NYCU-DCS-2025

HW03

Design: Convolution Neural Network (CNN)

Data Preparation

1. Extract files from TA's directory:
   % tar xvf ~DCSTA01/HW03.tar
2. The extracted LAB directory contains:
   a. **00_TESTBED**
   b. **01_RTL**
   c. **02_SYN**
   d. **03_GATE**
   e. **09_SUBMIT**

Design Description

Convolutional Neural Networks (CNNs) are commonly used in computer vision and image processing. They are good at extracting features in image, which makes them useful for tasks such as image classification and object detection.

For modern CNN architectures, each convolution is followed by an activation function to introduce non-linearity and a pooling layer to reduce the size of the data. These components together form a conv block, which is typically followed by fully connected layers to produce the final probability distribution.

In this homework, you are required to design a conv block, which includes a Convolution Layer, an Activation Function, and a Max-Pooling Layer. The conv block will process **two images**, each with **two channels**. After passing through the conv block, the outputs of the two images must be flattened and concatenated. The concatenated result will then be quantized into 8-bit values before being fed into a Fully Connected Layer

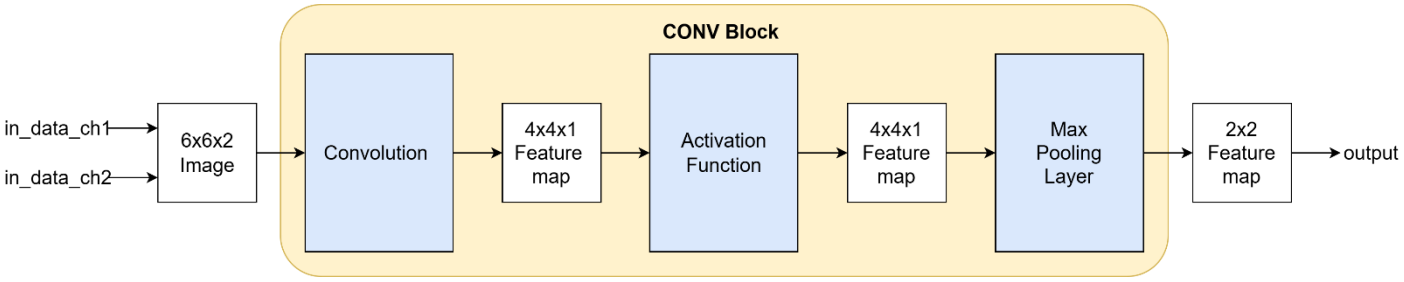


Fig. 1. The Convolutional Neural Network (CNN) architecture

Functional Description

System Flow

Step 1. Convolution

- Input image size: 6x6x2
- Kernel size: 3x3x2
- Output feature map: 4x4x1
- Stride = 1, No padding

