APPLIED HOMEWORK #5

Sequential Circuits

1 Description and Homework Objectives

Have you watched the video entitled "Proper Coding of State Machines" under the heading "Applied Homework Resources"? If not watch that first. Then return to this document.

By the end of AHW4 we had created a clock & stopwatch where we could set time, set the alarm, and operate the stop watch. However, there was not a decent ability to test the alarm function since we had no audible alarm. In this final AHW we will implement a couple of counters (note frequency & note duration counters) along with a SM to make an audible tune on a piezo buzzer.

After this homework, you should be able to:
1. Properly code a state machine in verilog
2. Use an FPGA prototyping system to verify sequential circuit funtionality
3. Describe what a piezo bender is.

2 Background (what is a piezo bender)

Piezo electric elements come in many forms. Piezo electric materials expand and contract based on applied electric charge. But the process is reversible. So they can also produce electric charge based on an applied mechanical force.

The ignitor on a gas grill (that big red button that is kinda hard to push) is piezo based. When you are pushing that button you are cocking a hammer that then releases and strikes a piezo element. The mechanical force imparted produces an electric charge that is routed to a spark gap that then ignites the gas. You may have used a newer gas grill where the ignitor circuit is easy to push and uses a battery. Still piezo based…just that the battery spins a motor that has a hammer that strikes a piezo for you.

We are using the piezo in an electrical charge to a mechanical deformation direction. A piezo bender is a round copper disk (diaphragm) coated with a piezo material. The charge applied to the leads causes a deformation of the diaphragm. Piezo benders are used in ink jet printers as the mechanism that pumps the ink. They can also be used to make sound. The I/O pins of an FPGA cannot drive the 8ohm load of a speaker, but they can drive the capacitive load of a piezo bender. We will test with a piezo bender on leads just hanging in the air. It will be very quiet. However, a piezo bender properly designed into an acoustic chamber can be very loud. That super annoying sound from a smoke detector is coming from a piezo bender.

3 Applied Homework Tasks

3.1 Copy Blocks Forward

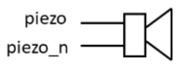
Download and unzip the **AHW5.zip** file into a folder labeled AHW5. Copy all verilog files (**.sv**) from your AHW4 folder to your new AHW5 folder.

3.2 Note Frequency Counter (really a period counter)

To achieve a given note frequency we can divide out 50Mhz system clock by some amount. Piezo benders operate at rather high frequencies and have a rather narrow band of frequencies that they have decent resonance over. We will be sticking to tones in the 1500 to 2000Hz range.

To produce frequencies in this range from a 50MHz clk requires a 15-bit counter. You are to produce a counter (**piezo_freq_cnt.sv**) that accepts a 15-bit number that specifies the period (**note_per**). It also has a synchronous clear signal (**clr**). The count value will reset to 15'h0000 if the current count matches the specified period, or if **clr** is asserted.

The MSB of this counter should form **piezo**. Then invert **piezo** to form **piezo_n**. By driving our piezo bender with complementary signals (**piezo** & **piezo_n**) we double our peak to peak voltage compared to just driving with **piezo** to ground.



You will find a file called **piezo_freq_cnt.sv** in the .zip. Flesh out the missing Verilog to implement the design. There is a self checking test bench (**piezo_freq_cnt_tb.sv**) available. You know the routine…build a ModelSim project, simulate, debug, and get it working. Capture the "YAHOO! test passed" message from the transcript window (**piezo_freq_cnt_yahoo.jpg**) Submit **piezo_freq_cnt.sv**, and **piezo_freq_cnt_yahoo.jpg** to the dropbox for AHW5.

3.3 Note Duration Counter (piezo_dur_cnt.sv)

We need a counter to also keep track of how long we have played any particular note. You will implement another counter (piezo_dur_cnt.sv) to accomplish this. Dividing a 50MHz clock down to something of seconds duration requires a really wide counter (*think a lot of gates/flops*). We already have a 0.01sec timebase coming from **clkdiv.sv** so we will make use of that to enable our duration counter. That way it can be much narrower (8-bits) and we can still get decent durations (2.5sec or longer) durations from it. Our counter will also require a synchronous clear (**clr**) so that we can reset the duration count when we move onto playing a new note.

An incoming 8-bit signal (**note_dur**) specifies the duration (*in 0.01sec increments*) of the note we desire to play. When the count value reaches the specified duration (**note_dur**) then **note_over** should be asserted and the count register should be set back to zero (*synchronously*).

You will find a file called **piezo_dur_cnt.sv** in the .zip. Flesh out the missing Verilog to implement the design. Pay attention to the comments specifying what to do. There is a self checking test bench (**piezo_dur_cnt_tb.sv**) available. You know the routine…build a ModelSim project, simulate, debug, and get it working. Capture the "YAHOO! test passed" message from the transcript window (**piezo_dur_cnt_yahoo.jpg**) Submit **piezo_dur_cnt.sv**, and **piezo_dur_cnt_yahoo.jpg** to the dropbox for AHW5.

