International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com, Volume 5, Issue 3 (May-June, 2017), PP. 102-106

MODIFIED MONTGOMERY MULTIPLICATION WITH KOGGE-STONE ADDER FOR HIGH THROUGHPUT

[1] SNEHA RAICHEL KOSHY, [2] JUBY RAJU
[1,2] ELECTRONICS AND COMMUNICATION ENGINEERING
[1] MUSALIAR COLLEGE OF ENGINEERING AND TECHNOLOGY
[2] MUSALIAR COLLEGE OF ENGINEERING AND TECHNOLOGY
PATHANAMTHITTA,KERALA, INDIA
[1] sneharaichelkoshy@gmail.com

Abstract — **This project proposes a efficient Montgomery multiplication algorithm such that the 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 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 format conversion can be hidden and high throughput can be obtained. For carry out the Montgomery multiplication more quickly configurable carry save adder is replaced with kogge-stone adder. and thus increases the frequency of operation and decreases propagation delay than previous designs.**

***Index Terms* — Carry-save addition, Montgomery modular multiplier, public-key cryptosystem ,kogge-stone adder.**

I. INTRODUCTION

In many public-key cryptosystems modular multiplication (MM) with large integers is the most critical and time-consuming operation. Therefore, numerous algorithms and hardware implementation have been presented to carry out the MM more quickly, and Montgomery's algorithm is one of the most well-known MM algorithms. 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 mod N. As a result, it can be easily implemented into VLSI circuits to speed up the encryption/decryption process. However, the three-operand addition in the iteration loop of Montgomery's algorithm requires long carry propagation for large operands in binary representation. To solve this problem, several approaches based on carry-save addition were proposed to achieve a significant speedup of Montgomery MM. 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. In the SCS strategy 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. Contrary to the SCS strategy, the FCS strategy 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 multiplier.

Kuang et al. have proposed an energy-efficient FCS based multiplier (denoted as FCS-MMM42 multiplier) in which the superfluous operations of the four-to-two (two-level) CSA architecture are suppressed to reduce the energy dissipation and enhance the throughput. However, the FCS-MMM42 multiplier still suffers from the high area complexity and long critical path delay. Other techniques, such as parallelization, high-radix algorithm, and systolic array design, can be combined with the CSA architecture to further enhance the performance of Montgomery multipliers. However, these techniques probably cause a large increase in hardware complexity and power/energy dissipation which is undesirable for portable systems with constrained resources.This paper

aims at enhancing the performance of CSA-based Montgomery multiplier. The proposed algorithm and hardware architecture have the following several advantages and novel contributions over previous designs. First, the one-level CSA is utilized to perform not only the addition operations in the iteration loop of Montgomery's algorithm but also B + N and the format conversion, leading to a very short critical path and lower hardware cost. However, a lot of extra clock cycles are required. the unnecessary carry-save addition operations while keeping a short critical path delay. Therefore, the required clock cycles for completing one MM operation can be significantly reduced. We also proposes a multiplier using kogge-stone adder. this will increases the frequency of operation and also decreases the propagation delay. As a result, the proposed Montgomery multiplier can obtain higher throughput.

II. EXISTING ARCHITECTURES

A. CSA Based Montgomery Multiplication

Montgomery multiplication algorithm is the most efficient algorithm available. The main advantage of Montgomery algorithm is that it replaces the division operation with shift operations. During two decades many alternative forms of Montgomery algorithms are introduced. These architectures use carry save addition. The work in [10] presented two types of Montgomery algorithms which use Carry Save Adder (CSA). One of the two types used four-to two CSA and the other used five-to-CSA. They had given a brief comparison between these two versions of Montgomery multipliers. They had found that the multiplier using four-to-two CSA architecture has shorter critical path than that of five-to-two CSA multiplier. But extra storage elements and multiplexers are required for 4-to-2 architecture which probably increases the energy consumption. The previous work proposed a Montgomery multiplication algorithm using pipelined carry save addition to shorten the critical path delay of five-to-two CSA. This method also required additional pipeline registers and multiplexers which will increase the area. Ming Der Shieh presented a new algorithm for high speed modular multiplication. This new Montgomery multiplier performs modular reduction in a pipelined fashion, so that the critical path delay is reduced from the four-to-two to three-to-two carry-save addition. Then it requires additional pipeline registers to store intermediate values. All the above works were not discussed about the energy consumption. Several previous works,have developed techniques to reduce the power/energy consumption of Montgomery multipliers. In [6], some latches named glitch blockers are located at the outputs of some circuit modules to reduce the spurious transitions and the expected switching activities of high fan-out signals in the radix-4 scalable Montgomery multiplier. Shiann Rong Kuang tried to reduce the energy consumption of CSAs and registers in the CSA-based Montgomery multipliers via a new technique. They modified the CSA based Montgomery multiplier algorithm and as a result of this, the number of clock cycles required to complete the multiplication is largely decreased. To achieve further energy reduction, they have adjusted the internal structure of barrel register full adder and then applied the gated clock design technique. But this energy efficient algorithm increased the total area of the design

