International Journal For Technological Research In Engineering
Volume 4, Issue 4, December-2016 *ISSN (Online): 2347 - 4718*

DESIGN OF IIR FILTER USING QSD NUMBER SYSTEM

A.Jayasurya[1], P.Rama Deepti[2]
[1]M.Tech, [2]Technical staff (seer akademi),
Dept of ECE, JNTUA College of Engineering, Anantapuramu, A.P, India.

Abstract: This paper presents a 3rd order IIR filter designed using a QSD number system and is compared with an another 3rd order IIR filter designed using Booth Multiplier (BM) and Ripple Carry Adder (RCA). The complexity of digital filters is dominated by the number of multiplications, many works have focused on minimizing the complexity of multiplier blocks. The problem of designing filters has received a great attention during the last decade, as the filters are suffering from a large number of multiplications, leading to excessive area and power consumption even if implemented in full custom integrated circuits. Many works have focused on replacing multiplications by decomposing them into simple operations such as addition, subtraction and shift and reducing the number of simple operations. Many algorithms have been proposed to make the multiplier block as simple as possible. Here QSD number system design is used to minimize the complexity of multiplier blocks and apply them to infinite impulse response (IIR) filter. Experimental results show that the proposed algorithms can reduce the area and delay of IIR filter. Synthesis and simulation is achieved through Synopsys 90nm technology.

Keywords: Quaternary Signed Digit (QSD) number system, Booth Multiplier, Ripple Carry Adder, Infinite-Impulse Response (IIR) filter.

I. INTRODUCTION

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range. The main purpose of this filter design is to minimize the number of additions/subtractions in order to reduce the delay. By using the binary number system, the computation speed is limited by formation and propagation of carry especially as the number of bits increases. Using a quaternary Signed Digit number system one may perform carry free addition, borrow free subtraction and multiplication. However the QSD number system requires a different set of prime modulo based logic elements for each arithmetic operation. A carry free arithmetic operation can be achieved using a higher radix number system such as Quaternary Signed Digit Number System.

II. IIR FILTER USING QSD NUMBER SYSTEM

3RD order IIR Filter

Infinite impulse response (IIR) is a property applying to many linear time-invariant systems. Common examples of linear time-invariant systems are most electronic and digital filters. Systems with this property are known as IIR systems or IIR filters, and are distinguished by having an impulse response which does not become exactly zero past a certain point.

The 3rd order filter is implemented using QSD adder for addition and QSD multiplier for multiplication [1]. Fig 1 used D flip flops to design the delay elements.

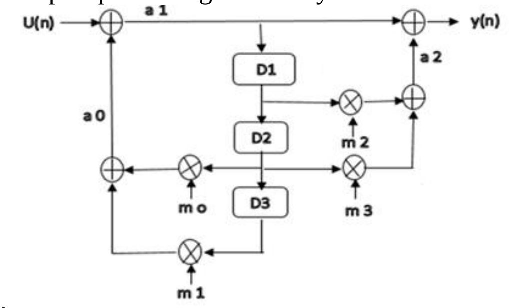


Fig.1. 3rd order IIR filter using QSD adder and QSD multiplier

Technique for Conversion of Binary Number to QSD Number 1-digit QSD can be represented by one 3bit binary equipollent as follows:

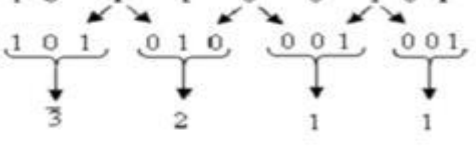
$$\overline{3} = 101$$
$$\overline{2} = 110$$
$$\overline{1} = 111$$
$$0 = 000$$
$$1 = 001$$
$$2 = 010$$
$$3 = 011$$

In order to convert *n*-bit binary data into its equivalent *q*-digit QSD data, we have to convert this *n*-bit binary data into *3q*-bit binary data. To achieve the target, we have to split the 3rd, 5th, 7th bit…. i.e. odd bit (from the LSB to MSB) into two portions. But we cannot split the MSB. To convert the given binary number into QSD number we have to follow these two set of rules:

- If the odd bit is 1 then, it is split into 1 & 0.
- If the odd bit is 0 then, it is split into 0 & 0.

