NYCU DCS 2025

HW02

Design: Vending Machine

Data Preparation

1. Extract files from TA's directory:

   % tar xvf ~DCSTA01/HW02.tar

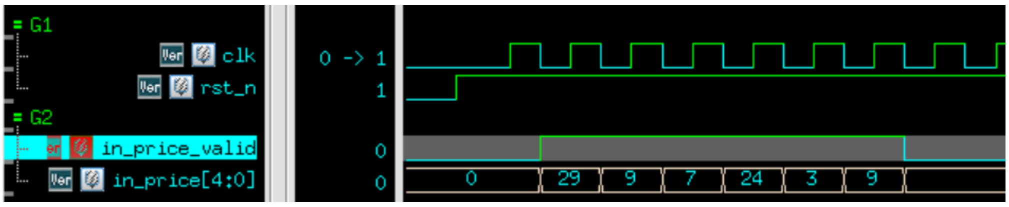
2. The extracted LAB directory contains:
   a. **00_TESTBED**
   b. **01_RTL**
   c. **02_SYN**
   d. **03_GATE**
   e. **09_SUBMIT**

Design Description

In this assignment, we are tasked with designing a vending machine and implementing functions for pricing, coin insertion, making change, and sales tracking. The vending machine will offer six products, namely item1 through item6. Additionally, note that only coins of the following denominations may be used in transactions: **$50, $20, $10, $5, and $ 1.**

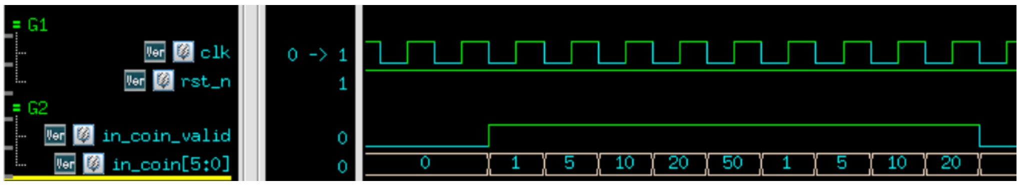
This vending machine has the following rules:

1. After rst_n, all output signals must be reset to 0.
2. When in_price_valid is 1, the pattern will input in_price sequentially **for 6 cycles** to set prices for items 1 to 6. (notice: not every pattern will reset the prices.)

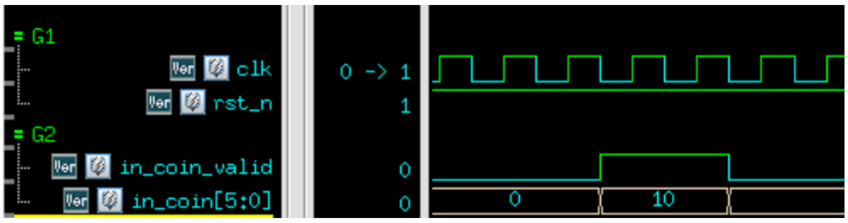


3. When in_coin_valid is 1, the pattern will input in_coin signal to insert coins. in_coin_valid **lasts for 1 to 9 cycles**, meaning at least 1 coin and up to 9 coins can be inserted.

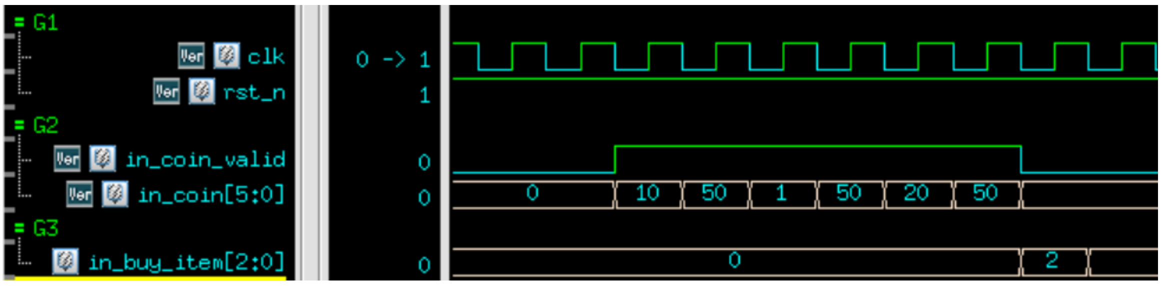
   - Insert coins 9 times:



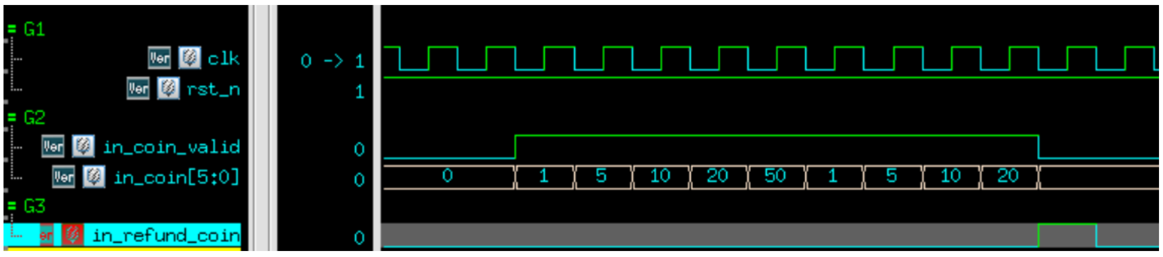
- Insert coins 2 time:



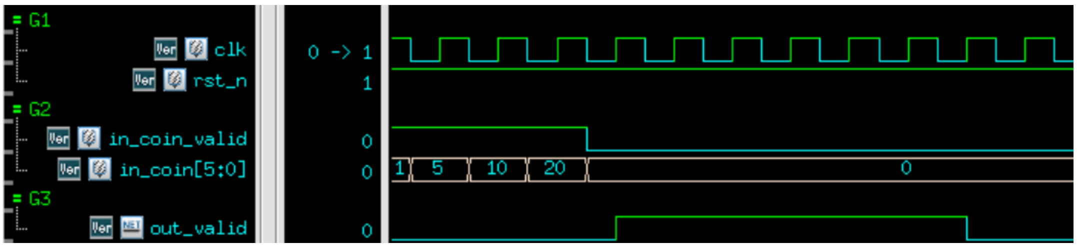
4. After in_coin_valid goes low, the pattern will **immediately** provide either a purchase signal(in_buy_item) or a refund signal(in_refund_coin).
   - Purchase(in_buy_item)



   - Refund(in_refund_coin)



5. When in_buy_item is pressed and the inserted coins are sufficient, the machine outputs the selected item number and returns the remaining coins.
6. When in_refund_coin is pressed, the machine outputs item number 0 and returns all inserted coins.
7. The out_valid signal must be pull high within 100 cycles after in_coin_valid pull down and remain high for 6 cycles.



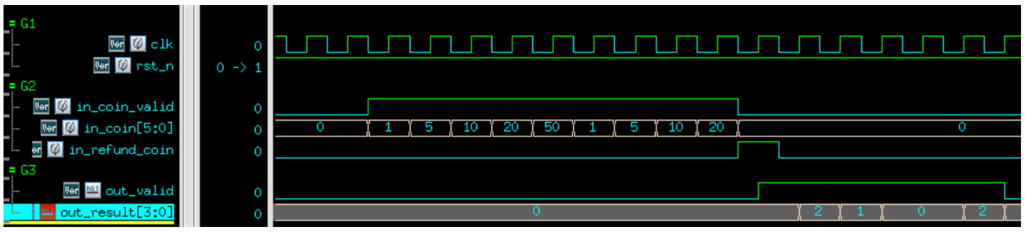
8. When **out_valid** is high, the pattern will check two output signals: **out_result** and **out_num**.
9. change rules(找錢規則):always prioritize using **larger** coin first in the following order: **$50 > $20 >$ 10 > $5 > $1**.
   Example: if the machine needs to give $90 as change, it must give 1*$50 and 2*$20. Giving 9×10 or any other combination is not allowed.

10. out_result will output 6 cycle : the first cycle should output the item number (**0-6,0 refer to appendix**), the second cycle From the second to the sixth cycle, the number of refunded coins for $50, $20, $10, $5, and $1 will be output in order.

(i) Refund(in_refund_coin):

Example: When $122 is inserted and the in_refund_coin signal is pull high, it means no item is purchased and $122 needs to be refunded.($50 *2, $20*1, $10*0, $5*0, $1*2).

