iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** IV **Month of publication:** April 2017

DOI: http://doi.org/10.22214/ijraset.2017.4061

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Systolic Hardware Architecture of Montgomery Modular Multiplication for Public Key Cryptosystems

Vinoth Kumar. D [1], Senthilkumaran. V[2]
[1] M. E VLSI Design, [2]Assistant Professor, ECE, Mahendra Engineering College, Namakkal, India

Abstract: ***The Montgomery modular multiplication is mostly used in the field public-key cryptosystems. This work presents how to relax the data dependency in conventional word-based algorithms to increase the possibility of reusing the current words of variables. With the greatly relaxed data dependency, i proposed a novel scheduling scheme to alleviate the number of memory access in the developed scalable micro architecture. Analytical results show that the memory bandwidth requirement of the proposed scalable architecture is almost 1=ðw _ 1Þ times that of conventional scalable architectures. The proposed one also retains a latency of exactly 1 cycle between the operations of the same words in 2 consecutive iterations of the Montgomery modular multiplication algorithm when employing enough processing elements. To Compared to the design with previous work, experimental results shows that the proposed one achieves an 55 percent reduction in power consumption with no degradation in throughput. The number of reduced memory access not only leads to lower power consumption, it also facilitates the design of scalable architectures for any precision of operands.***

Index Terms: Cryptosystems, low-power design, Montgomery modular multiplication, scalable architecture, VLSI

I. INTRODUCTION

The Montgomery multiplication [1] is complex than other cryptographic algorithms such as RSA, Digital Signature Algorithm (DSA), Elliptic Curve DSA and other emerging cryptographic algorithms, such as pairing-based systems. Despite improvements in the clock frequency and the level of parallelism of conventional microprocessors, software-based implementations of Montgomery multiplication remain insufficient, both in terms of performance and energy efficiency. These computations have long carry chains and their implementations are fundamentally different from systems that use composite fields with small characteristic (e.g. GF(2167)). Existing FPGA Montgomery multiplier implementations for large integer systems have one major weakness: they all use algorithms with O(*N*2) complexity i.e. the area and/or runtime increases quadratically with the bit-width of the computation. In this paper, i improve the implementation of new Montgomery multiplication of lengthy integers at the algorithmic level in order to lower complexity and high throughput. We have developed a parameterized Karatsuba multiplier using a combination of multiple-precision and coarse-grained carry-save addition techniques. This method has complexity *O*(*N*(log 3= log 2)). The major contributions of this work are: _ An FPGA-optimized design for irregular, recursive Karatsuba multiplication that is parameterizable to different bit-widths and a batch-pipelined Montgomery multiplier based on the Karastuba multiplier. I compare the performance and energy efficiency of the proposed Montgomery multiplier with existing software and hardware implementations.

II. BACKGROUND

In Many public-key cryptosystems [1]–[3], modular multiplication (MM) with large integers is the most critical and time-consuming operation. There are many 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 [4] 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 $R = 2^k \bmod N$. 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. In the SCS strategy [5]–[8], 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) [5] or reusing the carry-save adder (CSA) architecture [8] iteratively. Contrary to the SCS strategy,

©IJRASET: All Rights are Reserved

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

the FCS strategy [9], 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.

III. PREVIOUSLY PROPOSED ARCHITUCTURE

A. *Montgomery Modular Multiplier*

In the proposed new SCS-based Montgomery MM algorithm is to reduce the critical path delay of Montgomery multiplier. Then addition, the disadvantage of more clock cycles for completing one multiplication is improved while maintaining the advantages of less critical path delay and low hardware complexity [2].

Critical Path Delay Reduction The critical path delay of SCS-based multiplier is reduced by combining the advantages of FCS-MM-2 and SCS-MM-2. In this pre compute $D = B + N$ and reuse the one-level CSA architecture to perform $B+N$ and the format conversion. Fig.1 shows the modified SCS-based Montgomery multiplication (MSCS-MM) algorithm and possible hardware architecture .

