Dublin City University
School of Electronic Engineering

Coursework Submission Cover Sheet

Please use block capitals

Student No	18210473	**Degree Scheme**	Postgraduate -Meng
Student Name	OMKAR PABSHETWAR	**Year**	2018-2019
Module	RT-DSP	**Lecturer**	Dr Prince Anandarajah
Title	FIR filter Design and Implementation using FPGA	**Hours spent on this exercise**	41

I hereby declare that the attached submission is all my own work, that it has not previously been submitted for assessment, and that I have not knowingly allowed it to be used by another student. I understand that deceiving or attempting to deceive examiners by passing off the work of another as one's own is not permitted. I also understand that using another's student's work or knowingly allowing another student to use my work is against the University regulations and that doing so will result in loss of marks and possible disciplinary proceedings.

Signed:  **Date:** 20/11/2018

Note: Coursework examiners are entitled to reject any coursework which does not have a signed copy of this form attached.

For use by examiners only (students should not write below this line)

Comments:

FIR Filter Design and Implementation using FPGAs

Abstract:

Field Programmable Gate Array (FPGA) are at the edge of changing fundamentals of digital signal processing. The algorithms such as FIR or IIR were built with ASICs or PDSPs in the past and they are now replaced with FPGAs. Finite Impulse Response (FIR) filter is most widely used filter in the DSP. In this paper the design of FIR filter using several approaches on FPGAs are presented. The Distributed Algorithm (DA) is the used of FIR to implement based on FPGA, compared with traditional multiply-add structure. The other approaches for design and FPGA implementation are DA-offset binary coding (DAOBC), Kaiser window function, Least Mean Square (LMS) algorithm, Short word length technique (SWL), Canonical Signed Digit (CSD) number system described in this paper.

Introduction:

Digital Signal processing has become one of the mature technology and has replaced many traditional technologies over past 30 years. The techniques involved in DSP are filtering, convolution and transformations. The FIR filter is one of the most important parts of digital signal, FIR filters consist of finite number of sampling points, which always has the stable system structure, and due to its linear phase in the symmetry conditions compared to IIR filters, FIR filters has simple and regular structures which makes it easy to implement on hardware, so it is used widely. During Implementation of FIR filter on FPGA speed and power constraints need to be satisfied. Reconfigurable Finite Impulse Response (FIR) filter whose filter coefficients change rapidly during execution time plays an important role in many application scenarios like the Software Defined Radio systems, digital up/down converters and multi-channel filters [1] [2]. The DA architecture is limited up to one type of filter coefficient, to operate on variable filter coefficients Dynamic Distributed Algorithm (DDA) architecture is used [3]. Digital filter has two main development directions: Fixed and variable filter [4].The memory devices such as ROM, RAM, Look Up Table (LUT) which are used to store the filter coefficients, these filter coefficients are precomputed. The memory based algorithm which appeared as a very efficient solution especially suited for LUT based

FPGA architectures and can greatly improve the speed of operation is Distributed Arithmetic (DA) method [5] [1]. How to design a high-speed FIR digital filter implemented on FPGA has become a hot search field in signal processing in recent years. Based on Liu Xiongfei's research, 16-tap direct form FIR linear phase low pass filter using Kaiser window function was designed based on DSP builder system modelling approach and filter's performance was greatly improved [6]. Parallel FIR filter is widely used among various types of filter in DSP. The paper [7] gives details of area efficient 2-parallel FIR filter and its implementation on FPGA using image system. Parallel filter is widely used among various types of filters in DSP. The method to implement the high speed, narrow transitions width FIR filter, a use of modified frequency response masking approach to realize narrow transition width FIR filter [8]. Echo cancellation requires digital signal processing which can operate on real time entity and for such systems performance is often limited by processing capability of implementation, adaptive filter has become a useful building block in several systems. The algorithm used for updating the filter coefficient of adaptive filter is Least Mean Square (LMS) algorithm [9]. Another commonly used algorithm for implementing FIR on FPGAs is using alternate number representations for the multiplier, such as Canonical Signed Digit (CSD) number system or Signed Power of Two (SPT) representation [10]. Recently, the development of the Short Word Length (SWL) algorithms is used for simpler hardware implementation with low cost and increased flexibility. The SWL method for implementing DSP systems has limited applications [11].

Design and Implementation:

A. Distributed Arithmetic:

To improve working speed of the system work way of pipeline can be used. When compared with the traditional algorithm the Distributed algorithm improves the speed of circuit and reduces hardware complexity. The design procedure for distributed algorithm is given below:

1. According to the filter specification filter coefficient are derived.
2. The input register is used to store the inputs value.
3. Design LUT to shows all possible combination of filter coefficient.
4. Accumulate and shift value with LSB of input to the right and add it to next partial result.
5. 1^{st} value must be subtracted from the result due to negative bit of MSB.

