Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 - Introductory Digital Systems Laboratory

Laboratory 3 Check Off Sheet

Student Name:

TA Signature/Date:

Part 1: Analog Interface

You must show a TA the following at the beginning of the analog check off

- State transition diagram of your FSM(s). ☐
- Verilog code. ☐
- Timing diagrams for your D/A and A/D. ☐

Be Able to Demonstrate Your Working Lab ☐

- You will be asked to demonstrate the correct performance of your A/D and D/A by simply reconstructing the signal sampled by your A/D at the output of your D/A.

Be Able to Respond to any of the Following Questions (and possibly others) ☐

- What are the critical timing constraints of your A/D and D/A system?
- Explain the operation and importance of your tri-state bus.
- Can you have glitches on the control inputs $\overline{CS}$, $\overline{CE}$, R/$\overline{W}$ for the A/D?
- Can you have glitches on the control inputs $\overline{CS}$, $\overline{CE}$ for the D/A?

Student Name:

TA Signature/Date:

Part 2: Complete Digital Filter (including analog blocks)

You must show a TA the following at the beginning of the analog check off

- Block diagram for your system. ☐
- State transition diagrams for your major-minor FSM structure. ☐
- Verilog code printout. ☐

Be Able to Demonstrate Your Working Lab

- From reset, demonstrate that you can filter an input signal with any of the 4 impulse responses provided. Two simple tests: using a 600Hz square wave as the input, try the all-pass filter (output looks like input) and boxcar (output is a triangle wave that saturates after 16 steps). ☐

Be Able to Respond to any of the Following Questions (and possibly others) ☐

- Describe the critical timing issues of your system.
- What are the possible problem(s) with the shared tristate bus structure?
- For the clock frequency you used, what is the fastest sampling rate possible?
- Can you suggest ways to improve the throughput of the system?

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 - Introductory Digital Systems Laboratory

Laboratory 3 - Finite Impulse Response Filter

Analog Checkoff: October 22, 2004
Final Checkoff and Report Due: October 29, 2004

1. Introduction

You will design a 16-tap Finite Impulse Response Filter (FIR) in this lab, suitable for filtering inputs from a signal generator or audio signals from a music player or microphone. Like the previous labs, you will need to design finite state machines to control your subsystems. Additionally, you will need to instantiate some modules and build controllers to interface an analog to digital converter and a digital to analog converter to your FPGA.

2. Procedure

This lab consists of three parts. The first part is the design phase, which is very similar to the previous labs. You should read through the lab and plan your design. It will be helpful to schedule a design conference with your TA or the professor to help with your design.

The second part of this lab is the Analog Checkoff. You are asked to connect the A/D converter, FPGA, and D/A together to sample an analog signal, and reconstruct this signal at the output of the D/A converter.

The third part of the lab is to expand on the analog part and implement the convolution to complete the FIR filter. You will need to instantiate your custom parallel multiplier, an accumulator, a ROM, a RAM and and additional FSMs to control and coordinate these modules. Your design structure should be such that a top-level FSM controls other minor FSMs, which in turn control different blocks in your system. You are free to choose what modules your minor FSMs will control, for example, you might choose to control the multiplier and the accumulator by one FSM, memory units by another FSM and analog interfaces with other FSMs (for this design, it is probably just as easy to implement all the necessary logic in a single FSM, but we would like you to go through the process of modular design using the Major/Minor FSM approach).

You will be required to turn in a detailed report for this lab.

3. System Description

Your task is to build a machine which will accept analog signals and produce a filtered version of the input signal. We will choose the FIR (Finite Impulse Response) convolution approach for digital filtering, which is widely used by filter designers. Digital Signal Processors (DSP) are well suited to perform multiplication and addition operations and the FIR filter is a standard signal processing benchmark.

An overall block diagram is shown in Figure 1.

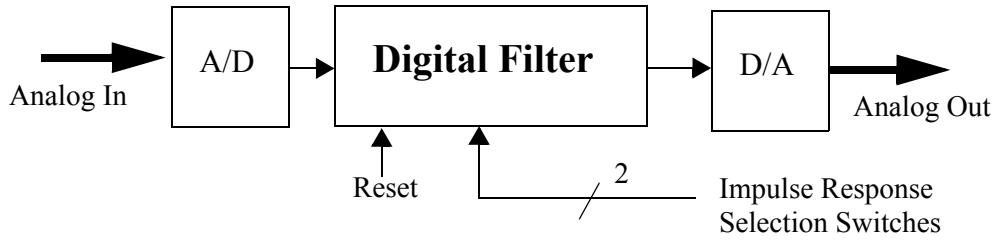

Figure 1: Overall block diagram.

