Polyphase Decimation Filters

SystemC Approach

Author:
Ahmed Shahein

email:
ahmed.shahein@ieee.org

May 11, 2012

1 Overview

This is a behavioral SystemC model for Polyphase Decimation filters. It can be used as for system design and functional verification. It has been tested with Matlab and Octave as well. If you need any further illustrations or further modifications, don't hesitate to contact me.

2 Introduction

Decimation filter is a filter and downsampler. Polyphase decimation filter is a digital filter (FIR/IIR) which is implemented in a polyphase decomposition.

The FIR filter transfer function is given by

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

where N is the filter order and h[n] is the filter coefficients (in other words the filter step response). A direct-form (DF) FIR filter is shown in Fig. 1.

The polyphase decomposition for FIR filter is given by

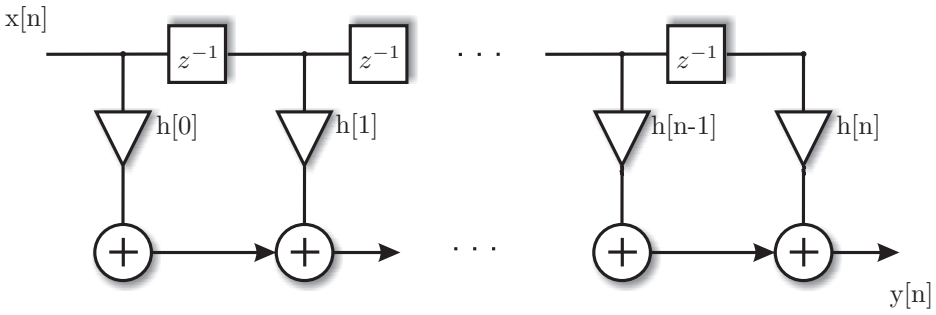

Figure 1: Direct-form FIR digital filter topology.

$$H(z) = \sum_{k=0}^{M-1} E_k(z^M)\ z^{-k} \tag{2}$$

where

$$E_k(z) = \sum_{n=0}^{\lceil \frac{N}{M} \rceil} h[nM + k]\ z^{-n} \tag{3}$$

where N is the filter order, M is the decimation factor (also called downsample factor) and $k = \lceil \frac{N+1}{M} \rceil$.

A polyphase decimation filter based on DF FIR is shown in Fig. 2 and detailed structural implementation is given in Fig. 3. The main advantage of PPD is that the filter is running at lower frequency compared to conventional

filter architecture. In other words, for clock frequency of f_s and decimation factor M the standard FIR filter is running at f_s while PPD is running at f_s/M.

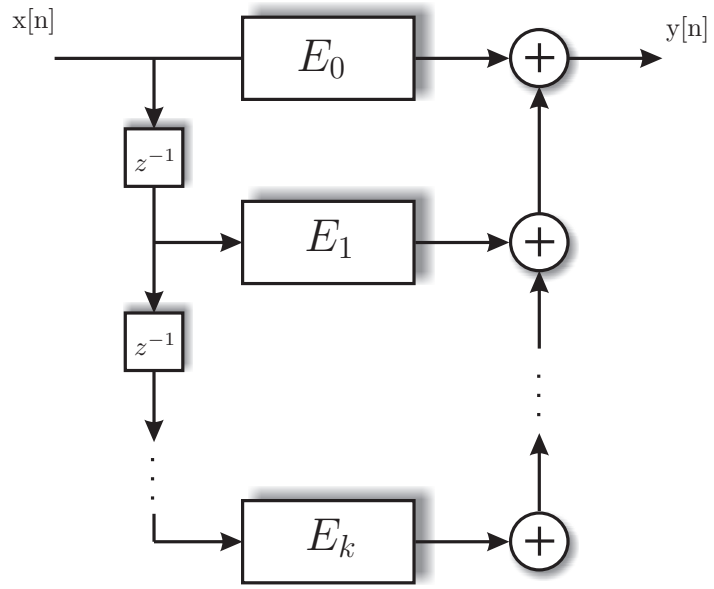

Figure 2: Polyphase decimation filter.

