International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.02, February 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

# DESIGN OF HIGH SPEED FLOATING POINT MAC USING RESIDUE NUMBER SYSTEM

**D Avinash[1], R Sushma[2]**

[1]*PG Scholar Dept. of, ECE, SVCET, Srikakulam, AP, (India)*

[2]*Asst.Professor Dept. of, ECE, SVCET, Srikakulam, AP, (India)*

**ABSTRACT**

*This paper presents implementation of floating point multiplier using residue number system (RNS). A floating point multiplier has inputs in terms of mantissa and exponent. For multiplication of two inputs, mantissas are multiplied and exponents are needed to be added together. Residue is the remainder obtained after division of two integers. Operations in residue number system are performed on remainders, which are smaller integers. RNS system possesses properties of carry free computation and parallelism which leads to improvement in speed. Floating point RNS MAC uses modulo adder for exponent addition and modulo multiplier for mantissa multiplication where operations are performed on moduli. The design is coded in Verilog HDL using Xilinx 14.2 ISE software.*

***Keywords:Floating Point, MAC, RSN***

## I.INTRODUCTION

Floating-point arithmetic is widely used in many areas, especially in scientific computation, numerical processing and signal processing. Due to progression in VLSI technology nowadays FPGA's with high speed, more embedded modules and more number of logic are available. These make them suitable for implementing complex applications like floating point arithmetic. If the performance of floating point arithmetic in FPGA is improved, then FPGA is an attractive platform for scientific and real time applications. With this goal of flexibility in mind, the processor is designed in such a way that it can be configured to perform several useful functions. Since multiplication, subtraction and addition are three of the most commonly used arithmetic operations, these operations are included in the Floating Point Arithmetic and Logic Unit, both in integer and floating point mode. Along with this, logic operations on integers are also included in the proposed floating point processor. This processor has separate data memory and program memory, 32 number of 32 bit register file, 32 bit A and B registers, 32 bit program counter (PC) and 32 bit instruction register (IR). For the effective implementation of the arithmetic operations on floating point and integer numbers, a residue number system (RNS) based floating point ALU is proposed for the processor. The design is coded using verilog HDL and synthesized for Xilinx virtex-4 device. The design is synthesized using Xilinx ISE tool. Multiplier is an important block in digital applications such as digital signal processing, image processing, 3D graphics, microprocessor, filtering. Design of multiplier with less delay and less hardware is desirable. Floating point number system is a standard used in many DSP applications. This paper deals mainly with time optimization for floating point multiplication. Floating point numbers attempt to represent real numbers with uniform accuracy.

A generic way to represent a real number is in the form: R =a*bn , Where, 'n' is chosen so that 'a' falls within a defined range of values and called as exponent; 'b' is usually implicit in the data type. Design presented in this paper has input of 16 bit floating point representation (half precision) and the output of 32 bit floating point representation (single precision), notations are shown in figure below.

