Fall 2018 / EC311
Lab 4
Recommended Due Date: November 6, 2018
Actual Due Date: November 30, 2018.

1. Goals

In this lab you will design a 7-segment display driver. The driver will be used to display the results of the counter from Lab 3 on the FPGA boards 7-segment display. The design will have two modes:

1. Automatic counting: the counter should increment automatically with a step of 1 second.
2. Manual counting: the counter should increment on the push of a button.

You will need a control switch and a multiplexer to enable selection and usage of one of the two modes. Your design will use clock dividers to create multiple clock domains:

1. Slow clock (1Hz): to use as an increment signal in the automatic counting mode.
2. Fast clock (1kHz): to iterate through different 7-Segment displays.
3. Normal clock (100MHz): to control everything else (debouncer, clock divider, etc).

1.1. Lab 4 I/O Requirements

The top level module must have inputs named clock, reset, mode_select and increment. The top level module must output a 4-bit digit_select signal to select the digit to light up (only 1 can be lit at a time) and the 7-bit encoded digit display value. The increment and reset inputs must be mapped to push buttons. The mode select must be mapped to a switch. The encoded digit value must be displayed on the board's hex digit display.

1.2. Components

Your design should have 6 modules:

1. seven_segment_decoder - A 7-Segment Display is composed of 7 LEDs (marked CA, CB, ..., CG on your board). The four 7-segment displays on the Nexys-3 board are common anode, meaning that the LEDs will light up when you set the input to 0 and turn off when you set the input to 1 (active low inputs). The 4 digits on the display are controlled by the corresponding active low AN line (AN0, ..., AN3). This AN line is named display_select in the block diagram below. Only one AN line should be set to 0 at a time, such that only one of the four 7-segment displays is ON at any given time. If several display_select bits are low at the same time, multiple 7-segment displays will be lit up with the current encoded value because all displays are connected to the same 7 encoded outputs (named "seven" in the block diagram).

   For this module you must implement a binary to 7-segment display decoder. The input is a 4-bit binary number (representing one hexadecimal digit), and the output is the seven segments of the display. Your Verilog code can include a case statement with one-to-one mapping of input number to output pattern.

   For more detail on the 7-Segment Display, refer to pages 18 and 19 of the Nexus-3 Reference Manual. You will need to consult the manual to determine which FPGA pins are connected to the hex display.

2. counter16 - We would like to use the four 7-segment displays to display a 4-digit hexadecimal number, which is 16-bit binary number. That is, the 4 least significant bits will be displayed on the right-most 7-segment display, and the 4 most significant bits will be displayed on the left-most 7-segment display. The number to be displayed will be the output of the counter from the previous lab. Modify your counter so that it will now output a 16-bit number. This should be a simple change.

3. display_control - The binary to 7-segment display decoder can drive one 7-segment display at a time, while we would like to display a different digit on each of the four 7-segment displays. We will use the decoder to drive each of the four displays in a round-robin fashion. To do so, we will make the displays flash each digit very fast (1kHz) such that the user would not notice that the digits are not being displayed concurrently.

   In order to do this, you need to apply the codes 4'b1110, 4'b1101, 4'b1011, and 4'b0111 to the display_select (AN) line. This will make sure that only one digit (one 7-segment display) is ON at a time. Make sure to synchronize your digit value output with the display alternation.

4. clock_divider - A clock divider receives a clock signal as input, and using an internal counter divides the clock by some number in order to achieve the desired frequency and duty cycle. If the content of internal counter is greater than some number (you will need to figure out what this number is), then the output should be 1, and otherwise the output should be 0. Note that your duty cycle for the 1Hz clock may not need to be 50% depending on your implementation.

   Your design will have two clock dividers. The first one will convert the 100MHz clock into a 1Hz clock. The second will convert the 100MHz clock into a 1kHz clock. You may use two separate modules for each clock divider or you may use Verilog parameters to create a parameterized module (you will have to learn about parameters on your own but the TAs will offer some help).

5. debouncer - This module can be reused from Lab 3 without any modification.

6. lab4 - The Top Level module should instantiate display_control, counter16, clock_divider, seven_segment_decoder and debouncer modules.

   Connect the modules as shown in Figure 1. You must demonstrate the working lab4 module on the FPGA to the TA. Refer to Lab B and the schematic capture tutorial for directions on implementing the design on the FPGA. The pin name for the FPGA clock input is v10. Refer to the Nexus-3 Reference Manual to determine appropriate pin assignments for the remaining I/O pins.

