International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 2, February 2013 ISSN 2319 - 4847

MONTGOMERY MULTIPLICATION METHODS - A REVIEW

Harmeet Kaur[1], Mrs.Charu Madhu[2]

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

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

ABSTARCT

RSA is the most widely used algorithm in public key cryptographic systems. It uses modular exponentiation of large numbers to encrypt data, which (although secure) is a slow process due to repeated modular multiplications. Thus the efficiency of an RSA encryption system depends on the speed of modular multiplications. Many hardware and software implementations for faster modular multiplication have been proposed, Montgomery Multiplication Algorithm is recognized as the most efficient among these. In this paper a survey of some known and recent Montgomery multiplier designs is presented, examining their strengths and weaknesses and a new high speed architecture for the same is proposed.

Keywords: Rivest, Shamir, Adleman Algorithm (RSA), Montgomery Multiplication, Processing Element(PE)

1. INTRODUCTION

With the exponential increase in electronic communication information security issues are also increasing proportionally. Applications like electronic mail, e-banking, e-commerce require secure channels for information exchange. Data exchanged has to be kept confidential and has to be protected against alteration. Cryptography is the answer to these problems. Through encryption of data, it is kept secret from all but those authorized to access it.

RSA encryption scheme is the most widely used cryptosystem. RSA algorithm is based on presumed difficulty of factoring large integers and is believed to be secure if its keys have a length of at least 1024 bits [1]. In RSA, a message is encrypted by representing it as a number M, raising M to a publicly specified power e, and then taking the remainder when the result is divided by the publicly specified product, n, of two large secret prime numbers p and q[2]. Decryption is similar, only a different, secret, power d is used where e.d ≡1 (mod (p-1). (q-1)). The security of the system rests in part on the difficulty of factoring the published divisor, n[2].

The modular exponentiation involved applies repetitive modular multiplications. So the performance of a RSA cryptographic system is essentially dependent on how fast modular multiplications are performed since these are at the base of computation. Thus high speed hardware architectures for modular multiplication are a subject of constant interest since the advent of RSA.

The most interesting advance in this field came with the Montgomery Multiplication Algorithm for Modular Multiplication given by Peter L. Montgomery [3]. This algorithm speeds up the multiplications and squaring required during the exponentiation process by eliminating trial division.

2. MONTGOMERY MULTIPLICATION

Montgomery Multiplication is an algorithm to perform modular multiplication quickly by replacing division (slow process) by multiplications. Montgomery multiplication of A and B (mod M), denoted by MP(A,B,M), is defined as $A.B.2^{-n}$ (mod M) for some fixed integer n. Before the actual multiplication the factors are converted to Montgomery domain and the result is computed in Montgomery representation which is then converted back to original form.

Ordinary Domain – A

Montgomery Domain- $A' = A.2^n \pmod M$

Despite the initial conversion delay, speed advantage over ordinary multiplication is achieved in case of large no. of Montgomery multiplications which is the case in RSA.

Pre-Conditions for Montgomery Algorithm [4]:

1. The multiplicand and multiplier need to be smaller than M.
2. Modulus M needs to be odd.
3. $n= [\log_2 M] +1$
4. Modulus M needs to be relatively prime to the radix.

A modified version of Montgomery Algorithm [4] is

Int R=0;
for i= 0 to n-1
    R= R + a_i×B;
If r_0 = 0 then
        R = R div 2
else
        R = (R + M) div 2 ;

In order to get the actual result, an extra Montgomery multiplication by the constant r^{2n} mod M is done.

2.1 MONTGOMERY MULTIPLIER ARCHITECTURE

The detailed architecture of the Montgomery multiplier [4] is given in Fig. 1.

- 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 a0 = ai.
- Controller synchronizes the shifting and loading operations of shift registers.

