Efficient Design and FPGA Implementation of Digital Filter for Audio Application

Gopal S. Gawande
Asso. Professor, Dept. of E&TC
SSGM College of Engineering, Shegaon,
Maharashtra State, India
gopalgawande@rediffmail.com

Dr. K.B. Khanchandani
Professor, Dept. of E&TC
SSGM College of Engineering, Shegaon,
Maharashtra State, India
kbkhanchandani@rediffmail.com

***Abstract*—Digital Filters are important elements in Digital Signal Processing (DSP). Major factors influencing in the designing an efficient digital filter are computational requirements, memory and finite word length effects. In order to meet these requirements, the order of the digital filter must be kept as small as possible by selecting appropriate filter design method. For Simulating the performance of the digital filter, the filter coefficients are specified with floating point data type but for implementation on Field Programmable Gate Array(FPGA) they must be represented with fixed point data type to minimize cost and power consumption by minimizing the word length. A minimum order Finite Impulse Response (FIR) digital filter is designed for filtering noise from audio signal. It is synthesized using Xilinx ISE9.1 for Spartan 3E-1200 FPGA board using Factored Canonic Signed Digit (FCSD) and Distributed Arithmetic architectures for optimal utilization of FPGA resources.**

Keywords- Digital Filter, FPGA, Filter coefficients, FCSD architecture, Distributed Arithmetic

I. INTRODUCTION

The developments in electronic technology are taking place at a mind boggling speed. Digital Signal Processing (DSP) finds applications in almost all walks of life. Now a day's multimedia requires the processing of the signal to be very fast with less power consumption. In multimedia audio signal processing is an important area. Many times we need to remove noise from the audio signals. Therefore Digital Filter is to be designed for removing noise. Designing Digital Filter is the process of calculating appropriate filter coefficients and order of the Digital Filter. The designed filter can be implemented on Field Programmable Gate Array (FPGA). This implementation must meet the sampling rates of the corrupted audio signal.

Digital Filter coefficients can be represented by fixed and floating point formats. Deciding whether fixed- or floating point is more appropriate for the given problem must be done carefully. In general it can be assumed that fixed point implementations have higher speed and lower cost, while floating point implementations have higher dynamic range and no need of scaling which may be attractive for more complicated algorithms [1][2][3]. The research is going on the optimized digital filter in terms of power, area and speed. There are various digital filter architectural optimization approaches like pipelining, parallel processing, Distributed Arithmetic, folding etc. used to generate efficient digital filter architecture.

In this paper, a Finite Impulse Response (FIR) filter is designed using Filter Design and Analysis (FDA) Tool available in MATLAB. Its performance is verified using simulink model and hardware related issues are handled by the Xilinx System Generator blocks. Finally, VHDL codes are generated and synthesized for Factored Canonic Signed Digit and Distributed Arithmetic (DA) architecture using Xilinx ISE9.1 for Spartan 3E-1200 FPGA board.

II. RELETED WORK

While implementing a Digital Filter on FPGA, the biggest challenge is to achieve specified performance at minimum cost. One way to minimize the cost is to reduce the wordlength. The effects of finite wordlength are discussed in [2]. The effects of using finite wordlength can be minimized by choosing higher order filters in cascaded or parallel form.

FIR filter can be designed using either windowing or optimal method. For the given specifications, the various methods are analyzed and it is concluded that equiripple filter design found to be the most suitable and optimized method to meet the given specifications [7].

The techniques which are used to achieve low power consumption in VLSI-DSP applications span a wide range from algorithm and architecture levels to logic, circuits & device levels. In [8] several approaches like pipelining, parallel processing, Retiming etc. are reviewed.

The algorithm presented in this paper can be used to determine the number of bits needed for each coefficient to implement FIR filter with variable precision coefficient. Furthermore CSD representation leads to compensation for the non uniform distribution of the CSD coefficient set [9].

Digital Filter coefficients can be represented using signed 2's complement number system. But reducing number of 1's in filter coefficient reduces the power significantly. Therefore CSD format can be used for filter representing filter coefficient. It reduces number of 1's in the filter coefficient thereby reducing complexity in the computation also. Moreover it is observed that there is sufficient scope for more work on complexity reduction in FPGA implementations especially for higher order filters [10] [11].