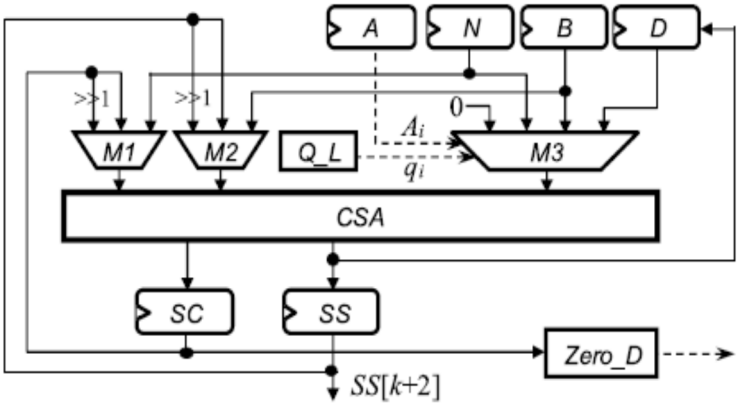


Fig.1. Diagram of Montgomery Modular Multiplier

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, I also pre compute *Ai* and *qi* in iteration $i-1$ (this will be explained more clearly in Section III-C) 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 *I [5]* Therefore, the critical path delay of the MSCS-MM multiplier can be reduced into *T*MUX4 + *T*FA, 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 $SS + SC$. That is, $3k$ clock cycles in the worst case are required for completing one MM. Thus, it is critical to reduce the required clock cycles of the MSCS-MM multiplier [1].

B. *Algorithm*

Input: T: w k-1 ≤ T < wk, where k is the number of bits.
R, S: 0 ≤ R, S < T
Output: O = R * S mod T
Algorithm:
O = 0;
for j = k − 1 downto 0 do
O = w * O + sj * R;
Uc = [O / T];
O = O – Uc * T;

©IJRASET: All Rights are Reserved

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

End for

IV. PROPOSED MONTGOMERY MODULAR MULTIPLICATION ALGORITHM

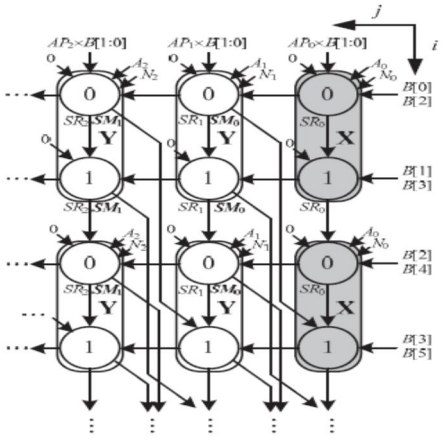


Fig. 2. Dependency graph of Algorithm Modified_MM_R for w ¼ 3.

Fig. 2 illustrates the dependency graph of Algorithm Modified_MM_R for w ¼ 3, where the jth word-based operation of the ith iteration is allocated in j th column and the I th row. For clarity, the symbols SM0, SR0, AP0 are replaced by SM, SR, and AP, respectively. By taking advantage of that SM generated in the ith iteration is accumulated in the ði þ wÞth iteration, a task can perform all of the operations in w 1 iterations instead of only one iteration. For example, since w ¼ 3 in Fig. 4, one PE can execute all of the operations in two iterations in an interleaved manner. The two circles indexed with 0 and 1 in the same group, say X, denote that the two X tasks for the same words in two consecutive iterations can be accomplished before moving to handle the Y tasks for the next words of variables.

