Midterm : Convolutional Neural Network (CNN)

Instructor: Lih-Yih Chiou

Speaker: Justin

Date: 2024/04/18

Outline

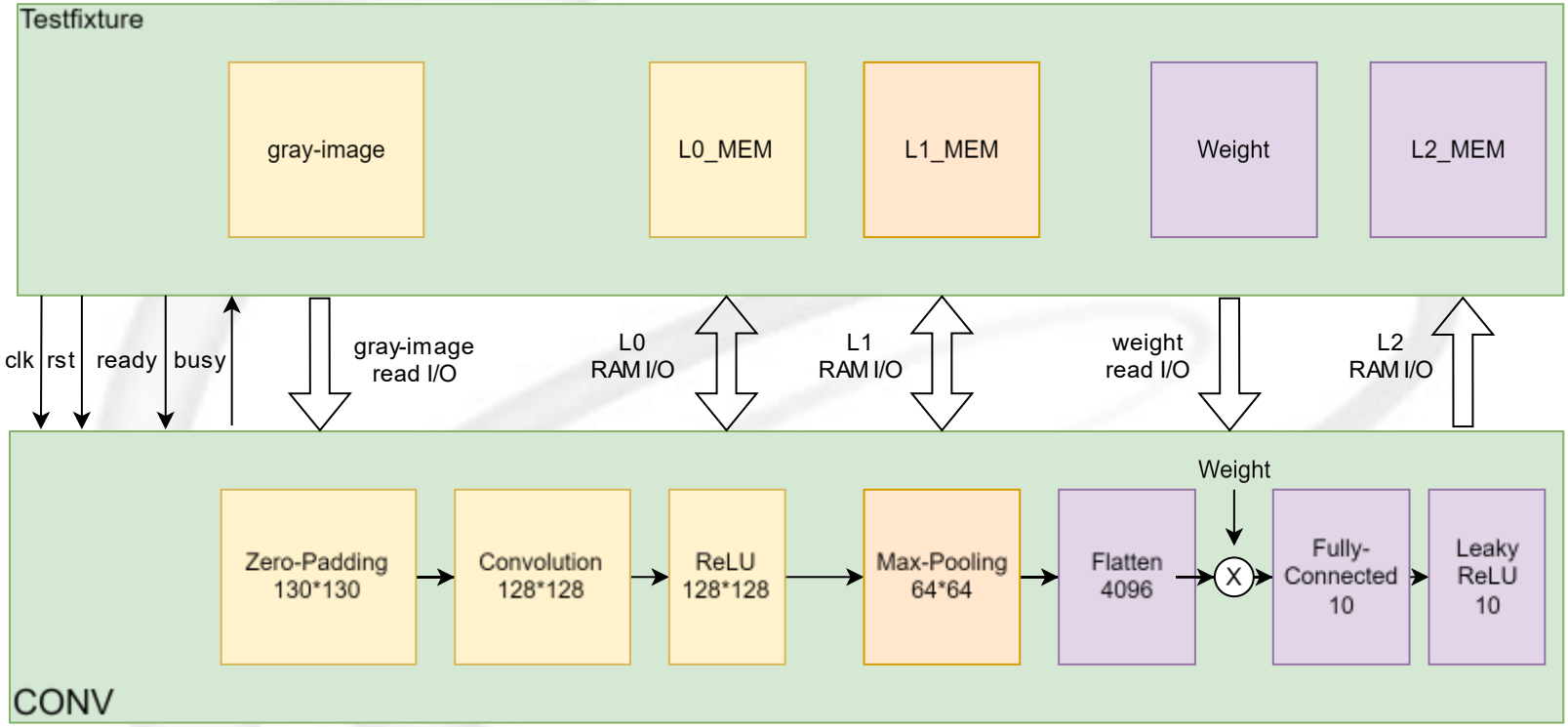
- Introduce to Convolutional Neural Network
- Hardware description
  - Block diagram
  - I/O Information
- Implementation
  - Convolution
  - ReLU
  - Max Pooling
  - Flatten
  - Fully Connected
  - Leaky ReLU
- Simulation Tips
- Criteria
  - Grading policy
  - Requirement & file format

Introduce to Convolution Neural Network

- Convolutional neural network (CNN) is a regularized type of **feed-forward** neural network that learns feature engineering by itself via **filters** (or kernel) optimization.
- Common CNN Layers:
  - Convolution Layer (w/ ,w/o padding)
  - Pooling Layer
  - Activation Function Layer
  - Batch Normalized Layer
  - Fully Connected Layer
  - Dropout Layer

Hardware description

Block Diagram



Hardware description

I/O Information

Signal	I/O	length	Desc.
clk	I	1	positive-edged triggered
rst	I	1	asynchronous positive-edged triggered
ready	I	1	enable signal to start processing
busy	O	1	System busy signal, active High, return Low to end Simulation
ioe	O	1	Active high output enable signal for gray-image ROM
iaddr	O	14	Address for gray-image ROM
idata	I	8	Read data from gray-image ROM, unsigned
addr_L0	O	14	Address for L0_RAM
wen_L0	O	1	Active high write enable signal for L0_RAM
w_data_L0	O	12	Write data from L0_RAM, unsigned
oe_L0	O	1	Active high output enable signal for L0_RAM
r_data_L0	I	12	Read data from L0_RAM, unsigned

