Design and Implementation of 32 Bit Convolution Architecture Using Higher Radix Algorithm

P.Ramakrishna[1], M.Mounika[2]
[1]*Associate Professor, Department of Electronics and Communication Engineering*
Anurag Group of Institutions , Hyderabad, Telangana, India
[2]*PG Scholar, Department of Electronics and Communication Engineering*
Anurag Group of Institutions , Hyderabad, Telangana, India

Abstract- Convolution and de convolution algorithms play a key role in digital processing applications. They involve many multiplication and division steps and consume a lot of processing time. As such, they play a vital role in determining the performance of the digital signal processor. Convolution and de convolution implemented with Vedic mathematics proved fast as compared to those using conventional methods of multiplication and division. This paper presents a novel VHDL implementation of convolution and de convolution algorithm with multiplier using radix-256 booth encoding to reduce the partial product rows by eight fold and carry propagate free redundant binary addition for adding the partial products, thus, contributing to higher speed. The design had been implemented for 32 bit signed and unsigned sequences. The delay was reduced by 18.27%. The entire design was implemented in Xilinx ISE 14.7 targeted towards SPORTON 3E.

Keywords - Convolution and de convolution, Radix-256, Redundant binary (RB) addition, Xilinx ISE.

I.INTRODUCTION

Convolution is a method that describes the relation between the input signal, impulse response and output signal of a Linear Time Invariant (LTI) system. It plays a very important role in digital signal processing. De convolution is the process used to regenerate the original signal as it was before convolution. Both convolution and de convolution involves several tedious steps of multiplication, division and addition. These processes are slow and time consuming. If the number of partial products in a multiplier is reduced to two and if these are added without carry propagation, the multiplication can be performed with small time delay and as such, the performance of convolution and de convolution can be increased. Several authors presented different methods for implementation of convolution and de convolution. Authors of [1] used ancient Indian Vedic mathematics-Urdhva Triyagbhyam for multiplication and Nikhilam for division. Authors of [3, 13] implemented FIR filter of various orders with Radix-256 booth multiplier and RB addition. The existing method [1] for finding convolution and de convolution was implemented on a 4 bit sequence. But nowadays 16 bit processors and 16 bit sample sequences are not uncommon. Hence, the existing and proposed method, as well, are implemented on 16 bit sample sequences and results are compared.

II. Proposed Method

2.1 Proposed Method using Higher Radix-256 Algorithm and Redundant binary addition:

In the proposed method, the multiplier in the convolution is implemented using Radix-256 booth encoding with redundant binary addition. For a 32-bit multiplication, the partial products generated are only Four in radix-256 booth encoding. These Four are added using carry propagation free Redundant Binary addition.

2.1. Convolution

1) Linear convolution: The convolution y(n) of two finite length sequences f(n) and g(n) are given by equations (1) and (2)

$$y(n) = f(n)*g(n) \tag{1}$$

$$y(n) = \sum_{k=-\infty}^{\infty} f(k)g(n-k) \tag{2}$$

This involves the multiplication of the first sequence with the reversed and shifted version of the 2nd sequence, where the shifting is from -∞ to ∞.

2) Circular convolution: The circular convolution of two sequences is given by equations (3) and (4).

$$y(n) = f(n)*g(n) \qquad (3)$$

$$y(n) = \sum_{k=0}^{N-1} f(k)g((n-k) \bmod N) \qquad (4)$$

where N represents length of the sequence. Example 1: f(n) = {3,4,2,6}, g(n) ={1,0,3,6}

$$Y(n) = \begin{pmatrix} 3 & 6 & 2 & 4 \\ 4 & 3 & 6 & 2 \\ 2 & 4 & 3 & 6 \\ 6 & 2 & 4 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 3 \\ 6 \end{pmatrix} = \begin{pmatrix} 33 \\ 34 \\ 47 \\ 36 \end{pmatrix}$$

2.2 Deconvolution:

While y(n) is the convolution of f(n) with g(n), finding f(n) from y(n) using g(n) is defined as the de convolution provided f(n) and g(n) are causal.

$$f(n) = \frac{y(n) - \sum_{k=0}^{n-1} f(k)g(n-k)}{g(0)}, \quad n \geq 1 \qquad (5)$$

Where f(0)=y(0)/g(0) provided g(0) ≠ 0. The de convolution is similar to the division process. It involves only a few steps of multiplication while implementing division with Nikhilam. The improvement in delay is found to be less significant when radix-256 booth encoding with RB addition is used for multiplier in place of Vedic multiplier.

