NYCU-EE IC LAB – Fall 2024

OT Exercise

Design: Ramen Vending Machine

Data Preparation

1. Extract file from TA's directory:
   % tar xvf ~iclabTA01/2024_fall_iclab_OT.tar
2. The extracted LAB directory contains:
   a. **00_TESTBED**
   b. **01_RTL**
   c. **02_SYN**
   d. **03_GATE**
   e. **09_SUBMIT**

Design Description

A company finds that in NYCU, students are constantly looking for quick, satisfying meals. With long lines at restaurants, the vending machine provides the perfect solution. By inserting a coin and selecting the desired ramen options, users can get a freshly prepared bowl of ramen in just a few minutes. The machine will be designed to efficiently handle orders and ensure fast service, making it a great chose for students.

The ramen vending machine sells a total of four types of ramen: TONKOTSU, TONKOTSU_SOY, MISO, and MISO_SOY. The ingredients include tonkotsu soup, noodles, broth, soy sauce, and miso. Additionally, customers can choose between a large or small bowl. **Table 1** shows the quantities of each ingredient required for the different flavors and bowl sizes.

Portion = 1'b0 : Small Bowl					
ingredient / ramen_type	**noodle (g)**	**broth (ml)**	**tonkotsu soup (ml)**	**soy sauce (ml)**	**miso (ml)**
TONKOTSU (2'b00)	100(g)	300	150	0	0
TONKOTSU_SOY (2'b01)		300	100	30	0
MISO (2'b10)		400	0	0	30
MISO_SOY (2'b11)		300	70	15	15

Portion = 1'b1 : Big Bowl					
ingredient / ramen_type	noodle (g)	broth (ml)	tonkotsu soup (ml)	soy sauce (ml)	miso (ml)
TONKOTSU (2'b00)	150	500	200	0	0
TONKOTSU_SOY (2'b01)		500	150	50	0
MISO (2'b10)		650	0	0	50
MISO_SOY (2'b11)		500	100	25	25

Table 1. Ingredients List

For each pair of pattern, you will get 100~110 orders, and initially, you will have

12000g noodle, 41000ml broth, 9000ml tonkotsu soup, 1000ml miso, 1500ml soy_sause.

1. When the vending machine is operating, the selling signal will be pulled high.
2. For each order, **in_valid will be pulled high for 2 cycles**. The first cycle you will get the **ramen_type**, and next cycle you will get the **portion**.
3. You have to response success or not, and the out_valid_order have to be high at the same time.

   (if all the ingredients are sufficient to fulfill the order, **success = 1'b1**. Otherwise, **success = 1'b0**)
4. The machine get the response, next order will be sent.
5. After all the orders be sent, the selling signal be be pulled low, and you should output the total gain and the sold_num of each type of ramen as well as pull the out_valid_tot signal high.
6. Next pattern will come in after 2~4 cycles

   - sold_num[27:21] : number of TONKOTSU ramen be sold
   - sold_num[20:14] : number of TONKOTSU_SOY ramen be sold
   - sold_num[13:7] : number of MISO ramen be sold
   - sold_num[6:0] : number of MISO_SOY ramen be sold

- TONKOTSU ramen: $250,
- TONKOTSU_SOY ramen: $200
- MISO ramen: $250
- MISO_SOY ramen: $200

price of big portion = price of small portion

So the total gain = sold_num[27:21] * 200
+ sold_num[20:14] * 250
+ sold_num[13:7] * 200
+ sold_num[6:0] * 250

Inputs and Outputs

The following are the definitions of input signals

Input Signals	Bit Width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when all input is valid.
portion	1	1'b0 : small portion, 1'b1: big portion
ramen_type	2	2'b00 : TONKOTSU 2'b01 : TONKOTSU_SOY 2'b10 : MISO 2'b11 : MISO_SOY
selling	1	High when the vending machine is operating

The following are the definitions of output signals

Output Signals	Bit Width	Definition
out_valid_order	1	High when order response is valid.
success	1	order response: 1'b0 means there are not enough ingredients 1'b1 means sell successfully
out_valid_tot	1	High after finish selling
sold_num	28	sold_num[27:21] : TONKOTSU ramen be sold sold_num[20:14] : TONKOTSU_SOY ramen be sold sold_num[13:7] : MISO ramen be sold sold_num[6:0] : MISO_SOY ramen be sold
total_gain	15	Total gain for each pattern (selling period)

1. The input signal **ramen_type** is delivered in the first in_valid cycle, and **portion** is delivered in next cycle
2. All input signals are synchronized at negative edge of the clock.
3. The output signal **success** can only be delivered for **1 cycle**, and **out_valid_order** should be **high** simultaneously.
4. The output signal **total_gain and sold_num** can only be delivered for **1 cycle**, and **out_valid_tot** should be **high** simultaneously.
5. The **out** signal should be **zero** when **out_valid** is low.
6. The **out_valid** cannot overlap with **in_valid** at any time.

