CS2110 Spring 2014
Homework 4

Overview

For this assignment you will be working with memory and state machines. This is a 3-part assignment,

In Part 1, you will make your 8-bit register that functions exactly like the register in Logisim.

In Part 2, I will give you a simple state diagram, and you will have to build it in Logisim using the “onehot” style of building state machines. Do not minimize it.

In Part 3, I will give you the same state diagram and ask you to minimize the logic. You will have to use K-maps to do so. I will ask you to also submit your K-maps, so be sure to do them! After you have minimized the logic, you will implement the circuit in Logisim.

In each of the parts for this homework assignment, you may use **ONLY** 1 register (or the appropriate number of D flip-flops). You DO NOT have to use the register you made in Part 1 for Parts 2 and 3. You can use the register in Logisim.

Part 1

You must make an 8-bit register that functions exactly like a register in Logisim. This includes:

8-bit data input (D):
This is where data is input into the register.

8-bit data output (Q):
This is where data is output from the register.

1-bit clock input (CLK):
This is the input for the clock pulse.

1-bit enable input (EN):
When this is 0, clock triggers are ineffective. When 1, the register stores the value on D when the CLK signal is asserted.

1-bit reset input (RES):
This ASYNCHRONOUSLY resets the register to all 0's.

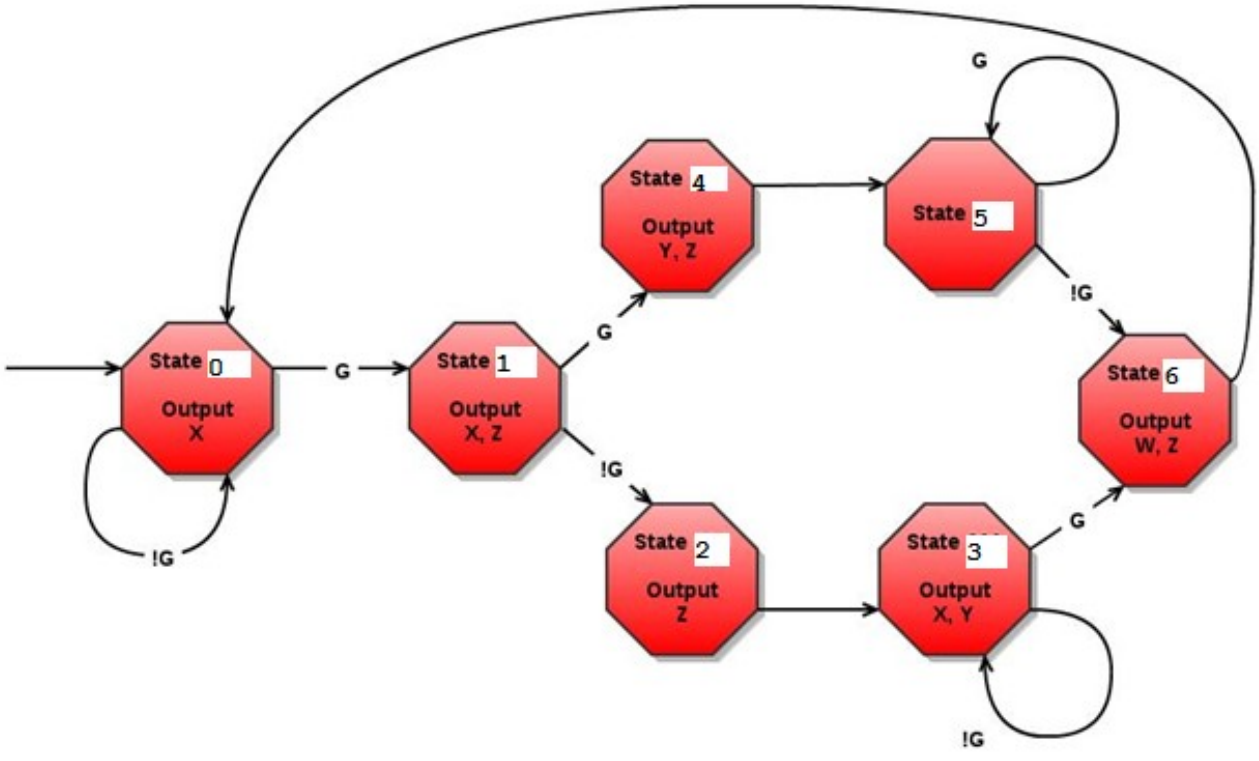
Do not fear! Much of this functionality is built into the **D Flip-Flops** you will be using to build this register. You simply need to connect the wires into the right places.

