International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 06 | June 2019 www.irjet.net p-ISSN: 2395-0072

VLSI Architecture for Montgomery Modular Multiplication

Minnu Varghese [1], Sanil K Daniel [2], R Nandakumar[3]

[1]M.Tech Student, Department of ECE, GEC Idukki, Kerala, India
[2]Assistant Professor, Department of ECE, GEC Idukki, Kerala, India
[3]Scientist/Engineer 'D' NIELIT Calicut, Kerala, India

Abstract - *In many public-key cryptosystem, like RSA (Riverst, Shamir and Adelman), modular multiplication (MM) with large integers is the most critical and time consuming operation [1]. This project addressed design, characterization and validation of MM multipliers for VLSI. Comparative analysis of 4 such MM multipliers viz Radix-2 Montgomery modular Multiplication, SCS-based Montgomery Multiplication, FCS-based Montgomery Multiplication and Modified SCS-based Montgomery Multiplication was done. Architecture was implemented to obtain a delay of 3.59ns, power of 20.36mW and frequency of 278MHz. Montgomery algorithms were simulated using ModelSim PE Student Edition 10.4a and it was implemented on Xilinx Spartan-6 FPGA(xc6slx16-2csg324).*

***Key Words*: Cryptography, ModelSim, Montgomery modular multiplication.**

1. INTRODUCTION

SEVERAL network services such as online shop, ecommerce, electronic mail, electronic shop for video, music is expected to make life more secure and helpful. Cryptographic algorithms are convenient tool for achieving security in those systems. To realize the network encryption technologies are used. Of the two broad category of cryptographic system as public key cryptosystem(PKC) and private key cryptosystem (secret key cryptosystem).Private key cryptography that uses same keys for encryption of plaintext and decryption of cipher text. Of these, public key cryptosystem are widely used because use of entirely different keys for encryption and decryption [1]-[3]. There are two types of PKC algorithm which are RSA (Riverst, Shamir, and Adelman) and DSA (Digital signature algorithm), among these RSA algorithm used by modern computers to encrypt and decrypt the message. In RSA, amount of computation is infeasible to perform in order to reduce computational P L Montgomery introduce new method which is Montgomery modular multiplication. Montgomery algorithm is one the most well-known modular multiplication (MM) algorithms. The algorithm incorporates a series of shifting modular addition by replacing complicated division and to produce $S=A\times B\times R^{-1} \pmod N$, where N is a k bit odd prime number, R^{-1} is inverse of R modulo N and $R=2^k$. There is long carry propagation in the calculation of intermediate result as shown in step 4 of MM algorithm. There are several approaches to

---

Algorithm MM:

Radix-2 Montgomery modular multiplication

Inputs: A, B, N (modulus)

Output: S[k]

3575. S[0]=0;

2. for i =0 to k-1{

3. $q_i = (S[i]_0 + A_i \times B_0) \bmod 2$;

4. $S[i+1] = (S[i] + A_i \times B + q_i \times N) / 2$; }

5. if(S[k] ≥ N) S[k] =S[k]-N;

6. return S[k];

Fig 1.Montgomery MM Algorithm

Speed up Montgomery modular multiplication. These algorithms come under the broad category of semi carry save (SCS) strategy and full carry save (FCS) strategy [9]. In the SCS strategy [4]-[8], inputs and output operands (A, B, N and S) of the Montgomery modular multiplication are represented in binary but intermediate result S[i+1] is in carry save format. So in the final step, a conversion is required from carry save format to binary. For this conversion carry propagation adder or iterative use of carry save adder can be adopted. FCS strategy, all the operands are kept in carry save format, so no need of format conversion. It reduces the number of clock cycles needed for completing one modular multiplication, it increase hardware complexity and critical path of entire system. Accordingly these papers make a simple VLSI architecture for Montgomery multiplication algorithm such that the less architecture and high-performance Montgomery modular multiplier can be analyzed by comparative study of different Montgomery MM algorithm.

