier and the modulus respectively and 'n' be the number of digit in their binary representation, i.e. the radix is 2. Montgomery multiplication of X and Y (mod M), denoted by MP(X, Y, M) is defined as $X.Y.2^{-n}$ (mod M)[12]. Before multiplication process there is a conversion step from ordinary domain to Montgomery domain which is summarized below: -

$$X \text{ (Ordinary Domain)} \leftrightarrow X' \text{ (Montgomery Domain)} = X.2^n \pmod M$$

$$Y \text{ (Ordinary Domain)} \leftrightarrow Y' \text{ (Montgomery Domain)} = Y.2^n \pmod M$$

$$X.Y \text{(Ordinary Domain)} \leftrightarrow (X.Y)' \text{ (Montgomery Domain)} = X.Y.2^n \pmod M$$

The conversion between each domain can be done using the same Montgomery operation, in particular X'= MP(X, $2^{2n}$(mod M), M) and X= MP(X', 1, M), where $2^{2n}$ (mod M) can be pre-computed[12]. Despite the initial conversion cost, an advantage is achieved over ordinary multiplication if many Montgomery multiplications are followed by an inverse conversion at the end, which is the case, for example, in RSA.

The pre-conditions of the Montgomery algorithm[4] are as follows:

- The modulus M needs to be relatively prime to the radix, i.e. there exists no common divisor for M and the radix;
- The multiplicand and the multiplier need to be smaller than M.

The Montgomery algorithm uses the least significant digit of the accumulating modular partial product to determine the multiple of M to subtract[4]. The multiplication order is reversed by choosing the least significant multiplier digit first and then shifting down. If R is the current modular partial product, then q is chosen so that R+q×M is a multiple of the radix r, and this is right-shifted by r positions, i.e. divided by r for use in the next iteration. So, after n iterations, the result obtained is $R = A \times B \times r^{-n} \bmod M$ [13]. Montgomery algorithm is given below:-

algorithm Montgomery(A, B, M) [4]

```
int R = 0;
for i= 0 to n-1
R = R + a_i × B;
if r0 = 0 then
R = R div 2;
else
R = (R + M) div 2;
return R;
end Montgomery
```

For the right result, the process should be followed by an extra Montgomery Multiplication by the constant $2^n$ mod M.

### *A. MWR2MM Algorithm:-*

The Multiple Word Radix-2 Montgomery Multiplication Algorithm given by Tenca and Koc is a scalable algorithm wherein the operand Y (multiplicand) is scanned word- by-word and the operand X is scanned bit-by-bit. The operand length is 'n' bits and word length is 'w' bits and e= [(n + 1)/w ] words are required to store sum. The MWR2MM algorithm[12] is given below:-

Input: odd M, $n = [\log_2 M] + 1$, word size w, e= [(n + 1)/w],
$X = \Sigma_{i=0}^{n-1} x_i.2^i$, $Y = \Sigma_{j=0}^{e-1} Y^{(j)}.2^{w.j}$,
$M = \Sigma_{j=0}^{e-1} M^{(j)}.2^{w.j}$, with $0 \le X, Y < M$
Output $Z = \Sigma_{j=0}^{e-1} S^{(j)}.2^{w.j} = MP(X,Y,M) \equiv X.\ Y.\ 2^{-n} \pmod M$, $0 \le Z < 2M$ 2.1
S= 0;
2.2 for i = 0 to n-1 do
2.3 [ $q_i = (x_i.Y^{(0)})$ xor $S_0^{(0)}$ ;
2.4 $(C^{(1)}, S^{(0)}) = x_i.Y^{(0)} + q_i.M^{(0)} + S^{(0)}$;
2.5 for j = 1 to e do
2.6 [ $(C^{(j+1)}, S^{(j)}) = C^{(j)} + x_i.Y^{(j)} + q_i.M^{(j)} + S^{(j)}$ ;
2.7 $S^{(j-1)} = (S_0^{(j)}, S_{w-1\ldots1}^{(j-1)})$; ]
2.8 $S^{(e)} = 0$; ]
2.9 return Z = S;

The n-bit multiplicand is broken into w-bit words. The kernel of the design has 'p' Processing Elements. Each of the PE handles one bit of the multiplier and w bits of the multiplicand. The kernel iteration continues until the multiplication process is complete. The design is highly flexible and can be configured to handle any number of bits.

