Welcome to The Hardware Lab!

Fall 2016

Lab 4: Finite State Machine

Prof. Chun-Yi Lee

Department of Computer Science
National Sting Hua University

Verilog Lab 4

- **Five basic questions** (**50%**)
  - Demonstration on **11/17/2016 (Thu)**, **in class**
- **Lab 4 report** (**30%**)
  - **Circuit** or **block diagram, and FSM state transition diagram** of your design, **and explain how it works**
  - Answer the questions listed in the handout (if any)
  - Contribution of each team member
  - What you have learned from Lab 4
- **Two optional questions** (**20%**)
  - Challenge it if you have extra time
  - Include in your Lab 4 report if you can make it
  - **Describe your design and how it works**

Verilog Lab 4 Rules

- You can use **ANY** modeling techniques
- Demonstrate your work by **waveforms (Q1~Q5, OQ2) and FPGA (OQ1)**
- Please follow the **module names** and **I/O ports** of the provided templates
  - Submit **only one testbench file** and **one design file**
  - **Avoid commenting out** modules in your submitted Verilog files
  - **Please keep in mind that we will deduct points if the module names and I/O port mismatch with the template**
- If not specifically mentioned, we assume the following SPEC
  - **CLK** is **positive edge triggered**
  - Reset the Flip-Flops when **RESET == 1'b0**

Verilog Lab 4 Submission

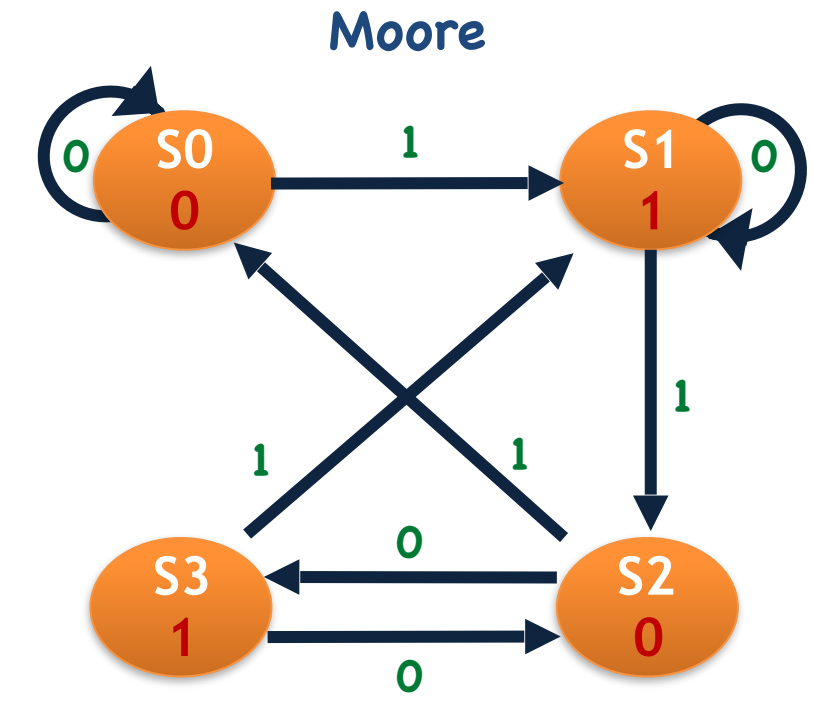
- **Demonstration**, **code and report** submission due date & time:
  5:30pm, 11/17/2016 (Thu)
  This is a HARD deadline

- Please submit your report and Verilog codes to **ILMS**
  - Format: **Lab4_Team1_Codes.v, Lab4_Team1_Tb.v, Lab4_Team1_Report.pdf**
  - **We will deduct points if the submitted files mismatch the requirements**
  - **Please don't uncomment the modules in the template file**

