Lab 2: Vending Machine Lab

EE312 Computer Architecture

Professor: John Kim

TA (Lab 2): Yujeong Choi

(EMAIL: kaist.ee312.ta@gmail.com)

1. Overview

Lab 2 teaches you more advanced concepts and skills of the Verilog language. You have learned the basics of Verilog in *Lab 1*, and now it is time to improve it. In *Lab 2*, you are required to implement a vending machine. After you finish *Lab 2*, you will have a good understanding of register-transfer level, synchronous circuit, and finite state machine which you have learned from the lectures. Have fun!

2. Backgrounds

Register-Transfer Level (RTL)

Register-Transfer Level (RTL) is a design abstraction which models a synchronous digital circuit in terms of the flow of digital signals (i.e., data) between hardware registers, and the logical operations performed on those signals. The RTL abstraction is used in Hardware Description Languages (HDLs) like Verilog and VHDL to create high-level representations of a circuit, from which lower-level representations and ultimately actual wiring can be derived. [1]

Finite State Machine (FSM)

A Finite State Machine (FSM) is a machine that can be in exactly one of a finite number of states at any given time. The FSM can transit from one state to another state in response to external inputs; the *transition* between states is predefined in the FSM. An FSM is defined with a list of states, conditions for each transition, and initial state. There are two types of FSMs, the Moore machine and the Mealy machine. A Moore machine is an FSM whose outputs are determined only by its current state, as shown in Fig. 1. This is in contrast to a Mealy machine whose outputs are determined by both its current state and inputs, as shown in Fig. 2. You can implement either Moore or Mealy machine.

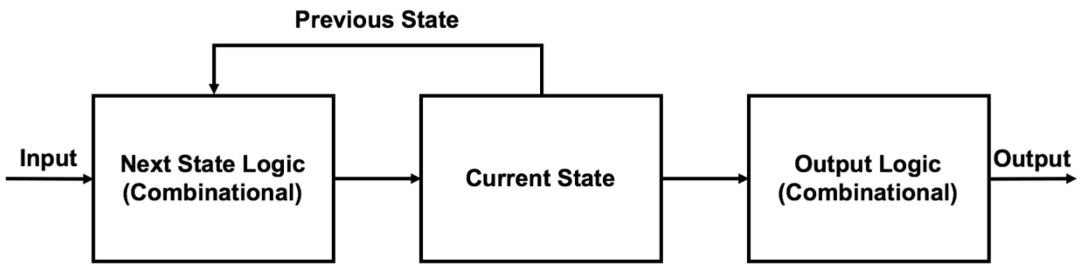

Previous State
Input
Next State Logic (Combinational)
Current State
Output Logic (Combinational)
Output

Figure 1. The Moore Machine

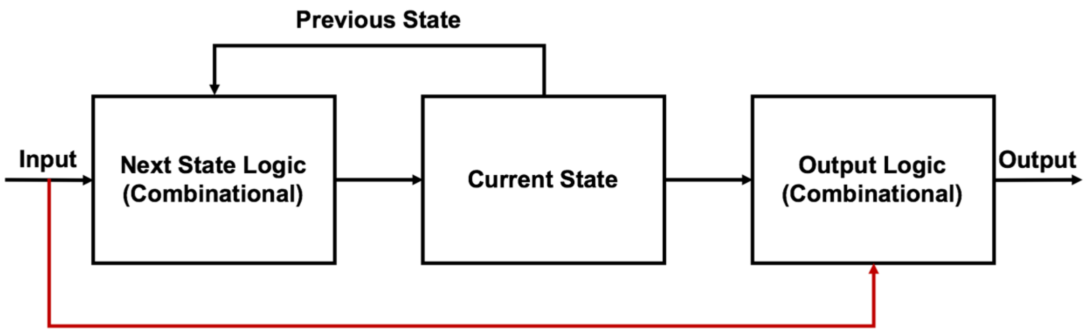

Previous State
Input
Next State Logic (Combinational)
Current State
Output Logic (Combinational)
Output

Figure 2. The Mealy Machine

An FSM in an RTL design consists of three parts:

1. A combinational logic that calculates the next state
2. A sequential logic that saves the current state
3. A combinational logic that calculates the output

In *Lab 2*, you are required to implement all three parts of a vending machine by yourself.

3. Files

You are given three files:

1. *vending_machine.v*: The file where you implement the vending machine module

2. *vending_machine_def.v*: The file where constants needed to implement the vending machine are defined

3. vending_machine_TB.v: The file which you can test and grade your module with

4. Vending Machine Lab

In *Lab 2*, you are required to implement a vending machine, which is an example of an FSM. There might be several corner cases when implementing the vending machine, so you have to make sure that your implementation of the vending machine functions properly for all use cases.

