CS2110 Summer 2014
Homework 4

This assignment is due by:

Day: Tuesday June 10th
Time: 11:54:59pm

Rules and Regulations

Academic Misconduct
Academic misconduct is taken very seriously in this class. Homework assignments are collaborative. However, each of these assignments should be coded by you and only you. This means you may not copy code from your peers, someone who has already taken this course, or from the Internet. You may work with others **who are enrolled in the course,** but each student should be turning in their own version of the assignment. We will be using automated code analysis and comparison tools to enforce these rules. **If you are caught you will receive a zero and will be reported to Dean of Students.**

Submission Guidelines
1. You are responsible for turning in assignments on time. This includes allowing for unforeseen circumstances. If you have an emergency let us know ***IN ADVANCE*** of the due time supplying documentation (i.e. note from the dean, doctor's note, etc). Extensions will only be granted to those who contact us in advance of the deadline and no extensions will be made after the due date.

2. You are also responsible for ensuring that what you turned in is what you meant to turn in. No excuses, what you turn in is what we grade. In addition, your assignment must be turned in via T-Square. When you submit the assignment you should get an email from T-Square telling you that you submitted the assignment. If you do not get this email that means that you did not complete the submission process correctly. Under no circumstances whatsoever we will accept any email submission of an assignment. Note: if you were granted an extension you will still turn in the assignment over T-Square.

3. There is a random grace period added to all assignments and the TA who posts the assignment determines it. The grace period will last **at least one hour** and may be up to 6 hours and can end on a 5 minute interval; therefore, you are guaranteed to be able to submit your assignment before 12:55AM and you may have up to 5:55AM. As stated it can end on a 5 minute interval so valid ending times are 1AM, 1:05AM, 1:10AM, etc. **Do not ask us what the grace period is we will not tell you**. *So what you should take from this is not to start assignments on the last day and depend on this grace period past 12:55AM.* There is no late penalty for submitting within the grace period. If you can not submit your assignment on T-Square due to the grace period ending then you will receive a zero, no exceptions.

General Rules
1. Any code you write (if any) must be clearly commented and the comments must be meaningful. You should comment your code in terms of the algorithm you are implementing we all know what the line of code does.
2. Although you may ask TAs for clarification, you are ultimately responsible for what you submit.

3. Please read the assignment in its entirety before asking questions.
4. Please start assignments early, and ask for help early. Do not email us the night the assignment is due with questions.
5. If you find any problems with the assignment it would be greatly appreciated if you reported them to the author (which can be found at the top of the assignment). Announcements will be posted if the assignment changes.

Submission Conventions

1. Failure to follow these may result in a max of 5 points taken off

2. All files you submit for assignments in this course should have your name at the top of the file as a comment for any source code file, and somewhere in the file, near the top, for other files unless otherwise noted.
3. When preparing your submission you may either submit the files individually to T-Square or you may submit an archive (zip or tar.gz only please) of the files (preferred). You can create an archive by right clicking on files and selecting the appropriate compress option in on your system.
4. If you choose to submit an archive please don't zip up a folder with the files, only submit an archive of the files we want. (See Deliverables).
5. Do not submit compiled files that is .class files for Java code and .o files for C code. Only submit the files we ask for in the assignment.
6. Do not submit links to files. We will not grade assignments submitted this way as it is easy to change the files after the submission period ends.

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. We 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.

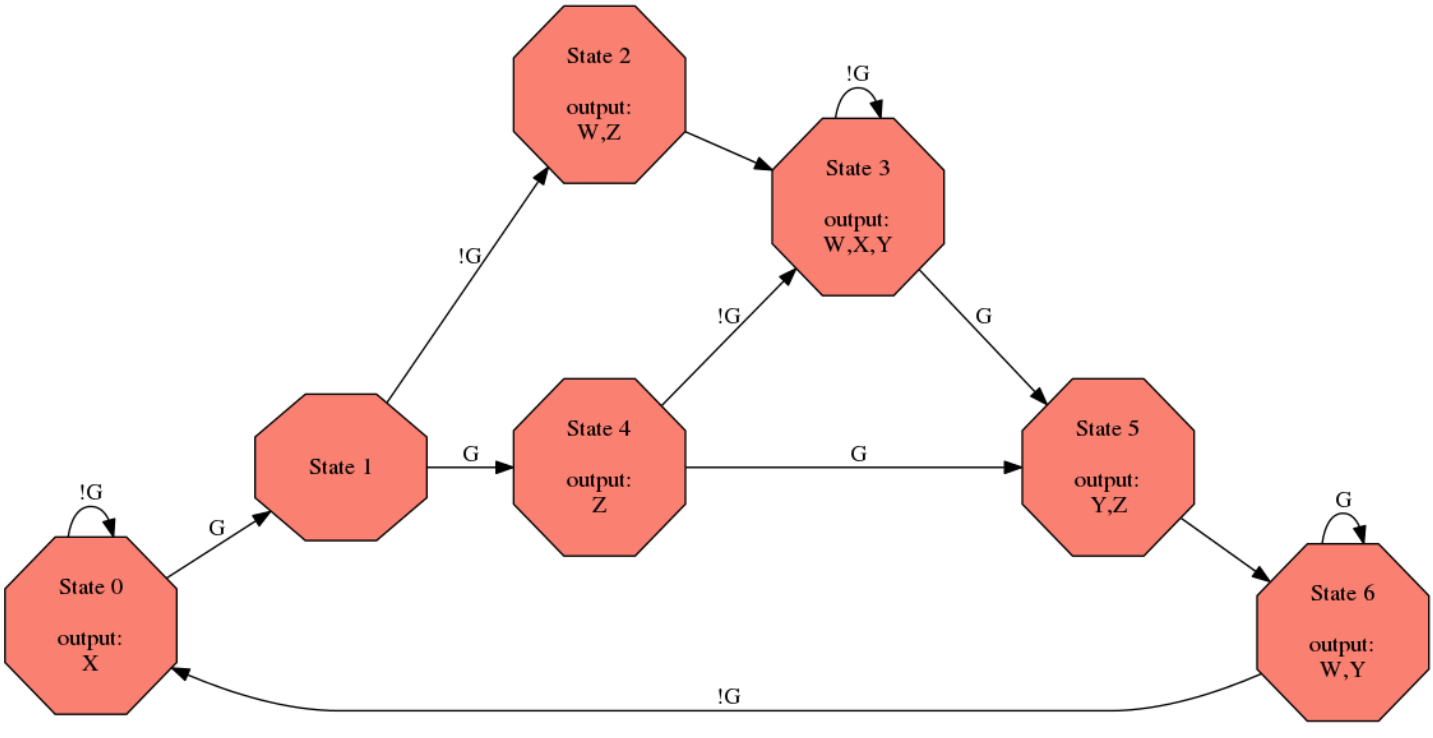
Do not fear! Much of this functionality is built into the **D Flip-Flops** you will be using to build this register. You simply need to connect the wires into the right places.

