NCTU-EE ICLAB – Autumn 2023

Lab09 Exercise: Design and Verification Using SystemVerilog

Design: Tea House

Data Preparation

1. Extract the test data from TA's directory:
   % tar xvf ~iclabTA01/Lab09.tar
2. The extracted LAB directory contains:
   Practice/
   Exercise/

Design Description

This assignment aims to simulate the beverage dispensing process. We need to inform customers about the current inventory levels of various ingredients in the raw material barrels based on the type, and the volume of the drinks they order, ensuring there is enough stock to fulfill their requests. After each beverage is prepared, we also want to monitor the inventory of ingredients in the background. In addition to customer orders, periodic replenishment of ingredients will be carried out to ensure an ample supply in the raw material barrels. Lastly, it is crucial to verify the expiration date of the beverage barrels during each functional operation.

Operations:

- **Make Drink**
  Input:
  1. Choose the type of beverage.
  2. Select the volume.
  3. Today's date.
  4. Select the ingredient box in DRAM.

- **Supply**
  Input:
  1. Expired date
  2. Select the ingredient box in DRAM.
  3. Material
     A. Black Tea
     B. Green Tea
     C. Milk
     D. Pineapple Juice

- **Check Valid Date**

  Input:

  1. Today's Date
  2. Select the ingredient box in DRAM.

Information:

Table 1: The Content of Ingredient Box (Store in the DRAM)

Bits	Ingredient Type	Max. Value (Dec.)
12	Black Tea	4095
12	Green Tea	4095
12	Milk	4095
12	Pineapple Juice	4095
8	Expire Date (Month)	12
8	Expire Date (Day)	31

Note:

1. The order of content in the ingredient box in the DRAM from MSB to LSB is: **[MSB] {Black_Tea, Green_Tea, Expire_Date (Month), Milk, Pineapple_Juice, Expire_Date (Day)} [LSB]**
2. February only has 28 days in this design.

Table 2: Beverage Volume

Size	Input	Beverage Volume (Dec.)
L	2'b00	960
M	2'b01	720
S	2'b11	480

Table 3: Type & Ingredient ratio of Beverage

Beverage Type	Input	Black Tea Ratio	Green Tea Ratio	Milk Ratio	Pineapple Juice Ratio
Black Tea	0	1	0	0	0
Milk Tea	1	3	0	1	0
Extra Milk Tea	2	1	0	1	0
Green Tea	3	0	1	0	0
Green Milk Tea	4	0	1	1	0
Pineapple Juice	5	0	0	0	1
Super Pineapple Tea	6	1	0	0	1
Super Pineapple Milk Tea	7	2	0	1	1

Table 4: The corresponding value of actions and error messages

ACTION		ERROR MESSAGE	
Make Drink	2'b00	No_Exp	2'b01
		No_Ing	2'b10
Supply	2'b01	Ing_OF	2'b11
Check Valid Date	2'b10	No_Exp	2'b01
All action		No_Err	2'b00

Note: The number in parentheses indicates the priority. The smaller, the higher. If the operation results in several errors at the same time, output the one with the highest priority.

The following are some rules about the actions:

1. The action will not be complete if the action outputs the error message.
2. There will be 1~4 cycles for each input.
3. The date should adhere to the real calendar, that is, 03/21, 07/22, 11/13, 11/15, 12/24 is legal; 02/29, 04/31, 05/00 is illegal.
4. If today's date is later (>) than the expired date, you need to set error message to "No_Exp (2'b01)".

Make Drink

5. When **Sel_action_valid** is high and input signal **D** = 0 (Make Drink), conduct this action.
6. The input order for **D** will be: Beverage Type -> Beverage Size -> Today's Date -> No. of Ingredient barrel in DRAM.
7. Check the expiration date of the selected ingredient box. If today's date is later (>) than the expiration date, stop making drink and raise the error message to "No_Exp (2'b01)"
8. Check the content of each ingredient in the ingredient box. If any one of the ingredients is insufficient for beverage production, stop the beverage production. Set the error message to "No_Ing (2'b10)"
9. If no errors occur, proceed to make the beverage according to the requirements. Upon completion, deduct the consumed amount of raw materials from the ingredient box. Finally, pull the signal 'Complete' to HIGH, and set the error message to ' No_Err' (2'b00).

