ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Implementation of Single Precision Floating Point RNS MAC Using Vedic Multiplier

A.Chandana
M.Tech,
VLSI,
Gokaraju Rangaraju Institute of Engineering & Technology, Hyderabad, India.

D.L.Chaitanya
Associate Professor,
Department of ECE,
Gokaraju Rangaraju Institute of Engineering & Technology, Hyderabad, India.

Abstract:

This paper presents the design and implementation of 32-bit floating point RNS Multiply and Accumulate (MAC) unit using Vedic Multiplier. MAC is one of the most important components in many of DSP applications like convolution and filtering. RNS (Residue Number System) gained popularity in implementation of fast arithmetic and fault tolerant computing applications because of parallel processing and carry free computations. Floating point RNS arithmetic units have obvious advantage over fixed point MAC units which are key units in Digital Signal Processors. Floating point RNS MAC uses modulo adder for exponent addition and modulo multiplier for mantissa multiplication where operations are performed on moduli. In previous methods, modulo multiplier is implemented using array multiplier that has more delay. The beauty of Vedic multiplier is that here partial products generation and additions are done concurrently reducing delay, which is the primary motivation behind this work. The design is coded in Verilog HDL and synthesized using Xilinx ISE 13.1. The results clearly show that Vedic multiplier can be used to improve the execution speed of Floating point RNS based MAC when compared to array multiplier and an application of Floating point RNS MAC, 8 tap FIR (Finite Impulse Response) filter is implemented.

Key words:

MAC, RNS, Floating point, Moduli, Vedic multiplier, Array multiplier.

I. INTRODUCTION:

MAC is the main component in many of the digital signal processing applications like convolution, filters and FFT. A basic MAC architecture consists of a multiplier and accumulator.

The products generated by the multiplier are added and stored in accumulator. The performance of MAC can be increased by the optimized design of multiplier and adder. RNS 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 [3]. That is because there is no need of communicating carry information between two residues. So carry free arithmetic can be performed. A high speed MAC capable for handling large range numbers with better precision will be required for many of the DSP applications. There are two types of arithmetic operation which are fixed and floating point operations. Fixed point number was inefficient for large 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 MxB^E where M is mantissa, B is base and E is exponent. The floating point representation used here is 32-bit floating point representation (single precision) where there is 1-bit sign, 8-bit exponent and 23-bit mantissa. In previous methods, floating point RNS based MAC was implemented using array multiplier which has more delay. To reduce the delay, Vedic multiplier is used here, which is quite different from the conventional method of multiplication like add and shift. "Urdhva Tiryakbhyam" [5] sutra is a general multiplication formula applicable to all cases of multiplication. It literally means "Vertically and Crosswise".The entire paper is divided into six sections. In first section introduction of project is given. In second section Vedic multiplier is discussed. Third section provides the information about the floating point RNS MAC using Vedic multiplier. Fourth section explains implementation of FIR filter using MAC. Fifth section explains the results and sixth section concludes this paper.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

II. VEDIC MULTIPLIER:

Urdhava Tiryakbhyam (Vertically and Crosswise), is one of sixteen Vedic sutras and deals with the multiplication of numbers. The sutra is illustrated in Figure 1 with multiplication of two numbers and the hardware architecture is depicted in Figure 3. The digits on both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result and a carry. This carry is added in the next step and hence the process goes on. If more than one line are there in one step, all the results are added to the previous carry. In each step, the least significant bit acts as the result bit and all other bits act as carry for the next step [4]. Initially the carry is taken to be zero.

