International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 08 | Aug -2017 www.irjet.net p-ISSN: 2395-0072

SCS-MCSA- Based Architecture for Montgomery Modular Multiplication

Pramoda B R[1], Shubha B[2], Veerabhadrappa S T[3]

[1]*M Tech Student, Dept. of ECE, JSSATE, Bengaluru, Karnataka, India*
[2]*Assistant Professor, Dept. of ECE, JSSATE, Bengaluru, Karnataka, India*
[3]*Associate Professor, Dept. of ECE, JSSATE, Bengaluru, Karnataka, India*

Abstract - *Modular multiplication is one of the important operation of many cryptographic algorithms. Montgomery modular multiplication (MMM) method is carried out in this work to solve the modular multiplication problems. This method enables real time security operations to perform faster way. The Montgomery multiplier will optimize the existing Montgomery multiplier by using Modified Carry Save Adder (MCSA) in place of configurable carry save adder and boosts the performance. In a typical Montgomery multiplier configurable carry save adder is repeatedly used for operand pre computation and format conversion from carry save format to binary format. Hence by making this pre computation operation better by using a modified carry save adder in place of configurable carry save adder leads to low hardware cost and short critical path delay. The Montgomery modular multiplication algorithm is implemented using Xilinx tool and we obtain programming results.*

Key Words: **Carry Save Adder, Low cost architecture, Montgomery Modular multiplication, Configurable CSA, Public-Key cryptosystem.**

1. INTRODUCTION

Modular multiplication is core operation of many cryptographic operation. So any improvement in this process of modular multiplication will increase the operation efficiency of the entire process. There are many algorithms proposed (eg: window MM, REDC, Montgomery MM, RSA) to improve the efficiency of this modular multiplication. The Montgomery's algorithm have been used in order to increase the efficiency of the modular multiplication operation. Montgomery achieves the quotient calculation just by shift operations and using only least significant digits of operands to produce $S = A \times B \times R^{-1} \pmod N$, where N is the k-bit modulus, R^{-1} is the inverse of R modulo N, and $R = 2k \bmod N$ [1]. The single shift operation in Montgomery Modular multiplication algorithm will reduce the time complexity and results in faster encryption and decryption. If the operands are larger value there would be longer carry propagation. So by having a methods like Full Carry Save (FCS) [2] and Semi Carry Save (SCS) [3, 4] provides the advantage of faster carry calculation leading to time complexity reduction of whole algorithm. The Semi Carry Save Configurable Carry Save Adder Based Montgomery Modular Multiplication is improvement of the semi carry save Montgomery modular multiplication, it uses one level CSA for operand pre-computation and format conversion [1]. This CCSA architecture also achieves smaller area, reduced number of clock cycle and delay. But this multiplier is better than CCSA based multiplier. In the following used Montgomery modular multiplier it is shown that higher throughput could be achieved by much smaller area-time product (ATP) than previous Montgomery multipliers.

1.1 Operation of MMM

The Fig-1 shows the semi carry save Montgomery modular multiplier using a modified carry save adder. That consists of multipliers (M1, M2, SM3, M4, and M5), D flip flops, Skip detector, Zero detector and Modified carry save adder. This system is used to reduce the numbers of clock cycle and critical path delay.

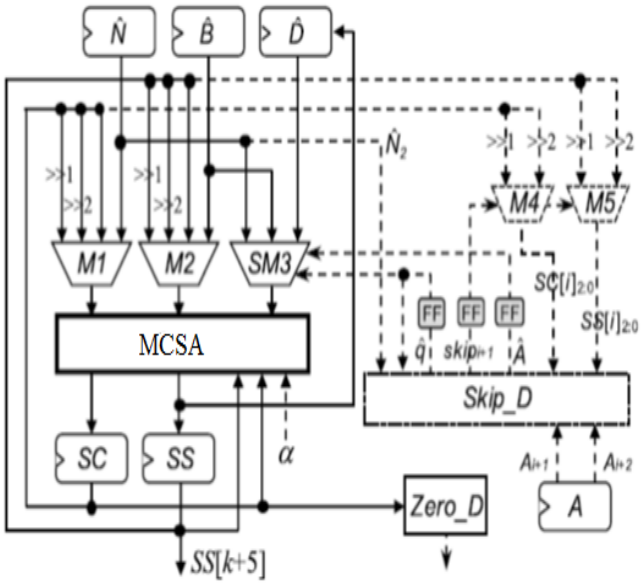

Fig-1: SCS-MCSA Based Montgomery Modular Multiplier

The variable x (output of SM3) value depends on the value of select lines Ai and qi (if select lines are "00" then x is '0', "01" then x is N, "10" then x is 'B', and "11" then x is D).

