IMPLEMENTATION OF FLOATING POINT MAC USING RESIDUE NUMBER SYSTEM

[1]DHANABAL R, [2] SARAT KUMAR SAHOO,[3]BARATHI V,[4] NAAMATHEERTHAM R SAMHITHA, [5] NEETHU ACHA CHERIAN, [6]PRETTY MARIAM JACOB

[1]Assistant Professor (Senior Grade), School of Electronics Engineering,
[2] Associate Professor, School of Electrical Engineering,
[3]Assistant Professor, GGR College of Engineering, Vellore
[4,5,6] MTECH Students, VLSI Division, SENSE Department,
VIT University ,Vellore- 632014, TN, INDIA

E-mail: [1]rdhanabal@vit.ac.in , [2]sksahoo@vit.ac.in ,[3]samhitha.nr@gmail.com,
[4]neethuachacherian@gmail.com, [5]prettymjacob@yahoo.co.in

ABSTRACT

This paper presents the design and implementation of 16-bit floating point RNS Multiply and Accumulate (MAC) unit. Residue Number System (RNS) gained popularity in the implementation of fast arithmetic and fault-tolerant computing applications. Its attractive properties such as parallelism and carry free computation have speed up the arithmetic computations. Floating Point can be represented as $M \times B^E$ where M is Mantissa, E is the Exponent and B is the Base. The MAC unit consists of three units - Floating-point multiplier, Conversion unit and an Accumulator. The floating-point multiplier makes use of Brickell's Algorithm, the conversion unit makes use of a parallel conversion for the forward conversion and the Chinese Remainder Theorem for reverse conversion and the accumulator includes an adder unit which can make use of any of the conventional adders that depends on the moduli of the RNS being used. The input takes form of half-precision format where there is 1-bit for sign, 5-bits for exponent and 10-bits for mantissa. The design is coded in Verilog HDL and the synthesis is done using Cadence RTL Compiler.
Keywords: *MAC, Residue Number System (RNS), Floating point, Moduli*

1. INTRODUCTION

As a result of the rapid advance in communication and multimedia systems, the signal processing techniques are highly in demand. The main components used in Digital signal processor (DSP) are multiplier, adder and multiplier and accumulator (MAC) unit. MAC is used extensively in many applications like convolution and filtering. A basic MAC architecture consists of a multiplier and accumulator. The products generated by the multiplier are added and stored in accumulator. MAC is the main component in many of the digital signal processing applications. Hence the performance of MAC unit plays an important role in the design of filters which are used in DSP applications. The performance of MAC can be increased by the optimized design of multiplier and adder. Residue number system gained popularity because of the parallel processing and carry free arithmetic. A large bit number can be represented in form of small bit residues and the residues can be processed in parallel and thus the performance of the multiplier or adder can be increased. That is because there is no need of communicating carry information between two residues [6]. A high speed MAC capable for handling large range numbers with better precision will be required for many of the DSP application. There are two types of arithmetic operations. They are fixed and floating point operations. Fixed point number was inefficient for big number arithmetic. So floating point arithmetic was invented. Real numbers can be represented as a floating point number with two parts, mantissa and exponent. A floating point number is represented as $M \times B^E$ where M is mantissa, B is base and E is exponent. The floating point representation used here is a half precision. In this design the input is in 16 bit floating point representation (half precision) and the output is in 32 bit floating point representation (single precision).

The entire paper is divided into five sections. In first section introduction to MAC, floating point and RNS is given. In second section Residue Number System is discussed. Third section provides the information about the floating point RNS MAC unit along with the Binary to RNS Converter and RNS to Binary Converter. Fourth and fifth section explains about results and conclusions that are obtained respectively.

I. Residue Number System

In recent times Residue Number System is becoming popular because of its carry free addition and multiplication capabilities. Since there is no carry propagation between arithmetic blocks, high speed processing can be obtained. RNS representation encodes large numbers into small residues so that computation can be performed more efficiently. The arithmetic can be implemented in parallel for these residues. This ensures that there is no dependency between each modulo unit. So, the complexity of the arithmetic units in each modulo unit is reduced. Here we need some extra hardware for converting binary to residue and residue to binary.

