CSIE DIC Lab 2024

Lab08. LCD Image Controller

Description

In this lab, you'll dive into the task of developing a circuit to control an LCD, known as the LCD_CTRL. The circuit's primary function is to manipulate an 8×8 grayscale image through a set of predefined commands that modify the display in various ways. These operations include moving the operation point horizontally or vertically within the set coordinate limits, and calculating the maximum, minimum, and average values of the image data, as illustrated in Figure 1. The image data, comprised of 64 pixels with 8 bits per pixel, will be sourced from an input read-only memory (IROM) and the processed results will be stored in an output random access memory (IRAM).

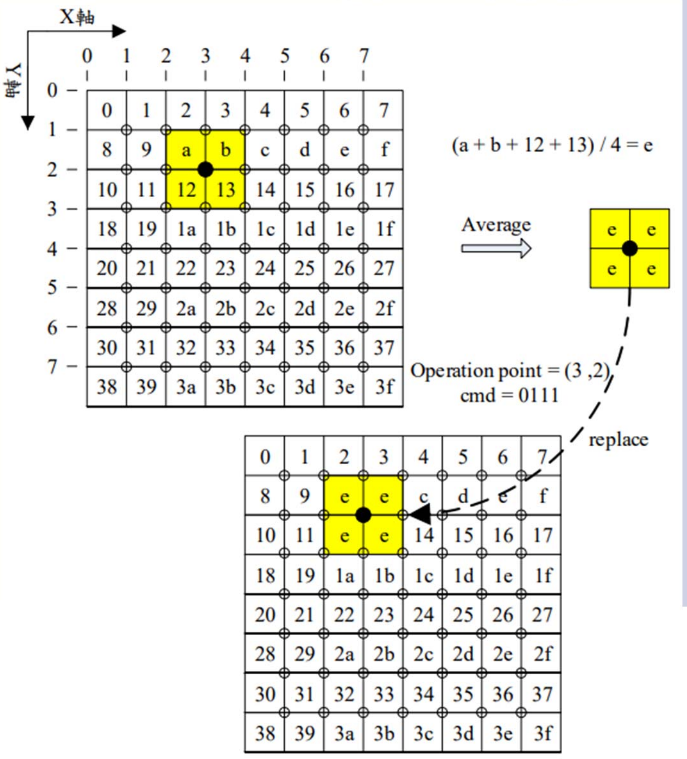


Figure 1: System function diagram.

After the circuit is reset, the LCD_CTRL must initialize by reading the image data from IROM. Throughout the operation, a busy signal indicates that the circuit is engaged in processing and is not ready to receive new commands. Once the busy signal is low, new commands can be inputted.

[Operation Point]

The term "operation point" refers to the current coordinate point in the LCD_CTRL circuit. The image pixel data in the four directions (up, down, left, right) from the operation point are the pixel data utilized by the LCD_CTRL circuit for computation, as illustrated in Figure 1. The LCD_CTRL circuit will perform calculations on these pixels based on the commands received. Simultaneously, in this lab, the coordinate axes of the input image are defined, with the horizontal direction being the X-axis and the vertical direction being the Y-axis.

As illustrated in Figure 2, the range of coordinates for both the X-axis and the Y-axis is from 0 to +8, with the operation point initially positioned at (4, 4). To ensure that the LCD_CTRL circuit does not access data beyond the input image's boundaries during computation, **the range of coordinates for the operation point on both the X-axis and the Y-axis is restricted to +1 to +7**. Consequently, any subsequent instructions for moving the operation point's coordinates must take this coordinate range limitation into consideration.

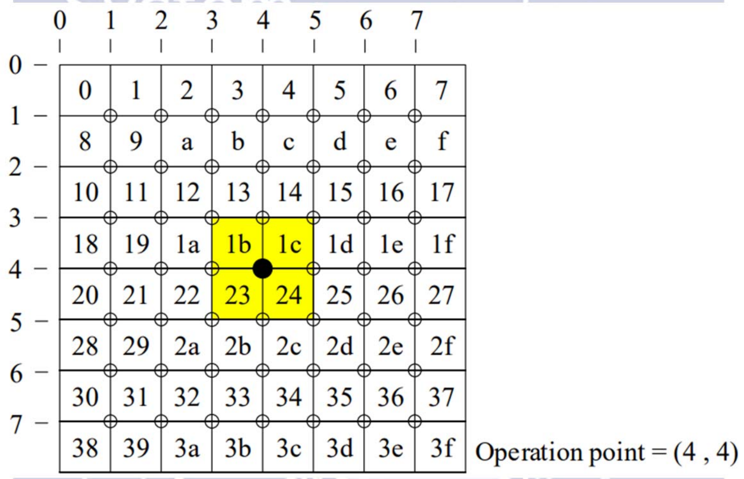


Figure 2: The range of coordinates along the X and Y axes.

