ISSN 2278-3091
Volume 9 No.2, March -April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse21922020.pdf
https://doi.org/10.30534/ijatcse/2020/21922020

Design and Implementation of Efficient Cryptographic Arithmetic based on Reversible logic and Vedic Mathematics

Kiran Kumar V G[1], Shantharama Rai C[2]
[1]Assistant Professor, Department of E & C Engineering, A J Institute of Engineering and Technology, Kottara Mangaluru, (Karnataka), INDIA,kiranvgk@gmail.com
[2]Professor and Principal, A J Institute of Engineering and Technology, Kottara Mangaluru, (Karnataka), INDIA, csraicec@gmail.com

ABSTRACT

Low area, low power and efficient arithmetic operations are the need of the decade. Arithmetic operations like modular operations have wide applications like cryptography where security is also a factor. The portability of a device, power consumed, response time of the system, power dissipation are some of the important aspects that need to be considered while designing a system with complex operations. The best and the most suitable method would be to employ efficient arithmetic algorithms that are the building blocks of complex operations like signal and image processing and DSP. The arithmetic operations include the addition, multiplication, and the modular operations. Vedic methodology and the reversible gates are infused together to work as multipliers and are analyzed. Montgomery modular operation is modified and its efficiency is checked with the different multipliers and the modular reduction algorithms implemented here. The area, timing and power consumed by the algorithms is tabulated and studied. The LUT's, slice registers and IOB's used by the design is tabulated. The tabulated values help the designer to choose an efficient algorithm based on the resources that are available while designing. An algorithm can thus be application specific. All the algorithms are implemented in Xilinx 14.2 with Spartan 6 as the family and in Cadence using 45nm technology. The hardware description language used is Verilog.

Key words: Reversible logic, Vedic Methodology, Barrett Reduction, Mod without division algorithm, Montgomery modular operation.

1. INTRODUCTION

The emergence of Internet of Things where billions of resource constrained devices referred as "smart objects", communicate effectively in tandem to share the highly confidential information. Moreover both conventional devices like the desktop, supercomputers and the resource constrained devices like RFID perform together in a concert, thus leading to a concern of wide range of new security and privacy issues [1]. The greatest challenge is to apply conventional standards to resource constrained devices. The design of computing resources to such resource constrained devices also is a major challenge [2][3].

Over the years there has been extensive research carried out in implementing arithmetic operations, to implement such cryptographic arithmetic in such highly resource constrained devices could be possible only by optimization or the designer may have to trade-off between security and performance. Hence, the need for efficient and effective implementation of arithmetic operation for cryptography is the greatest challenge. The term "effectiveness" means how effectively the data is secured and efficiency means implementing the conventional cryptographic arithmetic operations efficiently considering parameters speed, Area and Power

Engineers have been able to develop powerful devices that have resulted in a huge revolution in the digital world. The revolution has also surfaced issues related to power dissipation, clocks and leakage currents and so on. Developers had to integrate complex operations in mobile phones keeping into account the area, power and timing – important and critical parameters of any design. With wireless devices effortlessly making their way into the market, the key factor is portability and power consumption. The aim of any design is to run the devices for maximum time with minimum requirements like battery, memory, etc

2. LITREATURE SURVEY

Perumalsamy et al. [4] designed an innovative 8bit reversible ALU that dissipates 39% less power and is 10% faster than the conventional design. The same design can be extended to 16bit, 32bit, 64bit and higher bits. With a 3×3 reversible gate1 being the fundamental block of the ALU, a deduction in power dissipation is possible.

Xiaodong Yan and Shuguo Li [5] modified the hardware for modular inversion, expedited the algorithm and then analysed it. The algorithm was accomplished with 256 prime fields. It was observed that the algorithm occupied 10% larger space but was 31% brisk than the Extended Euclidean algorithm.

Ryan and Mill [6] proposed an algorithm that was implemented in a standard processor that usually faced issues because it is reliant on the length of the inputs. The lookup table was constructed so that it could be stored in the cache. Hence restricting the memory access to minimum.

Haniotakis et al. [7] integrated a Manchester Main Carry (MCC) adder in the Vedic multiplier. Analysis of the design was done by integrating a MCC adder in 4bit, 8bit Vedic multiplier and a conventional RCA Vedic multiplier. Even though an increase of 10.8% was observed in the timing, the Power Delay Product (PDP) and the area wa0s reduced by 106% and 80% respectively.

Gururaj et al. [8] studied the Vedic multiplier alongside with the array multiplier. The 8x8 Vedic multiplier exhibited a delay of 18.642ns which was less than the time taken by an array multiplier. Ramasubramanian et al. [9] proposed an algorithm to deduce the multiplicative modular inverse of two numbers that are coprime to each other. The designed incorporated a lookup table, hence it was named LUK-mod-inverse algorithm.

