NYCU-EE IC LAB – FALL 2023

Lab04 Exercise

Design: Siamese Neural Network

Data Preparation

1. Extract files from TA's directory:

   % openssl des3 -d -k zk5ZaSbq+yU= -salt -in ~iclabTA01/Lab04.tar | tar xvf -

2. The extracted LAB directory contains:
   a. **00_TESTBED**
   b. **01_RTL**
   c. **02_SYN**
   d. **03_GATE**
   e. **09_SUBMIT**

Design Description

The ***Siamese neural network*** is a type of neural network architecture designed for similarity learning and feature extraction tasks. It is called "Siamese" because the network consists of two identical subnetworks, known as twin networks, which share the same architecture and parameters. These twin networks are used to process two different input samples.

The primary application of ***Siamese neural network*** is in tasks that involve determining similarity or dissimilarity between two input samples. For example, Siamese networks are commonly used in tasks like face recognition, signature verification, one-shot learning, and similarity-based recommender systems.

In this lab, you are asked to design a ***Siamese Neural Network*** accelerator Fig 1. You have to take two Convolution Neural Networks (CNN) with identical structures and weights as sub-networks and concatenate them together to form a ***Siamese Neural Network***. The CNN sub-network includes operations such as **convolution, max-pooling, fully connected,** and **normalization**. The features computed by the two sub-networks are passed through an activation function to produce image encodings, which are then compared using the L1 distance to calculate the similarity score.

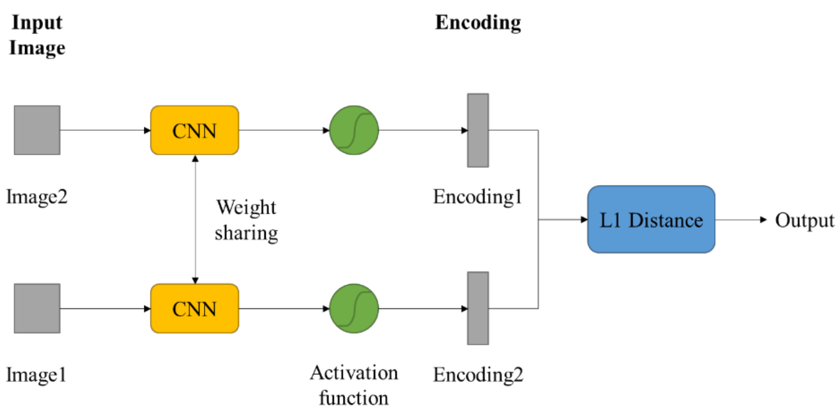


Fig 1. Siamese Neural Network architecture

- <u>Description of input signals</u>

When **in_valid** is high, the 32-bit **Img** signals will receive **4x4x3x2 = 96 cycles** continuously to represent 2 input 4x4x3 images.

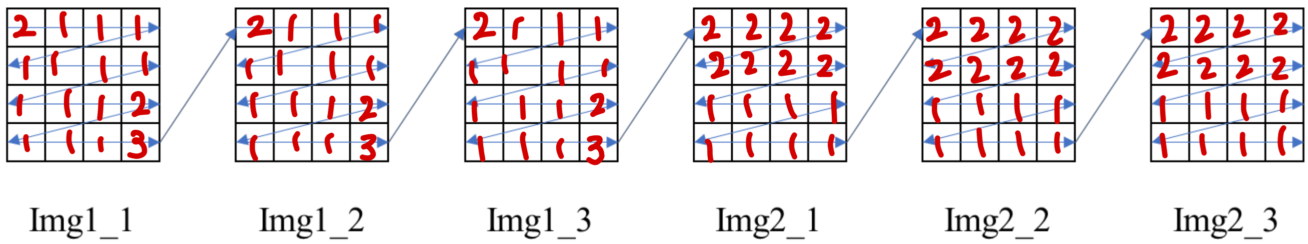
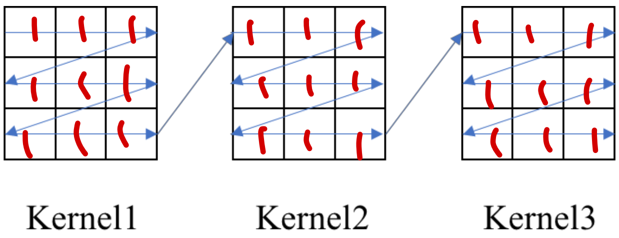


Fig 2. Sending order of the Img signal

The 32-bit unsigned **Kernel** signal will receive **3x3x3 = 27 cycles** continuously to represent the 3x3x3 kernel.



45 45

36 36

Fig 3. Sending order of the Kernel signal

The 32-bit **Weight** signal will also receive **4 cycles** continuously to represent the 2x2 matrix for the weight of the fully connected layer.

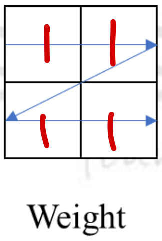


Fig 4. Sending order of the Weight signal

24	36	36	24
30	45	45	30
24	36	36	24
12	18	18	12

When the 2 images have been provided, i.e., after 96 cycles, **in_valid** will be pulled low. Note that the input signals **Img, Kernel**, and **Weight** are all sent in raster scan order.

zero padding

15	21	18	12
21	30	30	21
18	27	36	27
12	18	27	21

0	0	0	0	0	0
0	2	1	1	1	0
0	1	1	1	1	0
0	1	1	1	2	0
0	1	1	1	3	0
0	0	0	0	0	0

5	7	6	4
7	10	10	7
6	9	12	9
4	6	9	7

72

84

39 33 1 1

27 57 1 1

- <u>Description of CNN sub-networks</u>

Before doing the convolution, you must perform **Replication Padding or Zero Padding** according to the information given by Opt. The padding width is 1.

