International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

AREA EFFICIENT RNS-PPA MULTIPLIER DESIGN FOR HIGH-SPEED APPLICATION

Vidyasaraswathi. H.N[1], *Akarsha S.N*[2]

[1]Asst. Professor Dept. of ECE, BIT. Bengaluru, India
Vidyasaraswathi@bit-bangalore.edu.in
[2]M-tech student, Dept of VLSI &ES, BIT, Bangalore, India akarshasn@gmail.com

ABSTRACT:

The main purpose of this paper is to present the design and implementation of RNS (Residue Number System) based Area efficient high performance 8x8 multiplier. In this method conventional number is converted into its residues, this conversion of large bit numbers into a smaller number, i.e., Residue Number System (RNS) representations produce significant speedup for some classes of arithmetic-intensive algorithms in signal processing applications. Additionally, RNS arithmetic is a valuable tool for theoretical studies of the limits of fast arithmetic. These proposed methods also have some addition operation, by using convention adder will decreases the speed of operation and also increase the number of logic gates. So, to overcome those problems we are using Ladner Fischer parallel prefix adder to decrease the delay and area. Combination of these two algorithms producing a new architecture of a high speed and low implementation area in one multiplier. This fulfils the requirement of high-speed computer system applications and digital signal processing system nowadays. The algorithm was developed using Verilog HDL (Hardware Descriptive Language) in Xilinx 14.7 software and the design is implemented in FPGA Spartan 6 hardware to verify the results.

Keywords: RNS-Residue Number System, HDL-Hardware Descriptive Language, FPGA- Field Programable Gate Array, Computation- Intensive Arithmetic Functions (CIAF), Digital Signal Processing (DSP), Fast Fourier Transform (FFT).

1.INTRODUCTION

Multiplication is an important fundamental function in arithmetic operations. Multiplication-based operations such as Multiply and Accumulate (MAC) and inner product are among some of the frequently used Computation- Intensive Arithmetic Functions (CIAF)[1] currently implemented in many Digital Signal Processing (DSP)applications such as convolution, Fast Fourier Transform (FFT), filtering and in microprocessors in its arithmetic and logic unit. Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high-speed multiplier. Currently, multiplication time is still the dominant factor in determining the instruction cycle time of a DSP chip. The demand for high-speed processing has been increasing as a result of expanding computer and signal processing applications. Higher throughput arithmetic operations are important to achieve the desired performance in many real-time signal and image processing applications. One of the key arithmetic operations in such applications is multiplication and the development of fast multiplier circuit has been a subject of interest over decades. Reducing the time delay and power consumption are very essential requirements for many applications.

RNS is an arithmetic scheme where mathematical computations on large bit-width numbers is divided into smaller independent modular operations on multiple small modules. An RNS is defined by a chosen base of k relatively coprime moduli (r_1, r_2, ..., r_k)[2]. For each of these values, the RNS has an independent channel performing arithmetic over its corresponding channel modulus that contributes to the overall correct computation. Choosing the RNS moduli is also an important aspect of the hardware design for the individual channel arithmetic. Properly chosen values turn many of the modular operations into simplified logic.
This algorithm also involves some addition operation. Addition operation is the main operation in DSP (Digital Signal Process) Nowadays parallel prefix adders are most frequently used adders due to their fast computation properties. specifically, Ladner- Fischer parallel prefix adder is more efficient] [5] than the general adder because, in general adder each bit of addition is waited for the previous bit addition but in Ladner-Fisher adder, operation does not wait for previous bit addition operation and the modification is done at gate level so that it decreases the memory used.

MEDODOLOGY

RNS BASED MULTIPLIER

The process of RNS method-based multiplication is also caller as modular multiplication and the design of this multiplier mainly involves three steps as shown in the below figure 2.1.