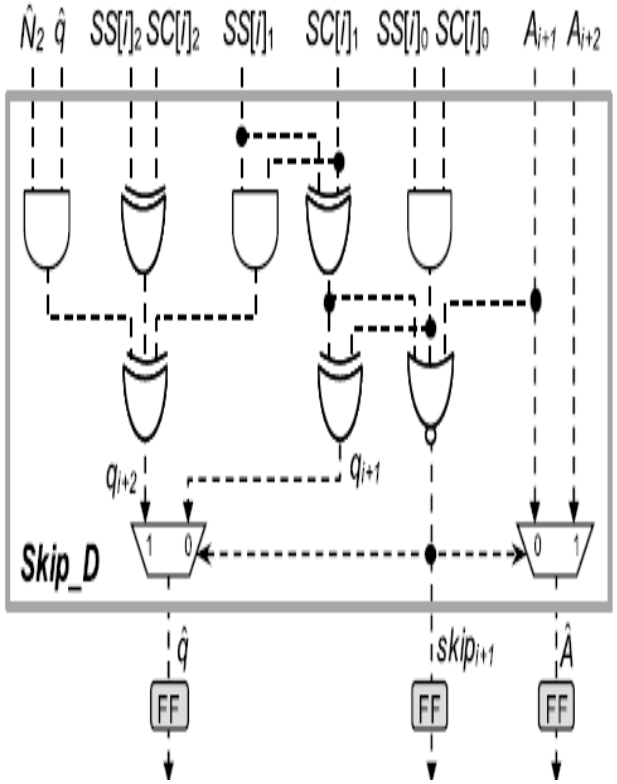

Fig-2: Skip Detector

The Fig-2 shows the Skip Detector (Skip_D). The Skip Detector is comprises with XOR gates, AND gates, a NOR gate, and 2:1 multiplexers. This produces the qi_{+1}, q_{i+2} and $skip_{i+1}$ signals

The following equations are used find the q_{i+1}, q_{i+2} and $skip_{i+1}$, the computation of q_{i+1} and q_{i+2} nothing but Quotient Pre-computation. For more information please refer [2].

$$q_{i+1} = (SS[i]_1 \oplus SC[i]_1) \oplus (SS[i]_0 \wedge SC[i]_0) \quad (1)$$
$$q_{i+2} = (SS[i+2]_1 + SC[i+2]_0) \bmod 2$$

$$q_{i+2} = (SS[i]2 \oplus SC[i]2) \oplus (qi \wedge \hat{} N2) \oplus (SS[i]1 \wedge SC[i]1 \quad (2)$$

$$\begin{aligned} Skip_{i+1} &= \sim (A_{i+1} \vee q_{i+1} \vee SS[i+1]_0) \\ &= \sim (A_{i+1} \vee (\delta_1 \oplus \delta_0) \vee \delta_1) \\ &= \sim (A_{i+1} \vee \delta_1 \vee \delta_0) \\ &= \sim (A_{i+1} \vee (SS[i]_1 \oplus SC[i]_1) \vee (SS[i]_0 \wedge S[i]\,0)) \end{aligned} \quad (3)$$

1.2. Modified Carry save Adder

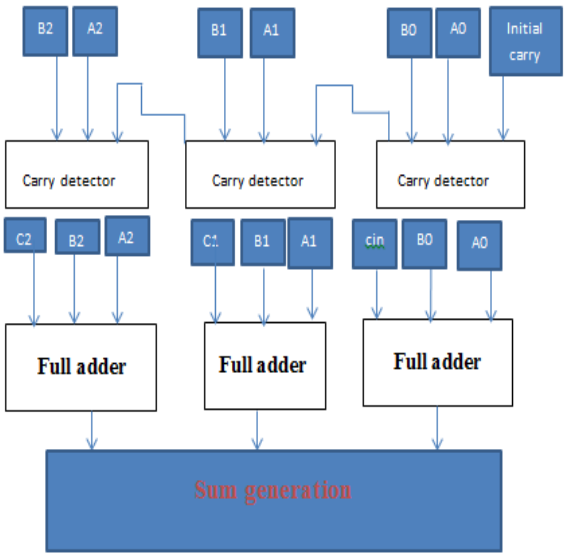

Fig-3: Modified Carry Save Adder

The modified carry save adder shown in Fig-3. It is a type of digital adder, used to find the sum of 3 or more (n) bit numbers in computer micro architecture. Basically carry are pre-calculated for all possibilities and stored in the memory, because there is no logic operation occurring to calculate the carry. So by using this approach we can reduce the number of clock cycle and operation is also fast.

