LABORATORY MANUAL

SC1005 : Digital Logic

[Location: Hardware Laboratory 3, N4-B1a-05]

Experiment 5:

Behavioural Verilog

COMPUTER ENGINEERING COURSE
COMPUTER SCIENCE COURSE

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

SC1005 Lab 5: Behavioural Verilog

Objective

To introduce the design approach for synchronous circuits using Behavioural Verilog. To reinforce the combinational and structural Verilog techniques you have developed in previous labs.

Equipment

As in Lab 3, we will use the Basys 3 FPGA board with an Artix 7 xc7a35tcpg236-1 part. You should also have to hand your Lab 3 & 4 manuals, the Basys 3 Reference manual and a copy of the lecture notes on Verilog.

Part 1 (Note: Part 1 can be done pre-Lab, using the virtual workstations at vasce.ntu.edu.sg)

In this lab, we will display 16 hexadecimal characters, by scrolling then from right to left across the four 7-segment displays on the Basys 3 board. We will display the sequence "_ _ C O F F E E _I S _ G O O D", where a "_" will be replaced by a space. We will use the sequence "_ _ C 0 F F E E _ 1 5 _ 9 0 0 d", which is close.

1. Start Vivado and from *Quick Start* create a new project called Lab5. Remember, select the project location given to you by the lab technician. Select *RTL Project* and tick the *Do not specify sources at this time* box (we will add them later). Select the **xc7a35tcpg236-1** part corresponding to the Basys 3 FPGA board. Click Finish to create the project.
2. Copy all the Verilog files from NTULearn zip file and place them in the same project location specified above (e.g. project_location/Lab5). Check the file extensions (e.g. *name.v*, *name.xdc*) are not modified.
3. Add the 4 files to your project
4. Open *slow_clkgen.v* and examine the code. This code uses a counter to divide the 100MHz system clock (*clk*) by 2^{25} to give a frequency of about 3Hz (that is, the period is ~1/3s). Firstly, a 25-bit register called *counter* is defined. It must be defined as type *reg* as we are assigning to it using a non-blocking assignment from inside an **always** block (a synchronous **always** block). The **always** block is triggered by the rising edge of the clock (**posedge** clk). If reset (*rst*) is high, the counter resets to all zeros, else it increments by 1. When it reaches all ones, it rolls over to all zeros and re-starts counting up. Then the MSB of *counter* (the bit changing the slowest) is assigned to the output (*clk_out*). Close *slow_clkgen.v*.
5. Next, open *seg7_driver.v*. The code inside seg7_driver is basically the same as the code you developed in Lab 4 (*segtoggle.v*). The major difference is that it inputs a 16-bit value, comprising four x 4-bit segment values (one for each of the 4 displays), and a 4-bit anode driver that turns the corresponding segment ON or OFF. See the top of the file for a description of each of the inputs and outputs. As described in Lab 4, the module uses a 20-bit counter (*count*) to slow down the 100MHz system clock to approximately 95Hz such that each digit is active low for 2.6ms with a refresh every 10.5ms. This is achieved by using the two MSBs (*count*[19:18]) as the input to a **case** statement, where the anode driver output (*anode_L*) is selected and the corresponding 4-bit value (*selnum*) is assigned. This value is then converted into the respective active low 7 segment format (*seg_L*) in the last **case** statement.
6. Before you close *seg7_driver.v*, it needs to be modified as:
   a. **You need to enter your Bench Number, as in Lab 4**. See after "You should NOT modify the code …".
   b. Examine the 7-segment code (at the bottom of *seg7_driver.v*). Note that to implement a space (NO segments illuminated), we just need to invoke the default assignment. But we already have all 16 hexadecimal digits (0 to F) defined, so there is no way to get to the default assignment. Instead, we perform a very rough modification to force a space. Firstly, pick an alphabetical character which is NOT in the above sequence. The first one is "A", so in *seg7_driver.v*, comment out the entry for "A" (*"// 4'd10 : seg_L = 7'b000_1000;"*) in the bottom **always @*** block. Then, if *selnum* is "A", it will go to the default statement at the bottom of the case block, which has all segments turned OFF. Thus, the new 16 character sequence we want is "AAC0FFEEA15A900D". Save and close *seg7_driver.v*.
7. Lastly, open *Lab5_top.v* and *Lab5_top.xdc*. *Lab5_top.v* implements the system shown in figure 1. You still need to instantiate the *slow_clkgen* and *scroll* modules, as well as implement the *scroll* module which itself instantiates four *convert* modules. You will design the *scroll* and *convert* modules in part 2. Next look at *Lab5_top.xdc*. It maps *clk, rst, seg_L* and *anode_L* to the FPGA pins corresponding to the off-FPGA components we are using. It is very similar to what you used in Lab 4.

Part 2

In this part we will implement *scroll.v* and *convert.v.* The module *scroll.v* will consist of a 4-bit counter which will count from 0 to 15 and repeat, while *convert.v* will map this 4-bit number to the new sequence defined in item 6, above. E.g. "0" & "1" will map to "A", "2" will map to "C", …, and "15" will map to "D", etc.

