IJWR

Novel Architecture for Efficient Implementation of Modular Exponentiation Algorithm

Abdalhossein Rezai*
Department of Electronical Engineering, University of science and culture, Tehran, Iran;
rezai@usc.ac.ir

ABSTRACT

One of the most difficult problems in web research is security. Cryptography is the fundamental technique utilized in secure communication. One key element of cryptography is Public-Key Cryptography (PKC). In many PKCs, the Modular Exponentiation (ME) with large modulus is a crucial process. Efficient architecture design and hardware implementation of large integer Modular Exponentiation (ME) plays a vital role in computer science such as public key cryptography. Therefore, many researchers have devoted special interest to provide efficient architecture design and hardware implementation of large integer ME. This study presents and evaluates a novel architecture for the hardware implementation of ME. To achieve the maximum architectural and timing improvements, the critical path of the Left-to-Right (LtR) and Right-to-Left (RtL) ME architectures is reorganized and reordered using a modified modular multiplication. The implementation results on a Xilinx Virtex 5 FPGA demonstrate that the developed ME architectures have a better performance in comparison with other well-known ME architectures so far in the literatures.

Keywords— Web Security, Public Key Cryptography, Modular Exponentiation Architecture, Modular Multiplication.

1. Introduction

Nowadays, infrastructures and industries must link to open access networks like the Internet. As such, one of the difficult issues in web research is security. Cryptography is the fundamental technique utilized in secure communication. One crucial element of cryptography is Public-Key Cryptography (PKC). In many PKCs, the Modular Exponentiation (ME) with large modulus is a crucial process. The Efficient architecture design and hardware implementation of large integer ME have received high attention in recent years due to their applications in computer science such as public key cryptography [1-4]. This operation computes $C= M^E \bmod N$, where N and E denote modulus and exponent, respectively, and $0 \leq M < N$. The ME is basically performed by repeating the Modular Multiplication (M2). Thus, the efficiency, throughput rate, and quantity of M2s needed determine the ME's performance completely [2-6]. Without the use of hardware acceleration, it is difficult to obtain the extremely efficient and high throughput rate for big integer ME. Therefore, many researchers [4, 5, 7-12] have devoted special interest to provide efficient architecture design and hardware implementation of large integer ME.

The Left-to-Right (LtR) and Right-to-Left (RtL) ME algorithms are typical used M2 algorithms. Several computational techniques such as common-multiplicand-multiplication technique [1, 9, 13-15] and sliding window technique [3, 8, 16] have been developed to reduce the number of required M2, but these techniques required extra area [3, 8]. So, these techniques are suitable for software or software/hardware implementation [4, 7, 9, 17].

On the other hand, Montgomery M2 (M3) [18] is a widely used M2 in the modular exponentiation. It is because in the M3, the trail division is replaced by simple right shift and addition, which are simple for hardware implementation [4, 5, 10]. To increase the efficiency of the M3, several hardware implementations have been developed that can be classified into three categories: systolic array architectures [19-22], high-radix architectures [4, 5, 7, 23-27], and scalable architectures [22, 26, 28-31].

http://dx.doi.org/10.22133/10.22133/ijwr.2024.449772.1212
Citation. A. Rezai, " Novel architecture for efficient implementation of modular exponentiation algorithm", *International Journal of Web Research*, vol.7, no.1,pp.61-67, 2024, doi: http://dx.doi.org / 10.22133/ijwr.2024.449772.1212.

*Coressponding Author
Article History: Received: 24 August 2023; Revised: 18 December 2023; Accepted: 7 January 2024.

A good review of hardware implementation for ME algorithms can be found in. [32].

Among them, Rezai and Keshavarzi [5] have proposed an efficient architecture for M2 and named it Compact Signed-Digit M2 (CSDM2) in which high-radix partial multiplication is replaced by one multi-bit shift and only one binary addition/subtraction.

This study presents a comprehensive algorithmic and architectural study on ME to utilize CSDM2 as its building block in the RtL and LtR modular exponentiation. The developed architectures are implemented on a Xilinx Virtex 5 FPGA. The FPGA implementation results indicate that the proposed architectures have advantages in comparison with other well-known modified ME architectures [4, 7, 9, 10].

The rest of this paper is as follows: section 2 briefly describes the preliminaries for the developed algorithms/architectures. Section 3 presents the proposed algorithms/architectures. Section 4 provides hardware implementation results and discussion. Finally, section 5 concludes this paper.

