NCTU-EE ICLAB – Autumn 2024

Lab05 Exercise:

Design: Template Matching with Image Processing (TMIP)

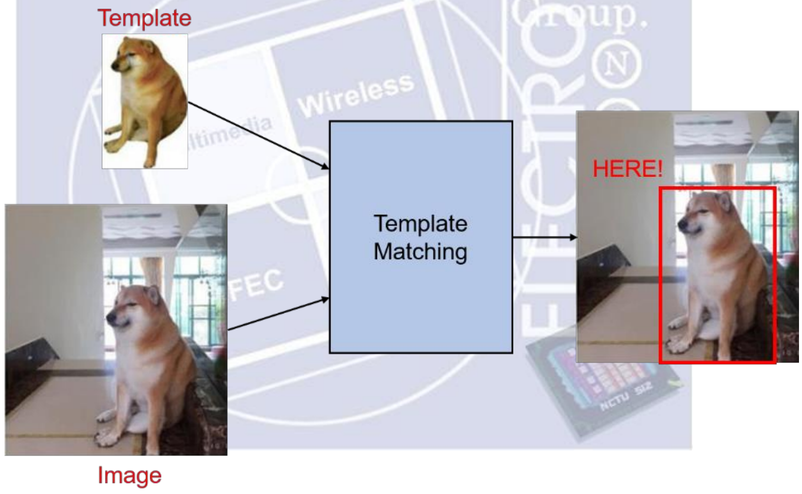
Data Preparation

- Extract the test data from TA’s directory:
  % tar xvf ~iclabTA01/Lab05_2024A.tar

- The extracted LAB directory contains:
  - Practice/
  - Exercise/

Design Description

Template Matching can be seen as a very basic form of object detection. It is a high-level machine vision technique that identifies the parts on an image that match a predefined template, which means we can detect objects in an input image using a “template” containing the object we want to detect. For example,



There are lots of methods of calculating image similarity for finding the matching parts, such as “Sum of Square Differences”, “Cross Correlation”, “Cross Coefficient”, etc. In this exercise, you are asked to use the “Cross Correlation” method, which is a dot product by taking every pair of pixels and multiply, then sum all products. Its formula is shown as below.

$$\boldsymbol{R(x,y) = \sum_{x',y'} (Template(x',y') \times Image(x+x',y+y'))}$$

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it.

In this exercise, you need to build a design to perform template matching by using correlation method and doing some image processing. For the matrix requires large space to store, you are asked to use memory for finishing this lab.

- Image & Template

  Image_R (256×8)
  Image_G (256×8)
  Image_B (256×8)

  - Size (Row × Col)

    The **image size** will be 4×4, 8×8 and 16×16 according to the given size value which will be given at the beginning of each pattern. The **template size** will only be 3×3.

  - Element

    The elements of an **image** are represented in 24-bit RGB format. The elements of a **template** are 8-bit unsigned integer.

    In order to reduce the number of IO ports, the R, G and B value of one element will be given sequentially in 3 cycles.

- Action

  There are 8 possible operations will be given:

  - Grayscale Transformation (Maximum method)

    Find the maximum of the RGB components directly. We will use the function shown as below.

$$\mathbf{grayscale}(\mathbf{x},\mathbf{y}) = \mathbf{max}\{\mathbf{red}(\mathbf{x},\mathbf{y}),\mathbf{green}(\mathbf{x},\mathbf{y}),\mathbf{blue}(\mathbf{x},\mathbf{y})\}$$

  - Grayscale Transformation (Average method)

    Calculate the average of the RGB components directly. We will use the function shown as below. In this operation, the result might be in float format, so we will transform it into integer with round-down approach. 操除以3

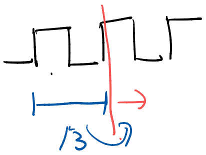
$$\mathbf{grayscale}(\mathbf{x},\mathbf{y}) = \frac{\{\mathbf{red}(\mathbf{x},\mathbf{y}) + \mathbf{green}(\mathbf{x},\mathbf{y}) + \mathbf{blue}(\mathbf{x},\mathbf{y})\}}{\mathbf{3}}$$

  - Grayscale Transformation (Weighted method)

    Calculate the weighted average of the RGB components based on human eye sensitivity. We will use the function shown as below. In this operation, whenever division operations occur, simply transform the result into integer with round-down approach.