3.4 SM to Control Alarm Tune (piezoSM.sv)

The alarm tune will simply be 3 ascending notes followed by a pause. This pattern repeats until the **stop** signal indicates the alarm should end.

Note:	**note_per setting required**	**note_dur setting:**
G octave 6 = 1568Hz	15'h7C90	8'h40
A octave 6 = 1760Hz	15'h6EF9	8'h20
B octave 6 = 1975Hz	15'h62E4	8'h10
Pause (no sound)	15'h038E	8'h20

The SM will sit in **IDLE** until it receives the start signal. Then it will start playing the 3 notes and the pause in order and continue that pattern until the stop signal is received. Every state should look for the stop signal, because we don't know what state (note) we are in when the user cancels the alarm. Is the pause state (no sound) really a no sound state? If a tree falls in the forest and no one is there to hear it does it make a sound? A note is being played in the Pause state, but trust me…not even a dog would be able to hear it.

3.4.1 Finish the State Diagram

At the end of this file you will find a state (bubble) diagram. The states have been given somewhat meaningful names. Print it out and complete it by hand. Take a clear picture of your completed state diagram (**state_diagram.jpg**) with your phone and submit to the dropbox for AHW5.

3.4.2 Implement & Test the State Machine

A file called **piezoSM.sv** is provided. It instantiates the state flops needed (**state5_reg**). The main intellectual property of any state machine is the ***always_comb*** (with ***case*** statement) block that determines the state transitions and state machine outputs. Complete this ***case*** statement using your state diagram as a guide.

As always a self checking test bench (**piezoSM_tb.sv**) has been provided. Using ModelSim compile, simulate and debug. Grab a screen capture of the transcript window showing the "YAHOO test passed" message (**piezoSM_yahoo.jpg**). Submit **piezoSM.sv**, and **piezoSM_yahoo.jpg** to the dropbox for AHW5.

4 Demo Tasks

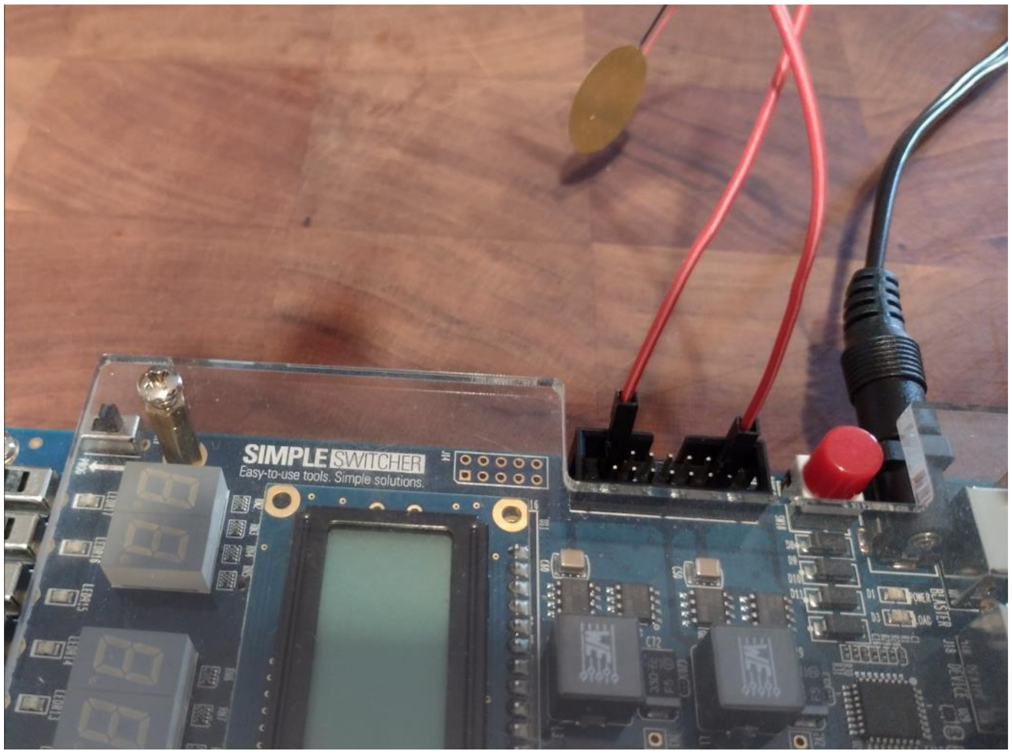
DO NOT wait until your demo to read this!

During your Applied Homework Demo, you will compile **AHW5.sv** in Quartus and map it to the FPGA board. You should be able to use the slide switches to select set time mode, stop watch mode, and set alarm mode. In addition to **SW[2:0]** performing those functions **SW[17]** serves as **RST_n,** and **SW[16]** can accelerate time by 4096x for testing purposes. PB3 can reset the stopwatch value being displayed.

Study **AHW5.sv** note around line 155 there is logic provided that compares the current time to the alarm set point. For the alarm to be enabled **SW**[3] (*which forms* ***alarm_on***) must be high.

The alarm can be cancelled by pushing **KEY**[3] (**PB**[3]).

One final note…**LEDR**[15:11] are mapped to display the 5-bit state of ***piezoSM***. This might help you with debugging.



