

NCTU-EE IC LAB – Fall 2022

Lab05 Exercise

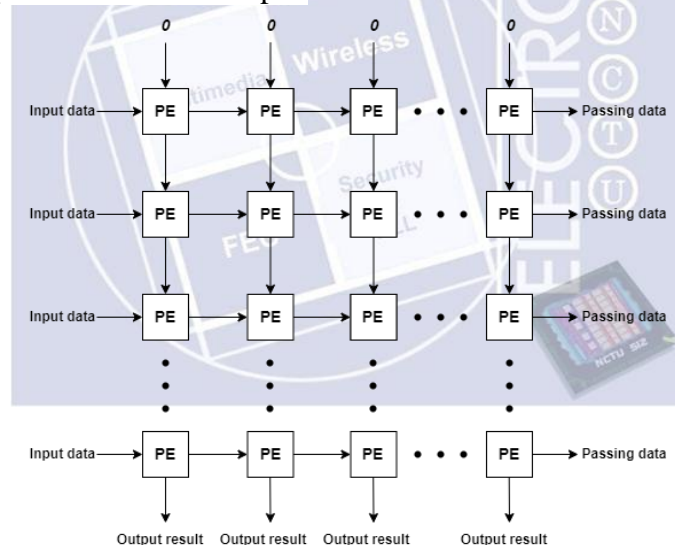
Design: Matrix Multiplication with Systolic Array

Data Preparation

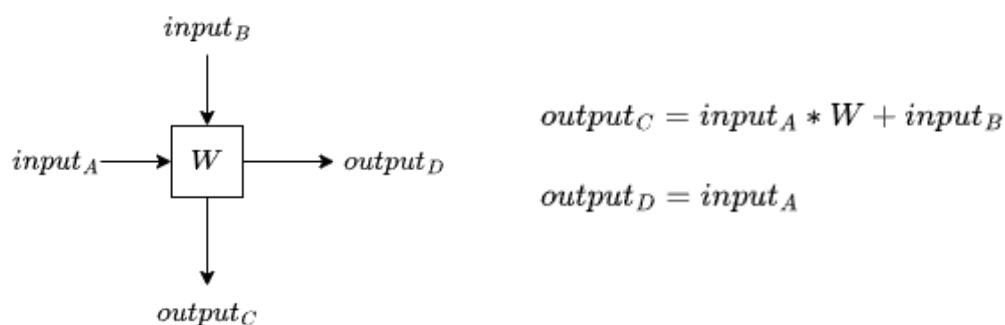
1. Extract files from TA's directory:
% tar xvf ~iclabta01/Lab05.tar
2. The extracted LAB directory contains:
 - a. Practice/
 - b. Exercise/

Design Description

A systolic array is a homogeneous network of tightly coupled data processing elements (PEs) called cells or nodes. Each node computes a partial result as a function of the data received from its upstream neighbors, stores the result within itself and passes it downstream. Systolic arrays often apply to specific operations, such as convolution, correlation, matrix multiplication or data sorting task. We will take **matrix multiplication** as an example.



The Behavior of each PE is Multiply-Accumulate (MAC) operation.