1. Forward conversion.
2. Residue's multiplication or residues arithmetic computation
3. Reverse conversion.

1. Forward conversion: The process of encoding the input data into RNS representation is called forward conversion. During this process a conventional number is Divided by each moduli in the moduli set to find remainders. For example [2], the moduli set is (3|4|5) and the inputs (x, y) is equal to (4,5) respectively. Then the inputs x and y should be divided by three numbers (3|4|5) and the result is x1= (1|0|2), y1= (2|1|0), where x1, y1 are the RNS representation of (x, y).A residue numeral system (RNS) is defined by a set of *k* integers called the moduli, which are generally supposed to be pairwise coprime (that is, any two of them have a greatest common divisor equal to one). one of the stander method to choice moduli set is $\{2^{n-1}, 2^n, 2^{n+1}\}$[1]. it is a special moduli set representation relatively co-prime of standard RNS .

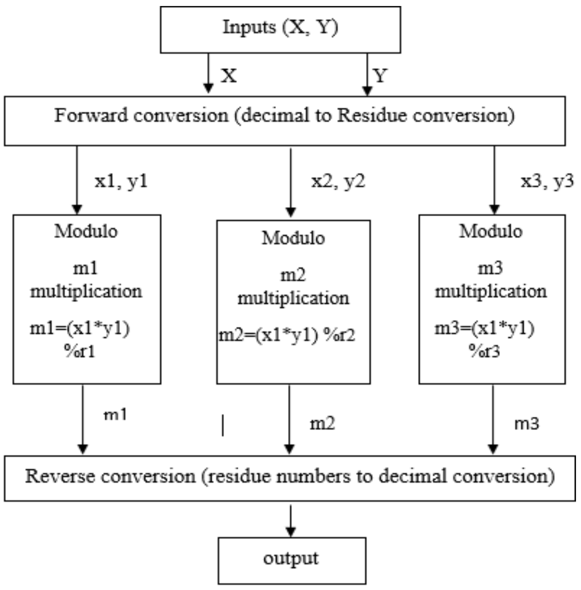
Inputs (X, Y)
X
Y
Forward conversion (decimal to Residue conversion)
x1, y1
x2, y2
x3, y3
Modulo m1 multiplication m1=(x1*y1) %r1
Modulo m2 multiplication m2=(x1*y1) %r2
Modulo m3 multiplication m3=(x1*y1) %r3
m1
m2
m3
Reverse conversion (residue numbers to decimal conversion)
output

Fig.2.1. General structure of RNS arithmetic operator

the number of coprimes that k integer numbers that can be represented as m_i. Notationally, we can write it as shown in equation (2).

$m_i=(x_i*y_i)\ \%\ r_i$ ....(2)

3. Reverse Conversion: The process of converting RNS representation to conventional number. One method for RNS-to- binary or conventional number conversion is to first derive the mixed-radix representation of the RNS number and then use the weights of the mixed-radix positions to complete the conversion. We can also derive position weights for the RNS directly based on the Chinese remainder theorem (CRT)[13].

Chinese remainder theorem (CRT): To perform the reversion conversion using CRT[14] method first we need to calculate the dynamic range of the system. The Dynamic range is the product of all the numbers in the moduli-set. And it is represented by M, mathematical we can represent it as show in equation (3).

$M=r_1*r_2*r_3 \ldots r_{k-1}$ ....(3).

After calculating the Dynamic range, divide it by moduli set that is divide M by each residue number r_i . mathematical we can represent it by using equation (4)

$\eta_i = M/r_i$ ....(4).

The process of conversion of RNS to binary is not only serial, but also requires multi-precision arithmetic. Chinese remainder theorem condonation can be formulated as show in the equation (5).

$(\eta_i \eta_i^{-1})$ %r =1 .... (5). i* i i

These moduli set has a unique advantage in which two or more

i

numbers do not have the same representation and these special moduli set shows better representational efficiency compared to that of other

k
Z= [∑ (ri*ηi*
i=0
η_i^{-1})]% M .... (6).

moduli set and also maintains a good balance between the different moduli in a given moduli-set. In a residue number system (RNS), a number x is represented by the list of its residues with respect to k pairwise relatively coprime moduli set $r_{k-1} > \cdots > r_1 > r_0$. The residue x_i of x with respect to the ith modulus r_i is a k in to a digit and the entire k-residue representation of x can be viewed as a k-digit number, where the digit set for the ith position is $[0, r_i - 1]$. Notationally, we can write it as shown in equation(1).

x_i = x mod r_i ....(1)

and specify the RNS representation of x by enclosing the list of residues, or digits, in parentheses.