2. Implementation and Algorithm

Algorithm

Inputs: A, B, $\hat{N}$ (new modulus)
Output: SS [K+5]

1. $\hat{B}$ =B<<3; $\hat{q}$=0; $\hat{A}$=0; $skip_{i+1}$=0;
2. (SS,SC)=1F_CSA($\hat{B}$, $\hat{N}$, 0);
3. While (SC!=0)
4. (SS,SC)=2H_CSA(SS,SC);
5. $\hat{D}$ =SS;
6. i= -1; SS[-1]=0; SC[-1]=0;
7. While (i ≤ k +4) {
8. if ($\hat{A}$=0 and $\hat{q}$=0) x=0;
9. if ($\hat{A}$=0 and $\hat{q}$=1) x=$\hat{N}$;
10. if ($\hat{A}$=1 and $\hat{q}$=0) x=$\hat{B}$;
11. if ($\hat{A}$=1 and $\hat{q}$=1) x=$\hat{D}$;
12. (SS[i+1],SC[i+1])=1F_CSA(SS[i],SC[i],x)>>1;
13. compute q_{i+1},q_{i+2},and $skip_{i+1}$ by (5),(7) and (8);
14. if($skip_{i+1=}$1){
15. SS[i+2]=SS[i+1]>>1,SC[i+2]=SC[i+1]>>1;
16. $\hat{q}=q_{i+2}$; $\hat{A}=A_{i+2}$; i=i+2;
17. }
18. else{
19. $\hat{q}=q_{i+1}$; $\hat{A}=A_{i+1}$; i=i+1;

20. }
21. }
22. q̂=0; Â=0;
23. while (SC[K+5]!=0)
24. (SS[K+5],SC[K+5]=2H_CSA(SS[K+5],SC[K+5]);
25. Return SS[K+5];

The semi carry save Montgomery modular multiplication algorithm using single stage modified carry save adder architecture is used to decrease the essential clock cycle for finishing single modular multiplication. Steps from 1 to 5 of this algorithm produce the $\hat{B}$ and $\hat{D}$. The values of q_{i+1} and q_{i+2} should be generated in i^{th} iteration. Where 'i' is the iterative index of MMM and its start from '-1' as a replacement of zero and initial value of $skip_{i+1,}$ q̂ and Â should be fixed to zero as shown in algorithm. When $skip_{i+1}$ is equal to '1', the system skips the unwanted iterations.Step-8 to step-12 is used to select the value of 'x' as shown in while loop of algorithm. By using equation (1), (2) and (3) we calculate the values of $q_{i+1,}$ q_{i+2} and $skip_{i+1}$ (step-13 algorithm). The step-14 to step-20 and step-8 to step-12 can be carried out simultaneously to select Â, q̂ and i. Montgomery multiplication starts with storing (skipi+1), q̂ and Â in FF and at first resets to 0 as shown in SCS-MM-New algorithm at step 1 then, $\hat{D} = \hat{B} + \hat{N}$ will be completed through one-level CCSA architecture. When while loop executes, Skip Detector (Skip_D) will be not executed and move to skipi+1, q̂ and Â as shown in figure 2. The Skip Detector is comprises with XOR gates, AND gates, a NOR gate, and 2:1 multiplexers. This produces the qi_{+1}, q_{i+2} and $skip_{i+1}$ signals in the i^{th} iteration according to equation (1), (2) and (3), and at that point choose the proper q̂ and Â based on $skip_{i+1,}$ and these values are stored in flip flops at the end of the iteration. If the value of skipi+1 1s equal to zero, SS_1 and SC_1 are selected else SS_2 and SC_2 are selected. I.e. one bit right shift operations in step 12 and step 15 of algorithm are executed simultaneously in following clock cycle of iteration i. Additionally, the inputs multiplexer 4 and 5 yields the correct output SC[i] 2:0 and SS[i] 2:0 by using $skip_{i+1}$. The SC[i] 2:0 and SS[i] 2:0 can also be gained from multiplexer 1 and multiplexer 2 but a longer delay is required because there are 4-to-1 multiplexers. The step 23 and step 24 execute the format conversion, it's similar to the operand precomputation as shown in step 3 and step 4 in the algorithm. At the end SS [K+5] is the output of the multiplier when SS [K+5] =0.

