Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.111 - Introductory Digital Systems Laboratory

Laboratory 3

Digital Filter

Text by Donald E. Troxel - August, 2002
Figures by James L. Kirtley - August, 2002

Issued: October 9, 2002
Analog Check-off: October 21, 2002
Design: October 25, 2002
Lab Check-off: November 6, 2002
Report Due: November 13, 2002

INTRODUCTION

This lab gives you the opportunity to design, construct, and test a reasonably complex digital system. A design review with a T.A. is to be held after you have completed the initial design portion. After the design review, you may proceed with the final design, construction, testing, demonstration, and the required lab report. There is an earlier check-off of the analog portion of your laboratory.

The purpose of Lab 3 is to familiarize you with the design of a complex system. Also this is your opportunity to become familiar with the use of the FPGA boards. You are strongly encouraged, though not required, to use an FPGA instead of RAM, ROM, PALs, CPLDs, etc. You will also have to use analog to digital and digital to analog converters. The lab has two check-offs, the first being the analog portion and the second being the digital part. The digital part should fit into either of the FPGAs on the FPGA board. The analog section must be designed and checked off before the rest of the lab is completed. Please also pay attention to the debugging strategy which is described later in this document.

A NOTE OF CAUTION

This problem, while not impossible, is not trivial. It is substantially more difficult in complexity than the previous lab exercises. You are strongly urged to start thinking about this design task as soon as possible.

This handout provides substantial guidance for your design. However, you are NOT required to follow these suggestions exactly. You are required to use an A/D and a D/A converter.
You are encouraged to use the FPGA boards and to simulate your design before testing your design.

We will provide, at suitable stations throughout the lab, sources of a variety of analog signals. You may view the results of processing by your digital filter by observing waveforms with an oscilloscope and also by listening to them with amplified speakers which are available at the Instrument Room desk. You may also find it interesting to cascade your Lab 3 with those of your friends to observe the results of more complicated impulse responses. Can you undo what an impulse response can do?

OVERVIEW

Your task is to build a machine which will accept analog signals and produce a filtered version of this input signal. There are many ways to accomplish this task. Digital filters can be implemented by convolution, difference equations, and also by FFT (Fast Fourier Transform). We will choose the FIR (Finite Impulse Response) convolution approach.

A simple, overall block diagram is shown in Figure 1. We provide a file, impulses.ntl, which contains all sixteen of the impulse responses you are required to demonstrate. You may implement additional impulse responses of any length and content you choose. The RESET signal (from a switch) should initialize your machine as required and start convolving the input analog signal with the impulse response which is selected by switches.

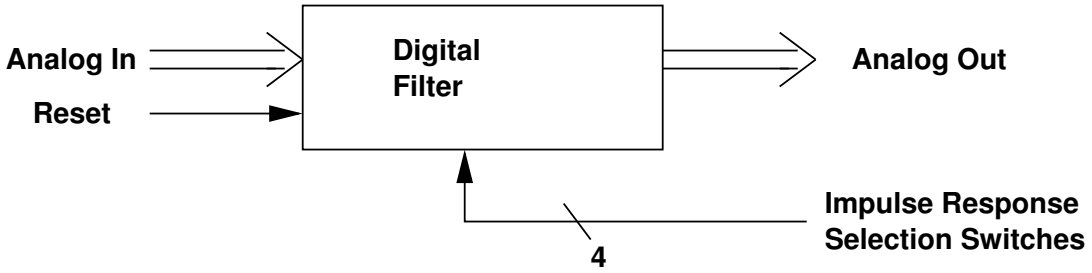

Figure 1: General Block Diagram

We expand Figure 1. to produce the block diagram of Figure 2. We do this simple expansion to isolate the analog to digital (A/D) and digital to analog (D/A) conversion from the complexities of the digital circuitry. As you will see later, we recommend that you use a single eight-bit data bus to both read the A/D and write to the D/A. Note that you could verify and demonstrate

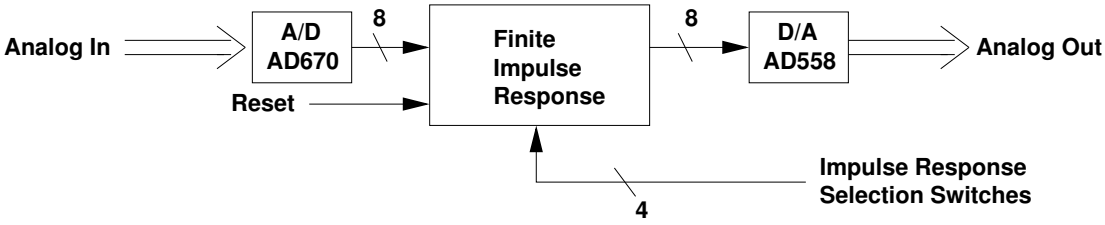

Figure 2: Analog/Digital Separation

the correct performance of your A/D and D/A by providing a rather simple fsm which provides the appropriate control signals. At your analog check-off be prepared to explain how your digital system verifies the timing of each analog operation and does not rely on the fact that a tristate bus holds its data due to its capacitance even when it is not being driven.

