Welcome to The Hardware Design & Lab!

Fall 2023

Lab 5: Keyboard and Audio Modules

Prof. Chun-Yi Lee

Department of Computer Science

National Tsing Hua University

Agenda

- Lab 5 Outline
- Lab 5 Basic Questions
- Lab 5 Advanced Questions

Lab 5 Outline

- Basic questions (1%)
  - Group assignments
  - Due on 11/16/2023 (Thu). Demonstration on your FPGA board (In class)
  - Only demonstration is necessary. Nothing to submit.

- Advanced questions (6%)
  - Individual and Group assignments
  - eeclass submission due on 11/23/2023 (Thu). 23:59:59.
  - Demonstration on your FPGA board (In class)
  - Assignment submission (Submit to eeclass)
    - Source codes and testbenches
    - Lab report in PDF

Lab 5 Rules

- Please note that grading will be based on NCVerilog
- You can use **ANY** modeling techniques

- If not specifically mentioned, we assume the following SPEC
  - **clk** is **positive edge triggered**
  - Synchronously reset the Flip-Flops when **rst_n == 1'b0, if there exists one rst_n signal in the specification**

Lab 5 Submission Requirements

- Source codes and testbenches
  - Please follow the templates EXACTLY
  - We will test your codes by TAs' testbenches
- Lab 5 report
  - Please submit your report in a single PDF file
  - Please draw the **block diagrams** and **state transition diagrams** of your designs
  - Please explain your designs in detail
  - Please list the contributions of each team member clearly
  - Please explain how you test your design
  - What you have learned from Lab 5

Agenda

- Lab 5 Outline
- Lab 5 Basic Questions
- Lab 5 Advanced Questions

Basic Questions

- Group assignment
- FPGA demonstration (due on 11/16/2023. In class.)
  - Keyboard sample code
  - Audio sample code 1 & 2

Basic FPGA Demonstration 1

- **Keyboard sample code**
  - Please implement the keyboard sample codes released on eeclass

- **Audio sample codes**
  - Please implement the audio sample codes 1 & 2 released on eeclass

Agenda

- Lab 5 Outline
- Lab 5 Basic Questions
- Lab 5 Advanced Questions

Advanced Questions

- Individual and Group assignments

- Verilog questions (Individual assignments)
  - Source codes and the report due on 11/23/2023. 23:59:59.
  - Optional: Sliding window sequence detector
  - Optional: Traffic light controller
  - Necessary: Greatest common divisor
  - **Bonus**: Booth multiplier

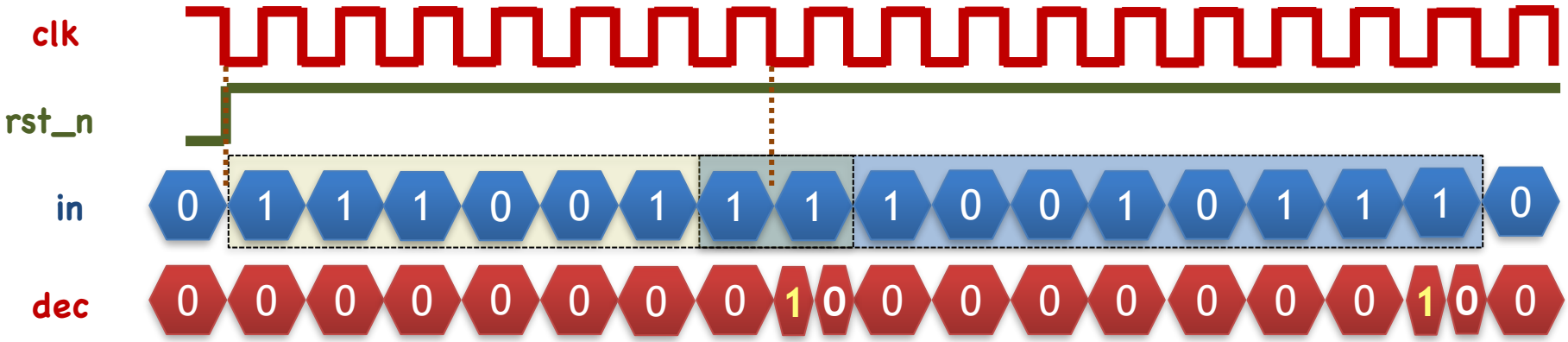
- FPGA demonstration (group assignments) on 11/23/2023, in class
  - Source codes and the report due on 11/23/2023. 23:59:59.
  - Necessary: Mixed keyboard and audio modules together
  - Necessary: Vending machine