The overall advantage of the design lies in its hardware simplicity[7].

This architecture performs a single Montgomery multiplication in approximately 2n clock cycles. An optimization of this design has been suggested in [12] by Huang, Gaz and Ghazawi wherein the two clock cycle delay between each of the PE is reduced to half through an approach of pre-computation of partial results using two possible assumptions regarding the most significant bit of the previous word. Actually each of the PE in MWR2MM design has to

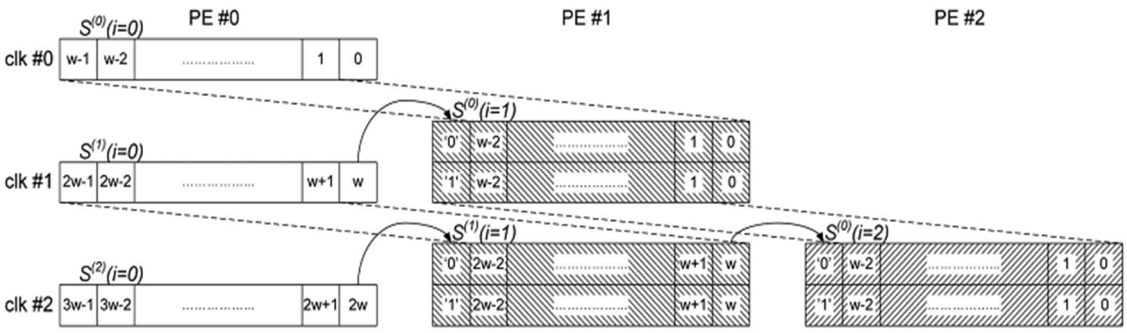


Fig.1 - MSB assumption in optimized architecture[12]

wait for 2 clock cycles for the value of S, before it starts its execution. This delay was reduced to a single clock by assuming MSB of S as '0' or '1' as shown in fig. 1.

*B. Parallelized Algorithm:-* In [14], Orup showed that the steps in the Montgomery Multiplication process can be re-ordered so that the multiplication and reduction processes could be executed in parallel. An implementation of the parallelized algorithm is given in [7]. It re-organizes the original algorithm to produce a new pre-calculated value $M^{\wedge}$ that allows the original algorithm's multiply and Result steps' multiplications to occur simultaneously. The algorithm[7] is

M' : n- bit integer satisfying $(RR^{-1} - MM') = 1$
$M^{\wedge}$ : $((M' \bmod 2)\, M + 1)/ 2$
w: multiplicand word length
e: [n/w] + 2 iterations per kernel
C: 1-bit carry digit
Z = 0
Q = 0
for i = 0 to n
C = 0
$Q = Z^0 \bmod 2$
for j = 0 to e -1
$( C, Z^{j+1} ) = Z^j + Q \times M^{\wedge j} + X^i \times Y^j + C;$

This design provided a significant cycle time improvement at the cost of a small increase in cycle count.

Our goal in this paper is to apply the optimization of MWR2MM architecture using pre-assumption of bits[12] to the radix-2 parallelized Montgomery Multiplier[7]. By combining both the approaches we achieve a significant amount of performance improvement in terms of speed of operation without escalating hardware cost.

## III. OPTIMIZED MULTIPLE WORD RADIX-2 MONTGOMERY MULTIPLICATION ALGORITHM

The Montgomery algorithm implemented in this paper is a hybrid between the optimized MWR2MM architecture and the parallelized algorithm for radix-2. The basic algorithm is similar to the parallel very high radix algorithm and the features of the optimized radix-2 algorithm are introduced by introducing pre-assumption of MSB of inter-mediate sum term at each PE. The resulting algorithm takes advantage of both one-cycle latency between Processing Elements and simultaneous multiplication of multiplication and result steps.