B. SCS-Based Montgomery Multiplication

In the SCS strategy the input and output operands 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 MM.This conversion can be accomplished by an extra Carry propogation adder(CPA).A 32-bit CPA with multiplexers and registers (denoted as CPA_FC), which adds two 32-bit inputs and generates a 32-bit output at every clock cycle, was adopted for the format conversion. Therefore, the 32-bit CPA_FC will take 32 clock cycles to complete the format conversion of a 1024-bit SCS-based Montgomery multiplication. The extra CPA_FC probably enlarges the area and the critical path of the SCS-MM-1 multiplier.

C. 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. 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 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, On the other hand, the FCS-MM-2 multiplier proposed 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. Generally speaking, SCS-based multipliers have lower area complexity than FCS-based Montgomery multipliers.

III. PROPOSED MONTGOMERY MULTIPLIATION

A. Critical Path Delay Reduction

The critical path delay of SCS-based multiplier can be reduced by combining the advantages of FCS-MM-2 and SCS-MM-2. That is pre compute D = B + N and reuse the one-level CSA architecture to perform B+N and the format conversion. Figure.1 shows the modified SCS-based Montgomery multiplication(MSCS-MM) hardware architecture,

respectively. The Zero D circuit is used to detect whether SC is equal to zero, which can be accomplished using one NOR operation. The Q_L circuit decides the qi value. 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 (SS, SC) = SS + SC + 0 until SC = 0.In addition, we also pre compute Ai and qi in iteration i−1 so that they can be used to immediately select the desired input operand from 0, N, B, and D through the multiplexer M3 in iteration. Therefore, the critical path delay of the MSCS-MM multiplier can be reduced into TMUX4 + TFA. However, in addition to performing the three-input carry-save additions k + 2 times, many extra clock cycles are required to perform B + N and the format conversion via the one-level CSA architecture because they must be performed once in every MM. Furthermore, the extra clock cycles for performing B+N and the format conversion through repeatedly executing the carry-save addition (SS, SC) = SS+SC+0 are dependent on the longest carry propagation chain in SS + SC. If SS = 111…1112 and SC = 000…0012, the one-level CSA architecture needs k clock cycles to complete SS + SC.

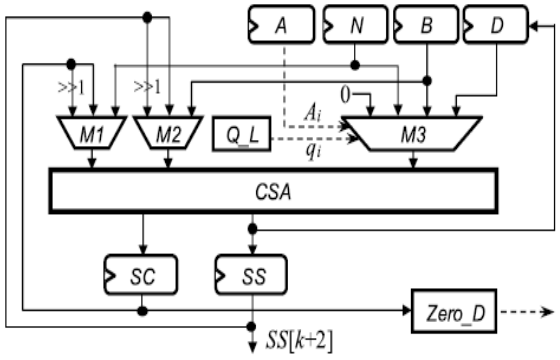

Fig1.MSCS-MM multiplier[1

B. Clock Cycle Number Reduction

To decrease the clock cycle number, a CCSA architecture which can perform one three-input carry-save addition or two serial two-input carry-save additions is proposed to substitute for the one-level CSA architecture. Two cells of the one-level CSA architecture in Figure.2each cell is one conventional FA which can perform the three-input carry-save addition. Two cells of the proposed configurable FA (CFA) circuit. If α = 1, CFA is one FA and can perform one three-input carry-save addition (denoted as 1F_CSA). Otherwise, it is two half-adders (HAs) and can perform two serial two-input carry-save additions (denoted as 2H_CSA). In this case, G1 of CF Aj and G2 of CFAj+1 will act as HA1 j and G3, G4, and G5 of CF Aj will behave as HA2j. Moreover, we modify the 4-to-1 multiplexer M3 into a simplified multiplier SM3 because one of its inputs is zero, where the INVERT operation. Note that M3 has been replaced by SM3 in the proposed one-level CCSA architecture.

