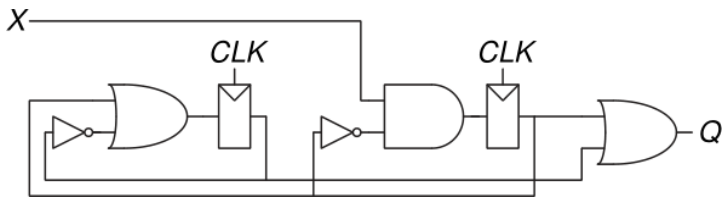
Problem Set #3, EE part

Issue date: Oct. 23, 2022; Deadline: 23:59, Nov. 6, 2022

Student Name:_____Student No.: _____

1. *FSM I*

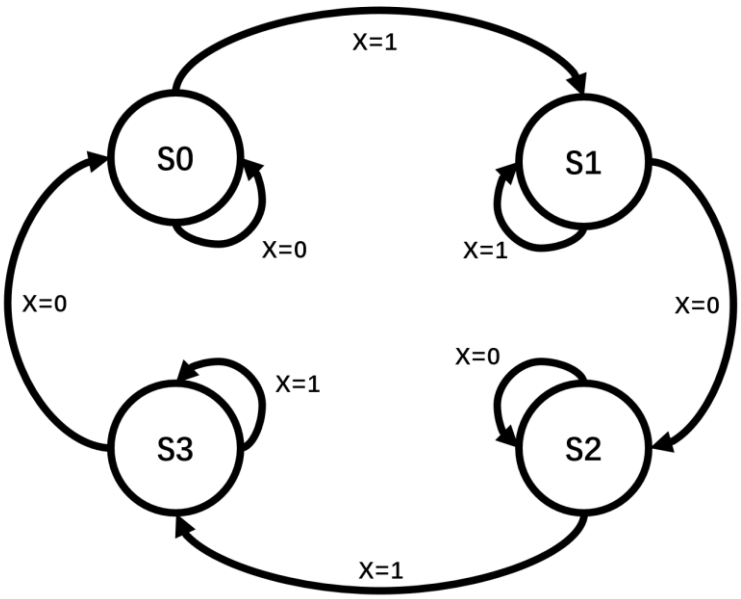
- Analyze the FSM shown as follows. The input of this FSM is X, clock signal is CLK, and output is Q. Write the state transition table, output table. Sketch the state transition diagram. (10')
- Suppose the clock frequency is 50 Hz. For this FSM, if the input X is HIGH, what will the output be? What if the input X is LOW? Describe both situation in words. You can show hand-drawn waveform, but you can't use simulation results. (10')



2. *FSM II*

- The state transition diagram of an FSM is shown below. The input is X. State encoding and output encoding is shown in the table below. Draw the state transition table of this FSM (20')
- According to the transition table you've drawn, simplify the state logic and output logic, implement the circuit in Multisim. The clock frequency should be 1000 Hz and the initial state should be S0. Test its performance. (20')

(You can use D_FF as your flip flop, and use DIGITAL_CLOCK as your clock signal. You can use INTERACTIVE_DIGITAL_CONSTANT as your input signal when debugging)
(To efficiently demonstrate your result and prove your circuit is correct, use 10Hz CLOCK_VOLTAGE as your input, and use transient analysis to plot the output. Please refer to appendix for more information)
(Please watch HOW_TO_FSM.mp4 on piazza if you are having difficulties)



State name	State encoding	Output0:3
S0	00	1000
S1	01	0100
S2	10	0010
S3	11	0001

3. *More combinational logic*

- Implement a full adder with basic logic gate (NOT, AND, OR, NOR, NAND, etc.). The inputs are A, B and CI (carry input). Outputs are S and CO (carry output). Use logic converter to show its truth table. (logic converter can only show one output at a time, so you will need to show two truth table, one is S and one is CO. Don't forget to mark the input so that we know which one is which) (10')
- Use the full adder you've built in the last problem, implement a 2 bits adder. The inputs are two binary number X and Y, each containing two bits. For X, it's X1, X0, and for Y it's Y1, Y0; Outputs are Z1, Z0 and one carry bit CO. The purpose is to calculate X+Y. Show your circuit schematic. Use logic converter to show the truth table of Z1, Z0 and C. (10')
- Bonus: Implement an 2bit ALU. The control bit is CTRL, when CTRL is LOW, your ALU will perform addition like the full adder in the last question. When CTRL is HIGH, your ALU will perform NOR operation (In other words, When CTRL=1, Z0 = X0 nor Y0, Z1 = X1 nor Y1). Use 7-segment display to show the input and output. (10')
  (The maximum score of this homework is capped at 100 points. So you don't have to do bonus to get full mark)

