Open access Journal International Journal of Emerging Trends in Science and Technology

Design of Floating Point Multiplier Using Residue Number System with Moduli Set $\{2^n-1, 2^n, 2^n+1\}$

Authors
Praveen Amrutkar[1], Prasanna Palsodkar[2]
[1]PG Scholar, Yeshwantrao Chavan College of Engineering,
RTMNU University, Hingna Road, Nagpur, India
Email: *praveenamrutkar@gmail.com*
[2]Asst. Prof, Yeshwantrao Chavan College of Engineering,
RTMNU University, Hingna Road, Nagpur, India
Email: *prasanna.palsodkar@gmail.com*

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. RNS multiplier unit consist of forward converter, modulo multiplier, modulo adder and reverse converter. A moduli set of the form $\{2^n - 1, 2^n, 2^n + 1\}$ is used to find residues of input integers. Input to the system is half precision floating point numbers i.e. sign (1bit), mantissa (10bit), exponent (5bit) and output is a single precision number, sign (1bit), mantissa (23bit), exponent (8bit). The design is coded in Verilog HDL using Xilinx 13.1 ISE software.

Keywords: *Residue Number System (RNS), floating point, moduli, modulo adder, modulo multiplier, forward converter, reverse converter*

1. Introduction

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: $R = a*b^n$, 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.

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

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

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

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

2. 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 $\{m_1, m_2, m_3 \ldots\ldots m_N\}$, 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 $\{r_1, r_2, r_3 \ldots\ldots r_N\}$. Each decimal number is represented uniquely by a single set of residue [9]. Moduli sets of the form $\{2^n - 1, 2^n, 2^n + 1\}$ are most popular in use.

Example. Consider n=2 in above moduli set→$\{m_1, m_2, m_3\} = \{3, 4, 5\}$.

It represents max number of integers→$m_1*m_2*m_3 = 3*4*5=60$(i.e. from 0 to 59).

a is represented as →$\{r_1 = rem(\boldsymbol{a}/m_1), r_2 = rem(\boldsymbol{a}/m_2), r_3 = rem(\boldsymbol{a}/m_3)\}$

$12 = \{rem(12/3), rem(12/4), rem(12/5)\} = \{0,0,2\}$.

Where *rem* is remainder.

3. 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).

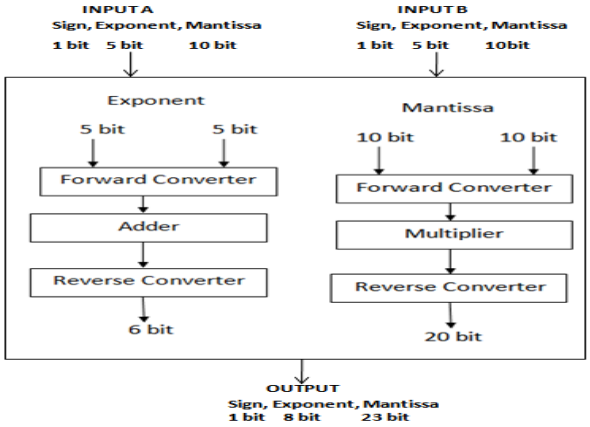


Figure 3: RNS multiplier unit.

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 $\{2^n - 1, 2^n, 2^n + 1\}$. 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.

Ex.2. inputs: - x=7, y=8, moduli set:- m={3,4,5}

Residue set for x = {1, 3, 2}.

Residue set for y = {2, 0, 3}.

Multiplication

Multiply res. set of x and y = {1*2, 3*0, 2*3} = {2, 0, 6}.

To obtain the set of modulo multiplication, find residue of 2 w.r.t 3, 0 w.r.t. 4, 6 w.r.t. 5.

Modulo multiplication = {2, 0, 1}.
Now perform reverse conversion. 7*8= {2, 0, 1} = 56.

Addition
Add res. set of x and y = {3, 3, 5}.
Modulo addition = {|3|3, |3|4, |5|5} = {0, 3, 0}.
Hence 7+8 = {0, 3, 0} =15.

3.1 Forward Converter

Forward converter is a binary to RNS converter. The moduli set used is $\{2^n - 1, 2^n, 2^n+ 1\}$. These special moduli are usually referred to as low-cost moduli, since conversion to and from their residues can be realized relatively easier and do not require complex operations. Here $m_1= 2^n - 1$, $m_2=2^n$, $m_3=2^n +1$. Then any integer X within dynamic range, $M= [0, 2^{3n} -2^n - 1]$ is uniquely defined by residue set $\{r_1, r_2, r_3\}$, where ri= |X|mi. For mantissa multiplication n=8, i.e. m= {255, 256, 257} and for exponent addition n=3, i.e. m= {7, 8, 9} is used. To find r_1, r_2 &r_3, first divide the given number in three n bit parts B_1, B_2, B_3 [1].