6. Analyse the output.

The coefficient FIR filter is h(n) that can be calculated by MATLAB the h(i)x(n-i) has become a constant multiplication, is the only input of scaling and this feature is prerequisite for the application of Distributed algorithm. The expression for the N order FIR filter is given below.

$$Y[n]=\sum i=0N\text{-}1h(i)x(n-i) \quad (1)$$

The equation of Distributed Algorithm for computing the sum of products is represented as

$$Y[n]=(c,x)= \sum i=0N-1c[n]x[n] = c(0)x(0)+ c(1)x(1)+ c(2)x(2)\ldots. c(N\text{-}1)x(N\text{-}1) \quad (2)$$

For Unsigned DA Let x[i] is the binary unsigned number B bits,

$$x(i)=\sum b=0B-1\ xb(i)\cdot 2\text{^}b \qquad xb\in[0,1] \quad (3)$$

It is observed that xb(i) is inner product of x(i), therefore from equation (2) and (3)

$$Y[n]=\sum i=0N-1c[i]\ x[\sum b= 0\ B-1\ \ xb(i)\cdot 2\text{^}b] \quad (4)$$

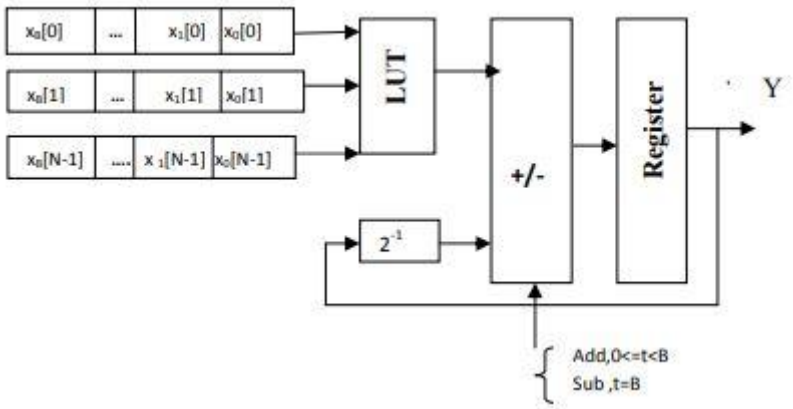
It can be abbreviated as

$$f[c(i),xb(i)] = \sum i=0N-1\ c(i)xb(i) \quad (5)$$

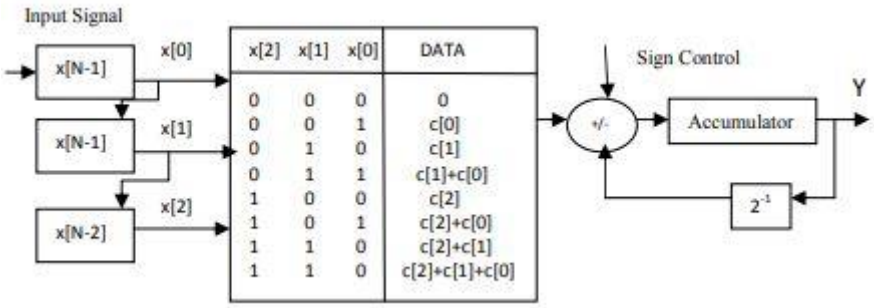
For Signed Distributed Algorithm, the (B+1) bits two's complement signed data x(i), the highest is that positive and negative.

$$x(i)=-2B\cdot xB(i) + \sum b = 0B-1xb(i)\cdot 2\text{^}b,\ xb(i) \qquad \in[0,1] \quad (6)$$

Distributed algorithms of complement are corrected only after the sign bit f[c(i),xb(i)], mapping values are weighted by its weight 2^B, it is the implementation of subtraction, other position by its weight 2^b can be accumulated.



Fig(a): DA Architecture



Fig(b): LUT based DA FIR filter

The DA filter require very small area, as only one LUT is in implementation due to 2^3 memory locations. The control unit clear the buffer and then input is collected by the input register during previously clock cycle serially injected to the circuit. These bits addresses a value in the input LUT structure and its partial result is accumulated and shifted by the shifter unit. Number of clock cycle depends on the number of bits in the inputs. The value of partial result gives output and this output is then shifted n times to the new value addressed in 4 input LUT units. The final result is observed on the (n+1) th clock cycle signal.

A.1 Dynamic Distributed Arithmetic (DDA):

When the system wants to operate using varying filter coefficients then Dynamic Distributed Arithmetic architecture is used. Distributed Arithmetic algorithm were FIR filter coefficient are constant. When there is requirement of change in filter coefficient the coefficient buffer is updated dynamically and perform defined operations. The coefficients are stored in buffer based on value of c[n], buffer then

generates address from the given inputs and then these are applied to LUT. Similarly, LUT get updated with filter coefficient based on the value of c[n] in the coefficient buffer. Once the shift register shifts the value to the accumulator. At last, the output of registered LUTs are added and loaded to the scaling accumulator from LSB to MSB and the result is accumulated on to the output register.

