SE 2DA4 Lab 4[1]

First lab Week of: Nov. 4, 2019
Prep Due week of: (8:40/14:40), Nov. 11, 2019
Demo Due Week of: (11:20/17:20), Nov. 11, 2019
Assigment due in class: 17:40, Nov. 21, 2019

Announcements:

Note 1: all lab due dates/times are for your scheduled lab in the indicated week.

For this and every lab: after you have demonstrated your lab to a TA , you must e-mail (as per instructions in lab 1) the final versions of your Verilog files (the ones demoed to the TA) as attachements to the following e-mail address: `rlta1@cas.mcmaster.ca`

Part 1: Assignment

Note: You must put your **lab section** on the top of your assignment as they will be handed back during your lab.

The following are relevant textbook questions to be handed in 17:40, Nov. 21, 2019:

Q 1-3) Ch. 5 # 1, 4, 14.

Q 4-5) Ch. 6 # 7 (for figure 6.52 only using state assignments A =00, B = 01, C = 10 and F = 11. Ignore cost discussion. Draw the circuits using D flip-flops and logic gates), and #9 (create state diagram and state table only.)
Q 6) What is the maximum clock frequency for the circuit in text Fig. 6.8? For timing purposes assume a gate delay of 2 ns for any logic gate, a flip flop setup time of $t_{su} = 2$ ns, a hold time of $t_h = 1$ ns, and a clock-to-output time of $t_{co} = 1$ ns.

For Extra practice (not to hand in):

1. Ch. 6 # 1 (draw the actual circuit), 5.
2. **Arithmetic Circuits**

   Figure 1 contains the block diagram symbols for a 1-bit full adder (FA) and a 4-bit adder.

   a) Let δ be the propagation delay for a single logic gate.

   i) What is the maximum time required to produce a valid output at c_4 by a 4 bit ripple carry adder circuit? (Assume the standard S-of-P implementation of c_{i+1} is used in FA.)

[1]This lab is based on lab from the ECE 342 course taught by Prof. Zaky at the University of Toronto. In fact, large sections are copied.

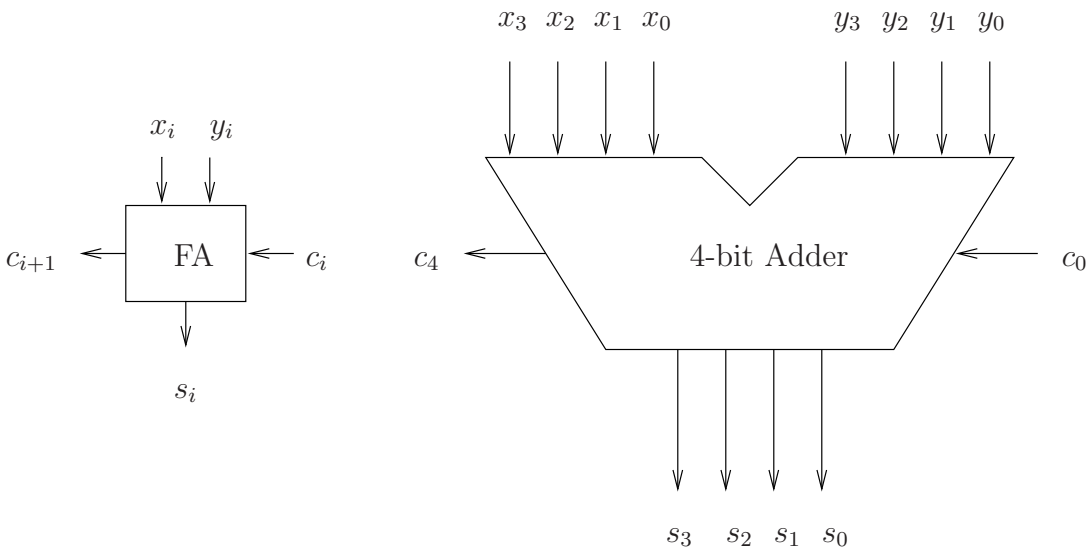

Figure 1: Adder Block Diagrams

ii) In class we discussed the design of "fast adders" that required less time to produce valid outputs. Derive the equations for the carries out of the first two stages of the carry look-ahead "fast" adder (i.e. write the equations for c_1 and c_2 in the carry look-ahead adder.).

iii) What is the the maximum amount of time required to produce a valid output at c_4 if the ripple carry adder is replaced by a carry look-ahead adder?