To test the alarm you will have to call a TA over so they can put a Piezo buzzer across the two driving pins (see image above).

5 Applied Homework Submission

If working with a partner, your group should submit only one report, and both group members will receive the same credit for the report component of the Applied Homework. The grade for the demo component may differ based on demonstrated knowledge and understanding of the submitted work.

You will upload a number of files to the AHW4 dropbox, be sure that your files are named correctly.

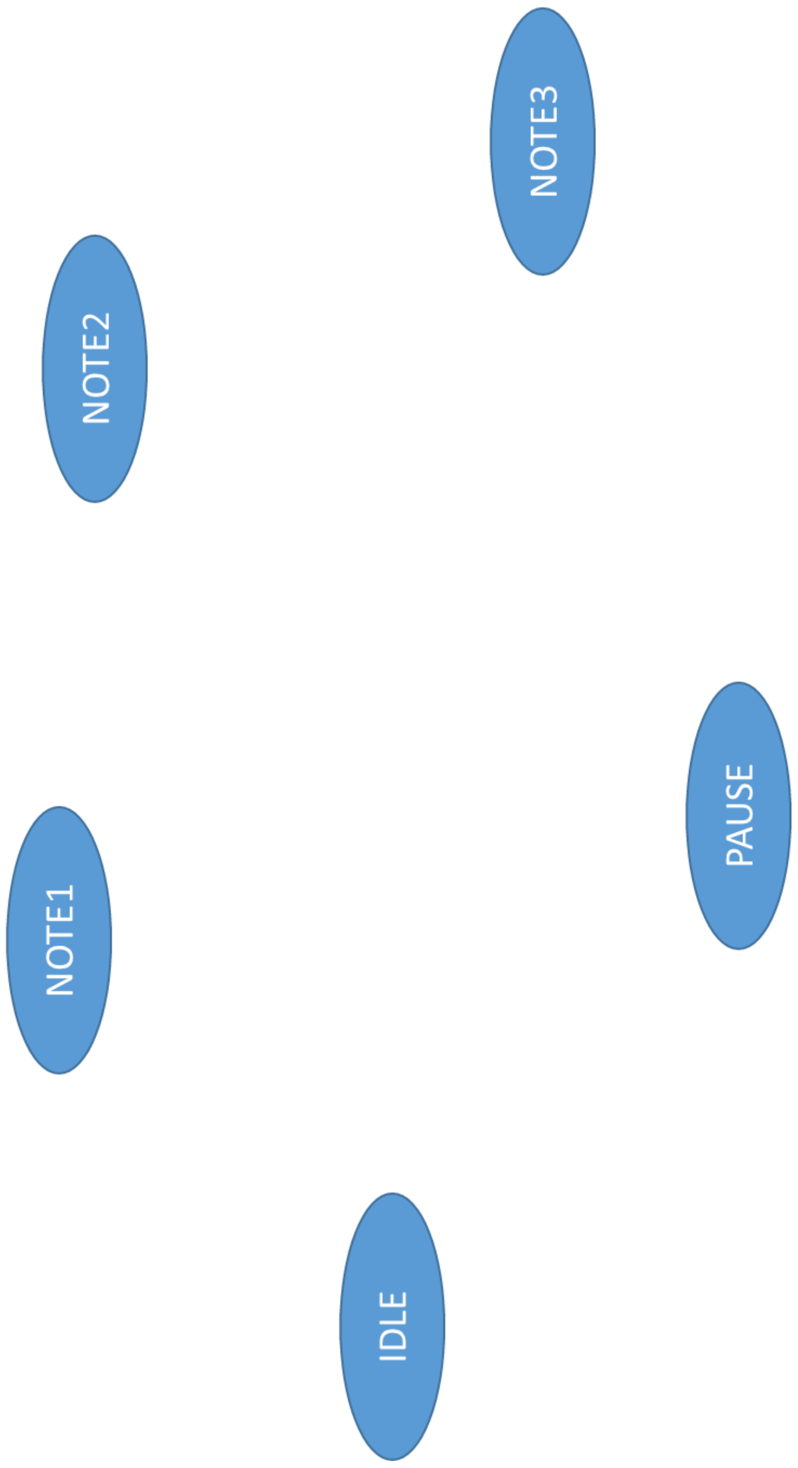
- **piezo_freq_cnt.sv** – verilog of frequency (period) counter
- **piezo_freq_cnt_yahoo**.jpg – Image of transcript window showing self-checking testbench passed
- **piezo_dur_cnt.sv** – verilog of note duration counter
- **piezo_dur_cnt_yahoo**.jpg – Image of transcript window showing self-checking testbench passed
- **state_diagram.jpg** – picture of your completed state (bubble) diagram
- **piezoSM.sv** – Completed Verilog of control SM.
- **piezoSM_yahoo.jpg** – capture of transcript window showing self-checking testbench passed

The deadline for your report appears online on the course webpage.

Name(s):______________________________________

Indicate Default Value of SM Outputs:

clr =
note_per =
note_dur =


NOTE3
NOTE2
NOTE1
PAUSE
IDLE