3. Result

The Fig 4, 5 and 6 shows RTL diagrams of SCS Modified Carry Save Adder based Montgomery modular multiplication and simulation result respectively. This system requires less area, critical path delay and number of clock cycle than SCS Configurable carry save adder based Montgomery modular multiplication. We can see available and utilization of flip flops, input LUT's and slices in area report. It takes 8.203 ns for completing one modular multiplication as shown in the timing report.

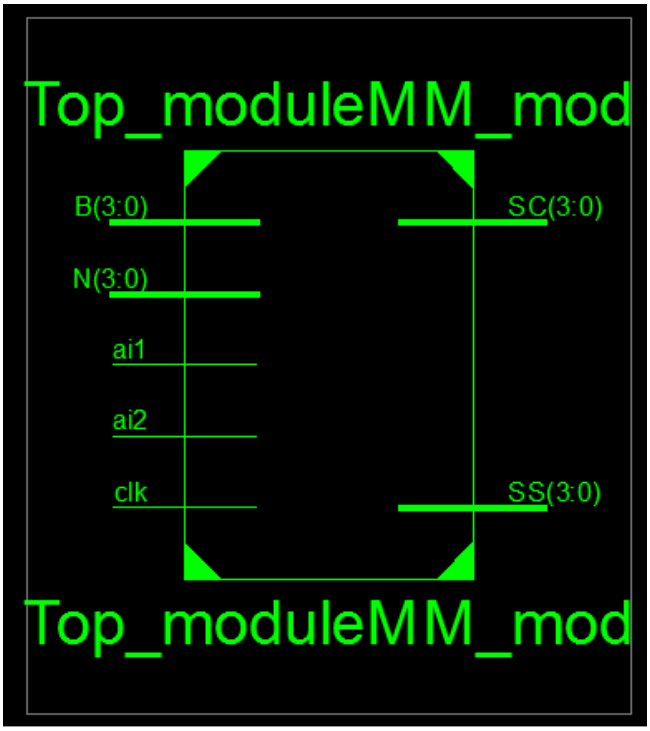

Fig-4: RTL Diagram of Proposed System

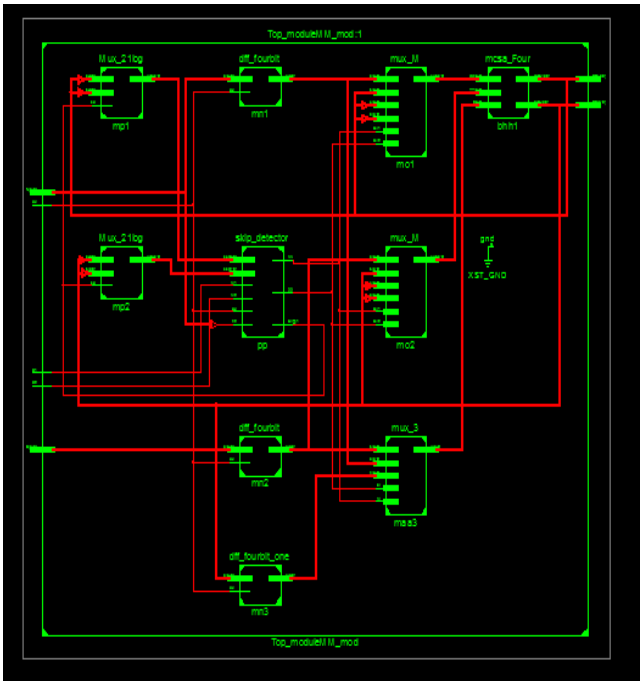

Fig-5: SCS- MCSA- based MMM detailed RTL

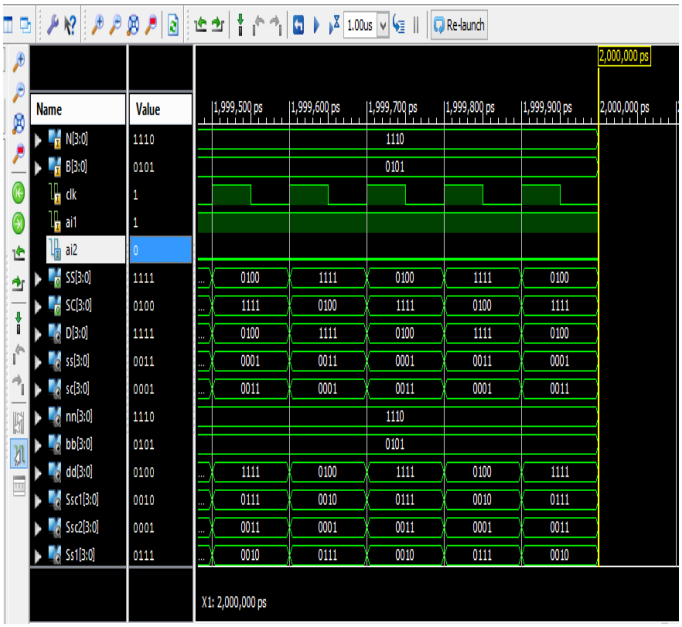

Fig-6: Simulation Report

Area report

Device utilization summary:

- Selected Device: 6s1xtqg144-3
- Slice Logic Utilization:
- Number of Slice Registers: 14 out of 11440
- Number of Slice LUTs: 27 out of 5720
- Number used as Logic: 27 out of 5720

Slice Logic Distribution:

- Number of LUT Flip Flop pairs used: 35
- Number with an unused Flip Flop: 21 out of 35
- Number with an unused LUT: 8 out of 35
- Number of fully used LUT-FF pairs: 6 out of 35
- Number of unique control sets: 3

IO Utilization:

- Number of IOs: 17
- Number of bonded IOBs: 17 out of 102

Specific Feature Utilization:

- Number of BUFG/BUFGCTRLS: 1 out of 16

Timing Report

Timing Report:

- Minimum period: 11.420ns (Max Freq: 87.568MHZ)
- Minimum input arrival time before clock: 3.484ns
- Maximum output required time after clock: 8.203ns
- Maximum combinational path delay: No path found

Timing Details:

- All values displayed in nanoseconds (ns)

4. CONCLUSIONS

By using a Full Carry Save based system, the necessity to have a format conversion is eliminated. Which led to a fewer clock cycle operation in operand precomputation stage. The proposed method modified the SCS based Montgomery multiplication and enhanced the computation speed. Thus enhancing the overall Montgomery multiplication operation and on the other hand