The basic idea is to sample analog signals using an Analog-to-Digital Converter (A/D) and store them in a memory element, perform the filtering operation in an arithmetic operation unit, and send the result to a Digital-to-Analog Converter (D/A). The major sequence of operations are shown in Figure 2.

First, initialize the system as required, then wait until the next sample (sample is a one cycle pulse every sampling period). Output the previously computed output signal sample to the D/A converter and store the result from the previous A/D conversion in the memory (e.g. SRAM). Initiate an A/D conversion so that the conversion time is overlapped (or concurrent) with the rest of the processing (why might this be important?). After that, do the arithmetic which implements the convolution filtering. Finally, loop back and wait until it is time for the next sample.

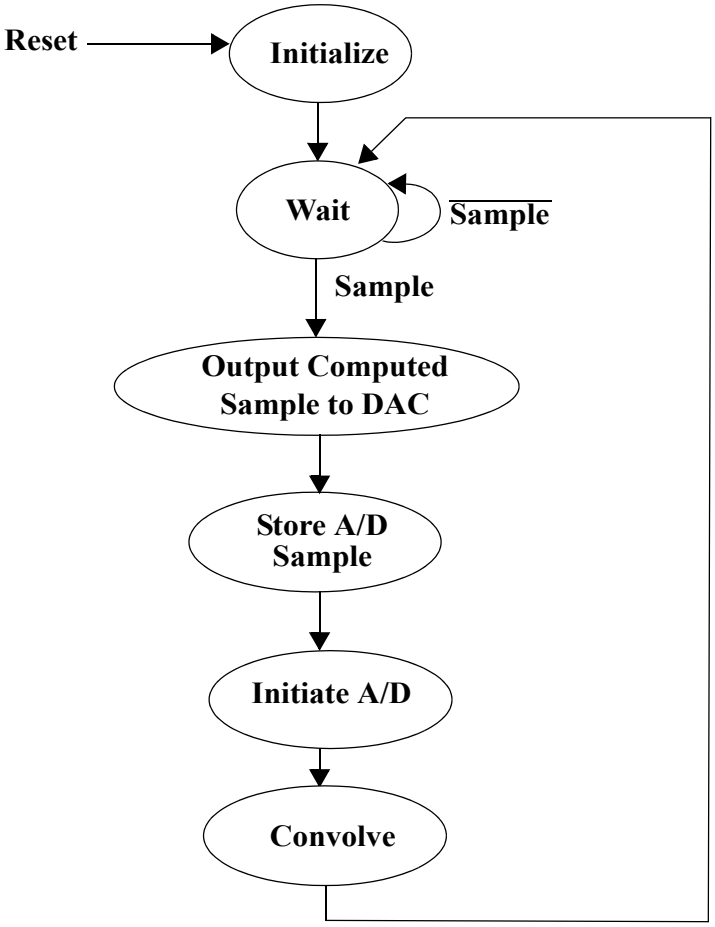

Figure 2: Overall control flow.

Note: If you are not familiar with convolution, there is problem in the 6.111 Spring 2004 Problem Set 3 that introduces the basics of discrete convolution. We will provide you with the mathematical formulas you will need for the purposes of this lab, however it will be useful to get a deeper understanding of convolution. You might also refer to a standard text book on digital signal processing.

4. System Organization

A logical system block diagram is shown in Figure 3. You are required to implement all the modules of your digital filter in your FPGA, and you are also **required to use a shared-bus between your FPGA, D/A and A/D**. As mentioned earlier, you are required to use the major-minor FSM structure to control your system in this assignment. The partitioning between these FSMs is a design choice you will have to make. There is a trade-off between modularity, system performance and complexity. You must be ready to justify the way you introduced modularity in your controller in your lab report.

