IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018 ISSN: 2320-1363

DESIGN OF A NEW SEMI CARRY SAVE(SCS) MONTGOMERY MODULAR MULTIPLIER

M.SUDHEERKUMAR REDDY[1], M.SHIREESHA[2]
[1]PG Student, Dept of ECE(VLSI-SD), SRIT, Proddatur, AP, India.
[2]Assistant Professor, Dept of ECE, SRIT, Proddatur, AP, India.

***Abstract*— This paper proposes a simple and efficient Montgomery multiplication algorithm such that the low-cost and high-performance Montgomery modular multiplier can be implemented accordingly. The proposed multiplier receives and outputs the data with binary representation and uses only one-level carry-save adder (CSA) to avoid the carry propagation at each addition operation. This CSA is also used to perform operand precomputation and format conversion from the carry- save format to the binary representation, leading to a low hardware cost and short critical path delay at the expense of extra clock cycles for completing one modular multiplication. To overcome the weakness, a configurable CSA (CCSA), which could be one full-adder or two serial half-adders, is proposed to reduce the extra clock cycles for operand precomputation and format conversion by half. In addition, a mechanism that can detect and skip the unnecessary carry-save addition operations in the one-level CCSA architecture while maintaining the short critical path delay is developed. As a result, the extra clock cycles for operand precomputation and format conversion can be hidden and high throughput can be obtained. Experimental results show that the proposed Montgomery modular multiplier can achieve higher performance and significant area–time product improvement when compared with previous designs.**

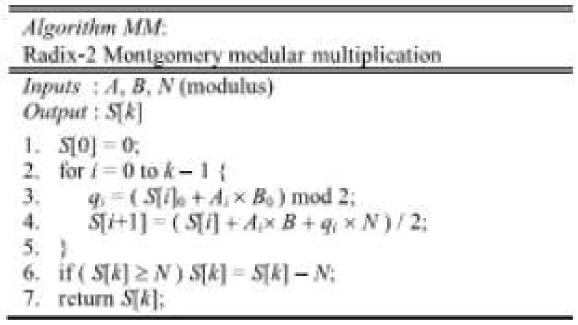
***Index Terms*— Carry-save addition, low-cost architecture, Montgomery modular multiplier, public-key cryptosystem.**

I. INTRODUCTION

Modular Multiplication (MM) with large integers is a time consuming operation in many public-key cryptosystems [1]. Therefore, many algorithms have been presented to carry out MM more quickly and Montgomery's algorithm is one of them. Montgomery's algorithm determines the quotient only depending on the least significant digit of operands [2]. It replaces the complicated division in MM with a series of shifting modular additions. Montgomery algorithm is classified into two based on the representation of input and output operands. They are Full Carry-Save Montgomery modular Multiplication (FCS-MM) and Semi-Carry-Save Montgomery modular Multiplication (SCS-MM). In FCS-MM both the obtained sum and carry are considered as output. But in SCS-MM only the obtained sum is considered as output. The adder levels in SCS-MM is less. Therefore SCS-MM requires lower area than FCS-MM. Hence SCS based multiplier is modified here. The remainder of this paper is organised as follows. Section II briefly describes about Montgomery MM algorithm. Section III briefly reviews the existing SCS based Montgomery multipliers. Section IV describes the proposed SCS based Montgomery multiplier. The comparisons of existing and proposed multipliers are made in Section V. The conclusion is drawn in Section VI.

II. MODULAR MULTIPLICATION ALGORITHMS

The Montgomery modular product S of A and B can be obtained as S = A × B × R−1 (mod N), where R−1 is the inverse of R modulo N. That is, R × R−1 = 1 (mod N). The length of A,B and N should be same. Also the value of N should be greater than A and B.



```
Algorithm MM:
Radix-2 Montgomery modular multiplication
Inputs  : A, B, N (modulus)
Output : S[k]
1.  S[0] = 0;
2.  for i = 0 to k − 1 {
3.      q_i = ( S[i]_0 + A_i × B_0 ) mod 2;
4.      S[i+1] = ( S[i] + A_i × B + q_i × N ) / 2;
5.  }
6.  if ( S[k] ≥ N ) S[k] = S[k] − N;
7.  return S[k];
```

Fig. 1 Montgomery MM Algorithm.

