

## 6.111 Lecture # 14

### Lab 3 Assignment:

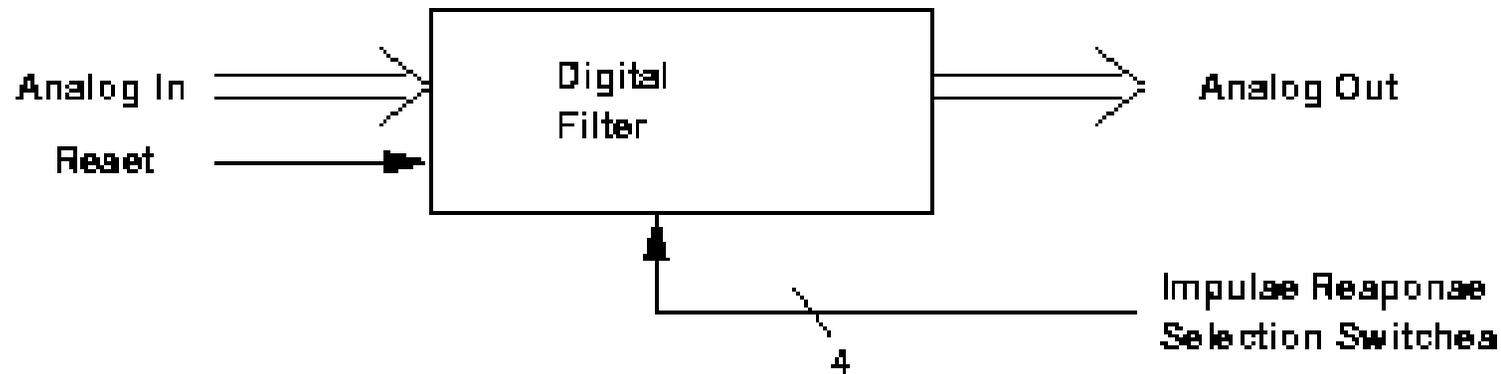
**Digital filter**

**Audio Frequencies (music, speech)**

**Input from waveform generator, microphone, 'boom box'**

**Output to oscilloscope or speaker**

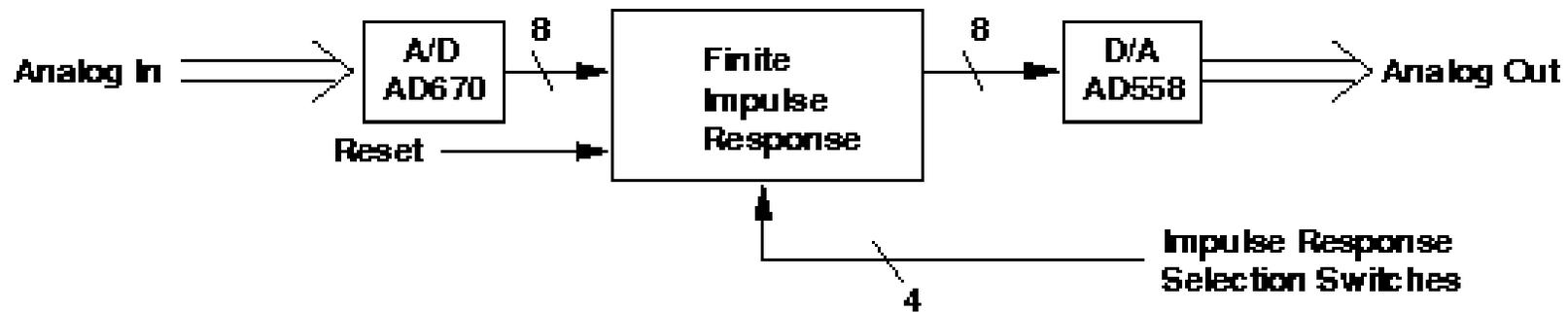
**We provide a variety of filters: your kit must provide for a selection of one of 16 (4 bits) from switches**



