Lab #10: Music player

Physics 127BL Winter 2024

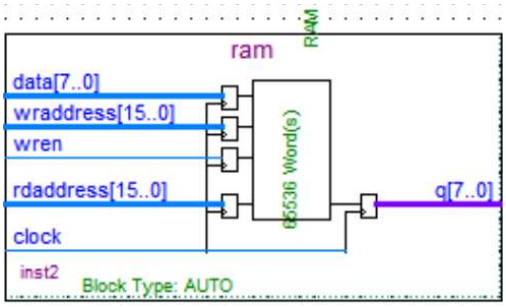
Lab report due **Thursday, March 21, at 11:55 P.M.**

Please read the lab report and homework guidelines handout on the course web page.

Introduction

In this lab, we will use the RS-232 serial interface to make a simple music player. We will also get some experience using random access memory (RAM), which is a widely-used component in digital systems.

The music player has two basic components: the download section that stores incoming RS-232 data to RAM, and a play section that sends this RAM data to the output. These two sections can be conveniently separated using a dual-port RAM, the Quartus module for which is shown below.



1. Create this dual-port RAM module using the `RAM:2-PORT` megafunction. Note that the write and read sections operate independently. Also note that the maximum size for the RAM is 64 k (= 2^{16} = 65536) 8-bit words, as determined by the internal resources of the FPGA, which corresponds to only a few seconds of music.

2. First code the download section, which takes data from the RS-232 interface and loads it sequentially into memory. The output of the RS-232 interface will have the 8-bit data and a `data_ready` line that pulses for one clock cycle when new data are available. Use this `data_ready` line to enable the RAM write enable line `wren`, storing the data. Also use this line to increment a counter that is connected to the address of the RAM, `wraddress[15..0]`.

3. The play section of the circuit will use a counter that increments its output at a rate of 16 kHz, the sample frequency set in the audio file. This counter is then connected to the read address, `rdaddress[15..0]`. The RAM output `q[7..0]` is connected to the VGA DAC input as in previous labs.

4. Since this is a complicated circuit, you might want to have the push-button switches clear the counters. You can display the values of the counters on the 7-segment LEDs to indicate that the loading and counting operations are functioning correctly.

5. In the `CHIMES_16s.txt` file under `lab10_audiofiles/` on the course web page is sound encoded with 8-bit resolution at a sampling frequency of 16 kHz. These data can be transferred to the FPGA via the RS-232 interface either by using the terminal emulator program or by following these steps in a Linux terminal window:

   (a) Either use the `cd` command to change to the directory (folder) where you placed the audio files, or copy the audio files to your home directory (`/home/student/` or similar).

   (b) Run the following command to set the serial port parameters:
   `stty -F /dev/ttyUSB0 115200 cs8 -parenb -cstopb crtscts`

   (c) This command will send the file `filename.txt`:
   `cat filename.txt > /dev/ttyUSB0`

   (d) If you have problems, you can test the connection using this command:
   `echo -n 'A' > /dev/ttyUSB0`
   This should put hex '41' on the FPGA board display if it is running the lab 9 code. (`'A' = 0x41`).

   Load the music player code into the FPGA, then send the audio file.

6. **External RAM interface.** A music player with about 1 minute of playing time can be constructed using the external RAM that is connected to the FPGA. This memory is single-port, and thus has a bidirectional data bus. The SRAM bus driver must use tristate outputs so that the write output can disconnect to avoid interfering with reads.

   The circuit for the interface is given below, where the input and output lines at the left and right sides correspond to those used in the FPGA dual port memory. The SRAM data bus is shown as a gated buffer symbol (can be located with the symbol tool explorer, not the IP catalog). The multiplexer chooses the address to be sent to the RAM using the "write enable" (`wren`) signal. The four connections on the bottom are various byte- and chip-enable lines that must be set to ground. The file `barracuda_8s.txt` contains music encoded at an 8 kHz sampling frequency that you can use to test this version of the player.