

## Checkpoint 3

### Overview

The purpose of checkpoint 3 is to create a logic block that will translate the digital audio output of a CD-ROM drive into another format that a dual 16-bit Audio DAC can recognize. The signal from the DAC is then amplified with an amplifier and is then outputted to a headphone jack. You can then attach a pair of speakers to the headphone jack and listen to cd music. **Note:** Do not attach headphones directly to the board. The volume is **extremely loud!**

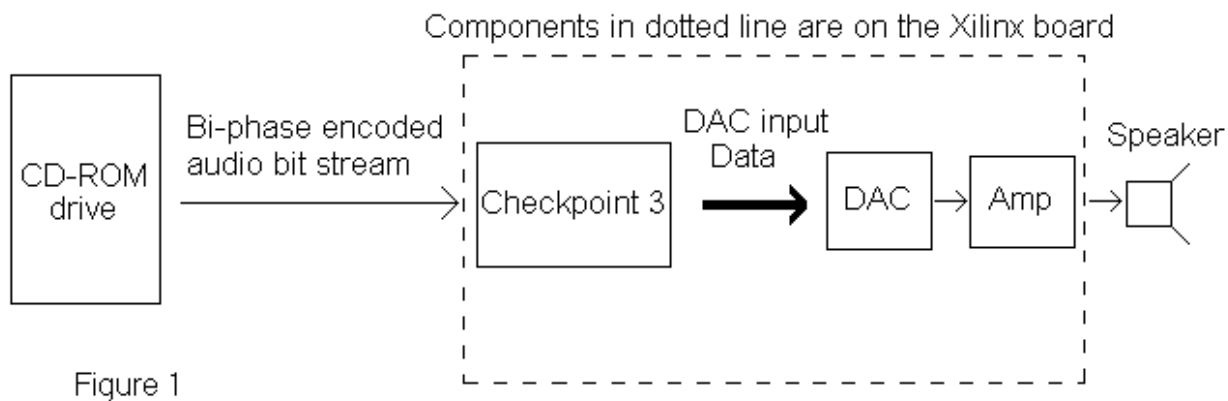


Figure 1

Figure 1 shows how checkpoint 3 is physically related to the CD-ROM drive and the rest of the Xilinx board. Additional components must be wire-wrapped to your board, a DAC, amplifier, and discrete packs containing resistors and capacitors.

## Inputs and outputs

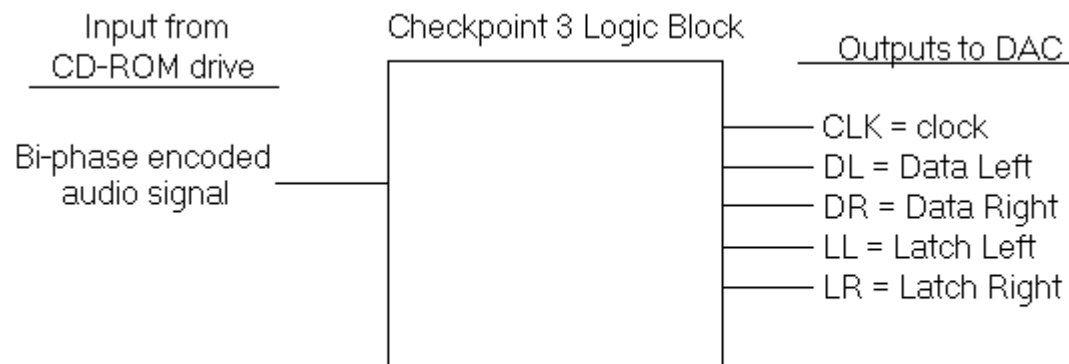


Figure 2 - Graphical representation of inputs and outputs

Figure 2 shows the inputs and outputs of the checkpoint2 logic block. There is one input, the digital audio output from the CD-ROM drive. There are 5 output signals, clock(CLK), Data Left(DL), Data Right(DR), Latch Left(LL), and Latch Right(LR). **Note:** We will also use a 16MHz clock not shown in the figure.

