ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING

National Conference on Advanced Innovation in Engineering and Technology (NCAIET-2015)

Alva's Institute of Engineering and Technology, Moodbidri

Vol. 3, Special Issue 1, April 2015

An Effective Algorithm ForMultiplication For Residual Number System

MrPhalguna P S[1], Mr Keith Raymond Fernandes[2]

Electronics and Communication Engineering, St. Joseph Engineering College, Mangalore, India[1]

Assistant Professor, Electronics and Communication Engineering, St. Joseph Engineering College, Mangalore, India[2]

Abstract: Residual Number System (RNS) represents a larger integer using a set of smaller integer for a set of selected moduli [1]. 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][2]. In RNS multiplication process the residues of the multiplier and multiplicand are obtained for 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 project, a comparison will be carried out between the Booth multiplier, Modified Booth multiplier and Radix-8 Booth multiplier with and without using RNS and are designed using Verilog HDL and implemented in FPGA. These multipliers are checked for Power and Efficiency.

Keywords: RNS (Residual Number System), CRT (Chinese Remainder Theorem).

I. INTRODUCTION

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 still it can be reduced by using the technique of radix 8 booth encoding.

Here the modified Booth multiplier is the advanced version of the normal Booth multiplier. This modified Booth multiplier gives the multiplied output in less number of steps compared to the normal Booth multipliers. So this modified Booth multiplier which can be used in the FIR filter consumes very less power compared to the Radix 2 Booth multiplier. Residual Number System (RNS) relies on the Chinese remainder theorem of modular arithmetic for its operation, a mathematical idea from Sun TsuSuan-Ching in the 4th century AD.

Residual Number System (RNS) represents a larger integer using a set of smaller integer for a set of selected moduli [2].A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems. Typical DSP applications where a multiplier plays an important role include digital filtering, digital communications and spectral analysis. Many current DSP applications are targeted at portable, battery-operated systems, so that power dissipation becomes one of the primary design constraints.

Since multipliers are rather complex circuits and must typically operate at a high system clock rate, reducing the delay of a multiplier is an essential part of satisfying the overall design. In this project radix 2, radix 4 and radix 8 Booth multipliers are designed with and without using Residual Number System.

II. DESIGN APPROACH

This section focus on the design approach for Radix-2, Radix-4 and Radix-8 Booth multipliers by considering the necessary specifications for develop the relevant source code in Verilog HDL using Finite State Machine.

A. Booth Multiplication Algorithm for Radix-2:

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation.

The algorithm was invented by Andrew Donald Booth while doing research on crystallography at Brikbeck College in Bloomsbury, London [3].

Booth used desk calculators that were faster at shifting than adding created the algorithm to increase their speed.

Let M and Q be the multiplicand and multiplier respectively.

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING

National Conference on Advanced Innovation in Engineering and Technology (NCAIET-2015)

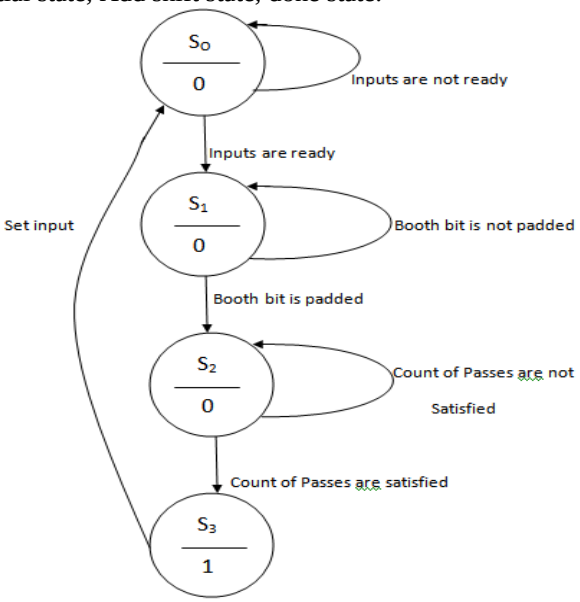
Alva's Institute of Engineering and Technology, Moodbidri

Vol. 3, Special Issue 1, April 2015

Table 1: Radix-2 Booth Encoding Table

BLOCK	PARTIAL PRODUCT
00	Arithmetic Shift Right (1 bit)
01	A=A+M and perform Arithmetic Shift Right (1 bit)
10	A=A-M and perform Arithmetic Shift Right (1 bit)
11	Arithmetic Shift Right (1 bit)

This algorithm works based on the last two bits of the partial product.Initially load the accumulator with zero's (4 bits) followed by accumulating the multiplier which is padded by a zero bit at the LSB forming a partial product. By considering last two bits of the partial product, do the operation as specified in the algorithm. After 'n' iterations discard the last bit of the obtained partial product to get the result.This Radix 2 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 2 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 2 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.

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 2 Booth multiplication and the sign bit extension concepts. The Radix 2 Booth multiplication process requires 'n' 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 2 Booth algorithm. In this state the last two 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 one bit. This state execute until the number of passes are satisfied (i.e. n passes). If the numbers of passes are less than n, then same state will be excecuted 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.

B. Booth Multiplication Algorithm for Radix-4:
One of the solutions of realizing high speed multipliers is to enhance parallelism which helps to decrease the number of subsequent calculation stages. The original version of the Booth algorithm (Radix-2) had two drawbacks. They are:

1) The number of add subtracts operations and the number of shift operations become variable and become inconvenient in designing parallel multipliers.
2) The algorithm becomes inefficient when there are isolated 1's. These problems are overcome by using modified Radix-4 Booth multiplication algorithm.

The salient features of this algorithm are:
Only n/2 clock cycles are needed for n-bit multiplication as compared to n clock cycles in Booth's algorithm.Isolated 0/1 is handled efficiently.For even n, the two's complement multipliers are handled automatically whereas for odd n an extension of sign bit is required.