Hardware description

I/O Information

Signal	I/O	length	Desc.
addr_L1	O	12	Address for L1_RAM
wen_L1	O	1	Active high write enable signal for L1_RAM
w_data_L1	O	12	Write data from L1_RAM, unsigned
oe_L1	O	1	Active high output enable signal for L1_RAM
r_data_L1	I	12	Read data from L1_RAM, unsigned
addr_weight	O	16	Address for weight_ROM
oe_weight	O	1	Active high output enable signal for weight_ROM
r_data_weight	I	8	Read data from weight_ROM, signed
addr_L2	O	3	Address for L2_RAM
wen_L2	O	1	Active high write enable signal for L2_RAM
w_data_L2	O	32	Write data from L2_RAM, signed

Implementation

Layer0 : padding

- We need zero-padding before Convolution to keep the image size after doing convolution.

pixel0	. . .	pixel 127
. . .	. . .	. . .
pixel 16256	. . .	pixel 16383

reading 128*128 pixels of unsigned 8bits data from gray-image ROM

Zero-padding

0	0	. . .	0	0
0	pixel0	. . .	pixel 127	0
. . .	. . .	. . .	. . .	. . .
0	pixel 16256	. . .	pixel 16383	0
0	0	. . .	0	0

zero-padding to 130*130 image (don't need to store the whole image in your circuit)

Implementation

Layer0 : Padding

- We need zero-padding before Convolution to keep the image size after doing convolution.

Zero-padding

reading 128*128 pixels of unsigned 8bits data from gray-image ROM

zero-padding to 130*130 image(don't need to store the whole image)

Implementation

Layer0 : convolution

- Doing Convolution after Zero-padding, we can get a same size as original gray image.

0	0	. . .	0	0
0	pixel0	. . .	pixel 127	0
. . .	. . .	. . .	. . .	. . .
0	pixel 16256	. . .	pixel 16383	0
0	0	. . .	0	0

130*130 pixels of unsigned 8bits data

⊗

3'b110 (-2)	3'b000 (0)	3'b001 (1)
3'b000 (0)	3'b001 (1)	3'b010 (2)
3'b001 (1)	3'b010 (2)	3'b101 (-3)

filter,3-bit signed

⇨

pixel0	. . .	pixel 127
. . .	. . .	. . .
pixel 16256	. . .	pixel 16383

Convolution(128*128),13-bit,signed (don't need to store the whole image in your circuit)

Implementation

- Layer0: convolution example
  - Convolution implementation ex:

1*(-2)+2*(0)+3*(1)+0*(0)+1*(1)+2*(2)+3*(1)+0*(2)+1*(-3)=6

2*(-2)+3*(0)+0*(1)+1*(0)+2*(1)+3*(2)+0*(1)+1*(2)+2*(-3)=6

1*(-2)+2*(0)+3*(1)+0*(0)+1*(1)+0*(2)+0*(1)+4*(2)+1*(-3)=7

  - Data format : 8-bit unsigned * 3-bit signed = 13-bit signed.

1	2	3	0	1
0	1	2	3	0
3	0	1	2	3
2	3	0	1	0
2	3	0	4	1

example

$\otimes$

-2	0	1
0	1	2
1	2	-3

=

6	0	-6
12	6	6
6	-5	7

Implementation

- Layer0 : convolution
  - Doing Convolution after Zero-padding, we can get a same size as original gray image.

130*130 pixels of unsigned 8bits data

3'b110 (-2)	3'b000 (0)	3'b001 (1)
3'b000 (0)	3'b001 (1)	3'b010 (2)
3'b001 (1)	3'b010 (2)	3'b101 (-3)

filter,3-bit signed

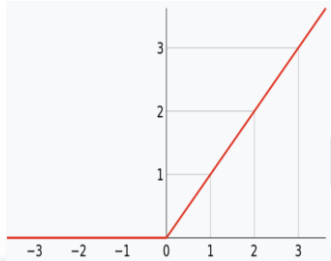
Convolution(128*128),13-bit,signed

Implementation

Layer0 : Activation Function - ReLU

- Doing ReLU calculation after convolution, we can get a sequence of 12 bits non-negative data, store back to L0 RAM.