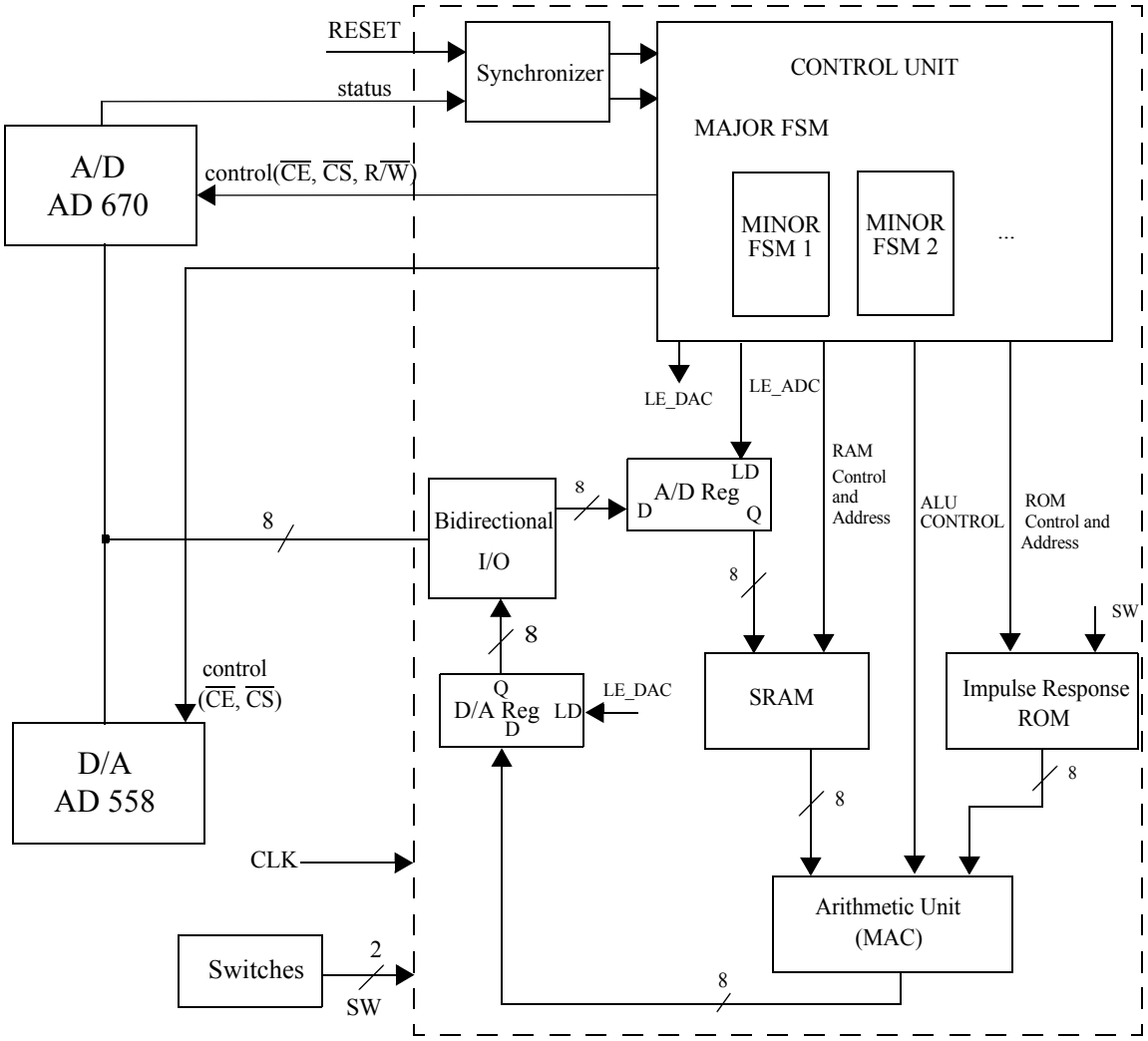

Figure 3: Detailed block diagram.

5. Analog Blocks and Checkoff

The analog checkoff requires using an A/D to sample an analog input at a sampling rate of 20 kHz and reconstruct this signal at the output of the D/A (i.e., a loopback through the FPGA). You will need to design an FSM to control the A/D and the D/A. **You are required to use a shared external bus for the data inputs of your D/A and the outputs of the A/D.** You must make sure that only one device is driving the bus at all times. You should structure the FSM in a major/minor approach so it can be easily extended to part 2.

A/D converter

The A/D converter is to be implemented by a single chip, the AD670. It serves to take samples of the analog signal from the microphone or the signal generator. You should wire it on the left hand proto strip of your kit which has special analog power supplies. You probably should configure the AD670 for bipolar output and twos complement format. Pay particular attention to the control signal that specifies the output data format as offset binary or as twos complement. Can you have glitches on the control inputs $\overline{CS}$, $\overline{CE}$, R/$\overline{W}$?