The input of the CNN sub-network is a 32-bit 6x6x3 image. **Img1_1, Img1_2**, **Img1_3** will be fed into the upper sub-network, while **Img2_1, Img2_2, and Img2_3** will be fed into the lower sub-network. First, computing the convolution of the image with the kernel will result in a 4x4x1 feature map. Then, the 32-bit 4x4x1 feature map will do the max pooling operation, resulting in a 2x2x1 feature map. The next step is to input the 2x2 feature map to a fully connected layer, resulting in a 4x1 feature map. In this step, the 2x2 feature map will be multiplied by a 2x2 weight matrix, where **Weight** represents the matrix elements and will be continuously provided for 4 cycles during the input stage. The resulting matrix will be flattened into a 4x1 feature map. And then perform normalization.

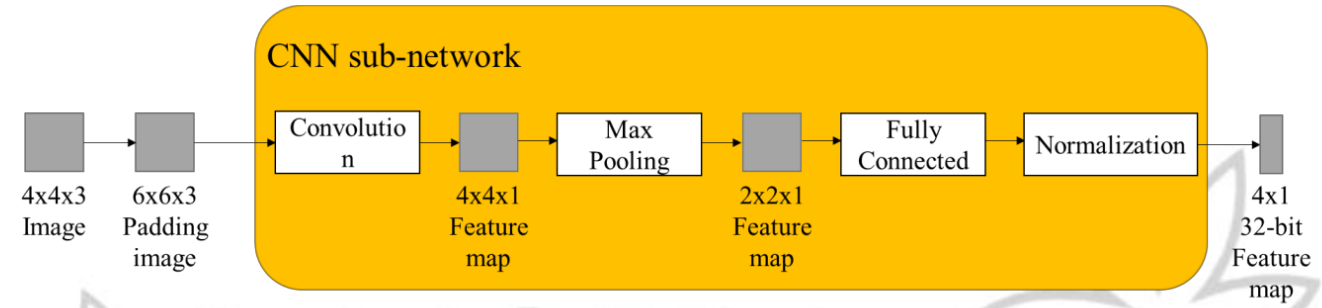


Fig 5. CNN sub-network

- <u>Description of connection of two sub-networks</u>

After computing the two sub-networks, the result is then passed through an **activation function.** The result of the activation function is called encoding vector. Next, the encoding vector passed through L1 distance calculation to obtain the **out** signal.

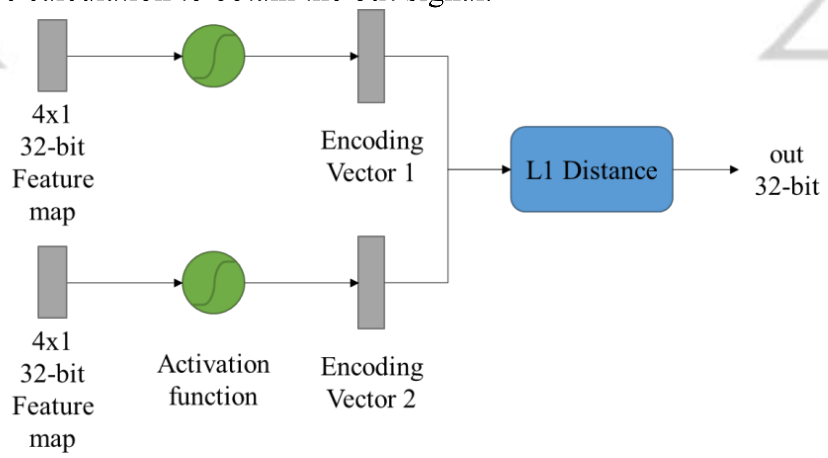


Fig6. Connection of two sub-network

- **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.

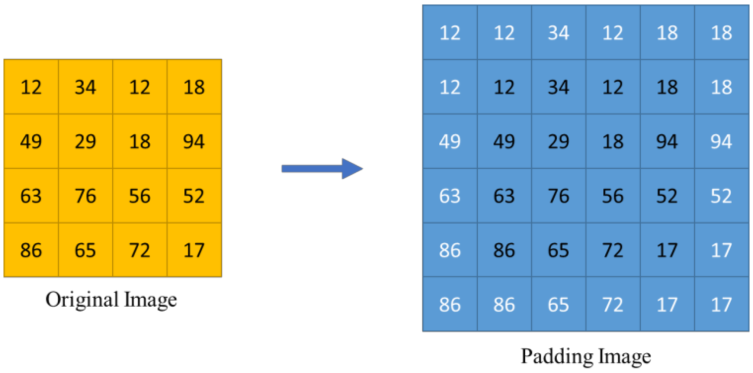


Fig7. Replication Padding

- **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.