## Inputs

### Bi-phase encoding

Bi-phase encoding is a method for encoding data. The basic building is a cell. Each data bit is composed of two cells. A 1 data bit is made up of two different cells, either a 1 and a 0, or a 0 and a 1. A 0 data bit is composed of two of the same cells, either two 1's or two 0's.

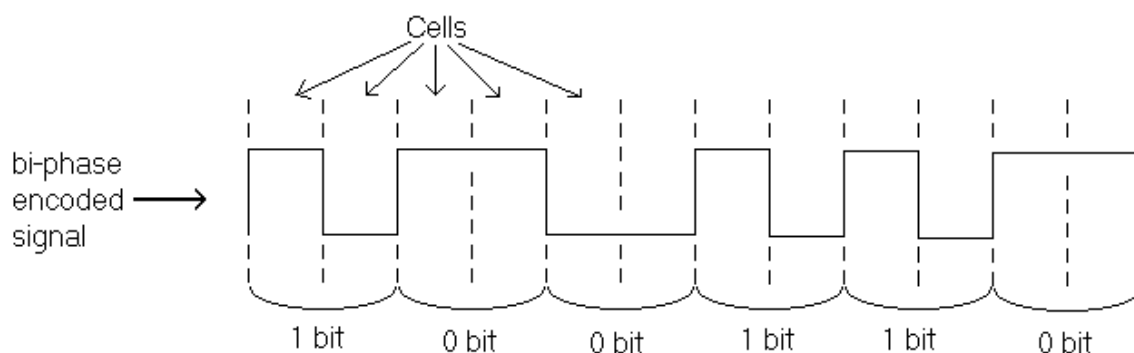


Figure 3

Figure 3 shows a section of a bi-phase encoded signal. This section is composed of 12 cells in the sequence "101100101011" which represents 6 data bits, "100110".

**Note:** No more than two consecutive cells can have the same polarity, (except preambles: please continue reading)

## Frames and Preambles

How do we extract the audio data from the bi-phase encoded bit stream? The bi-phase encoded stream is composed of 32-bit frames. Each frame contains one 16-bit audio sample. Each frame begins with a preamble. There are three types of preambles.

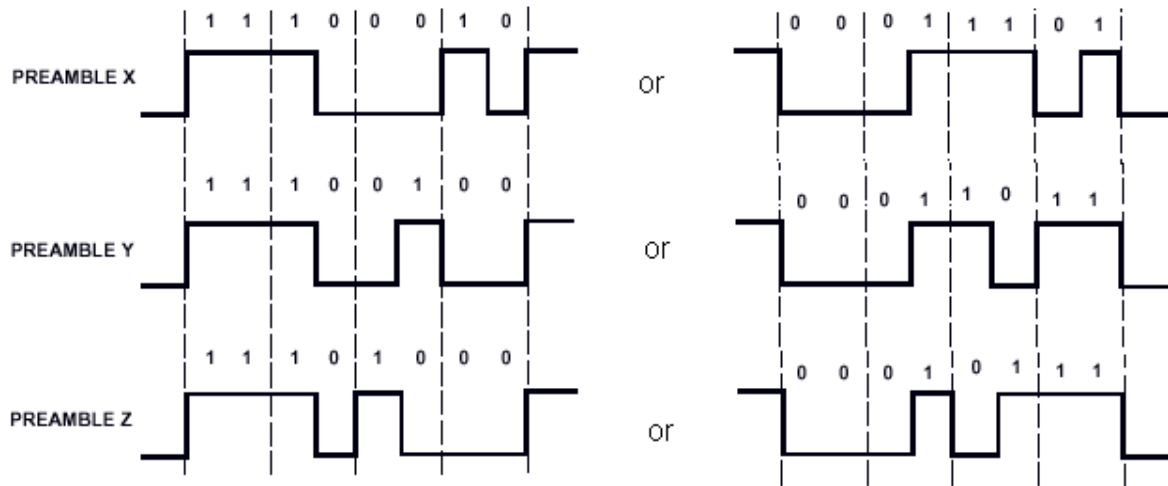


Figure 4

Figure 4 shows the three different Preambles. Please note that:

- Each Preamble is made up of 8 cells. (not data bits!)
- Notice that each Preamble has two representations. They are inverses of each other. This means that the Preamble X can be either “11100010” or “00011101”. (← These are **cells**)
- Preambles are distinguished from regular data as they contain 3 consecutive cells of the same polarity. Preambles begin with a “111” or “000” sequence of cells.
- Preambles begin with “111” if the previous cell was a “0” and begin with “000” if the previous cell was a “1”.

