Embedded DSP : Introduction to Digital Filters

1

Embedded DSP: Introduction to Digital Filters

- Digital filters are a important part of DSP. In fact their extraordinary performance is one of the keys that DSP has become so popular.
  - Audio processing
  - Speech processing (detection, compression, reconstruction)
  - Modems
  - Motor control algorithms
  - Video and image processing

2

Embedded DSP: Introduction to Digital Filters

- Historically, electronic designers implemented filters with analog components, as resistors, capacitors and inductors.
- With the develpment of special DSP processor (>1980), designers have a alternative: filter implementation by software on DSPs.
- Designers can now choose between the implementation on several technologies as
  - General purpose DSP
  - Gate-Arrays
  - General purpose microprocessors

3

Embedded DSP: Introduction to Digital Filters

- Analog filters
  - Electronic components are cheap.
  - Large dynamic range in amplitude and frequency.
  - Real-time.
  - Low stability of resistors, capacitors and inductors due to temperature.
  - Difficult to get the components accuracy as calculated by the formula.

Embedded DSP: Introduction to Digital Filters

- Digital filters:
  - Better performance than analog filters
    - Sharp Cut-off in the transition band.
  - DSP filters are programmable. The transfer function of the filter can be changed by exchanging coefficients in the memory. One hardware design can implement many different, loadable filters by executing a software development process.
  - The charachteristics of DSP filters are predictible.
  - Filter design software packages can accurately evaluate the performance of a filter by simulation before it is implemented in hardware.
  - Alternative digital designs are available by tools to adapt the filter to the application.
  - Unlike analog filters, the performance of digital filters is not dependent on the environment, such as temperature or voltage
  - In general, complex digital filters can be implemented at lower cost than complex analog filters.

5

Embedded DSP: Introduction to Digital Filters

Digital filters are used for two general tasks:

- Separation of different frequency components in signals if contaminated by
  - noisy
  - interference
  - other signal
- Restoration of signals which have been distorted in some ways
  - Improvement and correction of an audio signal recording which is distorted by poor equipment
  - Deblurring of an image from improperly focused lens

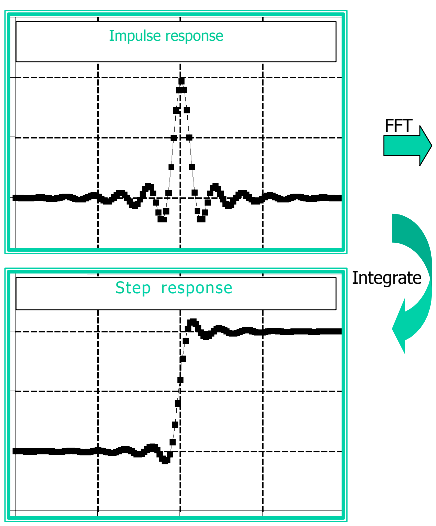
6

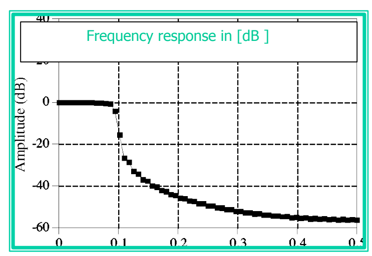
Embedded DSP: Introduction to Digital Filters

- Every linear filter has an
  - Impulse response
  - Step response
  - Frequency response
- Each of these responses contain the same information about the filter, but in different form.
- All representations are important because they describe how the filter will react under various circumstances.

7

Embedded DSP: Introduction to Digital Filters





The step response can be evaluated by discrete integration of the impulse response. The frequency response can be found from the impulse response by using the FFT (Fast Fourier Transformation).