Kaihara et al. [10] has presented the hardware for modular multiplication/division. It contains four steps with the first step being initialization, second step includes selection of the mode of operation, the respective logic for each case, third step comprises of addition operations while the last step produces the result. Nagaria et al. [11] compared a modified array multiplier with regular hydrid adder, an array multiplier using new hybrid adder and a conventional array multiplier. The modified array multiplier has the lowest PDP and driving capability, while the array multiplier with the new hybrid adder has optimal PDP. It was also observed that both the designs suffered from voltage swing problem.

3. ARITHMETIC OPERATIONS

The below section includes the description of the various arithmetic operations implemented.

3.1 Addition

Addition being the basic element of any arithmetic operation. 32bit adders are implemented using reversible logic and the performance is compared with the standard logic. Reversible

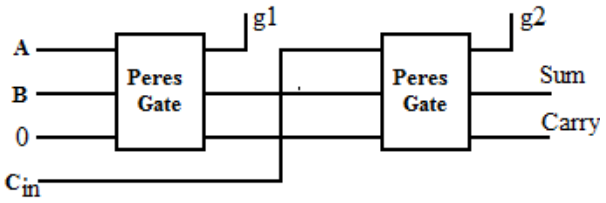

Figure 1: Full Adder using Peres Gates

gate is defined as a gate with equal number of inputs and outputs and is capable of retrieving the inputs from the outputs. The leaf module of the design is a reversible 1-bit full-adder constituting of two Peres-gates and two Feynman-gates. The design is modified to replace the four reversible gates with two reversible gates, Peres-gates. Figure 1 is a description of the modified reversible full adder. An N-bit adder is designed by iteratively placing a full adder. The other adder is implemented using basic gates. The inputs to both the adders are 32bit.

3.2 Multiplication

The performance of any system is dependent on the slowest element. Multiplier being one of the slowest element, designing of a multiplier needs to done taking into consideration critical parameters – area, timing and power. In this paper, three different algorithms are implemented. The first is the multiplication performed using only Reversible gates. Figure 2 is the shows the computation of Partial Products for a 4×4 Reversible multipliers. Figure 3 is the description of 4×4 Reversible multiplier using Peres gate and HNG Gate. Two different versions of Vedic methodology are implemented.

One is using Reversible gates and the other is using Basic

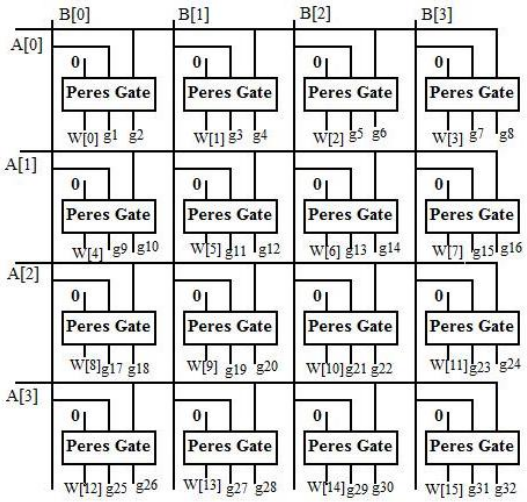

Figure 2: The partial products generation for a 4x4 Reversible

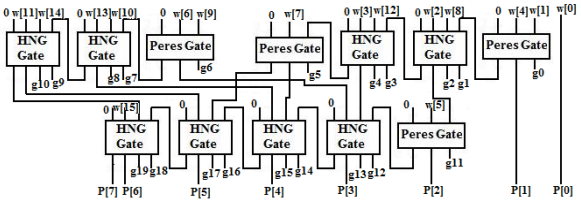

Figure 3: Reversible multiplier Product generation stage

gates. Vedic mathematics constitutes 16 sutras and 13 sub sutras. Among the 16 sutras, there are two multiplication sutras, Urdhva–Tiryagbhyam and Nikhilam Navatascaramam Dasatah sutra. The former is called vertically and crosswire algorithm. 2×2 Vedic multiplier is the base module for an Nbit multiplication. 4×4 multiplication involves breaking of the 4bit input into two groups. Figure 4 is the block description of N×N Vedic Multiplier

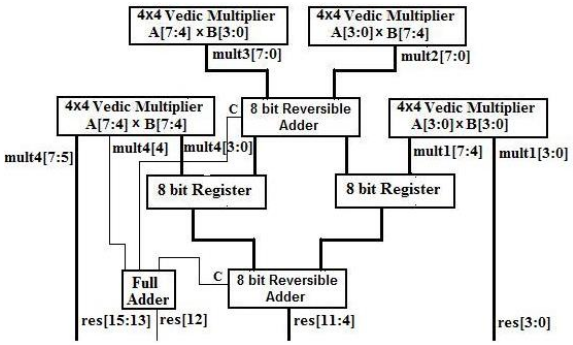

Figure 4: A 4×4 Vedic multiplier using reversible-logic

3.3 Modular Operations

The modular operations form an essential part of cryptography. In this paper, inverse modular, modular multiplication and modular reduction are some of the arithmetic operations that are implemented

The modular multiplication is implemented in two steps: First, the product P = A×B using Vedic/reversible multiplier is computed and then reducing P to compute M = A×B (mod N).

