DAT093
Introduction to Electronic System Design

Laboratory assignment 2

FIR Filters

Sven Knutsson
svenk@chalmers.se
Dept. Of Computer Science and Engineering
Chalmers University of Technology
Gothenburg
Sweden

Goal

Introduce methods to implement transversal digital filters

Finite Impulse Response Filters (FIR)

Assignments

- Direct (parallel) implementation of a FIR filter
- Serial implementation of a FIR filter
- Implementation of a FIR filter using distributed arithmetics

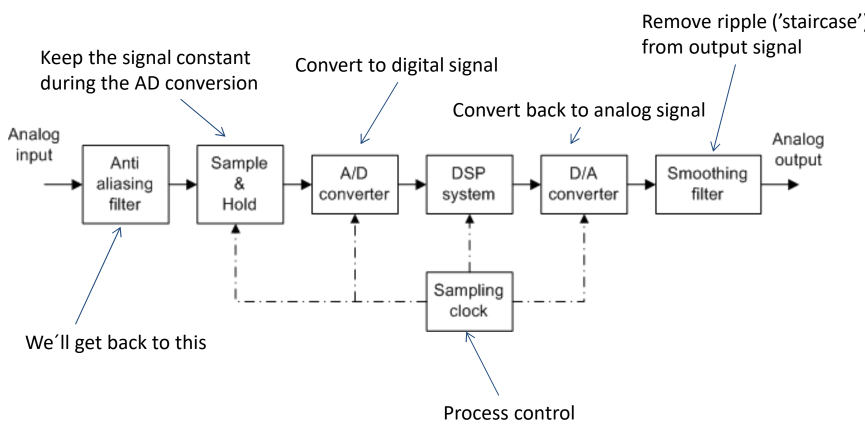
The distributed implementation is for the digilent student but won´t give any extra credits

There are test benches for the direct and the serial implementation on the PingPong homepage

To begin with we need to talk about DSP (Digital Signal Processing) systems, sampling and digital filters

DSP systems

Let´s look at a typical DSP system



DSP systems

Sampling

A digital system can not work with signals that are continous in time

We need to read the analog input signal at discrete times

This is normally done at constant time intervals

We sample the signal using a clock

How often do we need to sample the signal?

Demonstration!

DSP systems

Sampling cont.

Conclusion! The sampling theorem

We need to sample the signal more than twice each signal period.
That is the sampling frequency must be more than twice the highest signal frequency

If this is not accomplished we will get false signal frequences

We will get aliazing

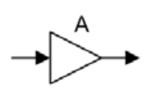
This kind of distorsion can not be removed afterwards

The frequencies of these false signals are not random though.
We can calculate them if we like

DSP systems

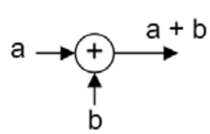
What are the mathematical tools we have in a DSP system?

- Scaling (amplification)



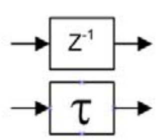
y[n] = A·x[n]

- Summation



y[n] = a[n] + b[n]

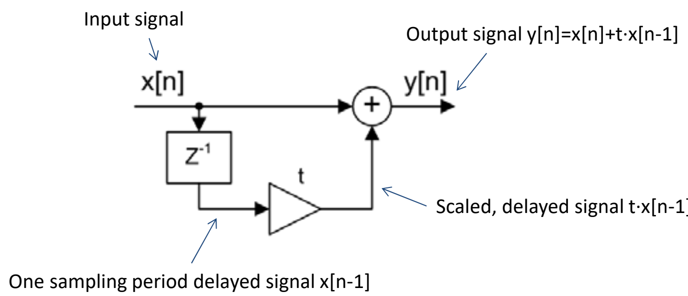
- Delay



y[n] = x[n-1]

DSP systems

A simple DSP system (filter)



Demonstration!