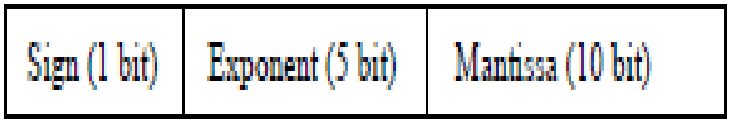


**Fig.1.** 16 bit half precision floating point number representation

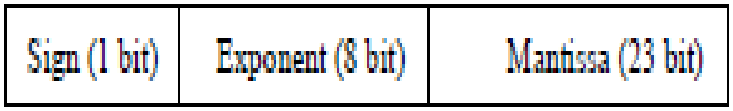


**Fig.2.** 32 bit half precision floating point number representation

## II. RELATED WORK

There are different types of processors. They are discrete processors, hard core processors and soft core processors. A discrete microprocessor is implemented as an ASIC with a specific peripheral set along with the processor core. A processor built from dedicated silicon is referred to as a hardcore processor. Such is the case for the ARM922T inside the Altera Excalibur family and the PowerPC 405 inside the Xilinx Virtex-II Pro and Virtex-4 families. And a soft core processor is built using the FPGA is general-purpose logic. The soft processor is typically described in a Hardware Description Language (HDL) or netlist. Processors can also be separated into two categories, fixed point and floating point. Fixed point processor has limited dynamic range compared to floating point processors. The first step in designing a processor is choosing an efficient instruction set architecture for our processor. Two architectures available are Reduced Instruction Set Computer (RISC)[1] and Complex Instruction Set Computer (CISC) architecture. A Floating point ALU is a processor or part of a processor that performs floating point calculations. When without a floating point unit, a CPU can handle both integer and floating point calculations. However, integer operations use significantly different logic than floating point operations, which makes it inefficient to use the same processor to handle both types of operations. An FPU provides a faster way to handle calculations with non-integer numbers. To design a Floating Point Unit we can follow different approaches like, Separate paths for multiplier and adder, merging common datapaths of multiplier and adders, use of effective algorithm for individual component design[2] or use multimode operations[5].

## III.RESIDUE NUMBER SYSTEM

Residue number systems are based on the *congruence* relation. Consider two integers $x$ and $y$,
these are said to be congruent modulo m, if m divides exactly the difference of $x$ and $y$; it is common, to write $x \equiv y \pmod{m}$ to denote this. Thus, for example, $10 \equiv 6 \pmod 2$; $10 \equiv 5 \pmod 5$; $17 \equiv 2 \pmod 3$ and $10 \equiv 4 \pmod 3$

etc[9]. The number *m* is called as *modulus* and assume that its value exclude unity, which produces only trivial congruence's. Residue is the remainder obtained after division of two integers. If q and r are the quotient and remainder respectively, of integer division of *a* by *m*, i.e. a = q*m + r. The number r is said to be the residue of *a* with respect to *m*. It is usually denoted by r=|a|m, where m is called as modulus [9].

*Representation of Decimal Number in Residue Number System*

Initially modulus set is chosen {m1, m2, m3…….. mN}, this set consists of N positive and pair wise relative prime moduli. Let M be the product of all moduli, Then every number X < M has a unique representation in RNS. Representation of numbers in a system in which the moduli are not pair wise relatively prime will not be unique i.e. two or more numbers will have the same representation. Decimal number is then divided by each modulus from moduli set and another set of remainders (residues) is obtained {r1, r2, r3…… rN}. Each decimal

number is represented uniquely by a single set of residue [9]. Moduli sets of the form {2n - 1, 2n, 2n+ 1} are most popular in use.

## IV.FLOATING POINT MULTIPLICATION ALGORITHM

Multiplying two numbers in floating point format is done by

1. Adding the exponent of the two numbers then subtracting the bias from their result.
2. Multiplying the significand of the two numbers
3. Calculating the sign by XORing the sign of the two numbers.

In order to represent the multiplication result as a normalized number there should be I in the MSB of the result (leading one).

The following steps are necessary to multiply two floating point numbers.

1. Multiplying the significand i.e. (I.MI * I.M2)
2. Placing the decimal point in the result
3. Adding the exponents i.e. (E I + E2 - Bias)
4. Obtaining the sign i.e. sl xor s2
5. Normalizing the result i.e. obtaining I at the MSB of the results "significand"
6. Rounding the result to fit in the available bits
7. Checking for underflow/overflow occurrence

## V. IMPLEMENTATION OF FLOATING POINT MUL TIPLTER

In this paper we implemented a double precision floating point multiplier with exceptions and rounding. Figure 3 shows the multiplier structure that includes exponents addition, significand multiplication, and sign calculation. Figure 3 shows the multiplier, exceptions and rounding that are independent and are done in parallel.