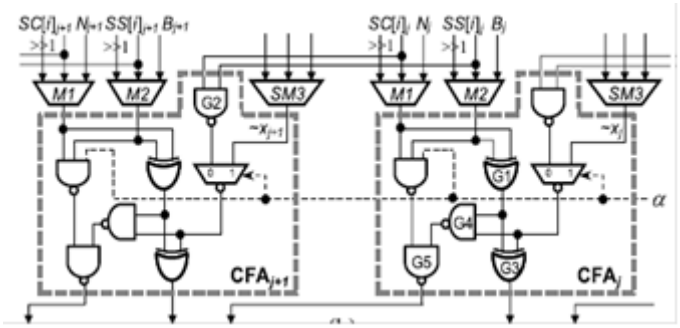

Fig2.Proposed CFA circuit [1]

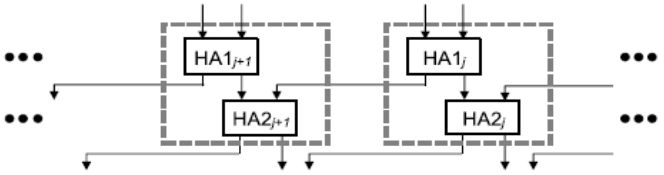

Fig3.Two serial HAs.[1]

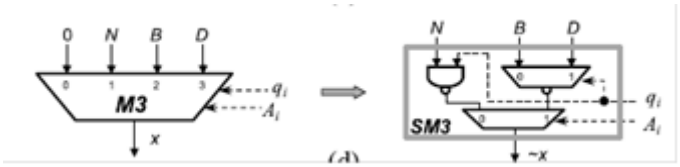

Fig4.Simplified multiplexer SM3.[1]

C. Proposed Hardware Architecture

On the bases of critical path delay reduction, clock cycle number reduction, above, a new SCS-based Montgomery MM algorithm (i.e., SCS-MM-New algorithm) using one-level CCSA architecture is proposed to significantly reduce the required clock cycles for completing one MM.

The hardware architecture of SCS-MM-New algorithm, denoted as SCS-MM-New multiplier, are shown in Fig. ,which consists of one one-level CCSA architecture, two 4-to-1 multiplexers (i.e., M1 and M2), one simplified multiplier SM3, one skip detector Skip_D, one zero detecto Zero_D, and six registers.Both M4 and M5 in Fig. 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. The Skip_D is composed of four XOR gates, three AND gates, one NOR gate, and two 2-to-1 multiplexers The modulus N and inputs will be allowed inside the two multiplexers. This partial product is then allowed inside the multiplier. Those partial outputs then enter into configurable carry save adder, where the carry save addition operation is performed. They are stored in the flip flops temporarily. When another partial output is executed, then that will be stored in the flip flop. The Skip detector will skip the previous multiplication which is not required in the operation so as to reduce the number of clock cycles. The partial product from SM3 is allowed to the multiplexers M4 and M5. Later on it allows inside the flip flops for temporary storage, then to the skip detector. The output can

be obtained from semi carry. This process is repeated until the output is obtained.

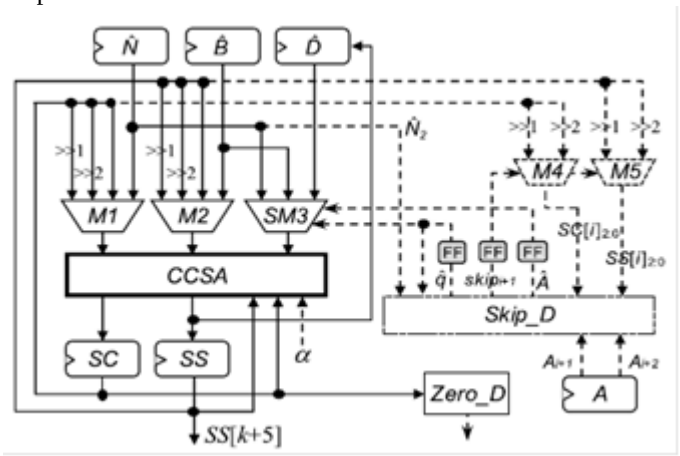

Fig 5. SCS-MM-New multiplier[1]

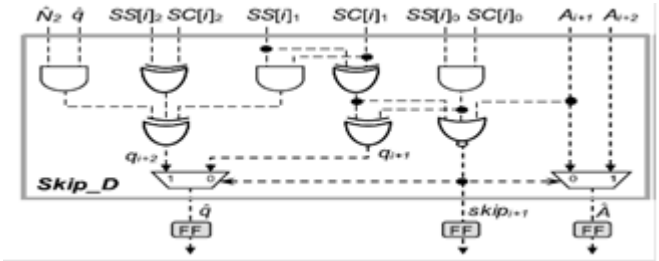

Fig6.Skip detector Skip_D[1]

The Skip_D is composed of four XOR gates, three AND gates, one NOR gate, and two 2-to-1 multiplexers. It first generates the qi+1, qi+2, and skipi+1 signal in the i th iteration respectively, and then selects the correct qˆ and Aˆ according to skipi+1. At the end of the i th iteration, qˆ, Aˆ, and skipi+1 must be stored to FFs. In the next clock cycle of the i th iteration, SM3 outputs a proper x according to qˆ and Aˆ generated in the i th iteration.