The reminder of paper is organized as follows. Section Ⅱ briefly explain different radix-2 Montgomery MM algorithm. In Section Ⅲ propose a simple VLSI architecture for Montgomery multiplication algorithm such that the less architecture and high-performance Montgomery modular multiplier can be analyzed by comparative study of different Montgomery MM algorithm in SectionⅥ. Finally, the conclusion is drawn in Section Ⅴ.

2. MONTGOMERY MM ALGORITHM

2.1 Montgomery MM Algorithm

Fig. 1 shows the radix-2 version of the Montgomery MM algorithm (denoted as MM algorithm) [9]. Montgomery's algorithm determines the quotient only depending on the least significant digit of operands and replaces the complicated division in conventional MM with a series of shifting modular additions 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 $R = 2^k \bmod N$. Since the convergence range of S in MM algorithm is $0 \le S \le 2N$, an additional operation S = S -N is required to remove the oversize residue if $S \ge N$. To eliminate the final comparison and subtraction in step 6 [10] of Fig 1 changed the number of iterations and the value of R to k + 2 and $2^{k+2} \bmod N$, respectively.

Nevertheless, the long carry propagation for the very large operand addition still restricts the performance of MM algorithm. Critical delay of radix-2 Montgomery multiplier is calculation of summation but one parameter of this delay is to calculate qi and multiply with n and then add the result to summation of the two other parameters. Based on the representation of input and output operands, these approaches can be roughly divided into semi-carry-save (SCS) strategy and full carry-save (FCS) strategy.

2.2 SCS-Based Montgomery Multiplication Algorithm

In the SCS strategy [4], the input and output operands (i.e., A, B, N, and S) of the Montgomery MM are represented in binary, but intermediate results of shifting modular additions are kept in the carry-save format to avoid the carry propagation. However, the format conversion from the carry-save format of the final modular product into its binary representation is needed at the end of each MM. This conversion can be accomplished by an extra carry propagation adder (CPA) or reusing the carry-save adder (CSA) architecture iteratively.

To avoid the long carry propagation, the intermediate result S of shifting modular addition can be kept in the carry-save representation (SS, SC)[11].The number of iterations has been changed from k to k + 2 to remove the final comparison and subtraction However, the format conversion from the carry-save format of the final modular product into its binary format is needed, as shown in step 6.

Algorithm SCS-based MM:

SCS-based Montgomery multiplication

Inputs : A, B, N (modulus)

Outputs: S [k+2]

1. SS[0]=0; SC[0]=0;
2. for i =0 to k+1{
3. $q_i=(SS[i]_0+SC[i]_0+A_i\times B_0) \bmod 2$;

4. (SS[i+1],SC[i+1]) = (SS[i]+SC[i]+A_i×B+q_i×N)/2;}
5. S[k+2]= SS[k+2]+SC[k+2];
6. return S[k+2]

Fig 2. SCS-based Montgomery multiplication algorithm

2.3 FCS-Based Montgomery Multiplication Algorithm

In FCS strategy [5], maintains the input and output operands A, B, and S in the carry-save format, denoted as (AS, AC), (BS, BC), and (SS, SC), respectively, to avoid the format conversion, leading to fewer clock cycles for completing a MM. Nevertheless, this strategy implies that the number of operands will increase and that more CSAs and registers for dealing with these operands are required. Therefore, the FCS-based Montgomery modular multipliers possibly have higher hardware complexity and longer critical path than the SCS-based multipliers. To avoid the format conversion, FCS-based Montgomery multiplication maintains A, B, and S in the carry save representations (AS, AC), (BS, BC), and (SS, SC), respectively[12].