2. Preliminaries

2.1. M2 Algorithm/Architecture

M3 [18] is a typical used M2 in computer arithmetic. This operation speeds up the M2 by utilizing the simple right shift instead of the trial division [4, 5]. Algorithm 1 displays the binary version of M3 algorithm.

In this algorithm, the inputs are n-bit integers X, Y and N. The output is $S(n) = X.Y.R \bmod N$, where x_i denotes the ith bit of X, $S(i)$ denotes S in the ith iteration, and $R=2^{-n}$. This method computes its output in terms of n clock cycles. Thus, it is a time-consuming process [5, 13, 33].

To increase the efficiency of the M3, several hardware implementations have been developed [4, 5, 7, 19-21, 23, 24, 28-31]. Among them, Rezai and Keshavarzi [5] have proposed the CSDM2 that is an efficient M2 architecture. In the CSDM2, a multi-bit shift and only one binary addition/subtraction is utilized instead of high-radix partial multiplication. They used a new integer representation for the multiplier and named it CSD representation. In this representation, each digit contains two parts $(Type_i, Length^{(i)})$, where $Type_i$ indicate the nonzero digit and $Length^{(i)}$ denotes the consecutive zero bits count. They used the canonical recoding [34, 35] and partitioning technique [3, 16] to increase the applicability of this idea. Algorithm 2 shows the CSDM2 algorithm.

The inputs of this algorithm are Y, N, and X_{CSD}, where Y, N and X_{CSD} denote the multiplicand modulus, and CSD representation of multiplier, respectively. The output is $S=XY2^{-(n+2)} \bmod N$. Using the CSDM2, the computation of $P:=S(i)+X^{(i)}Y$ is simplified to $P:=S(i)$, $P:=S(i)-Y$, or $P:=S(i)+Y$ based on $Length^{(i)}=l+1$, $Length^{(i)}\neq l+1$ and $Type_i=1$, or $Length^{(i)}\neq l+1$ and $Type_i=0$, respectively in steps 5-8 [5]. Figure 1 shows the CSDM2 architecture [5].

This architecture contains a NAND gate, a multiplexer (Mux), two modified Barrel shifters, a 3-bit shift register, two CSAs, two XORs, three registers, and a $q^{(i)}.M$ generator [5].

2.2. ME algorithm

This operation is usually implemented by utilizing the M3 and binary methods [1-4]. The LtR and RtL ME algorithms are two well-known algorithms in the binary methods [1-4].

Algorithm 3 shows the LtR ME algorithm utilized for computing where N, E, and M<N indicates an n-bit modulus, a k_e-bit exponent, an n-bit massage.

The value of R in algorithm 3 is $2^{-(n+2)}$ or 2^{-n} based on the M2 algorithm presented in [36]. In addition, the multiplication and square operations depend on the data, and the exponent bits are read from left to right. The LtR ME algorithm performs ME algorithm

Algorithm 1: The binary M3 algorithm (Mont(X, Y, N))
Input: X, Y, N;
Output: S(n) :=X.Y.R mod N;
1. S(0)=0;
2. **For** i=0 **To** n-1
3. $q_i = (S(i)+x_i.Y) \bmod 2$;
4. $S(i+1) =(S(i)+x_iY+q_i.N)/2$;
5. **End For**
6. **If** S(n)≥N **Then** S(n)=S(n)-N;
7. **Return** S(n);

Algorithm 2: The CSDM2 algorithm (CSD(X, Y, N)) [5]
1. **Input**: X_{CSD}(*Type*, *Length*),Y, N;
2. **Output**: $S=XY2^{-(n+2)} \bmod N$;
3. **For** i=0 **To** J /* *J shows the number of digits in CSD representation* */
4. k= $Length^{(i)}$;
5. **If** $Length^{(i)} = 3$ **Then** P := S(i), k = k - 1;
6. **Else**
7. **If** $Type_i = 0$ **Then** P := S(i) + Y;
8. **Else** P := S(i) - Y;
9. $q_i = P_{k...0}(2^{k+1} - N^{-1}_{k...0}) \bmod 2^{k+1}$;
10. $S(i+1)=(P + q_iN)/2^{k+1}$;
11. **End for**
12. **Return** S;

Algorithm 3: The LtR ME algorithm (LtRME(N, M, E))
Input: N, M, E;
Output: $C :=M^E \bmod N$;
1. F=Mont(M, R^2, N);
2. S=R mod N;
3. **For** i=k_e-1**To** 0
4. S=Mont (S, S, N);
5. **If** (e_i=1) **Then** S=Mont (F, S, N);
6. **End for**
7. C=Mont(S, 1, N);
8. **Return** C;