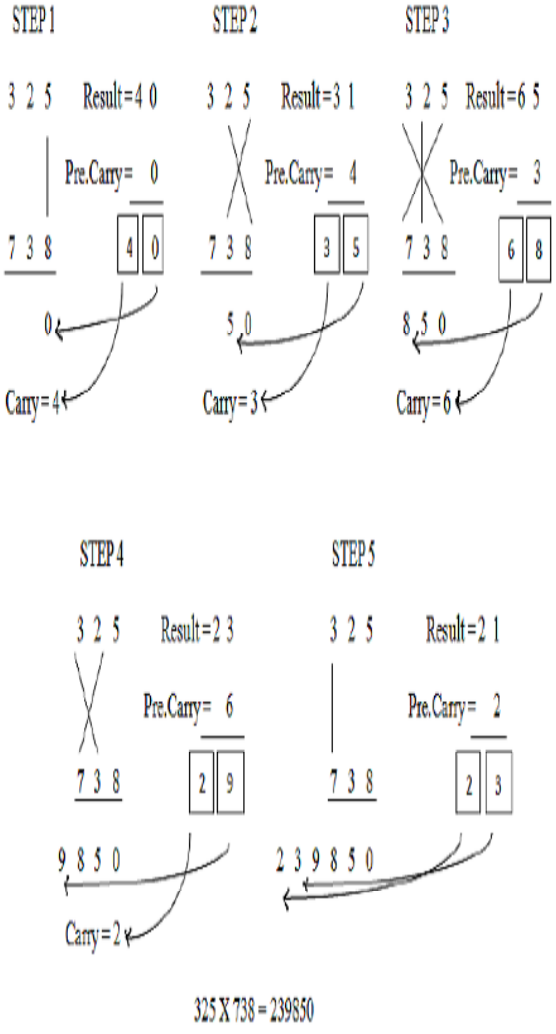

Figure 1. Multiplication of two decimal numbers by Urdhava Tiryakbhyam sutra.

From Figure 1, it is observed that all the partial products are generated in parallel. So the speed of the multiplier is higher compared to conventional multiplier.

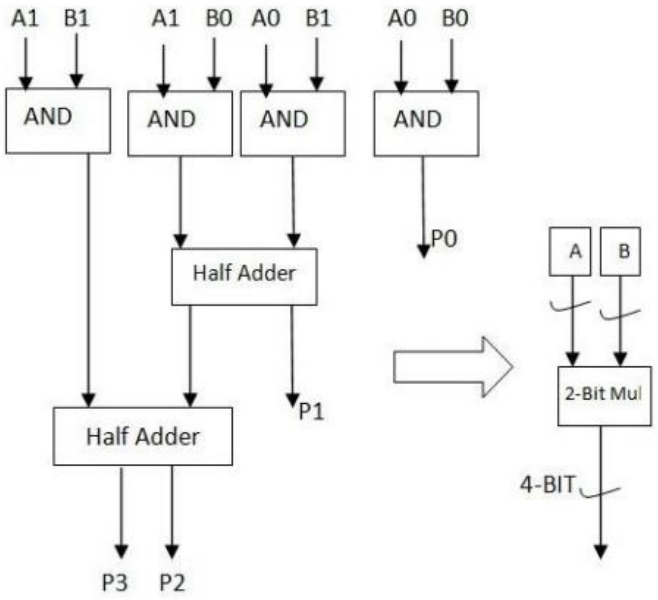

Figure 2. 2-bit Vedic multiplier

Note: 4-bit multiplier is designed using four 2-bt multipliers, 8-bit multiplier is designed using 4-bit and l6-bit using 8-bit as shown in figure3 below [5].

