FPGA Implementation of Enhanced Montgomery Modular for Fast Multiplication

[1]Akanksha Jain, [2]Prof Rajesh khtri, [3]Dr PP Bansod

[1,2] Department of Electronics and Instrumentation Engineering, [3]Department of Biomedical Engineering
Shri G.S. Institute of Technology and Science
Indore, India

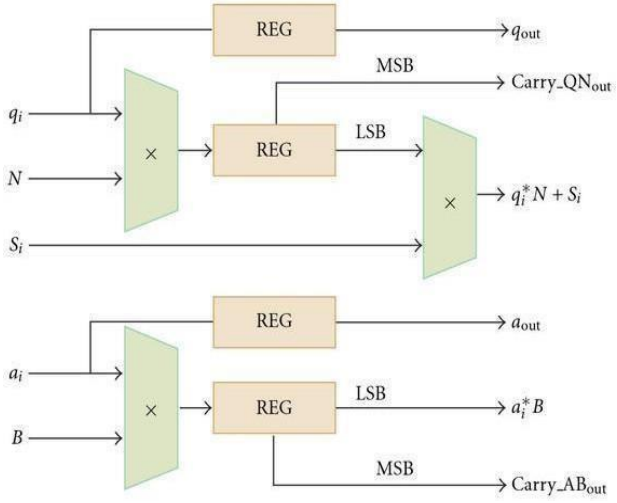
***Abstract*- This Paper proposed an enhanced Montgomery and efficient implementation of Modular Multiplication. Cryptography process is used for providing high information security when a data is transferred from transmitter to receiver. Various using methods like RSA, ECC, the Digital Signature Algorithm. The propose Montgomery algorithm usin RSA algorithim of cryptography is implemented in two different input both the inputs are 8 bit input. Coding have been done in Verilog language and the results are simulated on Vivado Software. For physical testing, we have used an FPGA NESYS 4 DDR hardware board that have Artix-7 FPGA chip on it produced by digilent company. The propose method shows good results in term of the number of slice flip flop , LUTs, and number of IOBs and power consumption. The proposed method shows better results as compare to other previous methods in term of different result parameters.**

INTRODUCTION:

A key issue for researchers in recent years has been data encryption. Technology is evolving at a rapid pace, necessitating the use of cutting-edge encryption techniques. Modular multiplication for advanced encryption and cryptography was proposed in this research article using the montgomery method. The advantage of using bit shift multiples of modulus to clear out the least significant bits before shifting them out. Even if the modulus of a regular modular multiplication results in a value that is less than its own modulus, it will be deducted from the result until it reaches zero. There is no need for subtractions in Montgomery multiplication since the bits are shifted out when the multiplicand is handled. In this study, a unified and dual-radix architecture is used to achieve Montgomery multiplication.

The Figure 1 to the right shows an example of a multiplier block layout. The Montgomery multiplication is the building block for Die-Hellman and RSA public-key cryptosystems' modular exponentiation operations. The most compelling reason to investigate fast and inexpensive modular multipliers for long integers is the search for faster and cheaper modular exponentiation processors. An elliptic key encryption over the niteeld GF has recently been implemented using Montgomery multiplications. (p). Additionally, discontinuous exponentiation in excess of GF(2k) and elliptic key cryptography became possible due to the introduction of Montgomery multiplication in GF(2k), which was first reported by. By developing an extensible Montgomery multiplication architecture, we can now look at different parts of the design space and see how different trade-offs affect performance on a small chip. Our design must be scalable, and we explain the broader theoretical issues with Montgomery multiplication afterward.

Fig. 1 Shows the Multiplier block architecture



This is followed by a discussion of the parallel evaluation of a word-based algorithm. This method is used to derive and present the modular multiplier's architecture. Furthermore, they conduct simulations to determine area/time tradeoffs and present an arts order evaluation of the multiplier's performance for various operand precisions. Despite more than three decades of research, public-key cryptography (PKC) is still considered computationally demanding, especially when used on embedded processors.