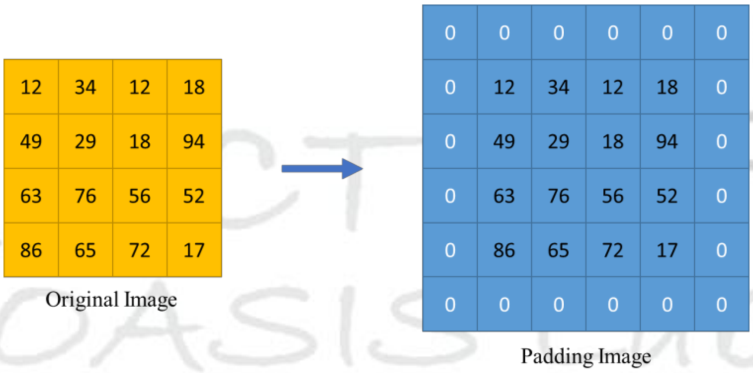


Fig8. Zero Padding

- **Convolution**

Formula of convolution:

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

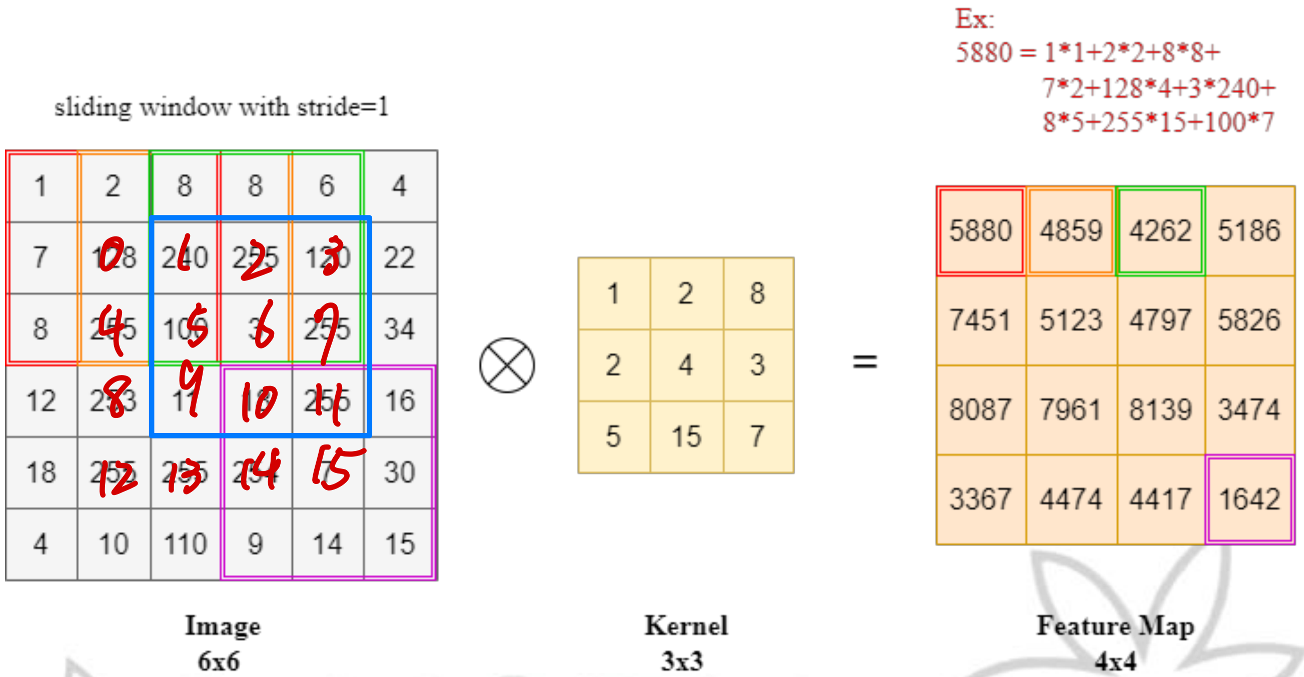


Fig9. Example of convolution operation

- **Max-Pooling**

The max-pooling operation works by sliding a 2x2 window, over the input feature map and taking the maximum value in each window as output.

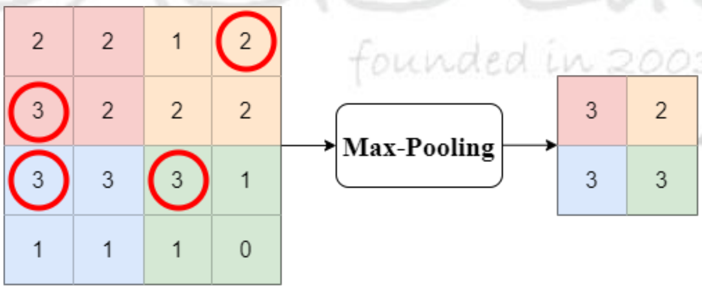


Fig10. Example of max-pooling operation

315
60

- **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.