2.3. Radix -256 Booth Encoding

The Booth Encoding Method for partial product generation is valid for both signed and unsigned numbers. Using Radix-256 booth encoding, the partial products are reduced to two. In Booth Encoding, the multiplier is divided into (N/k) groups [9] where k is the number of bits in the Radix-256, (256=28 =2k). The multiplier is divided into 2 overlapping groups each containing k+1 bits. Each of these groups maps to a signed digit Di ranging from 0, 1………., (N/k)-1.

2.4. Redundant Binary number system:

Avizienis [6] proposed the Redundant Binary (RB) number system as a part of representation of signed digits. If the number of digits used to represent the numbers is greater than the radix, then it is called RB number system. The digits {0, 1} of binary (radix-2) are represented by a digit set { 1, 0, 1}. In this representation 1 = -1. The number of digits in the digit set of RB number system is greater than the Radix in the Natural Binary (NB) number system. Digital Electronic Circuits can represent only 0 and 1. Hence each digit in the RB number system is encoded with two digits in the NB number system as shown in table I. To represent 3 RB bits 4 combinations are required.

Table I RB Encoding

X^+	X^-	RB Digit
0	0	0
0	1	$\bar{1}$
1	0	1
1	1	0

Example : A 3digit NB number 011 can be represented as 10 1 or 1 11 or 011 in the RB number system. The number 101 can be encoded as 100001 as per the scheme in table I. The following expression (9) can be used to find the sum of 2 NB numbers

$$x+y = x-(-y) = x-(\bar{y}+1) = (x-\bar{y})-1 = (x, \bar{y})-1$$

2.5 Adding two RB numbers:

Two numbers can be added without the need of propagating the carry in the following steps. If Xi is the augend digit and Yi is the addend digit then Xi+Yi = 2 Ci +Si where Ci € {1 ,0,1} is the intermediate carry digit and Si € { 1,0,1} is the intermediate sum digit. 2Ci means 'Ci generated in this process' is shifted to the left by one position. The intermediate sum Si and carry Ci are computed by following the rules given in table II. This addition must be performed considering the values added in the adjacent lower order positions.

Table Ii. Rules For Computation of Carry Propagation Free Addition

(x_i)	(y_i)	(x_{i-1},y_{i-1})	(S_i)	(C_i)
1	1	Any value	0	1
1	0	Both non-negative	$\bar{1}$	1
0	1	Other wise	1	0
0	0	Any value	0	0
1	$\bar{1}$			
$\bar{1}$	1			
0	$\bar{1}$	Both non-negative	$\bar{1}$	0
$\bar{1}$	0	Other wise	1	$\bar{1}$
$\bar{1}$	$\bar{1}$	Any value	0	$\bar{1}$

The sum digits Zi € {1 , 0, 1} are obtained by adding the Si and Ci starting from lower order position. This addition does not generate carry. Thus the RB addition can be performed using a combinational circuit in a constant time irrespective of size of operands. F. Conversion from RB to NB: Though it is advantages to deal with RB addition, we deal with only the NB number system in the real world. Hence the output obtained in RB form has to be converted into the NB form [7, 8]. This conversion is performed first by encoding the NB numbers to RB form separating the X+ and X- parts as given in table I. X+ is the positive digit of X and X- is the negative digit of X. The following example illustrates the conversion of RB number to NB number
.

2.6. Generation of partial products in Radix 256 Booth Encoding:

When the multiplier is encoded to get Dis, these Dis take any one of the value among 257 values i.e., between -128, -127, ……..,-1,0,1…….127,128. In this method all the Di.Bs are precomputed and the required Di.Bs are selected from among them. The algorithm is implemented in the following stages. RB adder v. RB to NB converter

2.6.1 Pre computer

The pre computer computes, from the input 'B', the following values using the algorithm shown in the pre computer block of Fig. 1. These values are called FDMPPs (Fundamental Digit Multiplied Partial Products). These are B, -1B, 3B, -3B, 5B, -5B, 7B and -7B. From these FDMPPs all the other SGDMPPs (Secondary Digit Multiplied Partial Products) and TGDMPPs (Tertiary Digit Multiplied Partial Products) can be derived by proper shifting.

2.6.2 Control signal generator

The control signal generator takes the multiplier 'A' as the input and generates 4 sets of control signals called 1) Sdigit0, 2) Tdigit0, 3) Sdigit1 and 4) Tdigit1. Out of these, the Sdigit0 and Sdigit1 are 4 bit control signals each and the Tdigit0 and Tdigit1 are 8 bit control signals. The algorithm for generation of control signals is Adding the SGDMPP0 and TGDMPP0 gives the first 32 bit partial product, PP0. The 16 bit multiplier A is concatenated with a zero on the right to get a 17 digit value. This 17 digit number is divided into two overlapping 9- digit groups which are the Dis. They are D1 and D0.