Before



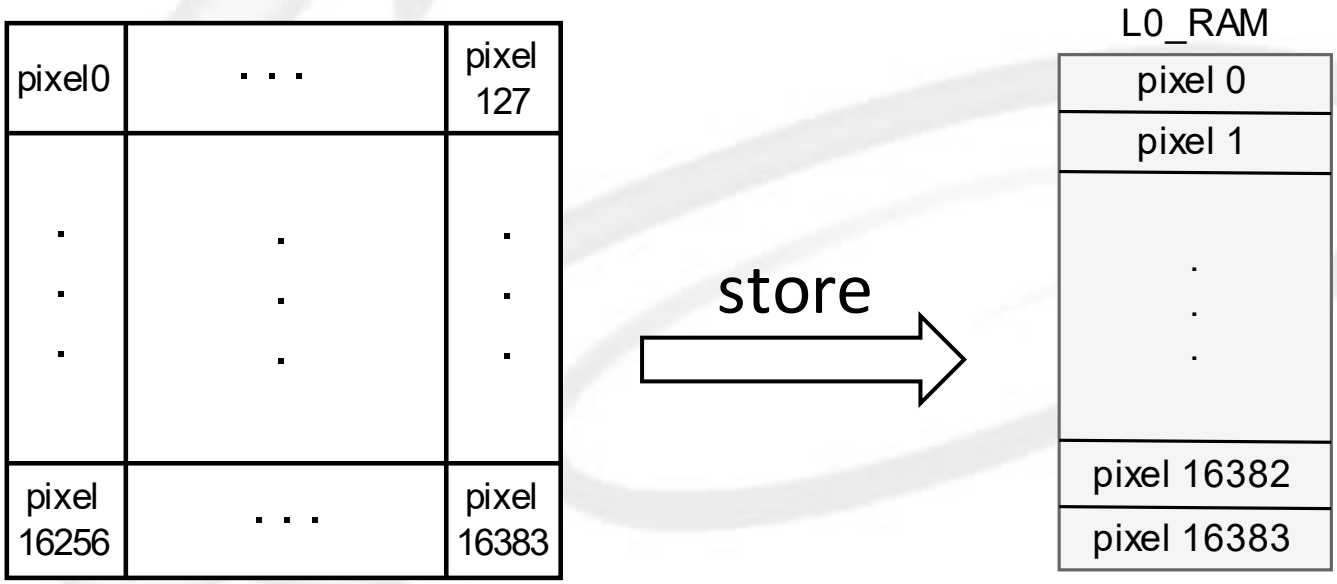
ReLU

$$f(x) = \max(0, x)$$

After

Implementation

- Layer0 : Store back to L0_RAM
  - Data structure : Row major
  - Storing the result to L0_RAM.
  - Data format : 12-bit unsigned (>=0).



After ReLU function

Implementation

Layer1 : Max Pooling

- We do 2*2 max pooling in Layer1,you can reuse the data you have stored into L0 RAM, or you have a better algorithm to catch the data directly before storing back to L0 RAM

pixel0	. . .	pixel 127
. . .	. . .	. . .
pixel 16256	. . .	pixel 16383

2*2 Max pooling

pixel0	. . .	pixel 63
. . .	. . .	. . .
pixel 4032	. . .	pixel 4095

128*128 pixels of unsigned 12bits data from Layer 0

Max-pooling to 64*64 image(don't need to store the whole image in your circuit)

Implementation

Layer1 : Max Pooling example

- Pooling layers allow us to reduce the size of the image so that the neural network works faster. It basically creates a smaller image by dividing the image in several n by n matrices (say 2 by 2 matrices).

56	13	84	91
17	48	75	43
9	63	58	40
49	71	11	88

Before

2*2 Max pooling

56	91
71	88

After

Implementation

Layer1 : Max Pooling

- Doing Max pooling in Layer1, we can get a smaller image size, and store it back to L1 RAM.

