DOI 10.29042/2018-4433-4440

Helix Vol. 8(6): 4433- 4440

# Performance Improvement of Montgomery Multiplier Architecture Using Pre-Computation and Unfolding Technique

[1]Prafulani Gajbhiye, [2]Pankaj Joshi

[1,2]Department of Electronics Engineering, Ramdeobaba College of Engineering and Management, Nagpur

*Email: gajbhiyepd@rknec.edu,joshipu@rknec.edu*

**Received: 20th September 2018, Accepted: 11th October 2018, Published: 31st October 2018**

**Abstract**

The Modern era of information and communication technology demands security. It is a prime important parameter along with other features. Data encryption and decryption algorithms such as RSA and ECC are popularly used to get the desired level of security. Modular multiplication is the integral part of these algorithms. Montgomery Multiplication is most efficient algorithm for modular multiplication. The modulo multiplication is a slow process for large bit size computations for key size more than 512. The critical path delay in architecture affects the iteration delay and overall computation time of encryption and decryption.

The slow ripples of the carries in the Montgomery multiplication are often replaced by the carry save architectures of additions. The paper optimizes this traditional approach using a novel combination of an unfolding algorithm and a pre-computation technique. A new architectureMM4_2 multiplier, which holds input and output in carry save format and consuming the smallest critical path, is also modified using unfolding approach .

The novel approach improves the overall computation time by 37.89% and 34.68% for ASIC and FPGA implementations as compared to traditional carry save approach. Further, unfolding of MM42_Multiplier in resent architecture gives improvement of 10.98 % in ASIC and 31.24% in FPGA as compare to original MM42 multiplier architecture.

**Keywords**

*Montgomery Multiplication, Delay Optimization, Pre-Computation, Unfolded Technique.*

## Introduction

The modern communication trends are more concerned over the security of the information. Cryptography, meaning a secret writing helps in encryption and decryption of the information to be transmitted and received respectively. The time required to encrypt and decrypt the information is one of the key role players in deciding the efficiency of these algorithms. The trend of using a larger key size for more security is popular but affects the overall time of encryption. Elliptic curve cryptography (ECC) uses much smaller key size as compared to traditional RSA algorithm to achieve the same level of security. For example, the security level achieved by the key size of 2048 in RSA algorithm is achieved by key size of 256 only in ECC algorithm. The basic operation for data encryption in cryptographic algorithm is series of modular multiplication given below.

$$Z = X.Y \bmod M \quad (1)$$

The time required to complete the modular multiplication increases with the key size. This makes the cryptosystem slow. The faster encryption and decryption makes the server job easier, with faster loading of web pages. Various algorithms like shift and add multiplication, double add and reduce, Montgomery Multiplication algorithm are developed for fast modular multiplication. The Montgomery Multiplication algorithm is most efficient, fastest and requires lowest computation over head. In Montgomery Multiplication add and shift operations are the only basic operations. Carry propagation in these adders increases the critical path delay of the circuit, resulting to limited speed of the operation. The main objective of the proposed work is to decrease the critical path delay and overall computation time of the Montgomery multiplication. To do so, the traditional Montgomery multiplier architecture is modified with unfolding and pre-computation techniques. In the past, one can witness some techniques based on pre computation. We further modify this pre-computation approach with a combination of unfolding algorithm giving better delay performances by sacrificing some area. The philosophy behind a use of an unfolding technique reduces the overall computation time theoretically to half. The next section discuss about the previous work on the Montgomery architecture based on pre-computation and a carry save approach and a new MM42 Multiplier architecture , and its unfolded version is also discussed.


