CS 2110 Homework 4
State Machines

Benjamin Yarmowich, Vivian Thiebaut, Farzam Tafreshian, Jarrett Schultz

Spring 2019

Contents

1 Overview

For this assignment, we'll be working with Flip Flops, state machines and K-maps. This will be a 3 part assignment.

Objectives:
1. Understand how information is stored in a computer
2. To learn how to make a state machine
3. To become familiar with different state machine style
4. To understand K-maps and their usefulness

2 Instructions

Part 1: For this part of the assignment you will build your own register from scratch.

- Implement your circuits in the "latches.sim" file

Part 2: Given a simple state diagram, you will build a state machine in CircuitSim using the "one-hot" style of building state machines.

- The circuit will be implemented in the "One-Hot FSM" subcircuit of the "fsm.sim" file

Part 3: Given the same state diagram from part 1, you will be minimizing the logic by using K-Maps.

- Fill out the truth table and respective K-Maps located in "KMap.pdf"
- The reduced circuit will be implemented in the "Reduced FSM" subcircuit of the "fsm.sim" file

In general, do not change/delete any of the input/output pins.
For parts 2 and 3 of this homework, you must use exactly 1 register. These are found under the Memory tab in CircuitSim. Failure to do so may result in large point deductions.

2.1 Part 1

For this part of the assignment you will build your own register from the ground up. For more information about each subcircuit refer to Patt & Patel textbook, pages 65, 66, and 80.

2.1.1 RS Latch

As described on page 65 of Patt & Patel, we will be building a RS latch using NAND gates.

- Build your circuit in the "RS Latch" subcircuit in the "latches.sim" file

2.1.2 Gated D Latch

Using your RS latch subcircuit, implement a Gated D Latch as described on page 66 of the Patt & Patel textbook. Notice that the Gated D Latch only needs one output, so you should disregard the inverse output of your RS Latch.

- Implement this circuit in the "Gated D Latch" subcircuit in the "latches.sim" file
- You are not allowed to use the built-in SR Flip-Flop in CircuitSim to build this circuit

2.1.3 D Flip-Flop

Using the Gated D Latch, create a D Flip-Flop. Your D Flip-Flop output should be able to change on the **rising edge**, which means that the state of the register should only be able to change at the exact instant the clock goes from 0 to 1.

- Implement this circuit in the "D Flip-Flop" subcircuit in the "latches.sim" file.

