NYCU-EE ICLAB – Autumn 2024

Lab09 Exercise: Design and Verification Using SystemVerilog

Design: Stock Trading Program

Data Preparation

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

Design Description

The **Stock Trading Program** is designed to assess current market risks by utilizing four key indices that provide a comprehensive analysis of market conditions. The program includes past data that stores past indices data in early trading session, enabling in-depth analysis and trend identification over time. It also features an automated system for updating both the date and the indices data to ensure that the information is always current and accurate. By leveraging the indices and regularly refreshed data, the program allows users to make decisions based on understanding of potential market risks.

Operations:

- **Index Check**
  Input:
  1. Choose the type of formula.
  2. Select the mode.
  3. Today's date.
  4. Select the No. of data in DRAM.
  5. Today's indices in late trading session.
     A. Index A
     B. Index B
     C. Index C
     D. Index D

- **Update**
  Input:
  1. A given data date
  2. Select the No. of data in DRAM.
  3. Variations of the indices data.
     A. Variation of Index A
     B. Variation of Index B

C. Variation of Index C
D. Variation of Index D

- **Check Valid Date**

  Input:

  1. Today's date
  2. select the No. of data in dram

Information:

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

Bits	Index Type	Max. Value (Dec.)
12	Index A	4095
12	Index B	4095
12	Index C	4095
12	Index D	4095
8	Data Date (Month)	12
8	Data Date (Day)	31

Note:

1. The order of content in the data in the DRAM from MSB to LSB is: **[MSB] { Index A, Index B, Data Date (Month), Index C, Index D, Data Date (Day)} [LSB]**
2. February only has 28 days in this design.

Table 2: Mode

Mode	Input
Insensitive	2'b00
Normal	2'b01
Sensitive	2'b11

Table 3: Threshold corresponding to Formula and Mode

Formula Threshold / Mode	Insensitive	Normal	Sensitive
Formula A Threshold	2047	1023	511
Formula B Threshold	800	400	200
Formula C Threshold	2047	1023	511
Formula D Threshold	3	2	1
Formula E Threshold	3	2	1
Formula F Threshold	800	400	200
Formula G Threshold	800	400	200
Formula H Threshold	800	400	200

Table 4: Formula

Formula	Input	Content
A	3'd0	$R = \left\lfloor \left(\frac{I(A)+I(B)+I(C)+I(D)}{4}\right) \right\rfloor$
B	3'd1	$R = max(I(A), I(B), I(C), I(D)) - \min(I(A), I(B), I(C), I(D))$
C	3'd2	$R = \min(I(A), I(B), I(C), I(D))$
D	3'd3	$R = 1_{\{I(A)\geq 2047\}} + 1_{\{I(B)\geq 2047)\}} + 1_{\{I(C)\geq 2047\}} + 1_{\{I(D)\geq 2047\}}$ $Ex: I(A) = I(B) = 0, I(C) = I(D) = 4095, R = 2$
E	3'd4	$R = 1_{\{I(A)\geq TI(A)\}} + 1_{\{I(B)\geq TI(B)\}} + 1_{\{I(C)\geq TI(C)\}} + 1_{\{I(D)\geq TI(D)\}}$
F	3'd5	$n = argmax_{n\in\{A,B,C,D\}}(G(n)),$ $R = floor(\frac{1}{3}(\sum_{i\neq n, i\epsilon\{A,B,C,D\}} G(i)))$ $Ex: G(A) = G(B) = G(C) = 3, G(D) = 5,$ $R = \frac{G(A)+G(B)+G(C)}{3} = 3$
G	3'd6	$N_0, N_1, N_2, N_3 = ascended.sort(G(A), G(B), G(C), G(D)),$ $R = \left\lfloor \left(\frac{N_0}{2}\right) \right\rfloor + \left\lfloor \left(\frac{N_1}{4}\right) \right\rfloor + \left\lfloor \left(\frac{N_2}{4}\right) \right\rfloor$ $Ex: G(A) = 5, G(B) = G(C) = 4, G(D) = 2$ $N_0, N_1, N_2, N_3 = 2,4,4,5 \;\; R = 3$
H	3'd7	$R = \left\lfloor \frac{G(A)+G(B)+G(C)+G(D)}{4} \right\rfloor$

Note: R means result. I(A), I(B), I(C), I(D) are early trading indices A, B, C, D from dram respectively. TI(A), TI(B), TI(C), TI(D) are today's late trading session indices A, B, C, D respectively. G(A), G(B), G(C), G(D) are $|I(A) - TI(A)|$, $|I(B) - TI(B)|$, $|I(C) - TI(C)|$, $|I(D) - TI(D)|$ respectively. If $R \geq Threshold$, you must issue the warning message.