Output: [0,2,1,0,0,2]

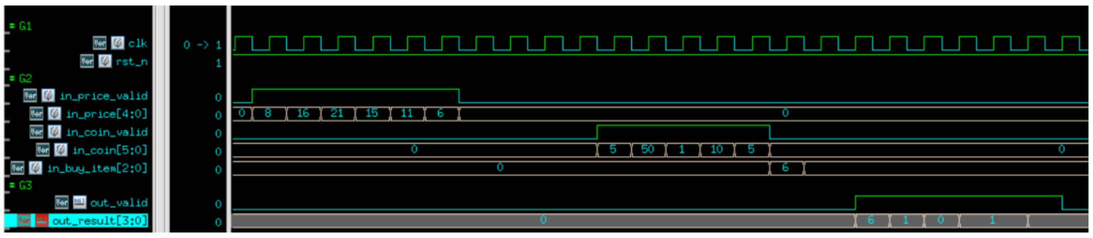


(ii) Purchase(in_buy_item)

Example: When $71 is inserted and buy item6($6)，machine should output purchased num(6) and refund $65.

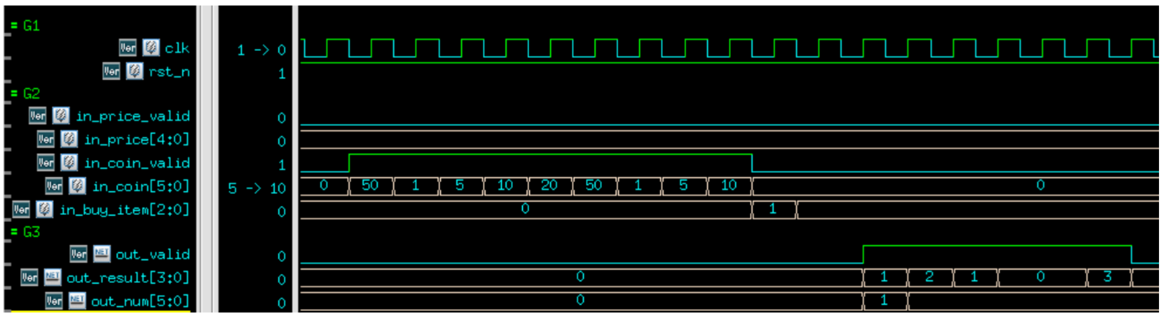
($50 *1, $20*0, $10*1, $5*1, $1*0)

Output[6,1,0,1,1,0]

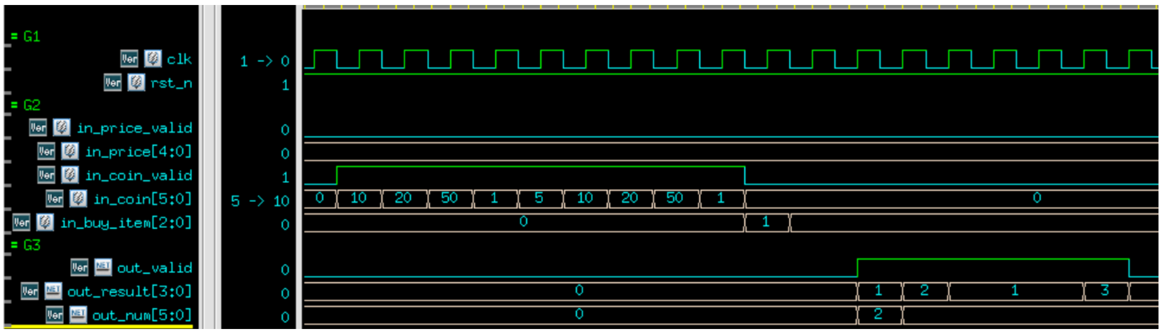


11. out_num will also output for 6 cycles: it outputs the number of items sold from item1 to item6 in order.This count will reset to 0 each time pricing is updated (每次重新訂價後累計數量重新計算).

