Linear Phase FIR Filter Implemented on FPGA

AN ECE 641 PROJECT

Presented by Xiao Zhang

Contents

INTRODUCTION

OBJECTIVE

IMPLEMENTATION

RESULT AND DISCUSSION

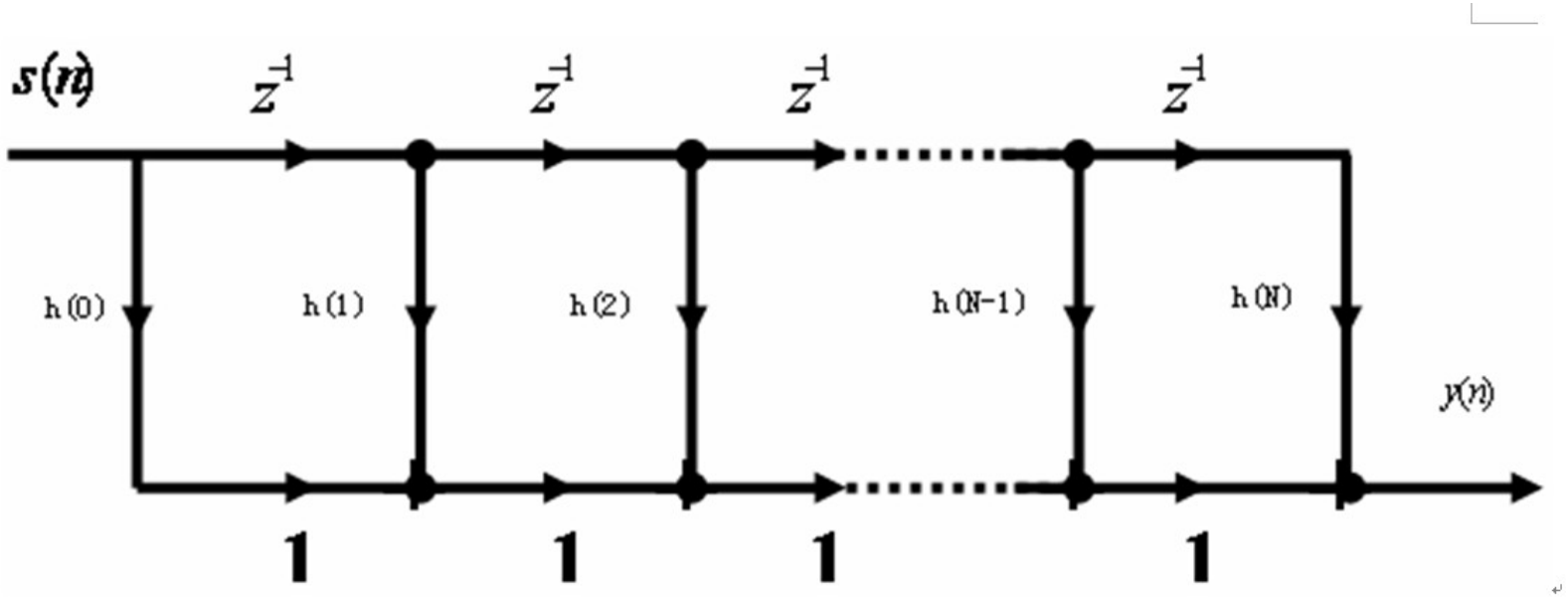
Introduction

- Transfer function of the filter is $H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 + \sum_{k=1}^{M} a_k z^{-k}}$
- Finite Impulse Response (FIR) Filters: M = 0, no feedback
- The function of N-order FIR Filter is $H(z) = \sum_{k=0}^{N} h(k) z^{-k}$
- The finite impulse response filter input-output relationship as follows