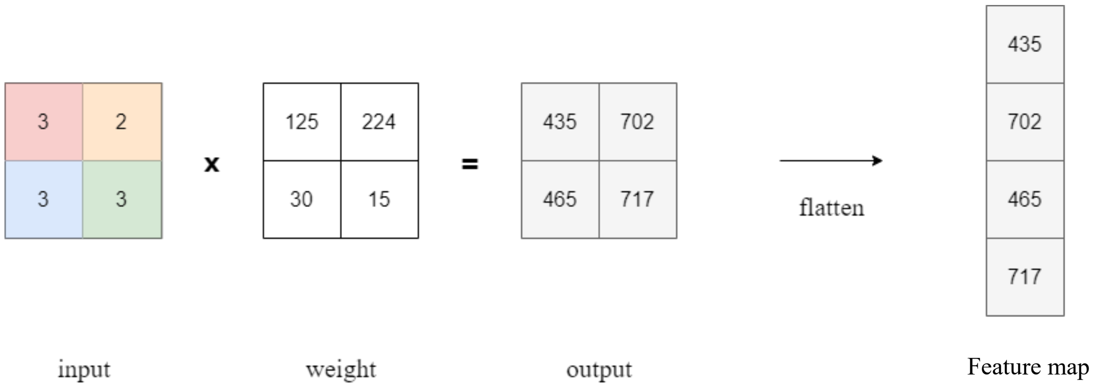


Fig11. Example of fully connected layer

- **Min-Max Normalization**

Min-Max Normalization is a data processing technique used to transform numeric features to a specific range, typically between 0 and 1.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- **Activation Function**

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

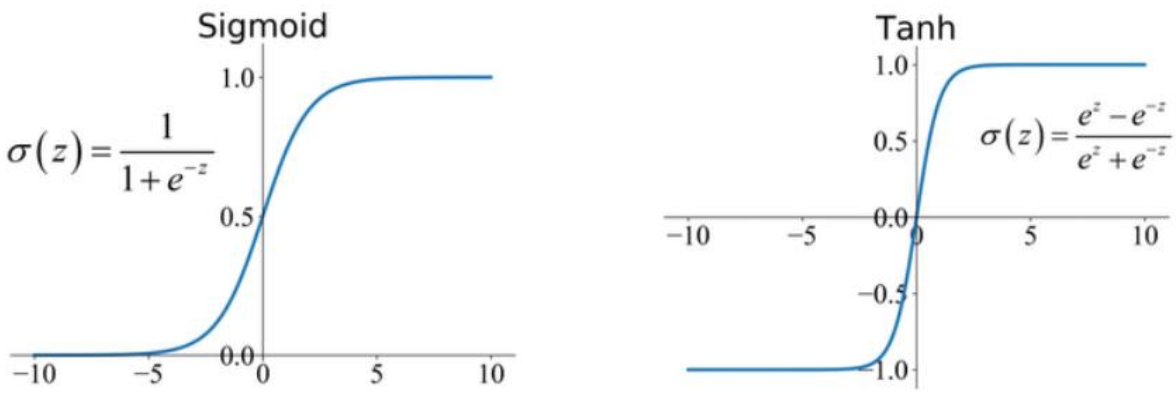


Fig12. Example of activation function

- **Encoding vector**

In this lab, an encoding vector refers to a mathematical vector used to represent images of information. This vector captures essential features of the input data and is commonly employed in fields such as machine learning.

- **L1 distance**

  L1 distance also known as Manhattan distance. It is defined as the sum of the absolute differences between the corresponding coordinates of the two points.

  Formula of L1 distance

$$L1\ distance = \sum_{1}^{n} |p_i - q_i|$$

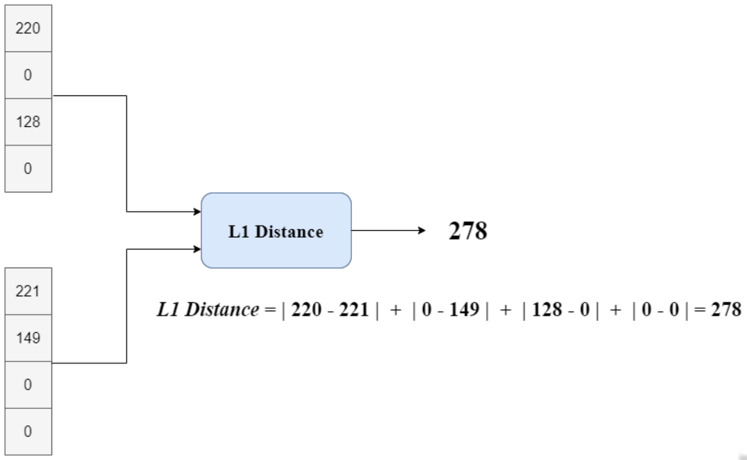


Fig10. Example of L1 distance

Inputs and Outputs

The following are the definitions of input signals

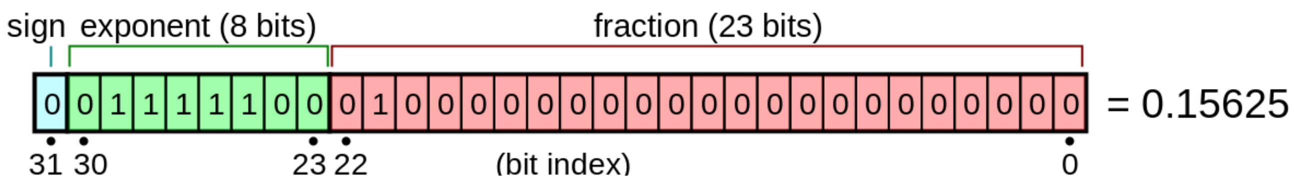
Input Signals	Bit Width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when all input is valid.
Img	32	The image signals which sent in raster ordering. The arithmetic representation follows the IEEE-754 floating number format. (Range: ∓ 0.5~255.0)
Kernel	32	The kernel signals which sent in raster ordering. The arithmetic representation follows the IEEE-754 floating number format. (Range: ∓ 0~0.5)
Weight	32	The weight signals which sent in raster ordering. The arithmetic representation follows the IEEE-754 floating number format. (Range: ∓