Supply

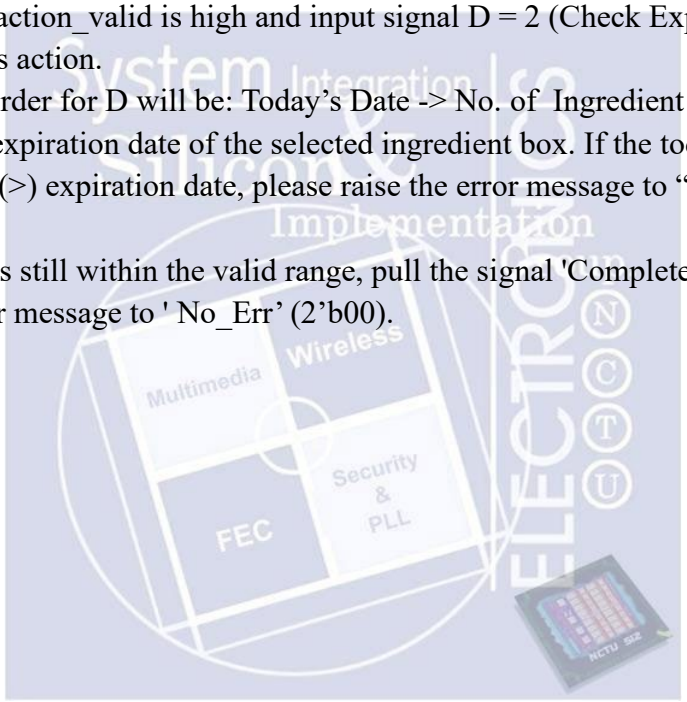
10. When Sel_action_valid is high and input signal D = 1 (Supply), conduct this

action.

11. The input order for **D** will be: Expired Date -> No. of Ingredient barrel in DRAM -> Black_Tea -> Green_Tea -> Milk -> Pineapple Juice.
12. If any of the ingredients exceed the upper limit, the error message will be “Ing_OF (2’b11)”.
13. No matter the ingredients is exceed or not, continue to add ingredients to the ingredient box; if it exceeds the upper limit, set it to the maximum capacity the ingredient box can hold (4095).
14. Replace the expired date saved in ingredient box to INPUT’s expired date.
15. If none of the ingredients exceed the upper limit, pull 'COMPLETE' to high. Set Error Message to “No_Err (2’b00)”

Check Expired Date

16. When Sel_action_valid is high and input signal D = 2 (Check Expired Date), conduct this action.
17. The input order for D will be: Today’s Date -> No. of Ingredient box in DRAM.
18. Check the expiration date of the selected ingredient box. If the today’s date is larger than (>) expiration date, please raise the error message to “No_Exp (2’b01)”.
19. If the date is still within the valid range, pull the signal 'Complete' to HIGH, and set the error message to ' No_Err’ (2’b00).