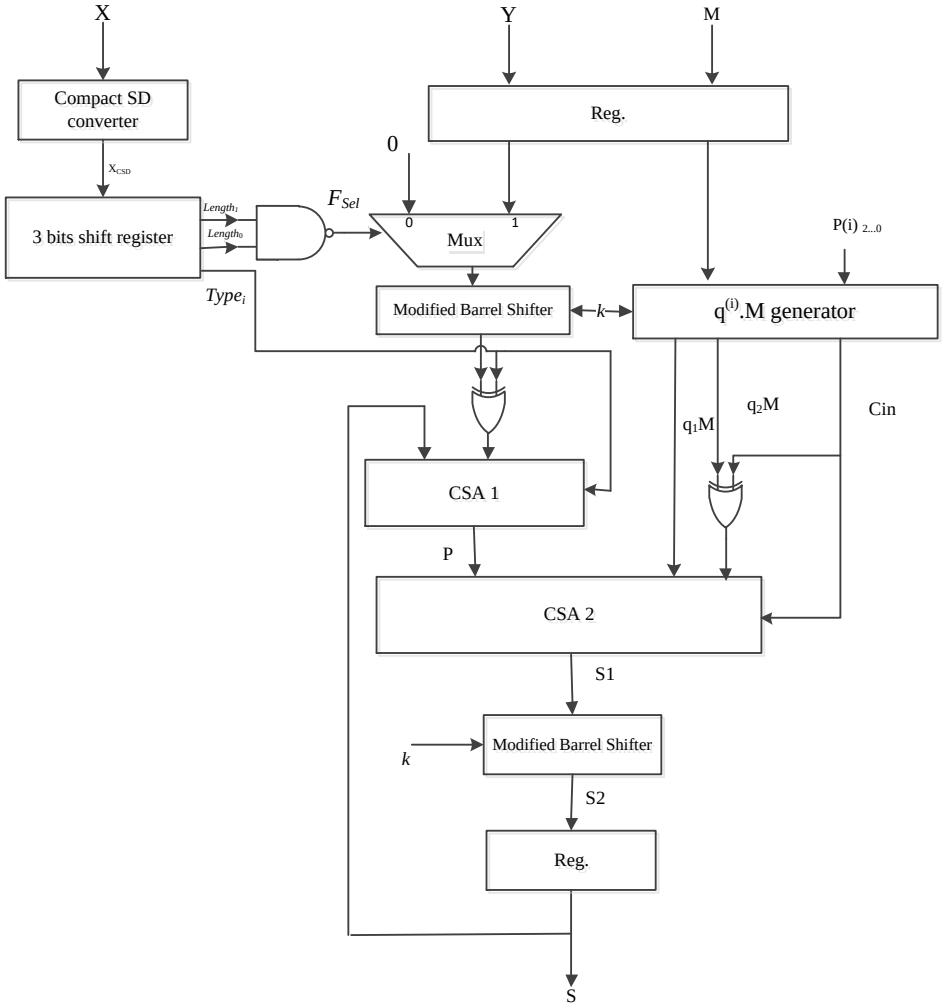

Figure. 1. The CSDM2 architecture

by using $1.5k_e+2$ multiplication operations on average [4, 7].

The RtL ME algorithm is also utilized to calculate $C= M^E \bmod N$. This algorithm scans the exponent bits from right-to-left [4, 7]. Algorithm 4 shows the RtL ME algorithm.

In this algorithm, the square and multiplication operations can run concurrently. As a result, area overhead is increased while the overall computation time is decreased. The RtL ME algorithm performs the ME algorithm by using k_e+2 multiplication operations [4, 7].

3. The proposed ME Algorithm/Architecture

In this section, a comprehensive algorithmic and architectural study on the ME is presented to achieve the maximum architectural and timing improvements, the critical path of the LtR and RtL ME architectures is reorganized and reordered using the CSDM2. So, the reformulation of the LtR and RtL ME algorithms are considered and then, the results are mapped to derive efficient ME architectures.

Algorithm 4: The RtL ME algorithm (RtLME(N, M, E))
Input: N, M, E;
Output: C :=M^E mod N;
1. F=Mont(M, R^2, N);
2. S=R mod N;
3. **For** i= 0 **To** k_e-1
4. **If** (e_i=1) **Then** S=Mont(F, S, N);
5. F=Mont (F, F, N);
6. **End for**
7. C=Mont(S, 1, N);
8. **Return** C;

3.1. The Proposed RtL CSDME

Algorithm 5 shows the developed RtL CSDME algorithm.

