Implementation of Chinese Remainder Theorem and Radix 8 Booth Algorithm to Perform Multiplication for Residual Number System using Verilog HDL

Phalguna P S[1]
Department of E & C Engineering
St Joseph Engineering College,
Mangaluru, D.K,India

Keith Raymond Fernandes[2]
Department of E & C Engineering
St Joseph Engineering College,
Mangaluru, D.K,India

***Abstract*— Residual Number System (RNS) represents a larger integer using a set of smaller integer for a set of selected moduli. The computation part of the RNS has an integer part multiplied with the selected modulo and a residual part. The selected moduli are absolute values, which are relatively prime [1]. In RNS multiplication process the residues of the multiplier and multiplicand are obtained for special set of moduli and multiplied respectively to get the residues of final product. The conversion of RNS to Decimal Number System is done by Chinese Remainder Theorem (CRT).**

In RNS multiplication process, multiplication of large numbers can be done at the same speed as on short numbers. The speed is determined by the largest modulo position. The computation complexity is decreased by representing the larger number as set of smaller numbers.

In this paper, a multiplier is implemented using CRT and Radix 8 Booth algorithm for RNS.This multiplier is checked for Power and Efficiency.

Index Terms*— RNS (Residual Number System), CRT (Chinese Remainder Theorem)

I. INTRODUCTION

A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems, communication systems, error control coding, cryptography etc.

Booth multiplication is a technique introduced by Andrew D. Booth in the year 1950. It allows smaller, faster multiplication by encoding the numbers that are multiplied. It is the standard technique used in chip design and provides significant improvement over the long multiplication technique.

The advantage of this method is the halving of the number of partial products. This is important in circuit design as it relates to the propagation delay in the running of the circuit, and the complexity and power consumption of its implementation. It is possible to reduce the number of partial products to half by using the technique of radix 4 Booth recoding and further it can be reduced by using the technique of radix 8 booth encoding.

The Radix 8 Booth multiplier is the advanced version of the normal Booth multiplier. This modified Booth multiplier gives the multiplied output using less number of partial products compared to the normal Booth multipliers.

RNS relies on the Chinese remainder theorem of modular arithmetic for its operation, a mathematical idea from Sun Tsu Suan-Ching in the 4th century AD. RNS represents a larger integer using a set of smaller integer for a set of selected moduli [2]. By considering a smaller equivalent integer in place of a larger integer will perform an efficient way of calculation. This will lead to a huge conservation of the bits which makes a fair impact on the arithmetic operations. Hence the efficiency of the operation increases by lowering the power consumption for the multiplier.

II. DESIGN APPROACH

This section focus on the design approach for Radix-8 Booth multipliers by considering the necessary specifications for develop the relevant source code in Verilog HDL using Finite State Machine.

The solutions of realizing high speed multipliers are to reduce the Partial products by factor of one third of the Booth multiplier method [4].

Since the numbers of partial products are less, the radix 8 Booth multiplier performs efficiently when compared to the normal Booth multipliers. The figure 1 shows the block diagram of the 9 bit Radix 8 Booth multiplier.

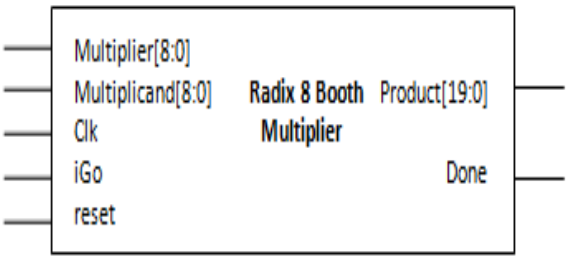

Figure1: Block diagram of Radix 8 Booth multiplier

The table 1 shows the algorithm for Radix 8 Booth multiplier.

Table 1: Radix-8 Booth Encoding Table

BLOCK	PARTIAL PRODUCT
0000	Arithmetic Shift Right (3 bit)
0001	1*Multiplicand and perform Arithmetic Shift Right (3 bit)
0010	1*Multiplicand and perform Arithmetic Shift Right (3 bit)
0011	2*Multiplicand and perform Arithmetic Shift Right (3 bit)
0100	2*Multiplicand and perform Arithmetic Shift Right (3 bit)
0101	3*Multiplicand and perform Arithmetic Shift Right (3 bit)
0110	3*Multiplicand and perform Arithmetic Shift Right (3 bit)
0111	4*Multiplicand and perform Arithmetic Shift Right (3 bit)
1000	-4*Multiplicand and perform Arithmetic Shift Right (3 bit)
1001	-3*Multiplicand and perform Arithmetic Shift Right (3 bit)
1010	-3*Multiplicand and perform Arithmetic Shift Right (3 bit)
1011	-2*Multiplicand and perform Arithmetic Shift Right (3 bit)
1100	-2*Multiplicand and perform Arithmetic Shift Right (3 bit)
1101	-1*Multiplicand and perform Arithmetic Shift Right (3 bit)
1110	-1*Multiplicand and perform Arithmetic Shift Right (3 bit)
1111	Arithmetic Shift Right (3 bit)

This algorithm scans strings of four bits as follows:

1) Extend the sign bit 1 position if necessary to ensure that n is even.

