National Cheng Kung University
Department of Electrical Engineering

Introduction to VLSI CAD (Spring 2022)

Lab Session 7

SOM Processing System

Name	Student ID	
吳紹齊	E34073071	
張育誠	E24066200	
Practical	Points	Marks
Lab 7_1	45	
Lab 7_2	45	
Demo	10	
Notes		

Due: 15:00 April 27, 2022@ moodle

Deliverables

1) All Verilog codes including testbenches for each problem should be uploaded.
   NOTE: Please **DO NOT** include source code in the paper report!
2) All homework requirements should be uploaded in this file hierarchy.
3) NOTE: 1. Please **DO NOT** upload waveforms (.fsdb or .vcd)!
4) If you upload a dead body which we can't even compile, you will get NO credit!

5) All Verilog file should get at least 90% SuperLint Coverage.
6) All homework requirements should be uploaded in this file hierarchy or you will not get full credit, if you want to use some sub modules in your design but you do not include them in your tar file, you will get 0 point.

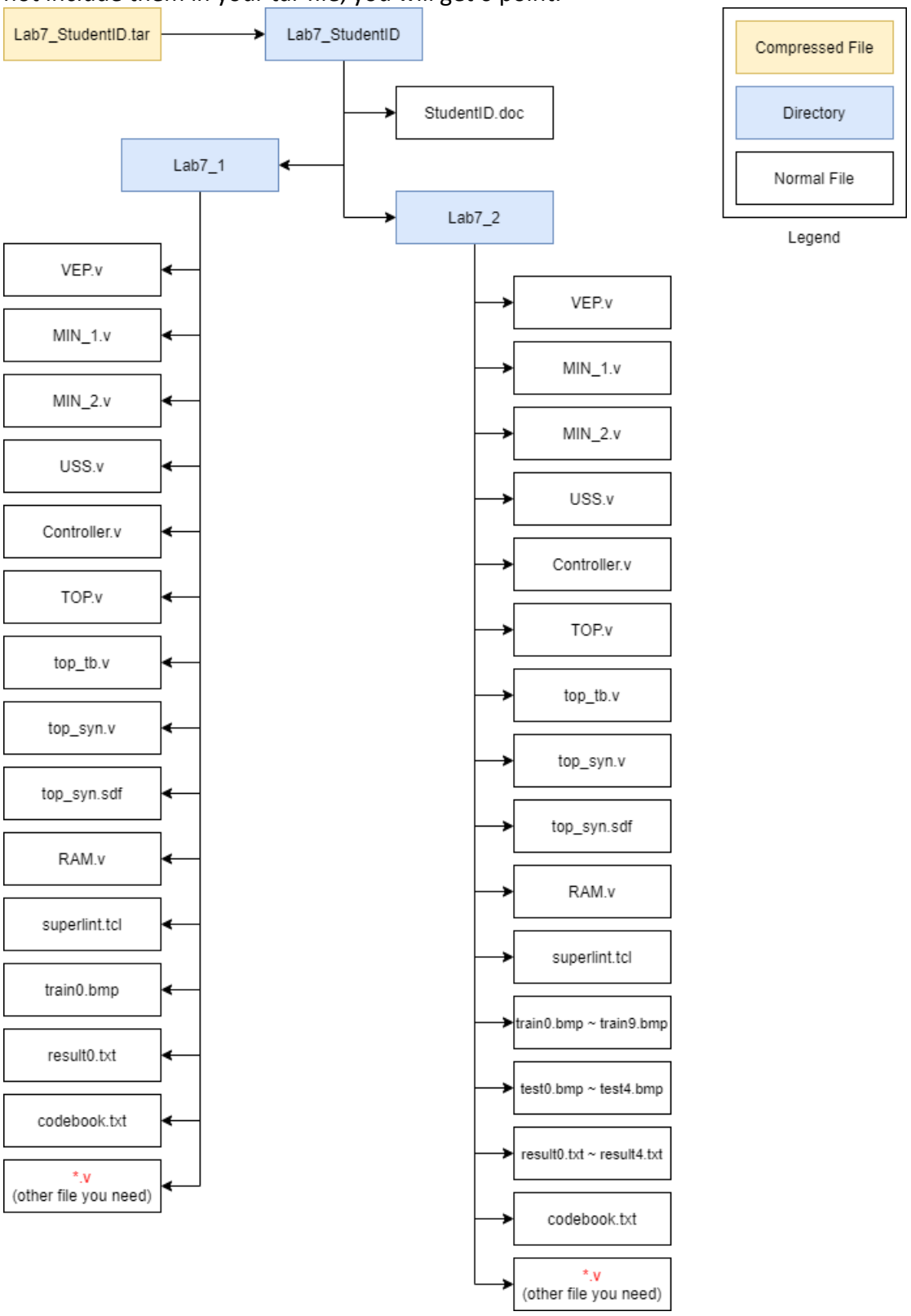
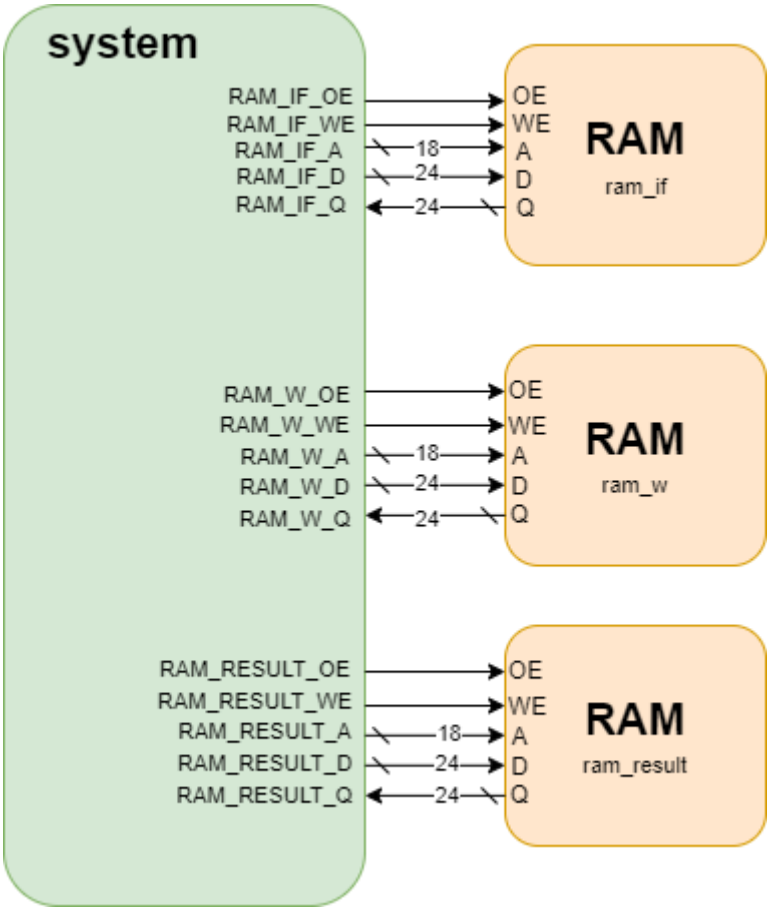
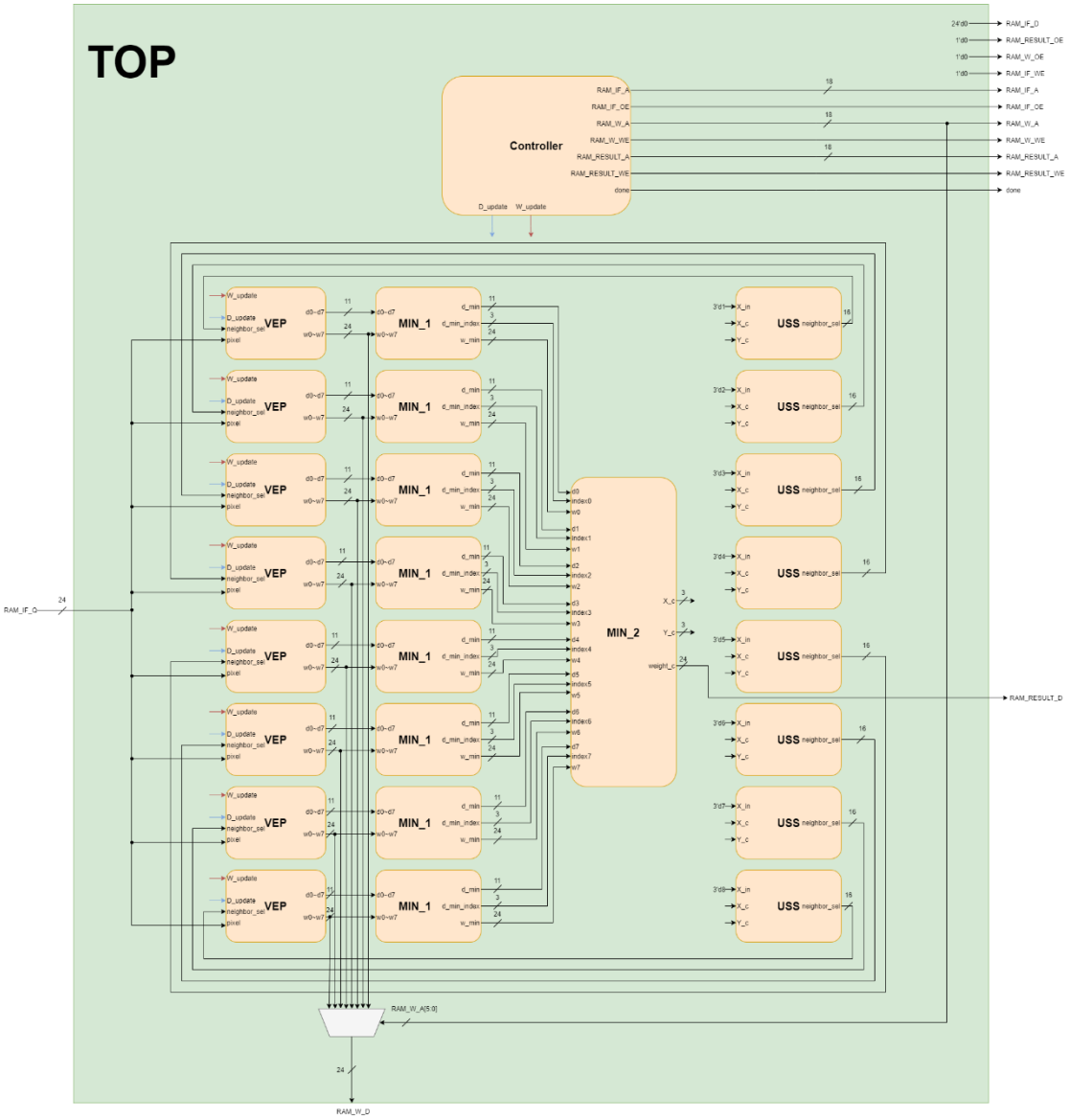


Fig.1 File hierarchy for Homework submission

You are about to integrate all components (VEP, MIN_1, MIN_2, USS, Controller…) to form a SOM processing system. The block diagram of system is as shown in **Fig2** and **Fig3**. (Clock pin and reset pin is ignored in the graph, but you should implement it)



▲Fig2. The block diagram of system (external)



