NCTU-EE DCS – 2017

Lab07 Exercise

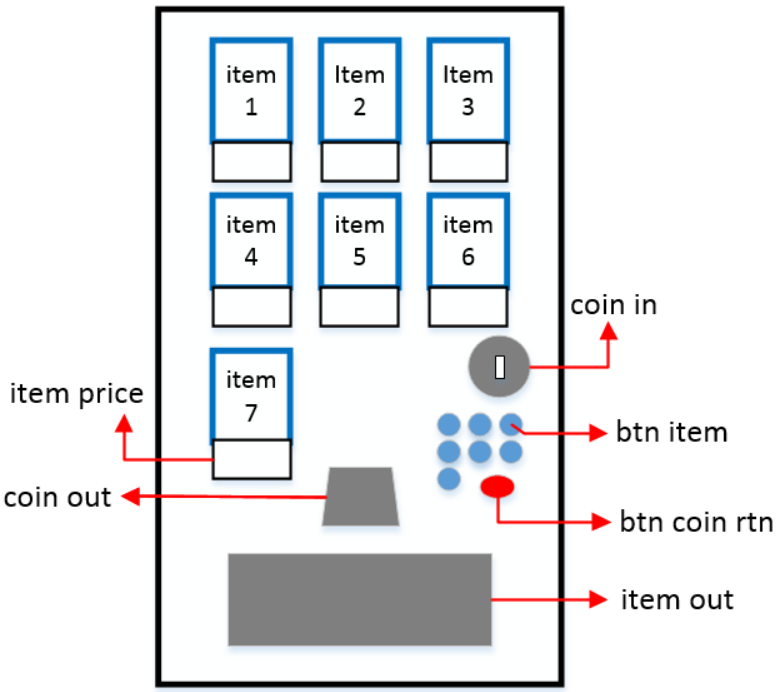
Design: Vending Machine

Data Preparation

1. Extract test data from TA's directory:
   % tar xvf ~dcsta02/LAB07.tar
2. The extracted LAB directory contains:
   a. Exercise/ :your design

Design Description and Example Wave

In this lab , you are asked to implement a vending machine. This vending machine contains seven difference items to sell. You can buy item by pressing the different button or you can press the return money button to get money back.



Ex1:
Price[item1, item2, item3, item4, item5, item6, item7]
= [15, 18, 22, 25, 18, 10, 22]
coin_in sequence is [6‘d5, 6’d10, 6’d50, 6’d50]
And you press the btn_coin_rtn.

Then the ouput signals will be:
btn_item = 3’d0
coin_out_50 = 2’d2
coin_out_10 = 2’d1
coin_out_5   = 1’d1
coin_out_1   = 2’d0

Ex2:
Price[item1, item2, item3, item4, item5, item6, item7]
= [5‘d15, 5d’18, 5’d22, 5’d25, 5’d18, 5’d10, 5’d22]
coin_in sequence is [6‘d50, 6’d1, 6’d5, 6’d50]
And you press the btn_item[2:0] = 3’d6

Then the ouput signals will be:
btn_item = 3’b6
coin_out_50 = 2’d1
coin_out_10 = 2’d4
coin_out_5   = 1’d1
coin_out_1   = 2’d1

Ex3:

Price[item1, item2, item3, item4, item5, item6, item7]

= [5‘d15, 5d’18, 5’d22, 5’d25, 5’d18, 5’d10, 5’d22]

coin_in sequence is [6‘d5, 6’d1, 6’d1, 6’d1]

And you press the btn_item[2:0] = 3’d6

Then the ouput signals will be:

btn_item = 3’b0

coin_out_50 = 2’d0

coin_out_10 = 2’d0

coin_out_5 = 1’d0

coin_out_1 = 2’d0

If the inserted coins are not enough to buy the item, **the coin can be used for the next input.**

The next coin_in sequence is [6‘d10, 6’d5, 6’d5, 6’d10]