reducing the hardware complexity. This was achieved by modifying the SCS-based Montgomery multiplication algorithm and proposed a SCS-MCSA-Based Architecture for Montgomery Modular Multiplication. The modification was by using one-level MCSA architecture and skipped the unnecessary carry-save addition operations to largely reduce the critical path delay and required clock cycles for completing one Montgomery Multiplication operation while keeping hardware complexity to minimum.

REFERENCES

[1]. S. R. Kuang, K. Y. Wu and R.Y. Lu, "Low –Cost High Performance VLSI Architecture for Montgomery Modular Multiplication"vol.24, no. 2, Feb. 2016.

[2]. C. McIvor, M. Mc Loone and J. V. Mc Canny, "Modified montgomery modular multiplication and RSA exponentiation techniques," IEEE Proc. Comput. Digit. Techn. vol. 151, no. 6, pp. 402–408, Nov. 2004.

[3]. Y. S. Kim, W. S. Kang, and J. R. Choi, "Asynchronous implementation of 1024-bit modular processor for RSA cryptosystem," in Proc. 2nd IEEE Asia-Pacific Conf. ASIC, pp. 187–190, Aug. 2000.

[4]. Y. Y. Zhang, Z. Li, L. Yang, and S. W. Zhang, "An efficient CSA architecture for Montgomery modular multiplication," Microprocessors Microsyst., vol. 31, no. 7, pp. 456–459, Nov. 2007.

[5]. V. S. Miller, "Use of elliptic curves in cryptography," in Advances in Cryptology. Berlin, Germany: Springer-Verlag, pp. 417–426, 1986.

[6]. P. L. Montgomery, "Modular multiplication without trial division," Math. Compute. vol. 44, no. 170, pp. 519–521, Apr. 1985.

[7]. H. Zhengbing, R. M. Al Shboul, and V. P. Shirochin, "An efficient architecture of 1024-bits crypto processor for RSA cryptosystem based on modified Montgomery's algorithm," in Proc. 4th IEEE Int. Workshop Intel Data Acquisition Adv. Compute. Syst. pp. 643–646. Sep. 2007.

[8]. S. R. Kuang, J. P. Wang, K. C. Chang, and H. W. Hsu, "Energy-efficient high-throughput Montgomery modular multipliers for RSA cryptosystems," IEEE Trans. Very Large Scale Integer. (VLSI) Syst. vol. 21, no. 11, pp. 1999–2009, Nov. 2013.