- We will test your codes by our own testbench

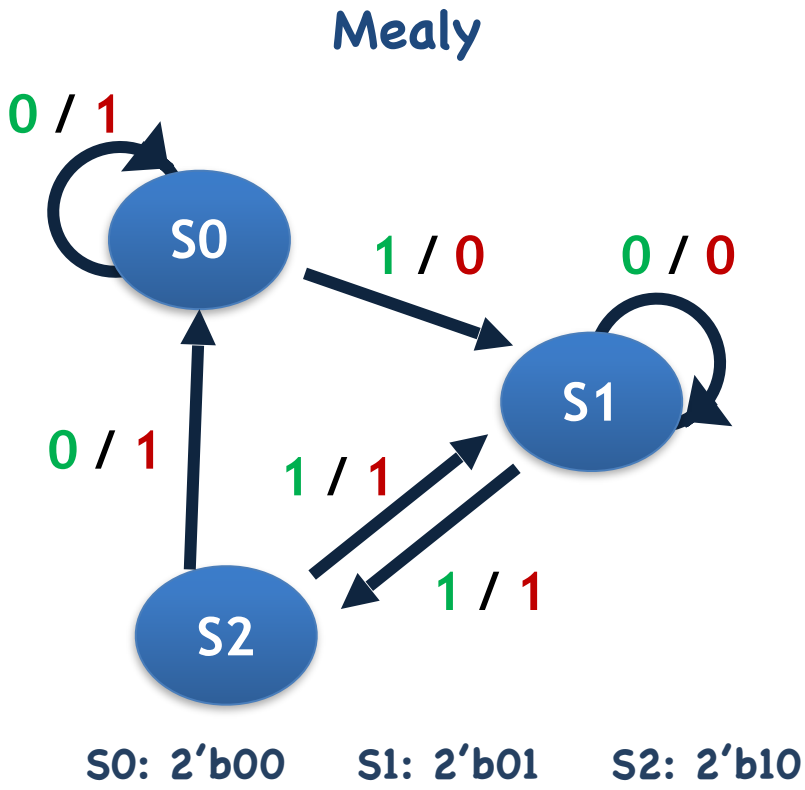
Question 1

- Please implement the following FSM
  - **Green** represents input, while **red** represents output
  - Output your **current state** as well
  - When RESET == 1'b0, State = S0



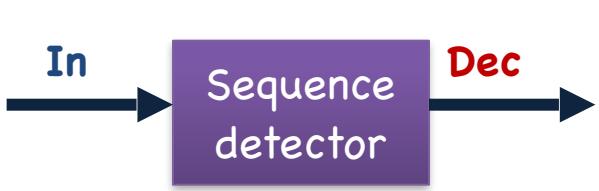
Question 2

- Please implement the following FSM
  - **Green** represents input, while **red** represents output
  - Output your **current state** as well
  - When RESET == 1'b0, State = S0

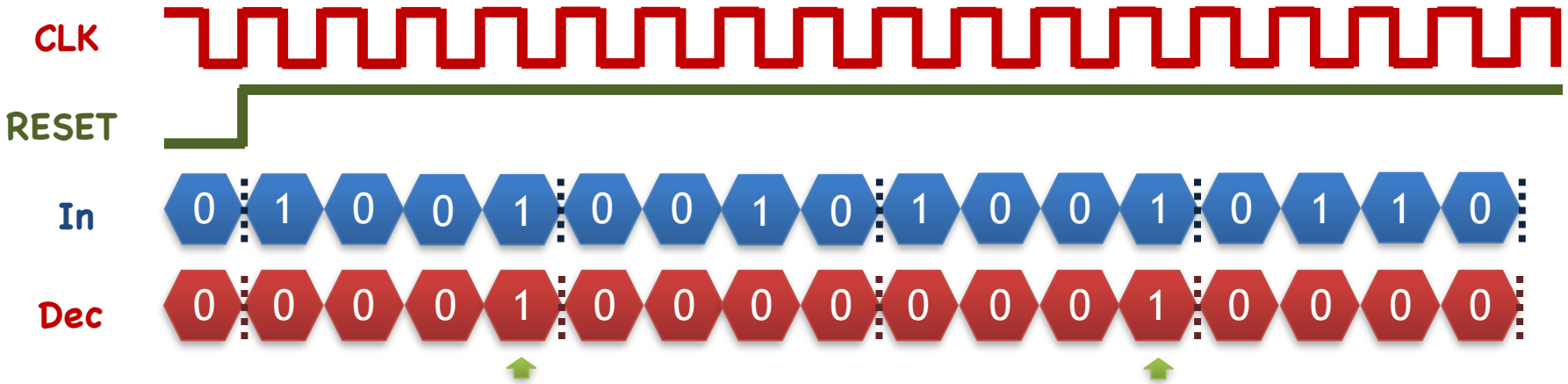


Question 3

- **Mealy machine sequence detector**
  - 1-bit input **In** and 1-bit output **Dec**
  - When the four bit sequence is either **1001**, **Dec** is set to **1**
  - Redetect the sequence **every four bits**
  - Please draw your state diagram in your report