APPROACH

The basic idea is to use a ROM to hold the impulse responses and to use switches for the high order bits of the ROM address to select the actual impulse response that you use. An SRAM is used to store the present and previous sample values of the input analog signal.

The major sequence of operations is shown in Figure 3. First, initialize as required, then wait

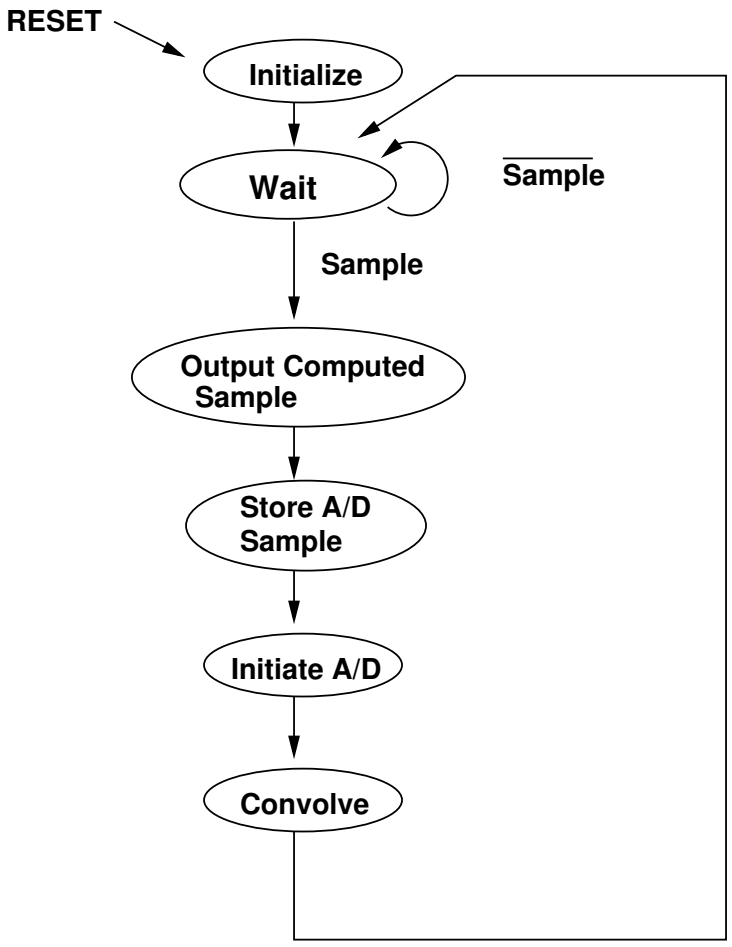

Figure 3: Sequence of Major Operations

until it is time for the next sample. Output the previously computed output signal sample to the D/A converter and store the result from the previous A/D conversion in the SRAM. Initiate an A/D conversion so that the conversion time is overlapped (or concurrent) with the rest of the processing. After that, do the arithmetic which implements the convolution filtering. Finally, loop back to wait unitl it is time for the next sample.

For simplicity, all impulse responses are of the same length, namely sixteen.

We will now consider the convolution arithmetic. Study Figure 4. Here we show a sample impulse response of length three. Samples of the input signal are shown and labeled where the zero

