NCTU-EE IC LAB – Spring 2022

Lab05 Exercise

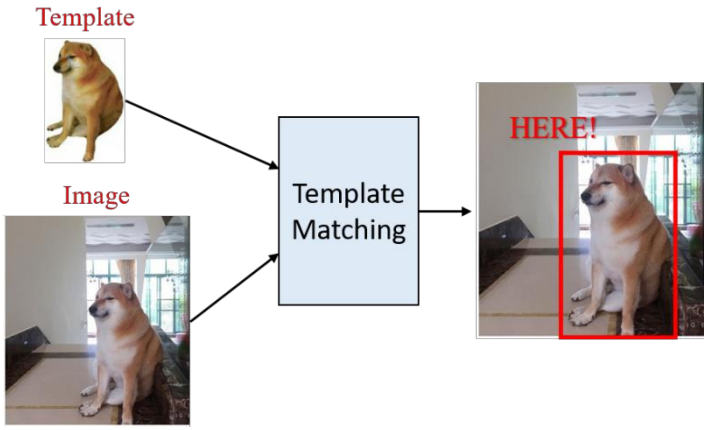
Design: Template Matching with Image Processing

Data Preparation

1. Extract files from TA's directory:
   % tar xvf ~iclabta01/Lab05.tar
2. The extracted LAB directory contains:
   a. Practice/
   b. 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're 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.

- **Formula of Cross Correlation :**

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

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 cross correlation method and doing some image processing. For the matrix requires large space to store, you are suggested to use memory for finishing this lab.

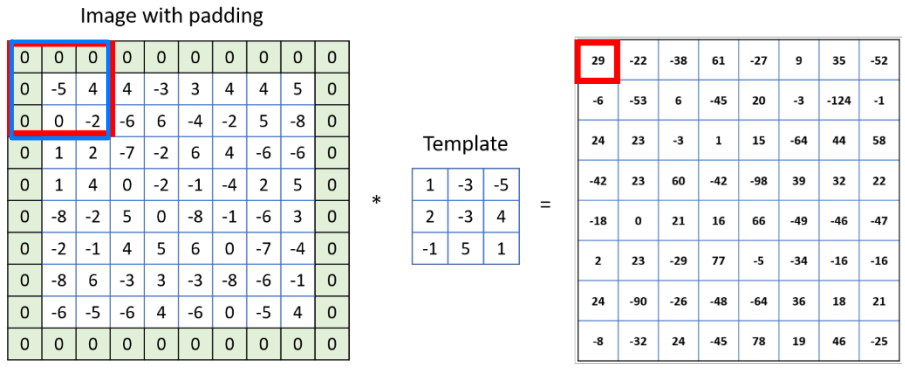
■ Size :

The **image size** will be **4x4, 8x8** and **16x16** according to the given size value which will be given at the beginning of each pattern. The **template size** will only be **3x3**.

■ There are 8 possible operations will be given :

- **Cross Correlation :**

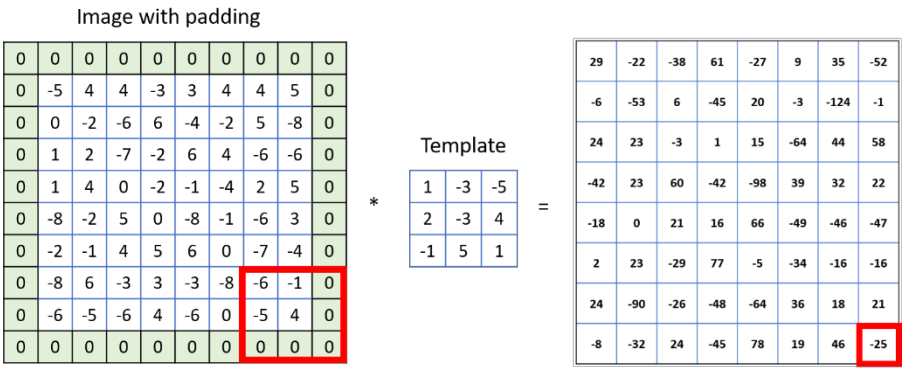
To keep the shape of the matrix, we will also apply zero padding to original data matrix X while doing the cross-correlation operation. Finally, we will get a matrix X' which is the same size as original data matrix X.



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

1	-3	-5
2	-3	4
-1	5	1

29	-22	-38	61	-27	9	35	-52
-6	-53	6	-45	20	-3	-124	-1
24	23	-3	1	15	-64	44	58
-42	23	60	-42	-98	39	32	22
-18	0	21	16	66	-49	-46	-47
2	23	-29	77	-5	-34	-16	-16
24	-90	-26	-48	-64	36	18	21
-8	-32	24	-45	78	19	46	-25



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

1	-3	-5
2	-3	4
-1	5	1

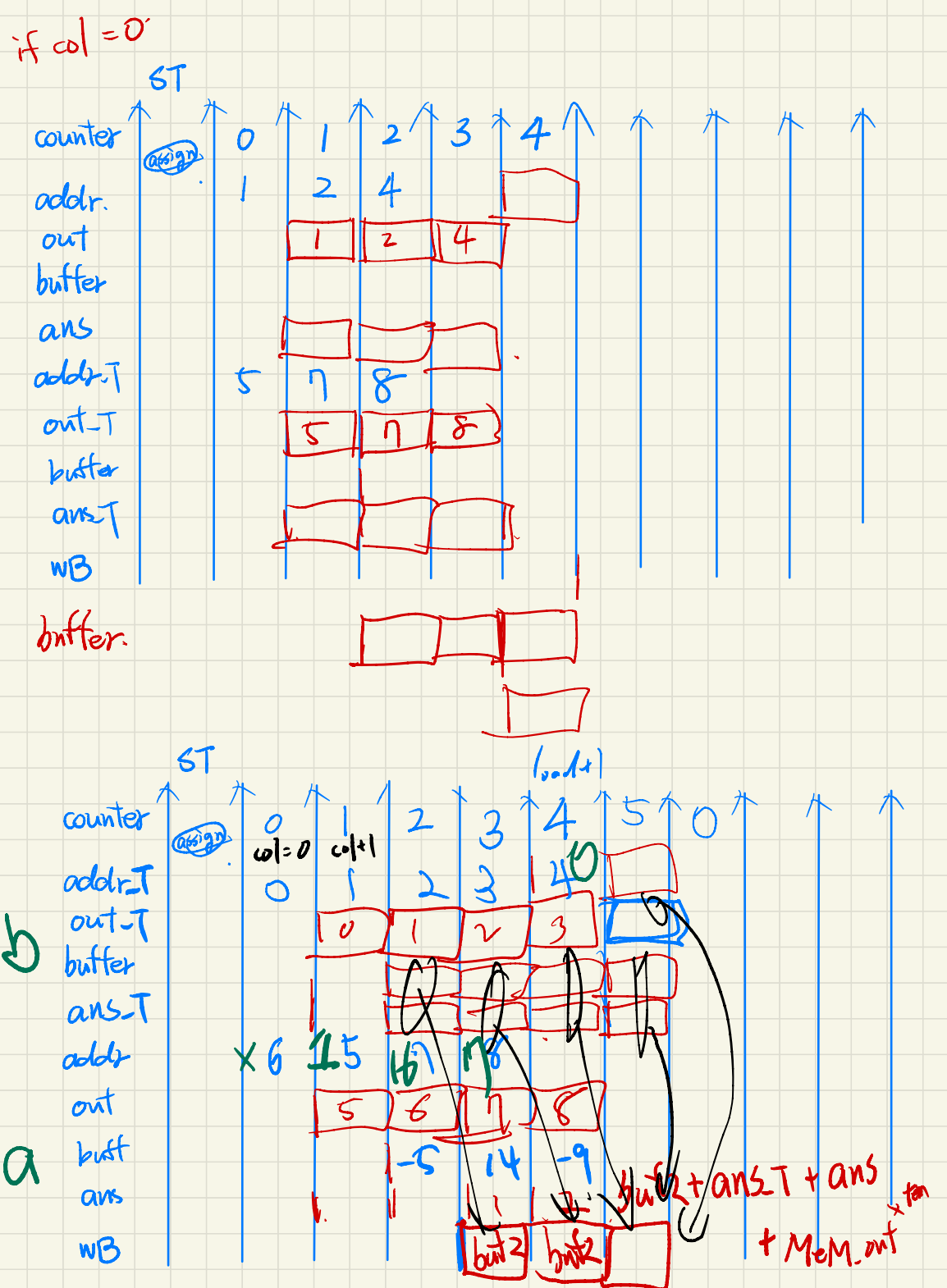
29	-22	-38	61	-27	9	35	-52
-6	-53	6	-45	20	-3	-124	-1
24	23	-3	1	15	-64	44	58
-42	23	60	-42	-98	39	32	22
-18	0	21	16	66	-49	-46	-47
2	23	-29	77	-5	-34	-16	-16
24	-90	-26	-48	-64	36	18	21
-8	-32	24	-45	78	19	46	-25