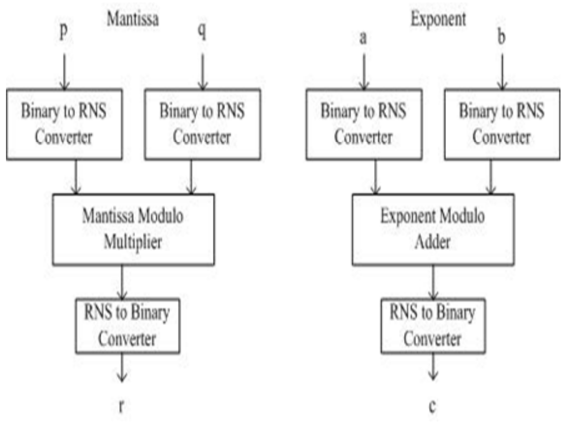
The RNS is defined by set of co-primes called Moduli. The Moduli set is denoted by $\{m_1, m_2, m_3 ... m_i\}$, where m_i is the ith modulus. Each integer X can be represented as set of small integers called residues. This residues are denoted by $\{r_1, r_2, r_3 ... r_i\}$, where r_i is the ith residue. Integer X is divided by each Moduli and corresponding remainder obtained is taken as residue [3]. Mathematically representation is given by Equation 1.

$$r_i = X \bmod m_i \qquad (1)$$

The MAC unit makes use the special Moduli set, $\{2^n - 1, 2^n, 2^n + 1\}$ that is taken to be a low-cost moduli set and improves the performance of the unit. For binary to RNS conversion the Moduli set we have chosen is $\{2^n - 1, 2^n, 2^n + 1\}$, where n is decided based on the number of bits of the input binary number. By making use of these Moduli a particular binary number can be converted into corresponding residues. This set of Moduli makes the forward conversion process fast and simple. In general, forward converters based on these Moduli set are the most efficient converters.

2.ARCHITECTURE OF FLOATING POINT RNS MULTIPLIER AND ACCUMULATOR UNIT

The modules that are required for implementing a Floating Point RNS MAC are Binary to RNS Converter, Modulo Adder, Modulo Multiplier, RNS to Binary Converter and Accumulator. The block diagram for the MAC unit is shown in Fig 1. Floating point multiplication involves multiplication of mantissa and addition of exponent. So, Floating Point in residue domain includes a RNS modulo



multiplier for mantissa and RNS modulo adder for exponent. The block diagram of Floating point RNS multiplier is shown in Fig. 4.

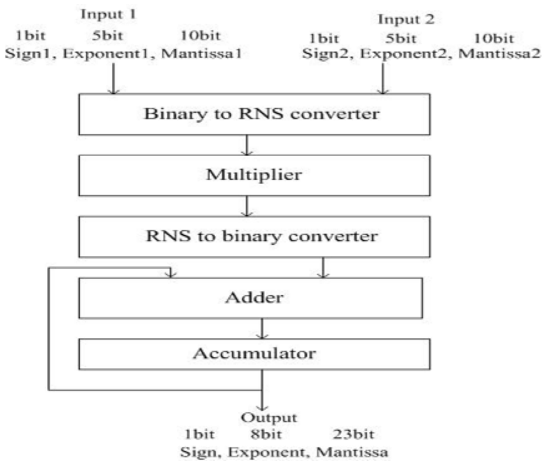


Fig. 1. Block Diagram Of Floating Point RNS MAC Unit

Basically floating point inputs are given as Mantissa and Exponent. The input to the MAC is in half-precision (16-bit) notation as shown in Fig. 2 and the output of MAC is in single precision (32-bit) notation as shown in Fig. 3. The sign bit indicates whether the number is positive (sign bit=0) or negative (sign bit=1). Exponent is biased exponent and the mantissa is in normalized form.

