ELEX 2117 : Digital Techniques 2
2020 Winter Term

Introduction to VHDL

Version 2: corrected purpose of Appendix A. Version 3: fixed labels on state transition diagram.

Introduction

In this lab you will use VHDL to design logic that displays the 8 successive digits of your student number on a 4-bit binary LED display.

For this lab you will need the same components as for the previous lab plus an additional pushbutton switch and three additional pin-header jumpers.

Pre-Lab

Write a VHDL description that has:

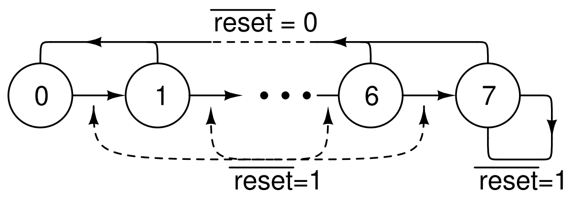
- an **`std_logic`** clock input,
- an active-low **`std_logic`** reset input whose name ends with the letter 'n' (e.g. **`reset_n`** or **`RSTN`**),
- a 4-bit **`std_logic_vector`** output

Your design should operate as follows:

- The output should change only on the rising edge of the clock.
- If the reset input is asserted (i.e. low) at the rising edge of the clock then the first digit of your student number should be displayed in binary on the led outputs.
- If the reset input is not asserted at the rising edge of the clock then the next digit digit of your student number should be displayed. Your design should not advance past the last digit.

Draw the state transition diagram and write VHDL source code for your counter and bring hard-copies to the lab.

Your state transition table should have conditions on each transition:


$\overline{\text{reset}} = 0$
0
1
6
7
$\overline{\text{reset}}$=1
$\overline{\text{reset}}$=1

Note that since your student number could have duplicates of the same digit, your states should not be the displayed values. Instead the state should be the digit position being displayed (0 through 7). Use a selected assignment to convert from the digit position to the displayed digit value.

If you've installed Quartus on your PC you can also follow the procedure below, complete the lab at home and bring your circuit to the lab to have it checked.

Procedure

Follow the general procedure in Appendix A of the previous lab to create a project, compile it, and configure your CPLD. However, note the following additional instructions specific to this lab...

Add Files to Project

When adding files to the project (step 4), download the **`debounce_pkg.vhd`** file from the course web site and copy it to your project folder. Add it to your project (**Project** > **Add/Remove Files in Project...**, **File name:** > ..., select the file, > **Add** > **OK**).

VHDL Entry

When adding files to the project (step 4), follow the procedure in Appendix A to create the VHDL source file for your design. You'll also need to add a **`debounce`** component as described next.

Debouncing

As in the previous lab, you'll need to debounce the clock switch input [1]. In your VHDL file include:

```
-- before the architecture:
use work.debounce_pkg.all ;

-- within the architecture:
   debounce1: debounce port map ( clk_in, clk ) ;
```

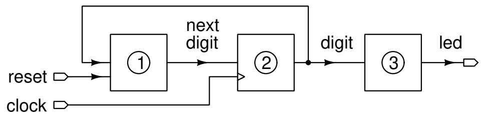
[1]But not the reset input since it is level-sensitive rather than edge-sensitive.

where **clk_in** and **clk** are the signal names you used for the un-debounced and debounced clock signals respectively.

The **use** clause will make the **debounce** component available. The component instantiation statement will include an instance of the **debounce** component in your design, name the instance **debounce1**, and connect **clk_in** to the **debounce** component's input port and **clk** to its output port (assuming you have signals with these names).

Hints

- Draw a block diagram of your design before you begin coding. For this lab it might be:



  where block ① computes the next digit position based on the value of the current digit position and the reset input (e.g. using a conditional assignment); block ② stores the current digit position value (e.g. by creating a register); and block ③ computes the output digit value based on the current digit position (e.g. by using a selected assignment).

- Declare signals within the architecture for the current and next digit values (e.g. **dig**, **dig_next**) and use a conditional assignment to set the value of the next digit based on the current digit. The internal signal declarations are placed just before the **architecture**'s **begin** keyword.

- You will also need to declare an internal signal within the architecture for the debounced clock (the clock declared in the entity's port is the un-debounced signal).

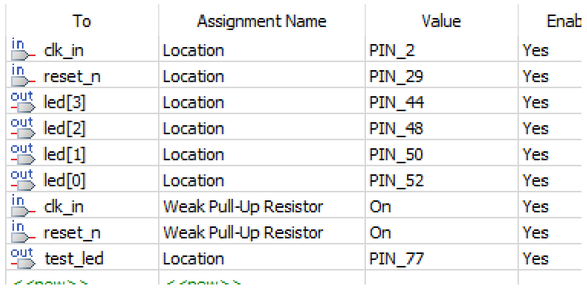
- You can use hexadecimal literals **x"5"** instead of binary ones (**"0101"**)

- Certain useful features of the 2008 version of VHDL can be enabled by checking: Assignments > Settings > Analysis and Synthesis Settings > VHDL Input > VHDL Version > VHDL 2008 One of these features is literals of any bit width and decimal base (e.g. **3d"5"** for a 3-bit decimal value).