Table 5: The corresponding value of actions and warning messages

ACTION		WARNING MESSAGE	
Index Check	2'b00	Date_Warn	2'b01
		Risk_Warn	2'b10
Update	2'b01	Data_Warn	2'b11

Check Valid Date	2’b10	Date_Warn	2’b01
All action		No_Warn	2’b00

Note: The number in parentheses indicates the priority. The smaller, the higher. If the operation results in several warnings 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 warning 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 earlier (<) than the dram’s date, you need to set warning message to “Date_Warning (2’b01)” on the operations of Index Check and Check Valid Date.
5. Keep the 'Complete' signal low until finishing an operation without any warnings.
6. In the Index Check operation, when the 'index_valid' signal is pulled high, it means that the D input represents today’s index in the late trading session, ranging from 0 to 4095. In the Update operation, when the 'index_valid' signal is pulled high, it means that the D input represents the variation of index A to D, ranging from -2048 to 2047.

Index Check

7. When **Sel_action_valid** is high and input signal **D** = 0 (Index Check), conduct this action.
8. The input order for **D** will be: Formula Type -> Mode -> Today’s Date -> No. of data in DRAM -> Today’s Index A in late trading session -> Today’s Index B in late trading session -> Today’s Index C in late trading session -> Today’s Index D in late trading session.
9. Check the data’s date of the selected dram data. If today’s date is earlier (<) than the dram date, stop the program and raise the warning message to “Date_Warn (2’b01)”.
10. Check the result of selected formula. If $Result \geq Threshold$, stop the program. Set the warning message to “Risk_Warn (2’b10)”.
11. Finally, if no warnings issued, pull the signal 'Complete' to HIGH, and set the warning message to ' No_Warn’ (2’b00).

Update

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

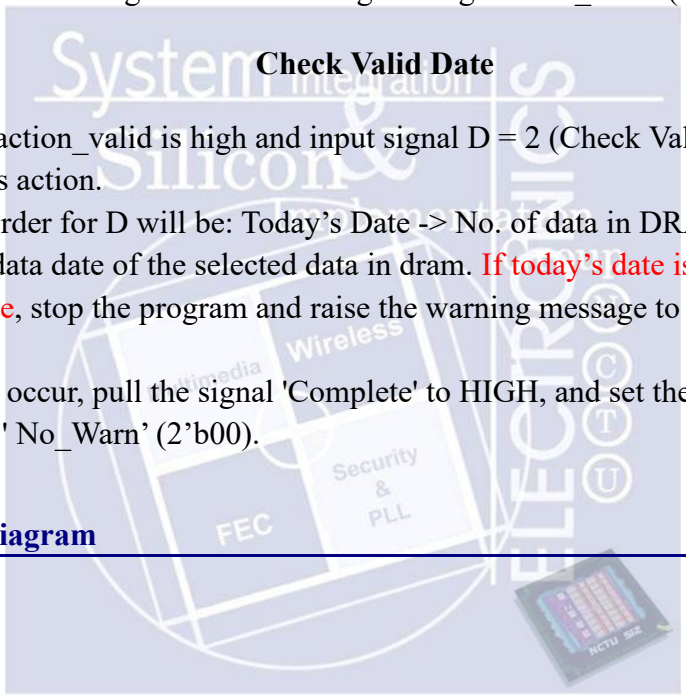
action.

