J. Basic. Appl. Sci. Res., 3(7)562-573, 2013

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

Efficient Hardware Implementation of Digital Filters using Distributed Arithmetic (DA)

Muhammad Hasnain[1], Saifullah Hammad[2], M. Aqeel Iqbal[3]

[1]Department of Electronics, Quaid-i-Azam University (QAU), Islamabad, Pakistan
[2]Department of Electrical & Electronics Engineering, Muhammad Ali Jinnah University (MAJU), Islamabad, Pakistan
[3]Dept. of CE, College of Electrical & Mechanical Engineering, National University of Science and Technology (NUST), Islamabad, Pakistan

ABSTRACT

The FPGA (Field Programmable Gate Array) constitute of many programmable modules like Configuration Logic Blocks (CLBs), Block Random Access Memories (BRAM), DSP 48 blocks and Input/output (I/O) modules. The CLBs are the main programmable logic units which consist of different number of logic slices and each slice contains different number of LUTs and flips flops depending upon the FPGA device family. The CLBs and DSP 48 blocks are the most expensive resources on the FPGA and are used wisely in the design of FPGA based system. In most of FPGA based DSP applications, CLB's and DSP 48 blocks are utilized in implementing algorithmic logic and digital filters whereas the BRAM remains unutilized. The non utilization of FPGA resources like BRAM motivates to implement various DSP module like filters (FIR or IIR) using BRAM. The Distributed Arithmetic (DA) is a technique which can be used to implement digital FIR and IIR filters. The DA logic replaces the MAC operation of convolution summation of any digital filter (IIR or FIR) into a bit serial look up table read and addition operation. Hence by implementation of digital filters using DA, expansive FPGA resources like DSP 48 block can be saved and used to implement the algorithmic logic for DSP algorithms. The research paper presents an efficient method for hardware implementation of digital filters (IIR and FIR) of any order by exploiting Distributed Arithmetic (DA). This paper presents critical analysis and optimization provided by the purposed design with respect to its conventional DA based design.

KEYWORDS: Distributed Arithmetic (DA), Finite Impulse Response (FIR) Filter, Infinite Impulse Response (IIR) filters, Multiply Accumulate (MAC), Bits Combination Factor (BCF).

1. INTRODUCTION

Distributed arithmetic (DA) is a way of implementing a dot product where one of the arrays has constant elements. The DA can be effectively used to implement FIR, IIR and FFT type algorithms [1, 2]. For example, in the case of an FIR filter, the coefficients constitute an array of constants in some signed Q format where the tapped delay line forms the array of variables which changes every sample clock. The DA logic replaces the MAC operation of convolution summation into a bit serial look up table read and addition operation [1, 2, 3]. Keeping in perspective the architecture of FPGAs, time/area effective designs can be implemented using DA techniques [4]. The DA logic works by first expanding the array of variable numbers in the dot product as a binary number and then rearranging MAC terms with respect to weights of the bits. A mathematical explanation of this rearrangement and grouping is given here. FPGAs with look up tables suit well DA based filter design [5]. A DA based design eliminates the use of a hardware multiplier and uses only a look up table to provide high throughput execution at bit rate irrespective of the filter length and width of the coefficients [6].Let the different elements of arrays of constants and variables be A_k and x_k, respectively. The length of both the arrays is K. Then their dot product can be written as:

$$y = \sum_{k=0}^{K-1} A_k x_k^{\,x} \quad \rightarrow \quad \text{Equation (1)}$$

Without lost of generality, let us assume x_k is an N bit Q1. (N-1) format number:

$$x_k = -x_{k0}2^0 + \sum_{b=1}^{N-1} z\, x_{kb}2^{-b}$$
$$x_k = -x_{k0}2^0 + x_{k1}2^{-1} + \ldots + x_{k(N-1)}2^{(N-1)}$$

*Corresponding Author: Muhammad Hasnain, Department of Electronics, Quaid-i-Azam University (QAU), Islamabad, Pakistan
muhammadhasnain@hotmail.com

The dot product of equation (1) can be written as:

$$y = \sum_{k=0}^{K-1} \left(-x_{k0}2^0 + \sum_{b=1}^{K-1} x_{kb}2^{-b} \right)^x A_k$$

$$y = \sum_{k=0}^{K-1} (\ -x_{k0}2^0 + x_{k1}2^1 + \ \ldots\ldots + x_{k(N-1)}2^{(N-1)} \ )^x A_k$$

Rearranging the terms yields:

$$y = -\sum_{k=0}^{K-1} x_{k0}A_k 2^0 + \sum_{b=1}^{N-1} 2^{-b} \sum_{k=0}^{K-1} x_{kb}A_k \quad ^x$$

For K = 3 and N = 4, the rearrangement forms the following entries in the ROM:

$$\begin{aligned} &-(x_{00}A_0 + x_{10}A_1 + x_{20}A_2)\ 2^0 \\ &+(x_{01}A_0 + x_{11}A_1 + x_{21}A_2)\ 2^{-1} \\ &+(x_{02}A_0 + x_{12}A_1 + x_{22}A_2)\ 2^{-2} \\ &+(x_{03}A_0 + x_{13}A_1 + x_{23}A_2)\ 2^{-3} \end{aligned}$$

$\rightarrow$ Equation (2)

The DA technique pre computes all possible values of

$$\sum_{k=0}^{K-1} x_{kb}A_k$$

For the example under consideration, the summations for all eight possible values of x_{kb} for a particular b and k = 0, 1 and 2 are computed and stored in ROM. The ROM is P bits wide and 2^K deep and implements a look up table. The value of P is:

$$P = floor\left(\log_2 \sum_{k=0}^{K-1} |A_k| \right) + 1$$

This size is prohibitively large and several techniques are used to reduce the ROM requirement [7, 8]. The conventional design for hardware implementation of DA based FIR filter with K=L and N bit data sample is shown in figure below [9],

