BME 328 LAB5

Lab 5 - VHDL for Combinational Circuits and Storage Elements

15 Marks (1 week)

Due Date: Week 9

1 Objectives

To construct combinational circuits and circuits with basic storage elements using VHDL

2 Pre-Lab Preparation

1. Start-up Quartus II. This window gives you access to an integrated suite of CAD tools
2. To save files for this lab, create subdirectories **mux**, **decode, encod,** and **johns** in your work directory.
3. Enter the name of the first project, **mux**, by clicking on File then Project on the pull down menu and then Name on the subsequent pull down menu. Type the Project Name, and click OK.
4. Open Text Editor and type the VHDL file from *Figure 6.28* of the textbook. Save the file as **mux.vhd**.
5. Start the compiler. Fix any errors and re-compile. Once the file compiles without errors, go to the next step. Copy the file *mux.vhd* to a usb drive
6. Repeat steps 3-5 for the remaining examples. Use files from the following figures accordingly (see textbook):
    i. **decod** – Figure 6.30
    ii. **encod** – Figure 6.41
    iii. **johns** – Figure 2 (In this Manual)
7. The last example shows one of the ways of implementing the Johnson counter. The last six digits of the student identification number must be represented by a four-bit vector variable **STUDENT_ID** which will be displayed cyclically in sequence with Johnson counter output. **Qreg** is an internal signal which can be fed back to the D's or fed out to **Q**.
8. Prepare a Truth Table for Johnson Counter for 6 clock cycles.

3 Laboratory Work

1. Create the subdirectory *lab4* in your work directory, and copy the all the subdirectories created as part of pre-lab to this subdirectory.
2. Compile your designs and create symbols for respective projects (**mux**, **decode, encod,** and **johns**) and save them.
3. Create new subdirectories inside lab4 folder of your working directory with names **muxModified and decodModified.**
4. Start-up Quartus II. This window gives you access to an integrated suite of CAD tools

5. Enter the name of the project, **muxModified**, by clicking on File then Project on the pull down menu and then Name on the subsequent pull down menu. Type the Project Name at top, and click OK.
6. Create a block schematic file **muxModified.bdf** for the project defined in (12) and implement a **4:1** multiplexer using two **2:1** multiplexer (**mux** symbols) as shown in *Figure 6.3* of the text book.
7. Repeat the steps 11-13 for the project **decodModified** and implement a **3:8** decoder using two **2:4** decoders (**decode** symbols) as outlined in *Figure 6.17* of the text book.
8. Assign all Input (Output) signals to any dedicated Input (Output) pins of the Cyclone- II EP2C35F672C6 FPGA on the prototype board (see Pin Assignment Tables in Lab3). Re-compile your design.
   NOTE:
   (a) All the output LEDs are active **HIGH**. (NOTE: This means high logic level will turn the LED's on).
   (b) All the 7-segment displays are active LOW (NOTE: This means low logic level will turn the 7-segment on).
   (c) The **resetn** signal must be assigned to the push button switch (PIN_G6) of the Cyclone- II EP2C35F672C6. There are four red buttons on the prototype board. The pin 1 is connected to the first button starting from the top.
   (d) The **clk** signal must be assigned to the Global Clock Input (pin 2) of the Cyclone- II EP2C35F672C6 FPGA.
9. Implement/program all your designs into the Cyclone® II 2C35 FPGA.
   NOTE: before programming double-check pin assignments. Incorrect pin assignment can result in failure of the Cyclone- II EP2C35F672C6 FPGA.
10. Every digit of last 6 digits of Student ID should be displayed on the seven-segment display while the current state of the **johns** is displayed on green LED's.
11. Design your circuits as outlined and demonstrate results to the instructor by displaying both the states and student identifier utilizing the displays of your prototype board.
    NOTE: Re-use the 7-segment module from Lab3 to display states and student identifier digits.
12. The circuit design must handle non-valid states and non-valid student identifier cases by displaying an "E" in the seven segment display.
13. Consider the last 6 digits of the student identifier D = {d_1, d_2, d_3, d_4, d_5, d_6} in its general representation. Then, as an example, a student with identifier: **500435429** will follow the display sequence as **435429**.