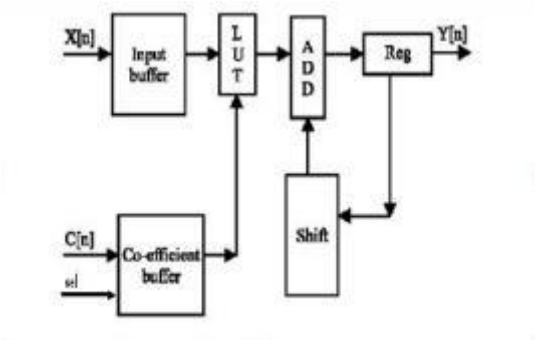


Fig c: Dynamic DA architecture

A.2 Distributed Arithmetic Offset Binary Coding (DA-OBC)

The DA-OBC algorithm is proposed and synthesized under the integrated environment of maxplus2. The simulation results using this algorithm show implementation works well and meet requirements. The DA is famous method of implementing FIR filters on FPGSs. The design procedure for DA is mentioned above in this paper, the DA algorithm greatly reduces complicated Multiply and Accumulate (MAC) operations to Look-up-Table at the cost of increasing storage area. The scale of LUT is reduced to half using DA-OBC algorithm and the expression for it is as follows

$$x_i = \frac{1}{2}[x_i - (-x_i)] = \frac{1}{2}[-(x_{i,m-1} - \overline{x_{i,m-1}}) + \sum_{j=1}^{m-1}(x_{i,m-1-j} - \overline{x_{i,m-1-j}})2^{-j} - 2^{-(m-1)}] \quad (7)$$

Equation (7) can be simplified as

$$x_i = \frac{1}{2}[\sum_{j=0}^{m-1} d_{i,m-1-j}2^{-j} - 2^{-(m-1)}] \quad (8)$$

Final expression after substituting Equation (8) in equation (1) we get

$$y = \sum_{j=0}^{m-1}(\sum_{i=0}^{N-1}\frac{1}{2}h_i d_{i,m-1-j})2^{-j} - (\frac{1}{2}\sum_{j=0}^{N-1}h_j)2^{-m-1} \tag{9}$$

For simplicity we can define as

$$D_j = \sum_{i=0}^{N-1}\frac{1}{2}h_i d_{i,m-1-j} D_{ex} = -\frac{1}{2}\sum_{j=0}^{N-1}h_j \tag{10}$$

TABLE I. LUT CONTENTS OF DA-OBC.(N=4)

Address $x_{1j}x_{2j}x_{3j}$ ($x_{0j}=0$)	LUT contents	Address $x_{1j}x_{2j}x_{3j}$ ($x_{0j}=1$)	LUT contents
0 0 0	$\frac{(h_0+h_1+h_2+h_3)}{2}$	1 1 1	$\frac{(h_0+h_1+h_2+h_3)}{2}$
0 0 1	$\frac{(h_0+h_1+h_2-h_3)}{2}$	1 1 0	$\frac{(h_0+h_1+h_2-h_3)}{2}$
0 1 0	$\frac{(h_0+h_1-h_2+h_3)}{2}$	1 0 1	$\frac{(h_0-h_1+h_2+h_3)}{2}$
0 1 1	$\frac{(h_0+h_1-h_2-h_3)}{2}$	1 0 0	$\frac{(h_0+h_1-h_2-h_3)}{2}$
1 0 0	$\frac{(h_0-h_1+h_2+h_3)}{2}$	0 1 0	$\frac{(h_0-h_1+h_2-h_3)}{2}$
1 0 1	$\frac{(h_0-h_1+h_2-h_3)}{2}$	1 0 1	$\frac{(h_0+h_1-h_2+h_3)}{2}$
1 1 0	$\frac{(h_0-h_1-h_2+h_3)}{2}$	0 0 1	$\frac{(h_0-h_1-h_2+h_3)}{7}$
1	$\frac{(h_0-h_1-h_2-h_3)}{2}$	0 0 0	$\frac{(h_0-h_1-h_2-h_3)}{2}$

The size of table can be reduced by half due the fact is that right half of LUT is mirror version of left half of LUT. The OBC coding unit generates the address of two LUTS which stores the precomputed value. The out of LUTS is sent to shifter and adder using sign control. The architecture is designed with VHDL, is simulated in function and synthesized under the integrated environment of maxplus 2. The Simulation waveform is shown below

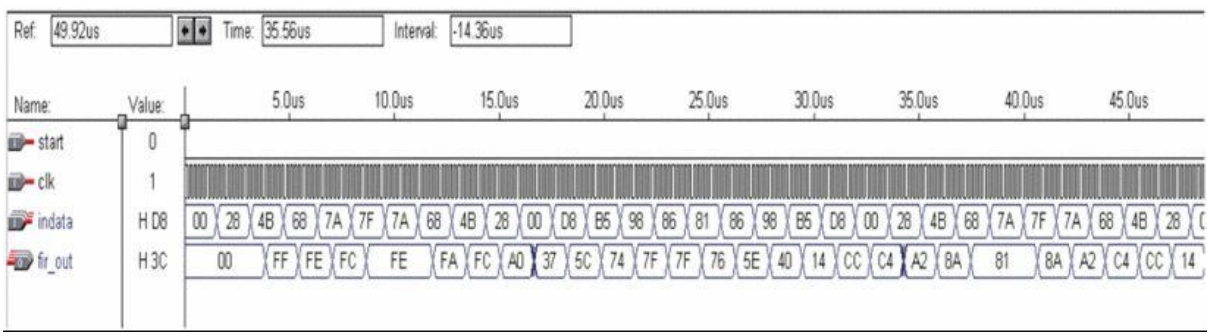


Fig d. The simulation waveform of FIR filter

The design of FIR filter works well because the error of simulation result is very small. The implementation of FIR filter base on FPGA was realized on EPF10K30EQ208–1 chips and need 272 LC.