Specification

1. Top module name: LCD_CTRL
2. Top module filename: LCD_CTRL.v
3. Input/output definition

Table 1: Input/output definition

Signal Name	I/O	Width(bit)	Description
clk	input	1	Clock signal.
reset	input	1	Reset signal.
cmd	input	3	Command Input Signal: The controller receives eight distinct

			command input signals. For specifics regarding these commands, please consult Table 2. A command input is deemed valid only when the "cmd_valid" signal is active (high) and the "busy" signal is inactive (low).
cmd_valid	input	1	When the signal is high, it indicates that the "cmd" command input is recognized as valid.
IROM_rd	output	1	IROM Read Enable Signal: A high state indicates that the LCD Controller requires data retrieval from the IROM.
IROM_A	output	6	IROM Address Bus: The LCD Controller uses this bus to access and request image data from the IROM at the specified address.
IROM_Q	input	8	IROM Data Bus: This bus is utilized by the IROM to send image data to the LCD Controller.
IRAM_valid	output	1	IRAM Data Valid Signal: A high state signals that both the image data being sent and the address bus utilized by the LCD Controller are valid.
IRAM_A	output	6	IRAM Address Bus: The LCD Controller uses this bus to store image data at the specified address.
IRAM_D	output	8	IRAM Data Bus: The LCD Controller employs this bus to write image data into the IRAM.
busy	output	1	System Busy Signal: A high state of this signal signifies that the controller is engaged in processing the ongoing command and is thus unable to accept new commands. Conversely, a low

			signal indicates readiness to accept new command inputs. By default, this signal should set to high during a system reset.
done	output	1	When the LCD Controller finishes writing to the IRAM, setting the "done" signal to high denotes the completion of the process.

4. **[Command definition]**:
   The specific functions associated with each input command (cmd) for LCD Controller are detailed in Table 2.

Table 2: Command definition

cmd number	Control instruction description
0(000)	Write
1(001)	Shift Up
2(010)	Shift Down
3(011)	Shift Left
4(100)	Shift Right
5(101)	Max
6(110)	Min
7(111)	Average

- **[Write]**
  - When executing the "Write" command, the LCD Controller writes image data into the IRAM sequentially, moving from left to right and then from top to bottom.
- **[Shift Up]**
  - Move the operation point upwards. Executing the "Shift Up" command will decrease the Y-coordinate of the operation point by 1, but the Y-axis coordinate must not fall below 1.
  - When the Y-axis coordinate of the operation point is already at 1, receiving another "Shift Up" command will keep the Y-axis coordinate unchanged at 1.
- **[Shift Down]**
  - Move the operation point downwards. Executing the "Shift Down" command will increase the Y-coordinate of the operation point by 1, but the Y-axis coordinate must not exceed 7.
  - When the Y-axis coordinate of the operation point is already at 7, receiving another "Shift Down" command will keep the Y-axis coordinate unchanged

at 7.

- **[Shift Left]**
  - Move the operation point to the left. Executing the "Shift Left" command will decrement the X-coordinate of the operation point by 1, but the X-axis coordinate must not fall below 1.
  - When the X-axis coordinate of the operation point is already at 1, receiving another "Shift Left" command will keep the X-axis coordinate unchanged at 1.
- **[Shift Right]**
  - Move the operation point to the right. Executing the "Shift Right" command will increase the X-coordinate of the operation point by 1, but the X-axis coordinate must not exceed 7.
  - When the X-axis coordinate of the operation point is already at 7, receiving another "Shift Right" command will keep the X-axis coordinate unchanged at 7.
- **[Max]**