### What do these different preambles mean?

Remember that each frame starts with a preamble and contains one 16-bit audio sample. The preamble tells you which channel the sample is for, either left or right. (We have stereo audio so we have both left and right channels)

	BIPHASE PATTERNS	CHANNEL
X	11100010 OR 00011101	LEFT
Y	11100100 OR 00011011	RIGHT
Z	11101000 OR 00010111	LEFT

Figure 5

From figure 5, we see that preamble X and Z are used to denote the left channel, and preamble Y denotes the right channel.

Note: You should still distinguish between preamble X and Preamble Z. We may use preamble X for future uses. For now, preamble X and Z will do the same thing.

**Extracting the 16-bit audio samples from frames**

Now we know that preambles denote the beginning of a frame. Now let’s examine how to extract the 16-bit audio sample from each frame.



Figure 6

Figure 6 shows a frame. Bit 0 is sent first. Remember that each frame is 32 bits, or 64 cells.

Bits	Purpose
0-3	= preamble
4-11	= ignored
12-27	= 16-bit audio sample (The least significant bit is sent first)
28-31	= ignored

For now, ignore bits 4-11 and 28-31. We may use them in the future.

Assume frames are separated by garbage bits. To extract a 16-bit audio sample, you should follow the simple process in figure 7.

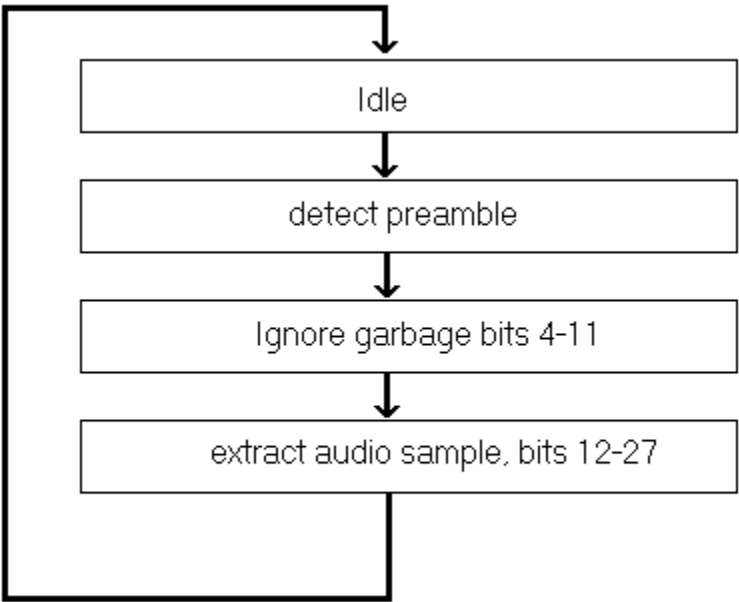


Figure 7

Now we know everything we need to extract the 16-bit audio samples from the bi-phase encoded input stream. Lets look at our output...

## A minor complication....

There is a slight challenge in extracting each 16-bit sample. Here's why. The fastest clock we have is 16MHz. A 16MHz clock has a period of 62.5ns. Unfortunately, the bi-phase encoded signal is sent at a non-multiple of 62.5ns. Each cell in the signal has a period of 180ns.

So What????

