Homework 10

Homework 10 is due on Wednesday, April 22, at the end of the day. Remember to include your *Discussions section* (e.g. ED1) and follow the complete Homework submission guidelines.

Please ask all questions about this assignment during the office hours, or post them on piazza.

All problems in this homework are design questions, so this homework is HARD. Therefore:

- Start early
- Come to office hours
- Elaborate your design clearly: follow the procedures we taught you in class.
- Tell us what each of your states mean in your state diagram
- Draw your circuit clearly (label DFFs, MUXs, etc)

This is not only a hard homework to do, but also a hard homework to grade. Be sure to label all states, transitions, inputs and outputs so we know what you mean. **If we cannot understand your design because it is not clearly labeled, you may not get points at all.**

Modular FSM Design and Memory

1. Serial adder

Draw a **datapath** (yes, only the datapath, not the FSM) of a circuit that adds two N-bit 2's complement binary numbers, A and B, stored in two N-bit right shift registers, R_A and R_B. Each shift register has serial output SO and serial input SI. Beginning with the least significant pair of bits in R_A and R_B, the circuit should add one pair at a time through a single fulladder (FA) circuit. The sum bit from the FA output should be stored back into shift register R_A.

Hint: You only need two N-bit shift registers, a single fulladder (FA) circuit, and a D flip-flop. Assume that the flip-flop starts in the 0 state.

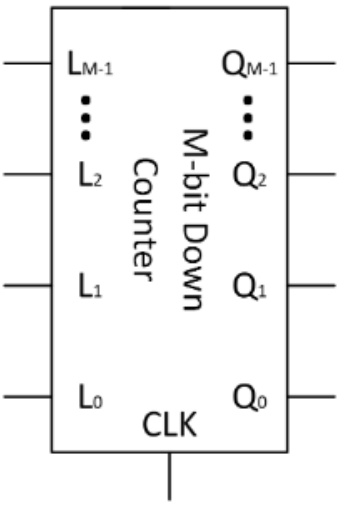
2. Counter-Based FSMs

Our goal is to design a traffic-light controller with the following properties; it lights up the green light (output G) for 24 seconds, followed by the yellow light (output Y) for 4 seconds, then the red light (output R) for 32 seconds. The controller will repeat this pattern forever. Our system will have a clock signal with a period of 4 seconds.

1. How many states, N, are needed in a FSM to implement this controller? In how many of these states will the light be green, yellow, and red, respectively? What is the minimum number of state bits, M, needed to implement this controller?

2. Draw the state diagram for a FSM implementing this controller. Label states in which the light is green as GREEN1, GREEN2, and so on, yellow as YELLOW1, and so on, red as RED1, and so on.

3. You will implement this FSM using a special down-counter that is available to you as a "black-box" circuit element. This M-bit counter (of the same minimum size you determined in Part 1) has the following properties: it normally counts down in binary by 1 each clock cycle; (2) after counting down to 0 (and holding that for the usual 1 clock period) it parallel loads as the next state the bit pattern L[M-1],...,L2,L1,L0, and then continues counting down from there. Otherwise, it ignores the Lx inputs. This device thus has outputs Q[M-1],...,Q2,Q1,Q0, and inputs CLK and L[M-1],...,L2,L1,L0. In your solution, draw it as a rectangular box with CLK (positive-edge-triggered) entering from the top, the L[i] inputs entering on the left side, and the Q[I] outputs on the right side, just like the picture below:



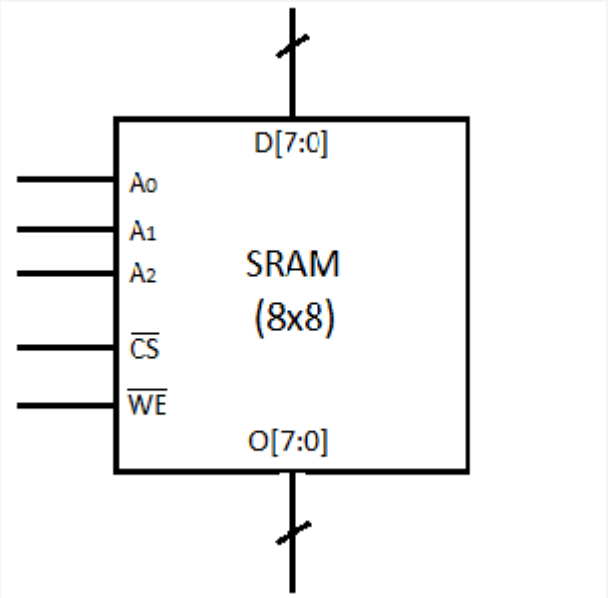
Assign the highest-numbered needed binary values (N1, N2, ...) to your green-light state(s), the next to your yellow state(s), and the lowest (...,2,1,0) to the red-light states. Based on this assignment, write the Karnaugh-maps for the three outputs G, Y, and R. Rather than filling unneeded state(s) with don't cares, choose a light color that ensures safe traffic control if the counter somehow enters an undefined state, and briefly explain why you chose what you did. Come up with expressions for each output that minimize area (use sum of literals and operators to approximate area). Consider SOP, POS, and using outputs to define other outputs. If you're careful, you can get the design down to four inverters and four larger gates (in addition to the down-counter). Sketch the circuit diagram for your solution, using the special down-counter as a circuit element that is already given to you; draw it as described above. (Points will be deducted for not using or not drawing this element as described!)

