Digital System Design

we gather, we share, you learn..

www.digitalsystemdesign.in

Parallel FPGA Implementation of FIR Filters

Shirshendu Roy

April 28, 2020

Abstract

Finite Impulse Response (FIR) filters are very crucial block in signal processing applications. In comparison to the Infinite Impulse Response (IIR) filters, FIR filters are always stable and also FIR filters provide linear phase response. There are many structures reported in literature to implement FIR filters efficiently and Field Programmable Gate Array (FPGA) device is a very popular platform for rapid prototyping of the FIR filters. In this work, a low pass FIR filter is designed using different parallel FIR filter structures and performances different structures are compared. Root Mean Squared Error (RMSE) is used to estimated the performance.

Keywords: Field Programmable Gate Array (FPGA), Finite Impulse Response (FIR), Direct Form, Linear Phase, Cascaded structure, Polyphase structure.

1 Introduction

The impulse response of a Finite Impulse Response (FIR) filter is of finite duration as it settles to zero in finite time. In comparison to the Infinite Impulse Response (IIR) filters, there is no feedback in FIR filter. This feature makes the FIR filter always stable. Another important feature of FIR filter is that it can produce linear phases and thus in application where linear phase should be used, FIR filters must be preferred. Implementation of FIR filters is also straightforward compare to the design of IIR filters.

Field Programmable Gate Array (FPGA) is a platform which is used to implement the FIR filters for many signal processing applications. Many FPGA implementations are reported in literature to implement FIR filters for many different applications. The FPGA manufacturing companies have also provided many advanced features for rapid prototyping of the FIR filters. The advanced DSP blocks can perform many mathematical functions with greater speed. Thus implementation of FIR filters is no longer a critical job.

The objective of this work to cover the aspects of implementation of FIR filters. In this work, FPGA implementation of a low pass FIR filter using different structures is presented. This filter is implemented with or without the advanced DSP blocks. Performance of all the structures is also compared in terms of resource utilization, latency and maximum frequency.

The manuscript is organized as follows: Section 2 describes transfer function of the basic low pass FIR filter. In the section 3, the advance DSP block is described. The low pass FIR filter is implemented using different structures in the section 4. In the Section 5, performance estimation is shown. Here performances of all the structures are compared. In the section 6, conclusions of this work are given.

2 FIR Low Pass Filter

In this work, design of a low pass filter is taken as an example to illustrate the implementation of FIR filter. The frequency response of a ideal low pass filter is

$$H(e^{jw}) = \begin{cases} 1 & \text{for } -\frac{\pi}{2} \leq w \leq \frac{\pi}{2} \\ 0 & \text{for } -\frac{\pi}{2} \leq |w| \leq \pi \end{cases} \tag{1}$$

Here, w is the normalized frequency and $w = \frac{\pi}{2}$ denotes the cutoff frequency in radian. This ideal low pass FIR filter can be realized using many techniques as illustrated in [2]. Here Hamming window based design is followed.

The transfer function $H(z) = y(z)/x(z)$ of a N tap FIR filter can be written as

$$H(z) = \sum_{n=0}^{N-1} c_n z^{-n} \tag{2}$$

Here, c_n is the co-efficients of the transfer function and z^{-n} denotes the delay by n samples. Here, n can be both even or odd. The time domain expression of the low pass FIR filter is shown below for $N = 13$.

$$y = x.(c_0 + c_1.z^{-1} + c_2.z^{-2} + c_3.z^{-3} + c_4.z^{-4} + ... + c_{12}.z^{-12}) \tag{3}$$

Here, there are $(N-1)$ delay elements and N co-efficients. The frequency plot of the low pass FIR filter is shown in Figure 1.

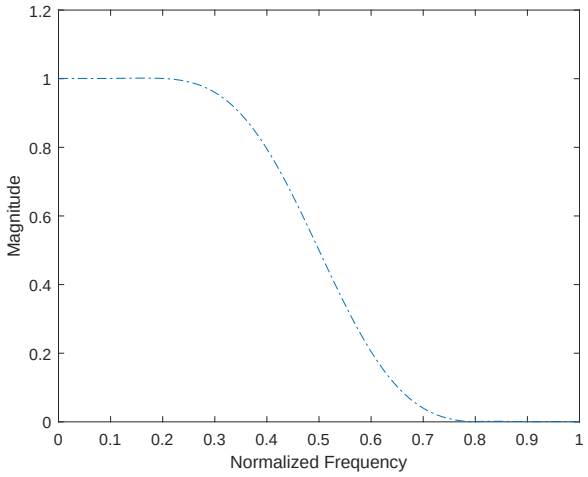


Figure 1: Frequency Spectrum of the Low Pass Filter.