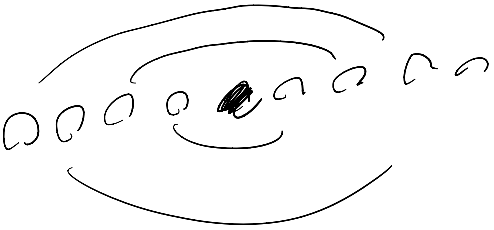
$$\mathbf{grayscale}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{red}(\mathbf{x},\mathbf{y})}{\mathbf{4}} + \frac{\mathbf{green}(\mathbf{x},\mathbf{y})}{\mathbf{2}} + \frac{\mathbf{blue}(\mathbf{x},\mathbf{y})}{\mathbf{4}}$$

2 3 7
V



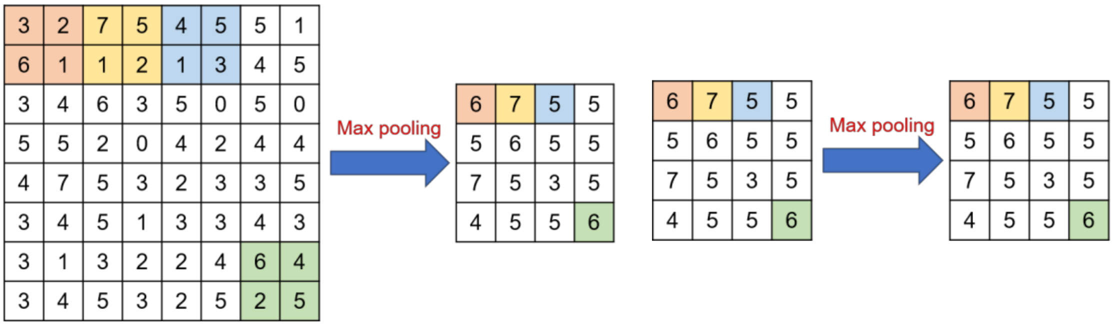
R1 G1 B1 - - - -

256×8



- Max Pooling

3 The 2×2 filter will slide through the data matrix, and the maximum pixels will be picked out and the rest will be discarded. The stride size will be 2 in max pooling operation. This operation will be performed only when the image size is 8×8 or 16×16. If the image size is 4×4, the result of max pooling will be the same as the original matrix.



3	2	7	5	4	5	5	1
6	1	1	2	1	3	4	5
3	4	6	3	5	0	5	0
5	5	2	0	4	2	4	4
4	7	5	3	2	3	3	5
3	4	5	1	3	3	4	3
3	1	3	2	2	4	6	4
3	4	5	3	2	5	2	5

6	7	5	5
5	6	5	5
7	5	3	5
4	5	5	6

6	7	5	5
5	6	5	5
7	5	3	5
4	5	5	6

6	7	5	5
5	6	5	5
7	5	3	5
4	5	5	6

X
- Negative

4 In a negative film, light areas appear dark and dark areas appear light, which is the opposite of the image as seen in the real world. We will use the function shown as below.

$$\mathbf{negative(x,y) = 255 - grayscale(x,y)}$$

(bit reversal)

↳不用減法

111
101
010

X
- Horizontal Flip

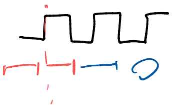
5 Mirror the image in the horizontal direction, which means each pixel will exchange its value with the corresponding horizontal column.

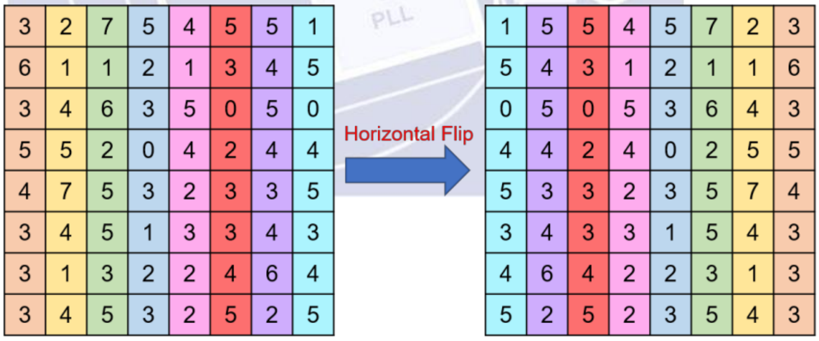
12345678
123

87654321
123

12345678
321

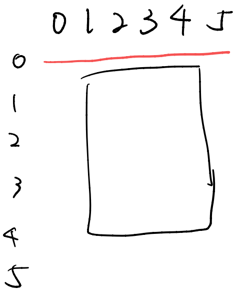
flip kernel and flip output





3	2	7	5	4	5	5	1
6	1	1	2	1	3	4	5
3	4	6	3	5	0	5	0
5	5	2	0	4	2	4	4
4	7	5	3	2	3	3	5
3	4	5	1	3	3	4	3
3	1	3	2	2	4	6	4
3	4	5	3	2	5	2	5