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

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

$$B_3 = \sum_{j=0}^{n-1} x_j 2^{j} \quad (3)$$

$$X = B_1 2^{2n} + B_2 2^n + B3 \quad (4)$$

Properties of addition and multiplication in RNS system

$$\|X+Y\|_m = \|X|_m + |Y|_m|_m$$
$$\|X*Y\|_m = \|X|_m * |Y|_m|_m$$

now,

$$r_3 = |X|_{2^n+1}$$
$$= |B_1 2^{2n} + B_2 2^n + B_3|_{2^n+1}$$
$$= \left\| |B_1 2^{2n}|_{2^n+1} + |B_2 2^n|_{2^n+1} + |B_3|_{2^n+1} \right|_{2^n+1}$$

$$|2^n|_{2^n-1} = |2^n -1+1|_{2^n-1}$$
$$= \| 2^n -1|_{2^n-1} * |1|_{2^n-1}|_{2^n-1}$$
$$= |1*1|_{2^n-1} = 1$$

$$|2^n|_{2^n+1} = |2^n -1+1|_{2^n+1}$$
$$= \| 2^n +1|_{2^n+1} * |-1|_{2^n+1}|_{2^n+1}$$
$$= |1*-1|_{2^n-1}$$
$$= -1$$

$$|2^{2n}|_{2^n-1} = \| 2^n * 2^n|_{2^n-1}|_{2^n-1}$$
$$= \| 2^n -1+1|_{2^n-1} * |2^n -1+1|_{2^n-1}|_{2^n-1}$$
$$= |1 *1|_{2^n-1} = 1$$

$$| 2^{2n}|_{2^n+1} = | 2^n * 2^n|_{2^n+1}$$
$$= \| 2^n +1-1|_{2^n+1} * | 2^n +1-1|_{2^n-1}|_{2^n-1}$$
$$= |-1 *-1|_{2^n-1} = 1$$

$$| 2^{2n}|_{2}^{n} = | 2^n * 2^n|_{2^n}$$
$$= |0 * 0|_{2^n}$$
$$= 0$$

$$|2^n|_{2^n} = |0|_{2^n}$$
$$= 0$$

Now, we can write

$$r_1 = |B_1 + B_2 + B_3|_{2^n-1}^{n} \quad (5)$$

$$r_2 = B_3 \quad (6)$$

$$r_3 = |B_1 - B_2 + B_3|_{2^n+1}^{n} \quad (7)$$

Fig.4. shows the architecture for finding r1, r2, r3 using equations (1), (2) and (3). If value of r3 comes negative, in such a case r3 is subtracted from 2^n+1.

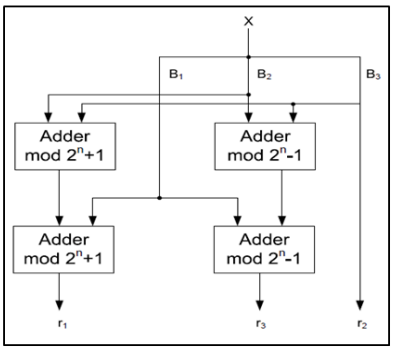


Figure 4: Binary to RNS Forward Converter

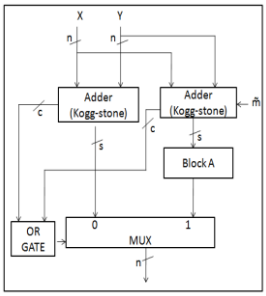
3.2 Modulo Adder

The result of modulo-m addition of two numbers X and Y can be given as:-

$$|X+Y|_m = \begin{cases} X+Y, \text{ for } X+Y < m. \\ X+Y-m, \text{ for } X+Y \geq m. \end{cases}$$

Where 0< X, Y <m

The above expression can be implemented using following architecture given below, fig.5 [3]. Architecture in figure 5 is used for modulo addition for modulus $2^n - 1$. m̃ is the 2's compliment of m, which is used for subtracting m from addition of X and Y. A parallel prefix Kogg-Stone adder is used for addition, any conventional adder can be used instead. Kogg-Stone adder is a faster one but requires more area. The architecture shown in figure 7 is for modulus $2^n + 1$ [3].