Design Block Diagram

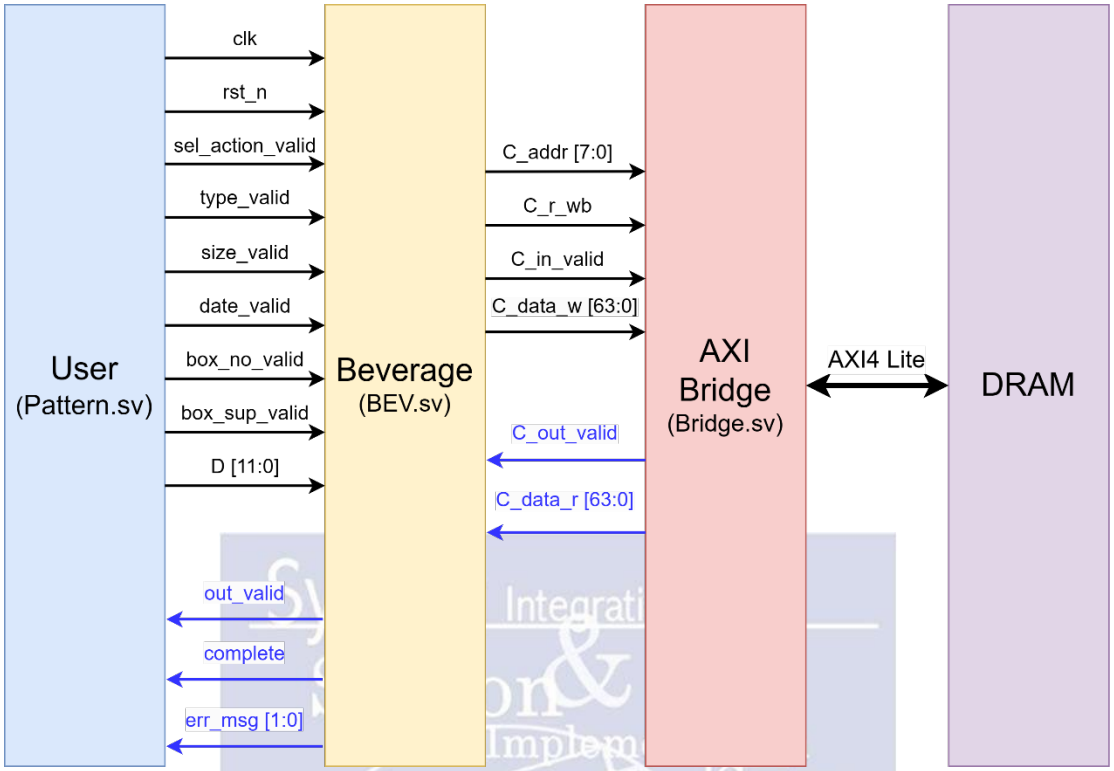


Figure 1: The Design Block Diagram

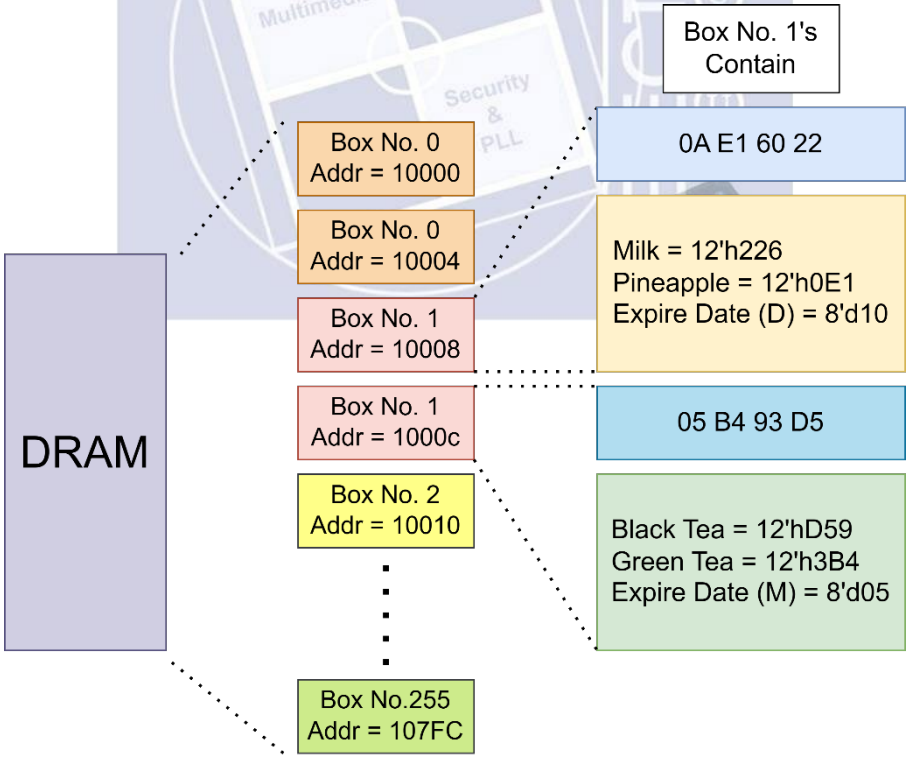


Figure 2: Ingredient box Content Diagram

Design Rules

1. After **rst_n**, all output signals (both Bev.sv and Bridge.sv) should be set to 0.
2. Please initialize DRAM at the beginning. (Please refer to 2023_Autumn_Lab09_Exercise_note.pdf)
3. DRAM_R_latency, DRAM_W_latency, DRAM_B_latency in pseudo_dram.sv is now set to 1, but you can modify it. **TA will change the value (1<= value <=100) while demo.**
4. The pattern will be inserted using **6 valid signals + 1 data signals**:

Signal	Means
sel_action_valid	High when input means select the action.
type_valid	High when input means type of beverage.
size_valid	High when input means the size of beverage.
date_valid	High when input means today's date or expired date.
box_no_valid	High when input means the number of ingredient barrel.
Box_sup_valid	High when input means supply material (Will pull HIGH 4 times)
D[71:0]	D = {10'b0, Action} (inf.D.d_act [0]) = {9'b0, Berverage_Type} (inf.D.d_type[0]) = {10'b0, Beverage_Size} (inf.D.d_size[0]) = {3'b0, Month[3:0], Day[4:0]} (inf.D. d_date[0]) = {4'b0, Box No.} (inf.D.d_box_no[0]) = {Material (Black/Green Tea, Milk, Pineapple Juice)} (inf.D. d_ing[0])