- **Max Pooling:**

The 2x2 filter will slide through the data matrix, 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 shape of X is 8x8 or 16x16)

(If the shape of X is 4x4, the result of max pooling will be still 4x4.)


if col = 0
ST
counter
assign
addr.
out
buffer
ans
addr_T
out_T
buffer
ans_T
WB
buffer.
ST
col+1
counter
addr_T
out_T
buffer
ans_T
addr
out
buff
ans
WB
col=0
col+1
b
a
buff+ans_T+ans
+MEM_out

0

0	0	0
0	16	1 5
0	17	17 9

0 0 0 0

0 -5 14 -9

10111

	8	9
23	24	25

1000

0001 0111

+ 304

0	16	17
0	32	33
0	48	49

	14	-9
	19	16
	59	37

load : 3

Image with padding

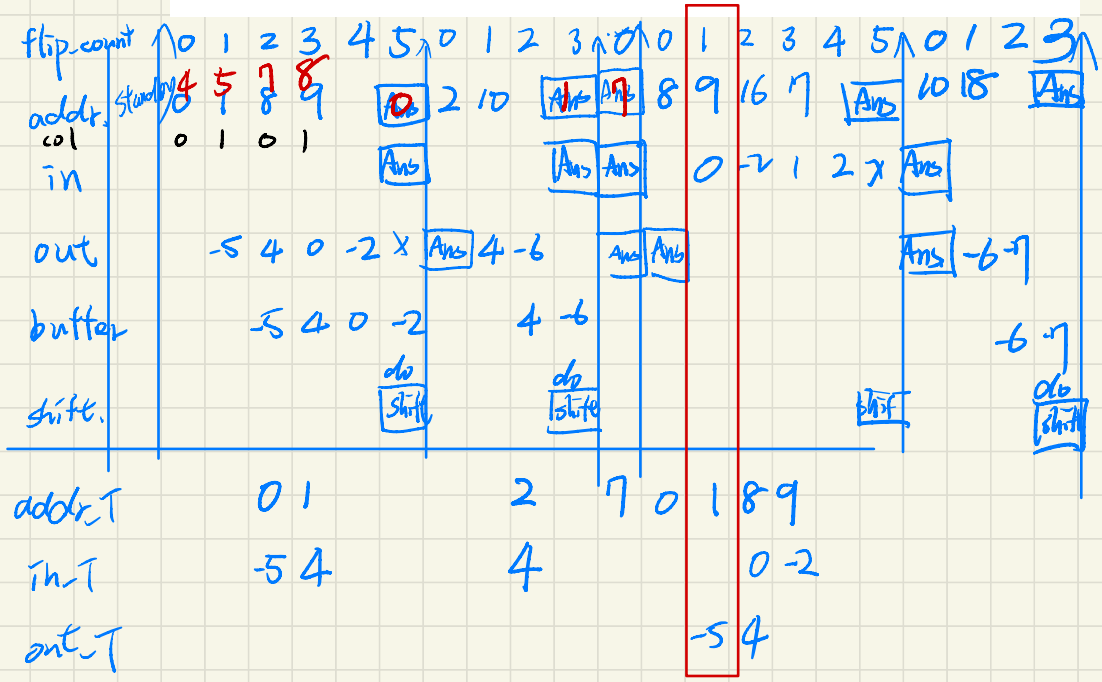
0	0	0	0	0	0	0	0	0	0
0	-5	4	4	-3	3	4	4	5	0
0	0	-2	-6	6	-4	-2	5	-8	0
0	1	2	-7	-2	6	4	-6	-6	0
0	1	4	0	-2	-1	-4	2	5	0
0	-8	-2	5	0	-8	-1	-6	3	0
0	-2	-1	4	5	6	0	-7	-4	0
0	-8	6	-3	3	-3	-8	-6	-1	0
0	-6	-5	-6	4	-6	0	-5	4	0
0	0	0	0	0	0	0	0	0	0

Template

1	-3	-5
2	-3	4
-1	5	1

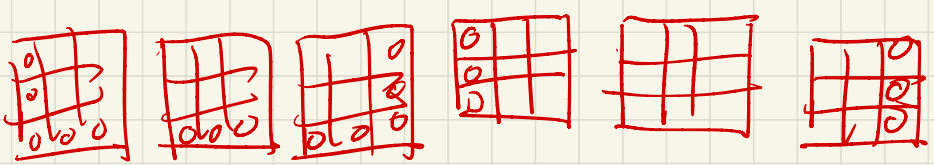
* =

29	-22	-38	61	-27	9	35	-52
-6	-53	6	-45	20	-3	-124	-1
24	23	-3	1	15	-64	44	58
-42	23	60	-42	-98	39	32	22
-18	0	21	16	66	-49	-46	-47
2	23	-29	77	-5	-34	-16	-16
24	-90	-26	-48	-64	36	18	21
-8	-32	24	-45	78	19	46	-25



9种case

$a^2 \approx 16 \times 40$



-5 4 4 -3 3 4 4 5

Image with padding

0	0	0	0	0	0	0	0	0	0
0	29	-22	-38	61	-27	9	35	-52	0
0	0	-2	-6	6	-4	-2	5	-8	0
0	1	2	-7	-2	6	4	-6	-6	0
0	1	4	0	-2	-1	-4	2	5	0
0	-8	-2	5	0	-8	-1	-6	3	0
0	-2	-1	4	5	6	0	-7	-4	0
0	-8	6	-3	3	-3	-8	-6	-1	0
0	-6	-5	-6	4	-6	0	-5	4	0
0	0	0	0	0	0	0	0	0	0

