The following is to be used as an example template ONLY. This is not an actual lab that any students will complete. Several parts of actual labs, including Procedures, Data, and Analysis sections will be expected to be a bit longer in order to meet all requirements specified in each lab assignment. Labs are NOT required to be in color. Any notes on key parts of this example will be shown in italics *only.*

Questions for this example lab are:

- Describe how output order on the LEDs relates to Behavioral Verilog index order for a shift register
- Explain your method for determining Behavioral Verilog index order

See the following pages for the example report

Player 1 Name: PID
Player 2 Name: PID
ECE 25
Section: A5_
Monday, 7p – 10p, WLH 2211

Lab 0: Gate the Clock

Introduction:

The purpose of this lab is to familiarize students with the index order of shift registers in Xilinx ISE Project Navigator as well as Behavioral Verilog syntax. In this lab, we will write custom Verilog code in order to practice our use of Behavioral Verilog and test how data bits travel through logic. We will investigate several index ordering schemes in order to become accustomed to Verilog shift register declarations.

Procedure:

- We downloaded the *gatetheclock.v* Verilog skeleton file from the course website section
- We wrote the Behavioral Verilog shown in **Figure 3**
- Using Xilinx ISE, we debugged our code until it compiled correctly and wrote a Verilog testbench testing various input cases.
- With Modelsim, we ran a behavioral simulation to ensure our logic was functioning correctly as shown in **Figure 4**
- Finally, after all debugging steps were completed, we generated a .bit file
- Using Digilent Adept we uploaded our .bit file onto the Basys2 board and cycled the power
- We connected the Function Generator and Oscilloscope as shown to provide the clock source for our Basys2 board logic with a 0-3.3V square wave at 1Hz.
- We obtained the LED results shown in **Figure 5** and **Figure 6** using switches to determine data inputs and a shift scheme selector.

Circuit Diagrams:

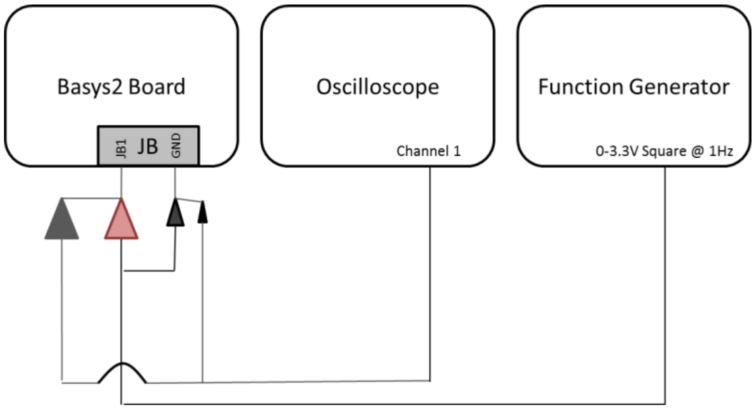

Figure 1: Basys2 Board Connection

Note: In this example, I only have the Basys2 Board and Function Generator connected. Normally, your labs will include Integrated Circuit chips. You are required to include the pinout diagram of those chips as well as the connections you made to them. An example pinout is shown below.

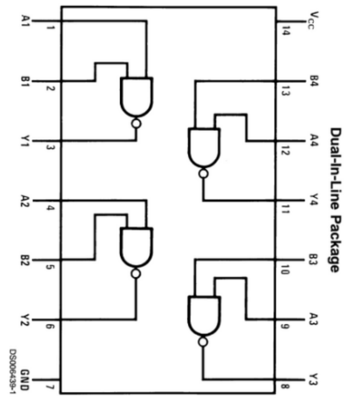

Figure 2: 74LS00N NAND IC

Data:

```
`timescale 1ns / 1ps
module gateclock(clock, reset, data, enable, led1, led2, led3, led4);
   input clock;
   input reset;
   input data;
   input enable;
   output led1, led2, led3, led4;
   reg[7:0] temp;               //all wires that do not change simultaneously must be declared as a registers
   reg led1, led2, led3, led4;  //all wires that change as a result of "if" statement must be declared as registers

always @ (*)      //sensitivity list for always block
begin
            led1 = temp[0];
            led2 = temp[1];
            led3 = temp[2];
            led4 = temp[3];
end
always@(posedge clock)             //this is the standard clock input to a shift register
begin
            if     (reset)             temp = 0;     //reset will clear register
            else if (enable == 1) temp = {data, temp[7:1]}; //while enabled, the register will shift data in one way
            else if (enable == 0) temp = {temp[7:1], data};     //else, the register will shift data in another direction
  end
endmodule
```