8

Embedded DSP: Introduction to Digital Filters

Implementation of a digital filter

By convolution:

- Convolving the input signal with the digital filter impulse response.
- Each sample in the output is calculated by weighting the samples in the input and adding them together.
- All linear filters can be realized by convolution (by a filter kernel)
- FIR-Filter (Finite Impulse Response)

By recursion:

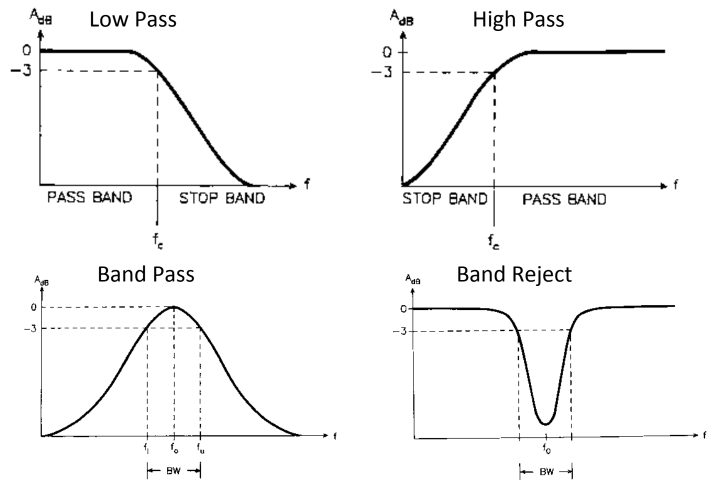
- Extension of the convolution by using previously calculated values from the output, besides the points from the input.
- Made of recursion coefficients.
- IIR-Filter (Infinite Impulse Response)

9

Filter Basics

- A filter is used to remove (or attenuate) unwanted frequencies in an audio signal
- "Stop Band" – the part of the frequency spectrum that is attenuated by a filter
- "Pass Band" – part of the frequency spectrum that is unaffected by a filter
- Filters are usually described in terms of their "frequency responses," e.g. *low pass, high pass, band pass, band reject (or notch)*

Frequency Response Curves



Essential Terminology

- *Cutoff Frequency* – point in the stop band where frequencies have been attenuated by 3 dB (½-power)
- *Center Frequency* – mid-point of the pass band in a Band Pass filter or the stop band of a Band Reject filter
- *Band Width* – distance (in Hertz) between the ½-power points of a Band Pass or Band Reject filter

Other Important Terms

- *Slope* – rate of attenuation within the stop band, measured in dB/Octave
- *Q* – the *Quality* of a filter. Definition:

$$Q = \frac{CF}{BW}$$

◆ *Q* is often a more useful parameter than *BW*, because the *BW* needs to vary with the CF to keep the same "musical interval"

◆ The higher the *Q*, the narrower the Band Width, and in *BP* filters, the more resonance may occur at the Center Frequency

A Simple Digital Filter

- All digital filters utilize one or more previous inputs and/or outputs
- A very simple digital filter:

$$y_t = .5x_t + .5x_{t-1}$$

◆ The current output is the average of the current input and the previous input

◆ A "moving average" filter, it has a low pass characteristic and a *Finite Impulse Response*

More Digital Filter Basics

- The *Impulse Response* of a filter is the output that will be produced from a single, instantaneous burst of energy, or "impulse"
- Given the input signal {1,0,0,0,0…}, the filter y(t)=.5x(t)+.5x(t-1) will output the signal {.5,.5,0,0,0…}, a "finite impulse response"
- A filter that uses only current and previous inputs produces a *Finite Impulse Response,* but a filter that employs previous outputs (a so-called "recursive filter") produces an *Infinite Impulse Response*
- If y(t) = .5x(t) + .5y(t-1), the impulse response is {.5,.25,.125,.0625,.03125…etc.}

Digital Filter Basics, cont.

- The *Order* of a filter is a measure of its complexity
- In a digital filter, the *Order* is proportional to the number of terms in its equation
- The *Slope* of the attenuation within the stop band of a filter is approximately –6 dB per *Order* of that filter
- Combining 2 filters by connecting them in series will double the total order, and hence, double the steepness of the slope

Digital Filter Basics, cont.

