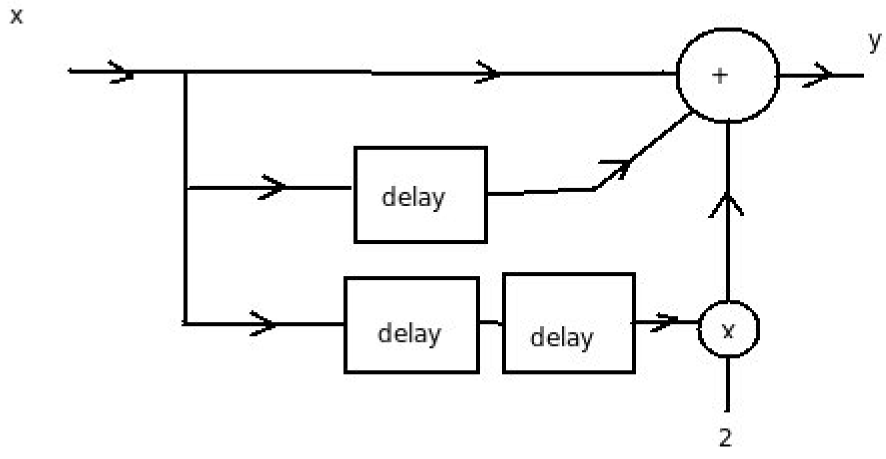
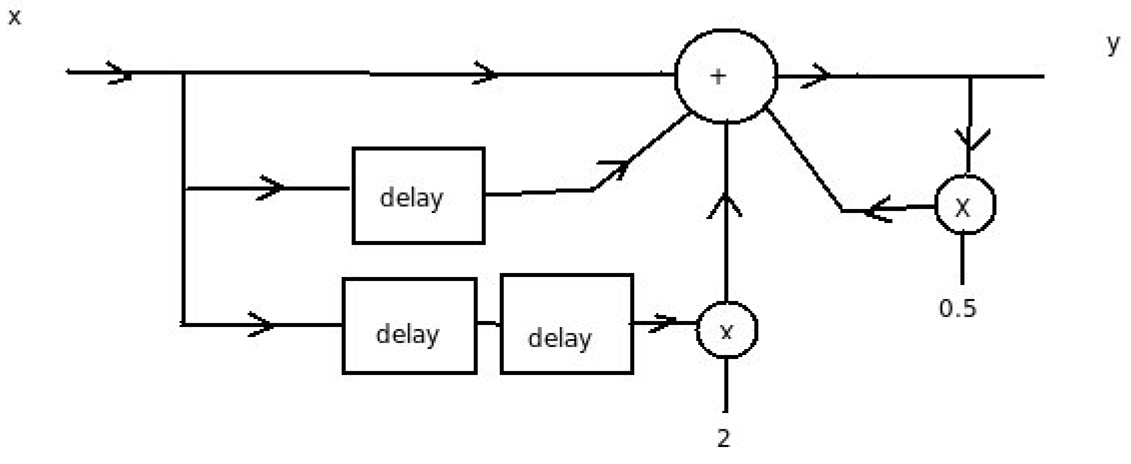
FIR, IIR and impulse invariance

FIR and IIR
Digital filters implementations are said to be FIR or IIR. What's the difference? It all comes down to where the inputs come from i.e. The form of the defining equation. An FIR design only uses weighted and/or delayed inputs, so the equation is similar to this:-
$y_n=x_n+x_{n-1}+2*x_{n-2}$



What happens when we apply a single '1' signal at the input (not binary 1, denary 1)? i.e. A sequence of samples like this are applied: 0,0,0,0,01,0,0,0,0. What comes out? This is easy for such a simple system, 0,0,0,0,0,1,1,2,0,0. Note that the output response starts at 0 and after three sample periods it goes back to 0. This is a 'Finite Impulse Response' (FIR). Another really useful and interesting thing to note is that that the samples making up the impulse response are the SAME in value as the coefficients of the terms on the RHS of the equation. (When the input is a unit impulse). We will come back to this.
Now for IIR. IIR uses weighted and/or delayed input and outputs. Modifying the original equation above, it might look like this:
$y_n=x_n+x_{n-1} +2*x_{n-2} +0.5*y_n$



Lets put in the 0,0,0,0,1,0,0,0,0 signal (the impulse). What comes out this time? Assuming the output of the summation is sample and hold (which I do), then I think it is: 0,0,0,0,1,1.5,2.75,1.375,0.6875.0.34375,0.171875.................... in theory at least the output never again falls to zero, it just gets smaller and smaller. In practice, there will be a threshold below which the output can be considered to be zero. The effect of the 'feedback' of the output sample is significant and gives an 'Infinite Impulse Response' (IIR) implementation.

IIR looks more complicated, why bother with it?