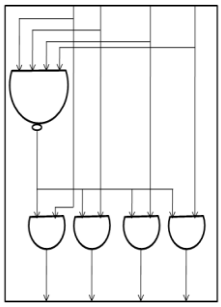


Figure 5: Modulo $2^n - 1$ adder **Figure 6**: Block A

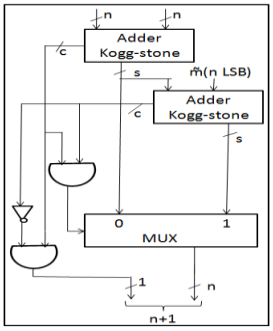


Figure 7: Modulo $2^n + 1$ adder

3.3 Modulo Multiplier

The output of forward converter is a residue set in the form {r1, r2, r3}. Residue sets of mantissa are needed to be multiplied. This process involves modular multiplication of corresponding residues, see ex (2). The modular partial products for each modulus can be found as,

3.3.1 Modulo 2^n-1 multiplier

The modular product nominally consists of modulo 2^n-1 sum of n partial products, each of which is ABi shifted left by i bits (i.e. multiplied by 2i), where Bi is the ith bit of B, This can be represented as

$$|AB|_{2^n-1} = \left|\sum_{i=0}^{n-1} AB_i 2^i\right|_{2^n-1}$$

$$= \sum_{i=0}^{n-1} P^{(i)}$$

where ith partial product $P^{(i)}$ is

$$P^{(i)} = \left|2^i \sum_{j=0}^{n-1} P_j^{(i)} 2^j\right|_{2^n-1}$$

$$= \left|2^i \sum_{j=0}^{n-1} A_j B_i 2^j\right|_{2^n-1}$$

$$= \left|2^i B_i (A_{n-1}2^{n-1} + A_{n-2}2^{n-2} + \ldots\ldots A_0 2^0\right|_{2^n-1}$$

The above equation can further be simplified as

$$P^{(i)} = 2^{n-1}A_{n-i-1}B_i + 2^{n-2}A_{n-i-2}B_i + \ldots\ldots + 2^i A_0 B_i + 2^{i-1}A_{n-1}B_i$$
$$+2^{i-2}A_{n-2}B_i + \ldots\ldots\ldots\ldots + 2^0 A_{n-i}B_i \quad (8)$$

The equation of partial product can be seen as i bit cyclic shift of ith partial product. After finding each partial products, all are added together and their modulo sum is the final result of multiplication.

3.3.2 Modulo 2^n multiplier

To calculate $|AB|_{2^n}$, initially find out all P^i given by

$$P^{(i)} = \left|2^i \sum_{j=0}^{n-1} P_j^{(i)} 2^j\right|_{2^n-1}$$

$$= \left|2^i \sum_{j=0}^{n-1} A_j B_i 2^j\right|_{2^n-1}$$

$$= \left|2^i B_i (A_{n-1}2^{n-1} + A_{n-2}2^{n-2} + \ldots\ldots A_0 2^0\right|_{2^n-1} \quad (9)$$

Now only consider the last n LSB bits of the each $P^{(i)}$, By modulo adding these $P^{(i)}$ gives result of multiplication.

3.3.3 Modulus 2^n+1 multiplier

Given two operands, *A* and *B*, the modulo (2^n +1) product is given by

$$|AB|_{2^n-1} = \left|\sum_{i=0}^{n-1} AB_i 2^i\right|_{2^n+1}$$

$$= \left|\sum_{i=0}^{n-1} P^{(i)}\right|_{2^n+1}$$

The $P^{(i)}$ can be given as

$$P^{(i)} = |B_i L - B_i U|_{2^n+1}$$

where $L = A_{n-i-1}A_{n-i-2}\ldots\ldots A_0 00\ldots.0$ (i 0's)

$U = 00\ldots\ldots 0A_{n-1}A_{n-2}A_{n-3}\ldots\ldots A_i$ (n-i 0's)

The $P^{(i)}$ can furter be simplified as

$$P^{(i)} = \left|\begin{matrix} B_i.(A_{n-i-1}A_{n-i-2}\ldots\ldots A_0 \overline{A}_{n-1}\overline{A}_{n-2}\overline{A}_{n-3}\ldots\ldots\overline{A}_{n-i}) \\ +\overline{B}_i.(00\ldots.011)+1 \end{matrix}\right|_{2^n+1} \quad (10)$$

3.4 Reverse Converter

Reverse converter for the moduli set {2n - 1, 2n, 2n+ 1}. Consider an integer, X is defined by residue set {r_1, r_2, r_3} for moduli set {m_1, m_2, m_3} then X can be represented as [10]

$$X = \left| m_2(r_2 - r_1) + r_2 + m_3 m_2 \left| \frac{m_1}{2} + \frac{r_1 + r_3}{2} - r_2 \right|_{m_1} \right|_M \quad (11)$$

$$X = \left| m_2(r_2 - r_1) + r_2 + \frac{m_3 m_2}{2} |m_1 + r_1 + r_3 - 2r_2|_{m_1} \right|_M$$

$$X = \left| m_2(r_2 - r_1) + r_2 + \frac{m_3 m_2}{2} |r_1 - 2r_2 + r_3|_{m_1} \right|_M \quad (12)$$