A.3 DA based re-configurable filter on FPGA

A high-speed reconfigurable design and implementation n FIR filters whose filter coefficient change during execution time is proposed was proposed [1] [2]. To achieve the best configuration in terms of coefficient of FIR filter, the storage resource and calculating field the DA algorithm is optimized and improvised in terms of algorithm, structure, memory and LUT. FPGA is one of the fastest growing technology and it is considered one of the popular choices in terms of communication base station instead of just being a prototype. These papers also show the reconfigurable DA-based FIR filter for FPGA implementation. The number of registers for the implementation of LUT for FIR filter of length N is designed. But, registers are scarce resources in FPGA since each LUT in many FPGA devices contains only two bits of registers. Therefore, LUT are required to be implemented by either Distributed RAM (DRAM) for FPGA implementation or using reconfigurable LUT (RLUT). The RLUT has one drawback the output of each CFGLUT is invalid during implementation. To overcome this drawback RLUT implementation uses two CFGLUTs instead of one to create glitch free reconfiguration architecture.

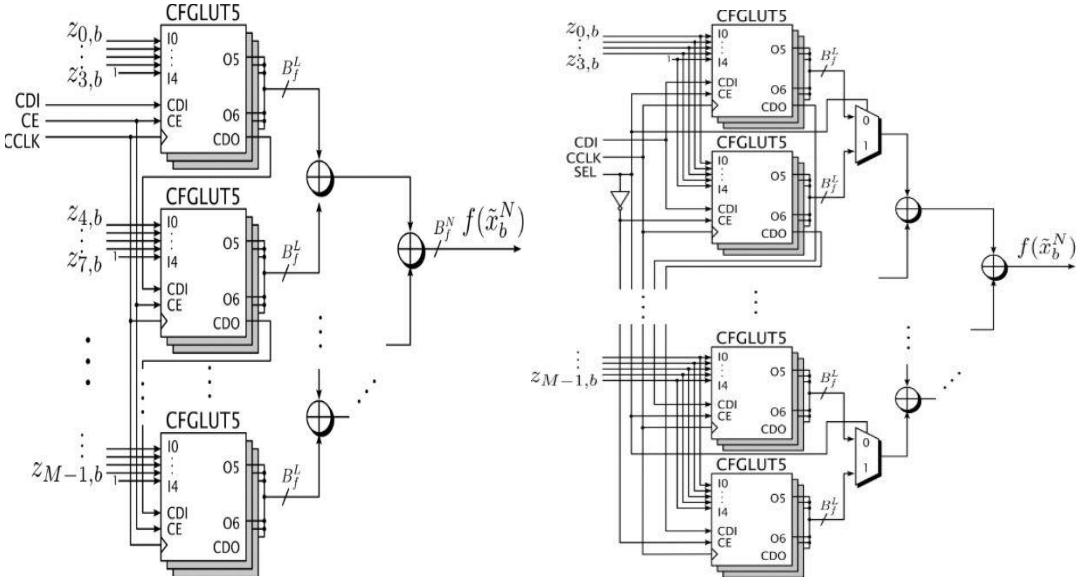


Fig 4: reconfigurable LUT realization of f(x~Nb) using single CFGLUT5 primitives for the resource minimized DA architecture

Fig5: A reconfigurable realization of f(x~Nb) using twice as many CFGLUT5 primitives for the glitch free reconfiguration architecture.

A reconfigurable FIR filter based on distributed arithmetic was presented which can be reconfigured with an arbitrary large number of filters which is only limited by configuration memory. The hardware cost could be extensively reduced by allocating same registers by the DA units for different bit slices.

B. Filter Design based on Kaiser Window Function:

Based on Liu Xiongfei's research, a new system modelling method for designing the FIR filter based on DSP Builder was proposed in this paper, through this way the filter performance and development efficient has been greatly improved. In an ideal filter, frequency response function h(n) is an infinite and non-casual sequence, but in an actual designed filter, h(n) must be finite. So in the practical application use of finite length window function sequence CD (n) to intercept, that is, to approximate the infinite as follows:

$$\mathrm{h}(n) = \omega(\mathrm{n})\mathrm{h}_d(n)$$

Among the usually used function such as rectangular window, Bartlett window, Hamming window, Hann window, Blackman window and so on. Kaiser window provides a variable transition bandwidth and it is selected to design FIR filter in this paper [6]. Window function is represented as

$$\omega(n) = \frac{I_0(\beta\sqrt{1-(1-\frac{2n}{N-1})^2})}{I_0(\beta)}$$

Design specifications: designing a low-pass FIR filter with a cut-off frequency of 100 KHz and a sampling frequency of 1MHz based on Kaiser window function. In the figure below the gain is about −6dB, then decreases rapidly to less than −20dB.