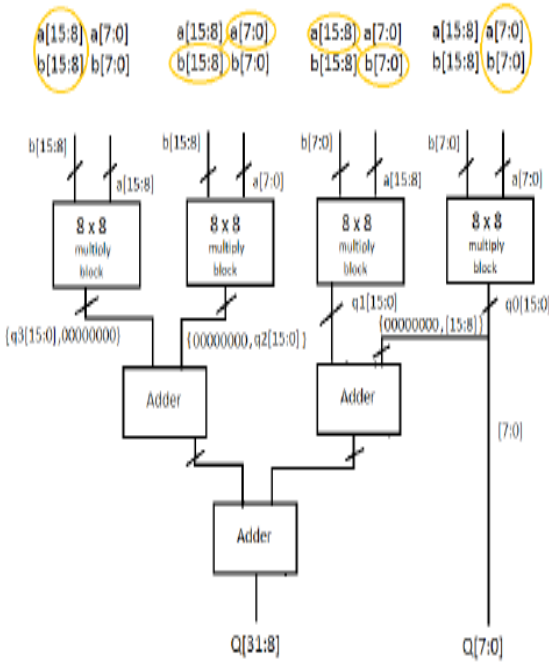

Figure 3. Block diagram for 16-bit Vedic Multiplier

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

III. ARCHITECTURE OF FLOATING POINT RNS MAC UNIT USING VEDIC MULTIPLIER:

The modules that are required for implementing a Floating Point RNS MAC are Binary to RNS Converter, Modulo Adder, Modulo Multiplier (using Vedic multiplier), RNS to Binary Converter, adder and Accumulator. The block diagram of Floating point RNS MAC unit is shown in Figure 4. Floating point multiplication involves multiplication of mantissa and addition of exponent. So, Floating point RNS multiplier uses RNS modulo multiplier for mantissa and RNS modulo adder for exponent. The block diagram of Floating point RNS multiplier is shown in Figure 5.

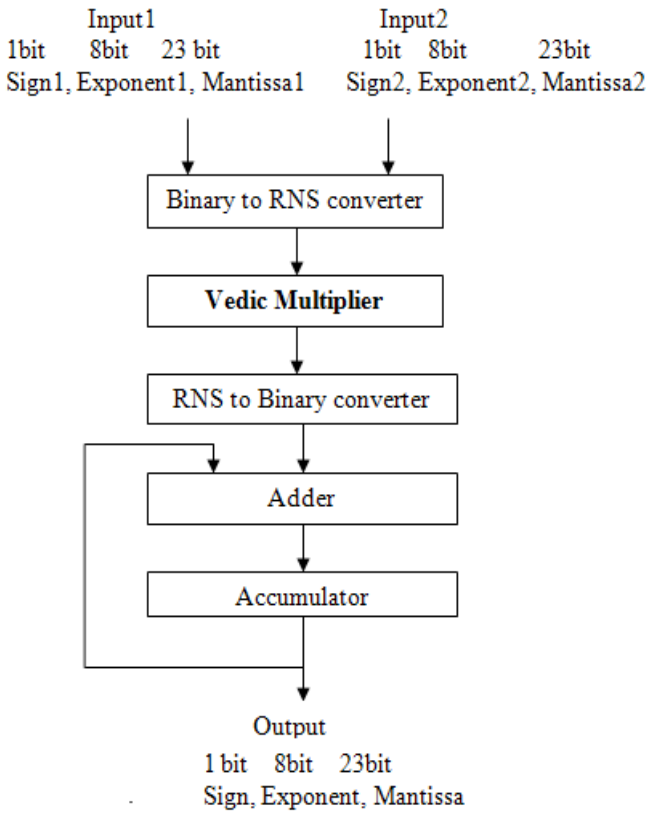

Figure 4. Block diagram of Floating Point RNS MAC using Vedic Multiplier.

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.

A. Conversions:

The process of converting binary data into RNS is referred to as the forward conversion. 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 representation is referred to as backward conversion. The forward conversion can be for an arbitrary moduli set or special moduli set [3]. 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. Special moduli set is taken to be a low-cost moduli set and improves the performance of the unit [2]. Since the exponent is of 8-bits, the moduli set required for the exponent is {7, 8, 9} which enables to represent 0 to 2^8-1. Similarly, as the mantissa is of 23-bits, the moduli set considered is {65535, 65536, 65537} which enables to represent 0 to 2^{23}-1. This paper includes a parallel method of forward conversion of the input binary block [1]. For backward conversion Chinese Remainder Theorem is used [3]. The mathematical equations for CRT are as follows. Given a set of moduli {m1,m2,…,mi} and the residues are {r1,r2,…,ri} then binary number 'X' is given as

