IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

LOW-COST HIGH-PERFORMANCE VLSI ARCHITECTURE USING MONTGOMERY MODULAR MULTIPLICATION

[1] P Ravi Kumar Reddy, [2]G Sai Shruthi, [3] Shivani Jupudi [4]R Nithin Kumar

[1]Assistant Professor, [2] Student, [3] Student, [4] Student

[1] Electronics and Communication Engineering,

[1]Matrusri Engineering College, Hyderabad, Telangana, India.

Abstract: Montgomery Modular Multiplication is a method for performing fast modular multiplication. It is used for encryption process in Public Key Cryptography. The Project consists of Semi Carry Save (SCS) based Montgomery Modular Multiplication, with high speed performance. One Carry Save Adder(CSA) is used to avoid carry propagation at each addition operation. The CSA is reused for operand pre-computation and format conversion leading to a short critical path delay. Montgomery modular Multiplier is implemented using VHDL language in Xilinx software. The multiplier is then compared with one of the existing Montgomery multiplier in terms of area, delay and power. Modular Multiplication (MM) with large integers is a time consuming operation in many public-key cryptosystems. Therefore, many algorithms have been presented to carry out MM more quickly and Montgomery's algorithm is one of them. To Overcome The Drawbacks The CCSA Is Replaced With PASTA (Parallel Self Timed Adder) In The Montgomery Modular Multiplier. The PASTA Adder Can Achieve less power consumption.

Index Terms **- CSA, MM, Semi carry save, Public key cryptography, PASTA.**

I. INTRODUCTION

In many public-key cryptosystems [1]–[3], 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 [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 $R = 2^k \bmod N$. As a result, it can be easily implemented into VLSI circuits to speed up the encryption/decryption process.

Kaung et al. [10] 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.They 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 [11]–[19], 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 [20], [21], which is undesirable for portable systems with constrained resources.

Adders being core building blocks in several VLSI circuits like microprocessors, ALU's etc. performance of adder circuit extremely affects the capability of the system. during this paper we tend to present the design and performance of Parallel Self-Timed Adder. it's supported a algorithmic formulation for acting multibit binary addition. The operation is parallel for those bits that don't would like any carry chain propagation. A sensible implementation is provided in conjunction with a completion detection unit. The implementation is regular and doesn't have any sensible limitations of high fanouts. The planned work principally geared toward minimizing the amount of transistors and estimation of varied parameters viz., area, power, delay for pasta. we've got conjointly designed four bit pasta as an example of planned approach.

In this space, the engineering and hypothesis behind pasta is displayed. The viper 1st acknowledges 2 data operands to perform half increases for each piece. on these lines, it emphasizes utilizing previous created convey and totals to perform half increases quite once till all convey bits are devoured and settled at zero level.

II. EXISTING SYSTEM

Consider the modulus N to be a k-bit odd number and an extra factor R is to be defined as $2^k \bmod N$, where $2^{k-1} \leq N < 2^k$. Given two integers a and b, where a, b<N, the N–residue of a and b with respect to R can be defined as,

$$A = a \times R \pmod N \quad \ldots\text{Eq. (1)}$$

$$B = b \times R \pmod N \quad \ldots\text{Eq.(2)}$$

Based on equation (1), the Montgomery modular product Z of A and B can be obtained as

$$Z = A \times B \times R^{-1} \pmod N \quad \ldots\text{Eq.(3)}$$

In this existing system, carry save addition with semi-carry approach is described. In which all the multiplicands are not recycled, that is whatever the multiplicand is needed to be multiplied at that time alone is used for determining the output. The carry save approach has higher benefits since it is the basic key for operating a Montgomery modular multiplier. In such a way, using this semi carry save type only one carry level adder is implemented which may be two serial half adders or a full adder can be used based on the requirement.

It thereby reduces the number of clock cycles and hence less delay. So the output will be optimized and it can be implemented using Verilog coding.