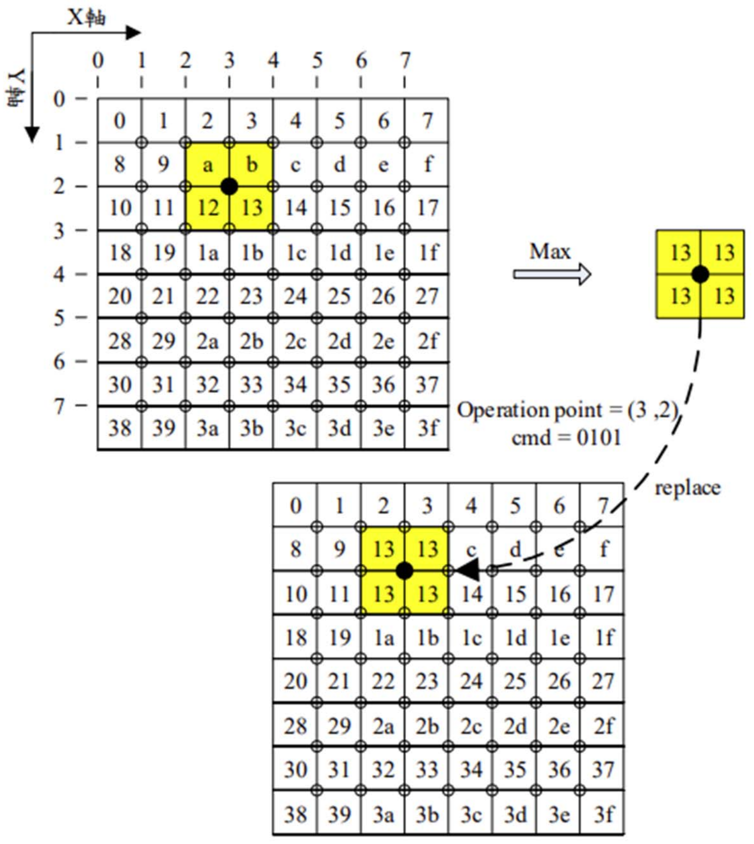


Figure 3: Maximum Value Output

- When executing the "Max" command, it compares the four pixel values above, below, left, and right of the current operation point, identifies the maximum value among them, and then modifies all four pixel values to this maximum value. Refer to the example in Figure 3, where the operation point is at (3,2). In this case, the maximum value of 13_{16} among the surrounding four pixels will replace their original values.

- **[Min]**
  - When executing the "Min" command, it compares the four pixel values above, below, left, and right of the current operation point, identifies the minimum value among them, and then modifies all four pixel values to this minimum value. Refer to the example in Figure 4, where the operation point is at (3,2). In this case, the minimum value of a_{16} among the surrounding four pixels will replace their original values.

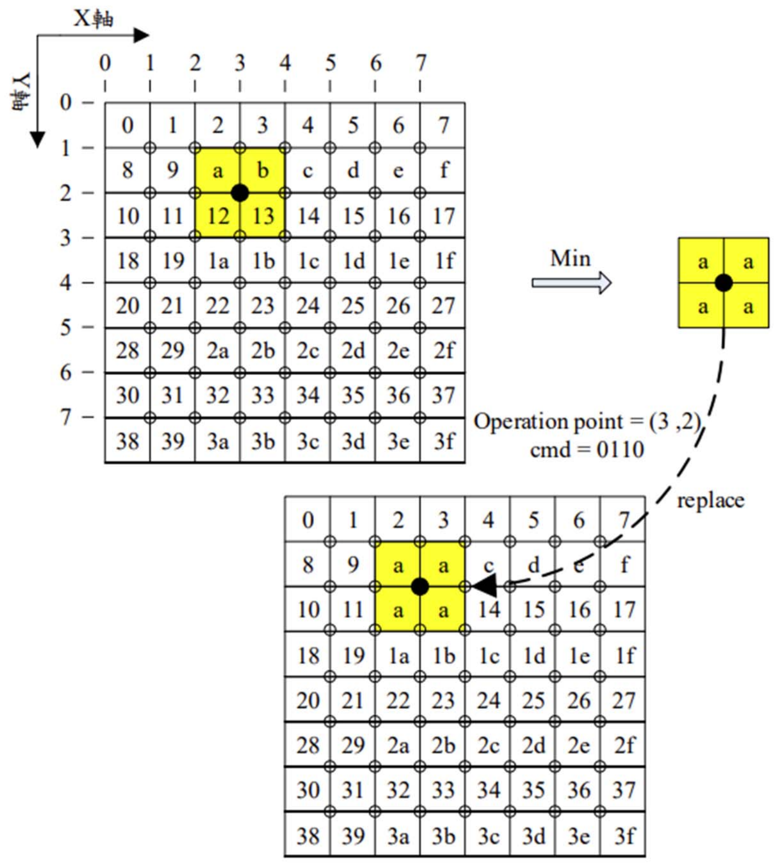


Figure 4: Minimum Value Output

- **[Average]**
  - When executing the "Average" command, the four pixel values above, below, to the left, and to the right of the current operation point are added together and then divided by 4. If the result is a decimal, the calculation will be rounded down to the nearest integer number. For example, if the calculation is $(a_{16}+b_{16}+12_{16}+13_{16}) / 4_{16} = 1110.1_2$, then the output will be $1110_2 = e_{16}$. Subsequently, these four pixel values are all modified to the average value. Refer to the example in Figure 5, where the operation point is at (3,2); at this point, the average value e_{16} will replace the original values of these

surrounding four pixels.

