Lab 2 Assignment

ECE 525.442 FPGA Design using VHDL

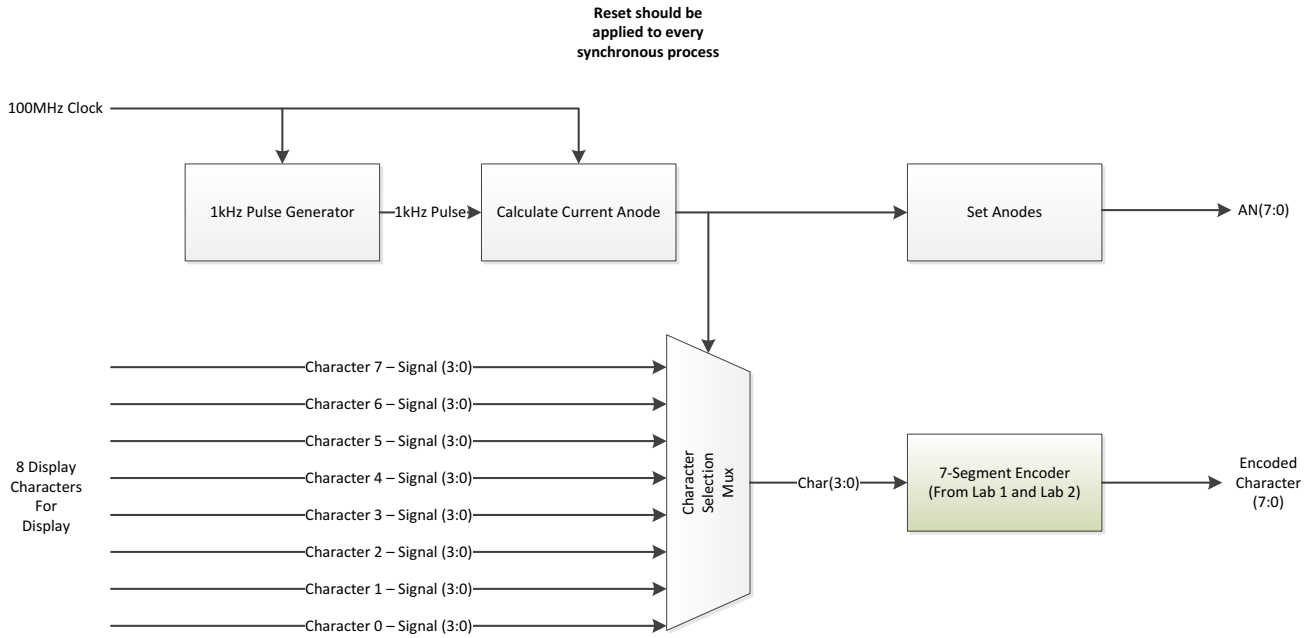
Seven-Segment Display Controller using Counters

This lab assignment builds on Lab 1 and Lab 2 to create a fully-functional controller for the eight-digit seven-segment display available on the Nexys-4 board. The seven segment display controller needs to be captured in an entity/architecture pair so that you can easily use it in future labs. This lab is designed to test your seven-segment display controller to make sure it works properly.

As described in the User's Guide for your Nexys-4 board, the board designers saved FPGA I/O pins by wiring the eight seven-segment digits to the same cathodes. Writing eight separate digits for "simultaneous" display is achieved by time-multiplexing each digit fast enough so that our eyes view ALL eight of the digits as illuminated while displaying the correct value. To do this, we continuously sweep each of anodes, see section 10.1 of the Nexy's 4 Board Reference Manual for more information on time multiplexing. The seven-segment controller requires a clock, reset, and eight characters (4 bits each) as inputs. The outputs of the controller are the eight cathodes (seven segments plus the decimal point) and the eight anode signals.

Task 1

Design a driver that displays different digits on the 7-segment display. For this task, create an entity called `seg7_controller` which requires the master 100MHz clock, an active high reset, and eight characters as 4-bit inputs each. The outputs of the controller are the encoded character as the cathodes and the anode signal. To initially test this task, drive in fixed values for the characters. Each character should be represented as a std_logic_vector(3 downto 0). Below is a block diagram of a possible implementation for the seven-segment controller.



Task 2

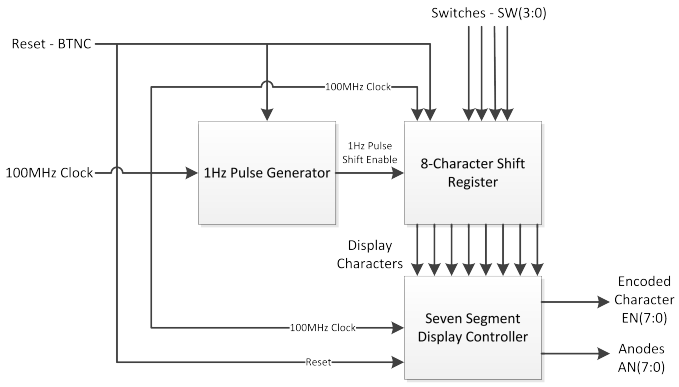
Create a new entity/architecture pair called `lab2_top` in a file named *lab2_top.vhd*. This new entity will instantiate the seg7_controller component and will test it for proper operation. The entity for `lab2_top` should look like the following:

```
entity lab2_top is port (
            --Clock
           CLK100MHZ : in STD_LOGIC;

            --Push Buttons
           BTNC : in STD_LOGIC;

            --Switches (16 Switches)
           SW : in STD_LOGIC_VECTOR (15 downto 0);

            --LEDs (16 LEDs)
           LED : out STD_LOGIC_VECTOR (15 downto 0);

            --Seg7 Display Signals
           SEG7_CATH : out STD_LOGIC_VECTOR (7 downto 0);
           AN : out STD_LOGIC_VECTOR (7 downto 0));
end lab2_top;
```