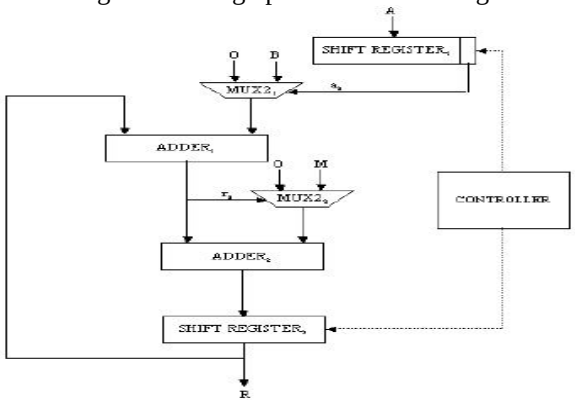

Figure 1 Montgomery Multiplier Architecture[4]

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 were developed [5], [6], [7]. 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 [8], [9] but due to their increased complexity ,low radix designs still remain an attractive choice for hardware implementation of Montgomery Multiplier.

2.2 WORD BASED RADIX-2 MONTGOMERY MULTIPLICATION ALGORITHM

Tenca and Koc[10] proposed a scalable architecture for Montgomery Multiplier called MWR2MM which proved to be the basis for several follow up designs of multipliers with very less computation time.

In this algorithm the multiplicand, Y is scanned word by word and the multiplier, X is scanned bit by bit. If word length is w bits then e(= n+1/w) words are required to store sum.

MWR2MM Algorithm [10] (for X.Y(ModM))

S=0
For i = 0 to m-1
$(C,S^{(0)}) := x_i Y^{(0)} + S^{(0)}$
If $S_0^{(0)} = 1$ *then*
$(C, S^{(0)}) := (C, S^{(0)}) + M^{(0)}$
for j= 1 to e – 1
$(C, S^{(j)}) := C + x_i Y^{(j)} + M^{(j)} + S^{(j)}$
$S^{(j-1)} := (S_0^{(j)} , S^{(j-1)}_{w-1...1}$
$S^{(e-1)} := (C , S^{(e-1)}_{w-1..1})$
else
for j= 1 to e – 1
$(C, S^{(j)}) := C + x_i Y^{(j)} + S^{(j)}$
$S^{(j-1)} := (S_0^{(j)}, S^{(j-1)}_{w-1...1})$
$S^{(e-1)} := (C, S^{(e-1)}_{w-1...1})$

The partial sum is computed for each bit of X and the process repeats for each bit of X. Thus there are no limitations on precision of operands, only the no. of iterations of loop varies with it. In this algorithm parallelism is possible among different i loops and the data dependency graph is of the form

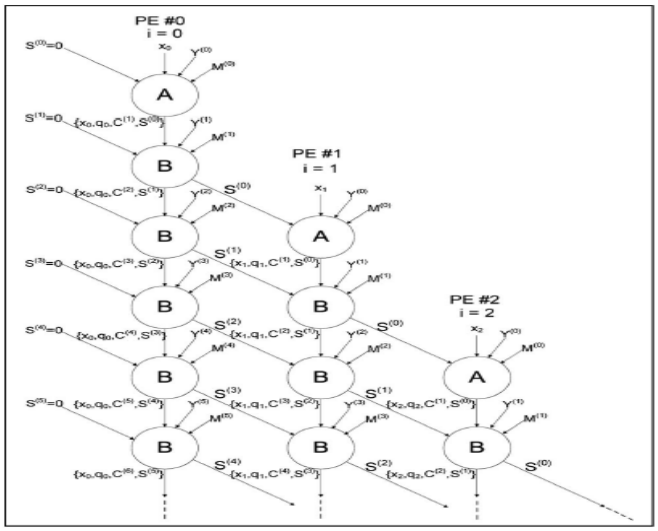

Figure 2 Data Dependency Graph of MWR2MM Algorithm[10]

Task A corresponds to iterations of i loop while Task B corresponds to j-loop iterations.

Each column in the graph is computed as a separate processing element (PE) and data generated from one PE is pipelined to another PE.