3.3.1Modulo-multiplicative inverse algorithm

Modulo multiplicative inverse algorithm also called as Extended Euclidean algorithm. It utilizes all the common arithmetic operations like addition, shift operation and subtraction, hence reducing the complexity of the algorithm. The algorithm duration is depended on the value of remainder; hence it results in different number loop for different values of the input. It is observed that if the modulo is in the range of $2^{16}+1$ then all the possible values for the input are {0, 1… 65536} and the maximum number of iterations would be eighteen. The Pseudo-code is given in algorithm 1 shows implementation of Modulo-multiplicative inverse algorithm.

Inputs:
(a, b positive integers);

Algorithm:

$q_1 = a \; div \; b;$
k := 1; $q_0 = a$; $r_0 = b$;
while $r_k \neq 0$
k := k + 1
$begin$
$q_k = r_{k-2} \; div \; r_{k-1}$;
$r_k = r_{k-2} \; mod \; r_{k-1}$;
end ; $\%comment$:
$gcd(a,b) = r_{k-1}$

Algorithm 1: Modulo multiplicative inverse algorithm

Algorithm 1 shows the steps for modulo multiplication algorithm.

3.3.2 Montgomery algorithm

Montgomery algorithm is the most efficient and widely applied modular multiplication algorithm. Simple arithmetic operations like addition and shift operations are realized for reduction, and can be easily implemented in VLSI.

INPUTS:T
$X = (X_{K-1}X_{K-2} \dots X_2X_1X_0)$
$Y = (Y_{K-1}Y_{K-2} \dots Y_2Y_1Y_0)$
$P = (P_{K-1}P_{K-2} \dots P_2P_1P_0)$

ALGORITHM:
STEPS:

$P' = -P^{-1} \text{ MOD } B$
$T = 0 \; (T = T_{N-1}T_{N-2}T_2T_1T_0)$
FOR $I = 0$ TO $K-1$
$A = ((T_0 + (X_I * Y_0))\, P') \text{ MOD } B$
$T = (T + X_IY + A_IP)/B$
WHILE $T \geq P$, $T = T - P$
$xyR^{-1} \bmod p = t$

Inputs:
$x = (x_{k-1}x_{k-2} \dots x_2x_1x_0)$
$y = (y_{k-1}y_{k-2} \dots y_2y_1y_0)$
$p = (p_{k-1}p_{k-2} \dots p_2p_1p_0)$

Algorithm:
Steps:
$p' = -p^{-1} \bmod b$
$t = 0 \; (t = t_{n-1}t_{n-2}t_2t_1t_0)$
for $i = 0$ to $k-1$
$a = ((t_0 + (x_i * y_0))\, p') \bmod b$
$t = (t + x_iy + a_ip)/b$
while $t \geq p$, $t = t - p$

Algorithm 2: Montgomery multiplication and montgomerry reduction

Montgomery-modulo algorithm computes *xy* mod *p* for positive integers *x*, *y*, and *p*. It reduces the computation time when multiplication of large numbers is to carry out using same mod m and with limited multipliers.

The Montgomery modular algorithm implemented is modified as in algorithm 2. The inputs, x, y and p are four hexadecimal digits each of 16 bits. The condition for value of R > p. Unlike normal algorithm, the value of R is taken to be equal to b^k where b is the base and k is the width of the input. Here the value of R taken as 16^4. This way the process of converting the inputs into Montgomery domain can done by appending four hexadecimal 0's thus there will be a considerable reduction of two multipliers. Next step is the reduction of the product and returning the product to the integer domain. For

Inputs:

$$z = (z_{k-1}z_{k-2} \dots z_2z_1z_0)$$
$$p = (p_{k-1}p_{k-2} \dots p_2p_1p_0)$$
such that $p_{k-1} \neq 0$

Outputs:

$$y = z \bmod p$$

Steps :