- Filters are often described in terms of *poles* and *zeros*
  - A *pole* is a peak produced in the output spectrum
  - A *zero* is a valley (not really zero)
- FIR (non-recursive) filters produce zeros, while IIR (recursive) filters produce poles.
- Filters combining both past inputs and past outputs can produce both poles and zeros

$$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) + \ldots + a_Mx(n-M)$$
$$-b_1y(n-1) - b_2y(n-2) - \ldots - b_Ny(n-N)$$

Embedded DSP: Moving Average Filters

- Moving average is the most common filter in DSP
  - Easy to understand
  - Easy to implement for DSP and FPGA
  - Less computation time
  - FIR filter
- the moving average filter operates by averaging a number of points from the input signal to produce each point in the output signal. In equation form, this is written

$$y[i] = 1/M \sum_{j=0}^{M-1} x[i+j] \quad \text{or symmetrical form: when } j=-(M-1)/2 \text{ to } (M-1)/2$$

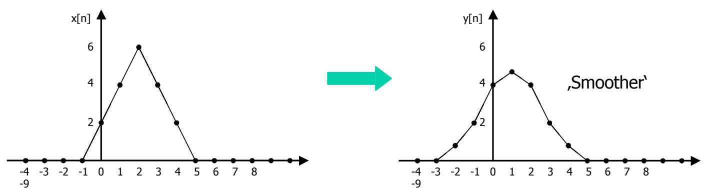
- Optimal filter for the following tasks:
  - Reducing random noise while retaining the sharp step response
  - Therefore usefull for time domain encoded signals, but
- Worst filter concerning frequency encoded signals (no frequency separation capabilities !)
- Relatives of the moving average filter include Gaussian and Blackman.

18

Embedded DSP: Moving Average Filters

- Example: A three point averager:
- y[n] = 1/3 [x(n) + x(n+1) + x(n+2)]

n	n < -2	-2	-1	0	1	2	3	4	5	n > 5
x[n]	0	0	0	2	4	6	4	2	0	n > 5
y[n]	0	2 / 3	2	4	14/3	2	3	5	0	n > 5



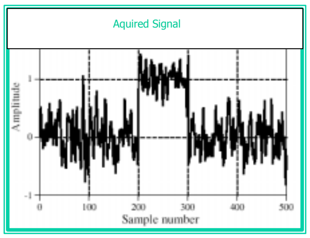
19

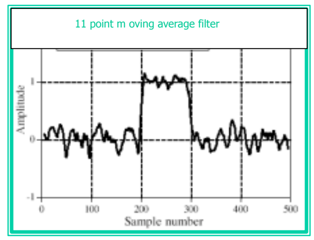
Embedded DSP: Moving Average Filters

- Examples:
- Smothing filter 5 points:
  - y[n] = 1/5 [x(n-2) + x(n-1) + x(n) + x(n+1) +x(n+2)]
    - Kernel is a rectangular pulse
- In contrast: Least square cubic:
  - y[n] = 1/35 [-3 x(n-2) + 12 x(n-1) + 17 x(n) + 12 x(n+1) - 3 x(n+2)]
    - Kernel is a more complex function !
- Symmetrical averaging requires that M be an odd number !
- An unsymmetrical filter produces an time offset of the output !
- Moving average is a convolution using a simple filter kernel: For 5-points the kernel is
  - ....0, 0, 1/5, 1/5, 1/5, 1/5, 1/5, 0, 0, ...   (a rectangular pulse)