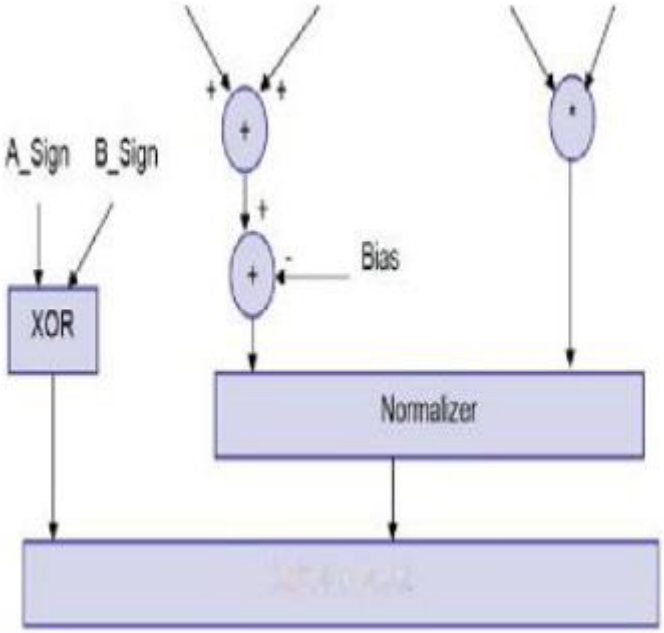


Fig 3. Multiplier Structure

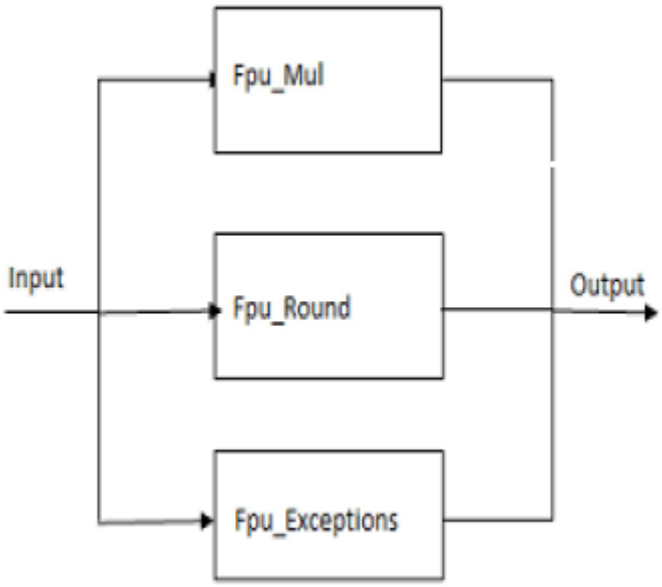


Figure 4. Multiplier structure with rounding and exceptions

## VI.FLOATING POINT RNS MULTIPLIER

In floating point multiplication mantissa of two inputs are multiplied together and exponents are added. The input to the proposed system is half precision (16 bit) and output is single precision (32 bit).