Example

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{bmatrix}$$

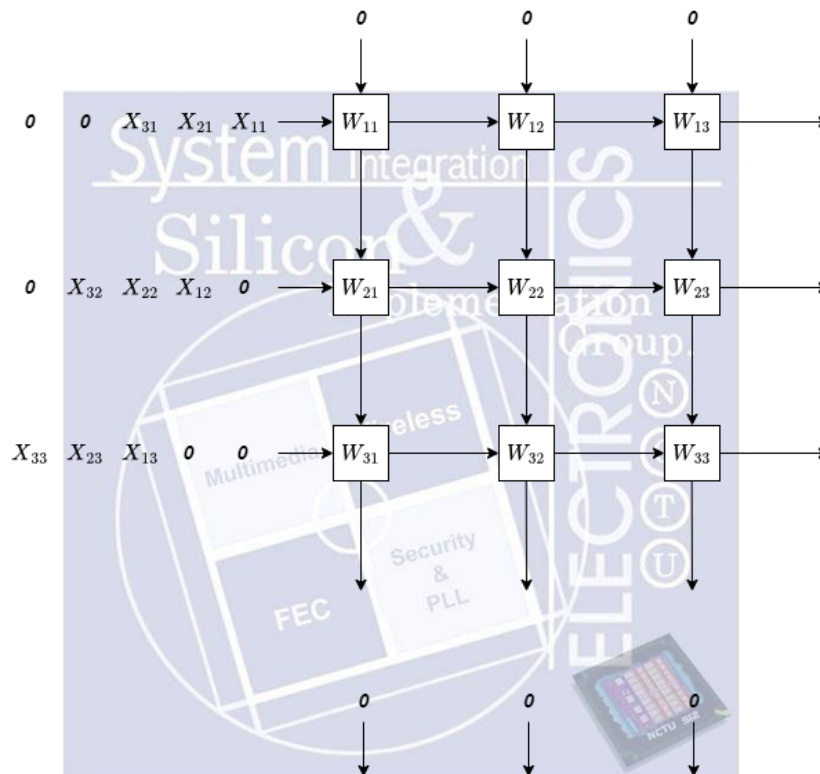
Input matrix

Weight matrix

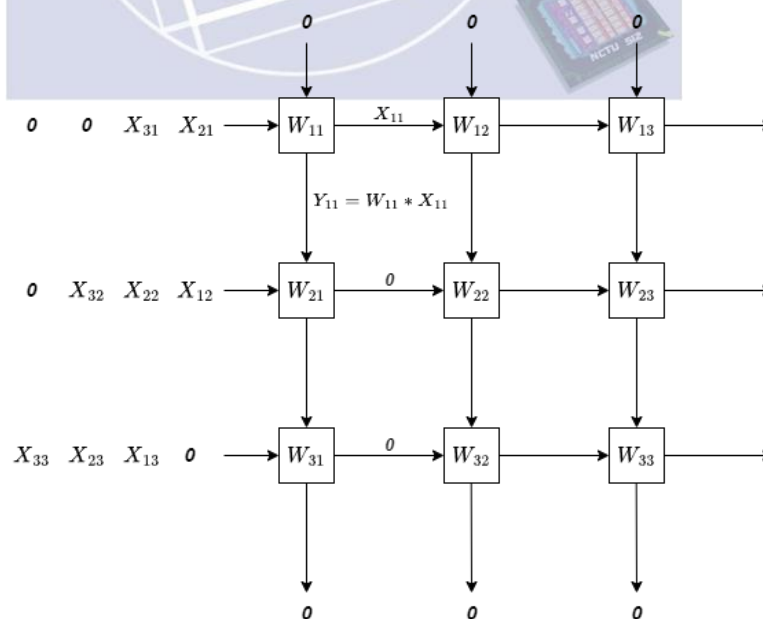
Output matrix

To resolve this $(3 \times 3) \cdot (3 \times 3)$ matrix multiplication, you need a systolic array which has 3×3 PEs. In each PE, you have to fix one element of weight matrix, then reschedule the input data flow.

