CPEN311 Winter 2017 Term 1
University of British Columbia

Lab 1: Tone Organ (Basic Verilog/VHDL)

In this lab we will construct a basic 1-octave frequency organ.

1. Download and decompress the template from the website.

2. Open the Quartus project Basic_Organ_Solution.qpf

3. Compile the project and verify that you can load the card.

4. Note that many modules in the code are intentionally given to you as a "black box", i.e., you cannot see their internal functioning. This is because some of those modules may be written by you in future labs or homework assignments.

5. Look at the LCD. (The DE1-SoC does not have an LCD. See the last page of this document for a tutorial on using SignalTap in place of the LCD).

   For the DE2: Make sure SW17 is in the "on" position, and that SW16 is also in the ON position. The LCD serves as a primitive 2-channel "oscilloscope". The top channel is connected to a 1KHz signal. The bottom channel is connected to SW1. Toggle SW1 in order to see that the oscilloscope indeed functions. Now set SW16 to the "off" position. As you may have deduced, you should see a square wave. SW16 is the "run/hold" setting for the oscilloscope. The 1 KHz signal was produced with a clock divider called Generate_Arbitrary_Divided_Clk32 which divides the 50 MHz input in order to generate 1 KHz. This signal also goes to the audio output. Find the relevant code in the template and study it.

6. (For the DE2 in the actual oscilloscope, for DE1-SoC: using SignalTap): The oscilloscope by default samples at a rate of 2 KHz per character (which means that a 1 KHz signal will look as:

   ```
   _-_-_-_-_-_-_-
   ```

   (i.e. 2 samples per period). You can change the timescale of the oscilloscope by pressing KEY0, KEY1, and KEY2. KEY0 will "expand" the time scale (i.e. make the oscilloscope sample faster), whereas KEY1 will make it sample slower. KEY2 will reset the sampling frequency. Play with KEY0, KEY1, and KEY2, as well as SW16, to take stock of the behaviour of the oscilloscope.

7. (for DE2, for DE1-SoC see appendix regarding Signal Tap): Now, set SW17 to the "off" position. You should see a bunch of letters and numbers in the LCD. This mode of the LCD is called "Information Console" mode.

8. Both the Oscilloscope and "Information Console" modes are controlled by the module "LCD_Scope_Encapsulated_pacoblaze" (for the DE1-SoC it is "LCD_Scope_Encapsulated_pacoblaze_wrapper"). You don't need to study the internals of this module (which are hidden from you anyway), however **do** study the inputs/outputs of this module, as you will need to interface to it.

9. You have to modify the code to do the following things:

   a. Audio: Let's build an organ tone one octave. That is, let's generate the notes "Do Re Mi Fa So La Si Do". Sound frequencies of these notes are: [587Hz 523Hz 659Hz 698Hz 783Hz 987Hz 880hz 1046Hz]. To do so, make a clock division controllable from the switches SW[3:1]. To start, you can use the module Generate_Arbitrary_Divided_Clk32 (which is not documented by the way, if you want to use it have to do "reverse engineering"), but in the version that you submit you must build your **own** clock divider. Clock dividers are a special type of counter which is controllable by the switches SW [3:1] to get the desired frequency. One of the objectives of this lab is to design that frequency divisor.

   b. SW[0] should enable audio output in the "On" position, and disable it when in the "Off" position.

   c. Modify the code so that when a certain audio frequency is selected, this wave will be seen in the "oscilloscope" with the appropriate description (i.e. is written in the caption to the left of the signal for the DE2 as "Do", "Re", "Mi", "Fa", "So", "La" "Si", "DO2"; for the DE1_SoC it will appear in the "info" line above the appropriate "scope" channel in the "Scope" section of SignalTap).

   d. Modify the code so that in "Information Console" mode, the bottom line will show the position of the switches as well as the current audio data (in hexadecimal). Make the top line say something interesting (For the DE1-SoC these are Line 1 and Line 2 in the "LCD" in SignalTap).

   e. LED Control: again using a frequency divider, make a 1 Hz clock and use it to control the eight green LEDs so that they go from side to side, changing LED every 1 second. It is up to you to figure out how to do this (there are many correct ways). (If you are using a DE1-SoC board, use the red LEDs, 0 to 7)

10. If you still have questions, contact your lab section's TA. But do not only contact them at the last minute! I do not want to see a situation that nobody comes to ask anything until the last 3 days and in a panic.

11. (10% Bonus question - can used towards the "creative bonus"): We generate sound in this laboratory with rectangular waves. This is not such a "clean" way to generate the sound (because "pure" sound is a sine wave). Humans can hear sounds with frequencies up to 20 KHz. Make a table of which frequencies we actually hear from the organ tones (hint: there is some advanced math involved), and explain how you arrived at these results.

It is possible, if you send data incorrectly, for the audio chip on the DE2/DE1SoC boards to enter an error state, where it will not respond and not play any audio. If at some point your design is no longer playing any audio, try powering off your board, turning it back on, and reprogramming your design. This will reset the chip.

Furthermore, there are several useful videos in the "Videos" page of the website which may be useful for you for this lab and future labs. Please take a look at those videos now.

Remember:

As always, if in doubt about what to do, load the card with the solution file (.sof) that can be found on the website.