2) Append a 0 to the right of the LSB of the multiplier.

3) According to the value of each vector, each Partial Product will be 0, +y, -y, +2y,-2y, +3y, -3y, +4y or -4y [4].

Radix-8 booth encoder performs the process of encoding the multiplicand based on multiplier bits. It will compare 4 bits at a time with overlapping technique. Grouping starts from the LSB, and the first block only uses three bits of the multiplier and assumes a zero for the third bit. The functional operation of Radix-8 booth encoder is shown in the Table 1.

The Radix 8 Booth multiplier can be designed by using the Finite State Machine Technique. In this technique four states are considered for the design of the Radix 8 Booth Multiplier. They are wait for Go state, initial state, Add shift state, done state.

Wait for Go state: This state checks for the availability of the inputs for Radix 8 Booth multiplier. If the inputs are ready, the next state i.e. initial sate is activated and if the inputs are not ready then this state will be continued until the inputs are given by the user.

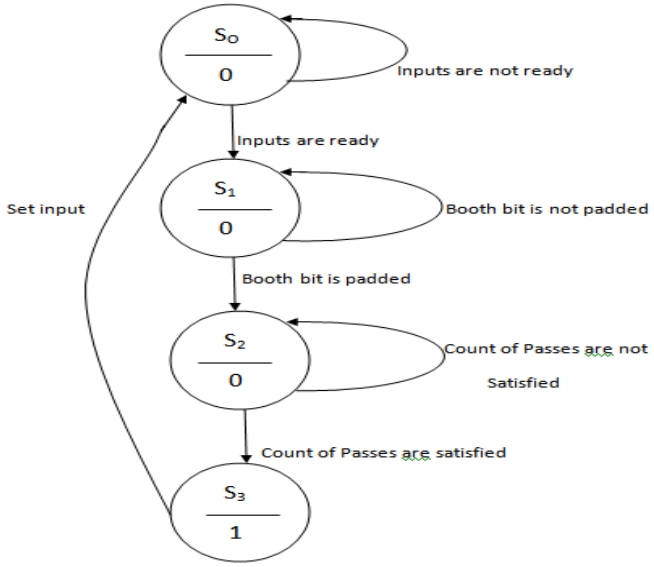

Figure 2: FSM design for Radix 8 Booth multiplier

Initial state**:** This state will mainly concentrate on the several issues such as addition of the booth bit (a zero bit) to the LSB of the first Partial Product, padding of the sufficient bits to the MSB of the first Partial Product, number of passes that has to be performed in the radix 8 Booth multiplication and the sign bit extension concepts. The Radix 8 Booth multiplication process requires 'n/3' passes for a n bit input.

If all the issues are satisfied the next state i.e. Add Shift state is activated or else it will remain in the same state.

Add Shift state: This state acts according to the Radix 8 Booth algorithm. In this state the last four bits of the partial product is considered and the particular addition operation is performed according to the algorithm followed by shifting the partial product to the right (i.e. Arithmetic Shift Right) by three bit. This state execute until the number of passes are satisfied (i.e. n/3 passes). If the numbers of passes are less than n/3, then same state will be executed until it is satisfied.

Done state: This is the final state of the Finite State Machine. Here LSB of the last partial product is discarded and the values are get stored the product register.

The radix 8 booth multiplier provides more efficiency compared to Radix 2 and Radix 4 Booth multipliers. Hence radix 8 Booth multiplier is chosen. Table 2 shows the various parameters obtained for different Booth multipliers.

Table 2: Design summary of Booth multipliers

Parameter	Radix 2	Radix 4	Radix 8
Time delay	80 ns	40 ns	30ns
Number of LUTs	56	69	109
Number of I/Os	36	36	42

III. RNS MULTIPLIER

This section focus on the design of RNS multiplier. This multiplier consists of three blocks namely Forward converter block, Multiplier Block and Reverse converter block.