3 Advanced DSP Blocks

The advanced FPGA devices provide high speed blocks to evaluate different arithmetic operations involved in signal processing. These blocks are popularly known as Digital Signal Processing (DSP) blocks. For example, the ARTIX7 FPGA device provides DSP48 [1] blocks. These DSP blocks incorporate high speed algorithms to perform addition, multiply, multiply-accumulate or shifting operations. A simple block diagram of a DSP block is shown in Figure 2. Here the DSP block is shown to perform the operations which are useful in realizing FIR filters.

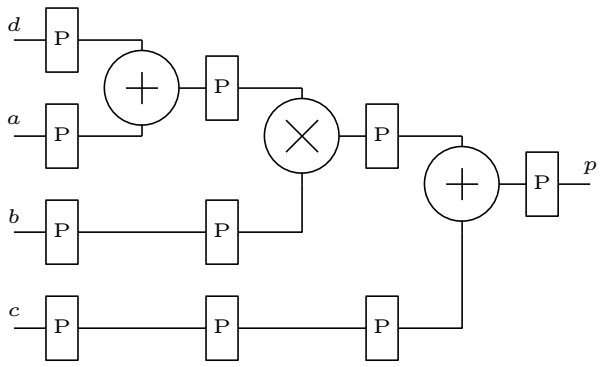


Figure 2: A simplified architecture of a DSP block.

The DSP block includes a pre-adder, a multiplier and an ALU. The ALU can be used to realize various functions but here ALU performs only addition or subtraction. The DSP block has a *sel* input and based on the status on this line the DSP block performs many different functions. For example, the DSP block shown here can evaluate $p = c \pm a \times b$ or $p = c \pm (a + d) \times b$. The pipeline registers are programmable, means they can be inserted or removed or increased. This DSP blocks are inbuilt and thus provides faster speed than the implementations using the LUTs.

4 Different Filter Structures

In literature, many different structures [2] to implement the FIR filters are reported and this structures are

1. Direct Form Structures
2. Linear Phase Structures
3. Polyphase Structures
4. Cascaded Structures
5. Lattice Structures

In this work, a low pass filter is implemented using Direct Form, Linear Phase Form, Polyphase Form and Cascaded Form. Lattice structures are extensively used in speech processing and are costly. Thus it is not discussed here but other structures are discussed in details with their FPGA implementation.

4.1 Direct Form Structures

In direct form structures, the multiplier coefficients are precisely the coefficients of the transfer function. A N-tap FIR filter, requires N multipliers and $(N-1)$ two-input adders. The direct form structure of the low pass filter is shown in Figure 3 and it is called as Direct Form 2 structure. Here, the discrete delay elements and the pipeline registers are shown separately. The pipeline registers are shown by 'P'. Here the maximum operating frequency is limited by the multiplier and the adder path. Combination of the adder and the multiplier can be implemented using DSP blocks as shown in Figure 3.

The structure shown in Figure 3, has latency of 12 clock cycles. The direct form structure can be implemented in another way as shown in Figure 4 to reduce the latency and it is called here as Direct Form 1. Here the output of the multipliers are fed to an adder tree. The adder

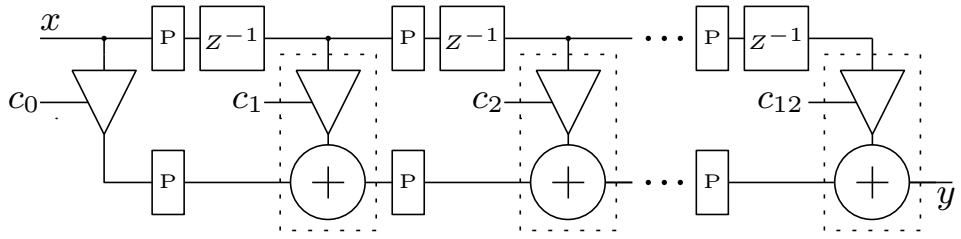


Figure 3: Direct form architecture of the FIR Low Pass Filter.

tree has less latency. This architecture uses same number of adders and multipliers but has latency of only three clok cycles. The DSP blocks also can be used here as shown in Figure 4.