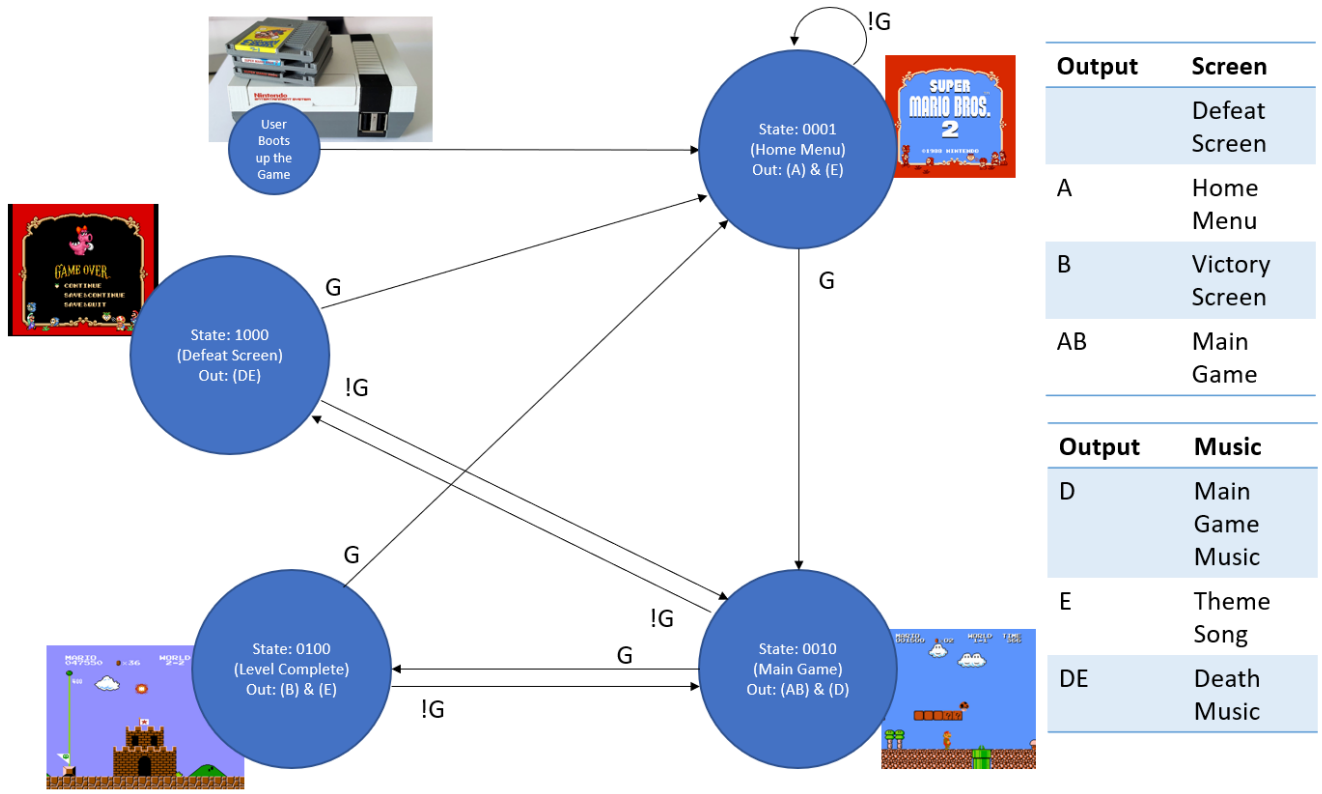
2.1.4 Register

Using the D Flip-Flop you just created, build a 4-bit Register. Your register should also use edge-triggered logic.

- This circuit will be implemented in the "Register" subcircuit in the "latches.sim" file

2.2 Part 2

Take a look at this state machine transition diagram. We are using an example of a GameBoy game to explain the diagram. The input G represents a button that carries the player through all the game states. The outputs A, B, D, E represent different game attributes that occur based on the current game state (Ex. Output D represents the main game music being played).



Output	Screen
	Defeat Screen
A	Home Menu
B	Victory Screen
AB	Main Game

Output	Music
D	Main Game Music
E	Theme Song
DE	Death Music

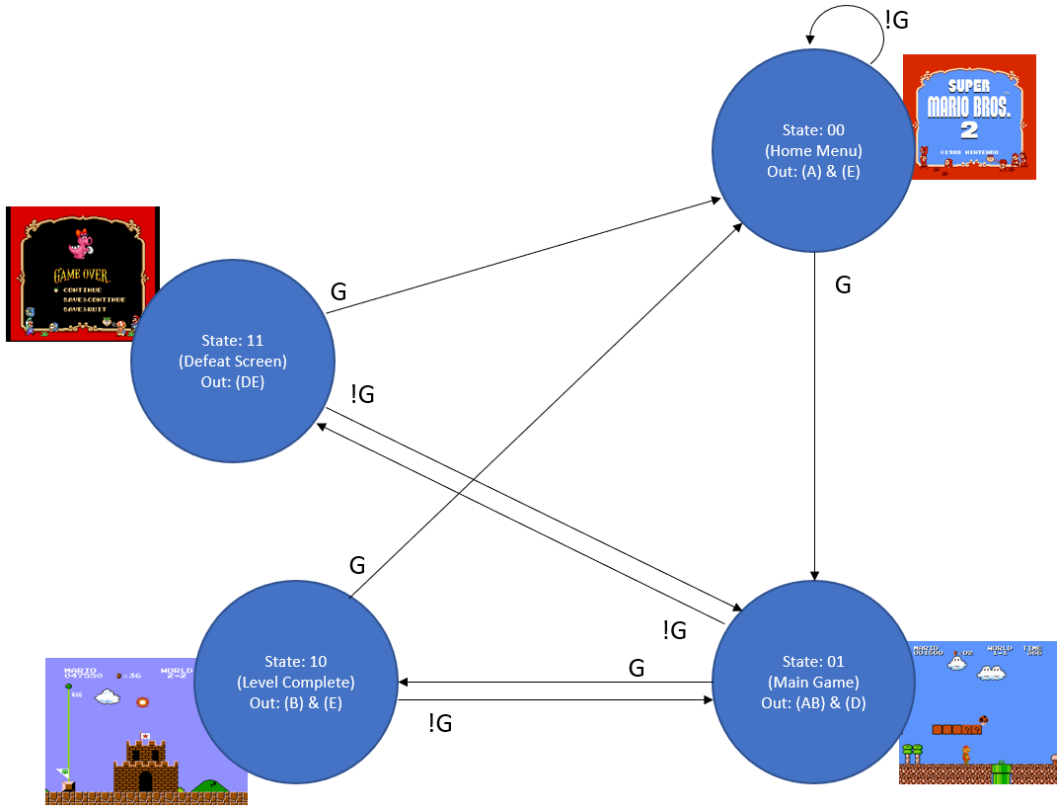
- You will be implementing this state transition diagram as a circuit using the one-hot style. Remember for one-hot you will have a register with the number of bits being the number of states you have. Each bit corresponds to a state, and you are in that state if the corresponding bit is a 1. Only one 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 will need to implement a reset input for this circuit, which will return your state machine to its Start state. You may design this so that the reset happens either immediately when the reset button is pressed (asynchronously) or when it is on during the rising edge of the next clock cycle.
- You must implement this using one-hot. A template file fsm.sim has been given to you. Implement the state machine in the provided "one-hot state machine" subcircuit.
- Note that there are 3 inputs to the "one-hot state machine" subcircuit: CLK, G, and RST. The CLK input turning off and on repeatedly will be used to represent clock ticks in your circuit. The G input corresponds to whether or not G is on, as in the diagram above. RST corresponds to an attempt to reset your circuit.