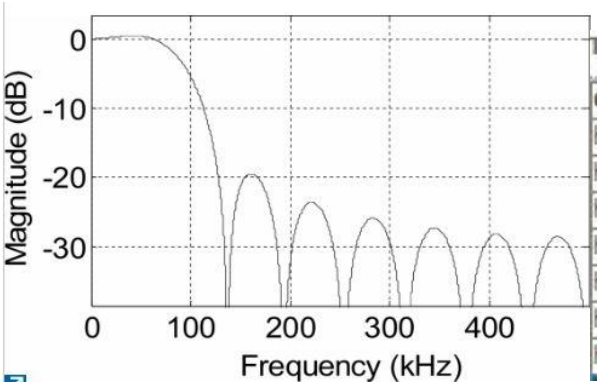


Table 1: Coefficients of the designed filter

Coefficient	Value	Coefficient	Value
h0	−0.04045	h8	0.19933
h1	−0.03835	h9	0.17357
h2	−0.01754	h10	0.12814
h3	0.02167	h 11	0.07359
h4	0.07359	h12	0.02167
h5	0.12817	h13	−0.01754
h6	0.17357	h14	−0.03835

The above graph indicates the amplitude-frequency response characteristics curve of designed FIR filter and the table shows the filter's performance meet the requirement, the designed filter must be not less than 16-tap and its coefficient were symmetric, and it was a linear phase filter. Using the new FPGA development software DSP Builder developed by Altera company, a 16-tap direct form FIR filter model was built. The frequency spectrums of input and output signal are shown below.

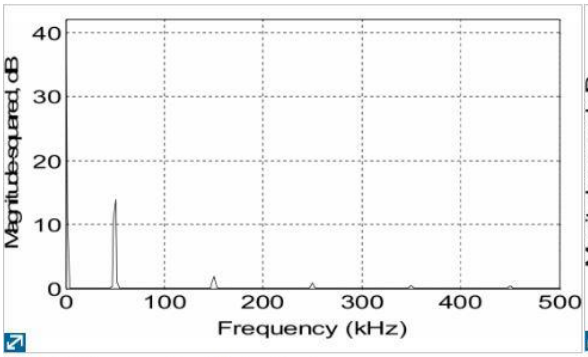


Figure 4. The frequency spectrum of filter input signal

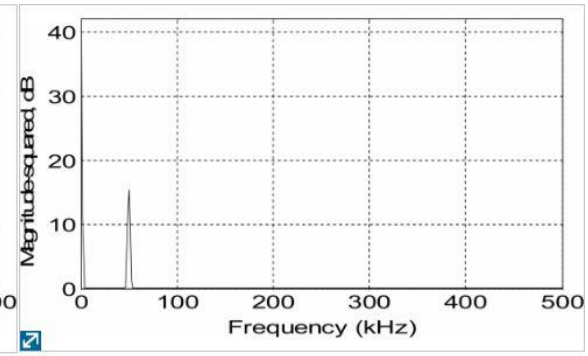


Figure 5. The frequency spectrum of filter output signal

Using Kaiser window function based on DSP builder system FIR low-pass filter was implemented on FPGA. The performance and designing efficiency was far better than traditional way. DSP builder approach accelerates FPGA development.

C. High speed Adaptive FIR filter on a FPGA:

One of the most common application being line echo cancellation (LEC). Due to increasing data rates in such communication system the need of adaptive filtering system that can handle the echo tail generated. To ensure efficient functionality and more processing power this algorithm is robust and stable. This paper describes the implementation of such an adaptive

filter on a Xilinx Spartan 3 FPGA for use in an echo cancellation system. An adaptive filter used for the removal of a local echo generated in a microwave point-to-point link was developed on an FPGA. Conventional, non-adaptive, filters that are used for the extraction of information from an input time series, x(n), are usually linear time invariant meaning that they perform the same set of linear operations on the time series x(n) to provide the output, irrespective of the value of n. In adaptive filtering this is not the case since this restriction on time invariance is removed by allowing the coefficients used in the linear filtering operation to change according to some predetermined optimization criterion [9].

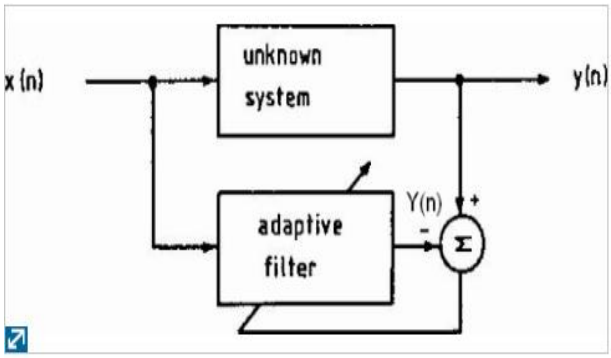


Fig. 2(a): System Identification using an adaptive Filter [3].

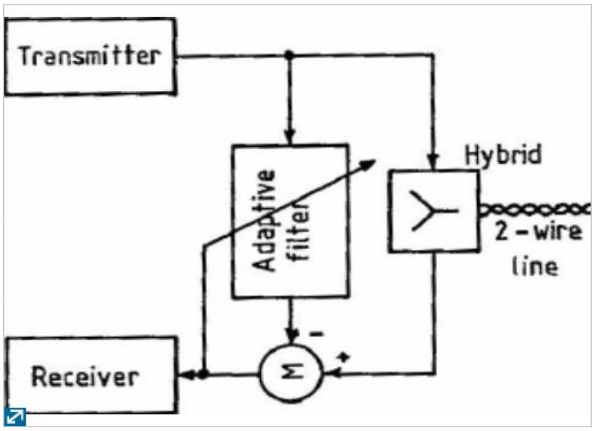


Fig. 2(c) Echo Cancellation using an Adaptive Filter [3].

