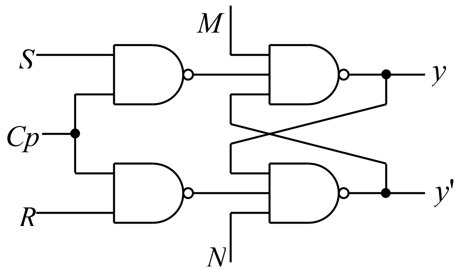
Lab #4: Sequential Logic Design Using Schematics

Introduction

In the previous lab you built combinational logic circuits where the output was an instantaneous function that depended on the current input only. In this lab you will build **sequential logic** circuits where the outputs depend on the current and *previous history* of outputs, and generally change only when 'clocked' by a clock signal. The **memory** components within the sequential logic are what allows for the dependence on *previous history*. A 1-bit memory device is known as a **flip-flop** (FF) or a **latch**.

- In part 4.1 you will design a **Set-Reset flip-flop (SRFF)**, with additional **Preset** and **Clear** inputs (which are useful to force the initial output to be a **1** (Preset) or **0** (clear), respectively).
- In part 4.2 you use three **D flip-flops (DFFs)** from the Component Library browser to design a **Counter** circuit which counts or steps to the next value in a sequence of numbers on each clock pulse.
- In part 4.3 you will use two **D flip-flops** to design a **Finite State Machine** circuit.
- You will manually assign pins to your input/output signals.
- Read through the entire part before starting!

4.1 Set-Reset Flip-flop (SRFF) Design and Implementation



1. A circuit for an SR flip-flop is shown here: →

- **Cp** is the clock pulse input
- The inputs **M** and **N** are signals that can be used to **Preset** or **Clear** the output **asynchronously** (i.e. without waiting for a clock pulse), but your cruel professor has not told you which is which*!*
- Assuming the clock pulse is *inactive*, derive the truth table from the circuit diagram: →
  - Hence determine which of the two inputs, **M** and **N**
    is the **Preset** input (sets **y** to 1): ___________
    is the **Clear** input (sets **y** to 0): ___________
  - Also determine whether the M, N are **active low** or **active high** signals: _______________

M	N	y^+	y'^+
0	0		
0	1		
1	0		
1	1		

2. When M & N are inactive, SRFF functions as follows
   - **S** sets the output y to **1** on the next **clock edge**
   - **R** resets the output y to **0** on the next **clock edge**
   - **y** stays the same value when SR are set to 00
   - **y'** is always the opposite of the output **y** when

Note: M and N should be used only for forcing the initial state, and should be held ***inactive*** most of the time, since they override the regular SR behavior.

3. Implement your SRFF design within Quartus:
   - Create a new Project called **Lab4P1**, and within it a **File>New>Schematic** diagram called **SRflipFlop.bdf**. Create the SRFF diagram on Page 1 in this schematic. Add an **inverter** to the clock signal Cp (because the push-buttons go low when pressed and you want the clock to go high when pressed). Set SRflipflop.bdf as the top design. Compile.
   - **Assign pins:** In the past, you have been importing pin assignments from a given file. Starting this lab, you will assign pins to your input/output signals using **Pin Planner**. A **pin assignment table of the DE10 board** is posted in the Lab Assignments folder.
     - Click **Assignments> Pin Planner.** Adjust the window boundaries to see the table listing the inputs/outputs of your design. Adjust the columns so you can see the full pin names. Pin Assignments will be placed in the Location column.
     - Open the **DE10 pin assignment table** in BB.
     - For each of the signals below, click in the **Location** cell to select the pin name you want to connect to (check the pin table) and choose "3.3-V LVTTL" in the **I/O standard** column.
       - Assign **Cp** to the pin of **KEY[0]**
       - Assign the pins of **SW[3]..SW[0]** to the **S, R, M**, and **N** inputs, respectively.
       - Assign the pins of **LEDR[1]** and **LEDR[0]** to the **y** and **y'** outputs, respectively.
     - Compile (compile is needed every time you make change to your project).
4. Compile, Program, and **test** your circuit on the **DE10** board.

   Demonstrate the following to the instructor/TA:
   - Show you have completed the analysis in section 4.1.2
   - With the **clock** input *inactive*, demonstrate that the **M** and **N** inputs work correctly.
   - With **M** and **N** *inactive*, demonstrate that the **S** and **R** inputs work correctly along with the **clock**.