Verilog Advanced Question 1

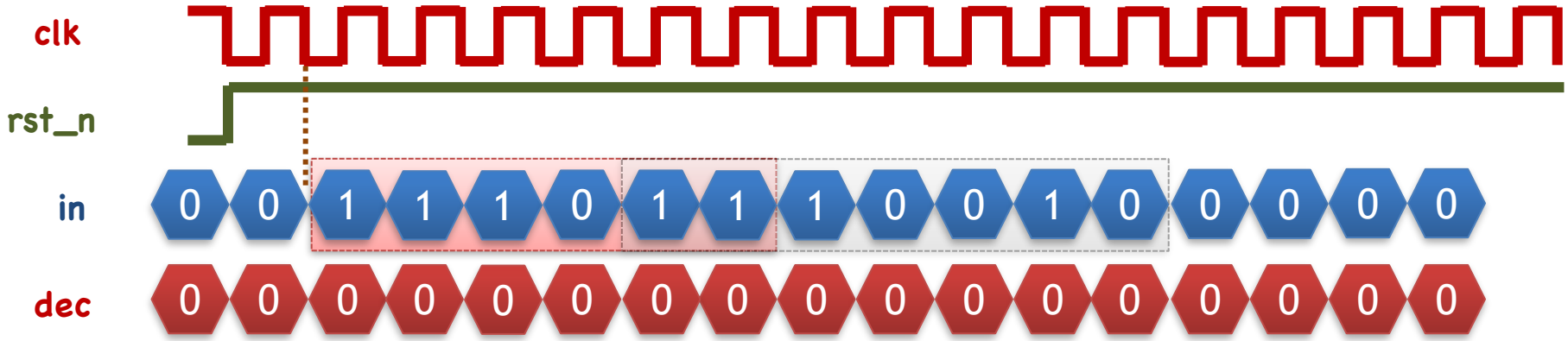
- **Sliding Window** sequence detector (**mealy machine**)
  - Detect the sequence **1110(01)+11** (in regular expression)
  - The pattern **01** in the middle have to appear at least once and can be repeated.
  - For example, **11100111** is a match, **1110010111** is also a match. On the other hand, **111011** is a mismatch.
- **Continuous detection**
  - Detect the sequences whenever they occur, and set **dec** to **1'b1**
  - Please draw a state transition diagram in your report
  - A sample waveform is provided in the next page
- **I/O port definition**
  - Input: **clk**, **rst_n, in**
  - Output: **dec**

Verilog Advanced Question 1 (Con't)