Before
128*128

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

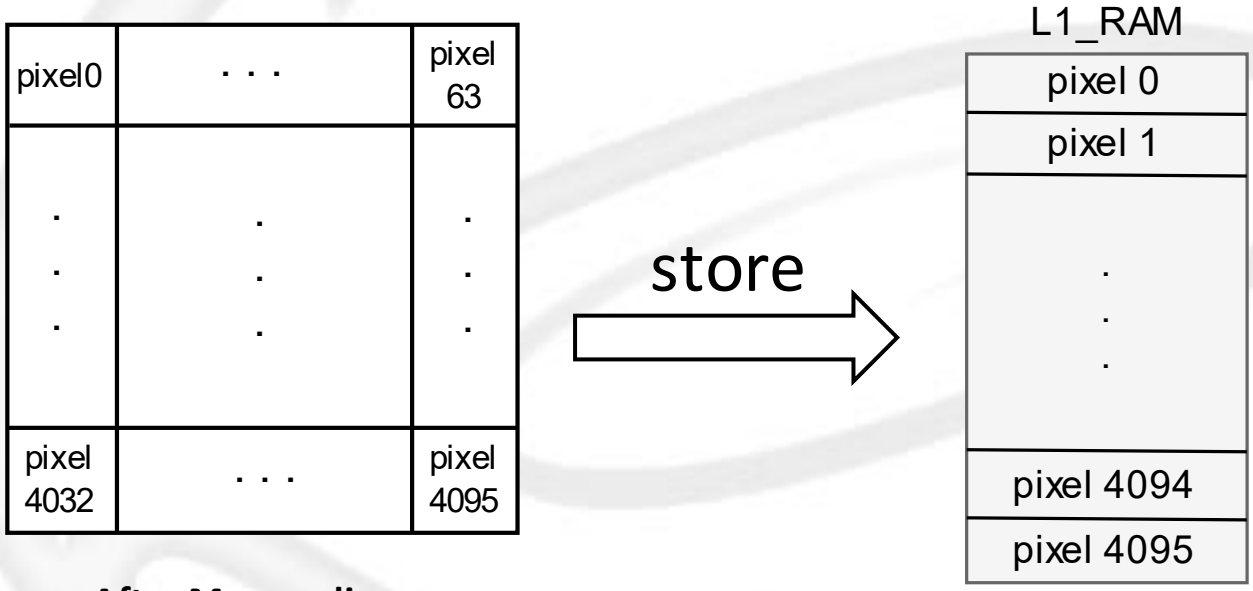
6	8
9	7

2*2 Max pooling

After
64*64

Implementation

- Layer1 : Store back to L1_RAM
  - Data structure : Row major
  - Storing the result to L1_RAM.
  - Data format : 12-bit unsigned (>=0).

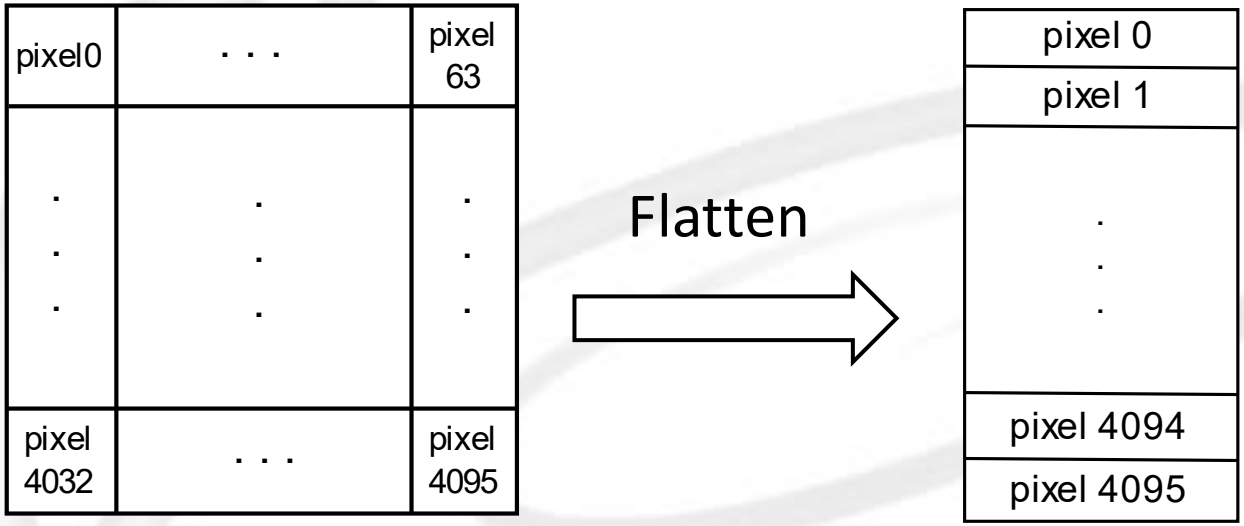


After Max pooling

Implementation

Layer2 : Flatten

- Data structure : Row major
- Data format : 12-bit unsigned (>=0)
- Hint : You don't need to do any change in your circuit!