Do not forget the rules of good design:

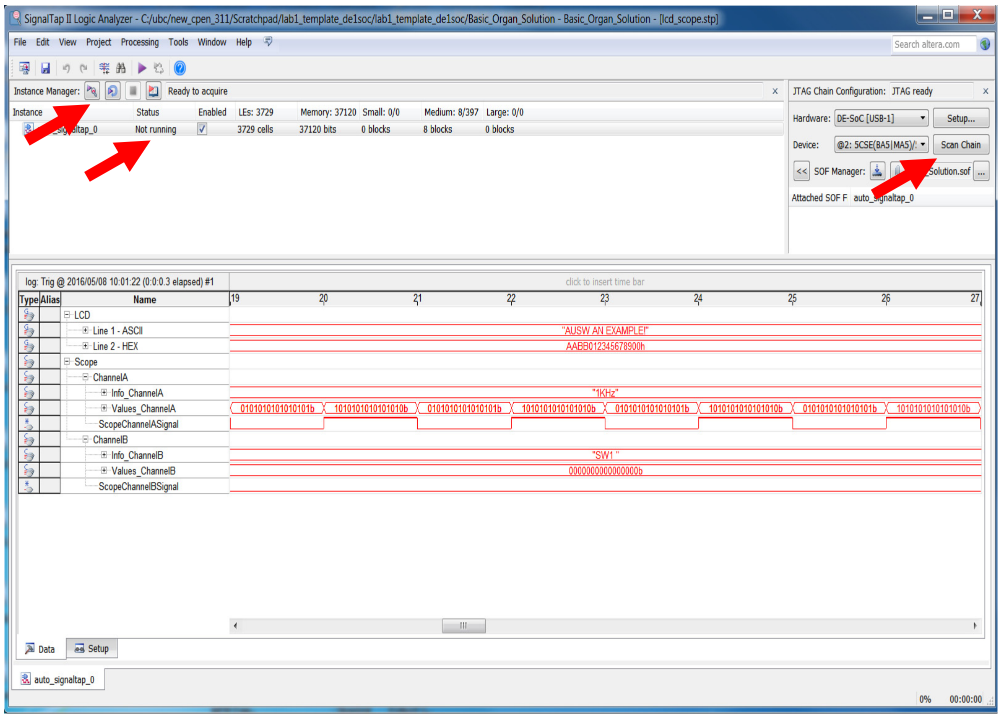
- Always design while thinking about the **hardware** implementation
- Use **simple** structures
- **Incremental** design and test
- **Modular design**: **divide** the work, work in **parallel** in the group
- Use the **RTL viewer**
- Verification and test: Use **simulations**, LEDs, 7-segments
- Write **Clean**, **Neat**, and **Legible** code
- Give **meaningful** names to variables, use **comments**
- Go over and understand the warnings given by Quartus during compilation
- The circuit should be correct by **design**, not just by simulation.
- The code should be **verifiable** by **inspection**
- Design in a **modular** fashion, always thinking about future **reutilization** of the module.
- **Re-use** proven modules, instead of re-inventing the wheel all the time

Good luck and have fun!

Appendix: Using SignalTap to emulate an LCD on the DE1-SoC

1. Launch SignalTap. In Quartus, in the Project Navigator, open the 'Files' tab, and double click the "lcd_scope.stp" file. This will open SignalTap and load the configuration for viewing the internal circuit signals that would go to the LCD (if the DE1 had an LCD).

2. Program the DE1-SoC board with your design.


SignalTap II Logic Analyzer - C:/ubc/new_cpen_311/Scratchpad/lab1_template_de1soc/lab1_template_de1soc/Basic_Organ_Solution - Basic_Organ_Solution - [lcd_scope.stp]
File Edit View Project Processing Tools Window Help
Search altera.com
Instance Manager:
Ready to acquire
JTAG Chain Configuration: JTAG ready
Setup...
Scan Chain
SOF Manager:
Solution.sof
click to insert time bar
Data
Setup
auto_signaltap_0
0%
00:00:00

Instance	Status	Enabled	LEs: 3729	Memory: 37120	Small: 0/0	Medium: 8/397	Large: 0/0
_signaltap_0	Not running	✓	3729 cells	37120 bits	0 blocks	8 blocks	0 blocks

Hardware: DE-SoC [USB-1]
Device: @2: 5CSE(BA5|MA5)/:
Attached SOF F auto_signaltap_0

log: Trig @ 2016/05/08 10:01:22 (0:0:0.3 elapsed) #1

Name	Value
LCD	
Line 1 - ASCII	"AUSW AN EXAMPLE!"
Line 2 - HEX	AABB012345678900h
Scope	
ChannelA	
Info_ChannelA	"1KHz"
Values_ChannelA	0101010101010101b, 1010101010101010b, 0101010101010101b, 1010101010101010b, 0101010101010101b, 1010101010101010b, 0101010101010101b, 1010101010101010b
ScopeChannelASignal	
ChannelB	
Info_ChannelB	"SW1 "
Values_ChannelB	0000000000000000b
ScopeChannelBSignal	

3. Click the "Scan Chain" button on the right hand side. This will communicate with the SignalTap logic that is programmed onto the device, and the "Status" field should display "Not running"

4. To retrieve a snapshot of data from the FPGA, click the entry "auto_signaltap_0", then click the "Run Analysis" button which is shown in the left-most arrow in the diagram above.

5. The data from the FPGA should load into the waveform.

   a. The "LCD" data is shown first. The "Scope" data is shown second. Remember that SW[8] acts as a capture control, and the Scope data will only be updated while SW[8] is high.

      Note: SW[9] on the board switches between whether the "LCD" or "Scope" data is shown on the LCD. Since we are using SignalTap and not a physical LCD we can view both of these signals simultaneously. However the way the LCD scope is designed, you must set SW[9] high at least once before it will start capturing "Scope" data.

6. Each time you want to capture new data, just click the "Run Analysis" button again.

Note that the above is talking about using SignalTap to debug your lab when using the lab template as a basis for your code. If you want to use SignalTap to look at the "LCD" contained in the solution SOF file, refer to the relevant video in the "Videos" page of the course website for instructions.