CS2110 Spring 2012
Homework

Version: 2

This assignment is due by:

Day: February 7th, 2012
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. Be very careful when supplying your work to a classmate that promises just to look at it. If he/she turns it in as his own you will both be charged.

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 also 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. In addition 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.

Submission Conventions

1. 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.
2. When preparing your submission you may either submit the files individually to T-Square (preferred) or you may submit an archive (zip or tar.gz only please) of the files.
3. 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.
4. Do not submit compiled files that is .class files for Java code and .o files for C code.
5. 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 state machines. This is a 2 part assignment, for the first part we will be asking you to draw **ONLY** the state diagram for a pinball machine based on behavior which will be described below. Be sure to account for all states, transitions, inputs, and outputs. You will be submitting some sort of image of the state diagram, you can use third party software to help draw up your diagrams as long as you submit an image file.

In Part 2, we will give you the 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.

Part 1

Please construct and turn in an image of a state diagram for a pinball machine that has the following behavior:

Symbols

Inputs: N = Nickel placed in coin slot

D = dime placed in coin slott

BIM = a ball is in motion, coming out of the queue

BOOP = ball out of play. This means the user has lost a ball through the bottom.

HB = Has ball. This input is on so long as the user has not used up all of the balls.

Outputs: SL = score light

ICL = insert coin light.

BG = Ball gate, this output opens the ball gate blocking the queue.

1. The machine initially starts by waiting for the user to put coins in. The insert coin light (ICL) should be lit up while the machine still needs coins. The pinball machine takes nickels and dimes, and it takes 15 cents to start a game (Don't worry about change, if user puts in more than 15 cents, like 2 dimes, then the machine keeps it). **Check the 5 cent coke problem below! It will take a few cycles of the clock for the coin to roll past the sensor, use dummy states!**
2. Once 15 cents is put in, the ICL no longer lights up, the score light (SL) turns on, and the state machine must release a ball from the ball queue. This is where the ball gate (BG) must open. The BG must stay open while the first ball is rolling out of the queue. The ball in motion (BIM) input to the state machine will be turned on during this time. While the ball is in motion, the gate must remain open.
3. From here, the user will fire the ball and play the game (playing the game and firing the ball are all mechanical, you don't worry about it), during gameplay, the (SL) must always be on to illuminate the score. You don't worry about keeping score, other mechanisms on the machine do this automatically. The score light is there to make the score visible on the board mounted at the back of the pinball machine.
4. Once the user loses the ball through the bottom, the ball out of play (BOOP) censor becomes a 1, and the state machine will have to open the ball gate to put another ball into play if there are any balls available in the queue. If not, the game is over and the machine will require more coins before . Assume the BOOP turns off automatically whenever the ball gate opens again.

5. As long as the user has at least one ball left, the has ball signal (HB) will be turned on. If the user loses a ball and the HB is off, meaning they have no more balls left, the game is over.

Your state diagram should use the signals given in the legend above (N, D, BOOP, and so on). **You must mark your start state on the diagram. You must show the outputs for each state. You must put conditions are your transitions where appropriate You must label your states**. Do this!