3. SRAMs

The following chip is a 8x8 bit RAM. Its logic symbol and function table are shown below:

Inputs		Operation	Outputs
CS'	WE'		
0	0	Write	hi-Z
0	1	Read	Stored Data
1	x	Inhibit	hi-Z

Logic Symbol



A_0-A_2 are the address lines, D_0-D_7 are the data lines, and O_0-O_7 are the data outputs. The outputs are gated with tri-state buffers and are in the high impedance (hi-Z) state whenever the Chip Select (CS') input is 1. They are available only in the Read mode. Chip Select (CS') and Write Enable (WE') inputs are activeLOW.

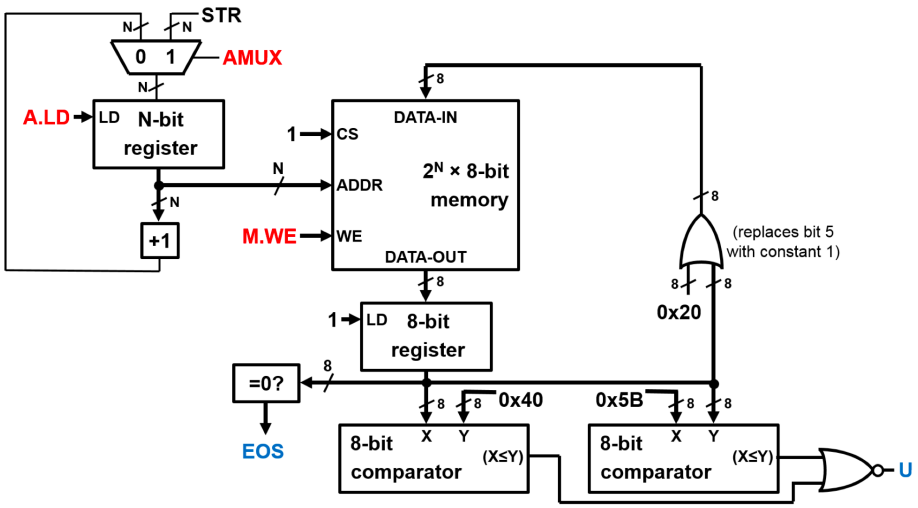
1. Using **only** two such RAM chips, implement an 8x16 bit RAM circuit.

2. Using **only** two such RAM chips and a decoder, implement a 16x8 bit RAM circuit.

3. Using **only** four such RAM chips and a decoder, implement a 16x16 bit RAM circuit.

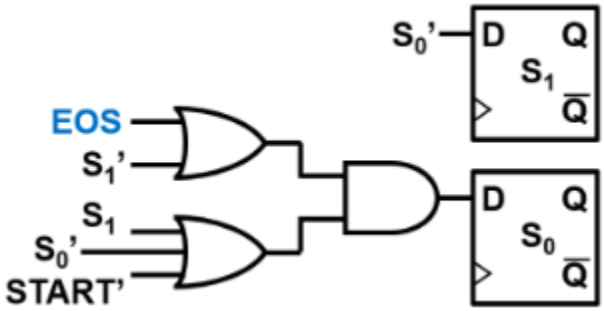
4. Converting Strings to Lower Case

The figure below shows the datapath for an FSM that converts any upper-case letters in an ASCII string to the corresponding lower-case letters. An ASCII string consists of a sequence of ASCII characters ending in 0x00, the ASCII NUL character.



Inputs for use by external logic, including the STR input, are shown in black. Datapath control signals appear in red. Datapath outputs are shown in blue.

The circuit below shows the next-state logic for the FSM that controls the datapath. The START signal is another external control signal used to tell the FSM to start changing the string beginning at the memory location specified by STR into lower case.



a) The system starts in a WAIT state (S_1S_0=01), allowing external logic to write a string into the memory (logic not shown). The external logic provides the starting address of the string through the STR input and raises the START input to make the FSM change any upper-case letters in the string to lower case.
Using the next-state logic and the description above, draw a state transition diagram for the FSM. Draw only those states reachable from the starting state. Transition arcs between states should be drawn without crossing, and should be labeled with the signals START and EOS. START comes from external logic, and EOS (end of string) comes from the datapath. A transition arc label of "10," for example, means that START=1 and EOS=0. Do not mark outputs in these states.

b) Based on the datapath, the state transition diagram from **part (a)**, and the desired operation of the FSM, calculate the three datapath control signals for each state (AMUX, A.LD, and M.WE). Note that you may need to use the datapath output signals EOS and/or U when calculating the control signals for the states.

c) Based on your answers to **part (b)**, find expressions with optimal area to express each datapath control signal (AMUX, A.LD, and M.WE) in terms of the current state S_1S_0 and the datapath output signals EOS and U.