Sign (1-bit)	Exponent (5-bit)	Mantissa (10bit)

Fig. 2. 16-bit Floating Point Representation

Sign (1-bit)	Exponent (8-bit)	Mantissa (23-bit)

Fig. 3. 32-bit Floating Point Representation

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

1. The unbiased Exponent is converted to biased by adding 15 (this value depends on number of bits used to represent exponent). This biasing is done to ensure that the Exponent is unsigned.
2. The Mantissa and biased Exponent is converted to Residue Number System. In RNS, based on the moduli, residues are obtained.
3. For multiplication, the Mantissa should be multiplied and Exponent should be added. For this, an RNS Mantissa modulo multiplier and RNS Exponent modulo adder are used.
4. Using accumulator the products are added and saved.

A. *Floating Point RNS Multiplier*

Floating point multiplication involves multiplication of mantissa and addition of exponent. Floating Point in residue domain includes a RNS multiplier for mantissa and RNS modulo adder for exponent. There are different methods for multiplication in residue domain – look up tables, array of adders and combination of look up tables and adders. For large number of bits the delay and area of residue array multiplier is preferred [2]. The block diagram of Floating point RNS multiplier is shown in Fig. 4. For residue multiplication Brickell's algorithm is used.

Fig. 4. Block Diagram of Floating Point Multiplier

The Brickell's algorithm is as follows:

1. Initialize accumulator to 0. For $i=0$ to $N-1$ where N is no of bits of multiplier.
2. Double the contents of accumulator and add partial products to it.
3. Partial products are obtained by AND operation of Bi and A where A is multiplicand and Bi is ith bit of multiplier.
4. Check the contents of accumulator (acc), if

acc > m_i , then acc = acc - m_i

acc > 2m_i , then acc = acc - 2m_i

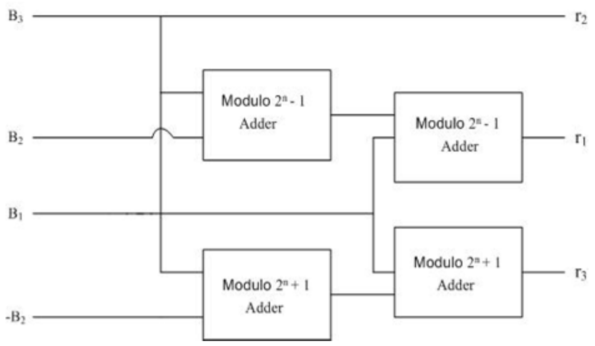
Where m_i is the modulus

The exponents are added in residue domain. The addition in residue domain is done by RNS modulo adder. The algorithm of RNS modulo adder is as follows. Let $c_i = a_i + b_i \bmod m_i$ where a_i and b_i are the residues and m_i is the modulus, then

$c_i = a_i + b_i$, if $a_i + b_i < m_i$

$c_i = a_i + b_i - m_i$, if $a_i + b_i >= m_i$

In order to perform modulo M addition the architecture in Fig. 6 is used [4], the adder structure which we have used here is a Parallel Prefix adder called Kogge-stone adder.



B. *Binary to RNS Converter*

The data available will be in the form of binary. In order to process in RNS, the binary data has to be converted into RNS. The process of converting binary data into RNS is referred to as the forward conversion. The forward converter must be area, power and speed efficient. After the data is being processed through modulo processing units of RNS, they must be converted back to their conventional representations. The process of converting back into the conventional representations is referred to as backward conversion.

The forward conversion can be for an arbitrary Moduli set or special Moduli set [3] [8], the special Moduli set used in this paper include $\{2^n-1, 2^n, 2^n+1\}$ where n is decided based on the number of bits of the input binary number. Since the exponent is of 5-bits, the Moduli set required for the exponent is {3, 4, 5} which enables to represent 0 to 2^5-1 in the RNS form. Similarly, as the mantissa is of 10bits, the Moduli set considered is {255, 256, 257} which gives the RNS representation for numbers from 0 to 2^{10}-1. These special Moduli set are considered for the RNS operation because they make the system fast and simple along with being efficient [3] [8]. In order to obtain the residue of an input binary number with respect to a certain Moduli, the distributive property of the addition in RNS is taken into consideration i.e.,

$$|X+Y|_m = || X|_m + |Y|_m|_m$$

(2)

Consider a binary number X given by $X = x_{n-1}x_{n-2}....x_1x_0$ which can be given as

$$X = \sum_{j=0}^{n-1} 2^j x_j$$

The modulus of the binary number using equation (1) gives,

$$|X| = |\sum_{j=0}^{n-1} 2^j x_j |_m = |\sum_{j=0}^{n-1} | 2^j x_j |_m|_m$$

Where x_j can be either 0 or 1.

