CMSC 313 Spring 2024
Homework 6
due Monday, March 18, 11:59pm

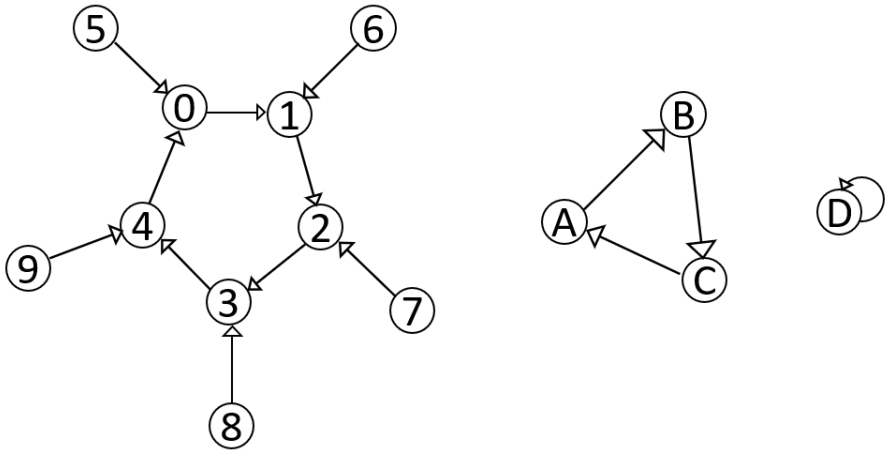
Please use Logisim to implement the following 3 exercises. You can only use the following gates: AND, OR, NOT, NAND, NOR, XOR, XNOR, and the following flip flops: D, T, JK, SR, and other components: clock, constants, splitters, MUX, DEMUX, DECODE. No other components can be used. Do not add any inputs or outputs; the Input/output (I/O) for each exercise is listed exactly.

This assignment introduces the concept of a logisim library. You are provided with 2 files: the driver file (which is for testing your circuit) and the homework file (which is the only one that you need to edit and submit). The homework file is used as a "library" in the driver file. Libraries are listed in the left hand panel in logisim.

In this assignment we also introduce the usage of ROM and RAM used for the testing tool. This circuit does not use test vectors to evaluate the circuit, instead it uses ROM and RAM. The testing drive circuits created for this assignment are labeled as "<circuit>_driver" in logisim. **You do not need to edit these driver circuits**. ROM stores the expected output, and the output of your circuit is compared and written to RAM. Your solution will get full credit if you get all 1s in RAM after simulating. You can inspect the contents of RAM and ROM by right clicking and selecting edit contents. Note, more bits than can be seen on the circuit block diagram may exist after right clicking and checking contents. You can speed up the simulation by adjusting the clock frequency by following the menu options: Simulate>Auto-Tick frequency. Trying to bypass the driver by hardcoding a solution in ROM or RAM will not work. We will use our own copy of the driver for testing and load your homework file in.

Only edit and only submit the file labeled "hw6_student.circ". Please rename your file to "<Firstname>_<Lastname>.circ" (underscore between first and last name).

Exercise 1. (20 pts) Implement the circuit "seqgen" with the following positive edge triggered synchronous sequence generator with the following graph. **The state values are in hexadecimal**. Note states 0xE and 0xF are unassigned and can transition to any next state including themselves.



Implement this state diagram with a circuit that has inputs that are exactly **clk**, **a[4]**, and **set**. The output is exactly **b[4]**. When **set**=1, the state of the circuit gets set to whatever value is currently in **a[4]** on a positive clock edge (If you're using JK flip flops, you have to set $J_A = 1$ and $K_A = 0$ to make $Q_A = 1$; $J_A = 0$ and $K_A = 1$ to make $Q_A = 0$. Choosing the right kind of flip flop may make this easier to implement). As long as **set**=1, the value in **a[4]** will keep updating the state in the circuit on every positive clock edge and won't let it progress to the next state. When **set**=0, on the clock edge, the state just moves to the next state. When the internal state updates, the state should be immediately visible on **b[4]**. With respect to the state: **a[0]** is the LSB and **a[3]** is the MSB; same for **b**.

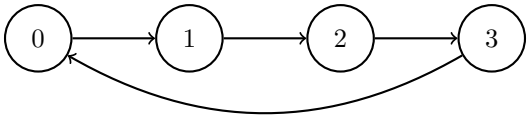
Your solution will get full credit if you get all 1s in RAM after testing.

If you are curious: the ROM in the "seqgen_driver" circuit uses the numbers 0x00 - 0x0F to represent "from" states, and numbers 0x10 - 0x1F to represent "to" states. Each value at 0x0i will transition to the next state 0x1i.