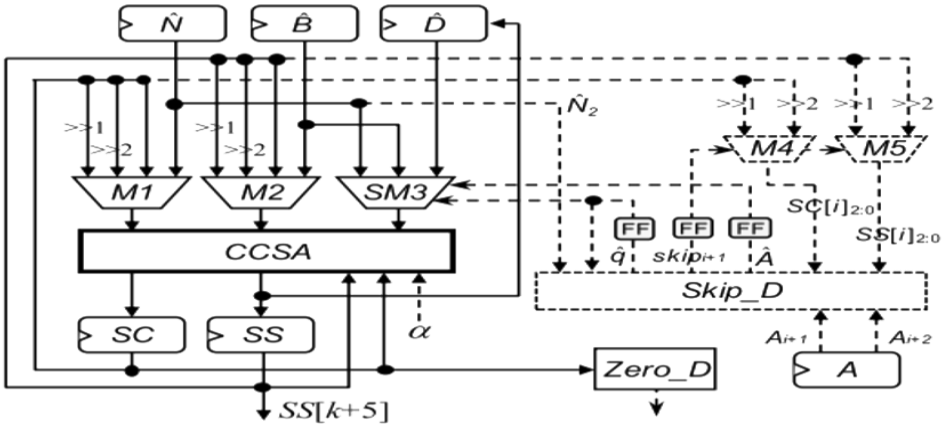

Fig.1. Block diagram of Montgomery Modular Multiplication using CCSA

The above architecture is the semi-carry save based Montgomery multiplier. In which the loop is reduced on comparing to the existing one. It consists of two multiplexers, one multiplier, one configurable carry save adder, flip-flops, skip detector and zero detector.

A. Architecture of Montgomery Modular Multiplication using CCSA

Fig. 1 illustrates the block diagram of proposed semi carry save multiplier. It is first used to precompute the four-to-two carry save additions. Then the required multiplication can be performed. 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. The zero detectors can also be used to detect zero in many situations, which is most required. The complexity is very less compared to the previous one.

B. Critical Path Delay Reduction

In order to reduce the critical path delay, the operations in semi carry save and full carry save is performed jointly. The carry save format conversion as well as the binary format should be taken place.

Then pre-computation must be done in order to reuse the multiplicand values. Another method is using zero detectors. SC will produce the output only when the zero is detected. Then the pre computation can be done i-1 iteration.

C. Clock Cycle Number Reduction

In order to decrease the clock cycle number, a configurable carry save architecture to perform one three-input carry-save addition or two serial two-input carry-save addition is used. Furthermore the number of iteration can also be reduced to reduce number of clock cycles.

Then a signal skip is used. In order to verify whether i+1 is required or not to be happen in the upcoming events. This can be found in the previous i iteration itself. By again using the same condition, signal skip will use i+1, so that it increases by a factor 2. Hence it directly goes to i+2. So that clock cycle gets reduced.

D. Quotient Pre-computation

Ai+1, Ai+2 and qi+1, qi+2 should be known already in order that the unwanted steps in the (i +1) iteration can be reduced by determining i iteration. So as to pre compute the quotients. Another method is using skip detector so that it will pre computes the values. And also since the shortest path in this multiplier is lengthened, it has to be minimized. As modulus N is an odd number, it can be used directly for the multiplication. So that time is consumed highly.