And you press the btn_item[2:0] = 3’d6

Then the ouput signals will be:

btn_item = 3’d6

coin_out_50 = 2’d0

coin_out_10 = 2’d2

coin_out_5 = 1’d1

coin_out_1 = 2’d3

Inputs

1. All input signals will be **synchronized** at **negative edge** of the clock.

I/O	Signal name	Description
Input	**clk**	Clock
Input	**rst_n**	Asynchronous active-low reset. You should set all your outputs to 0 when rst_n is low
Input	**in_valid_price**	item_price[4:0] is valid when in_valid is high.
Input	**in_valid**	Input signals is valid when in_valid is high.
Input	**item_price[4:0]**	item_price[4:0] will be given in continuously 7 cycles
Input	**coin_in[5:0]**	When in_valid is high, coin_in[5:0] will be given in continuously 4 cycles
Input	**btn_item[2:0]**	When in_valid is high, at the 5th cycle btn_item[2:0] will be given
Input	**btn_coin_rtn**	When in_valid is high, at the 5th cycle btn_coin_rtn will be given

Outputs

1. Output signals are synchronized at clock positive edge.
2. The TA's pattern will capture your output for checking at clock negative edge.

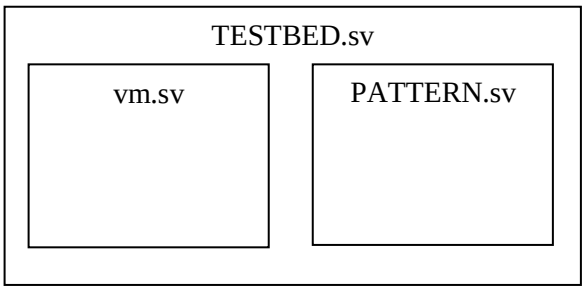
I/O	Signal name	Description
Output	**out_valid**	**out_valid** should be low after initial reset and not be raised when **in_valid is high**. **out_valid** should be set to high when your output value is ready, and will be high for **1** cycle.
Output	**item_out[2:0]**	Item_out = 2'b00 means no item is bought
Output	**coin_out_50**	The 50 coins which vending machine returnes
Output	**coin_out_10**	The 10 coins which vending machine returnes
Output	**coin_out_5**	The 5 coins which vending machine returnes
Output	**coin_out_1**	The 1 coins which vending machine returnes

Specifications

1. Top module name： **vm** (Filename: vm**.sv**)
2. Input ports: **clk, rst_n, in_valid_price, in_valid, btn_coin_rtn, coin_in[5:0], btn_item[2:0], item_price[4:0]**
3. Output ports: **out_valid, item_out[2:0], coin_out_50[2:0], coin_out_10[2:0], Coin_out_5, coin_out_1[2:0]**
4. It is an **asynchronous active-low reset** and **positive edge clk** architecture
5. The clock period of the design is **4ns**.
6. The input delay is set to **1.5ns**.
7. The output delay is set to **1.5ns**, and the output loading is set to **0.05**.
8. The next group of inputs will come in **1~9** cycles after your **out_valid** is over.

9. After synthesis, please check **syn.log** file that can not include any **Latch & Error**.
10. After synthesis, you can check alu.area and alu.timing. The area report is valid when the slack in the end of **timing report** is **non-negative**.
11. The latency for one case should be smaller than **1000 cycles**.