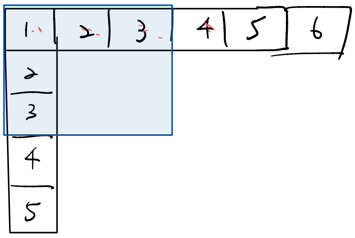
1	5	5	4	5	7	2	3
5	4	3	1	2	1	1	6
0	5	0	5	3	6	4	3
4	4	2	4	0	2	5	5
5	3	3	2	3	5	7	4
3	4	3	3	1	5	4	3
4	6	4	2	2	3	1	3
5	2	5	2	3	5	4	3



- Image Filter

轉 template

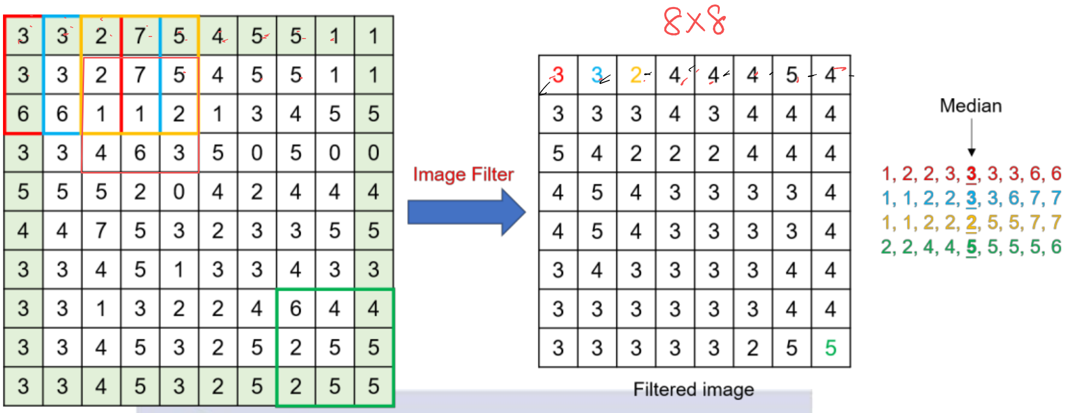
6 We usually do noise cancelling or enhance the edges of specific objects on images with image filter. In this operation, we will use median filter with 3×3 mask to implement this function. The steps for medium filter are shown as below.



- ■ Step 1: Apply replication padding on the image.
- ■ Step 2: Collect the element values within the 3×3 window.

還要排序,幹

- Step 3: Find the median value in the list of collected element values.
- Step 4: Replace the original center element value with the median value.
- Step 5: Repeat step 2~5 to slide through the image with stride of 1.



3	3	2	7	5	4	5	5	1	1
3	3	2	7	5	4	5	5	1	1
6	6	1	1	2	1	3	4	5	5
3	3	4	6	3	5	0	5	0	0
5	5	5	2	0	4	2	4	4	4
4	4	7	5	3	2	3	3	5	5
3	3	4	5	1	3	3	4	3	3
3	3	1	3	2	2	4	6	4	4
3	3	4	5	3	2	5	2	5	5
3	3	4	5	3	2	5	2	5	5

Image with replication padding

3	3	2	4	4	4	5	4
3	3	3	4	3	4	4	4
5	4	2	2	2	4	4	4
4	5	4	3	3	3	3	4
4	5	4	3	3	3	3	4
3	4	3	3	3	3	4	4
3	3	3	3	3	3	4	4
3	3	3	3	3	2	5	5

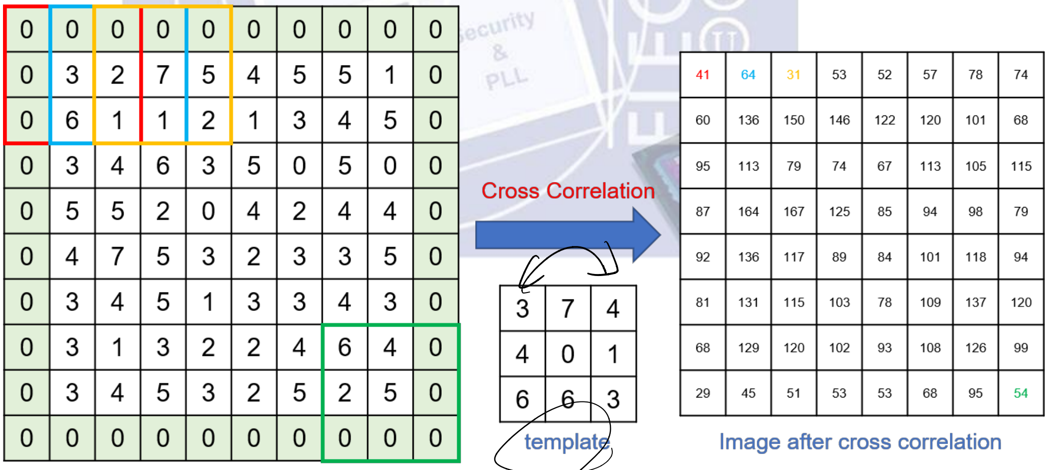
Filtered image

1, 2, 2, 3, **3**, 3, 3, 6, 6
1, 1, 2, 2, **3**, 3, 6, 7, 7
1, 1, 2, 2, **2**, 5, 5, 7, 7
2, 2, 4, 4, **5**, 5, 5, 5, 6

- Cross correlation

This operation is the main goal of our design to do template matching. The steps are shown as below.