Comparision Of Various Stages Of Multipliers

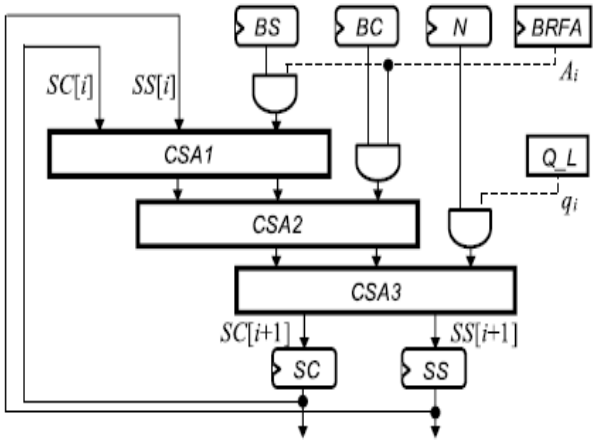

Fig 2.1 Three Stage CSA

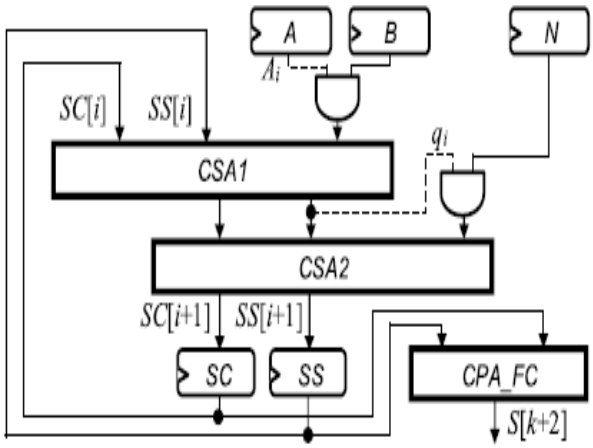

Fig.2.2 Two Stage CSA

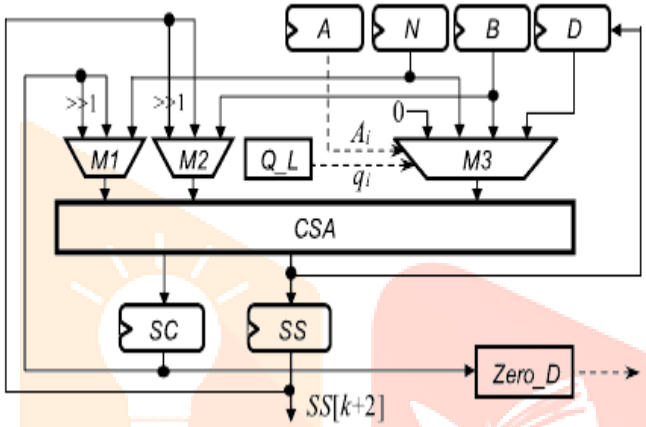

Fig. 2.3 Single Stage CSA

In the existing system, the semi carry save adder based Montgomery multiplier design has more hardware complexities and large critical path due to the usage of multiple carry save adder stages. So the proposed system will overcome this disadvantage by reducing the carry save adder stages into only one level CSA architecture. And this one level CSA is further modified into configurable CSA. This is done by connecting a skip detector circuit to the CSA circuit so as to skip unnecessary carry save addition operations. The drawback of more clock cycles for completing one modular multiplication can also be improved. As a result by using only one level carry save adder architecture, we can expect better results than previous methods which used several carry save adder stages.

III. PROPOSED SYSTEM

To implement a better and efficient method of performing modular multiplication which works with high performance in low cost. Design of Parallel Self Timed Adder is the primary objective. The interconnection and space demand is linear that makes it possible to fabricate. The planning operates during a parallel manner for those bits that don't need any carry propagation. It is self - regular, which implies that as presently the addition is finished, it'll signal the completion of addition thereby overcoming the duration limitations. Self-coordinated adders will presumably run speeder than usual adders.

PASTA is to be implemented using CMOS technology. The general block diagram of Parallel Self-timed Adder includes following circuits modules:

1. **Half-Adder**
2. **Multiplexer**
3. **Completion detection circuit.**

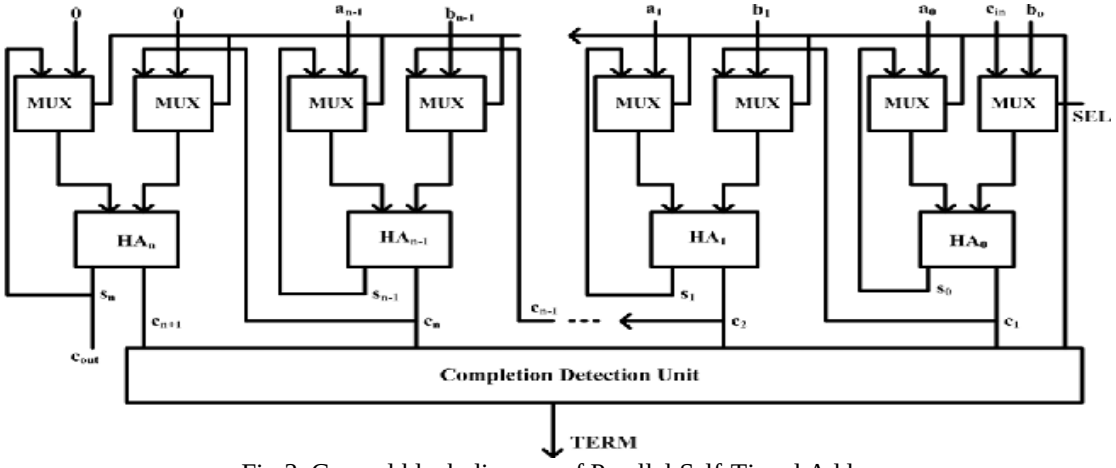

Fig.3. General block diagram of Parallel Self-Timed Adder

The selection input for two-input multiplexers corresponds to the Request handshake signal and will be a single 0 to 1 transition denoted by SEL. It will initially select the actual operands during SEL=0and will switch to feedback/carry paths for subsequent iterations using SEL=1. The feedback path from the HAs enables the multiple iterations to continue until the completion when all carry signals will assume zero values are show in Fig .3.

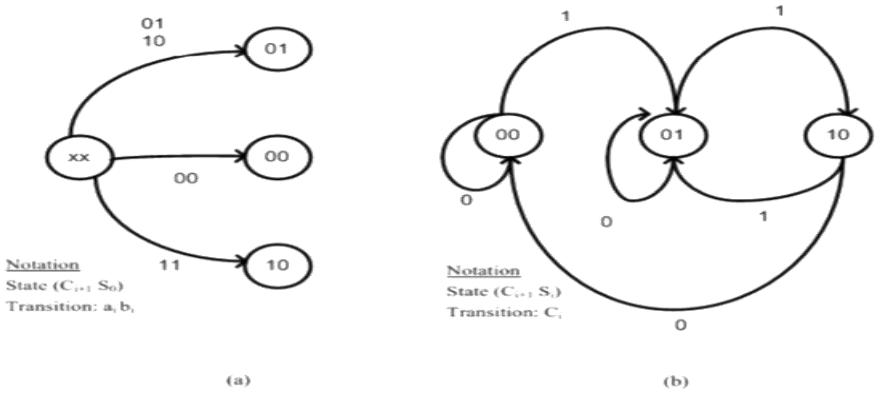

Fig.4. State Diagram of PASTA Adder

In Fig. 4, two state diagrams are drawn for the initial phase and the iterative phase of the proposed architecture. Each state is represented by (Ci+1 Si) pair where Ci+1, Si represent carry out and sum values, respectively, from the i^{th} bit adder block. During the initial phase, the circuit merely works as a combinational HA operating in fundamental mode. It is apparent that due to the use of HAs instead of FAs, state (11) cannot appear.