Example: Let (-155)10 = (101100101)2 have be converted to its equipollent QSD.

(101100101)2 „is 9-bit binary data. Its 3rd bit is 1, 5th bit is 0 and 7th bit is 1.we can verbally express that, its QSD equipollent is of 4-digit. Hence according to the splitting technique verbally expressed above the binary data can be expressed as follows:



So the equivalent of $(101100101)_2$ is $(\bar{3}211)_4$

Table I The intermediate carry and sum between -6 to 6

SUM	QSD REPRESENTED NUMBER	QSD CODED NUMBER
-6	$\bar{2}2,\bar{1}\bar{2}$	$\bar{1}\bar{2}$
-5	$\bar{2}3,\bar{1}\bar{1}$	$\bar{1}\bar{1}$
-4	$\bar{1}0$	$\bar{1}0$
-3	$\bar{1}1,0\bar{3}$	$\bar{1}1$
-2	$\bar{1}2,0\bar{2}$	$0\bar{2}$
-1	$\bar{1}3,0\bar{1}$	$0\bar{1}$
0	00	00
1	$01,1\bar{3}$	01
2	$02,1\bar{2}$	02
3	$03,1\bar{1}$	$1\bar{1}$
4	$11,2\bar{3}$	10
5	$12,2\bar{2}$	11
6		12

QSD Adder

In QSD number system carry propagation chain are eliminated which reduce the computation time substantially. As range of QSD number is from -3 to 3, the addition result of two QSD numbers varies from -6 to +6. The decimal numbers in the range of -3 to +3 are represented by one digit QSD number. In the Table I the QSD representation of the sum from -6 to 6 is calculated, from that QSD representation the required QSD coded number is selected based on the requirement.

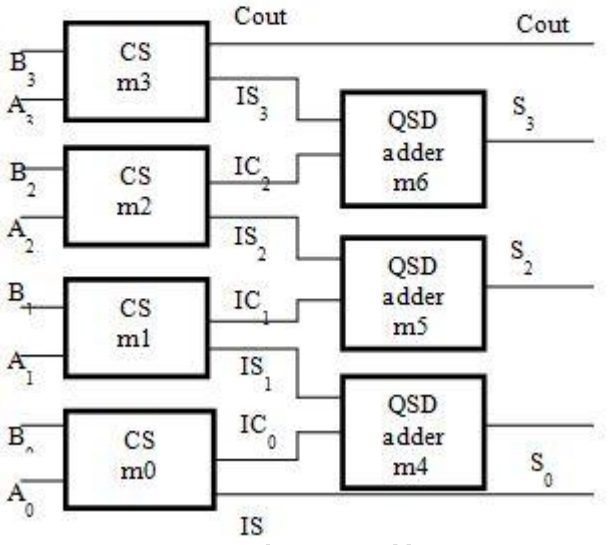


Fig.2. 4-digit QSD adder.

In figure 2 the addition of two QSD numbers is done in two stages. First stage of adder generates intermediate carry and intermediate sum from the input digits. Second stage of adder adds the intermediate sum of current digit with the intermediate carry of lower significant digit. To remove the further rippling of carry there are two rules to perform QSD addition in two steps:

Rule 1: First rule states that the magnitude of the intermediate sum must be less than or equal to 2 i.e., it should be in the range of -2 to +2.

Rule 1: Second rule states that the magnitude of the intermediate carry must be less than or equal to 1 i.e., it should be in the range of -1 to +1.

We can see in the first column of Table I that some numbers have multiple representations, but only those that meet the above defined two rules are chosen.