		0~0.5)
Opt	2	2'd0 : Sigmoid & {Replication} 2'd1 : Sigmoid & {Zero} 2'd2 : tanh & {Replication} 2'd3 : tanh & {Zero}

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 0.002 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.**



$$\text{sign} = +1$$
$$\text{exponent} = (-127) + 124 = -3$$
$$\text{fraction} = 1 + 2^{-2} = 1.25$$
$$\text{value} = (+1) \times 1.25 \times 2^{-3} = +0.15625$$

Binary form: 00111101101100101010000001000101

IEEE floating number: 0.08721975

nWave:

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.

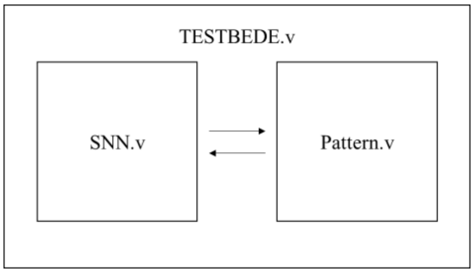
※ e-02 = 10^{-2}

1. The input signal **Img** is delivered in raster scan order for **96 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 **27 cycles** continuously.
3. The input signal **Weight** is delivered in raster scan order for **4 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 **only 1 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: SNN (File name: SNN.v)
2. **You have to check an error under 0.002 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 **1000 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 **5500000**. 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 SNN.area and SNN.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 SNN.resource in 02_SYN/Report/.**

Block Diagram



Grading Policy

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

Note

1. **Please submit following files under 09_SUBMIT before 12:00 at noon on October. 16:**
    - SNN.v
    - If uploaded files violate the naming rule, you will get 5 deduct points.
    - In this lab, you can adjust your clock cycle time.
    - The 2nd demo deadline is **12:00 at noon on October.18** .
    - 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/SNN**.timing**)