A match case

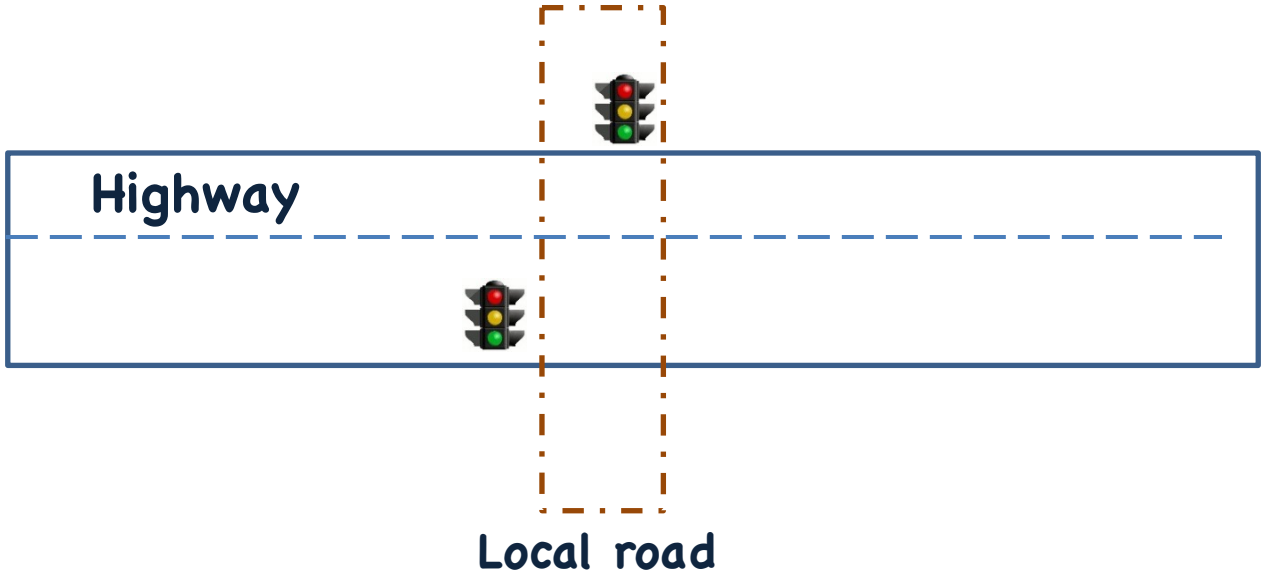


A mismatch case



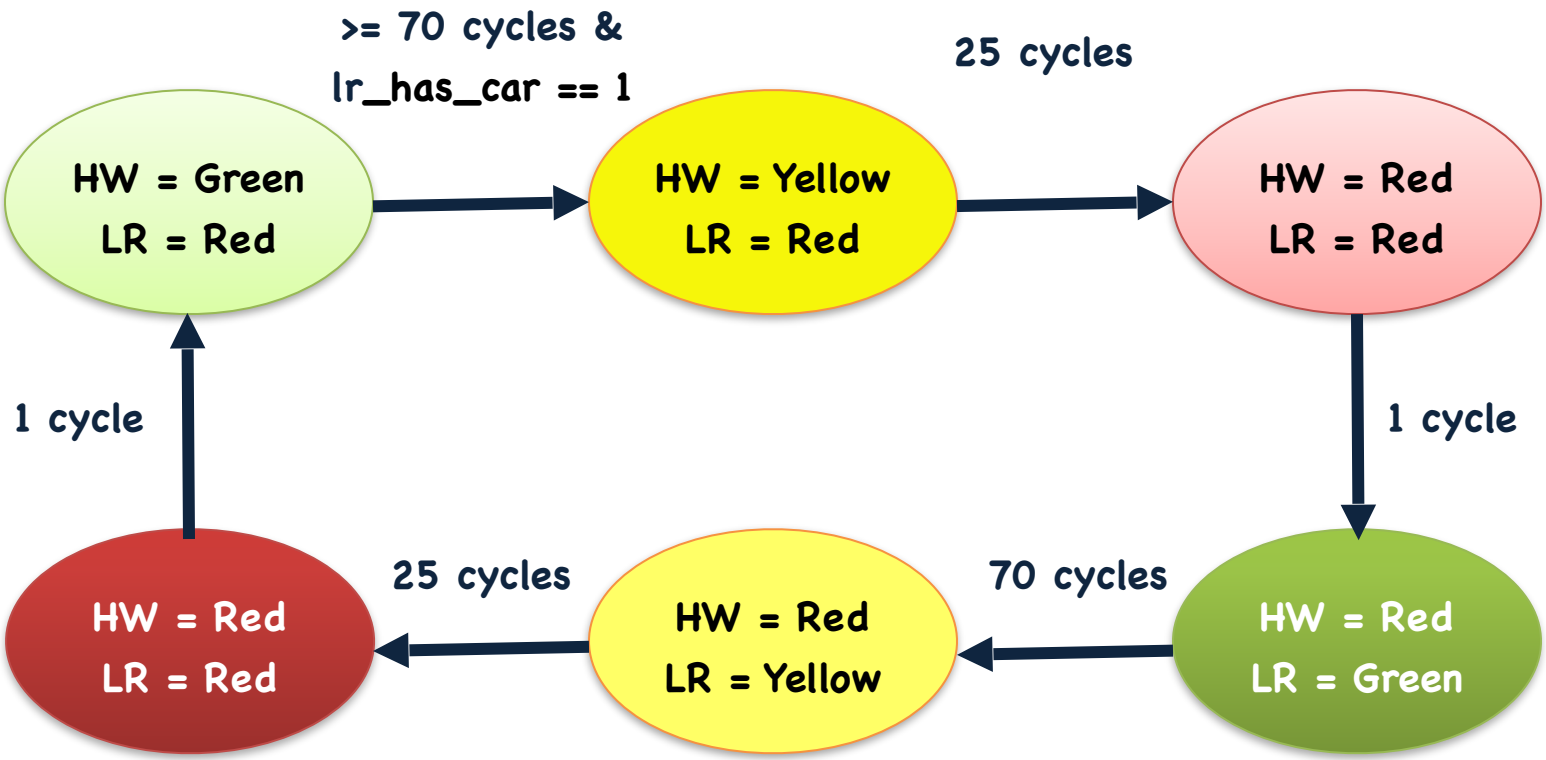
Verilog Advanced Question 2

- **Traffic light controller** for a 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 70** clock cycles and yellow light is **25** clock cycles