Copyright © 2018 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

**Previous Work**

Modulo arithmetic circuits undergo several modifications in the past, either to increase the speed or to decrease the power. The literature discusses different architectures of Montgomery multiplication. In [5] Modified Montgomery Modular Multiplication Carry Save Adders (CSA) are used to perform the large word length additions. Two algorithms presented in [5] use five-to-two CSA and four-to-two CSA to reduce critical path of Montgomery multiplication. In [6] Fast Montgomery Modular Multiplication by Pipelined CSA Architecture is discussed. In this [6] paper CSA adders are broken into 1/3 of its original length and pipelined by adding extra registers to store the intermediate results. This increased the throughput of the multiplier. In [8] a lookup table based technique is used and one of the CSA is removed. We propose a modification in [8] using a combination of unfolding and pre computation technique. In[12] energy efficient high-throughput Montgomery Modular Multipliers are explained which reduce energy consumption and improve throughput .We attempt to improve the traditional Montgomery multiplication architecture evolved using CSA technique with the combination of pre-computation and unfolding algorithms. MM42_Multiplier is modified with unfolding , to get a better improved speed multiplier , to serve the purpose to resolve the bottleneck of slow operation of Montgomery Multiplier .

**Montgomery Modular Multiplication Algorithm**:

Montgomery modular multiplication is a method for performing modular multiplication. It was developed in 1985 by the mathematician Peter L. Montgomery. Given two integers *A* and *B* and modulus M the modular multiplication algorithm gives result *AB* mod *M*. Given below is the Montgomery Multiplication algorithm.

Input: X, Y, M; $0 \le X, Y < M$. Output:
X.Y. $R^{-1}$ Mod M S = 0 fori = 0 to n-1
S = S+ Xi× Y if S is odd then
S = S+ M S=S/2 if S= M then
S= S – M

This algorithm computes = MM(X,Y ) $=X.YR^{-1}$ mod M. The main operations involve in algorithm are add and shift operation. Addition operation is implemented using adders, carry propagation causes long critical path in adders, which limits the speed of the operation.

**Proposed Architectural Modifications** Several techniques were proposed that reduce carry propagation delay which limit the speed of circuit. In [5] Modified Montgomery Modular Multiplication and RSA exponentiation technique the addition is broken into multiple stages and implemented with carry save adder. Bunimov*et.al* [8] proposed a method that uses Carry Save-Adder to reduce critical path delay.

Given below is Fast Montgomery Multiplication algorithm.

**Inputs**: *X, Y, M* with $0 \le X, Y < M$ **Output**: $P= (XY(2^n)^{-1})$ mod *M* n: number of bits in *X*,
*X*i*: i*thbit of X
*S*o: LSB of *S*

1. *S* := 0; *C* := 0;
2. For *i*:=0 to *k*-1 do
3. *S,C*:= *S* + *C* + *X*i x *Y*;
4. if S,C is odd then
5. *S,C*:= *S* + *C* + *S*o x *M*;
   *S*:=*S* div 2; *C* := div 2;
6. *P* := *S* + *C*
7. If *P*>*M* then *P* := *P* -*M*;

The algorithm [8] uses two carry save adder as shown in Fig 1. The critical path of carry save adder is equal to a full adder, as the carry bits are store in a register. Carry save adder add three operands and gives two results sum and carry. The final result is added with the conventional adders. In this [8] architecture carry save adders consume most of the area. The delay is further reduced by eliminating one of the carry save adder by pre-computing the result of one carry save adder.

 Copyright © 2018 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

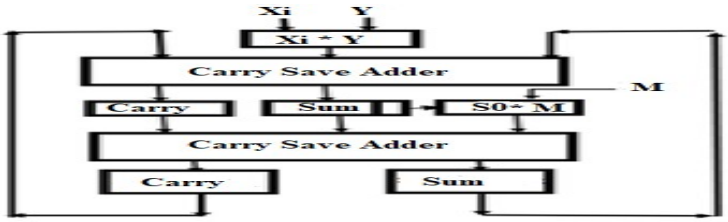


**Figure 1: Montgomery Multiplication Algorithm [8]**

In the Montgomery Multiplication Algorithm given in [8] step 3 and step 4 are implemented using Carry Save Adder. In step 3
when Xi=0, there is no need to add Y to the Sum. In step 4 when So=0, Co=0 (ie. So, Co are even) then, there is no need to add M to the Sum. Using the approach author in [8] pre- computed the output of first Carry Save Adder. These values are stored in the Look up table. As shown in Table I Address (Xi, Yo, So, Co) is give to the lookup table, Z is result generated by the look up table. This Z is given as input to the carry save adder.