**Input is digitized and sampled**

**Filter is FIR (next slides)**

**Output is through D/A converter (as in sample and hold)**



**This is what your machine should do:**

**Wait for a fixed time interval**

**Sample the input signal**

**Output what was computed in the last sample**

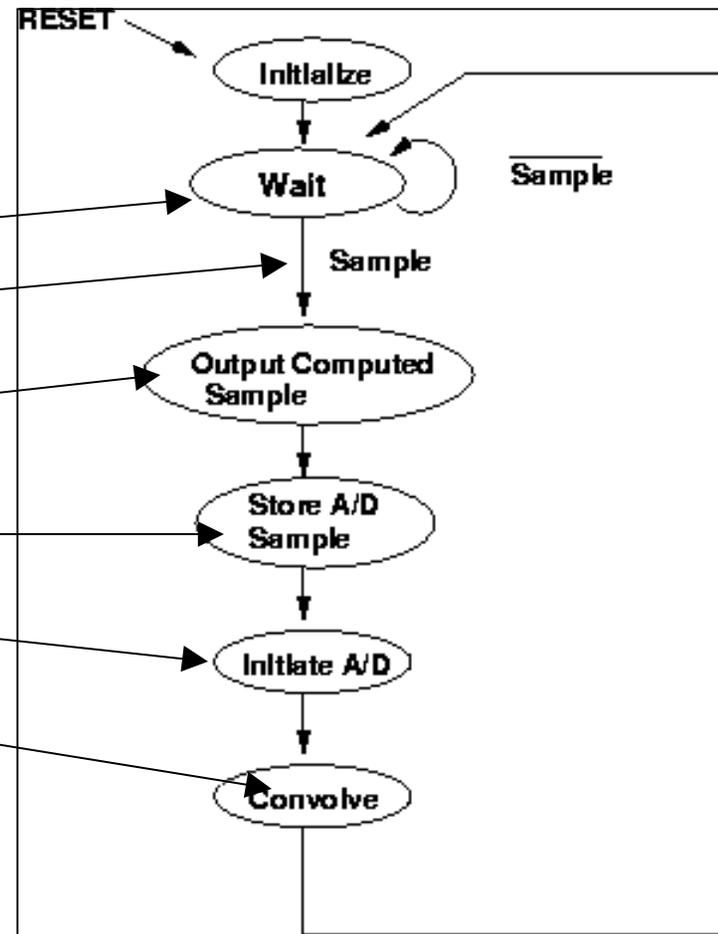
**Store the sample just taken**

**Start the A/D conversion**

**Do the FIR filter algorithm, which is a convolution**

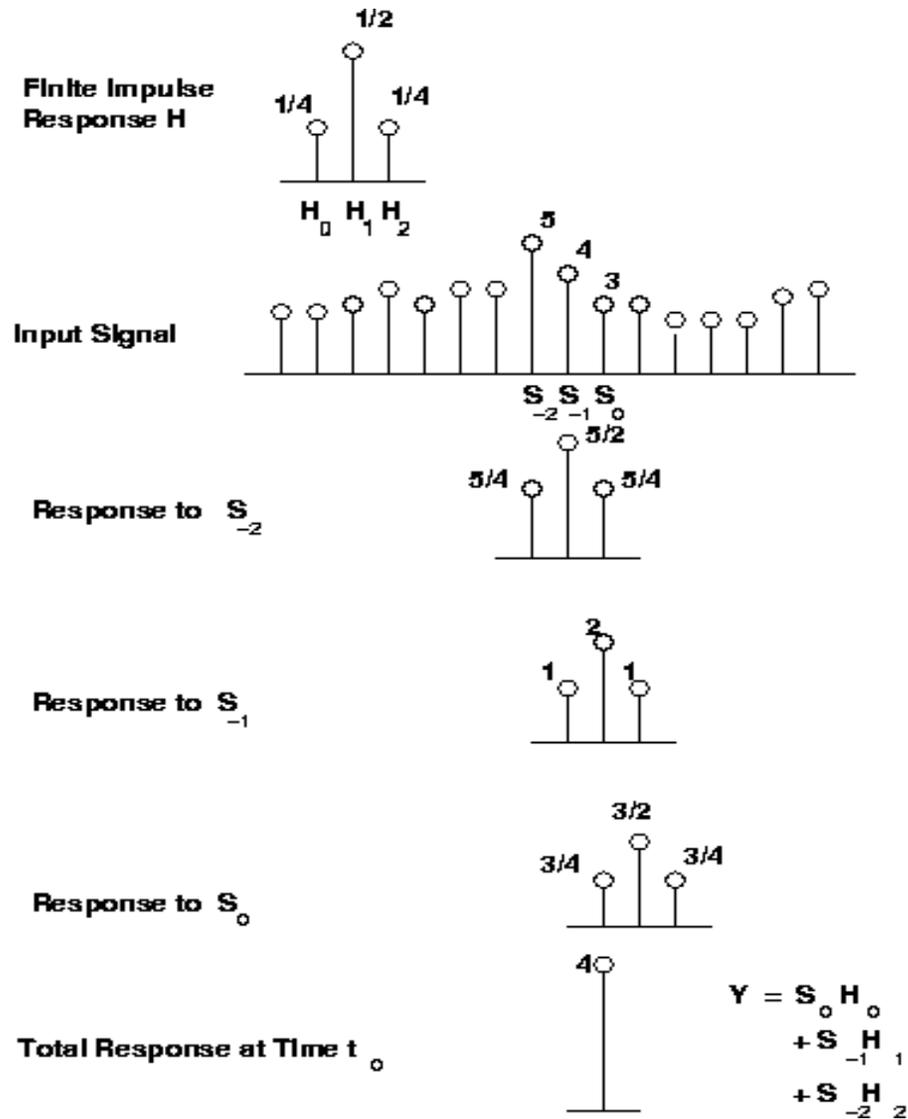
**(No, this is not out of order)...**

**Go back for more...**



Convolution is a weighted sum of the last n samples.