Pin Assignments

When assigning pins also configure internal pull-up resistors on pins 2 and 29, the clock and reset input pins, following the instructions in the previous lab.

You should end up with the following assignments, possibly using different signal names[2]:

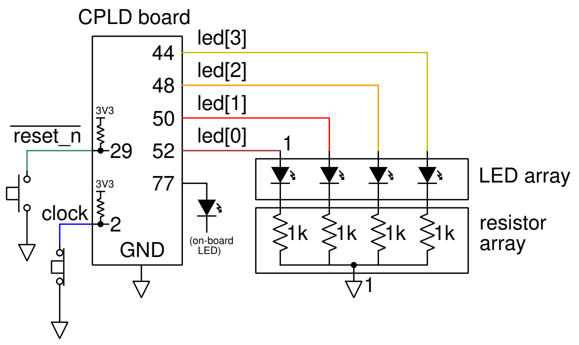


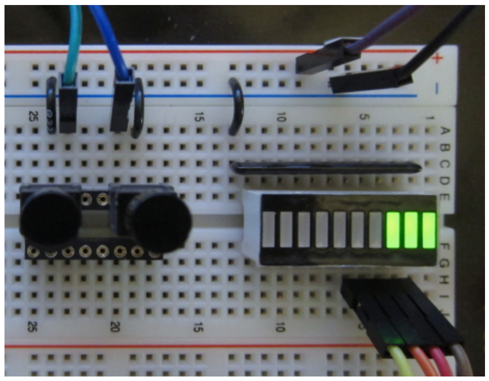
To	Assignment Name	Value	Enab
in clk_in	Location	PIN_2	Yes
in reset_n	Location	PIN_29	Yes
out led[3]	Location	PIN_44	Yes
out led[2]	Location	PIN_48	Yes
out led[1]	Location	PIN_50	Yes
out led[0]	Location	PIN_52	Yes
in clk_in	Weak Pull-Up Resistor	On	Yes
in reset_n	Weak Pull-Up Resistor	On	Yes
out test_led	Location	PIN_77	Yes

CPLD I/O

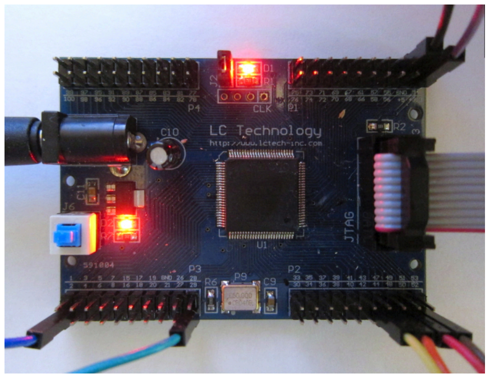
After the CPLD is programmed, the schematic and photos below show how it can be connected to switches for the clock and reset inputs, and to four LEDs that display the state.

The black wires are ground, blue and green are the clock and reset respectively, red through yellow are the color-coded **led** outputs:





[2]The **test_led** output is for troubleshooting.



Configure internal pull-up resistors on pins 2 and 29 (the clock and reset input pins). Use the N.C. (normally closed) pushbutton switch contacts to generate a positive clock edge each time the clock button is pressed. Use the N.O. (normally open) contacts to assert the reset input when the reset button is pushed.

Connect four LEDs with 1 kΩ current-limiting resistors to CPLD pins 50 and 52 as in the previous lab. If you have them available, use the LED and resistor arrays to minimize wiring and chance of errors.

The schematic above also connects an output (named `test_led`) to the LED on your CPLD board (pin 77) so you can verify the operation of your clock.

Demonstration

Pressing the pushbutton should make the on-board LED light. On each press (rising clock edge) the count value should change to the next digit of your student number. Holding the reset button and pressing the block button should display the first digit of your student number.

Show your working counter to the lab instructor. After your demonstration the instructor will ask you to modify the behaviour of your design to make sure you understand how it works.

A VHDL Entry with Quartus

To create a new VHDL source (`.vhd`) file:

- select **File** > **New...** > **Design Files** / **VHDL File** > **OK**
- select **File** > **Save as...** and enter a file name. Use the project name as the file name and the name of the top-level entity (e.g. `lab3`, `lab3.vhd` and `lab3`)[3]
- the editor has syntax highlighting, keyword and signal/variable name completion and language templates (right-click > **Insert Template...**)

Optional Extensions

If you found this lab too easy, you can try adding[4] the following features (listed in order of increasing complexity):

- Define additional outputs to display the digits on one of your 7-segment LED displays. Use an additional selected assignment to covert the binary digit to seven active-low signals to drive the LED sgments (the displays are common-anode). Define additional outputs (e.g. **a** through **g**), assign pins and connect them to your display.
- Use two 7-segment displays and have the digits scroll through the display as you advance through the digits. You'll need an additional register and logic to load/clear it.
- Multiplex the two displays so you only need one set of pins (7 segments plus 2 anodes) to drive both of them. You can divide down the on-board 50 MHz clock available on pin 12.

[3]You can change the name of the top-level entity and the project files to be compiled in **Assignments** > **Settings**, but its easier to use the defaults.

[4]You must leave the required functionality intact.