2.3 Part 3

Take a look at this state machine transition diagram. Again, we are using an example of a GameBoy game to explain the diagram. The input G represents a button that carries the player through all the game states. The outputs A, B, D, E represent different game attributes that occur based on the current game state (Ex. Output D represents the main game music being played).



Output	Screen
	Defeat Screen
A	Home Menu
B	Victory Screen
AB	Main Game

Output	Music
D	Main Game Music
E	Theme Song
DE	Death Music

- First, convert this state diagram to a truth table and fill out the corresponding truth table on the first page of the KMap pdf.

  The inputs for the truth table are:

  - C_0 = Current State 0th bit (least significant)

- C_1 = Current State 1st bit (most significant)
- G

The outputs for the truth table are:

- N_0 = Next State 0th bit (least significant)
- N_1 = Next State 1st bit (most significant)
- A, B, D, E

Please Note: This State Machine is a Moore State Machine, meaning that the output values are determined solely by the current state (you should not use the N_1 and N_0 outputs for determining the values for A, B, D, E. **Use the character 'x' to denote don't care states**

- Next, produce a K-map from the truth table on paper.
  - You will fill out one K-map per output and one per next state bit for a total of 6 K-maps (N_1, N_0, A, B, D, E). The respective K-maps are located on the second page of the K-map pdf.
  - 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. Use these groupings to write the reduced expressions next to each respective K-Map
  - 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.
  - It may be helpful to check with others on piazza to see if your circuit is optimal. In order to do this without giving away your answer you may share the number of AND and OR gates used. For example in the K-map below if you thought the answer was BD + ABC then you might ask if someone got better than 2 + 3.

- Implement this circuit in the "reduced state machine" subcircuit of the provided fsm.sim file. 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.

3 Testing

To test locally, navigate to the directory with the latches.sim and fsm.sim files and run the tester jar file via

```
java -jar hw04-checker.jar
```

4 Deliverables

You will need to submit latches.sim, fsm.sim to the Homework 04 assignment on Gradescope. **Scan and submit your K-map pdf as one pdf file** to the Homework 04 K-Map/Truth Table assignment on Gradescope. **Please ensure that the scan of your K-Map pdf is readable and takes up the entire page. If we cannot read your pdf, you will not receive credit for this portion of the assignment.**

The sim files will be autograded on gradescope.

Note: The checker may not reflect your actual grade on this assignment. We reserve the right to update the checker as we see fit when grading.

5 Demos

This homework will be demoed.
The demos will be ten minutes long and will occur in the CS2110 TA lab - COC 104b during the week of February 11th - February 15th.

- Demo signups will go up by February 8th and you can sign up for a demo via the calendar on Canvas
- You must sign up for your demo at least 24 hours in advance
- If you cannot make any of the available demo signups, email the Head TA at mbrickell3@gatech.edu **by Monday February 11th**. This is the only way you can ensure you will have a demo.
- Your demo is 30 points of your Homework 04 grade. If you miss your demo you will not receive these points and the maximum you can receive on the homework is a 70%.
- You cannot cancel your demo within 24 hours or else you will lose all 30 of your demo points.
- You will be able to makeup one of your demos at the end of the semester for 50% of the demo grade.

6 Rules and Regulations

6.1 General Rules

1. Starting with the assembly homeworks, any code you write must be meaningfully commented. You should comment your code in terms of the algorithm you are implementing; we all know what each line of code does.
2. Although you may ask TAs for clarification, you are ultimately responsible for what you submit. This means that (in the case of demos) you should come prepared to explain to the TA how any piece of code you submitted works, even if you copied it from the book or read about it on the internet.
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.

6.2 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 unless otherwise noted.
2. When preparing your submission you may either submit the files individually to Canvas/Gradescope or you may submit an archive (zip or tar.gz only please) of the files. You can create an archive by right clicking on files and selecting the appropriate compress option on your system. Both ways (uploading raw files or an archive) are exactly equivalent, so choose whichever is most convenient for you.
3. 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.
4. Do not submit links to files. The autograder does not understand it, and we will not manually grade assignments submitted this way as it is easy to change the files after the submission period ends.