- Step 1: Apply zero padding on the image.
- Step 2: Do convolution (stride of 1) on the image with the 3×3 template.



0	0	0	0	0	0	0	0	0	0
0	3	2	7	5	4	5	5	1	0
0	6	1	1	2	1	3	4	5	0
0	3	4	6	3	5	0	5	0	0
0	5	5	2	0	4	2	4	4	0
0	4	7	5	3	2	3	3	5	0
0	3	4	5	1	3	3	4	3	0
0	3	1	3	2	2	4	6	4	0
0	3	4	5	3	2	5	2	5	0
0	0	0	0	0	0	0	0	0	0

Image with zero padding

3	7	4
4	0	1
6	6	3

template

41	64	31	53	52	57	78	74
60	136	150	146	122	120	101	68
95	113	79	74	67	113	105	115
87	164	167	125	85	94	98	79
92	136	117	89	84	101	118	94
81	131	115	103	78	109	137	120
68	129	120	102	93	108	126	99
29	45	51	53	53	68	95	54

Image after cross correlation

- Rules of input data

In this exercise, you will get the image, image size and template at the beginning for each pattern. After that, you will get several numbers which range in 0 to 7 as a set of actions. The following are some rules about the actions:

- ➢ The number of actions ranges from 2 to 8 for each set.
- ➢ The first action is the grayscale transformation, which represents as 3’b000, 3’b001 or 3’b010.
- ➢ The last action is the cross correlation, which represents as 3’b111.

→只會出現一次grayscale

- ➢ Only one of the actions 0, 1, or 2 will appear once and it must be the first action in each set.
- ➢ The action 7 will only appear once and must be the last action in each set

After 8 sets of action, you will get the next pattern.

- Rules of output result

When finishing a set of actions, you need to output the result matrix in raster scan order with serial out format from the MSB.

- The flow for input/output

Flow	Pattern 1	Set1		Set2			Set 8		Pattern 2	Set1		
Behavior	Input	Input	Output	Input	Output	●●●	Input	Output	Input	Input	Output	●●●
Objects	image template image_size	action	result matrix (serial out)	action	result matrix (serial out)		action	result matrix (serial out)	image template image_size	action	result matrix (serial out)	

First action	0~6 actions	Last action
0~2	3~6	7

↓ 有可能會出現也有可能不會 (小心) maxpooling!]

- Example

At the beginning, you will obtain the image consisting of 8-bit values for red, green and blue channel, image size and template.

↘ 有可能: 16x16、8x8、4x4

image_size = 1 (8×8)

6	2	5	7	7	6	1	1
4	2	0	4	1	3	4	1
0	8	7	6	8	1	9	1
9	9	2	2	6	0	0	7
2	9	0	6	0	7	9	0
7	2	0	0	2	3	3	5
4	0	5	9	4	2	1	7
4	4	8	2	1	6	5	4

Red channel

0	5	8	4	5	9	8	1
8	2	3	1	0	4	3	9
5	3	7	2	2	0	6	0
4	6	0	0	5	5	6	5
8	9	9	1	2	5	1	7
0	4	9	1	7	6	8	2
3	3	4	1	0	9	9	3
3	3	3	5	3	8	1	9

Green channel

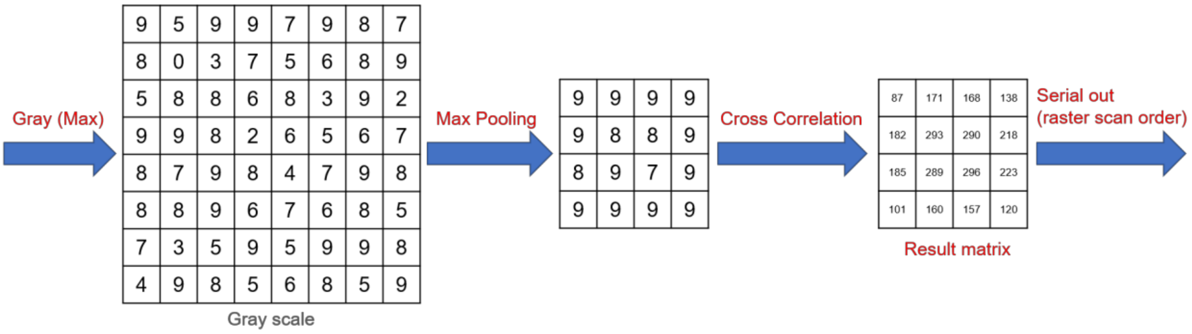
9	2	9	9	4	0	6	7
4	0	2	7	5	6	8	5
5	6	8	4	8	3	1	2
6	1	8	2	6	1	6	5
1	7	5	8	4	3	4	8
8	8	7	6	0	0	1	5
7	1	3	3	5	1	9	8
4	9	8	4	6	2	5	0