4.2 Counter Design and Implementation

In this part, you will design a counter and implement it using Quartus. You can obtain your personalized counter specification from the instructor/TA.

1. Design the circuit using **three** D flip-flops and whatever other gates required. The circuit must be *minimized; i.e., it should use the minimum number of logic gates possible.*
   - Make a **Next State** table for your counter, including unused states.
   - Create a **Karnaugh map** for each D input of your counter, and from them derive minimized expressions for the excitation functions **D2..D0**.
   - Also predict the next state when the counter is set to one of the **unused states**.
   - Sketch the circuit you will build to implement your counter with **three** DFFs.
   - Take a picture of your counter page with your work and keep it in your 4.2 folder.
   - You will also need your counter design in Lab 5. So keep it (and your specification sheet)!

   Have your circuit checked by the instructor/TA before proceeding further.

2. Implement your circuit in Quartus:
   - Create a new Project called **Lab4P2**, and within it a **File>New>Schematic** diagram called **Counter.bdf**. Recreate your counter circuit in this schematic.
     - Use **Primitives>Storage>DFF** (D-type Flip-Flop) from the Component Library browser for your flip flops.
     - Add an inverter to the clock signal of the DFFs. Compile.
   - Assign pins in **Pin Planner**:
     - Connect the Preset (**PREN**) signals of your three DFFs to switches **SW[9]** to **SW[7**], respectively. Note: these signals are active low.
     - Connect the Clear (**CLRN**) signals of your three DFFs to switches **SW[6]** to **SW[4**], respectively. Note: these signals are active low.
     - Connect the counter output (the outputs of your three flip-flops) to **LEDR[2]..LEDR[0]**, respectively.
     - Connect **Key[0]** to the **clock** signal.
3. Compile, Program and **test** your circuit on the **DE10** board. Test the preset and clear (reset) signals first, then make them inactive when testing the counter circuit.

Demonstrate the following to the instructor/TA:

- When PRENs and CLRNs are set inactive, the main counting **sequence**/cycle is generated correctly as you clock your circuit
- Force the circuit into each of the three **unused** states (using the corresponding Clear and Preset inputs) and verify your predictions of unused states.

4.3 Finite State Machine Design and Implementation

In this part, you will design a finite state machine and implement it using Quartus. You can obtain your personalized **state machine** specification from the instructor/TA.

1. Design the circuit using **two** D flip-flops and whatever other gates are required.
   - **Substitute** state variable values into the next state table and **rearrange** the rows into Karnaugh map form.
   - Hence create a **Karnaugh** map for each state bit **Q1..Q0** and input of your state machine and from them derive minimized expressions for the excitation functions **D1, D0**, and for the output function **z**.
   - **Sketch** the circuit you will build to implement your machine with the **two** flip-flops.
   - Take a picture of your counter page with your work and keep it in your 4.3 folder.
   - You will also need your state machine design in Lab 5. So keep it (**and your specification sheet)**!

Have your circuit checked by the instructor/TA before proceeding further.

2. Implement your circuit in Quartus:
   - Create a new Project called **Lab4P3**, and within it a **File>New>Schematic** diagram called **FSM.bdf**. Recreate your state machine circuit in this schematic.
   - Assign pins in **Pin Planner**
     - Connect the FSM states **Q1..Q0** to **LEDR[1]..LEDR[0],** respectively, and the output **z** to **LEDR[2]**.
     - Connect your input **x** to **SW[0]**, and the Preset (**PREN**) and Clear (**CLRN**) signals of your two DFFs to switches **SW[5]** and **SW[4]**, respectively. Note: these signals are active low.
     - Feed input **Key[0]** through an **inverter** and then use this to **clock** your flip-flops.
   - Compile, Program and **test** your circuit on the **DE10** board.

Demonstrate the following to the instructor/TA:

- Force the circuit into state A using the preset/clear inputs. Then demonstrate that you can step through all states of the **next state table** in your original state machine assignment paper.

Submission for Lab 4:

After you have been checked off all the parts by the instructor/TA, zip all the Lab4 project folders into one zip file, name the zip file Lab4_<your last name>.zip, and submit it to Lab assignment in Blackboard. Please only submit one submission to blackboard, and please ensure everything is contained in the single zip file.