Lab 03

Lab03 Lab03

Prep:

Prepare the Hardware:

- Plug in the power adapter.
- Connect the USB cable - leftmost port on the DE2-115 board to the Computer.
- Power ON. (did you hear the OS register the USB device?)

Preparing for ALL New Quartus Projects, this lab forward (always bring the FPGA board):

- Navigate or create (if not already) your 252 directory (folder) "*ENSC252*"
- Copy the file "*DE2_115_pin_assignments.csv"* (location: desktop shortcut "**DE2-115..SystemCD**" - "**DE2_115_tutorials**" - "**design_files**") to the 252 directory. Verify that this file is read-only. If not flag the file as READ-ONLY.
- You will need this file for every project throughout the semester.

Preparing a New Quartus Project:

- Create a new **project directory** (folder) inside "***ENSC252***". – call it "***Lab03****"*.
- Start Quartus II.
- New Project wizard.
  - Set the path to your project directory. Name the project **Lab03**
  - Choose Cyclone IV E – **EP4CE115F29C7**
  - Leave all other choices alone as you have in the previous lab
  - Once the project wizard is done, select ***Assignments->Import Assignments***
  - browse to the location of your copy of the file "***DE2_115_pin_assignments.csv****"* (good idea to place in your main 252 folder, with subfolders as your labs)

Every project must contain a Top-Level Entity.

- The Entity at the top level may contain instances (components) of lower-level Entities
- The Entity at the top-level may also contain statements that explicitly describe behaviour.
- The PORT Signals of the Top-Level Entity are the pin names of the FPGA. In order to prevent damage to the FPGA, Terasic has provided the file "***DE2_115_pin_assignments.csv***" that defines names for pins.

WE ONLY USE PIN NAMES IN THE TOP-LEVEL ENTITY.

- ✓ The port specification of the **top-level Entity** must contains **ONLY pin names**.
- ✓ **Do not** use pin names in lower level entities.
- ✓ The correct pin names can be found in the DE2-115 **user manual** on the CDROM. Alternatively you may also find the names of the pins in the .csv file you copied. If you look closely on the FPGA board, you will see the names of the peripherals (ex: SW0 labels the right-most switch on the board).

Part 1:

Objective: **To build a circuit that connects the 18 switches to the 18 RED LEDs.**

- Enter the design using the **schematic capture tool**.
  - ***File -> New -> Design Files.Block Diagram/Schematic File (bdf)***.
- save as "***Part1.bdf***" - verify that the file is stored in the project directory.
- To set the Entity contained in "***Part1.bdf***" as the **TOP-LEVEL** circuit.
  - right-click on "***Part1.bdf***" in the files tab of the Project Navigator Pane, select "Top Level Entity"
- To connect one switch to one LED.
  - Insert an input pin and an output pin.
  - Connect these pins with a wire.
  - Name these pins **SW[0]** and **LEDR[0]**.
- **Analyse/Elaborate** and **Synthesize** this simple circuit. **<ctrl>-k**
- Using the RTL circuit viewer, **Quickly Check** that the synthesis is correct.
  Tools -> Netlist Viewers -> RTL Viewer
- Connect four more switches to four LEDs. **Quickly Check** that the design entry is correct.

Part 1: ***continued***

- Now make Quartus fit the design inside a Cyclone IV FPGA. <ctrl>-l
  - ignore all the warnings.
  - The final design is contained inside a file called "part23.*sof*"
    Determine the location of this file. Determine what .sof stands for

- **Configure the FPGA**:
  And transfer the design in "part23.*sof*" to the FPGA over the USB cable:
  - *Tools -> Programmer.*
    (this procedure is called programming the FPGA. It would be better to refer to the procedure as configuring the FPGA.)
  - Check that the programmer can see your USB-Blaster interface. **If not** the click Hardware setup and select the **USB-Blaster** device (double click and ensure it's listed in "currently selected hardware").
  - Click the *Add File* button and browse to the location of your .sof file. It will likely be a file called "part23.*sof*" (or equivalent project name) in **output_files** directory.
  - Click Start.
- Test the circuit. Flick the switches. Is this circuit functioning as expected?
- To enter the remaining 13 connections as a bus.
  - insert one input pin and one output pin.
  - connect these pins using a bus connecting tool.
  - Name the pins SW[17..5] and LEDR[17..5] respectively.
- Synthesize, Quickly Check, Fit, Program and Test this final circuit.

Part 2:

Objective: **To build a circuit In VHDL - connect 18 switches to the 18 RED LEDs.**

- To **Enter** the design using **VHDL**.
  - Create a new folder in **LB03** called **part23**. Create a new project called part23, and follow directions as specified in the "Preparing a new Quartus project" section of the lab
  - ***File -> New -> Design Files.VHDL File***.
  - Save as "***Part2.vhd***" - verify that the file is stored in the project directory.
  - Enter the two context clause lines that precede all design units.
  - Declare an **ENTITY** called **Part2**
  - Declare the **PORT** signals for this entity

```
Port ( SW : in  std_logic_vector( 17 downto 0 );
     LEDR : out std_logic_vector( 17 downto 0 ) );
```

  - Write the **ARCHITECTURE** for this **ENTITY**.
  - Enter the signal assignment

```
LEDR <= SW;
```

- Set this **ENTITY** as the **TOP-LEVEL** Entity.
  - Right-click the file "**Part2.vhd**" (in pane Project navigator->Files)
- **Synthesize**, **Quickly Check**, **Fit**, **Program** and **Test** this final circuit.

