NYCU-EE IC LAB – Fall 2024

Lab04 Exercise

Design: Convolution Neural Network

Data Preparation

1. Extract file from TA's directory:

   % tar xvf ~iclabTA01/Lab04.tar

2. The extracted LAB directory contains:

   a. **00_TESTBED**

   b. **01_RTL**

   c. **02_SYN**

   d. **03_GATE**

   e. **09_SUBMIT**

Design Description

Convolution Neural Networks (CNN) is a class of artificial neural networks that has become dominant in various computer vision tasks, attracting interest across a variety of domains, including radiology, image recognition, and so on. CNN is designed to automatically and adaptively learn spatial hierarchies of features through back propagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers.

In this exercise, you are asked to implement a ***Convolution Neural Network*** as Fig1.

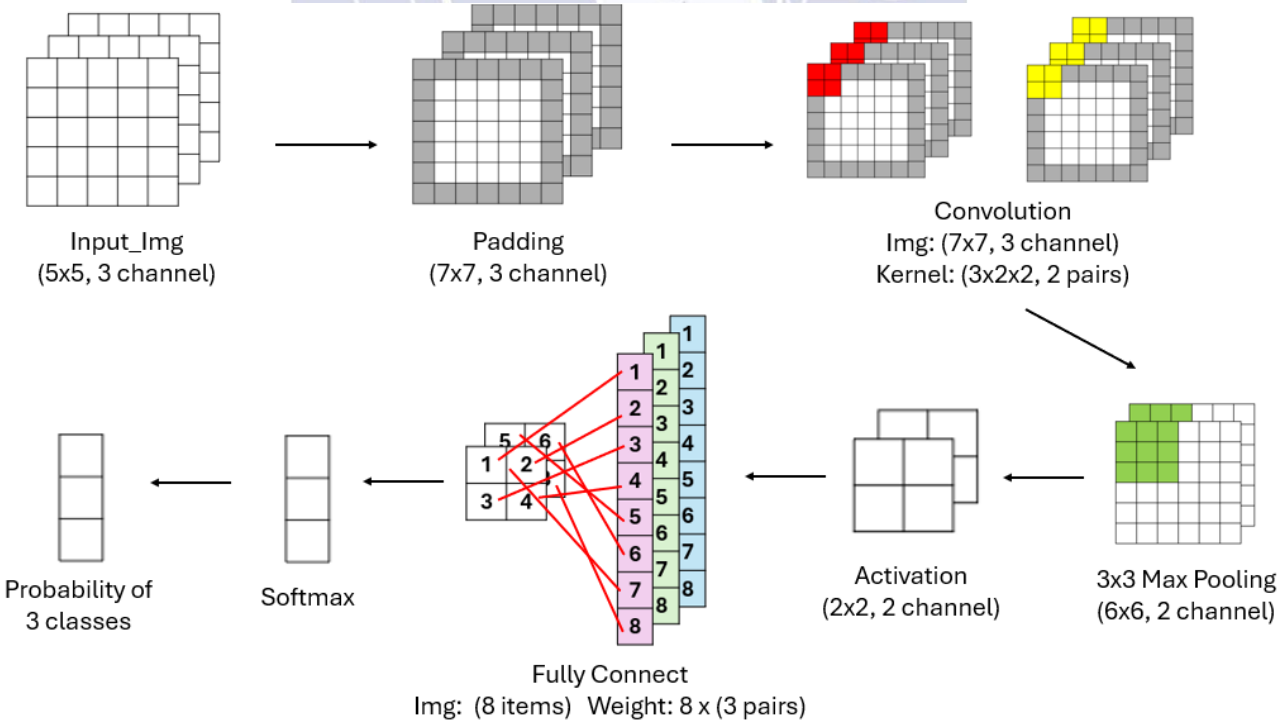


Fig 1. CNN Process

Before doing the convolution, you must perform **Replication Padding or Zero Padding** according to the information given by Opt. The padding width is 1. And before fully connect layer, you have to apply max pooling to downsize, and activation to introduce non-linearity. At length, softmax is applied to convert our data into a probability distribution.

- Description of input signals

When **in_valid** is high, the 32-bit **Img** signals will receive **(5 x 5) x 3 = 75** cycles continuously to represent the 5x5x3 images.