The FIR ('Finite Impulse Response') filter can do most all filtering functions (high pass, low pass, etc.)



**You should implement this filter using:**

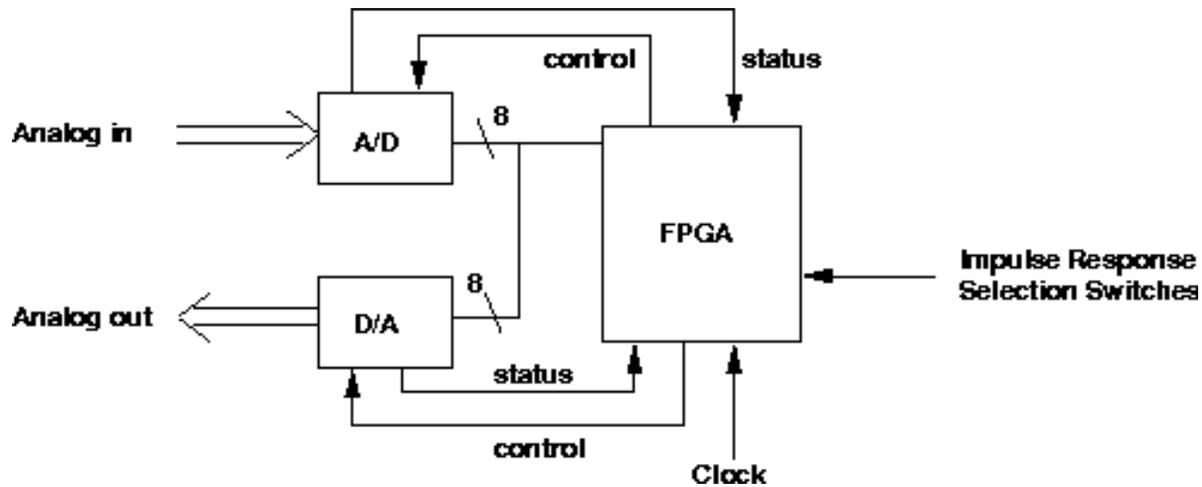
**Altera Flex: EPF10K10 or EPF 10K70 FPGA**

**Analog Devices AD670 A/D converter**

**Analog Devices AD 580 D/A converter**

**(If necessary) External RAM**

**The staff thinks you can fit it into a 10K10, and if so that is good...**



This is a functional block diagram of what must be built.