2. Arithmetic computation: The process of performing arithmetic's like addition subtraction and multiplication and this process is done for all the moduli's concurrently. The total number of moduli's is equal to Where Z is the final output of multiplication of X and Y.

From equation (6) we can say that final process of conversion involves some addition operation. If we write code in Verilog by default addition operation will be takes place as Ripple carry adder method. we know that operation speed of Ripple carry adder [10] is slow compare to other adders.so increase the speed of operation multiplier we can Ladner Fischer parallel prefix adder.

PARALLEL PRE-FIX ADDER

These parallel prefix adders increase the pace at which a carry is propagated and in turn speed up the summation. Parallel prefix adder is a technique for increasing the speed in DSP processor while performing addition. The process of operation of addition in parallel prefix adder[19] employs the 3-stage structure as show

in the bellow figure 2.2

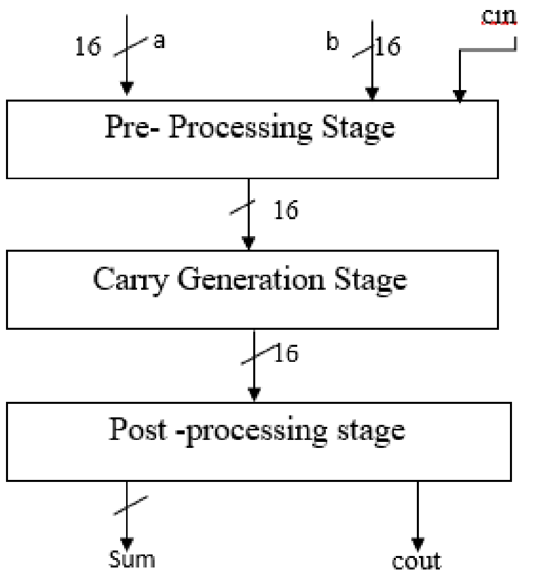

Figure 2.2. block diagram of parallel prefix adder

1. **Pre-Processing stage:** In pre-processing stage propagate and generate signal are computed by using Boolean expression as shown in the below equation (6) and (7). And this stage is also called as pre- fix stage because propagate and generate signals are prefixing in this stage for further carry generation.

$P[i] = a[i] \oplus b[i]$..(6)

$G[i] = a[i] \cdot b[i]$ ....(7)

2. **Carry Generation Stage:** This is the second stage of parallel pre-fix adder, In this stage carry propagate and carry generate signals are generated simultaneously for each bit by using Boolean expression as show in the below equation (9) and (8) respectively and it involve k number of stages or levels. For example, if we are performing 16 bit addition then k can calculated by 2^k=16 that equal to 4 similarly for 8 bit , k=3 and for 4bit k=2.

   $G_{i:j} = G_{i:k}$ OR ($P_{i:k}$ AND $G_{k-1:i}$)......................(8).

   $P_{i:j} = P_{i:k}$ AND $P_{k-1:i}$.. (9).

3. **Post-Processing Stage:** This is the final stage of an Ladner Fischer adder, the final sum is the XOR operation of the inputs with carry generated by the previous bit in the carry Generation stage. Sum is given by equation (10).

   Sum = a XOR b XOR C_{i-1} .... (10).

   $C_{i-1}=G_{i-1}$...(11).

Ladner Fischer parallel prefix adder:

The Ladner Fischer adder structure looks likes tree structure as shown in figure 2.4. for high performance of arithmetic operation .it requires less number of gates implement hence decreases the delay and memory used in this proposed architecture.