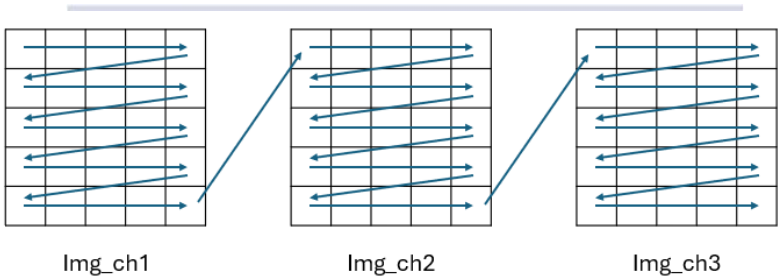


Fig 2. Sending order of the Img signal

The 32-bit unsigned **Kernel** signal will receive 12 cycles, representing two 3x2x2 kernels (**Kernel_ch1** and **Kernel_ch2**), with both kernels transmitted **simultaneously** over the 12 cycles.

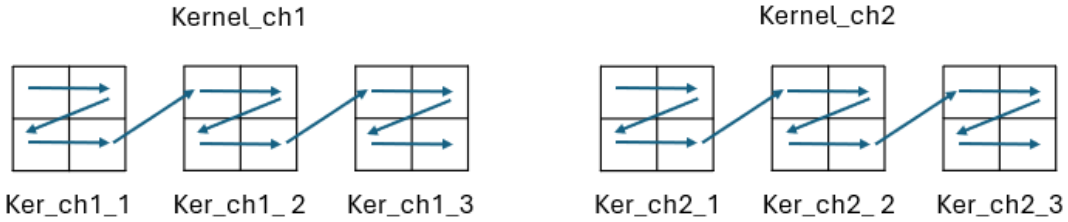


Fig 3. Sending order of the Kernel signals

The 32-bit **Weight** signal will receive **24** cycles continuously to represent the 3x8 matrix for the weight of the fully connected layer.

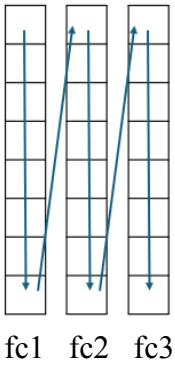


Fig 4. Sending order of the Weight signal

When the images have been provided, i.e., after 75 cycles, **in_valid** will be pulled low. Note that the input signals **Img**, **Kernel_ch1 & Kernel_ch2**, and **Weight** are all sent in raster scan order.

- **Replication Padding**

  Replication padding, is a technique used in image processing and computer vision to extend the borders of an image by replicating or mirroring the existing pixels.

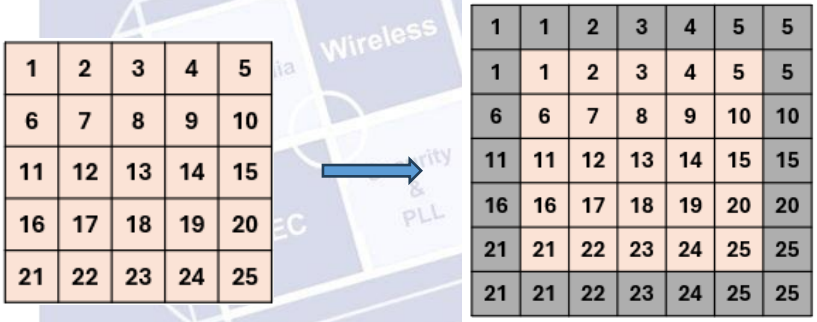


Fig 5. Replication Padding

- **Zero Padding**

  Zero padding involves adding zeros around the borders of an image or signal before applying certain operations, such as convolutions or Fourier transforms.