1. $\mu = \left\lfloor \frac{b^{2n}}{p} \right\rfloor$
2. $q_1 = \left\lfloor \frac{z}{b^{n+1}} \right\rfloor$
3. $q_2 = \left\lfloor \frac{q_1 \times \mu}{b^{n+1}} \right\rfloor$
4. $r_1 = z \bmod b^{n+1}$
5. $r_2 = (q_2 \times p) \bmod b^{n+1}$
6. $r = r_1 - r_2$
7. if $r < 0, r = r + b^{n+1}$
8. while $r \geq p, r = r - p$

Algorithm 3 : Barrett Reduction

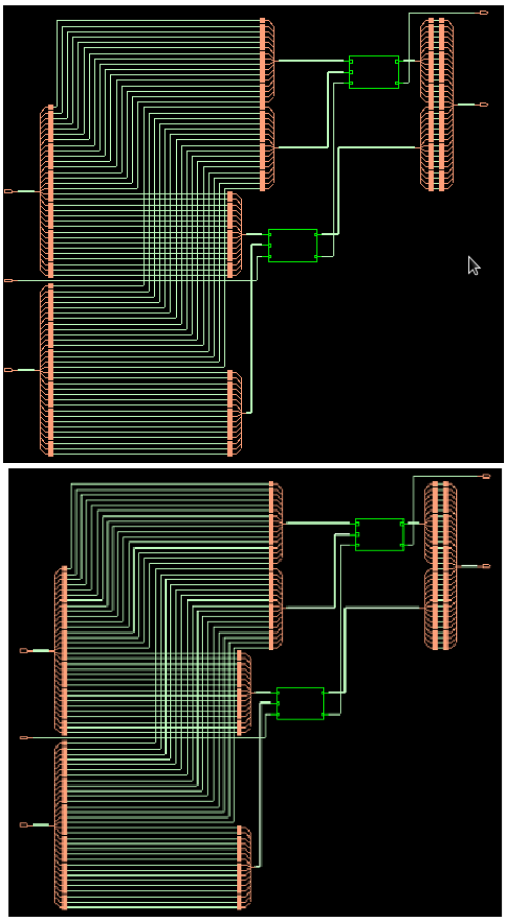

Figure 5: RTL schematics of 32 bit adders using(a) standard logic and (b)using reversible logic

this purpose, another algorithm called Montgomery Reduction is used which is also the last step of Montgomery modular operation. The additional data that is needed for the algorithm is inverse of p that can be calculated using an inverse algorithm.

Modular multiplication with Barrett reduction

The Barrett reduction is based on the knowledge that $\left\lfloor \frac{z}{p} \right\rfloor$ can be written in the form

$$Q = \left\lfloor \left(\frac{z}{b^{n-1}}\right)\left(\frac{b^{2n}}{p}\right)\left(\frac{1}{b^{n+1}}\right) \right\rfloor \quad \dots\dots \text{(3.1)}$$

The purpose of Barrett reduction is to compute z mod p where z and p are inputs given to the algorithm and n is the length of p. The lengths of z and p are 32bits and 16bits respectively. *b* is the word-size of the processor, hence it is usually assumed to be greater than 3. Here it is assumed to be 16. The algorithm 3 needs a pre-computed value μ making the algorithm very beneficial if many reductions are to be performed using the same modulus

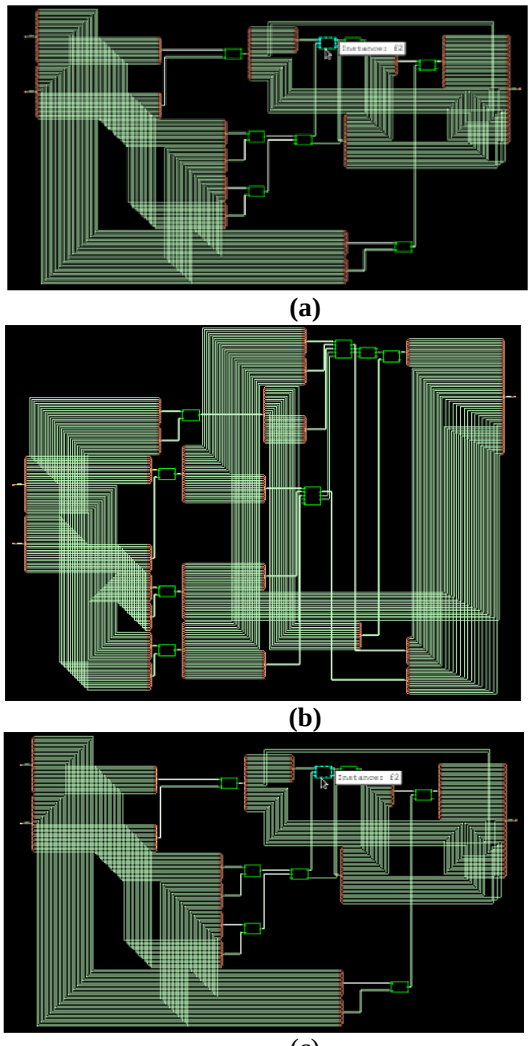

Figure 6 :(a)RTL schematics of 32 bit multiplier using(a) standard logic (b)using reversible logic (c) reversible multiplier

4. IMPLEMENTATION RESULTS AND ANALYSIS

Addition

The 32 bit adders are implemented using basic gates and reversible gates. Figure 5 shows the RTL schematic using Xilinx 14.2 of a 32-bit adder using standard logic gates and 32bit adder using reversible logic gates.