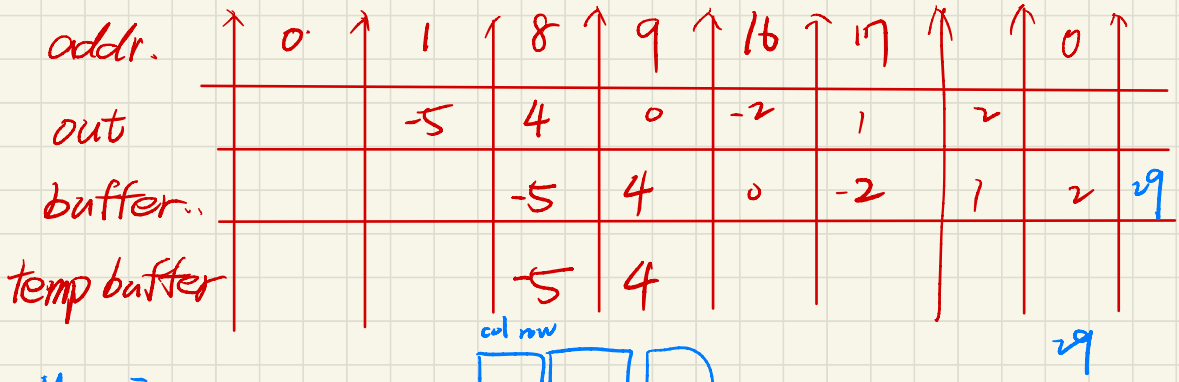
*

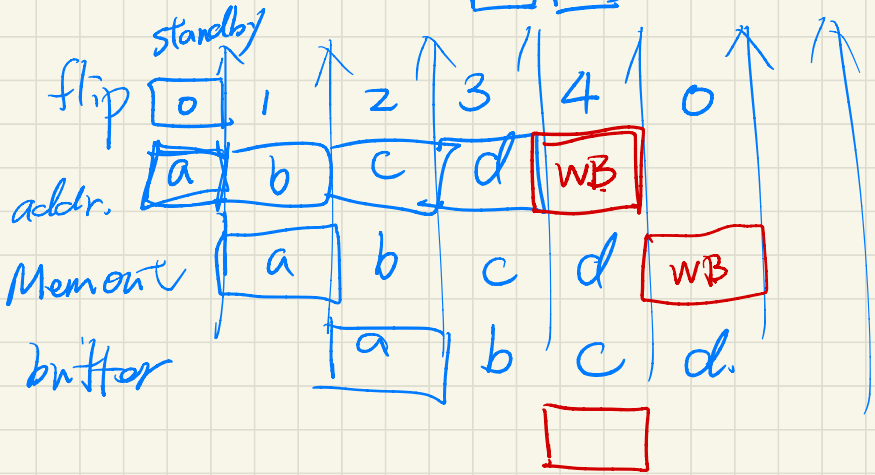
Template

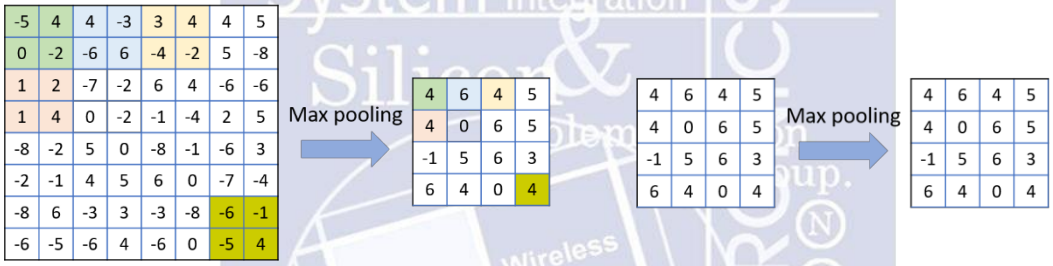
1	-3	-5
2	-3	4
-1	5	1

=

29	-22	-38	61	-27	9	35	-52
-6	-53	6	-45	20	-3	-124	-1
24	23	-3	1	15	-64	44	58
-42	23	60	-42	-98	39	32	22
-18	0	21	16	66	-49	-46	-47
2	23	-29	77	-5	-34	-16	-16
24	-90	-26	-48	-64	36	18	21
-8	-32	24	-45	78	19	46	-25