- **Input**: **clk**, **rst_n**, **lr_has_car**; **Output**: **hw_light[2:0]**, **lr_light[2:0]**
  - **hw_light** & **lr_light**: bits [2:0] represent **Green**, **Yellow**, and **Red**, respectively

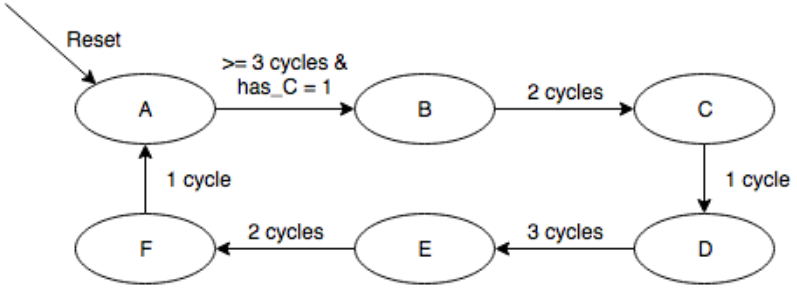


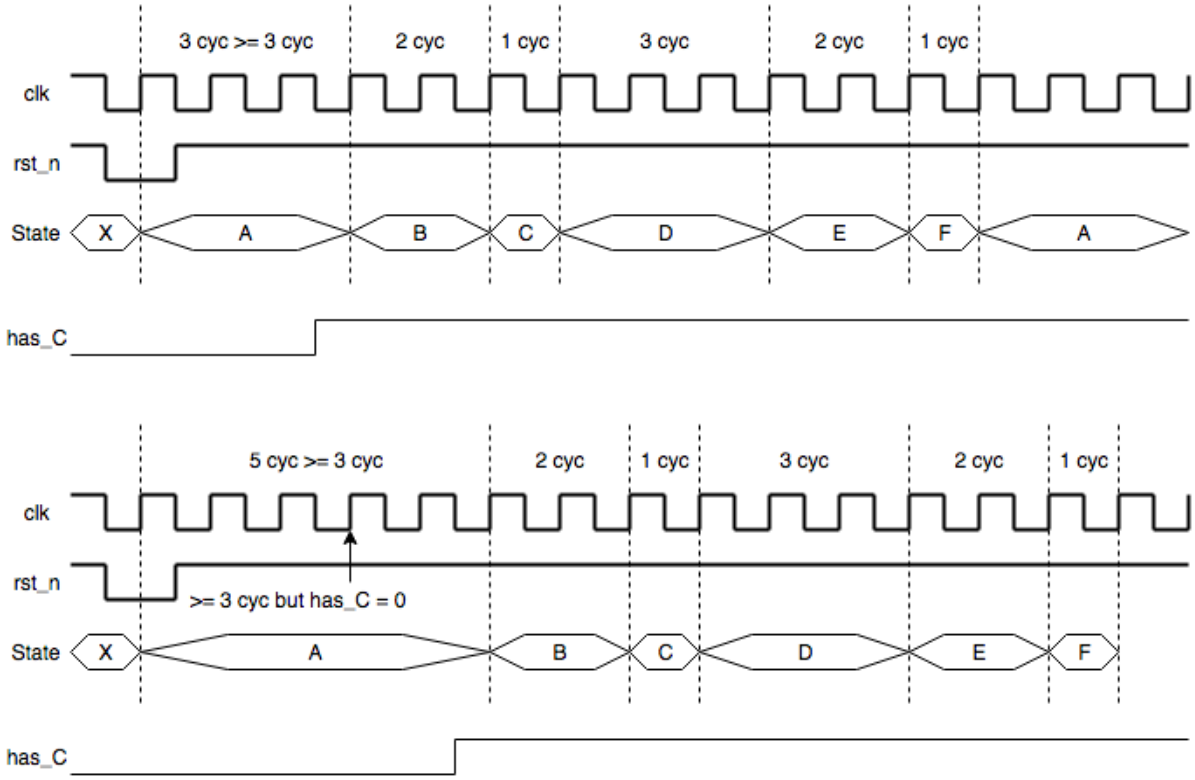
Verilog Advanced Question 2 (Con't)