Restrictions – The only things you are allowed to use here are inputs, outputs, wires, splitters, gates, and D-Flip-Flops. ALSO it is important that you make the 8-bit inputs and outputs as multi-bit inputs/outputs, NOT 8 individual inputs and outputs. We WILL take off (a few) points if you do not do it this way.

Put this in the template file provided called hw4part1.circ

Part 2

Take a look at this state machine transition diagram.



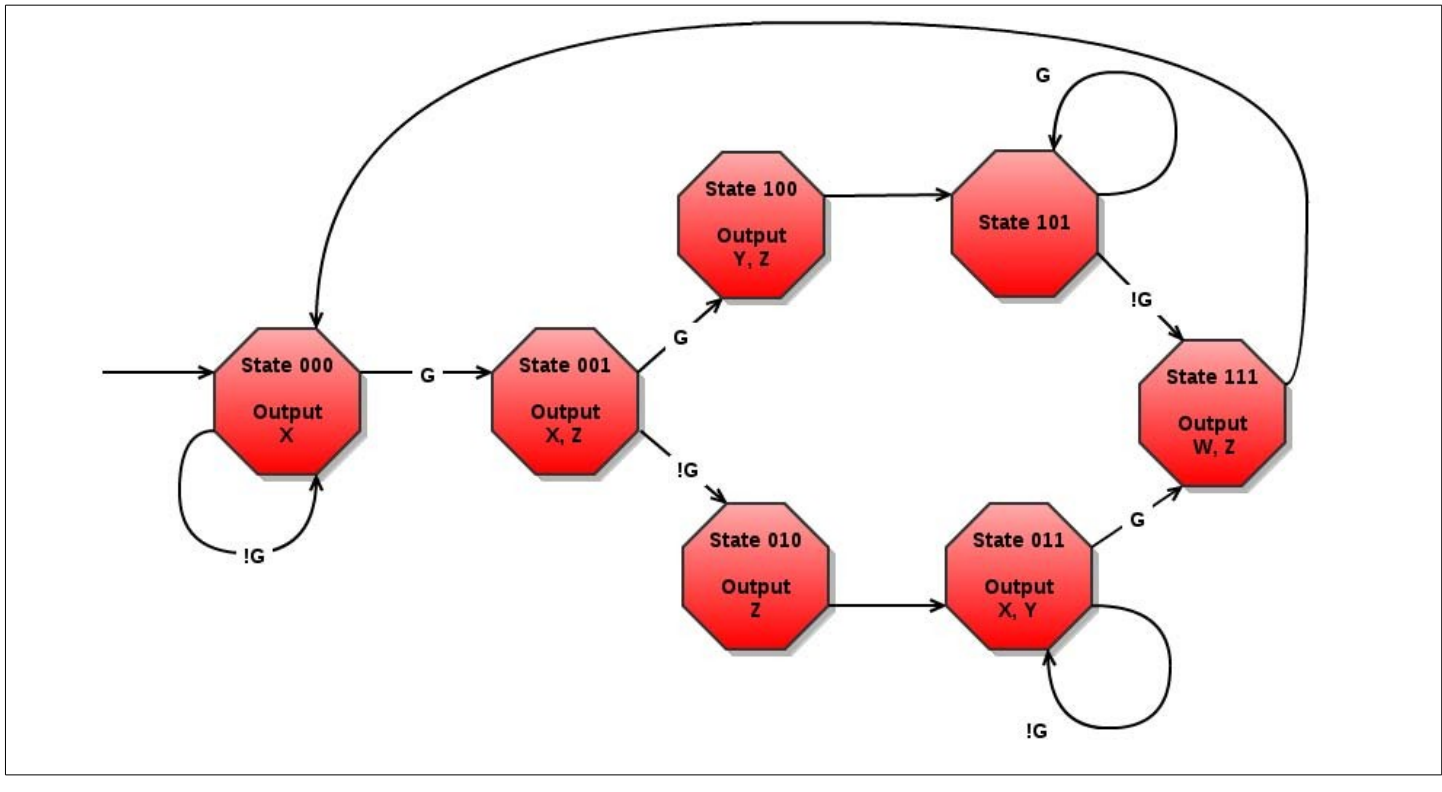
You will be implementing this state transition diagram as a circuit using the **onehot** style. Remember for onehot you will have a register with # bits being the number of states you have. Each bit corresponds to a state, and you are in that state if the corresponding bit is a 1. Only one of these bits will be on at a time (the only exception being when the state machine starts up). At most, one of the inputs will be turned on.