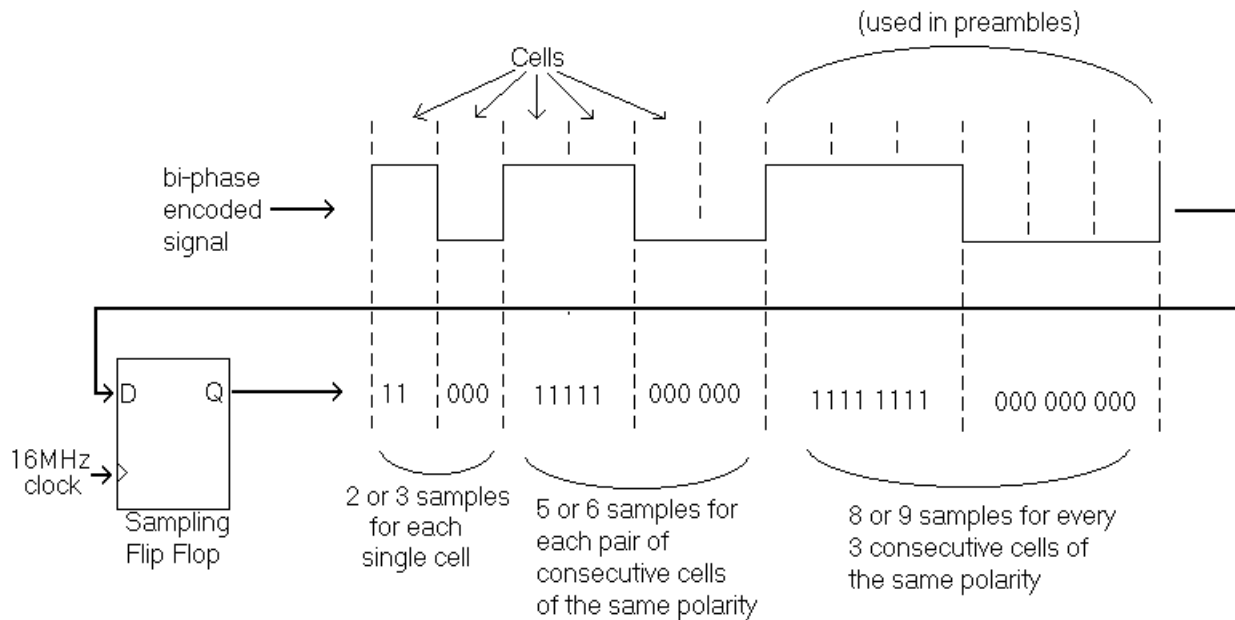


Figure 8

**Note:** Use an IFDI latch as the “sampling flip flop” to sample the bi-phase encoded signal from the CD-ROM drive. (An IFDI latch does not need an ibuf)

Figure 8 shows what happens when the bi-phase encoded signal is sampled using a flip flop. Because the periods of the cell and clock are not multiples of each other, 62.5ns verses 180ns, identical cell configurations will yield different data when sampled.

Lets examine Figure 8:

- A single cell can have 2 or 3 samples. (See the left part of the signal in Figure 8)
- Each pair of consecutive cells of the same polarity, as shown in the middle of fig. 8, can yield 5 or 6 samples. Remember that this denotes a data 0 and is also found in preambles.
- 3 consecutive cells of the same polarity, as shown on the right, can yield 8 or 9 samples. Remember that this is used in preambles only.

This means we can't just sample the input signal. We need to convert the signal into something a little more friendly. Keep reading...

