Digital IIR Filter

SystemC Approach

Author:
Ahmed Shahein

email:
ahmed.shahein@ieee.org

July 20, 2012

1 IIR Fundamentals

Infinite Impulse Response (IIR) filters is a typical DSP component. IIR filters can be be implemented as analog or digital filter. However it depends on their analog counterparts. Compared to FIR filters, IIR filters have less order, narrow transitions, and approximate to analog filter responses. On the other hand, very sensitive to fixed point representation, can't be used for multi-rate computation (decimation filter), and non-linear phase (variable group delay). Its transfer function is given by

$$H(z) = \frac{Y(z)}{X(z)} \quad (1)$$

$$H(z) = \frac{\Sigma_{i=0}^{N} b[i]z^{-1}}{1 - \Sigma_{i=1}^{N} a[i]z^{-1}} \quad (2)$$

Its difference equation is depicted by

$$y[n] = b[0]x[n] + b[1]x[n-1] + b[2]x[n-2] \cdots + a[1]y[n-1] + a[2]y[n-2] \cdots \quad (3)$$

2 IIR Structures

The IIR structure does not affect the functionality of the filter, i.e., the transfer function does not change. The choice of the implementation depends on the hardware realization theme. The common structures are:

1. Direct-form I (DF I)
2. Direct-form II (DF II)
3. Transposed-form I (TF I)
4. Transposed-form I (TF II)
5. Second Order Sections (SOS)

There are several trade-offs between the different implementations. As an example, the DF/TF-I and DF/TF-II are recommended for high speed applications. However, they are very sensitive to round-off errors. On the other hand, the cascaded implementation, e.g., second order sections (SOS), are less sensitive to round-off errors. Practically, it is recommended to use SOS implementation for IIR filters. Each section could be implemented as either of the previous basic structures.