20

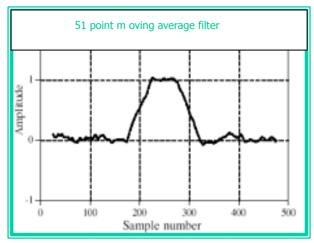
Embedded DSP: Moving Average Filters





Filtering by a 11 point moving average filter

Processing time ++



Filtering by a 51 point moving average filter

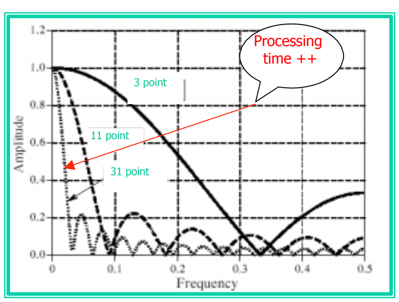
A rectangular pulse with noise

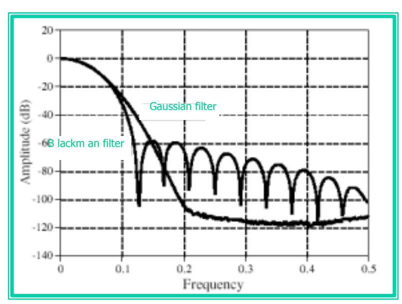
Increasing the number of points in the filter leads to a better noise performance. But the edges are then less sharp. This filter is the best solution providing the lowest possible noise level for a given sharpness of the edges. The possible amount of noise reduction is equal to the square-root of the number of points in the average (a 16 point filter reduces the noise by a factor of 4)

21

Embedded DSP: Moving Average Filters

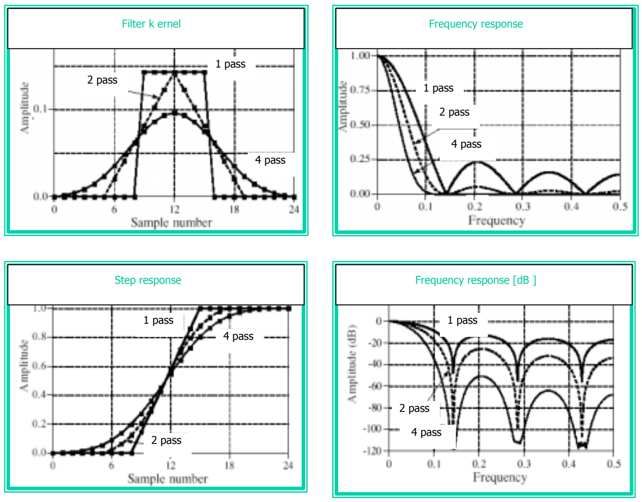
- The frequency response is mathematically described by the Fourier Transform of the rectangular pulse.
- H[f]=sin(Pi f M) / M sin(Pi f)
- The roll-off is very slow, the stopband attenuation is very weak !
- The moving average filter is a good smoothing filter but a bad low-pass-filter !





22

Embedded DSP: Moving Average Filters



- Multiple-pass averaging filter:
- passing the input data several times through a moving average filter.

23

Embedded DSP: Moving Average Filters

- A great advantage of the moving average filter is that the filter can be implemented with an algorithm which is very fast.
- Example: 9-point moving average filter
- Y[30] = x[26] + x[27] + x[28] + x[29] + x[30] + x[31] + x[32] + x[33] + x[34]
- Y[31]= x[27] + x[28] + x[29] + x[30] + x[31] + x[32] + x[33] + x[34] + x[35]
- x[27] to x[34] must be calculated for y[30] and y[31] !
- If y[27] has already been calculated the most efficient way for y[31] is:
- y[31] = y[30] + x[35] - x[26]
- y[i] = y[i-1] + x[i+p] - x[i-q]; with: p = (M - 1) / 2, q = p + 1

24

Embedded DSP: FIR Filters

- An FIR filter is a weighted sum of a limit set of inputs. The equation for a FIR filter is

$$y[n] = \sum_{k=0}^{m-1} b_k x[n-k]$$

x(n-k) is a previous history of inputs
y(n) is the filter output at time n
b_k is the vector of filter coefficients

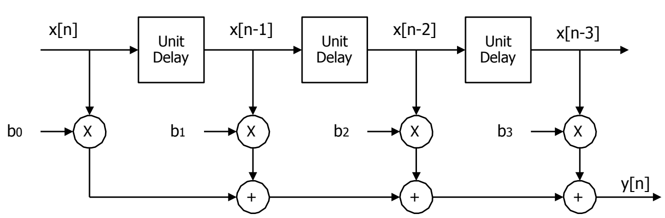
$$y(n) = b_0\, x(n) + b_1 x(n-1) + b_2 x(n-2) + \ldots + b_M x(n-m-1)$$