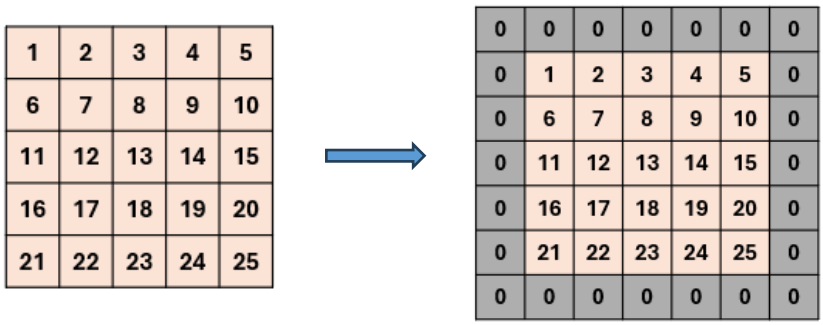
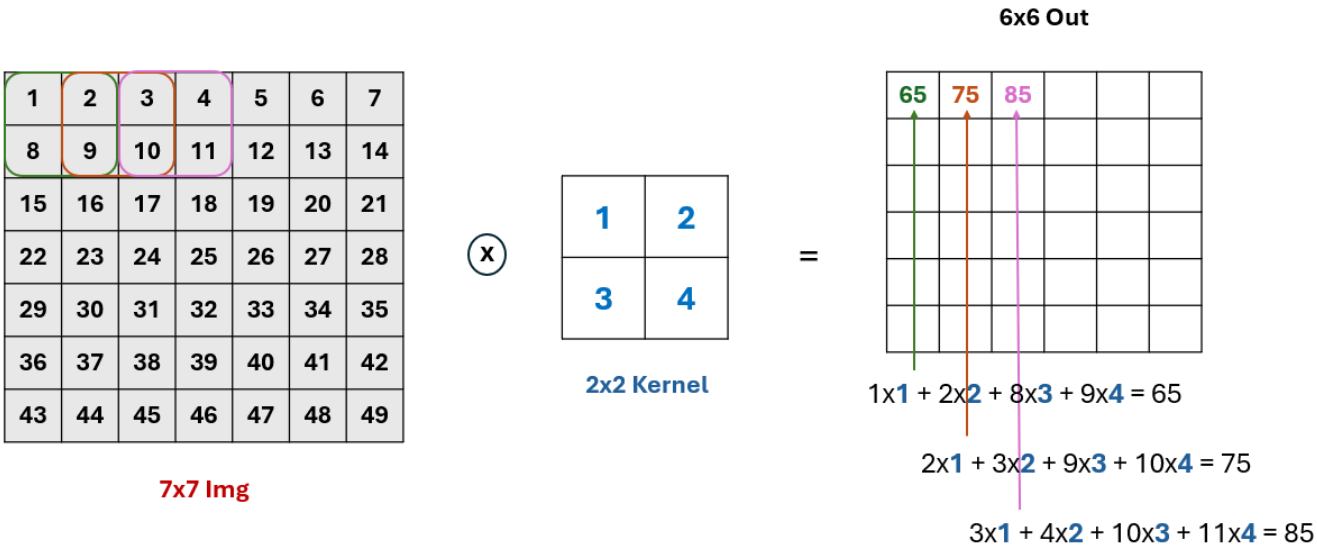


Fig 6. Zero Padding

– **Convolution**

Formula of convolution:

$$Out[\mathrm{m},\mathrm{n}] = \sum_j \sum_i Img[m,n] \cdot Kernel[m-i, n-j]$$



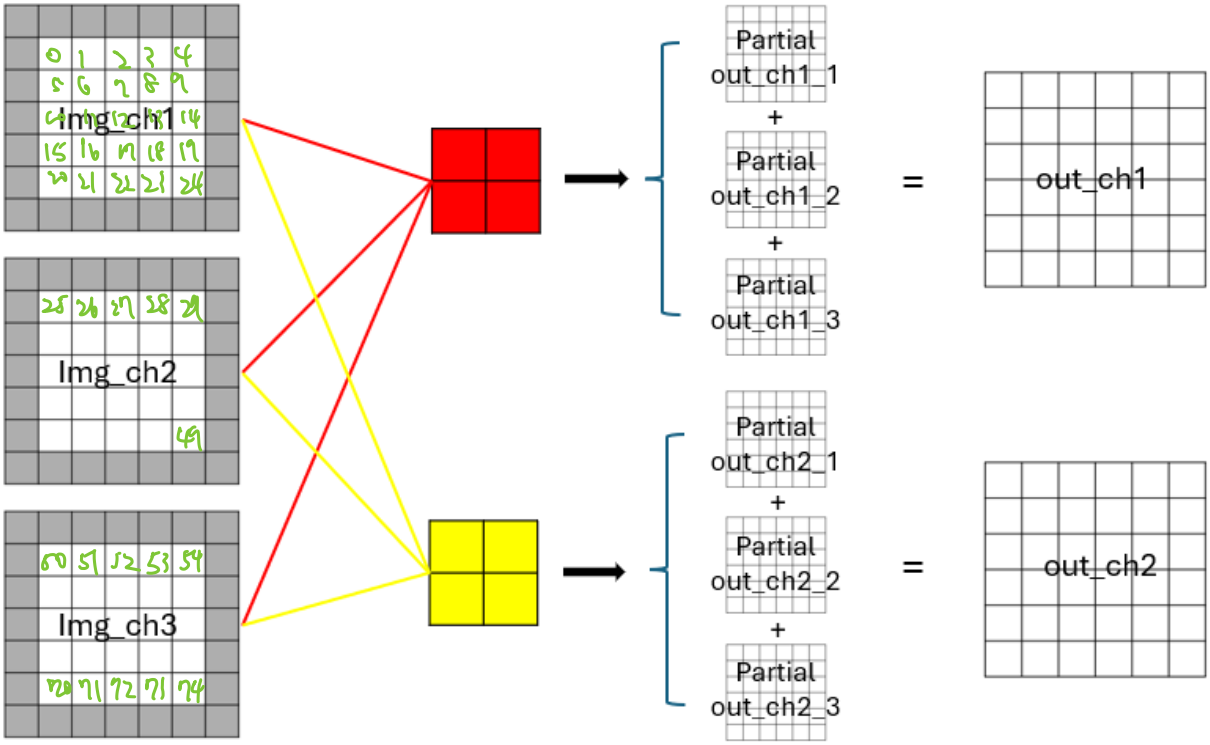


Fig 7. Example of convolution operation

– **Max-Pooling**

The max-pooling operation works by sliding a 3x3 window, over the input feature map and taking the maximum value in each window as output.

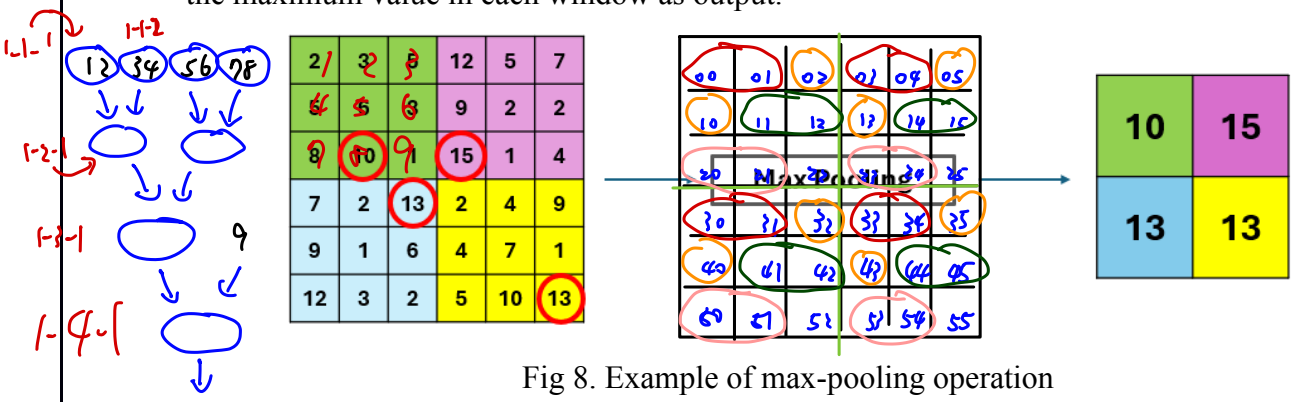


Fig 8. Example of max-pooling operation

– **Activation Function**

An activation applies a non-linear transformation to the result of CNN.

