San Jose State University
Department of Electrical Engineering
EE178, Fall 2021

Laboratory Assignment #4

Objectives

This lab practices logic design using familiar building blocks. The goal of this lab is for you to implement a circuit that generates square waves at specific frequencies based on inputs, and outputs them to headphones using the audio jack present on the Real Digital Blackboard. When you successfully complete this lab, you will have developed a piece of intellectual property that you might be able to re-use in the future.

Figure 1: Mechanical Music Player

Now that you are familiar with the tools from Laboratory Assignment #1, you should be able to concern yourself with digital design. Figure 2 shows a symbol for the design you will create. The inputs are shown on the left and the outputs are shown on the right.

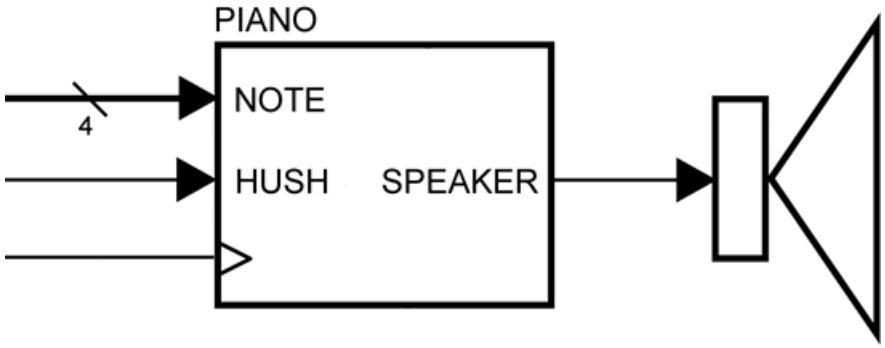

Figure 2: A Symbol for the Design

Bibliography

This lab draws heavily from documents on the Real Digital website https://www.realdigital.org. I would like to thank Real Digital for making this material available.

Square Wave Audio

There are many different techniques for creating audio output. The simplest technique for creating elementary tones is to generate a periodic waveform and then send that waveform to a speaker. For example, if you take a function generator (available in most analog laboratory courses) and connect the output to a speaker, you can hear the output when it is in the range of human hearing. This range is approximately 20 Hz to 20,000 Hz. With the function generator, you can control:

- The frequency – which your brain interprets as the pitch, or note heard.
- The amplitude – which your brain interprets as the volume, or loudness heard.
- The waveshape – which your brain interprets as a timbral characteristic, or "instrument" heard.

You are probably familiar with frequency and amplitude. The waveshape changes the quality of the note you hear, because the waveshape changes the number and amplitude of harmonics associated with the fundamental frequency of the waveform. A sinusoidal waveshape gives a perfect tone; that is, if you were to look at the Fourier transform of the time domain signal, you would see a spike at the fundamental frequency of the waveform itself, and no other frequency components. On the other hand, if you were to look at the Fourier transform of a square, saw tooth, or triangle waveshape, you would still see the largest spike at the fundamental frequency of the waveform, but you would see other, smaller spikes at multiples of the fundamental frequency. These are called overtones, or harmonics.

Without the aid of digital to analog converters or other analog circuitry, it is cumbersome to produce anything other than a square wave with digital logic, so we will create square waves as output. Square waves are simple to create; you can periodically toggle a signal, and then send that signal to a speaker to create audio output. For the amplitude parameter, we will simply use the output of the FPGA device directly, with no provision for controlling the amplitude.

The final parameter is the frequency parameter. Keyboard instruments are tuned using what is called the equal tempered music scale. In short, what this means is that the frequency of each note is related to the frequency of the adjacent notes by a constant multiple. The result is that music can be played equally well, or equally badly, in any key. Figure 3 shows a keyboard of sixteen notes. The piano design will be capable of playing any note on this keyboard.

