University of Cincinnati
Department of Electrical Engineering and Computing Systems

EECE 2060C – Digital Design ELTN 1040C – Digital Systems

Laboratory Project 7
A Sequence Detector Implementation using Breadboard
Fall 2017

Introduction

The purposes of this lab are to help you fully understand the working principle of sequential circuits and to get familiar with every design step. This lab requires you to start from design specification all the way down to circuit diagram and simulation. Finally, you have to implement the circuit using breadboard, wires, FFs, and gates.

Design Specification

In this lab, a synchronous sequential circuit is design to recognize a sequence of 11*0 where 1* represents any number of logic 1's, e.g., 0 logic 1, 1 logic 1, 2 logic 1's, ….. The sequential circuit has one input I and one output Z, and Z outputs logic 1 when the machine identifies a sequence of 11*0. A sample input sequence and output response is given below.

I = 0 1 1 0 0 1 0 1 0 0 1
Z= 0 0 0 1 0 0 1 0 1 0 0

Task 1

The sequential circuit must be a Moore machine and can use 3 states: S0, S1, and S2. You must design this machine using *D-FFs*. You will see lots of don't cares to help you reduce the circuit size since two FFs will be used and S3 is NOT required (and thus can be used for don't cares). The workstation has a **clock generator** with tunable frequency. You must use an appropriate clock rate, e.g., one clock cycle per second. Note that if the clock is too fast, you cannot input the sequence on time. However, if the clock is too slow, you lose your patience, since you have to wait for so long to give the next input bit. **Input I** can be implemented using a *switch*, and **output Z** can be implemented by an *LED*. Use whatever circuit minimization methods, e.g., K-map, to reduce the circuit size such that your wiring efforts can be minimized. Go through **pages 236 to 239** of the textbook to learn the design procedure.

Note that you can also **use push-button to generate the clock signal** such that it can be fully controlled during the debugging phase. The clock generator keeps pumping clock signal to the sequential machine which may make debugging extremely difficult. You are **recommended** to use a push-button (instead of clock generator) to generate the clock signals. The breadboard orientation contains an introduction to breadboard workstation where operations for push-buttons can be found. The paragraph is duplicated below.

Two Debounced Pushbuttons

Two pushbuttons are provided on the CADET workstation each of which can be configured to provide a momentary Gnd-5V-Gnd (LO – HI-LO) pulse. Since mechanical switches naturally "bounce" when they are switched, a simple electronic circuit can be added to the switch (thereby making the switch "debounced") that delays the output of the switch by several milliseconds to ensure the switch is solidly closed or open. By default the switch is in the normally closed (NC) which ties the NC outputs to ground. When depressed, the NO are tied to ground. The most important thing is to **add a 300 Ohm resistor** from Vdd to a NC (NO) pin as shown in Fig. 1(a) (Fig. 1(b)) to generate a 0-1-0 (1-0-1) pulse when the button is pressed. Clock implemented by **0-1-0 pulse in Figure 1(a) is preferred in this lab.**