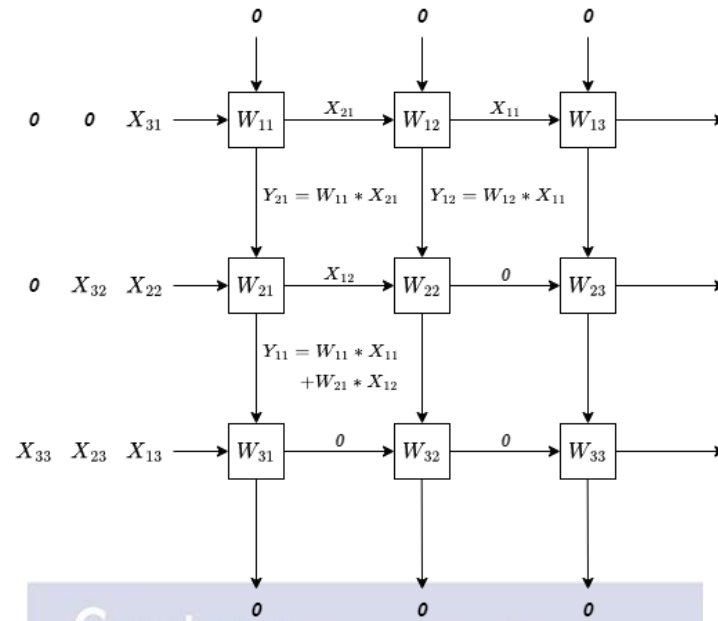
Cycle=0



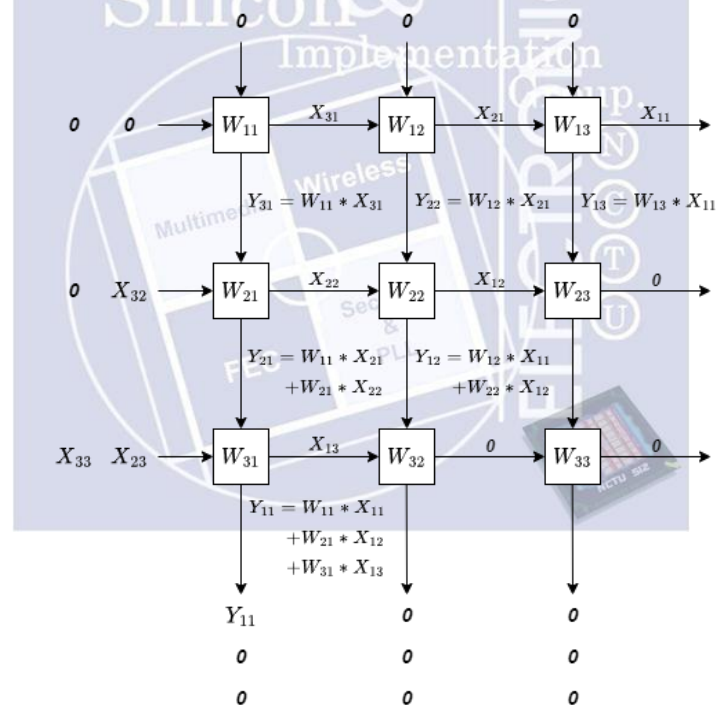
Cycle=1



Cycle=2

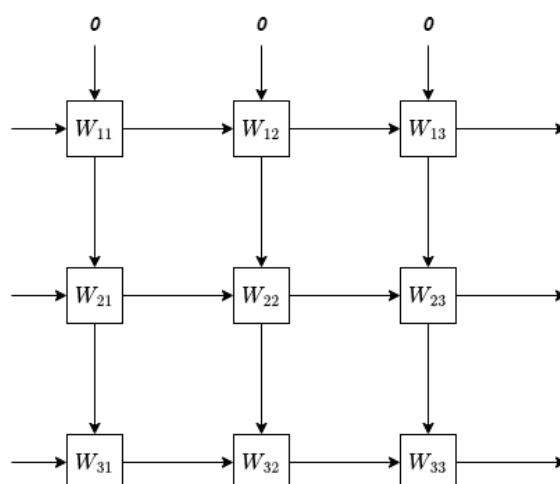


Cycle=3



•
•
•
•
•
•

Cycle=7



In this exercise, you need to build a design to perform a square matrix multiplication. Because the matrix requires large space to store, you are suggested to **use memory** for finishing this lab.

■ Size :

The **input matrix size** and **weight matrix size** will be **2x2, 4x4, 8x8** and **16x16** according to the given size value which will be given at the beginning of each pattern. The **output matrix size** will be the same as given input and weight matrix size.

■ Rules of input data :

In this exercise, you will get a matrix size and 32 matrices continuously at the beginning of each pattern. **The first 16 matrices will be input matrices, and the last 16 matrices will be weight matrices.** After that, you will get an input matrix index and a weight matrix index, and the two matrices will be used in matrix multiplication. These two indices range from 0 to 15, and each index **will not repeat**, so you will get pairs of indices for total 16 times in a given pattern.

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix} \dots \begin{bmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix}$$

Input matrix 0 Input matrix 1 Input matrix 15

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix} \dots \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix}$$