5. You need to raise **out_valid** only when your design has done current input operation.
6. We can get inventory of materials in ingredient box from DRAM via AXI Lite protocol.
7. C_in_valid can only be high for one cycle and cannot be pulled high again before C_out_valid.
8. If action is '**Make Drink**'
   - **Success**: When out_valid is HIGH, "success" should be HIGH and "error message" should be No_Err (2'b00).
   - **Fail:**
     i. **Expired Date is OVER:** When out_valid is HIGH, "success" should be LOW and "error message" should be "No_Exp" (2'b01).
     ii. **Ingredient not enough:** When out_valid is HIGH, "success" should be LOW and "error message" should be "No_Ing"(2'b10).

9. If action is '**Supply**':
    - Update the inventory of raw materials in the ingredient box.
    - **Success:** When out_valid is HIGH, "success" should be HIGH and "error message" should be No_Err (2'b00).
    - **Fail:**
        i. **Overflow:** When out_valid is HIGH, "success" should be LOW and "error message" should be "No_Ing (2'b11). The inventory of raw materials in the ingredient box STILL need to be updated, up to 4095.
10. If action is '**Check Expired Date**':
    - **Success**: When out_valid is HIGH, "success" should be HIGH and "error message" should be "No_Err" (2'b00).
    - **Fail**: When out_valid is HIGH, "success" should be LOW and "error message" should be "No_Exp" (2'b01).

11. The 6 valid input signals will not overlap with each other.
12. The next valid input signal will be valid for 1~4 cycles after the valid input signal fall.
13. Out_valid cannot overlap with the 6 input valid signals.
14. Out_valid should be high for exactly one cycle.
15. Out_valid can only be high after giving all necessary input valid signals.
16. TA will check the output signal when the out_valid is high.
17. If actions complete, complete should be high and err_msg should be 2'b00.
18. If action is not complete, complete should be the low, err_msg should be corresponding value.
19. The next operation will be valid for **1-4** cycles after the out_valid fall.
20. The system will NOT check the data stored in DRAM.

For the definition of cycles between signals, please refer to the **2023_Autumn_Lab09_Exercise_note.pdf**

Input: Bev.sv (Bev : Beverage System)

Signal	Width	from	Note
clk	1	testbench	System clock
rst_n	1	pattern	Asynchronous reset active low reset. Every output signal should be zero after **rst_n**.
sel_action_valid	1	pattern	High when input means select the action.
type_valid	1	pattern	High when input means type of beverage.
size_valid	1	pattern	High when input means the size of beverage.
date_valid	1	pattern	High when input means today's date or expired date.
box_no_valid	1	pattern	High when input means the number of ingredient barrel.
Box_sup_valid	1	pattern	High when input means supply material (Will pull HIGH 4 times)

D[71:0]	72	pattern	Represents the contents of the current input. = {10'b0, Action} = {9'b0, Berverage_Type} = {10'b0, Beverage_Size} = {3'b0, Month[3:0], Day[4:0]} = {4'b0, Box No.} = {Material (Black/Green Tea, Milk, Pineapple Juice)}
C_out_valid	1	bridge	High when the data from the DRAM is ready.
C_data_r	64	bridge	The returned data from DRAM.

Output: Bev.sv

Signal	Width	Send to	Note
out_valid	1	pattern	Should set to high when your output is ready. **out_valid** will be high for **only one** cycle.
err_msg	2	pattern	err_msg will be 2'b00 (No error) if operation is complete, else it needs to be corresponding value.
complete	1	pattern	1'b1: operation complete 1'b0: some error occurred
C_addr	8	bridge	Indicates which address we want to access.
C_data_w	64	bridge	The data to overwrite DRAM.
C_in_valid	1	bridge	High when the beverage system is ready to communicate with the bridge.
C_r_wb	1	bridge	1'b1: Read DRAM. 1'b0: Write DRAM.