A. Forward converter

The conversion of decimal number to residual number is done using Forward converter. One of the most important considerations when designing RNS Systems is the choice of the moduli set. The choice of moduli affects the complexity of forward and reverse converters as well as RNS arithmetic circuits [5].

The moduli set ($m_1, m_2 \ldots\ldots m_i$) should be chosen such that the moduli m_i s satisfy the following criteria:

1. They should be pair wise prime. That is, $gcd(m_k, m_j) = 1$ for all $m_k \neq m_j$.
2. Each moduli m_i should be as small as possible so that operations modulo m_i require minimum computational time.
3. The moduli m_is should imply simple binary to RNS and RNS to binary conversions as well as simple RNS arithmetic.
4. The moduli product should be large enough to implement the desired dynamic range.
5. The moduli should provide a well balanced decomposition of the dynamic range. This means that the difference in word length between the moduli should be as small as possible [6].

Sets with all elements being of the forms ($2^n + 1$, 2^n , $2^n - 1$, $2^{2n} + 1$) for n=2,3 which satisfy the requirement of simple conversions and efficient modulo arithmetic is considered for the design of Forward converter. By considering n=2, a moduli set (3, 4, 5, 17) is selected for the design of RNS multiplier. Hence the dynamic range of the RNS multiplier is 1020. Figure 3 shows the block diagram of 16 bit forward converter.

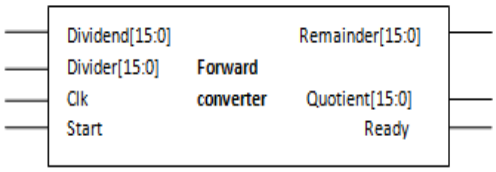

Figure 3: block diagram of Forward converter

B. Multiplier

The residual output of the Forward converter which is having less number of bits is fed to the Radix 8 Booth multiplier as discussed in section II which produces the product in the form of residual number. Since inputs to the multiplier are very small, the multiplier performs efficiently.

C. Reverse converter

This converts the residual number into decimal number. The product generated by the multiplier is considered. This product is again converted back to the decimal number by applying CRT. The equation for CRT is given as,

$$X = \sum_{i=1}^{N} A_i T_i r_i \bmod M \quad (1)$$

Where, 'A' is the result of M/m_i, 'T' is the multiplicative inverse, 'r' is the residual number and 'M' represents the dynamic range of the selected set of prime moduli (3,4,5,17) i.e. 1020.

The multiplicative inverse term 'T' should be selected in such a way that congruence should be 1. The block diagram of reverse converter is as shown in Figure 4, where r, s, t represents the residual numbers which are the product of radix 8 Booth multipliers.

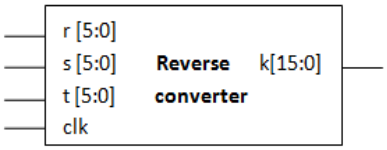

Figure 4: block diagram of Reverse converter

IV. PROPOSED ARCHITECTURE

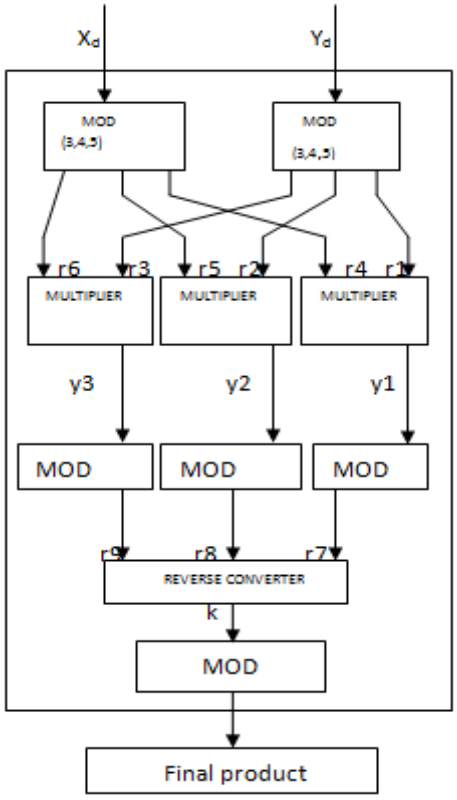

Figure 5: Architecture for RNS multiplier

The RNS multiplier consists of forward converter, radix 8 Booth multiplier and a reverse converter. The inputs are given in decimal numbers which gets converted into residual number by the forward converter and fed into the multiplier. The product is obtained in the form of residue number which again converted back to the decimal number by applying CRT to get the final product.

V. RESULTS

This section focuses on the results obtained for various blocks of the RNS multiplier. Figure 6 represents the forward converter in which the modulo 3 is applied for the decimal number 10 resulting in a residual number 1.