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

    09_SBMIT / (submit your files) **./00_tar**

    ./01_submit

    ./02_check

※You should make sure the two 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 "SNN"
set CYCLE 50
```

Sample Waveform

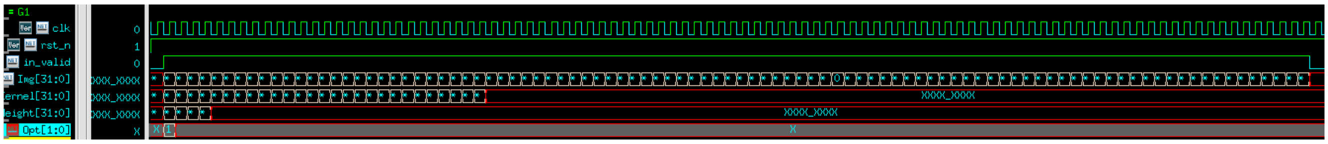
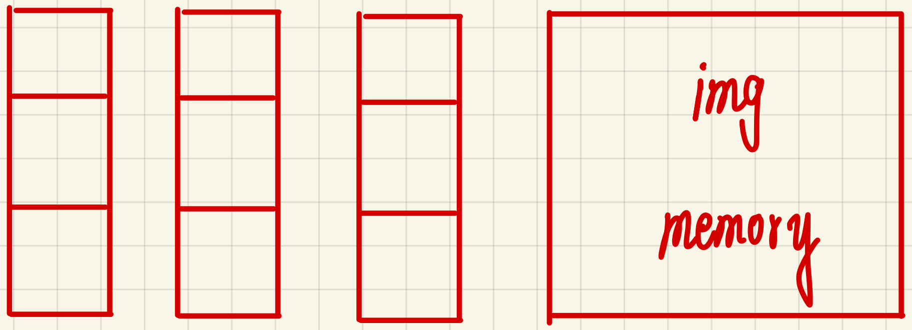


Fig1. Input waveform

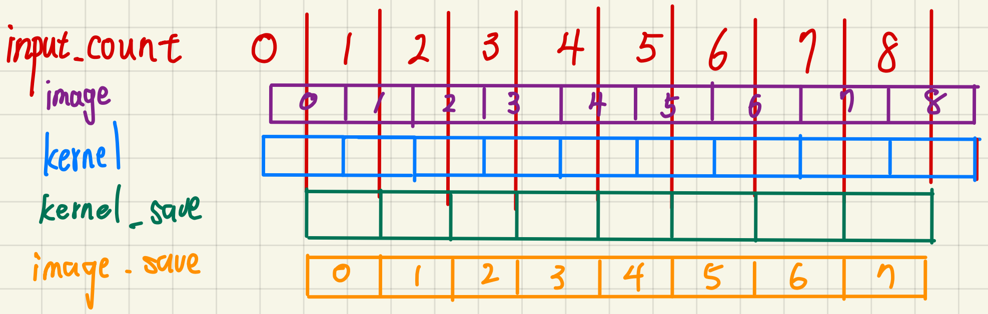
Fig2. Output waveform



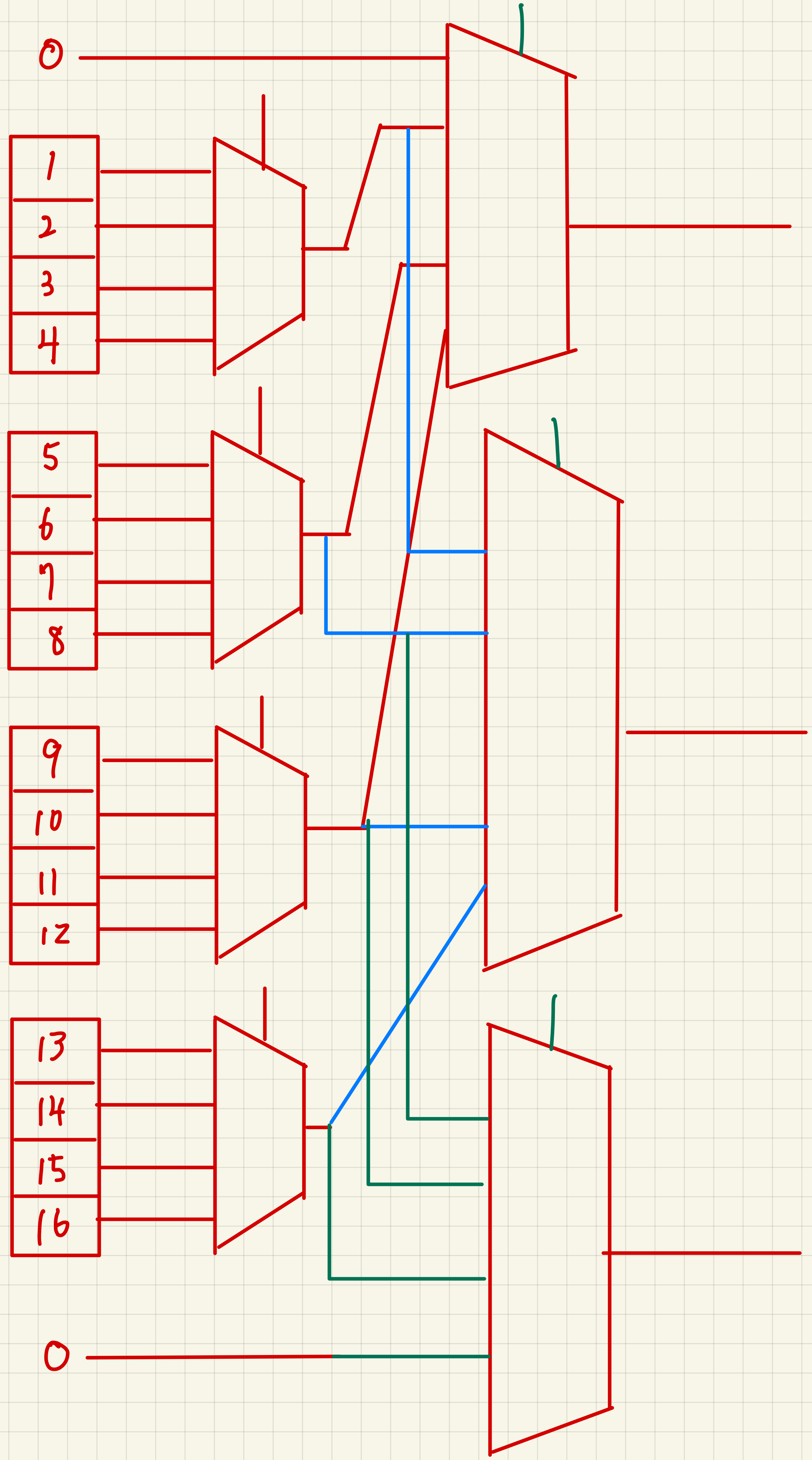
col % 4 = 0
1
2
3

row % 4 = 0
1
2
3

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 1 & 0 \\ 0 & 3 & 2 & 1 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 4 & 3 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$



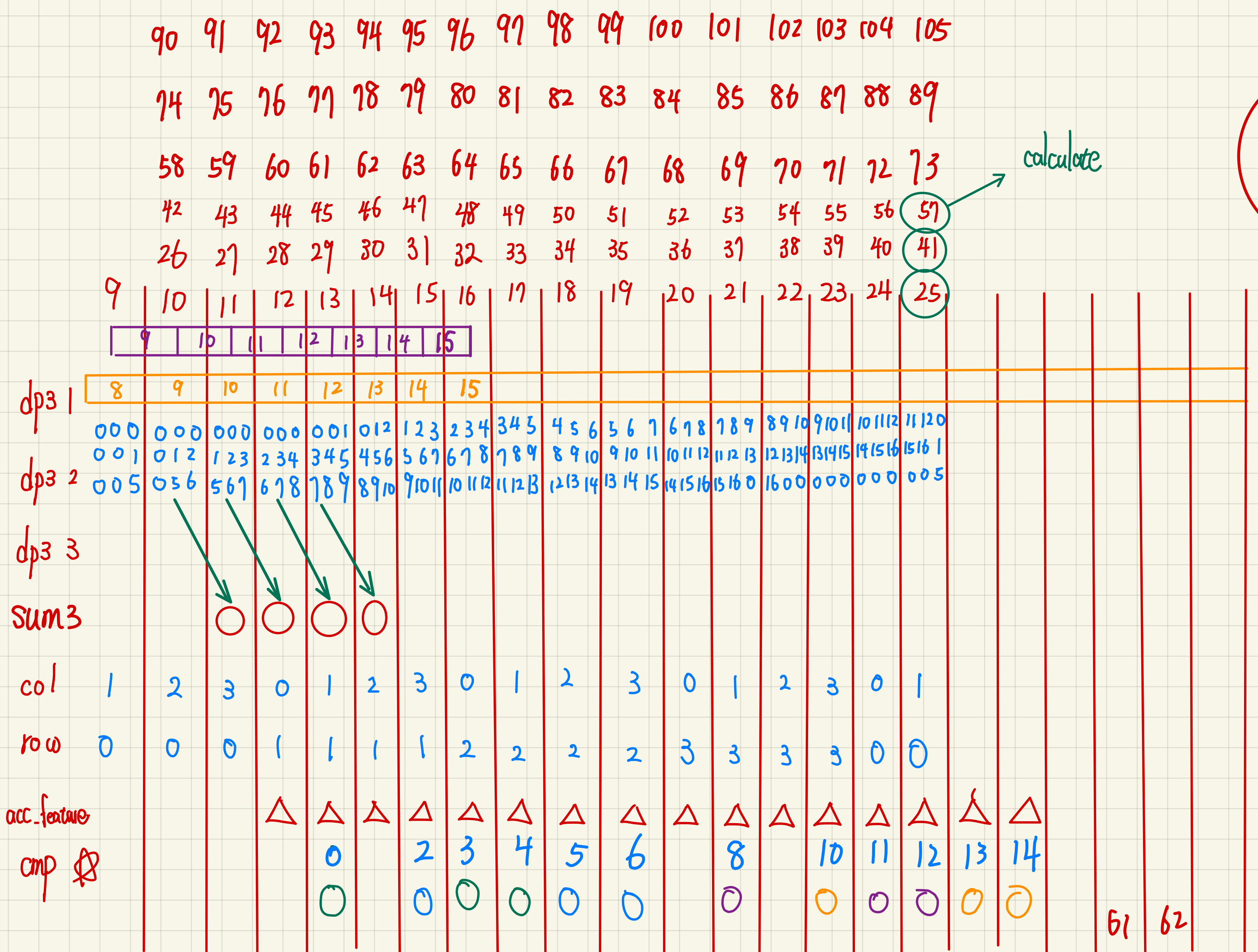
11 + 3 + 4 + 2

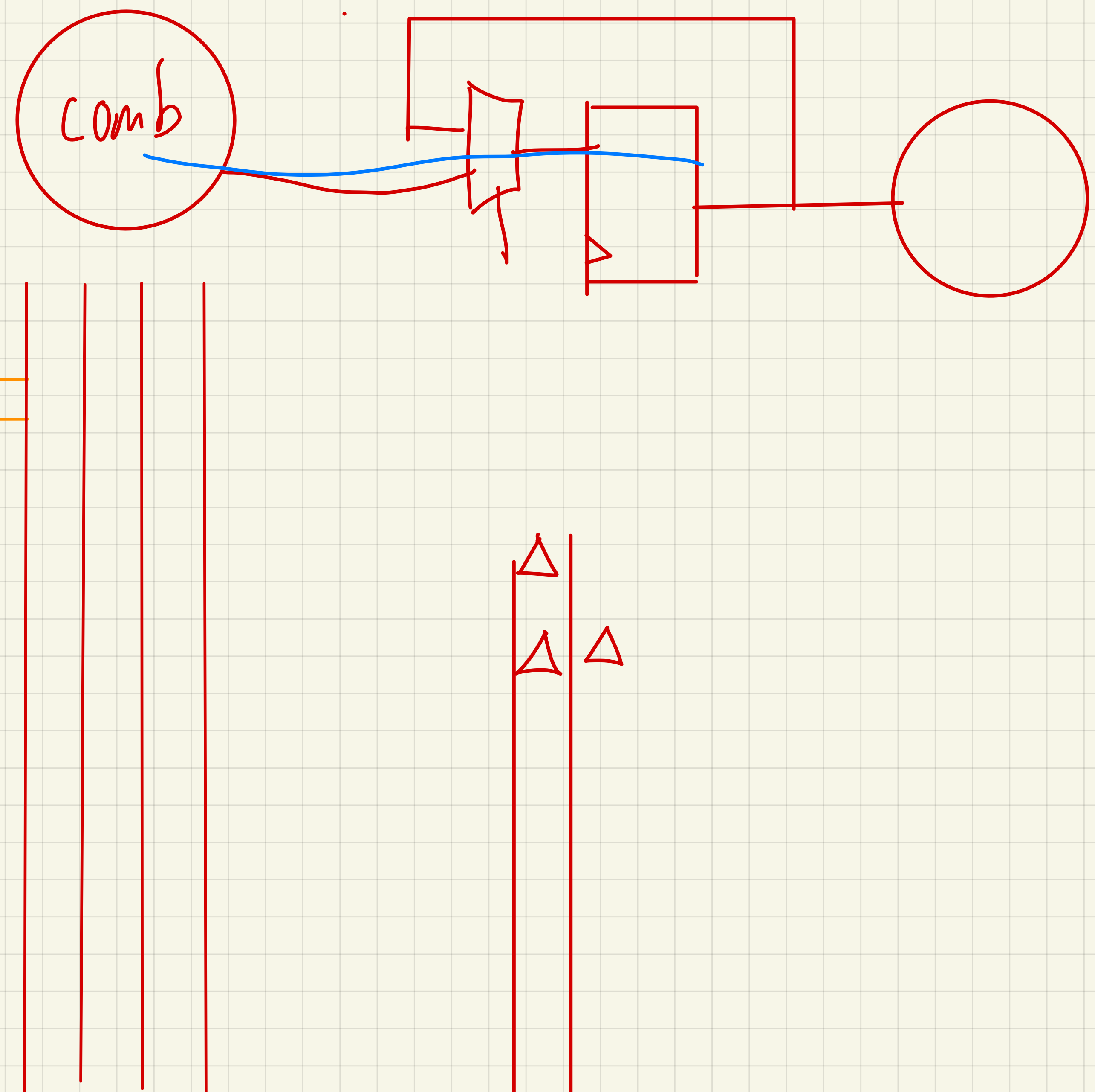