In the setup shown above, the output of the adaptive filter is subtracted from the output of the hybrid (Echoed data) and an error term is fed into the receiver [9].

C.1 LMS Algorithm

The LMS algorithm is preferred because it removes the requirement of having explicit knowledge of auto-correlation and cross correlation. The LMS algorithm utilizes time recursion method which uses an estimate of gradient rather than gradient itself.

$$\underline{h}(k+1) = \underline{h}(k) - \mu\hat{\underline{\nabla}}(k) \tag{1}$$

The vector h(k) is an estimate for the Wiener filter estimate h_{opt}, the latter being the filter estimate that minimises the cosine t function. The vector ∇^—(k) is an estimate of the gradient ∇—(k) of the Mean Square Error cost function at the point where the impulse response is h(k). The variable μ is the convergence factor of the system and for ensuring convergence this value must fall within certain bounds (to be defined later) [9].

$$\underline{\nabla}(k) = \frac{\partial \xi}{\partial \underline{h}}(\underline{h}(k)) = \frac{\partial}{\partial \underline{h}} E\left[(y(n) - \underline{h}^{\tau}(k)\underline{x}(n))^2\right] \tag{2}$$

In the formation of the estimate ∇^—(k) of the actual gradient ∇—(k) the group average of Eqn. 3 is replaced by a time average over n. Since h(k) changes at each data point, the time average is reduced to a single value at n=k+1 which yields.

$$\hat{\underline{\nabla}}(k) = -2\underline{x}(k+1)e(k+1) \tag{3}$$

Substituting equation (3) into equation (1) gives LMS algorithm.

$$\underline{h}(k+1) = \underline{h}(k) + 2\mu\underline{x}(k+1)e(k+1) \tag{4}$$

For coding purpose and simplicity otherwise, it would represent time advance, the above equation is used as

$$h(k+1) = h(k) + 2\mu e(k)x(k)$$

The block diagram of final implementation is shown below. The Digital Phase Locked Loop (DPLL) ensures all stages running at constant lock speed. The output block (a) is used by FSK modulator. The clock denoted (b) is subtractor to remove echo from the link d[n], from the filter output, y[n], to give the error term, e[n], which is then fed to the LMS block. The latter uses e[n] to generate the new set of filter coefficients. There is a continuous exchange between the LMS block and the filter ensuring that both move in lock-step [9].

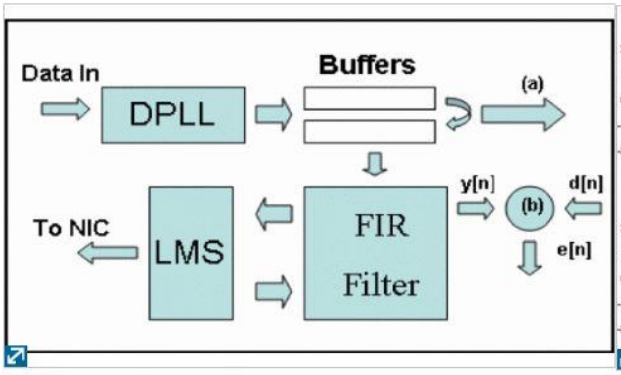


Fig. 4: Block diagram of implementation

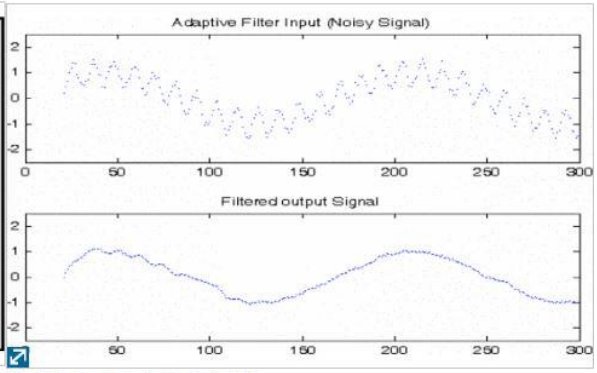


Fig. 3: Test Results obtained using Matlab®

The graph mentioned above indicates the result after implementation of LMS algorithm. It confirms that system performs well in obtaining original signal from noisy input signal. The overall system performs well and the echo generated by transceivers was cancelled out by the adaptive filter without any loss.

D. CSD Implementation in Filter Implementation

In these implementation, rather than implementing multiplication of inputs by coefficients using multipliers, the multiplication takes advantage of the *a priori* knowledge of the coefficient values to implement the multiplication by a series of shifts and adds/subtracts [10]. The number of add/shift operations is directly related to the power consumption and area required by the implementation. When the multiplier is known a*priori* a more efficient multiplier can be achieved by using CSD number representations.