Step 2. Activation Function

- ReLU
- Absolute

Step 3. Max-Pooling

- Stride = 2, window size = 2
- Input size: 4x4
- Output size: 2x2

Step 4. Flatten

- Input size: 2x2
- Output size: 4x1

Step 5. Quantization

- Input N-bit signed number
- Output 8-bit signed number

Step 6. Feature Concatenation

- Input size: (4x1) x 2
- Output size: 8x1

Step 7. Fully Connected Layer

- Input size: 8x1
- Weight size: 4x8

  Output size: 4x1

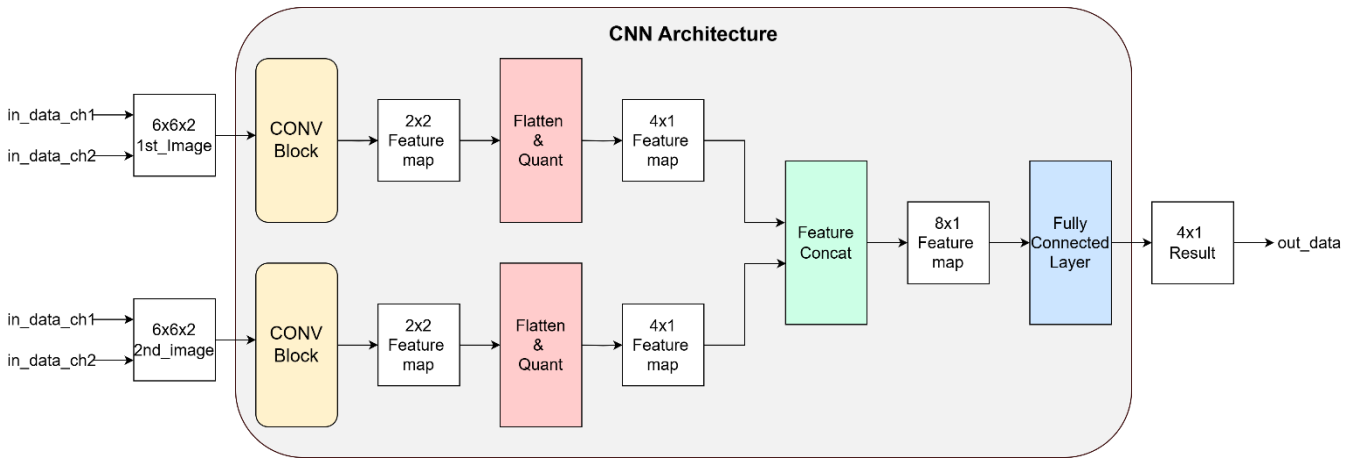


Figure 2. Overall system

◆ **Convolution**

The convolutional formula:

$$\text{FeatureMap}[x,y] = \sum_k \sum_j \sum_i Image[h+i, w+j, k] * Kernel\_c[i,j,k]$$

where h is height, w is width , and k represents channel

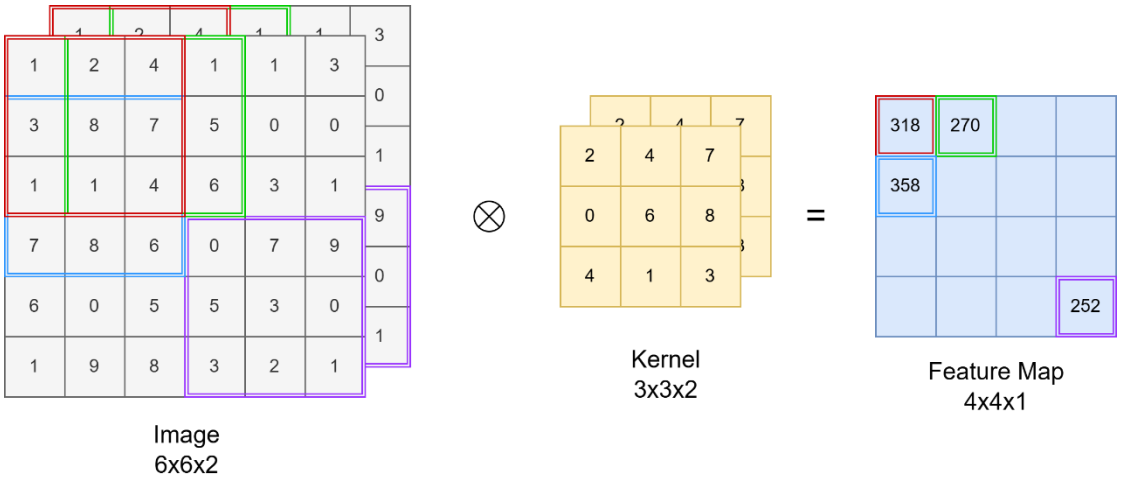


Figure 3. The example of convolution operation

The input image and kernel are sent in raster scan order as follows:

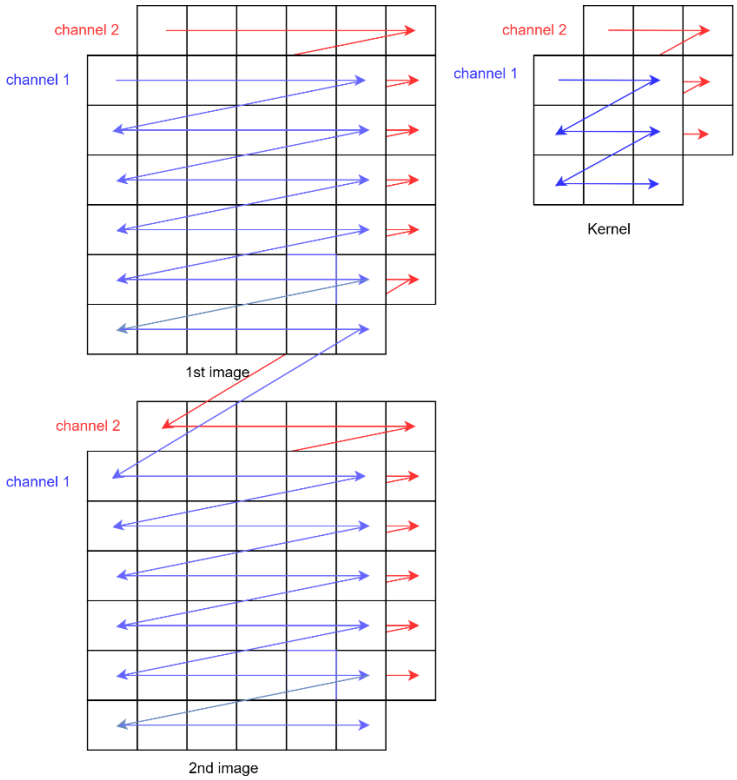
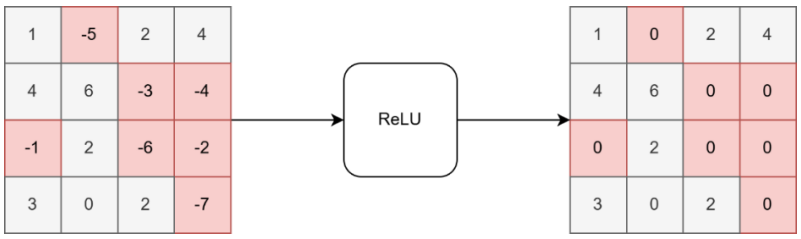


Figure 4. The example of input image and kernel order

◆ **Activation Function**

This design applies one of two activation functions, selected by the input **mode.**

1. ReLU(x) = max (0, x)



2. $\text{Abs}(x) = \begin{cases} x, \text{if } x \geqq 0 \\ -x, \text{if } x < 0 \end{cases}$

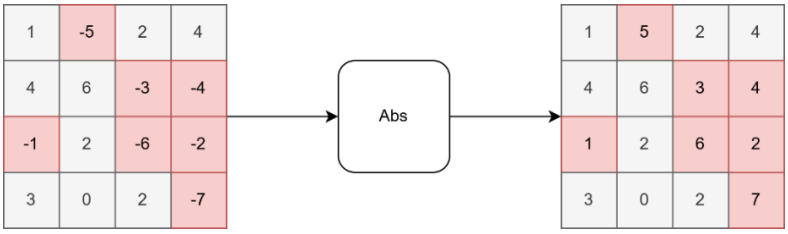


Figure 5. The example of activation function

◆ **Max-Pooling**

The max-pooling carries out with sliding a 2x2 window over the input, which takes the maximum value in each window as output.

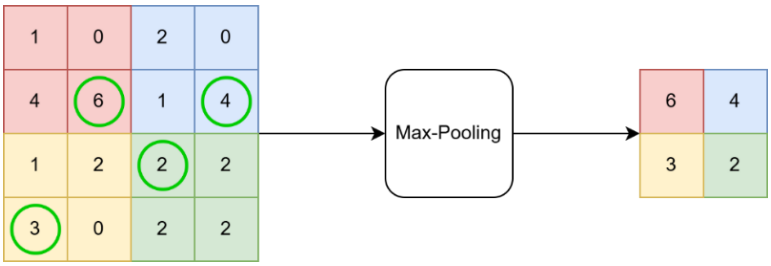


Figure 6. The example of max-pooling

◆ **Flatten**

Flattening reshapes the 2x2 output from the max pooling layer into a 4x1 vector in raster scan order.

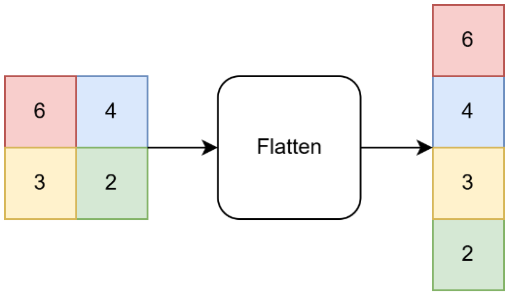


Figure 7. The example of Flatten

◆ **Quantization**

This flattened output is passed into the quantization module, where each element is quantized to **8 bits**. You need to map the input to the output using intervals of 512, with each interval corresponding to a specific output value.

input	output
[-∞, -65025]	-128
-	-
[-512, -1]	-1
[0, 511]	0
[512, 1023]	1
[1024, 1535]	2
-	-
[64512, 65023]	126
[65024, ∞]	127