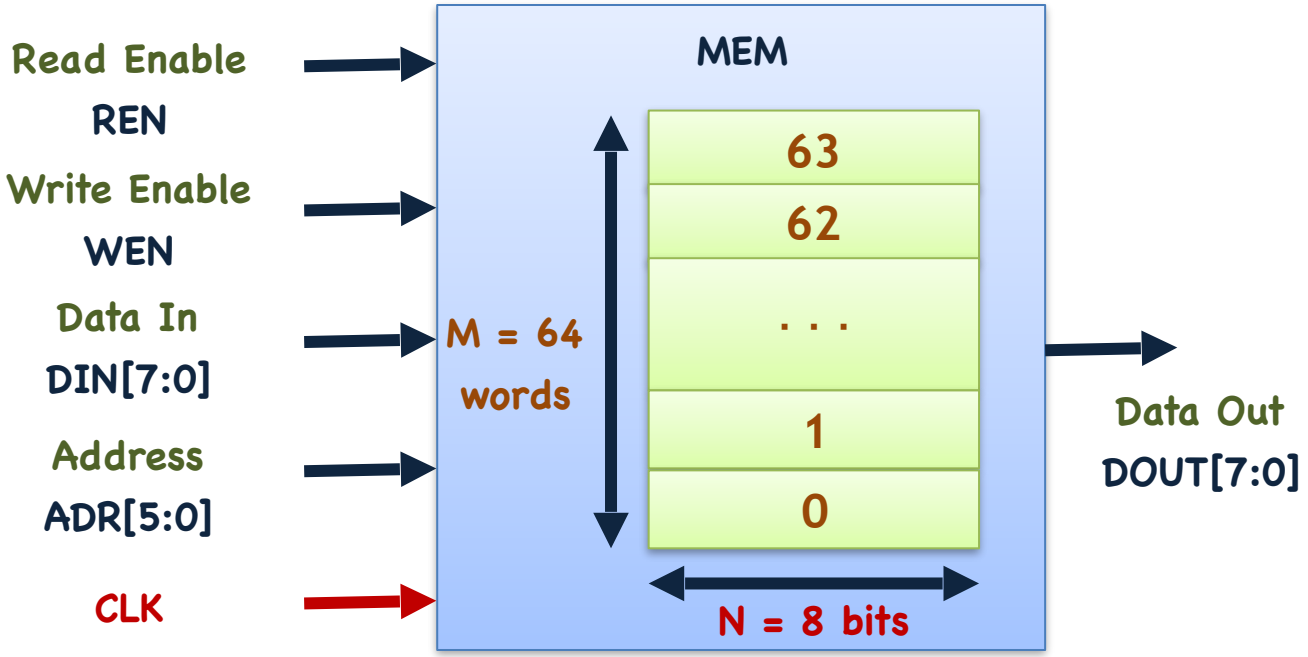


Complete the state diagram in your report



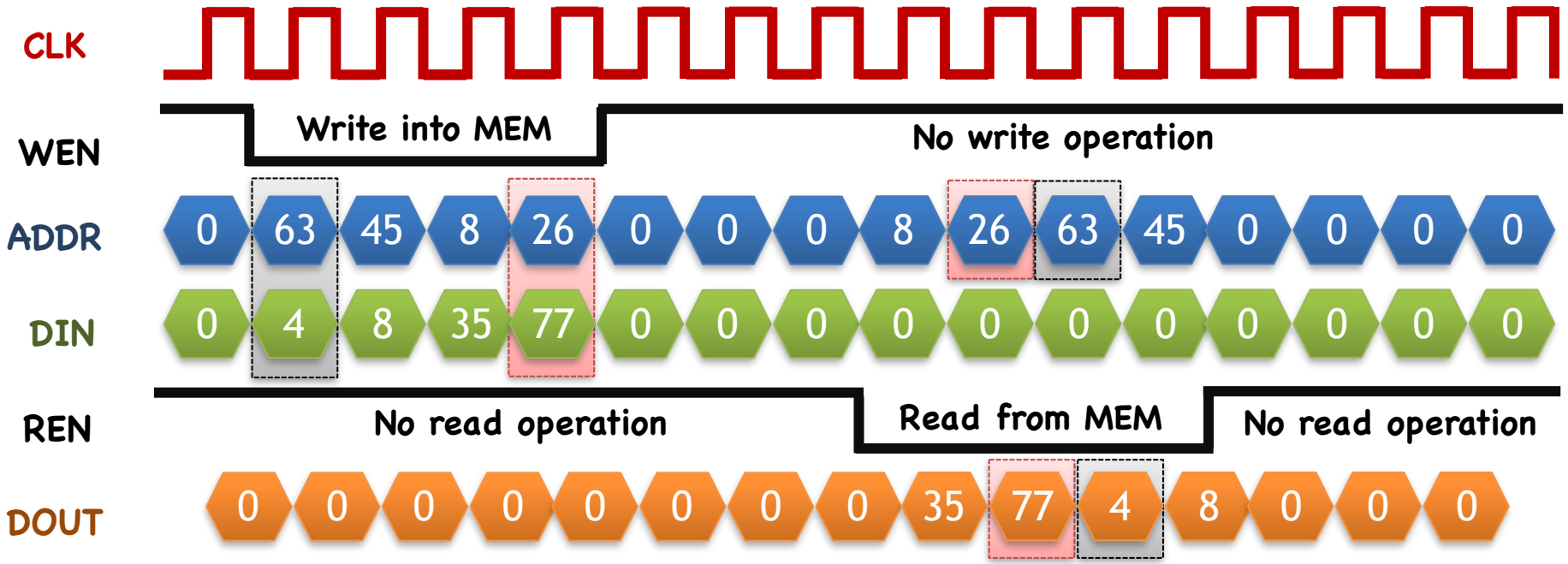
Question 4

- Assume that we have a **64** x **8** memory array **MEM**
  - **M** = **64**, **N** = **8**
  - **Inputs**: CLK, REN, WEN, ADDR[5:0], DIN[7:0]
  - **Outputs**: DOUT[7:0]