Table 3: Radix-4 Booth Encoding Table

BLOCK	PARTIAL PRODUCT
000	Arithmetic Shift Right (2 bit)
001	1*Multiplicand and perform Arithmetic Shift Right (2 bit)
010	1*Multiplicand and perform Arithmetic Shift Right (2 bit)
011	2*Multiplicand and perform Arithmetic Shift Right (2 bit)
100	-2*Multiplicand and perform Arithmetic Shift Right (2 bit)
101	-1*Multiplicand and perform Arithmetic Shift Right (2 bit)
110	-1*Multiplicand and perform Arithmetic Shift Right (2 bit)
111	Arithmetic Shift Right (2 bit)

This algorithm scans strings of three bits as follows:
1) Extend the sign bit 1 position if necessary to ensure that n is even.

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING

National Conference on Advanced Innovation in Engineering and Technology (NCAIET-2015)

Alva's Institute of Engineering and Technology, Moodbidri

Vol. 3, Special Issue 1, April 2015

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 or -2y [4].Radix-4 both encoder performs the process of encoding the multiplicand based on multiplier bits.

It will compare 3 bits at a time with overlapping technique. Grouping starts from the LSB, and the first block only uses two bits of the multiplier and assumes a zero for the third bit.

The functional operation of Radix-4 booth encoder is shown in the Table 3.This Radix 4 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 4 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 4 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.

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 4

Booth multiplication and the sign bit extension concepts. The Radix 4 Booth multiplication process requires '$\lceil n/2 \rceil$' 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 4 Booth algorithm. In this state the last three 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 two bits.

This state execute until the number of passes are satisfied (i.e. n passes).

If the numbers of passes are less than n/2, 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.

C. Booth Multiplication Algorithm for Radix-8:

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

Table 4: 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 4.

This 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.

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING

National Conference on Advanced Innovation in Engineering and Technology (NCAIET-2015)

Alva's Institute of Engineering and Technology, Moodbidri

Vol. 3, Special Issue 1, April 2015

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.

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 extention concepts. The Radix 8 Booth multiplication process requires '$\lceil n/3 \rceil$' 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 2 Booth algorithm. In this state the last two 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 one bit. This state excecute until the number of passes are satisfied (i.e. n passes). If the number of passes are less than n, then same state will be excecuted 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.

III. RESIDUAL NUMBER SYSTEM

The multiplier and the multiplicand numbers which are selected is fed to the decimal to residual converter where the modulo operation is performed for both the multiplier and the multiplicand by considering the selected prime moduli set (3,5,7) The residual part obtained after the modulo operation is considered (i.e larger integer is converted into smaller set of integers in RNS). These residual numbers are multiplied by using effective multiplier (booth multiplier) and the final product obtained will be in the residual form. This product is again converted back to the decimal number by applying Chinese Remainder Theorem.

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

Where 'T' is the multiplicative inverse, 'r' is the residual number and 'M' represents the dynamic range of the selected set of prime moduli (3,5,7) i.e 105.

IV. RESULT

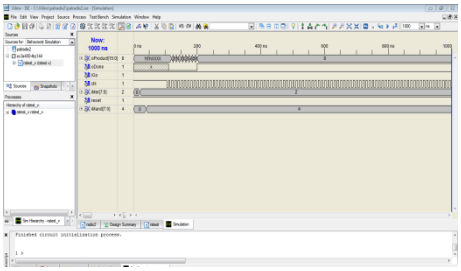


Fig 1: Output waveform for Radix 2 Booth multiplier

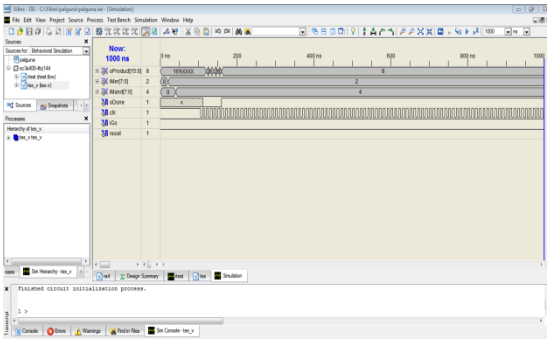


Fig 2: Output waveform for Radix 4 Booth multiplier

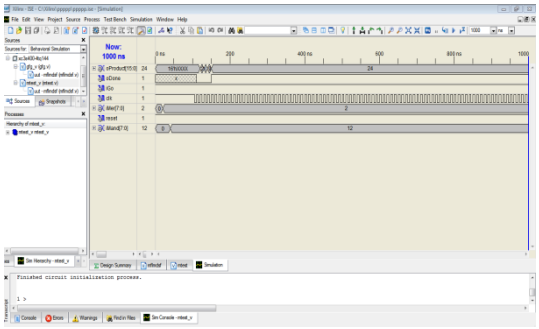


Fig 3: Output waveform for Radix 8 Booth multiplier

V. CONCLUSION

This project discuss about the implementation of efficient algorithm for modified booth multiplier which are used in the design of FIR filters. Here the brief description about Booth multiplier, Modified Booth Multiplier and Radix 8 Booth multiplier are explained.It is observed that the partial products obtained by the multiplication process of two signed or unsigned numbers are reduced for the Booth multiplier, Modified Booth Multiplier and Radix 8 Booth multiplier respectively. Hence the efficiency of the multipliers 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 future implementations of these multipliers are done with using RNS and are compared in power and efficiency factors.

REFERENCES

[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. BhaskaraRao, B.ChinnaRao, "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",QuarterlyJ. Mechan. Appl. Math., vol. IV, part 2, 1951.

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