Since operations like exponentiation and scalar multiplication use operands with sizes in the hundreds or thousands of bits, this is most likely the case. RSA and ECC are two types of public-key algorithms that use multi-precision modular arithmetic for their

security. This is especially true for modular multiplication on embedded CPUs. For modular multiplication, cryptographers have devised a number of effective reduction algorithms that can be implemented in the most efficient manner. An essential modular reduction technique is the Montgomery algorithm, which was first presented in 1985[11] and has since been widely used in real-world applications. Further examples of reduction algorithms include Barrett [2] and Quisquater [13, 14]

MONTGOMERY MODULE MULTIPLICATION

The following are the definitions of unified and dual-radix. All the hardware and software needed to work with operands in both prime and binary extension fields is referred to as a unified architecture. The multiplier for GF(p) in [4] was modified in [3] to illustrate that a unified multiplier is possible with relatively modest alterations. If the unified multiplier uses a bigger radix value for GF(2n) than the radix for GF, it is referred to as a dual-radix multiplier (p). For the Montgomery multiplier's hardware, the term "architecture" is often employed. Every clock cycle, n bits of the multiplicand are processed by a radix–2n multiplier. Radix (2,4) multipliers are multipliers that operate in radix–2 for GF(p) and radix–4 for GF(p) (2n).

The radix (2,4) multiplier design is used in this thesis. There are important time–area concerns in the design of a dual radix multiplier, as adding an extra radix should have minimal impact on signal propagation time while keeping silicon area to a minimum. Multiplying GF(2n) with the same radix as multiplying GF(2n) is known as dual-radix architecture, becauseit uses a different radex for the two operations (p). The hardware of the Montgomery multiplier is referred to as its "architecture." The multiplicand is multiplied by a radix-2n multiplier per clock cycle, processing n bits of the multiplicand. A radix (2,4) multiplier is a multiplier that works with GF(p) in radix–2 and with GF in radix–4 (or vice versa) (2n). The radix (2,4) multiplier design is used in this thesis. When designing a dual-radix multiplier, there are important time-area concerns. Adding an extra radix should have little effect on the time it takes for a signal to travel and take up as little silicon space as possible. The Montgomery modular multiplication is discussed in detail in the preceding section.

In Section II, we'll go over the many types of Montgomery modular multiplication. This section-III focuses on past research that was given by other researchers. Finally, explain the strategy given in Section IV. In Section V, the proposed method's simulations and outcomes are examined. Section VI concludes with a discussion of the conclusion.

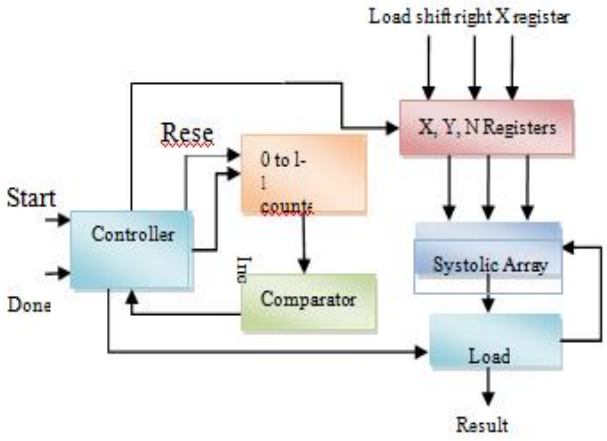


Fig.2 Montgomery Modular Multiplier Circuit

LITERATURE REVIEW

In this section, we will discuss the review of literature on the Montgomery modular multiplier and also the different VLSI. *Pajuelo-Holguera, et.al. (2021),* An FPGA device and the HLS programming approach were used to create the Montgomery Modular Multiplier. Parallel multiplier and parallel adder were built in order to implement the MMM in this manner. Authors tested this parallel hardware proposal against a sequential hardware version, a software version, and fifteen other studies in the literature. The hardware sequential version and the software implementation, respectively, benefit from a speedup of 8 and 18.5. In addition, our concept outperforms the competition in terms of turnaround time and effectiveness [1]. Gu, Z., et al. (2020, June) A new approach of modular multiplication based on Karatsuba-like multiplications was described in this study. NIST primes, as well as generic moduli based on Montgomery modular multiplication, benefit from this strategy. With our strategy, we can reduce the number of steps required to do integer multiplications from three to one [3]. Parihar, and others (2019), Presented in this study are the findings. Tests show that the new suggested multiplier requires less clock cycles than earlier versions. A whole MM can be executed in the least amount of time possible by the multiplier under consideration. Because of the multiplier's fast speed and the tiny number of clock cycles, the system achieves an extremely high throughput rate. In order to include more hardware for format conversion, the suggested multiplier requires a larger footprint. This multiplier's surface area is, nonetheless, equivalent to other multipliers. There is a 44.8 percent reduction in clock cycles and a 50.2% reduction in the time it takes to complete an MM with the proposed multiplier compared to the current MM CCSA.