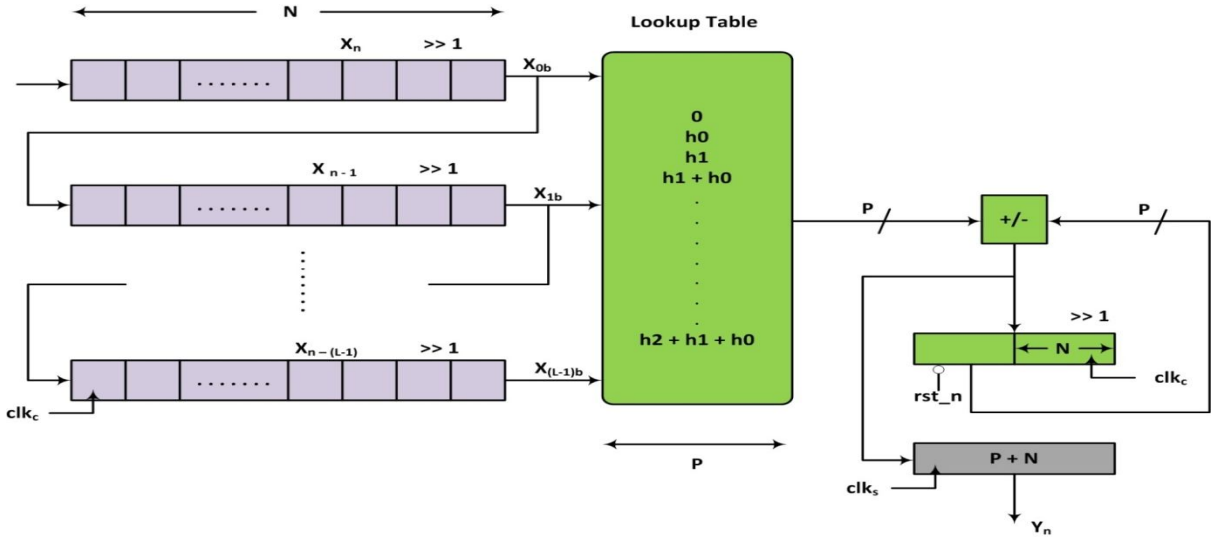

Figure 1: Conventional DA based architecture for implementing an FIR filter of length L and N bit data samples

There are few drawbacks with conventional design of DA, which are

1. The clk_C (circuit clock) should be N times faster than the clk_S (system clock) where N is with of sampled data.
2. The BRAM requirement for implementation.
3. Design operating speed compared with pipelined MAC based implementation.

In this research paper we will purpose a fixed point design which caters the drawbacks in conventional DA based design. We will first purpose a design for L order and N samples FIR and IIR filters, method of computing the ROM/RAM contents and the architecture of the purposed design. Also we will used the purposed design to implemented second order IIR filter (biquad filter) and compares the performance of purposed design with conventional DA based implementation, pipelined MAC based implementation and normal implementation of biquad IIR filter.

2. PURPOSED DESIGN

In this research paper we will purpose the design which can be used to implement any K order FIR and IIR filters. The design exploits the core concept of distributed arithmetic by pre-computing the result of X-bits specified by BCF (Bits Combination Factor) of N-bit sample of K variables in tapped delay line at a time and then stores the result into BRAM contrary to conventional DA design in which result of 1-bit of N-bit sample of K variables in tapped delay line is computed at a time and stored in the BRAM. In this way we can perform computation of X bits specified by BCF of N-bits sample at circuit clock which will increase the throughput of the design by a factor of X and hence performance of the design. Also another single port RAM/ROM is used to store the 2's complement of twice the value of contribution of MSB; it is discussed in detail in the next subsection. The only constraint in the purposed design is X (specified by BCF) should be selected such that it is multiple of N (number of bit in variable). For example for Q2.22 format fixed point number, N = 24 and value of 1, 2, 3, 4, 6, 8, 12 and 24 can be selected as X i.e. Bit Combination Factor.

2.1 ROM/RAM content Generation

In the purposed design in order to pre-compute the content of the ROM/RAM, result of X-bits from each N-bits sample of K variables in tapped delay line is pre-computed by using value corresponding to the first bit which is right shifted by one and then added into the value corresponding to second bit (obtained by adding first bit's result with shifting right and second bits result without shifting), then right shifting the accumulated result by 1 and then adding the value corresponding to third bit in the accumulated value (obtained by adding first bit's and second bit's result (accumulated result) with shifting right and second bits result without shifting) and so on until value corresponding to X^{th} bits is added into the accumulated sum. After adding the contribution of X^{th} bits, the accumulated sum should not be shifted right. The computed value is stored into the ROM/RAM at address corresponding to the value represented by concatenation of X-bits of each N-bits sample of variables in tapped delay line. The above method is applicable to FIR filters only. For IIR filters to pre-compute content of ROM/RAM, feed forward path and feedback path should be catered separately and same method as discussed above is applied to them.

Also in Distribute Arithmetic (DA) we need to subtract the contribution of MSB (Most Significant Bit) from the accumulated result of remaining bits. To accommodate it, in the purposed design we use another single port ROM/RAM which contains the 2's complement of twice the value of contribution of MSB. The contribution of MSB is simply added into the accumulated result of remaining bits. The twice value of 2's complement eliminates the effect of inherent addition of MSB in the accumulated result of all bits (including MSB) along with the required subtraction of contribution of MSB.

For the purposed design, the width of single port ROM/RAMs containing pre-computed values are R-bit wide and 2^K deep and implemented as look up table. The value of R is

$$R = floor\left(\log_2 \sum_{k=0}^{K-1} |A_k|\right) + BCF$$

In above equation, BCF is the Bits Combination Factor.