The main portion of DA based computation is Look Up Table (LUT) that stores the pre-computed values & can be read out easily, which makes DA based computation well suited for FPGA realization. Designs of pipelined architectures are derived for high speed implementation of FIR filter. Filters implemented with Q8.7 representation

978-1-4799-6892-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ICCUBEA.2015.180

results in nearly 53% lesser slice LUTs when compared to Q16.14 representation [12].

The work in [13] presents the design and implementation of serial and parallel distributed Algorithm for FIR filter. The results of the implementation are analyzed in terms of area & speed. For high order filters, speed of parallel DA architecture becomes 3 times faster than that of conventional FIR filter.

III. Designing Cost Efficient Digital Filter

Digital Filters are of two types namely Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). FIR filters are preferably used because of stability. In general, an FIR filter is described by the convolution sum given by (1):

$$y(n) = \sum_{K=0}^{M-1} b_k\, x(n-k) \qquad (1)$$

Where b_k represent FIR filter coefficient. The realization of equation (1) for M-tap FIR filter is as shown in the fig 1. It uses multiplier, adder and delay blocks.

The simplest technique used to design FIR filter is windowing technique. This technique is based on designing a filter using well known frequency domain transition function called window. The use of windows often involves choosing the lesser of two evils. Some windows such as rectangular, yield fast roll off in the frequency domain but have limited attention in stopband along with poor group delay characteristics. Other windows like Blackman have better stopband attenuation and group delay but a wide transition band.

The designed frequency response of any Digital Filter is periodic in frequency and can be represented by equation (2),

$$H_d e^{(jw)} \sum_{n=-\infty}^{\infty} h_d(n)\, e^{-j\omega n} \qquad (2)$$

where, $h_d(n) = \frac{1}{2\pi}\int_0^{2\pi} H(e^{j\omega}) e^{-j\omega n} dw$

But the resultant $h_d(n)$ for ideal Low pass filter is of infinite duration. So to make it finite, we need to multiply this finite $h_d(n)$ by a time limited discrete window sequence w(n). The resultant sequence would be then of finite duration.

The order of various windows is calculated for the given specifications and it has been found that Kaiser window gives minimum order compare to other window functions [7]. The other category of FIR filter design is the optimal method. Under this category, the Equiripple FIR design uses the Remez/Parks McLellan algorithm to design a linear phase FIR filter which just meets the specifications without over performing. Many of the window method designs actually perform better as you move further away from the passband. That means they use more filter coefficients than they need. The Remez/Parks McLellan method performs just to meet the specifications at the lowest possible order. Remez/Parks McLellan designs have equal ripple up to the specification but no more in both passband and stopband.

For removing noise from a noisy audio signal, an FIR Band Pass Equiripple filter is designed as per the specifications given in Table I. The frequency response of the designed filter is shown in Fig 2. The minimum order is obtained as 91 to meet the specifications given in Table I.

IV. Data Formats

A. Quantization to Q16.14 format

Using the FDA Tool, the designed filter is quantized to a Q16.14 fixed point numeric representation format. This means that the total word length is 16 bits, out of which the most significant bit (MSB) is used to represent the sign of the number, the next bit for representing the non-fractional magnitude and the remaining the fractional magnitude. Table II illustrates this distribution in detail.

TABLE I. FIR Filter Specifications

Filter performance parameter	Value
F stop1	400Hz
F pass1	700Hz
F pass2	2000Hz
F stop2	2300Hz
Density Factor	20
Passband attenuation	1dB
Stopband attenuation	50dB

TABLE II. Numeric Representation Format Q16.14- Bit Distribution

MSB (15)	Bit (14)	Bit (13)	Bit (12)	Bit (11)	Bit (10)	Bit (9)	Bit (8)	Bit (7)	Bit (6)	Bit (5)	Bit (4)	Bit (3)	Bit (2)	Bit (1)	LSB (0)
S	M	M_F	M_F	M_F	M_F	M_F	M_F	M_F	M_F	M_F	M_F	M_F	M_F	M_F	M_F

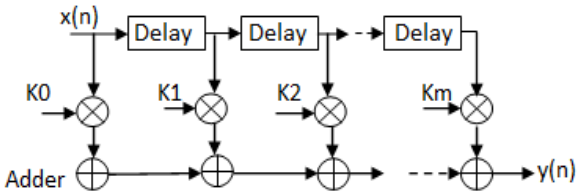

Figure 1. The realization of M-tap FIR filter

Where, 'S' denotes the sign bit, 'M' denotes the non fractional magnitude and 'MF' denotes the fractional magnitude. After applying the aforesaid quantization to the filter coefficients, the response of the filter is compared against that of the reference filter (which uses double precision floating point representation).