## Converting the bi-phase encoded signal into a cell-stream

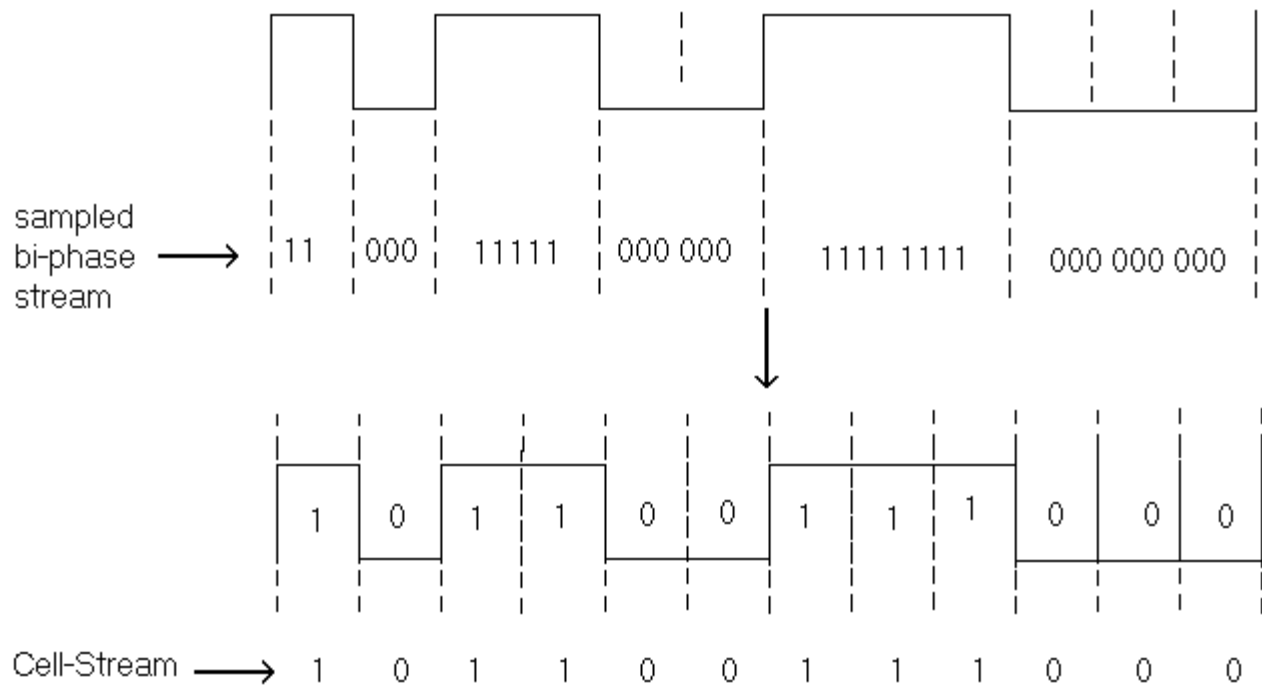


Figure 9

Figure 9 shows a “friendly” signal of cells. This cell stream has one bit for each cell. To convert this signal, create a mechanism that:

- When it sees 2 or 3 consecutive bits of the same polarity, it outputs 1 bit of that polarity.
- When it sees 5 or 6 consecutive bits of the same polarity, it outputs 2 bits of that polarity.
- When it sees 8 or 9 consecutive bits of the same polarity, it outputs 3 bits of that polarity.

As you can see, the “unfriendly” input signal is coming in at 16MHz. The cell-stream is at some almost random frequency, depending on the bit-stream. Your system must be synchronized with the cell-stream.

## Outputs

### DAC Data Format

A DAC is simple. You can think of the DAC as having a 16-bit shift register. It constantly shifts data in. When you tell it, by giving it a falling edge on a control line, the DAC latches onto the current data in the 16 bit register, and converts it to an audio signal.

This means that to output a 16-bit sample to the DAC, all you need to do is serially shift it into the DAC and pulse its control line immediately after the last bit goes into the DAC. Here's a diagram that may help explain this:

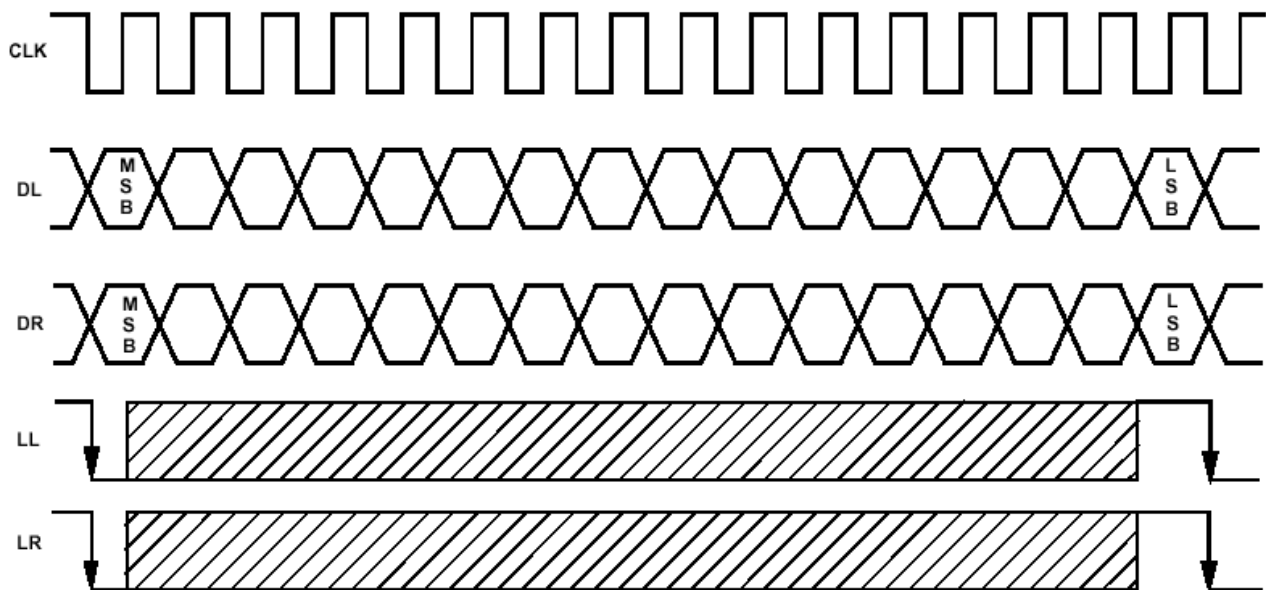


Figure 10

**Note:** You must output your signal, MSB, (most significant bit) first. This is opposite the bi-phase input stream which is LSB, (least significant bit) first.

DL and DR are the data lines you use to shift the samples into the DAC. DL and DR are for the left and right channels, respectively. LL and LR are the control lines. When the DAC sees a falling edge on this control line, it latches onto the last 16 bits you just sent it.

To create the pulse on LL and LR, invert the clock and AND that with a one clock cycle wide pulse occurring one clock cycle after the last bit has been sent on DL/DR. Please see figure 11.

When you don't want to shift data into the DAC, don't put a falling edge on LL or LR. The DAC only latches data when it sees a falling edge on these control lines.

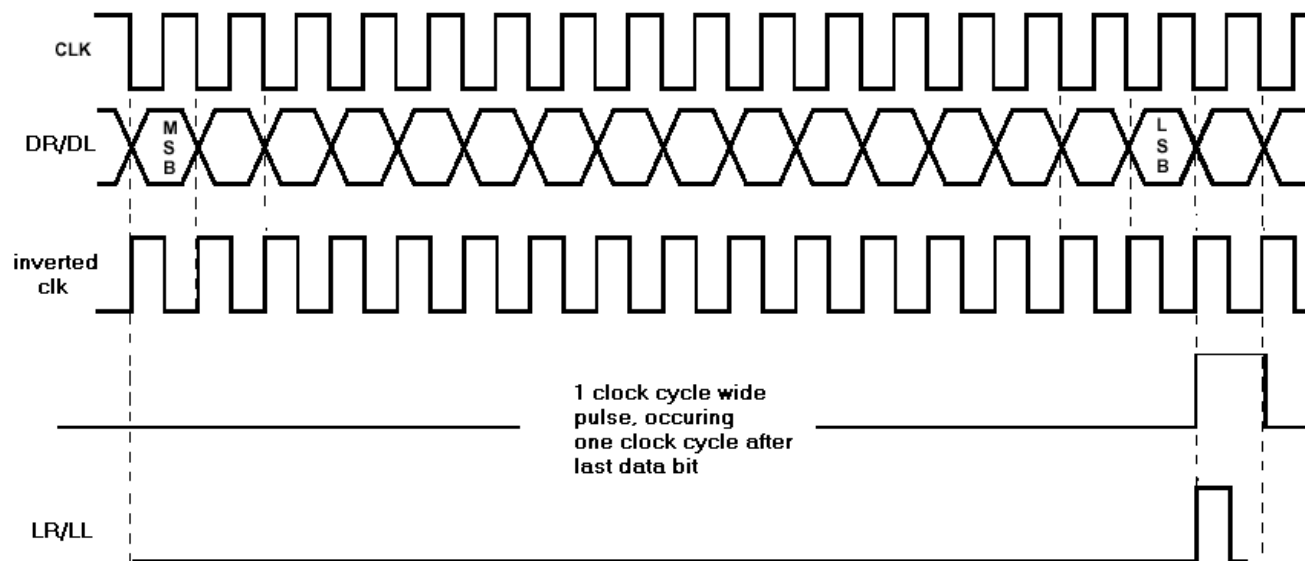


Figure 11

Figure 11 shows how to create the proper pulse on LR and LL.