Part 2: Preparation

As your preparation, hand in the Verilog code and simulations. Also, hand in a copy of your state diagram and a block diagram of your design showing each main part of the design as a separate block (clearly labelled), showing all inputs and outputs to each block, and how the blocks connect to one another, as described in class. For instance, you might have a block for your 4-bit register to store the combination, a block for a 4 bit compartor to determine if the entered number matches the saved value, a block for your LED display driver, a block for your state machine and so on. Each block should be a separate subcircuit, combined together in the final design. You should have separate simulation for each subcircuit, as well as a simulation for the final design.

Background:
You need to be able to create a Moore finite state machine. See introduction to Chapter 6, Sections 6.1-6.1.3, 6.1.5, 6.4.1, as well as Appendix A, Sections A.11, A.14.8.

As the switches tend to be noisy, you will need to use the hardware debounce pushbuttons KEY[0]-KEY[3] (see page 23 of the DE1-SoC user manual for more details) for all of your input signals (excluding the 4 bit input X - make sure you understand why X does not need to be debounced). If you generate clock pulses yourself while testing, make sure you use a pushbutton.

Input Conditioning: External inputs to a state machine are often asynchronous (change in input signals are not synchronized to the clock) and they change much slower than the clock signal. If inputs are not properly conditioned, then the circuit may operate incorrectly. As an example, a pop machine controller might count a nickel more than once if the internal signal representing the nickel remains high for multiple clock cycles. Design a circuit that produces a single pulse when the input A is held high, as shown in Figure 2. Hint: create a Moore finite state machine.

Use pushbutton KEY[0] for input A, and pushbutton KEY[1] to generate clock pulses. Connect output A_pulse to red LED LEDR[9]. Hand in Verilog code and simulation as part of preparation. Also hand in a copy of your state diagram. Download to Altera board and verify correct operation. You don't have to demo this to a TA.

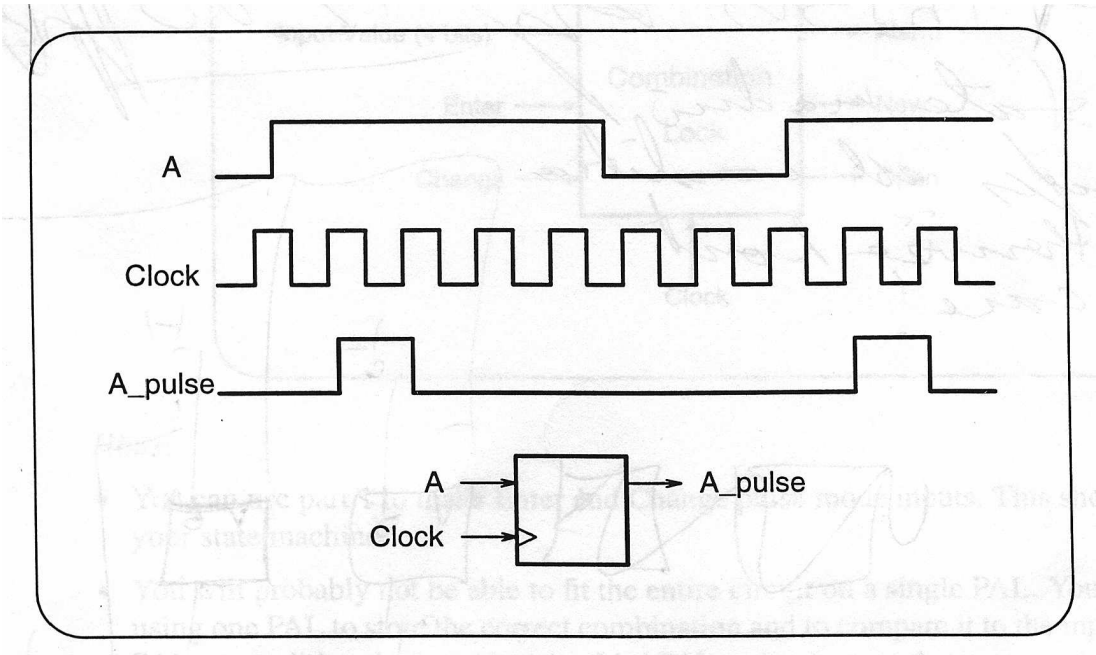

Figure 2: Input Conditioning

Combination Lock: Design a 4-bit combination lock (see Figure 3) that functions as follows:

Set the 4 input bits ($X = x_3x_2x_1x_0$) to the desired value and "press" Enter (set Enter input HIGH). If X matches the stored 4-bit combination, then the door will open (output Open goes HIGH). The door should stay open until Enter is "pressed" again. If two incorrect combinations are entered in a row, the alarm goes off (output Alarm goes HIGH). The alarm can only be cancelled by resetting the system (set Reset input HIGH).