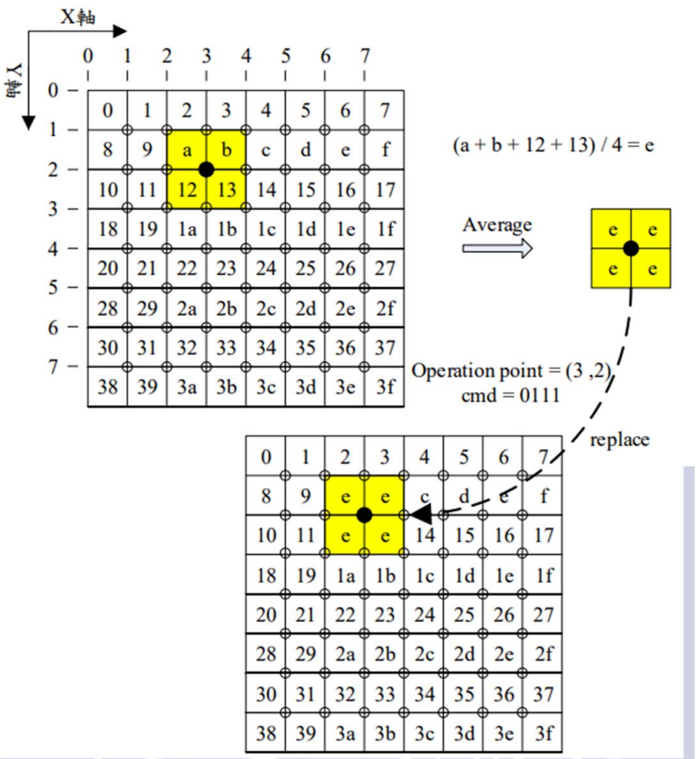


Figure 5: Average Value Output

5. **[IROM Mapping and Timing Specifications]**

   The grayscale image input is set at a resolution of 8×8 pixels, where each pixel is represented by 8 bits of data. Consequently, the grayscale image stored on the Host side consists of 64 pixels in total. The IROM is characterized by a data width of 8 bits and can accommodate 64 addresses. Each address holds 8 bits of data, which accurately encodes the grayscale data for a single pixel. This arrangement is depicted in Figure 6.

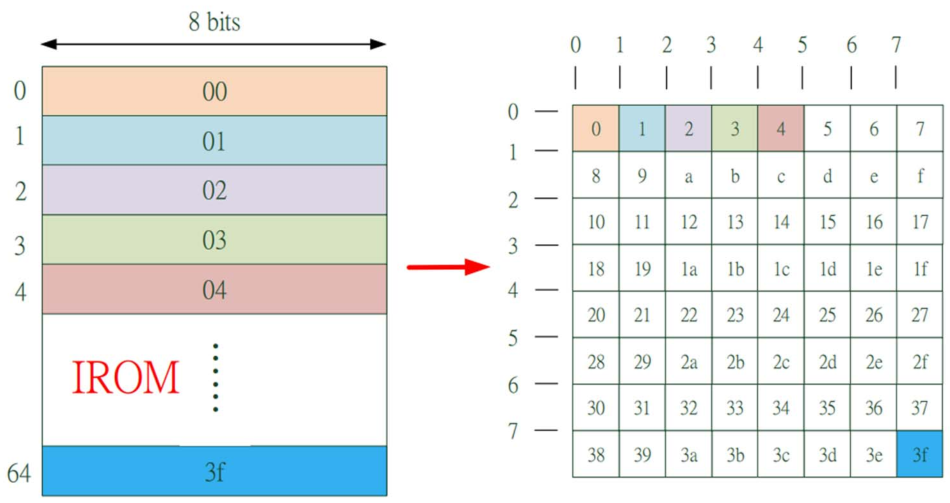


Figure 6: Input Grayscale Image Memory (IROM) Mapping

The timing sequence for the IROM operates as shown in the Figure 7. **With each falling edge of the clock signal**, if the IROM read signal (IROM_rd) is high, as seen at times T1 and T2 in Figure 7, the data located at the address indicated by the IROM Address signal (IROM_A) is instantly sent from the IROM Data Out bus (IROM_Q) to the LCD Controller. Conversely, when the IROM_rd signal is low, as seen at time T3 in Figure 7, the IROM takes no action. This memory eliminates the need to account for read delay concerns.

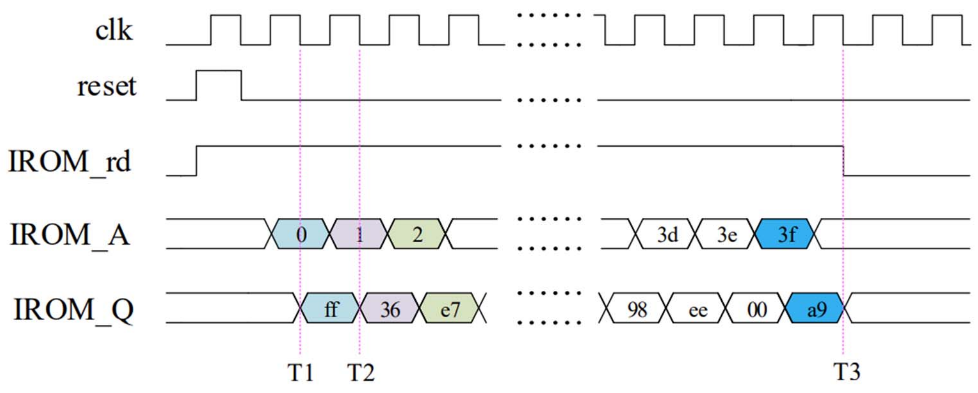


Figure 7: Input Grayscale Image Memory (IROM) Timing Diagram

6. **[Output IRAM Mapping and Timing Specifications]**
   The grayscale image output computed by the LCD Controller is formatted as 8×8 pixels, with each pixel encapsulating 8 bits of data. As a result, the Host's output IRAM is equipped with 64 addresses, allocated to store the processed results for each individual pixel. This specific mapping of the data is depicted in Figure 8.