Table .2. The conversion between the inputs and outputs of the Intermediate carry and Intermediate sum

INPUT				OUTPUT				
QSD		Binary		Decimal	QSD		Binary	
A_i	B_i	A_i	B_i	Sum	C_i	S_i	C_i	S_i
3	3	011	011	6	1	2	001	010
3	2	011	010	5	1	1	001	001
2	3	010	011	5	1	1	001	001
3	1	011	001	4	1	0	001	000
1	3	001	011	4	1	0	001	000
2	2	010	010	4	1	0	001	000
1	2	001	010	3	1	-1	001	111
2	1	010	001	3	1	-1	001	111
3	0	011	000	3	1	-1	001	111
0	3	000	011	3	1	-1	001	111
1	1	001	001	2	0	2	000	010
0	2	000	010	2	0	2	000	010
2	0	010	000	2	0	2	000	010
3	-1	011	111	2	0	2	000	010
-1	3	111	011	2	0	2	000	010
0	1	000	001	1	0	1	000	001
1	0	001	000	1	0	1	000	001
2	-1	010	111	1	0	1	000	001
-1	2	111	010	1	0	1	000	001
3	-2	011	110	1	0	1	000	001
-2	3	110	011	1	0	1	000	001
0	0	000	000	0	0	0	000	000
1	-1	001	111	0	0	0	000	000
-1	1	111	001	0	0	0	000	000
[illegible]	[illegible]	[illegible]	110	0	0	0	000	000
-2	2	110	010	0	0	0	000	000
-3	3	101	011	0	0	0	000	000
3	-3	011	101	0	0	0	000	000
0	-1	000	111	-1	0	-1	000	111
-1	0	111	000	-1	0	-1	000	111
-2	1	110	001	-1	0	-1	000	111
1	-2	001	110	-1	0	-1	000	111
-3	2	101	010	-1	0	-1	000	111
2	-3	010	101	-1	0	-1	000	111
-1	-1	111	111	-2	0	-2	000	110
0	-2	000	110	-2	0	-2	000	110
-2	0	110	000	-2	0	-2	000	110
-3	1	101	001	-2	0	-2	000	110
1	-3	001	101	-2	0	-2	000	110
-1	-2	111	110	-3	-1	1	111	001
-2	-1	110	111	-3	-1	1	111	001
-3	0	101	000	-3	-1	1	111	001
0	-3	000	101	-3	-1	1	111	001
-3	-1	101	111	-4	-1	0	111	000
-1	-3	111	101	-4	-1	0	111	000
-2	-2	110	110	-4	-1	0	111	000
-3	-2	101	110	-5	-1	-1	111	111
-2	-3	110	101	-5	-1	-1	111	111
-3	-3	101	101	-6	-1	-2	111	110

QSD Multiplier

There are generally two methods for a multiplication operation: parallel and iterative. QSD multiplication can be implemented in both ways, requiring a QSD partial product generator and QSD adder as basic components. A partial product, Mi, is a result of multiplication between an n-digit input, An-1 to A0, with a single digit input, Bi, where i = 0 to n-1.

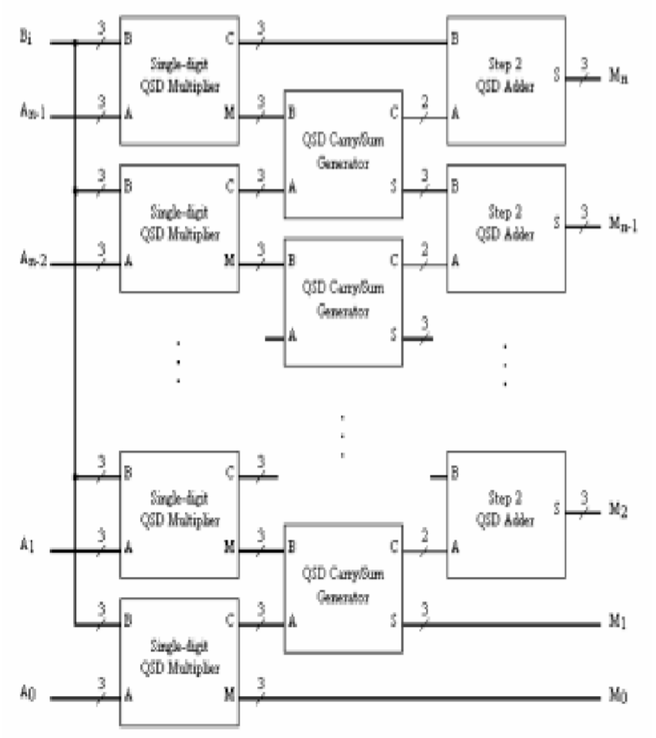


Fig.2. n-digit QSD Partial Product Generator

According to these two rules the intermediate multiplier and intermediate carry from the first step QSD multiplier can have the range of -9 to +9. When the second step QSD adder adds the intermediate multiplier of current digit, which is in the range of -2 to +2, with the intermediate carry of lower significant digit, which is in the range of -1 to +1, the multiplier result cannot be greater than 3 i.e., it will be in the range of -3 to +3. The multiplier result in this range can be represented by a single digit QSD number: hence no further carry is required.