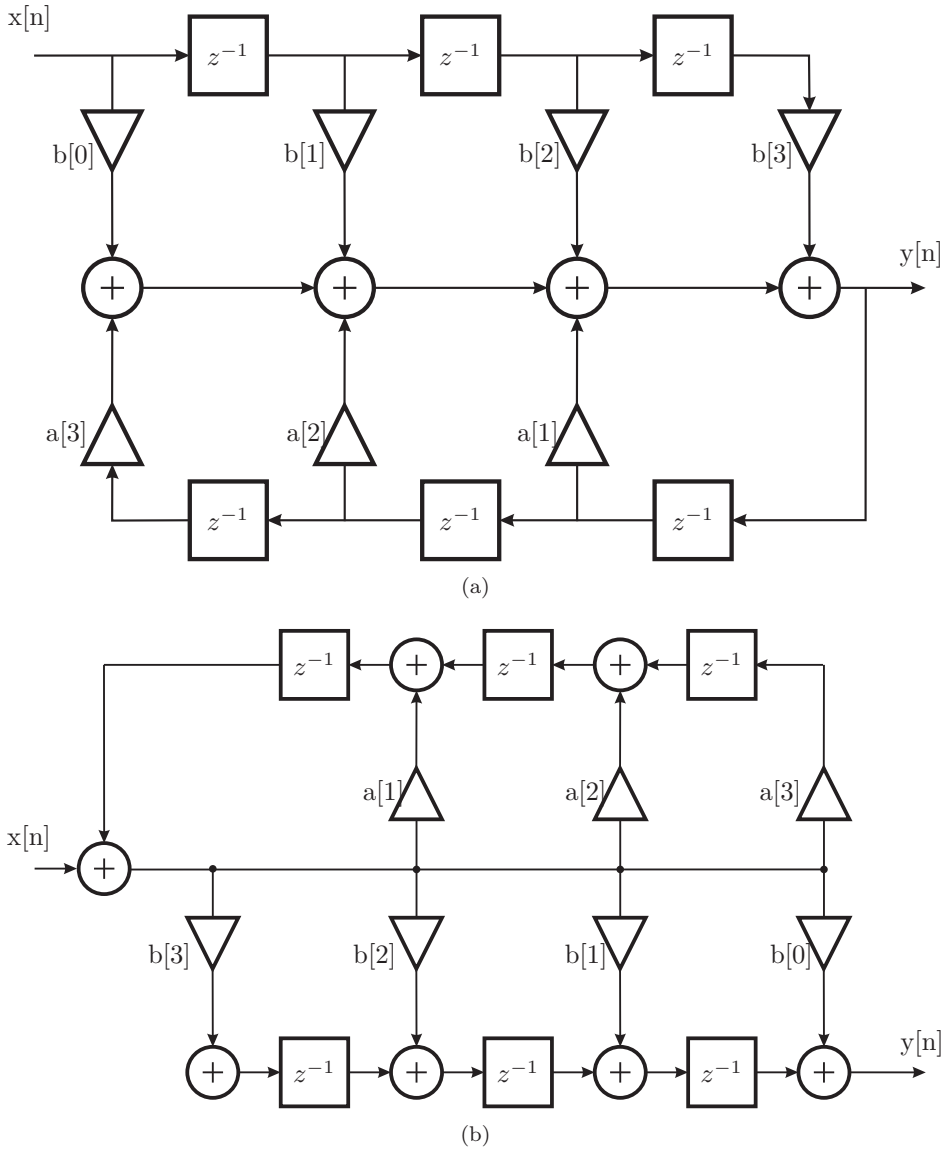
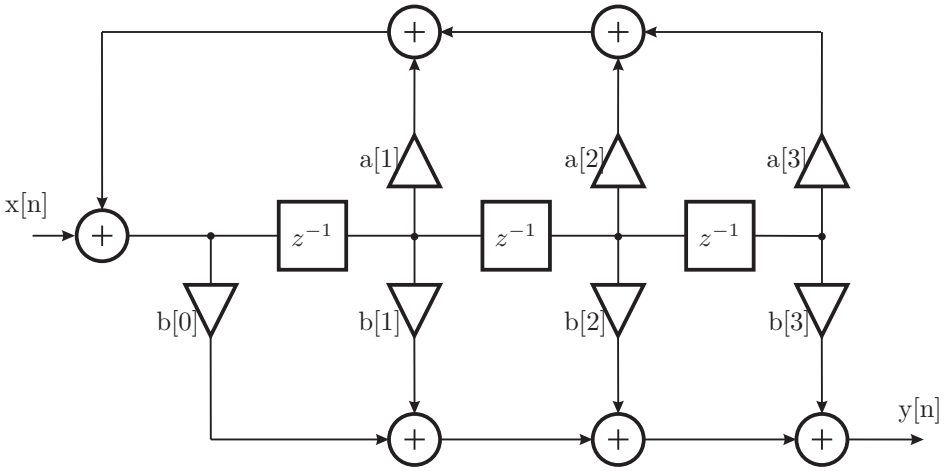
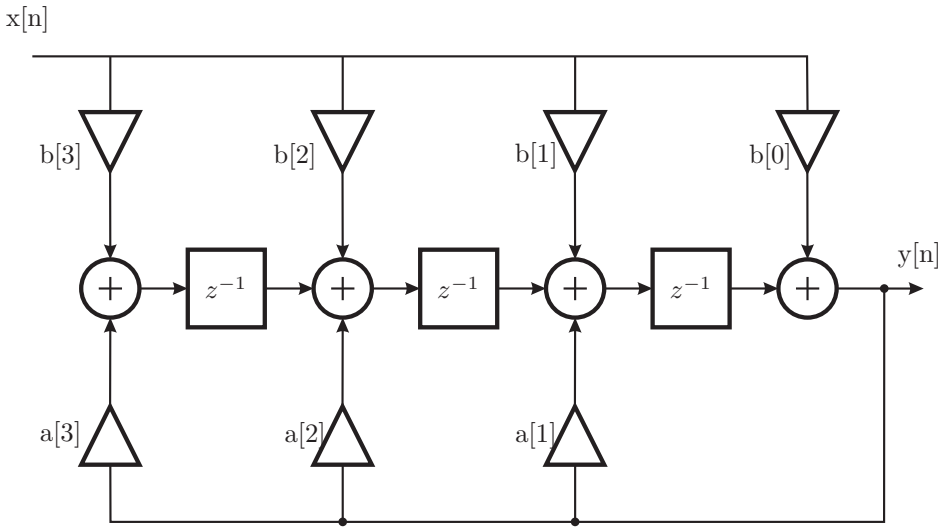


Figure 1: IIR filter structures (a) Direct-form I (b) Transposed-form I.



(a)



(b)

Figure 2: (a) Direct-form II (b) Transposed-form II.