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

As we have moduli set $\{m_1,m_2,m_3\}$, so

$$M_1 = m_1*m_2*m_3/m_1 = M/m_1$$
where $M = m_1*m_2*m_3$

$$M_2 = m_1*m_2*m_3/m_2 = M/m_2$$

$$M_3 = m_1*m_2*m_3/m_3 = M/m_3$$

And M_i^{-1} can be obtained by,

$$|M_i^{-1} \;\; M_i| = 1$$

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

For moduli set $\{2^n-1, 2^n, 2^n+1\}$, M_i^{-1} values can be obtained as

$$M_1^{-1} = \left(\frac{m1+1}{2}\right)$$

$$M_2^{-1} = (m_2-1)$$

$$M_3^{-1} = \left(\frac{m3+1}{2}\right)$$

So CRT requires three main processes

1. Calculating M_i and inverses M_i^{-1}
2. Multiplying M_i and inverses with residues and accumulating it
3. Modulo M_i adder as the final stage

B. Floating Point RNS Multiplier:

Floating Point in residue domain includes a RNS multiplier for mantissa and RNS modulo adder for exponent. For large number of bits, the delay and area of residue array multiplier is more [7]. In order to reduce the delay, Vedic Multiplier is used instead of array multiplier. The block diagram of Floating point RNS multiplier is shown in Figure 5.

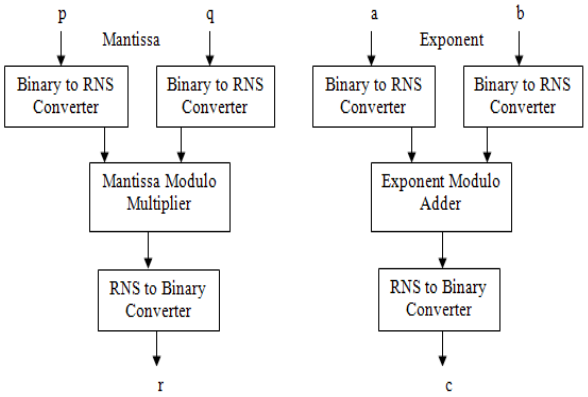

Figure 5. Block Diagram of Floating Point RNS Multiplier

C. Modulo-M adder:

The Modulo-M adder forms the basic arithmetic unit for any RNS operation or RNS conversion [1]. According to it, the modulo-m addition is done as follows

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

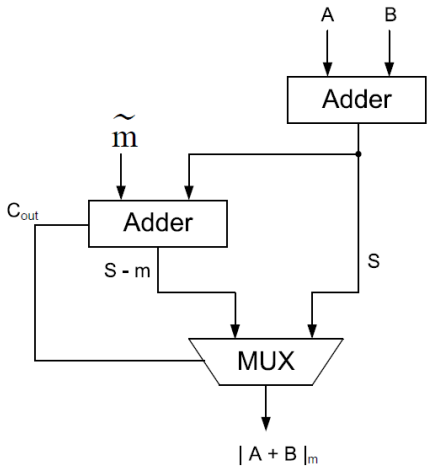

Figure 6. Modulo-M adder architecture

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. Kogge-Stone adder [1], a parallel prefix adder is used here.

D. Modulo-M multiplier:

For residue multiplication modulo multiplier is designed using Vedic multiplier. When 23-bit binary mantissa is converted into RNS, the residues each of 16-bits are obtained. So 16-bit Vedic multiplier is used.

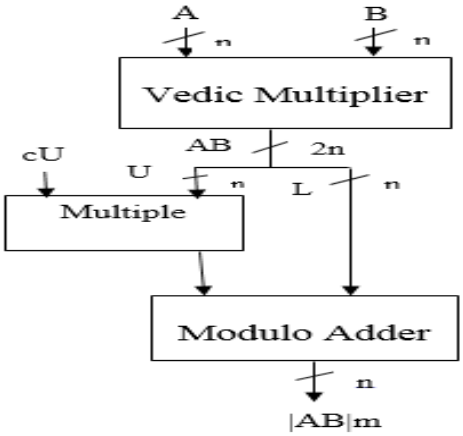

Figure 7. Modulo-M multiplier using Vedic Multiplier

Suppose A, B and m are each represented in n bits. Then AB is 2n bits wide, and $m = 2^n - c$, where $1 \leq c < 2^n - 1$. If we split AB into an upper half, U, and a lower half, L, then

$$\begin{aligned} |AB|_m &= |2^nU + L|_m \\ &= | \, |2^nU|_m + |L|_m \, |_m \\ &= |cU + L|_m \quad \text{since } |2^n|_m = c \end{aligned}$$

and $L < m$

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Application of this last equation requires another multiplication, but, given that 'c' will generally be small (c=1 for modulo 65535, 0 for modulo 65536 and -1 for modulo 65537), this operation need not be costly.