III EXISTING SCS BASED MULTIPLIERS

A) SCS-Based Montgomery Multiplication

The Montgomery modular product S of A and

IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018 ISSN: 2320-1363

B is obtained as S = A × B × R−1 (mod N), where R−1 is the inverse of R modulo N. That is, R × R−1 = 1 (mod N). The intermediate result S of shifting modular addition is kept in the carry save representation (SS, SC) to avoid long carry propagation. The format conversion from the carry-save format of the final modular product into its binary format is needed. The two existing SCS based Montgomery Multiplier are SCSMM1 and SCS- MM2. Fig 2 shows the architecture of SCS-based MM algorithm proposed in [3] (denoted as SCSMM1 multiplier) ,composed of two Carry Save Adders (CSA) architecture and one format converter, Carry Propagation Adder (CPA), where the dashed line denotes a 1-bit signal.

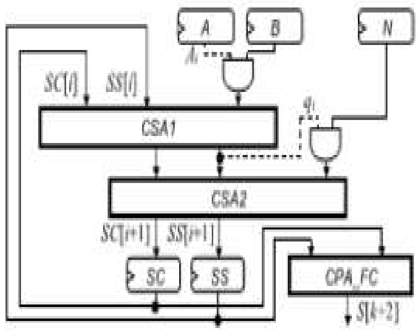


Fig. 2 SCS-MM1 multiplier

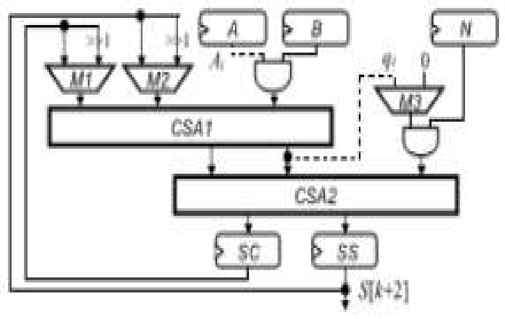


Fig. 3 SCS-MM2 multiplier

The extra CPA enlarges the area and the critical path of the SCS-MM-1 multiplier SCS MM 1 is modified by reusing the two-level CSA architecture for performing the format conversion so that the CPA can be removed. Fig. 3 shows the architecture of the Montgomery multiplier proposed in [4] (denoted as SCSMM-2 multiplier). This multiplier is modified further to reduce the critical path delay and area to increase the performance.

B) **FCS-Based Montgomery Multiplication:**

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. McIvor et al. [9] proposed two FCS based Montgomery multipliers, denoted as FCS-MM-1 and FCS-MM-2 multipliers, composed of one five-to two (three-level) and one four-to-two (two-level) CSA architecture, respectively. The algorithm and architecture of the FCS-MM-1 multiplier are shown in Figs. 5 and 6, respectively. The barrel register full adder (BRFA) consists of two shift registers for storing AS and AC, a full adder (FA), and a flip-flop (FF). For more details about BRFA, please refer to [9] and [10]. On the other hand, the FCS-MM-2 multiplier proposed in [9] adds up BS, BC, and N into DS and DC at the beginning of each MM. Therefore, the depth of the CSA tree can be reduced from three to two levels. Nevertheless, the FCS-MM-2 multiplier needs two extra 4-to-1 multiplexers addressed by Ai and qi and two more registers to store DS and DC to reduce one level of CSA tree. Therefore, the critical path of the FCS-MM-2 multiplier may be slightly reduced with a significant increase in hardware area when compared with the FCS-MM-1 multiplier.

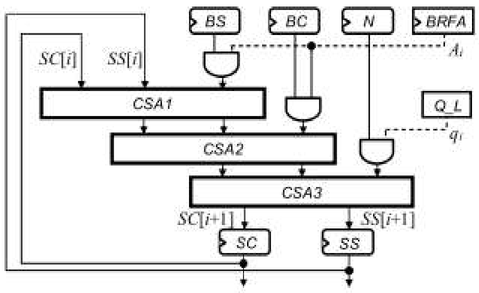


Fig 4.FCS-MM-1 multiplier

IV. Proposed Algorithm and Hardware Architecture

The critical path delay of SCS-based multiplier is reduced by pre-computing D = B + N . Two CSA's are replaced by one CSA [6]. The CSA is reused for performing B+ N and the format conversion. Fig.3 shows the hardware architecture of modified SCS-based Montgomery multiplier (MSCS-MM) . The Zero_D circuit in Fig. 3 is used to detect whether SC is equal to zero, which can be accomplished using one NOR operation.