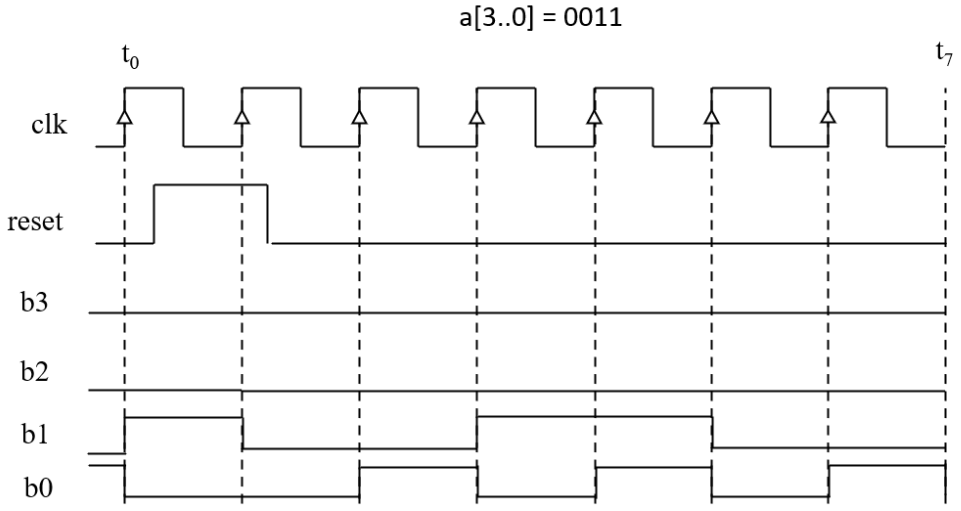
Exercise 2. (15 pts) Design the circuit "multicounter" that counts from 0 to a (inclusive) where a is between 0 and 15 (inclusive). The circuit inputs are exactly **clk**, **{a3,a2,a1,a0}** where a3 is the MSB (most significant bit) and a0 is the LSB (least significant bit), and **reset**. The outputs are exactly**{b3,b2,b1,b0}** where b3 is the MSB (most significant bit) and b0 is the LSB (least significant bit) of the counter. If the reset bit is high on the rising edge of a clock pulse, all values accessible on the b wires will reset to 0.

For example: if $a = 0$, i.e. $a3 = 0$, $a2 = 0$, $a1 = 0$, and $a0 = 0$, then $b = 0$ for every clock cycle, since to count from 0 to a, where $a = 0$, I will have a counter from 0 back to itself.

Another example: if $a = 3$ (i.e. $a3 = 0$, $a2 = 0$, $a1 = 1$, and $a0 = 1$), the circuit should count from 0 through 3 and implement the state diagram where b represents the state:



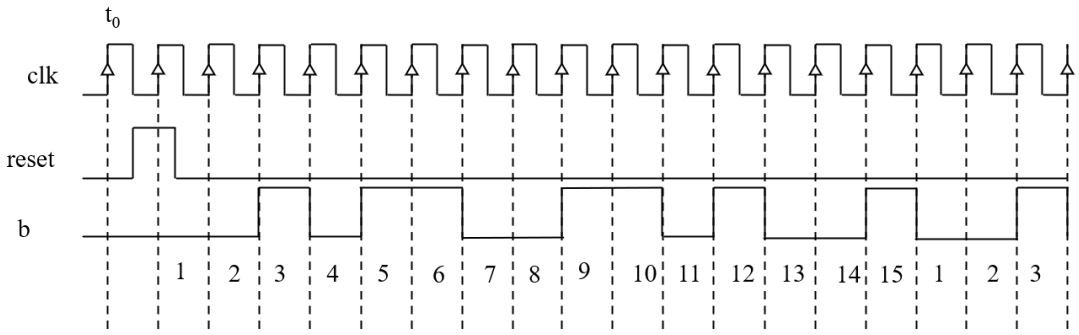
The timing diagram would look like this:



Your solution will get full credit if you get all 1s in RAM after testing.

Exercise 3. (15 pts) Design the circuit "foobar": the output is a single wire that goes high every 3rd clock cycle and high every 5th clock cycle, otherwise it is low. The circuit inputs are exactly **clk, reset** and the output is exactly **b**. If the reset bit is high on the rising edge of a clock pulse, the circuit will reset its internal counter to 1. *(hint: You can reuse the circuits from exercise 2 to build this, although there are multiple ways to implement this.)* Your internal counter system or way of maintaining state can be implemented in any way, presuming you only use the circuit components specified at the top of this assignment.

For example:



Your solution will get full credit if you get all 1s in RAM after testing.