| Xi | Y0 | C0 | S0 | Z |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | M |
| 0 | 0 | 1 | 0 | M |
| 0 | 0 | 1 | 1 | Y |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | M |
| 0 | 1 | 1 | 0 | M |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | Y |
| 1 | 0 | 0 | 1 | Y + M |
| 1 | 0 | 1 | 0 | M + Y |
| 1 | 0 | 1 | 1 | Y |
| 1 | 1 | 0 | 0 | M + Y |
| 1 | 1 | 0 | 1 | Y |
| 1 | 1 | 1 | 0 | Y |
| 1 | 1 | 1 | 1 | Y + M |

**Table I: Lookup Table for Pre Computation [8]**

The approach suggested by [8] of using a look up as shown in Fig. 2 can be further modified to decrease delay and the overall computation time of the Montgomery multiplication. We unfolded the architecture of Montgomery Multiplication which requires K iteration to obtain a final modular multiplication result, where K is number of bits in input A and B. For the circuit shown in Fig. 3 the addition logic is implemented by AND- XOR gates while the shifting operation by one bit is done by shift register.

 Copyright © 2018 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

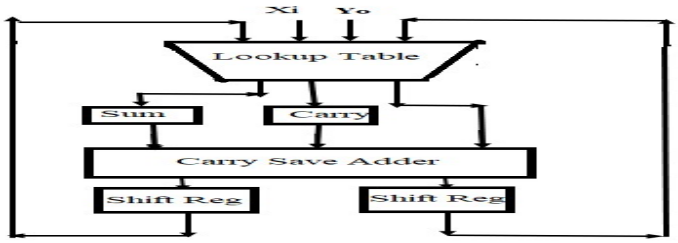


**Figure 2: Montgomery Multiplication using Lookup Table [8]**

The subtraction operation which is performed at fixed stage of algorithm if the Sum $\geq$ $M$ *inc*reasing the number of iteration. Walter [3] resolve maintaining the range of A,B and Sum within [0, 2M] . And also increasing the number of iteration perform from K to K+2. The value of R is $A = 2^{K+2} Mod\ M$.But addition operation involve in the algorithm still remain the problem as it require conventional adders. Carry ripple propagation in these conventional adders is the responsible for the long critical path of the circuit. This bottleneck is resolve , by a new Montgomery Multiplication design, MM42_ Multiplier[12].

This new architecture maintain the inputs i.e. A, B and output Sum in carry save format i.e. AS,AC,BS,BC and (SS,SC) respectively. Two extra register are required D1, D2 registerare required to store precomputed value (BS+BC+ M).

This reduce the critical path, accelerating the operation of Montgomery multiplier. Further to enhance the multiplier processing speed, we contribute to reduce speed by applying unfolding technique, which result in reduced delay for final result.

**Algorithm: MM42_Multiplier [12]**

Inputs : $A1, A2, B1, B2, N$ (modulus)
Outputs : $S1[k], S2[k]$
1. $(D1, D2) = B1 + B2 + N + 0;$
2. $S1[0] = 0;\ S2[0] = 0;$
3. for $i = 0$ to $k - 1$ {
4. $q_i = (S1[i]_0 + S2[i]_0 + A_i \times (B1_0 + B2_0))$ mod 2;
5. if ($A_i = 0$ and $q_i = 0$)
6. $(S1[i+1], S2[i+1]) = (S1[i]+S2[i]+0+0) / 2;$
7. else if ($A_i = 0$ and $q_i = 1$)
8. $(S1[i+1], S2[i+1]) = (S1[i]+S2[i]+N+0) / 2;$
9. else if ($A_i = 1$ and $q_i = 0$)
10. $(S1[i+1], S2[i+1]) = (S1[i]+S2[i]+B1+B2) / 2;$
11. else if ($A_i = 1$ and $q_i = 1$)
12. $(S1[i+1], S2[i+1]) = (S1[i]+S2[i]+D1+D2) / 2;$
13. }
14. return $S1[k], S2[k];$