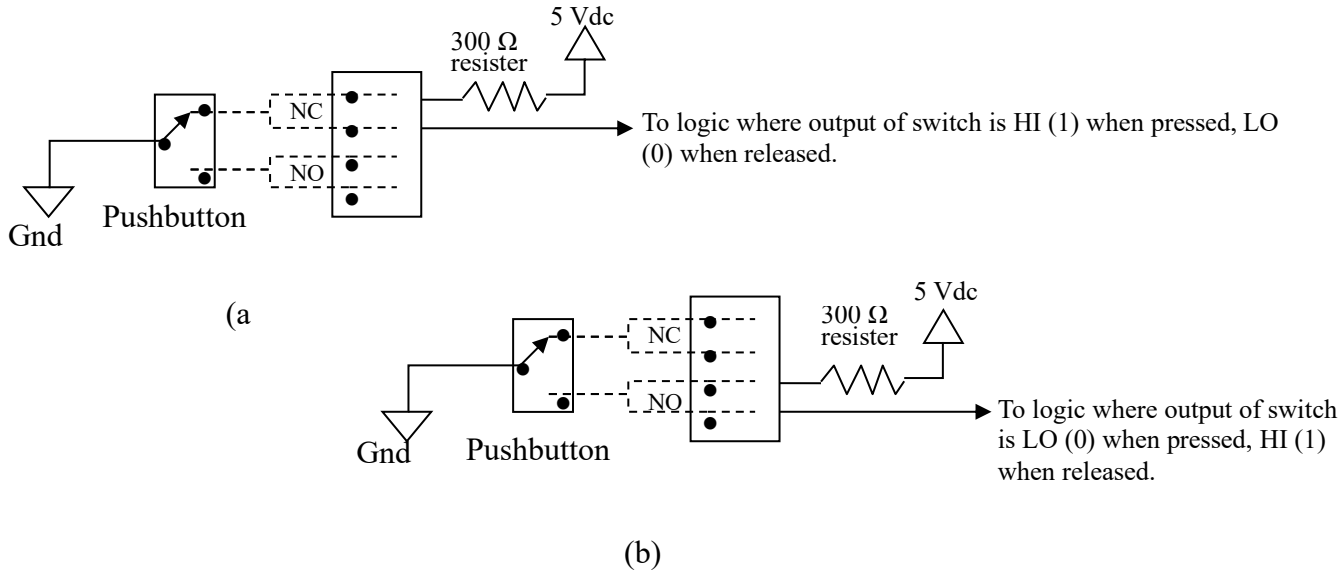


Figure 1. Wiring diagrams for pushbutton switches. Normally HI output when button is depressed is shown in (a), OUT output when button is depressed is shown in (b).

Notes: (1) Never try to use a switch to generate clock signals since they can be very noisy and can make your sequential circuit non-deterministic. (2) Make sure that you have PR = CLR = 1 after CLR =0 for state initialization where PR is an active low preset and CLR is an active low clear.

Circuit Simulation and Testing

Before wiring the circuit, use *logicSim* (again, tutorial and installation guide for logiSim is given in the end of this lab description) or *ISE* to simulate the sequential circuit using a well-designed test sequence. Without using simulation, there is no way to verify your circuit before implementing to the breadboard. You have to show the TA your simulation waveform generated by ISE or input

and output patterns by logicSim, and *take a picture* of the waveform (or inputs/outputs) to be included in your final report.

The following codes will help you simulate your sequential circuit.

```
module D_FF(Q, D, CLK, Clr
   );
   output Q;
   input D, CLK, Clr;
   reg Q;

   always @(posedge CLK, negedge Clr)
      if (~Clr) Q <= 1'b0; else Q <= D;
endmodule

module Moore_sequential(output y_out, A, B,   //A, B are used for debugging ONLY
                  input x_in, clock, reset
   );
     wire Da, Db;
      assign Da = x_in | B; //Change the Da function to describe your Da (next-state) circuit.
      assign Db = x_in & A; //Change the Db function to describe your Db (next-state) circuit.

       assign y_out = A & B; //Change y_out for your output circuit

       D_FF  m1 (A, Da, clock, reset);
       D_FF  m2 (B, Db, clock, reset);
endmodule

module Sim;

          // Inputs
          reg x_in;
          reg clock;
          reg reset;

          // Outputs
          wire y_out;
```

```
wire A;
wire B;

// Instantiate the Unit Under Test (UUT)
Moore_sequential uut (
        .y_out(y_out),
        .A(A),
        .B(B),
        .x_in(x_in),
        .clock(clock),
        .reset(reset)
      );

        // Initialize Inputs
        initial begin
          x_in = 0;
          clock = 0;
          reset = 1;
          #2 reset = 0;
          #2 reset = 1;
          repeat (16)
            #5 clock = ~ clock; //used to generate 15 clock cycles
        end

        // Add stimulus here
        initial begin
                #12 x_in = 1; //Change the input patterns to fully simulate your circuit
                #20 x_in = 0;
        end

endmodule
```

Note that you just need to modify the Da, Db, and y_out (marked **BOLD)** circuit to describe your design using data-flow modeling. Further, modify the input patterns (x_in) in the above testbench to fully verify your design. At least, you should use the following test sequence in your testbench.

I = 0 1 1 0 0 1 0 1 0 0 1 ←--------- This is your x_in sequence above
Z= 0 0 0 1 0 0 1 0 1 0 0 ←----------This is the expected output sequence for y_out

Have the TA sign for your logicSim or ISE simulation result when the result is correct. Take a picture of the waveform or input/output patterns and then start wiring the circuit on the breadboard.

After wiring the circuit and testing your breadboard circuit, take a picture of the board operation to be included in your final report, and have TA check and sign for you. Fill the test sequence below.

Input I:

Output Z:

Waveform simulation; TA sign here_____

FPGA board functions: TA sign here_____

Project Pre-lab Design Report (PDR) Format:
The PDR is a computer-edited or hand-written report documenting the development activities conducted during the pre-laboratory phase of the lab project.
The format of the PDR must include:

1. Identification Section (Name, M#, Date, Project #, etc.)
2. Section # is worth 5%
3. Design specification.
4. Major design steps, e.g., state diagram, state assignment, state table, FF-type selection (D-FFs), simplified FF input equations and output equation, logic diagram. Note that you **MUST include the design steps** in the pre-lab design report. This part has very high weighting in the grading.
5. Test patterns for the circuit.
6. Verilog codes (D_FF module, 11*0 detector module, and testbench) for ISE simulation or logicSim simulation.
7. It is **required** that you finish the logic simulation before the lab starts.
8. Design Time (in hours) to complete the pre-lab phase of this task.

