CSO-Recitation 13

CSCI-UA 0201-007

R13: Assessment 11 & Assessment 12 & Lab5 & FSM

Today’s Topics

- Assessment 11
- Assessment 12
- Lab5
- Sequential Logic, FSM

Before we start…

- Lab5 due is postponed to Dec 14
- Lab6 is changed to be optional..
  - If completed, it gives you an extra 5% of your total grade
- A final-assessment on the final week
  - 5% of your total grade
  - just another assessment, not a final exam thing, don’t worry~

Lab5

using Logisim Evolution

Setup

- Clone the Lab 5 repository – new link (campuswire)
- cd lab5new-<YourGithubUsername>
- wget https://nyu-cso.github.io/labs/logisim-evolution-2.15.jar
- download the Java runtime environment:
  - sudo apt-get install default-jre
- Launch Logisim by typing (in your lab5new repo):
  - java -jar logisim-evolution-2.15.jar

- Save frequently when you work~

Part 2 exercise FSM

- There are 4 states
- what you write on the arrow edge is “transition condition”
- 0/1: bit before the backslash represents input, after the backlash is the output.
- it represents that at what condition we need to transit from which state to which state..

Sequential logic

Building Blocks

Sequential Logic

- There is memory
  - Outputs depend on prior state as well as the current inputs
  - State can be stored and used later
- We rely on clock signals
  - Clock signals tell us when things should happen
  - We should only write to state when the clock is set a certain way

SR Latch

- Constructed from two NOR gates
  - You can either Set the latch (make it remember 1), or Reset it (make it remember 0)
- Two inputs: S and R
- Two outputs: Q and NOT Q
  - Q is what it remembers, NOT Q is the opposite
- Both S and R cannot be 1 at the same time, or sadness occurs

D Latch

- Constructed from some additional logic and an SR Latch
- Two inputs: C and D
- You can have the latch remember D as long as C is true
- Two outputs: Q and NOT Q
  - Q is what the latch remembers, NOT Q is the inverse
- Ensures that S and R inputs to the SR Latch aren’t both true

D Flip Flop

- Constructed from some additional logic and two D latches
- Same inputs and outputs as D latches
- But, the output is only stored on a chosen clock edge

Finite State Machines

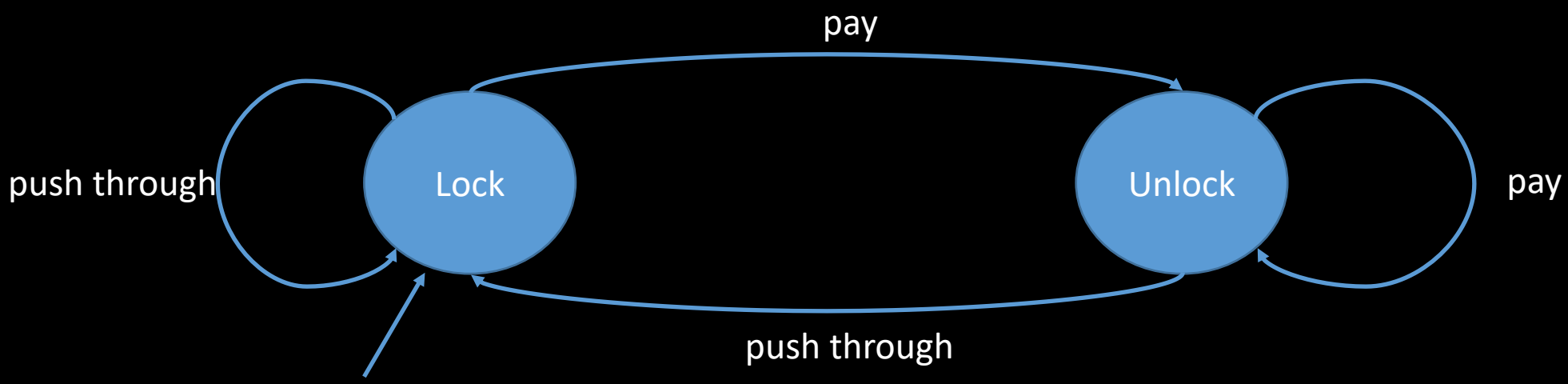
Finite State Machines

- There are a number of states, inputs, and outputs
- To the beat of the clock, we read in inputs and go to new states, and set the outputs
- Both the output and the next state are defined by the current state and the inputs
- You can think of it as following a flow chart “when I’m on this step, and this is true, I go here”

An FSM Example

- The NYC Subway Turnstile
- There is a lock controlled by the FSM
- If the user didn’t pay yet then the lock is active and the user can’t push through
- If the user pays the lock unlocks until they push through
- Draw an FSM for this
- Write out a truth table
- Create the circuit

An FSM Example



current state	input	next state
(lock / unlock)	(pay / push through)	