▲Fig3. The block diagram of system (internal)

- **Port list of each module:**
- Controller

Signal	I/O	bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
D_update	Output	1	Distance update enable, active high
W_update	Output	1	Weight update enable, active high
RAM_IF_A	Output	18	Input feature RAM address
RAM_IF_OE	Output	1	Input feature RAM output enable
RAM_W_A	Output	18	Weight RAM address
RAM_W_WE	Output	1	Weight RAM write enable
RAM_RESULT_A	Output	18	Result RAM address
RAM_RESULT_WE	Output	1	Result RAM write enable
done	Output	1	Pull to 1 if the system is done

- **VEP**

Signal	I/O	Bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
W_update	Input	1	Weight update enable, active high
D_update	Input	1	Distance update enable, active high
neighbor_sel	Input	16(2x8)	Neighborhood function of 8 VEP weights (00=>1, 01=>0.25, 10=>0.125, 11=>0)
pixel	Input	24	Input pixel from RAM_if
d0~d7	Output	11	Manhattan distance between 8 weights
w0~w7	Output	24	8 weights

- **MIN_1**

Signal	I/O	bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
d0~d7	Input	11	Manhattan distance between 8 weights
w0~w7	Input	24	8 weights
d_min	Output	11	Minimum distance between d0~d7
d_min_index	Output	3	Index of minimum distance
W_min	output	24	Weight of minimum distance

- **MIN_2**

signal	I/O	bit	Description
clk	Input	1	clock
rst	Input	1	Reset signal, active high
d0~d7	Input	11	Minimum distance from MIN_1
w0~w7	Input	24	Weight of minimum distance from MIN_1
index0~index7	Input	3	Index of minimum distance from MIN_1
X_c	Output	3	The X coordinate of center weight
Y_c	Output	3	The Y coordinate of center weight
weight_c	output	24	Center weight

- **USS**

signal	I/O	bit	Description
clk	input	1	Clock
rst	Input	1	Reset signal, active high
X_in	Input	3	USS module index
X_c	Input	3	The X coordinate of center weight
Y_c	Input	3	The Y coordinate of center weight
neighbor_sel	Output	16(2x8)	Neighborhood function of 8 VEP weights (00=>1, 01=>0.25, 10=>0.125, 11=>0)

- Top

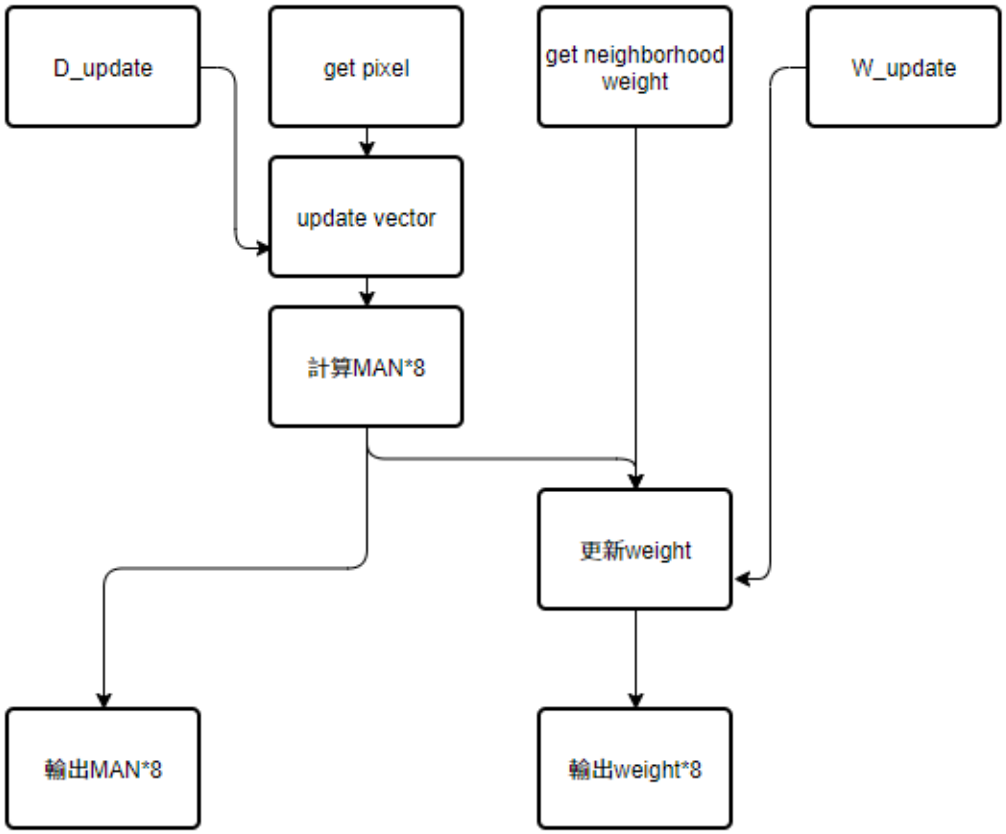
Signal	I/O	bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
RAM_IF_Q RAM_W_Q RAM_RESULT_Q	Input	24	Data output from RAM
RAM_IF_OE RAM_W_OE RAM_RESULT_OE	Output	1	RAM output enable signal
RAM_IF_WE RAM_W_WE RAM_RESULT_WE	Output	1	RAM write enable signal
RAM_IF_A RAM_W_A RAM_RESULT_A	Output	18	RAM address
RAM_IF_D RAM_W_D RAM_RESULT_D	output	24	Data written into RAM
done	Output	1	Pull to 1 if the system is done

- Understanding the function:

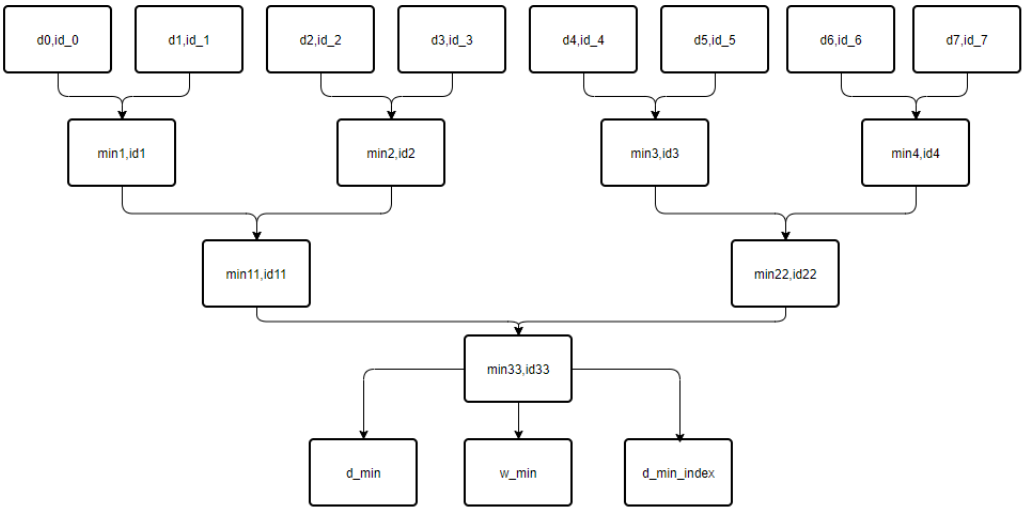
Once system is initialized, it

a) Read input pixel from RAM_if
b) Calculate Manhattan distance and fin the minimum distance
c) Update the weight memory
d) Repeats the process step(a)~(c)until the last pixel of RAM_if is read
e) writes the trained codebook to the RAM_w
f) read input pixel from RAM_if and inference the picture
g) writes the lossy compression picture to the RAM_result
h) repeats the process step (f)~(g) until the last pixel of RAM_result is writed;
i) flags “done” when system is completed

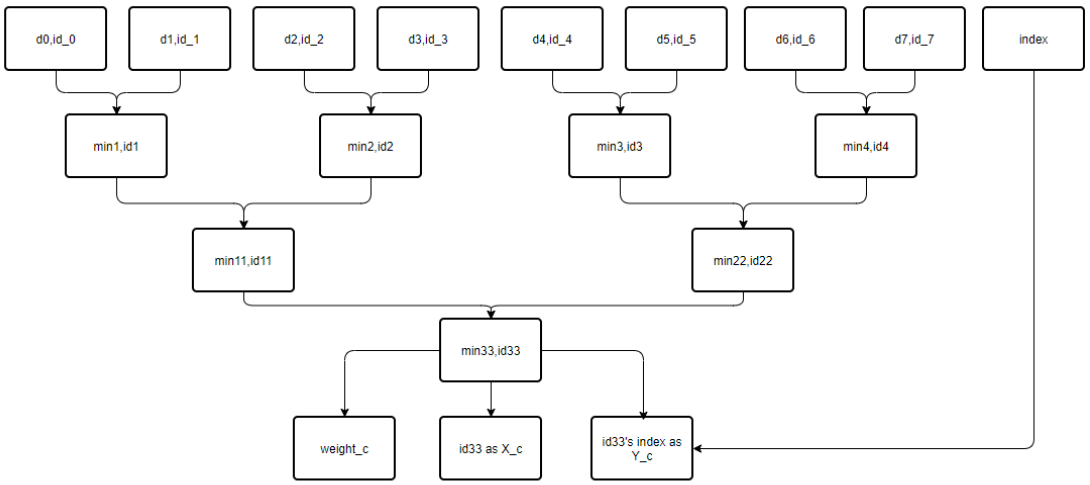
- Describe your design in detail. You can draw internal architecture or block diagram to describe your dsign.
  - VEP



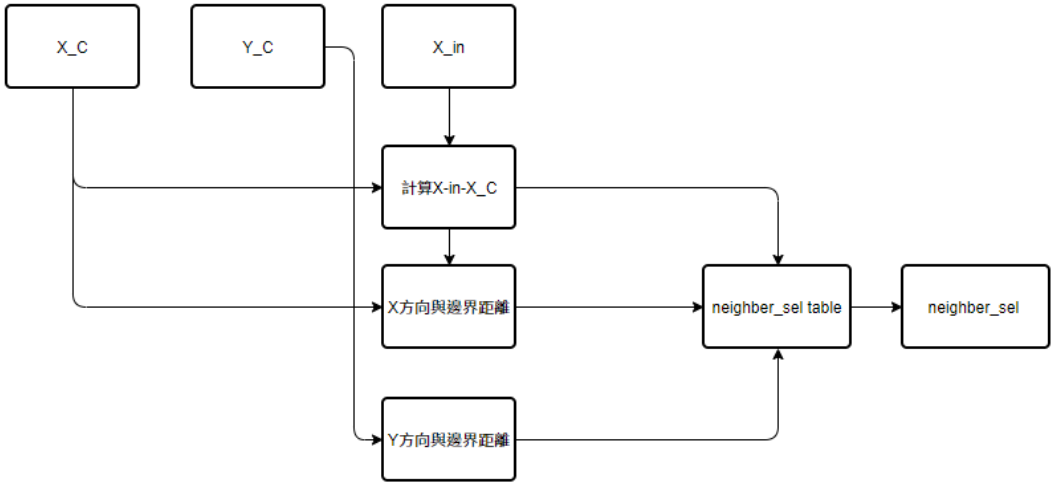
MIN_1


d0,id_0
d1,id_1
d2,id_2
d3,id_3
d4,id_4
d5,id_5
d6,id_6
d7,id_7
min1,id1
min2,id2
min3,id3
min4,id4
min11,id11
min22,id22
min33,id33
d_min
w_min
d_min_index