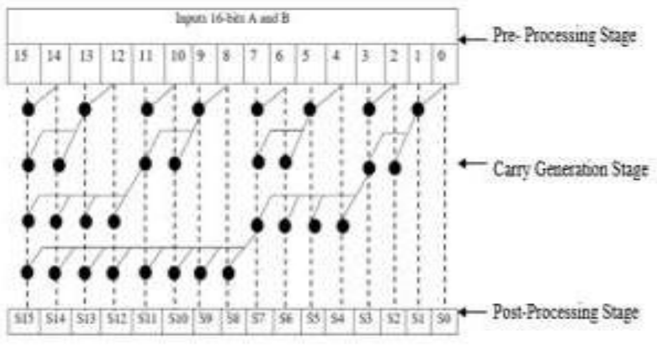

Figure 2.3.16bit Ladner Fischer adder

2.2. PROPOSED SYSTEM

The proposed system is the combination of RNS based multiplier and the Ladner-Fischer Parallel prefix adder. During resides inverse conversion the final output of the multiplier is the summation of the three residues so for that addition operation we require a 16-bit adder. In Verilog code the default addition operation will be carried out in Ripple Carry adder method, this Ripple Carry Adder requires a greater number of logic gates to design this will increases area and power. In Ripple carry Adder it will wait the previous stage carry out generation so the speed of design will be decreases. By using parallel prefix adder for addition, we can achieve area efficient and high-speed multiplier. Below flow chart or block diagram is the design of RNS based Multiplier using Parallel prefix adder is as shown in the figure 2.4.

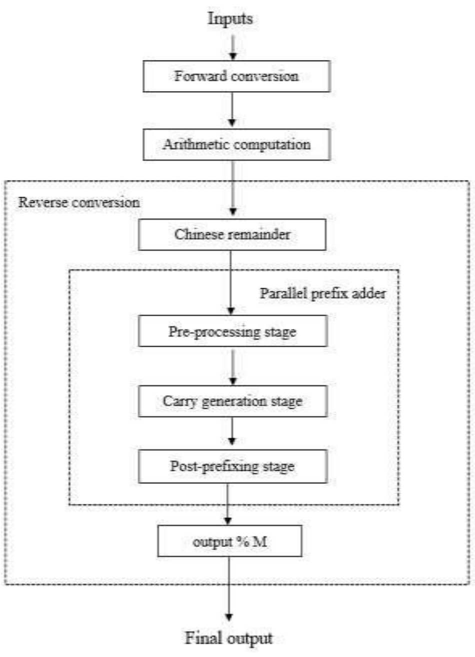

Figure 2.4. flow chart of proposed system.

RESULTS AND DISCUSSIONS 3.1.RNS BASED MULTIPLIER RESULTS

8X8bit multiplier based on RNS method is verified using Xilinx 14.7 and output is observed by simulation as shown in the figure 2.5. and compared the power and delay of another multiplier with RNS multiplier as shown in the table 1.

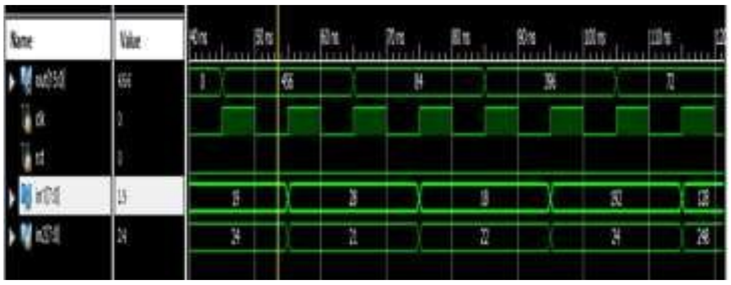
figure 2.5. Simulation result of RNS multiplier.