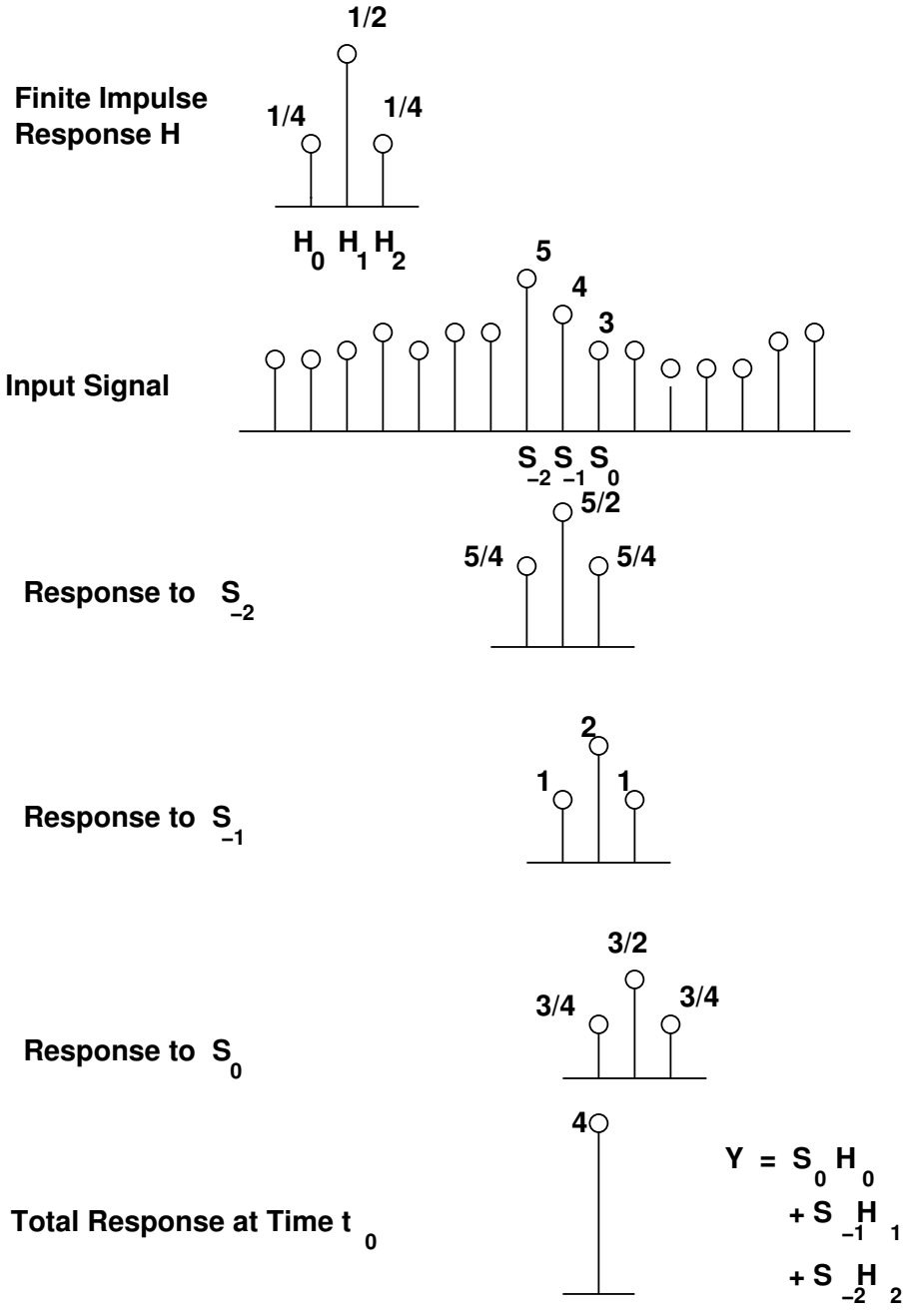

Figure 4: Convolution Arithmetic

subscript refers to the most recent sample. The -1 subscript refers to the sample before the most recent sample, etc.

We construct the output signal value by superposition. In Figure 4 we show the separate responses to input signal samples on three lines. These are simply copies of the impulse response as scaled by the input signal sample value. If we then sum up these individual responses vertically, we get the sample of the final output signal. The formula for that sum is shown at the bottom of Figure 4.

Thus, our task is to compute the sum of n products where n is the number of samples in the impulse response. In our case n=16. Each product term is the multiplication of successive impulse response samples with the corresponding past input signal samples. Once we have computed an output signal sample, then we go back and do the same operation all over again when we get another input signal sample.

Since we only have to use n of the input signal samples, it will be convenient to only use n RAM locations and overwrite older samples by simply letting the signal sample address wrap around assuming n is a power of two. How should the RAM address counter be implemented if n were not a power of two?

SYSTEM ORGANIZATION

A physical system block diagram is shown in Figure 5.