MIN_2

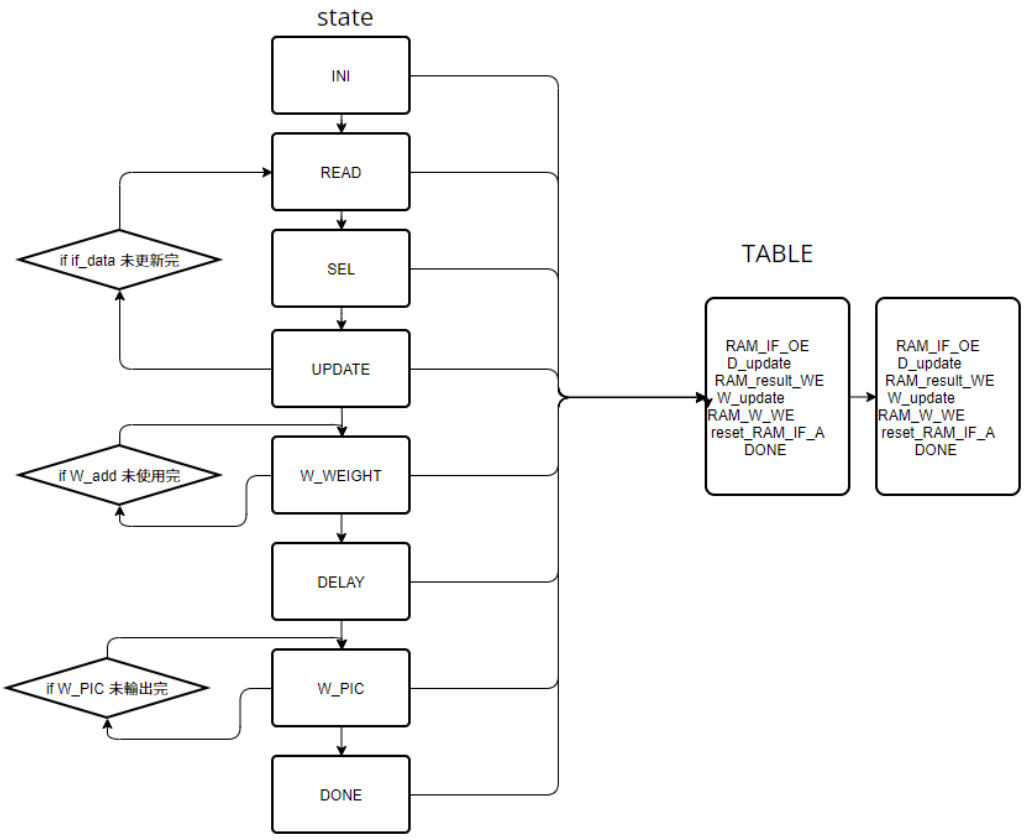

d0,id_0
d1,id_1
d2,id_2
d3,id_3
d4,id_4
d5,id_5
d6,id_6
d7,id_7
index
min1,id1
min2,id2
min3,id3
min4,id4
min11,id11
min22,id22
min33,id33
weight_c
id33 as X_c
id33's index as Y_c

■ USS



■ Controller

◆ Draw your state diagram in controller and explain it



1) Complete the Controller, VEP, MIN_1, MIN_2, USS, and TOP module, in the system.

2) Compile the verilog code to verify the operations of this module works properly.

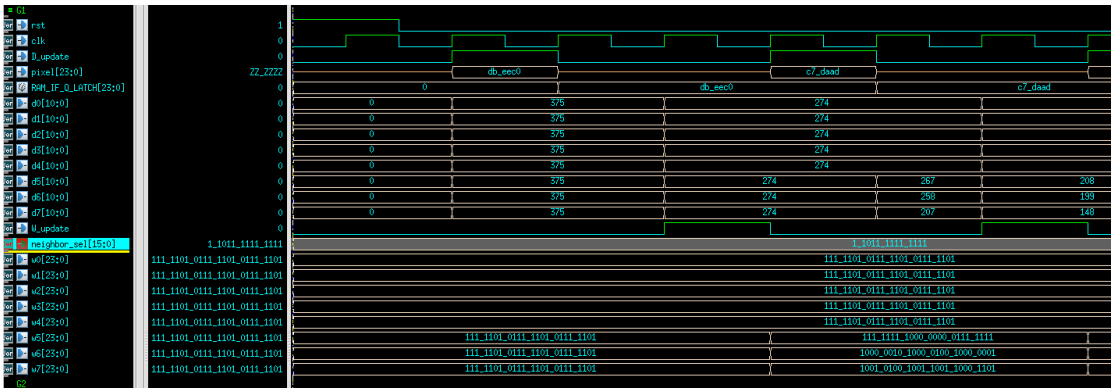
3) Synthesize your *TOP.v* with following constraint:

- Clock period: no more than 20 ns.

- Don’t touch network: clk.
- Wire load model: saed14rvt_ss0p72v125c.
- Synthesized verilog file: *top_syn.v*.
- Timing constraint file: *top_syn.sdf*.

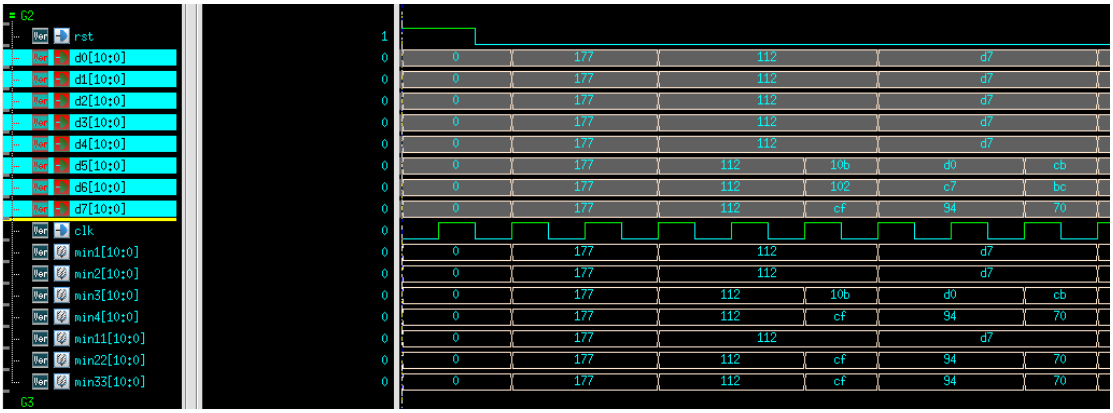
4) Please **attach your waveforms** and **specify your operations** on the waveforms.

VEP(VEP7 為例)



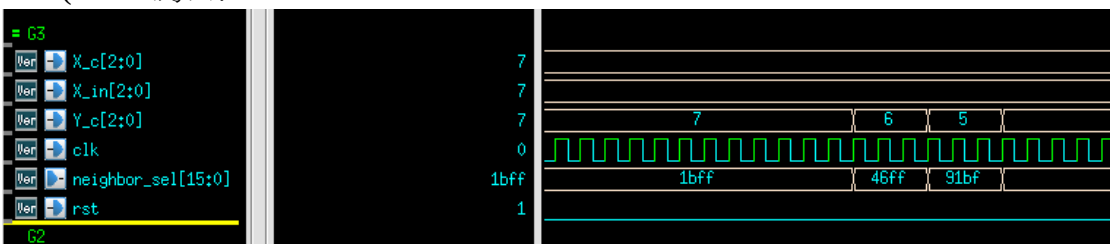
Reset 後所有 D 變成 0，所有 W 變成(125,125,125)並在第一個 D_update 時進來第一個 pixel，並在下一個 clk 吃進 RAM_IF_Q_LATCH，在經過 combinational 的運算後於下一個 clk 輸出到 D0~D7，下方的 W0~W7 也在 W_update 之後改變數值。

MIN(MIN7 為例)

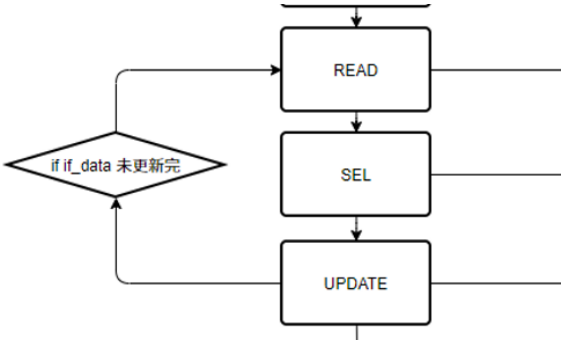


由於全部都是 combinational 所以只要有 posedge，所有數值全部更新

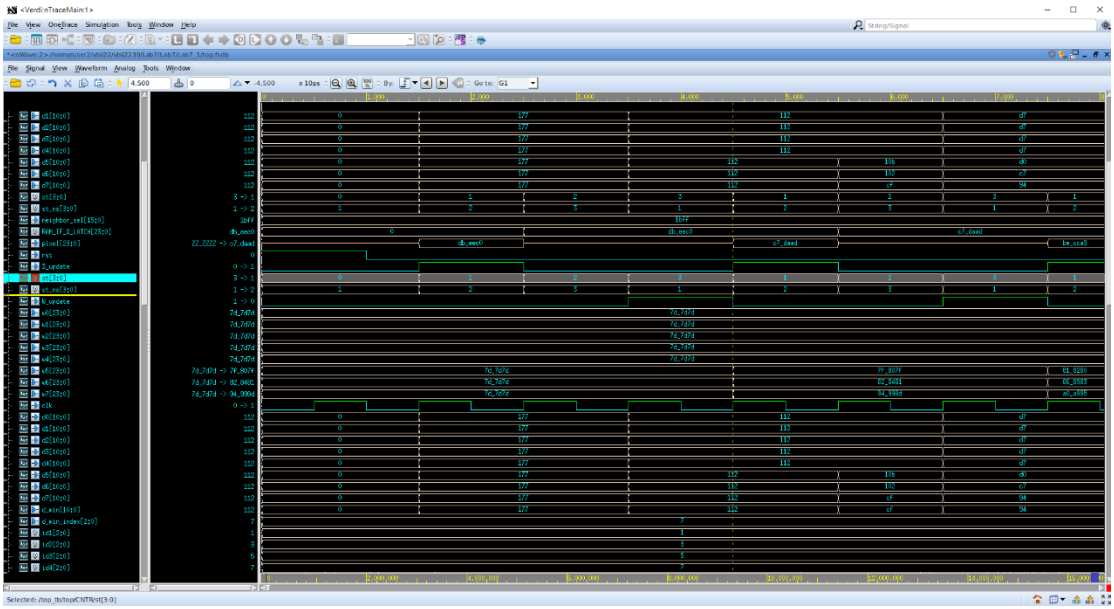
USS(USS7 為例)