For mathematical prove of the purposed design, consider the example for K = 3, N = 4 and bit combination factor (BCF) = 2, the rearrangement forms the following entries in the ROM mentioned in introduction section. From equation (2),

$T = (x_{00}A_0 + x_{10}A_1 + x_{20}A_2)\ 2^0$ → Equation (3)
$U = (x_{01}A_0 + x_{11}A_1 + x_{21}A_2)\ 2^{-1}$ → Equation (4)
$V = (x_{02}A_0 + x_{12}A_1 + x_{22}A_2)\ 2^{-2}$ → Equation (5)
$W = (x_{03}A_0 + x_{13}A_1 + x_{23}A_2)\ 2^{-3}$ → Equation (6)

According to distributed arithmetic technique,

$Z = -T + U + V + W$ → Equation (7)

Adding equation (5) and (6)

$V + W = (x_{02}A_0 + x_{12}A_1 + x_{22}A_2)\ 2^{-2} + (x_{03}A_0 + x_{13}A_1 + x_{23}A_2)\ 2^{-3}$
$V + W = ((x_{02}A_0 + x_{12}A_1 + x_{22}A_2) + (x_{03}A_0 + x_{13}A_1 + x_{23}A_2)\ 2^{-1})\ 2^{-2}$

Or $V + W = A * 2^{-2}$ → Equation (8)

Where $A = ((x_{02}A_0 + x_{12}A_1 + x_{22}A_2) + (x_{03}A_0 + x_{13}A_1 + x_{23}A_2)\ 2^{-1})$ → Equation (9)

Similarly adding equation (3) and (4)

$T + U = (x_{00}A_0 + x_{10}A_1 + x_{20}A_2)\ 2^0 + (x_{01}A_0 + x_{11}A_1 + x_{21}A_2)\ 2^{-1}$

or

$T + U = (x_{00}A_0 + x_{10}A_1 + x_{20}A_2) + (x_{01}A_0 + x_{11}A_1 + x_{21}A_2)\ 2^{-1}$

Let $T + U = B$ → Equation (10)

Now to obtain the accumulated result, add equation (8), (10) and compensate the subtraction of MSB contribution,

$C = B + A * 2^{-2} - 2T$

According to the purposed design equation (9) and equation (10) should be the content of the single Port RAM/ROM placed at the address correspond by concatenation of 2-bits (specified by BCF) of 4-bits sample of 3 variables of tapped delay line. Also twice the 2's complement of the T would be the content of single port ROM/RAM which will be used to compensate the contribution of MSB.

Substituting value of *A* and *B* from equation (9) and (10)

$C = T + U + ((x_{02}A_0 + x_{12}A_1 + x_{22}A_2) + (x_{03}A_0 + x_{13}A_1 + x_{23}A_2)\ 2^{-1}) * 2^{-2} - 2T$
$C = T + U + V + W - 2T$
$C = -T + U + V + W$

Above equation is same as equation (7), which represents the result obtained from conventional distributed arithmetic approach. Similarly the purposed design can be mathematical proved for filter of any order K, N bits sample width and bit combination factor (BCF) = bcf.

2.2 Architecture

In the purposed design architecture, two single port RAM/ROMs are used; one for storing the pre-computed content of distributed arithmetic logic and the other one for storing the 2's complement of twice the value of contribution of the MSB. In this way we can perform computation of X bits (specified by BCF) of N-bits sample of K variables in tapped delay line at circuit clock which will increase the throughput of the design by a factor of BCF. Also, instead of using ripple carry adder to compute the summation of intermediate results we will use Wallace compression tree which will further increase the design performance. We will discuss the hardware architecture of FIR filter in detail whereas for IIR filter the same concept is applied.

The implemented hardware architectures for purposed design for FIR and IIR filters are given below,