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.

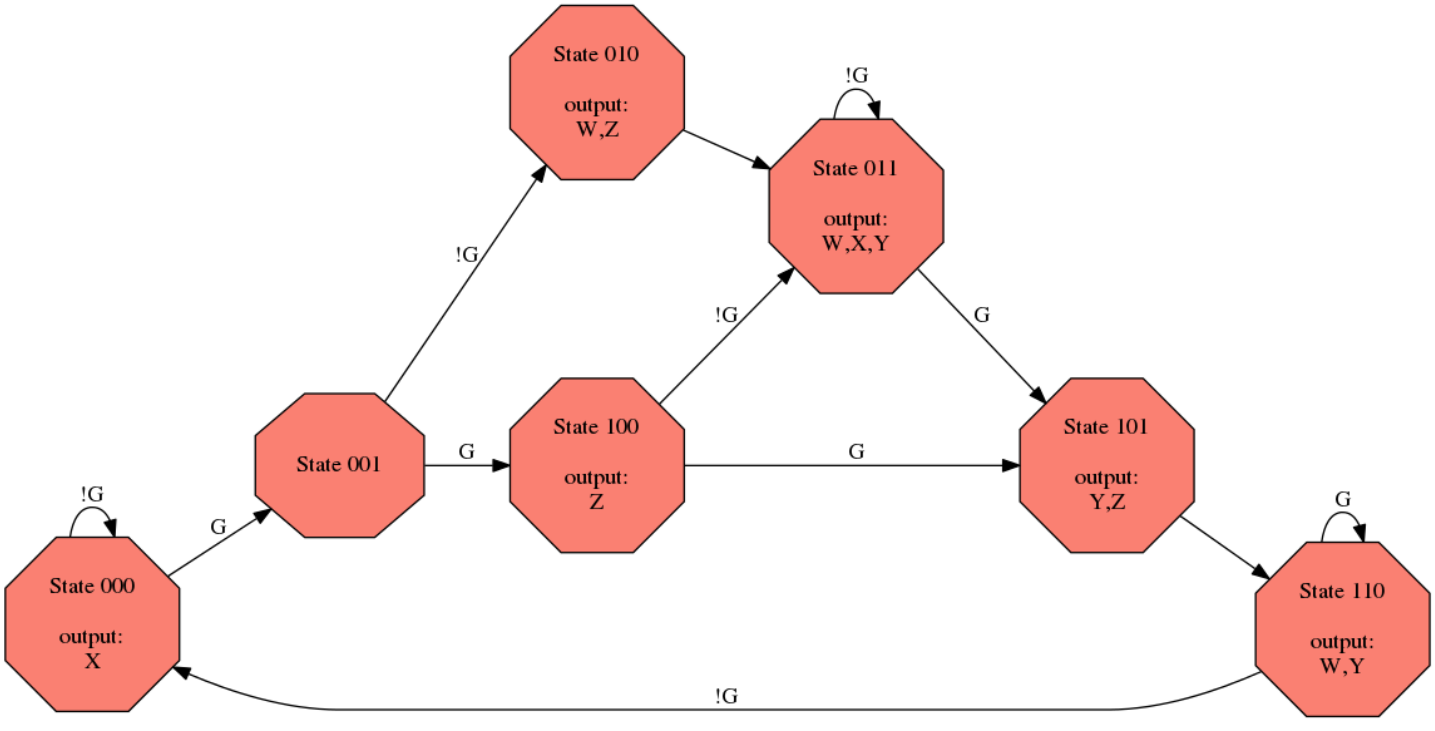


You will be implementing this state transition diagram as a circuit using the **onehot** style. Remember for onehot you will have a register with # bits being the number of states you have. Each bit corresponds to a state, and you are in that state if the corresponding bit is a 1. Only one of these bits 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 011 then S2 = 0, S1 = 1, S0 = 1. This means that S2 is the most significant bit. Note that the state 111 isn't shown above. This is an invalid state, and we don't care about the behavior of this state nor which state it goes to next. Use this 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 hw4part3.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 011 and G is pressed, then W, X, and Y should be on. If the current state is 001 and G is not pressed, then W, X, and Y 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 (that is unrelated to the state machine we gave you):

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 two 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.

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 111 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 take a clear picture of it) or in Excel/LibreOffice Calc.
8. Make sure your name is clearly visible in all of your files (including K-maps).

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.
2. The modified hw4part3.circ file that implements your minimized state machine from the K-Maps