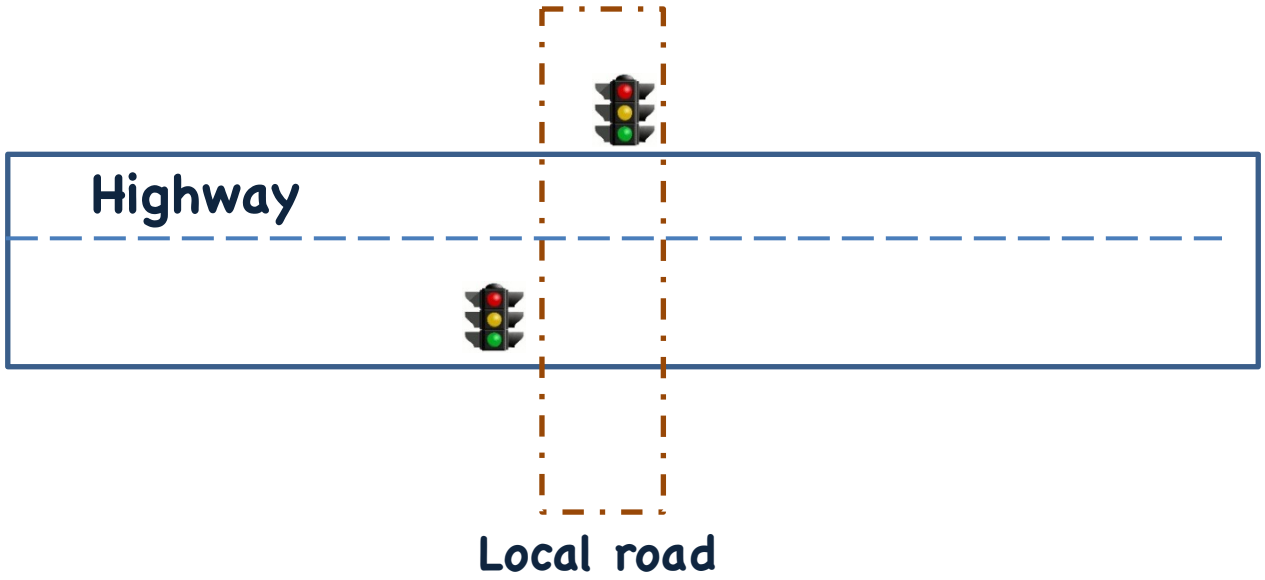
Question 4 Specification

- Specification
  - When WEN == 1'b0, write DIN to MEM[ADDR]
  - When REN == 1'b0, output MEM[ADDR] to DOUT; otherwise DOUT = 8'd0
  - REN and WEN won't be zero at the same time. **If both are 0, do only the read operation**
  - **MEM** does not need to be reset