Figure 3: gatetheclock.v Behavioral Verilog

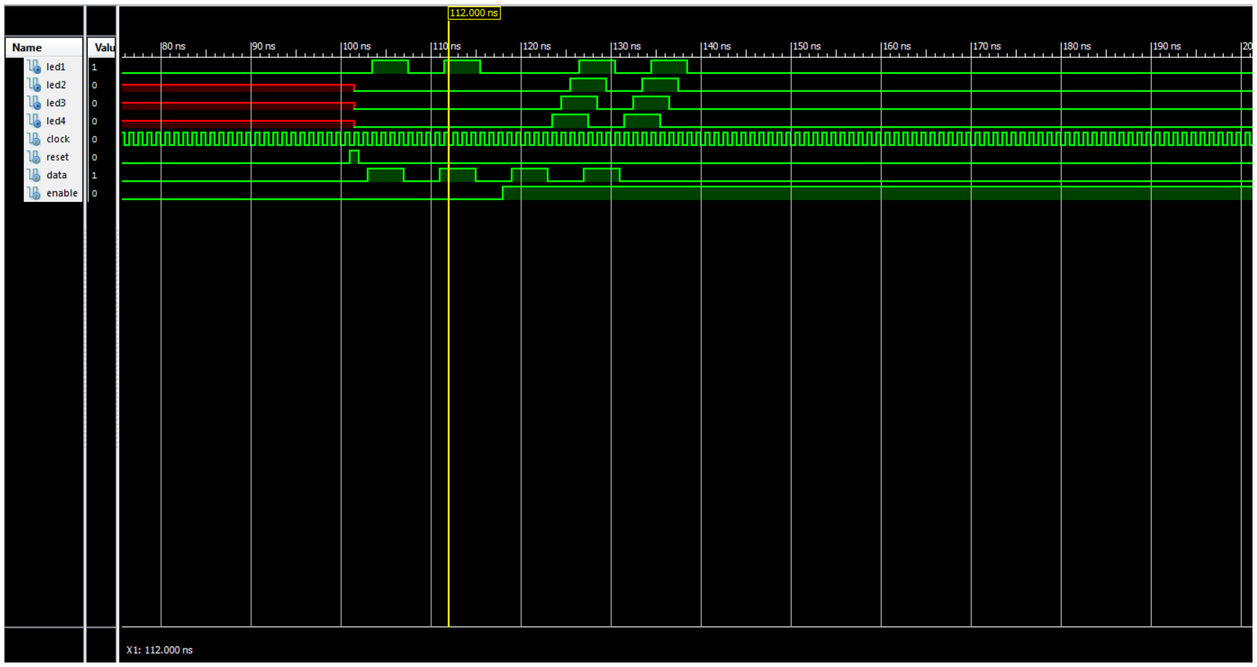

Figure 4: ModelSim Output

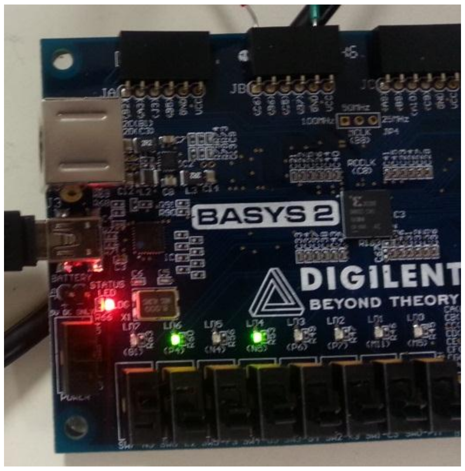
Figure 5: LEDs With Correct Shifting Scheme

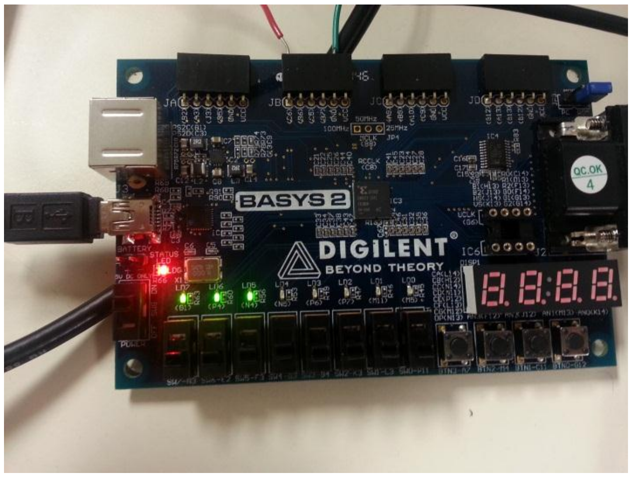
Figure 6: LEDs With Incorrect Shifting Scheme

Analysis:

Note: Analysis section should be written in full paragraphs with complete sentences. The analysis is NOT a question and answer format but MUST include answers to all questions in the lab.

Behavioral Verilog indices keep track of which input is being operated on in a shift register. For this lab, we created a shift register that was eight bits in size, half of which was shown on four LEDs. We set these LEDs to shift left to right and show the oldest bit value on the far right. However, being new to Verilog, we were not sure whether correct shifting was achieved with the data input on the left of the bracket statement, or on the right as is shown in lines 24 and 25 of **Figure 3**. After experimenting with the switched inputs, we found that the *enable == 1* case had the desired shift register effect. This implies that Behavioral Verilog assigns registers much the same way that we write binary, with the MSB to the right of the bracket assignment. Also, because it took a long time for the 1's and 0's that we toggled on the switch to appear on the LEDs, and because the far right LED is at the 0 index, we can deduce that the 0 index of the shift register is the "oldest" bit of the sequence.

To test this index scheme, we first connected the clock signal from the function generator to the Basys2 Board. We expected results much like our ModelSim simulation, in which when *enable* was set to 0, only the first bit of the register changed, and when *enable* was set to 1, the correct shifting scheme occurred in all bits. After connecting the function generator and ground, we set the frequency of the waveform to be 1Hz, the voltage to be 0-3.3V, and got to toggling the data switch. As we saw, the data only shifted further than the first bit when the *enable* switch was set to 1. **Figure 5** shows our LED sequence that shifted through the register successfully, and **Figure 6** shows that only the last bit in the register changed when *enable* was left at 0.

Conclusion:

Note: Conclusions should include a short summary of the purpose of the lab as well as any unexpected results that may have come about during lab.

In conclusion, this lab familiarized us with basic Behavioral Verilog syntax, as well as default ordering of bits and default bit concatenation schemes. One thing we did not expect, however, was that the shifting did not occur at all when *enable* was left at 0. We had anticipated that reordering the variables inside the brackets would simply make the shifting change directions, but instead it caused only one bit to be changed every clock cycle. This lab can be considered a success!