University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS150
Fall 2001

Jason Hu
Mark Feng
Mike Lowey

Checkpoint 4

Overview:

The purpose of checkpoint 4 is to test your skills as a logic designer. The specification for this checkpoint is very simple, and there are few constraints, so it is up to your imagination and creativity to finish this project. Given the correct design, checkpoint 4 can be very simple to implement, otherwise it can be a nightmare. This checkpoint is 25% of your final project grade, so start early. There is no CLB limit for this checkpoint, but it does need to fit on the chip with your working checkpoint two.

Specification:

Part 1: Tone Control

In checkpoint 4 you will implement FIR filters to filter the digital signals put out by checkpoint 3. You are expected to implement a high-pass and a low-pass filter in Xilinx. The entire design will be in done in schematic capture, so you do not have to wire wrap or add any additional hardware for this checkpoint. If your project is implemented correctly, then you should be able to hear changes in the bass or treble of the music you are playing when you switch on the high-pass or low-pass filter respectively.

You are given the following formulas:
You do not have to understand them, just know that they work.

Low pass filter:
Output[n] = ¼ × Input[n] + ½ × Input[n-1] + ¼ × Input[n-2]

High pass filter:
Output[n] = -¼ × Input[n] + ½ × Input[n-1] - ¼ × Input[n-2]

Input[n-1] means the previous input sample from one time unit ago. For example, to get one sample of a low pass filtered signal, take the current input sample divided by four, add the previous input sample divided by two, and then add the sample from two samples ago divided by four.

Your input samples are 16-bit samples coming in at 11 kHz, and your output samples are 16-bit samples leaving at 11 kHz. The CD-ROM sends samples at 44 kHz, so to get the sampling rate down to 11 kHz, drop 3 out of every 4 input samples. This means that for every four samples the CD-ROM sends out, keep only one of them to use for the filter. Dropping the samples makes the high pass and low pass filter effects more pronounced and keeps the filter simple.

If you want an example of what a high pass and low pass filter sound like, try playing with the treble and bass controls on your speakers or in your audio program such as Winamp. The low pass filter should sound like the bass turned up and treble turned down, while the high pass filter should sound like the treble turned up and the bass turned down.

Just as in checkpoint 3, you need to keep the left and right channel samples separate. Right channel samples should only be used with other right channel sample for the filter, and the same should be done for the left channel samples.

The filter formulas should really look like this:

For left channel:
Low pass filter:
OutputL[n] = ¼ × InputL[n] + ½ × InputL[n-1] + ¼ × InputL[n-2]

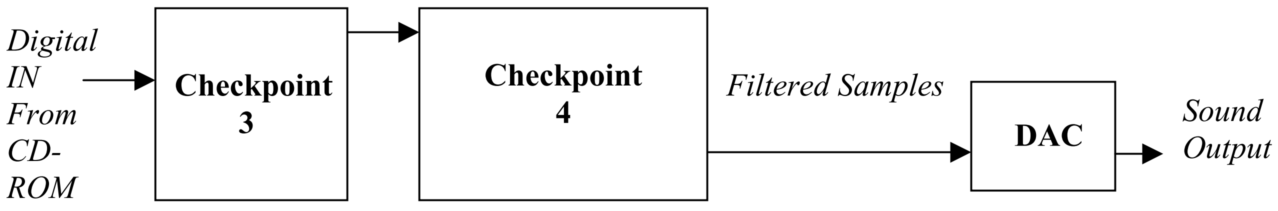
High pass filter:
OutputL[n] = -¼ × InputL[n] + ½ × InputL[n-1] - ¼ × InputL[n-2]

For right channel:
Low pass filter:
OutputR[n] = ¼ × InputR[n] + ½ × InputR[n-1] + ¼ × InputR[n-2]

High pass filter:
OutputR[n] = -¼ × InputR[n] + ½ × InputR[n-1] - ¼ × InputR[n-2]

Just like checkpoint 3, you should assert LL signal when OutputL is ready, and LR signal when OutputR is ready. You should output to the DAC every 11 kHz.

Here's a system diagram of where checkpoint 4 falls:



Requirements:

The following are restrictions we will impose to test your creativity and logic design skills, but mostly importantly to save you CLBs. Be very conservative in your CLB usage.