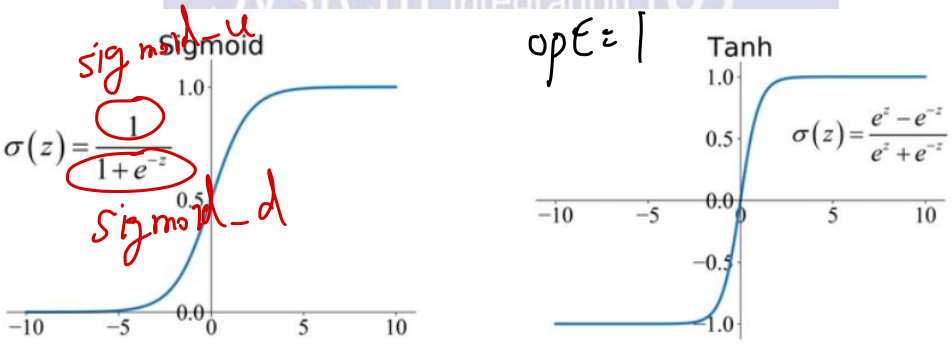


Fig 9. activation function figure

– **Fully Connected**

A fully connected layer can be represented as a matrix multiplication operation between the input matrix and weight matrix. Flattening the output matrix in obtaining a feature map

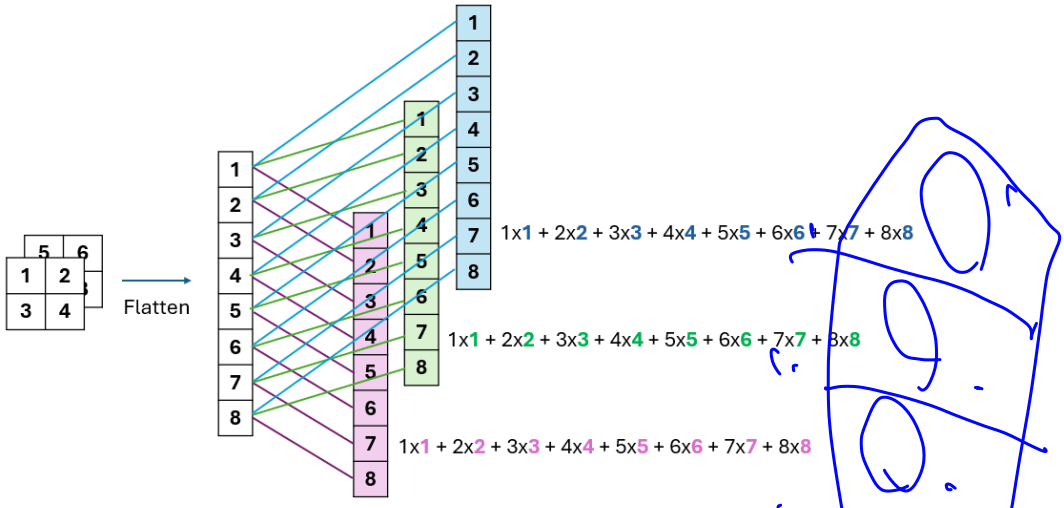


Fig 10. Example of fully connected layer

1.155 + 0.576 + 0.40

IDLE → IN → counter

counter = 113

- **Softmax**

  Softmax applies an exponential function to each score and normalizes them, representing the probability of a class being the correct one.

  For a vector $\mathbf{z} = [z_1, z_2, \ldots, z_{n-1}, z_n]$, the Softmax function for the i-th element is defined as:

  $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$, where $\sigma(z)_i$ is the probability of the i-th element.

z_1	5.82	→ $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$ →	σ_1	0.79
z_2	3.14		σ_2	0.05
z_3	4.22		σ_3	0.16

Fig 11. Example of softmax layer

$\frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$

Inputs and Outputs

The following are the definitions of input signals

Input Signals	Bit Width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when all input is valid.
Img	32	The image signals which sent in raster order. The arithmetic representation follows the IEEE-754 floating number format. (Range: ∓ 0~0.5)
Kernel_ch1	32	The kernel signals which sent in raster order. The arithmetic representation follows the IEEE-754 floating number format. (Range: ∓ 0~0.5)
Kernel_ch2	32	The kernel signals which sent in raster order. The arithmetic representation follows the IEEE-754 floating number format. (Range: ∓ 0~0.5)

Weight	32	The weight signals which sent in raster order. The arithmetic representation follows the IEEE-754 floating number format. (Range: ∓ 0~0.5)
Opt	1	1'b0 : sigmoid & {Zero} 1'b1 : tanh & {Replication}

The following are the definitions of output signals

Output Signals	**Bit Width**	**Definition**
out_valid	1	High when out is valid.
out	32	The output signals of image. The arithmetic representation follows the IEEE-754 floating number format.