D1 = A15……A7; D0=A7…..A0 A-1 (11)

The partial products corresponding to these D1 and D0 are to be selected. • From each of the Dis, two temporary variable groups are generated using the set of eqns (12 to 15).

Sgroup0 = 4*A2 + 2*A1+1*A0+1*A-1 (12)

Tgroup0 = 16 * (4 * A6 + 2 *A5 + 1 *A4 + 1 * A3) (13)

Sgroup1=4*A10 +2*A9+ 1*A8 +1*A7 (14)

Tgroup1=16*(4 * A14 + 2 *A13 + 1 *A12 + 1 *A11) (15)

Sgroup2 =4 * A18 + 2 *A17 + 1 *A16 + 1 *A15 (16)

Tgroup2 = 16 * (4 * A22 + 2 *A21 + 1 *A20 + 1 * A19) (17)

Sgroup3 = 4 * A26 + 2 *A25 + 1 * A24 + 1 *A23 (18)

Tgroup3 = 16 * (4 * A30 + 2 *A29 + 1 *A28 + 1 *A27) (19)

Sgroup4 = 4 * A34 + 2 *A33 + 1 *A32 + 1 *A31 (20)

Tgroup4 = 16 * (4 * A38 + 2 *A37 + 1 *A36 + 1 * A35) (21)

From the Sgrop0, Tgroup0, Sgroup1 and Tgroup1, the Sdigit0, Tdigit0, Sdigit1 and Tdigit1 are derived as per the flow chart in Fig. 2. • These digit signals are the control signals used for selecting the proper Di.B.

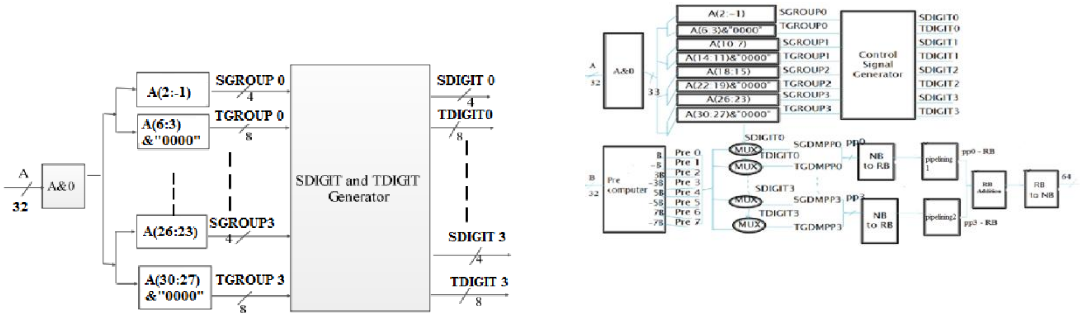


Fig. 1. Implementation block diagram of Radix-256 Booth Encoding and RB addition

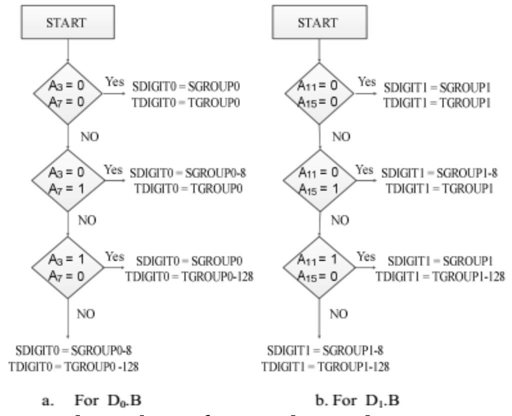


Fig. 2. Flow Chart of Control Signal Generation

2.6.3. Selector

The selector consists of 4 multiplexers, the pre computer outputs are fed to these 4 multiplexers in parallel.The outputs of these multiplexers are the SGDMPP0, SGDMPP1, TGDMPP0 and TGDMPP1 which are all 31 bit values. SGDMPP's means S-Group Digit Multiplied Partial Products, TGDMPP's are T-Group Digit Multiplied Partial Products. The SGDMPPs and TGDMPPs are obtained from the precomputer outputs by suitably shifting them to the left by suitable number of digits. The selection of as to what SGDMPP or TGDMPP is to be obtained is determined by the select signals Sdigit0, Tdigit0, Sdigit1, and Tdigit1.Adding the SGDMPP0 and TGDMPP0 gives the first 32 bit partial product, PP0.Adding the SGDMPP1 and TGDMPP1 and shifting the result to the left by 8 positions gives the second partial product PP1. These partial products are in NB form. Adding these partial products in NB form takes too many numbers of units of delay if added with ripple carry adder because it is a carry propagate addition. To avoid carry propagation during the addition, an RB adder is used

.