The study by Verma et al. FPGAs were used to implement RSA on early word- based radix 2 and 4 architectures given inthis study. In early word-based systems, the most significant bits may be computed using just the most fundamental operations. As a result of the DSP48Es, which add 48 bits and run at high frequency, the word size was set at 48 bits for this project. The cycle time of a Montgomery design is mostly dictated by the addition operation in word-based Montgomery designs. This improvement has been made possible by the use of DSP48Es for addition and an early word-based technique for determining bits on FPGAs. [9].

Rabet, et.al 2017, This study, presented an algorithm for large prime-characteristic finite fields (Fp). They show the results of our design after routing and Placement on Xilinx's Artix-7 and Virtex-5 Field Programmable Gate Arrays. For anyimplementation that requires modular multiplication and uses cryptographic algorithms like RSA, ECC, or pairing-based cryptography, we have systolic implementations that can be applied to it. In order to work with the architectures and designswe created, the features of the Field-Programmable Gate Arrays were adapted. A satisfactory performance was achieved in the latency area with the NW-8 design. This architecture can run all bit lengths associated with traditional security levels in 33 clock cycles or less (128, 256, 512, or 1024 bits). Although it takes 66 clock cycles to complete the same amount of work as the NW-8, the NW-16 offers significant improvements in terms of surface area compared to that processor. A variety of word counts can be accommodated by our systolic design, which employs the method CIOS. Kuang, et.al. It was in 2016 that a radix-4 scalable architecture for Montgomery modular multiplications was presented in this research work. Our experiments have shown that our design uses significantly less power and significantly less space for the hardware than any previous work.

To use the multiplier, one must meet various requirements, such as the amount of space in hardware and the amount of power it consumes, before it can be used in various applications. A radix-4-based architecture for Montgomery modular multiplications has been proposed in this short paper. The results of our experiments show that our design requires significantly less space for hardware and consumes significantly less power than any other. It is possible to use the proposedmultiplier for a wide range of applications [15] provided that they meet various requirements, including the amount of space that the hardware needs to occupy. These include the amount of space that the hardware occupies and how much energy it uses.

This research was done by Kuang et al. (2015, FCS-based multipliers keep the input and output operands of the Montgomery MM in the carry- save form to avoid the format conversion, resulting in fewer clock cycles but a larger area than the SCS- based multiplier.This paper proposes a low-cost and high-performance Montgomery multiplier based on a modified version of the SCS-based Montgomery multiplication algorithm. Through the use of k-partitions, the multiplier operand is divided into smaller pieces that can be processed in parallel and independently, reducing the overall amount of computational complexity that is needed. Another method for modular exponentiation, known as the Square and Multiply method, is implemented and compared to an ordinary Montgomery multiplier and the k-partition method for four sets of input bit lengths of 128, 256, 512, and 1024 bits, respectively.

In comparison to the other two methods, the Square and Multiply method uses significantly less power [21]. The high- speed Montgomery modular multipliers now have more registers and higher energy consumption, allowing for faster decryption and encryption thanks to redundant carry save formats for all inputs and outputs of the modular multiplication. Kuang et al. presented their findings in this study (2012).

PROPOSED METHODOLOGY

In this part of the article, we will go over the suggested procedure. In reality, many complicated cryptographic algorithms are built on top of relatively straightforward modular arithmetic. Since integers are the only numbers that can be used in modular arithmetic, only addition, subtraction, multiplication, and division can be performed on them. Only in modular arithmetic are all operations performed in relation to a positive integer, also known as the modulus, as opposed to the elementary arithmetic you learned. This is the only significant difference between the two. The proposed approach is based on modular arithmetic computation, specifically Montgomery modular multiplication, more commonly known as Montgomery multiplication.

It's a method for quickly multiplying modular numbers. The Montgomery modular multiplication technique employs a specialised representation of numbers known as the Montgomery form. Algorithms use Montgomery forms of a and b for the Montgomery form of ab modified by N, which is more efficient. By dividing ab by N and keeping only the remainder, the conventional method of modular multiplication reduces the size of the double- width product ab.

This division necessitates an estimate and correction of the quotient digits. If R > N is coprime to N, then all that must be divided for a Montgomery multiplication is R by the Montgomery form's only dependent variable R. Selecting the constant R's value in such a way that division by R is simple can significantly speed up the algorithm's computation time.