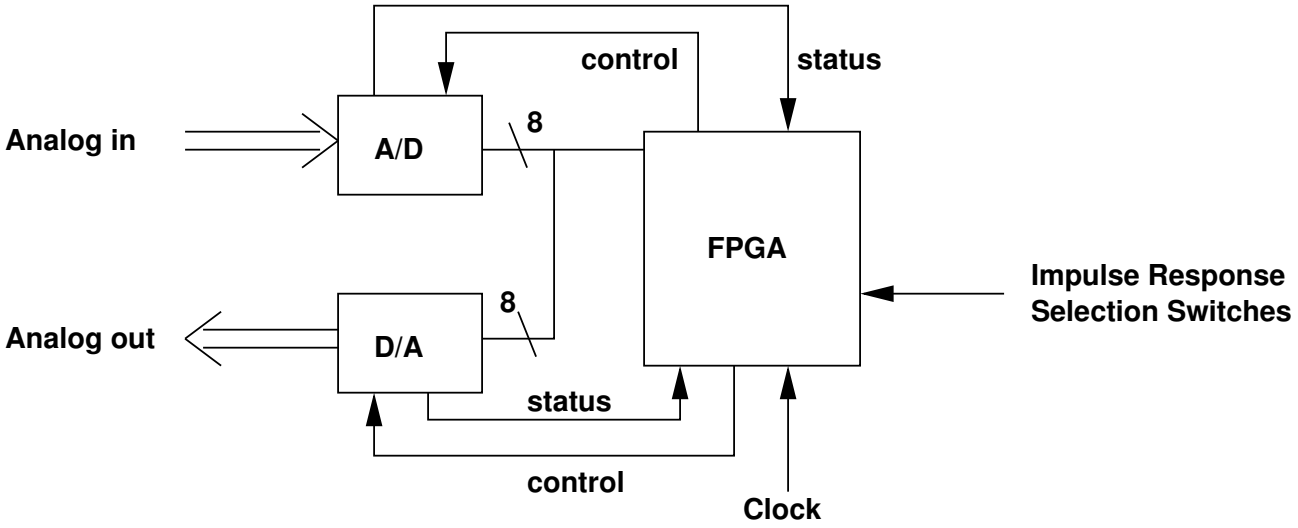

Figure 5: Physical System Block Diagram

While this is a helpful beginning to the generation of a detailed physical block diagram for wiring (or, in industry, printed circuit layout), it is not particularly helpful for the planning of circuitry to be implemented in the FPGA.

A logical system block diagram is shown in Figure 6.

The SAMPLE TIMER produces an output to indicate that another sample should be taken of

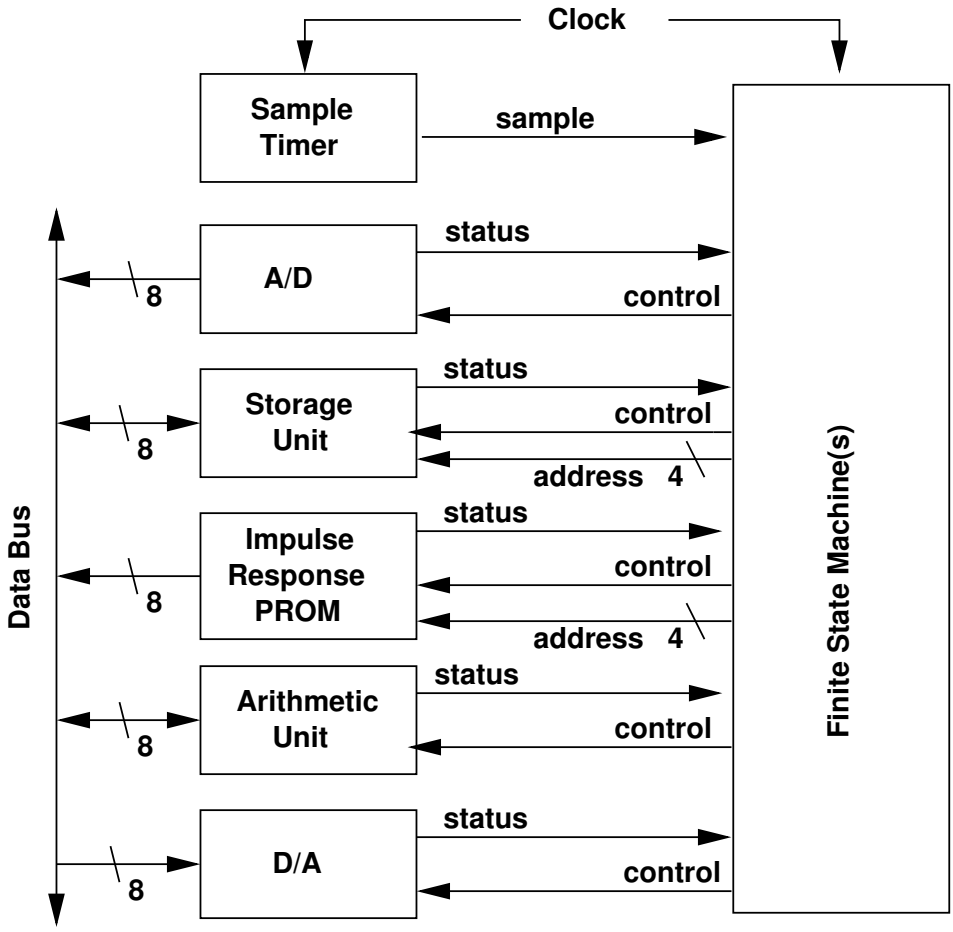

Figure 6: Logical System Block Diagram

the input signal and also that the most recently computed output signal value is to be output to the D/A.

The A/D converter is to be implemented by a single chip, AD670. You should wire it on the left hand proto strip of your kit which has special analog power supplies. You likely want to configure the AD670 for bipolar input. Pay particular attention to the control signal that specifies the output data format as offset binary or as two's complement. Note that switching between the two formats simply involves complementing the most significant bit. You may wish to use a different format for the initial analog demonstration and the actual digital filtering operation.

The D/A is also to be implemented by a single chip, AD558. It also should be wired on the left hand proto strip of your kit. This chip is a registered digital to analog converter which functions to convert the data bytes computed by the ARITHMETIC UNIT to an analog voltage. Note that there is only a single relationship between the binary input and the output voltage. What would you have to do if the number that represents the output is in two's complement? Note that the register is a latch, not an edge triggered register. Read the data sheet! Be careful and do not allow any glitches on either the /CS or /CE inputs. Other than the name, there is no difference between these two signals. The analog output signal can be viewed on an oscilloscope or it can be used to drive a speaker. When you are ready to listen to your filtered output signal, check out an amplified speaker from the Instrument Room and connect your D/A to the speaker through a capacitor (0.05 μF).