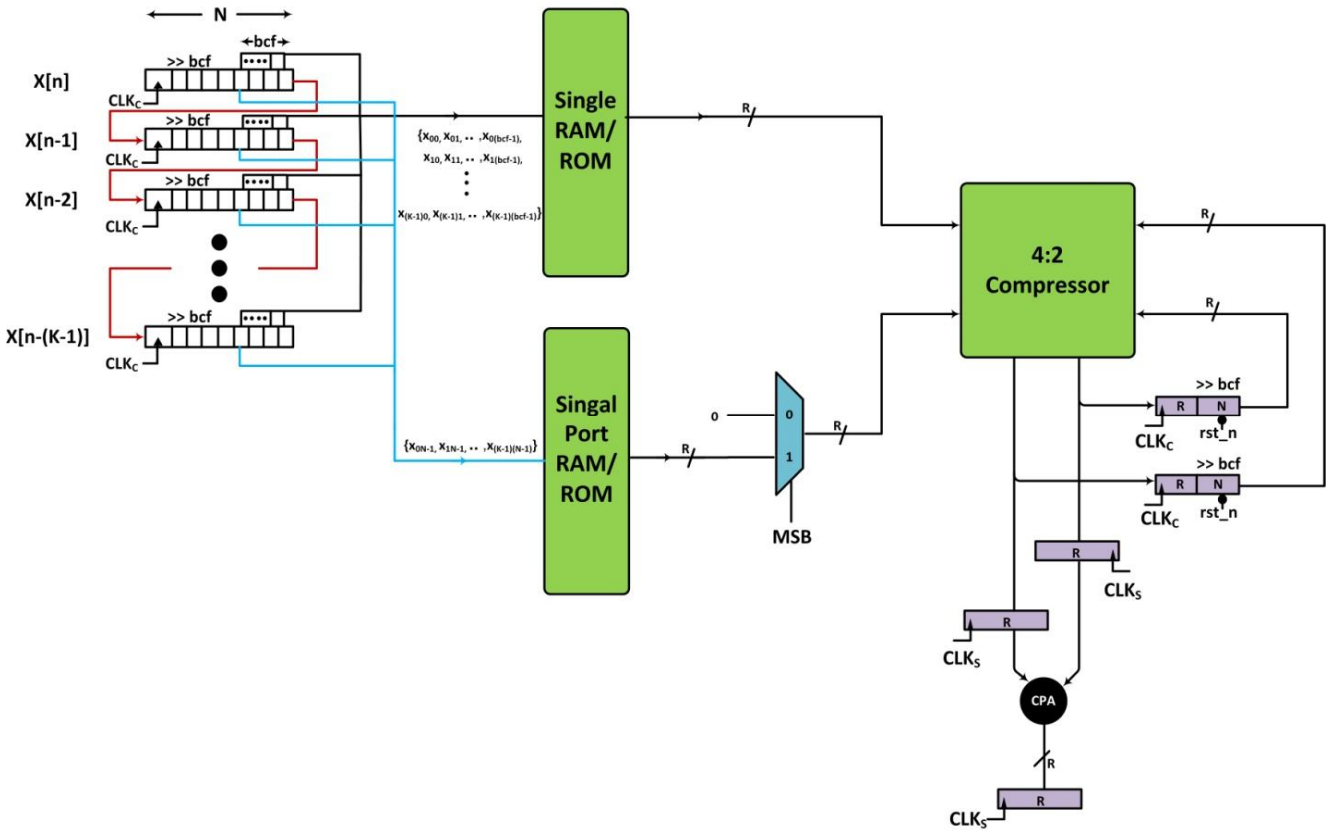

Figure 2: Purposed Architecture for FIR filter of length K, N bit data samples and 'bcf' as Bits Combination Factor

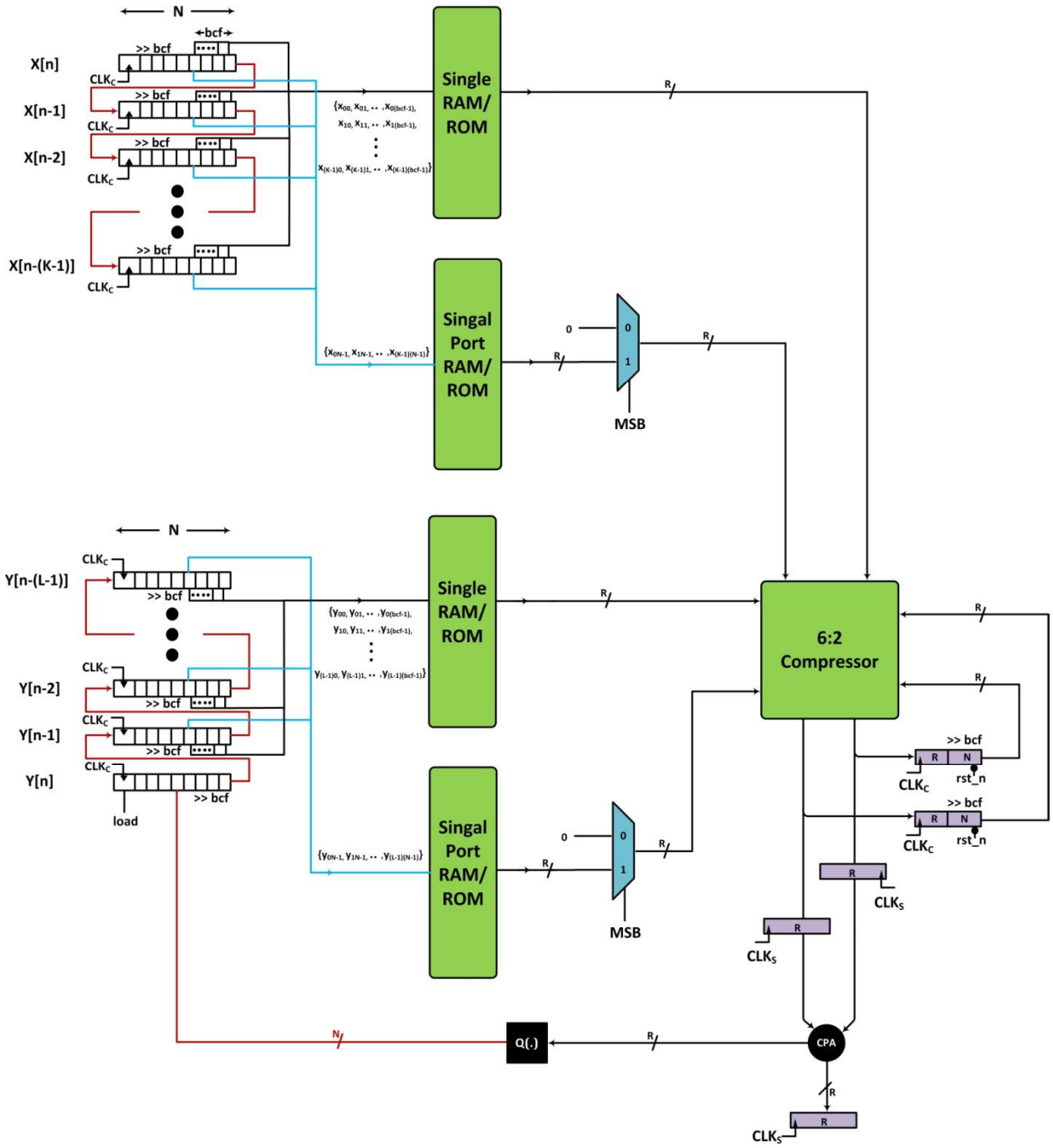

Figure 3: Purposed Architecture for IIR filter of length K for feed forward path, length L for feed backward path, N bit data samples and 'bcf' as Bits Combination Factor