6.3 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. After submitting you should be sure to download your submission into a brand new folder and test if it works. No excuses if you submit the wrong files, what you turn in is what we grade. In addition, your assignment must be turned in via Canvas/Gradescope. 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 Canvas/Gradescope.

3. There is a 6-hour grace period added to all assignments. You may submit your assignment without penalty up until 11:55PM, or with 25% penalty up until 5:55AM. So what you should take from this is not to start assignments on the last day and plan to submit right at 11:54AM. You alone are responsible for submitting your homework before the grace period begins or ends; neither Canvas/Gradescope, nor your flaky internet are to blame if you are unable to submit because you banked on your computer working up until 11:54PM. The penalty for submitting during the grace period (25%) or after (no credit) is non-negotiable.

6.4 Syllabus Excerpt on Academic Misconduct

Academic misconduct is taken very seriously in this class. Quizzes, timed labs and the final examination are individual work.

Homework assignments are collaborative, In addition many if not all homework assignments will be evaluated via demo or code review. During this evaluation, you will be expected to be able to explain every aspect of your submission. Homework assignments will also be examined using computer programs to find evidence of unauthorized collaboration.

What is unauthorized collaboration? Each individual programming assignment should be coded by you. You may work with others, but each student should be turning in their own version of the assignment. Submissions that are essentially identical will receive a zero and will be sent to the Dean of Students' Office of Academic Integrity. Submissions that are copies that have been superficially modified to conceal that they are copies are also considered unauthorized collaboration.

You are expressly forbidden to supply a copy of your homework to another student via electronic means. This includes simply e-mailing it to them so they can look at it. If you supply an electronic copy of your homework to another student and they are charged with copying, you will also be charged. This includes storing your code on any site which would allow other parties to obtain your code such as but not limited to public repositories (Github), pastebin, etc. If you would like to use version control, use github.gatech.edu

6.5 Is collaboration allowed?

Collaboration is allowed on a high level, meaning that you may discuss design points and concepts relevant to the homework with your peers, share algorithms and pseudo-code, as well as help each other debug code. What you shouldn't be doing, however, is pair programming where you collaborate with each other on a single instance of the code. Furthermore, sending an electronic copy of your homework to another student for them to look at and figure out what is wrong with their code is not an acceptable way to help them, because it is frequently the case that the recipient will simply modify the code and submit it as their own. Consider instead using a screen-sharing collaboration app, such as `http://webex.gatech.edu/`, to help someone with debugging if you're not in the same room.

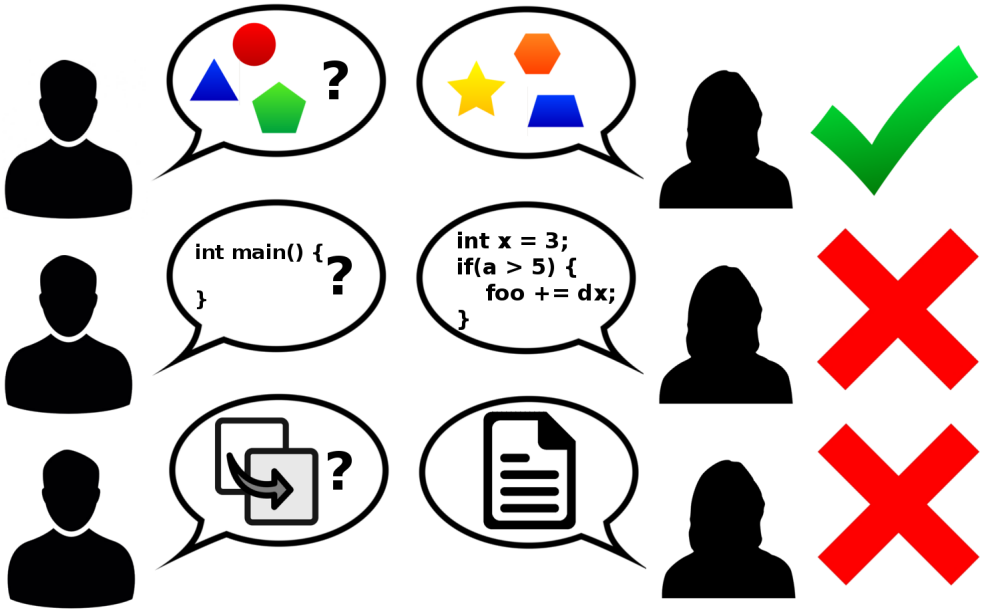


Figure 1: Collaboration rules, explained colorfully