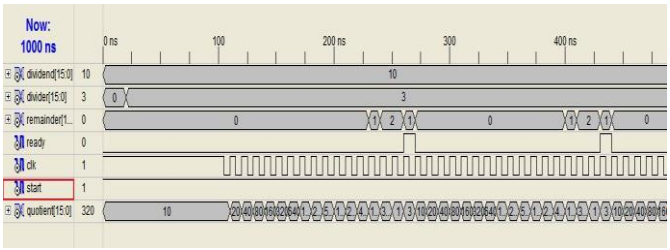

Figure 6: Output waveform for Forward converter

Figure 7 shows the result obtained for Radix 8 Booth multiplier where 12 and 2 are the inputs given resulting 24 as the product.

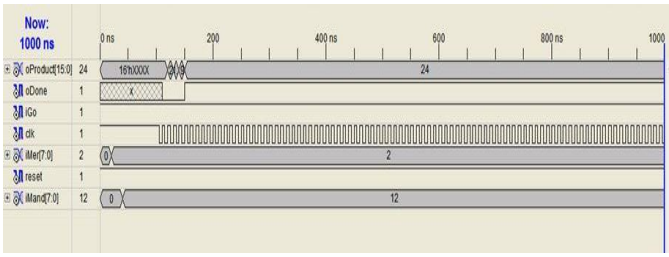

Figure 7: Output waveform for Radix 8 Booth multiplier

Figure 8 shows the result obtained for reverse converter where the input is the residual number (0,3,6) for moduli set (3,5,7) resulting in a decimal value 153.

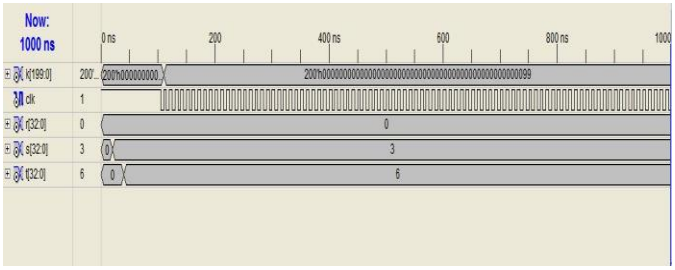

Figure 8: Output waveform for Reverse converter

V. CONCLUSION

This project discuss about the implementation of efficient algorithm for Booth multiplier which are used in the design of FIR filters. Here the brief description about Radix 8 Booth multiplier is explained.

It is observed that the partial products obtained by the multiplication process of two signed or unsigned numbers are reduced for the Radix 8 Booth multiplier. Hence the efficiency of the multiplier will be increased.

Here the efficiency of Modified Booth Multiplier will be more than the normal Booth multiplier, where as the Radix 8 Booth multiplier will give more efficiency compared to Booth multiplier and Modified Booth multiplier because of the reduction of partial products.

The use of RNS multiplier will cause huge reduction in the number of bits used for the multiplication process and hence increasing the efficiency.

As the future work, power consumption of this RNS multiplier is estimated with various other multipliers.

REFERENCE

[1] R. Muralidharan and C. H. Chang, "*Radix-8 Booth encoded (2^n-1) modulo multipliers with adaptive delay for high dynamic range Residue Number System*", *IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58*, no. 5, pp. 982–993, May 2011.

[2] K. Bhaskara Rao, B.Chinna Rao, "*Radix-8 Booth Encoded Modulo (2^n-1) Multipliers with Parallel Prefix Adder For High Dynamic Range RNS*", IJERD, Volume 5,Issue 1 ,2012.

[3] A. D. Booth, "A signed binary multiplication technique", *Quarterly J. Mechan. Appl. Math*., vol. IV, part 2, 1951.

[4] Neredimelli VVP Hide, Dr. I. Shanthi Prabha, "*Design of Modulo 2^n-1 Based on Radix -8 Algorithm for RNS and MAC applications*", IJRCCT, ISSN-2278-5841, Issue 3,Aug-2012.

[5] Andreas Persson, Lars Bengtsson, "*Forward and Reverse Converters and Moduli Set Selection in Signed-Digit Residue Number Systems*" 8 March 2007 / Revised: 21 May 2008 / Accepted: 12 June 2008 © 2008 Springer Science + Business Media, LLC. Manufactured in The United States.

[6] Abdallah, M., & Skavantzos, " *A systematic approach for selecting practical moduli sets for residue number systems*". In *Proceedings of the 27th IEEE southeastern symposium on system theory* (pp. 445–449) (March).