The STORAGE UNIT consists of a static, byte-wide RAM (SRAM) and an address counter. The implementation of this unit is rather straightforward. Use the library of parameterized modules provided by Altera as part of MAXPLUSII and choose a RAM similar to that which you used in Lab 2. If you really want to use separate components, then you may use an SRAM as in Lab 2. The STORAGE UNIT is used to store the digitized analog input signal as converted by the A/D. As input sample data are received, they are to be stored sequentially in a circular buffer implemented by a block of sixteen locations in the SRAM. Naturally you should use a counter to provide the SRAM addresses for storing new input signal samples and for accessing previous input samples. Should it be an UP counter, a DOWN counter, or both? What should be the initial state of the RAM address counter?

The IMPULSE RESPONSE ROM consists of a 256-byte ROM. The high order four bits of the ROM address select which impulse response is to be used. The low order four bits of the ROM address select which byte of the impulse response it to be accessed. The hex number 31 specifies the second byte of the third impulse response is accessed. If you use Altera's library of parameterized modules you can implement this rather easily. The file that you want can be copied from `/mit/6.111/handouts/labs/lab3.f2002/impulses.ntl` or you can reference that file directly if you always do setup 6.111 before running MAXPLUSII. The ROM output is shown as going to the data bus but it could go directly to the ARITHMETIC UNIT. Again, you can use a separate component if you wish; but, then, it is likely a good idea to connect the ROM outputs to the data bus and use the same address lines as for the SRAM. Naturally you should use a counter to provide the ROM addresses. Should it be an UP counter, a DOWN counter, or both? What should be the initial state of the ROM address counter?

The ARITHMETIC UNIT actually implements the convolution of the impulse response with the input signal. It then provides the digital data to the D/A which in turn produces the analog output signal.

A possible implementation of the data structure of the arithmetic section is shown in Figure 7. Beware - VHDL has a multiplication operator, *. What circuitry would be synthesized if we were to

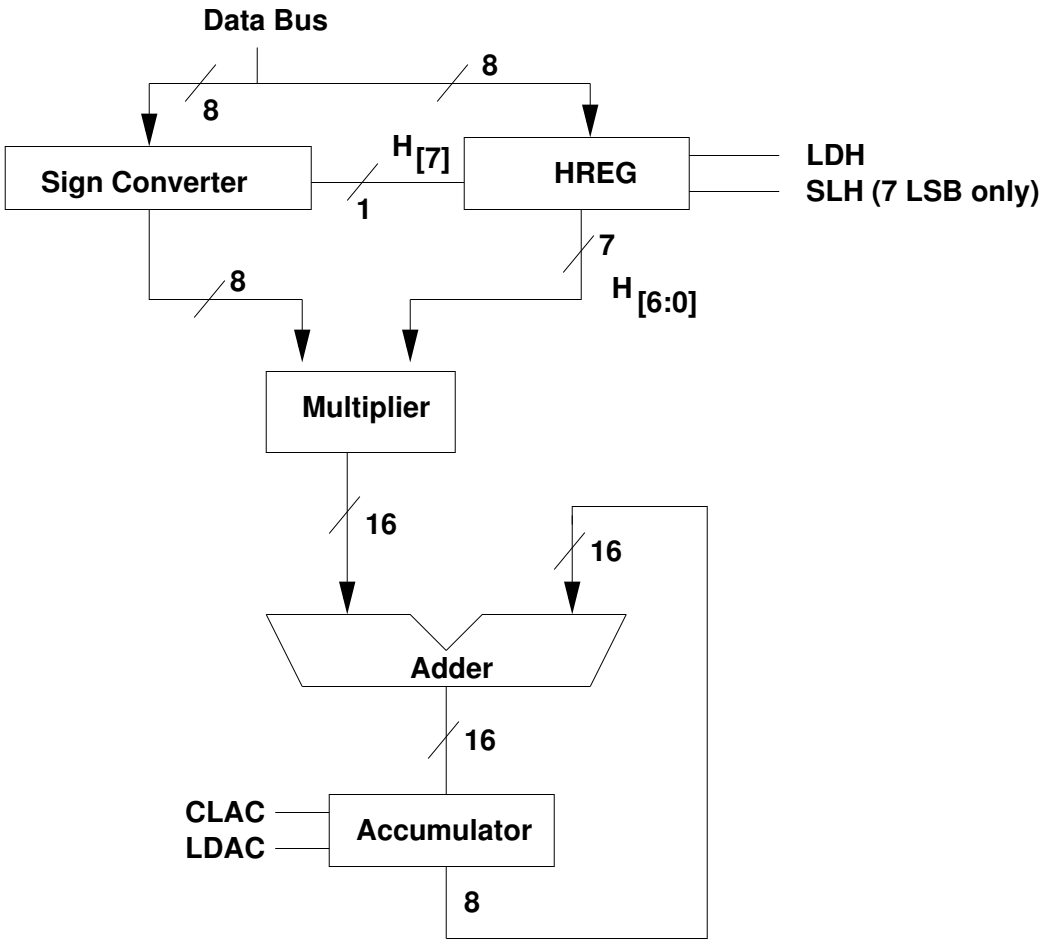

Figure 7: Logical Arithmetic Block Diagram

use it? We doubt that you want to spend the resources required for a combinational logic multiplier.