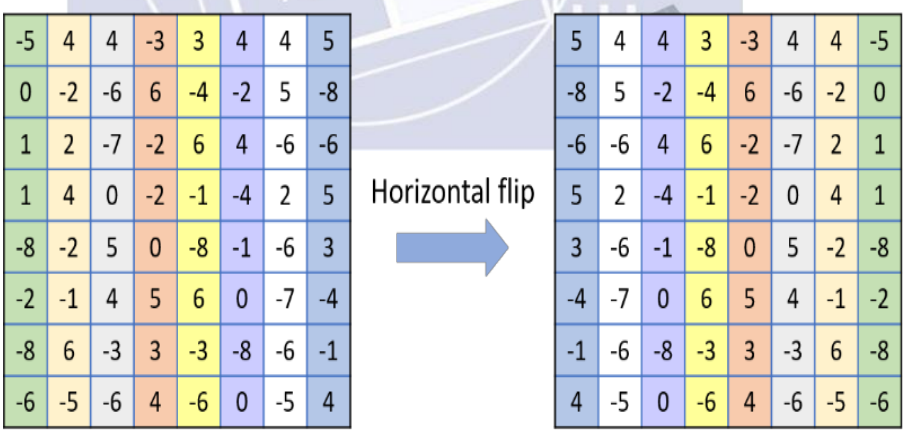






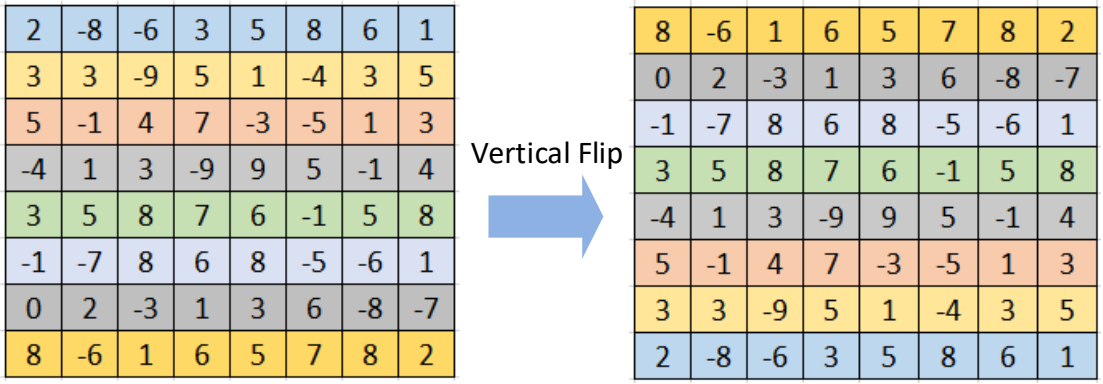
- **Horizontal Flip:**

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



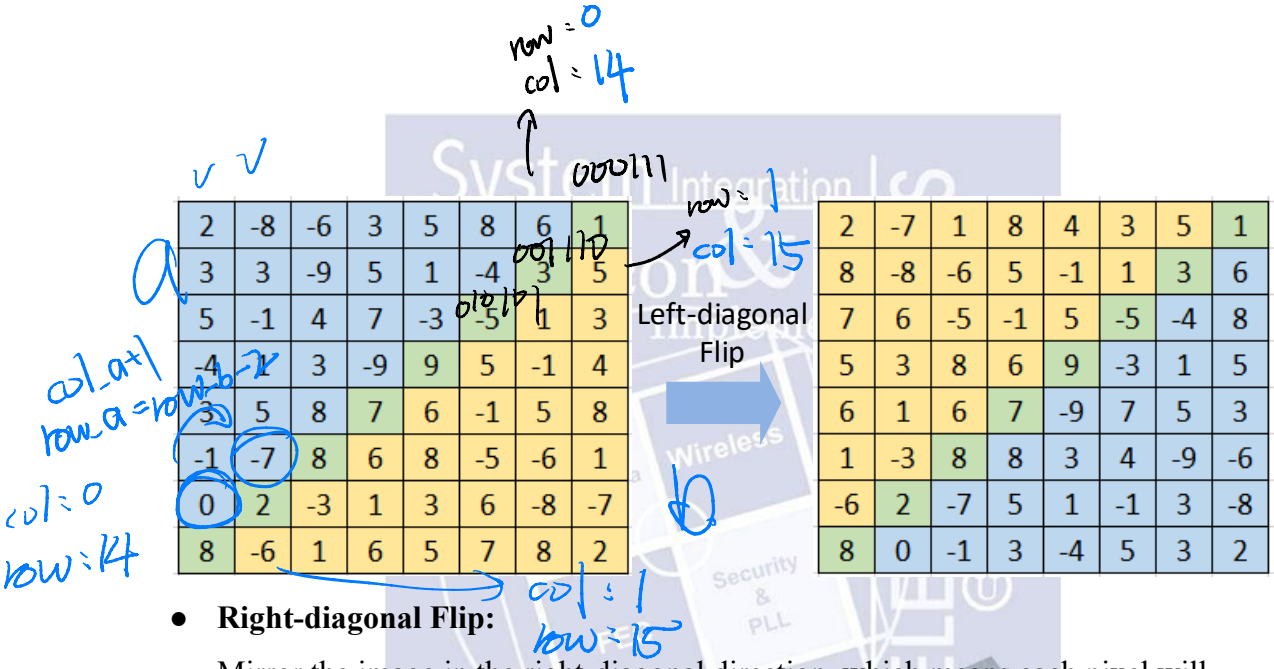
- **Vertical Flip:**

Mirror the image in the vertical direction, which means each pixel will exchange its value with the corresponding vertical row.



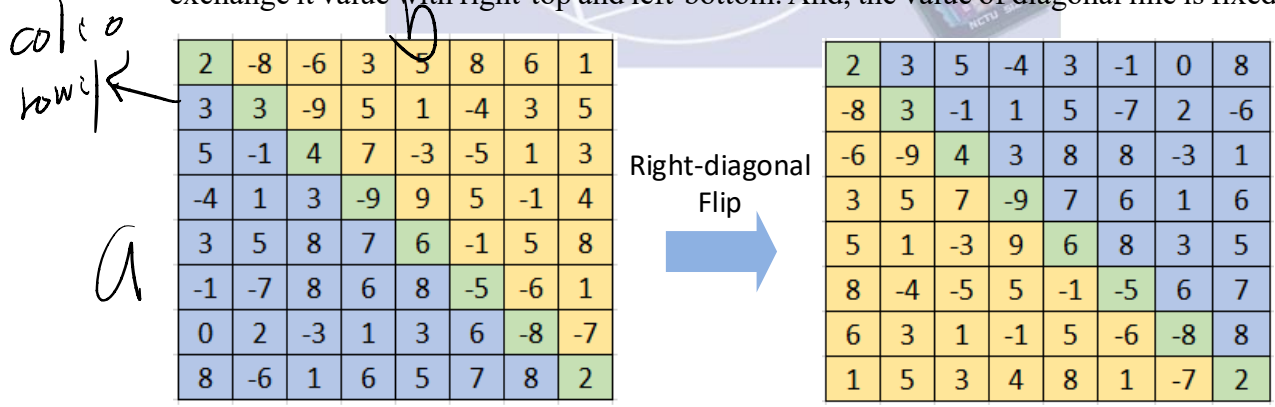
- **Left-diagonal Flip:**

Mirror the image in the left-diagonal direction, which means each pixel will exchange its value with left-top and right-bottom. And, the value of diagonal line is fixed.



- **Right-diagonal Flip:**

  Mirror the image in the right-diagonal direction, which means each pixel will exchange it value with right-top and left-bottom. And, the value of diagonal line is fixed.



- **Zoom-in:**

  Use a linear function to do Zoom-in, follow the equation to adjust the image size and resolution. The original pixel is corresponding to the new left-top of pixel, where alpha set as 0.5 and adjust set as 20 in Zoom-in. **After doing Zoom-in, the image size will be twice of the original one.**

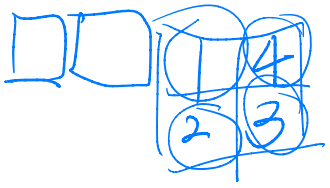
(This operation will be performed only when the shape of X is 4x4 or 8x8)

(If the shape of X is 16x16, the result of zoom-in will be still 16x16.)

$$Image'(x', y' + 1) = Image(x, y) \div 3$$

$$Image'(x' + 1, y') = (Image(x, y) \times 2) \div 3 + adjust$$

$$Image'(x' + 1, y' + 1) = alpha \times Image(x, y)$$



-60	-39	-54	-8
94	-19	87	15
89	-38	-28	-8
-4	-54	3	2

Zoom-in

-60	-20	-39	-13	-54	-18	-8	-2
-20	-30	-6	-20	-16	-27	15	-4
94	31	-19	-6	87	29	15	5
82	47	8	-10	78	43	30	7
89	29	-38	-12	-28	-9	-8	-2
79	44	-5	-19	2	-14	15	-4
-4	-1	-54	-18	3	1	2	0
18	-2	-16	-27	22	1	21	1

- In Zoom-in, the result might be in float format, so we will use the **floor function** to transform it into integer.
- The floor function, denoted as floor(x) or ⌊x⌋, is the function that takes as input a real number x, and gives as output the greatest integer less than or equal to x. **About divide by 3, ignore the floor function and do general calculation.**
  - *For example :*
    - $(94)/3 = 31$
    - $(-19)/3 = -6$
    - $floor\ (-199 \times 0.5) = -100$
    - $floor\ (89 \times 0.5) = 44$
    - $\left((-38 \times 2)/3 + 20\right) = -5$
    - $\left((94 \times 2)/3 + 20\right) = 82$

- **Shortcut + Brightness Adjustment:**

For shortcut, select the center of image and do the brightness adjustment. Use a linear function, $Image'(x,y) = alpha \times Image(x,y) + bias$ to adjust the image brightness, where alpha set as **0.5** and bias set as **50** in this exercise. **After doing shortcut, the image size will be half of the original one.**

(This operation will be performed only when the shape of X is 8x8 or 16x16)

(If the shape of X is 4x4, the result of shortcut will be still 4x4, and also do brightness adjustment.)

再加一個 state.