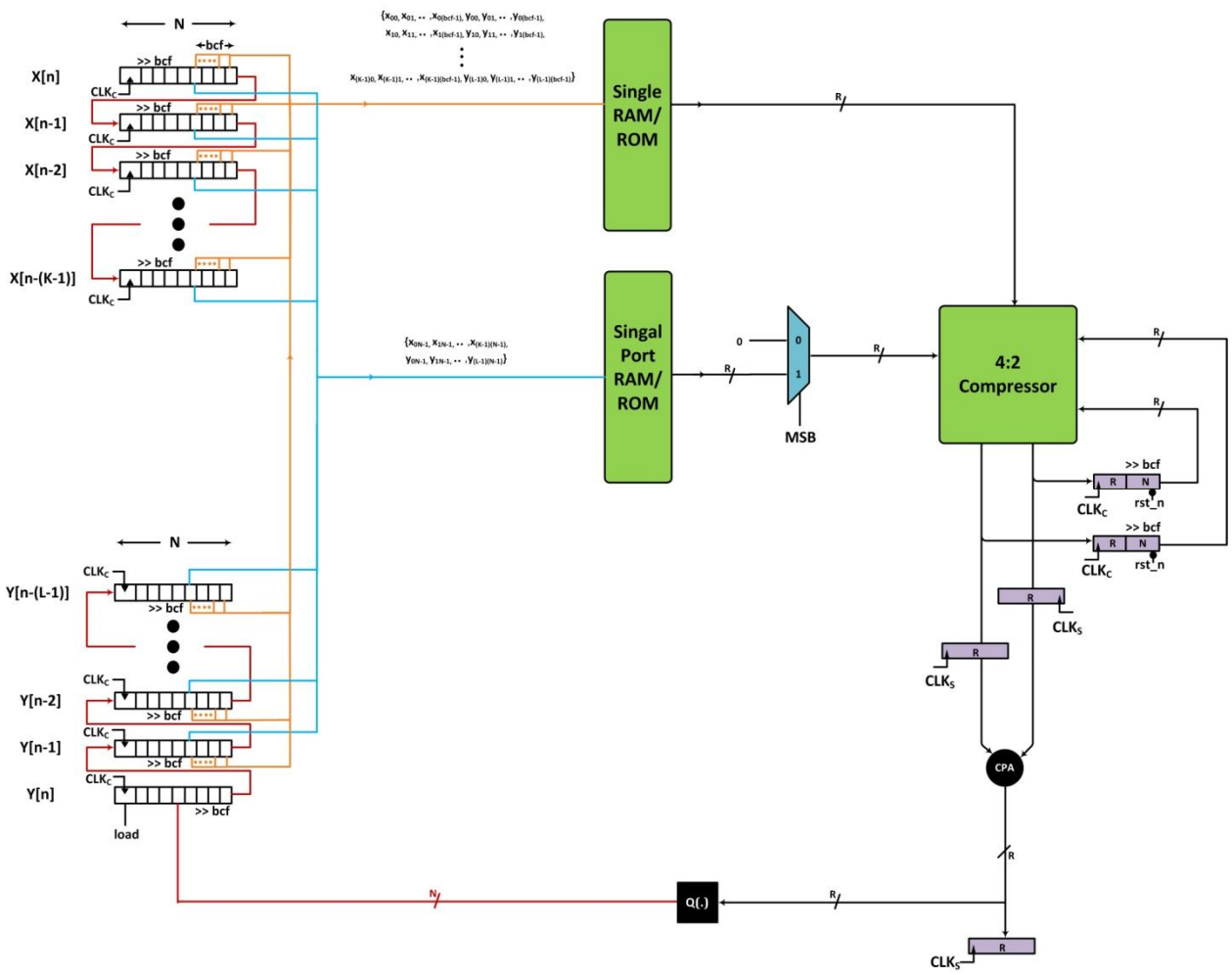

Figure 4: Purposed Architecture for IIR filter using single ROM/RAM (for storing coefficients – both feedback & feed forward paths) of length K for feed forward path, length L for feed backward path, N bit data samples and 'bcf' as Bits Combination Factor

3. APPLICATION

Biquad IIR filter is very commonly used in DSP applications like digital audio processing, digital video processing, signal conditioning and channel selection filtering. Biquad is a second order (two poles and two zeros) IIR filter. It is high enough order to be useful on its own and because of coefficient sensitivities in higher order filters biquad is often used as basic building block for more complex filters. For instance, a biquad low pass filter has a cutoff slope of 12 dB per octave which is useful for tone controling. If you need a 24 dB/octave slope, you can cascade two biquads and it will have less coefficient-sensitivity problems than a single fourth-order design. In this research paper, we will use the purposed design to implement second order IIR filter (biquad filter) for Q1.15 format with different number of BCF and will compare the performance of purposed design with conventional DA based implementation, pipelined MAC based implementation and normal implementation of biquad IIR filter. A software application is developed to compute the content of the ROM/RAM which uses the coefficient of the filters and generates the content of the ROM/RAM. The biquad IIR filter can be represented mathematically as,

$$y(n) = a0.x(n) + a1.x(n-1) + a2.x(n-2) + b1.y(n-1) + b2.y(n-2)$$

Following are the architecture of normal implementation of biquad IIR filter (figure 5), pipelined MAC based implementation (figure 6), conventional DA design (figure 7) and the purposed design (figure 8). The specifications

of the FPGA, tools and programming language used to implement all the described architectures is given in the table below

FPGA SPECIFICATION	
Family	Virtex 4
Device	XC4VLX15
Package	SF363
Speed	-12
TOOL & PROGRAMMING LANGUAGE	
HDL Language	Verilog HDL
Xilinx Implementation Tools	Xilinx ISE v12.1 M.53d
Simulation	Mentor Graphics Model SIM v6.5

3.1 Normal Hardware Implementation

The hardware architecture for implementation biquad IIR filter in direct form I is given below.