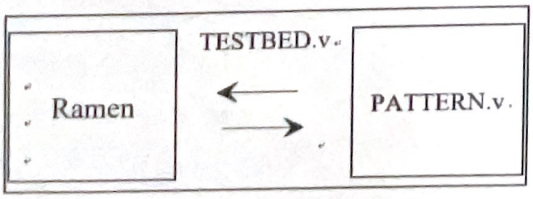
Specifications

1. Top module name: Ramen (design file name: Ramen.v)
2. **It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock starting) in your design, you may fail to reset signals.**
3. The reset signal (rst_n) would be given only once at the beginning of simulation. All output signals should be reset after the reset signal is asserted.
4. The **out** should be reset after your **out_valid** is pulled down.
5. The execution latency is limited in **100 cycles**. The latency is the clock cycles between the falling edge of the **in_valid** and the rising edge of the first **out_valid**.
6. The clock period is **10 ns,** and **you can't adjust your clock period** by yourself
7. The input delay is set to **0.5*(clock period)**.
8. The output delay is set to **0.5*(clock period)**, and the output loading is set to **0.05**.
9. The synthesis result of data type **cannot** include any **latches**.
10. After synthesis, you can check Ramen.area and Ramen.timing. The area report is valid when the slack in the end of timing report should be **non-negative (MET)**.

Grading Policy

1. Function Validity: 100%
2. 2nd_Demo will be 50%

Block diagram



Note

1. **Please submit following files under 09_SUBMIT before OT time finish:**
   - Ramen.v
   - If uploaded files violate the naming rule, you will get 5 deduct points.The 2nd demo deadline is **00:00 at noon on Oct. 30**.
   - Check whether there is any wire / reg / submodule name called "error", "fail", "pass", "congratulation", "latch", if you used, you will fail the lab.
2. **Template folders and reference commands:**

   01_RTL/ (RTL simulation) **./01_run_vcs_rtl**

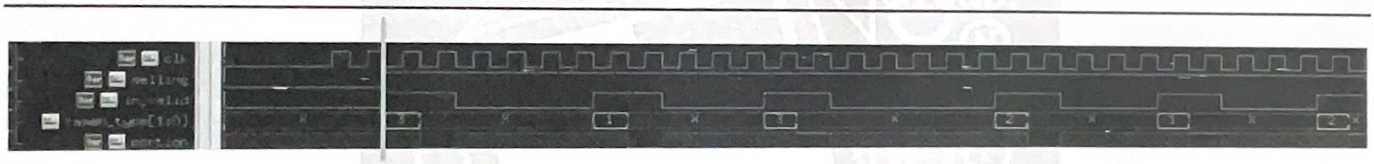
   02_SYN/ (Synthesis) **./01_run_dc_shell**

   (Check if there is any **latch** in your design in **syn.log)**

   (Check the timing of design in /Report/Ramen.**timing**)