V. RESULTS AND DISCUSSION

The following figures titled Figure 1, Figure 2, Figure 3, Figure 4 and Figure 5 implies the simulation results of the Classical, Montgomery, Bipartite, Tripartite and for the Proposed algorithm. The result analysis had been done for all possible values of four bit wide. The same algorithms were also extended for higher level of bits. The output is given in the terminal t. All the above algorithms are coded in VLSI Hardware Description Language. The HDL codes are synthesized and simulated in Xilinx 9.1i version of ISE simulator. The simulation results are compared with the manually calculated value and the logic of HDL code is verified.

The comparison table for all the above algorithms are based on the hardware utilized and time delays consumed for synthesis and to implement all the above stated algorithms. The hardware computation can be summarized by considering the amount of slices, LUTs used, adders/ subtractions and total number of comparators needed. The time make use of can be stated from the CPU time usage and time delay. The comparison table is stated in Table 1. Thus the above results show that the proposed algorithm produces significant reduction in time delay and in hardware computation.

VI. CONCLUSION AND FUTURE WORK

These results clearly depicts that the proposed algorithm produces a significant advantage in hardware and time consumption in executing and implementing the algorithm. This drastic reduction in hardware and time delay is due to complete elimination of the classical algorithm by normal interleaved multiplication method. On practicing this scenario, the bulk program which is needed to execute the heavy division algorithm is completely avoided. The replacement of the division process by repeated subtractions is the reason behind the trim down of the hardware. This makes a greater advantage while implementing the cryptosystems while considering heavy input operands of larger size.

The current work is extended to make any modifications further to produce a significant reduction in hardware utilization and time delay. This may be achieved by reducing the steps in algorithm which skip a bulk operation in this case a division operation, say. The future work focuses on the modular multiplication to produce a speed in the calculation of final value.