Multiplication

Three multipliers are implemented using two methodologies, Vedic mathematics and Reversible logic. Figure 6 a shows the RTL schematic of multiplier using only reversible gates and figure b shows Vedic methodology using reversible gates and fig c vedic multiplier using standard logic gates.

Modular Arithmetic

Figure 7 shows the RTL schematic of the modulo multiplicative inverse algorithm

Figure 8 shows the RTL schematics of the barett reduction algorithms.

Figure 9 shows the RTL schematics of the mod- without division algorithms.

Figure 10 is the RTL schematic of Montgomery modular algorithm.

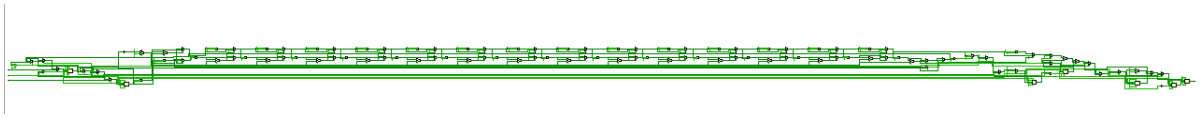

Figure 7: Modulo Multiplicative inverse

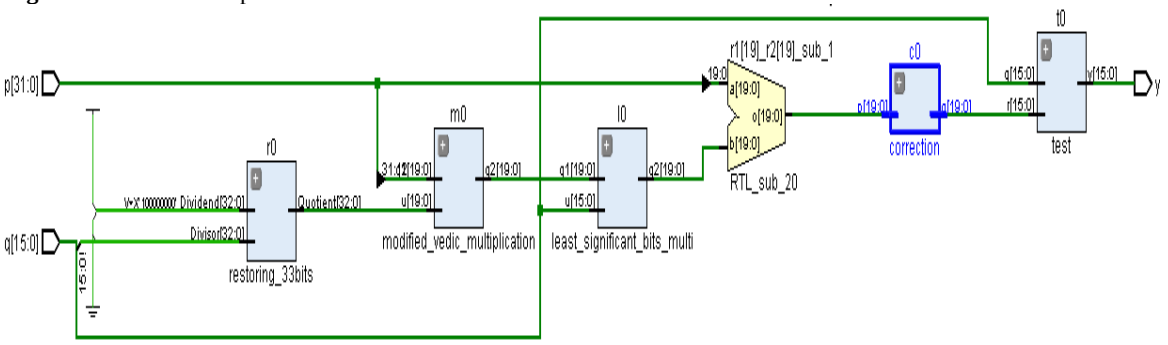

Figure 8: RTL schematics of the barett reduction algorithms

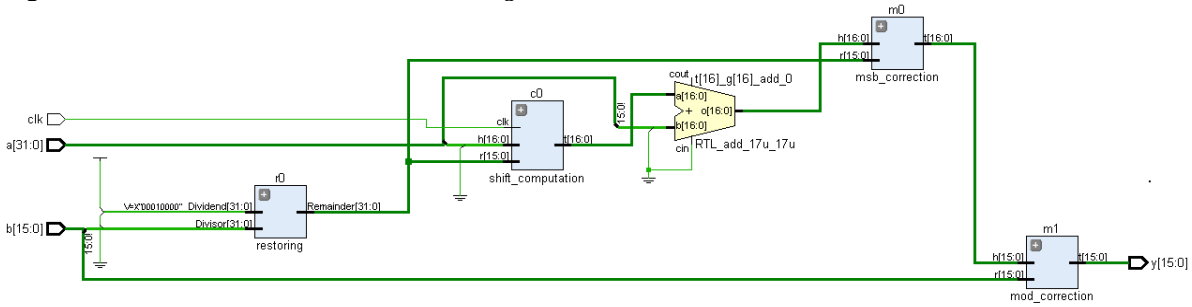

Figure 9: RTL schematics of the modwithout division algorithms.