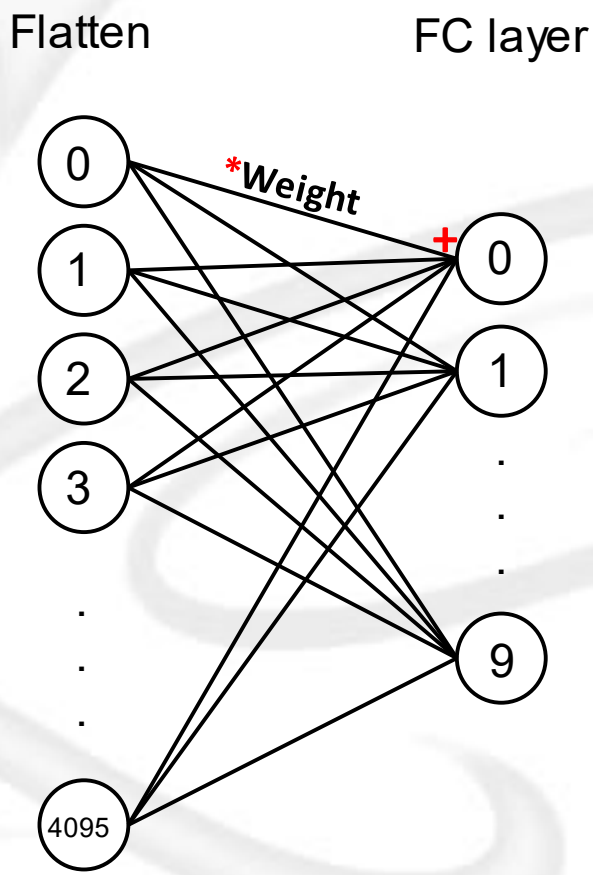
Layer 1 result
64*64

Flatten
4096*1

Implementation

Layer2 : Fully connected

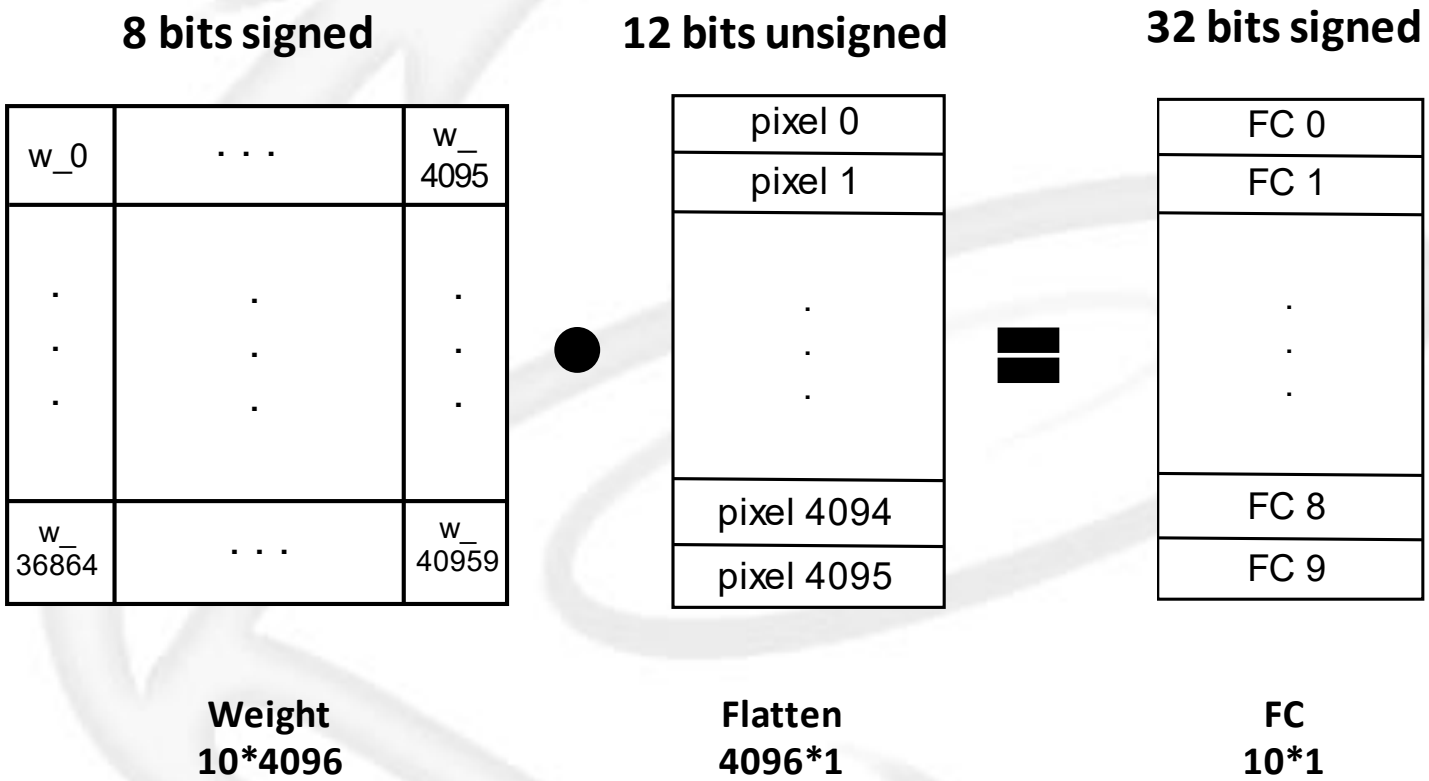
- A fully connected layer refers to a neural network in which each input node is connected to each output node.