The structure shown in Figure 4 is the most direct way to implement FIR filters whereas

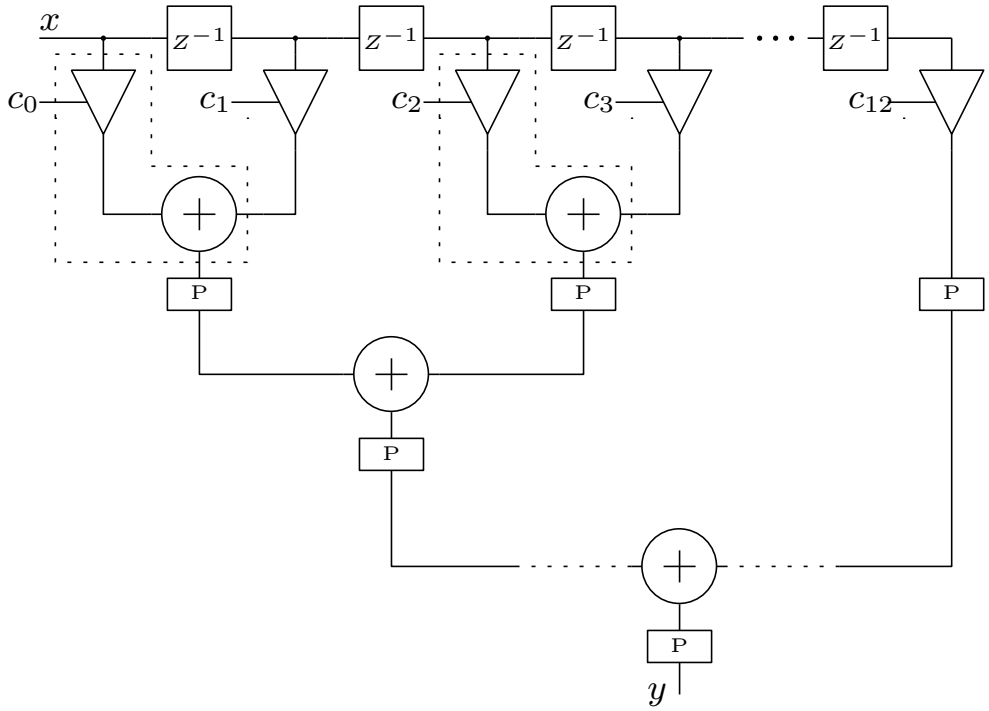


Figure 4: Another Implementation of the Direct form architecture of the FIR Low Pass Filter.

the architecture shown in Figure 3 is the systolic array structure. An inverted structure is also possible which is in Figure 5 and it is called here as Direct Form 3. The inverted structure has very less latency compared to other two structures. The major advantage of the inverted structure is that the pipeline registers are not required compared to the other two structures.

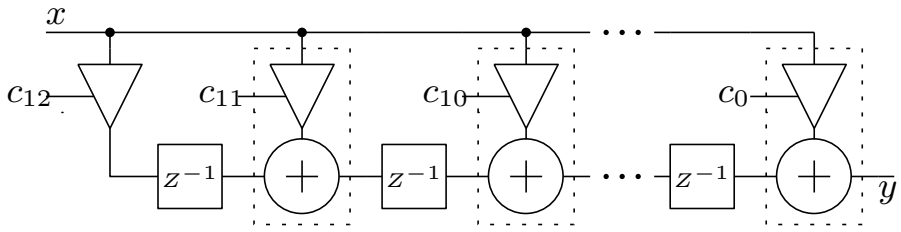


Figure 5: Transpose of the Direct form architecture of the FIR Low Pass Filter.