13. The input order for **D** will be: Data Date -> No. of data in DRAM -> Variation in Index A in early trading session -> Variation in Index B in early trading session -> Variation in Index C in early trading session -> Variation in Index D in early trading session.
14. If any of the indices exceed the upper limit or are below the lower limit, the warning message will be "Data_Warn (2'b11)".
15. No matter whether the index data exceeds or is below, continue to add the variation to the index data; if it exceeds the upper limit, set it to the maximum capacity the index data can hold (4095); if it falls below the lower limit, set it to the minimum capacity the index data can hold (0).
16. Replace the data date saved in data to INPUT's given date.
17. If none of the index data exceeds the upper limit or falls below the lower limit, pull 'COMPLETE' high. Set the warning message to 'No_Warn (2'b00)'.

Check Valid Date

18. When Sel_action_valid is high and input signal D = 2 (Check Valid Date), conduct this action.
19. The input order for D will be: Today's Date -> No. of data in DRAM
20. Check the data date of the selected data in dram. If today's date is earlier (<) than the data date, stop the program and raise the warning message to "Date_Warn (2'b01)".
21. If no warns occur, pull the signal 'Complete' to HIGH, and set the warning message to ' No_Warn' (2'b00).

Design Block Diagram



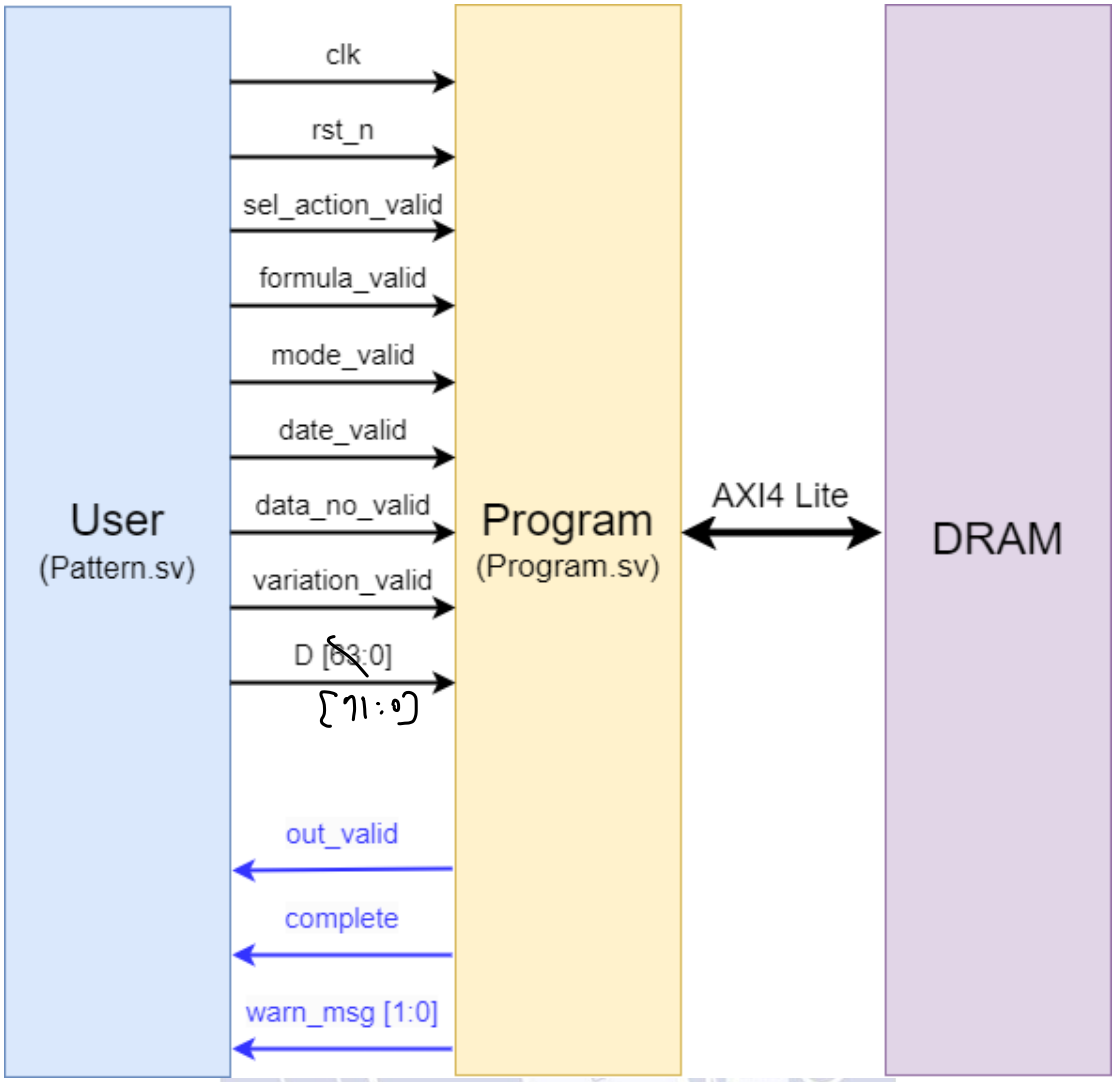


Figure 1: The Design Block Diagram

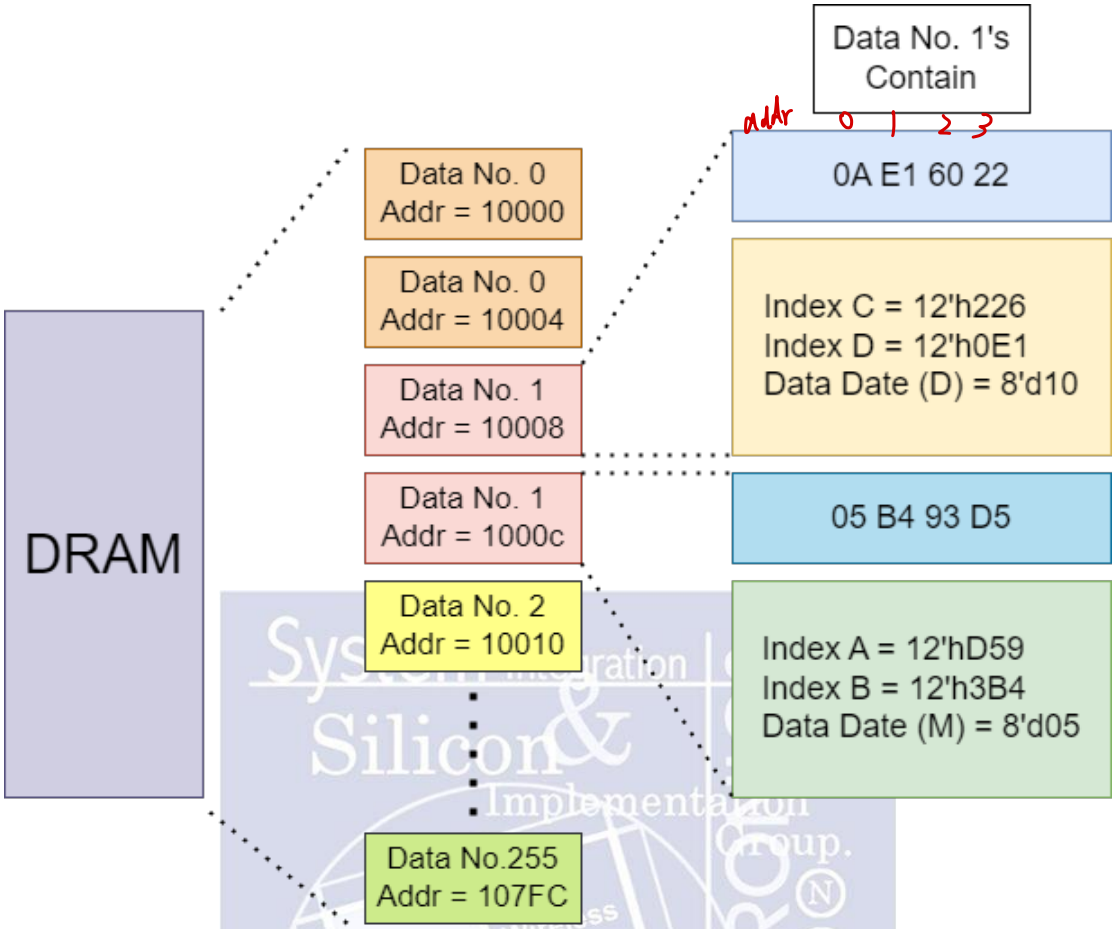


Figure 2: Dram Data Content Diagram

Design Rules

1. After **rst_n**, all output signals in Program.sv should be set to 0.
2. Please initialize DRAM at the beginning.
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.
formula_valid	High when input means type of formula.
mode_valid	High when input means the Mode.
date_valid	High when input means today's date or data's date.
data_no_valid	High when input means the number of data.