D/A converter

The D/A converter is also to be implemented by a single chip, the AD558. It also should be wired on the left hand proto strip of your kit. This chip is a digital to analog converter which functions to convert the data bytes computed by the arithmetic unit to an analog voltage. The $\overline{CS}$ or $\overline{CE}$ inputs should be glitch-free. 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 (0.05μF).

Bidirectional Output, A/D and D/A Registers

You will need to make sure that you can read from and write to the 8 bit bus that connects your FPGA to your A/D and D/A. You must not drive these pins from your FPGA when AD670 is driving them, and additionally you should tristate the outputs from the AD670 when you drive the bus from the FPGA. D/A and A/D registers will be controlled by your A/D and D/A FSM, and will provide data to the appropriate data paths.

System Clock

The A/D sampling rate is specified to be 20kHz (i.e., the rate at which we acquire data from the data converter). You have the flexibility to choose the internal clock rate for the FPGA. It can be the crystal oscillator clock frequency of 1.8432MHz or a 10MHz reference.

6. Major Finite State Machine

The finite state machine for the FIR filter should put together the different parts of the lab. This FSM therefore manages the communications between the FPGA, D/A and A/D, and it also carries out the convolution by controlling the two memory blocks, the multiplier, and the accumulator. Your major FSM should be a very simple FSM, controlling all the minor FSMs in your design.

When designing your minor FSM, that carries out convolution, remember to reset your accumulator after 16 points. The convolution is simply a sum of products; therefore, the accumulator should be reset at the beginning of the convolution, and each product should be accumulated. After accumulating 16 points, the value in the accumulator should be the resulting value of the convolution, ready to be sent to the AD558.

All of your minor FSMs should run off the same system clock as your major FSM, and should be reset to their idle states when the global reset signal is asserted. In designing your state machine(s), pay attention to the timing constraints for each of your modules and introduce appropriate delays as necessary.

7. FIR Filter

Convolution Operation

A FIR filter uses convolutions to generate the data points for the filtered output. Convolutions are a weighted accumulation of sampled points from a signal. In this case, there are 16 points in the convolution. Given that h(n) is the Finite Impulse Response for a specific filter and x(n-k) is the k^{th} recent sample, the equation that describes the convolution that we are going to implement is:

$$Y(n) = \sum_{k=0}^{15} h(k) \cdot x(n-k)$$

To carry out convolution, we only need multiplication, addition, and some addressing logic to carry out the shifting operation.

Functional Block Diagram

Figure 3 shows the functional block diagram for the FIR filter. The entire implementation is within the FPGA except for the analog part. The next few sections describe the details for each block. The ROM is the memory element that stores the filter coefficients and the RAM is the memory element that stores sampled values. The multiplier and the accumulator are the two blocks that actually carry out the computations for the convolution, and the FSM is the block that sequentially implements the convolution.

Parallel Multiplier

A parallel multiplier can be implemented in a variety of ways. Note that the Verilog "*" operator implements unsigned multiplication, so you need another plan for implementing the required multiplication: follow the implementation outlined lecture or use Altera's MegaWizard Plug-In Manager (arithmetic => LPM_MULT). **You must pay particular attention to the number formats at the inputs and outputs of your multiplier.** The filter coefficients are in sign-magnitude format (one bit for the sign and 7 bits for the magnitude), whereas the sampled values coming from A/D should be in twos complement. You must make sure to feed a twos complement number into the accumulator, therefore you might need to carry out appropriate conversions at the inputs and/or outputs of the multiplier. Figure 4 shows a possible interface between your multiplier and the accumulator.

Accumulator

To implement the convolution, you will need to accumulate 16 products. Draw a block diagram for an accumulator using components such as adders and registers. Be sure to have the input bitwidth match the bitwidth of the output of the multiplier you will be instantiating for the lab, and the output of the accumulator width be as wide as necessary to accommodate for all 16 products.

The "gain" of the filter coefficients will determine which 8 bits of the accumulator should be sent to the DAC. One way to put off the decision is to use a 8-bit 4:1 "range" mux to select between 4 different ranges, e.g., for an N-bit accumulator the choices might be bits [N-1:N-8], [N-2:N-9], [N-3:N-10] and [N-4:N-11]. Look at the coefficients in impluse.mif or simply run some experiments to determine the correct choices. The range mux can be controlled by two additional switches.