In the above discuss the basic of montgomery multiplication. Now discuss the proposed method fast montgomery multiplication that is based on counter approach. In the proposed method use counter approach shown in below algorithms.

Montgomery Modular Multiplication Algorithm

Let N be a k-bit odd number, and let R be an additional factor defined as 2k mod N, where 2k1 N 2k.The N-residue is the product of two integers, x and y, where x, y N.with respect to R can be written as$X = x \times R \pmod N$

$Y = y \times R \pmod N$ (1)

Based on (1), the Montgomery modular product Z of X and Y can be obtained as $Z= X \times Y \times R{-}1 \pmod N$ (2) where $R{-}1$is the inverse of R modulo N, i.e., $R \times R{-}1 = 1 \pmod N$.

Based on (1), the Montgomery modular product Z of X and Y can be obtained as $Z= X \times Y \times R{-}1 \pmod N$ (2) where $R{-}1$is the inverse of R modulo N, i.e., $R \times R{-}1 = 1 \pmod N$.

Algorithm 1 illustrates the Montgomery modular product of X and Y using the radix-2 version of the Montgomery modular multiplication algorithm, designated as Algorithm MM. Observe that Algorithm 1's Xi notation indicates the i^{th} bit of X in binary form. Furthermore, a segment of X from the i^{th} bit to the j^{th} bit is denoted by the notation Xi: j. Algorithm MM's S has a convergence range of 0 S 2N/2 + 2N/4 + + + 2N/2k1 2N.

Algorithm 1- Algorithm MM52: 5-to-2 CSA Montgomery Multiplication

Inputs : $A1, A2, B1, B2, N$ (modulus)
Outputs : $S1[k], S2[k]$

1. $S1[0] = 0;$
2. $S2[0] = 0;$
3. for $i = 0$ to $k - 1$ {
4. $q_i = (S1[i]_0 + S2[i]_0 + A_i \times (B1_0 + B2_0)) \bmod 2;$
5. $(S1[i+1], S2[i+1]) = (S1[i] + S2[i] + A_i \times (B1 + B2) + q_i \times N) / 2;$
6. }
7. return $S1[k], S2[k];$

Algorithm 1 illustrates the Montgomery modular product of X and Y using the radix-2 version of the Montgomery modular multiplication algorithm, designated as Algorithm MM. Observe that Algorithm 1's Xi notation indicates the i^{th} bit of X in binary form. Furthermore, a segment of X from the i^{th} bit to the j^{th} bit is denoted by the notation Xi: j. Algorithm MM's S has a convergence range of 0 S 2N/2 + 2N/4 + + + 2N/2k1 2N.

Flow Chart

In the next section, we'll talk about how the proposed Montgomery Multiplication design was tested and whattheresults were. The Montgomery multiplication algorithm [3], [6,] [10], [11] can be used to efficiently perform modular multiplication. Two numbers are multiplied by this procedure modulo P.avoiding division by P's modulus to get the finished item, a series of additions are made. Let themultiplicand multiplier, and modulus each be represented by an integer (a, B, P).