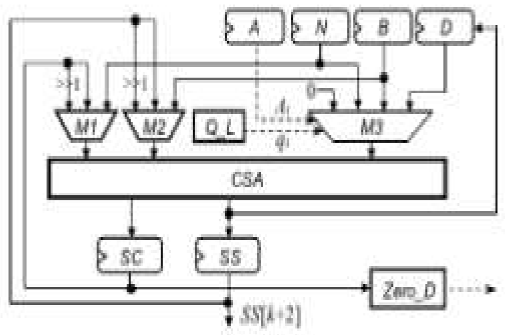


Fig.5 MSCS-MM multiplier

The carry propagation addition operations of B + N and the format conversion are performed by the one-level CSA architecture of the MSCS-MM multiplier through repeatedly executing the carry-save addition. Therefore, the critical path delay of the MSCS-MM multiplier can be reduced. The area complexity is also reduced as only one level CSA is used here. The structure of carry save adder used is shown in Fig 5. The CSA block internally consists of full adders which is realized using and gates and xor gates.

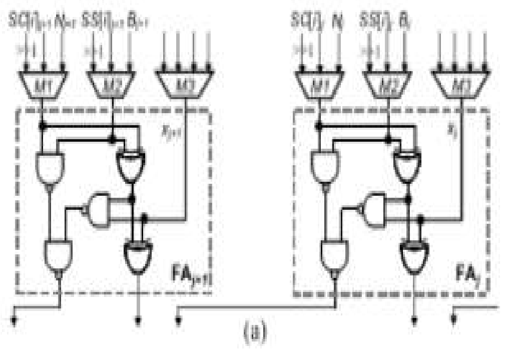


Fig. 6 Two cells of CSA

The carry save adder is used because it have less propagation delay. Carry Save adder for n-bit means it have n-parallel adders, which produce n-bit sums and n-bit carry’s. The inputs for carry save adder are SS,SC and mux output. The hardware architecture of SCS-MM-New algorithm, denoted as SCS-MM-New multiplier, are shown in Fig. 6, which consists of one one-level CCSA architecture,two4-to-multiplexers (i.e., *M1* and *M2*), one simplified

```
Algorithm SCS-MM-New:
Proposed SCS-based Montgomery multiplication
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 ( $i \le k + 4$ ) {
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. }
```

Fig. 7. SCS-MM-New algorithm

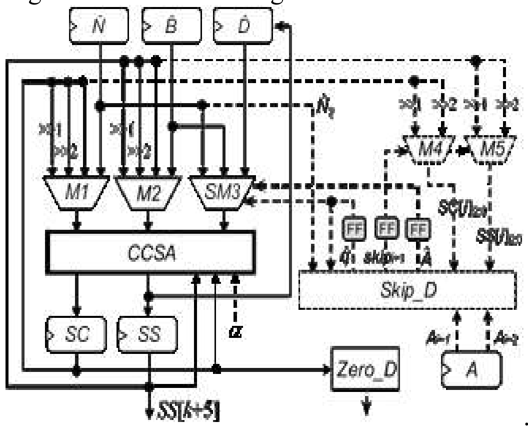


Fig.8. SCS-MM-New multiplier

multiplier *SM3*, one skip detector *Skip_D*, one zero detector *Zero_D*, and six registers. *Skip_D* is developed to generate skip$_{i+1}$, *q*, and $\hat{A}$ in the *i* th iteration. Both *M4* and *M5* in Fig. 11 are 3-bit 2-to-1 multiplexers and they are much smaller than *k*-bit multiplexers *M1*, *M2*, and *SM3*. In addition, the area of *Skip_D* is negligible when compared with that of the *k*-bit one-level CCSA architecture. Similar to Fig.5, the select signals of multiplexers *M1* and *M2* in Fig.6 are generated by the control part, which are not depicted for the sake of simplicity.

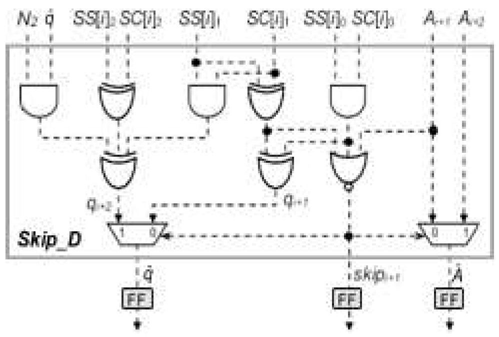


Fig.9. Skip detector *Skip_D*

V. EXPERIMENTAL RESULTS

The design of Modified SCS-MM (MSCS-MM) has been made by using Verilog VHDL. The simulation results have been evaluated by using Xilinx 14.7, for 4-bit and 8-bit. The simulation results are shown in Figures below. The critical path delay, area and power of the proposed multiplier is analyzed. This is then compared with the area, delay and power of SCS MM-2. The delay, area and power of the proposed multiplier have been decreased. Therefore the speed the proposed multiplier is increased. The below figure show the RTL schematic of proposed Montgomery modular multiplication when the RTL code is executed using Xilinx.