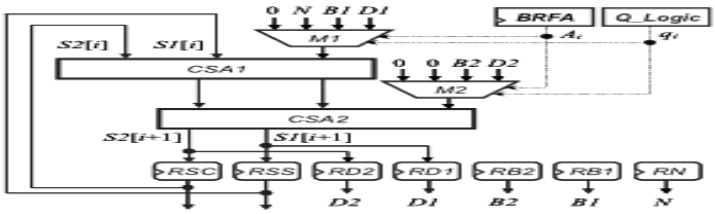


**Figure 3: MM42 Multiplier [12]**

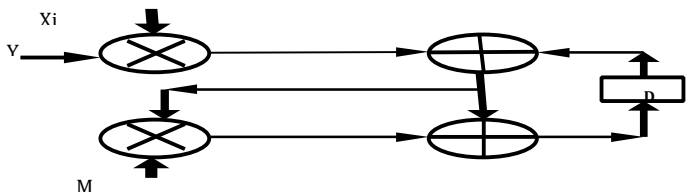


**Figure 4: Montgomery Multiplication Logic Diagram**

The DFG of circuit in Fig. 3 is given in Fig. 4, where a, b, c, d are the nodes representing the operational element.

 Copyright © 2018 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

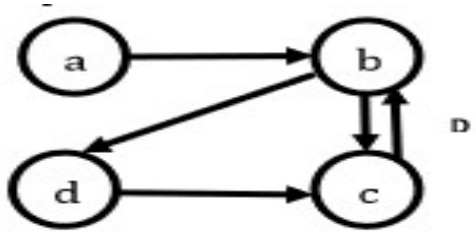


**Figure 5: DFG of Montgomery Multiplier**

Using unfolding technique, we can unfold DFG in Fig 4, by using the formula:
Ui à V(i + w) % J
Where, w is Delay, J is unfolding factor, i =0,1,…..J-1.
U is source node, V is destination node.

i) For i=0, aoàb(0+0)%2 aoàbo for i=1,
a1àb(1+0)%2
a1à b1

ii) For i=0, b0àc(0+0)%2
b0à c0
for i=1,
b1àc(1+0)%2 b1à c1

iii) For i=0, b0à d(0+0)%2 b0à d0
for i=1, b1àd(1+0)%2 b1à d1

iv) For i=0, c0àd(0+0)%2 c0àd0
for i=1,
c1àd(1+0)%2 c1à d1

v) Fori=0, d0àb(0+1)%2 d0àb1
for i=1,
d1àb(1+1)%2 d1à b0

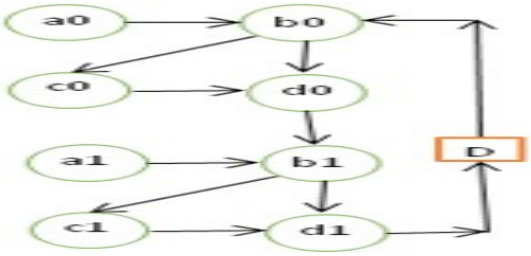


**Figure 6: Unfolded DFG of Montgomery Multiplier**

The Fig. 5 shows the unfolded architecture, for the unfolding factor J= 2. The data flow graph shows the hardware required increases by factor 2. Therefore, the area overhead is also increased. The unfolded Montgomery architecture allows us to compute the Montgomery Multiplication in „K/2" cycles as compared to „K" cycles in the original algorithm, where „K" is number of bits in inputs. This technique decreases the time delay and increase the throughput of the Multiplier. Similarly we also solve for the unfolding of Montgomery multiplier with the pre-computation approach for the unfolding factor 2 and obtain the architecture shown in Fig. 6.
The proposed and the original architectures of Montgomery multiplier are designed and implemented in VHDL. The designs are synthesized for ASIC technology in Leonardo Spectrum with TSMC 350nm technology library.

 Copyright © 2018 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

The functional and post rout simulation of the designs are also carried out on Xilinx ISE 14.5 and implemented on the Atlys Spartan 6 FPGA development board.