$$y[n] = x[n] * h[n] = \sum_{k=0}^{N} x[k]h[n-k]$$

Introduction

The structure of N-order FIR Filter



Introduction

- Linear-phase FIR filters

1. The ability to have an exactly linear phase response is the one of the most important of FIR filters

$$H(w) = |H(w)|e^{j\phi(w)} \quad \text{where} \quad \phi(w) = -wn_0$$

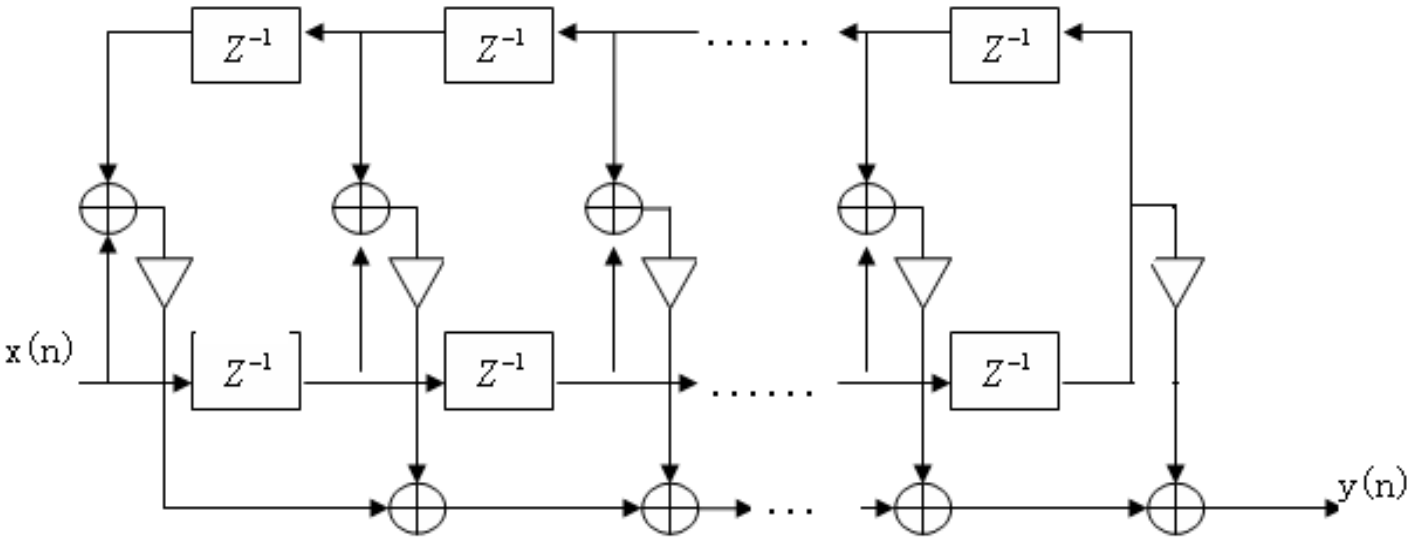
2. A general FIR filter does not have a linear phase response but this property is satisfied when h(n)=h (N-1-n)

Introduction

- Symmetric impulse response linear phase FIR transfer function can be written as follows:

When N is odd:

$$H(z)=\sum_{n=0}^{N} h[n]z^{-n}=\sum_{n=0}^{\frac{N}{2}-1} h[n]\left(z^{-n}+z^{-(N-n)}\right)+h\left(\frac{N}{2}\right)z^{-\frac{N}{2}}$$

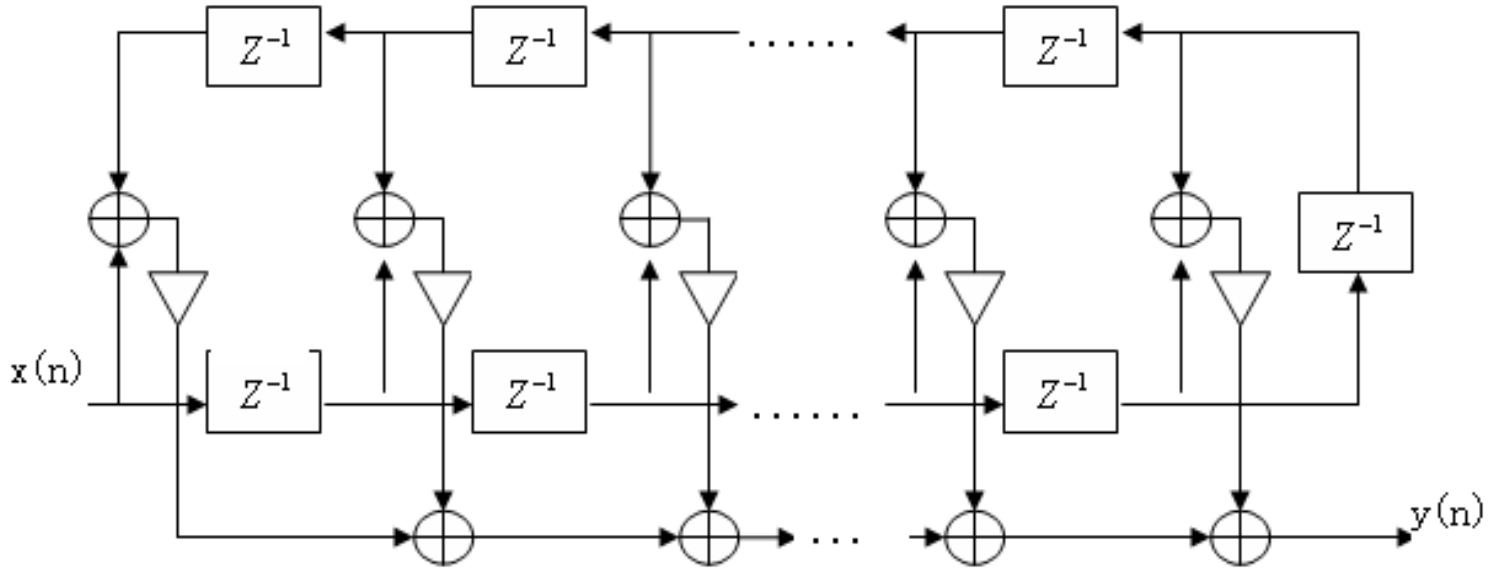


Introduction

- Symmetric impulse response linear phase FIR transfer function can be written as follows:

When N is even:

$$H(z) = \sum_{n=0}^{N} h[n]z^{-n} = \sum_{n=0}^{\frac{N+1}{2}-1} h[n](z^{-n} + z^{-(N-n)})$$