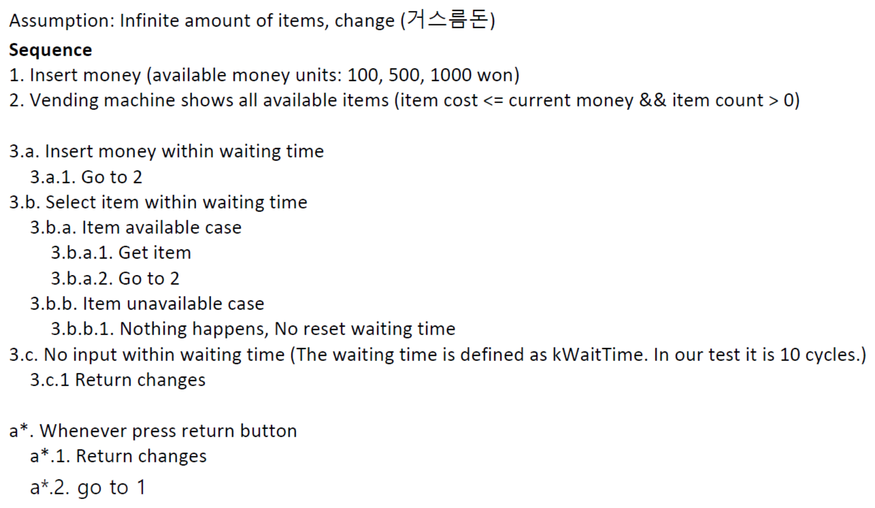
You are given a template of a vending machine in *vending_machine.v*. The template defines the interface of the vending machine, as shown in Table 1.

- Some of the variables in the given template might not be used, but must be connected to the testbench module.

Input		
Signal	**Description**	**# of bits**
i_input_coin	Insert a coin	1 for each coin
i_select_item	Select an item	1 for each item
i_return_trigger	Return change	1
clk	Clock	1
reset_n	Reset	1
Output		
o_output_item	Get an item	1 for each item
o_available_item	Available items	1 for each item
o_return_coin	Return coins	1 for each item

Table 1. Interface of the vending machine

Please follow the below rule, and figure out what the given variables in the template correspond to.



```
Assumption: Infinite amount of items, change (거스름돈)
Sequence
1. Insert money (available money units: 100, 500, 1000 won)
2. Vending machine shows all available items (item cost <= current money && item count > 0)

3.a. Insert money within waiting time
   3.a.1. Go to 2
3.b. Select item within waiting time
   3.b.a. Item available case
      3.b.a.1. Get item
      3.b.a.2. Go to 2
   3.b.b. Item unavailable case
      3.b.b.1. Nothing happens, No reset waiting time
3.c. No input within waiting time (The waiting time is defined as kWaitTime. In our test it is 10 cycles.)
   3.c.1 Return changes

a*. Whenever press return button
   a*.1. Return changes
   a*.2. go to 1
```

Figure 3. Vending machine use-cases

5. Grading

The TAs will grade your lab assignments with *vending_machine_TB.v* that is already given to you. Your score is relevant to how many tests you pass in the testbench file (you must not modify the testbench file). 80% of the lab 2 score is determined by how well you implemented the vending machine, 10% is determined by your report, and the remaining 10% will be given if you submit your honor code.

Also, there are some guidelines that can affect your grades.

- **Do not** use "blocking operator" in the sequential logic part (e.g., always @(posedge clk)). Use non-blocking operator for sequential logic and blocking operator for combinational logic. If you do not abide by this rule, there will be a deduction.

- **Do not** use delay operator (e.g., #50, #100). If you use "delay operator" in your code (except the given testbench code), you will automatically get zero points for that lab assignment.

- **Do not** use “wait()” function for your design. We will not give any points for a design that is using “wait()” function.

6. Lab Report Guidance

You are required to submit a lab report for every lab assignment. You can write your report either in Korean or in English.

Your lab report **MUST** include the following sections:

1. Introduction
    a. *Introduction* includes what you think you are required to accomplish from the lab assignment and a brief description of your design and implementation.
2. Design
    a. *Design* includes a high-level description of your design of the Verilog modules (e.g., the relationship between the modules).
    b. Figures are very helpful for the TAs to understand your Verilog code.
    c. The TAs recommend that you include figures because drawing the figures helps you how to *design* your modules.
3. Implementation
    a. *Implementation* includes a detailed description of your implementation of what you design.
    b. Just writing the overall structure and meaningful information is enough; you do not need to explain minor issues that you solve in detail.
    c. **Do not copy and paste your source code.**
4. Evaluation
    a. *Evaluation* includes how you evaluate your design and implementation and the simulation results.
    b. *Evaluation* must include how many tests you pass in vending_machine_TB.v
5. Discussion
    a. *Discussion* includes any problems that you experience when you follow through the lab assignment or any feedbacks for the TAs.
    b. Your feedbacks are very helpful for the TAs to further improve *EE312* course*!*
6. Conclusion
    a. *Conclusion* includes any concluding remarks of your work or what you have accomplished through the lab assignment.

7. Requirements

You **MUST** comply with the following rules:

- You should implement the lab assignment in **Verilog**.
- You should only implement the **TODO** parts of the given template.

- You should name your lab report as **lab2_YourName1_StudentID1_YourName2_StudentID2.pdf**.
- You should compress the honor pledge(s), lab report, simulation results, and source code, then name the compressed zip file as **Lab2_YourName1_StudentID1_YourName2_StudentID2.zip**, and submit the zip file on the KLMS.
- You should not change the interface of the vending machine defined in *vending_machine.v*.
- Your submission file should look like this:
    - Lab2_YourName1_StudentID1_YuourName2_StudentID2.zip
        - vending_machine.v
        - vending_machine_def.v
        - vending_machine_TB.v
        - lab2_yourname1_studentID1_yourname2_studentID2.pdf
        - honor_pledge_for_the_assignment.pdf

If your submission file does not contain the expected files or your code cannot be compiled or run with ModelSim, we cannot give you any points. Please double check before you submit.

Reference

1. Frank Vahid (2010). *Digital Design with RTL Design, Verilog and VHDL* (2nd ed.). Jorn Wiley and Sons. p. 247. ISBN 978-0-470-53108-2