3 Matlab

```
f = [0 0.25 0.35 1.0];      % Corner frequencies
m = [1 1 0 0];              % Filter magnitudes
n = 3;                      % Filter order

[b,a] = yulewalk(n, f, m);  % Design IIR Filter

[h,w] = freqz(b,a,128);     % FFT

figure
plot(f,m,w/pi,abs(h))       % Plot frequency response

figure
step(filt(b,a))             % Plot step response

figure
impulse(filt(b,a))          % Plot impulse response

figure
zplane(b,a)                 % Plot poles/zeros

[sos, g] = tf2sos(b, a);    % Second Order Sections
```

4 Implementation

Here I will describe how the developed model is constructed. I will consider only the TF I as an illustrative case study since. I developed it in a structural manner in order to easily transfer it to hardware model using any HDL. The filter is constructed from right-to-left as indicated by the bold arrow in the figure shown below.

- **x[n]** input
- **y[n]** output
- **b[i]** feed-forward coefficients
- **a[i]** feed-back coefficients

The internal signals are defined as arrays, i.e., each signal is an element in the array. The internal signals are defined as follow:

- **oMultiplierFF[i]** multipliers' outputs in the feed-forward path
- **oMultiplierFB[i]** multipliers' outputs in the feed-back path
- **oAdderFF[i]** adders' outputs in the feed-forward path
- **oAdderFB[i]** adders' outputs in the feed-back path
- **oDelayFF[i]** delays' outputs in the feed-forward path

- **oDelayFB[i]** delays' outputs in the feed-back path

The figure shows a detailed construction of an IIR filter of 3^{rd} order.

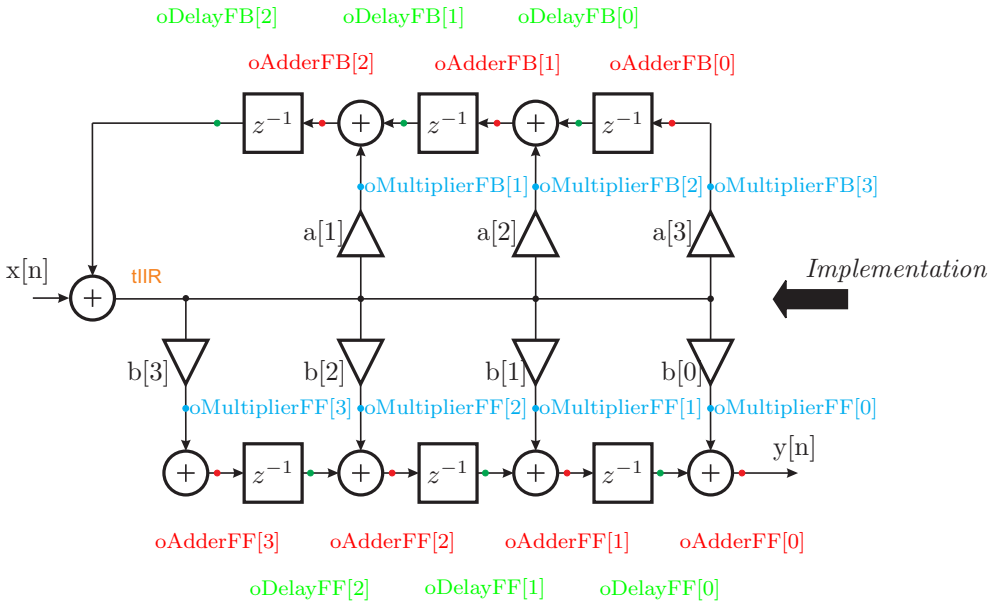


Figure 3: Transposed-form I

5 Getting Start

Generally you will just need to insert the filter coefficients, i.e., feed-forward (b's) and feed-back (a's) coefficients. Further, the filter order and the stimuli file. The code comes with stimuli files for step and impulse responses. At the main file '.cpp' the compilation command is commented by the end of the file. The model is developed as a template so that you can change the data type as the design or user wish. As an example, you can simply investigate the effect of fixed point representation by replacing the float data type by sfixed¡¿ data type.

At the header filter **IIR_TFI.h** you can change:

- Define the filter order

```
#define order 3
```

At the header filter **Stimuli.h** you can change:

- Give in the name of the stimului file

```
FILE* pFile = fopen("Step.txt","r+t");
```

At the main file **IIR_TFI.cpp** you can change the following items:

- Adjust the clock frequency

```
sc_clock  CLOCK("CLOCK", 1, SC_US);
```

- Enter the filter coefficients

```
float b[orderFF] = {0.0995,0.1486,0.1481,0.0999};
float a[orderFB] = {0.9828,-0.5450,0.0671};
const int Size   = 16;
```

- Change the data type from float to sc_fixed<16,3>

```
IIR_TFI<float > DUT("DUT", b, a);
```