3 Symbolic Example

Consider a filter with order of 8 ($N = 8$) and decimation factor of 3 ($M = 3$)

$$H(z) = h[0]+h[1]z^{-1}+h[2]z^{-2}+h[3]z^{-3}+h[4]z^{-4}+h[5]z^{-5}+h[6]z^{-6}+h[7]z^{-7}+h[8]z^{-8}$$

$$E_k(z) = \sum_{n=0}^{\lceil \frac{8}{3} \rceil} h[3n+k]\ z^{-n}$$

$$\begin{aligned} E_0(z) &= \sum_{n=0}^{2} h[3n+0]\ z^{-n} \\ E_1(z) &= \sum_{n=0}^{2} h[3n+1]\ z^{-n} \\ E_2(z) &= \sum_{n=0}^{2} h[3n+2]\ z^{-n} \end{aligned}$$

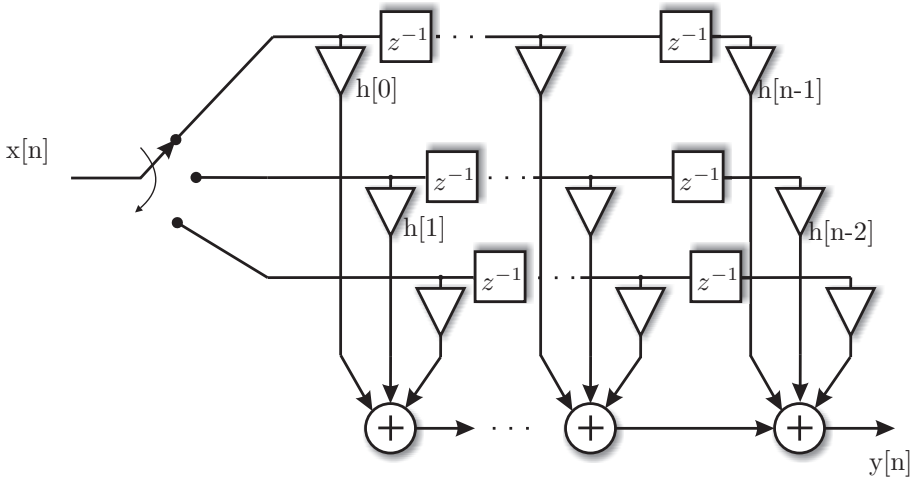

Figure 3: Direct-form FIR digital filter topology.

$$\begin{aligned} E_0(z) &= h[0] + h[3]z^{-3} + h[6]z^{-6} \\ E_1(z) &= h[1] + h[4]z^{-3} + h[7]z^{-6} \\ E_2(z) &= h[2] + h[5]z^{-3} + h[8]z^{-6} \end{aligned}$$

4 How to Build your PPD?

@ ppd.cpp

Replace h[N] = 0.0017,0.0073,0.0107 ... with your filter coefficients, where the N is the filter order and it is defined at **ppd.h**. Setting N is a bit trick and it is explained later on.
You can then change the clock frequency by changing 10, where 10 is the clock period.

```
sc_clock CLK("CLK", 10, SC_NS);
```

Then you can change the simulation period by changing 360

```
sc_start(360, SC_NS);
```

@ ppd.h

Replace N, M and P first set M to your decimation factor (has to be integer) adjust and set N to be a valid integer value taking into cosidration M. As an example, if you have a filter order of $N = 55$ and a decimation factor of $M = 6$. Then how many columns do we have in the PPD? In other words, how many coefficients at each row?

$$\lceil \frac{55+1}{6} \rceil = 10$$

However

$$10 \times 6 = 60$$

Which means there is 4 empty coefficients, those extra coefficients has to be filled by zeros. Actually, I could write a simple function to adjust this but I leave it to you. In this case the input parameters would be

```
#define N 60
#define M 6
#define P 10
```

Actually, you can add a simple check like that

```
if (M*P != N)
{
cout << "#################" << endl;
cout << "ERROR : MxP != N" << endl;
cout << "#################" << endl;
}
```

@ stimuli.h

The the proceeding value to the length of samples in the stimuli file. In the default case it is set to the number of samples at the impulse_response.dat.

```
#define textLength 40
```

In the following line just replace "impulse_response.dat" to the name and extension of your stimuli file.

```
FILE* pFile = fopen("impulse_response.dat","r+t");
```