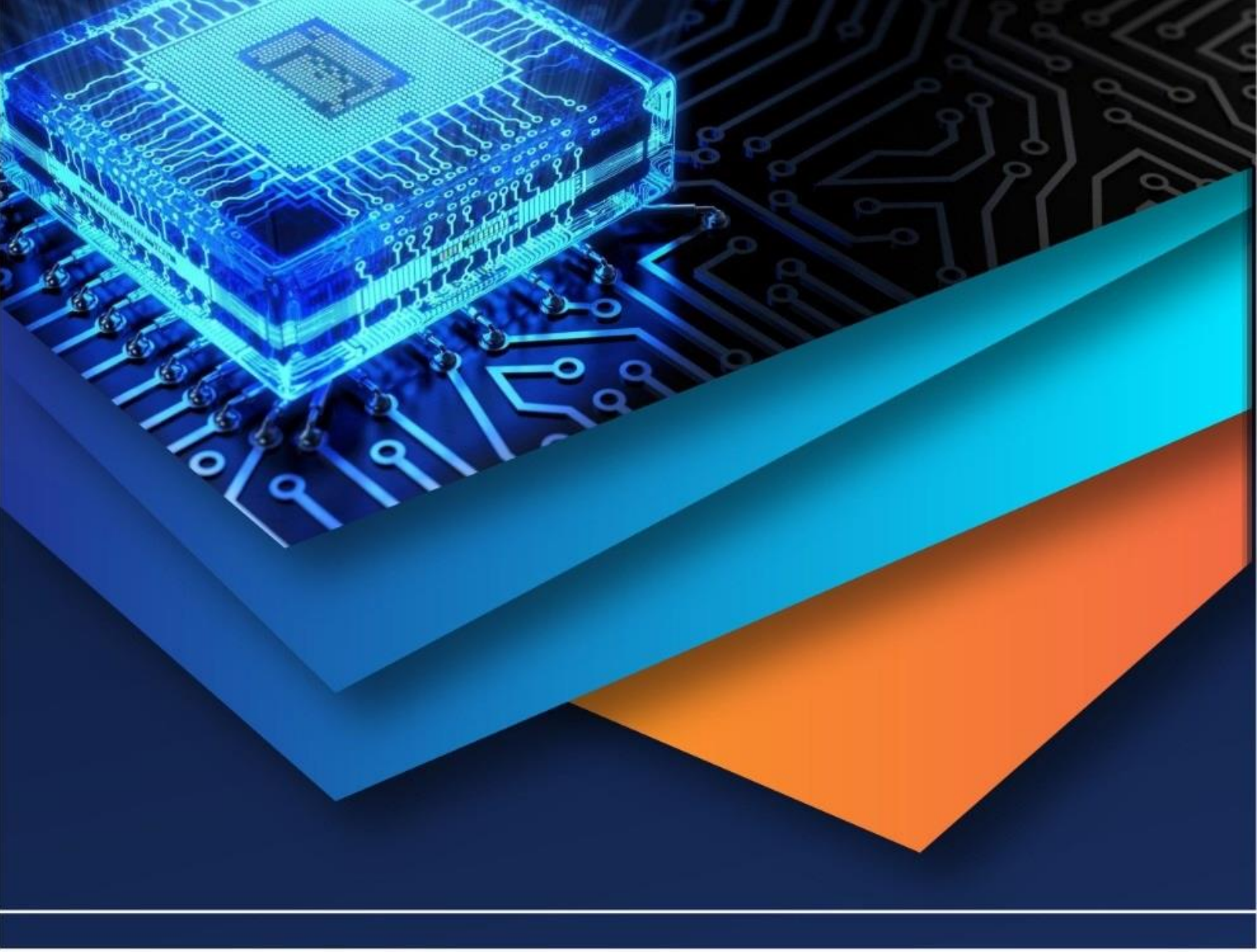
©IJRASET: All Rights are Reserved

Volume 5 Issue IV, April 2017
ISSN: 2321-9653

REFERENCES

[1] Amber.P, Pinckney.N, and Harris, D. M. "Parallel high-radix Montgomery multipliers,"(2008) in Proc. 42nd Asilomar Conf. Signals, Syst., Comput., pp. 772–776.
[2] Bunimov.V, Schimmler.M, and Tolg.B, "A complexity-effective version of Montgomery's algorihm," (2002) in Proc. Workshop Complex.Effective Designs.
[3] Gang.F, "Design of modular multiplier based on improved Montgomery algorithm and systolic array," (2006) in Proc. 1st Int. Multi-Symp. Comput. Co mput. Sci., vol. 2. Jun. 2006, pp. 356–359.
[4] Han, J. Wang S., Huang W., Yu Z., and Zeng X, "Parallelization of radix-2 Montgomery multiplication on multicore platform,"(2013) IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 12, pp. 2325–2330,.
[5] Kuang S.-R., Wang J.-P., Chan K.-C., and Hsu. H.-W., "Energy-efficient high-throughput Montgomery modular multipliers for RSA cryptosystems," (2013) IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 11,pp. 1999–2009,.
[6] McIvor.C, McLoone.M, and McCanny, J. V. "Modified Montgomery modular multiplication and RSA exponentiation techniques,"(2004) IEE Proc.-Comput. Digit. Techn., vol. 151, no. 6, pp. 402–408,.
[7] Miyamoto A., Homma N., Aoki, T. and Satoh.A, "Systematic design of RSA processors based on high-radix Montgomery multipliers,"(2011) IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 7, pp. 1136–1146.
[8] Neto, J. C. Tenca A. F., and Ruggiero W. V., "A parallel k-partition method to perform Montgomery multiplication,"(2011) in Proc. IEEE Int. Conf. Appl.-Specific Syst., Archit., Processors, , pp. 251–254.
[9] Sassaw.G,. Jimenez.C.J, and Valencia.M, "High radix implementation of Montgomery multipliers with CSA," (2010) in Proc. Int. Conf. Micro electron., Dec. 2010, pp. 315–318.
[10] Saemen.J and Rijmen.V, The block cipher Rijndael, Smart Card research and Applications, (2010)LNCS 1820, Springer-Verlag, pp. 288-296

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)