根據 table 輸出



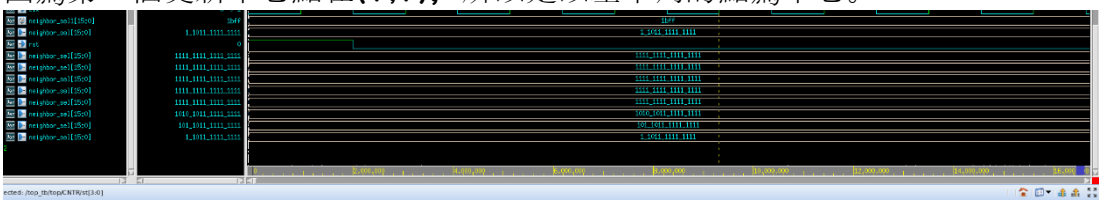
在訓練的部分，我分成 3 個 state，第 1 個 state 是 READ 主要是將資料讀進來，例如第一筆資料是 db_eec0，資料讀進來後，儲存到 RAM_IF_Q_LATCH，RAM_IF_Q_LATCH 會維持 db_eec0 直到 Update 完權重。

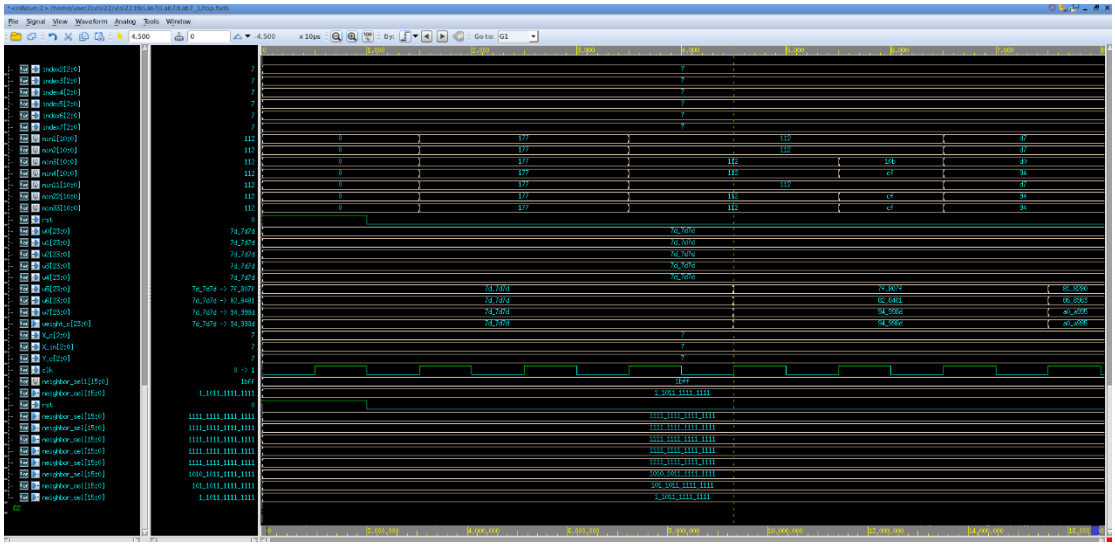


第 2 個 state 是 SEL，主要是計算的部分，這裡我設計的都是 combinational block，MIN_1、MIN_2、USS 都在 1 個 clk 裡，所以 1 個 clk 就可以算完 neighbor_sel，並輸出 neighbor_sel。

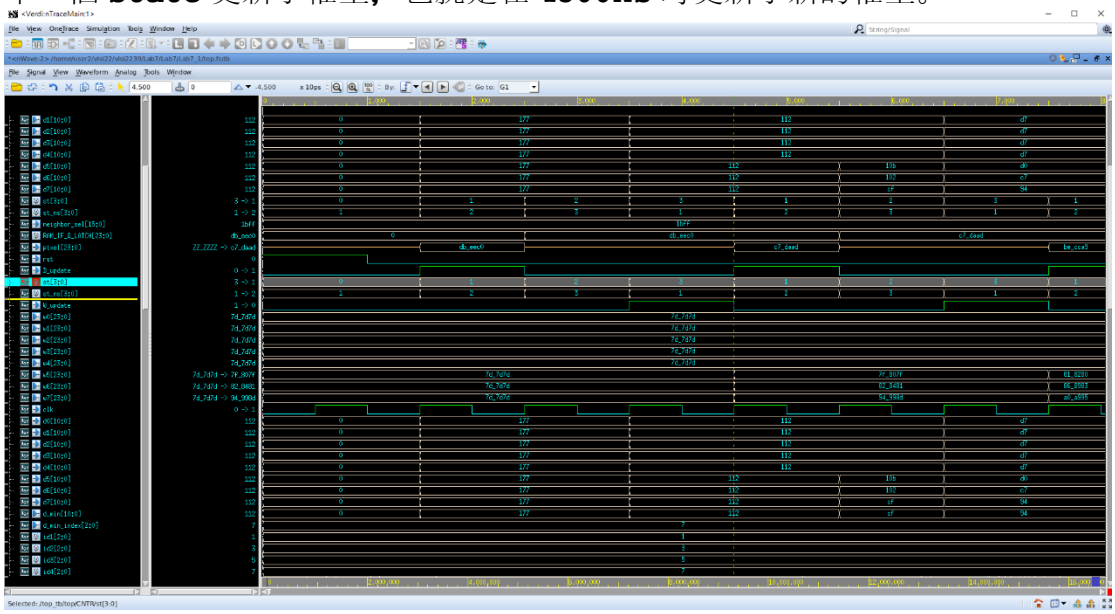
下圖可以看到完整的 neighbor function

因為第一個更新中心點在(7,7)，所以是以左下角的點為中心。

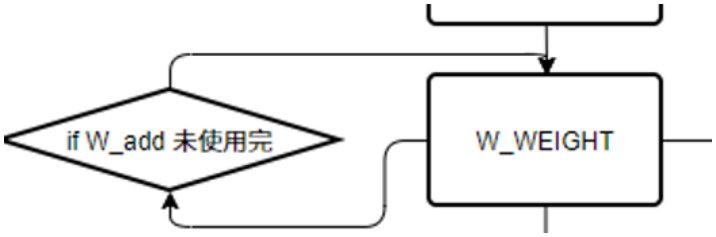




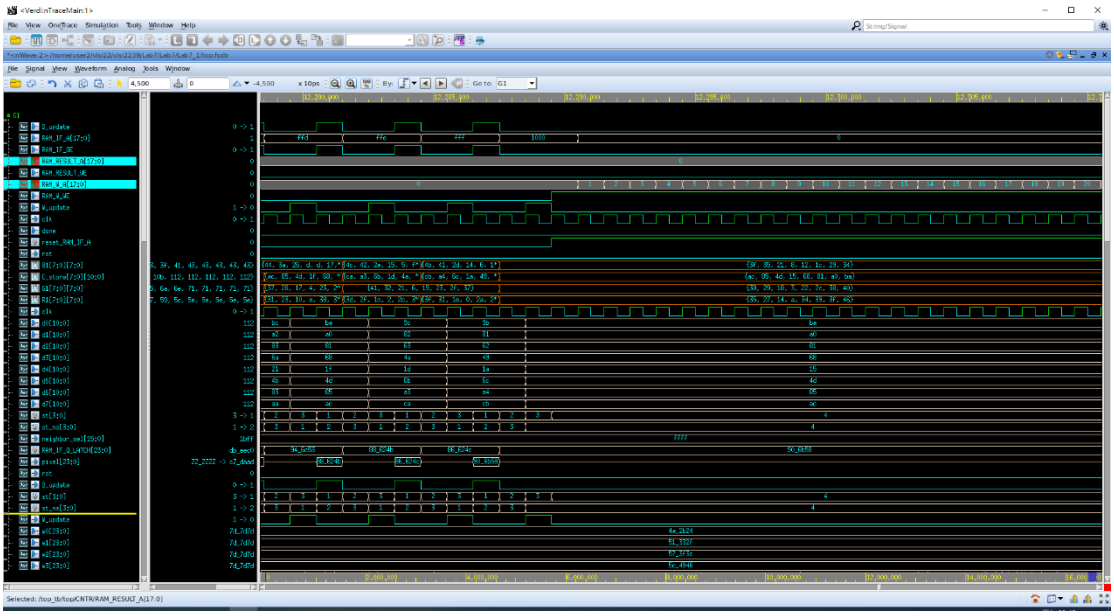
第三個state是UPDATE，主要是更新權重，根據接收到neighbor_sel來更新權重，可以看到在 UPDATE state 時，W_update 升高，所以在下一個 state 更新了權重，也就是在 4500ns 時更新了新的權重。



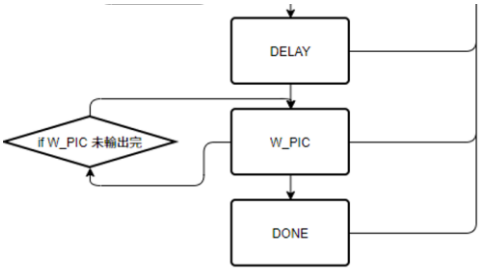
在訓練了 4096 次後，拿到了訓練完的權重，接著進到 W_WEIGHT，更新 codebook 資料，將更新的權重寫入 RAM_W_Q。



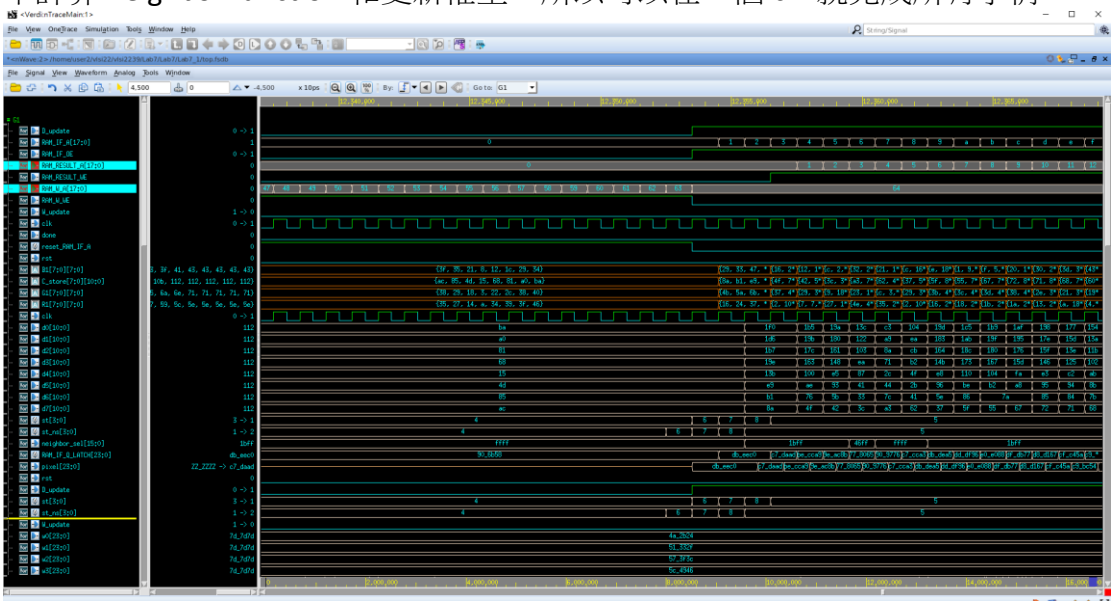
可以看到下圖的波型圖，進到 W_WEIGHT，RAM_W_WE 為 1，開始更新 codebook 資料，一共 64 組資料。



