Name: Harshil Rajesh Patel Student Number: 35437326

CPEN 311 LAB 1 – Basic Tone Organ

1. Directory / Path of .sof File:

harshil_rajesh_patel_35437326_Lab_1/lab1_template_de1soc/rtl/Basic_Organ_Solution.sof

Additional Info on Important Files:

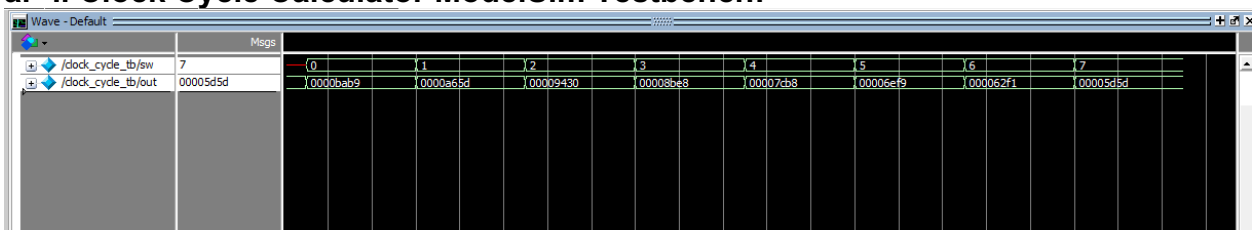
- All .sv files created (modules) are in /rtl.
- All sim files (ModelSim project, testbenches, waveforms) are in /sim.
- Screenshots are in /doc

2. Status of the Lab:

- Completed and tested the clock divider module (clk_divider.sv, clk_divider_tb.sv).
- Completed and tested the LED control (moving from side to side) on DE1_SOC as well as ModelSim (led_control.sv, led_control_tb.sv).
- Added code to change the LCD Display of the oscilloscope in SignalTap as requested (Basic_Organ_Solution.v).
- The DE1-SOC behaves as intended when programmed, the switches [3:1] can successfully produce the octave and switch [0] can turn audio on/off.

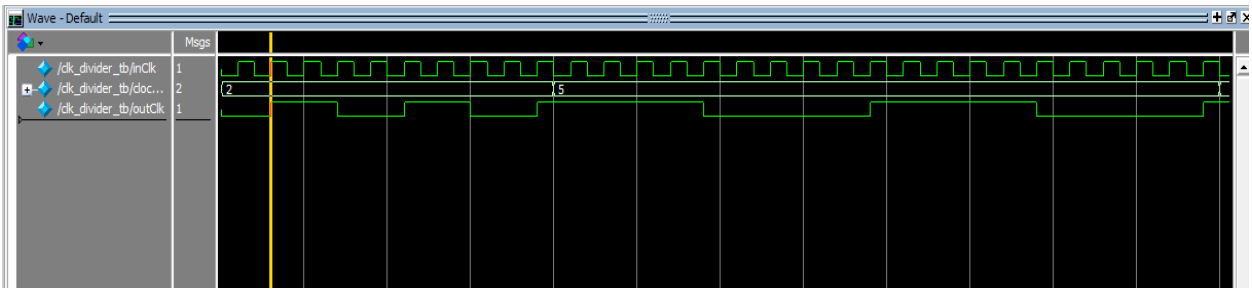
3. Simulation Screenshots:

a. i. Clock Cycle Calculator ModelSim Testbench:



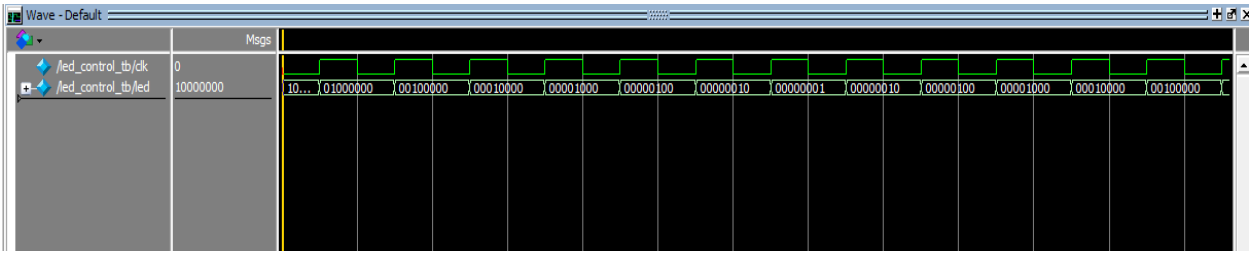
- The Clock Cycle Calculator module is used to calculate the number of clock cycles from the frequencies of the octave notes input through the switches.
- As seen in the above waveform, as we cycle through the different notes (0 – do, 1 – re, …., 7 – do2) from the switches (sw), the output (out) corresponds to the number of cycles required for division of CLOCK_50M to achieve that note.
- Note: The cases coded for the output (out) correspond to half the clock cycle count that the 50 MHz clock input needs to be divided by. The reason for this will be better understood in the explanation of the waveform of the clock divider module.
- If we look at the possible clock cycle outputs for the different switch inputs in clock_count_calc.sv file, it is seen that the waveform produced is as expected.
- Hence, as this waveform shows that the output is as intended, the testbench was successful and the module is working correctly.

ii. Clock Divider ModelSim Testbench:



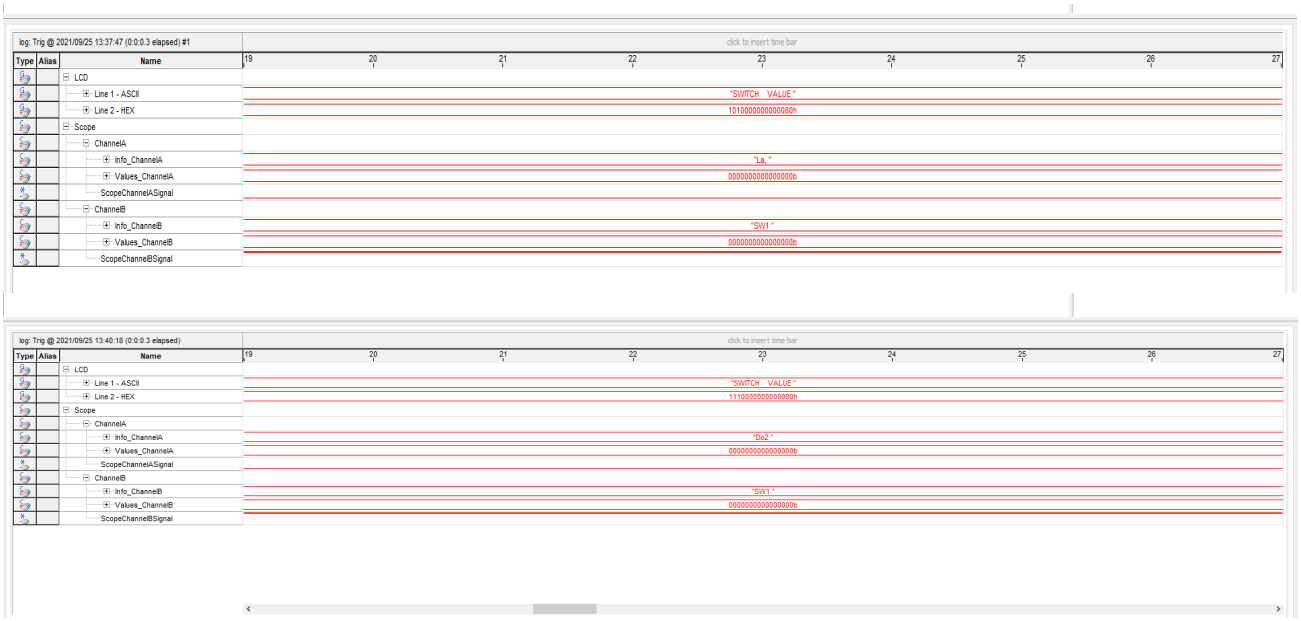
- The Clock Divider module is used to divide the frequency of the CLOCK_50M signal to produce the frequencies of the octave notes.
- It takes as input the CLOCK_50M (to be divided), reset (set to 0 so excluded from waveform), clock_count (from the Clock Count Calculator module), and outputs outClk (the new divided wave) with the new frequency.
- However, my clock divider works by outputting a wave that is divided by double the input clock_count.
- For example, in the first test case when the clock_count input is set to 2, the outClk signal is divided by 4 (double of the input). This is because for every 4 cycles of inClk, one cycle of outClk is produced.
- Similarly, in the next test case where the clock_count input is set to 5, the outClk signal is divided by 10.
- However, the correctness of the module's use in context of the basic organ solution is unaffected because the clock_cycles input into the clock divider are of half the actual value (to maintain 50% duty cycle). Hence, the output produced by the clock_divider would be correct and as intended.
- There is a reason for making the clock divider behave in this way. Originally, I used an implementation that did not divide by double the input. However, I realized upon testing that if I was to manually divide my clock_count in the module, the output (outClk) would not work for odd divisors (3, 5, 7, etc). This is because the hardware performs integer division by using the closest even number. So, division by 2 and 3 would produce the same and thus, incorrect result.
- Therefore, by halving the input clock_cycles instead, we can get the desired and correct output.
- The saved waveform (clock_divider_wave.do) includes another testcase (division by 10) which it passes as well (ends up producing a wave that is divided by 20). Hence after this analysis, I concluded that the clock divider is working as I intended it to for my implementation.

b. LED Control ModelSim Testbench:



- The LED Control module is used to shift the LED input on the DE1-SOC from side-to-side at a frequency of 1Hz (every second).
- As seen in the waveform above, on every positive edge of clk, the 1 bit on the 8 bit led output is shifted from left-to-right first and when it reaches the least significant bit, it then begins shifting from right-to-left.
- The process continues in a similar pattern while there is a clock input.
- Hence, I concluded that the LED control module is working as intended and upon uploading it to the DE1-SOC board, I can see that the LEDs behave as required.

4. SignalTap Screenshots:



- Upon looking at the screenshots of the LCD and Oscilloscope in SignalTap, we can observe the following things are working as required:
- Line 1 – ASCII shows the text “SWITCH VALUE”
- Line 2 – HEX shows the switches that are on/off (most significant bits) and the least significant bits show the value of ‘audio_data’.
- In the Scope Channel A Info section, we can see the text of the note as set by switch (for example – SW[3:1] = 101 => ‘La’ and SW[3:1] = 111 => ‘Do2’).
- Hence, the SignalTap screenshots above show that solution is working as intended.

5. **How I ran the simulations:**

- The simulations were run making a ModelSim project lab1.mpf and uploading the testbench (_tb.sv) files.
- Upon adding the files to the project, they were compiled and then uploaded using ‘vsim module’, and then the commands ‘restart -f’, and ‘run -all’.
- Each of the three modules were tested (clk_divider, clock_cycle_calc, led_control) and their waveforms analyzed.
- The annotations/explanations of the waveforms can be found in the earlier sections of this document, while the .do files are also available to view in the ./sim folder.
- The waveforms were used to debug the modules.

6. **Additional Info:**

- References from the textbook are mentioned in comments where possible.
- Any calculations are also mentioned in comments.