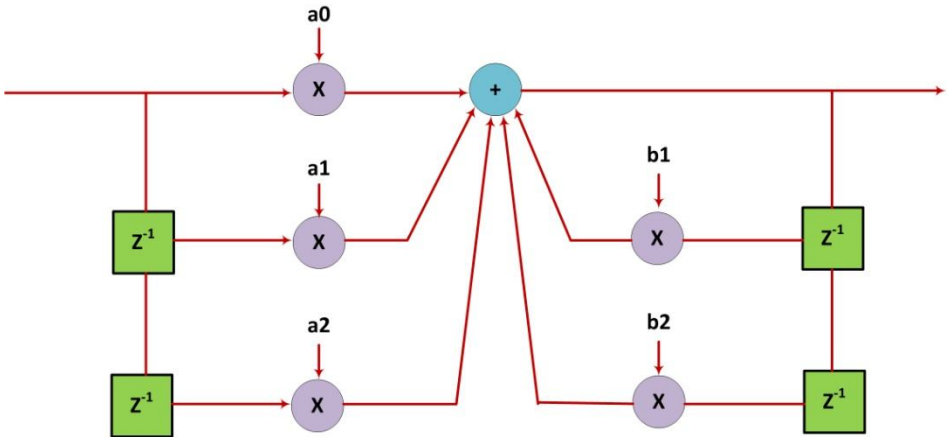

Figure 5: Direct form I implementation of biquad IIR filter

3.2 Pipelined MAC (Multiply Accumulate) based Design

The hardware architecture of biquad IIR filter with pipelined MAC (Multiply Accumulate) module is given below [10].

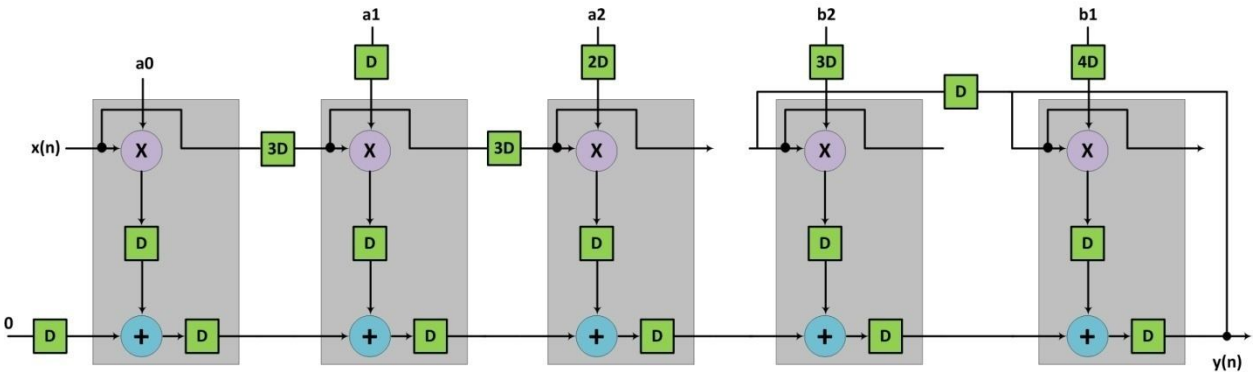

Figure 6: Biquad IIR filter implementation using pipelined MAC module

3.3 Conventional DA Design

The hardware architecture of biquad IIR filter with conventional DA design is given below.

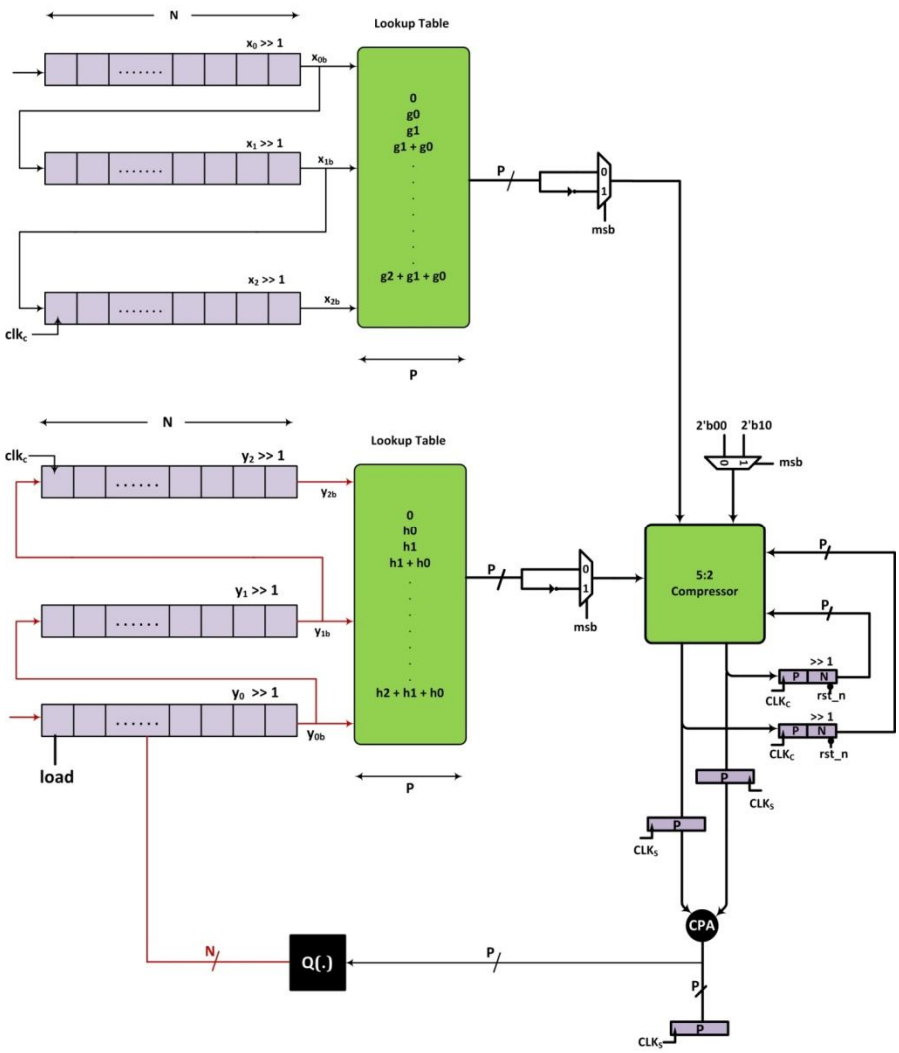

Figure 7: Biquad IIR filter implementation using conventional DA

3.4 Purposed Design

The hardware architecture of biquad IIR filter with purposed design is given below.