1 4 1
3 2 3
1 2 2

2 + 3 + 6 + 4 = 15

= 20

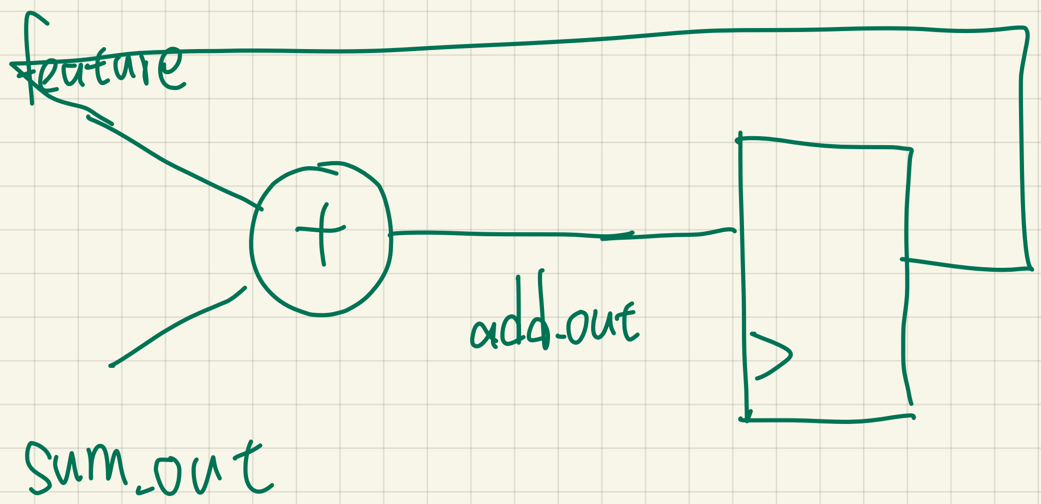




2x2 maxpool	get0		get1			get2		get3																
fully multi	0x0	0x1	1x2	1x3		2x0	2x1	3x2	3x3															
fully add		0x0 + 0	0x1 + 0	0x0 + 1x2	0x1 + 1x3		2x0 + 0	2x1 + 0	2x0 + 3x2	2x1 + 3x3														
1x4 fully				get0	get1				get2	get3														
cmp									16 max	17 max		64	65	66	67	68	69	70	71	72	73	74	75	
cmp △					9 0				16 min	17 min	18	19	20	21	22	23	24	25	26	27	28	29	30	31
norm_add											max − min	S_1 − min	S_2 − min	S_3 − min						△	△			
norm_div												0	1	2	3									
exp													0	1	2	3								
recip														0	1	2	3							
add											S_0 − min				$1+e^0$	$1+e^1$	$1+e^2$	$1+e^3$						
															e^0+e^0	e^1+e^1	e^2+e^2	e^3+e^3						