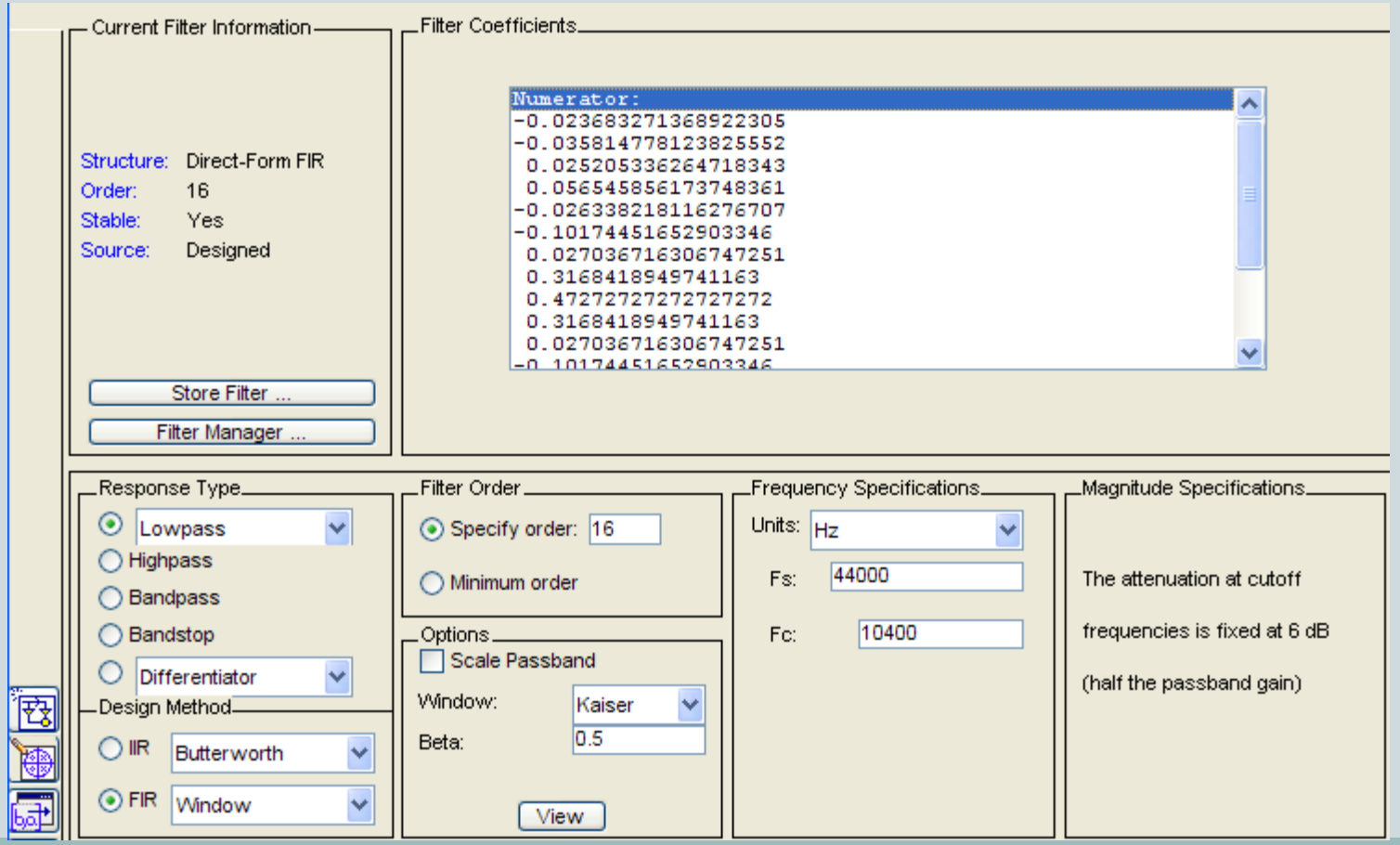


Objective

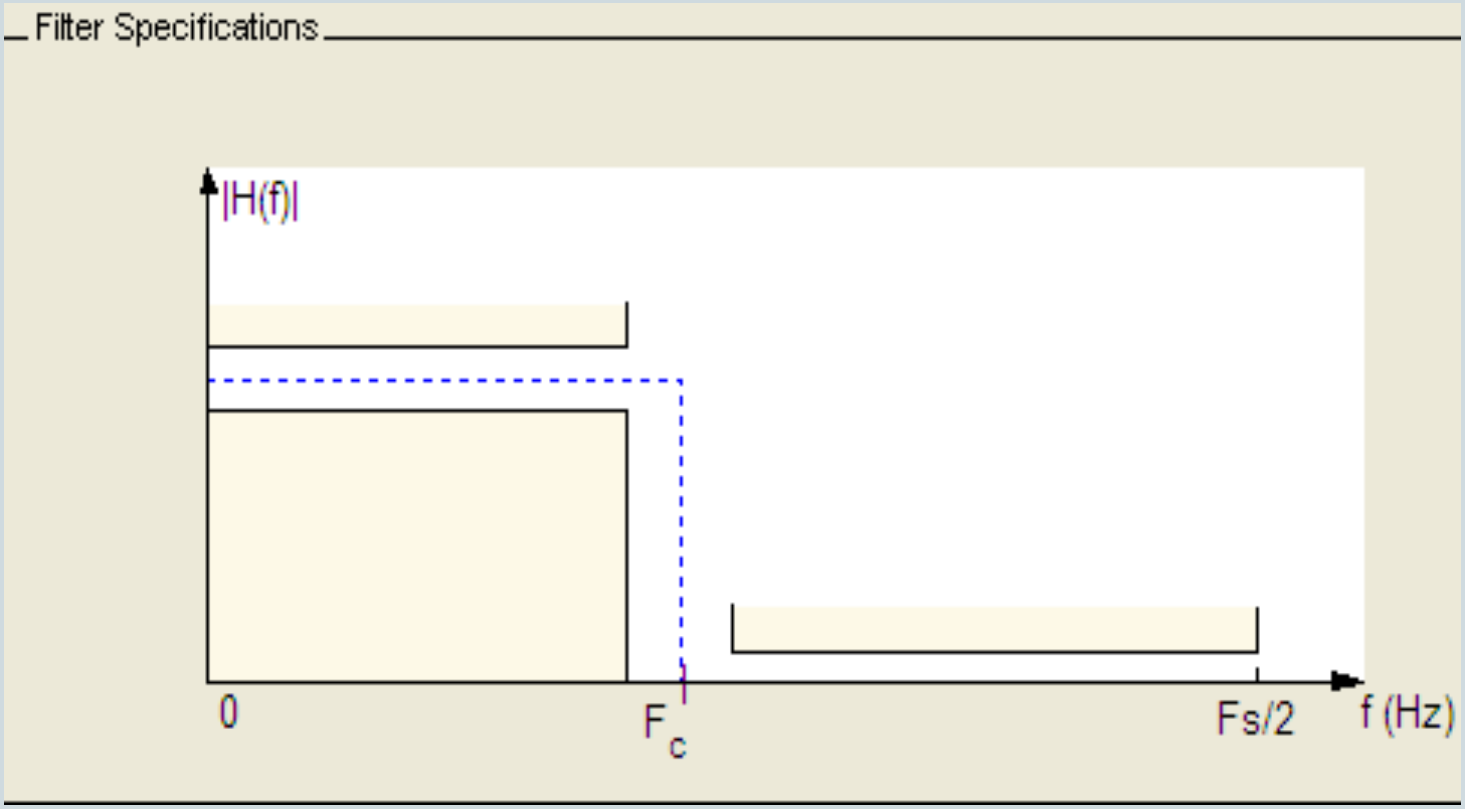
- The input signal is [99，0，0，0，70，0，0，0，99，0，0，0，70，0，0，0…]
- This project is to use FPGA to design and implement a 17-Order Linear Phase FIR Filter which has 8-bit input and 8-bit output, Fs=44kHz, Fc=10.4kHz.