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



Verilog Advanced Question 2 (Con't)

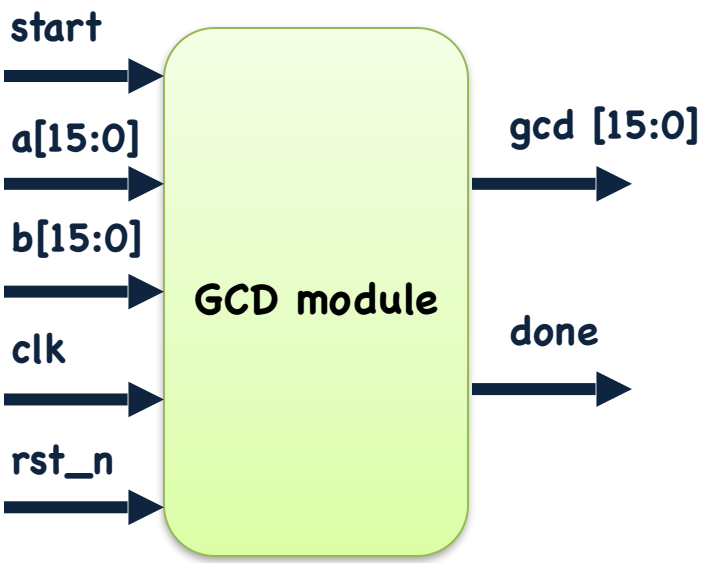




- **A Traffic light controller "example"** timing diagram is illustrated on the left
- **Please make sure that your state transitions follows the timing digram correctly**

Verilog Advanced Question 3

- **Greatest common divisor**
- Calculate the greatest common divisor of two numbers **a** and **b**
- Top level block diagram and pseudo code are as follows
  - You **shall not** use **loop statements and modulus (%)** in your Verilog codes