write back

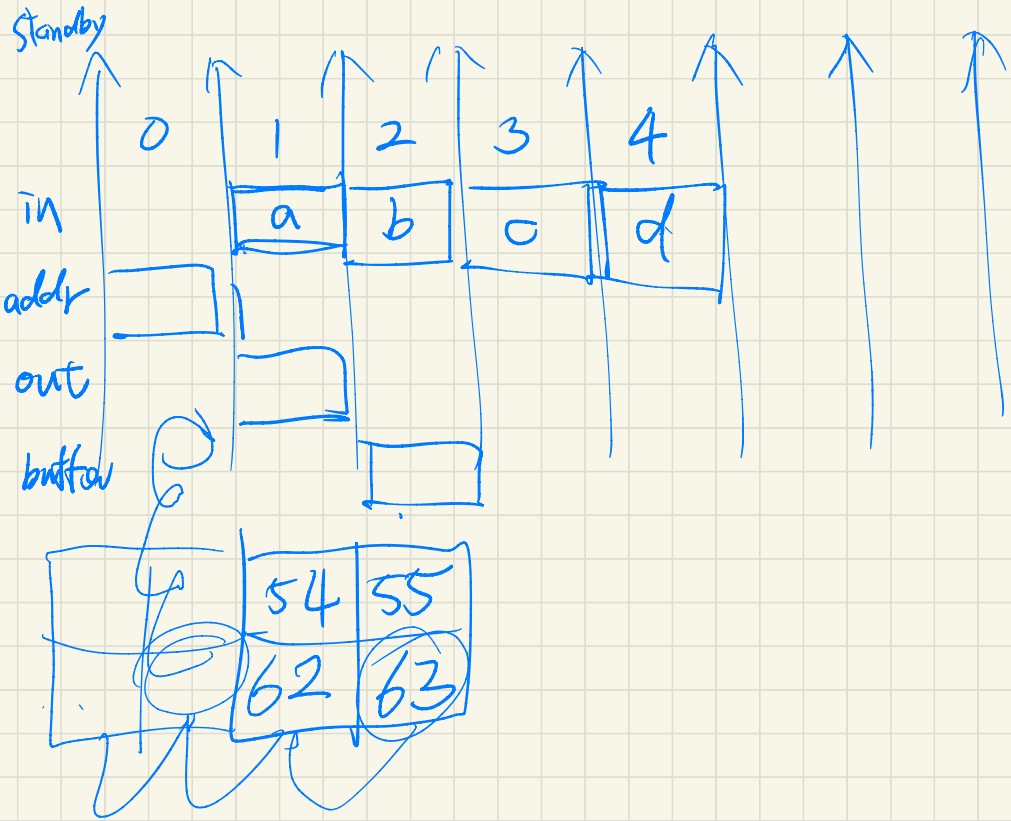
WB

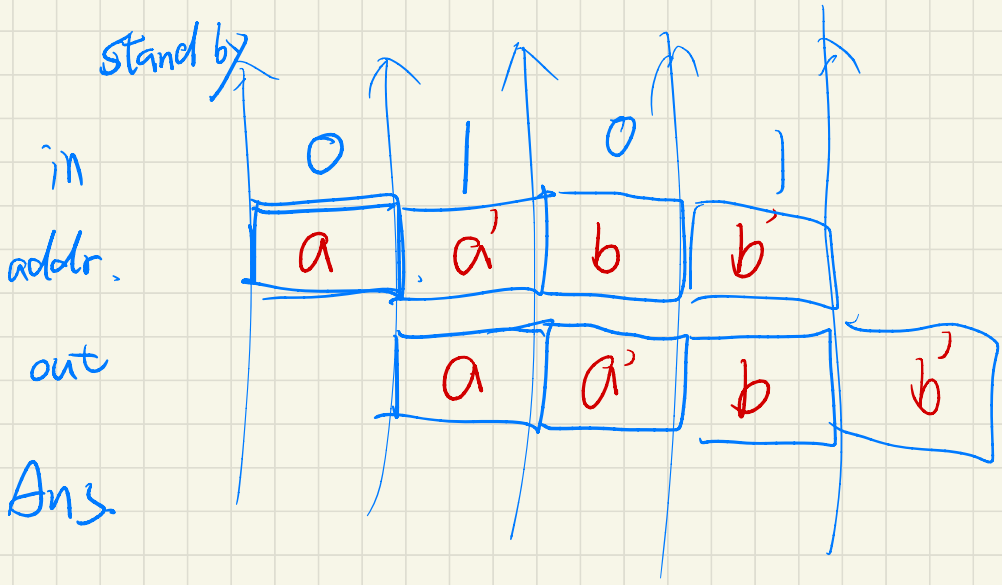
從後面開始放

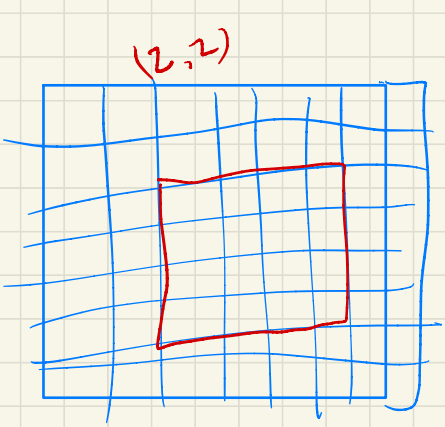
CAL : if (action_reg[0]=0, 6, 7 && outflag)

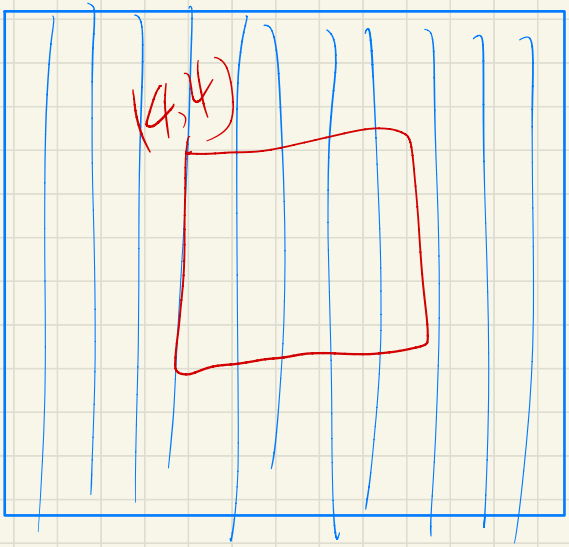
next_state = WB

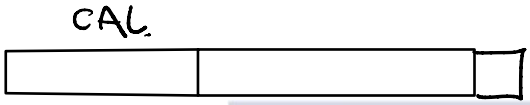
else = STANDBY

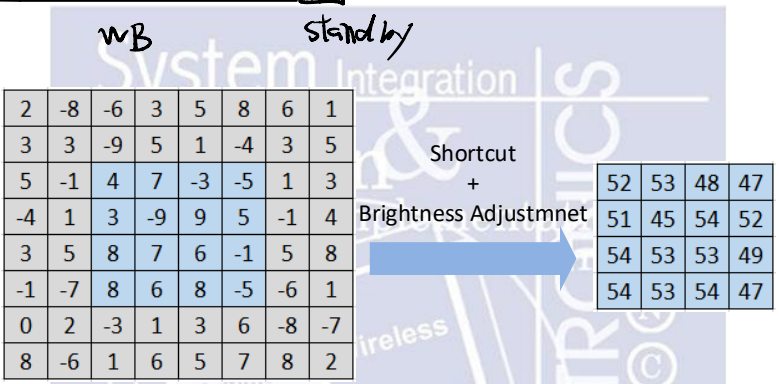

Standby
0
1
2
3
4
in
a
b
c
d
addr
out
buffer
54
55
62
63


Stand by
0
1
0
1
in
addr.
out
Ans.
a
a'
b
b'
a
a'
b
b'


(2,2)


(4,4)





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

52	53	48	47
51	45	54	52
54	53	53	49
54	53	54	47

- In Brightness Adjustment, the result might be in float format, so we will use the **floor function** to transform it into integer.
- The floor function, denoted as floor(x) or $\lfloor x \rfloor$, is the function that takes as input a real number x, and gives as output the greatest integer less than or equal to x.
  - *For example：*
    - $floor\left((27 \times 0.5) + 50\right) = 63$
    - $floor\left((-199 \times 0.5) + 50\right) = -50$

In this exercise, you will get the image, image shape and template at the beginning, then you will get several numbers which range in 0 to 7 as actions. The number of instructions ranges from 1 to 16, and **the last instruction is the cross correlation**, which represents as 3'b00. Noted that the action 0 will only **appear once** and must be the last action in each pattern. Ex：4→6→2→5→3→1→0. After doing cross correlation, you must **output the x and y coordinate of the maximum value**, **the sequences of matching template positions** and **all the cross-correlation values** simultaneously. **If the indexes of the maximum value are more than one, choose the index which is smaller**.