We can see in the first column of Table IV that some numbers have multiple representations, but only those that meet the above defined two rules are chosen. The chosen intermediate carry and intermediate multiplier are listed in the last column of Table IV as the QSD coded number.

An nxn-digit QSD multiplication requires n partial product terms. In an iterative implementation, a 2n digit QSD adder is utilized to perform integrate-shift operations between the partial product generator and the accumulator. In contrast, a parallel implementation requires n partial product circuits and n-1 QSD adder unit.

Table IV: QSD representation of a single-digit multiplication output

PRODUCT	QSD REPRESENTED NUMBER	QSD CODED NUMBER
-9	$\bar{2}\bar{1}$, $\bar{3}3$	$\bar{2}\bar{1}$
-6	$\bar{2}2$, $\bar{1}\bar{2}$	$\bar{1}\bar{2}$
-4	$\bar{1}0$	$\bar{1}0$
-3	$\bar{1}1$,$0\bar{3}$	$\bar{1}1$
-2	$\bar{1}2$,$0\bar{2}$	$0\bar{2}$
-1	$\bar{1}3$,$0\bar{1}$	$0\bar{1}$
0	00	00
1	01,$1\bar{3}$	01
2	02,$1\bar{2}$	02
3	03,$1\bar{1}$	$1\bar{1}$
4	10	10
6	12,$2\bar{2}$	12
9	21,$3\bar{3}$	21

III. IIR FILTER USING BOOTH MULTIPLIER AND RIPPLE CARRY ADDER

The 3rd order IIR filter is designed using Booth Multiplier (BM) for multiplication and Ripple Carry Adder (RCA) for addition. A Ripple Carry Adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. Booth Multiplier is used to reduce the number of partial products generated in a multiplication process.

MODIFIED BOOTH Multiplier

The modified Booth algorithm reduces the number of partial products by half in the first step. The modified Booth encoding (MBE) scheme is proposed. It is known as the most efficient Booth encoding and decoding scheme. To multiply X by Y using the modified Booth algorithm starts from grouping Y by three bits and encoding into one of {-2, -1, 0, 1, 2}. Table V shows the rules to generate the encoded signals by MBE scheme. The Booth decoder generates the partial products using the encoded signals as shown in Fig.6.

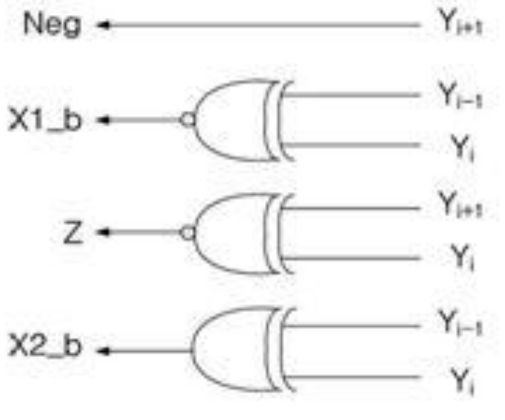


Fig 5. Booth Encoder

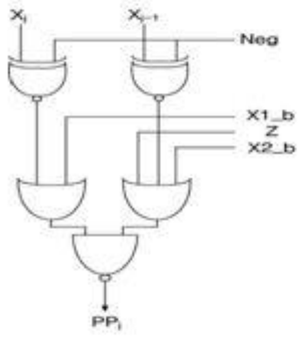


Fig 6. Booth Decoder

To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Figure.7. shows the grouping of bits from the multiplier term for using in the modified booth encoding.