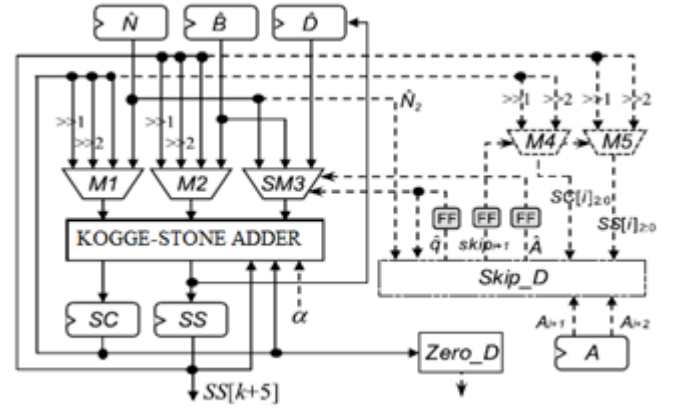

Fig7.Proposed Architecture using kogge-stone adder

A. When replacing the configurable carry save adder with a kogge-stone adder the frequency of operation increases and also decreases the propagation delay. so the montgomery multiplication can be carry out more quickly

IV. IMPLEMENTATION RESULTS

Table 1.comparisons of different montgomery multipliers

Multiplier	#Cycle	ΔC (%)	Delay (*ns*)	Area (μm^2)	Time (μs)	Throughput Rate (*Mbps*)	ATP ($10^3 \mu m^2 \times \mu s$)
SCS-MM-1(32) [5]	1072	+4.2	4.93	487171	5.2850	193.8	2574.68
SCS-MM-2 [8]	1049	+1.9	5.60	406127	5.8744	174.3	2385.75
FCS-MM-1 [9]	1029	-	5.80	518214	5.9682	171.6	3092.80
FCS-MM-2 [9]	1029	-	6.01	677121	6.1843	165.6	4187.51
FCS-MMM42 [10]	822	−20.1	5.56	749076	4.5703	224.1	3423.52
SCS-MM-New	880	−14.5	4.00	498379	3.5200	290.9	1754.29

To evaluate the average clock cycles for completing one Montgomery MM, the above-mentioned multipliers, including SCS-MM-1 , SCS-MM-2 ,FCS-MM-1, FCS-MM-2, FCS-MMM42 and the proposed SCS-MM-New were designed and specified in Verilog hardware description language.

As the results shown in Table 1, the proposed SCS-MM-New multiplier has the shortest critical path delay and needs fewer clock cycles to complete one Montgomery MM,and thus spends the least execution time and achieves he highest throughput rate and needs fewer clock cycles to complete one Montgomery MM, and thus spends the least execution time and achieves the highest throughput rate. On the other hand, the SCS-MM-2 multiplier generally has smaller area than other designs. The proposed SCS-MM-New multiplier also needs more area than the SCS-MM-2 multiplier due to extra multiplexers introduced to shorten the critical path delay and reduce the required clock cycles. Nevertheless, the area of the proposed SCS-MM-New multiplier is still less than that of FCS-based multipliers. As a consequence, SCS-MM-New can obtain the smallest ATP than previous radix-2 Montgomery multipliers.

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. In this paper also proposed a multiplier using kogge-stone adder and it increases the frequency of operation and decreases the propogation delay. Thus the Montgomery multiplication can carry out more quickly. Experimental results showed that the proposed

approaches are indeed capable of enhancing the performance of Montgomery multiplier

REFERENCES

[1] Shiann-Rong Kuang, Kun-Yi Wu, and Ren-Yao Lu"Low-Cost High-Performance VLSI Architecture for Montgomery Modular Multiplication" IEEE transactions on very large scale A. F. Tenca integration (vlsi) systems, 1063-8210,February 15,2015

[2] 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., pp. 643–646,Sep.2007

[3] 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 multipliers,"IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 7,pp. 1136–1146,July 2011

[4] 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.

[5] .J. C. Neto, , 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,Sep.2011

[6] 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.

[7] 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.

[8] 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.

[9] JS.-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. 12..