- *For example：*

Position index

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Cross correlation Result

x \ y	0	1	2	3
0	29	-22	-38	61
1	-6	-53	8	-35
2	24	61	5	-5
3	-43	23	50	-48

Position of maximum value：3
Index of x coordinate：0
Index of y coordinate：3

- **Example：**

At the beginning, you will obtain the image, image shape and template.

Image

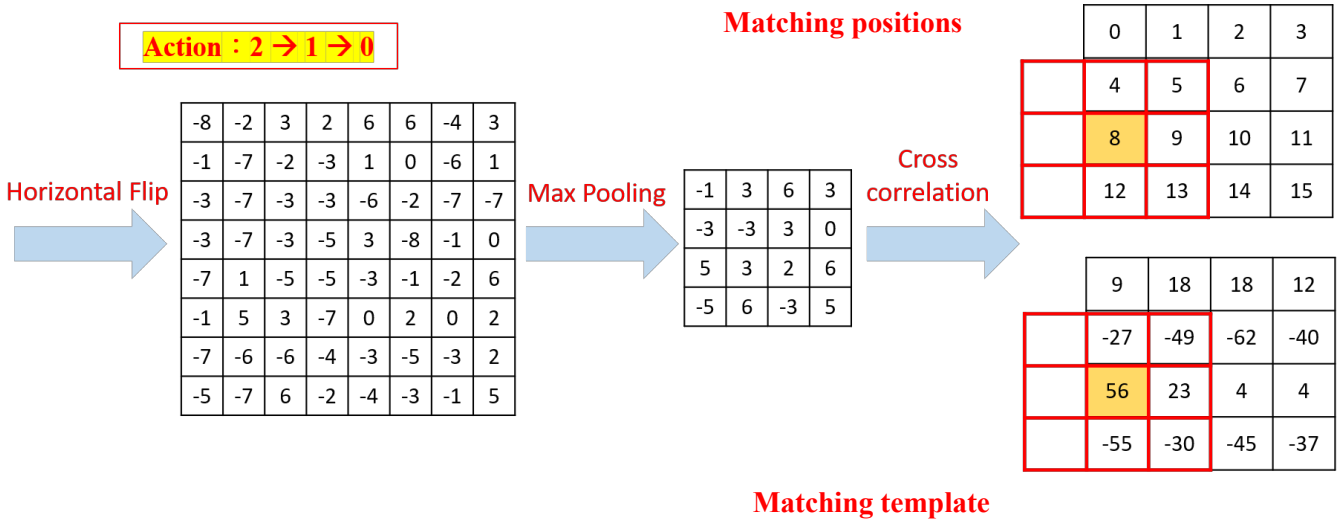
3	-4	6	6	2	3	-2	-8
1	-6	0	1	-3	-2	-7	-1
-7	-7	-2	-6	-3	-3	-7	-3
0	-1	-8	3	-5	-3	-7	-3
6	-2	-1	-3	-5	-5	1	-7
2	0	2	0	-7	3	5	-1
2	-3	-5	-3	-4	-6	-6	-7
5	-1	-3	-4	-2	6	-7	-5

Image shape：8

Template

-2	-8	-4
0	3	2
1	-1	-1

Then you will get the action numbers in sequence. In this case, the sequence of the action number is 2→1→0, which means doing “Horizontal Flip”, “Max Pooling” and “Cross Correlation”.



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

-1	3	6	3
-3	-3	3	0
5	3	2	6
-5	6	-3	5

	0	1	2	3
	4	5	6	7
	8	9	10	11
	12	13	14	15

	9	18	18	12
	-27	-49	-62	-40
	56	23	4	4
	-55	-30	-45	-37

Output the index of x and y coordinate of the maximum value, the sequences of matching template positions and all the cross-correlation values.

- `X coordinate of the maximum value：2`
- `Y coordinate of the maximum value：0`
- `Sequences of matching template positions：`4→5 →8→ 9→12 →13
- `Cross correlation values：`9 → 18 → 18 → 12 → -27 → …… → -37

Inputs

Input	Bit Width	Definition and Description
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when input signals are valid.
in_valid2	1	High when input signals are valid.
image	16	Element of input image. It will be sent in **raster scan order** continuously when **in_valid** is high. The elements are signed integers, which are represented in 2's complement format.
template	16	Element of input template. It will be sent in **raster scan order** continuously when **in_valid** is high. The elements are signed integers, which are represented in 2's complement format.
img_size	5	The signal will be given at the first cycle of **in_valid**. It defines the size of the image. **5'd4** : 4x4. **5'd8** : 8x8. **5'd16** : 16x16.
action	3	The signal will be given when **in_valid2** is high. The definition is as following: **3'd0**: Cross Correlation **3'd1**: Max pooling. **3'd2:** Horizontal flip **3'd3**: Vertical flip. **3'b4:** Left-diagonal flip **3'b5**: Right-diagonal flip **3'b6:** Zoom-in **3'b7**: Shortcut+Brightness Adjustment

Outputs

Output	Bit Width	Definition and Description
out_valid	1	High when out is valid. It cannot be overlapped with **in_valid** signal.
out_value	40	Result matrix outputted in raster scan order. The elements are signed integers, which are represented in 2's complement format.
out_x	4	The X coordinate of the maximum value. Range from 0 to 15.
out_y	4	The Y coordinate of the maximum value. Range from 0 to 15.
out_img_pos	8	The index of matching template positions. Range from 0 to 255.

1. The input of **image, template** are delivered in **raster scan order** for **current size of matrix(4x4, 8x8…) cycles continuously**. When input of template finish delivered, it will be tied to unknown state. When **in_valid** is low, the input of **image is** tied to unknown state.

2. The input of **action** is delivered for **several cycles which depends on the numbers of actions continuously**. When **in_valid2** is low, the input of **action** is tied to unknown state.

3. in_valid2 will be triggered at next negative edge after in_valid is low.

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

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

6. The output signal **out_img_pos** must be delivered for **several cycles which depends on current numbers of matching template positions continuously**, and **out_valid** should be high simultaneously.