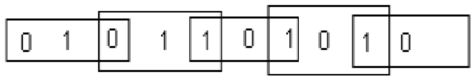
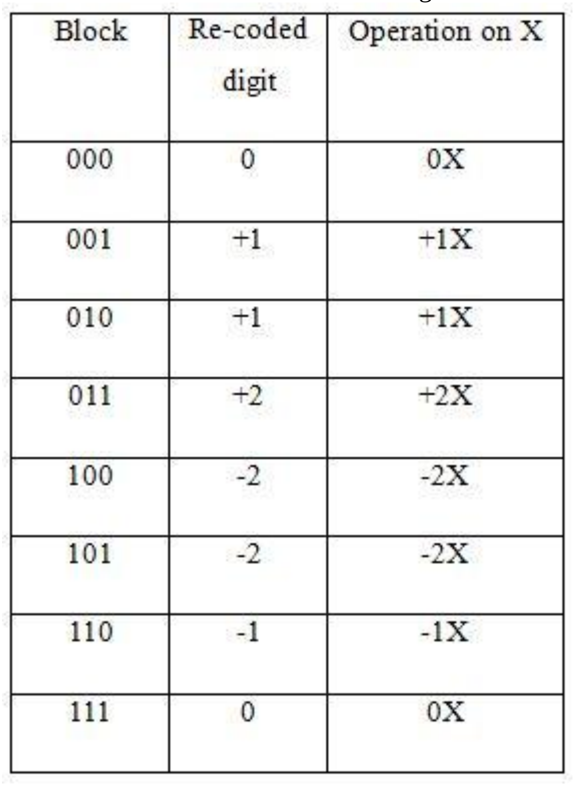


Figure 7. Grouping of bits from the multiplier

The PP generator generates five candidates of the partial products, i.e., {-2A,-A, 0, A, 2A}. These are then selected according to the Booth encoding results of the operand B. Booth encoder and partial product generator affects the efficiency of the partial product generation. The number of partial products that can be saved by this stage impacts the cost, performance, and power consumption of the multiplier as a whole. Figure 8 shows the generated partial products and sign extension scheme of the 8-bit modified Booth multiplier. The partial products generated by the modified Booth algorithm are added in parallel using carry save Adder.

Table V. Booth re-coded digit



Block	Re-coded digit	Operation on X
000	0	0X
001	+1	+1X
010	+1	+1X
011	+2	+2X
100	-2	-2X
101	-2	-2X
110	-1	-1X
111	0	0X

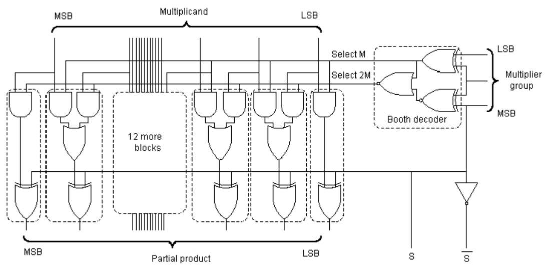


Figure 8. Partial Product Generator

The next step involves the addition of these partial products in the fastest manner possible. Speeding up the addition of partial products required faster adders. For adding these partial products carry save adder (CSA) is used.

Ripple Carry Adder

Ripple Carry Adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into the next stage. Propagation delay is time elapsed between the application of an input and occurrence of the corresponding output. The ripple carry adder is designed using Full Adders.

IV. SIMULATION RESULTS AND PERFORMANCE COMPARISONS

The programming has been done in Verilog HDL and it is simulated using Synopsys 90nm technology. The simulation and synthesis results of both the filters are compared. The simulation results of the two filters are shown in figure 9, 10. The schematic of both the filters are shown in figure 11, 12. The area, and time comparisons of both the Filter using different multipliers and adders in the filter design are presented in Table VI. The simulated result for 8-bit 3rd order IIR Filter is shown below.