In the developed RtL CSDME algorithm, $R=2^{-(n+2)}$, MCSD, and R_{CSD} denote F and R in the CSD representation, respectively. It should be noted that the format conversion in the developed RtL CSDME

algorithm affects the calculation time. It is because the format conversion of F in the developed RtL CSDME algorithm is processed in parallel with previous step. More specifically, steps 4 and 8 are performed after one multiplication delay in comparison with steps 3 and 7, respectively. Figure 2 shows the proposed RtL CSDME architecture.

In the proposed RtL CSDME architecture, both multiplication operation and square operation are performed in parallel. To control the process of step 3 and step 7 of algorithm 5, the signal Select1 is used as follows: when Select1=0, step 7 is executed and when Select1=1, step 3 is executed. To control the performance of step 6 and step 10 of this algorithm, the signal Select2 is used as follows: when Select2=0, step 6 is performed, and when Select2=1, step 10 is performed. The proposed RtL CSDME algorithm performs ME algorithm by using k_e+4 multiplication operations.

3.2. The Proposed LtR CSDME

Algorithm 6 displays the developed LtR CSDME algorithm.

In this algorithm, the format conversion reasonably affects the computation time. Figure 3 shows the developed LtR CSDME architecture.

In the proposed LtR CSDME architecture, only one CSDM2 unit is used. To control the operand 1 and operand 2 in this architecture which executes steps 3, 5, 8 and 12 of the CSDME algorithm, the signal Select is used. The developed LtR CSDME algorithm performs ME algorithm by using 1.5k_e+4 multiplication operations on average.

4. Hardware Implementation and Performance Comparison

In this section, the proposed RtL and LtR CSDME architectures have been implemented using synthesizable VHDL code, and synthesized, placed and routed to Xilinx XC5VLX20T-2FF323 FPGA by executing Xilinx ISE 14.1.

The implementation results of the proposed RtL and LtR CSDME architectures compared to other well-known modified RtL and LtR ME architectures in [4, 7-10] for 1024-bit length modulus are displayed in Table 1. In this table, Method denotes the used method for performing modular exponentiation. f_{max} denotes the maximum frequency in terms of MHz. The total delay time (Time) denotes in terms of µs. The number of occupied slices for FPGA design is shown by Area. The A×D denotes the delay time by area measurement in slice×milisecond. The throughput rate is displayed in terms of Kb/s.

Based on the results that are indicated in Table 1, the developed RtL CSDME architecture has the best performance in terms of the throughput rate, and total

Algorithm 5: The developed RtL CSDME algorithm

Input: N, M, E;
Output: C :=M^E mod N;
1. Convert R^2 and 1 from binary representation to the CSD representation, R^2_{CSD}, 1_{CSD};
2. S=R mod N;
3. F=CSD(M, R^2_{CSD}, N);
4. Convert F from binary representation to the CSD representation, M_{CSD}.
5. **For** i=0**To** k_e-1
6. **If** (e_i=1) **Then** S= CSD (S, M_{CSD}, N);
7. F= CSD (F, M_{CSD}, N);
8. Convert F from binary representation to the CSD representation, M_{CSD}.
9. **End for**;
10. C= CSD (S, 1_{CSD}, N);
11. **Return** C;

Algorithm 6: The proposed LtR CSDME algorithm

Input: N, M, E;
Output: C :=M^E mod N;
1. Convert R from binary representation to the CSD representation, R_{CSD};
2. S= R_{SD};
3. F=CSD(M, R^2_{CSD}, N);
4. **For** i=k_e-1**To** 0
5. S= CSD (S, S_{CSD}, N);
6. Convert S from binary representation to the CSD representation, S_{CSD}.
7. **If** (e_i=1) **Then**
8. S= MCSD (F, S_{CSD}, N);
9. Convert S from binary representation to the CSD representation, S_{CSD}.
10. **End If**;
11. **End for**;
12. C= CSD (1, S_{CSD}, N);
13. **Return** C;

delay time in comparison with other well-known modified ME architectures in [4, 7] for 1024-bit modulus. In addition, the developed LtR CSDME architecture has a better performance in terms of the throughput rate, and total delay time compared to other modified LtR ME architectures in [4, 7, 9, 10] for 1024-bit modulus. Our developed LtR CSDME architecture has also better performance in terms of area×time complexity in comparison with LtR ME architectures in [8, 10] for 1024 modulus. The area×time complexity in our CSDME architecture is improved by about 58% and 24% in comparison with ME architecture in[10] and [8], respectively .The only ME architecture that has slightly better performance in terms of throughput and latency in comparison with our CSDME architecture is the ME architecture presented in [8]. Although the area and area×time complexity in the ME architecture in [8] are 2 and 1.33 times bigger than our CSDME architecture.