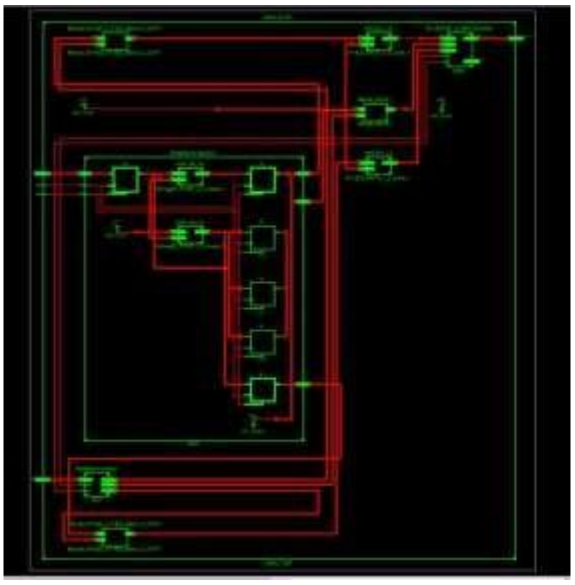
Figure.2.6. RNS based multiplier RTL schematic

Compare to RNS to modulo multiplier other multiplier had more delay and consume more power. And also, the multiplier like Array multiplier requires more area for implementation.

Table 1. comparison of 8x8 different multiplier

Sl. no	**Type of Multipliers(8-bit)**	**Power (mW)**	**Delay(ns)**
01	Booth Multiplier	27.36	7.58
02	Array Multiplier [15}	40	14.886
03	Booth encoded Wallace Tree [10]	18.716	7.04
04	Wallace Tree multiplier [10]	23.675	7.81
05	RNS modulo multiplier (proposed RNS Multiplier)	14.14	4.666

3.2. LADNER FISCHER ADDER RESULTS

16-bit three input parallel prefix adder using Ladner-Fischer structure is designed and verified by simulation as shown in the figure
2.7. and performed synthesis of the Ladner-Fischer adder using Xilinx 14.7., the RTL schematic of the adder is shown in the below figure.2.8. The proposed parallel prefix adder delay and power are compared with other conventional adder is as shown the below table 2.

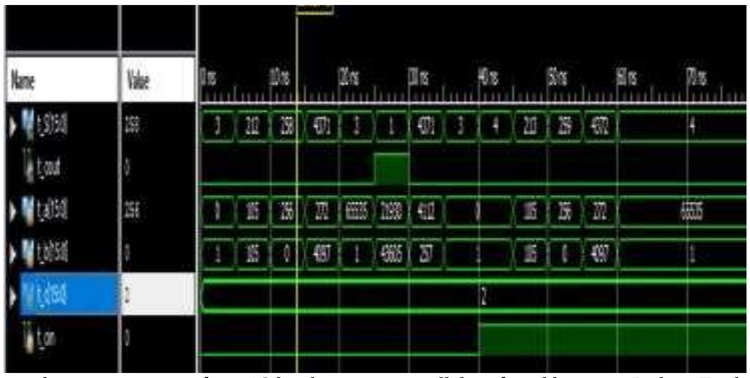

Figure.2.7. simulation output waveform 16-bit three input parallel prefix adder using Ladner-Fischer structure.

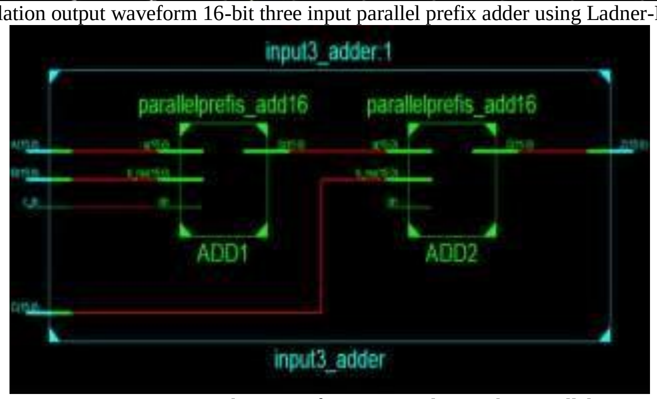

Figure 2.8. RTL Schematic of 3 input Lander-Fischer parallel

prefix adder

Table 2. Comparison of 16-bit different adder

Sl.no	Type of parallel prefix Adder (16 bit)	Delay(ns)	Power (W)	No. of logic LUTs
01	Kogge-stone Adder [19]	26.895	1.076	37
02	Bent kung adder [19]	27.31	0.897	32
03	Han-Carlson adder [19]	27.09	1.001	20
04	Lander fisher Adder [19]	21.416	0.01366	24
Other adders				
05	Ripple carry adder (16 bit) [10]	27.331	0.125	45
06	Carry save adder (16 bit) [10]	24.667	0.126	63