The most commonly used number representation this type of multiplier less implementation of filters is the canonical signed digit (CSD) representation [10]. CSD are represented in radix-two number system which has canonical property .i.e. no two consecutive bits in CSD number are non-zero. CSD numbers have proven to be useful in implementing multipliers with less complexity, because the cost of multiplication is a direct function of the number of nonzero bits in the multiplier, which can be reduced by using CSD numbers. The total number of bits required for the design for the example used in this paper is given below.

Table 2. The results of filter order, wordlength of coefficients required, total number of binary and nonzero CSD bits, when stopband attenuation is changed (not including the sign bit)

A_s (dB)	Order	Number of bits per coefficient	Total number of binary bits	Total number of nonzero CSD bits
80	178	20	3580	758
	183	18	3312	624
	189	17	3230	600
70	149	19	2850	626
	152	18	2754	580
	155	16	2496	486
65	134	18	2430	520
	138	16	2224	450
	142	15	2145	430
55	105	17	1802	404
	108	14	1526	304
	114	13	1495	287
50	90	19	1729	430
	95	13	1248	258

Table: The results of filter order, word length of coefficients required, total number of binary and nonzero CSD bits, when stopband attenuation is changed (not including the sign bit).

The trade-off between filter order and coefficient length is shown in this paper. Non-minimum order FIR filters are designed for implementation using canonical signed digit (CSD) multiplierless implementation techniques. By using non-minimum order designs, the length of the coefficients can be reduced, and thus an overall hardware savings can be achieved [10].

E. Implementation of Short Word Length Ternary FIR filter in FPGA

This technique is demonstrated as one of the new efficient approach for implementing DSP systems. The paper [11] represents design and implementation of Sigma-delta modulator (ΣΔM) based on SWL ternary FIR filter. The system using SWL was first simulated in Matlab and then implemented on FPGA. Also, to examine the trade-off between hardware efficiency and performance, they also evaluated the design with four different oversampling rates (8, 16, 32, 64). Sigma Delta Modulator (ΣΔM) used in SWL technique oversample the input signal and produce single-bit input data. The input bit can be represented using 1-bit binary stream and then process using SWL blocks. Like a conventional FIR filter, a SWL FIR filter is described by a convolution equation between the binary input signal with a coefficient.

$$y(k) = \sum_{i=0}^{N} h_i x(k - i) \tag{1}$$

In above equation N is the order of filter and x and h are input signal coefficient. For hardware implementation, the following requirements were applied:

- Hardware designed using Verilog language and synthesized with 90nm technology node for both FPGA and ASIC.
- Supported pipeline structure.
- Maximum frequency is analyzed in the range between 64 MHz and 250 MHz.
- The Hardware/Software co-design flow was adopted in this project:

- Filter modelling and coefficients generated using MATLAB.
- Hardware designed using Verilog.
- Test environment development and then verification [11].

For filter modelling and coefficient generation using MATLAB steps are shown below in the black diagram.

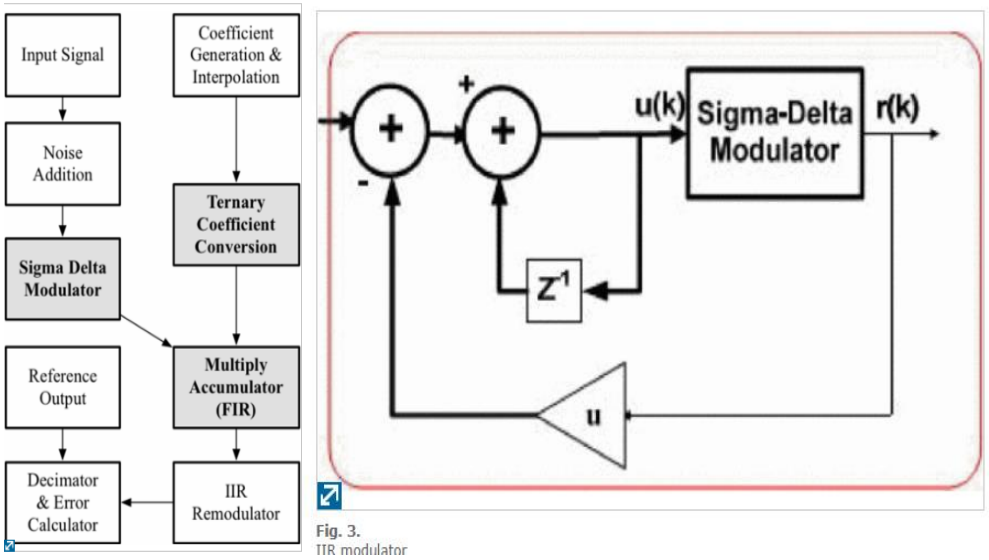


Fig. 3.
IIR modulator

The MAC block multiply and accumulate all coefficients and it is given as input to calculate output. The IIR modulator shown above is used to convert Multi-bit output from MAC into single bit.