5. Conclusion

Efficient hardware implementation of computer arithmetic algorithms such ME algorithms has been in the focal point of major research efforts for the last decades. This paper presented a comprehensive algorithmic and architectural study to improve the performance of the hardware implementation of the ME algorithm. The proposed RtL and LtR CSDME architectures were implemented on Xilinx virtex 5 FPGA. The FPGA implementation results showed that the developed ME architectures provided an improvement performance in terms of throughput rate and total delay time compared to other modified exponentiation architectures in [4, 7, 9, 10].

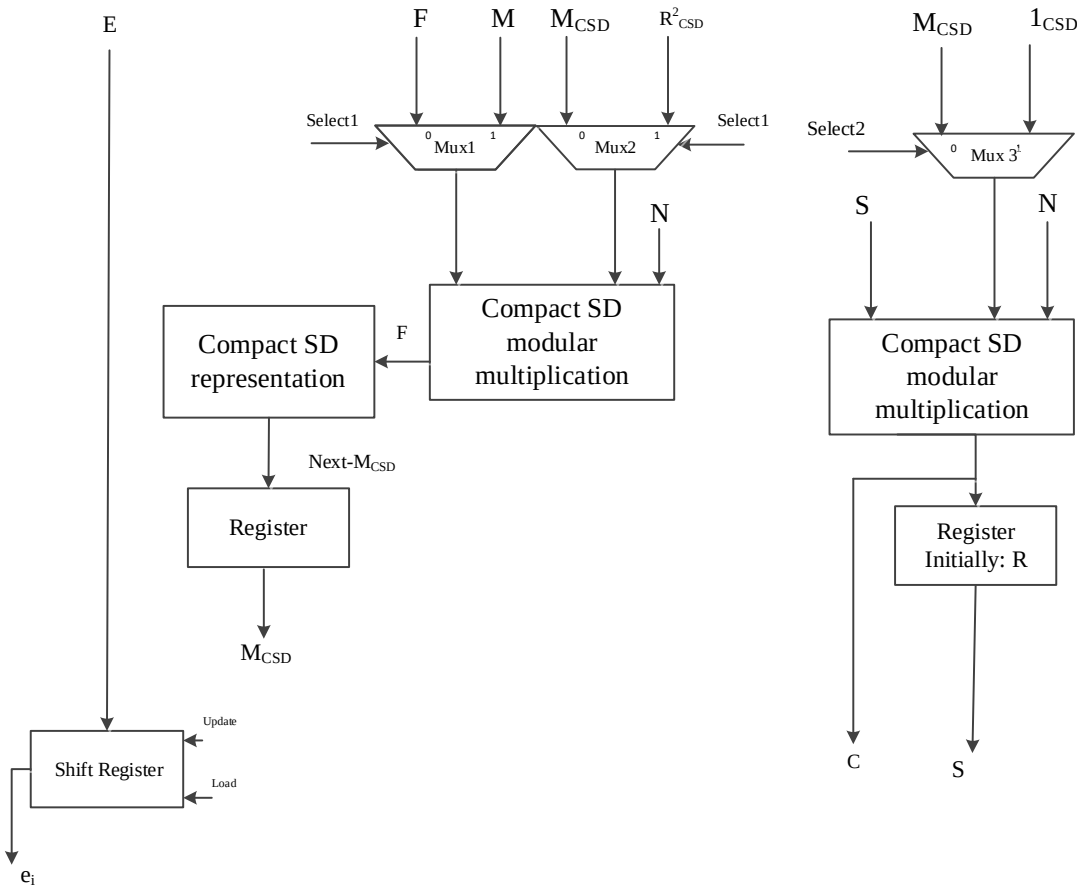

Figure. 2. The proposed RtL CSDME architecture

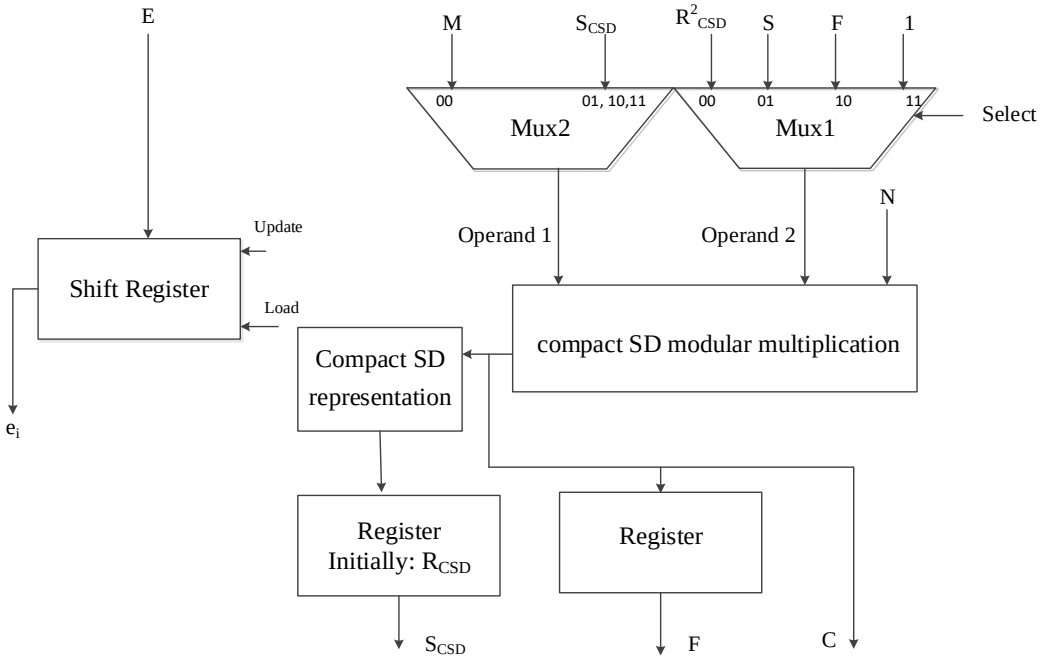

Figure. 3. The proposed LtR CSDMEarchitecture

Table 1: Comparison of ME implementations for 1024- bit length of modulus in FPGA

Reference	Method	Device	f_{max} (MHz)	Time (ms)	Area (Slice)	A×D (Slice×ms)	Throughput (kb/s)
[4] d=1	RtL	Virtex 5	526	2.98	2982	8.88	343.2
[4] d=4	RtL	Virtex 5	222	1.79	6217	11.13	572.5
[7]	RtL	Virtex 5	401	1.37	6776	9.28	747.4
This paper	RtL	Virtex 5	419	1.31	6757	8.85	783.2
[4] d=2	LtR	Virtex 5	385	1.38	7303	10.08	744.6
[9] Work II	LtR	Virtex 5	345	3.18	3218	10.23	322
[9] Work IV	LtR	Virtex 5	290	1.95	5225	10.2	525.1
[10]	LtR	Virtex 5	274	3.83	7158	27.42	267.4
[7]	LtR	Virtex 5	401	0.92	12716	11.70	1113
[8]	LtR	Virtex 6	165	0.567	26489	15.02	1805.9
This paper	LtR	Virtex 5	419	0.88	12683	11.29	1165.9

Declarations