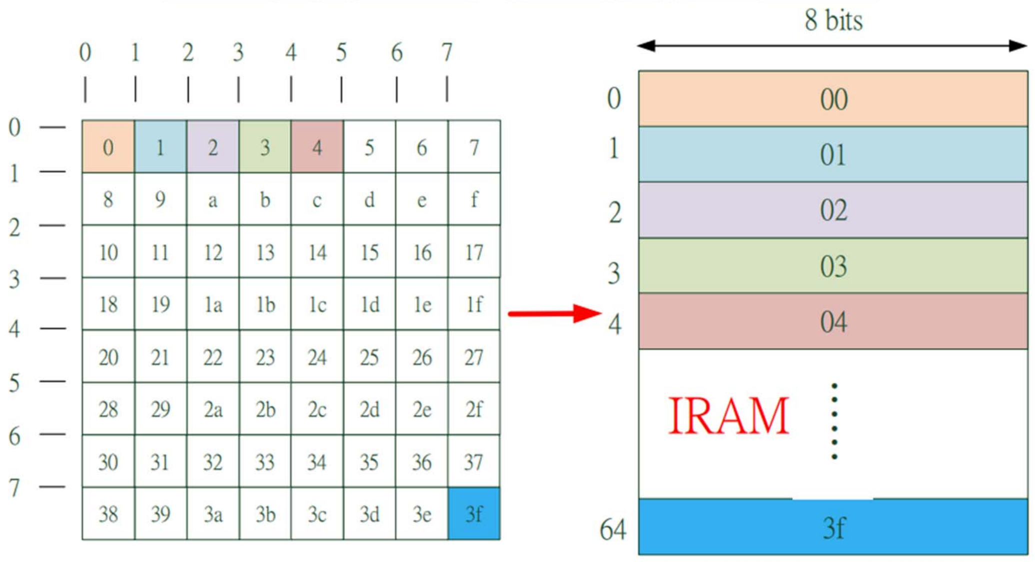


Figure 8: Output Grayscale Image Memory (IRAM) Mapping

The timing sequence for the IRAM, responsible for storing the output result grayscale image, is illustrated in Figure 9. At each negative edge of the clock signal, if the IRAM valid signal (IRAM_valid) is set to high (as observed at time T5 in Figure 9), the specific address and data meant for writing are positioned on the IRAM Address bus (IRAM_A) and the IRAM Data bus (IRAM_D), respectively. **The writing process is executed when the Host's clock signal negative edge occurs, marked at time T5.** For the purpose of writing data continuously, the IRAM_valid signal should remain high while the IRAM_A and IRAM_D are updated as necessary. To conclude the data writing process, IRAM_valid is turned low at time T6. This memory eliminates the need to account for write delay concerns.

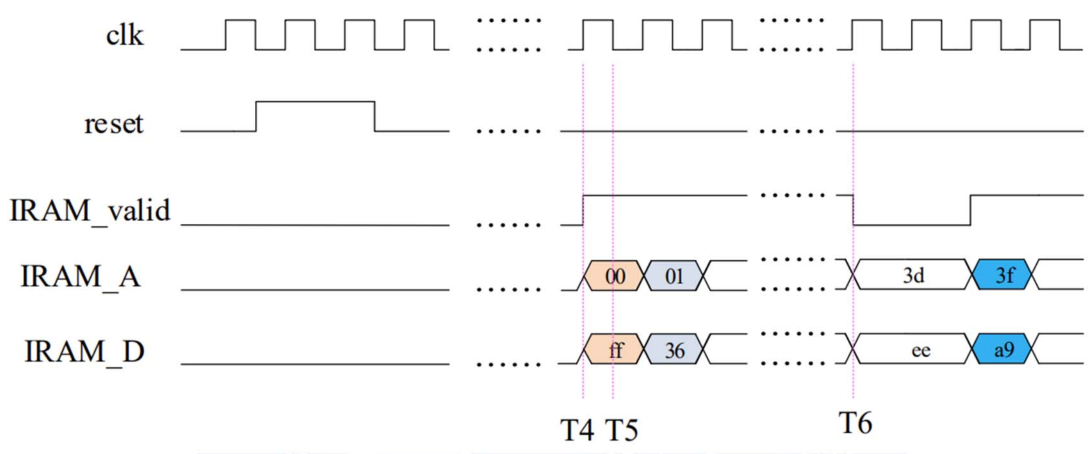


Figure 9: Output Grayscale Image Memory (IRAM) Timing Diagram

7. **[IROM Data Reading Timing Specifications]**
   After the circuit is reset, the LCD Controller will sequentially read 64 pieces of image data from the IROM, as shown in Figure 10. When IROM_rd is high, an address signal can be input to read image data from the IROM. Simultaneously, during the reading of IROM data, the busy signal must be maintained at a high level. After the completion of IROM data reading, busy is set back to a low level to accept new command inputs.