This paper includes a parallel method of forward conversion of the input binary block. Binary to RNS converter architecture is given in Fig.5. The input binary number is divided into 3 blocks each of 'n' bits. Thus, if a given binary number X is divided into blocks, B_1, B_2, B_3, then we have,

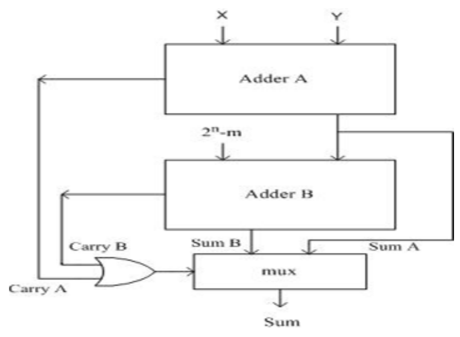
$$X = \sum_{j=0}^{n-1} 2^{jB} B_j$$

Hence, we can have,

$$|X|_m = |\sum_{j=0}^{n-1} 2^{jB} B_j |_m = |\sum_{j=0}^{n-1} | 2^{jB} B_j |_m|_m$$

Fig. 5.Binary to RNS Converter for moduli set $\{2^n - 1, 2^n, 2^n + 1\}$

Due to the usage of a special moduli set, $\{2^n - 1, 2^n, 2^n + 1\}$, the number to be converted to RNS is divided into three blocks each of 'n' bits. The three blocks B_1, B_2, B_3 are represented as given below.



$$B_1 = \sum_{j=2n}^{3n-1} 2^{j-2n} x_j$$

$$B_2 = \sum_{j=n}^{2n-1} 2^{j-n} x_j$$

$$B_3 = \sum_{j=0}^{n-1} 2^j x_j$$

Thus, the input binary number, X can be given in terms of the blocks as,

$$X = B_1 2^{2n} + B_2 2^n + B_3$$

Then the residues can be obtained as follows,

$$r_1 = | B_1 + B_2 + B_3 |_{2^n - 1}$$

$$r_2 = B_3$$

$$r_3 = | B_1 - B_2 + B_3 |_{2^n + 1}$$

The overall architecture of Binary to RNS converter architecture is given in Fig. 5. In this for calculating residue r_1, blocks B_2 and B_3 are first added using modulo $2^n - 1$ adder and then the result is added with the block B_1 using same modulo $2^n - 1$ adder, where as for calculating residue r_3, block B_3 is added with the 2's compliment of block B_2 using modulo $2^n + 1$ adder and then the result is added with the block B_1 using same modulo $2^n + 1$ adder. The residue r_2, is just same as block B_3 of the input binary number X [8].

C. *Modulo-M Adder*

The modulo-m adder forms the basic arithmetic unit for any RNS operation or RNS conversion. The basic modulo-m adder architecture is given in Fig. 6.

According to it, the modulo-m addition is done as follows

$$|X+Y|_m = \begin{cases} X+Y : X+Y < m \\ X+Y-m : X+Y \geq m \end{cases}$$

Fig. 6. Modulo-M adder architecture [4]

The adder structure can be any conventional adder that can be used like a ripple carry adder (RCA), a carry look ahead adder or any parallel prefix tree. Here we make use of a Kogge-stone adder as shown in Fig. 7.and Fig. 8. Since we are providing mantissa and exponent binary inputs separately, they should be converted separately using the moduli mentioned for them respectively. When a 5-bit binary exponent is converted into RNS using moduli set {3, 4, 5} then the residues each of 3-bit is obtained. So we are considering a 3-bit Kogge-Stone adder in Fig.7. Similarly when a 10-bit binary exponent is converted into RNS then the residues each of 9-bit is obtained. So for this we are considering 9-bit Kogge-Stone adder in Fig. 8.