Weight matrix 0 Weight matrix 1 Weight matrix 15

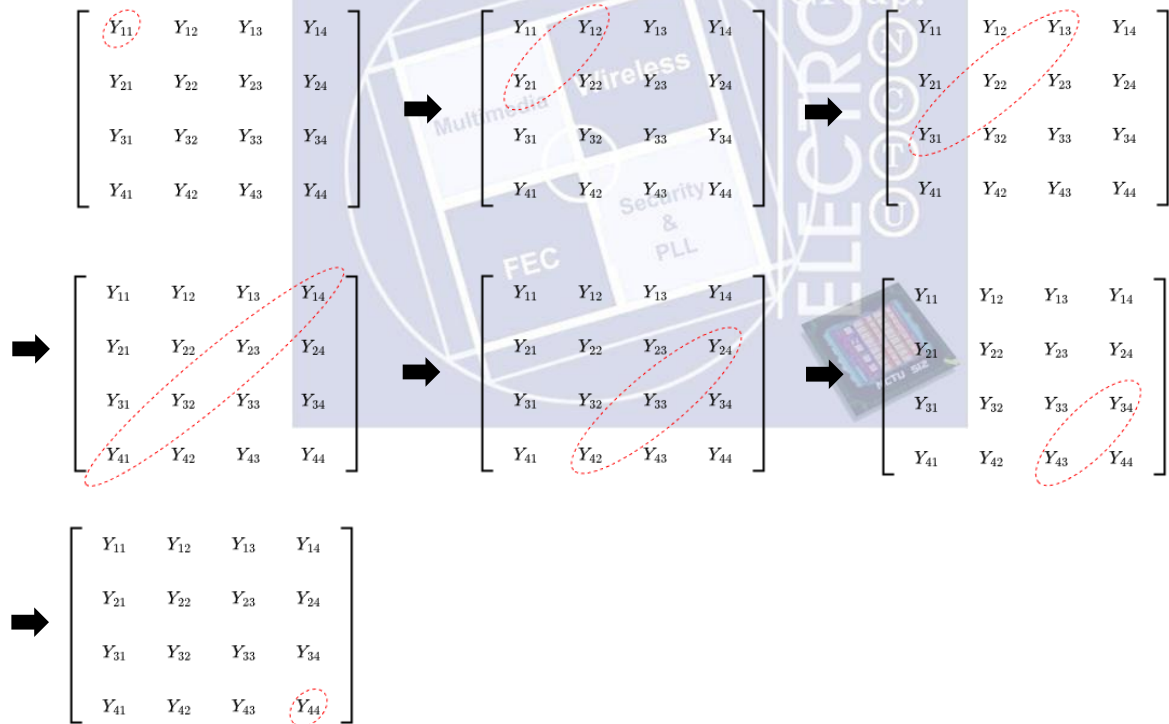
Input matrix index: 1, 0, 15, 13, 10,, 14, 9, 7

Weight matrix index: 0, 15, 1, 8, 11,, 2, 12, 5

1. There are total 16 pairs
2. The number in each index will not repeat
3. Index sequence is random

Rules of output result :

In each matrix multiplication, you will get an output matrix, and then compute the **summation along antidiagonal direction**. After that, you have to output the summation in sequence.



In this example, you will output the results with 7 cycles, and the output value will be:

$$Y_{11} \rightarrow Y_{12} + Y_{21} \rightarrow Y_{13} + Y_{22} + Y_{31} \rightarrow Y_{14} + Y_{23} + Y_{32} + Y_{41} \rightarrow Y_{24} + Y_{33} + Y_{42} \rightarrow Y_{34} + Y_{43} \rightarrow Y_{44}$$

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.
matrix	16	Elements of input matrix and weight matrix. 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.
matrix_size	2	The signal will determine the dimension of input matrix, weight matrix and output matrix. It will be given in first cycle when in_valid is high. 2'b00: 2x2. 2'b01: 4x4. 2'b10: 8x8. 2'b11: 16x16.
i_mat_idx	4	Input matrix index, this signal will be given when in_valid2 is high.
w_mat_idx	4	Weight matrix index, this signal will be given when in_valid2 is high.

Outputs

Output	Bit Width	Definition and Description
out_valid	1	High when out is valid. It cannot be overlapped with in_valid and in_valid2 signal.
out_value	40	It will output the summation along antidiagonal direction of output matrix in sequence. The elements are signed integers, which are represented in 2's complement format.