Implementation

- First we use Matlab Simulink to design and calculate the filter coefficients

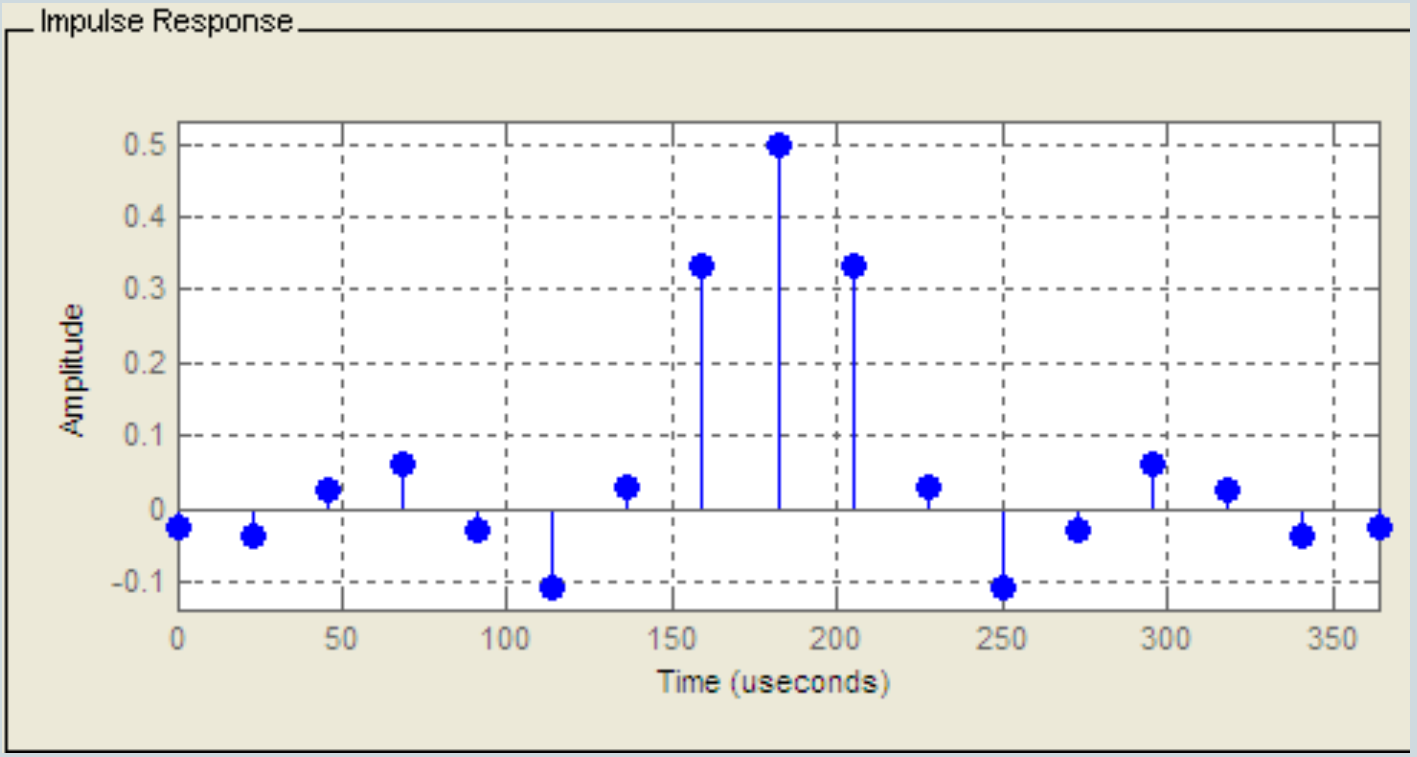


Implementation



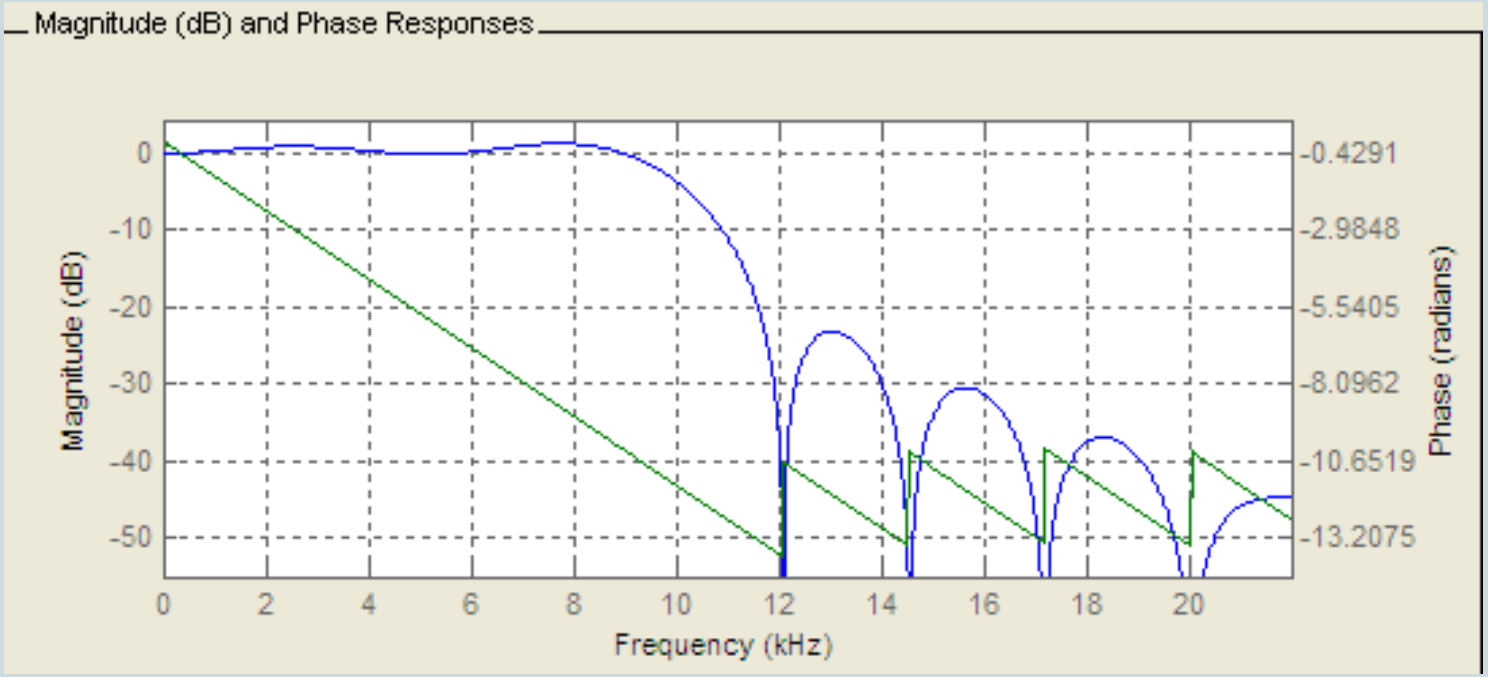
Implementation

- Impulse Response



Implementation

- Magnitude and Phase Response



Implementation

- Make the FIR coefficients to be integers

```
  Columns 1 through 7

   -0.0469   -0.0703    0.0508    0.1133   -0.0508   -0.2031    0.0547