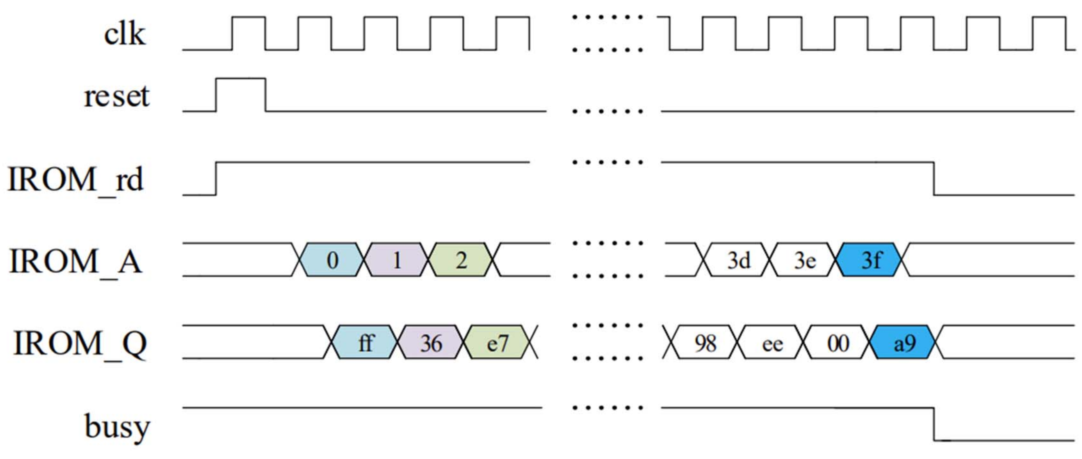


Figure 10: IROM Data Reading Timing Diagram

8. **[Command Input Timing Specifications]**

   Upon receiving a new command (for example: shift up, shift down, shift left, shift right, max, min, average), the timing for inputting control commands is as illustrated in Figure 11. After a new command is received, during the process of handling the command, the busy signal needs to be continuously maintained at a high level. After the command action is completed, busy is set to a low level to accept new command inputs.

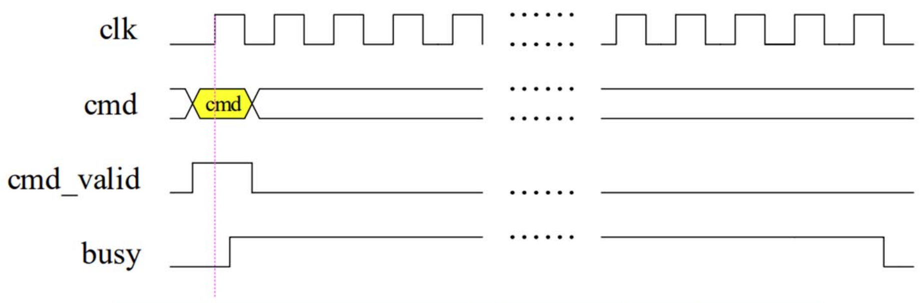


Figure 11: Input Control Commands Timing Diagram

9. **[Write Command Timing Specifications]**

   When executing a Write command, the LCD Controller begins writing the processed image data back to the IRAM, as shown in Figure 12. When IRAM_valid is high, it indicates that data is being written into the IRAM. The LCD Controller outputs the address and the image data to be written back, sequentially writing the data back into the IRAM. Simultaneously, during the processing of the Write command, the busy signal needs to be maintained at a high level. **After the completion of the Write command, done is set to a high level.** At this point, the testing module will compare the image data in the IRAM with the correct answer.