DSP systems

A more general system

We can use more samples to build a more complex filter

Output

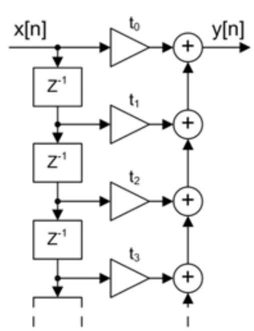
The sample one sampling period ago

$$y[n]=t_0 \cdot x[n]+t_1 \cdot x[n-1]+t_2 \cdot x[n-2]+t_3 \cdot x[n-3]+\Lambda$$

Current sample

The sample two sampling periods ago

$$y[n]=\sum_{k=0}^{N-1} t_k \cdot x[n-k]$$



DSP systems

A more general system cont.

$$y[n] = t_0 \cdot x[n] + t_1 \cdot x[n-1] + t_2 \cdot x[n-2] + t_3 \cdot x[n-3] + \Lambda$$

$$y[n] = \sum_{k=0}^{N-1} t_k \cdot x[n-k]$$

The number of terms in the sum, N, is called the order of the filter or the number of taps

Observe that an impulse will pass the filter in N sampling periods and that the output series will be the same as the filter coefficients

That is the impulse response is finite in time

We call this a Finite Impulse Response filter (FIR)

It is also called a transversal filter

The filter coefficients are given by the impulse response of the system

Demonstration!

DSP systems

An even more general system

We can also use delayed output signals to form our system

$$y[n] = t_0 \cdot x[n] + t_1 \cdot x[n-1] + t_2 \cdot x[n-2] + t_3 \cdot x[n-3] + \Lambda \; +$$

$$+ \; r_1 \cdot y[n-1] + r_2 \cdot y[n-2] + r_3 \cdot y[n-3] + \Lambda$$

We can not use the current output sample since that is the result of the calculation

$$y[n] = \sum_{k=0}^{N-1} t_k \cdot x[n-k] + \sum_{p=1}^{M-1} r_p \cdot y[n-p]$$

We call the system a recursive system

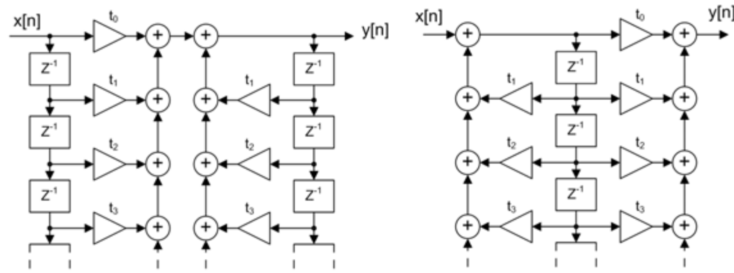
or an Infinite Impulse Response (IIR) filter

This will give more efficiant filters than the transversal filters

but they can get unstable because of the feedback loop

DSP systems

An even more general system cont.



The feedback loop means that the impulse response in the general case never will die, that is it is infinite in time. Therefore the name

Warning! The feedback paths can make the system unstable if the filter coefficients are badly choosen

Demonstration!

DSP systems

How do we design a FIR filter?

There are a number of methods.

Most known are

- Inverse fourier transform
- Equi-Ripple filter (Parks-McClellan)

We will have a look at design using inverse fourier transform

DSP systems

FIR filter design using inverse fourier transform

The discrete version of the inverse fourier transform will give us the impulse response

Impulse response

Normalized angular frequency $\Omega = 2 \cdot \pi \cdot \frac{f}{f_s}$

$$h[n] = \frac{1}{2 \cdot \pi} \cdot \int_{2 \cdot \pi} H(\Omega) \cdot e^{j \cdot \Omega \cdot n} \cdot d\Omega$$

The sampling frequency

The frequency spectra that should correspond to the impulse response

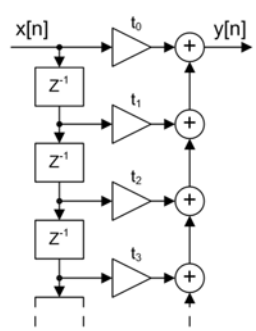
Integration over 2π of the normalized angular frequency is the same as an integration over the frequency range zero to f_s

DSP systems

FIR filter design using inverse fourier transform cont.