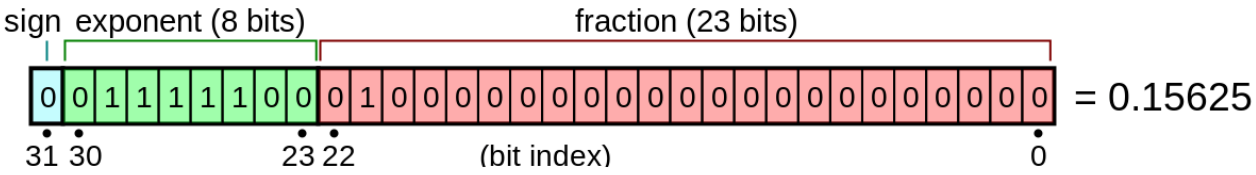
Each time you output the result, our pattern will check the correctness of it. Basically, if you follow the formulas and use IEEE floating point number IP, you should get same result as our answer. **However, we release the constraint; you may have an error under 0.005 for the result after converting to float number. This means that we will convert your output from binary format into real float number, and compare with our answer. Error will be calculated by '(golden-ans)/golden' and get its absolute value. If the error is higher than the value, you will fail this lab.**

Ex:

Binary form: 00111101101100101010000001000101

IEEE floating number: 0.08721975

nWave:



$$\text{sign} = +1$$

$$\text{exponent} = (-127) + 124 = -3$$

$$\text{fraction} = 1 + 2^{-2} = 1.25$$

$$\text{value} = (+1) \times 1.25 \times 2^{-3} = +0.15625$$

128 64 32 16 8 4 2 1

nWave will round the number for display, but the computation will not be affected. The following numbers are all from nWave, thus the computations are performed by IPs in Verilog.

※ e-02 = 10^{-2}

1. The input signal **Img** is delivered in raster scan order for **75 cycles** continuously. When **in_valid** is low, input is tied to unknown state.
2. The input signal **Kernel** is delivered in raster scan order for **12 cycles** continuously.
3. The input signal **Weight** is delivered in raster scan order for **24 cycles** continuously.
4. The input signal **Opt** is delivered for **only 1 cycle during the first cycle of in_valid tied high**. After 1 cycle, input is tied to unknown state.
5. All input signals are synchronized at negative edge of the clock.
6. The output signal **out** must be delivered for **3 cycle**, and **out_valid** should be **high** simultaneously.
7. The **out** signal should be **zero** when **out_valid** is low.
8. The **out_valid** cannot overlap with **in_valid** at any time.
9. **Please follow the parameter TA set, or you might fail in this lab.**
10. **You don't need to worry about infinity in the calculation process.**