Table I. Quantized table

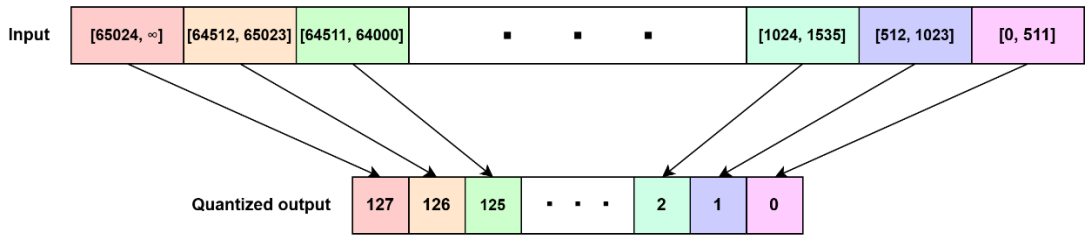


Figure 8. Quantization Mapping

◆ **Concatenation**

The first image and the second image are concatenated together.

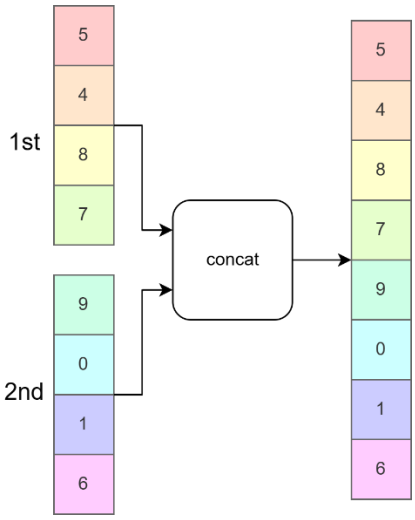


Figure 9. The example of Concatenation

◆ **Fully Connected Layer**

The fully connected layer formula:

$$y[j] = \sum_i w[i,j] * x[i]$$

where x[i] is input matrix, w[i, j] is weight matrix, y[j] is output matrix.

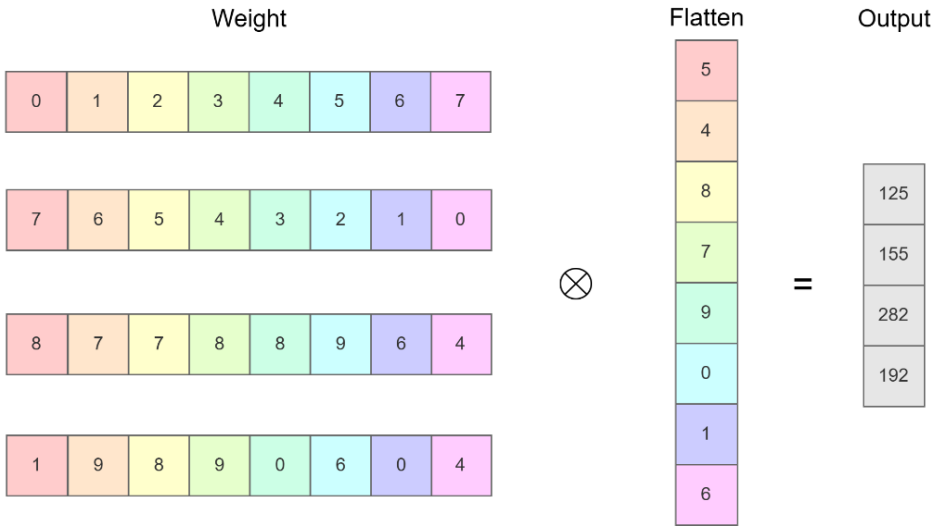


Figure 10. The example of fully connected operation

The input weights are sent in raster scan order as follows:

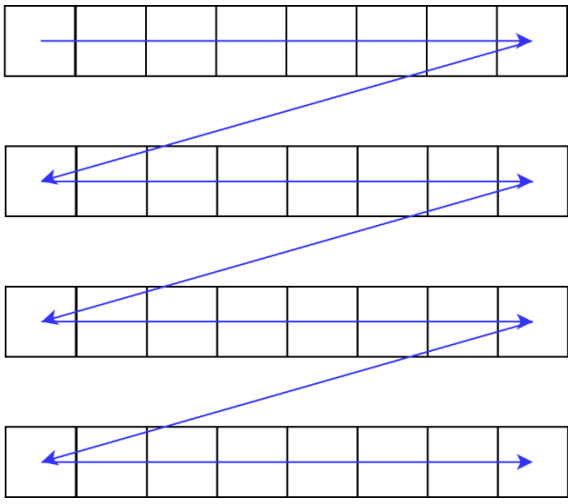


Figure 11. The example of input weight order

Input

Input	Bit	Sign	Definition and Description
clk	1	unsigned	Clock signal
rst_n	1	unsigned	Asynchronous active-low reset
in_valid	1	unsigned	High when input data signals are valid
mode	1	unsigned	The option signal for activation function 1'b0: ReLU 1'b1: Abs
in_data_ch1	8	signed	The input signals for image (channel 1) (6x6) x 2
in_data_ch2	8	signed	The input signals for image (channel 2) (6x6) x 2
kernel_ch1	8	signed	The input signals for CONV kernel (channel 1) (3x3)
kernel_ch2	8	signed	The input signals for CONV kernel (channel 2) (3x3)
weight	8	signed	The input signals for FC weight (4x8)

1. When **in_valid** is high, **in_data_ch1** and **in_data_ch2** are delivered with **(6x6)x2 cycles** continuously (the first 36 cycles are 1st_image data and second 36 cycles are 2nd_image data) then tied to unknown state.
2. When **in_valid** is high, **kernel_ch1** and **kernel_ch2** are delivered with **3x3 cycles** continuously then tied to unknown state.
3. When **in_valid** is high, **weight** are delivered with **4x8 cycles** continuously then tied to unknown state.
4. When **in_valid** is high, **mode** are delivered with only **1 cycles** then tied to unknown state.

Output

Output	Bit	Sign	Definition and Description
out_valid	1	unsigned	High when output signals are valid
out_data	20	signed	out_data is the final result of fully connected layer, which must be sent in 4 cycles continuously

1. The **out_valid** must be **low** with the **asynchronous reset** when rst_n is **low**.
2. The **out_data** must be **zero** with the **asynchronous reset** when rst_n is **low**.
3. The **out_data** must be delivered for **exact 4 consecutive cycles** and **out_valid** should be **high** simultaneously.
4. The **out_valid** **cannot** overlap with **in_valid** at any time.

SPEC

Top module

1. Top module name: CNN (File name: CNN.v)
2. The execution latency is limited in **1000 cycles** for single pattern. The latency is the clock cycles between the rising edge of the **in_valid** and the rising edge of the **out_valid.**

Reset

3. It is **asynchronous reset** and **active-low** architecture. If you use synchronous reset (considering reset after clock starting), 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 to 0 after the reset signal is asserted.**

in_valid & input signals

5. All input signals are synchronized at **negative edge of the clock**.
6. in_valid will come after reset.
7. The input of 2 images (**in_data_ch1** for channel 1 and **in_data_ch2** for channel 2), which are represented as **2's complement signed numbers**, are delivered in raster scan order for **(6x6)x2** cycles continuously when in_valid is tied high. When all elements of two images are delivered, it is tied to unknown state, and in_valid will also be tied low.
8. The input of **kernel** (**kernel_ch1** for channel 1 and **kernel _ch2** for channel 2), which are represented as **2's complement signed numbers**, are delivered in raster scan order for the first **3x3** cycles continuously when in_valid is tied high. When all elements of a kernel are delivered, it will be tied to unknown state.
9. The input of **weight**, which are represented as **2's complement signed numbers**, are delivered in the first **4x8** cycles continuously when in_valid is tied high. When all elements of a weight are delivered, it will be tied to unknown state.
10. The input of **mode** will be given in the first cycle when the **in_valid** is high. When mode is delivered, it will be tied to unknown state.
11. In each pattern, the **in_valid** will come in 2~4 negative edge of clock after **out_valid** falls.

out_valid & out_data

12. All output signals should be synchronized at **positive edge of clock**.
13. The TA's pattern will capture your output for checking at clock negative edge.
14. **out_valid cannot** overlap with **in_valid** at any time.
15. **out_data** should be correct when **out_valid** is high, and **out_data** should be low when **out_valid** is low.
16. **out_data** must be delivered for **4 cycles continuously** and **out_valid** should be high simultaneously.

