Finite State Machines (FSMs)
Finite State Machines (FSMs)

• Can model behavior of any sequential circuit
• Useful representation for designing sequential circuits
• As with all sequential circuits: output depends on present and past inputs
  – effect of past inputs represented by the current state
• **Moore**: output depends only on present state, delayed by a clock cycle
• **Mealy**: output depends on present state and input, output is immediate
Finite State Machines (FSMs)

- Allows us to design/model complex systems by viewing a system as having a set of possible states it can be in
  - The machine can only be in one state at a time
  - There are rules dictating how the machine moves between states
- Very common in digital logic
  - Often used to design “control logic”
    - ASIC and FPGA labs will both be using FSMs like this
    - So common that many EDA tools (including Vivado) have special optimization passes specifically for FSMs
- Can also be used in software, particularly in Real-Time Systems & Mechatronics
  - EECS 149/249A uses FSMs extensively as well as other formal methods classes
FSM Design Process Review

Review of Design Steps:
1. Specify circuit function (English)
2. Draw state transition diagram
3. Write down symbolic state transition table
4. Write down encoded state transition table
5. Derive logic equations
6. Derive circuit diagram
   - Register to hold state
   - Combinational Logic for Next State and Outputs
Step 1: Specify Circuit Function

Given a description of the state machine, there are several things you need to suss out and clearly define:

- Inputs
- Outputs
- States (this is the non-trivial part part)
Step 2: Draw State Transition Diagram

- Circuit is in one of two “states”.
- Transition on each cycle with each new input, over exactly one arc (edge).
- Output depends on which state the circuit is in.
- Moore or Mealy? (on the homework and exams we will specify)
Step 3: Symbolic State Transition Table

The table needs to have:
- Present State
- Inputs
- Outputs
- Next State

The values of the states, inputs, outputs, and next states can be physical values (ex. Input: added a quarter to the vending machine, Present State: traffic light is green)
Step 4: Encode State Transition Table

The table needs to have:
- Present State
- Inputs
- Outputs
- Next State

Now prescribe a binary code to these physical values (0 or 1 if something is or isn’t, N-bits for the states if there are multiple states)
Step 5: Derive Logic Equations

Step 4 is a truth table!

Use your Boolean Algebra tools to derive logic equations that describe the encoded state transition table.
Step 6: Derive Circuit Diagram

This should be the easiest step, having done all 5.
- Need a register to hold states
- Logic equations clearly define which logic gates are needed for the next state and outputs
Finite State Machines Optimization

• State Reduction (why waste time use lot state when few state do trick)
  – Row Matching
  – Implication Chart
• State Assignment (which state gets what code)
  – Minimum bit transition heuristic
  – Extreme example: One hot encoding

All of these are procedural heuristic methods, just follow the algorithms outlined in lecture.
Questions?

HW3 or Lecture Topics