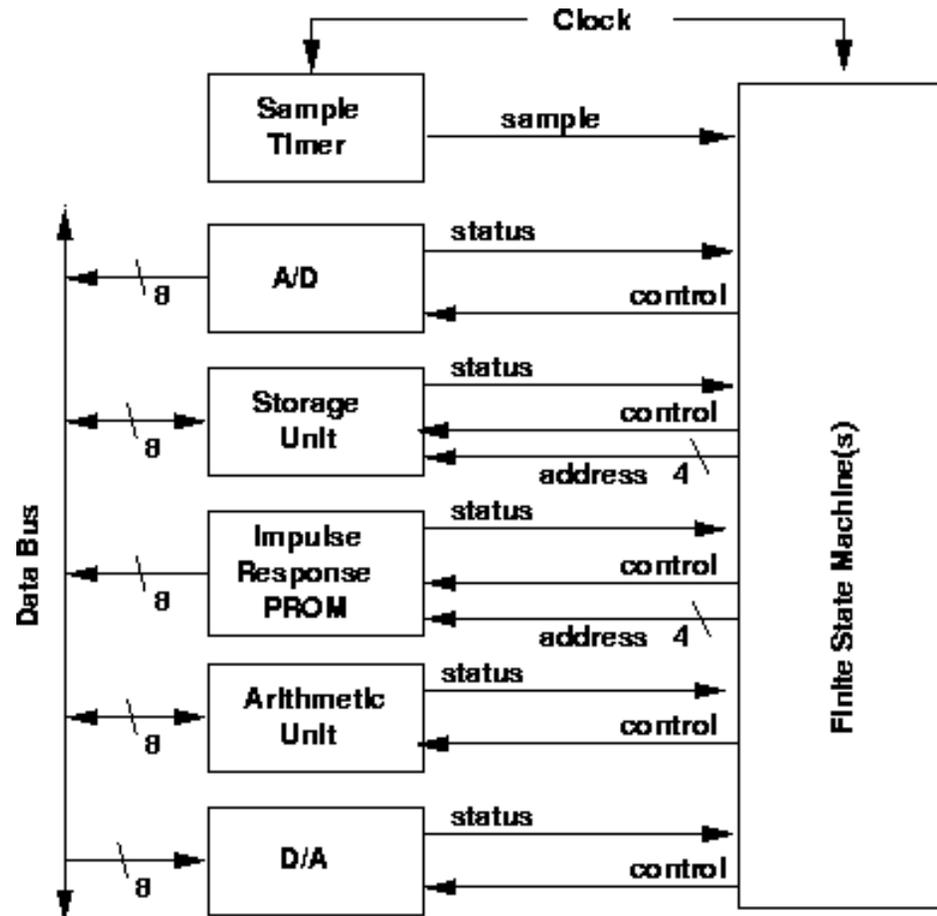
Timer figures out when to start each operation

A/D converter gets the signal from the analog

Storage, Arithmetic unit and impulse response ROM are used to compute the output.