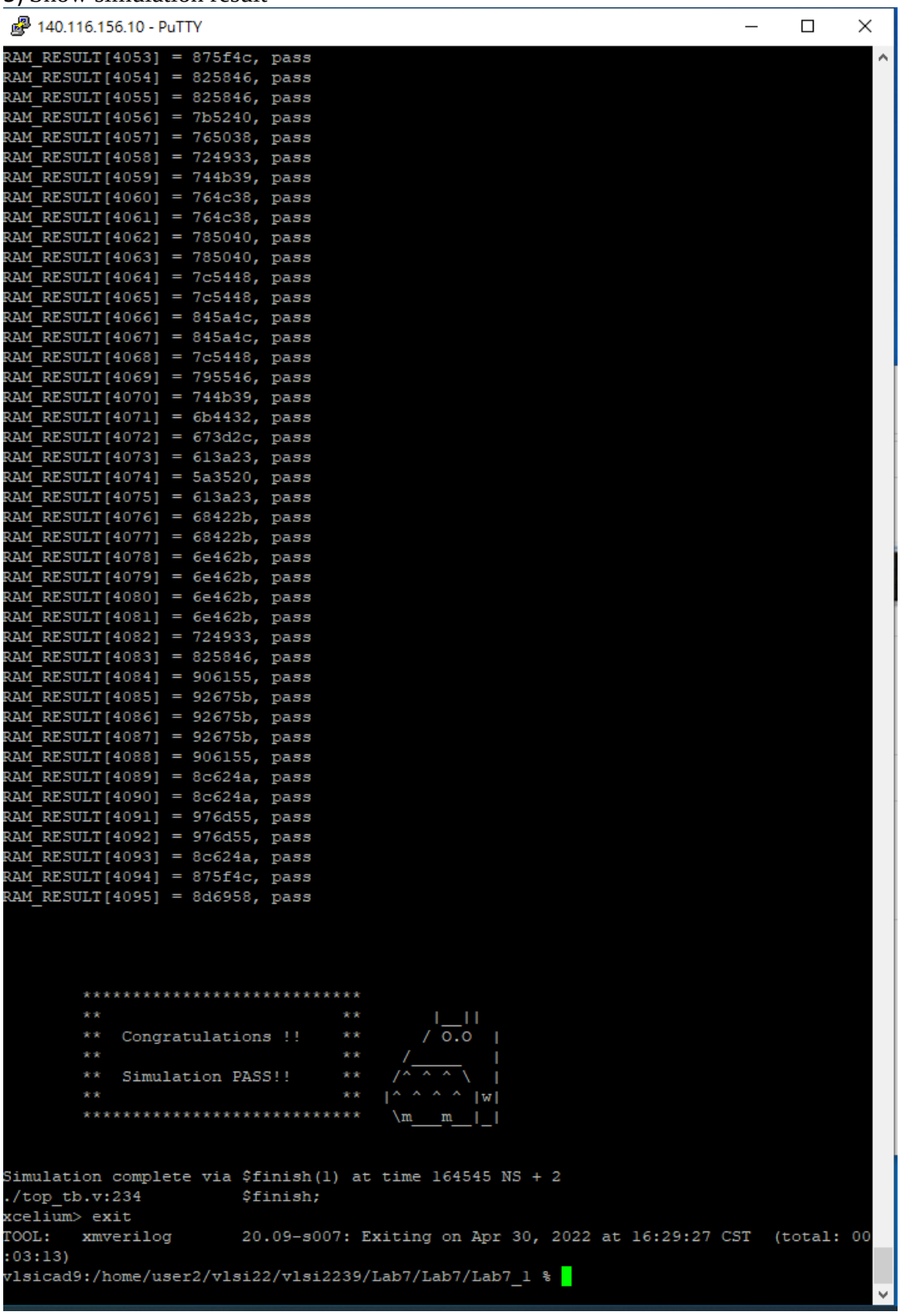
更新完 **codebook** 資料接著進到 **DELAY,** 這裡設計是因為有 **delay clk** 所以多設計一個 **state** 來符合傳輸資料的 **clock**。
這裡主要是 **W_PIC**。



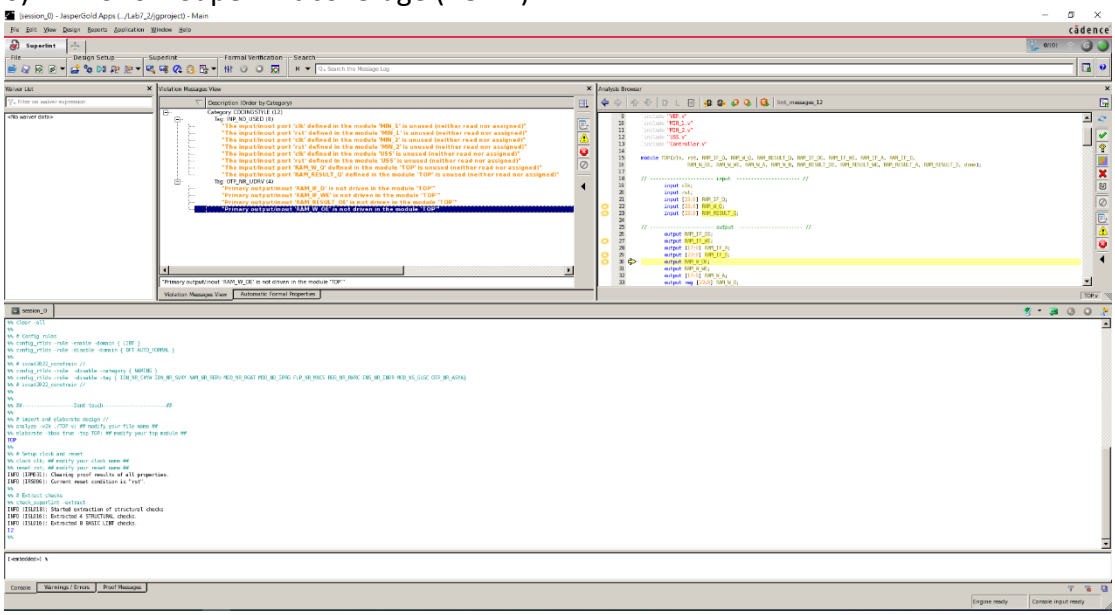
這裡 write compress picture result，可以使用 pipeline 的方式來更新，這裡資料傳遞方式是先從 RAM_IF 讀資料到 RAM_IF_Q_LATCH 然後進到 VEP，算 calculate manhattan distance，進到 MIN_1 比較最小的 index，最後進到 MIN_2 找中心點然後直接寫入 RAM_RESULT，這裡跟 1~3 STATE 的差別就是不計算 neighbor function 和更新權重，所以可以在一個 clk 就完成所有事情



5) Show simulation result



6) Show SuperLint coverage (TOP.v)



99%

都是 unused 的訊號

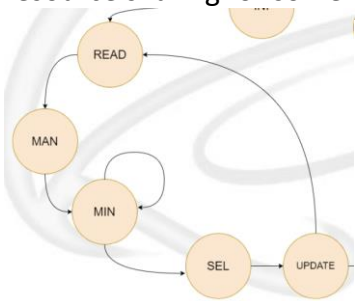
7) Your clock period, total cell area, post simulation time (TOP.v)

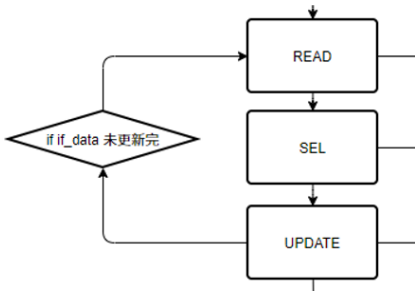
Clock period:10

Total cell area: 24657.273257

Post simulation time:164545ns

8) Please describe how you optimize your design when you run into problems in synthesis .ex: plug in some registers between two instances to shorten your datapath, resource sharing for some registers to reduce your cell area.





上圖可以看到原本助教的在更新 state 切成 6 個 state，但其實不用切到那麼細，我設計成 3 個 state 就可以更新一次 weight 了，甚至還有足夠的時間去壓 clock。

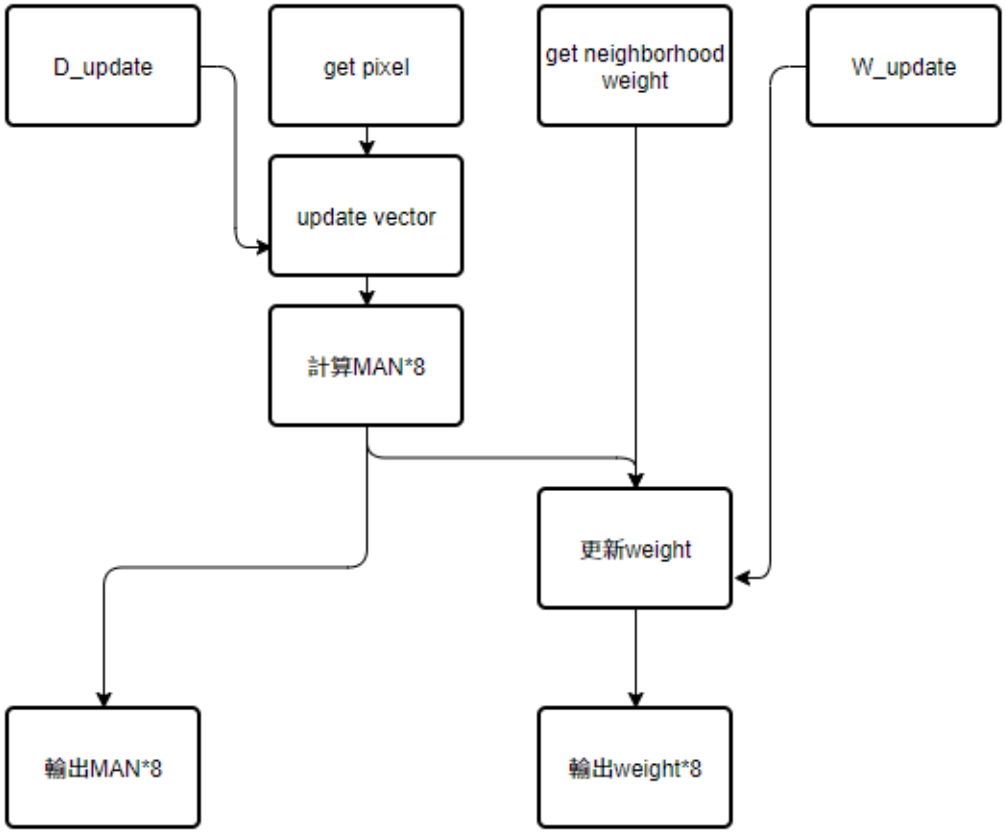
縮小面積的部分是我原本是用 counter 來控制 state 的切換，但後來發現 RAM_IF_A、RAM_W_A、RAM_RESULT_A 就是天然的 counter，直接利用他們來控制 state 的切換，省下另外建一個 counter 的 register。