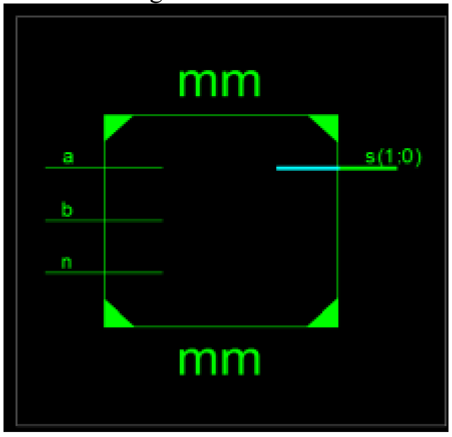


Fig.10 RTL schematic of SCS-MM-NEW multiplier

From the above module we can say that the scs-mm multiplier ha 3 inputs(a,b,n) and output(s[1:0]).

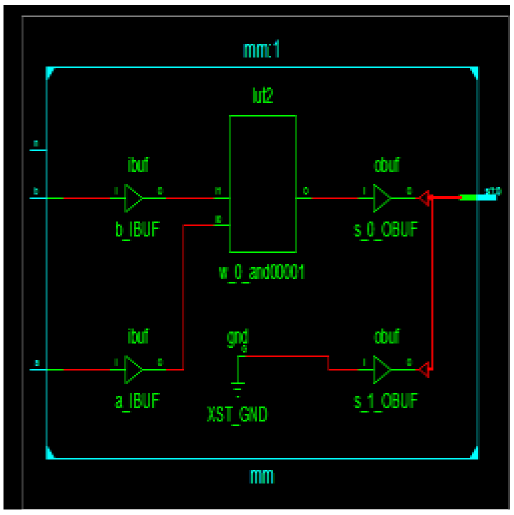


Fig.11 Technology schematic of scs-mm-new-multiplier

The figure 11 represents the internal technology schematic of proposed scs-mm-new multiplier when run using Xilinx. Which consist of lut2,two input buffers and two output buffers.

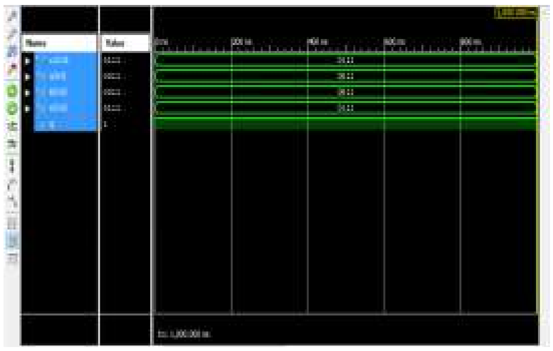


Fig.12 simulation result of Montgomery modular multiplication

The above figure show the simulation result of ssc-mm-new-multiplier when input stimulus a[3:0],b[3:0],n[3:0] is supplied to module using Verilog test feature the output ss[3:0] can be absorbed from above figure when input is applied.

The below figures show the simulation results of existing multiplier

Fig.13 simulation result of FCS-MM-1

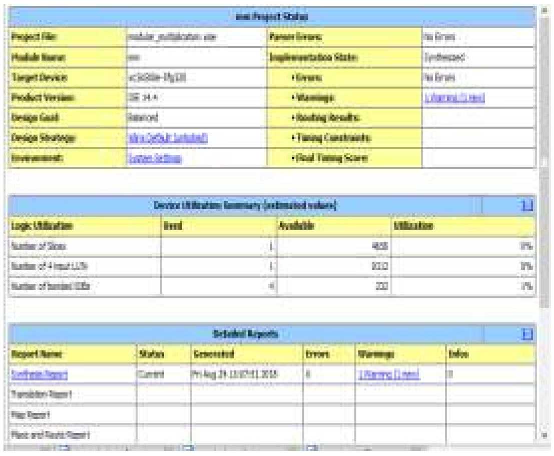
The above figure shows the simulation of fcs-mm-1 when it is executed in Xilinx.from the above figure we can say that it has input a,b,n which is of size 3 bit I,e [2:0] and output ss[2:0] .we can apply any number of inputs from test bench and absorb the result by simulating the design.

Fig.14 simulation result of MOD SCS

Figure 14 shows the simulation result of mod scs when executed the design using Xilinx software. We can use this simulation result to compare the proposed method with existing methods.