4.2 Linear Phase Structures

If the transfer function of FIR filter is symmetric or asymmetric then the filter can be implemented more efficiently. This symmetry or asymmetry property can be written as

$$c_n = \pm c_{(N-1-n)} \tag{4}$$

This property can be used to reduce the multipliers by half of that in the direct form implementations. The transfer function can be written as

$$H(z) = \begin{cases} \sum_{n=0}^{\frac{N-2}{2}} [z^{-n} + z^{-(N-1-n)}], & \text{When N is even.} \\ c_{(\frac{N-1}{2})} z^{-(N-1)/2} + \sum_{n=0}^{\frac{N-3}{2}} [z^{-n} + z^{-(N-1-n)}], & \text{When N is odd.} \end{cases} \tag{5}$$

The low pass filter implemented using the linear phase structure is shown in Figure 6 and it is called as Linear Phase 1 structure. In this structure, the critical path of a multiplier and an adder is retained. Thus more pipeline registers are required to achieve high frequency.

The structure shown in Figure 6 is modified in Figure 7 and this structure is called as Linear

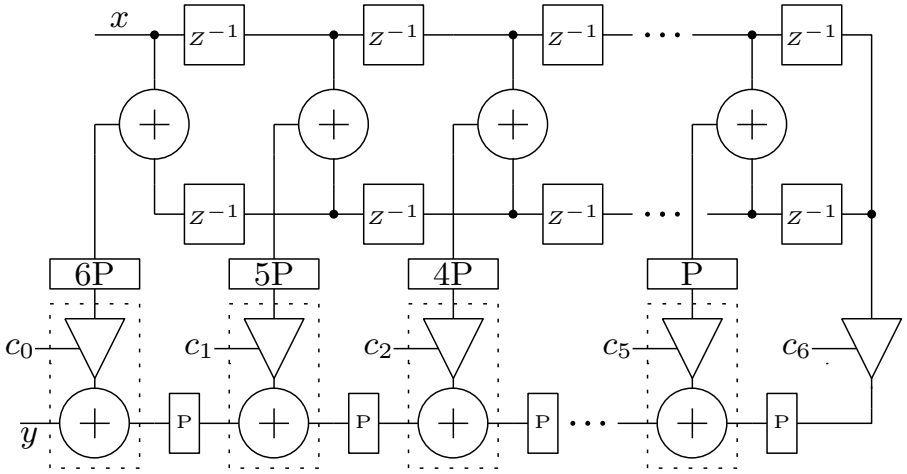


Figure 6: Linear Phase Architecture of the FIR Low Pass Filter.

Phase 2 structure. It is the transposed architecture of the previous linear phase structure. Thus requirement of the pipeline registers is less. Here in thus structure, the critical path is adder-multiplier-adder. The maximum frequency achieved for this structure is less than that for the previous structure. This structure is again implemented using the DSP blocks to improve the frequency performance and we named it as Linear Phase 3 structure.

4.3 Polyphase Structures

The transfer function of a FIR filter can be written as summation of two terms where a term contains all the even indexed co-effcients and the other term contains odd indexed co-efficients. For example the transfer function of the low pass filter can be expressed as

$$H(z) = (c_0 + c_2 z^{-2} + c_4 z^{-4} + ... + c_{12} z^{-12}) + (c_1 z^{-1} + c_3 z^{-3} + ... + c_{11} z^{-11}) \tag{6}$$

This equation can also be written as

$$H(z) = (c_0 + c_2 z^{-2} + c_4 z^{-4} + ... + c_{12} z^{-12}) + z^{-1}(c_1 + c_3 z^{-2} + ... + c_{11} z^{-10}) = P_0(z^2) + z^{-1} P_1(z^2) \tag{7}$$

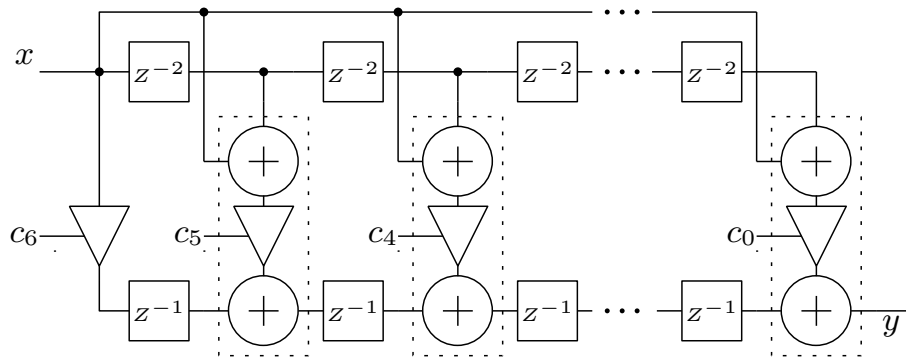


Figure 7: Transposed Linear Phase Architecture of the FIR Low Pass Filter.