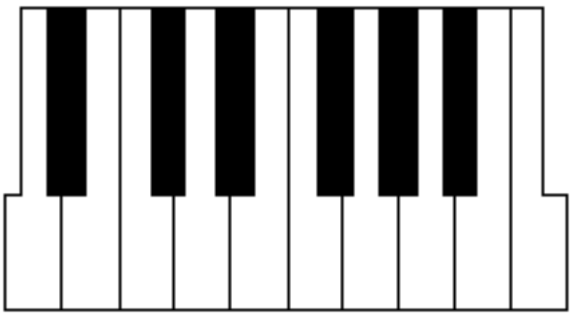

Figure 3: Our Virtual Keyboard

For those of you who play instruments, some of this may be familiar. I do not play an instrument, by the way, and I still made this circuit work correctly! While generating the data shown in Figure 4, I was running under the following assumptions:

1. Sixteen keys cover more than one octave, which should be sufficient.
2. The frequency of a key called "Concert A" or A4 is 440 Hz.
3. The twelfth root of two relates the frequencies of adjacent keys.
4. The single clock used in this design is running at a nominal 100 MHz.

Oscillator Frequency		100000000			
Oscillator Period		0.00000001			
Binary Note Code	Keyboard Note	Desired Frequency	Note Period	Half Period Clk Cycles	Rounded Term Cnt
0000	A4	440.00	0.002273	113636.36	113635
0001	A#/Bb	466.16	0.002145	107258.44	107257
0010	B/Cb	493.88	0.002025	101238.49	101237
0011	C/B#	523.25	0.001911	95556.41	95555
0100	C#/Db	554.37	0.001804	90193.24	90192
0101	D	587.33	0.001703	85131.08	85130
0110	D#/Eb	622.25	0.001607	80353.04	80352
0111	E/Fb	659.26	0.001517	75843.17	75842
1000	F/E#	698.46	0.001432	71586.42	71585
1001	F#/Gb	739.99	0.001351	67568.59	67568
1010	G	783.99	0.001276	63776.25	63775
1011	G#/Ab	830.61	0.001204	60196.77	60196
1100	A5	880.00	0.001136	56818.18	56817
1101	A#/Bb	932.33	0.001073	53629.22	53628
1110	B/Cb	987.77	0.001012	50619.25	50618
1111	C/B#	1046.50	0.000956	47778.21	47777

Figure 4: Calculating Half-Period Cycle Counts

The rounded terminal count column contains integer values that could be used in conjunction with a binary up counter, or a loadable binary down counter, to create the sixteen different notes. In addition, we will also need some method of telling the tone generator to be silent. This is useful for creating short pauses between notes; otherwise, you could not distinguish between two repeated notes and one long note.

Design Description and Requirements

In this design, you are not allowed to use latches. You are allowed to use only one clock. The clock must be the 100 MHz clock signal available from the oscillator on the board. You will receive zero points if you do not follow these requirements.

As shown in Figure 2, the module has three inputs. There is a clock input, plus a four-bit input to select one of 16 tones and an additional input to silence the audio output. There is one output signal, which is intended to be connected to the on-board audio amplifier.

clk	clock signal, 100 MHz from oscillator
note[3:0]	frequency select
hush	be silent
speaker	audio output

The module must drive the speaker output to generate a square wave at the frequencies listed in Figure 4. The speaker output must not toggle if the hush signal is asserted. Assertion of hush should simply disable speaker output transitions – it should not cause the speaker output to change value.

Describing the Design

Before you begin writing any code, you must sit down with scratch paper and draw a block diagram of a circuit that will satisfy the design requirements. Once you have a possible solution, write a description of it in Verilog-HDL and proceed to test it in simulation. Use the following template for your design.

```
// File: piano.v
```

```
// This is the top level design for EE178 Lab #4.

// The `timescale directive specifies what the
// simulation time units are (1 ns here) and what
// the simulator time step should be (1 ps here).

`timescale 1 ns / 1 ps

// Declare the module and its ports. This is
// using Verilog-2001 syntax.

module piano (
  input wire clk,
  input wire hush,
  input wire [3:0] note,
  output wire speaker
  );

  // Describe the actual circuit for the assignment.

endmodule
```

To facilitate re-use of your completed design, you must implement it in this single module – you are not allowed to use hierarchical design with sub-modules for this assignment. If you have further questions, or need clarification, consult the instructor.