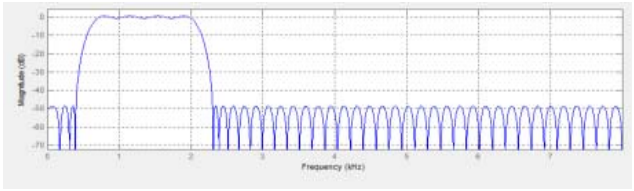

Figure 2. Frequency Response of an Equiripple FIR filter

B. Quantization to Q8.7 format

A larger effect of coefficient quantization can be observed for the representation format Q8.7. Here, the total word length is 8 bits and the number of bits available for representing the fractional magnitude is 7. Table II illustrates this distribution in detail.

Here, 'S' denotes the sign bit, 'M' denotes the non fractional magnitude and 'MF' denotes the fractional magnitude. After quantizing the filter coefficients to Q8.7 format, the response of the filter is compared with the reference filter. Fig 4 shows the response of the filter after quantizing coefficients to Q8.7 format. The pass band response is relatively unaffected by the quantization. However, the difference is drastic in the stop band response of the quantized filter. This is due to the substantial loss of precision as a result of quantization to the sub-optimal Q8.7 format. The effects of quantized word lengths on frequency response are shown in Fig 3 for Q16.14 format and in Fig 4 for Q8.7format [3] [12].

The effect of coefficient inaccuracy resulting from finite word length fixed point representation is more pronounced for a high-order filter when it is realized in the direct form than when it is realized in the parallel or cascade form. Therefore, the parallel or cascade form should be used for high-order filters whenever possible. The saving in the number of coefficient bits can be quite substantial [9].

TABLE III. NUMERIC REPRESENTATION FORMAT Q 8.7-BIT DISTRIBUTION

MSB (8)	Bit (6)	Bit (5)	Bit (4)	Bit (3)	Bit (2)	Bit (1)	Bit (0)
S	M	M_F	M_F	M_F	M_F	M_F	M_F

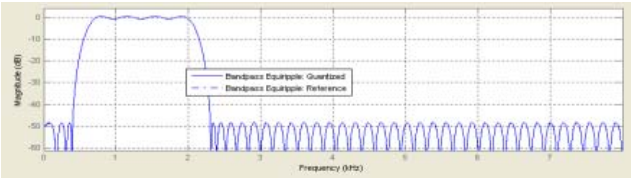

Figure 3. Frequency response for Q16.14 format

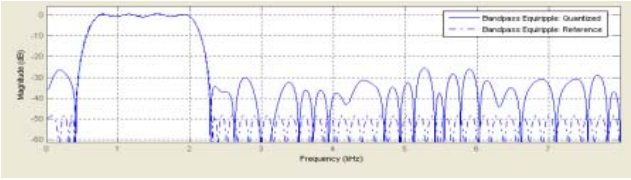

Figure 4. Frequency response for Q8.7 format

V. BLOCK MODELLING OF FIR FILTER

Simulink is a dynamic system and simulation software based on MATLAB for modeling and high level simulation. This environment is useful for the rapid implementation of DSP application in terms of functional blocks. These functional blocks can be translated into VHDL code using a system Generator for FPGA implementations. The developed model for audio filtering to meet the filter specifications given in Table1 is shown in Fig 5.

The Gateway in & Gateway out blocks are used to support parameters to control the conversion from double precision to Q 16.14 fixed point format for the sampling rate. It must be ensured that each signal must be sampled and each block in a System Generator design has an appropriate sample period. Thus the sample period must be set explicitly for each block in the design. The System Generator token controls the system sample period.

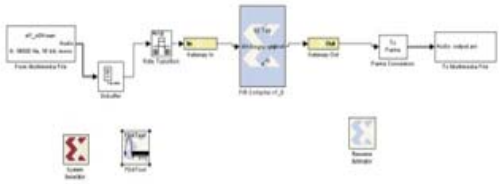

Figure 5. Simulink model of Audio FIR filter

VI. ARCHITECTURAL OPTIMISATION APPROCHES

There are various optimization techniques like Parallel processing, Pipelining, Distributed Arithmetic etc. available for low power and high speed digital filter architectures[5][7]. In this paper FCSD architecture and Distributed Arithmetic architectures are studied and FPGA resources utilization for both architectures are obtained for audio filtering application.