Synthesis

17. You are **not** allowed to modify syn.tcl, except for cycle time (CYCLE).
18. You **can** adjust your clock period, but the maximum period is **20 ns**. The precision of clock period is 0.1, for example, 4.5 is allowed, 4.55 is not allowed.
19. The area is limited in **5,000,000**.
20. The synthesis time should be less than **2 hours**.
21. The synthesis result of data type **cannot** include any **latches** (using ctrl+F to find the term of "**Latch**" or using the command ./08_check in 02_SYN/).
22. After synthesis, you can check CNN.timing. The timing report should be **non-negative (MET)**.

Gate-level simulation

23. The gate-level simulation **cannot** include any **timing violations**.
24. The cycle time used in Gate-Level Simulation must be the **same** as the cycle time used in synthesis.

Supplement

25. In this lab, you are **NOT** allowed to use DesignWare IP.
26. Please check whether there is any wire/reg/submodule being named as "error", "fail", "pass", "congratulations", "latch". All letters used in the formats of uppercase or lowercase is prohibited. If there is, you will **fail** this homework.
27. Don't write Chinese or other language comments in the file you sent.
28. Any error messages during synthesis and simulation, regardless of the result will lead to failure in this lab.
29. The score is only based on the final version you submit to 09_SUBMIT in your last upload. Please ensure that you successfully upload the latest version.
30. It is strongly recommended to optimize the circuit by incorporating the **pipeline & parallel processing** methods you learned in class, as the performance calculation is cubic in terms of cycle time and square in terms of latency.

Block Diagram

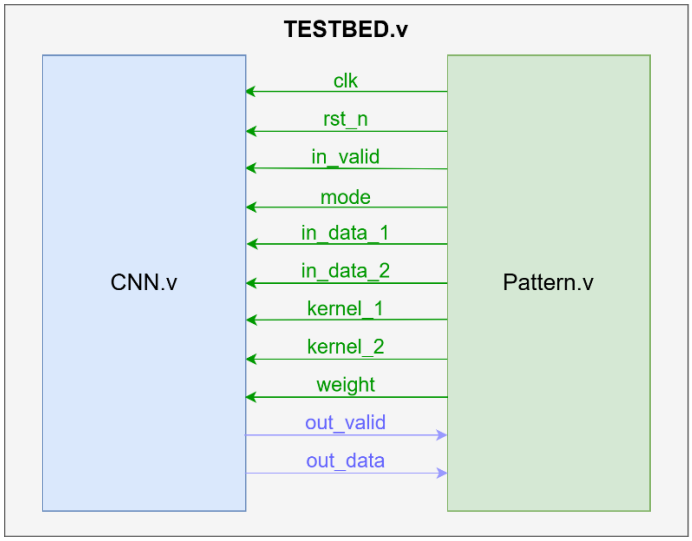


Figure 12. System Block Diagram

Homework Upload

1. Please use the commands we provided to tar whole file under 09_SUBMIT/ and to submit your homework
   - **1st_demo deadline : 2025/05/01 12:00:00**
   - **2nd_demo deadline : 2025/05/08 12:00:00**
2. Please check the homework file again after you upload it. If you upload the wrong file, you will **fail** this homework.
3. When TA demos your design, Random Seed definition will be changed in the PATTERN provided by the TA.

Grading Policy

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

The score is only based on the demo result you submit with 09_SUBMIT. Please ensure that you successfully upload the latest version.

1. Function Validity: 80%
   - RTL: 70%
   - SYN: 5%
   - GATE: 5%

2. Performance: 20%
   - **Performance = Area * Total_Latency^2 * Cycle_Time^3**

Command List

- Verilog RTL simulation (01_RTL/):
  - **./01_run_vcs_rtl**
  - **./08_check**
- Synthesis (02_SYN/):
  - **./01_run_dc_shell**
  - **./08_check**
- Gate level simulation (03_GATE/):
  - **./01_run_vcs_gate**
  - **./08_check**
- Submit your files (09_SUBMIT/):
  - **./00_tar XX.X**
    - XX.X should be replaced with the cycle time you use.
  - **./01_submit**
    - You must type '**y**' when you are ready to hand in your homework.
  - **./02_check**
- Waveform for debug:
  - **./05_nWave & or nWave &**
  - find *.fsdb
  - shift+L for reloading the *.fsdb file after you simulate your design again.