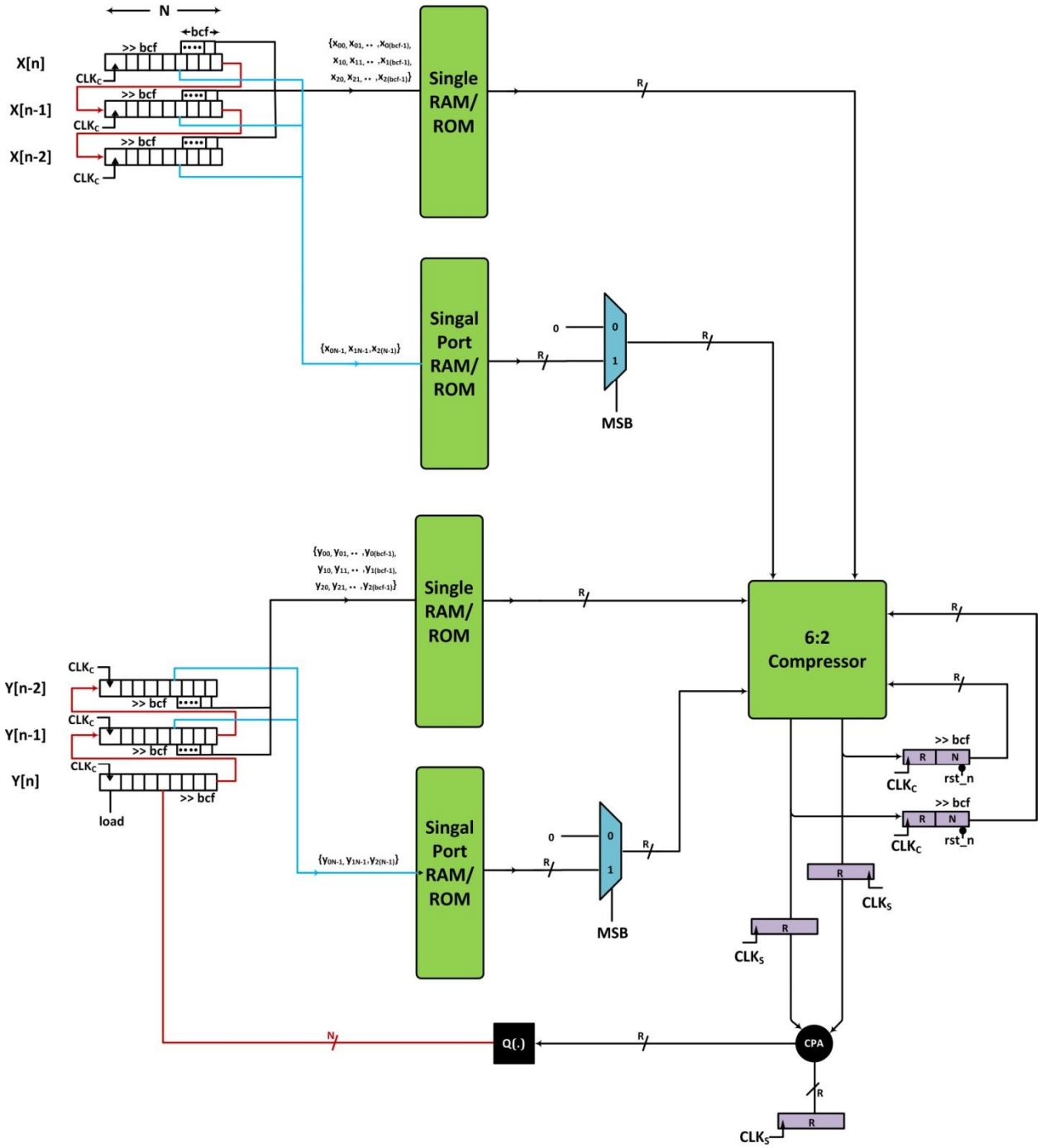

Figure 8: Biquad IIR filter implementation using purposed design

4. RESULTS

Following are the results obtained from the hardware implementation of biquad filter using the purposed design;

4.1 Data Processing Throughput

The table below compares the data processing throughput of the conventional and the purposed design. It is evident from the table that purposed design processes more bits per circuit clock (clk_C) then the conventional design.

Moreover the circuit clock (clk_C) requirement is also significantly reduced by using the purposed design as compared with the conventional design.

Table 1: Data processing throughput comparison between the purposed design and conventional designDATA PROCESSING THROUGHPUT COMPARISON

$Q_{N.M}$ FORMAT	SYSTEM CLOCK (CLK_S)	BCF	CONVENTIONAL DESIGN (Q1.15)		PURPOSED DESIGN (Q1.15)	
			CIRCUIT CLOCK (CLK_C)	BITS PROCESSED/ CLK_C	NO. OF CIRCUIT CLOCK (CLK_C)	BITS PROCESSED/ CLK_C
Qn.m	X MHz	bcf	16 * X MHz	1	Qn.m / bcf MHz	bcf
EXAMPLE						
Q1.15	10 MHz	4	160 MHz	1	40 MHz	4
Q1.15	10 MHz	16	10 MHz	1	10 MHz	16
Q2.22	10 MHz	8	240 MHz	1	30 MHz	8
Q2.22	10 MHz	24	240 MHz	1	10 MHz	24

4.2 Speed

The table below compares the maximum frequency (Synthesize Frequency) at which the hardware designs can be operated for various implementations of biquad filter.

Table 2: Synthesize frequency analysis of various implementations of biquad filter

SYNTHESIZE FREQUENCY (MHz)			
NORMAL IMPLEMENTATION	PIPELINED MAC IMPLEMENTATION	CONVENTIONAL DA IMPLEMENTATION	PURPOSED DA IMPLEMENTATION
115.827 MHz	324.654 MHz	549.753 MHz	549.753 MHz

4.3 Area

The table below compares the FPGA area and resources utilized by the hardware designs of various implementations of biquad filter.