   Note that your design must have a global reset input (omitted from the figure for clarity), which will reset all modules in the design.

   Note that your simulations do not need to use clock dividers. Simulations with 1Hz and 1kHz clocks would be very long.

2. Deliverables

You need to show your design hierarchy (under Design → Implementation window), your Verilog code, and FPGA implementation (on the FPGA board) to the TA. The TAs do not need to check off your test benches before you move on to other modules or the FPGA implementation.

Your design must abide by the following requirements:

1. Use multi-bit vector/bus/array (just different ways of naming) instead of 1-bit signals for all multi bit signals.

2. Follow the input/output specifications.

3. Create test benches for all modules. Modules re-used from Lab 3 can re-use testbenches from Lab 3 (you may have to make minor changes). The test benches should be thorough but do not need to provide complete coverage. Complete test coverage in sequential designs can be difficult to achieve.

4. Be hierarchical and use a separate file for each module. Test each module and demonstrate that each works as a standalone unit. Generally, it is good engineering practice to test submodules before going to the next level. You must turn in a test bench for each module.

You must turn in a lab report describing what you did in this lab, how you did it and what happened when you did it. The lab report should be broken up into these sections: Objective, Methodology, Observation (can include screenshots timing diagrams, schematics, etc.) and Conclusion. Your report should contain enough detail such that any other engineer can replicate your experiments after reading your lab report. Please include all the required details in your lab report. The lab report should be 2-3 pages long with figures and diagrams. Your report must be written in paragraph form with complete sentences.

You must submit a PDF to TurnItIn. The PDF must contain your lab report and all of your Verilog code (including test benches).

3. Due date

Your lab report and lab must be checked off by a TA and submitted to TurnItIn by November 30.

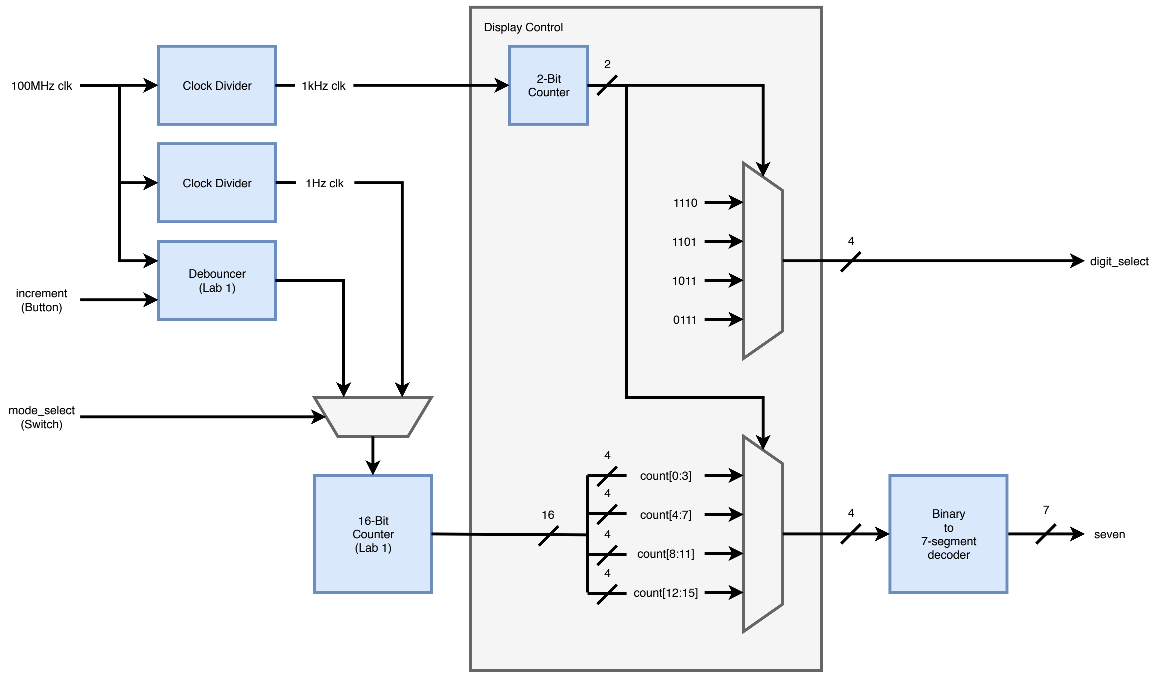

Figure 1: Lab 4 top level module. Note that reset is not shown on the diagram but it must be included in your design.