7. If all the matching positions output are finished, please set **out_img_pos** to 0.

Specifications

1. Top module name: TMIP (Design file name：TMIP.v)
2. **It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock staring) in your design, you may fail to reset signals should be reset after the reset signal is asserted.**
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. You can adjust your clock period by yourself, but the maximum period is **12 ns**.
5. The data type in the synthesis result **CAN NOT** include any **LATCH**.
6. After synthesis, the area report is valid only when the slack in the end of timing report is **non-negative** and the result should be **MET**.
7. The next input will come in **2~5** cycles after your **out_valid** is pulled down.
8. The **out_valid** cannot overlap with **in_valid**.
9. The execution latency is limited in **10000 cycles.** The latency is the clock cycles between the falling edge of the **in_valid2** and the rising edge of the **out_valid**.
10. 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/ folder. 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
```

Fig 1. The area of your memory

11. **The total cell area should not larger than 800,000 μm^2.**
12. **If any port of memory is connected with mismatch width, the memory will not be synthesized and you will get an error message as shown in Fig 2. Even though the design may still pass gate level simulation, this situation will be regarded as synthesis**

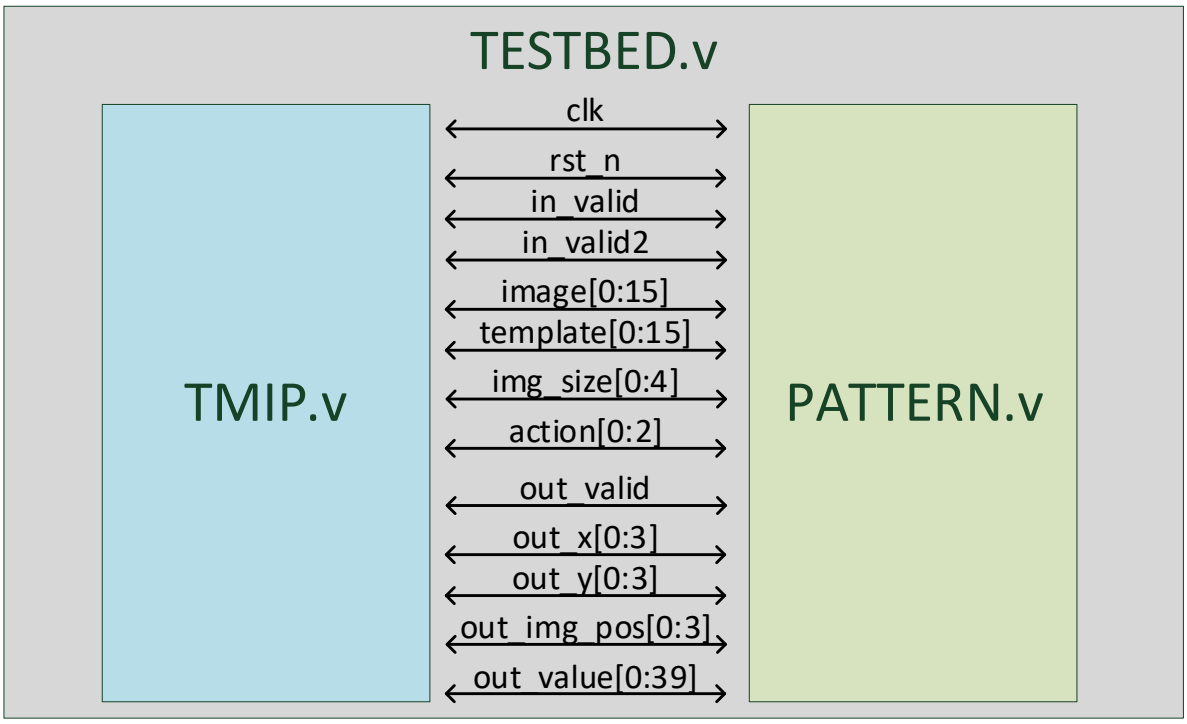
fail. In this case, memory area will be 0 in TMIP.area. We will check it at syn.log and TMIP.area.