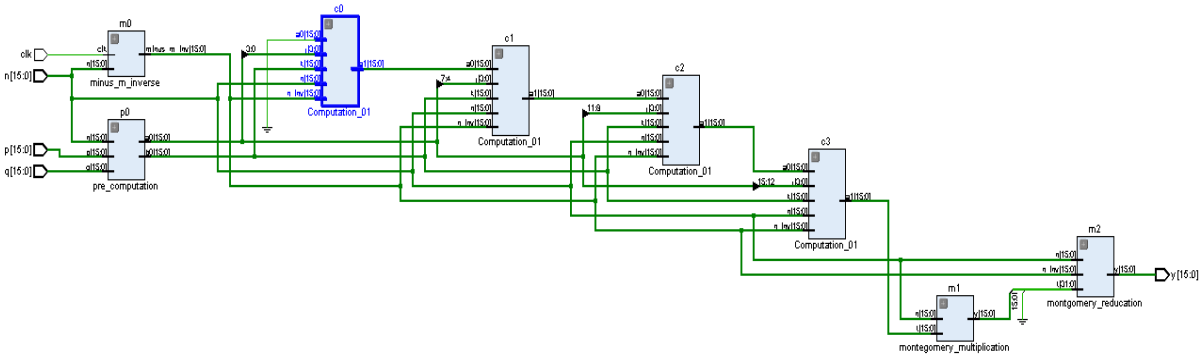

Figure 10: RTL schematic of Montgomery modular algorithm

The below section tabulates the area, power required and the timing of the implemented circuits. Table 1 tabulates the synthesis report of the implemented addition, multiplication algorithms.

Table 2 tabulates the synthesis report of the implemented Modular algorithms.

Figures 11 and 12 show the comparative FPGA performance analysis for the proposed multiplication and modulo algorithms on Virtex-6 FPGA. Figure 12 shows the area and power dissipation using 45nm Cadence tool shows that the area and power dissipation in better than the standard gate implementation for both addition and multiplication

Table 1: Synthesis report of addition, multiplication algorithms.

Cell Name	Spartan 6			45nm Technology		
	Area			Timing (ns)	Area	Power (µW)
	Slices Registers	LUT	IOB's			
32bit Adder using Basic Gates	48	0	98	9.309	501	36.548
32bit Adder using Reversible Gates	48	0	98	11.720	545	37.708
Reversible Multiplier (RM)	2843	0	128	18.612	12228	1236.625
Vedic Methodology using Reversible Gates	2609	0	128	22.384	35630	2293.531
Vedic Methodology using Basic Gates	2358	0	128	16.868	16103	1426.011

Table 2: Synthesis report of modular algorithms

Algorithm	Spartan 6			45nm Technology		
	Area			Timing (ns)	Area	Power (µW)
	Slices Registers	LUT	IOB			
Modulo Multiplicative Inverse	83	610	49	54.029	5988	5079.74
Barrett Reduction (BR)	18	2771	64	3.358	7701	8253.96
Mod without Division (MWD)	75	1555	64	11.808	9700	11304.02
Modified Montgomery Modular(MM)	71	4229	65	54.028	1793	92.70
Ref[10]	--	--	--	303.3	2695	--

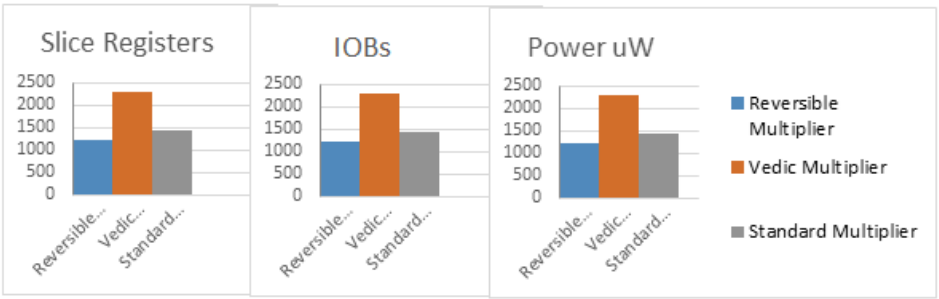

Figure 11 : Comparison of Slices IOBs and power of Multipliers using different logic (Xilinx Tool)

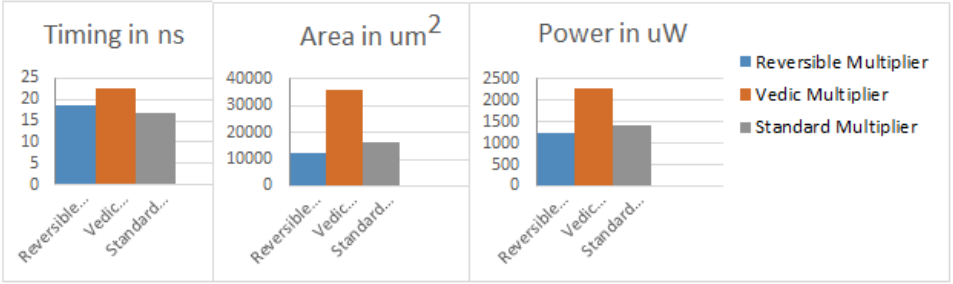

Figure 12 : Comparison of Timing Area and power of Multipliers using different logic (Cadence Tool 45nm Technology)