1. Create a new Verilog module. Select **File** > **Add Sources** from the menu (or click the + icon in the Sources window). Make sure the File type is Verilog, then enter *convert* as the File name and click **OK**, then **Finish**. Give the module a single input called *in*, select bus, and set the MSB to 3. Add an output called *out*, select bus, and again set the MSB to 3. This creates a module with a 4-bit input and a 4-bit output. Click OK. Double click on *convert.v* (in the Sources window inside the PROJECT MANAGER window) and you should see a bare module declaration with the inputs and outputs you declared.
2. You should now populate the *convert.v* module with a single **always @ *** block, that contains a case statement, switched on *in*. Then, for each case (0 to 15), make an assignment to the new sequence "AAC0FFEEA15A900D". The first statement in the case body should be "4'd0 : out = 4'hA;". Set the default to "A" (segments not lit). End the case statement and the module, then save and correct any syntax errors (they usually show up on the line below as wavy red underlines, like in MS Word).
3. Next, create a new Verilog module called *scroll.v*. It should have two 1-bit inputs called *clk* and *rst*, and a single 16-bit ([15:0]) output called *display*.
4. Add a 4-bit up counter (*count*, which counts 0 to 15), triggered on the positive edge of *clk* with active high synchronous reset (*rst*). See the **always** block in *slow_clkgen.v* for a counter example.
5. Then, we need to generate the 4 active digits that will be displayed on the four 7-segment displays. Declare a **wire**, with four 4-bit values called *a, b, c* and *d*, as "**wire** [3:0] a, b, c, d;". Using a continuous **assign** statement, assign the current *count* value to *a*. Using three other continuous **assign** statements, assign *count+1* to *b*, *count+2* to *c* and *count+3* to *d*. So, if *count* is "0", *a*=0, *b*=1, *c*=2 and *d*=3. If *count* is '14", *a*=14, *b*=15, *c*=0 and *d*=1, etc. That is, as count increments, we have a sequence of 4 sequential values starting with the current value of count.
6. Then, we need to instantiate four *convert* modules, with inputs *a, b, c* and *d*, respectively. Each of the instantiated modules will output 4 bits of the 16-bit *display* output, with the *a* value corresponding to the 4 MSBs (*display* [15:12]) and the *d* module corresponding to the 4 LSBs (*display* [3:0]). Save and then correct any syntax errors.
7. Lastly, open *Lab5_top.v* and instantiate the missing *slow_clkgen* and *scroll* modules as per figure 1.
8. Run Synthesis. Correct any errors and warnings (you can ignore the warning [Constraints 18-5210] No constraints selected for write.). Then Run Implementation. You should make sure that the project Summary window is active so that you can see the status of the various processes.
9. Then Generate Bitstream and program the device. Open the hardware Manager. Open Target and select Auto Connect. Verify the Digilent device is shown. Then Program Device and select xc7a35t_0.
10. Test the circuit. You should see 4 of the characters in the sequence slowly scrolling from right to left.

Part 3 (optional)

Change the sequence to display "_ _ C O F F E E _ I S _ g o o d". To do this you will need to change one of the unused case lines in the bottom **case** block of seg7_driver.v (perhaps 'B') to display lower case "o", instead of "0", and them also modify *convert.v*.

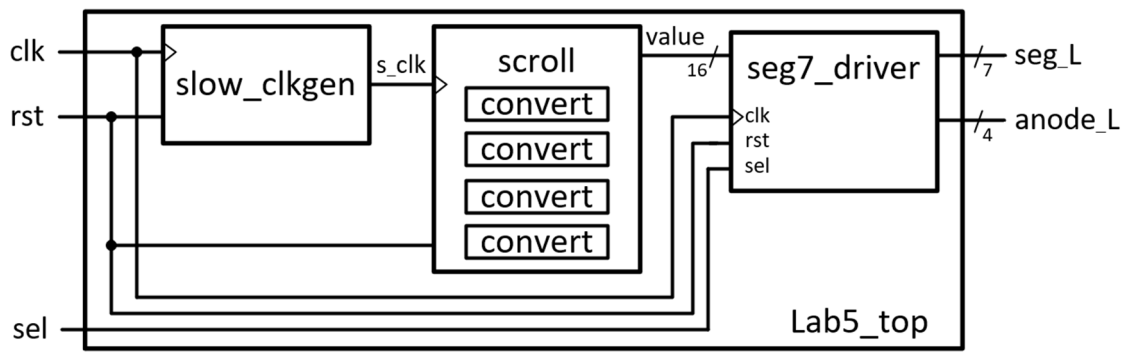

Figure 1

Note: The following system warnings can be safely ignored

Synthesis (Part 2):

WARNING: [Constraints 18-5210] No constraints selected for write.

Implementation (Part 2):

Place Design: [Place 46-29] place_design is not in timing mode. Skip physical synthesis in placer.

Bitstream (Part 2):

There should be NO warnings.

Notes:

The warnings (above) are system warnings and can be ignored.

ANY other errors or warnings probably mean that you have made a coding/syntax mistake, and you will need to fix these.