Fig.15 simulation output of SCS-MM-2

Fig 15 shows the simulation result of SCS-MM-2 multiplier which is advanced of SCS-MM-1 which is obtained by making changes in existing multiplier that is scs-mm-1.The purpose of above simulation results is to compare the existing methods with proposed method which is very much important.



mm Project Status			
Project File:	modular_multiplication.xise	Parser Errors:	No Errors
Module Name:	mm	Implementation State:	Synthesized
Target Device:	xc3s500e-4fg320	• Errors:	No Errors
Product Version:	ISE 14.4	• Warnings:	1 Warning (1 new)
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1	4656	0%
Number of 4 input LUTs	1	9312	0%
Number of bonded IOBs	4	232	1%

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Fri Aug 24 13:07:51 2018	0	1 Warning (1 new)	0
Translation Report					
Map Report					
Place and Route Report					

Fig.16 design summary of Montgomery modular multiplication

The above figure shows the design summary of proposed method which specify the number of slices ,4 input LUT'S, bonded IOBs used .and how many are available after the utilization. the proposed method used 1 slices available 4656, and 1 LUT(look up table) ,available 9312,and 4 IOBs and available 232.

VI. CONCLUSION

FCS-based multipliers maintain the input and output operands of the Montgomery MM in the carry-save format to escape from the format conversion, leading to fewer clock cycles but larger area than SCS-based multiplier. To enhance the performance of Montgomery MM while maintaining the low hardware complexity, this paper has modified the SCS-based Montgomery multiplication algorithm and proposed a low-cost and high-performance Montgomery modular multiplier. The proposed multiplier used one-level CCSA architecture and skipped the unnecessary carry-save addition operations to largely reduce the critical path delay and required clock cycles for completing one MM operation. Experimental results showed that the proposed approaches are indeed capable of enhancing the performance of radix-2 CSA-based Montgomery multiplier while maintaining low hardware complexity.

REFERRENCES

[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] 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.
[10] 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.
[11] J. C. Neto, A. F. Tenca, and W. V. Ruggiero, “A parallel k-partition method to perform Montgomery multiplication,” in Proc. IEEE Int. Conf. Appl.-Specific Syst., Archit., Processors, Sep. 2011, pp. 251–254.
[12] J. Han, S. Wang, W. Huang, Z. Yu, and X. Zeng, “Parallelization of radix-2 Montgomery multiplication on multicore platform,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 12, pp. 2325–2330, Dec. 2013.
[13] P. Amberg, N. Pinckney, and D. M. Harris, “Parallel high-radix Montgomery multipliers,” in Proc. 42nd Asilomar Conf. Signals, Syst., Comput., Oct. 2008, pp. 772–776.
[14] G. Sassaw, C. J. Jimenez, and M. Valencia, “High radix implementation of Montgomery multipliers with CSA,” in Proc. Int. Conf. Microelectron., Dec. 2010, pp. 315–318.
[15] A. Miyamoto, N. Homma, T. Aoki, and A. Satoh, “Systematic design of RSA processors based on high-radix Montgomery multipliers,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 7, pp. 1136–1146, Jul. 2011.
[16] S.-H. Wang, W.-C. Lin, J.-H. Ye, and M.-D. Shieh, “Fast scalable radix-4 Montgomery modular multiplier,” in Proc. IEEE Int. Symp. Circuits Syst., May 2012, pp. 3049–3052.
[17] J.-H. Hong and C.-W. Wu, “Cellular-array modular multiplier for fast RSA public-key cryptosystem based on modified Booth’s algorithm,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 11, no. 3, pp. 474–484, Jun. 2003.
[18] F. Gang, “Design of modular multiplier based on improved Montgomery algorithm and systolic array,” in Proc. 1st Int. Multi-Symp. Comput. Comput. Sci., vol. 2. Jun. 2006, pp. 356–359.
[19] G. Perin, D. G. Mesquita, F. L. Herrmann, and J. B. Martins, “Montgomery modular multiplication on reconfigurable hardware: Fully systolic array vs parallel implementation,” in Proc. 6th Southern Program. Logic Conf., Mar. 2010, pp. 61–66.
[20] A. Cilardo, A. Mazzeo, L. Romano, and G. P. Saggese, “Exploring the design-space for FPGA-based implementation of RSA,” Microprocessors Microsyst., vol. 28, no. 4, pp. 183–191, May 2004.
[21] D. Bayhan, S. B. Ors, and G. Saldamli, “Analyzing and comparing the Montgomery multiplication algorithms for their power consumption,” in Proc. Int. Conf. Comput. Eng. Syst., Nov. 2010, pp. 257–261.
[22] C. D. Walter, “Montgomery exponentiation needs no final subtractions,” Electron. Lett., vol. 35, no. 21, pp. 1831–1832, Oct. 1999.