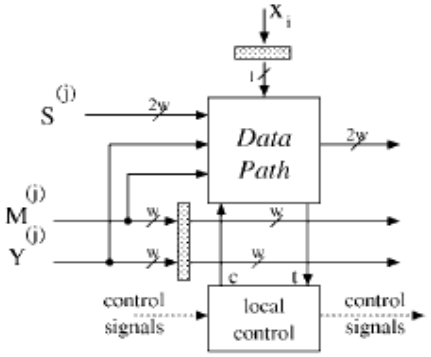

Figure 3 Processing Element[10]

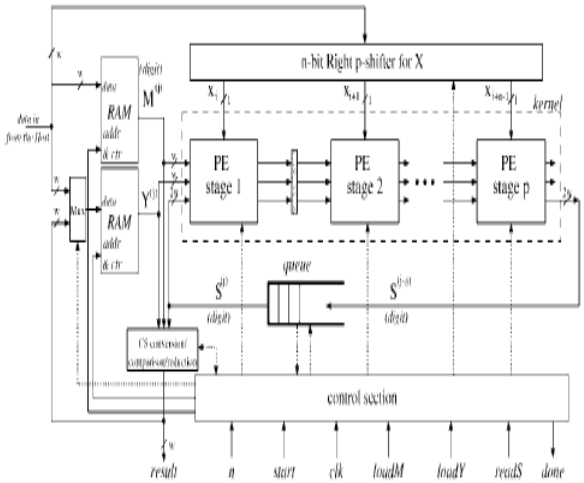

Figure 4 Pipelined Organization [10]

There is a delay of 2 clock cycles between processing of column x_i and x_{i+1}.For example, PE#1 has to wait for two clock cycles before computing $S^{(0)}(i=1)$. Thus an opportunity for improving the performance of this algorithm is to reduce this delay to 1 clock cycle.In this design, for 1024-bits, at 90 MHz a delay of 34,177 ns was observed.

2.3 HIGH RADIX SYSTOLIC MODULAR MULTIPLIER[11]

A high radix systolic modular multiplier was proposed by McIvor, McLoone, McCanny which is very well suited for implementations on FPGAs and provides high throughput rate.

Each Processing Element has an adder and a multiplier. Inclusion of multipliers led to the elimination of need of pre-computing any operands. The algorithm proposed in [10] is of the form:-

Input:

$$A = \sum_{i=0}^{m+2} (2^k)^i a_i, a_i \in \{0,1 \ldots 2^k - 1\}, a_{m+2} = 0$$

$$B = \sum_{i=0}^{m+1} (2^k)^i b_i, b_i \in \{0,1 \ldots 2^k - 1\};$$

$$\overline{M} = (M' \bmod 2^k) M,$$

$$\overline{M} = \sum_{i=0}^{m} (2^k)^i \overline{m}_i, \overline{m}_i \in \{0,1 \ldots 2^k - 1\};$$

$$A, B < 2\overline{M}; \; 4\overline{M} < 2^{km}; \; M' = -M^{-1};$$

Output: $S_{m+3} = ABR^{-1} \pmod M$;

1. $S_0 = 0$
2. for i in 0 to $m+2$ loop
   $q_i = S_i \bmod 2^k$
   $S_{i+1} = (S_i + q_i \overline{M})/2^k + A_i B$
   end loop
3. return $S_{m+3} = ABR^{-1} \pmod M$

Figure 5 High radix Systolic Multiplier Architecture

Each PE calculates the k-bit multiplications $q_i M$ and $A_i B$ and also performs the additions.

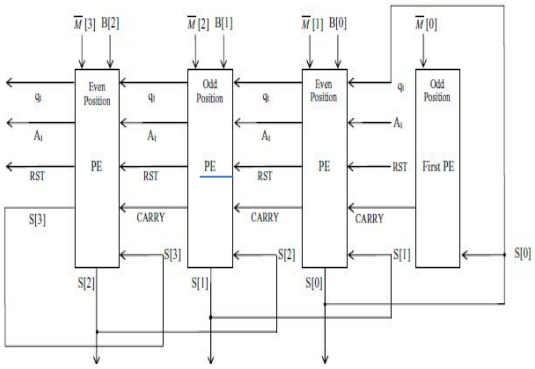

Figure 6 High radix Systolic Multiplier from [11]

RST is initially for a clock cycle to initialize S[0] to zero and shifted left through the array to initialize S[1] and S[2].Similar is the case for inputs q_i and A_i.