### Wire wrapping...

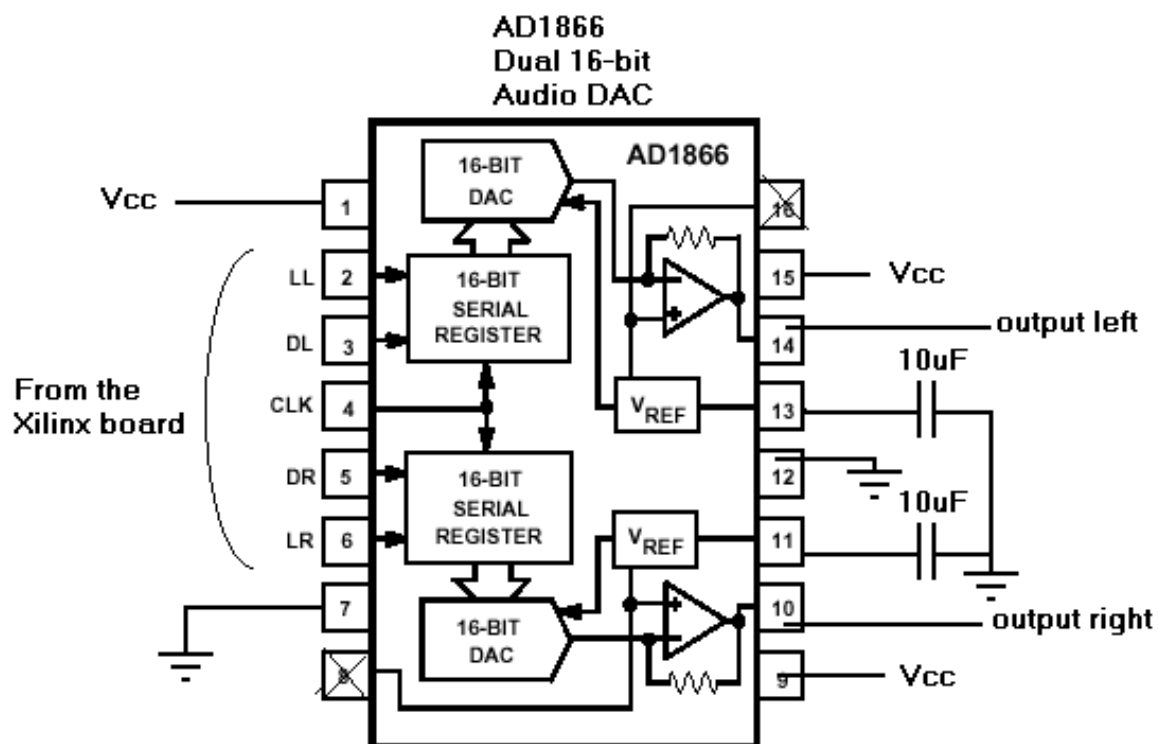


Figure 12



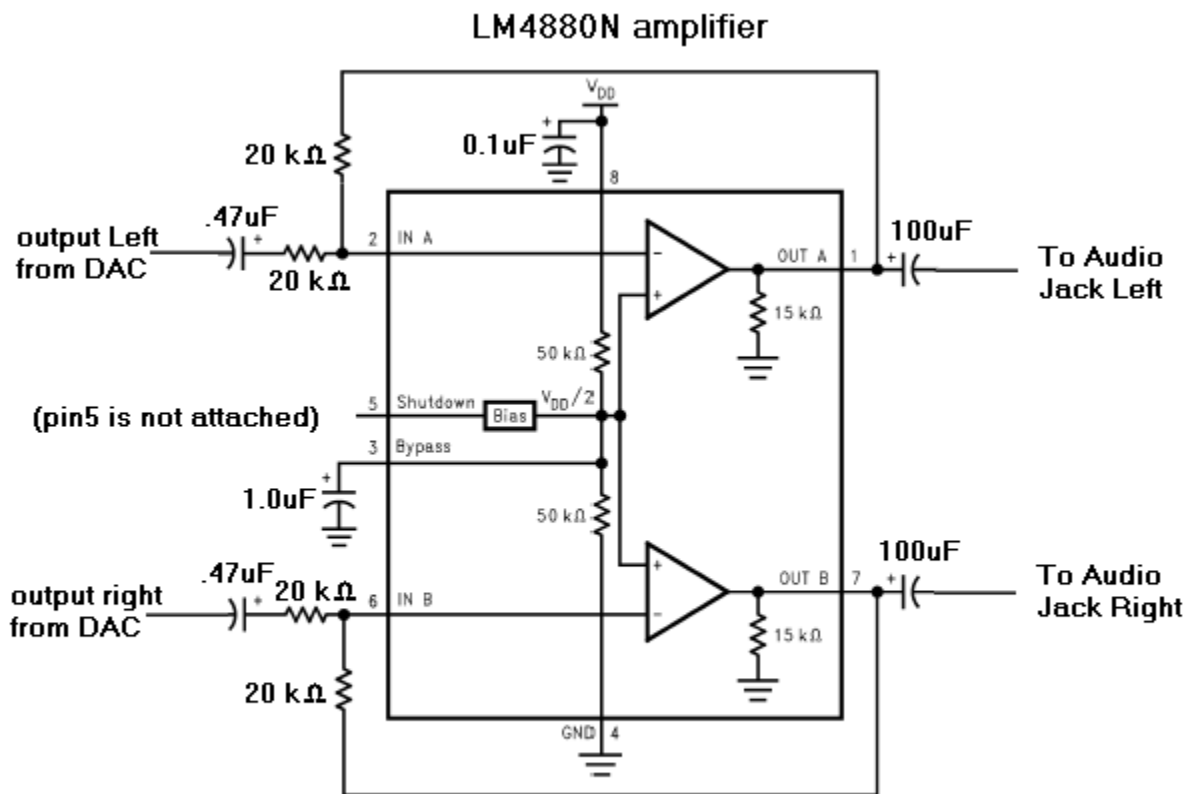


Figure 13

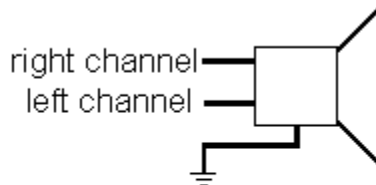


Figure 15- wire wrapping the audio jack

pin	function
61	DR
62	DL
68	LR
69	LL
78	CLK
79	input from CD-ROM

Figure 14 - Checkpoint 3 pinouts

### Tips:

- Start early. Go for some extra credit, but don't burn yourself out. You should probably spend time in other classes too.
- Simulate all you logic using script files.
- Understand the checkpoint. Read this specification several times. If you need more clarification, please refer to the data sheets or ask me questions.
- **USE SCRIPT FILES!!!!!!**

Name\_\_\_\_\_

Name\_\_\_\_\_

### Project Checkpoint 3

#### Checkoff Sheet

Everything works! \_\_\_\_\_

- Stereo sound
- Check LL and LR on pins 68 and 69

Works after unplugging and reattaching digital input \_\_\_\_\_

Done with Checkpoint 1 \_\_\_\_\_

Done with Checkpoint 2 \_\_\_\_\_

(Checkpoints 1 and 2 must be completed in order to get early extra credit)

Turned in two weeks early (11/2/01) \_\_\_\_\_(120%)

Turned in a week early (11/9/01) \_\_\_\_\_(110%)

**Turned in on time** (11/16/01) \_\_\_\_\_(100%)

Turned in a week late (11/23/01) \_\_\_\_\_(50%)