E. Floating Point RNS MAC:

Implementation of Floating point RNS MAC unit requires Floating point RNS multiplier, Adder, and accumulator. It takes floating point data and performs sequence of multiplication and addition. The products generated by the multiplier are added and stored in accumulator. 32 bit floating point adder is used to add the products generated by multiplier.

IV. Implementation of floating FIR filter using Floating point RNS MAC:

Finite Impulse Response (FIR) filters are widely used in various DSP applications. FIR filter is implemented as a series of multiply and accumulate operations [6]. Figure 8 shows N- tap FIR filter. The output of FIR filter is described by the following equation

$y(n)=a0x(n)+a1x(n-1)+\ldots..aN-1x(n-N)$

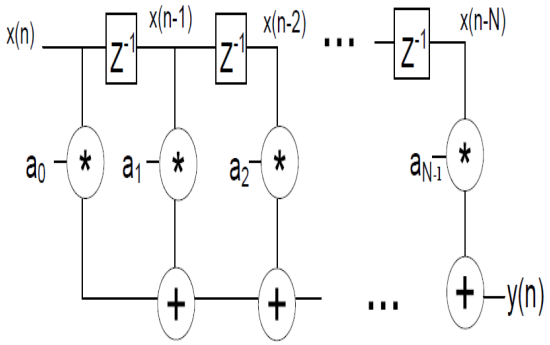

Figure 8. N-tap FIR filter

x(n) and filter coefficients are given as inputs to filter. Output y(n) is the sum of product of every multiplier. Here Z^{-1} represents unit delay, provided by D flip-flops.

V. Results:

Design is coded in Verilog HDL, synthesized using Xilinx 13.1. Simulation results for Floating point RNS multiplier are shown in Figure 9, Simulation results for Floating point RNS MAC, 8-tap FIR filter are shown in Figure 10 and Figure 11 respectively.

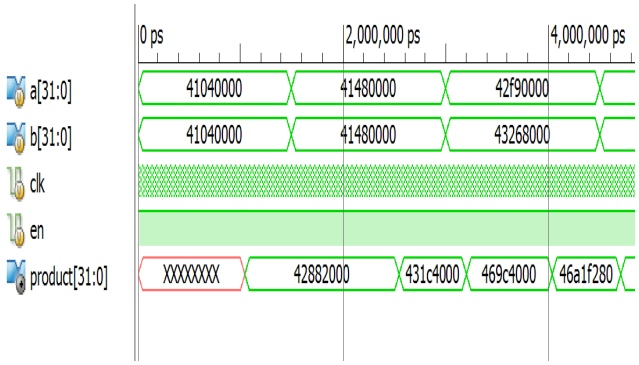

Figure 9. Simulation results for Floating point RNS multiplier.

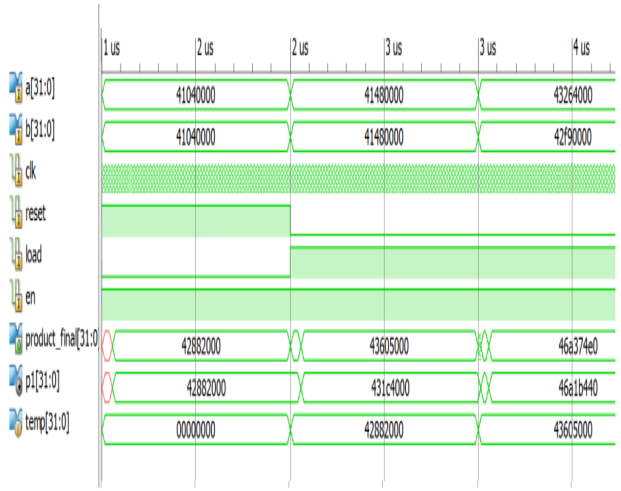

Figure 10. Simulation results for Floating point RNS MAC .