Algorithm FCS –MM:

FCS-based Montgomery multiplication

Inputs: AS, AC, BS, BC, N (modulus)

Outputs: SS[k+2], SC[k+2]

1. SS[0] =0; SC[0]=0;
2. for i = 0 to k+1 {
3. q_i=(SS[i]$_0$+SC[i]$_0$+A_i×(BS_0+BC_0)) mod 2;
4. (SS[i+1],SC[i+1])=(SS[i]+SC[i]+A_i×(BS+BC)+q_i×N)/2;}
5. return SS[k+2], SC[k+2];

Fig 3.FCS-based Montgomery multiplication algorithm

3. PROPOSED MONTGOMERY MULTIPLICATION

Propose a new SCS-based Montgomery MM algorithm to reduce the critical path delay of Montgomery multiplier. In addition, the drawback of more clock cycles for completing one multiplication is also improved while maintaining the advantages of short critical path delay and low hardware complexity.

Algorithm Modified SCS-MM:

Modified SCS-based Montgomery multiplication

Inputs: A, B, N (modulus)

Outputs: SS[k+2]

1. (SS,SC) = (B+N+0);
2. While (SC!=0)
3. (SS,SC) = (SS+SC+0)

4. D =SS;
5. SS[0]=0; SC[0]=0;
6. for i = 0 to k+1 {
7. qi =$(SS[i]_0+SC[i]_0+A_i \times B_0)$ mod 2;
8. if (Ai =0 and qi =0) x=0;
9. if (Ai =0 and qi =1) x=N;
10. if (Ai =1 and qi =0) x=B;
11. if (Ai =1 and qi =1) x=D;
12. (SS[i+1],SC[i+1]) =(SS[i]+SC[i]+x)>>1;}
13. while (SC[k+2]!=0)
14. (SS[k+2],SC[+2]) =(SS[k+2]+SC[k+2]+0);
15. return SS[k+2];

Fig 4. Modified SCS-based Montgomery modular multiplication

The critical path delay of SCS-based multiplier can be reduced by combining the advantages of FCS-MM and SCS-MM. That is, we can precomputed D = B + N so that the computation of Ai × B + qi × N in step 4 of SCS and FCS can be simplified into one selection operation. One of the operands 0, N, B, and D will be chosen if (Ai , qi) = (0, 0), (0, 1), (1,0), and (1, 1) respectively. As a result, only one-level CSA architecture is required in this multiplier to perform the carry-save addition at the expense of one extra 4-to-1 multiplexer and one additional register to store the operand D.

4. EXPERIMENTAL RESULTS

In this section we analyze the critical path delay and area of Modified SCS-MM. Then area and delay are compared with that of previous designs. In addition, the power of different Montgomery multipliers are also measured. Finally, several Montgomery multipliers are implemented and synthesized to demonstrate the efficiency of the proposed approach.

4.1 Implementation Results

To verify efficiency of proposed design, we synthesized those Montgomery modular multiplication listed in Table 1 and Table 2 by Xilinx Spartan-6 FPGA(xc6slx16-2csg324). The behavioral description of the design is written in verilog HDL and simulated using ModelSim PE Student Edition 10.4a platform. Design verification is performed with the help of ChipScope ILA tool of Xilinx. Power utilization is analyzed by power report. The implementation results, includes the critical path delay (Delay), dynamic and static power (Power) and Frequency which is the inverse of delay are given in the Table 1. The performance of various Montgomery Modular Multiplication designs is presented in Table 1.The performance of Modified SCS-based Montgomery Multiplication design is compared with other design on the basis of delay, frequency and power. Delay and power of Modified design is very low compared with other design.

Sl.No	Multiplier	Delay (ns)	Power (mW)	Frequency(MHz)
01	Radix-2 Montgomery modular Multiplication	4.277	20.91	233.8
02	FCS-based Montgomery Multiplication	5.837	20.96	171.3
03	SCS-based Montgomery Multiplication	4.979	20.92	200.8
04	Modified SCS-based Montgomery Multiplication	3.597	20.36	278

Table1: Comparisons of different Montgomery multipliers

The design summery of various Montgomery Modular Multiplication is presented in Table2.Based on the different parameter, we analyze that modified design has less architecture compared to other design. Less architecture means it has less area complexities.