  Columns 8 through 14

    0.6328    0.9453    0.6328    0.0547   -0.2031   -0.0508    0.1133

  Columns 15 through 17

    0.0508   -0.0703   -0.0469

>> Num*(2^8)

ans =

  Columns 1 through 13

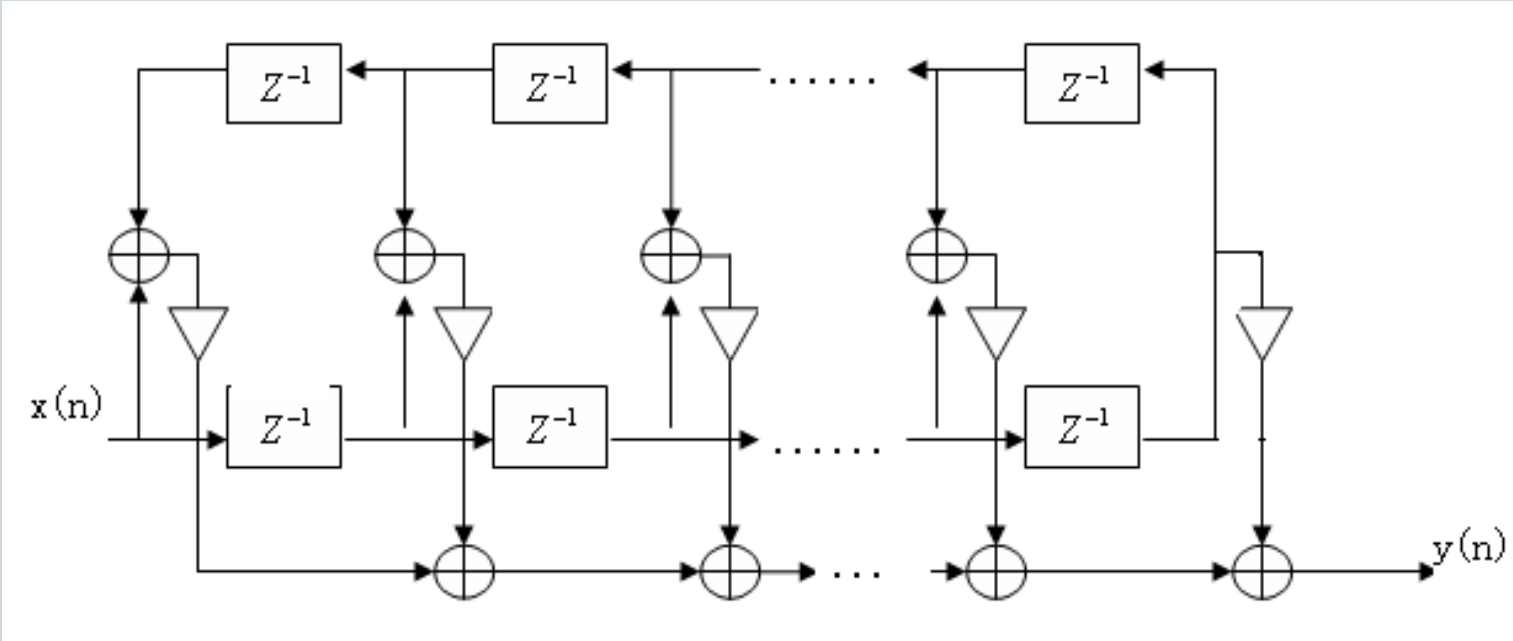
   -12   -18    13    29   -13   -52    14   162   242   162    14   -52   -13

  Columns 14 through 17

    29    13   -18   -12
```