index_valid	High when input means variation in index (Will pull HIGH 4 times for index A, B, C, D)
D[71:0]	D = {70'bX, Action} (inf.D.d_act [0]) = {69'bX, Formula_Type} (inf.D.d_formula[0]) = {70'bX, Mode} (inf.D.d_mode[0]) = {63'bX, Month[3:0], Day[4:0]} (inf.D. d_date[0]) = {64'bX, Data No.} (inf.D.d_data_no[0]) = {60'bX, today's Index or variation in Index} (Today's index in the late trading session for the index check action or variation in the index for the update action.) (inf.D. d_index[0])

5. You need to raise **out_valid** only when all input_valid signals are transmitted and your design has done current input operation.
6. We can get selected data from DRAM via AXI Lite protocol.
7. If action is '**Index Check**'
   - **Success**: When out_valid is HIGH, "complete" should be HIGH and "warning message" should be No_Warn (2'b00).
   - **Fail:**
     i. **Today's Date is earlier:** When out_valid is HIGH, "complete" should be LOW and "warning message" should be "Date_Warn" (2'b01).
     ii. **Risk Warning:** When out_valid is HIGH, "complete" should be LOW and "warning message" should be "Risk_Warn"(2'b10).
8. If action is '**Update**':
   - Update the selected data.
   - **Success:** When out_valid is HIGH, "complete" should be HIGH and "warning message" should be No_Warn (2'b00).
   - **Fail:**
     i. **Index overflow or underflow:** When out_valid is HIGH, "complete" should be LOW and "warning message" should be "Data_Warn (2'b11)". The selected indices data STILL need to be updated, up to 4095 or to 0.

9. If action is '**Check Valid Date**':
   - **Success**: When out_valid is HIGH, "complete" should be HIGH and "warning message" should be "No_Warn" (2'b00).
   - **Fail:**
     i. **Today's Date is Earlier:** When out_valid is HIGH, "complete" should be LOW and "warning message" should be "Date_Warn" (2'b01).
10. The 6 valid input signals will not overlap with each other.