You must implement this using onehot. A template file hw4part2.circ has been given to you. Implement the state machine in the subcircuit provided.

Part 3

Take a look at the following state diagram.

You will be required to make a K-map and minimize the logic here.
• First, convert this state diagram to a truth table.
• Next, produce a K-map from the truth table. You will need one K-map per output and one per next state bit for a total of 7 K-maps (s2,s1,s0,X,Y,Z,W). Please be clear in labeling your K-maps. In the above diagram, the binary numbers on each state represent the state number. You must use these numbers in your K-map. For instance, if you see the number 0, 1, 1 then s2 = 0, s1 = 1, s0 = 1. This means that s2 is the most significant bit. Yes we know that the state 110 isn't shown above. This is an invalid state, and we don't care about the behavior of these two states nor



which state it goes to next. Use both of those when making your K-map. In your K-map, you must circle the groups (or highlight the groups) you have elected to use in your minimized SUM OF PRODUCTS expression. **Your K-map must give the BEST minimization possible to receive full credit. This means you must select the BEST values for the don't cares in your K-maps to do this.**
• Implement this circuit by adding onto the template file hw5part3.circ template already provided for you. You will lose points if your circuit does not correspond to your K-map or if your circuit is not minimal. You should use only the minimal components possible to implement the state machine.
• For the outputs in your K-map. **The outputs should depend only upon the current state you are in.** That is if the state is 001 and G is pressed, then X and Z should be on. If the current state is 001 and G is not pressed, then X and Z should be on. You see, the output does not depend on the G input, but only what state you are in.

• Your K-map must also choose the BEST values for the don't cares to achieve the minimal logic necessary to implement the state diagram as a circuit. For instance consider the following K-map:

AB\CD	00	01	11	10
00	0	0	0	0
01	0	1	X	0
11	0	X	1	X
10	0	0	0	0

Doing this the naïve way you would select the 2 1's in this K-map and be done, however this is not minimal. The X's in this K-map are don't cares, meaning you don't care if it is a 1 or a 0. You can use these don't cares to make your expression even more minimal. For this K-map you can let the X's in the center be 1's this way you can select a 2x2 square in the center. Also note the other don't care we will let be 0 as you do not get a better minimization by having it be a 1. You must do your minimization this way or else your K-maps will be wrong and you will lose a lot of points. For more information about don't cares and simplification, you can read the document under resources Boolean Algebra and Simplification / 06Simplification.pdf .

Summary of the rules

Failure to follow these will result in a heavy point deduction, if not a zero, for this part. **This is your only warning!**

1. Outputs are ONLY to depend upon the current state.
2. Your K-maps should show the labeled groups and the minimized boolean expression as a sum of products. Do not factor anything else out. Your expression should not include any parenthesis, and the circuit shouldn't use XOR gates as part of the logic.
3. Your K-maps should give the BEST minimization you should choose the BEST values for all of the don't cares.
4. States 110 is not used so we DON'T CARE what is outputted or what the next state is.
5. Your K-map should match your circuit.
6. The left most bit of the state is s2, the right most is s0.
7. You are to do your K-map on paper (and scan it) or in Excel/LibreOffice Calc.
8. Make sure your name is clearly visible in your file here too.

Deliverables

Part 1
Your modified version of hw4part1.circ

Part 2
Your modified version of hw4part2.circ

Part 3
1. Truth table and K-maps complete with circled groups and your sum of products expressions. Again, please keep the file size low if you have scanned it.
2. The modified hw4part3.circ file that implements your minimized state machine from the K-Maps