Table 3: Device utilization analysis of various implementations of biquad filter

DEVICE UTILIZATION (XC4VLX15-12SF363)												
DESIGN	DSP 48			SLICES			SLICE FF			4 – INPUT LUTS		
	USED	TOTAL	%	USED	TOTAL	%	USED	TOTAL	%	USED	TOTAL	%
Normal	0	32	0	408	6144	6.64	96	12288	0.78	674	12288	5.49
Pipelined MAC	5	32		66	6144	1.07	116	12288	0.94	1	12288	0.0
Conventional DA (Q1.15)	0	32	0	39	6144	0.63	41	12288	0.33	62	12288	0.50
Purposed DA (Q1.15) with BCF = 2	0	32	0	35	6144	0.56	41	12288	0.33	55	12288	0.44

4.4 BRAM requirements

The graph below shows the utilization of BRAM resources of FPGA for implementation of biquad IIR filter using conventional and purposed design;

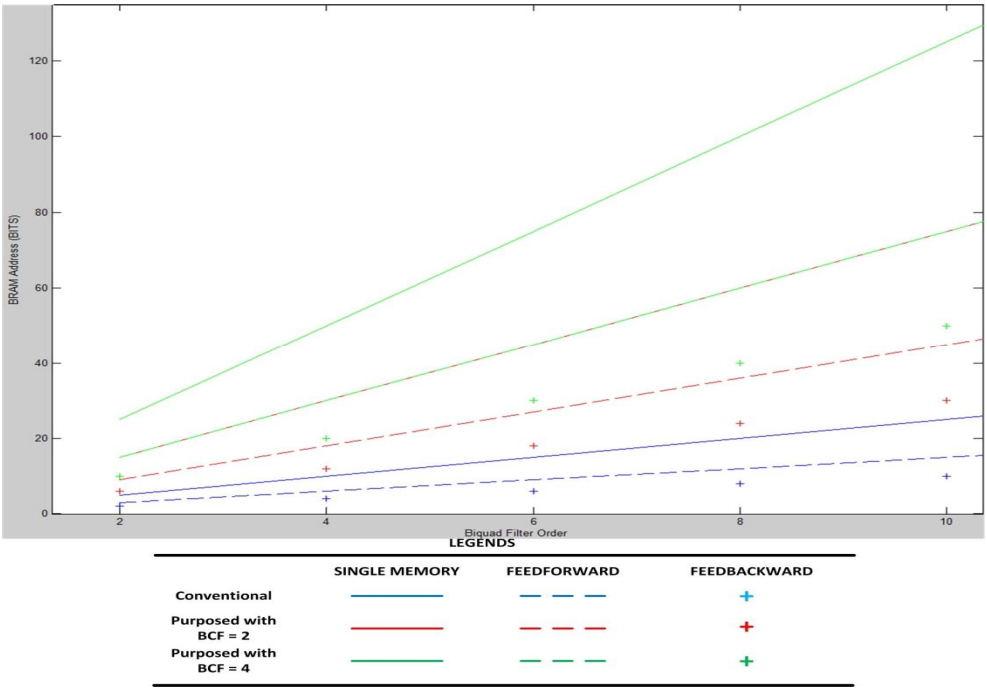

Figure 9: Block RAM requirement for implementation Biquad IIR filter

5. CONCLUSION

From the results, it is evident that the purposed design is the optimized (from above all designs) with minimum used of FPGA recourses. Also no expensive recourses like DSP 48 block is being used which can now be used for algorithmic logic of DSP or other algorithms. Most importantly, the purposed design is the fastest of all discussed designs in terms processing of bits per circuit clock cycle. The only drawback is the utilization of BRAM, which will increases with the increase in filter order.

6. REFERENCES

[1]. D. J. Allred, H.Yoo,V. Krishnan,W. Huang and D.V. Anderson,"LMS adaptive filters using distributed arithmetic for high throughput," IEEE Transactions on Circuits and Systems, 2005, vol. 52, pp. 1327 1337.

[2]. H. Yoo and D. V. Anderson, "Hardware efficient distributed arithmetic architecture for high order digital filters," Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2005, vol.5, pp. 125 12

[3]. S. Hwang, G. Han, S. Kang and J. Kim, "New distributed arithmetic algorithm for low power FIR filter implementation," IEEE Signal Processing Letters, 2004, vol. 11, pp. 463 466.

[4]. P. K. Meher, S. Chandrasekaran and A. Amira, "FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic," IEEE Transactions on Signal Processing, 2008, vol. 56, pp. 3009 3017.

[5]. W. Sen, T. Bin and Z. Jim, "Distributed arithmetic for FIR filter design on FPGA," in Proceedings of IEEE international conference on Communications, Circuits and Systems, Japan, 2007, vol. 1, pp. 620 623.

[6]. S. A. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," IEEE ASSP Magazine, 1989, vol. 6, pp. 4 19.

[7]. P. Longa and A. Miri, "Area efficient FIR filter design on FPGAs using distributed arithmetic," Proceedings of IEEE International Symposium on Signal Processing and Information Technology, 2006, pp. 248 252.

[8]. H. Natarajan, A. G. Dempster and U. Meyer Base, "Fast discrete Fourier transform computations using the reduced adder graph technique," EURASIP Journal on Advances in Signal Processing, 2007, December, pp. 1 10.

[9]. DIGITAL DESIGN OF SIGNAL PROCESSING SYSTEMS A PRACTICAL APPROACH by Shoab Ahmed Khan National University of Sciences and Technology (NUST), Pakistan

[10]. "FPGA-based Implementation of Signal Processing Systems by Roger Woods, John McAllister, Gaye Lightbody, Ying Yi", figure 9.4, page 179