Blue channel

3	7	4
4	0	1
6	6	3

Template

Then you will get the action numbers in sequence. In this case, the sequence of the action number is 0 → 3 → 7, which means doing “Grayscale conversion (Maximum method)”, “Max pooling” and “Cross Correlation”.



9	5	9	9	7	9	8	7
8	0	3	7	5	6	8	9
5	8	8	6	8	3	9	2
9	9	8	2	6	5	6	7
8	7	9	8	4	7	9	8
8	8	9	6	7	6	8	5
7	3	5	9	5	9	9	8
4	9	8	5	6	8	5	9

9	9	9	9
9	8	8	9
8	9	7	9
9	9	9	9

87	171	168	138
182	293	290	218
185	289	296	223
101	160	157	120

I/O Description

一開始、

Input	Bit Width	Definition and Description
clk	1	Clock
rst_n	1	Asynchronous active-low reset
in_valid	1	High when **image**, **template** and **image_size** are valid.
in_valid2	1	High when **action** is valid.
image	8	Element of input **image**. It will be sent in raster scan order continuously when **in_valid** is high. The RGB data for each element in the matrix will be provided sequentially over three cycles from the 8-bit image. The RGB data for the next element will be provided in the subsequent three cycles, continuing this process until all elements have been delivered. (Example: R_1, G_1, B_1, R_2, G_2, B_2, …, R_n, G_n, B_n)
template	8	Element of input **template**. It will be sent in raster scan order continuously when in_valid is high. The elements are unsigned integers.
image_size	2	Size of the input image matrix. It will be given in the first cycle when **in_valid** is high. **2'd0**: 4×4, **2'd1**: 8×8, **2'd2**: 16×16
action	3	The signals will be given when **in_valid2** is high. The definition is as following: **3'd0**: Grayscale Transformation (Max method) **3'd1**: Grayscale Transformation (Average method) **3'd2**: Grayscale Transformation (Weighted method) **3'd3**: Max Pooling √ **3'd4**: Negative **3'd5**: Horizontal Flip √ **3'd6**: Image Filter **3'd7**: Cross Correlation √
Output	Bit Width	Definition and Description
out_valid	1	High only when **out_value** is valid. It cannot be overlapped with **in_valid** and **in_valid2**
out_value	1	It will output the final result matrix in raster scan order and serial out format from the MSB after finishing a set of action. Each **out_value** contains 20 bits for an element of the result matrix.

只有一個bit

Specification

Top module

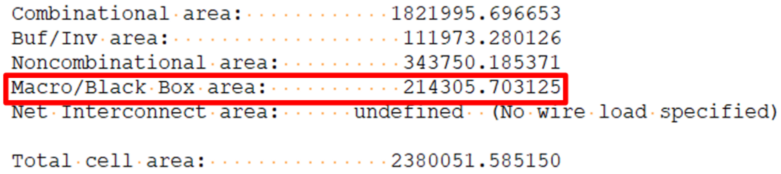
1. Top module name: **TMIP** (file name: **TMIP.v**)

2. You can adjust your clock period by yourself, but the maximum period is 20 ns. The precision of clock is 0.1, for example, 19.5 is allowed but 19.55 is not.

3. The execution latency is limited to 5000 cycles. The latency is the clock cycles between the falling edge of the last cycle of **in_valid2** and the rising edge of the first cycle of **out_valid**.

4. The total cell area should not larger than 1,500,000 μm^2.

5. In this lab, you must use the memory and generate it yourself. The number of words and the bits per each word is defined by yourself. The total number and kind of memory is unlimited. We will check it at **TMIP.area in 02_SYN/Report**. The area of **Macro/Black Box** must not be 0. The example is shown in following figure.



```
Combinational area:            1821995.696653
Buf/Inv area:                   111973.280126
Noncombinational area:          343750.185371
Macro/Black Box area:           214305.703125
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:               2380051.585150
```

6. If any port of memory is connected with mismatched width, the memory will not be synthesized and you will get an error message. Even though the design may still pass gate level simulation, this situation will be regarded as a synthesis fail. In this case, the memory area will be 0 in **TMIP.area**. We will check it at **syn.log** and **TMIP.area**.

Reset

7. 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 that should be reset after the reset signal is asserted.