Implementation

Layer2 : Fully connected

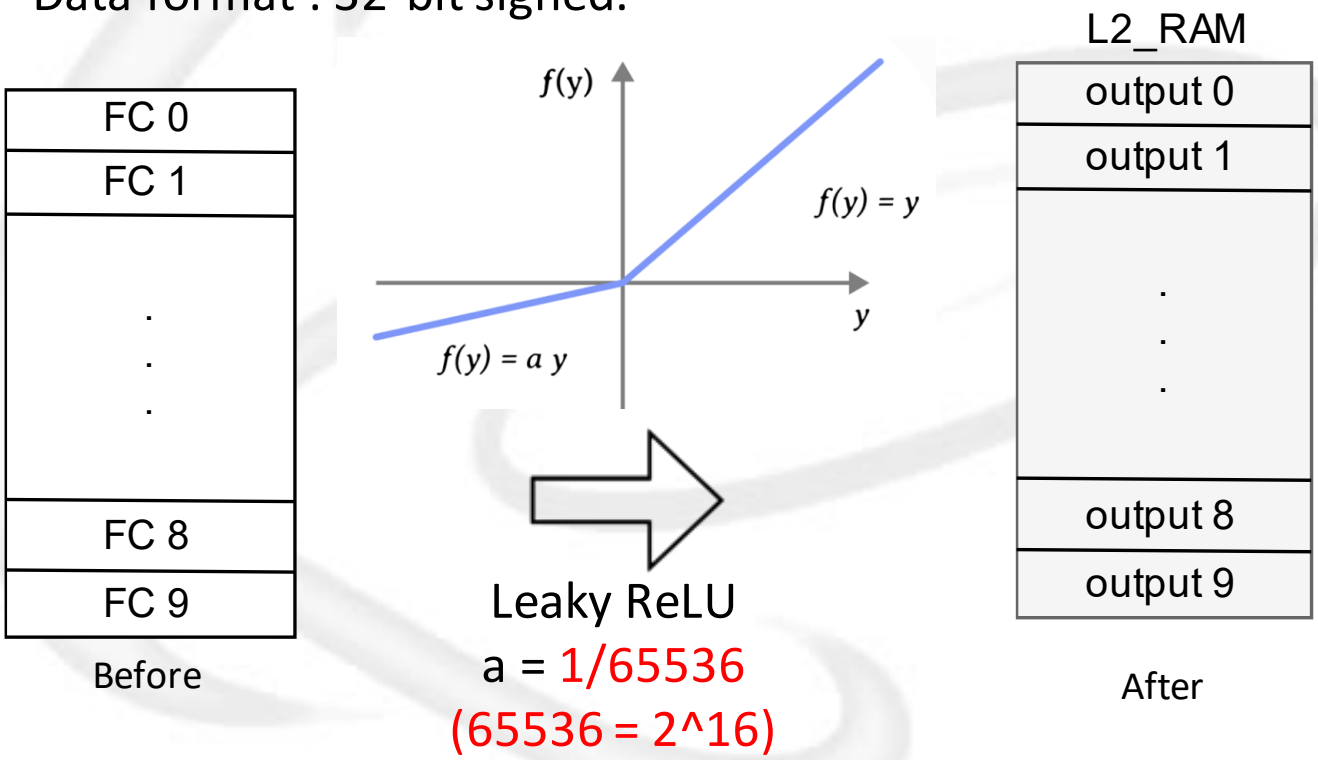
- Use matrix multiplication, accumulate each weight * pixel
- Weight's data structure in ROM is row major



Implementation

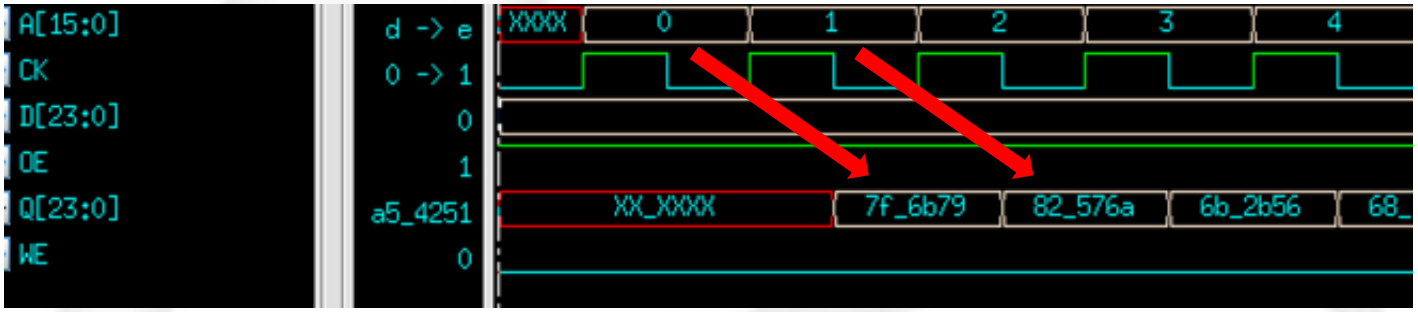
Layer2 : Activation Function – Leaky ReLU

- Doing Leaky ReLU calculation after fully connected layer, we can get a sequence of 32 bits signed data, store back to L2 RAM.
- Leaky ReLU : If y<0, f(y) = y * (1/65536); else, f(y) = y
- Hint : you don't need a divider in this case
- Data format : 32-bit signed.



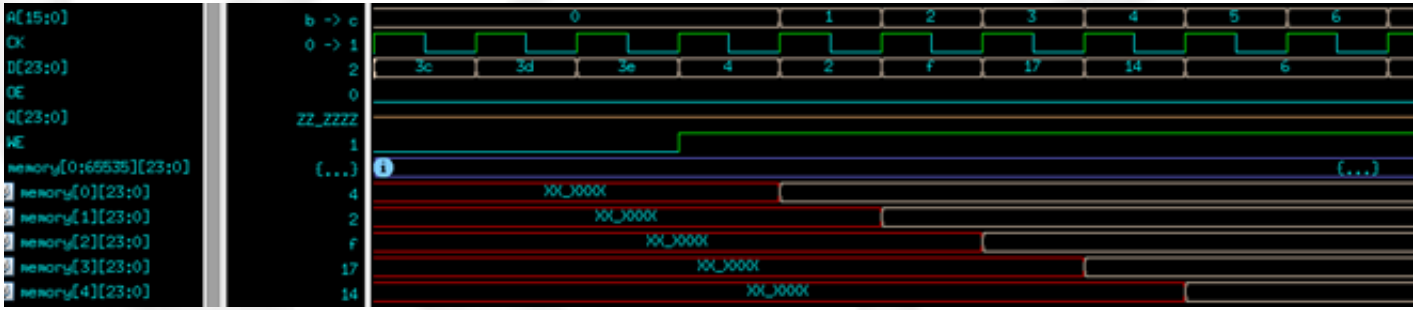
Implementation

- The timing information for Read/Write SRAM
  - Read operation(delay one cycle)