The algorithm is a scalable one and can be used to perform multiplication for any no. of bits. Each of the Processing Element runs multiple times during a kernel cycle to process all bits of M^ and Y. Thus, an inner for loop iterates over the n/w words of M^ and Y. Moreover, a side effect of the algorithm could be seen in form of an increased iteration of inner loop since the result in this case could be larger. The algorithm is of the form:-

M: n-bit odd modulus
n' : length of pre scaled X = n + 1
R : $2^{n'}$
$R^{-1}$: modular multiplicative inverse of R
M' : n-bit integer satisfying $RR^{-1} - MM' = 1$
M^ = (( M'mod2)M+1)/2
w = word length, e = [n/w] +1

1.Z = 0
2.For i = 0 to n
3.C = 0
4.Reduce = $Z_0$
5.for j = 0 to e-1

6.( $C0^{(j+1)}$ , $Z0^j_{w-1}$ ,$Z^j_{w-2,\ldots0}$) = ( 1,$Z^j_{w-1\ldots1}$) + Reduce * $M\hat{}^j$ + $X_i$ *$Y^j$ + $C^j$
7.( $CE^{(j+1)}$ , $ZE^j_{w-1}$ ,$Z^j_{w-2,\ldots0}$) = ( 0,$Z^j_{w-1\ldots1}$) + Reduce * $M\hat{}^j$ + $X_i$ *$Y^j$ + $C^j$
8.If $Z_0^{j+1} = 1$ then $C_j^{j+1} = C0_j^{j+1}$
9.$Z^j_{w-1\ldots1} = Z0^j_{w-1\ldots1}, Z^j_{w-2\ldots1}$
10.Else $C_j^{j+1} = CE_j^{j+1}$
11.$Z^j_{w-1\ldots1} = ZE^j_{w-1\ldots1}, Z^j_{w-2\ldots1}$

## IV. HARDWARE IMPLEMENTATION

The hardware architecture of our design is similar to that in [7]. Figure 2 shows the overview architecture of the design with p Processing Elements. Every PE receives one bit of X and Reduce, and w bits of M^ , Y and Z on each step. In one kernel cycle 'p' digits of X are processed against all bits of M^ and Y. So k= [n/p] + 1 kernel cycles are required for multiplication process. The results from the last PE could either be stored in a FIFO till kernel cycle completion of first PE or can be passed directly to first PE.

*A. Processing Element-* Figure 3 shows a processing element for the design. The multiplications of $X_i.Y^j$ and Reduce.$M^j$ are carried out using multiplexers. Additions in line 6 and 7 in the algorithm are carried out using ripple carry adders. The two additions differ only in the Most Significant bit of the Z term so the shared part of the two additions is consolidated and remaining part is then carried out using two adders. The basic representation is similar to that of [12], the difference being in value of M^.

Latency of the design is similar to that of [12] since the MSB is pre-assumed at each PE. The parallel modification does not show any effect on latency. With the change in the no. of Processing Elements time parameter changes. If the no. of Processing Elements is more than the no. of Processing cycles the first PE has to wait until the last PE is complete with its execution. In other case that is, with lesser no. of Processing Elements, hardware is utilized to maximum efficiency since by the time first PE is completed with its kernel cycle the Z term from last PE is already available.