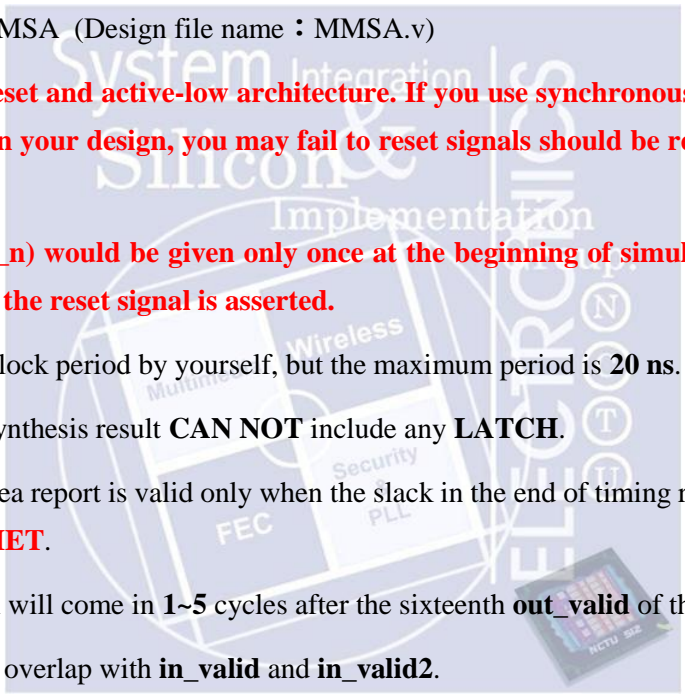
1. The **matrix** signal is delivered in **raster scan order** for **32*current size of matrix(2x2, 4x4...)** cycles continuously when **in_valid** is tied high. The first 16 matrices are input matrices and the last 16 matrices are weight matrices. When matrix finish delivered, it will be tied to unknown state, and **in_valid** will also be tied low.
2. The input of **matrix_size** is delivered for **only one cycle** during the **first cycle of in_valid tied high**. After that, the **matrix_size** signal is tied to unknown state.
3. Every time **in_valid2** is triggered, it is tied high **for only one cycle**.
4. The **in_valid2** signal will be triggered **for total 16 times** after **in_valid** is tied low in a single pattern. After **each time** **in_valid2** triggers, your design will do the matrix multiplication with specific matrices and then **out_valid** will be tied high for several cycles (3, 7, 15, 31 for 2x2, 4x4, 8x8, 16x16).
5. In each pattern, the **in_valid2** signal will be first time triggered at next negative edge after **in_valid** is

tied low, and the other fifteen times in_valid2 will be triggered at next negative edge after out_valid is tied low.

6. The next input pattern will be triggered 1~5 cycles after **the sixteenth** out_valid of this pattern falls.
7. The input of **i_mat_idx** and **w_mat_idx** are delivered for **only one cycle** during in_valid2 tied high. After that, the **i_mat_idx** and **w_mat_idx** are tied to unknown state.
8. All input signals are synchronized at negative edge of the clock.
9. The **out_value** must be delivered for **several cycles continuously** (3, 7, 15, 31 for 2x2, 4x4, 8x8, 16x16) **which depends on current size of matrix**, and **out_valid** should be high simultaneously.

Specifications

1. Top module name: MMSA (Design file name : MMSA.v)
2. **It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock starting) 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 **20 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 pattern will come in **1~5** cycles after the sixteenth **out_valid** of this pattern is pulled down.
8. The **out_valid** cannot overlap with **in_valid** and **in_valid2**.
9. The execution latency is limited in **2000 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 MMSA.area in 02_SYN/Report/ folder. The area of Macro/Black Box must not be 0. The example is shown in following figure.**

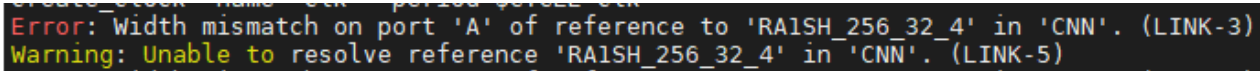


```
Combinational area:.....1821995.696653
Buf/Inv area:.....111973.280126
Noncombinational area:.....343750.185371
Macro/Black Box area:.....214305.703125
Net interconnect area:.....undefined (No wire load specified)

Total cell area:.....2380051.585150
```

Fig 1. The area of your memory

11. **The total cell area should not larger than $15,000,000\mu m^2$.**
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 MMSA.area. We will check it at syn.log and MMSA.area.**