1. In order to store your music samples, you are required to use a RAM. Try to pick out the smallest possible RAM for your design. It is important that you do not store your samples via 5 or 6 16-bit FFs. (You may not meet the CLB count) You may, however, use one 16bit FF to latch the RAM output.
2. To compute your output samples, you are only allowed one ADDER unit. Figure out how you are suppose to do subtract. You may not use a 16-bit FF to store your temporary sum results. The most obvious solution is to write your computed results back to the RAM and perform a RAM read operation later on, to fetch the previous summation and added it to a new incoming sample. Since you are only allowed one RAM for your entire design, consider partitioning it. For example, you may consider store left-channel samples and summations in the upper half of the RAM and right-channel samples and summations in the lower half.
3. Your design is required to be able to switch between NORMAL mode, HIGHPASS mode, and LOWPASS mode for each channel separately. To do this you should have four buttons:
   a. HP-Left
   b. LP-Left
   c. HP-Right
   d. LP-Right

   These buttons should each toggle on and off their respective functions independently. It will save you some work to notice that implementing LP and HP on the same channel amounts to not changing that channel at all.
4. To help foster logical organization, we expect you to group all of your checkpoint 4 control signals in a control block and keep all your datapath separate from your control signals.

How we designed the filters:

You don't need to understand how we designed the filters, but for those of you who are interested, here's the explanation. This was done using what was learned in EE120 and EE123.

Highpass filter:

$$H[w] = -\tfrac{1}{2} \times \cos(2\pi w) + \tfrac{1}{2}$$
$$= -\tfrac{1}{4}\left(e^{-j2\pi w} + e^{j2\pi w}\right) + \tfrac{1}{2}$$

Inverse Discrete Fourier Transform of H[w] = h[n]

$H[n] = -\tfrac{1}{4} \times \delta[n-1] + \tfrac{1}{2} \times \delta[n] - \tfrac{1}{4} \times \delta[n+1]$

$\text{Output}[n] = -\tfrac{1}{4} \times \text{Input}[n] + \tfrac{1}{2} \times \text{Input}[n-1] - \tfrac{1}{4} \times \text{Input}[n-2]$

Lowpass filter:

$$H[w] = \tfrac{1}{2} \times \cos(2\pi w) + \tfrac{1}{2}$$
$$= \tfrac{1}{4}(e^{-j2\pi w} + e^{j2\pi w}) + \tfrac{1}{2}$$

Inverse Discrete Fourier Transform of $H[w] = h[n]$

$H[n] = \tfrac{1}{4} \times \delta[n-1] + \tfrac{1}{2} \times \delta[n] + \tfrac{1}{4} \times \delta[n+1]$

$\text{Output}[n] = \tfrac{1}{4} \times \text{Input}[n] + \tfrac{1}{2} \times \text{Input}[n-1] + \tfrac{1}{4} \times \text{Input}[n-2]$

Part 2: Volume and Balance Control

For this section you will implement separate volume controls for each channel. The goal will be to allow each channel to step through six volume settings: zero volume (mute), 1/5 volume, 2/5, 3/5, 4/5, and 5/5 (full volume). To change volume we need to scale the amplitude samples before they are fed in to the tone control or the DAC. Because our samples already span the full 16-bits of their range, we will only be reducing the volume from the levels that we extract from the CD-ROM drive's digital output. To change the volume we cannot just subtract from the values, we must divide them all by the same constant to keep the amplitude relationships in the music intact. We will be approximating the volume settings by dividing the sample by powers of two and then summing combinations of the various quotients.

For example:
Starting with a full volume sample V
4/5 volume can be approximated as $0.8125V = V/2 + V/2^2 + V/2^4$
$= .5V + .25V + .0625V$

The approximation should be as close as you can get by adding any three of the quotients down to $V/2^5$. The volume control interface should be four buttons:

- Left Volume +
- Right Volume +
- Left Volume –
- Right Volume –

The volume control should not change anything if it is asked to lower the volume below 0/5 or raise it above 5/5.

The volume control should be inserted before the tone controls. It should be able to operate at the same time as the tone controls.

The volume control should utilize the parts from the part one. You can increase the size of your RAM and add a shift register, but you should not need to add any other major new components to implement the volume control. The volume feature should mainly require an additional level of control of the tone-control resources.

Name________________________________   Name___________________________

Project Checkpoint 4

Check Off Sheet

We will be looking at the schematics during check off to make sure that the requirements have been met.

Either tone control **or** volume control
work correctly ____________**(20%)**

Both tone control **and** volume control
work correctly ___________**(100%)**

Extra Credit !!!

Implement reverb (echo) by delaying and attenuating samples and then adding them to the present sample. You should compensate for the possibility that the present sample could be all ones by attenuating it just enough to add the attenuated signal without overflow. It must sound like an echo to get points! Reverb should have no effect when it is not turned on. A single button is enough to turn it on and off for both channels. This will be tricky because it will require storing and manipulating a lot of data.

Reverb ___________**(+10%)**