Since it is easy to implement multiplication with a shift and add technique, we can implement the ARITHMETIC UNIT data paths as shown in Figure 8. This is similar to the way we humans usually multiply two numbers (at least when we do so manually). Of course, we have to provide the appropriate control so that we implement the convolution arithmetic as shown in Figure 4.

A MATTER OF SIGNS

Please note that the impulse response stored in the ROM is in Sign/Magnitude format, so that the most signifigant bit contains the sign (if the sign bit is 1 the number is negative), and the other seven bits have the magnitude. You should wire the A/D converter to give two's complement. When H[7] is one it is necessary to turn the number from the A/D converter into its negative and you should convince yourself that the arrangement of the eight exclusive-ors and the sign bit carried in to the adder does this job. [You are NOT required to use this scheme. If you have a method you like better, go ahead and do it your way.]

Note also that the number that is generated by the convolution is in Two's complement form,

but the D/A converter can output only positive voltage. A good way of handling this is to convert this number to 'offset binary', so that the most negative output is a small number, zero is about in the middle of the range and the most positive number is at about the positive limit of the converter. It is possible to do this by simply inverting the MSB of the output number before sending it to the D/A converter (You should convince yourself this is right!).

CLOCK FREQUENCY

What clock frequency should you use? Using a shift and add multiplier takes seven cycles per multiply. There are sixteen multiplies per input sample. If you want a 20 KHz bandwidth the Nyquist rate is 40 KHz. This requires a clock faster than 7 * 16 * 40 KHz = 4.48 MHz. So the 20 MHz clock available on the kit is more than fast enough. Actually you will get decent quality sound if you use an 8 KHz sampling rate.

CONTROL UNIT

The CONTROL for this digital filter is quite complex. If you were to do it all in one state machine the FSM would be very complex. Any changes (We doubt anyone will get it just right the first time.) would require retesting of a lot of the design already tested. Also, having all the control in one FSM would make some of the desired testing difficult, if not impossible.

The sequence of major operations has already been suggested by Figure 3. You are to break up the control into meaningful minor FSMs operated by a major FSM. Carefully consider your testing strategy. Do not test your design implementation by looking at the output waveform with music as an input signal!

TROUBLESHOOTING HINTS

1. For test purposes, a reasonable approach is to initially provide the data inputs to your STORAGE UNIT by wiring the memory address counter outputs to the memory tristate I/O pins through a tristate buffer. This way, you can store a ramp into the memory and check out the STORAGE UNIT and its control.
2. Be careful with your memory timing. Remember lessons learned with Lab 2. It is not necessary to operate the SRAM as fast as possible.
3. Do NOT attempt to debug everything using only your logic probe! The logic probe is provided as a convenience for you when you cannot get to the lab. Your first instinct should be to look at signals with the scope or the logic analyzer.
4. Next, work on your ARITHMETIC UNIT. This section will probably cause you more trouble than any other (except for the CONTROL) as we have not made very detailed suggestions. Consider what testing and simulation will aid you in debugging and verifying this part. Consider providing some special input signal samples, e.g., an impulse train of isolated unit samples.
5. You can include several test FSMs (and even the "real" one) in your FPGA by using inputs to allow selection of the FSM to use.