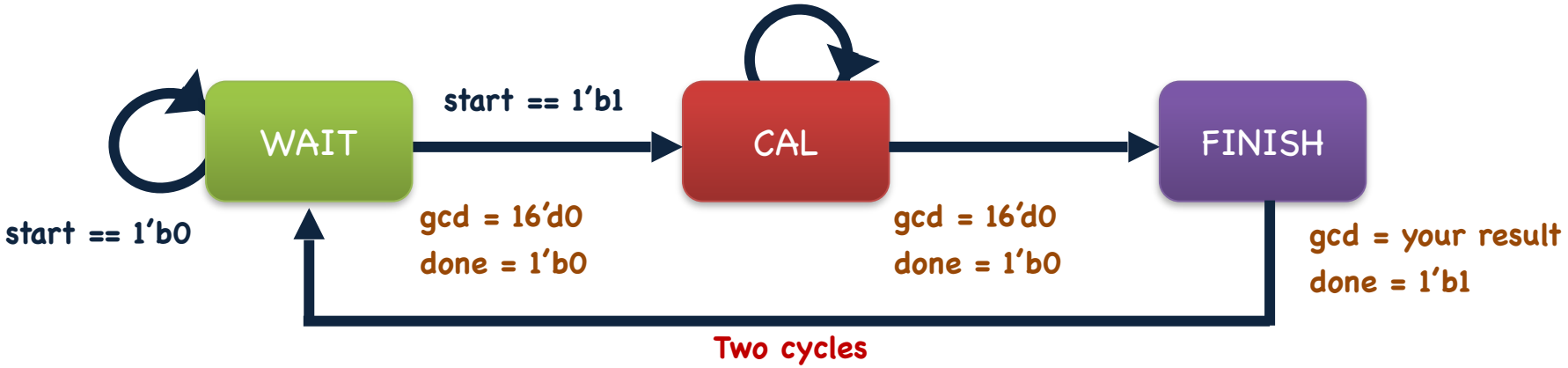


GCD pseudo code

```
Function gcd (a, b)
begin
    if (a == 0)
        return b;
    while (b != 0)
    // Do the following operation once per clock cycle
    begin
        if (a > b)
            a = a - b;
        else
            b = b - a;
    end
    return a;
end
```

Verilog Advanced Question 3 (Cont'd)

- Three states are used: **WAIT**, **CAL**, and **FINISH**
- **WAIT state**
  - Wait for **start == 1'b1** (**one cycle**) to begin the operation (**and fetch the inputs**)
  - The values of **a** and **b** may change during operation. Be sure to fetch and buffer them when the state changes from **WAIT** to **CAL**
  - When **rst_n == 1'b0**, reset the module to the **WAIT state**
- **CAL state**
  - Perform the **subtraction operations once per cycle**
- **FINISH state**
  - Output the **gcd** result for **two cycles**
  - **done** == **1'b1** for **two cycles**



Bonus: Verilog Advanced Question 4

- **Booth Multiplier**
- Please design a **booth multiplier** to compute the product of two signed input (the product is also signed).
- Your design should follow the mechanism of the booth multiplication. Otherwise, no bonus credit will be granted
- For more information about the **booth multiplier**, please refer to the following references:
  - https://tinyurl.com/kvsyspuj
  - https://tinyurl.com/4bzyayf8
- **I/O port definition**
  - Input: **clk**, **rst_n**, **start**, **a[3:0] (signed)**, **b[3:0] (signed)**
  - Output: **p[7:0] (signed)**