Radix 8 and Radix 16 designs have been reported to have very fast throughput rates as compared to their lower radix counterparts. But due to the increase in hardware as well as area involved these have limited application.

2.4 OPTIMIZED MWR2MM ALGORITHM [12]

MWR2MM algorithm was optimized by Huang, Gaz and Ghazawi, proposing a new hardware architecture for Montgomery Modular Multiplication which reduces the 2 clock cycle delay to half by pre-computing the partial results using assumptions.

In MWR2MM, PE#1 can take w-1 bits of $S^{(0)}$ (i=0) from PE#0 at the beginning of clock#1 and two different assumptions regarding the MSB (0 or 1) are made and two versions of $S^{(0)}$(i=1) are computed. At beginning of clock cycle#2, the missing bit is available as LSB of $S^{(1)}$(i=0) and is then used to choose between the two pre-computed versions. The pattern is repeated throughout.

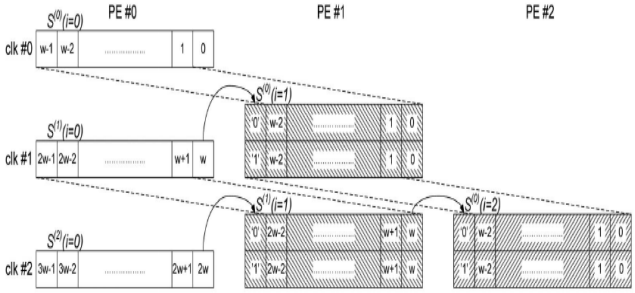

Figure 7 MSB assumptions in optimized MWR2MM algo

Similar to the tasks A and B in MWR2MM Algorithm it has tasks D and E.
Computations in Task D include:

1. Computation of q_i.
2. Calculation of two possible results
3. Selection between the two results on basis of $S_0^{(1)}$

The algorithm for computations of D[12] is summarized below:-

Input: $x_i, Y^{(0)}, M^{(0)}, S_0^{(1)}, S_{w-1..1}^{(0)}$
Output: $q_i, C^{(1)}, S_{w-1..1}^{(0)}$
$q_i = (x_i \cdot Y_0^{(0)}) \oplus S_1^{(0)};$
$(CO^{(1)}, SO_{w-1}^{(0)}, S_{w-2..0}^{(0)}) = (1, S_{w-1..1}^{(0)}) + x_i \cdot Y^{(0)} + q_i \cdot M^{(0)}$
$(CE^{(1)}, SE_{w-1}^{(0)}, S_{w-2..0}^{(0)}) = (0, S_{w-1..1}^{(0)}) + x_i \cdot Y^{(0)} + q_i \cdot M^{(0)}$
if $S_0^{(1)} = 1$ **then**
  $C^{(1)} = CO^{(1)};$
  $S_{w-1..1}^{(0)} = (SO_{w-1}^{(0)}, S_{w-2..1}^{(0)});$
else
  $C^{(1)} = CE^{(1)};$
  $S_{w-1..1}^{(0)} = (SE_{w-1}^{(0)}, S_{w-2..1}^{(0)});$

Figure 8 Optimized MWR2MM Algorithm

Task E forwards $S_0^{(j)}$ and $S^{(j)}_{w-1..1}$ and computations in E[12] include:-