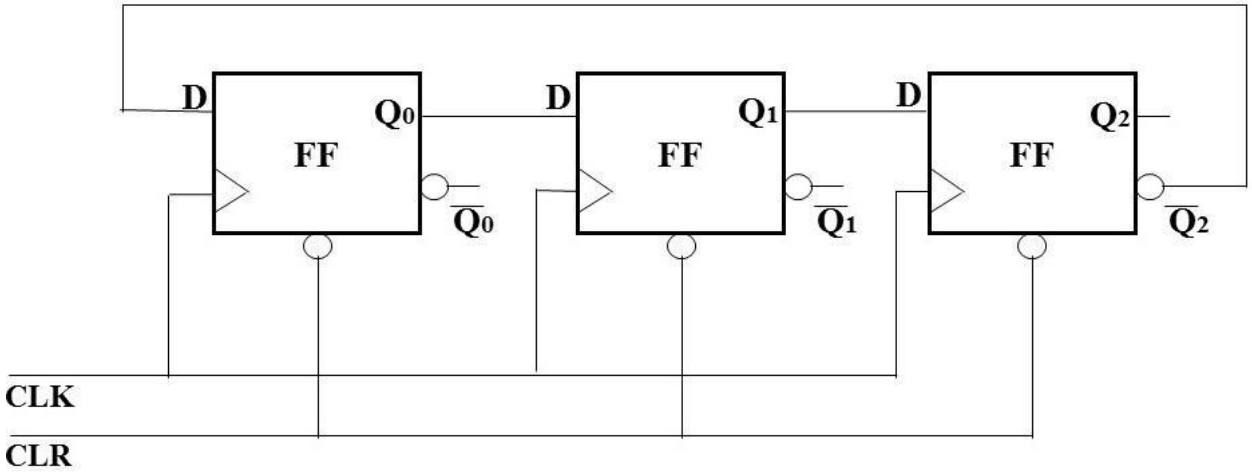

Figure 1: Johnson Counter (Note feedback Connection)

```
LIBRARY ieee;
USE ieee .std_logic_1164.all;
ENTITY johns IS

PORT (Clrn, E, Clkn : IN STD_LOGIC; --clrn is your reset button
 STUDENT_ID : out   std_logic_vector(3 downto 0);
 Q : OUT STD_LOGIC_VECTOR (0 TO 2));
END johns;

ARCHITECTURE Behavior OF johns IS
signal Qreg : STD_LOGIC_VECTOR (0 TO 2);
BEGIN
    PROCESS (Clrn, Clkn)
     BEGIN
         IF Clrn = '0' THEN
            Qreg <= "000";
         ELSIF (Clkn'EVENT AND Clkn = '0') THEN
             IF E = '1' THEN
                 .. -- complete your johns flip-flop outputs here..
                 ..
                 Qreg(1) <= Qreg(0);
                 ..
                  ..
             ELSE
                Qreg <= Qreg;
             END IF;
         END IF;

    -- STUDENT_ID variable represents the last 6 digits of your student ID
    hence d4 is the fourth digit of your
    --student ID in four bits, d5 is the fifth and so on. For example, for
    Student ID 500435429,
    --d4 is 0100, d5 is 0011 and so on
         CASE Qreg  IS
             WHEN  "000" =>
                 STUDENT ID <= ..... --d1
             WHEN  "100" =>
                 STUDENT ID <= ..... --d2
             .                       --d3
             .                       --d4
             .                       --d5
                                     --d6
             WHEN OTHERS => STUDENT_ID <= "----";--error
         END CASE;
    END PROCESS;
Q <= Qreg;

END Behavior;
```