   03_GATE / (Gate-level simulation) **./01_run_vcs_gate**

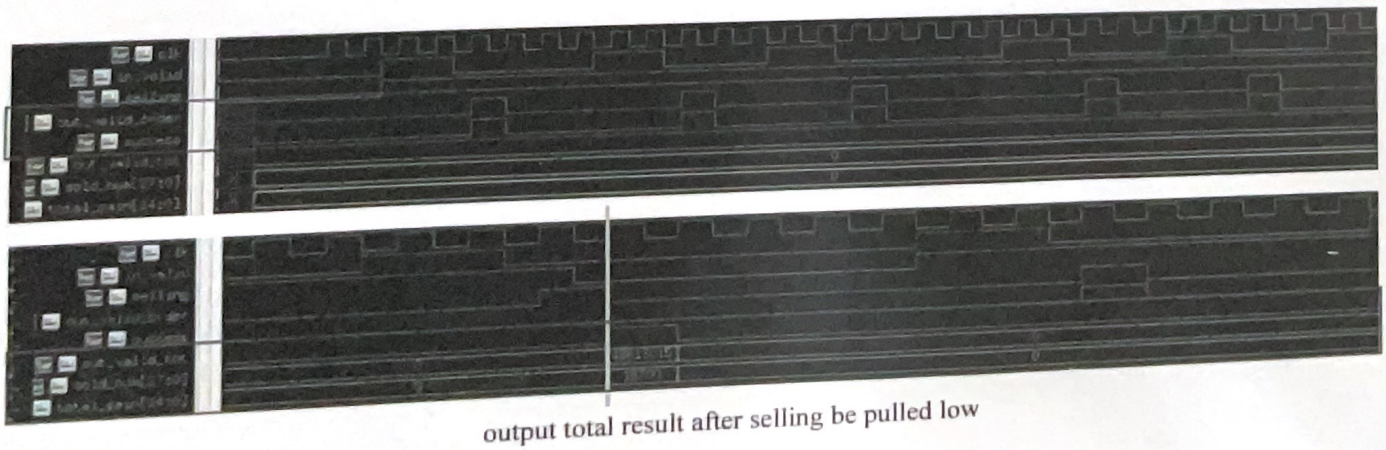
Sample Waveform



selling will be pulled high when the first order come in

selling will be pulled low when after the last order finish

Fig1. Input waveform



output total result after selling be pulled low

Fig2. Output waveform

You can check ing_left.txt in 00_TESTBED to track the situation of ingredients !!!

```
PAT_NUM=0
(ramen_type=1 porttion=1 success=1) noodle:-150=11850 broth:-500=40500 tonkotsu_soup:-150=8850 miso:-0=1000 soy_sauce:-50=1450
(ramen_type=1 porttion=0 success=1) noodle:-100=11750 broth:-300=40200 tonkotsu_soup:-100=8750 miso:-0=1000 soy_sauce:-30=1420
(ramen_type=1 porttion=1 success=1) noodle:-150=11600 broth:-500=39700 tonkotsu_soup:-150=8600 miso:-0=1000 soy_sauce:-50=1370
(ramen_type=2 porttion=1 success=1) noodle:-150=11450 broth:-650=39050 tonkotsu_soup:-0=8600 miso:-50=950 soy_sauce:-0=1370
(ramen_type=3 porttion=1 success=1) noodle:-150=11300 broth:-500=38550 tonkotsu_soup:-100=8500 miso:-25=925 soy_sauce:-25=1345
(ramen_type=2 porttion=0 success=1) noodle:-100=11200 broth:-400=38150 tonkotsu_soup:-0=8500 miso:-30=895 soy_sauce:-0=1345
(ramen_type=2 porttion=0 success=1) noodle:-100=11100 broth:-400=37750 tonkotsu_soup:-0=8500 miso:-30=865 soy_sauce:-0=1345
(ramen_type=0 porttion=1 success=1) noodle:-150=10950 broth:-500=37250 tonkotsu_soup:-200=8300 miso:-0=865 soy_sauce:-0=1345
(ramen_type=0 porttion=0 success=1) noodle:-100=10850 broth:-300=36950 tonkotsu_soup:-150=8150 miso:-0=865 soy_sauce:-0=1345
(ramen_type=0 porttion=1 success=1) noodle:-150=10700 broth:-500=36450 tonkotsu_soup:-200=7950 miso:-0=865 soy_sauce:-0=1345
(ramen_type=0 porttion=1 success=1) noodle:-150=10550 broth:-500=35950 tonkotsu_soup:-200=7750 miso:-0=865 soy_sauce:-0=1345
(ramen_type=2 porttion=0 success=1) noodle:-100=10450 broth:-400=35550 tonkotsu_soup:-0=7750 miso:-30=835 soy_sauce:-0=1345
(ramen_type=0 porttion=1 success=1) noodle:-150=10300 broth:-500=35050 tonkotsu_soup:-200=7550 miso:-0=835 soy_sauce:-0=1345
(ramen_type=2 porttion=0 success=1) noodle:-100=10200 broth:-400=34650 tonkotsu_soup:-0=7550 miso:-30=805 soy_sauce:-0=1345
(ramen_type=3 porttion=0 success=1) noodle:-100=10100 broth:-300=34350 tonkotsu_soup:-70=7480 miso:-15=790 soy_sauce:-15=1330
(ramen_type=3 porttion=0 success=1) noodle:-100=10000 broth:-300=34050 tonkotsu_soup:-70=7410 miso:-15=775 soy_sauce:-15=1315
(ramen_type=2 porttion=1 success=1) noodle:-150=9850 broth:-650=33400 tonkotsu_soup:-0=7410 miso:-50=725 soy_sauce:-0=1315
(ramen_type=3 porttion=1 success=1) noodle:-150=9700 broth:-500=32900 tonkotsu_soup:-100=7310 miso:-25=700 soy_sauce:-25=1290
(ramen_type=3 porttion=0 success=1) noodle:-100=9600 broth:-300=32600 tonkotsu_soup:-70=7240 miso:-15=685 soy_sauce:-15=1275
(ramen_type=0 porttion=0 success=1) noodle:-100=9500 broth:-300=32300 tonkotsu_soup:-150=7090 miso:-0=685 soy_sauce:-0=1275
(ramen_type=0 porttion=0 success=1) noodle:-100=9400 broth:-300=32000 tonkotsu_soup:-150=6940 miso:-0=685 soy_sauce:-0=1275
(ramen_type=1 porttion=1 success=1) noodle:-150=9250 broth:-500=31500 tonkotsu_soup:-150=6790 miso:-0=685 soy_sauce:-50=1225
(ramen_type=3 porttion=0 success=1) noodle:-100=9150 broth:-300=31200 tonkotsu_soup:-70=6720 miso:-15=670 soy_sauce:-15=1210
(ramen_type=1 porttion=0 success=1) noodle:-100=9050 broth:-300=30900 tonkotsu_soup:-100=6620 miso:-0=670 soy_sauce:-30=1180
(ramen_type=0 porttion=1 success=1) noodle:-150=8900 broth:-500=30400 tonkotsu_soup:-200=6420 miso:-0=670 soy_sauce:-0=1180
(ramen_type=2 porttion=0 success=1) noodle:-100=8800 broth:-400=30000 tonkotsu_soup:-0=6420 miso:-30=640 soy_sauce:-0=1180
(ramen_type=2 porttion=0 success=1) noodle:-100=8700 broth:-400=29600 tonkotsu_soup:-0=6420 miso:-30=610 soy_sauce:-0=1180
```