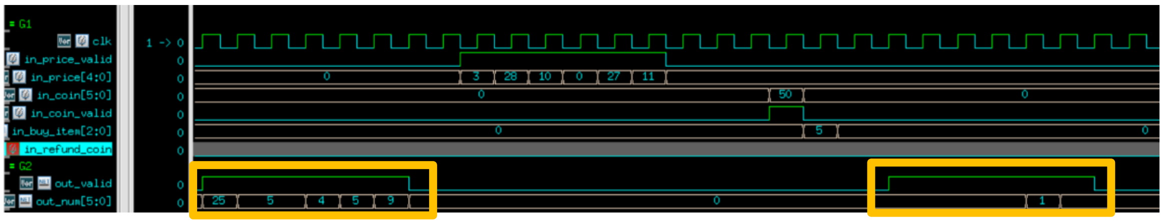
a. The first time for selling item1



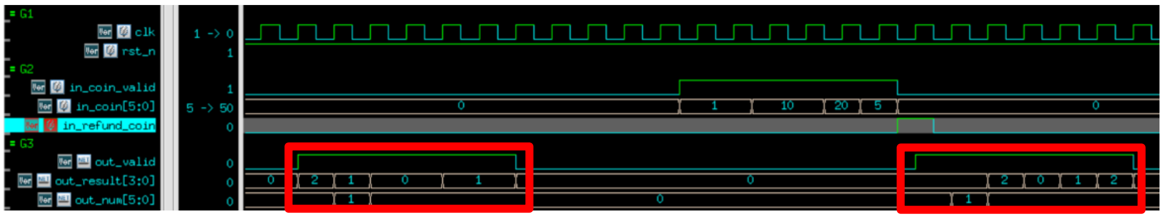
b. The second time for selling item1



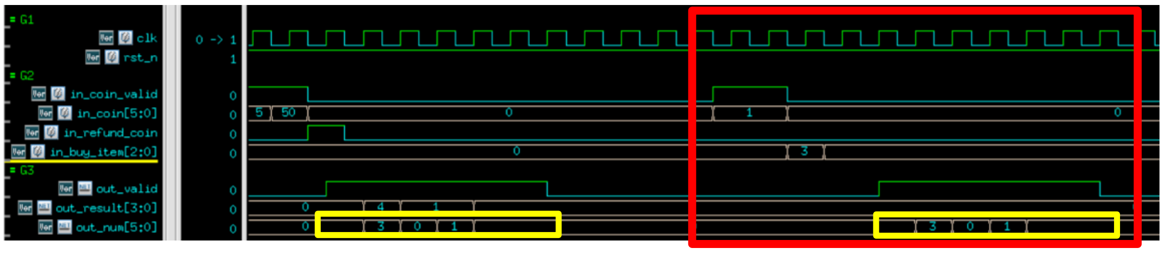
c. When in price_valid, out_num resets to 0 and restart to accumulate the sold number. (重新計算累計數量)



d. When in_refund_coin pull high, Since no product was sold this time, out_num outputs the same as the previous result: [0, 1, 0, 0, 0, 0] (as shown in the red box in the image).



12. Special case: When the inserted coins are less than the price of the selected item, the purchase will fail. The machine will **not refund the money** and will wait for the next coin insertion, accumulating the total amount.
Example: Insert $1 and try to buy item 3 ($11). The purchase fails.
out_result output[0,0,0,0,0,0].
Since no item was sold this time, out_num outputs the same as the previous result (as shown in the yellow box in the image).



Input

Input Signal	Bit Width	Description
clk	1	Clock signal
rst_n	1	Asynchronous active-low
in_price_valid	1	Be high when in_price signal is valid.
in_coin_valid	1	Be high when in_coin signal is valid.
in_price	5	Pricing signal total 6 cycles,setting the prices for items 1 to 6 in order.
in_coin	6	Coin insertion signal (random in 1-9 cycles)
in_refund_coin	1	Refund signal
in_buy_item	3	Purchase signal where 1-6 represent items 1 to 6.

Output

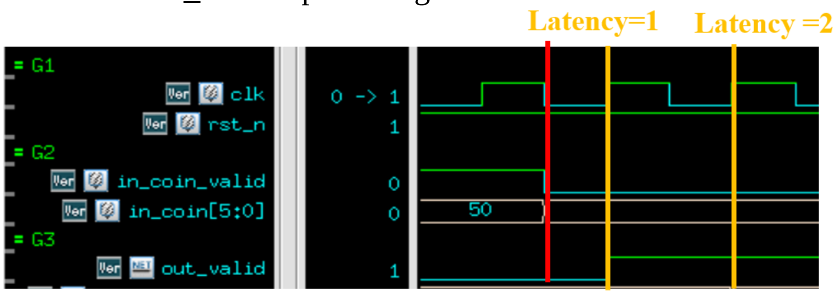
Output Signal	Bit Width	Description
out_valid	1	Be high when output signals are valid. Maintain with 6 cycles.
out_result	4	Sequentially output the item number(refer to appendix),the number of 50 coins, 20 coins, 10 coins, 5 coins, and 1 coins (notice:item number will be 0 if not buy anthing) Maintain with 6 cycles.
out_num	6	Sequentially output the cumulative sold quantity of item 1 to 6. Each time in_price_valid signal is pulled high, the cumulative count should be reset. (每次重新訂價後累計累積數量需重新計算). Maintain with 6 cycles.