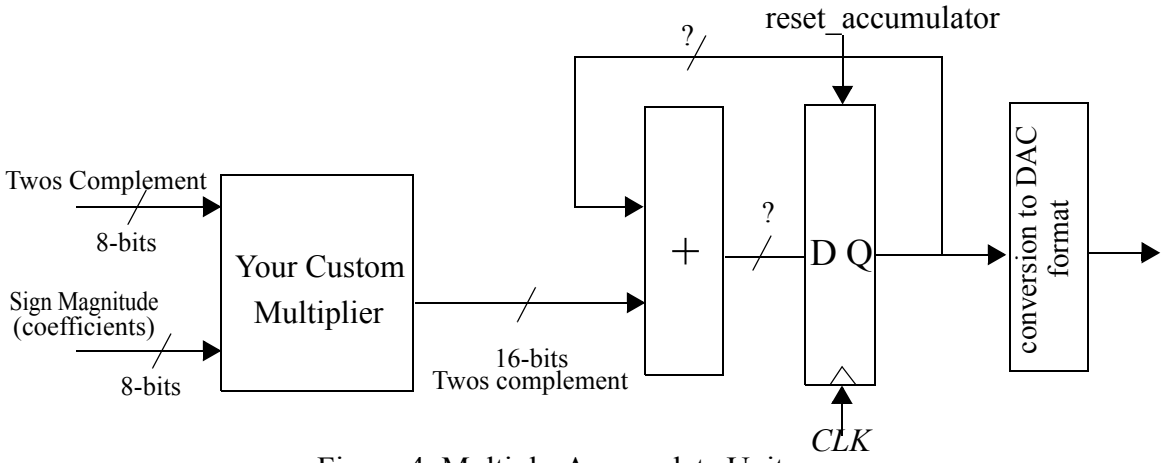

Figure 4: Multiply-Accumulate Unit.

Global Reset Signal

You must provide a way to reset the state of your digital filter. Your digital filter should not produce any output to the D/A or samples from the A/D as long as the reset signal is asserted. The reset signal should be synchronized.

Storage Unit (SRAM)

This unit consists of a static, byte-wide RAM and an address counter. You will need to use the library of parametrized modules provided by the Altera MegaWizard Plug-In Manager (memories => LPM_RAM_DQ). Make sure to choose an appropriate size to be able to store 16, 8-bit data points. 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 samples. You can use a separate data bus for the input data port and output data port. **You should not use a simple delay line (i.e., back to back registers) to implement the memory and must use the SRAM macro.**

ROM for Impulse Responses

You will need to use the MegaWizard to generate a Single Port Block Memory for the ROM (memories => LPM_ROM) to store four different impulse responses. These four impulse responses are the All Pass, High-Pass, Box Car, and Exponential filters (see Table 1 below). When implementing the memory block, you should use the file *impulses.mif* (/mit/6.111/www/f2004/handouts/impulses.mif) to load the ROM with the appropriate values.

The file *impulses.mif* has a certain addressing pattern for easy access to any particular datapoint for the four impulse responses. The memory should be addressed using a 6-bit internal address bus. The top two address bits will be connected to two switches and will select the appropriate impulse responses. The bottom four bits are the offset for each impulse response.

The top two address bits of your ROM should be connected to two switches. Using these switches, the user will specify which digital filter to use.

Table 1: Switches

00	01	10	11
All pass	High pass	Box car	Exponential

Therefore, 0x33 will select the fourth data point for the exponential (fourth) impulse response.

8. Laboratory Report Requirements

We require that you turn in a brief but detailed report for this lab. Your report should emphasize both the theory of the design and the problems of practical implementation.

Cover Page/Abstract

Include a cover page with a title, your name, the name of your TA, the course name, and the date.

Introduction

Give a brief description of the problem and a block diagram of your system.

Module Description/Implementation

(a) Describe your control FSMs

- Define your major and minor FSMs.
- Include a state transition diagram for each FSM.
- Describe (in words) the operation of each FSM.
- Include your commented Verilog code.

(b) Detailed block diagrams of logic implemented in your FPGA

- Try to make a reasonable compromise between legibility and detail. You do not have to draw detailed equivalent circuits to describe the contents of the FPGA. You should include a detailed block diagram showing major functional units with annotated widths. Each block should be accompanied by a paragraph describing the function of the FPGA circuitry.

(c) Timing diagrams for major signals - refer to these timing diagrams in your detailed descriptions.

Testing/Debugging

Describe how you tested your digital system. Provide us with a description of the design methodology you used in the creation of your digital system, and how you planned on testing each block in the design stage and how you actually ended up testing it. Be sure to include specific details.

Conclusion

We are particularly interested in hearing about what you learned from the completion of this lab and what you think are the important concepts to take away from the design of this digital system.