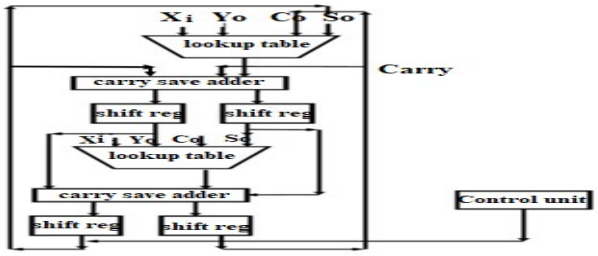


**Figure 7: Proposed Two Unfold Pre-Computed Montgomery Multiplier**

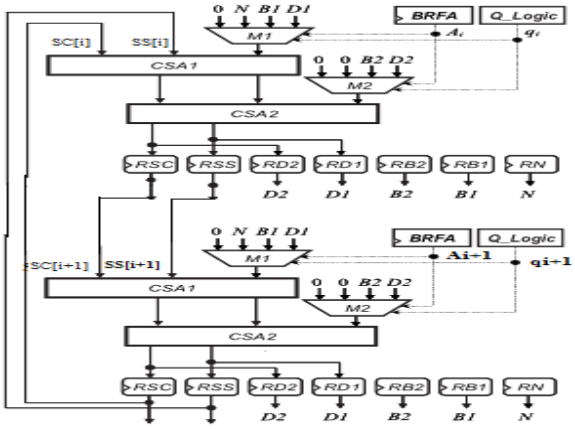


**Figure 8: Proposed Unfolding of MM24_ Multiplier in [12]**

**Results and Discussion**

This section presents the details of implementation and results obtained. All the compared architectures are designed and synthesized for FPGA and ASIC technology. The Xilinx 14.5 ISE and Mentor Graphics Leonardo spectrum tool chains are used for FPGA and ASIC implementations respectively. The Xilinx Spartan 6 xc6xlx45-3csg324 and TSMC 350 nm technology library are used as test vehicles for FPGA and ASIC performance analysis. The designs shown in Table IV are designed using VHDL and post route simulations are verified for functional check in mentor graphics tool chain. The results are shown in Table IV gives area in terms of number of gates and LUT"s for ASIC and FPGA technologies respectively. The time required for computation of one bit, given input to Montgomery multiplier is shown by computation delay. The product Area-delay is also analyzed.

It can be seen from Table IV that the pre-computation improves the performance of the Two CSA architecture. The carry save adder replaced by the lookup table in the pre-computed architecture reduces both area and delay as seen in the Table IV. The decrease is due to the reduction in the critical path. Though the decrease in area is very little, a significant decrease in delay by 19.05% and 13.87% for ASIC and FPGA respectively is observed.

| Types of Montgomery Multiplier | Area | | Critical Path delay (ns) | | Area x Delay Product (ns) | |
|---|---|---|---|---|---|---|
| | ASIC | FPGA | ASIC | FPGA | ASIC | FPGA |
| Two-CSA[8] | 437 | 56 | 4.46 | 3.46 | 1949 | 194 |
| Two-CSA [8]With proposed Unfolding | 528 | 77 | 3.38 | 2.64 | 1785 | 203 |
| Pre-computed [8] | 432 | 52 | 3.61 | 2.98 | 1560 | 155 |
| Pre-computed [8] with proposed Unfolding | 470 | 94 | 2.77 | 2.26 | 1302 | 212 |
| MM42_Multiplier[12] | 57 | 6 | 1.73 | 2.989 | 98.61 | 17.93 |

 Copyright © 2018 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

| | | | | | | |
|---|---|---|---|---|---|---|
| MM42_Multiplier[12] with proposed Unfolding | 44 | 6 | 1.54 | 2.049 | 52.36 | 12.294 |