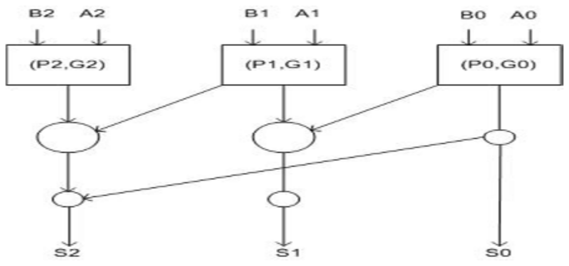


Fig. 7. 3-bit Kogge-stone adder

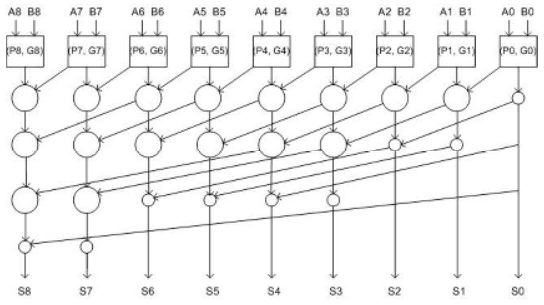


Fig. 8. 9-bit Kogge-stone adder

D. RNS to Binary Converter

RNS to binary conversion is the bottle neck in using RNS. RNS to Binary conversion is based on two algorithms. They are mixed radix conversion (MRC) and Chinese remainder theorem (CRT) [4]. MRC is sequential whereas CRT is parallel. The design of modulo M adder in the final stage of CRT is design bottleneck of it. In this paper CRT algorithm is used for conversion of RNS to binary.

The mathematical equations for CRT are as follows. Given a set of moduli $\{m_1, m_2, m_3 ... m_i\}$ and the residues are $\{r_1, r_2, r_3 ... r_i\}$, then binary number X is given as

$$X = \left|\sum_{i=1}^{n} r_i M_i^{-1} M_i\right|_M$$

Suppose we have three moduli set $\{m_1, m_2, m_3\}$ then,

$$M_1 = \frac{m_1 * m_2 * m_3}{m_1},\ M_2 = \frac{m_1 * m_2 * m_3}{m_2},$$

$$M_3 = \frac{m_1 * m_2 * m_3}{m_3}$$

Now M_i^{-1} can be obtained from following equations

$$|M_1^{-1} * M_1| = 1,\ |M_2^{-1} * M_2| = 1,\ |M_3^{-1} * M_3| = 1,$$

Where $M = m_1 * m_2 * m_3$

So CRT requires three main processes.

1) Calculating M_i and inverses $|M_i^{-1}|_{m_i}$.

2) Multiplying M_i and inverses with residues and accumulating it.

3) Modulo M adder as the final stage.

II. Results

The design is developed using Verilog HDL, simulated using Altera ModelSim and synthesized using Cadence RTL Compiler. The simulation results of 16 bit floating point RNS fig 1 MAC unit are shown in Fig. 9. The MAC performs signed

Fig. 9. Simulation Results for 16-bit Floating Point RNS MAC Unit operations. The simulation result shows the output for both positive and negative floating point numbers. The synthesis results are tabulated in Table I. The speed of floating point MAC in [9] was found to be 500MHz. While comparing [9] with the proposed MAC unit, the time for MAC is 1.6ns and thus the speed is 800.11MHz. So an increase of 25% is obtained. This is being shown in Fig 10.