An Aside: 5 Cent Coke Machine

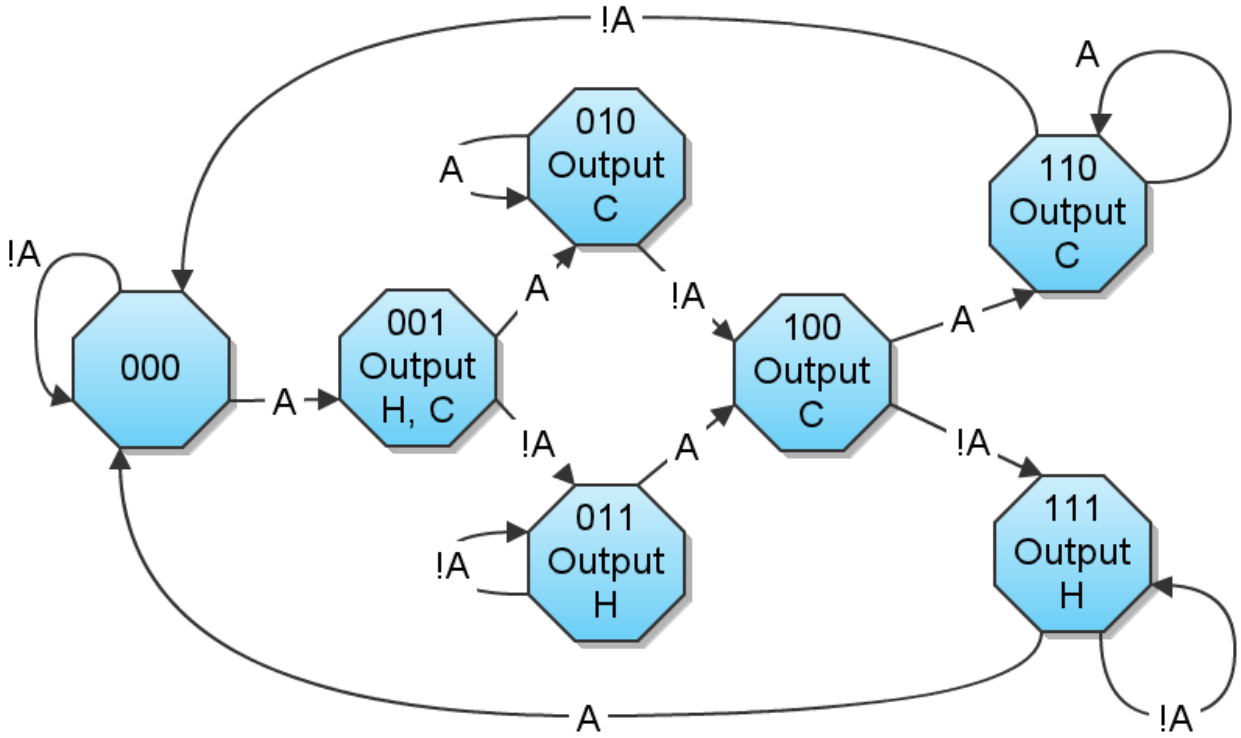
Imagine an FSM for a coke vending machine. The machine outputs a coke after it gets a nickel. A naïve approach would be to have 2 states, A and B. A waits for a nickel. When a nickel is received, it transitions to state B. State B has an output of a cola and then unconditionally transitions back to A to wait for another nickel. This is bad.

The clock on the state machine is pulsing many, many times. Let's assume that in the fraction of a second it takes for the coin to travel past the coin sensor, the clock pulses 10 times (a conservative estimate at best!). This means that a transition will occur from A to B because the sensor reads a nickel, then in the next cycle, the machine will output a coke. In the third cycle, the sensor still reads that a nickel is being inserted though, because the nickel has not cleared the sensor. So a transition from A to B occurs yet again and another coke is given to the customer. This is very bad!

So how do we remedy this problem? We do it through dummy states. We place a state in between state A and B called dummy. The transition condition to get from A to dummy is the presence of a nickel, let's call this signal N. The dummy state has a self-transition with a condition of N again. This means that while the nickel is moving past the sensor, we are in this dummy state. The only transition out of the dummy state occurs in the absence of a nickel, let's refer to this as !N. The transition out of the dummy state transitions into state B, where precisely 1 coke is dispensed. The idea here is to keep the FSM in a dummy state while we account for inputs that may be on for multiple clock cycles but should only be counted once.

Part 2

Take a look at the following state diagram.



The outputs depend **ONLY** on the current state. For instance, if we are in state 011, then we output H during the same clock cycle.

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 for a total of 5 k-Maps (s0,s1,s2,H,C).** 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 s0 = 0, s1 = 1, s2 = 1. This means that s0 is the most significant bit. **Use this rule when making your k-map.** In your K-map, you must circle the groups you have elected to use in your minimized **SUM OF PRODUCTS** expression.

You will then implement this circuit by adding onto the .circ template already provided for you. You will lose points if your circuit does not correspond to your K-map!

Deliverables

Part 1:

An image of your state diagram complete with labels for states and transitions. You must also mark your start state! Also, be use the legend we have provided in this file for your signals (N, D, BIM, BOOP, HB, SL, ICL, BG)!

Part 2:

1. Truth table and K-maps complete with circled groups and your sum of products expression

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