In digital systems an impulse is a signal that has the value one (1) at time zero and is equal to zero at all other times.

We can see that if we apply a impulse to this system the output will be a series of values that are the same as our filter coefficients.



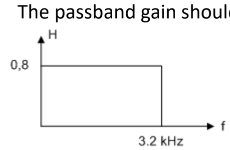
We can conclude that the impulse response has the same values as the coefficients in our FIR filter so we can use the inverse fourier transform to design our filter

DSP systems

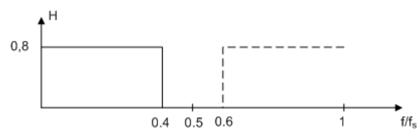
FIR filter design using inverse fourier transform cont.

Example

Use inverse fourier transform to design a low pass filter with the cutoff frequency 3.2 kHz when the sampling frequency is 8 kHz. The passband gain should be 0.8.



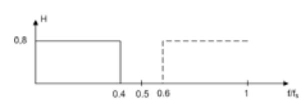
Our digital characteristics gives that there will be a mirror image of the passband above half the sampling frequency. Let´s also normalize the frequency $\frac{f_{cut\,off}}{f_s} = \frac{3.2}{8} = 0.4$



Observe that we have only specified the amplitude behavour. We want a filter with zero phase.

Such a filter can not be realized in real time but we can get linear phase which is just a delay of the signal

DSP systems

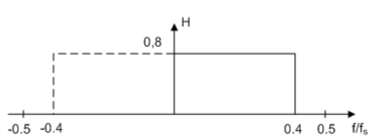


FIR filter design using inverse fourier transform cont.

Now we have to use the inverse fourier transform and integrate to get our time samples.

We can see that there are two intervals where H is separated from zero meaning that we have two integrals to solve.

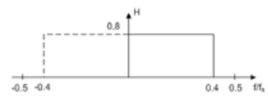
Since the digital spectra is cyclic we can redraw the picture of the spectra using negative frequencies



This will give us only one interval and therefore only one integral.

Let´s calculate

DSP systems



FIR filter design using inverse fourier transform cont.

$$h[n] = \frac{1}{2\cdot\pi}\cdot\int_{2\cdot\pi} H(\Omega)\cdot e^{j\cdot\Omega\cdot n}\cdot d\Omega = \frac{1}{2\cdot\pi}\cdot\int_{-0.4\cdot\pi}^{0.4\cdot\pi} 0.8\cdot e^{j\cdot\Omega\cdot n}\cdot d\Omega =$$

$$= 0.8\cdot\frac{e^{j\cdot 0.4\cdot\pi\cdot n} - e^{j\cdot 0.4\cdot\pi\cdot n}}{2\cdot\pi\cdot n} = \frac{0.8}{\pi\cdot n}\cdot\sin(0.4\cdot\pi\cdot n)$$

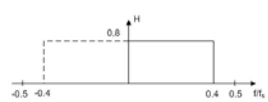
For this to be true we need to include all values of `n` in the impulse response from –infinity to infinity.

This is something that we obviously can not do since it would give a filter with a infinite number of coefficients and this can not be realized.

We have to take a smaller number of coefficients and this will make our representation of the frequency spectra an approximation. It will be less accurate.

The more coefficients the better the approximation but at the same time the complexity and delay through the filter increases.

DSP systems



FIR filter design using inverse fourier transform cont.

It can also be shown that we have to use symmetrical terms, that is use values of `n` that are ±1, ±2 and so on. We will also include the value for `n=0`

This will give a filter

$$y[n] = \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} t_k \cdot x[n-k]$$

Here we have some samples of the type `x[n+k]` and these can not be realized in real time since they are future samples that don´t exist yet.