➢ Lessons learned from this lab

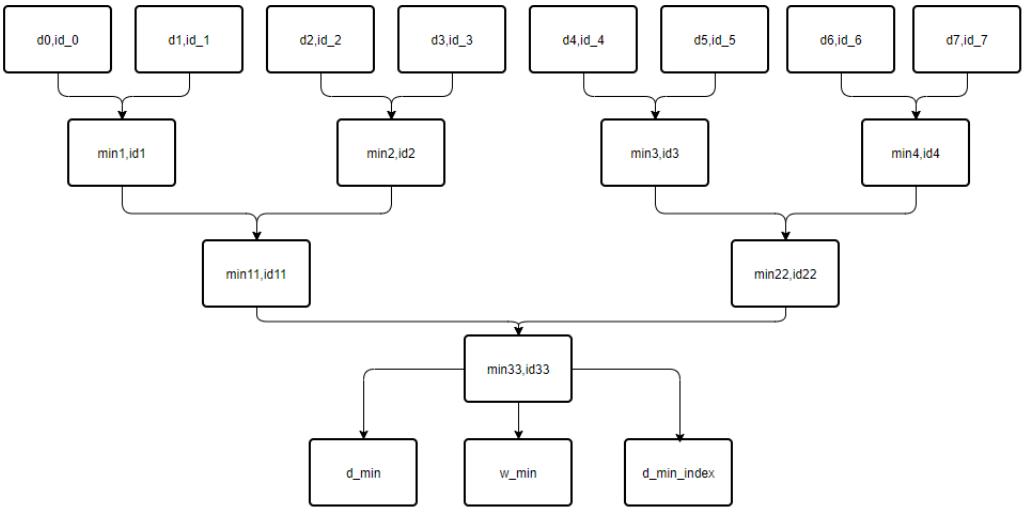
這次的電路架構是最大的一次，所以 debug 起來相對較難，寫架構花了 2 天，debug 也花了 2 天，尤其是更新權重的部分特別難 debug，因為我是中間有少部分地方算錯，所以導致最後結果錯，前面的波行都是對的，最後一個一個 module 檢查，才發現是自己要白癡複製貼上參數忘記改，經過這次其中 project，感覺比之前前幾個 lab 進步幅度多很多。

Lab 7_2

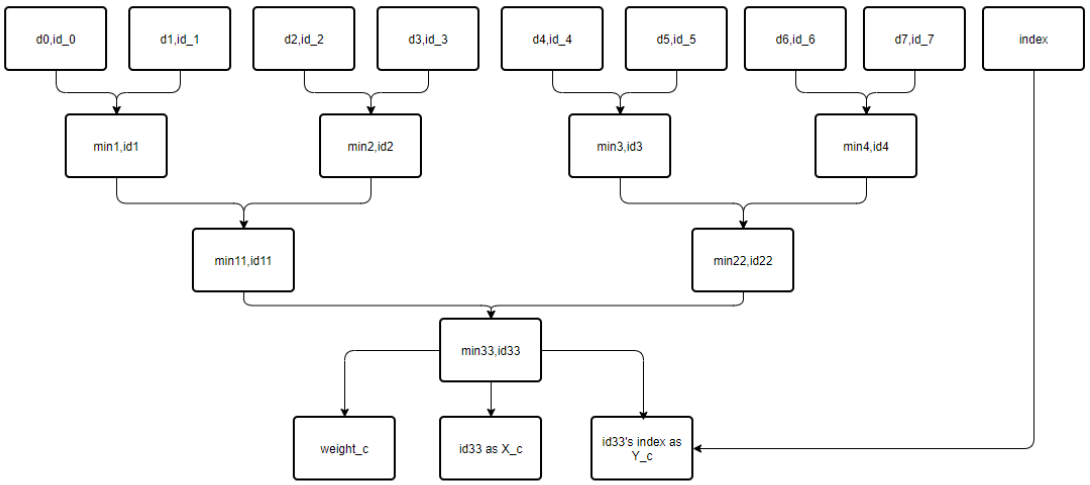
- Draw your state diagram and explain your design. You can draw internal architecture to describe your design
- Describe your design in detail. You can draw internal architecture or block diagram to describe your dsign.
  - VEP



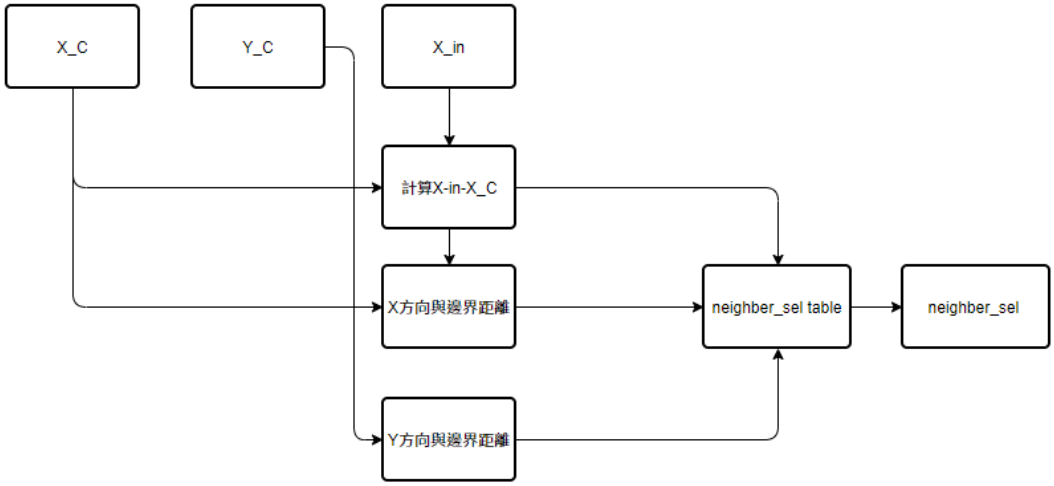
MIN_1


d0,id_0
d1,id_1
d2,id_2
d3,id_3
d4,id_4
d5,id_5
d6,id_6
d7,id_7
min1,id1
min2,id2
min3,id3
min4,id4
min11,id11
min22,id22
min33,id33
d_min
w_min
d_min_index

MIN_2

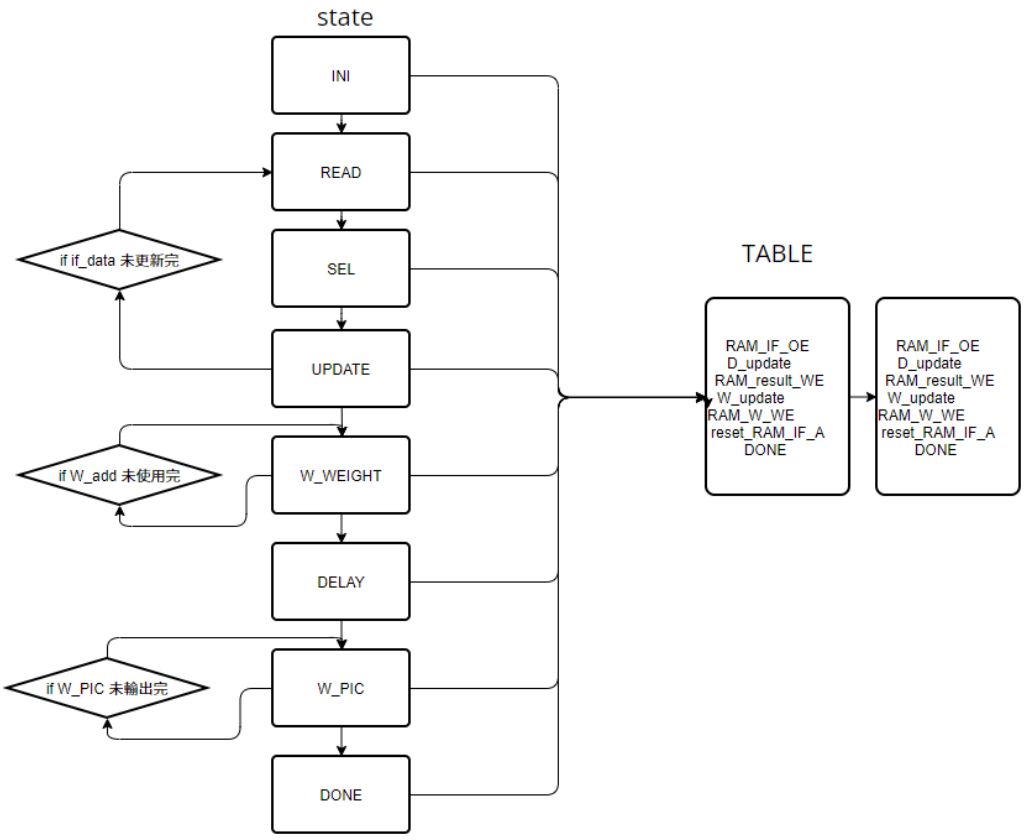

d0,id_0
d1,id_1
d2,id_2
d3,id_3
d4,id_4
d5,id_5
d6,id_6
d7,id_7
index
min1,id1
min2,id2
min3,id3
min4,id4
min11,id11
min22,id22
min33,id33
weight_c
id33 as X_c
id33's index as Y_c

■ USS



■ Controller

◆ Draw your state diagram in controller and explain it



9) Complete the Controller, VEP ,MIN_1, MIN_2, USS, and TOP module, in the system.

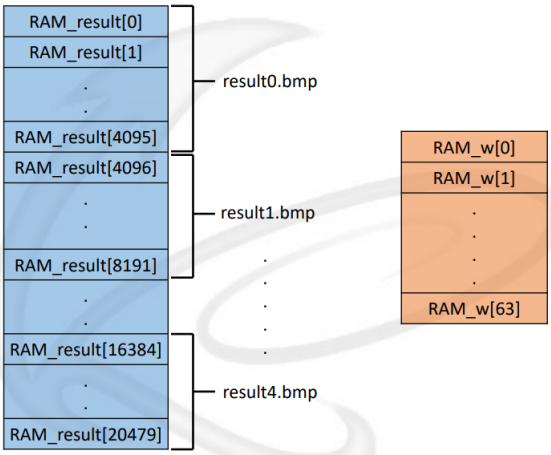
10) Compile the verilog code to verify the operations of this module works properly.

11) Synthesize your *TOP.v* with following constraint:

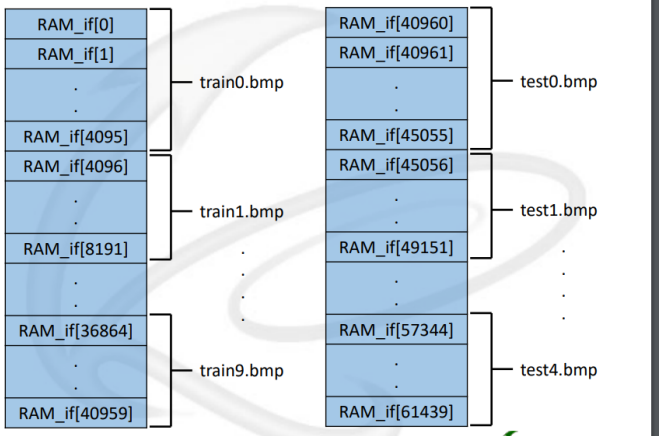
- Clock period: no more than 20 ns.

- Don’t touch network: clk.
- Wire load model: saed14rvt_ss0p72v125c.
- Synthesized verilog file: *top_syn.v*.
- Timing constraint file: *top_syn.sdf*.