LABORATORY REPORT REQUIREMENTS

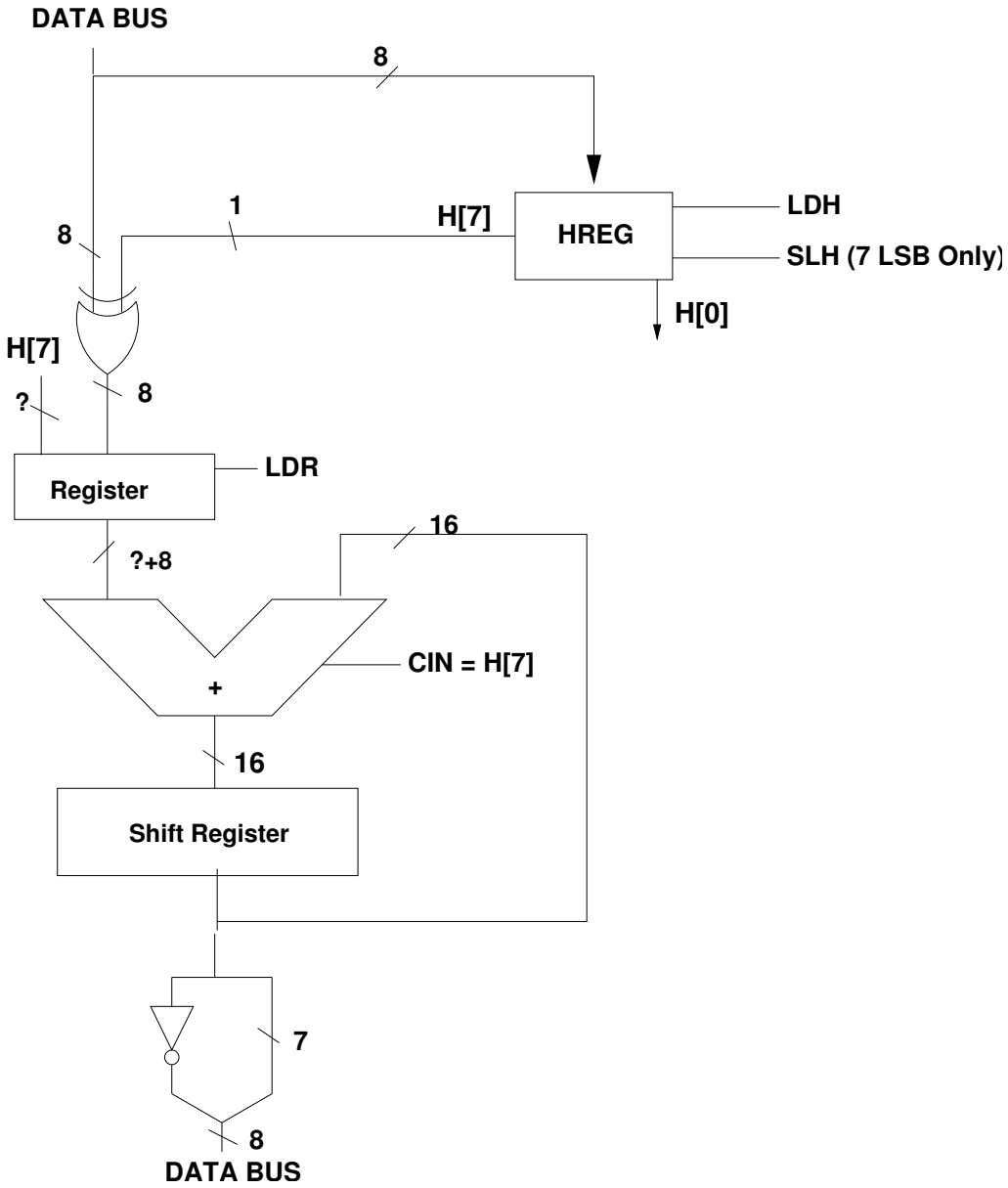

Figure 8: Expanded Arithmetic Block Diagram

After the completion of Lab 3, a typewritten (or neatly handwritten) report is required. It should emphasize both the theory of the design and the problems of practical implementation.

1. Introduction - a brief description of the problem and a block diagram
2. Information and description of your CONTROL FSMs
   (a) Define your major and minor FSMs.
   (b) Include a diagram for each FSM.
   (c) Describe the operation of each FSM in English.
   (d) Include your VHDL file(s) (with comments).
3. Detailed block diagrams of logic implemented in your FPGA
   (a) Try to make a reasonable compromise between legibility and detail. Include analog IC position designations and pin numbers. Where reasonable, draw the wire connecting two (or more) pins. Do not, however, run wires all over your diagram when this makes it hard to understand. Instead, label the pins with (unique) signal names.
   (b) You do not have to draw detailed equivalent circuits to describe the contents of FPGA(s). However, you should include a detailed block diagram showing major functional units and their width, e.g., a sixteen-bit adder. You must include your VHDL file(s). Each block should be accompanied by a paragraph describing the function of the FPGA circuitry. This, of course, could be in the form of comments in the VHDL file(s).
4. Timing diagrams for major signals - refer to these timing diagrams in your detailed descriptions.

The keys to a successful report are organization and clarity. A short paragraph with a diagram or table usually communicates your intent to the reader far better than a long-winded written explanation.

DESIGN REVIEW

You are not required to, but are STRONGLY encouraged to, meet with a T.A. for a design review before you program your FPGA. You do not have to have detailed VHDL code completed before the design review.

You should think about the problem and come for a design review with a block diagram, structure of your control, and a reasonably clear idea of the approach. Specifically, you should think about how you are going to test your circuits. A complicated design rarely works the first time you turn on the power. Consider how you might test individual modules in isolation. A little bit of well chosen extra test circuitry can actually reduce the time and effort required to get your design to work.

Since you have received the least guidance on the ARITHMETIC UNIT and Control, you should have fairly detailed information for these sections before the design review. Beware that an unoptimized design may not fit in an FPGA.

IMPLEMENTATION

After arriving at the final design, you should prepare detailed block diagrams for both the data paths and your control FSMs. Then write your VHDL code and simulate the modules to verify correctness. Finally, test the system. When operational, your system should be demonstrated to any staff member in the laboratory. Bring your detailed block diagrams and your VHDL code to the lab when coming for a demonstration as the T.A. will initial and date your diagrams. Be sure to include this witnessed copy in your report. After the demo, complete your report and turn it in.