- For linear phase FIR filters, all coefficients are real and symmetrical
- FIR filters are easy to realize in either hardware (FPGA) or DSP software
- FIR filters are inherently stable
- FIR filtering is a convolution in time: $y[n] = \sum_{k=0}^{\propto} h(k)x[n-k]$

25

Embedded DSP: FIR Filters

- The basic building-block system are the multiplier, the adder and the unit-delay-operator.



$$y(n) = b_0\, x(n) + b_1 x(n-1) + b_2 x(n-2) + b_3 x(n-3) \ldots + b_M x(n-m-1)$$

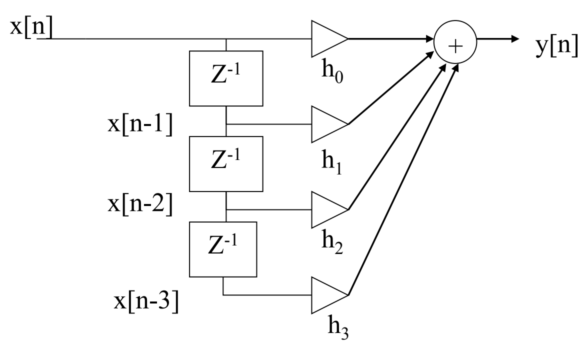
26

Embedded DSP: FIR Filters

- FIR filter have several advantages that make them more desirable than IIR filters for certain design applications:
  - FIR can be designed to have linear phase. In some applications phase is critical to the output. For example, in video processing, if the phase information is corupted the image becomes fully distorted.
- FIR filters are always stable, because they are made only of zeros in the complex plane.
- Overflow errors are not problematic because the sum of products operation is realized ona finite set of data.
- FIR filters are easy to understand and implement.
- FIR filter costs computation time (dependant on filter length !)

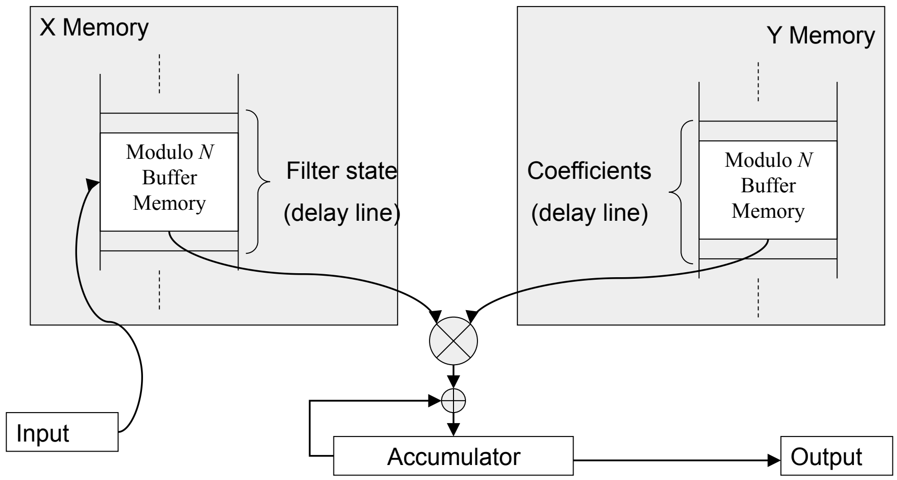
27

FIR Filter



$$H(z) = \sum_{n=0}^{3} h_n z^{-n}$$

FIR Setup



IIR Filters

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_N z^{-N}}$$

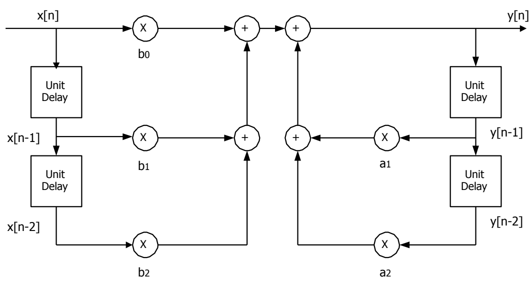
- Advantages
  - Fewer coefficients for sharp cutoff filters.
  - Able to calculate coefficients for standard filter (Bessel, Butterworth, Dolph-Tschebyscheff, Elliptic ....).
- Disadvantages
  - Existing non-linear phase response.
  - Filter can be unstable: precision of coefficients is important, adaptive filters are difficult to realize.