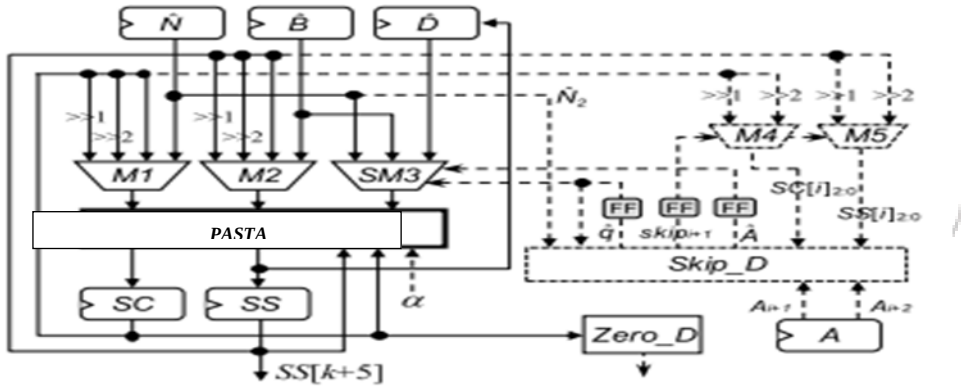

Fig .5 Montgomery Modular Multiplication using PASTA adder

The proposed architecture of Montgomery Modular Multiplication using PASTA adder, which consists of one one-level Parallel Self Timed Adder(PASTA) architecture, two 4-to-1 multiplexers (M1 and M2) one simplified multiplier SM3, one skip detector Skip_D, one zero detector Zero_D, and six registers. Zero detector Zero_D is used to detect SC is equal to zero. The Skip_D is composed of four XOR gates, three AND gates, one NOR gate, and two 2-to-1 multiplexers the skip detector is used to detect the unnecessary multiplication operations.

IV. RESULTS

Simulation Results -The simulated results of enforced style for standard approach and projected approach of Parallel Self-Timed Adder are shown below. All the circuits are designed, simulated and also the performance is evaluated supported starting from block diagram [Fig.6.1] to RTL Schematic [Fig.6.2], Clock Cycles[Fig.6.3].