When the system is reset, the stored combination should be set to '0110'. To change the combination, set the input to the old combination and "press" Change (set Change input HIGH). If this is done correctly, output New goes HIGH to signal user to enter new combination. Set the 4 input bits to the new combination and "press" either Change or Enter to store the new value. Two incorrect values will set off the alarm as above. Hint: use the input conditioning circuit to make Enter and Change pulse mode inputs.

To provide output for your circuit, use seven-segment display HEX5. If all outputs are LOW, then display '-'. If Alarm active, then display 'A'. If New active, display 'n'. Finally, if Open active display

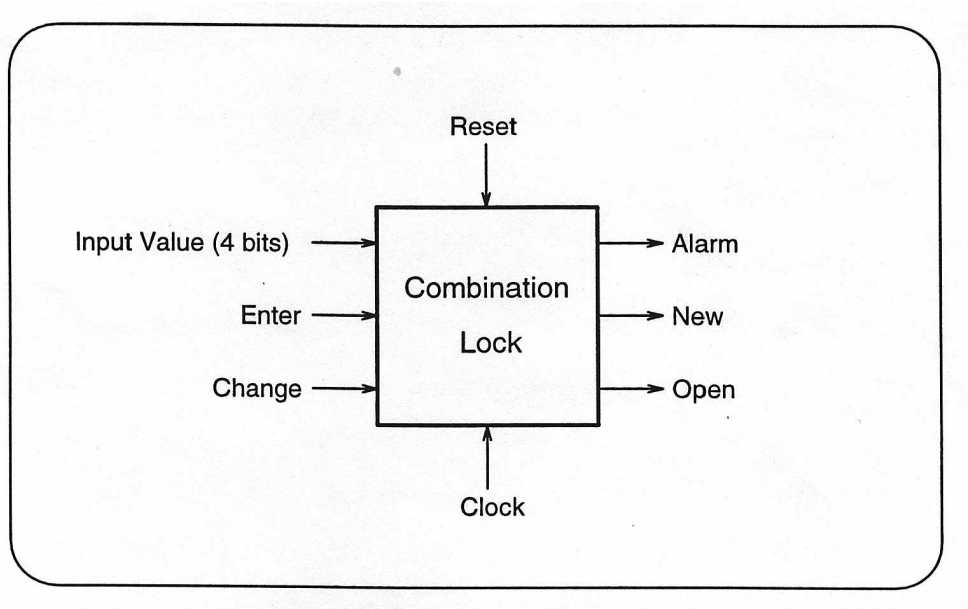

Figure 3: Combination Lock

'O'.

Debugging tips: Test and debug each subcircuit separately, before combining them together. Test your final design initially using a debounced pushbutton to generate clock pulses. For the demo, you will need to use one of the four 50 MHz clock sources to Cyclone V FPGA (see Page 22 of the DE1-SoC user manual for more details). If your circuit isn't working, try bringing internal signals out, and connect them to red or green LEDs. Good candidates are the state bits of your state machine.

Please remember to set all of the unused DE1-SoC I/O pins to "As input tri-stated with weak pull-up" or "As input tri-stated." See: `"Assignments | Devices,"` and then click the `"Device and Pin Options >> Unused Pins."` This should ensure a more stable behavior as unused pins that are allowed to float can cause unusual behavior in the rest of the circuit and this lab is more sensitive to this than the earlier labs.

Part 3: Demonstration

Demonstrate the circuit to the TAs by compiling it for the Cyclone V FPGA (5CSEMA5F31C6) device and downloading it to the DE1-SoC board.

Use toggle switches SW[9], SW[8], SW[7], SW[6] for inputs x_3, x_2, x_1, x_0 respectively. Connect Enter to KEY[0], Change to KEY[1], and connect the Reset signal to KEY[2]. Use one 50 MHz free running clock on the Altera board for your Clock input.