8. The reset signal **(rst_n)** will be given only once at the beginning of simulation. All output signals should be reset to 0 after the reset signal is asserted.

in_valid & input signals

9. All input signals are synchronized at negative edge of the clock.

10. **in_valid** will come after reset. $R_1 G_1 B_1$ - - -

11. The input of **image** is delivered in raster scan order for 3×current size of image (4×4, 8×8, 16×16) cycles continuously when **in_valid** is tied high. When all elements of an image are delivered completely, it is tied to unknown state, and **in_valid** will also be tied low.

12. The input of **template** is delivered in raster scan order for 9 cycles continuously when **in_valid** is tied high. When all elements of a template are delivered completely, it will be tied to unknown state.

13. **image_size** will be given in the first cycle when the **in_valid** is high.

14. Every time **in_valid2** is triggered, it is tied high for several cycles depending on the numbers of actions continuously. When **in_valid2** is low, the input of **action** is tied to unknown state.

15. **in_valid2** will be triggered for a total 8 times after **in_valid** falls in a single pattern. After each time **in_valid2** triggers, your design will do template matching with image processing and then **out_valid** will be tied high for corresponding cycles.

16. In each pattern, the first **in_valid2** will come in 2~4 negative edge of clock after **in_valid** falls, and the other 7 times **in_valid2** will come in 2~4 negative edge of clock after **out_valid** falls.

out_valid & out_value

17. All output signals should be synchronized at clock positive edge.

18. The TA's pattern will capture your output for checking at clock negative edge.

19. **out_valid** cannot overlap with **in_valid** and **in_valid2** at any time.

20. **out_valid** is limited to be high only when **out_value** is valid.

21. **out_value** should be correct when **out_valid** is high. And **out_value** should be low when **out_valid** is low.

22. **out_value** must be delivered for several cycles which depends on current size of matrix continuously, and **out_valid** should be high simultaneously.

23. Each **out_value** contains 20 bits for an element of the result matrix.

24. The next input pattern will come in 2~4 negative edge of clock after the eighth **out_valid** of this pattern falls.

Synthesis

25. The input delay is set to 0.5*(clock period).

26. The output delay is set to 0.5*(clock period), and the output loading is set to 0.05**.**

27. The input delay of **clk** and **rst_n** should be zero.

28. The synthesis time should be less than 1 hour.

29. The synthesis result (syn.log) of data type cannot include any latches and error.

30. After synthesis, you can check **TMIP.area** and **TMIP.timing**. The area report is valid when the slack in the end of timing report should be non-negative and the result should be MET.

Gate-level simulation

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

Supplement

32. In this lab, you are NOT allowed to use Designware IP.

33. 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.

34. Don’t write Chinese or other language comments in the file you sent.

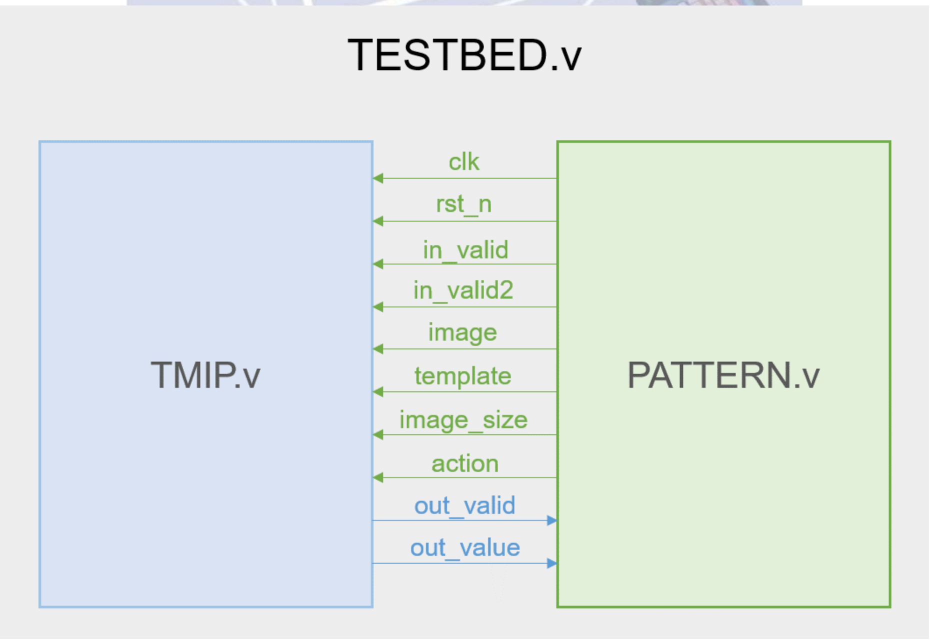
35. Verilog command //synopsys dc_script_begin, //synopsys dc_script_end //synopsys translate_off, //synopsys translate_on are only allowed during the usage of including and setting designware IPs, other design compiler optimizations are forbidden.