3.3. RESULTS OF RNS MULTIPLIER USING LANDER- FISHER PARALLEL PREFIX ADDER

8X8bit multiplier based on RNS method using 16-bit parallel prefix adder is verified using Xilinx 14.7 and output is observed by simulation. simulation results of RNS multiplier using Ladner Fischer parallel prefix adder as shown in figure 2.9. RTL schematic of top module and proposed system are shown in figure 2.11 and 2.10 respectively. Designs using the Register-Transfer Level specify the characteristics of a circuit by transfer of data between the registers, and also the functionality. An explicit clock is used. RTL design contains exact timing possibility; and data transfer is scheduled to occur at certain times Power ,Aelay and Area of RNS based Multiplier is compared with proposed system as shown in below table 3 table 4 respectively.

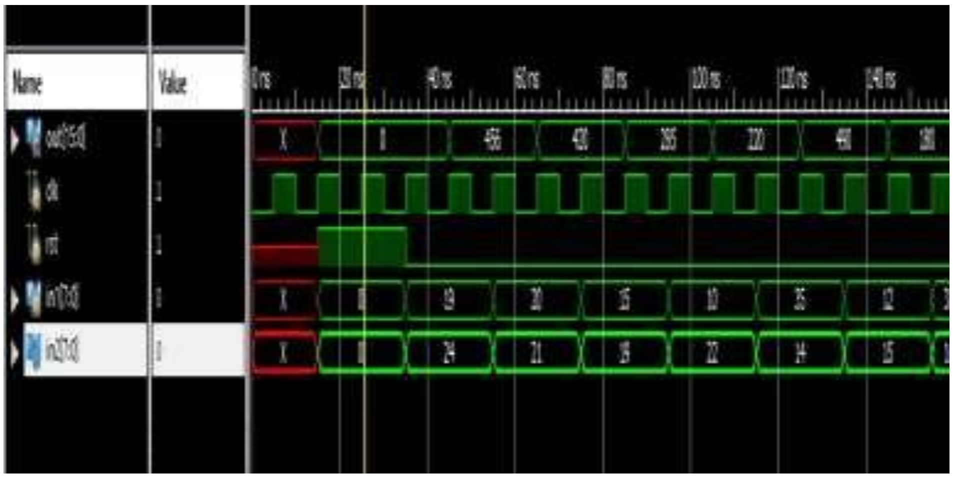

figure 2.9. Simulation result of RNS multiplier using Ladner Fischer parallel prefix adder.