To get a complex specified implementation in FIR can often require huge numbers of terms. So much so that the computation required gets out of hand for practical speed. Also, some analogue defining equations lend themselves much more easily to IIR. Here's an example of a 2nd order low pass Butterworth filter (maximally flat in the passband). As the algebraic simplification gets a bit tedious without using tools like MATLAB I'll use the example from emfs1.eps.hw.ac.uk/~pf21/pages/page3/assets/Ch5Lec1.pdf
where the author designs a filter with a cut off freq of 100Hz and a sampling frequency of 800 Hz. After simplification the Z-plane transfer function is:

$$H(z) = \frac{0\cdot098 + 0\cdot195z^{-1} + 0\cdot098z^{-2}}{1 - 0\cdot942z^{-1} + 0\cdot333z^{-2}}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0\cdot098 + 0\cdot195z^{-1} + 0\cdot098z^{-2}}{1 - 0\cdot942z^{-1} + 0\cdot333z^{-2}}$$

multiply out:

$$Y(z)\left(1 - 0\cdot942z^{-1} + 0\cdot333z^{-2}\right)$$
$$= X(z)\left(0\cdot098 + 0\cdot195z^{-1} + 0\cdot098z^{-2}\right)$$

and carrying on with the process:

$$Y(z) - 0.942\,z^{-1}.\,Y(z) + 0.333\,z^{-2}.\,Y(z)$$
$$= 0.098.\,X(z) + 0.195z^{-1}.\,X(z) + 0.098z^{-2}.\,X(z$$

$$Y(z) = 0.98\,X(z) + 0.195z^{-1}.\,X(z) + 0.098\,z^{-2}.\,X(z)$$
$$+\,0.942\,z^{-1}.\,Y(z) + 0.333\,z^{-2}.\,Y(z)$$

which yields the difference equation!

$$y(n) = 0.098\,x(n) + 0.195\,x(n-1) + 0.098\,x(n-2)$$
$$+\;0.942\,y(n-1) + 0.333\,y(n-2)$$

Note that the output is a summation of weighted/delayed input and output samples and so the implementation is IIR by virtue of how the algebra pans out.

About impulses and their uses

Let's kick off in the analogue world. Remembering Fourier synthesis it is clear that to get a high speed edge you have to include a large number of harmonics etc. In fact to arrive at an instantaneous rise-time, in theory, you need all frequencies. So, an edge or an impulse with a fast rise-time and short duration must, by implication contain a vast range of frequencies. If you apply an impulse to a circuit, its output will respond. (a time domain response). If you Fourier transform this output you will have the frequency response of the circuit.
Clearly then, the impulse response of a system is linked unambiguously with the frequency response that it has. The impulse response defines the performance every bit as accurately as the frequency response, its just a different viewpoint.

Remember the first topic in this paper (FIR), in that example the output sample values for an impulse input were the same as the coefficients of that filter. This proves to be a general rule. If you know the or can define the ideal impulse response, just sampling it gives you the coefficients (multipliers) you need to specify the structure. The structure is a manifestation of the difference equation and the difference equation is a time domain prescription of how to multiply-accumulate the samples for the performance you want. Sampling the ideal impulse response to get the coefficients like this always produces an FIR solution.

If you were to Fourier transform the impulse response you will get the frequency response and we can use this fact if we want to build an IIR solution to an analogue design that is itself IIR in nature.

IIR filter design by sampling the impulse response

You can find the frequency response of a digital system if you know the coefficients. Unfortunately this does not work in the reverse direction, or at least no-one knows how to do it yet! It is an impossibility (as far as I know) to find the filter coefficients directly from a specified frequency response.

Now, we have seen how the the impulse response characterizes the frequency response. We have also seen how the coefficients of an FIR system defined by the difference equation are the same as the impulse response. Thus we can put two and two together and suggest a way to get a desired filter, but because of the facts outlined in the paragraph above we cannot start a digital design with with a desired frequency response specification directly. What we must do is:

1. Define a desired frequency response
2. Design an analogue filter to do the job
3. Use an impulse input to produce the analogue impulse response
4. Sample the analogue impulse response to produce a series of samples defining the shape
5. Inverse z-transform the sample set to get the z-plane transfer function ($H(z)=Y(z)/X(z)$, and just like the Butterworth equations above multiplying out etc will produce an IIR difference equation. (Provided the analogue filter in 2. above was IIR).

Doing things this way is called the 'impulse invariance' method. Impulse invariance means that digital filter impulse response exactly equals samples of the analog prototype impulse response.

Any problems with impulse invariance design

Oh yes! Solutions arrived at this way are very susceptible to issues arising from aliasing and also from frequency resolution factors. It could be argued that the Bilinear transform, as an alternative, is better.

Rationale for this paper

To follow on with more background on the DSP terms and their meanings before the evening on using the PIC for DSP applications that is hopefully to be arranged.