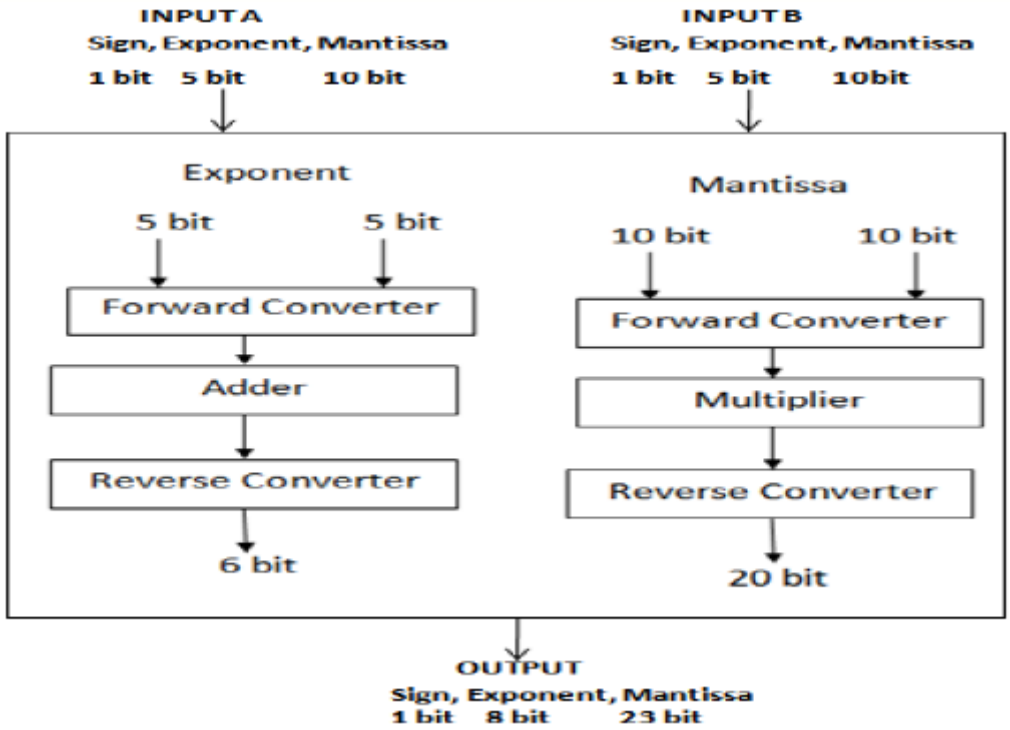


Fig 5. RNS multiplier unit.

In RNS multiplier, initially 10 bit mantissas of both the floating point inputs are converted into residue domain (RNS). This process is called as forward conversion. Special moduli set used is {2n - 1, 2n, 2n+ 1}. After Forward conversion, the set of residue (remainder) for each input A & B are obtained. Each set consist of three residues. Next step is multiplication of corresponding residues. This multiplication is modular i.e. result of multiplication of two corresponding residues is also a residue w.r.t. the same modulus (ex. 2). Modular multiplication gives another set of residue which converted into binary form by using reverse converter. Output of reverse converter is the final result obtained after multiplication of mantissa of given two inputs. Exponent of half precision floating point number is of 5 bits. Exponents of each input are also converted into RNS using forward converter. The obtained

residues are correspondingly added together using modulo adder. And the result of addition is

converted back into binary form using reverse converter.