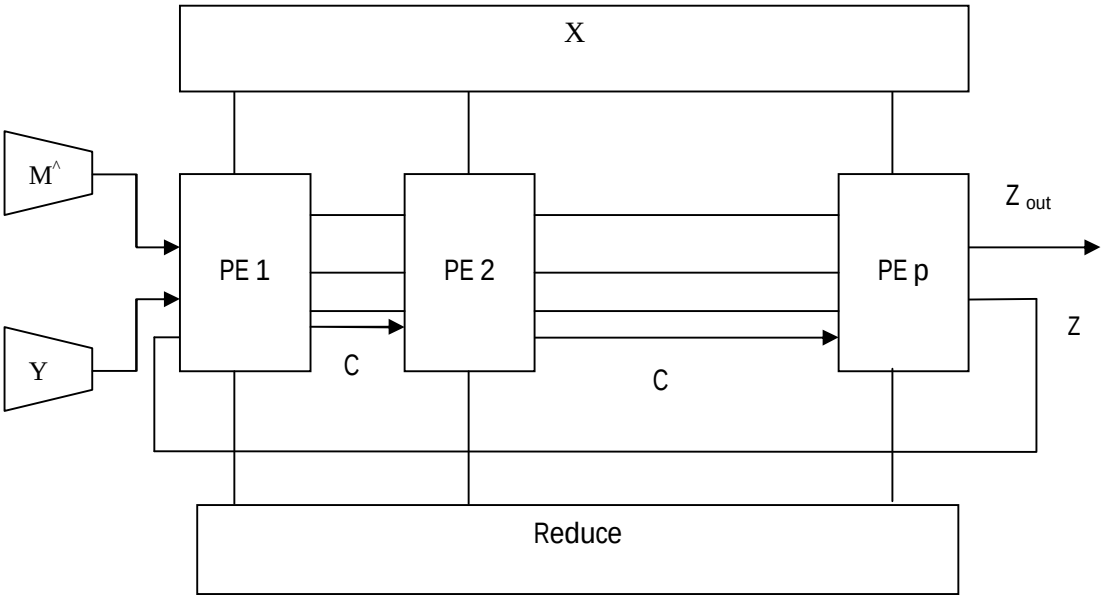


Fig. 2- Radix-2 Hardware Diagram

## V. RESULTS

The parallel radix-2 Montgomery Multiplier design described above was coded in VHDL and simulated using Xilinx ISE 10.1 Simulator. A maximum combinational path delay of 0.14 µs was observed in case of 1024-bit multiplication of numbers with 1024 processing elements and a word size of 4 bits which is considerably a lower value in comparison to the existing designs analyzed. The high speed attained could be a result of the reduced data dependency between Processing Elements and the parallelization of operations within PE's. Further performance boost is expected with reduction in no. of PEs and increase of word size.

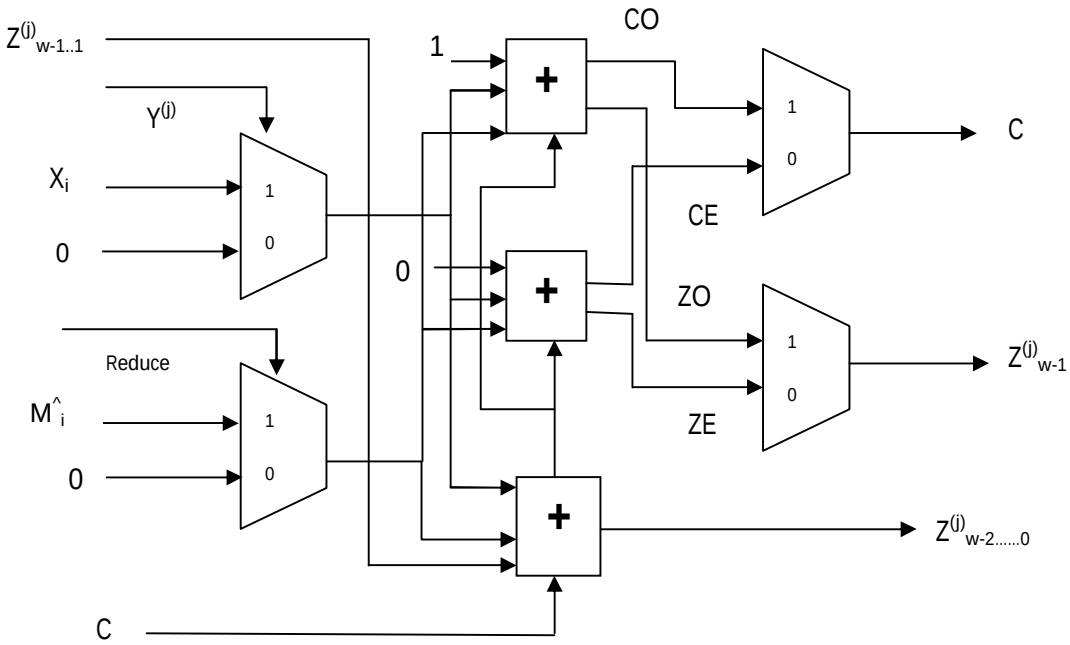


Fig. 3- Processing Element of our design

## VI. CONCLUSIONS AND FUTURE SCOPE

In this paper a modified algorithm of the MWR2MM algorithm and its corresponding architecture is presented. The design is basically a hybrid of the Parallelized scalable multiplier and optimized architecture of MWR2MM algorithm with reduced data dependency. The resulting algorithm thus takes advantage of both one-cycle latency between Processing Elements and simultaneous multiplication of multiplication and result steps. Good results in terms of speed have been seen in the work with an approximate maximum combinational path delay of 0.14 µs.