Bonus: Verilog Advanced Question 4

- Three states are used: **WAIT**, **CAL**, and **FINISH**

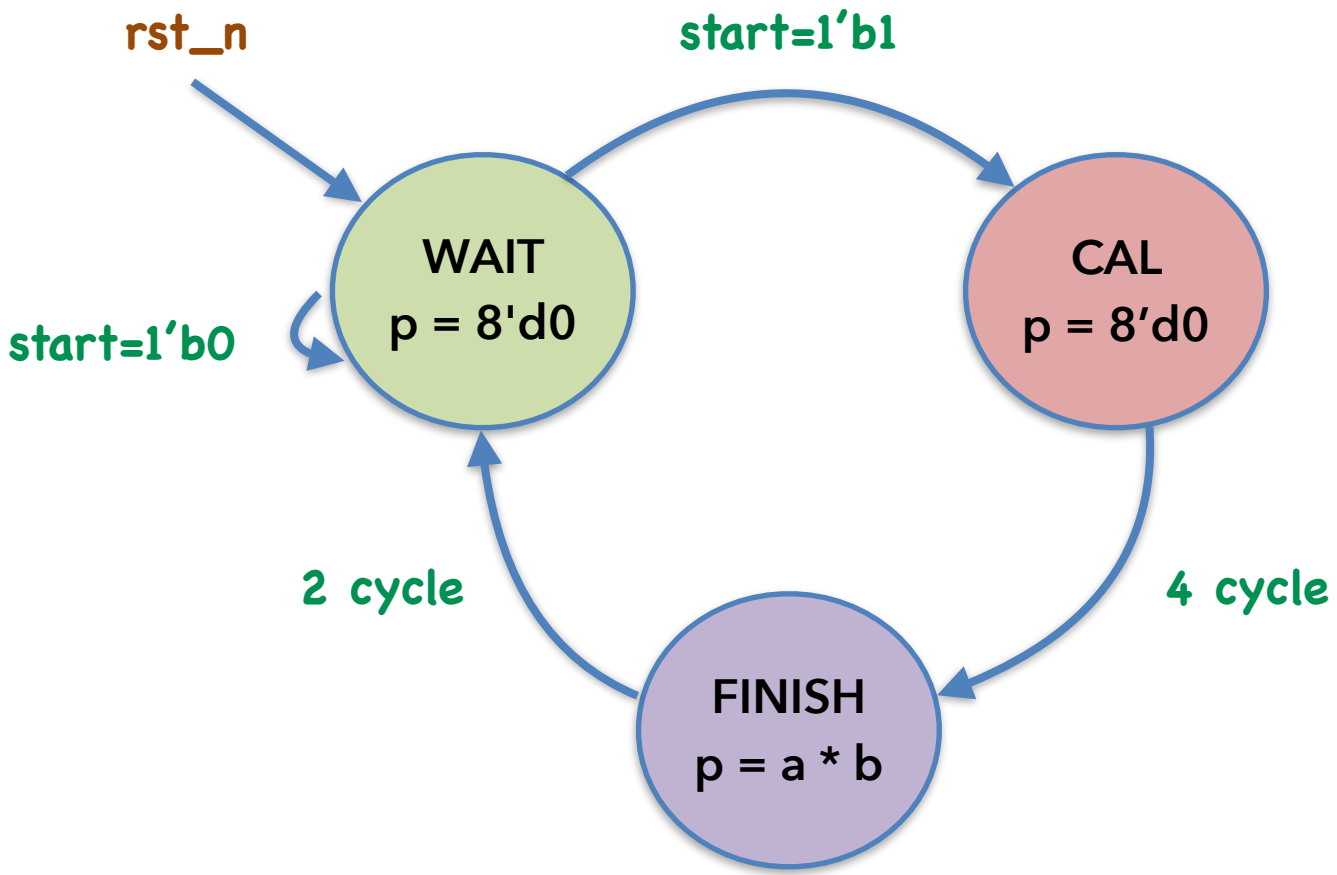
- **WAIT state**
  - Wait for **start == 1'b1** (**one cycle**) to begin the operation (**and fetch the inputs**)
  - The value of **a** and **b** may change during operation. Be sure to fetch and buffer them when the state changes from **WAIT** to **CAL**
  - When **rst_n == 1'b0**, reset the module to the **WAIT state**

- **CAL state**
  - Perform the **booth multiplication operations once per cycle**
  - Transition to FINSH state after 4 cycles of calculation.

- **FINISH state**
  - Output the result **p** for **two cycles**

Bonus: Verilog Advanced Question 4

- The state transition diagram of the booth multiplier



Advanced Questions

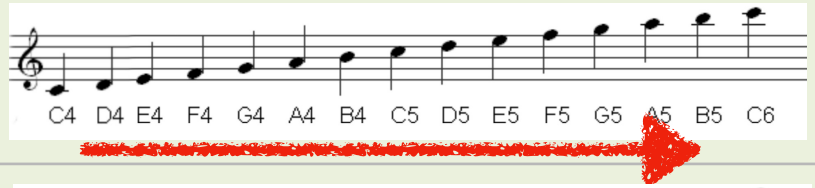
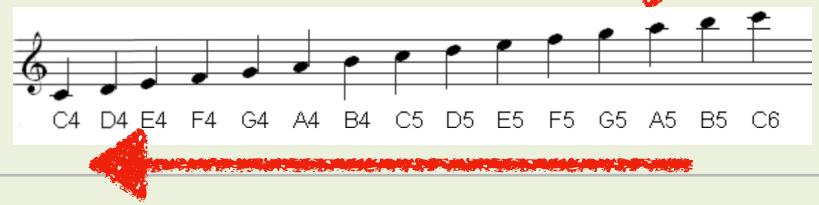
- Individual and Group assignments

- Verilog questions (Individual assignments)
  - Source codes and the report due on 11/23/2023. 23:59:59.
  - Optional: Sliding window sequence detector
  - Optional: Traffic light controller
  - Necessary: Greatest common divisor
  - **Bonus**: Booth multiplier

- FPGA demonstration (group assignments) on 11/23/2023, in class
  - Source codes and the report due on 11/23/2023. 23:59:59.
  - Necessary: Mixed keyboard and audio modules together
  - Necessary: Vending machine

FPGA Demonstration 1

- Use the numbers ("**w**" and "**s**") on the keyboard to control the scale to ascend or descend, ranging from **C4** to high **C8**.
- Change a note every **1 second**. If "**r**" is pressed, change to a note every **0.5 second**. If "**r**" is pressed again, go back to **1 second** per note.
- When it reaches **C4** or **C8**, stay on the note until the direction changes (keyboard pressed).