36. Using the above commands are allowed, however any error messages during synthesis and simulation, regardless of the result will lead to failure in this lab.

37. 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 synthesizable.

11/14 4:30

Block Diagram



Grading Policy

The performance is determined by the area and latency of your design. The less cost your design has, the higher grade you get.

- Function Validity: 70%
- Performance: 30%
  - Area^2 * Total latency
  - Total latency = cycle time * latency

Note

- Please submit your design (Bev.sv, Bridge.sv) in Lab09/EXERCISE/09_SUBMIT
  - 1st_demo deadline : 2024/10/14 (Mon.) 12:00:00
  - 2nd_demo deadline: 2024/10/16 (Wed.) 12:00:00
- Please upload the following files under 09_SUBMIT
  - **TMIP.v**
  - 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.
  - If your files violate the naming rule, you will get 10 deduction points.
- 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/TMIP.timing
    - Check the area of design in /Report/TMIP.area
  - 03_GATE/ (Gate-level simulation) **./01_run_vcs_gate**
  - 04_MEM/ (Memory location)
    - Your should generate your own memory and put the required files here

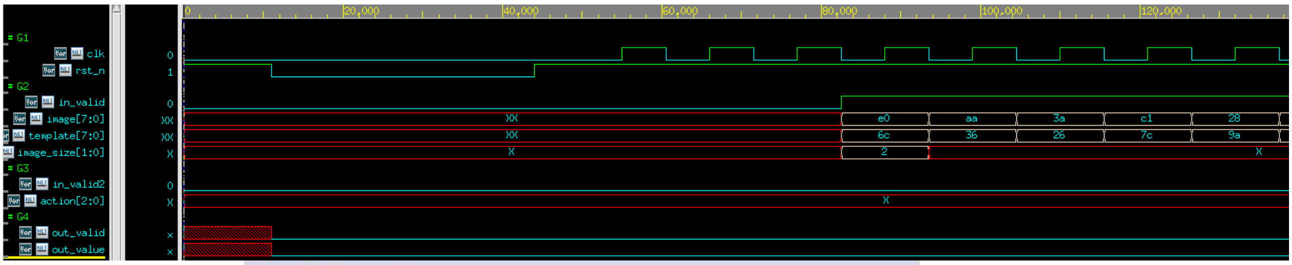
- ➢ 09_SUBMIT/ (Submit your files) **./00_tar**

  ./01_submit

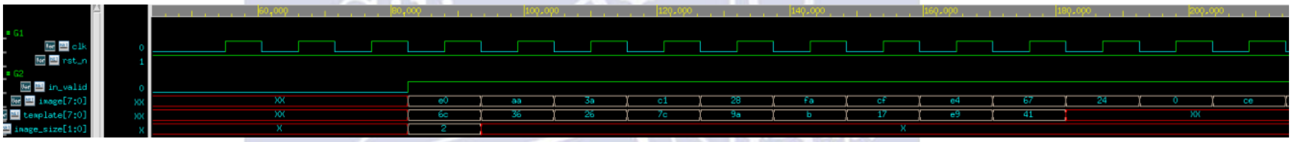
  ./02_check

Reference Waveform

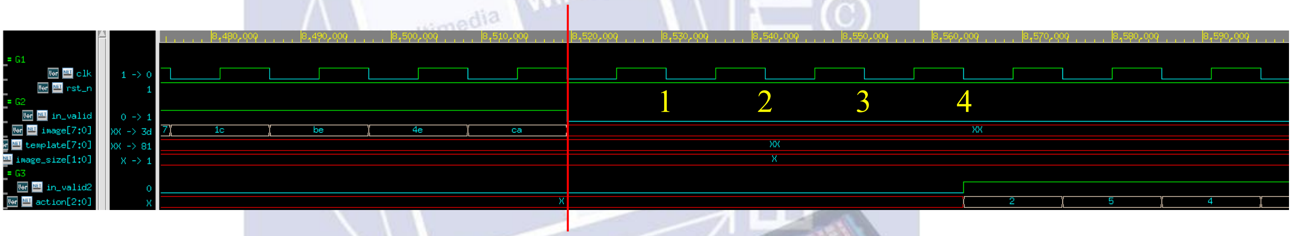
- Asynchronous reset and active-low reset all output to low