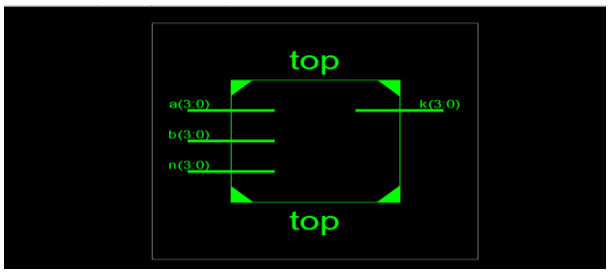

Fig.6.1 Block Diagram of Montgomery PASTA Adder

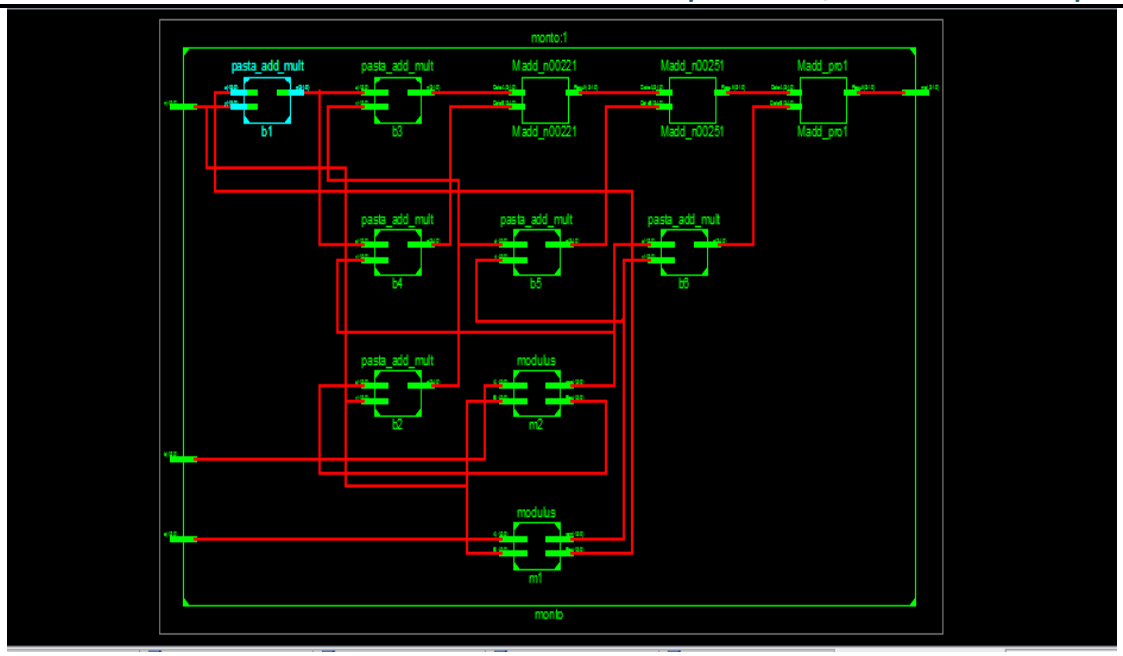

Fig.6.2 RTL Schematic

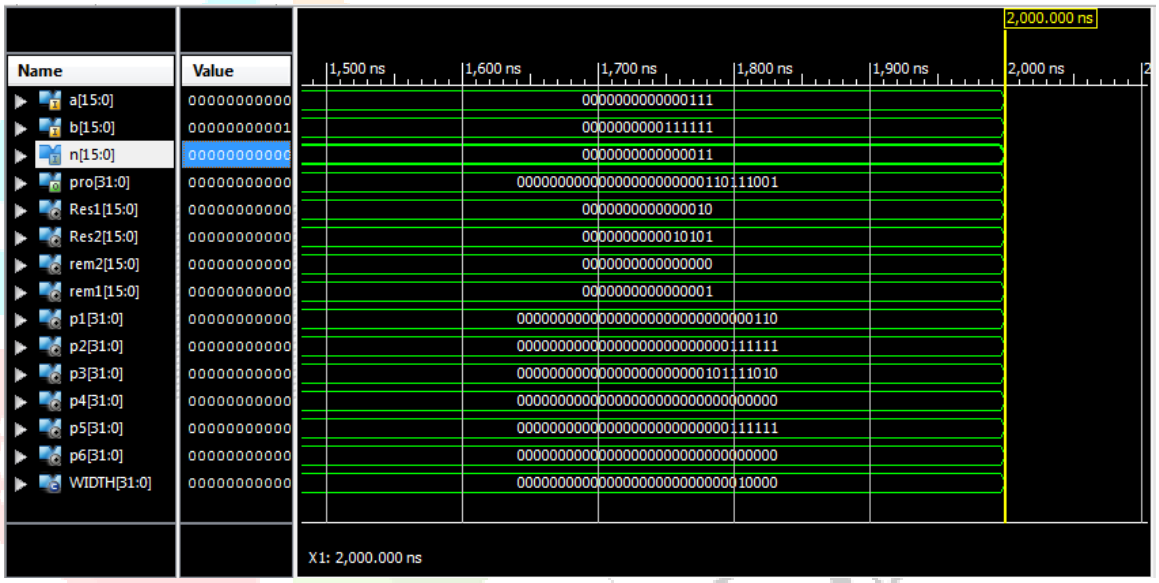

Fig.6.3 Waveform

Table 1. Comparison of Results of CCSA and PASTA with various parameters

Parameter	CCSA(Existing)	PASTA(Proposed)
No. of Slices	3147	23
Power Consumed	136.43 mW	82.16 mW
Delay	1150.550 ns	40.97 ns
No. of Bonded IOs	131	80

V. CONCLUSION & FUTURE SCOPE

SCS-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 smaller area than FCS-based multiplier. In the existed architecture disadvantages are carry propagation delay and extra clock cycles. To overcome the disadvantages, we go for PASTA adder. The PASTA adder is using in

Montgomery Modular Multiplier in these advantages are low hard ware cost short critical path delay and required clock cycles are reduced for completing one MM operation. In this project the used MUX takes inputs in parallel manner but not the carry is generated parallel manner. In future this project can be modified in such a way that the carry generated may also be taken in parallel manner. So, the delay can be reduced further more.

VI.REFERENCES

[1]. Adi, W.: Fuzzy Modular Arithmetic for Cryptographic Schemes with Applications to Mobile Security, Proc. IEEE International Conference EuroComm 2000, pp. 263-265, IEEE 2000.

[2]. Bajard, J., Didier, L., Kornerup, P.: An RNS Montgomery modular multiplication algorithm, IEEE Transactions on Computers, Vol. 47, pp. 766-776, July 1998.

[3]. Bertil Schmidt, Manfred Schimmler and Wael Adi: Area Efficient Modular Arithmetic for Mobile Security, ICWN'02, Las Vegas, USA (2002).