**Table IV: Comparative Analysis of Performance Parameters on ASIC and FPGA**

The pre-computation technique gives overall improvement on area delay product by 19.95% and 20.01% for ASIC and FPGA respectively. We further observe the effect of unfolding technique on these architectures by applying the unfolding algorithm discussed in previous sections. The unfolding technique increases the hardware. Therefore, the area consumed by the unfolded architectures is more than Two CSA and precomputed Montgomery multiplier. As the unfolded architecture computes two bits at a time, the speed of the unfolded architectures has been improved than the original designs. We observed the speed improvement by 24.21% and 23.7% for Two-CSA design in ASIC and FPGA respectively. Similarly, we observe 23.2% and 24.16% of speed improvement in precomputed architecture due to unfolding technique. The overall improvement of the area delay product in ASIC is different than FPGA. We observe improvement of 8.4% and 16.5% in area delay product for ASIC technology due to unfolding. Whereas the area delay product is increased in FPGA technology by 4.4% and 26% due to the effect of unfolding. This is due to the percentage increase in the area consumption in FPGA as compared to ASIC is observed to be more when unfolding technique is applied to the architectures. We observe the percentage increase of 17.2% and 8.1% in area for ASIC for Two CSA and pre-computed architectures respectively. Whereas 27.2 % and 44.6% for FPGA for Two CSA and precomputed architectures respectively. This large increase in the area decreases the overall performance of area delay product for technology as compared to ASIC. Further, MM42_Multiplier [12] is examined applying unfolding does show improvement in delay and area. So, unfolding of MM42_Multiplier[12] show 31.44% and 10.98% decrease in delay as compared to original MM42_Multiplier[12] in FPGA and ASIC technology respectively.

The gate level net list generated in the Mentor Graphics Leonardo Spectrum is given to Pyxis layout generation tool. It uses the standard cells of TSMC 350 nm library to convert the synthesized net list to layout obtained in Fig 9. And Fig 10. The VLSI implementation and post layout simulation verifies the feasibility of proposed architectural modification in the traditional approach.

Pyxis layout of the MM42_Multiplier, is generated combining gate, flip flops , routing them and placing using transistors of 0.35 micro meter size. The layout of original MM42_Multiplier and unfolded version does not show very much difference as the common functional blocks are efficiently reuse optimized.