11. The next valid input signal will be valid for 1~4 cycles after a input valid signal or out_valid signal fall.
12. Out_valid cannot overlap with the 6 input valid (sel_action_valid, formula_valid, mode_valid, date_valid, data_no_valid, index_valid) signals.
13. No checks will be performed by the TA for any overlaps between AXI4 Lite signals and the 6 input valid or out_valid signals.
14. Out_valid should be high for exactly one cycle.
15. Out_valid can only be high after giving all 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 warn_msg should be 2'b00.
18. If action is not complete, complete should be the low, warn_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.

To understand how AXI4LITE works, please refer to the **Lab09_AXI4LITE_Note.pdf**

Input: Program.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**.
sel_action_valid	1	pattern	High when input means select the action.
formula_valid	1	pattern	High when input means type of formula.
mode_valid	1	pattern	High when input means the Mode.
date_valid	1	pattern	High when input means today's date or expired date.
data_no_valid	1	pattern	High when input means the number of data.
index_valid	1	pattern	High when input means variation (Will pull HIGH 4 times)
D[71:0]	72	pattern	Represents the contents of the current input. = {70'bX, Action} = {69'bX, Formula_Type} = {70'bX, Mode} = {63'bX, Month[3:0], Day[4:0]}

			= {64'bX, Data No.} = {60'bX, Index or Variation}
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: Program.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.
warn_msg	2	pattern	warn_msg will be 2'b00 (No warn) if operation is complete, else it needs to be corresponding value.
complete	1	pattern	1'b1: operation complete 1'b0: some warnings occurred
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: **Program** (file name: **Program.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 Program.sv) should be reset after the reset signal is asserted.
5. All signal should be reset to 0.