✓ The memory will output values on the negative edge, and you need to capture data on the positive edge

  - Write operation



Simulation Tips

- Testbench
- You can only modify those red box in testbench

```
`timescale 1ns/10ps
`define CYCLE       25.0                    // Modify your clock period here (do not larger than 25)
`define End_CYCLE   200000000               // Modify cycle times once your design need more cycle times!

`define gray_image      "../../dat_grad/gray_image.txt"                 // Open in Quartus
`define L0_EXP0         "../../dat_grad/relu_image.txt"
`define weight          "../../dat_grad/random_weight.txt"
`define L1_EXP0         "../../dat_grad/pooled_image.txt"
`define L2_EXP          "../../dat_grad/result_leaky_relu.txt"

// `define gray_image      "./dat_grad/gray_image.txt"                  // Open in ModelSim
// `define L0_EXP0         "./dat_grad/relu_image.txt"
// `define weight          "./dat_grad/random_weight.txt"
// `define L1_EXP0         "./dat_grad/pooled_image.txt"
// `define L2_EXP          "./dat_grad/result_leaky_relu.txt"

module testfixture;

logic       check0=1, check1=1, check2=1;   ////you can only modify this line

logic   [11:0]  L0_EXP0 [0:16383];
logic   [11:0]  L1_EXP0 [0:4095];
logic   [31:0]  L2_EXP  [0:99];
```

Simulation Tips

- CONV.sdc
- You can only modify this red box in CONV.sdc
- Do not set this clock period more than 25(ns)

```
# operating conditions and boundary conditions #

set cycle 25.0
create_clock -name clk  -period $cycle   [get_ports  clk]


#Don't touch the basic env setting as below
set_input_delay  5.0   -clock clk [remove_from_collection [all_inputs] [get_ports clk]]
set_output_delay 5.0    -clock clk [all_outputs]

```

Simulation Tips

- Golden files
- Those file can help you debug easier.

gray_image.txt	Gray Image ROM
pooled_image.txt	Layer 1 golden
random_weight.txt	Weight ROM
relu_image.txt	Layer 0 golden
result_dot.txt	Fully connected golden
result_leaky_relu.txt	Layer 2 golden

Criteria

- Grading policy(100%)
  - Layer 0 RTL simulation pass (10%)
  - Layer 1 RTL simulation pass (10%)
  - Layer 2 RTL simulation pass (10%)
  - Layer 0 Gate-Level simulation pass (10%)
  - Layer 1 Gate-Level simulation pass (10%)
  - Layer 2 Gate-Level simulation pass (10%)
  - PA(Simulation time * Area) (15%)
  - Hand In Time (code only) (5%)
  - You will only get the point that previous mission pass, Include PA & Hand in time
    ex : Layer 0 RTL + Gate-Level pass & others fail, only 10 points
  - Report (20%)

Criteria

- Grading policy - Performance & Area
  - Performance : Simulation time in Gate-Level Simulation
  - Area : Total logic elements + 1000 * embedded Multipliers
  - This 10 points need Gate-Level all pass to attend the PA Ranking
  - Top 10% got 15 points, Top 10%~20% got 13.5 points….

```
# Loading work.RAM_32
# Loading instances from CONV_7_1200mv_85c_v_slow.sdo
# Loading altera_ver.PRIM_GDFF_LOW
# Loading timing data from CONV_7_1200mv_85c_v_slow.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Instance: /testfixture File: C:/Users/justi/Documents/NCKU/HDL/midterm/testfixture.sv
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ------------------------------------------------------------
# START!!! Simulation Start ..... ------------------------------
# Layer 0 (Convolutional Output) with Kernel 0 is correct !
# Layer 1 (Max-pooling Output) with Kernel 0 is correct!
# Layer 2 (Flatten  Output) is correct!
# -------------------- S U M M A R Y ----------------
#
# Congratulations! Layer 0 data have been generated successfully! The result is PASS!!
#
# Congratulations! Layer 1 data have been generated successfully! The result is PASS!!
#
# Congratulations! Layer 2 data have been generated successfully! The result is PASS!!
#
#  ****************************
#  **                        **       |__||
#  **  Congratulations !!    **      / O.O  |
#  **                        **    /_____   |
#  **  Pattern   All Pass    **   /^ ^ ^ \  |
#  **                        **  |^ ^ ^ ^ |w|
#  ****************************   \m___m__|_|
#
#
# Your score in     RTL    Simulation = 30
# Your score in Gate-Level Simulation = 65
# ------------------------------------------------------------
# ** Note: $finish    : C:/Users/justi/Documents/NCKU/HDL/midterm/testfixture.sv(331)
#    Time: 14747227294 ps  Iteration: 0  Instance: /testfixture
```