Specifications

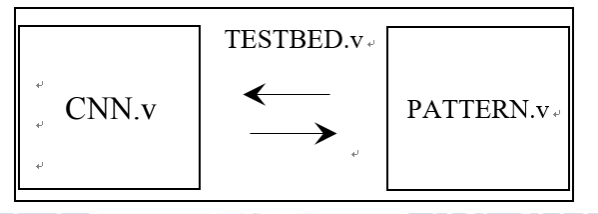
1. Top module name: CNN (design file name: CNN.v)
2. **You have to check an error under 0.0001 for the result after converting to float number. If the error is higher than the value, you will fail this lab.**
3. **It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock starting) in your design, you may fail to reset signals.**
4. 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.
5. The **out** should be reset after your **out_valid** is pulled down.
6. The execution latency is limited in **200 cycles**. The latency is the clock cycles between the falling edge of the **in_valid** and the rising edge of the first **out_valid**.
7. The area is limited in **5000000**. Also, the synthesis time should be less than **3 hours**.
8. You can adjust your clock period by yourself, but the maximum period is **50 ns**. The precision of clock period is 0.1, for example, 4.5 is allowed, 4.55 is not allowed.
9. The input delay is set to **0.5*(clock period)**.
10. The output delay is set to **0.5*(clock period)**, and the output loading is set to **0.05**.
11. The synthesis result of data type **cannot** include any **latches**.
12. After synthesis, you can check CNN.area and CNN.timing. The area report is valid when the slack in the end of timing report should be **non-negative (MET)**.
13. **In this lab, you must use at least one IEEE floating point number IP from Designware. We will check it at CNN.resource in 02_SYN/Report/. The example shows in following figure.**

```
============================================================
=======
| Cell        | Module       | Parameters | Contained Operations      |
============================================================
=======
| add_x_5     | DW01_inc     | width=4    | add_194 (CNN.v:194)       |
| add_x_6     | DW01_inc     | width=11   | add_207 (CNN.v:207)       |
| lt_x_7      | DW_cmp       | width=11   | lt_230 (CNN.v:230)        |
| lt_x_8      | DW_cmp       | width=11   | lt_237 (CNN.v:237)        |
| M0          | DW_fp_mult   | sig_width=23 | M0 (CNN.v:1741)         |
|             |              | exp_width=8 |                          |
|             |              | ieee_compliance=0 |                     |
| M1          | DW_fp_mult   | sig_width=23 | M1 (CNN.v:1742)         |
|             |              | exp_width=8 |                          |
|             |              | ieee_compliance=0 |                     |
```

Grading Policy

1. Function Validity: 70%
2. Performance: 30 %
   - Area * Computation time: 30%
   - Computation time = Latency * clock cycle time

Block diagram



Note

1. **Please submit following files under 09_SUBMIT before 12:00 at noon on Oct. 7:**
   - CNN.v
   - If uploaded files violate the naming rule, you will get 5 deduct points.
   - In this lab, you can adjust your clock cycle time. Consequently, make sure to key in your clock cycle time when you submit. It's means that the TA will demo your design under this clock cycle time.
   - The 2nd demo deadline is **12:00 at noon on Oct. 9** .
   - Check whether there is any wire / reg / submodule name called "error", "fail", "pass", "congratulation", "latch", "DW_fp", if you used, you will fail the lab.

   -

2. **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/CNN**.timing**)

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

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

```
`define CYCLE_TIME      50.0
`define SEED_NUMBER     28825252
`define PATTERN_NUMBER 1000
```

```
#=========================
# (A) Global Parameters
#=========================
set DESIGN "CNN"
set CYCLE 50
```

Sample Waveform

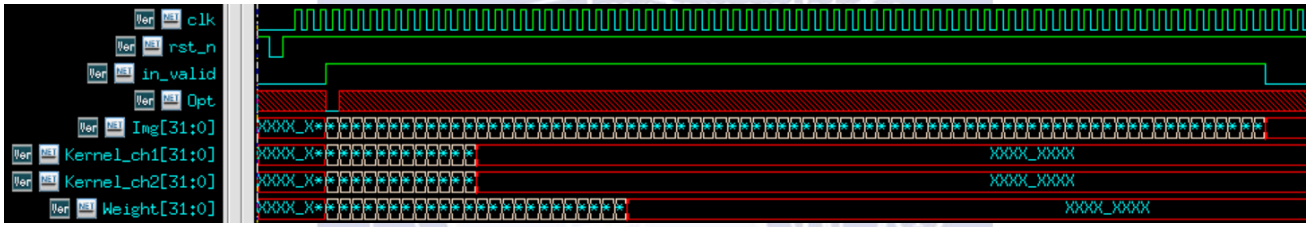


Fig1. Input waveform

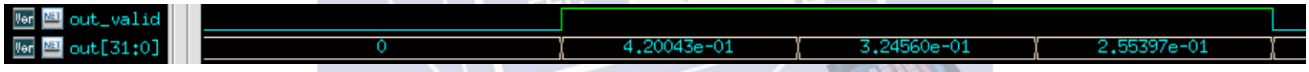


Fig2. Output waveform