Block Diagram

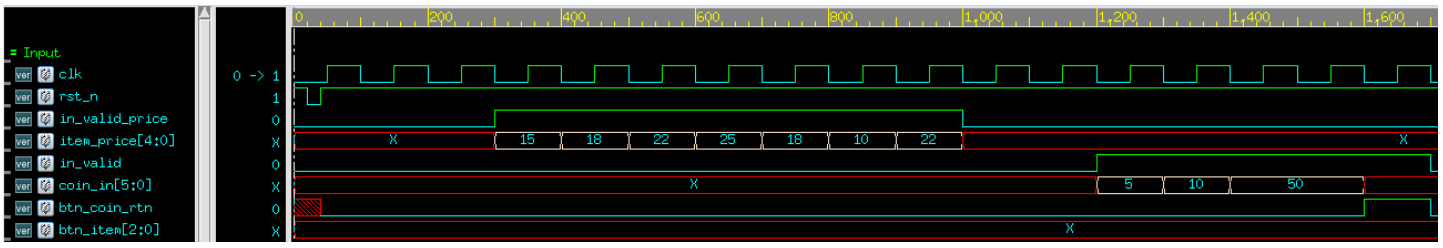


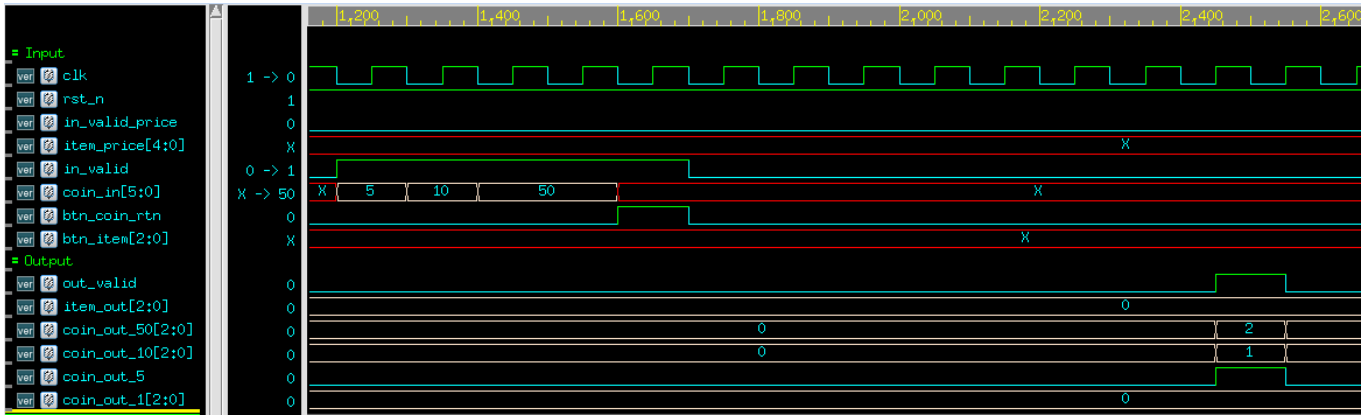
Note

1. Grading policy:
   a. Pass the RTL simulation
   b. Synthesis successful
      (**no Latch and Error**)
      (**slack in the timing report is non-negative**)
   c. Using For loop is not allowed.
   d. All registers should have reset function.
   e. Plagiarism is not allowed.
2. Template folders and reference commands:
   a. 01_RTL/ (RTL simulation) **./01_run**
   b. 02_SYN/ (Synthesis) **./01_run_dc**
      (Check the design which contain **latch** or not in **syn.log**
      (Check the design's timing in /Report/**core.timing**)

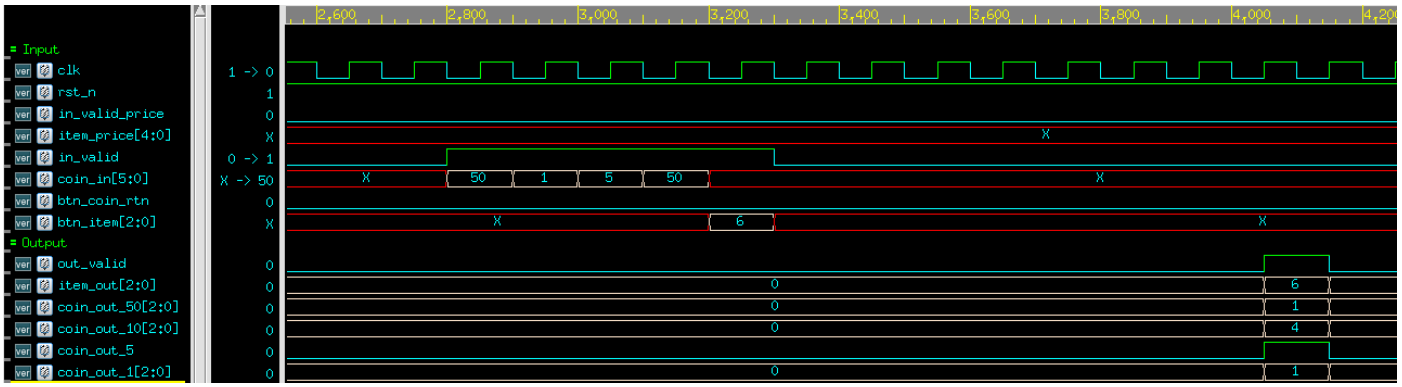
Sample Waveform

Return money





Buy item(get)



Buy item(not get)