The above equation is derived for moduli set of the form {2n- 1, 2n, 2n+ 1} in [11]. The same equation can be used for moduli set used in proposed architecture. The above equation for n=8 can be written as

$$X = \left| 2^8(r_2 - r_1) + r_2 + \frac{2^8(2^8+1)}{2} |r_1 - 2r_2 + r_3|_{2^8-1} \right|_M$$

$$= \left| 2^8(r_2 - r_1) + r_2 + 2^7(2^8+1) |r_1 - 2r_2 + r_3|_{2^8-1} \right|_M$$

4. Experimental Result

Design has two inputs of 16 bit each and output is of 32 bits. Output consists of mantissa of 23 bit and exponent of 8bit. Result of floating point multiplication is shown in Fig. 3. Here ra1, ra2, ra3 and rb1, rb2, rb3 are residues of a_mantissa, b_mantissa respectively and rm1, rm2, rm3 are the residues of multiplication of a_mantissa, b_mantissa. Design is targeted to FPGA of Virtex6 xc6vlx240t-2ff784 device.

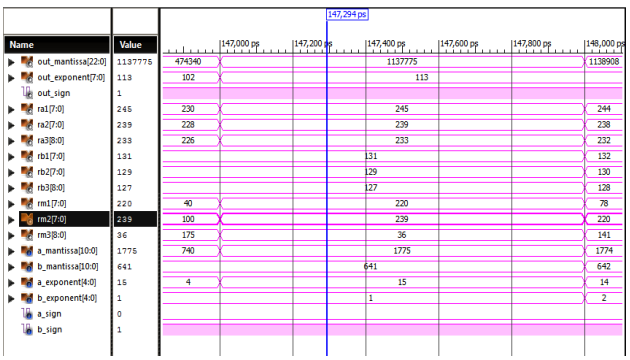


Figure 8: Simulation result of multiplier

TABLE I
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

5. Conclusion

This paper presents floating point multiplier unit which supports IEEE 754 half precision binary floating point number format. Design is targeted to FPGA of Virtex6 xc6vlx240t-2ff784 device. Multiplier is compared with previously designed multiplier and it is found that proposed architecture is faster, in the expense of increase in number of slice flip-flop. Further research is required for reducing computation time.

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 Hossein-zadeh, 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 $2^n - 1$ and $2^n + 1$ 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

Signal Processing", In Proceedings of the IEEE Transactions on, vol. 39, no. 5, pp 325-329,1992.

5. C.-L. Chiang and L. Johnson, "Residue arithmetic and VLSI", In Proceedings of the IEEE International Conference on Computer Design: VLSI in computers, 1983.
6. Jeremy Yung Shern Low and Chip-Hong Chang, "A VLSI efficient programmable power-of-two scalar for RNS", In Proceedings of the IEEE Transactions On Circuits And System-I: Regular Papers, Vol. 59, No. 12, December 2012.
7. J. Claude Bajard and T. Plantard, "RNS bases and conversions".
8. Naamatheertham R Samhitha, Neethu Acha Cherian, Pretty Mariam Jacob, P Jayakrishnan, "Implimentation of 16-bit floating point multiplier using residue number system", In Proceedings of the International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), 2013.
9. R. Amos Omondi, Benjamin, "Residue Number System: Theory and Implementation", Imperian College Press 2007.
10. Kazeem Alagbe Gbolagade, George Razvan Voicu, and Sorin Dan Cotofana, "An efficient FPGA design of residue-to- binary converter for the moduli set {2n - 1, 2n, 2n+ 1}", In Proceedings of the IEEE Transaction On Very Large Scale Integration (VLSI) System, vol. 19, no. 8, August 2011.
11. Maryam Saremi, Somayeh Timarchi, "Efficient 1-out-of-3 Binary Signed-Digit Multiplier for the moduli set {2n-1, 2n, 2n+1}", In Proceedings of the IEEE 2013.
12. Sunil M, Ankith R D, Manjunatha G D and Premananda B S, "Design And Implementation Of Faster Parallel Prefix Kogg Stone Adder", In Proceedings of the International journal Of Electrical And Electronics Engineering & Telecommunication.

Author Profile

Praveen V Amrutkar received the B.E. degree in Electronics Engineering from Rajiv Gandhi College of Engineering Research and Technology, RTMNU University, Maharashtra in 2012. He is currently a M.Tech Scholar at Department of Electronics Engineering, Yeshwantrao Chavan College of Engineering, Nagpur.

Prasanna M Palsodkar worked as a lecturer in central India's premier engineering college," G. H. Raisoni College of Engineering" since 2003. He completed M.Tech (Electronics) in 2006. He left G. H. Raisoni College of Engineering in 2010 and now working in Yeshwantrao Chavan College of Engineering from July 2010. Specialties: VLSI, Signal Processing