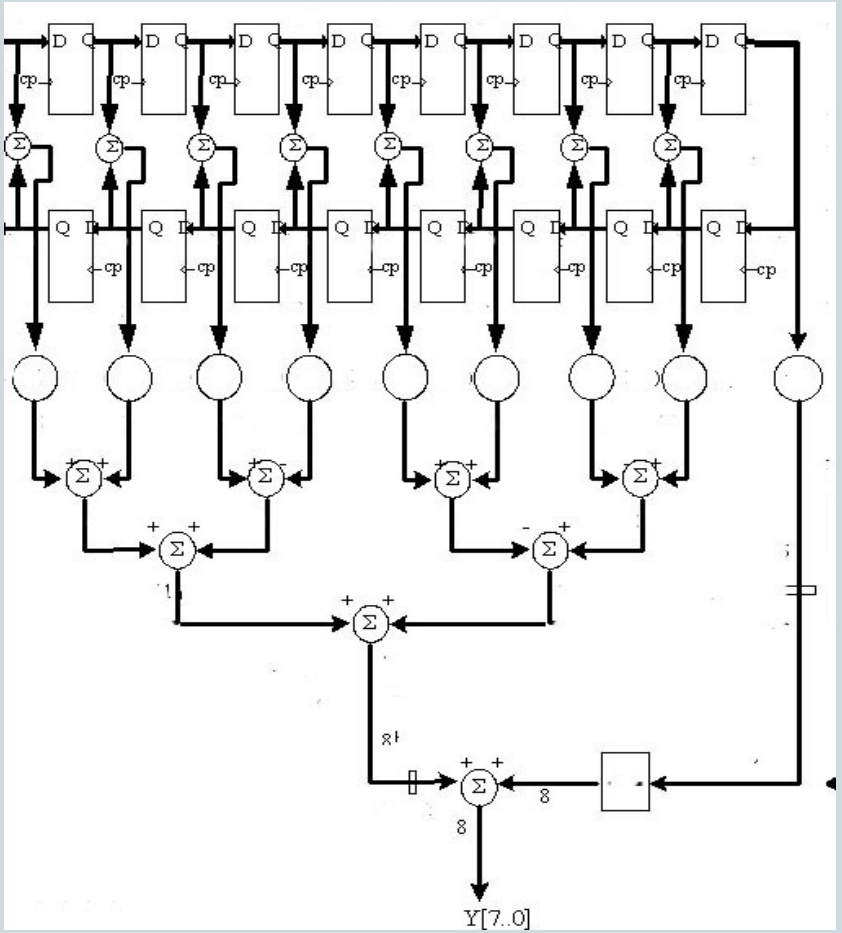
Implementation

- We get FIR filter parameters are [-12 -18 13 29 -13 -52 14 162 242 14 -52 -13 29 13 -18 -12]
- 17 is odd, so we use the diagram (see below) to design.



Implementation

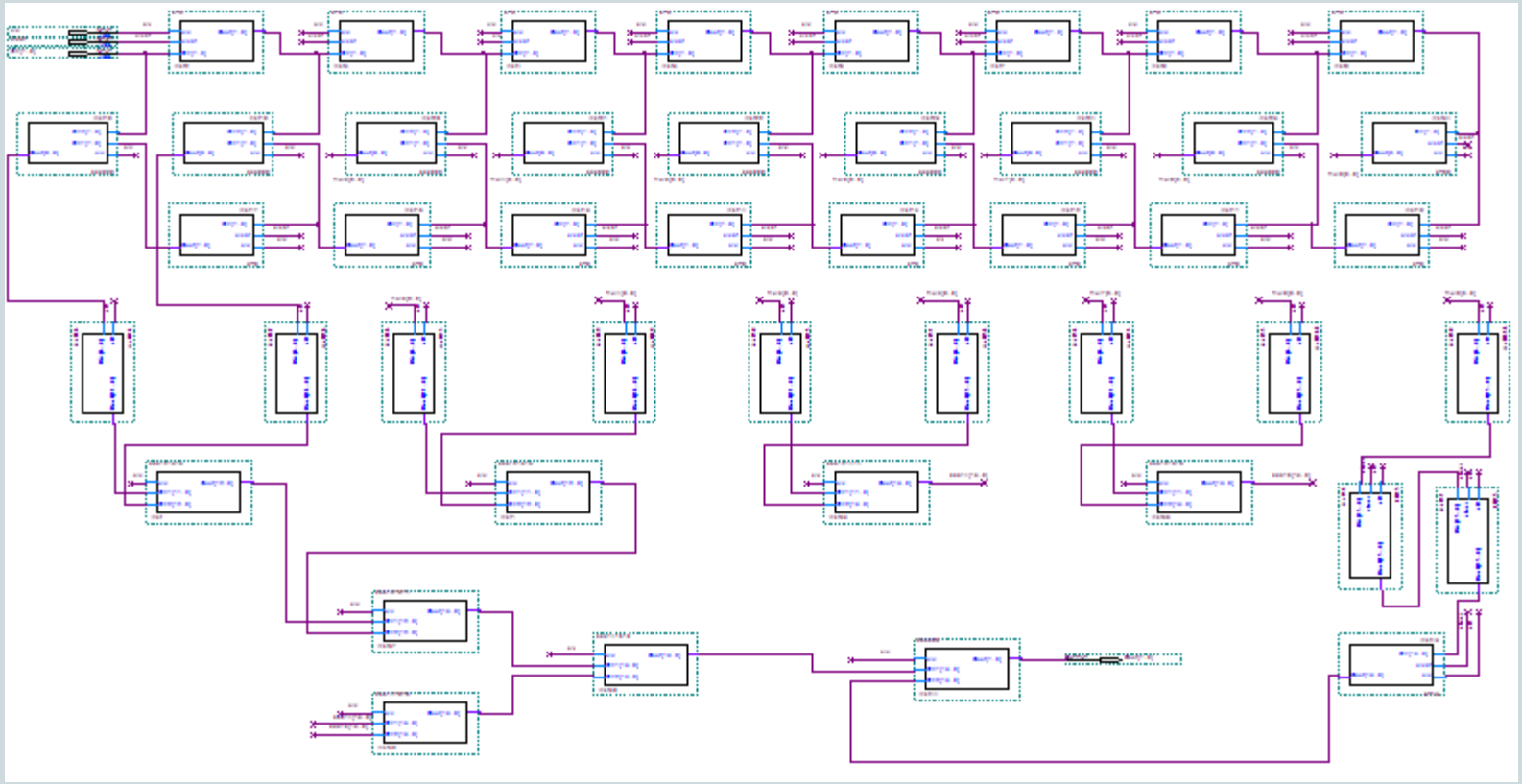
- Use 4 kinds of modules (Registers, Adders, Subtractors, Multipliers) to design the FIR filter.
- According to functions of the various modules, I use VHDL to make them work well.