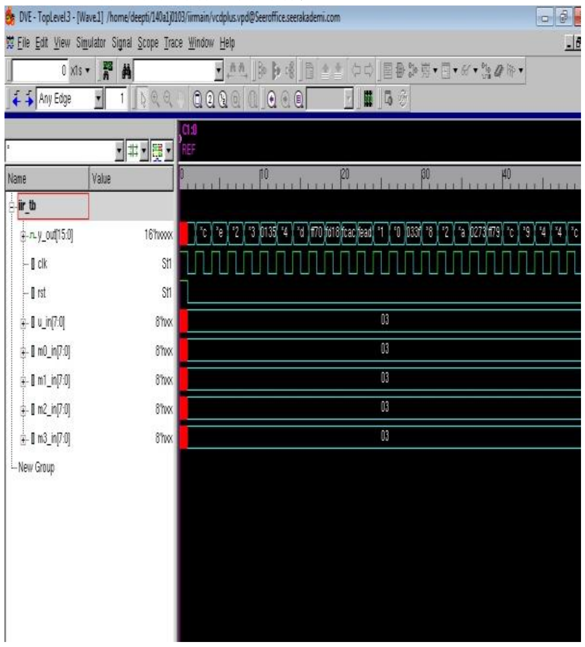


Fig.9. Simulation results of 3rd order IIR filter designed using QSD Number System

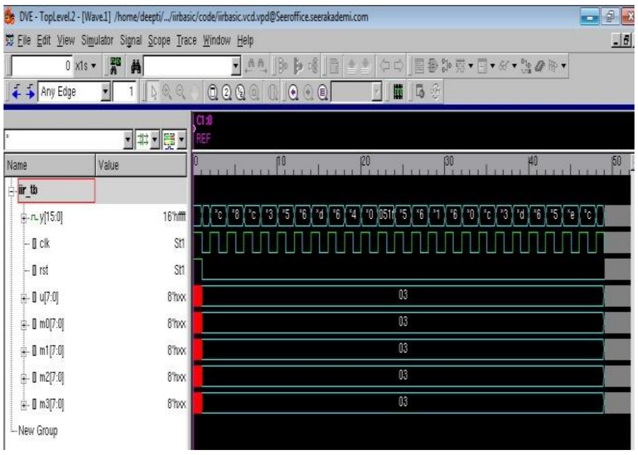

DVE - TopLevel.2 - [Wave.1] /home/deepti/.../iirbasic/code/iirbasic.vcd.vpd@Seeroffice.seerakademi.com
File Edit View Simulator Signal Scope Trace Window Help
Name
Value
iir_tb
y[15:0]
clk
rst
u[7:0]
m0[7:0]
m1[7:0]
m2[7:0]
m3[7:0]
New Group
03

Fig.10. Simulation results of 3rd order IIR filter designed using Booth Multiplier and Ripple Carry Adder