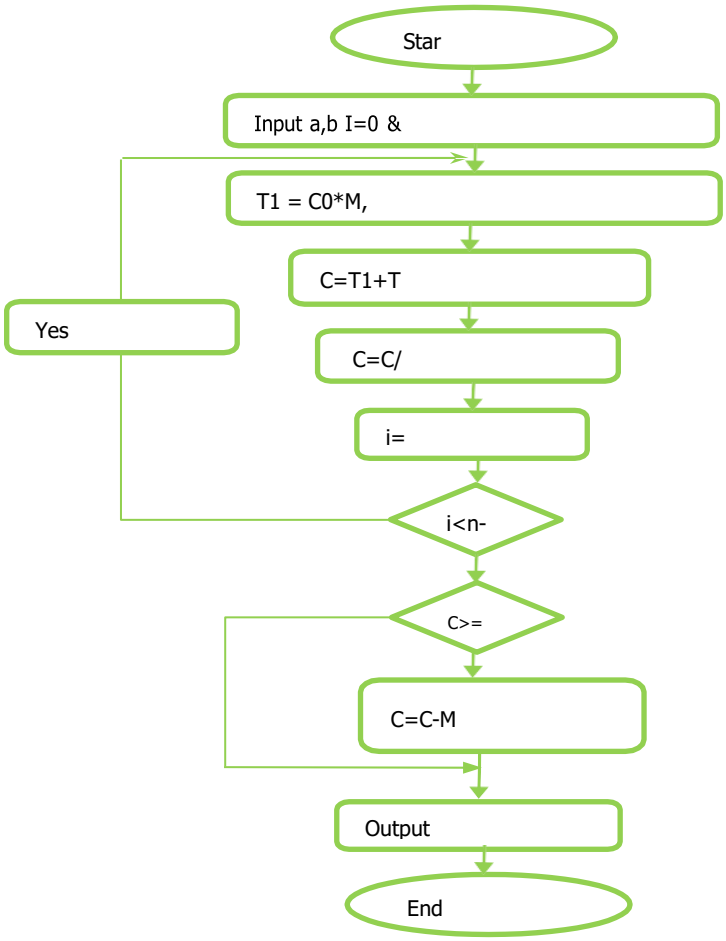


Fig 3: Flowchart of Propose Design

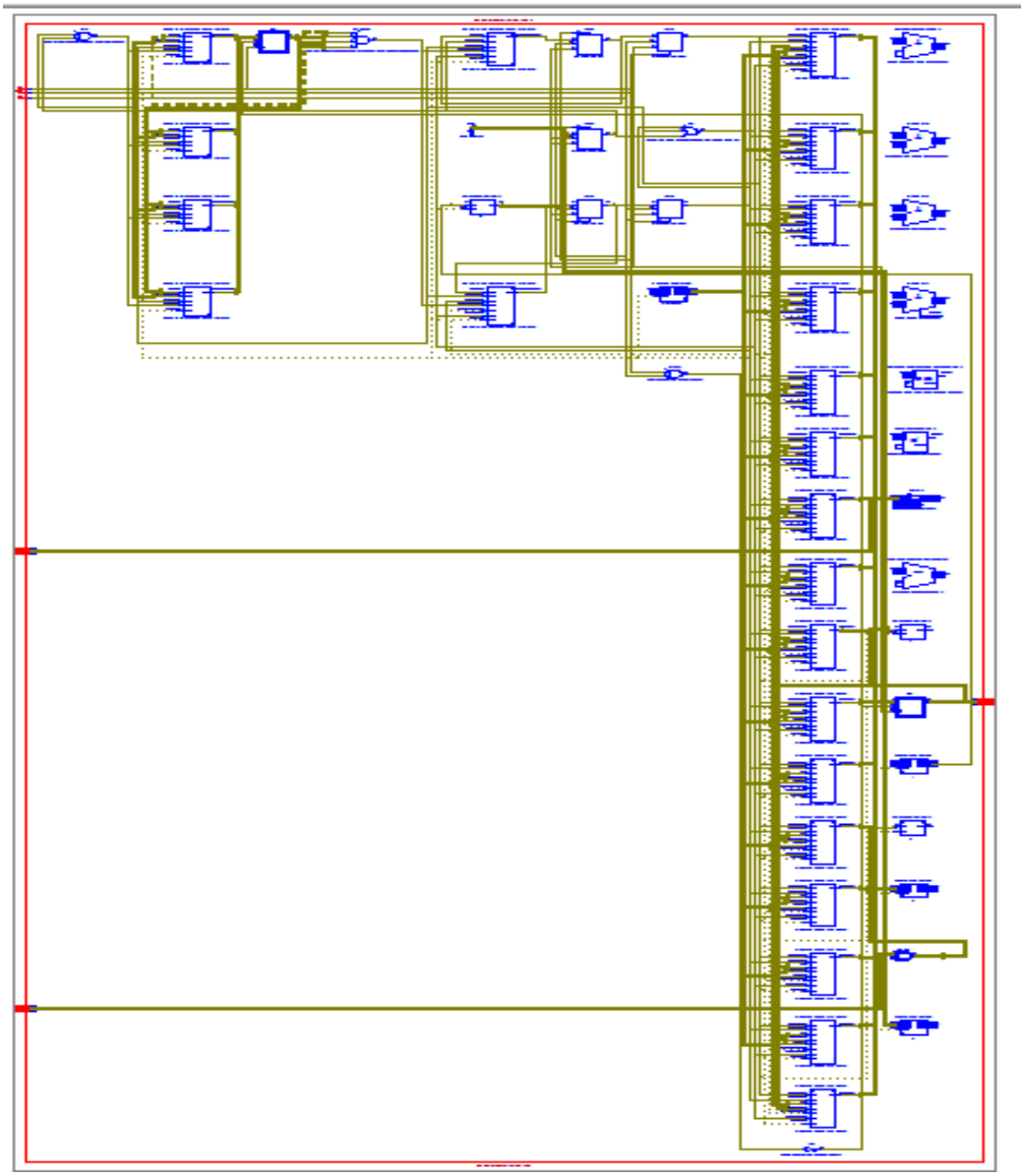
4. SIMULATION RESULTS

In this section, we are describing the implementation details and design issues for our proposed research work. By searching, we have observed that for our proposed work, Vivado Software is the well- known platform of Xillinx to performthe suggested approach. We tend to perform some experimental tasks in verilog code on VIvado Software.

4.1 Result Parameters

The RTL design of the proposed improved Montgomery Multiplication design can be seen in Figure 4, which can be found below. Register-transfer-level extrapolation is used in hardware description languages (HDLs) [16] [20] like Verilog and VHDL to construct high-level models of a circuit, from which lower- level views and, eventually, real routing can be determined. Figure 4 (a) is a demonstration of and Figure 4 (b) is a view of the proposed FIR design as seen from the perspective of inter RTL technology.