Table II. FPGA performance summary

OSR		Total Logic Element	Total Registers	Total Memory Bits	Fmax (MHz)
8	Pipeline	4140	3601	1024	176.68
	Non-pipeline	4015	1566	1024	61.93
16	Pipeline	8240	7181	2048	164.85
	Non-pipeline	8061	3105	2048	53.94
32	Pipeline	16438	14357	4096	148.10
	Non-pipeline	16113	6180	4096	46.71
64	Pipeline	32833	28695	8192	137.17
	Non-pipeline	32161	12327	8192	43.33

It is observed for SWL applications, Oversampling Rates (OSR) does affect significantly the hardware implementation hence the hardware performance was analysed with varying

Oversampling rates as shown in table above. This paper proves that SWL has been an effective approach for simplified hardware implementations for DSP algorithms. In this paper the performance of system was compared using both FPGA and ASIC and it showed that ASIC is more flexible than FPGA. However, in terms of performance FPGA did not outperform due to limitation of current target ASIC library.

Conclusion:

In this paper different techniques are described for design and Implementation of FIR filter on FPGA. Filter design based on FPGA is one of the preferred method for high order digital filter. Basically, all algorithms aim is to improve circuit performance, reduce complexity, obtain accuracy and speed. Different tools were used for comparing test results and verified by simulation, all tools used are different depending on paper and the logic used by the Authors. The design and implementation of Distributed Arithmetic (DA) algorithm was represented on low pass filter but it can be implemented on high pass and band pass filter by simply recombining filter parameters. The DDA algorithm provided flexibility and can be applied on variable filter coefficient as per requirement. The Implementation of DA-OBC algorithm provided good flexibility and portability and it was applied to high-pass, band-pass and band-stop filters by changing their coefficients. The new idea of modelling approach of DSP builder system using Kaiser Window designing method showed improvement in designing efficiency and filter's performance as compared to traditional methods. The use of LMS algorithm on the echo generated by transceivers proved no loss of data. Non-minimum FIR filters are designed and implemented using CSD multiplierless implementation techniques due to which the length of coefficient reduced. The latest technique used for design and implementation of FIR is Short Word Length (SWL). In this paper the technique was implemented on both FPGA and ASIC and proved that FPGA performance was good as compared to ASIC. The use of delta sigma modulation and ternary coefficients reduced size of adder and multiplier blocks.

References

[1] B. N., R. K. R and N. K. R, "Design and implementation of DA-based reconfigurable FIR digital filter on FPGA," in *2015 International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT)*, Mandya, India, 2015.

[2] M. Kumm, K. Möller and P. Zipf, "Reconfigurable FIR filter using distributed arithmetic on FPGAs," in *IEEE*, Beijing, China, 2013.

[3] T. R. Kumar, "Design and implementation of DDA architectures using FIR Filters," in *International Journal of Engineering Trends and Technology (IJETT)*, Kukatpally, Hyderabad, 2013.

[4] X. Liangfen and L. Yi, "Implementation of distributed FIR digital filter on FPGA," in *IEEE 2011 10th International Conference on Electronic Measurement & Instruments*, Chengdu, China, 2011.

[5] B. Hong, H. Yin, X. Wang and Y. Xiao, "Implementation of FIR filter on FPGA using DAOBC algorithm," in *IEEE*, Hangzhou, China, 2010.

[6] G. Jinding, H. Yubao and S. Long, "Design and FPGA Implementation of Linear FIR Low-pass Filter Based on Kaiser Window Function," in *IEEE*, Shenzhen, Guangdong, China, 2011.

[7] L. K. Phimu and M. Kumar, "Design and implementation of area efficient 2-parallel filter on FPGA using image system," in *2017 International Conference on Innovative Research In Electrical Sciences (IICIRES)*, Nagapattinam, India, 2017.

[8] Y. Lian, "FPGA implementation of high speed multiplierless frequency response masking FIR filters," in *IEEE*, Lafayette, LA, USA, USA, 2000.

[9] M. Vella and C. Debono, "The Implementation of a High Speed Adaptive FIR Filter on a Field Programmable Gate Array," in *MELECON 2006 - 2006 IEEE Mediterranean Electrotechnical Conference*, Malaga, Spain, 2006.

[10] L. S. DeBrunner and Y. Wang, "Optimizing Filter Order and Coefficient Length in the Design of High Performance Fir Filters for High Throughput FPGA Implementations," in *2006 IEEE 12th Digital Signal Processing Workshop & 4th IEEE Signal Processing Education Workshop*, Teton National Park, WY, USA, 2006.

[11] T. C. Pham, B. X. Hoang, Q. T. Chiem, L. D. Tran and A.-V. Ho, "Implementation of a short word length ternary FIR filter in both FPGA and ASIC," in *2018 2nd International Conference on*

Recent Advances in Signal Processing, Telecommunications & Computing (SigTelCom), Ho Chi Minh City, Vietnam, 2018.

[12] J. H. G. Jiafeng Xie, "FPGA realization of FIR filters for high-speed and medium-speed by using modified distributed arithmetic architectures," *Microelectronics Journal,* vol. 41, no. 6, pp. 365-370, 2010.

[13] U. Meyer-Baese, Digital Signal Processing with Field Programmable Gate Arrays, 3rd ed., Berlin: Springer, 2007.

[14] C. Guo-wei and W. Feng-ying, "The implementation of FIR low-pass filter based on FPGA and DA," in *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, Beijing, China, 2013.

[15] K. Vinger and J. Torresen, "Implementing evolution of FIR-filters efficiently in an FPGA," in *NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings.*, Chicago, IL, USA, USA, 2003.