You should prove this scream shot is Gate-Level simulation & run in the same cell library case

Simulation time : 14747227294(ps)

Flow Summary

<<Filter>>

Flow Status	Successful - Mon Apr 15 11:58:00 2024
Quartus Prime Version	17.1.0 Build 590 10...017 SJ Lite Edition
Revision Name	CONV
Top-level Entity Name	CONV
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	452 / 114,480 (< 1 %)
Total registers	107
Total pins	173 / 529 (33 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	2 / 532 (< 1 %)
Total PLLs	0 / 4 (0 %)

Area = 452 + 1000 *2 = 2452

Criteria

- Grading policy – Hand In Time (code only)
  - This 5 points need Gate-Level all pass to attend the HIT Ranking
  - Top 20% got 5 points, Top 20%~40% got 4 points….
  - Hand in time Ranking based on the last Moodle & google form submit time. TA will choose the latest.
  - Please update your PA every time you upload your code!!!

Midterm Performance & Area google form

  - Report can submit individually after submitting your code.
  - You can check your PA & hand in time from the excel link

Midterm Performance & Area excel report

A	B	C	D	E	F
時間戳記	電子郵件地址	Student ID	Gate-Level Simulation Time (ps)	Area	Simulation time * Area
2024/4/16 上午 1:17:47	m16121116@gs.ncku.edu.tw	test	14747227294	2452	36160201324888

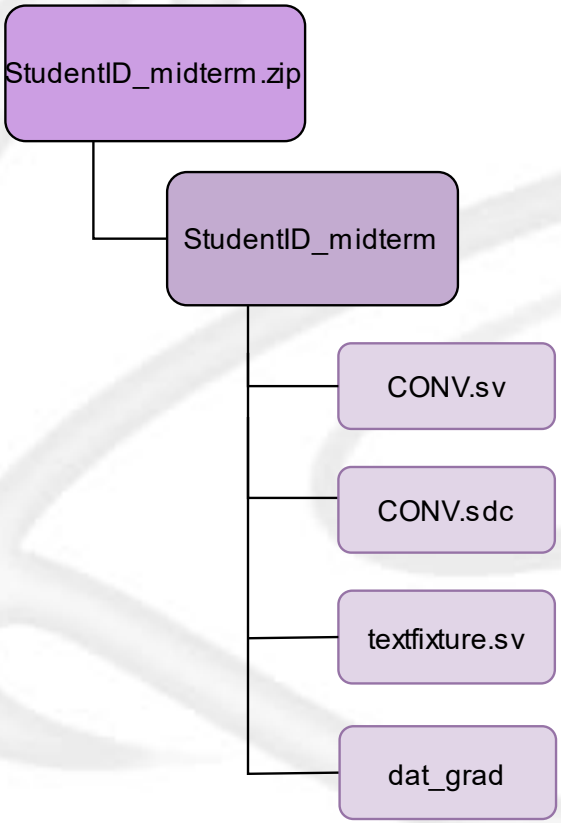
Criteria

- Grading policy – Report
  - Your Report should include those components
    - Performance & Area table
    - Block Diagram (10%)
    - Explain how your design works? (10%)
    - Draw the flowchart for your Finite State Machine (FSM). (10%)
    - Explain the result by waveform. (10%)
    - RTL & Gate-level simulation result on the terminal. (5%)
    - Scream shot form Quartus and give an explanation (5% each)
      - Schematic view of the design netlist
      - Finite State Machine view
      - Fmax summery
      - worst-case timing paths
      - power analyzer summery
      - flow summery
    - Q&A (5% each)
      - What is the difference between RTL simulation & Gate level simulation?
      - What is your strategy to get a batter PA?
    - lesson learned from midterm.(15%)

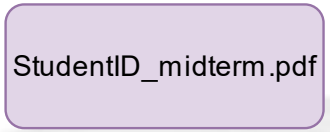
Requirement & file format

- File format
- Deadline:2024/05/02 8:59

Midterm project code



Midterm project report



Requirement & file format

Friendly reminder

- Please complete the assignment by your own, discussion with peers is recommended, but do not cheat.
- Warning! Any dishonesty found will result in zero grade.
- Warning! Designing responses tailored to the golden files will result in zero grade.
- Warning! Any late submission will also receive zero.
- Warning! Please make sure that your code can be compiled in Modelsim & Quartus, any dead body that we cannot compile will also receive zero.
- Warning! Please submit your work according to the specified file format, making sure not to include any unnecessary files. Any unnecessary file found, will lead to 10% deduction from the overall score.
- Please start this project As Soon As Possible, Quartus synthesize & Gate Level simulation will take you a lot of time.
- A bad coding style may cause your Gate Level simulation unsuccessful!!!

Thanks for listening