In general there can be more number of branches where each branch can be realized using direct form. This type of structure is known as Polyphase structure and shown in Figure 8 (Polyphase 1). This type of structures has many uses in multi-rate signal processing and also has less latency than the direct form. The structure shown in Figure 9 is also the Polyphase structure but with sharing the delay elements and called as Polyphase 2 structure.

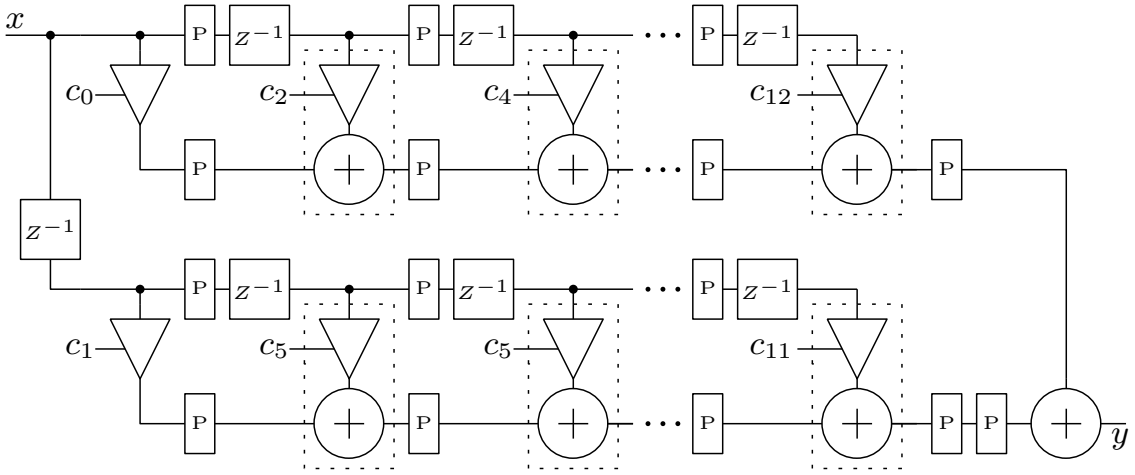


Figure 8: Polyphase structure of the FIR Low Pass Filter.

4.4 Cascade Structures

Higher order FIR filters can be realized in the cascaded form also. In the cascaded form, the higher order transfer function is realized by cascading lower order FIR sections where each section realizes either a first order or second order transfer function. A higher order transfer function can be written as

$$H(z) = \begin{cases} (c_{10} + c_{11}z^{-1}) \prod_{n=2}^{\frac{N}{2}} (c_{n0} + c_{n1}z^{-1} + c_{n2}z^{-2}), & \text{When N is even.} \\ \prod_{n=1}^{\frac{N-1}{2}} (c_{n0} + c_{n1}z^{-1} + c_{n2}z^{-2}), & \text{When N is odd.} \end{cases} \tag{8}$$

Here the higher order transfer function is factorized into second order transfer functions. The second order realization is shown in Figure 10. The cascade realization of the low pass filter is shown in Figure 11. Here total $(N-1)$ adders are used and $(N+5)$ multipliers are used. Unlike the other implementations, in the cascade implementation the co-efficients are modified. The