Authors' contributions

The author did not receive support from any organization for the submitted work.

Conflict of interest

The author declares that he has no conflict of interest.

References

[1] A. Rezai and P. Keshavarzi, "Algorithm design and theoretical analysis of a novel CMM modular exponentiation algorithm for large integers," RAIRO-theoretical informatics and applications, vol. 49, no. 3, pp. 255-268, 2015. https://doi.org/10.1051/ita/2015007.

[2] M. Issad, B. Boudraa, M. Anane and N. Anane, "Software/hardware co-design of modular exponentiation for efficient RSA cryptosystem," Journal of Circuits, Systems, and Computers, vol. 23, no. 03, p. 1450032, 2014. https://doi.org/10.1142/S0218126614500327.

[3] N. Nedjah and L. de Macedo Mourelle, "High-performance hardware of the sliding-window method for parallel computation of modular exponentiations," International journal of parallel programming, vol. 37, pp. 537-555, 2009. https://doi.org/10.1007/s10766-009-0108-7.

[4] G. D. Sutter, J. P. Deschamps and J. L. Imana, "Modular multiplication and exponentiation architectures for fast RSA cryptosystem based on digit serial computation," IEEE Transactions on industrial electronics, vol. 58, no. 7, pp. 3101-3109, 2010. https://doi.org/10.1109/TIE.2010.2080653

[5] A. Rezai and P. Keshavarzi, "Compact SD: A new encoding algorithm and its application in multiplication," International journal of computer mathematics, vol. 94, no. 3, pp. 554-569, 2017. https://doi.org/10.1080/00207160.2015.1119269.