Sl.No	Multipliers	No. of Slice Register	No. of Slice LUTs	No. of Slice used as logic	No. of Slice used as memory	No. of IOBs	Total
01	Radix-2 Montgomery modular Multiplication	547	3075	3021	54	17	6714
02	FCS-based Montgomery Multiplication	589	3961	3905	56	19	8530
03	SCS-based Montgomery Multiplication	589	3789	3733	56	19	8186
04	Modified SCS-based Montgomery Multiplication	360	3089	3089	56	19	6612

Table 2: Resource Utilization Summery

Device utilization summery includes slice logic utilization, logic distribution, IO utilization and specific feature utilization. From the design summery, Radix-2 multiplier that it uses 547 slice registers , 3075 LUTs,3021 slice logic, no of bonded IO as 17,no of Memory used as 54.Only 3% of slice register is utilized, 33% of logic used, IO used as 7% and 2% of memory is utilized. SCS-based, that uses 589 slice registers ,3789 LUTs,3733 slice logic, no of bonded IO as 19,no of Memory used as 56.Only 3% of slice register is utilized, 40% of logic used, IO used as 8% and2% of memory is utilized. Compared to radix-2 utilization of registers is same but more area used than radix-2. FCS-based contains 589 slice registers, 3961 LUTs,3905 slice logic, no of bonded IO as 19,no of Memory used as 56.Only 3% of slice register is utilized, 42% of logic used, IO used as 8 % and 2% of memory is utilized.

From this we can understand that its architecture is utilized more so that its area is also high. Compared to SCS-based utilization of register is same but FCS has more area complexities than SCS. Modified SCS multiplier contains360 slice registers , 3081 LUTs,3089 slice logic, no of bonded IO as 74.Only 1% of slice register is utilized, 33% of logic used, IO used as 31% utilized. Modified design has less architecture compared to other design. Less architecture means it has less area.

5. CONCLUSION

This project work proposes the design and implementation of the VLSI architecture for Montgomery Modular Multiplication. FPGA implementation of MM is presented in this report. The behavioral description of the design is written in verilog HDL and simulated using ModelSim PE Student Edition 10.4a platform. Then the design is successfully implemented on Xilinx Spartan-6 FPGA(xc6slx16-2csg324). Design verification is performed with the help of ChipScope ILA tool of Xilinx. Different algorithms are numerically verified and they are compared with values obtained using on-line calculator. Power utilization is analyzed by power report. The maximum frequency of operation is obtained as 278MHz with 3.59ns delay and a power consumption of 20.36mW. Compared to other architecture this design has less hardware complexity.

ACKNOWLEDGMENT

This work is supported by National Institute of Electronics and Information Technology (NIELIT), Calicut and government engineering college idukki.

REFERENCES

[1]R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, no. 2,pp. 120–126, Feb. 1978.

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

[3] N. Koblitz, “Elliptic curve cryptosystems,” *Math. Comput.*, vol. 48,no. 177, pp. 203–209, 1987.

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

[5] 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*, Aug. 2000, pp. 187–190.

[6] V. Bunimov, M. Schimmler, and B. Tolg, "A complexity-effective version of Montgomery's algorihm," in *Proc. Workshop Complex. Effective Designs*, May 2002.

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

[8] 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.

[9] Shiann-Rong Kuang, "Low-Cost High-Performance VLSI Architecture for MontgomeryModular Multiplication" *IEEE Trans.VLSI Syst.*, vol. 24, no. 2, February 2016

[10] C. D. Walter, "Montgomery exponentiation needs no final subtractions,"*Electron. Lett.*, vol. 35, no. 21, pp. 1831–1832, Oct. 1999.

[11] C. McIvor, M. McLoone, and J. V. McCanny, "Modified Montgomery modular multiplication and RSA exponentiation techniques," *IEE Proc.-Comput. Digit. Techn.*, vol. 151, no. 6, pp. 402–408, Nov. 2004.

[12] 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 Integr. (VLSI) Syst.*, vol. 21, no. 11, pp. 1999–2009, Nov. 2013.