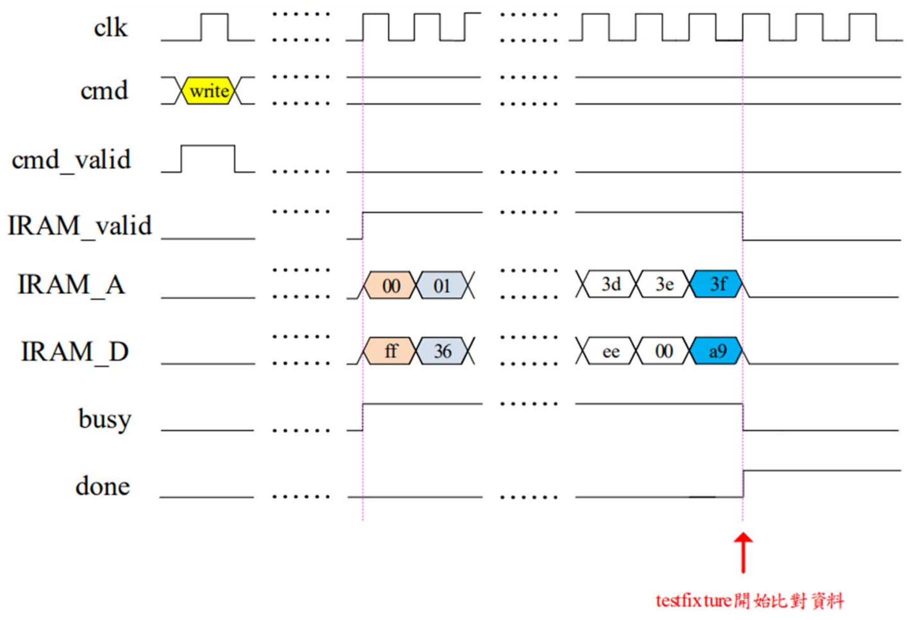
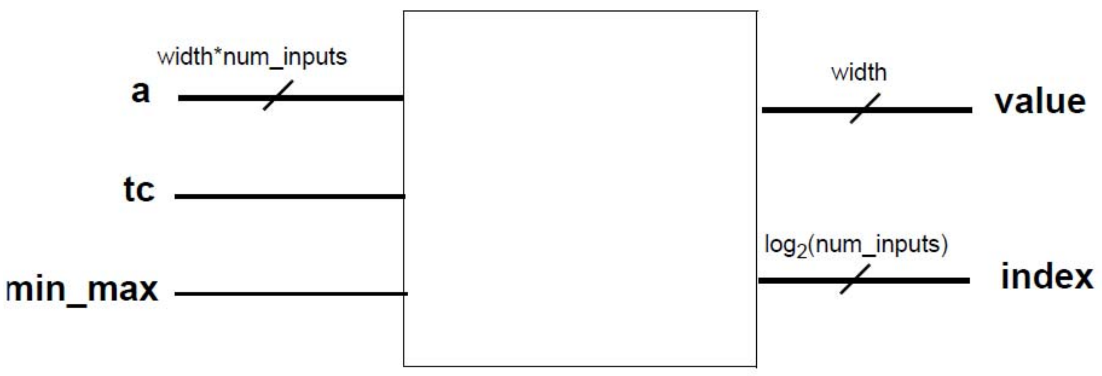


Figure 12: Write Command Timing Diagram

10. In this lab, you are required to use the **CW_minmax** provided by the Logic Synthesizer to implement the calculation of the maximum and minimum pixel values. The I/O port description of **CW_minmax** is illustrated in Figure 13.

CW_minmax



Port Description

Port Name	Type	Size (bits)	Description
a	Input	*num_inputs*width*	Integer input
tc	Input	1	0: a and value are interpreted as unsigned integers 1: a and value are interpreted as signed integers
min_max	Input	1	Minimum/maximum switch. 0: value = Minimum segment of a. 1: value = Maximum segment of a.
value	Output	*width*	Minimum/maximum of input bus
index	Output	$\log_2$(*num_inputs*)	Index of minimum/maximum value within the input bus

Figure 13: I/O ports of CW_minmax

11. You should perform low-power synthesis to optimize the power consumption of your circuit. In addition, you should provide power information of your circuit with and without low-power synthesis.
12. In logic synthesis, the timing constraint for the clock period **MUST** be **0.44ns**. Thus your design should operate correctly at this clock speed.

Data Preparation

1. Extract LAB data from TA's directory:

   % tar xzvf ~dicta/lab08.tar.gz
2. The extracted LAB directory (**lab08**) contains:
   a. **Democase/** :Design example, including RTL code and synthesis script
   b. **Exercise/** :Test pattern for this lab

Demo Case

3. Change directory to lab08:

   % cd lab08
4. Change to directory: **Democase/**

   % cd Democase
5. This directory contains:

   a. RTL/ :RTL source code of design example

   b. Synthesis/ :Synthesis script

   c. Netlist/ :Synthesized netlist and test pattern for gate-level simulation

Generate the Toggle file (Switching Activities)

6. Change to directory: **RTL/**
7. Open and edit "TEST.v", add the following system tasks to generate TCF file during Verilog RTL simulation.

```
      …………………
22    $toggle_count("TESTBED.I_STACK");
23    $toggle_count_mode(1);
      …………………
198   $toggle_count_report_flat("STACK_rtl.tcf", "TESTBED.I_STACK");
      …………………
```

   (You only need to remove comments // in these three lines)
8. Run Verilog RTL simulation and then generate the required switching activities file "**STACK_rtl.tcf**"

   % ncverilog –f run.f
9. Check the "**STACK_rtl.tcf**" file

   You can find that a lot of pin/net information in this file, the format is

   "PIN/NET NAME" : "Probability of being 1" "toggle count"