[6] B. Zhang, Z. Cheng and M. Pedram, "Design of a High-Performance Iterative Barrett Modular Multiplier for Crypto Systems," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 32, no. 5, pp. 897-910, 2024. https://doi.org/10.1109/TVLSI.2024.3368002.

[7] A. Rezai and P. Keshavarzi, "High-throughput modular multiplication and exponentiation algorithms using multibit-scan–multibit-shift technique," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 9, pp. 1710-1719, 2015. https://doi.org/10.1109/TVLSI.2014.2355854.

[8] T. Wu, "High-Performance RNS Modular Exponentiation by Sum-Residue Reduction," in IEEE Canadian Journal of Electrical and Computer Engineering, vol. 46, no. 2, pp. 137-143, 2023, https://doi.org/10.1109/ICJECE.2023.3243888.

[9] T. Wu, S. Li and L. Liu, "Fast, compact and symmetric modular exponentiation architecture by common-multiplicand Montgomery modular multiplications," Integration, vol. 46, no. 4, pp. 323-332, 2013. https://doi.org/10.1016/j.vlsi.2012.09.002.

[10] A. P. Fournaris, "Fault and simple power attack resistant RSA using Montgomery modular multiplication," in Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010: IEEE, pp. 1875-1878. https://doi.org/10.1109/ISCAS.2010.5537879.

[11] S. Vollala, "Energy efficient triple-modular exponential techniques for batch verification schemes," Journal of Cryptographic Engineering, vol. 14, pp. 295–309, 2024, https://doi.org/10.1007/s13389-024-00348-2.

[12] U. Tiwari, S. N. R. Vollala and S. Begum, "Improving the performance of authentication protocols using efficient modular multi exponential technique," Multimedia Tools and Applications, vol. 83, no. 4, pp. 11061-11076, 2024, https://doi.org/10.1007/s11042-023-15726-x.

[13] A. Rezai and P. Keshavarzi, "A new CMM-NAF modular exponentiation algorithm by using a new modular multiplication algorithm," Trends in applied sciences research, vol. 7, no. 3, p. 240-247, 2012. https://scialert.net/abstract/?doi=tasr.2012.240.247.

[14] C. L. Wu, "An efficient common-multiplicand-multiplication method to the Montgomery algorithm for speeding up exponentiation," Information Sciences, vol. 179, no. 4, pp. 410-421, 2009. https://doi.org/10.1016/j.ins.2008.10.004.

[15] J. C. Ha and S. J. Moon, "A common-multiplicand method to the Montgomery algorithm for speeding up exponentiation," Information processing letters, vol. 66, no. 2, pp. 105-107, 1998. https://doi.org/10.1016/S0020-0190(98)00031-3

[16] N. Nedjah and L. D. M. Mourelle, "A hardware/software co-design versus hardware-only implementation of modular exponentiation using the sliding-window method," Journal of Circuits, Systems, and Computers, vol. 18, no. 02, pp. 295-310, 2009. https://doi.org/10.1142/S0218126609005071.

[17] A. Rezai, M. Abbasi, and A. Karimi, "Algorithm Design and Theoretical Analysis of a New Bit Forwarding Large Integer

Modular Exponentiation Algorithm," Computational Sciences and Engineering, vol. 2, no. 2, pp. 227-238, 2022. https://doi.org/10.22124/cse.2023.23755.1041.

[18] P. L. Montgomery, "Modular multiplication without trial division," Mathematics of computation, vol. 44, no. 170, pp. 519-521, 1985.

[19] C. D. Walter, "Systolic modular multiplication," in IEEE transactions on computers, vol. 42, no. 3, pp. 376-378, 1993. https://doi.org/10.1109/12.210181.

[20] J. Xie, J. jun He, and P. K. Meher, "Low latency systolic Montgomery multiplier for finite field $ GF (2^{m}) $ based on pentanomials," IEEE transactions on very large scale integration (VLSI) systems, vol. 21, no. 2, pp. 385-389, 2012. https://doi.org/10.1109/TVLSI.2012.2185257.

[21] A. P. Fournaris and O. Koufopavlou, "A new RSA encryption architecture and hardware implementation based on optimized Montgomery multiplication," in 2005 IEEE International Symposium on Circuits and Systems, IEEE, 2005, pp. 4645-4648. https://doi.org/10.1109/ISCAS.2005.1465668.

[22] L. Noyez, N. E. Mrabet, O. Potin and P. Veron, "Montgomery Multiplication Scalable Systolic Designs Optimized for DSP48E2," ACM Transactions on Reconfigurable Technology and Systems, vol. 17, no. 1, pp. 1-31, 2024. https://doi.org/10.1145/3624571.