Button	Direction Reset: Set back to C4 and ascend (1sec/note) (Use *Enter* as rst_n)
w	
s	
r	0.5 sec per note or 1 sec per note

FPGA Demonstration 2

- Four options available: **Coffee**, **Coke**, **Oolong**, and **Water**
  - Prices are: **Coffee** (**NT$ 80**), **Coke** (**NT$ 30**), **Oolong** (**NT$ 25**), **Water** (**NT$ 20**)
- The **rightmost three 7-segment displays** show the money inserted into the machine
  - When **rst_n** == **1'b1**, please display "0"
  - The maximum value is **NT$ 100**
  - **Do not prepend '0' when you only have one or two digits to display**
- Use **five buttons** to implement your design:
  - **Left: NT$ 5**
  - **Center: NT$ 10**
  - **Right: NT$ 50**
  - **Top: rst_n**
  - **Bottom: Cancel**

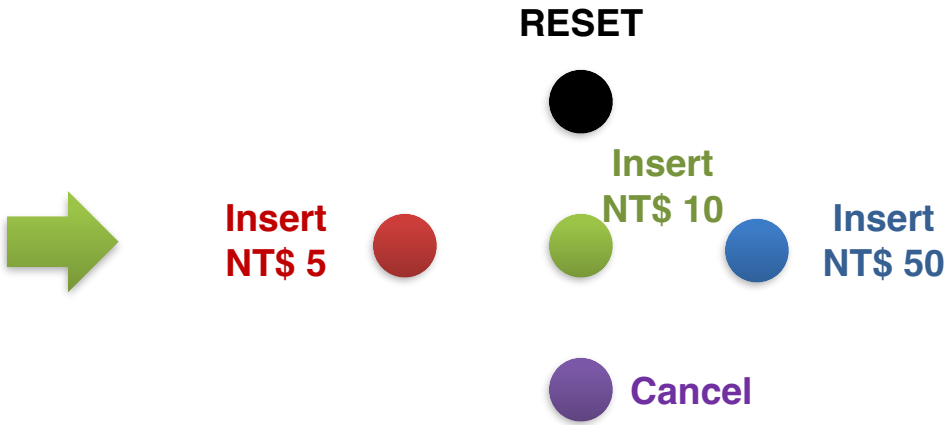
FPGA Demonstration 2

- Use **four LEDs** to indicate which drinks you can buy
  - **LED[3:0]** corresponds to Coffee, Coke, Oolong, and Water, respectively
- Use the **keyboard** to select which drinks you buy
  - **'a', 's', 'd', 'f'** corresponds to **Coffee**, **Coke**, **Oolong**, and **Water**, respectively
  - Assume that the machine allows you to buy **ONLY ONE DRINK** at a time
- Use the **rightmost three 7-segment display** to **show the rest of the money** after buying a drink
  - E.g., if you inserted **NT$ 40** and bought a can of **Oolong** (**NT$ 25**), the 7-segment display will show **NT$ 15**

FPGA Demonstration 2

- Remember to add debounce and one-pulse circuits to your buttons
- Decrement the **7-segment display** by **NT$ 5** every second to mimic the process of returning changes
  - Return the changes until it becomes zero
- If the buyer does not want to buy a drink, he/she can use a **Cancel Button** to cancel it
  - The inserted money will be returned the same way (**NT$ 5** per second)

The layout of the buttons used in this question



Thank you for your attention!

*Lake Helen at Lassen Volcanic National Park, Shasta County, California, USA
This picture is taken by Chun-Yi Lee himself, who is also a fan of photography