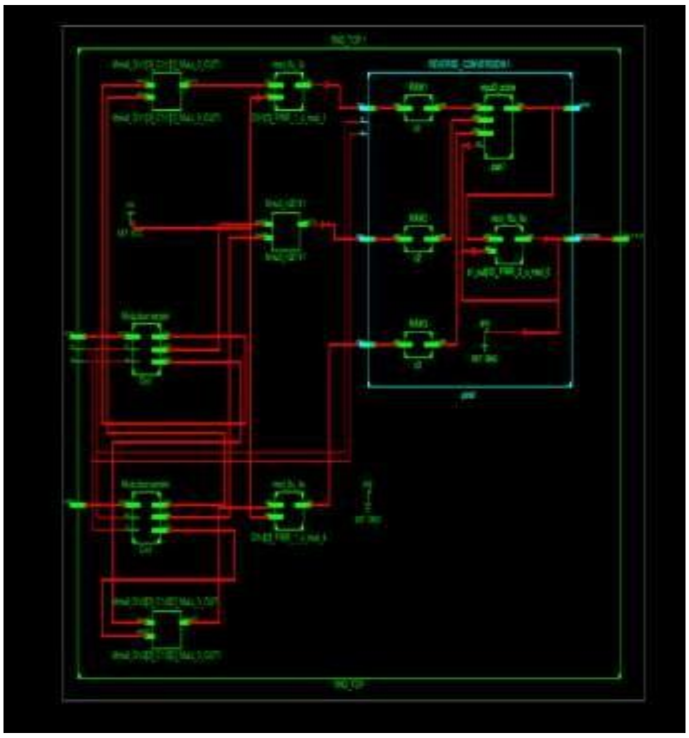

Figure 2.10. RTL Schematic RNS multiplier

Sl.no	Type of Multipliers(8-bit)	Comparison	
01	Normal RNS Multiplier	Number of Slice Registers	56
		Number of Slice LUTs	310
		Number of fully used LUT-FF pairs	41
		Number of bonded IOBs	34
02	RNS Multiplier using parallel prefix adder (Lander fisher adder)	Number of Slice Registers	54
		Number of Slice LUTs	229
		Number of fully used LUT-FF pairs	39
		Number of bonded IOBs	34

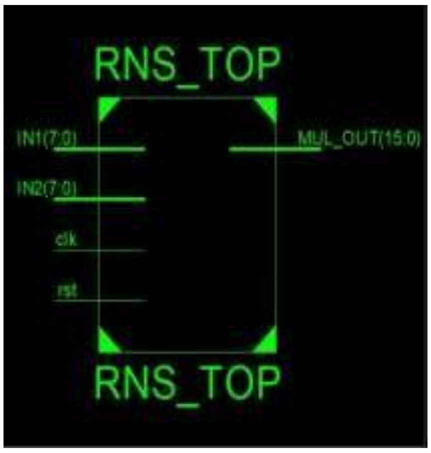

Figure 2.10. RTL Schematic RNS multiplier top module

Table.3. Comparison of power and delay RNS based multiplier and RNS based multiplier with parallel prefix adder.

Sl. no	Type of Multiplier(8-bit)	comparison	
		Power(mW)	Delay(ns)
01	Normal RNS based Multiplier	14.14	4.666ns
02	RNS Multiplier using parallel prefix adder (Lander fisher adder)	14.47	27.234

Table 4. Area utilization summary of RNS based Multiplier and RNS based Multiplier using parallel prefix Adder

Sl.no	Type of Multipliers(8-bit)	Comparison	
01	Normal RNS Multiplier	Number of Slice Registers	56
		Number of Slice LUTs	310
		Number of fully used LUT-FF pairs	41
		Number of bonded IOBs	34
02	RNS Multiplier using parallel prefix adder (Lander fisher adder)	Number of Slice Registers	54
		Number of Slice LUTs	229
		Number of fully used LUT-FF pairs	39
		Number of bonded IOBs	34

4.CONCLUSION

In this paper we design the area efficient parallel modulo based high speed RNS multiplier unit using LUT model for high performance. The performance metrics of proposed RNS based multiplier model is analysed for high performance DSP applications. The effectiveness of improved RNS coded is verified using exhaustive test bench unit. And finally, the complete trade of metrics of RNS encoded multiplier unit is validated using spartan6 FPGA hardware synthesis.

And the comparison of different multipliers and adder are done. From this comparison we are getting less delay and area efficient compare to other convention multipliers**.** Finally, we designed a 8 bit multiplier of area efficient with the high performance and verified using Verilog test bench code in Xilinx 14.7.

REFRENCESS

[1] Shang Ma , Shuai Hu , Zeguo Yang, Xuesi Wang, Meiqing Liu and Jianhao Hu High Precision Multiplier for RNS {2^{n-1}, 2^n, $^{2n+1}$}:Electronics 2021, 10, 1113.

[2] Schoinianakis, D. Residue arithmetic systems in cryptography: A survey on modern security applications. *J. Cryptogr. Eng.* 2020.