Appendix:

Item number	Definition
0	Buy nothing
1	Buy item 1
2	Buy item 2
3	Buy item 3
4	Buy item 4
5	Buy item 5
6	Buy item 6

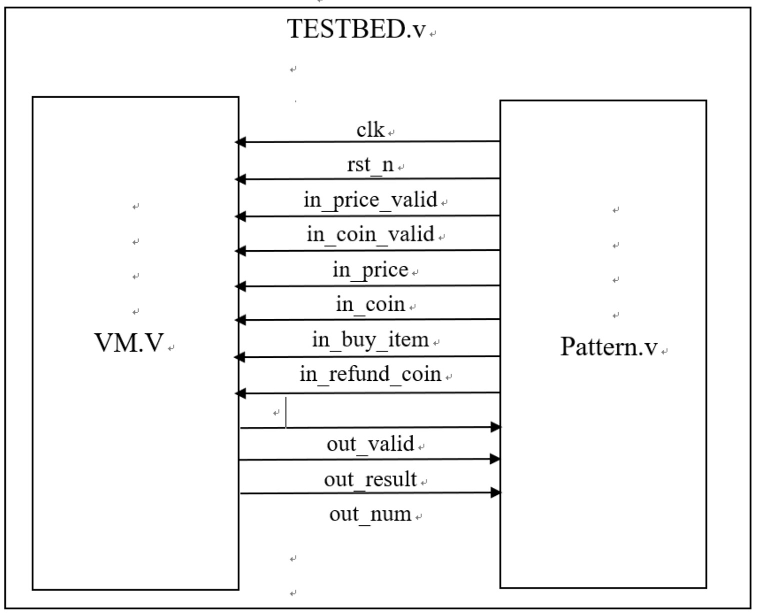
SPEC

1. Top module name: VM (File name: VM.v)
2. It is positive edge trigger architecture.
3. It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock starting), you may fail to reset signals.
4. The reset signal (reset) would be given only once at the beginning of simulation.output signals for valid should be reset after the reset signal is asserted.
5. Clock period is 5 ns,you can’t modify clock period in this homework 。
6. Input delay = 0.5 * clock period; Output delay = 0.5 * clock period
7. Out_valid singal should be pulled high in 100 cycle after in_coin_valid signal pull down
8. Out_valid signal should be pulled high for 6 cycle, no more and no less.
9. Definition of latency: Start counting after **in_coin_valid** is pulled down, and count until out_valid is pulled high



10. 02_SYN results cannot include any latches or error.
11. Slack should be non-negative(MET) at the end of the timing report.
12. 03_GATE results cannot include any time violation.

Block Diagram



Homework Upload

1. Please use the commands we provided to tar whole file under 09_SUBMIT/

   First demo deadline: **23:59:59 on 4/4(Fri)**

   Second demo deadline: **23:59:59 on 4/11(Fri)**

   A. Noticed that the time **is only based on** the demo result you submit with 09_SUBMIT. It is not allowed that the homework handed in late, even if the time just overs 1 second.

Grading Policy

1. Verilog Function Validity: 70%
   - ➢ The score **is only based on** the demo result you submit with 09_SUBMIT. Please ensure that you successfully upload the latest version.
   - ➢ This Homework do not have hidden test cases.
   - ➢ Please check whether there is any wire/reg/submodule being named as "error", "fail", "pass", "congratulations", "latch". All letters used in the formats of uppercase or lowercase is prohibited. If there is, you will **fail** this homework.
2. Performance: Total cell area * total latency **30%**
3. **1st demo: total score * 1, 2nd demo: total score * 0.7**

Command List

- ❖ Verilog RTL simulation (01_RTL/):
  - ◆ **./01_run_vcs_rtl**
- ❖ Synthesis (02_SYN/):
  - ◆ **./01_run_dc_shell**
  - ◆ **./08_check**
- ❖ Gate level simulation (03_GATE/):
  - ◆ **./01_run_vcs_gate**
- ❖ Submit your files (09_SUBMIT/):
  - ◆ **./00 tar 5.0 (do not change the cycle time in this homework)**
  - ◆ **./01_submit**
  - ◆ **./02_check**
- ❖ Waveform for debug:
  - ◆ **nWave &**
  - ◆ find *.fsdb
  - ◆ shift+L for reloading the *.fsdb file after you simulate your design again.
- ❖ You can key in **./09_clean_up** to clear all log files and dump files in each folder.