

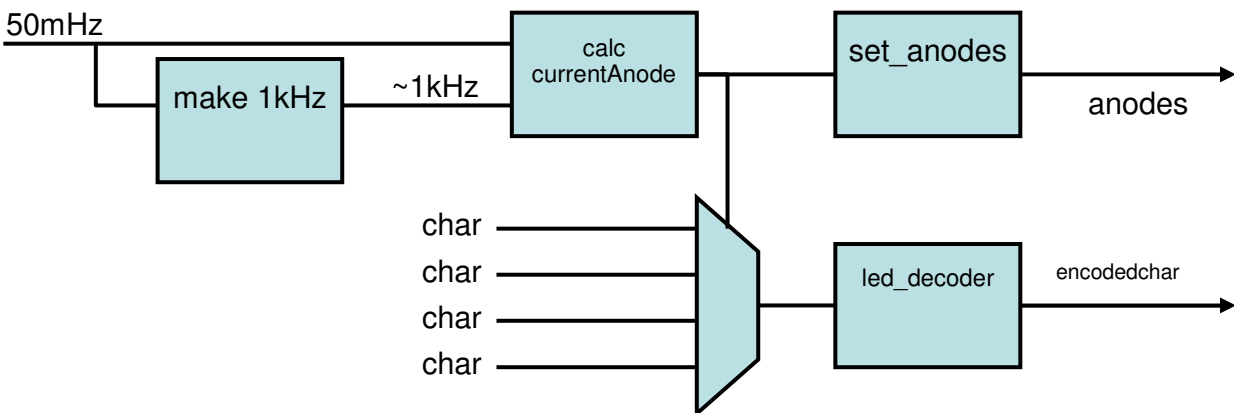
Full Multiplexed Seven-Segment Display Controller

This assignment will build off of Lab 1 to create a fully-functional controller for the four seven segment displays available on the Nexys2 board.

Task 1

As described in the User's Guide for our evaluation board, the board designers saved I/O pins by wiring the four seven-segment displays to the same control lines. Writing four separate characters for "simultaneous" display is achieved by time-multiplexing which seven-segment display is being driven fast enough so that our eye views all four of the seven-segments as on and displaying the correct value. To do this, we continuously sweep which of the four seven segment display anodes are low (see Figure 10 of the Nexys2 Board Reference Manual). Our seven-segment controller is going to take a clock and four characters (4-bit each) as inputs, and is going to write the seven-segment control signals as well as the four anode signals to display all four characters simultaneously.

To do this, create an entity/architecture pair called **seg7_driver** in a file named **seg7_driver.vhd**. This entity should have six inputs: *clk50*, *rst*, *char0*, *char1*, *char2*, *char3* and 2 outputs: *anodes*, and *encodedChar*. The reset should be asynchronous. The anodes should be swept at approximately a kilohertz. Also, your led decoder should decode all possible values (from 0, 1, 2, ...A, B, C, D, E, F), which will require a modification of Assignment 1. A block diagram of a possible implementation of the **seg7_driver** is given below (yours does not have to follow this):



Task 2

Create a new entity/architecture pair called **lab2_top** in a file named **lab2_top.vhd**. This new entity will make use of the **seg7_driver**. The entity for **lab2_top** should look like this:

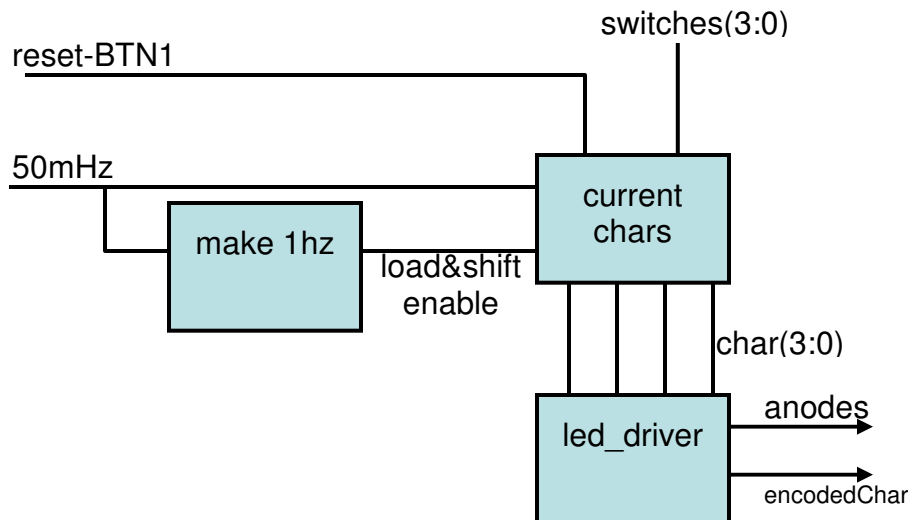
```

entity lab2_top is
port (
  clk50 : in std_logic;
  BTN1 : in std_logic;
  sliderSwitches : in std_logic_vector(3 downto 0);
  Seg7 : out std_logic_vector(6 downto 0);
  anodes : out std_logic_vector(3 downto 0) );

```

Your top level will test your seven-segment driver as follows. BTN1 is an asynchronous reset, and when pressed, the seven-segment display will display all zeros. During normal operation, the values on the seven-segment display will be shifted left, and the value currently on the sliderSwitches(3 downto 0) will be visible on the rightmost seven-segment display. So, as you change the values on the switches you will see the characters slowly “scroll” left.

A small block diagram of how this might be implemented is below (yours does not have to follow this):



Your design will need to store the four current seven-segment characters (unencoded) and will need to shift them. Therefore, the block “current chars” can be thought of as a four bit wide shift-register (4 deep) with an enable signal that is the 1Hz signal divided down from the 50MHz clock.

Task 3

Using the constraints editor (or a text editor) set the pin assignments for the ports of lab2_top to the appropriate Xilinx pin numbers as shown in the document. Synthesize, generate a programming file, and upload this file to the board using IMPACT. Test your design to verify that it works.

To Turn In

- 1) Readable paper copies of appropriately commented VHDL for lab2_top.vhd and seg7_driver.vhd.
- 2) The zipped project directory (lab2_yourlastname.zip) and the bit file (lab2_yourlastname.bit) via the submission link on the webpage. This must be turned in by 4:25 on the day of class.
- 3) Bring your board with the assignment loaded into the non-volatile memory so that all that is required to demo the assignment is simply powering the board. An instructor will check for correct operation during the lab period

Grading Sheet

20 points

DEMO: 7 segment display capable of displaying 4 separate characters.

20 points

DEMO: Switch values shifted into display at approximately 1 Hz rate.

10 points

DEMO: Brief press and release of BTN1 resets display to all 0's.

15 points

DEMO: PROM programmed correctly to allow power-on operation without manual configuration.

35 points

Quality of VHDL code and hardware design. Graded from printed copies submitted in class.