The work could be further improved through architectural changes to optimize the time and area requirements. Lesser no. of processing elements could be utilized to decrease the hardware requirements and strike a balance between the delay and hardware requirements trade-off. Moreover, the proposed Montgomery design could be used to implement a full cryptosystem like RSA or ECC. The power requirements of the circuit have not been dealt with in this work and may prove to be an interesting task.

The optimized Montgomery design may prove to be of great use in future cryptographic applications with more stringent demands of speed, area and power.

## REFERENCES

[1] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for ObtainingDigital Signatures and Public-Key Cryptosystems," Comm. ACM, vol. 21, no. 2, pp. 120-126, 1978.
[2] en.wikipedia.org/wiki/RSA(algorithm)
[3] P.L. Montgomery, "Modular Multiplication without Trial

Division,"Math. of Computation, vol. 44, no. 170, pp. 519-521, Apr. 1985.
[4] Nadia Nedjah, Luiza de Macedo Mourelle, "A Review of Modular Multiplication Methods and Respective Hardware Implementations" Informatica 30 (2006) 111–129.
[5] A.F. Tenca, G. Todorov, and C ¸ .K. Koc ¸, "High-Radix Design of a Scalable Modular Multiplier," Proc. Third Int'l Workshop Cryptographic Hardware and Embedded Systems (CHES '01), pp. 185-201,2001.
[6] D. Harris, R. Krishnamurthy, M. Anders, S. Mathew, and S. Hsu,"An Improved Unified Scalable Radix-2 Montgomery Multiplier,"Proc. 17th IEEE Symp. Computer Arithmetic (ARITH), pp. 172-178,June 2005.
[7] N. Jiang and D. Harris, "Parallelized Radix-2 Scalable Montgomery Multiplier," Proc. IFIP Int'l Conf. Very Large Scale Integration(VLSI-SoC '07), pp. 146-150, Oct. 2007.
[8] N. Pinckney and D.M. Harris, "Parallelized Radix-4 Scalable Montgomery Multipliers," J. Integrated Circuits and Systems, vol. 3,no. 1, pp. 39-45, Mar. 2008.
[9] K. Kelly and D. Harris, "Parallelized Very High Radix ScalableMontgomery Multipliers," Proc. 39th Asilomar Conf. Signals,Systems and Computers, pp. 1196-1200, Oct. 2005.
[10] E.A. Michalski and D.A. Buell, "A Scalable Architecture for RSACryptography on Large FPGAs," Proc. Int'l Conf. Field Program-mable Logic and Applications, (FPL '06), pp. 145-152, Aug. 2006.
[11] C. McIvor, M. McLoone, and J.V. McCanny, "High-Radix Systolic Modular Multiplication on Reconfigurable Hardware," Proc. IEEE Int'l Conf. Field-Programmable Technology (ICFPT '05), pp. 13-18,Dec. 2005.
[12] Miaoqing Huang, Kris Gaj, and Tarek El-Ghazawi, "New Hardware Architectures for Montgomery Modular Multiplication Algorithm," IEEE transactions on computers, vol. 60, no. 7, july 2011.
[13] Koç, Ç.K., High speed RSA implementation, Technical report, RSA Laboratories, RSA Data Security Inc. CA, version 2, 1994.
[14] H. Orup, "Simplifying quotient determination in high-radix modular multiplication," Proc 12th IEEE Symp. Computer Arithmetic, pp- 193-199, 1995.

Authors

HarmeetKaur received the B.Tech degree in Electronics and Communication in 2011. She is pursuing M.Tech (Microelectronics) from UIET, Panjab University, Chandigarh. Her area of interest includes image processing and VLSI Design.

Mrs Charu Madhu is M.E(Electronics and Communication) from Beant College of Engineering and Technology, PTU, Gurdaspur. Her area of research includes VLSI, nanoscale devices and optoelectronics. Currently she is working as Assistant Professor (ECE). She has 4 publications in International Journals/Conference proceedings.