Actually, there is nothing wrong with writing your report before the final check-off demonstration. Often it is a good idea to write the report while the debugging and testing are proceeding. This can not only provide a change of pace, but the thought going into the organization of the report can actually be beneficial in the debugging process. If you write your report before finishing the debugging, you will likely have to change the report only in a minor way, if at all, when your system is fully operational.

PROBLEMS?

If you encounter problems in your design, please come in and ask. We regard this lab exercise as a most important experience for you in preparation for the final project. We want you to succeed and will do our best to help. Please do not leave everything to the last minute as this will make it difficult to have time to help you.

HINTS

(1) Start early!!! This lab is larger in size and complexity than Lab 2. This lab will require a **FULL THREE WEEKS** of effort to be completed. Do not let it slide into the second week before you begin.

(2) Try to keep your wiring as compact as possible. Long wires introduce delays and are more susceptible to analog noise. This is especially important when dealing with your clock. Noise-corrupted clock signals account for many problems encountered.

(3) If you use more digital components than an FPGA, be careful with the loading of your clock. You may run into trouble if you try to drive every clock signal on your kit from a single output of a crystal oscillator. Sending the clock through several parallel buffers will help you avoid putting too much load on your clock signals.

(4) The filters that are provided to you are in sign and magnitude format. That is, the left hand bit is a sign bit, Logic 0 for a positive number and Logic 1 for a negative number. The right hand seven bits represent the magnitude of the impulse response sample. The maximum area under each impulse response is 127 and the largest that the sum of the impulse response coefficients reaches is 143. One has to look at the values of the impulse response samples to determine this, but it allows one to need only a sixteen-bit accumulator.

(5) The first four filters (0 through 3) are provided to help you in debugging your circuit. The first filter is simply a single impulse; therefore, the output signal from this filter should be the same as the input signal. The second filter is a negative impulse so its output should be the negative of the input signal. The third filter is a "box car" filter. The fourth filter is an "exponential" filter. By

inputting a square wave into these filters you should get outputs shown in Figure 9. The complete list of the filters is given in Table 1. The filter coefficients are provided in sign-magnitude HEX format.

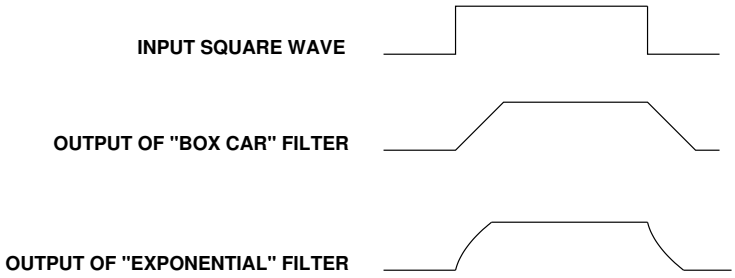

Figure 9: Outputs of "Box Car" and "Exponential" Filters

	Filter															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h[0]	7F	FF	08	20	00	83	00	00	81	03	06	00	88	86	08	00
h[1]	00	00	08	18	00	03	00	01	04	0B	03	8A	88	07	01	87
h[2]	00	00	08	12	82	04	00	00	00	04	8C	0A	00	0A	88	89
h[3]	00	00	08	0E	83	82	00	84	87	86	02	85	0D	8B	0F	1D
h[4]	00	00	08	0A	81	8A	00	05	07	91	10	87	11	87	8D	05
h[5]	00	00	08	08	09	82	00	0A	0B	8E	87	12	06	12	01	08
h[6]	00	00	08	06	18	18	00	A5	A4	03	92	94	8C	03	0E	03
h[7]	00	00	08	04	24	31	40	33	31	14	0E	08	63	63	59	45
h[8]	00	00	08	03	24	31	C0	A5	A4	14	0E	08	8C	03	0E	03
h[9]	00	00	08	02	18	18	00	0A	0B	03	92	94	06	12	01	08
h[10]	00	00	08	02	09	82	00	05	07	8E	87	12	11	87	8D	05
h[11]	00	00	08	01	81	8A	00	84	87	91	10	87	0D	8B	0F	1D
h[12]	00	00	08	01	83	82	00	00	00	86	02	85	00	0A	88	89
h[13]	00	00	08	01	82	04	00	01	04	04	8C	0A	88	07	01	87
h[14]	00	00	08	01	00	03	00	00	81	0B	03	8A	88	86	08	00
h[15]	00	00	08	00	00	83	00	00	00	03	00	00	00	00	06	00
Σh[i]	7F	FF	80	82	7E	7E	00	25	45	18	06	0C	73	7F	7D	7F
Max | Σ h[i]|	7F	7F	80	82	84	88	40	26	46	25	10	12	83	88	7D	87

Table 1: Filter Coefficients (columns, in HEX)