30

Embedded DSP: IIR Filters

- Because IIR filters corresponds directly to analog filters, one way to design IIR filters is to create a desired transfer function in the analog domain and then transform it to the z-domain. Then the coefficients of a direct form IIR filter can be calculated from the z-domain equation. The following equation is the direct form of the biquad difference equation of an IIR filter:

$$y(n) = b_0\, x(n) + b_1 x(n\text{-}1) + b_2 x(n\text{-}2) + a_1 y(n\text{-}1) + a_2 y(n\text{-}2)$$



31

IIR Filters

- IIR (infinite impulse response) filters allow zeros *and* poles; FIR allow zeros only. IIR can be more *selective* for a given filter order
- IIR also called *recursive* filters: output depends on past inputs *and* past outputs
- IIR designs are not guaranteed to be stable
- IIR filters can be particularly sensitive to coefficient quantization

IIR Issues: Stability and Sensitivity

- Finite precision of coefficients can lead to several issues:
  - In order to be unconditionally stable and causal, all system poles must be inside the unit circle ($|z|<1$). Coefficient roundoff may inadvertently move a pole outside unit circle
  - Finite coefficient precision "quantizes" pole locations: may change frequency response from ideal case even if still stable

Overflow Issues

- Gain from input to storage nodes in the filter may exceed unity. This can cause filter state to be saturated (clipped), resulting in distortion
- Typically must scale down (attenuate) the input signal, then scale up (amplify) by an equal amount on the output

Second-Order Sections

- High-order filter polynomials involve terms that are products and sums involving many poles and zeros. Small roundoff errors when implementing filter can lead to large response errors
- As with analog filters, it is typical to reduce sensitivity by using *second-order sections*

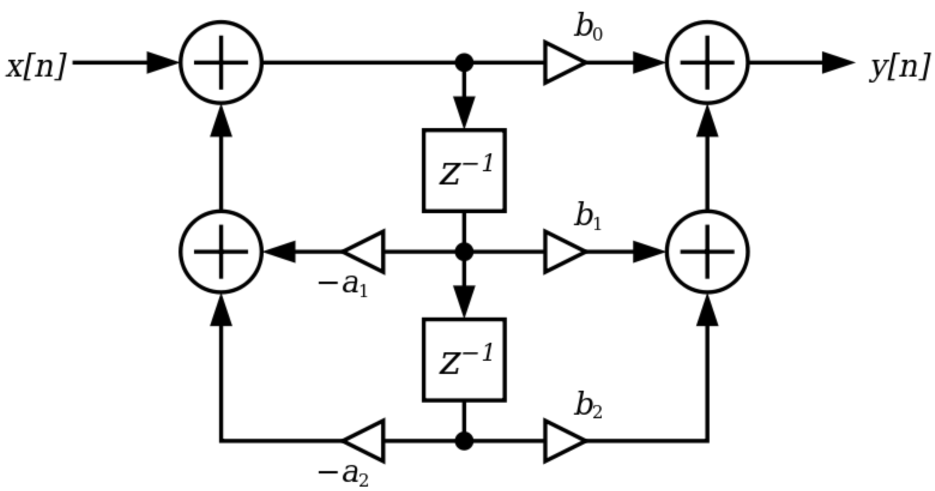
Implementing 2nd Order Sections

- 2nd Order (bi-quad) expression

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$
$$= b_0 \frac{1 + k_1 z^{-1} + k_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

- Numerator implements 2 zeros, denominator implements 2 poles (real or complex conj.)

Canonical Direct Form Bi-Quad 2



Filter Design on-line

- Interactive Filter Design Tool-   IIR and FIR with C code
  http://www-users.cs.york.ac.uk/~fisher/mkfilter/
- T-filter– FIR designer
  http://t-filter.appspot.com/fir/index.html
- Example C Code for FIR and IIR Filters
  http://iowahills.com/A7ExampleCodePage.html

And there is always Matlab– available on Attu