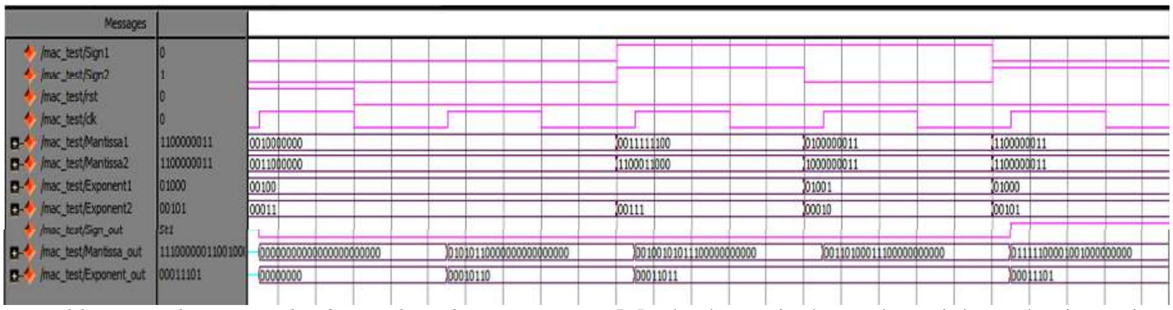

Messages
/mac_test/Sign1 0
/mac_test/Sign2 1
/mac_test/rst 0
/mac_test/clk 0
/mac_test/Mantissa1 1100000011
/mac_test/Mantissa2 1100000011
/mac_test/Exponent1 01000
/mac_test/Exponent2 00101
/mac_test/Sign_out St1
/mac_test/Mantissa_out 1110000001100100
/mac_test/Exponent_out 00011101

Table I. Synthesis Results for 16-bit Floating Point RNS MAC Unit

Power (mW)	Timing (ns)	Area (µm^2)
6.87	1.6	83873.44

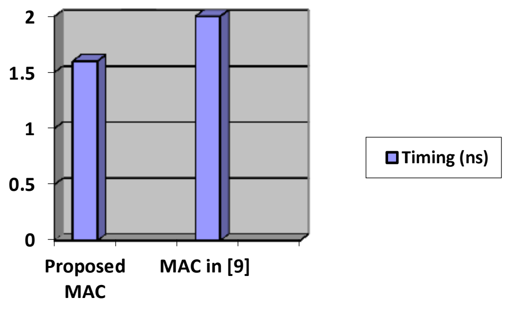


Fig 10. Speed Comparison of Proposed with [9]

6. CONCLUSION

The 16 bit MAC unit is designed in Verilog HDL and implemented in Altera ModelSim and synthesized in Cadence TSMC 45nm technology. By using Residue Number System parallel and carry free arithmetic's can be obtained. The large number is represented in the form of residues. So addition and multiplication is performed parallely on all the residues. Thus arithmetic is performed in a faster rate. Thus 16-bit Floating Point RNS MAC unit is found to be of efficient.

REFRENCES:

[1] Behrooz Parhami, "Computer arithmetic-Algorithms and hardware designs", Oxford University Press, 2000.

[2] C.-L. Chiang and L. Johnson, "Residue arithmetic and VLSI," IEEE International Conference on Computer Design: VLSI in computers, 1983.

[3] Ghosh, S. Singha, and A. Sinha, "Floating point RNS: a new concept for designing the MAC unit of digital signal processor," SIGARCH Comput. Archit. News, vol. 40, no. 2, pp. 39–43, May 2012.

[4] M. Dugdale, "VLSI implementation of residue adders based on binary adders," Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, vol. 39, no. 5, pp. 325–329, 1992.

[5] Taylor.F, "Residue arithmetic a tutorial with examples," *Computer*, vol. 17, no. 5, pp. 50–62, 1984.

[6] Kinoshita, H. Kosako, and Y. Kojima, "Floating-point arithmetic algorithms in the symmetric residue number system," Computers, IEEE Transactions on, vol. C-23, no. 1, pp. 9–20, 1974.

[7] R. Amos Omondi, Benjamin , "Residue Number Systems: Theory and Implementation," Imperial College Press, 2007.

[8] Strenski, D., Cappello, J.D.; "A practical measure of FPGA floating point acceleration for High Performance Computing," Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on , vol., no., pp.160,167, 5-7 June 2013

[9] Dhanabal R,,Bharathi V,Saira Salim, Bincy Thomas, Hyma Soman, Dr Sarat Kumar Sahoo "DESIGN OF 16-BIT LOW POWER ALU - DBGPU " ,International Journal of Engineering and Technology (IJET) 2013.

[10] R Dhanabal,V Bharathi, Anand N, George Joseph, Suwin Sam Oommen, Dr Sarat Kumar Sahoo ,"Comparison of Existing Multipliers and Proposal of a New Design for Optimized Performance " ,International Journal of Engineering and Technology (IJET) 2013.

[11] R Dhanabal, Ushashree, "Implementation of a High Speed Single Precision Floating Point Unit using Verilog" ,International Journal of Computer Applications (0975 – 8887),2013.