Fig. 4 RTL Design of Proposed Montgomery Multiplication



4.2 Simulation wave

Shows Fig. 5. in this section represents the simulated waveform of Proposed Montgomery Modular Multiplication on Vivado Software. When 8 bit inputs A and B are given in the program and the modulo value is fixed in the code then theoutput is come shown in the figure it gives Remainder of the multiplication of the inputs no. in help in the Cryptographykeys generated for security purpose.

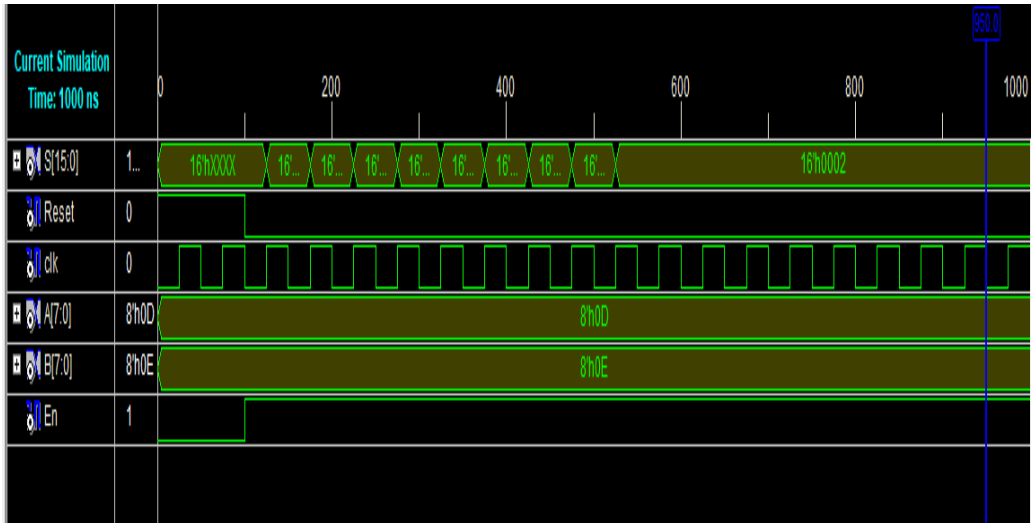


Fig 5 Simulated Output Of Proposed Montgomery Modular Multiplication

5 . Implemented Results

In the given figure 6, which shows the FPGA board that's NEXYS 4, apply the UCF file to this board. This FGPA board is made by Digilent. The proposed method is validated on this board and achieves the same result as the simulation of Montgomery Multiplication. It shows same outputs in Synthesize on FPGA Board which shown in figure 5 in Simulated wave.