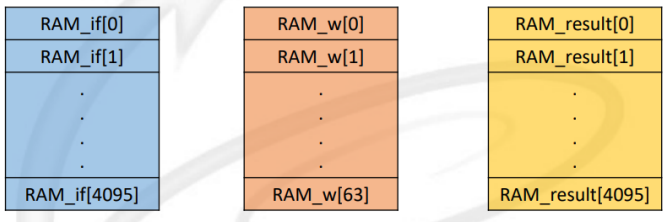
RAM


RAM_result[0]
RAM_result[1]
RAM_result[4095]
result0.bmp
RAM_result[4096]
RAM_result[8191]
result1.bmp
RAM_result[16384]
RAM_result[20479]
result4.bmp
RAM_w[0]
RAM_w[1]
RAM_w[63]

RAM

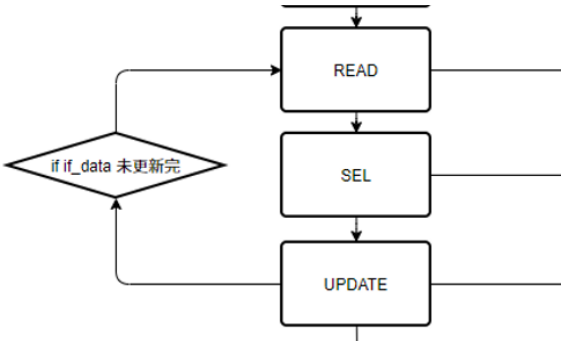

RAM_if[0]
RAM_if[1]
RAM_if[4095]
train0.bmp
RAM_if[4096]
RAM_if[8191]
train1.bmp
RAM_if[36864]
RAM_if[40959]
train9.bmp
RAM_if[40960]
RAM_if[40961]
RAM_if[45055]
test0.bmp
RAM_if[45056]
RAM_if[49151]
test1.bmp
RAM_if[57344]
RAM_if[61439]
test4.bmp

RAM

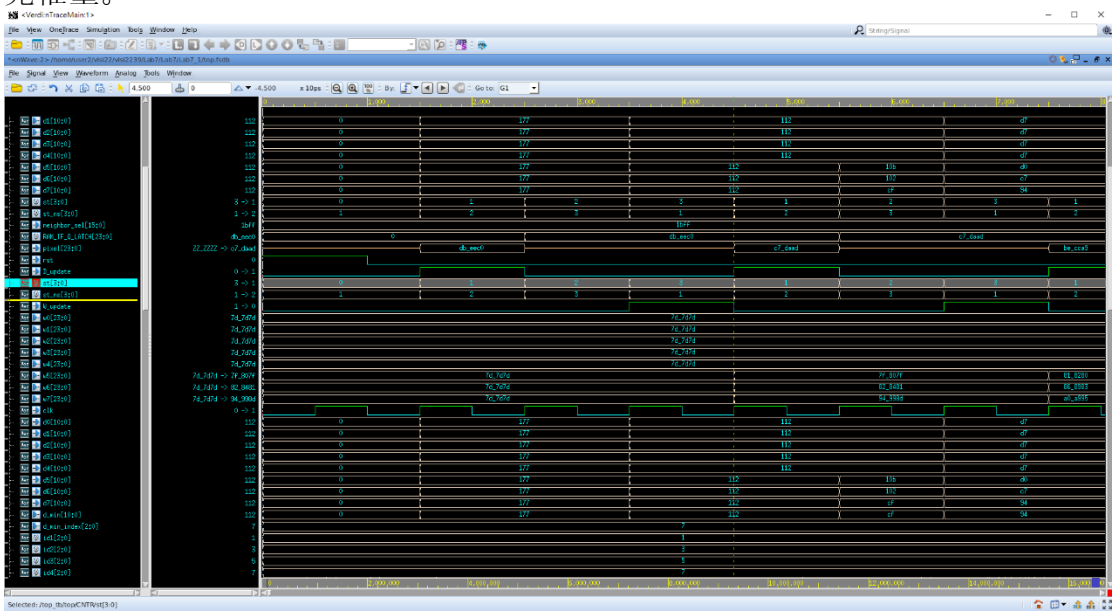

RAM_if[0]
RAM_if[1]
RAM_if[4095]
train0.bmp
RAM_w[0]
RAM_w[1]
RAM_w[63]
codebook
RAM_result[0]
RAM_result[1]
RAM_result[4095]
result0.bmp

12) Please **attach your waveforms** and **specify your operations** on the waveforms.

Lab7_1 和 Lab7_2 的區別是在 Train picture 和 Inference picture 的數量，Lab7_2 的數量分別是 Lab7_1 的 10 倍 Train picture，Inference picture 的 5 倍



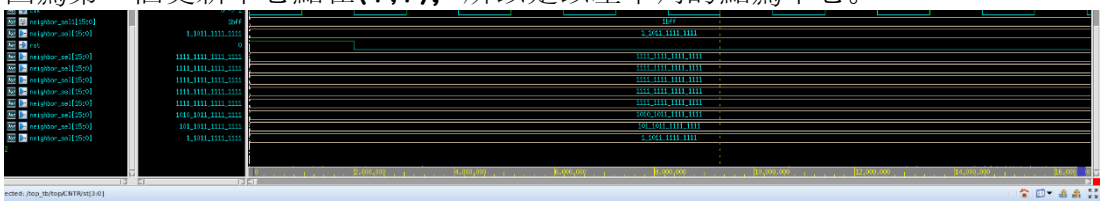
在訓練的部分，我分成 3 個 state，第 1 個 state 是 READ 主要是將資料讀進來，例如第一筆資料是 db_eec0，資料讀進來後，儲存到 RAM_IF_Q_LATCH，RAM_IF_Q_LATCH 會維持 db_eec0 直到 Update 完權重。

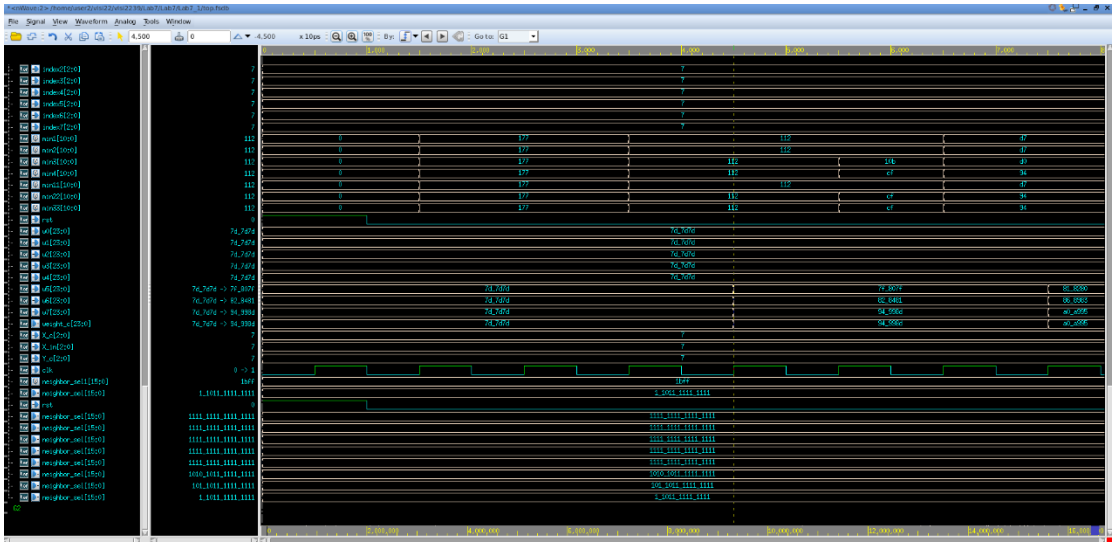


第 2 個 state 是 SEL，主要是計算的部分，這裡我設計的都是 combinational block，MIN_1、MIN_2、USS 都在 1 個 clk 裡，所以 1 個 clk 就可以算完 neighbor_sel，並輸出 neighbor_sel。