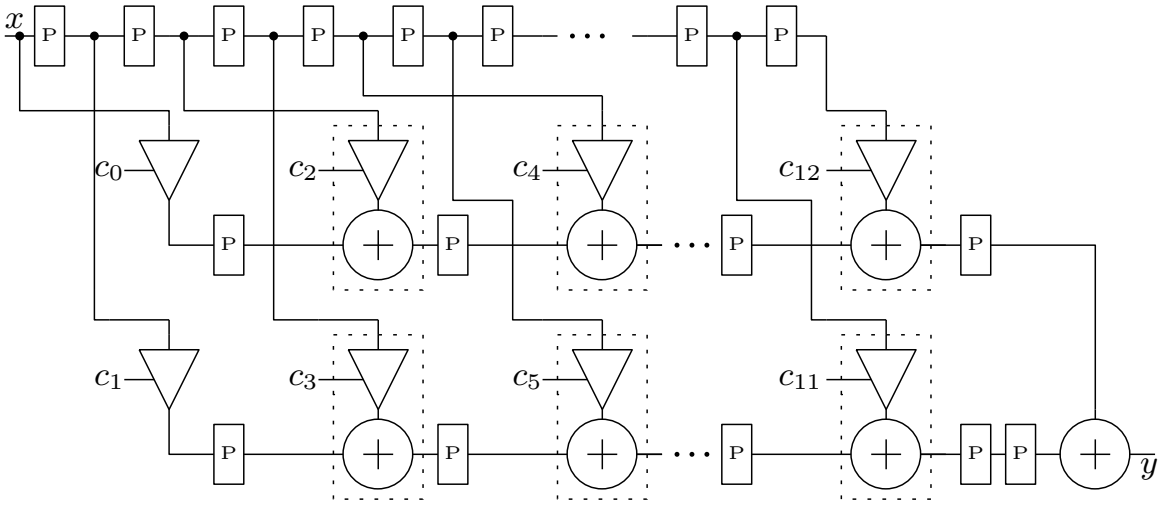


Figure 9: Polyphase structure of the FIR Low Pass Filter by sharing delay elements.

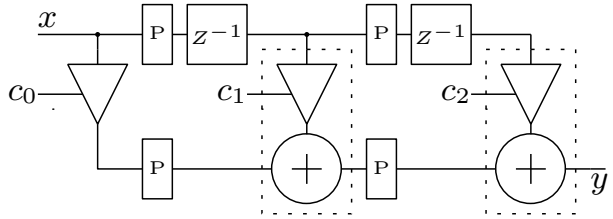


Figure 10: Second Order stage for a cascade form of FIR Low Pass Filter.

main advantage of this type factorization is that it has smaller sensitivity towards the word length.

5 Performance Estimation

5.1 Implementation Issues

In the implementation of FIR filters, three major blocks are used, viz., multiplier, delay elements and adders. The major block is the multiplier which multiplies the input signal by the known constants. Thus constant multipliers can be used in place of complete multipliers. The constant multipliers multiply a constant using add and shift method. This customization of multipliers can be useful in case ASIC implementation but in case of FPGA implementation the DSP blocks blocks are designed optimize according to the multiplicands.

An implementation of the FIR filter can be serial or parallel. In this work, all the possible parallel architectures are mentioned. But in some applications the serial implementations may be use full. The frequency at which an implementation can work is useful. This why the pipeline registers are inserted to support the higher frequencies. But if the application frequency is low then this registers may not be used.

The FIR filters suffer from the quantization error. The quantization error depends on firstly on the data format. There are two formats, floating point and fixed point. It is obvious that floating point format gives better accuracy but uses more hardware. The fixed point format is preferred here and the word length controls the quantization error. The word length should be chosen in such a way that minimum resources are used with acceptable accuracy.

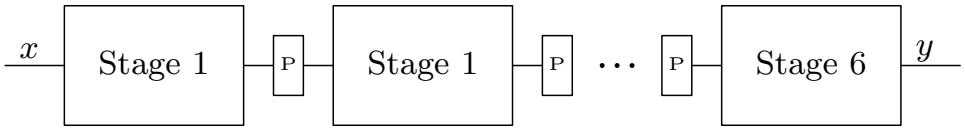
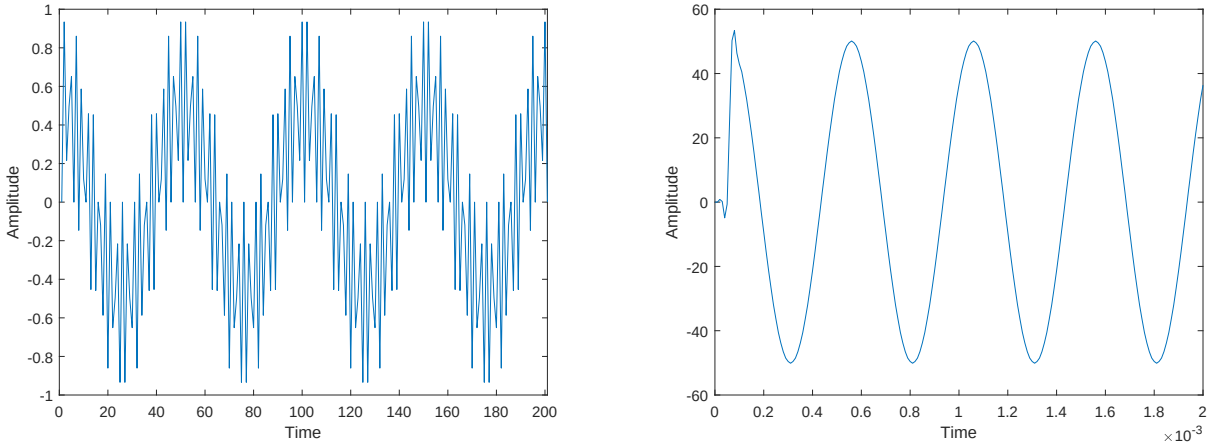