Design Constraints

6. The maximum clock period is **15 ns**.
7. Your latency should be **less than 1000 cycles** for each operation.
8. All outputs (including Program.sv) are synchronized at clock rising edge.
9. The type defined in Usertype.sv by TA **should not be modified, but you are encouraged to define a new datatype if needed**.
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 (Program_area) should be **less than 120000.**

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: (Program_Area) x latency x cycle_time**

Note

- Submit your design (Programe.sv) in Lab09/EXERCISE/09_SUBMIT
  a) 1st_demo deadline : 2024/11/25 (Mon.) 12:00:00
  b) 2nd_demo deadline: 2024/11/27 (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
```

- Program.sv, Usertype.sv
- If your files violate the naming rule, you will get 10 deduction points.

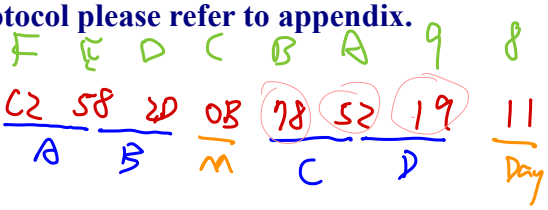
※ **Since Lab10 is about pattern with SystemVerilog, if you need to share your pattern with others, use the following command to encrypt your PATTERN. It will generate "PATTERN.svp" file.**

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

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

The detail about AXI4 Lite protocol please refer to appendix.

Reference Waveform and Note



DRAM.dat file example

```
@10000
05 2A FA F8
@10004
0A 4D 6B 2F
@10008
11 19 52 78
@1000C
0B 2D 58 C2
@10010
0B 15 7C 35
@10014
09 71 5C BB
@10018
1B 49 3C C0
@1001C
02 03 7B 4D
@10020
01 FB 84 0E
@10024
03 42 48 0A
```

8 9 A B

@10008

11 19 52 78

Data Date (Day) Index D Index C

C D E F

@1000C

0B 2D 58 C2

Data Date (Month) Index B Index A

Category	Amount (Hex.)
Index A	C25
Index B	82D
Index C	785
Index D	219
Data Date (Month)	0B
Data Date (Date)	11

5

DRAM note

- You may modify the following part in ../00_TESTBED/pseudo_DRAM.sv.

DRAM latency →

```
parameter DRAM_R_latency = 1;
parameter DRAM_W_latency = 1;
parameter DRAM_B_latency = 1;
```

(Will be adjusted to 1~100 during DEMO)

- If you want to initialize dram in pattern, you may use the following code.

Declaration of dram reg array →

```
parameter DRAM_p_r = "../00_TESTBED/DRAM/dram.dat"

logic [7:0] golden_DRAM[ ((65536+256*8)-1) : (65536+0)] ;

initial $readmemh(DRAM_p_r, golden_DRAM);
```

7

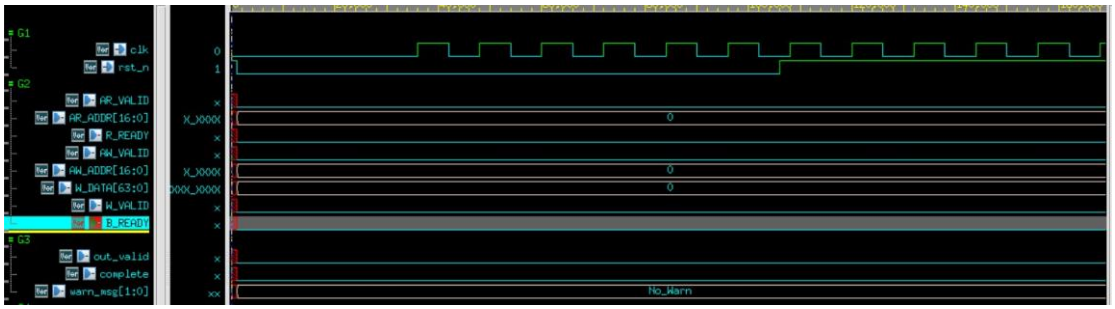
You need to generate dram.dat yourself and the DRAM address range starts from @10000 and ends at @107FF.