Output goes out to analog through D/A converter

A/D, D/A, PROM are external to the FPGA

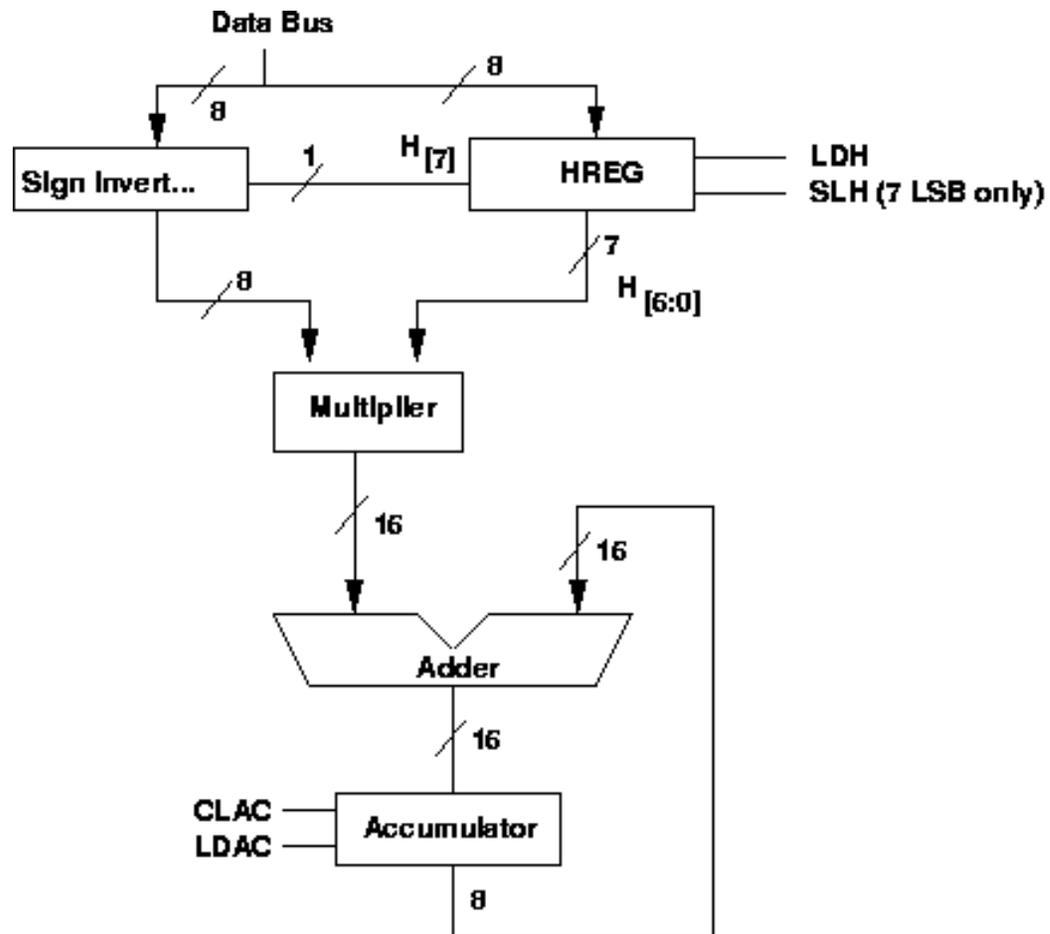


Convolution is a series of multiplications and summations. These are (fairly) easily done in a circuit like this:

H holds the impulse response (in sign/magnitude form).

The data word needs to be inverted if H is negative.

The signal is accumulated in the bottom part of the circuit.