Input: Bridge.sv

Signal	width	from	Note
clk	1	testbench	System clock
rst_n	1	pattern	Asynchronous reset active low reset. Every output signal should be zero after **rst_n**.
C_addr	8	BEV	Indicates which address we want to access.
C_data_w	64	BEV	The data to overwrite DRAM.
C_in_valid	1	BEV	High when the Beverage system is ready to communicate with the bridge.
C_r_wb	1	BEV	1'b1: Read DRAM. 1'b0: Write DRAM.
AR_READY	1	DRAM	AXI Lite signal
R_VALID	1	DRAM	AXI Lite signal
R_DATA	64	DRAM	AXI Lite signal
R_RESP	2	DRAM	AXI Lite signal
AW_READY	1	DRAM	AXI Lite signal
W_READY	1	DRAM	AXI Lite signal
B_VALID	1	DRAM	AXI Lite signal
B_RESP	2	DRAM	AXI Lite signal

Output: Bridge.sv

Signal	width	Send to	Note
C_out_valid	1	BEV	High when data from DRAM is ready.
C_data_r	64	BEV	The returned data from DRAM
AR_VALID	1	DRAM	AXI Lite signal
AR_ADDR	17	DRAM	AXI Lite signal
R_READY	1	DRAM	AXI Lite signal
AW_VALID	1	DRAM	AXI Lite signal
AW_ADDR	17	DRAM	AXI Lite signal
W_VALID	1	DRAM	AXI Lite signal
W_DATA	64	DRAM	AXI Lite signal
B_READY	1	DRAM	AXI Lite signal

Specifications

Top module

1. Top module name: **BEV** (file name: **Bridge.sv** & **BEV.sv**)
2. Your design should trigger at positive edge clock, and the pattern will trigger and receive input/output signal at negative edge clock.

Reset

3. It is asynchronous reset and active-low architecture. If you use synchronous reset (reset after clock starting) in your design, you may fail to reset signals.
4. The reset signal(**rst_n)** would be given only once at the beginning of simulation. All output signals (including Bev.sv and Bridge.sv) should be reset after the reset signal is asserted.

Design Constraints

5. The maximum clock period is **15 ns**.
6. Your latency should be **less than 1000 cycles** for each operation.
7. All outputs (including Bev.sv and Bridge.sv) are synchronized at clock rising edge.
8. The type defined in Usertype_BEV.sv by TA **should not be modified, but you are encouraged to define a new datatype if needed**.
9. C_in_valid can only be high for one cycle, and cannot be pulled high again before C_out_valid.
10. Out_valid can only be high for exactly one cycle.
11. You CANNOT use any designware IP and SRAM in this lab.

Synthesis

12. The input delay and the output delay are **'0.5*clock period**'.
13. The output load should be set to **0.05.**
14. The synthesis result of data type cannot include any **LATCH.**
15. Synthesis time should less than 30 minutes.
16. The total area (bridge_area + BEV_area) should be **less than 70000.**

Gate level simulation

17. The gate-level simulation cannot include any timing violations without the *notimingcheck* command.

Supplement

18. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *latch* or *fail* otherwise you will fail the lab. Note: * means any char in front of or behind the word. e.g: error_note is forbidden.
19. Don't write Chinese or other language comments in the file you sent.

20. Verilog command for **design compiler optimizations are forbidden**.
21. **Any error messages** during synthesize and simulation, regardless of the result will lead to failure in this lab.
22. Any form of display or printing information in Verilog design is forbidden. You may use this methodology during debugging, but the file you turn in **should not contain any coding that is not synthesized.**

Grading

- **70%: Function**
- **30%: Performance: (bridge_Area + Bev_Area) x latency x cycle_time**

Note

- Submit your design (Bev.sv, Bridge.sv) in Lab09/EXERCISE/09_SUBMIT
  a) 1st_demo deadline : 2023/12/04 (Mon.) 12:00:00
  b) 2nd_demo deadline: 2023/12/06 (Wed.) 12:00:00
- **Please upload the following files under 09_SUBMIT:**
  - **In this lab, you can adjust your clock cycle time. Consequently, make sure to key in your clock cycle time after the command like the figure below. It's means that the TA will demo your design under this clock cycle time.**

    ```
    ./01_submit 8.0
    ```
  - BEV.sv, Usertype_BEV.sv, Bridge.sv
  - If your files violate the naming rule, you will get 10 deduction points.

※ **Since Lab10 is about pattern with SystemVerilog, if you need to share your pattern with others, use the following command to encrypt your PATTERN. It will generate "PATTERN.svp" file.**

```
[~/Lab09/Exercise/00_TESTBED]$ ./00_ENC_PATTERN
```

Since we have provided the way to protect your file, there will be no excuse for plagiarism.

Reference Waveform

Please refer to **2023_Autumn_Lab09_Exercise_note.pdf**