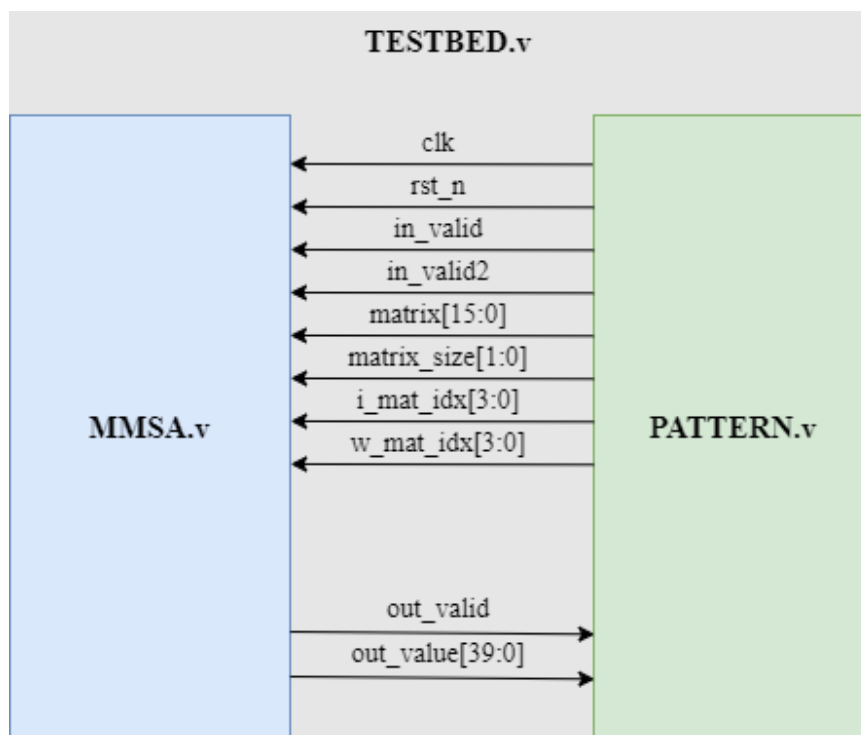


```
Error: Width mismatch on port 'A' of reference to 'RA1SH_256_32_4' in 'CNN'. (LINK-3)
Warning: Unable to resolve reference 'RA1SH_256_32_4' in 'CNN'. (LINK-5)
```

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.

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% **Area² * Total Latency**

Note

1. Please submit your files under **09_SUBMIT** before 12:00 at noon on Oct. 24:

- If uploaded files **violate the naming rule**, you will get **5 deduct point**.
- 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

```
[Exercise/09_SUBMIT]% ./01_submit 10.9
```

After that, you should check the following files under **09_SUBMIT/Lab05_iclabXXX/**

RTL design : **MMSA_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

If you miss any files on the list, you will fail this lab.

Then use the command like the figure below to check the files are uploaded or not.

[Exercise/09_SUBMIT]% ./02_check 1st_demo

- **Example:**

- Submit your design files :

MMSA_iclab999.v

10.9_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 and submit it :**

../04_MEM/RA1SH1.v

../04_MEM/RA1SH2.v

- 2. **Template folders and reference commands:**

01_RTL/ (RTL simulation) **./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/MMSA.timing)

(Check the design’s area in /Report/MMSA.area)

03_GATE/ (Gate-level simulation) **./01_run**

04_MEM/ (Memory location)

(You should generate your own memory and put the required files here)