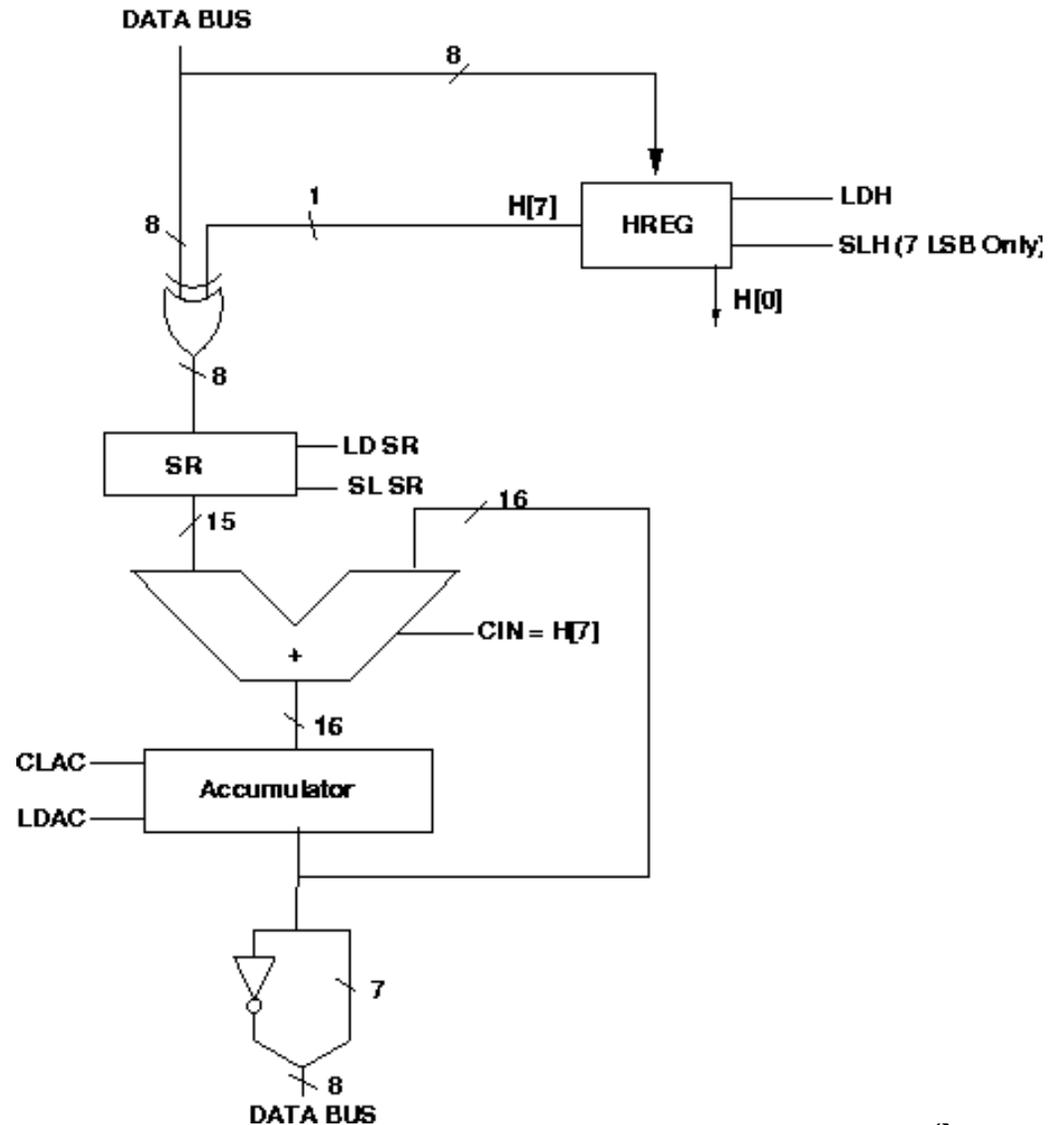


Here is a little more detail on how this might be done.

The xor and CIN do the 2's complement inversion if H is negative.

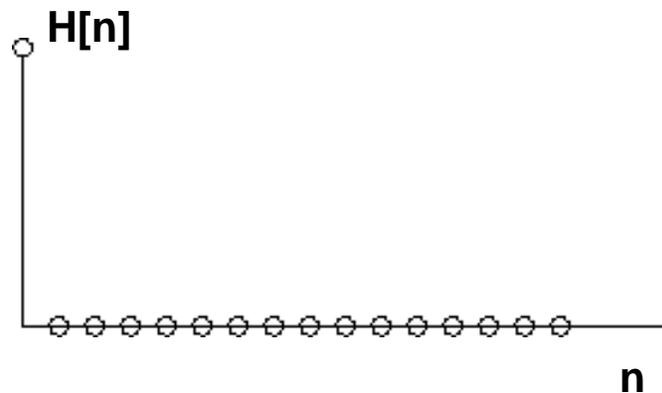
This is a shift and add multiplier: H[0] = 1 implies 'load' the accumulator (add the shifted H to the result).

The accumulator keeps a running sum. Successive numbers can be multiplied and added.



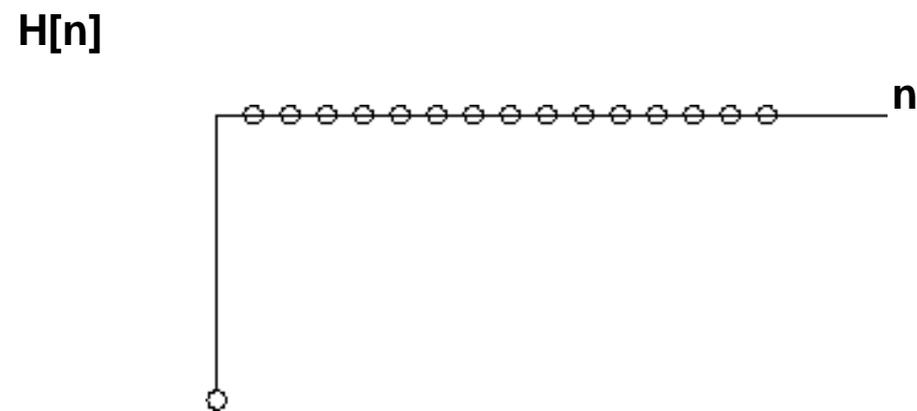
The filters you will be working with are available as a file of 16 filters with 16 numbers each. The file is supplied as a handy-dandy, ready to load into a ROM file.

This is the form of the first filter (an impulse), which, when applied to a signal should return the signal unaltered.