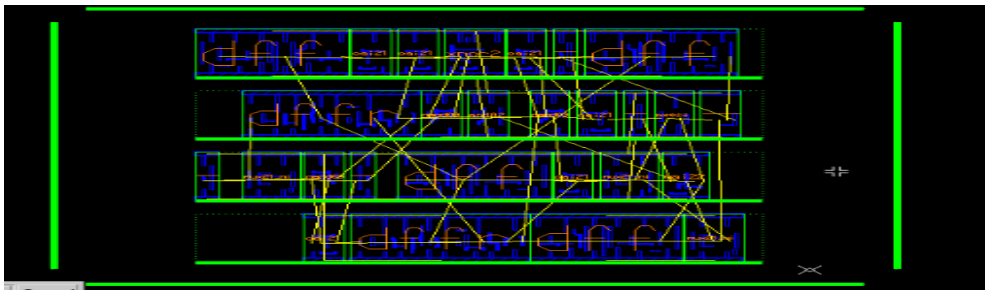

**Figure.9. Layout of MM42_ Multiplier**

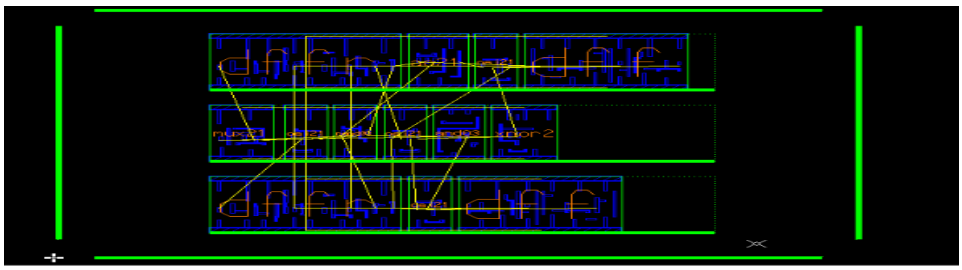

**Figure.10. Layout of Unfolded MM42_Multiplier**

 Copyright © 2018 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

## Conclusion

The VLSI implementations of Montgomery multiplication algorithm with a combined approach of pre-computation technique and unfolding technique is studied in this paper. The pre-computation in the Montgomery architecture uses only one carry save adder as compared to traditional two carry save design. Therefore, the operating frequency increases. We further improved this operating frequency using unfolding technique. The speed performance of traditional Two Carry save architectures is improved by pre-computation in [8] by 19%. We improve the performance of traditional Two CSA architecture using unfolded pre-computation architecture to 37.89% and 34.68% for ASIC and FPGA implementations. Further, unfolded version of MM42_ Multiplier gives reduced delay by 65.47 % in ASIC and 40.78% in FPGA than traditional CSA architecture and require least area.

## References


[1] P.L. Montgomery, "Modular Multiplication without Trial Division", Mathematics of Computation, Vol. 44, Number 107, pp. 519-521,April 1985.

[2] Ç.Koç, T. Acar, and B. Kaliski, "Analyzing And Comparing Montgomery Multiplication Algorithms", *IEEE Micro*, Vol. 16, pp. 26-33, June 1996.

[3] C.D. Walter, "Montgomery Exponentiation Needs No Final Subtraction", Electronic Letters, Vol. 35, pp. 1831-1832, 1999.

[4] A.Tenca and C.K. Koc, "A Scalable Architecture for Modular Multiplication Based on Montgomery‟s algorithm", IEEE Transactions on Computers, Vol. 52, pp. 1215 - 1221 Issue: 9, Sept. 2003.

[5] C. McIvor, McLoone, M., and McCanny, J.V. "Modified Montgomery modular multiplication and RSA exponentiation techniques", Proceedings of Computers and Digital Techniques, Vol. 151, pp. 402408, 2004.

[6] Kooroush Manochehri, Saadat Pourmozafari, "Fast Montgomery Modular Multiplication by Pipelined CSA Architecture", Proceedings of The 16thInternational Conference on Microelectronics, 6-8 Dec. 2004.

[7] Xin Wang, Peter Noel and Tad Kwasniewski, "Low Power Design Techniques for a Montgomery Modular Multiplier", International Symposium on Intelligent Signal Processing and Communication Systems, 13-16 Dec. 2005.

[8] Viktor Bunimov, Manfred Schimmler, Boris Tolg, "A Complexity Effective Version of Montgomery‟s Algorihm", Presented at the Workshop on Complexity Effective Designs (WECD02), May 2002.

[9] David Harris, Ram Krishnamurthy, Mark Anders, Sanu Mathew, and Steven Hsu "An Improved Unified Scalable Radix-2 Montgomery Multiplier", Proceedings of the 17th IEEE Symposium on Computer Arithmetic, 2005.

[10] DilekBayhan, S. Berna Ors, GokaySaldamli, "Analyzing and comparing the Montgomery multiplication algorithms for their power consumption", The 2010 International Conference on Computer Engineering & Systems, 30 Nov.2 Dec. 2010.

[11] Miaoqing Huang, Member, Kris Gaj, Tarek El-Ghazawi, Senior Member, "New Hardware Architectures for Montgomery Modular Multiplication Algorithm", IEEE Transactions on Computers, Vol 60, pp. 923-936, 2010.

[12] Shiann-Rong Kuang, Member, IEEE, Jiun-Ping Wang, Kai-Cheng Chang, and Huan-Wei Hsu, "Energy-Efficient High-Throughput Montgomery Modular Multipliers for RSA Cryptosystems", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.920. (2012).

[13] Shiann-Rong Kuang, Member, Kun-Yi Wu, and Ren-Yao Lu ,"Low-Cost High-Performance VLSI Architecture for Montgomery Modular Multiplication", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.(2015).

[14] M. Morales-Sandoval, A. Diaz-Perez, "Scalable GF(p) Montgomery multiplier based on a digit–digit computation approach", IET Computer. Digit. Tech., pp 102-109, Vol. 10, Issue -3, (2016).


 Copyright © 2018 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495