```
Error: Width mismatch on port 'A' of reference to 'RA1SH_256_32_4' in 'CNN'. (LINK-3)
Warning: Unable to resolve reference 'RA1SH_256_32_4' in 'CNN'. (LINK-5)
```

Fig 2. Memory port width mismatch error

13. **All numbers are signed integers and expressed in 2's complement format. Be sure the operations are done with signed operations.**
14. **Every** output signal should be correct when **out_valid** is high.
15. The input delay is set to **0.5*(clock period)**.
16. The output delay is set to **0.5*(clock period)**, and the output loading is set to **0.05**.
17. The gate level simulation cannot include any timing violations without the *notimingcheck* command.
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 comments or other language comments in the file you turned in.
20. Verilog commands //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.
21. 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.
22. **The synthesis time in 02_SYN can't over 1 hour.**

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^3 * (Total Latency)**

Note

1. **Please upload the following file on e3 platform before 23:59 on Apr. 3:**
   RTL design : **TMIP_iclabXXX.v** (**XXX** is your account no.)
   clock_cycle_iclabXXX.txt
   Memory file : **MEMORY_NAME_iclabXXX.v**
   MEMORY_NAME_iclabXXX.db
   file_list_iclabXXX.f

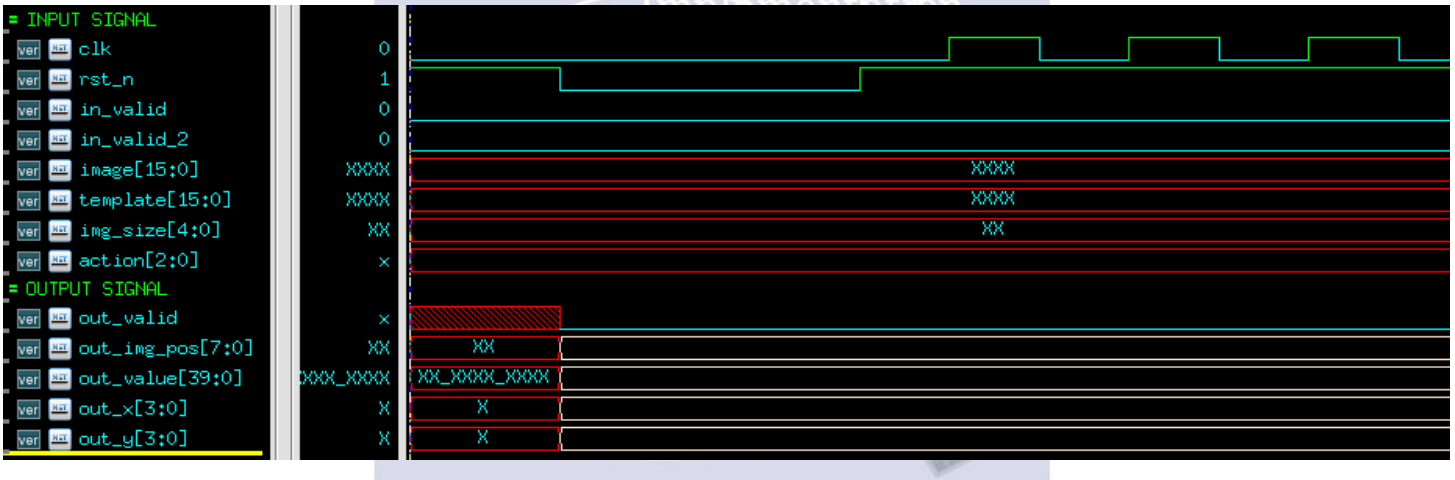
- ***Example:***
  - Submit your design files：
    TMIP_iclab999.v
    18.0_iclab999.txt
  - Given two memories in your design, RA1SH1 and RA1SH2
    A. **Submit these memory files:**
    RA1SH1_iclab999.v RA1SH1_iclab999.db
    RA1SH2_iclab999.v RA1SH2_iclab999.db
    B. **Type following in file_list_iclab999.f：**
    ../04_MEM/RA1SH1_iclab999.v
    ../04_MEM/RA1SH2_iclab999.v

2. Template folders and reference commands:
   In RTL simulation, the name of template folder and reference commands is:
   01_RTL:
   "**./01_run**"
   02_SYN/ (Synthesis):
   ./01_run_dc
   (Check **latch** by searching the keyword "**Latch**" in 02_SYN/**syn.log**)
   (Check the design's timing in /Report/TMIP.timing)
   (Check the design's area in /Report/TMIP.area)
   03_GATE_SIM/:
   ./01_run
   04_MEM/ (Memory location)
   (You should generate your own memory and put the required files here and remember that use the command ./02_lib_gen_syntax_match.sh)

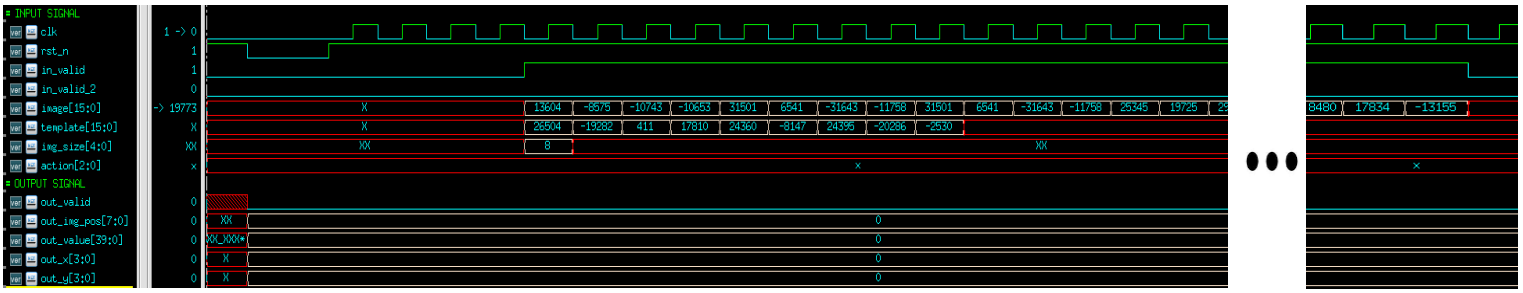
- You can key in **./09_clean_up** to clear all log files and dump files in each folder

Example Waveform

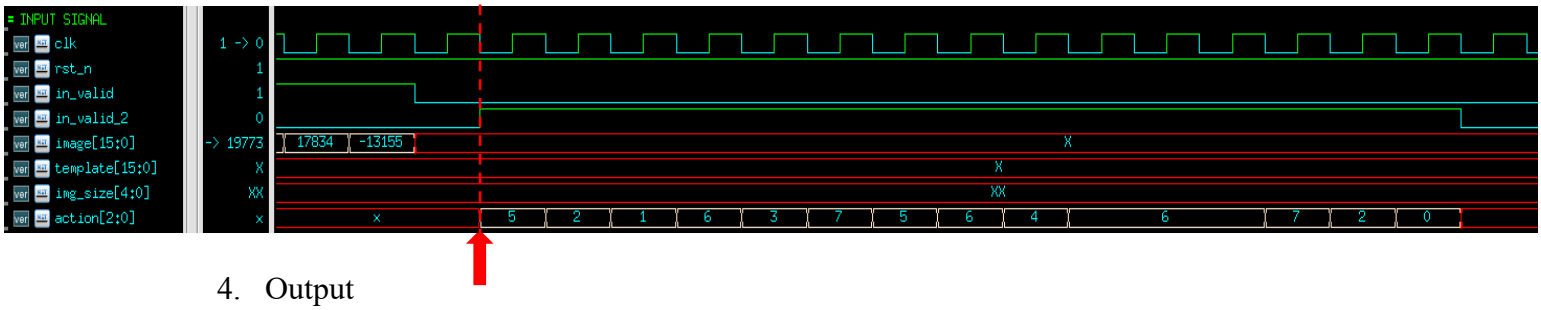
1. Reset signal



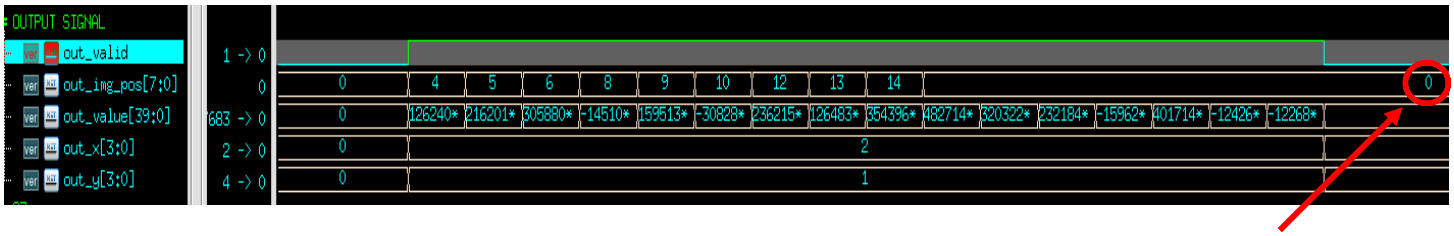
2. Input of **image, template** and **image shape**



3. Input of **action**



4. Output



If all the matching positions output are finished, set **out_img_pos** to 0

File_list.f and .tcl file

1. The content of file_list.f like the following figure. If you have more than two different SRAM.v file, you can add the other new line in below.

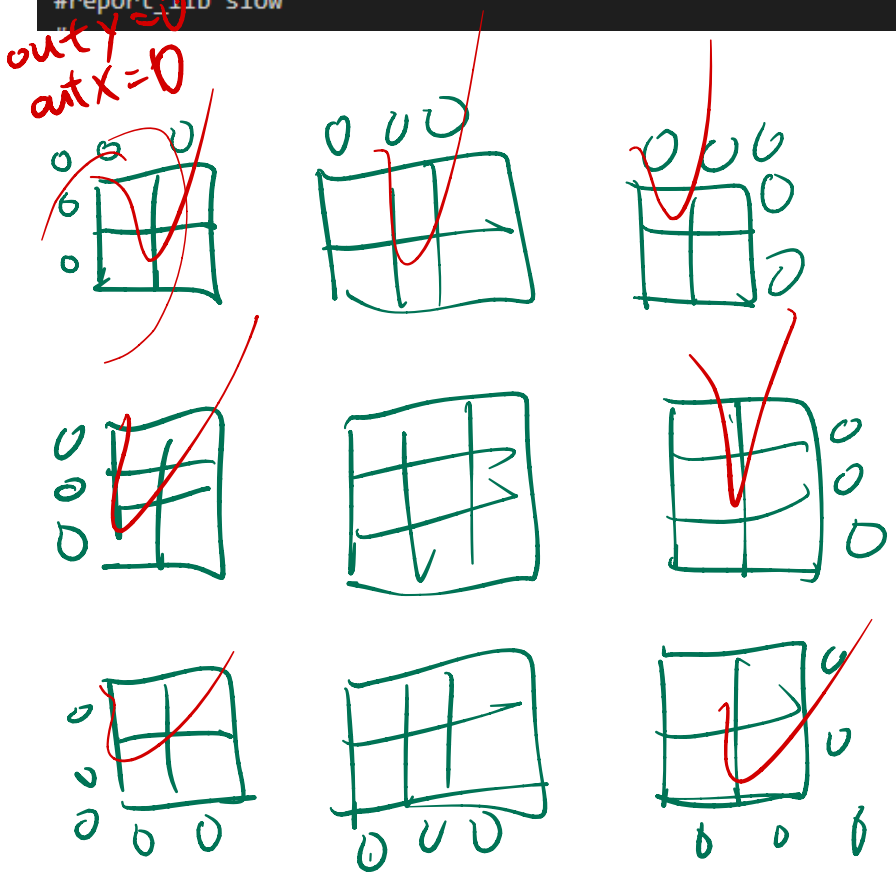
```
1    ../04_MEM/RAISH.v
```

2. Before Synthesis, you need to add SRAM.db file into .tcl file to let compiler know the information of SRAM.

```
set search_path {./../01_RTL \
                 ./../04_MEM \
                 ~iclabta01/umc018/Synthesis/ \
                 /usr/synthesis/libraries/syn/ \
                 /usr/synthesis/dw/ }

set synthetic_library {dw_foundation.sldb}
set link_library {* dw_foundation.sldb standard.sldb slow.db RAISH.db}
set target_library {slow.db}

#report_lib slow
```



$12 \times (593246)^3$ 2.5×10^{18}

$8 \times (635974)^3$ 2.05×10^{18}