Schematic diagram of the FIR filter

Implementation

- The overall circuit diagram



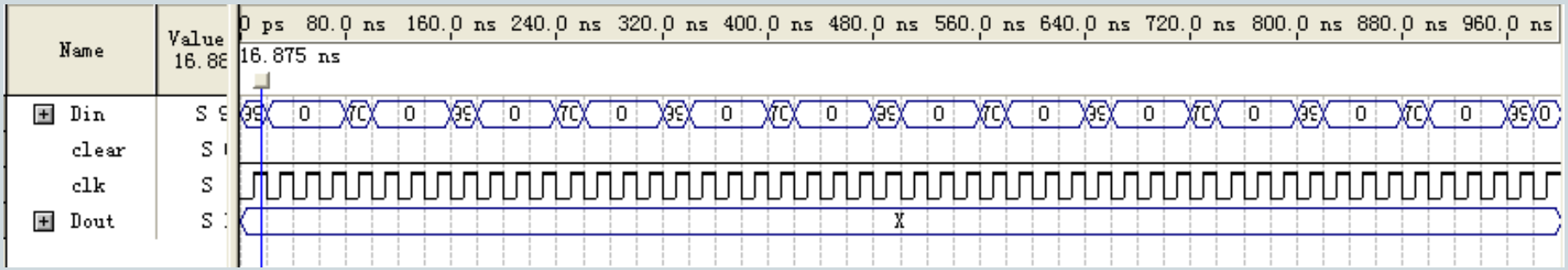
Result and Discussion

Compilation Report –Flow Summary

Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	fir
Top-level Entity Name	fir
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	697 / 68,416 (1 %)
Total combinational functions	527 / 68,416 (< 1 %)
Dedicated logic registers	435 / 68,416 (< 1 %)
Total registers	435
Total pins	18 / 622 (3 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)

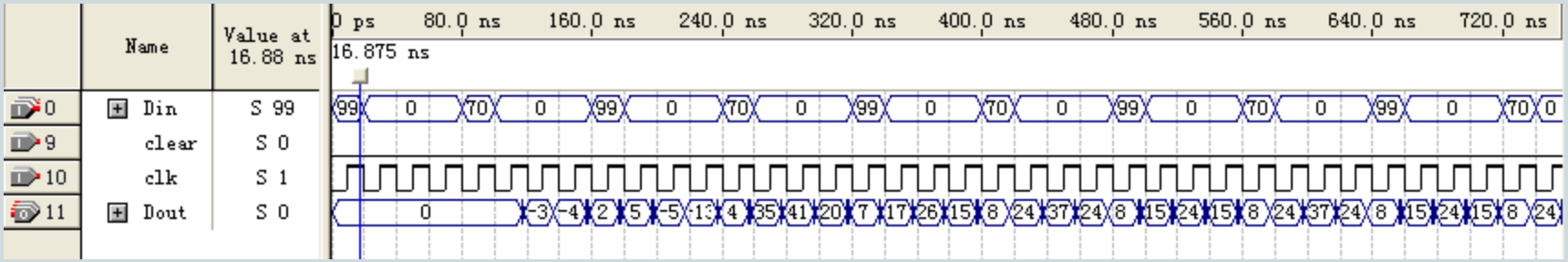
Result and Discussion

- We use the input signal X=[99, 0, 0, 0, 70, 0, 0, 0, 99, 0, 0, 0, 70, 0, 0, 0, 99, 0, 0, 0, 70, 0, 0, 0, 99, 0, 0, 0, 70, 0, 0, 0]



Result and Discussion

- The simulation report of the FIR filter



- Use Matlab to calculate the convolution.

H=[-12,-18,13,29,-13,-52,14,162,242,162,14,-52,-13,29,13,-18,-12];
X=[99,0,0,0,70,0,0,0,99,0,0,0,70,0,0,0,99,0,0,0,70,0,0,0,99,0,0,0,70,0,0,0];
Y=conv(X,H)/512

Then we will get the theoretical value.

Result and Discussion

	theoretical value	Simulation result
y[0]	38.598	37
y[1]	24.699	24
y[2]	8.9121	8
y[3]	15.24	15
y[4]	24.777	24
y[5]	15.24	15
y[6]	8.9121	8
y[7]	24.699	24
y[8]	38.598	37
y[9]	24.699	24
y[10]	8.9121	8
y[11]	15.24	15
y[12]	24.777	24
y[13]	15.24	15
y[14]	8.9121	8
y[15]	24.699	24
y[16]	38.598	37
y[17]	24.699	24

- By comparison, We see that the simulation results and the theoretical value of the output signal are nearly the same. So we get the expected outcome!

DEMO

REFERENCES

- [1] http://en.wikipedia.org/wiki/Finite_impulse_response
- [2] Digital Signal Processing, 3/e Sanjit K Mitra, University of California---Santa Barbara
- [3] Understanding Digital Signal Processing, Second Edition By: Richard G. Lyons
- [4] Signal processing with Field Programmable Gate Arrays Third Edition By: U. Meyer-Baese

THANK YOU!