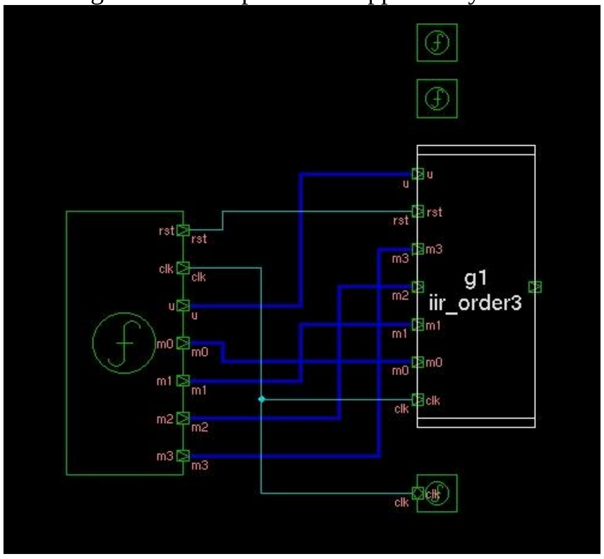


Fig.11.Schematic of 3rd order IIR filter designed using QSD Number System

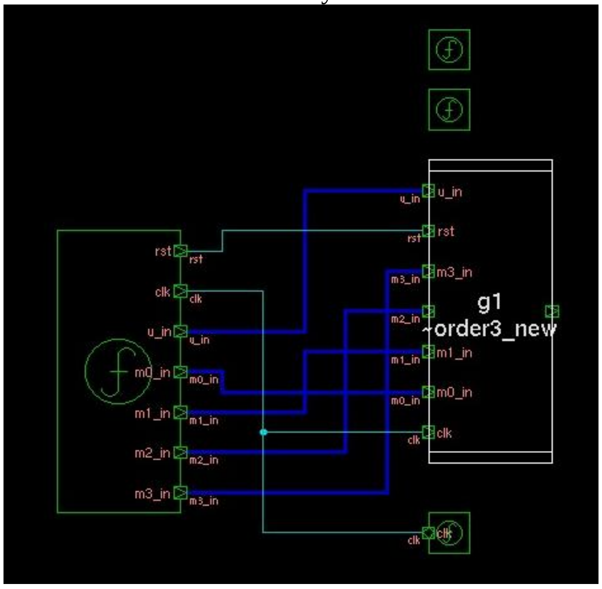
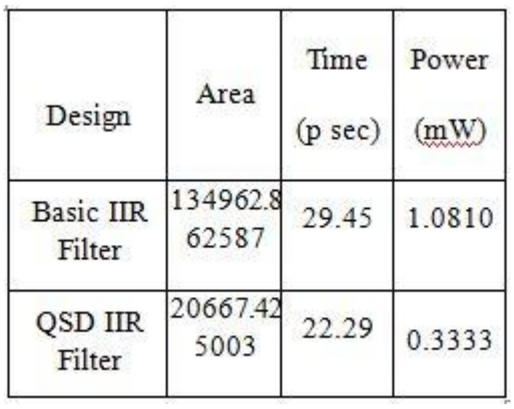


Fig.12.Schematic of 3rd order IIR filter designed using Booth Multiplier and Ripple Carry Adder

TABLE VI. Area, time and Power comparisons of both the 3^{rd} order IIR Filters



Design	Area	Time (p sec)	Power (mW)
Basic IIR Filter	134962.862587	29.45	1.0810
QSD IIR Filter	20667.425003	22.29	0.3333

V. CONCLUSION

The main purpose of the filter is to occupy less memory so while designing an IIR filter if we use QSD adder and multiplier it gives better area and power results. The area of IIR filter designed using QSD number system occupied 25% less space than the filter designed using ripple carry adder and booth multiplier. The ripple carry adder has carry propagation delay so the filter designed using QSD number system has less delay. The filter designed using QSD number system has better area and time result compared to the basic filter.

REFERENCES

[1]. International Journal of Ethics in Engineering & Management Education “VLSI Implementation of Fast Addition Using Quaternary Signed Digit Number System”.

[2]. “Obfuscating DSP Circuits via High-Level Transformations”, Free Scale Semiconductor – DDR SDRAM Layout Considerations, Rev.1, Aug 2004 Hynix – DDR SDRAM Device Operation, Rev 1.1, April 2006.

[3]. Survey Paper of “Digital IIR Filter Design” by International Journal of Advanced Research in Computer Science and Software Engineering Er.Daljit Singh Bajwa, Er. Karamjeet Singh, Navpreet Kaur Chahal ECE & BBSBEC, Fatehgarh Sahib India.

[4]. “FPGA Implementation of Fast Arithmetic Unit Based on QSD” Shrikesh A. Dakhane, A. M. Shah Department of Electronics & Telecommunication, Government College of Engineering, Amravati, India using the vertex select I/O Resource, Application note, 1999. (Xilinx Inc., XAPP133).

[5]. International Journal of Scientific and Research Publications, Volume 4, Issue 5, May 2014 “Study of Combinational and Booth Multiplier” Department Of Computer Science & Engineering Shri Ram College of Engineering & Management, Palwal, Haryana, INDIA.

Ms.A.Jayasurya received B.Tech degree in Electronics & Communication Engineering from JNTUK and presently pursuing M.Tech degree in VLSI in JNTUA College of Engineering, Anantapuram. My research interests include VLSI Design, Low Power VLSI and DSP Architectures.

Ms. P.Rama deepti received her B Tech electronics and communication Sri Padmavati mahila visvavidyalayam, tirupati in 2010. M Tech VLSI design and embedded system Dayanandasagar college of engineering, VTU Bangalore in 2013. Internship and worked as trainer at smart chip pvt ltd, Bangalore from 2014 jan to 2014 dec. Working as Member of technical staff, Seer Akademi pvt ltd., Hyderabad jan 2015 to dec 2016. Her areas of interests are Verilog, System verification and Physical Design.