Input: $q_i, x_i, C^{(j)}, Y^{(j)}, M^{(j)}, S_0^{(j+1)}, S_{w-1..1}^{(j)}$
Output: $C^{(j+1)}, S_{w-1..1}^{(j)}, S_0^{(j)}$
$(CO^{(j+1)}, SO_{w-1}^{(j)}, S_{w-2..0}^{(j)}) =$
$(1, S_{w-1..1}^{(j)}) + C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)};$
$(CE^{(j+1)}, SE_{w-1}^{(j)}, S_{w-2..0}^{(j)}) =$
$(0, S_{w-1..1}^{(j)}) + C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)};$
if $S_0^{(j+1)} = 1$ **then**
  $C^{(j+1)} = CO^{(j+1)};$
  $S_{w-1..1}^{(j)} = (SO_{w-1}^{(j)}, S_{w-2..1}^{(j)});$
else
  $C^{(j+1)} = CE^{(j+1)};$
  $S_{w-1..1}^{(j)} = (SE_{w-1}^{(j)}, S_{w-2..1}^{(j)});$

Figure 9 Task E Algorithm

This method reduces the delay between the PEs while maintaining the scalability of architecture proposed by Tenca and Koc[10].At 90 MHz, a delay of the order of 9.349 µs was observed for 1024-bits.

2.5 PARALLELIZED SCALABLE MONTGOMERY MULTIPLIERS[13]A parallel approach to Montgomery Multiplier is proposed which parallelizes the multiplications within each Processing Element thereby, improving speed.In this algorithm, multiplication and reduction steps of Montgomery Multiplication are performed in parallel by prescaling X by 2^v, n×n-bit multipliers are replaced by v×n-bit multipliers here.Parallelized Scalable radix-2^valgorithm[13] given is of the form

$Z = 0$
for $i = 0$ to f
  $C = 0$
  $reduce = Z_{v-1:0}$
  for $j = 0$ to $e - 1 + \left\lfloor \frac{v}{w} \right\rfloor$
    $(C, Z_{(j+1)w-1:jw}) = (Z_{(j+1)w+v-1:(j+1)w}, Z_{(j+1)w-1:jw+v}) +$
    $reduce \times \hat{M}_{(j+1)w-1:jw} +$
    $X_{(j+1)w-1:v} \times Y_{(j+1)w-1:jw} + C$

Figure 10 Parallelized scalable algorithm

The pre-computation of M^ here eliminates the use of multiplexers and reduce the critical path.

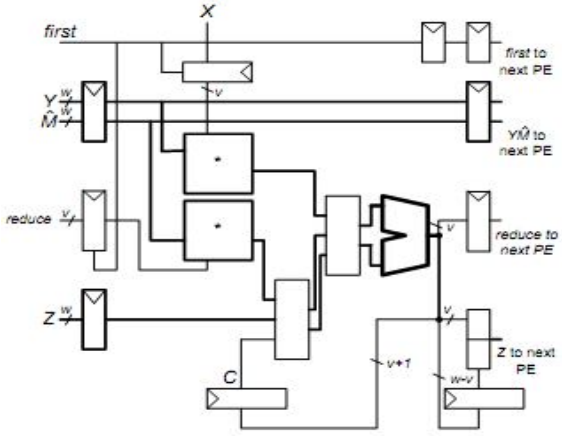

Figure 11 Processing Element [13]

Large improvements in time required for a modular exponentiation, of the order of ms are reported. Parallel multipliers thus are a good option to work with to enhance speed.

2.6 PARALLELIZED RADIX-2 SCALABLE MULTIPLIER [14]In order to eliminate the extra clock delay between Processing Elements an improved parallel scalable radix-2 design was given by Jiang and Harris. The extra clock cycle delay between successive processing elements in the original MWR2MM architecture is removed by left shifting Y and M in place of right shifting the partial element. Thus the PEs do not have to wait for the next word.

w: multiplicand word length
e: $\lceil n/w \rceil + 2$ PE 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 $e-1$
    $(C, Z^{j-1}) = Z^j + Q \times \hat{M}^j + X^i \times Y^j + C$

Figure 12 Parallel Scalable Radix 2 algorithm [14]