Pre-lab Design Report Due: 10:00am, Nov. 6 (Monday), 2017. Submit an **electronic version to BB** to the link of BB, and a hard copy to Rhodes 808. Late submissions will incur a 30% penalty for each day delay. Design reports will not be accepted after beginning of the lab.

Note that next Friday (11/10) is a holiday. If your lab on that day, please go to join another lab section.

Project Final Report (PFR) Format:
The PFR is computer-edited report that documents all aspects of the completed tested project.
Title page with Identification Section (Name, M#, Date, Project #, etc.)

Section is worth 5% of the total grade

1. Design requirements and specification
2. Major design steps, e.g., state diagram, state assignment, state table, FF-type selection (D-FFs), simplified FF input equations and output equation, logic diagram. (you can reuse those in the pre-lab design report by copy and paste or attach your pre-lab design report).
3. Test pattern for the circuit. (you can reuse those in the pre-lab design report by copy and paste, or attach your pre-lab design report).
4. Photo of ISE simulation waveform.
5. Photo of breadboard and circuits.
6. TA's signatures for items 4 and 5 above.
7. Answer the following common questions.

(a) What is the utilization of your components (i.e., total number of used gates (or FFs) divided by the number of IC components? Can you think of any way(s) you could improve the component utilization?
(b) How many test vectors were required to *assure* correctness of the circuit?
(c) Define in detail the contribution of each team member to the accomplishment of the project for each phase (i.e., Pre-lab, In-Lab, and Post-Lab). The idea is that over the quarter all team members share equally in all aspects of the laboratory activity (requirements, design, simulation, test, implementation, and report writing).

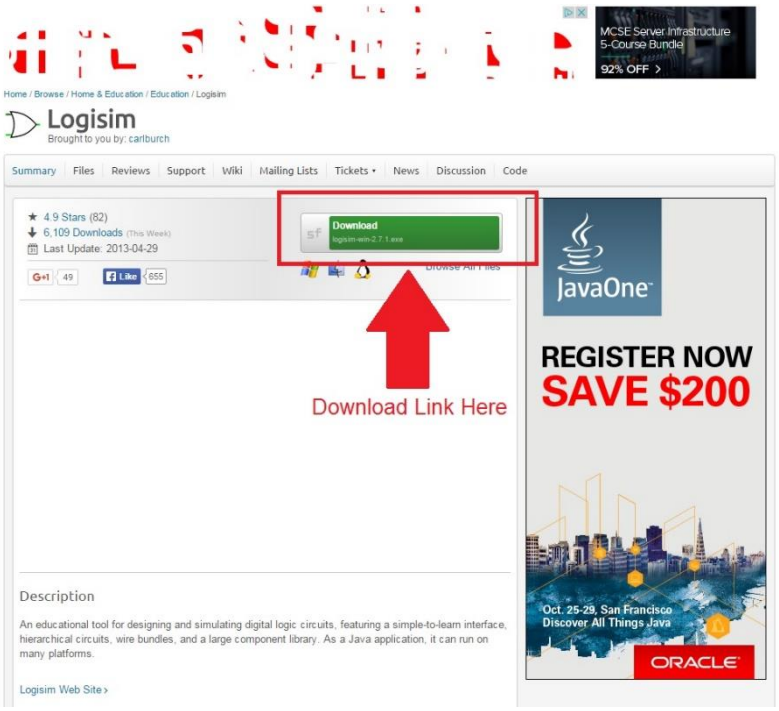
Final Report Due: Submit a hard copy in the lab hours in the week of 11/13 (you have one week to work out the final report as always) to the TA. Late submissions will incur 20% of late penalty each day.

Tutorial and installation guide for logiSim

1. Download Link for Logisim: http://www.cburch.com/logisim/download.html

2. Follow the SourceForge link on this page (listed as step 2 in their description) to download Logisim.

3. Once you get to the SourceForge page, if you cannot find the download link, it's here:



4. Windows systems simply only need to download the .exe from the SourceForge page. There is no "installation" required for Logisim. Just run the .exe and Logisim will work.

5. For Mac systems, just unzip the .tar.gz. Once unzipped, double-click the Logisim icon to run the program.

https://www.youtube.com/watch?v=ATPqpFMlVdw
LOGISIM Tutorial & Demo