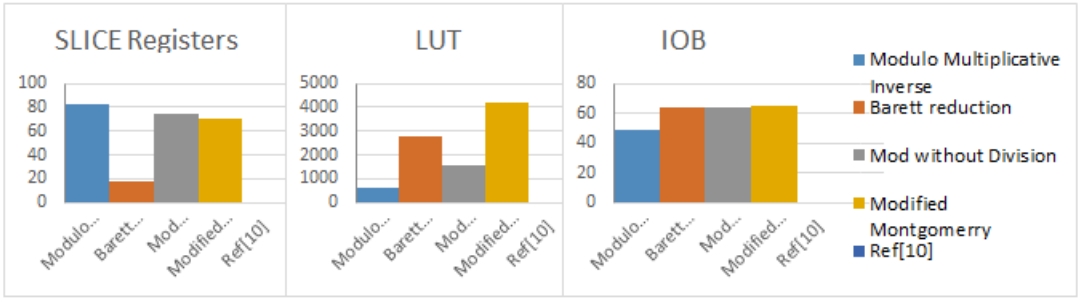

Figure 13 : Comparison of Slices, LUT and IOBs of Modulo Algorithms (Xilinx Tool)

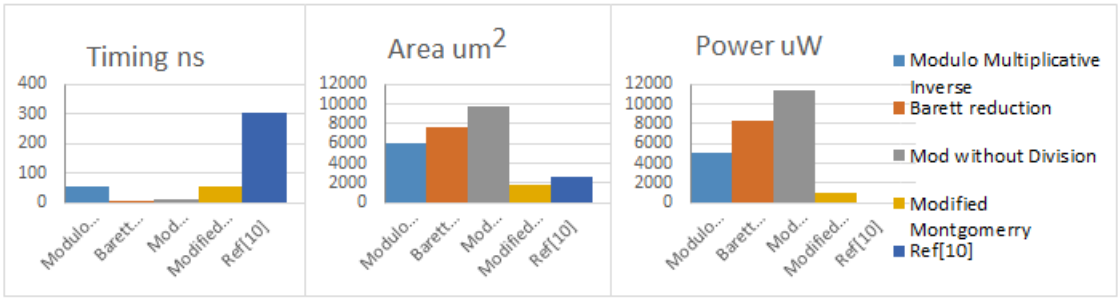

Figure 14 : Comparison of Timing Area and power for modulo Algorithm (Cadence Tool 45nm Technology)

algorithms. Figure 13 and 14 shows the analysis of modular multiplication algorithms using both tools. And the above algorithms has been compared with the implementation in Ref[10] and has found better timing and area

5. CONCLUSION

Parametric analysis of the algorithms helps the designer to decide the best and the most effective algorithm based on the resources that are available. Form the tabulated values it can be observed that despite of the adders having more or less same area and timing constraints in Spartan 6, a notable difference can be observed in Cadence. Hence from the results it can be concluded that the Basic adder appears superior with a reduction of 20%, 8% and 3% in timing, area and power respectively. In multiplication, a mixed response is observed. In Spartan 6, Vedic methodology with Basic gates occupies the least area of 2358 slice registers while in Cadence, Reversible multiplier occupies the least area of 12228 cell area. Reversible multipliers consumes the least power of 1236.625µW. In Spartan 6, the quickest algorithm is Reversible multiplier with a duration of 52.694ns while in Cadence it is Vedic methodology with Basic gates with a duration of 16.868ns. The timing and the area of the implemented Montgomery modular algorithm is compared with [10]. The number of slice registers utilized is reduced by 72% and an increase in slice LUT's by 30% is observed. They need to be modified in such a way that they can be applied to inputs unconditionally. There is scope for modification in reversible multiplier and the reversible adder as well. Other multiplication algorithms like Kasturba multipliers and so on need to be compared with the reversible multipliers and Vedic methodology to determine the most effective and efficient multiplier. The modular operations also need to be researched more extensively. The condition on the base selected in Barrett reduction makes it a selectively used algorithm. The requirement that the length of the input should be greater than modulus value is a limitation of the mod without division algorithm. Further reduction of timing in the proposed Montgomery modular operation is required.

ACKNOWLEDGEMENT

The authors would like to thank the Department of Electronics and Communication and Engineering, Canara Engineering College Mangalore and Visvesvaraya Technological University, Belagavi for the support for carrying out the research work.