- Input of image, template and image_size

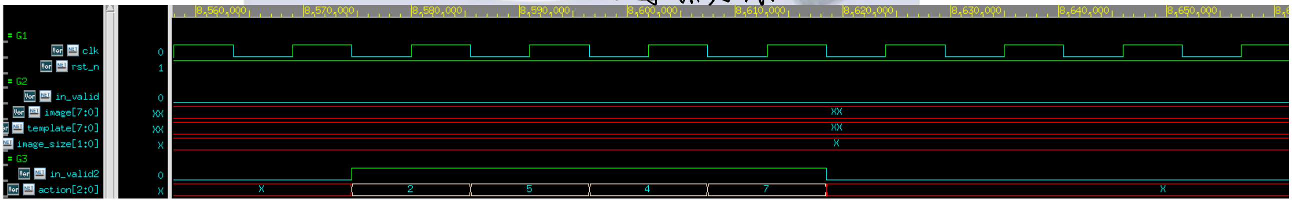


- in_valid2 will come in 2~4 negative edge of clock after in_valid is tied low

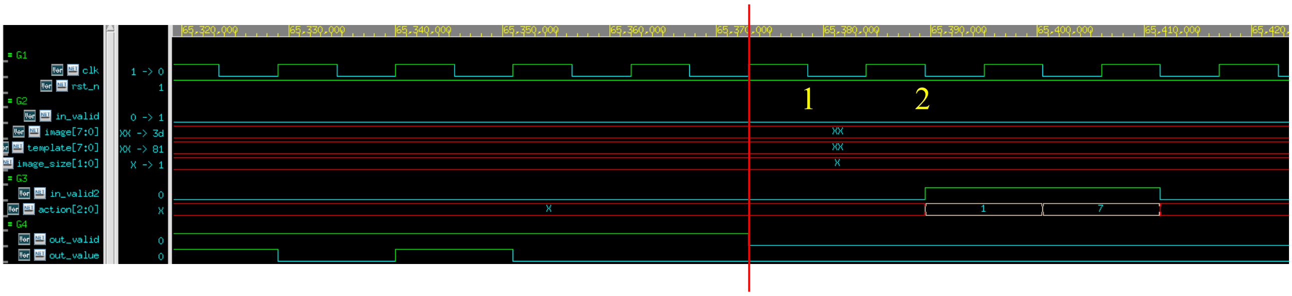


- Input of action (In this case, the number of actions is 4)

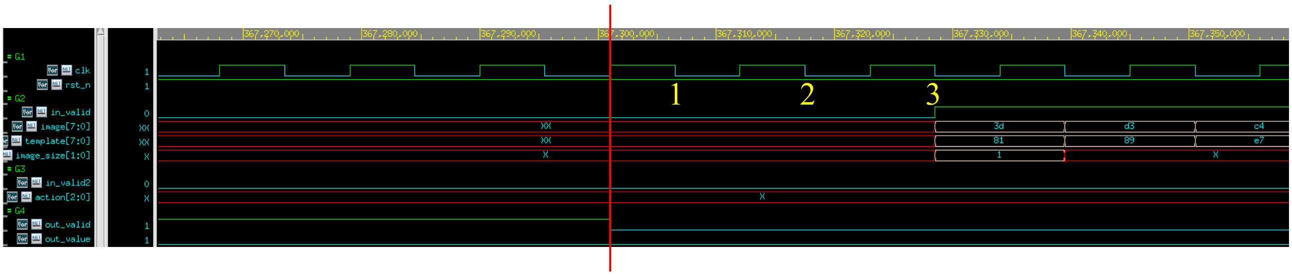
↳連續輸入 action



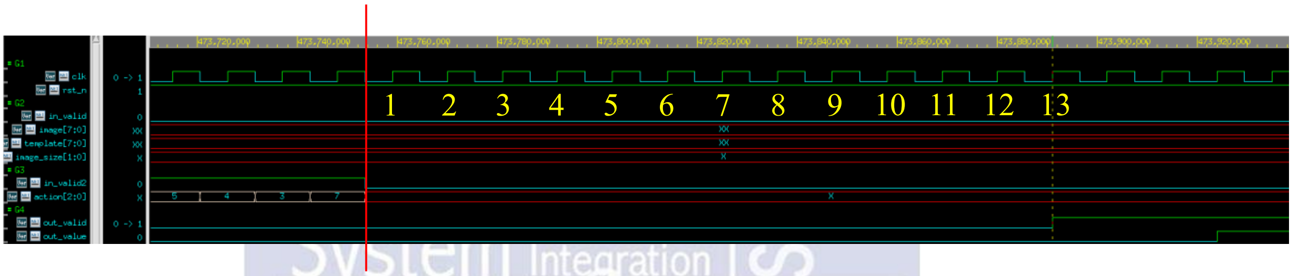
- The other 7 times in_valid2 will come in 2~4 negative edge of clock after out_valid falls



- The next input pattern will come in 2~4 negative edge of clock after the eighth out_valid of this pattern falls



- The definition of latency


G1
clk
rst_n
G2
in_valid
image[7:0]
template[7:0]
image_size[1:0]
G3
in_valid2
action[2:0]
G4
out_valid
out_value
1
2
3
4
5
6
7
8
9
10
11
12
13