To solve this we delay or samples and only use current and past samples

$$y[n] = \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} t_k \cdot x\left[n-k-\frac{N-1}{2}\right]$$

Our phase contribution will because of this no longer be zero but it will be linear, which means that there will be a constant and equal time delay through the filter for all frequencies, all frequencies are delayed the same amount of time

DSP systems

FIR filter design using inverse fourier transform cont.

Since we are using symmetrical values and the value for `n=0` is included the number of filter coeffients will be odd.

We can use an even number of filter constants but then we have to calculate the coefficients using

$$n = -\frac{N}{2} + 0.5, -\frac{N}{2} + 1.5, \Lambda\ , -0.5, 0.5, \frac{N}{2} - 1.5, \frac{N}{2} - 0.5$$

Demonstration!

DSP systems

Our system

We will focus on FIR filters and create a four tap filter in three different ways.

Just four taps is not enough coefficients to give a good filter but it is enough to demonstrate the principle and still keep the debugging pretty simple.

We have to start by describing how to represent the filter coefficients

We will talk about fractional numbers

DSP systems

Fractional numbers

A fractional number is a number with
a magnitude less or equal to one.

If we look at a four bit signed number this means that
for a positive number we have

$$0b_{-1}b_{-2}b_{-3} = b_{-1}\cdot 2^{-1} + b_{-2}\cdot 2^{-2} + b_{-3}\cdot 2^{-3} = b_{-1}\cdot\frac{1}{2^1} + b_{-2}\cdot\frac{1}{2^2} + b_{-3}\cdot\frac{1}{2^3} =$$

$$= b_{-1}\cdot 0.5 + b_{-2}\cdot 0.25 + b_{-3}\cdot 0.125$$

MSB is always zero for a positive number

We can see that the maximal value would be

$$0.5 + 0.25 + 0.125 = 0.875$$

That is almost one, but not quit

DSP systems

Fractional numbers cont.

The maximal positive value of a fractional number is always

$$1 - 2^{-weight\ of\ LSB}$$

Let´s look at a negative number, still with four bits.

The negative number with the highest magnitude is 1000.

We convert it to magnitude (2´s complement).

```
        111
1000 → 0111
      +   1
       ----
       1000
```

and we will get

$$1000 = 1\cdot 2^0 + 0\cdot 2^{-1} + 0\cdot 2^{-2} + 0\cdot 2^{-3} = 1$$

That is our largest negative number is -1 so there is a slight
difference between the positive and the negative side

t_k

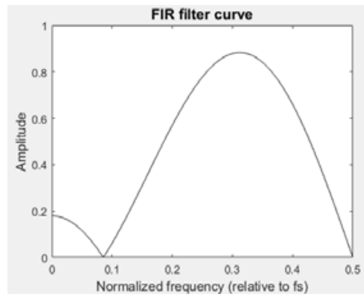
DSP systems

Fractional numbers cont.

In the assignment you have four fractional filter constants

k	t_k
0	-0,32
1	0,23
2	0,23
3	-0,32

The filter will have the filter curve given below



Use the given method to convert the cofficients to binary fractional numbers.

You will not get the exact values but go as close as possible with the choosen number of bits. The testbench assumes that you truncate your values

There is an additional text on fractional numbers on the homepage

Demonstration!

t_k

DSP systems

Filter implementations

Direct implementation

We have the filter equation

$$y[n] = t_0 \cdot x[n] + t_1 \cdot x[n-1] + t_2 \cdot x[n-2] + t_3 \cdot x[n-3] = \sum_{k=0}^{3} t_k \cdot x[n-k]$$

This equation can be directly implemented as a number of multiplications and summations and this is our first goal.

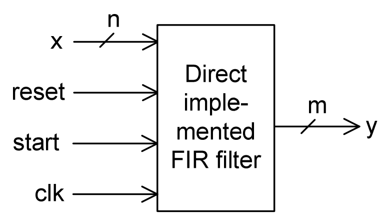
Make sure to choose the number of bits in your vectors so there is no risk of overflow.

DSP systems

Filter implementations

Direct implementation cont.