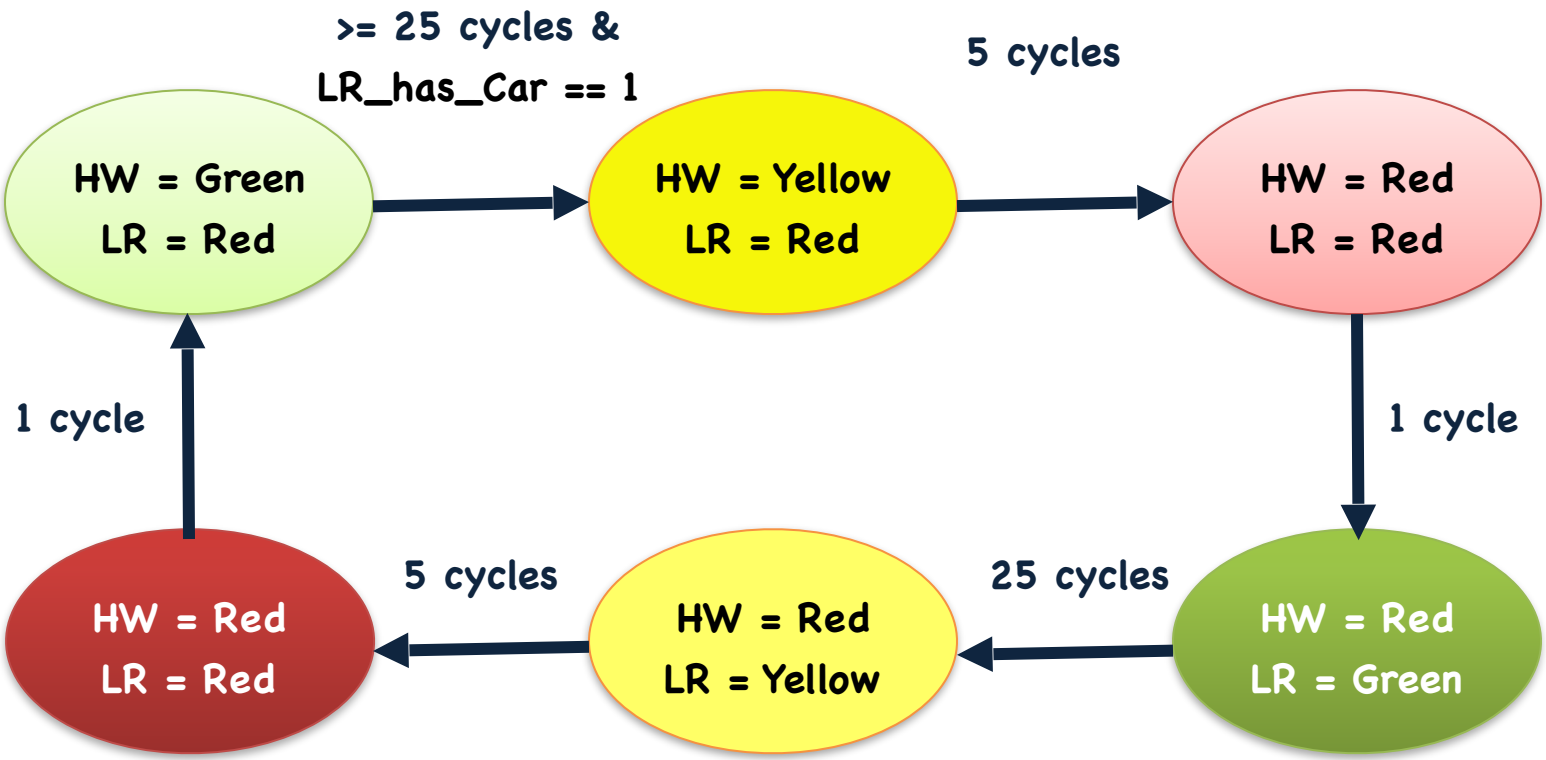
Question 5

- **Traffic light controller** for highway (**HW**) and local road (**LR**) intersection
- **HW** has higher priority and should be green as long as possible
- **LR** has a sensor to detect cars on it. When a car is sensed, LR turns green shortly
- Green light is **at least 25** clock cycles and yellow light is **5** clock cycles
- **Input**: **CLK**, **RESET**, LR_has_Car; **Output**: HW_light[2:0], LR_light[2:0]
  - HW_light & LR_light: bits [2:0] represent **Red**, **Yellow**, and **Green**, respectively



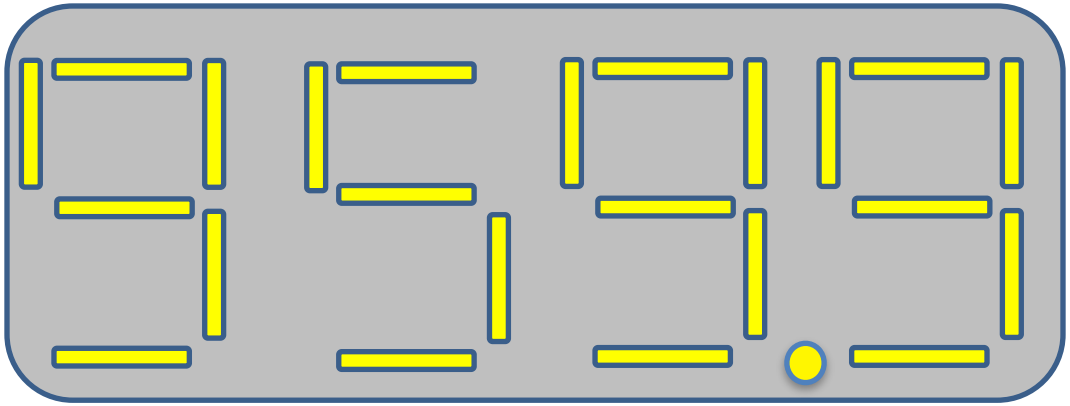
Question 5 Finite State Machine

- **Traffic light controller** Finite State Machine
- **Please complete the FSM in your report** (some arrows are removed intentionally)



Optional Question 1

- Please design a stopwatch and display it on your 7-segment display
- The four digits represent:
  - Digit[3]: Minutes
  - Digit[2:1]: Seconds
  - Digit[0]: 0.1 Seconds
  - Dot: Separates the third and fourth digits
- Use a push-bottom (**UP button, T18**) with debounce to start counting up
- Use another push-button (**RIGHT button, T17**) with debounce to reset