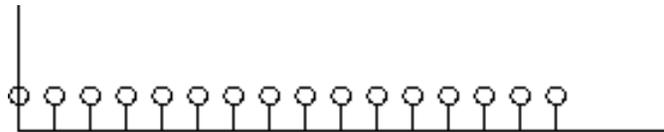


**This is filter number two: an inverse impulse**

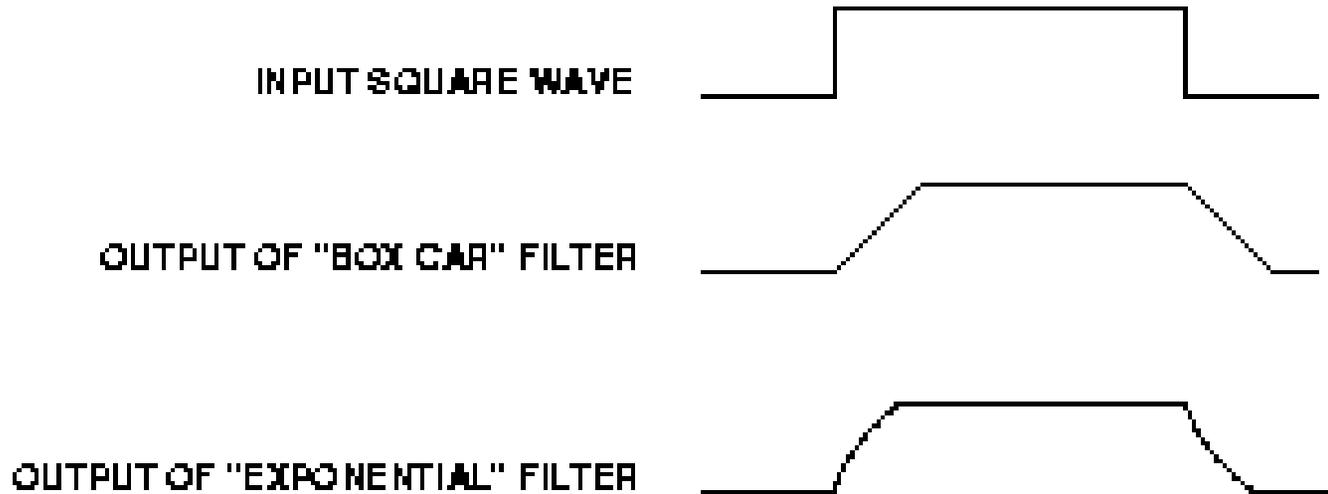
**Output = - Input**



**This is the third filter: a 'boxcar'**



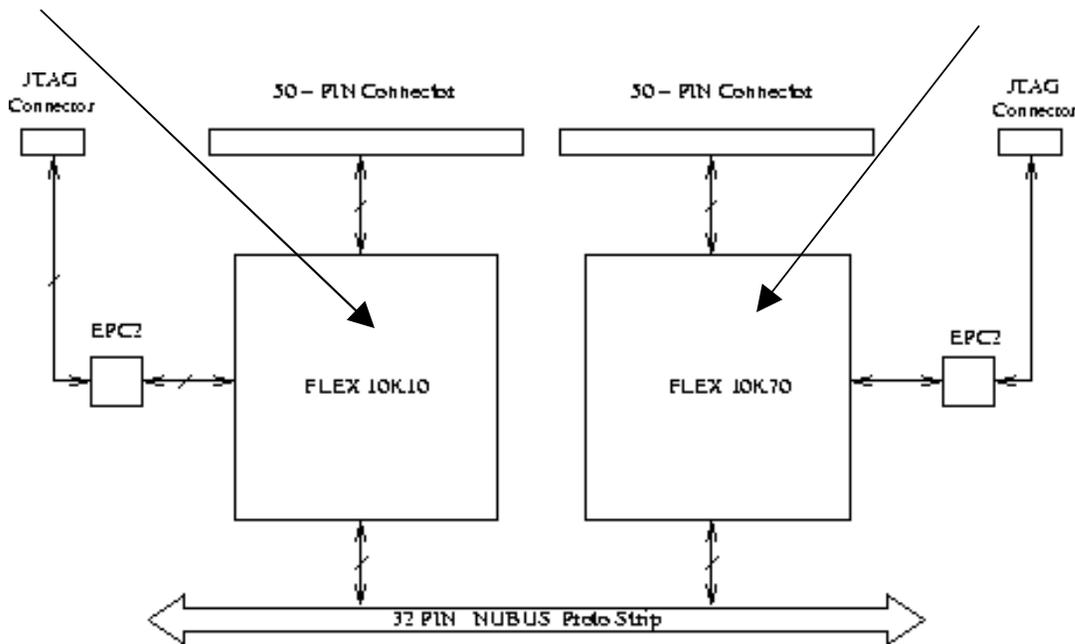
**And here are outputs for a couple of different filters:**



**That FPGA board (soon to be) in your kit has:  
one FLEX10K10 and one FLEX 10K70**

**About 10 k gates  
Socketed**

**About 70 k gates  
Surface mount (hard to replace!)**



**Note these parts are interconnected and connected to the Nubus and two 50-pin connectors. Must be erased (tr-stated) before you can use them for anything!**