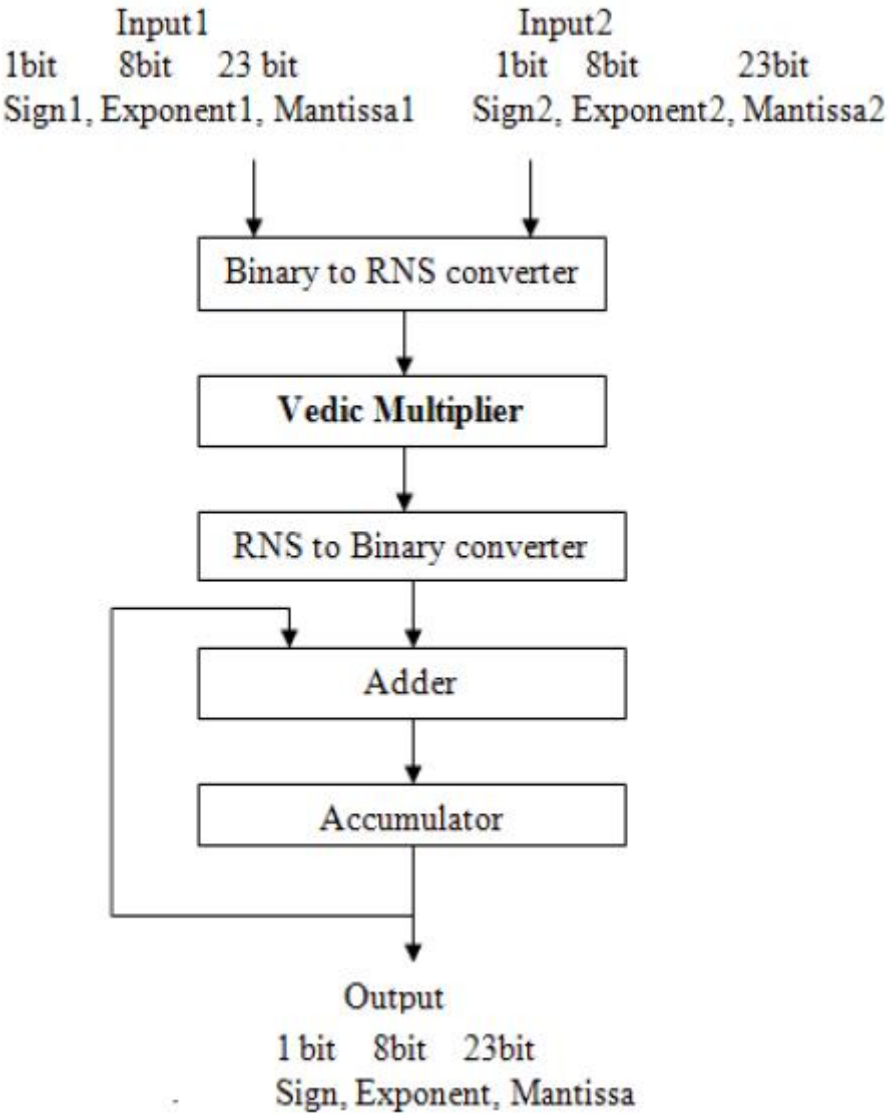


Fig 6. Block diagram of Floating Point RNS MAC using Vedic Multiplier.

The flow of operations for Floating point RNS MAC unit is as follows:

1. The Mantissa and biased Exponent is converted to Residue Number System. In RNS, based on the moduli, residues are obtained.

2. For multiplication, the Mantissa should be multiplied and Exponent should be added. For this, Mantissa modulo multiplier and Exponent modulo adder are used.

3. The results obtained are converted back into Binary numbers. 4. Using accumulator the products are added and saved.

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.02, February 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

## VII. RESULTS

Design is coded in Verilog HDL, synthesized using Xilinx 13.1. Simulation results for Floating point RNS multipli-er are shown below.

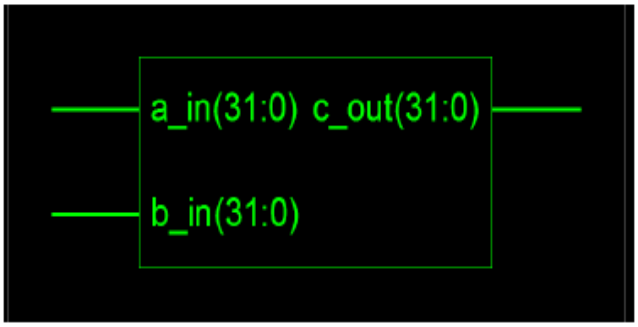


Fig 7. RTL Schematic

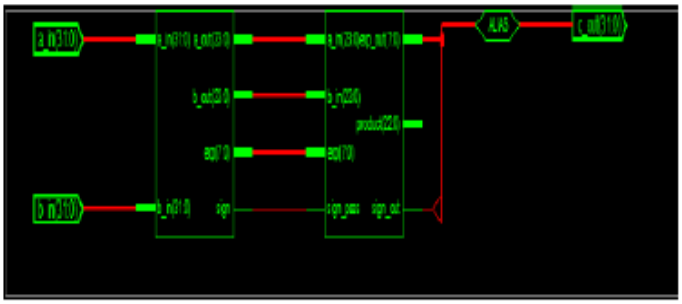

Fig 8. Internal RTL Schematic

The synthesis of proposed system is shown below,

### Synthesis Report:

```
*********************************************************************
Final Results
RTL Top Level Output File Name     : floating_point_multiplication.ngr
Top Level Output File Name         : floating_point_multiplication
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO

Design Statistics
# IOs                              : 96

Cell Usage :
# BELS                             : 415
#      GND                         : 1
#      LUT1                        : 6
#      LUT2                        : 74
#      LUT3                        : 36
#      LUT4                        : 82
#      MUXCY                       : 96
#      MUXF5                       : 24
#      VCC                         : 1
#      XORCY                       : 95
# IO Buffers                       : 96
#      IBUF                        : 64
#      OBUF                        : 32
# MULTs                            : 4
#      MULT18X18SIO                : 4
```