Reference Waveform

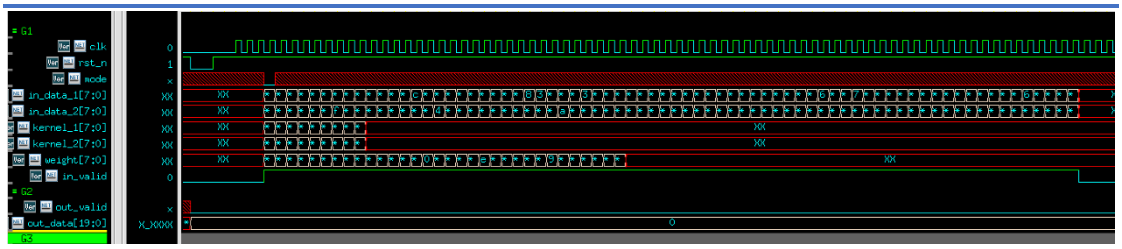


Figure 13. The example of input waveform

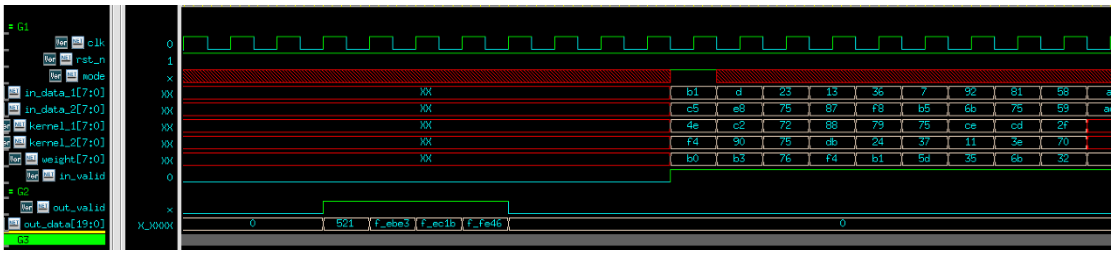


Figure 14. The example of output waveform

Debug

You can use "input.txt" and "output.txt" located in 00_TESTBED/debug/ to assist with debugging.

```
*Verdi* : End of traversing.
PASS PATTERN NO.    0,      Execution Cycle:  91
**********************************************************
                      FAIL!
 Expected:  =       14762,       30759,       -6045,      -22378
 Received:  =         978,       30759,       -6045,      -22378
**********************************************************
Input  data is written to ../00_TESTBED/debug/input.txt
Output data is written to ../00_TESTBED/debug/output.txt
$finish called from file "PATTERN.v", line 502.
$finish at simulation time              419000
           V C S   S i m u l a t i o n   R e p o r t
Time: 4190000 ps
```

input.txt

```
Pattern No.   1

1st image channel 1:
   -79     13     35     19     54      7
  -110   -127     88     10    104   -107
    13     90     11     17      7   -114
    81    -68     20    -11    -37    -61
  -126    -13    122     23    124     94
  -117    -47    104    -24     88   -115

1st image channel 2:
   -59    -24    117   -121     -8    -75
   107    117     89    -84    -65    -39
   113     37    -37    -41     37     41
   -75    118     19    -45     43     -2
   -13    -78    103     89    -12     34
    56    113     35    -77    123   -109

--------------------

2nd image channel 1:
   -35    108     83    -47     22     62
   -75     52     24     81   -119    -41
   -90      6     49     94    -66    -70
    65     83   -128    -98     94      8
    12   -101     25    120     95     53
   -44     34     56     98    -94    -16

2nd image channel 2:
   -56   -112     90     30   -102     25
   -36      7   -117     29   -116   -109
  -122    -37     11    -72    -99     53
  -106    124    -17   -113    -25    -37
  -120    -78     18    -94     62    -86
   103     83   -111    -90     81      4
```

output.txt

```
1st image ofmap result:
 22732 -15169  11731 -10383
 19611 -47851  22490 -22321
 -1505  20852   6429 -16629
 58893   5280  19008  -3225

2nd image ofmap result:
 40392 -10433 -38198 -30183
 -8850  28397 -37083 -23661
   673 -50718  -4978  31173
-39824   2481  28127  -5375

--------------------

1st image pool result:
 47851  22490
 58893  19008

2nd image pool result:
 40392  38198
 50718  31173

--------------------

Flatten & Quantization & Concatenation:
    93     43    115     37     78     74     99     60

--------------------

Final output:
 14762  30759  -6045 -22378

Your output:
   978  30759  -6045 -22378
```

Figure 15. Contents of the File Used for Debugging