   Please find the "Probability of being 1" of the "***CLK***" PIN:______________

   Please find the "toggle count" of the "***CLK***" PIN:______________________

Synthesis without power optimization

10. Change to directory: **Synthesis/**
11. Open and read the synthesis script and design constraint file.

    % gedit SYN_ADFP_RC.tcl &

    % gedit SYN_RTL.sdc &
12. Start logic synthesis,

% ./01_run_synthesis

13. Open and read the synthesis report

    % gedit report.area &

    % gedit report.timing &

    % gedit report.power &

    What is the area of the module STACK?_______________________________

    What is the slack time of the critical path?______________________________

14. Change to the directory: **Netlist/**
15. Open and edit "**TEST_gate.v**" as following

    ```
          …………………
    23    $toggle_count("TESTBED.I_STACK");
          …………………
    201   $toggle_count_report_flat("STACK_gate.tcf", "TESTBED.I_STACK");
          …………………
    ```

    (You only need to remove comments // in these two lines)

16. Run the gate-level simulation and generate the required switching activities file "**STACK_gate.tcf**",

    % ncverilog –f run.f

17. Return to the directory: **Synthesis/**
18. Report the power consumption of this design after gate-level simulation.

    % ./02_report_power

19. Open and read the power report after gate-level simulation

    % gedit report.power.gate &

    What is the power consumption of the module STACK without power optimization?__________________

Synthesis with power optimization

20. Change to directory: **Synthesis/**
21. Clean up previous synthesized results,

    % ./03_clean_up

22. Open and read the synthesis script and design constraint file with low-power synthesis

    % gedit SYN_ADFP_RC_LP.tcl &

    % gedit SYN_RTL.sdc &

23. Start low power synthesis,

    % ./04_run_low_power_syn

24. Open and read the synthesis report

    % gedit report.area &

    % gedit report.timing &

% gedit report.power &

What is the area of the module STACK?______________________________

What is the slack time of the critical path?______________________________

25. Change to the directory: **Netlist/**
26. Run the gate-level simulation and generate the required switching activities file "**STACK_gate.tcf**",

    % ncverilog –f run.f
27. Return to the directory: **Synthesis/**
28. Report the power consumption of this design after gate-level simulation.

    % ./05_report_power
29. Open and read the power report

    % gedit report.power.gate &

    What is the power consumption of the module STACK with power optimization?

    ____
30. Compare the power consumption to ″**without power optimization**″. Does power decrease a lot? Why or why not? (hint: open and reading the report **report.clock_gating**)

    ____

Exercise

31. Change directory to lab08:

    % cd lab08
32. Change to the directory: **Exercise/**

    % cd Exercise
33. This directory contains:

a.	tb3_goal.dat	: Input data
b.	cmd3.dat	: Input data
c.	image3.dat	: Input data
d.	TEST.v	: Design test bench
e.	TEST_gate.v	: Design test bench for Gate-level simulation
f.	LCD_CTRL.v	: Empty design module
g.	lab08.rc	: nWave saved signals file
h.	SYN_RTL.sdc	: Design constraint file for logic synthesis

RTL Design

34. Write your RTL code in synthesizable coding style:

    % gedit LCD_CTRL.v &

    or % gvim LCD_CTRL.v &

or % joe LCD_CTRL.v

or % vim LCD_CTRL.v

35. Run RTL simulation:

    % ncverilog –f run.f

36. Open simulation waveform:

    % nWave &

    Choose File → Open, then select **"LCD_CTRL.fsdb"** and press **OK**

37. Restore signal record for debugging:

    Choose File → Restore Signal, then select **"LCD_CTRL.rc"** and press **OK**

38. Meaning of signals in TEST group:

    err[31:0] : accumulated errors for your answer

Synthesis without power optimization

39. Repeat step 10 to step 19 in Synthesis without power optimization described in the Democase. Also, you need to prepare all of your files required for logic synthesis. We only provide the design constraint file (**SYN_RTL.sdc**) for logic synthesis and the test bench (**TEST_gate.v**) for gate-level simulation.

40. Open and read the power report after gate-level simulation

    % gedit report.power.gate &

    What is the power consumption of the module LCD_CTRL without power optimization?________________

Synthesis with power optimization

41. Repeat step 20 to step 30 in Synthesis with power optimization described in Democase. In addition, you need to prepare all of your files required for logic synthesis. We only provide the design constraint file (**SYN_RTL.sdc**) for logic synthesis and the test bench (**TEST_gate.v**) for gate-level simulation.

42. Open and read the power report after gate-level simulation

    % gedit report.power.gate &

    What is the power consumption of the module LCD_CTRL with power optimization?________________

43. Compare the power consumption to "**without power optimization**". Does power decrease a lot? Why or why not? (hint: open and reading the report **report.clock_gating**)

    ____