A. Distributed Arithmetic

The traditional ways to implement an FIR filter rely on Multiply-Accumulate (MAC) structure. But multipliers are limited resource on FPGA. Therefore for FPGA implementation point of view the architecture should use either minimum multipliers or it can be multiplier less. The Distributed Arithmetic (DA) architecture provides multiplier less architecture as it relies on Look-Up Tables (LUT), delays and scaling accumulator to implement FIR filter. DA efficiently implements the MAC using basic building blocks (LUTs) in FPGAs.

The arithmetic sum of products that defines the response of linear, time-invariant filter at any discrete time is given by equation (3).

$$y = \sum_{k=1}^{K} A_k X_k \tag{3}$$

Let X_k be N-bits scaled 2's complement number i.e. $|X_k|<1$ and X_k: $\{b_{k0}, b_{k1}, b_{k2}\ldots\ldots, b_{k(N-1)}\}$ where b_{k0} represents sign bit. Therefore x_k can be expressed as:

$$x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \tag{4}$$

Substituting (4) in (3) results in

$$y = \sum_{k=1}^{K} A_k \left[-b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \right]$$

$$y = -\sum_{k=1}^{K} (b_{k0} \cdot A_k) + \sum_{k=1}^{K} \sum_{n=1}^{N-1} (A_k \cdot b_{kn}) 2^{-n}$$

$$y = -\sum_{k=1}^{K} (b_{k0} \cdot A_k) + \sum_{k=1}^{K} \left[\sum_{n=1}^{N-1} (b_{kn} \cdot A_k) 2^{-n} \right]$$

$$y = -\sum_{k=1}^{K} (b_{k0} \bullet A_k) + \sum_{k=1}^{K} \left[(A_k \bullet b_{k1}) 2^{-1} + (A_k \bullet b_{k2}) 2^{-2} + \cdots + (A_k \bullet b_{k(N-1)}) 2^{-(N-1)} \right]$$

$$
\begin{aligned}
y = &-\left[b_{10} \bullet A_1 + b_{20} \bullet A_2 + \cdots + b_{K0} \bullet A_K\right] \\
&+\left[\left(b_{11} \bullet A_1\right)2^{-1} + \left(b_{12} \bullet A_1\right)2^{-2} + \cdots + \left(b_{1(N-1)} \bullet A_1\right)2^{-(N-1)}\right] \\
&+\left[\left(b_{21} \bullet A_2\right)2^{-1} + \left(b_{22} \bullet A_2\right)2^{-2} + \cdots + \left(b_{2(N-1)} \bullet A_2\right)2^{-(N-1)}\right] \\
&\vdots \\
&+\left[\left(b_{K1} \bullet A_K\right)2^{-1} + \left(b_{K2} \bullet A_K\right)2^{-2} + \cdots + \left(b_{K(N-1)} \bullet A_K\right)2^{-(N-1)}\right]
\end{aligned}
$$

$$
\begin{aligned}
y = &-\left[b_{10} \bullet A_1 + b_{20} \bullet A_2 + \cdots + b_{K0} \bullet A_K\right] \\
&+\left[\left(b_{11} \bullet A_1\right) + \left(b_{21} \bullet A_2\right) + \cdots + \left(b_{K1} \bullet A_K\right)\right]2^{-1} \\
&+\left[\left(b_{12} \bullet A_1\right) + \left(b_{22} \bullet A_2\right) + \cdots + \left(b_{K2} \bullet A_K\right)\right]2^{-2} \\
&\vdots \\
&+\left[\left(b_{1(N-1)} \bullet A_1\right) + \left(b_{2(N-1)} \bullet A_2\right) + \cdots + \left(b_{K(N-1)} \bullet A_K\right)\right]2^{-(N-1)}
\end{aligned}
$$

$$
y = -\sum_{k=1}^{K} (b_{k0}) \bullet A_k + \sum_{n=1}^{N-1}\left[b_{1n} \bullet A_k + b_{2n} \bullet A_2 + \cdots + b_{Kn} \bullet A_K\right]2^{-n}
$$

$$
y = -\sum_{k=1}^{K} A_k \bullet (b_{k0}) + \sum_{n=1}^{N-1}\left[\sum_{k=1}^{K} A_k \bullet b_{kn}\right]2^{-n} \qquad (5)
$$

Thus basic DA technique is bit-serial in nature and is basically a bit-level rearrangement of the MAC operation. DA hides the explicit multiplications by LUTs which makes it an efficient technique to implement on FPGA [4][6].