4. *MCU development*

Control the brightness of the on board led of your ESP32 by using PWM. Let the LED's brightness increase linearly from 0% to 100%, then let it decrease from 100% to 0%, then turn off the LED. The speed of increase and decrease should be the same, and the whole process should be done between 8 to 12 seconds. (20')

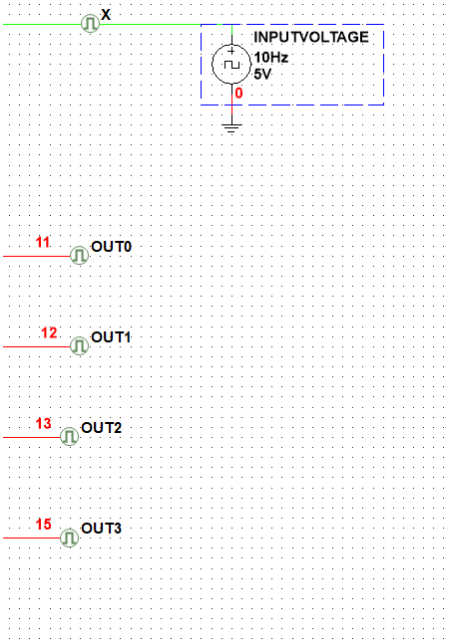
(You are expected to search online and learn about PWM yourself. It's okay to use the machine.PWM class.)

(When submitting your code, we recommend copy and paste your code directly, you can visit http://word.wd1x.com/ to make your code more legible. Then you can copy and paste the code into your word document)

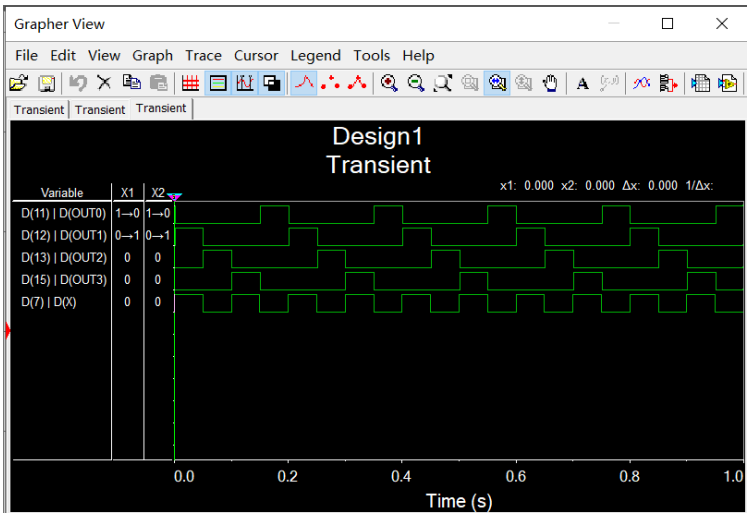
** When capturing circuit schematics and simulation results, taking a screenshot is mandatory. Please refrain from using your phones to take a photo of the screen. If you do so, 20% of the full grades will be deducted from that problem.*
** Please submit the softcopy of your solutions to the problems on gradescope. When uploading to gradescope, please select all corresponding pages related to each question.*
** Please use English. Answers using Chinese or other language will be deducted 5 points.*
** Discussion on methodology is allowed, yet, the assignment should be done individually. Plagiarism, once found, grades zero for the whole homework assignment.*

Appendix:

Simulation setup: (your design is on the left)



Correct result:



(note how the output starts with S0 as initial state but instantly jumps to S1 since the input is high at the beginning)