Testing the Design

You must perform some minimal functional simulation of the design. This is important for two reasons. First, it will give you confidence your design is working properly before you implement it. Second, if the design does not behave as expected when you download it, you will have a mechanism to quickly create additional test cases to help debug the problem. The instructor will not help you debug logic problems (incorrect design behavior) unless you have a block diagram and are able to run a simulation.

In order to help you get started, here is a template for a test bench that works with the design. Feel free to enhance this as you see fit:

```
// File: testbench.v
// This is a top level testbench for the piano design,
// which is part of the EE178 Lab #4 assignment.

// The `timescale directive specifies what the
// simulation time units are (1 ns here) and what
// the simulator time step should be (1 ps here).

`timescale 1 ns / 1 ps

module testbench;

  // Generate a free running 100 MHz clock
  // signal to mimic what is on the board.

  reg clk;

  always
  begin
```

```
    clk = 1'b0;
    #5;
    clk = 1'b1;
    #5;
  end

  // In this block, include a mechanism
  // to exercise the design and finally
  // stop the simulation.

  reg [3:0] note;
  reg hush;
  integer loopvar;

  initial
  begin
    $display("If simulation ends before the testbench");
    $display("completes, use the menu option to run all.");
    note = 4'h0;
    hush = 1'b1;
    #100;
    $display("Beginning note loop.");
    // Loop through all 16 possible notes
    // and also exercise the hush signal.
    for (loopvar = 0; loopvar < 16; loopvar = loopvar + 1)
    begin
      hush = 1'b0; // make noise
      note = loopvar[3:0]; // assign note
      #10000000; // allow it to run
      hush = 1'b1; // go quiet
      #1000000; // allow it to run
    end
    // End the simulation.
    $display("Simulation is over, check the waveforms.");
    $stop;
  end

  // Now instantiate the top level design.

  wire speaker;

  piano my_piano (
    .clk(clk),
    .hush(hush),
    .note(note),
    .speaker(speaker)
  );

endmodule
```

Synthesizing the Design

Synthesize your design exactly as you did in the tutorial. Do not forget to check the reports. As a general practice, you will want to review all errors and warnings. These point to areas of concern that you should either address or justify.

Implementing the Design

Before you implement your design, you will need to add a constraints file. The following constraints are similar in nature to those used previously:

```
# Constraints for CLK
set_property PACKAGE_PIN H16 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -name external_clock -period 10.00 [get_ports clk]

# Constraints for SW0
set_property PACKAGE_PIN R17 [get_ports {note[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {note[0]}]

# Constraints for SW1
set_property PACKAGE_PIN U20 [get_ports {note[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {note[1]}]

# Constraints for SW2
set_property PACKAGE_PIN R16 [get_ports {note[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {note[2]}]

# Constraints for SW3
set_property PACKAGE_PIN N16 [get_ports {note[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {note[3]}]

# Constraints for BTN3
set_property PACKAGE_PIN M14 [get_ports {hush}]
set_property IOSTANDARD LVCMOS33 [get_ports {hush}]

# Constraints for SPEAKER
set_property PACKAGE_PIN G18 [get_ports {speaker}]
set_property IOSTANDARD LVCMOS33 [get_ports {speaker}]
```

Do not forget to check the reports. As a general practice, you will want to review all errors and warnings. If the design fails one or more timing specifications the reports will indicate this is the case.

Test your design in hardware. Does the circuit behave as you expect? If it does not, seek assistance. Once you are confident it works properly, demonstrate your final result to the instructor.

WARNING! The audio is extremely loud. Do not wear the headphones, but instead bring them only as close to your ears as necessary to hear the output of the circuit. With stereo headphones, the audio output will be on one channel only.

Laboratory Hand-In Requirements

Once you have completed a working design, prepare for the submission process. You are required to demonstrate a working design. You are also required to submit an archive of your project in the form of a ZIP file. Use the Vivado "File" → "Project" → "Archive..." option to create the ZIP file. Name the archive lab4_yourlastname_yourfirstname.zip.
You will submit your archive to the instructor through Canvas by the due date and time along with a narrated video of your demonstration. If your circuit is not completely functional by the due date, you should demonstrate and turn in what you have accomplished to receive partial credit. See the syllabus for the late penalty guideline.