[3] Raj Kumar, Ram Awadh Mishra, Ritesh Kumar Jaiswal. Perspective and opportunities of Modulo Multipliers in Residue Number System. IEEE conference 2020.

[4] Elango Sekar, Sampath Palaniswami. Hardware Implementation of Residue Multipliers based Signed RNS Processor for Cryptosystems- 2020.

[5]A New Implementation of 16-bit Parallel Prefix Adder for High Speed and Low Area, Proceedings of the 2020 4th International Conference on Digital Signal Processing**.**

[6] Implementation of High-Performance Hierarchy-Based Parallel Signed Multiplier for Cryptosystems, IEEE conference 2020.

[7]Qing Liao and Shuguo Li. A New Implementation of 16-bit Parallel Prefix Adder for High Speed and Low Area. 2020

[8]S Elango , P Sampath, “Implementation of High Performance Hierarchy Based Parallel Signed Multiplier for Cryptosystems, J Circuit Syst. Comp., vol. 29, no. 13, pp. 2050214-1-2050214-25,
2020.

[9] Esmaeidout.M, Schinianakis.D., Javashi.H., Stourasitis.T, Navi.K: Efficient RNS implementation of elliptic curve poiunt multiplication over GF(pp).IEEE Trans very large scale integrator(VLSI)syst-2019.

[10] Hybrid variable latency carry skip adder in multiplier structures by Cristin.R.- researchgate-2019**.**

[11] Rohan Pinto and Kumara Shama. Low power Modified shift Add Multiplier Design using Parallel prefix Adder-March 7 2018.

[12] Shalini R.V and Dr.P.Sampath. Multiplier Design Incorporating Logarithmic Number System for Residue Number System in Binary Logic-2018

[13] Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme Ahmad Al Badawi, Yuriy Polyakov, Khin Mi Mi Aung, Bharadwaj Veeravalli, and Kurt Rohloff DECEMBER 5, 2018

[14] Data Flow Oriented Hardware Design of RNS-based Polynomial Multiplication for SHE Acceleration by Joël Cathébras1 , Alexandre Carbon1 , Peter Milder2 , Renaud Sirdey1 and Nicolas Ventroux- IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925,-2018

[15] Implementation of time efficient hybrid multiplier for FFT computation.-IJET-2018.

[16] Molahosseini, A.S., Zarandi, A.A.E., Martins, P., Sousa, L., “A multifunctional unit for designing efficient RNS-based datapaths,”
IEEE Access, vol.5, pp. 25972–25986,2017-2017

[17] Hiasat.A. sign detector for the extended four-moduli set {2^n- 1,2^n+1,2^{2n}+1,2^{n+k}}. *IET* comput.Digit.Tech-2017.

[18] Bajard J.C, Merkiche.N,:Doble level Montgomery cox-rower architechture, new bound. Springer, Hedelberg-2016.

[19] Performance Analysis of Parallel Prefix Adders Using Zynq- 7000 Soc Neeraj Kumar Cheryala, Nikhil Madhavaneni, Rajesh Odela-2016**.**

[20] Cheung, R.C.C., Duquesne.S, Fan.J, Guillermin.N, Verbauwhede.I.,Yao,G.X: FPGA implementation of pairing using residue number system and lazy reduction. Springer, Heidelberg- 2015.

[21] 16-bit velociously fault lenient parallel prefix adder- EEE *Xplore*: 16 April 2015.

[22] Bigou, K., Tisserand, A.: RNS modular multiplication through reduced base extensions. In: Proceedings 25th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP), IEEE, June 2014.

[23] Design of Efficient 16-Bit Parallel Prefix Ladner-Fischer Adder by S.Baba Fariddin ,E.Vargil Vijay-International Journal of Computer Applications (0975 – 8887) Volume 79 – No 16, October 2013.

[24] Gandino.F., Lamberti.F, Paravati.G, Bajard.J.C., Montuschi.P: An algorithm and architecture study on Montgomery exponential in RNS.IEEE Trans.Comput-2012