Figure 2: Code Template for Johnson Counter to sequence STUDENT ID

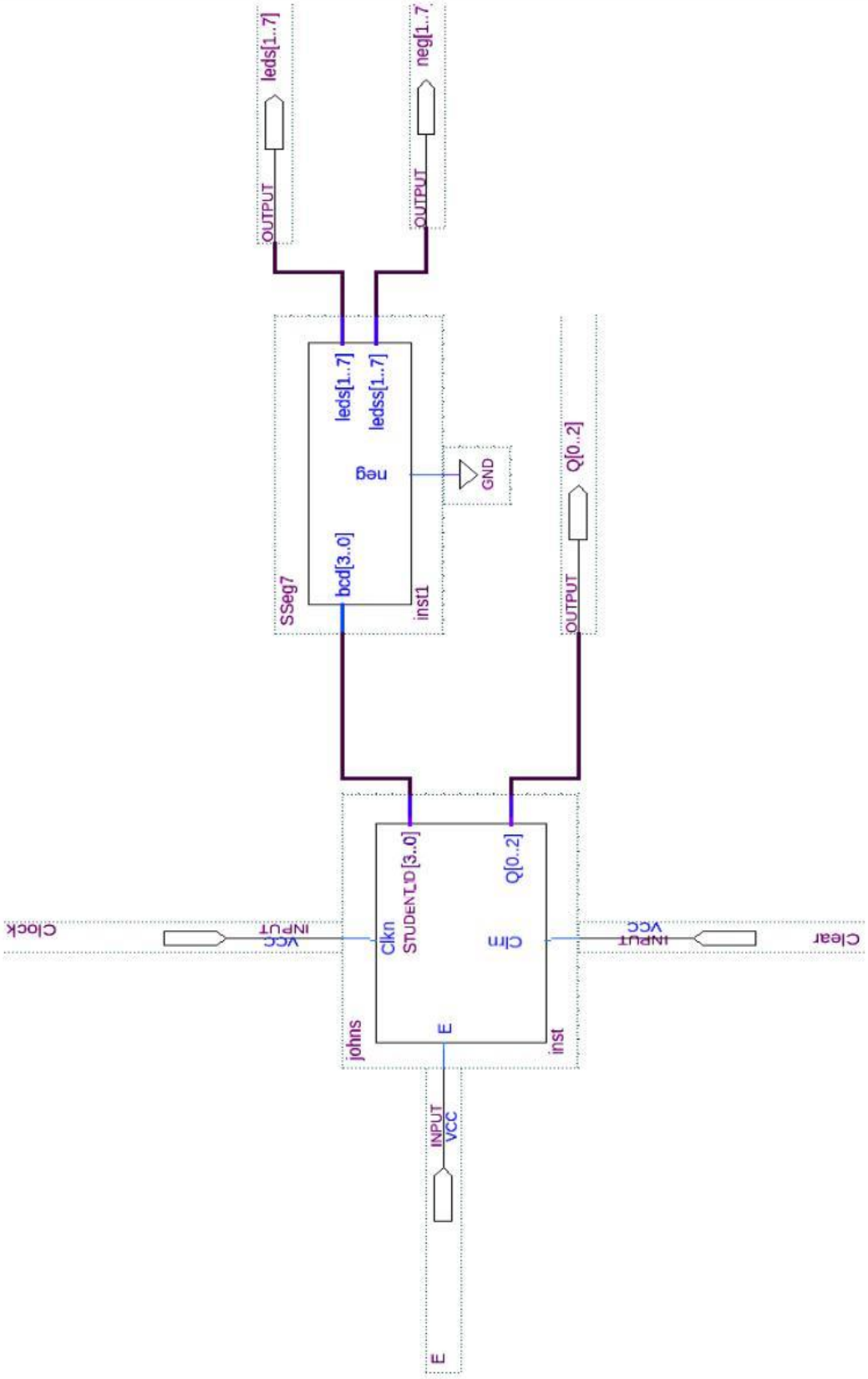

Figure 3: Typical Schematic produced using Quartus II