REFERENCES

1. Bhimani, Anish, "**Securing the commercial Internet",** Communications of the ACM 39.6, pp. 29-35, 1996.
2. Mario Weber, Marija Boban, "**Security challenges of the Internet of Things**", MIPRO 2016, May 30 - June 3, 2016, Opatija, Croatia.
3. Kiran Kumar V.G., Shantharama Rai C. (2019) **Implementation and Analysis of Cryptographic Ciphers in FPGA**. In: Abraham A., Dutta P., Mandal J., Bhattacharya A., Dutta S. (eds) Emerging Technologies in Data Mining and Information Security. Advances in Intelligent Systems and Computing, vol 755. Springer, Singapore.
4. Kamaraj Arunachalam, Marichamy Perumalsamy, C. Kalyana Sundaram, and J. Senthil Kumar, "**Design and implementation of a reversible logic based 8 bit arithmetic and logic unit**", International Journal of Computers and Applications vol. 36, Pp. 49-55, July 2015.

5. Xiaodong Yan and Shuguo Li, "**Modified modular inversion algorithm for vlsi implementation**", 2007 7th International Conference on ASIC, 22-25 Oct. 2007.
6. Mark A. Will and Ryan K. L. Ko, "**Computing mod without mod**", Cryptology ePrint Archive, Report 2014/755, 28 Sep 2014.
7. Raghava Katreepalli and Themistoklis Haniotakis, "**Power-delay-area efficient design of vedic multiplier using adaptable manchester carry chain adder**", 2017 International Conference on Communication and Signal Processing, 6-8 April, 2017.
8. V Meghana, Sandhya S, Aparna R and C Gururaj, "**High speed multiplier implementation based on vedic mathematics**", 2015 International Conference on Smart Sensors and Systems (IC-SSS), 21-23 Dec. 2015.
9. Satyanarayana Vollala, B. Shameedha Begum and N. Ramasubramanian, "**Hardware design for multiplicative modular inverse**",2015 International Conference on Computing and Network Communications (CoCoNet'15), 16-19 Dec. 2015.
10. Marcelo E. Kaihara and Naofumi Takagi, "**A hardware algorithm for modular multiplication/division**", IEEE Transactions On Computers, vol. 54, pp. 12-21, January 2005.
11. Priyanka Srivastava, Vishant, Rakesh K. Singh and R. K. Nagaria, "**Design and implementation of high performance array multipliers for digital circuits**", 2013 Students Conference on Engineering and Systems (SCES), 12-14 April 2013.
12. S.P.Pohokar, R.S.Sisal, K.M.Gaikwad, M.M.Patil and Rushikesh Borse, "**Design and implementation of 16 x 16 multiplier using vedic mathematics**", 2015 International Conference on Industrial Instrumentation and Control (ICIC), 28-30 May 28-30, 2015.
13. Ankush Yete, Ananya Kajava P, Hazel Melita Rodrigues, Namratha P and Kiran Kumar V.G, "**Implementation of montgomery modular multiplication using high speed multiplier**", 2013 International Journal of Current Engineering and Scientific Research (IJCESR), 7-8 Dec. 2013.
14. M.Siva Kumar, Syed Inthiyaz, J. Dhamini, A.Sanjay, U.Chandu Srinivas, "**Delay Estimation of Different Approximate Adders using Mentor Graphics**", International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, No.6, November – December 2019. https://doi.org/10.30534/ijatcse/2019/141862019
15. M Siva Kumar, Fazal Noorbasha, Syed Inthiyaz, M. Jameela, A. Sandhya, Md. Imran, Sanath Kumar Tulasi, "**Low Power Carry Look-Ahead Adder using Transmission Gate Multiplexer**", International Journal of Emerging Trends in Engineering Research, Volume 8, No. 1 January 2020. https://doi.org/10.30534/ijeter/2020/03812020.
16. J. Lakshmi Prasanna , R.S. Ernest Ravindran , M. Ravi Kumar , K. Sree Pooja , U V S Sumanth , Shaik Ahamed and Chella Santhosh, "**Design of BCD Adder using Quantum Cellular Automata", International Journal of Advanced Trends in Computer Science and Engineering**, Volume 9, No.1, January – February 2020. https://doi.org/10.30534/ijatcse/2020/79912020
17. Asmita Poojari , Nagesh HR , Kiran Kumar V G, Shantharama Rai C, "**A Novel Key Scheduling Algorithm for Lightweight Cryptographic Applications**", International Journal of Advanced Trends in Computer Science and Engineering, Volume 9, No.1, January – February 2020. https://doi.org/10.30534/ijatcse/2020/96912020.
18. Kiran Kumar V G, C Shantharama Rai, "Low Power High Speed Arithmetic Circuits", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-2, July 2019.