- Three FSM states: **RESET**, **WAIT**, **COUNT**
  - **please draw a state transition diagram in your report**



Optional Question 2

- Design a **4-bit Ping-Pong Counter** with **MAX** and **MIN**
  - **Input:** **CLK, RESET, Enable, FLIP, MAX[3:0], MIN[3:0]**
  - **Out:** **0,1,2,…,7,8,9,8,7,…,2,1,0,1,2,…**
  - **Direction:** **0,0,0,…,0,0,0,1,1,…,1,1,1,0,0,…**
  - In the above example, MAX is 9 and MIN is 0

Optional Question 2 SPEC

- **4-bit Ping-Pong Counter** SPEC
  - When **RESET == 1'b0**, the counter resets its value to **MIN**
  - When **Enable == 1'b1**, the counter begins its operation. Otherwise, the counter holds its current value

- **MAX** and **MIN**
  - **MAX** and **MIN** values are the maximum and minimum values for the counter
  - **MAX** >= **MIN**. Otherwise, counter holds its current value
  - When counter > **MAX** or counter < **MIN**, counter holds its current value

- **FLIP**
  - When **FLIP** == 1'b1, counter flips its direction
  - Flip is **only one cycle** in length

Thank you for your attention!

*The first Starbucks at Seattle, Washington, USA
This picture is taken by Chun-Yi Lee himself, who is also a fan of photography