```
@10000
05 2A FA F8
@10004
0A 4D 6B 2F
```

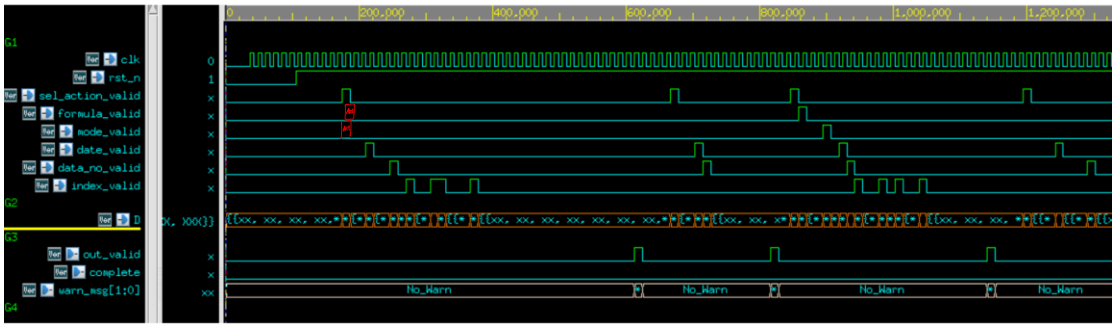
to

```
@107F8
1E 27 28 BE
@107FC
06 A2 06 A3
```

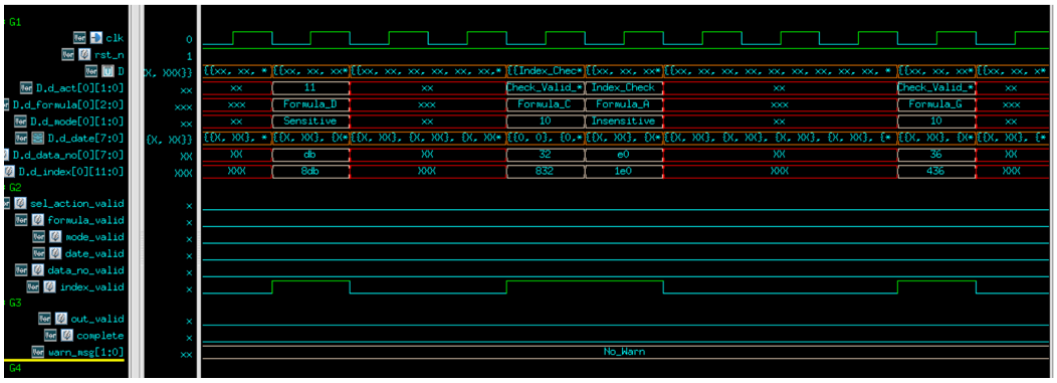
All output signals should reset to 0 after rst_n getting low.



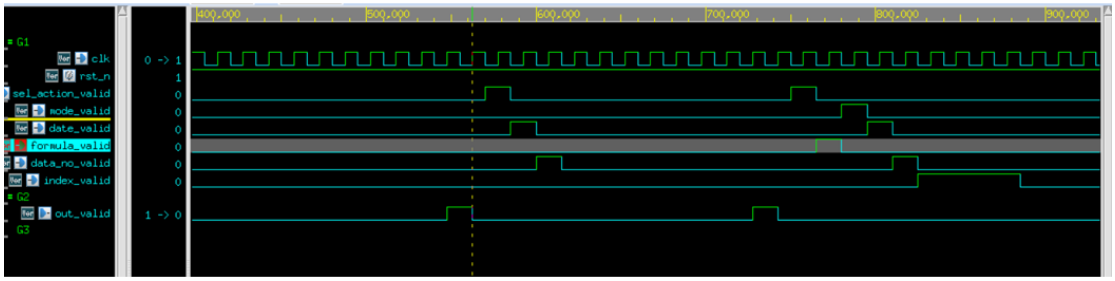
The next input valid signal among 6 input valids will become valid 1 to 4 cycles after a valid in 6 input valids or out_valid falls.



index_valid may or may not be continual.



One cycle delay between all valid signals



Four cycles delay between all valid signals