Equation (5) can be realized as shown in fig 6 and it is called Distributed Arithmetic.

B. Factored Canonic Signed Digit

Encoding a binary number such that it contains the fewest number of non-zero bits is called Canonic Signed Digit (CSD).Here, mapping a number to a ternary system {-1, 0, 1} versus a binary system {0, 1} is done. The following are the properties of CSD numbers:

- No two consecutive bits in a CSD number are non-zero.
- The CSD representation of a number contains the minimum possible number of non-zero bits, thus the name canonic.
- The CSD representation of a number is unique.

Among the N-bit CSD numbers in the range [-1,1] , the average number of non-zero bits is N/3 + 1/9 + O(2-N).Hence, on average, CSD numbers contains about 33% fewer non-zero bits than 2's complement number representation. The conversion of W-bit number in SW.(W-1) notation to CSD format is given as:

A = a'_{W-1}. a'_{W-2}… a'_1 a'_0 = 2's complement number and its CSD representation is $a_{W-1}.a_{W-2}$… a_1 a_0.

The number of computations and complexity can be reduced by searching the common sub-expression in coefficient terms before filter implementation and sharing them to eliminate extra computational complexity. This process is called Common Sub- expression Sharing (CSE).

VII. Simulation Results

Using FDA Tool of MATLAB 7.10.0 (R2010a) an Equiripple FIR filter is designed as per the specifications given in Table I for removing noise from a raw audio signal. Double precision floating point representation allows the tool to achieve a fair degree of precision [2]. But the FPGA implementation prefers fixed point representation.

Fast Fourier Transform (FFT) spectra of the raw audio signal and filtered audio signal for Double precision floating point and Fixed point filter coefficients is shown in Fig 7 and fig 8 respectively. From these Spectra it can be seen that the performance with Double precision floating point and fixed point filter coefficients is observed to be very similar.

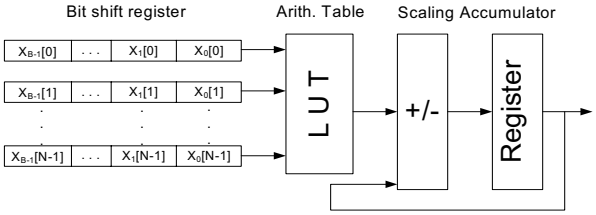

Figure 6. Block schematic of Distributed Arithmetic Architecture

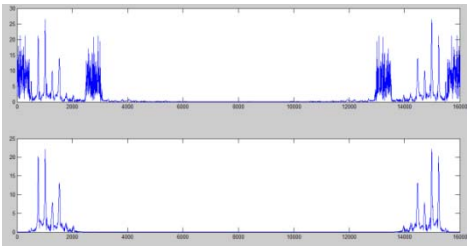

Figure 7. FFT with Double precision Floating point

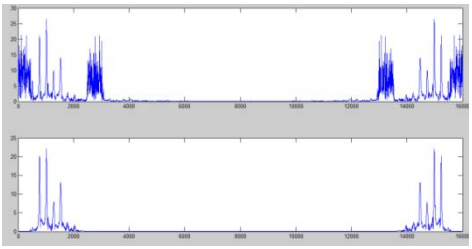

Figure 8. FFT with Fixed point

VIII. Experimental Resulsts

Distributed Arithmetic and FCSD architectures are synthesized using Xilinx ISE 9.1 for target device as a Spartan 3E-1200 FPGA device. The RTL schematic and layout for DA architecture and for FCSD architecture are shown in fig 9 and fig 10 respectively. The summary of the resources utilized is obtained from the synthesis reports of both the architectures and is presented in Table VI.