The top level should test your seven-segment driver as follows. BTNC is an asynchronous reset, and when pressed, the seven-segment display resets all the digits to zero. During normal operation, the values on the seven-segment display are shifted left at a rate of 1Hz, and the value currently on the SW(3 downto 0) becomes visible on the rightmost seven-segment display and remains there for at least one second before shifting to the left. As you change the values on the switches, you should see the characters slowly "scroll" to the left at a rate of 1Hz. An example scenario is shown at the end of this lab description.

The value on the SW(3 downto 0) starts scrolling from the rightmost digit as soon as BTNC is released (from reset) or a new value is selected on the SW(3 downto 0). The value should continue to scroll to the left at a rate of 1Hz until it gets to the leftmost digit.

A block diagram of how this may be implemented in VHDL is shown below.



Your design needs to store the eight current seven-segment digits (unencoded) and needs to shift them at a rate of 1 Hz. Therefore, the block "8-Character Shift Register" is a 4-bit wide shift register with depth eight and a 1Hz enable. The 1Hz enable signal is derived from the 100MHz clock. Make sure to trigger the flip flops using the 100 MHz clock and use the 1Hz enable to activate the flip flops at a rate

of 1Hz. It is considered incorrect design practice to design the flip flops using the rising edge of the 1Hz pulse.

Task 3

Using the constraints editor (or a text editor) set the FPGA IO in the UCF to the appropriate cathodes, anodes, switches, pushbutton and clock from your `lab2_top` component. Synthesize, generate a programming file, and upload this file to the board using Xilinx Hardware Manager. Test your design to verify that it works.

Example Flow:

Time	Switches (3:0)	Display Output
Reset	N/A	0000 0000
1 second	0001	0000 0001
2 seconds	0011	0000 0013
3 seconds	0111	0000 0137
4 seconds	1111	0000 137F
5 seconds	1111	0001 37FF
6 seconds	1111	0013 7FFF
7 seconds	1111	0137 FFFF
8 seconds	1111	137F FFFF
9 seconds	1111	37FF FFFF
10 seconds	1111	7FFF FFFF
>11 seconds	1111	FFFF FFFF

To Submit

Files to Submit (minimum), zipped together as Lab2_<last_name>.zip:

- VHD, XDC, BIT files for Lab 2
- Complete Synthesis Results
- Report summarizing the resource utilization for Lab 2 and overall design principles

- This report should have good estimates of resource utilization with explanation for the estimate which will be compared to the actual synthesis results in a table format
- Explain any discrepancies between estimate and actual resource utilizations

An example table for resource utilization could be as follows (with appropriate values):

Resources	**Estimate Used**	**Actual Used**
Slice Registers		
Register Latches		
Clock Buffers		
Number of IOs		

The instructor will focus on accurate estimates for the number of clock buffers, FPGA IO and slice registers, or flip flops. The FPGA IO can be estimated by counting the number of bits in your top level input and output ports. This includes the clock input port. The flip flops can be accurately estimated by counting the number of bits on the left hand side of an assignment in a process triggered by a clock edge. If your estimate of flip flops is different than the actual flip flops reported by the synthesis tools, you will need to provide a detailed explanation of where that discrepancy lies. If this part is not clear to you, please post questions to the forum and we can collaboratively discuss this topic in detail.

Recommendation: First you complete the design and test of the seven-segment display controller described in Task 1. To test the design of this component, instantiate the seven-segment display controller in the `lap2_top` and drive the eight character inputs of the components using constant 4-bit values. Make sure you can clearly display eight different hexadecimal characters on the seven segment display (no scrolling affect yet). For example, try to display 1, 3, 5, 7, 9, B, D, F simultaneously on digits 0 to 7, respectively. Once this part of the design is working, you can implement the scrolling effect, which is task 2 of the lab assignment.

Note: The grading sheet on the next page provides a good overview of the key points of the design and what the instructor will look for when grading. Please note that this is a guide and isn't comprehensive. Please read over the lab assignment carefully to verify all the proper documents are submitted.

Grading Sheet

Lab 2: Seven-Segment Display controller using counters

Lab 2

Student:

Grade	Out of	Notes
	100	
	10	DEMO: All hexadecimal characters 0-F appear on display when selected by switches
	20	DEMO: 7-segment display capable of clearly displaying 8 separate characters
	10	DEMO: Switch values SW3-SW0 shifted from right to left into display at ~1Hz
	10	DEMO: First character remains visible on rightmost digit for at least one second before scrolling to the left
	10	DEMO: brief press and release of BTNC resets displays to all 0's and then restarts with shifting of value selected by switches
	20	Well documented, commented and appropriate VHDL Code
	20	Synthesis report with device resource utilization summary -- Include utilization report by synthesis tools and discuss if results match your estimates