2.6.4 Rb Adder

RB addition is a carry propagate free addition. With two units of delay, all the bits of the partial products can be added in parallel. The 32 bit partial products, PP0 and PP1 which are in NB form are first converted into 64 bit partial products in RB (Redundant Binary) form by encoding as shown in table I.These partial products PP0_RB and PP1_RB are added in an RB adder to give the 66 bit output which is the RB output of the multiplier.

2.6.5. Rb To Nb Converter

The RB output from the above RB adder is separated into positive digits and negative digits. The positive digits are added to the 2's complement of negative digits using equation (10) to get the 33 bit NB output. This 33 bit output of the RB-NB converter is the multiplier output

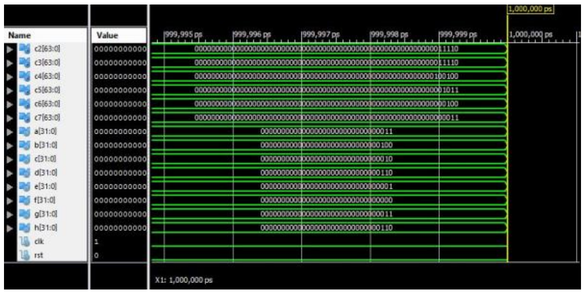
.

2.6.6. Pipelining Method

Pipelining method is used to transform a sequential circuit to that circuit which has high clock speed or sample speed. It reduces critical path which will increase the sample speed or clock speed thus speed of opetation is improved.

III. EXPERIMENT AND RESULT

The project requires Xilinx ISE 14.7, spartan3E , ModelSim: Xilinx Edition III v6.2g and verilog HDL.



(a)simulation result of proposed method

Table -1 Experiment Result

Design	Area	Delay
Existing design	1006 lut's	38.334ns
Proposed Design	6134 lut's	34.892 ns

Design	Number of bits	Number of partial products
Existing design	16	2
Proposed design	32	4

IV. CONCLUSION

The convolution algorithms are implemented with multiplier using radix- 256 booth encoding and RB addition. It has been found that there is a considerable decrease in the delay with the proposed algorithm with a little increase in hardware

V. REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Sandeepsani, Surabhi jain, "High Speed Convolution and Deconvolution Algorithm(Based on Ancient Indian Vedic Mathematics), 11thIEEE International conference on Electrical Engineering / Electronics, Computer, Tele communication and Information Technology (ECTICON), 2014.

[2] Itawadiya, Akhalesh K., et al. "Design a DSP operations using vedic mathematics" , International Conference on Communications and Signal Processing (ICCSP), IEEE, 2013.

[3] S.K.Sahoo, Srinivasa Reddy K, "A High Speed FIR filter Architecture based on Novel Higher Radix Algorithm", 25th IEEE International Conference on VLSI Design, 2012.

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

[5] Lomte, Rashmi K., P. C. Bhaskar, "High Speed Convolution and Deconvolution Using Urdhva Triyagbyam", VLSI (ISVLSI) IEEE Computer Society Annual Symposium , 2011.

[6] Avizienis, "Signed-digit number representation for fast parallel arithmetic," IRE Trans. Electron. Computer., vol.EC-10, pp. 389-400,mSept.1961.

[7] S. Kuninobu, T. Nishiyama, T. Edamatsu, T. Taniguchi, and N. Takagi,"Design of high speed MOS multiplier and divider using redundant binary representation," In Proc. 8th Symp. Computer Arithmetic, Italy, pp. 80-86, May 1987.

[8] Yum Kim, Bang-Sup Song, John Grosspietsch, and Steven F. Gilling , "A carry-free 54b x 54b multiplier using equivalent bit conversion algorithm, " IEEE J. Of Solid State Circuits, Vol. 36, No.10, pp. 1538-1544, Oct. 2001.

[9] A.D. Booth, "A signed binary multiplication technique," Quarterly J. Mech. Appl. Math, Vol. 4, Part 2, pp. 236-240, 1951

[10] Pierre, John W, "A novel method for calculating the convolution sum of two finite length sequences.", IEEE Transactions on Education, 1996, pp. 77-80.

[11] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja, "Vedic Mathematics", Motilal Banarsidass, New Delhi, India, 1994.

[12] J. G. Proakis and D. G. Manolakis, "Digital Signal Processing: Principles, Algorithm, and Applications", 2nd Edition, New York Macmillan, 1992.

[13] FilixAsaJyothi, V. koteswara Rao, "An efficient architecture of the radix based FIR filter design", International journal of reviews on recent electronics and computer sceience (IJRRECS), August 2013, pp.300-305.