Figure 9 (a): RTL schematic for DA architecture

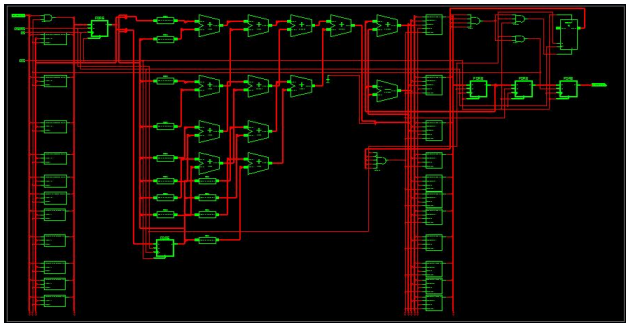

Figure 9 (b): Layout for DA architecture

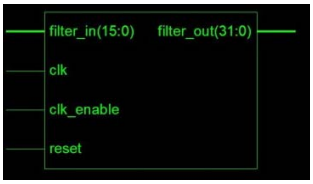

Figure 10 (a): RTL schematic for FCSD

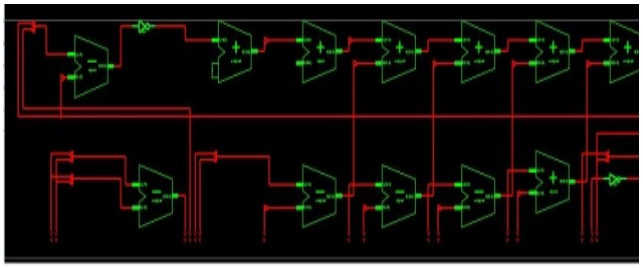

Figure 10 (b): Layout for FCSD Architecture

TABLE IV. COMPARISON OF FPGA RESOURCES UTILIZATION

Logic Utilization	**Distributed Arithmetic**			**FCSD**		
	Used	**Avail-able**	**Utiliz-ation**	**Used**	**Avail-able**	**Utilization**
Number of Slices	1680	8672	19%	5266	8672	60%
Number of Slice Flip Flops	1718	17344	9%	1504	17344	8%
Number of 4 input LUTs	2011	17344	11%	8633	17344	49%
Number of bonded IOBs	54	304	17%	51	304	16%
Number of MULT18X18SIOs	-	-	-	-	-	-
Number of GCLKs	1	24	4%	1	24	4%

IX. CONCLUSION

Equiripple FIR filter design method results in the filter with minimum order compare to other optimal and windowing design methods. DA architecture can be used to for high speed implementations. FCSD architecture reduces the power consumption and computation complexity significantly. Still there is a scope to apply appropriate techniques to reduce the number of resources utilized and reduction in the computation complexity.

REFERENCES

[1] U. Meyer Baese, "*Digital signal processing with FPGA*", Springer Publications.

[2] Shanthala S & S. Y. Kulkarni " High speed and low power FPGA implementation of FIR filter for DSP applications" *European Journal of Scientific Research* ISSN 1450-216X Vol.31 No.1 (2009), pp.19-28 .

[3] Floating-Point Number Specification IEEE 754 (1985), "IEEE Standard for Binary Floating-Point Arithmetic

[4] Xilinx App Note, "The Role of Distributed Arithmetic In FPGA Based Signal Processing".

[5] Parhi K.K., *VLSI Digital Signal Processing Systems: Design and Implementation*, Wiley, New York, (1999).

[6] A Tutorial on Distributed Arithmetic: Implementations and Applications

[7] Prof. Gopal S. Gawande , Dr. K. B. Khanchandani , T. P. Marode, "Performance Analysis Of Fir Digital Filter Design Techniques", *International Journal of Computing and Corporate Research*, Volume 2 Issue 1 January 2012.

[8] Keshab K. Parhi, "Approaches to Low-Power Implementations of DSP Systems", *IEEE Transactions On Circuits And Systems—I: Fundamental Theory And Applications*, Vol. 48, No. 10, October 2001

[9] X. Hu, L. S. DeBrunner, and V. DeBrunner, 2002, "An efficient design for FIR filters with Variable precision", *Proc. 2002 IEEE Int. Symp. on Circuits and Systems*, pp.365-368, vol.4,May 2002.

[10] Anshika Rajolia,Maninder Kaur, "Finite Impulse Response(FIR) Filter Design Using Canonic Signed Digit (CSD)", *IJSR*, Vol-2Issue 7,July 2013.

[11] Kanu Priya,Rajesh Mehra, "FPGA based cost efficient FIR filter using Factored CSD technique", *IJRTE* ISSN:2277-3878,Vol.I,Issue 6,Jan 2013.

[12] V.Sudhakar, N.S.Murthy, L.Anjaneyulu, "Area Efficient Pipelined Architecture for Realization of FIR filter using Distributed Arithmetic", *IPCSIT Press* Vol 31, Singapore.

[13] Maheshbabu Ketha, CH.Venkateshwarlu Kantipudi Raghuram, "Design & FPGA Implementation of Reconfigurable FIR filter Architecture for DSP Applications", *IJERT* ISSN:2278-0181,Vol.1 Issue 7 Sept 2012.