To simplify testing we will use a predefined interface for our design



In this lab assignment we leave out the sampling of the signal and just use a start signal to trigger the calculatation of a new output sample.

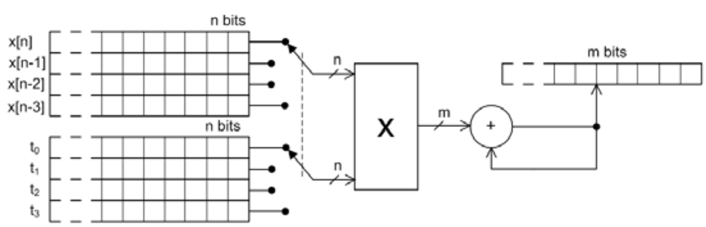
For this to work properly you must include aclock signal i your design.

DSP systems

Filter implementations cont.

Serial implementation

We serialize the circuit and only use only one multiplier and one adder



This will of course be slower but we use less hardware

The implementation requires a clock do shift the values

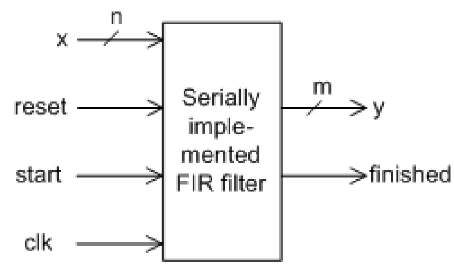
Demonstration!

DSP systems

Filter implementations cont.

Serial implementation cont.

Once again we stick to a defined interface for our design



Here we need a clock signal to shift in the bits and a signal to indicate when the calculation is done

DSP systems

Filter implementations cont.

Distributed arithmetics

Let´s have a look at a very simple FIR filter with just three taps and three bits word length

$$y[n] = \sum_{k=0}^{2} t_k \cdot x[n-k] = t_0 \cdot x[n] + t_1 \cdot x[n-1] + t_2 \cdot x[n-2]$$

We break down the samples to individual bits

$$y[n] = (t_0 \cdot x[n]_{b0} + t_1 \cdot x[n-1]_{b0} + t_2 \cdot x[n-2]_{b0}) \cdot 2^0 +$$ ← LSB of samples

$$+ (t_0 \cdot x[n]_{b1} + t_1 \cdot x[n-1]_{b1} + t_2 \cdot x[n-2]_{b1}) \cdot 2^1 +$$

$$+ (t_0 \cdot x[n]_{b2} + t_1 \cdot x[n-1]_{b2} + t_2 \cdot x[n-2]_{b2}) \cdot 2^2$$ ← MSB of samples

DSP systems

Distributed arithmetics cont.

If we look at the sum of products for LSB of the samples

$$t_0 \cdot x[n]_{b0} + t_1 \cdot x[n-1]_{b0} + t_2 \cdot x[n-2]_{b0}$$

this is really only a sum of constants since we only have one bit from each sample and this bit can only be one or zero

We can have $2^{\text{number of taps}}$ different sums

We can calculate these sums one time for all in the design phase and then store these values in a memory where we use the current bit from the samples as addresses

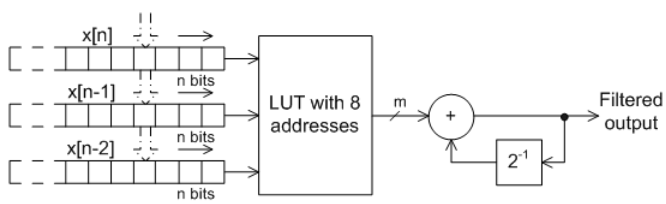
To perform the filter calculation we can now shift out bit by bit from the samples, use them as addresses to the memory and step by step sum up our filter output

Remember that the new product have to be shifted to the left before it is added to the earlier temporary sum

We can accomplish the same thing by shifting the earlier sum to the left

DSP systems

Distributed arithmetics cont.



Demonstration!

DSP systems

Distributed arithmetics cont.

We can use the same interface as we used for the serial implementation