[23] T. Blum and C. Paar, "High-radix Montgomery modular exponentiation on reconfigurable hardware," IEEE transactions on computers, vol. 50, no. 7, pp. 759-764, 2001. https://doi.org/10.1109/12.936241.

[24] G. Sassaw, C. J. Jimenez and M. Valencia, "High radix implementation of Montgomery multipliers with CSA," in 2010 International Conference on Microelectronics, IEEE, 2010, pp. 315-318. https://doi.org/10.1109/ICM.2010.5696148.

[25] S. Immareddy, A. Sundaramoorthy and A. Alagarsamy, "Design and implementation of hybrid (radix-8 Booth and TRAM) approximate multiplier using 15-4 approximate compressors for image processing application," Journal of Real-Time Image Processing, vol. 21, no. 2, p. 50, 2024, https://doi.org/10.1007/s11554-024-01427-7.

[26] B. Zhang, Z. Cheng and M. Pedram, "High-radix design of a scalable montgomery modular multiplier with low latency," IEEE Transactions on Computers, vol. 71, no. 2, pp. 436-449, 2022. https://doi.org/10.1109/TC.2021.3052999.

[27] A. Arunachalamani, V. Venkatasubramani, V. V. Thyagarajan and S. Rajaram, "High Radix Design for Montgomery Multiplier in FPGA platform," in 2023 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI), IEEE, 2023, pp. 1-5. https://doi.org/10.1109/RAEEUCCI57140.2023.10134351.

[28] A. F. Tenca, G. Todorov and C. K. Koç, "High-radix design of a scalable modular multiplier," in Cryptographic Hardware and Embedded Systems—CHES 2001: Third International Workshop Paris, France, May 14–16, 2001 Proceedings 3, Springer, 2001, pp. 185-201. https://doi.org/10.1007/3-540-44709-1_17.

[29] A. Ibrahim, H. Elsimary and F. Gebali, "Low-power, high-speed unified and scalable word-based radix 8 architecture for montgomery modular multiplication in GF (p) and GF (2 n)," Arabian Journal for Science and Engineering, vol. 39, pp. 7847-7863, 2014. https://doi.org/10.1007/s13369-014-1363-5.

[30] A. F. Tenca and Ç. K. Koç, "A scalable architecture for modular multiplication based on Montgomery's algorithm," IEEE Transactions on computers, vol. 52, no. 9, pp. 1215-1221, 2003. https://doi.org/10.1109/TC.2003.1228516.

[31] J. H. Ye, T. W. Hung and M. D. Shieh, "Energy-efficient architecture for word-based Montgomery modular multiplication algorithm," in 2013 International Symposium onVLSI Design, Automation, and Test (VLSI-DAT), IEEE, 2013, pp. 1-4. https://doi.org/10.1109/VLDI-DAT.2013.6533882.

[32] A. E. Cohen and K. K. Parhi, "Architecture optimizations for the RSA public key cryptosystem: A tutorial," IEEE Circuits and Systems Magazine, vol. 11, no. 4, pp. 24-34, 2011. https://doi.org/10.1109/MCAS.2011.942747

[33] H. R. Ahmadi and A. Afzali-Kusha, "A low-power and low-energy flexible GF (p) elliptic-curve cryptography processor," Journal of Zhejiang University SCIENCE C, vol. 11, no. 9, pp. 724-736, 2010. https://doi.org/10.1631/jzus.C0910660

[34] G. W. Reitwiesner, "Binary arithmetic," in Advances in computers, vol. 1, pp. 231-308. https://doi.org/10.1016/S0065-2458(08)60610-5

[35] G. A. Ruiz and M. Granda, "Efficient canonic signed digit recoding," Microelectronics journal, vol. 42, no. 9, pp. 1090-1097, 2011. https://doi.org/10.1016/j.mejo.2011.06.006.

[36] C. D. Walter, "Montgomery exponentiation needs no final subtractions," Electronics letters, vol. 35, no. 21, pp. 1831-1832, 1999. https://doi.org/10.1049/el:19991230.

Abdalhossein Rezai is an Associate professor in University of science and culture, Tehran, Iran. He received Ph.D. degree in electrical engineering from Semnan University, Semnan, Iran in 2013, M.S. and B.S. degree in electrical engineering from Isfahan University of technology, Isfahan, Iran in 1999, and 2001, respectively. His research interests include VLSI design, nanoelectronics, computer arithmetic, cryptography engineering and WBAN.