Part 3:

Objective: **To build a circuit that connects 8 switches to two 7-segment displays.**

- **Top-Level:** Enter the design using VHDL. Use the same folder and project as the previous section
  - *File -> New -> Design Files.VHDL File*.
  - Save as "*Part3.vhd*" - verify that the file is stored in the project directory.
  - Enter the two context clause lines that precede all design units.
  - Declare an ENTITY called Part3
  - Declare the PORT signals for this entity

```
Port ( SW : in std_logic_vector( 17 downto 0 );
   HEX0, HEX1 : out std_logic_vector( 6 downto 0 ) );
```

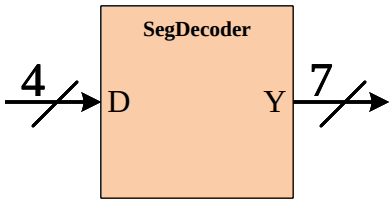
  - Write the ARCHITECTURE structure for this ENTITY.
    Leave the body empty for now, as we can easily enter the design later.

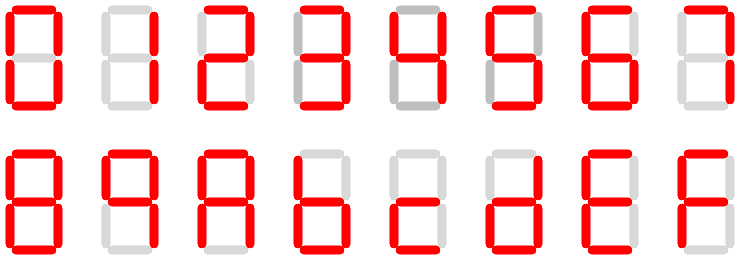
- Set this ENTITY as the TOP-LEVEL Entity.
  - Right-click the file "Part3.vhd" (in pane Project navigator->Files)
- Analyse/Elaborate and Synthesise this simple circuit. <ctrl>-k

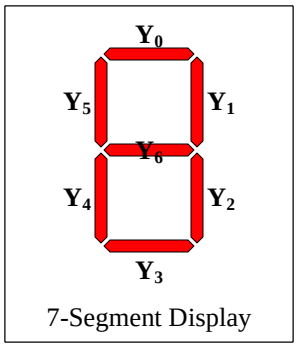
Part 3: ***continued***

➢ **Design:** a Circuit (**LOWER-LEVEL ENTITY)** called **SegDecoder**.

- The ports for **SegDecoder** are specified in the block diagram below.
- **Write a truth table** so that the 16 rows produce the patterns specified below.
- Which value, '1' or '0' makes the individual segments Turn On?







7-Segment Display

➢ **Code the circuit:** Enter the design of your combinational circuit using **VHDL**.

- ***File -> New -> Design Files.VHDL File***.
- Save as "***SegDecoder.vhd***" - verify that the file is stored in the project directory.
- Enter the two context clause lines that precede all design units (library …).
- Declare an **ENTITY** called **SegDecoder**
- Declare the **PORT** signals for this entity

```
Port ( D : in  std_logic_vector( 3 downto 0 );
       Y : out std_logic_vector( 6 downto 0 )   );
```

- Write the **ARCHITECTURE** for this **ENTITY**.
  Review/Study **Selected Signal Assignment**.
  Enter only one **Selected Signal Assignment Statement** to implement your truth table.

➢ It is now time to create the **ARCHITECTURE** for the **TOP-LEVEL** Entity.

- Enter two instances of **SegDecoder** in the **ARCHITECTURE** of the **ENTITY part3**.
- **MAP** the **PORT** signals so that the 8-Right-Most switches, **SW(7 downto 0),** control two 7-segment displays, **HEX0(6 downto 0)** and **HEX1(6 downto 0)** after being converted by your truth table.

➢ **Synthesize**, **Quickly Check**, **Fit**, **Program** and **Test** this final circuit.

Objective: **Modify the design to allow the HEX displays to be used by either the Left-Most or the Right-Most 8 switches.**

- Use a push button switch, **KEY(0)**, to select whether the display shows the Left or Right switches.
  - Design the circuit. **Draw a diagram** of your circuit in your notebook. Label all Internal Signals and Instances.
  - Review/Study **Conditional Signal Assignment**. Think about how Conditional Signal Assignment relates to a Multiplexer (**MUX**) that has two channels. Using the RTL viewer, look at the results produced by the synthesizer when you enter Conditional Signal Assignments.
  - Modify the **ARCHITECTURE** of the **TOP-LEVEL ENTITY**. (as shown below)
- **Synthesize**, **Quickly Check**, **Fit**, **Program** and **Test** this final circuit.