[4]. Blakley, G. R.: A Computer Algorithm for Calculating the Product A*B modulo M, IEEE on Computers, Vol. c-32, No. 5, May 1983, pp. 497-500.

[5]. Blum, T, Paar, C.: High Radix Montgomery Modular Exponentiation on Reconfigurable Hardware, IEEE Transactions on Computers, July 2001 (Vol. 50, No. 7)

[6]. Blum, T., Paar, C.: Montgomery Modular Multiplication on Reconfigurable Hardware, 14th IEEE Symposium on Computer Arithmetic (ARITH-14), April 14-16, IEEE 1999.

[7]. Brickell, E.F.: A Fast Modular Multiplication Algorithm with Application to Two Key Cryptography, Proc. of Crypto'82, pp. 51-60, Plenum Press 1983.

[8]. Eldridge, S.E., Walter, C.D.: Hardware implementation of Montgomery's modular multiplication algorithm, IEEE Transaction on Computers, Vol. 42, pp. 693-699, July 1993.

[9]. K.Lakshmi Durga , V.Priya Darshini VLSI Architecture For Montgomery Modular Multiplication Algorithm By Using Pasta Adder , , Gudlavalleru Engineering College, Gudlavalleru, A.P, India.

[10]. Crypto processor PLD001, Master Thesis, Department of Computer science, Tallinn Technical University, June 1998, www.pld.ttu.ee/~prj/master.pdf

[11]. Kim, Y. S., Kang, W. S., Choi, J. R.: Implementation of 1024-bit modular processor for RSA cryptosystem, School of Electronic and Electrical Engineering, Kyungpook National University, 1370 Sankyok-Dong, Book-Gu, Taegu, Korea, 702-701, www.ap-asic.org/2000/proceedings/10-4.pdf

[12]. Knuth, D. E. the Art of Computer Programming, Vol. 2, Seminumerical Algorithms. Reading, MA: Addison-Wesley, Nov. 1971, 2nd printong, p. 423.

[13]. Koc, C. K.: RSA Hardware Implementation, RSA Laboratories, RSA Data Security, Inc. August 1995, http://security.ece.orst.edu/koc/papers/reports.html

[14]. Koc, C.K., Acar, T., Kaliski, B.S.: Analyzing and Comparing Montgomery Multiplication Algorithms, IEEE Micro, pp. 26-33, June 1996.

[15]. Menezes et. al: Handbook of Applied Cryptography, CRC Press, 1997.

[16]. Montgomery, P. L.: Modular Multiplication without trial division,Mathematics of Computation, 44, pp. 519-521, 1985.

[17]. Morita, H.: A Fast Modular-multiplication Algorithm based on a Higher Radix, NTT Communications and Information Processing Laboratories 3-9-11, Midori-cho.

[18]. Shand, M., Vuillemin, J.: Fast implementation of RSA cryptography, Proc. 11th IEEE Symposium on Computer Arithmetic, pp. 252-259, IEEE 1993.

[19]. Sloan, K. R.: Comments on "A Computer Algorithm for Calculating the Product A*B modulo M", IEEE Transactions on Computers, Vol. c-34, No. 3, March 1985.

[20]. Stinson D. R.: Cryptography, Theory and Practice, CRC Press, 1995.

[21]. Walter, Colin D.: Precise Bounds for Montgomery Modular Multiplication and Some Potentially Insecure RSA Moduli, CT-RSA 2002: San Jose, CA, USAhttp://link.springer.de/link/service/series/0558/bibs/2271/22710030.htm

[22]. Walter, Kolin D.: Logarithmic Speed Modular Multiplication. Electronics Letters 30 (1994), No 17. pp. 1397-8.

[23]. Wu, C., Chou, Y.: General Modular Multiplication by Block Multiplication and Table Lookup, IEEE Int. Symposium on Circuits and System, ISCAS'94, pp. 295-298, IEEE 1994.

[24]. Yong-Yin, J., Burleson, W.: VLSI array algorithms and architectures for RSA modular multiplication, IEEE Transactions on VLSI Systems, Vol. 5, pp. 211-217, June 1997.