Figure 11: Cascade form of FIR Low Pass Filter.



(a) Input signal to the FIR filter

(b) Output of the low pass filter

Figure 12: Input and output signal of the filter

5.2 Design Performance

In this work, 13-tap low pass FIR filter is implemented on NEXYS DDR2 artix7 FPGA device (xc7a100t-3csg324). The low pass filter is verified by taking two sinusoidal signals of frequencies $22KHz$ and $20KHz$. These two signals are multiplied and output of the multiplier is given to the low pass filter. The sampling frequency is taken as $100KHz$ and thus the low pass filter filters out the signals whose frequency greater than $25KHz$. The output of the filter is a tone of $2KHz$ which is shown in Figure 12(b). The original output signal obtained from MATLAB and the FPGA based filtered output is compared in Figure 12. Here, 20-bit fixed point data width is chosen for implementation where 12-bit is reserved for fractional part. Here, Root Mean Squared Error (RMSE) is used to measure the design performance. RMSE is computed as

$$RMSE = \frac{\|(\hat{y} - y)\|_2}{\|y\|_2} \tag{9}$$

Here y is MATLAB based filtered output and $\hat{y}$ is FPGA output. A RMSE of 0.0006728 is achieved using 20-bits of word length.

5.3 Comparison of the Different Structures

Comparison of the implementation of the different structures is shown in Table 1. Performance of the structures is evaluated in terms of consumption of slice registers, LUTs, DSP blocks and occupied slices. The minimum clock period represents the maximum frequency that can be achieved. The dynamic power is computed at the maximum frequency. It is clear that transposed direct form architecture is better than the other direct form structures as it do not consume pipeline registers. The linear phase structures can only be used when the co-efficients are symmetrical. Linear phase 2 architecture achieves less frequency as it has a long critical path. But Linear phase 3 architecture implements same linear phase 2 structure but with DSP blocks. Here, higher maximum frequency is achieved and also other resources are are less used

Table 1: Performance Comparison of Different Structures.

Structures	CLK_{min}	Slice Reg	Slice LUT	DSP48	Occupied Slices	Power
Direct Form 1	5.5 ns	432	327	11	155	0.048 W
Direct Form 2	5.7 ns	631	337	11	283	0.048 W
Direct Form 3	5.5 ns	259	273	11	83	0.044 W
Linear Phase 1	5.5 ns	533	347	6	125	0.044 W
Linear Phase 2	7.5 ns	390	302	6	159	0.048 W
Linear Phase 3	6 ns	352	168	7	70	0.064 W
Cascaded	5.5 ns	290	279	16	105	0.036 W
Polyphase 1	6 ns	542	309	11	261	0.051 W
Polyphase 2	5.5 ns	522	334	11	212	0.053 W

with higher power consumption. This shows the advantage of using the inbuilt DSP blocks. Cascaded or Polyphase structures are less used to implement FIR filters but they have other advantages which is beyond the scope of this work.

6 Conclusion

In this work, FPGA implementation of the different FIR filter structure is presented and a comparison of the performances is presented here. Here, a low pass filter is designed using different structures to demonstrate difference in implementation. Direct form structures directly implements the FIR transfer functions and these are systolic architectures. Transposed structures are very popular as they don't need the pipeline registers. Linear phase structures are only possible when the transfer function is symmetric or anti-symmetric. Cascaded structures are useful to ignore the effect of word length. The DSP blocks can efficiently increase the performance of the filter structures.

References

[1] XILINX, "7 series dsp48e1 slice," vol. UG479 (v1.10), march 2018. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf

[2] P. R. Babu, *Digital Signa Processing.* SCITECH, 2009.