09_SUBMIT/ (submit your files) **./01_submit** **./02_check**

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

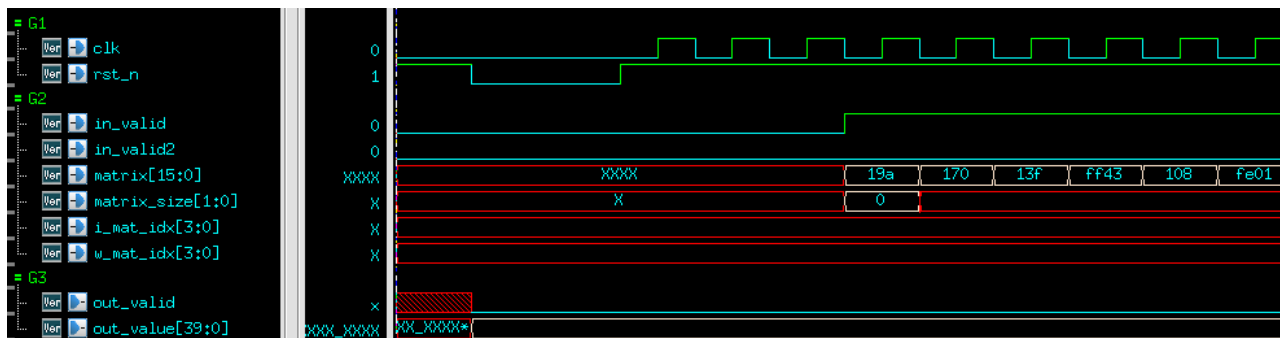
※ You should make sure the **three clock period values identical** in 00_TESTBED/PATTERN.v and 02_SYN/syn.tcl

```
^ifdef RTL
  `timescale 1ns/10ps
  `include "MMSA.v"
  `define CYCLE_TIME 20.0
^endif
^ifdef GATE
  `timescale 1ns/10ps
  `include "MMSA_SYN.v"
  `define CYCLE_TIME 20.0
^endif

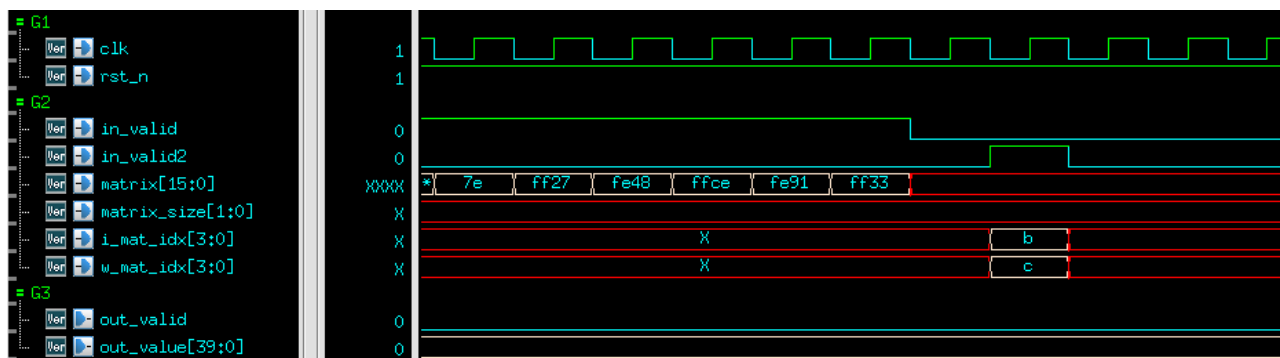
# Global Parameters
set DESIGN "MMSA"
set CYCLE 20.0
```

Example Waveform

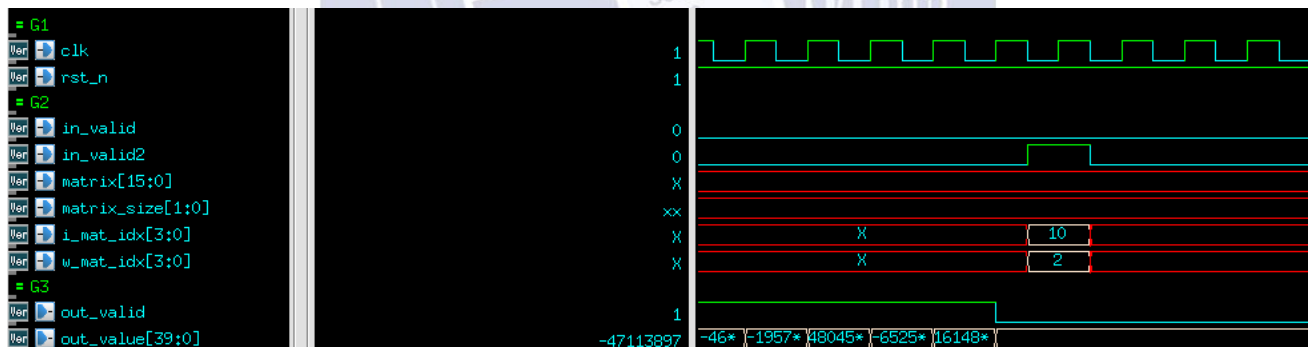
1. Asynchronous reset and active-low reset all output



2. The in_valid2 signal will triggered first time at next negative edge after in_valid is tied low



3. The other fifteen times in_valid2 will be triggered at next negative edge after out_valid next



4. The next input pattern will be triggered 1~5 cycles after the sixteenth out_valid of this pattern falls.