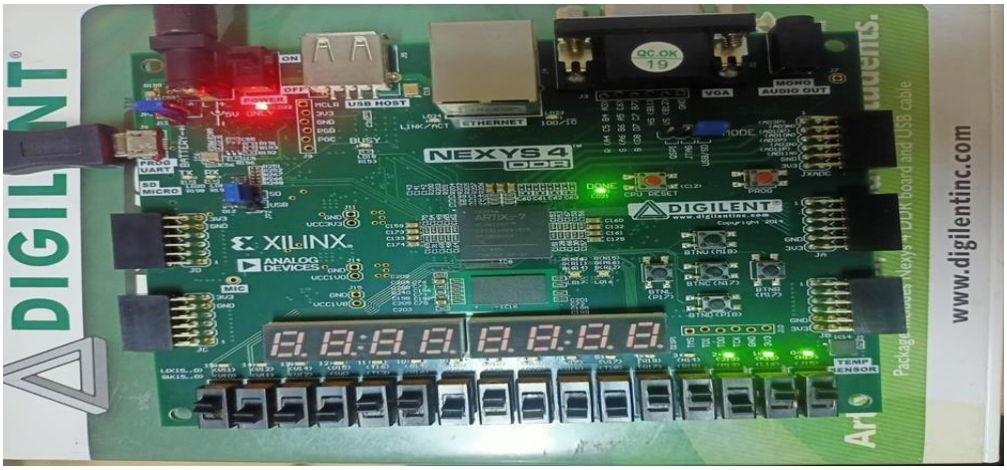


Fig 6 Shows FPGA NEXYS DDR 4 Board

In the above TABLE I and II shows the different result parameters calculated in this research work. This is taken less Noof LUT(Look up Table). Less no of Flip- flops. Based on the implementation results, it is know that the proposed designrequires less area consumption than the existing deigns. the performance of Montgomery multiplication algorithm is improved there by the low space complexity is achieved in performing RSA cryptosystems.

Table 1. Result Device Dynamic Of Proposed Method

Name	Unit	Percentage
Total on chip power	6.531W	-
Junction temperature	54.80 C	-
Device static	0.138W	2%
Device dynamic	6.394W	98%
Signal	0.263W	4%
Logic	0.202W	3%
Input/output	5.929W	93%

layout design of the Synthesize design code gives correcponding outputs. In this section, input output ports are 35, cells182 and 264 nets.
For example, on-chip power and various other performance parameters are shown below. calculation of power andactivity based on the implemented net list and other data sources such as constraint and simulation files.

S. No.	Year / Ref	Method	FGPA board	LUT's	Power(W)
01	2022/ Presented	Proposed Montgomery Multiplication.	Artix-7	64	6.53
02	2020	Montgomery modular multiplier	Artix-7	80	6.61
03	2017	Broken-Karatsuba multiplication	Aitix-7	164	7.0
04	2013	Montgomery Modular Multiplication	Artix-7	180	10.8

TABLE 2 Result Of Device Utilization Summary

In the above figure 6, which shows the FPGA board that's NEXYS 4, apply the UCF file to this board. This FGPA boardis made by Digilent. The proposed method is validated on this board and achieves the same result as the simulation of Montgomery Multiplication.
In the below figure 7, the input output synthesized design outcomes of the proposed design. The I/O synthesized proposed montgomery multiplication algorithm was designed with the help of the UCF file. The UCF file is generated after the verification of simulation outcomes.

In the next figure below, figure 8, shows the outcome of the proposed Montgomery Multiplication Design I-sim simulator. We can clearly verify the outcomes of the proposed design in Figure 8. Input is denoted by A(7:0) and B(7:0) andoutput is denoted by S(15:0).
Table III gives comparision of different method in terms of no of LUT's and power consumption our proposed method take less power as compare to previous methods its takes 6.53 uw but previous methods take more power as compare to the proposed method. We also compare in terms of no of LUT's means it take less area as compare to previous method it take less no offlip flops and registers.

Resource	Utilization	Available	Utilization %
LUT	64	63400	0.10 %
FF	24	126800	0.02%
IOB	35	210	16.67%
BUFG	1	32	3.13%

TABLE 3. Result of Compariosion Different Method

6. CONCLUSION

This research work presented a enhanced Montgomery and efficient implementation of Modular Multiplication. The method that is presented here makes the RSA algorithm more time efficient. The Montgomery Multiplication algorithm that has been presented has the benefit of being able to replace the division operation with the bit shift operation.
The implementation of Montgomery multiplication requires a trade-off between the amount of space on the chip and the amount of time it takes to perform the computation. The advantages of shifting at least significant bits of the partial product by setting it to zero outweigh the disadvantages of the earlier approach. The proposed approach produces satisfactory resultsin terms of the number of slice flip flops, LUTs, and IOBs. In terms of the amount of power used, the results obtained using the proposed method are superior to those obtained using other, more traditional approaches. The comparison is shown in the third column of the table above.