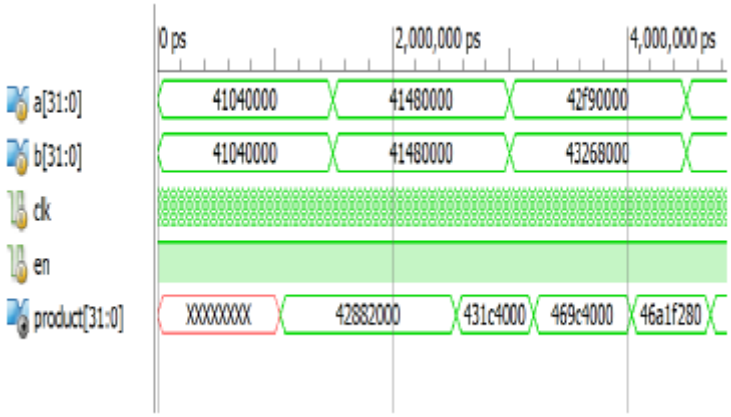


Figure 9. Simulation results for Floating point RNS multiplier.

Device Utilization Summary And Timing Details

| LOGIC UTILIZATION (VIRTEX6-XC6VLX240T) | MULTIPLIER IN [1] | MULTIPLIER IN [9] | PROPOSED MULTIPLIER |
|---|---|---|---|
| DELAY | 4.087 | 2.700 | 2.232 |
| NUMBER OF SLICE LUTS | 634 | 497 | 563 |

## VIII.CONCLUSION

This paper deals with development of an efficient Floating Point Processor. In this project a Single precision Floating Point Processor with Residue Number System based ALU is designed. The ALU is capable of performing both floating point and integer operations. A floating point RNS based MAC using Vedic multiplier is designed using Verilog HDL and synthesized using Xilinx ISE 14.2. By using Residue Number System parallel and carry free arithmetic can be obtained.

## REFERENCES

[1.] Dhanabal R, Barathi V, Sarat Kumar Sahoo, "Implementation of floating point MAC using residue number system", In Proceedings of the International Conference on Reliability, Optimization and Information Technology - ICROIT 2014, India, Feb 6-8 2014.

[2.] Azadeh Alsadat Emrani Zarandi, Amir Sabbagh Molahosseini, Mehdi Hosseinzadeh, Saeid Sorouri, Samuel Antao, and Leonel Sou, "Reverse converter design via parallel-prefix adders: Novel components, methodology and implementations", In Proceedings of the IEEE transaction on VLSI systems, 1063-8210.

[3.] Laurent-Stephane Didier & Luc Jaulmes, "Fast modulo 2n − and 2n + 1 adder using carry-chain on FPGA", In Proceedings of the IEEE 2013, Asilomar, 978-1-4799-2390- 8/13.

[4.] M. Dugdale,, "VLSI implementation of residue adders based on binary adders, "Circuits and SystemsII: Analog and Digital

[5.] B. Lee and N. Burgess, "Parameterisable Floating-point Operations on FPG A," Conference Record of the ThirtySixth Asilomar Conference on Signals, Systems, and Computers, 2002.

[6.] Xilinx13.4, Synthesis and Simulation Design Guide", UG626 (v13.4) January 19, 2012. [7] N. Shirazi, A. Walters, and P. Athanas, "Quantitative Analysis of Floating Point Arithmetic on FPGA Based Custom Computing Machines," Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM'95), pp.155-162, 1995.

[7.] L. Louca, T. A. Cook, and W. H. Johnson, "Implementation of IEEE Single Precision Floating Point Addition and Multiplication on FPGAs," Proceedings of 83 the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM'96), pp. 107-116, 1996.

D Avinash pursuing his M.Tech in the department of Electronics and Communication Engineering (VLSI), Sri Venkateswara College of Engineering & Technology, Etcherla, Srikakulam, A.P., India. Affiliated to Jawaharlal Nehru Technological University, Kakinada. Approved by AICTE, NEW DELHI. He obtained his B.Tech(ECE) from Sri Venkateswara College of Engineering & Technology, Srikakulam

R Sushma working as Assistant Professor, in the Department of Electronics and Communication Engineering(VLSI), Sri Venkateswara College of Engineering & Technology, Etcherla, Srikakulam. She obtained her M.Tech from Avanthi College of Engineering, Narsipatnam.