Here, p1 represents multiplier output, 'temp' is a register used to store previous MAC output, 'product_final' is the MAC output. Inputs and corresponding outputs of MAC in hexadecimal notation are shown in Table 1.

Table 1. Input/ output table for MAC:

Input a	Input b	Multiplier output	MAC output
41040000	41040000	42882000	42882000
41480000	41480000	431c4000	43605000
43264000	42f90000	46a1b440	46a374e0

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

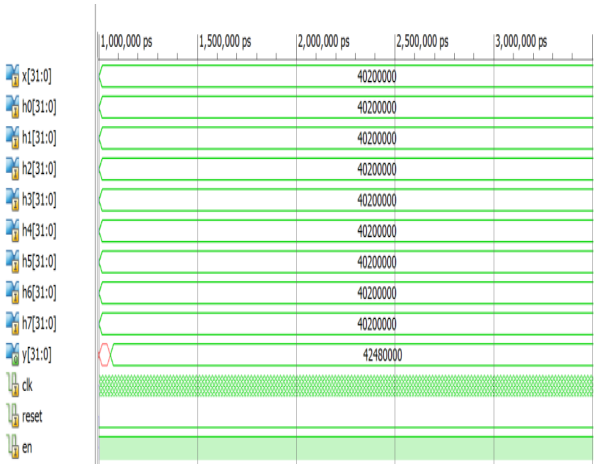

Figure 11. Simulation results for 8 tap FIR filter

Delay comparison of Floating point RNS MAC using array multiplier and Vedic multiplier is shown in Table 2. It shows that Vedic multiplier can be used to improve the execution speed of Floating point RNS based MAC when compared to array multiplier.

Table 2. Delay comparison of Floating point RNS MAC using array multiplier and Vedic multiplier:

Multiplier used	MAC Delay(ns)
Array multiplier	81.537
Vedic multiplier	70.243

VI. Conclusion:

Single precision (32-bit) floating point RNS based MAC using Vedic multiplier is designed using Verilog HDL and synthesized using Xilinx ISE 13.1. By using Residue Number System parallel and carry free arithmetic can be obtained. Using Urdhva Tiryakbhyam sutra, partial products can be generated in parallel. The speed comparison of floating point RNS based MAC using array multiplier and Vedic multiplier is presented. The results clearly show that Vedic multiplier can be used to improve the execution speed of Floating point RNS based MAC when compared to array multiplier. An application of Floating point RNS based MAC i.e. 8 tap FIR filter is implemented.

References:

[1] Dhanabal, R., et al, "Implementation of floating point MAC using Residue Number System", Optimization, Reliability, and Information Technology (ICROIT), 2014 International Conference on, IEEE, pp. 461-465, Feb 2014.

[2] Ghosh, Aniruddha, Satrughna Singha, and Amitabha Sinha, "Floating point RNS: a new concept for designing the MAC unit of digital signal processor", ACM SIGARCH Computer Architecture News, vol. 40, no. 2, pp. 39-43, May 2012.

[3] Omondi, Amos, and Benjamin Premkumar, "Residue number systems: theory and implementation", Imperial College Press, 2007.

[4] Poornima, M., et al, "Implementation of multiplier using Vedic algorithm", International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 6, no. 6, pp. 219-223, May 2013.

[5] Rudagi, J. M., Vishwanath Ambli, Vishwanath Munavalli, Ravindra Patil, and Vinaykumar Sajjan, "Design and implementation of efficient multiplier using Vedic mathematics", pp.162-166, 2011.

[6] Borkar, Shraddha S., and Awani S. Khobragade. "Optimization of FIR digital filter using low power MAC" International Journal of Computer Science & Engineering Technology (IJCSET), vol. 2, no. 4, pp.1150-1154, April 2012.

[7] Singh, Avtar, and Srinivasan, "Digital signal processing implementations: using DSP microprocessors with examples from TMS320C54xx", Ch. 2, CI-Engineering, 2004.