add															e^0-e^0	e^1-e^1	e^2-e^2	e^3-e^3						
tanh. or sig div																$\frac{e^0-e^0}{e^0+e^0}$	$\frac{e^0-e^0}{e^0+e^0}$	$\frac{e^0-e^0}{e^0+e^0}$	$\frac{e^0-e^0}{e^0+e^0}$					
																$\frac{1}{1+e^0}$	$\frac{1}{1+e^0}$	$\frac{1}{1+e^0}$	$\frac{1}{1+e^0}$					
get encoding vector																								
cmp △																								
add																				△				
out																					outreq			

$-7.25 + 6 - 7.5 = -8.75$

$2.5 - 13.75 - 16 =$



wei

0.5	-2
1	2

-8

-0.5 -4 -4 +1

ch1

0	0	0	0	0	0
0	-0.5	1	-4	-1	0
0	4	-4	2	-0.5	0
0	-2	-0.5	-0.5	2	0
0	0.5	-0.5	0.5	2	0
0	0	0	0	0	0

-2	-0.5	-0.5	0.5
-4	-1	2	4
0.5	-2	2	4
-4	2	1	-2

-1	-2	-1	4
4	2	2	2
0.5	1	2	-2
-1	-2	2	-4

2	2	4
2	0.5	-1
-2	-2	-0.5

-1	1	-2
4	-0.5	2
0.5	-1	-2

-0.5	1	2
-4	0.5	2
-4	-1	0.5