REFERENCES:

1. Pajuelo-Holguera, Francisco, José M. Granado-Criado, and Juan A. Gómez-Pulido. "Fast Montgomery Modular Multiplier using FPGAs." IEEE Embedded Systems Letters **100**, 9 (2021).
2. Zhao, Shilei, Hai Huang, Zhiwei Liu, Bin Yu, and Bo Yu. "An efficient signed digit montgomery modular multiplication algorithm." Microelectronics Journal **114**, 80-90 (2021).
3. Gu, Zhen, and Shuguo Li. "A Novel Method of Modular Multiplication Based on Karatsuba-like Multiplication." In 2020 IEEE 27th Symposium on Computer Arithmetic (ARITH),**50**, pp. 33-40 (2020).

4. Abd-Elkader, Ahmed AH, Mostafa Rashdan, El-Sayed AM Hasaneen, and Hesham FA Hamed. "Advanced implementation of montgomery modular multiplier." Microelectronics Journal **106 ,**pp.75-78(2020).
5. Reddy, Venkata, Simranjeet Singh, Vivian Desalphine, and David Selvakumar. "A Low Latency Montgomery Modular Exponentiation." Procedia Computer Science **171,**800-809 (2020).
6. Parihar, Aashish, and Sangeeta Nakhate. "High-speed high-throughput vlsi architecture for rsa montgomery modular multiplication with efficient format conversion.*"* Journal of The Institution of Engineers *(*India*)*: Series B **100**, no. 217-222 (2019).
7. Pajuelo-Holguera, Francisco, José M. Granado-Criado, and Juan A. Gómez-Pulido. "Fast Montgomery Modular Multiplier using FPGAs." IEEE Embedded Systems Letters (2021).
8. Zhao, Shilei, Hai Huang, Zhiwei Liu, Bin Yu, and Bo Yu. "An efficient signed digit montgomery modular multiplication algorithm." Microelectronics Journal **114** (2021).
9. Gu, Zhen, and Shuguo Li. "A Novel Method of Modular Multiplication Based on Karatsuba-like Multiplication.*"* In 2020 IEEE 27th Symposium on Computer Arithmetic (*ARITH*), pp. 33-40. (2020).
10. Abd-Elkader, Ahmed AH, Mostafa Rashdan, El-Sayed AM Hasaneen, and Hesham FA Hamed. "Advanced implementation of montgomery modular multiplier." Microelectronics Journal **106 ,** P.no. 80-90 (2020).
11. Reddy, Venkata, Simranjeet Singh, Vivian Desalphine, and David Selvakumar. "A Low Latency Montgomery Modular Exponentiation." Procedia Computer Science **171,**800-809. (2020):
12. Parihar, Aashish, and Sangeeta Nakhate. "High-speed high-throughput vlsi architecture for rsa montgomery modular multiplication with efficient format conversion." Journal of The Institution of Engineers (India*)* Series B **100,** no. 3 (2019).
13. Liu, Ruirui, and Shuguo Li. "A design and implementation of Montgomery modular multiplier." IEEE International Symposium on Circuits and Systems (ISCAS), **217-222,** pp. **1-4.** ,(2019)
14. Liu, R., & Li, S. (2019, May). A design and implementation of Montgomery modular multiplier. InIEEE International Symposium on Circuits and Systems *(ISCAS*) no.4 (**2019**).
15. Verma, Rupali, Maitreyee Dutta, and Renu Vig. "RSA Cryptosystem based on early word based montgomery modular multiplication." *In World Congress on Services*, pp. **33-47**. Springer, Cham, 2018
16. Gu, Zhen, and Shuguo Li. "A division-free toom–cook multiplication-based montgomery modular multiplication." IEEE Transactions on Circuits and Systems II: Express Briefs **66**, no. 8 (2018).
17. Yu Liting, Dongrong Zhang, Liang Wu, Shuguo Xie, Donglin Su, and Xiaoxiao Wang. "Aes design improvements towards information security considering scan attack." 12th IEEE International Conference On Big Data Science And Engineering *(TrustCom/BigDataSE*), **322-326**,no. 10 (2018).
18. El Mrabet, Nadia, Amine Mrabet, Ronan Lashermes, Jean-Baptiste Rigaud, Belgacem Bouallegue, Sihem Mesnager, and Mohsen Machhout. "A scalable and systolic architectures of montgomery modular multiplication for public key cryptosystems based on dsps." *Journal Hardware and Systems Security* 1, **219-236** no. 3 (2017).