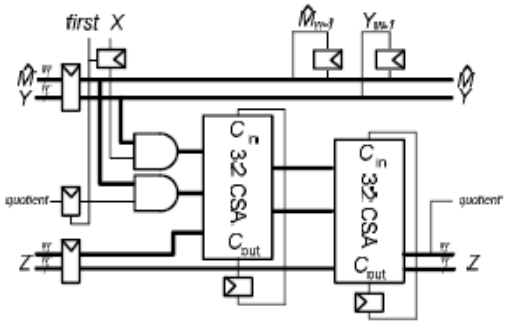

Figure 13 Processing Element[14]

The design is not only faster but also smaller than the previous designs since it eliminated the no. of gates used in previous designs. But overhead does increase on account of extra left shift operations.

3. PROPOSED DESIGN

Analyzing the various architectures, we proposea new high speed radix-2 Montgomery Multiplier Architecture which will be a hybrid of the parallel multiplier of [13] and optimized MWR2MM multiplier of [12].Our design will be a radix 2 parallel multiplier which will eliminate the data dependency between Processing Elements by pre-assuming the MSB of intermediate sum term thus removing the delay which arises when a PE waits for the MSB of partial sum element for an extra clock cycle. Moreover in parallel implementation the processes within a Processing Element are executed in parallel further speeding up the system. The design will be then compared with the scalable parallel multiplier of [13]. We suspect our design will have an increased speed as well as decreased overhead since the extra left shift operation is eliminated on account of pre-assumption of bits.

4. CONCLUSION & FUTURE WORK

Considering all the multipliers discussed above hybrid multiplier can give good results for operands with large no. of bits. Its high speed is a main advantage in VLSI design. The work can be further extended through analysis of power dissipation and hardware requirements of the multiplier and optimizing the three parameters for an efficient Montgomery multiplier design.

REFERENCES

[1]en.wikipedia.org/wiki/rsa_(algorithm)
[2] r.l. rivest, a. Shamir, and l. Adleman, "a method for obtaining digital signatures and public-key cryptosystems," comm. Acm,vol. 21, no. 2, pp. 120-126, 1978.
[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 MacedoMourelle, "A Review of Modular Multiplication Methods and Respective Hardware Implementations", Informatica 30 (2006)
[5]Nedjah, N. and Mourelle, L. M.,"Reconfigurable hardware implementation of Montgomery modular multiplication and parallel binary exponentiation", Proceedings of the Euro Micro Symposium on Digital System Design − Architectures, Methods and Tools, Dortmund, Germany, IEEE Computer Society Press, pp. 226-235, 2002
[6] Mourelle, L.M. and Nedjah, N., "Compact iterative hardware simulation model for Montgomery's algorithm of modular multiplication", Proceedings of ACS/IEEE International Conference on Computer Systems and Applications, Tunis, Tunisia, July 2003.
[7] C.Y. Su, S.A. Hwang, P.S. Chen, "An improved Montgomery's algorithm for high-speed RSA public-key cryptosystem [J]," IEEE Trans on VLSI Systems, 7(2), 1999, pp. 280-284.
[8] 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.
[9] T. Blum and C. Paar, "Montgmery modular exponentiation one configurable hardware," 14th IEEE Symposium on Computer Arithmetic, Adelaide, Australia, April 14-16, 1999.
[10] A.F. Tenca and C¸ .K. Koc¸, "A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm," IEEE Trans. Computers, vol. 52, no. 9, pp. 1215-1221, Sept. 2003.
[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, Member, IEEE, Kris Gaj, and Tarek El-Ghazawi, "New Hardware Architectures for MontgomeryModular Multiplication Algorithm" IEEE Transactions on Computers ,2011
[13] K. Kelly and D. Harris, "Parallelized Very High Radix Scalable Montgomery Multipliers," Proc. 39th Asilomar Conf. Signals, Systems and Computers, pp. 1196-1200, Oct. 2005.
[14] 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.

AUTHORS

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

Mrs Charu Madhuis 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 3 publications in International Journals/Conference proceedings.