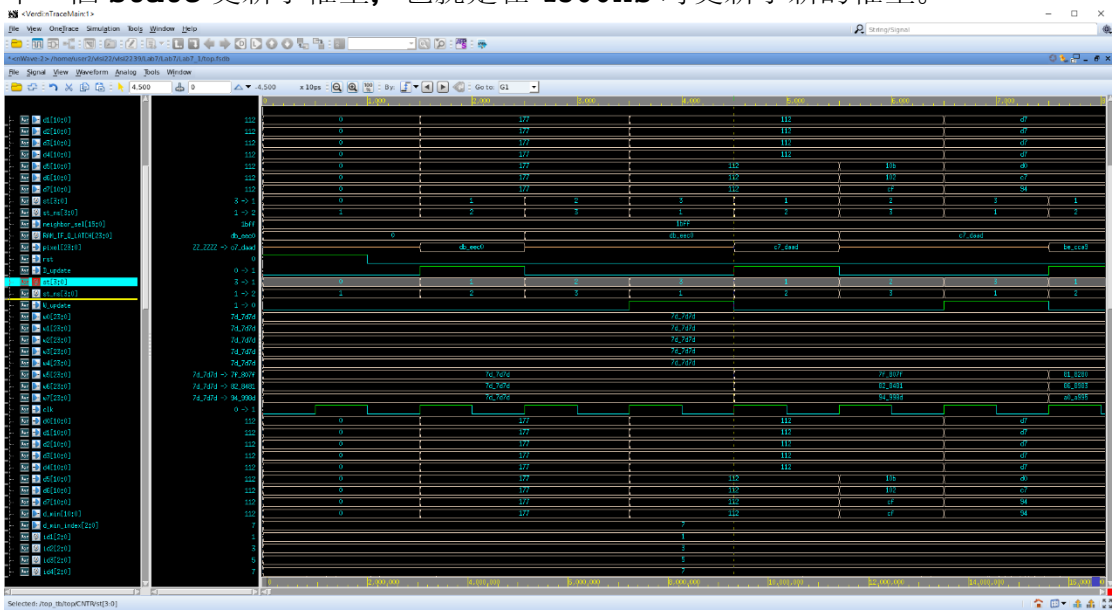
下圖可以看到完整的 neighbor function

因為第一個更新中心點在(7,7)，所以是以左下角的點為中心。

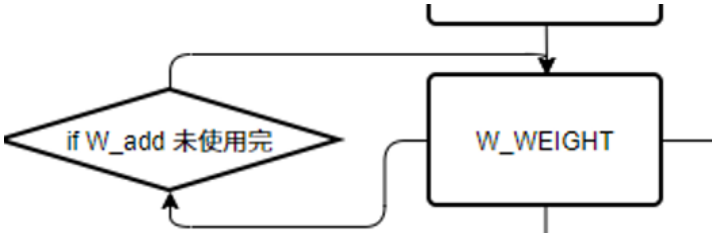




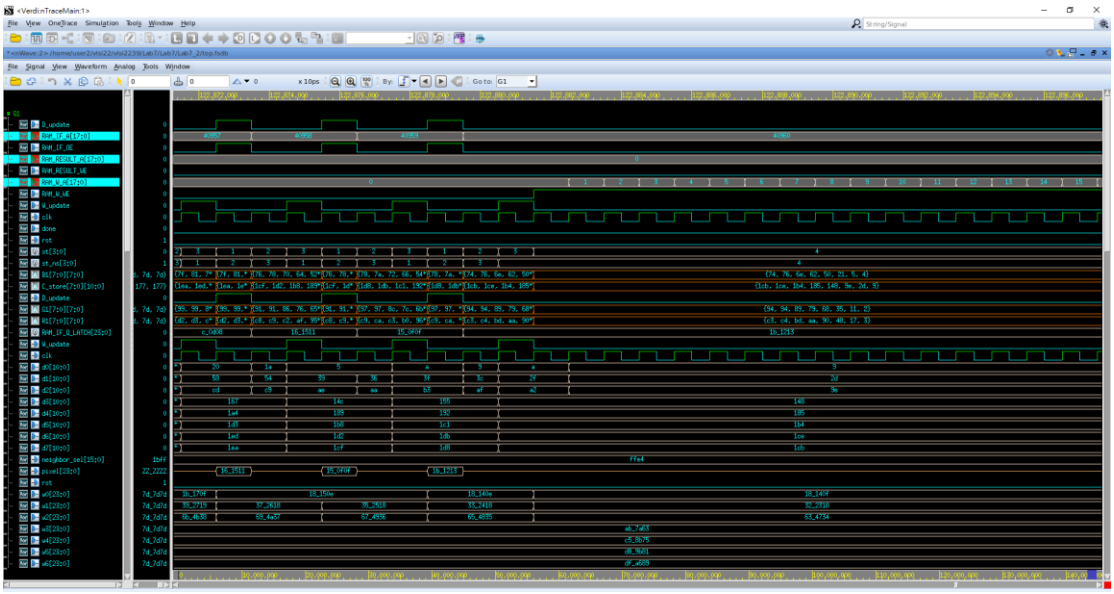
第三個 state 是 UPDATE，主要是更新權重，根據接收到 neighbor_sel 來更新權重，可以看到在 UPDATE state 時，W_update 升高，所以在下一個 state 更新了權重，也就是在 4500ns 時更新了新的權重。



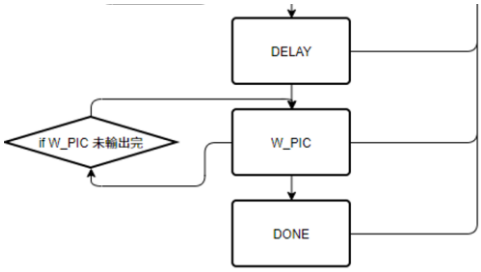
在訓練了 40960 次後，拿到了訓練完的權重，接著進到 W_WEIGHT，更新 codebook 資料，將更新的權重寫入 RAM_W_Q。



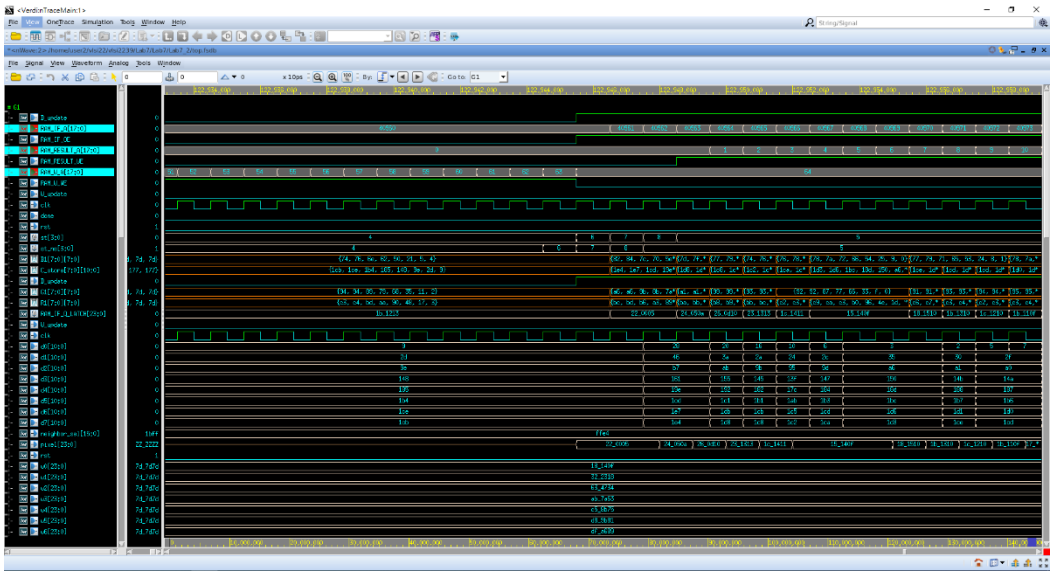
可以看到下圖的波型圖，進到 W_WEIGHT，RAM_W_WE 為 1，開始更新 codebook 資料，一共 64 組資料。



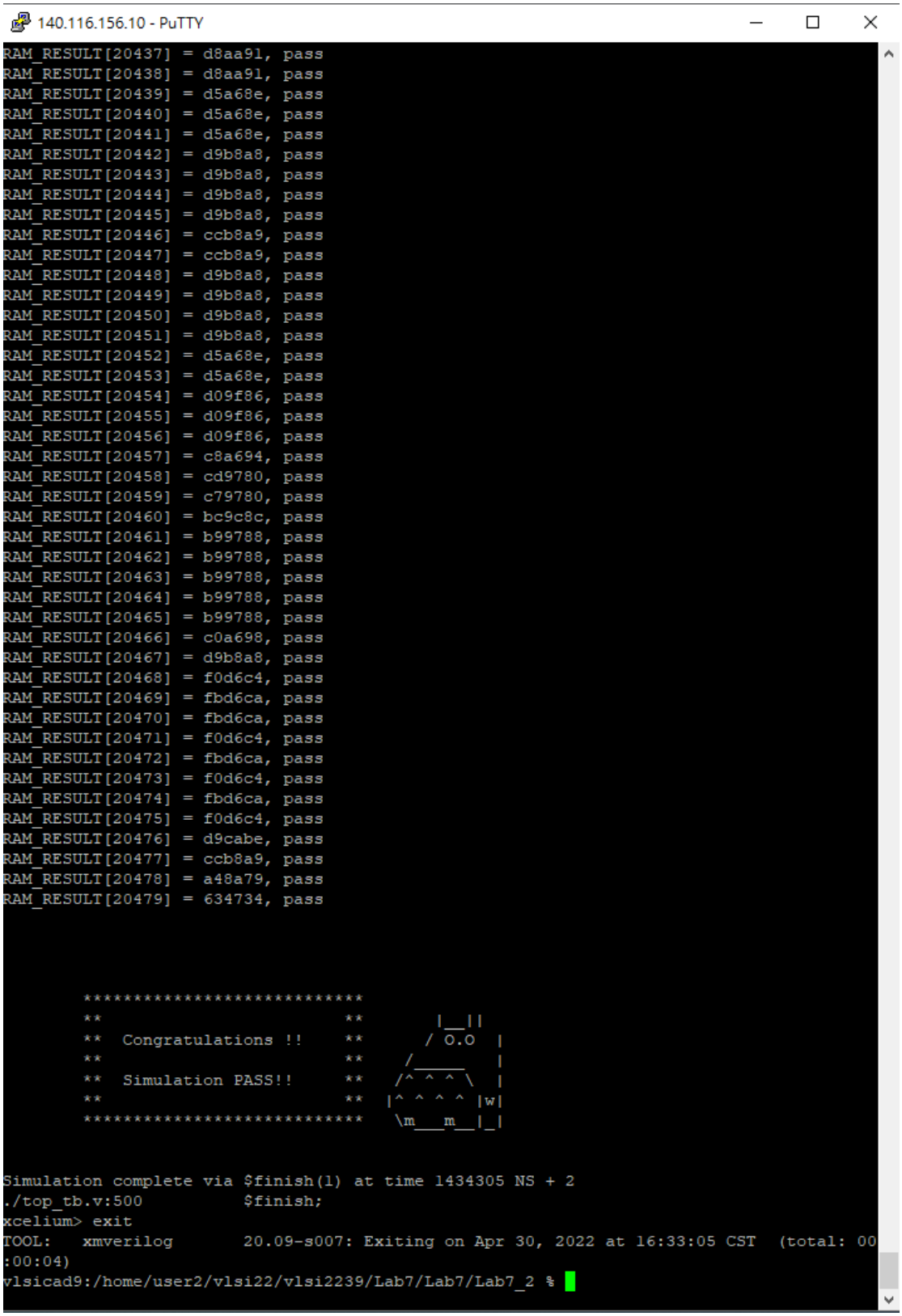
更新完 codebook 資料接著進到 DELAY, 這裡設計是因為有 delay clk 所以多設計一個 state 來符合傳輸資料的 clock。
這裡主要是 W_PIC。



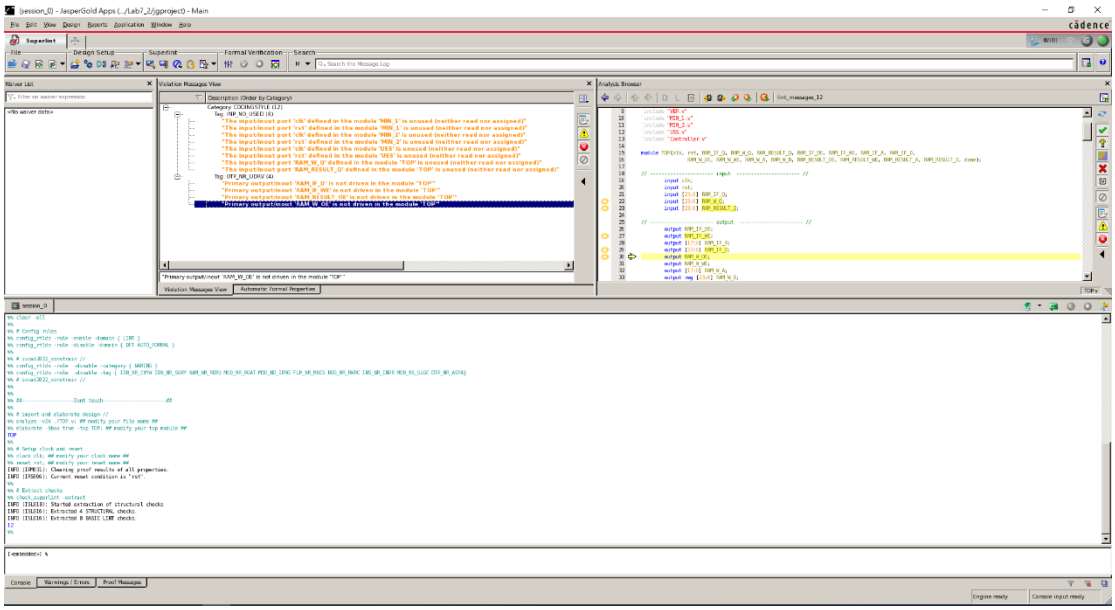
這裡 write compress picture result，可以使用 pipeline 的方式來更新，這裡資料傳遞方式是先從 RAM_IF 讀資料到 RAM_IF_Q_LATCH 然後進到 VEP，算 calculate manhattan distance，進到 MIN_1 比較最小的 index，最後進到 MIN_2 找中心點然後直接寫入 RAM_RESULT，這裡
跟 1~3 STATE 的差別就是不計算 neighbor function 和更新權重，所以可以在一個 clk 就完成所有事情



13) Show simulation result



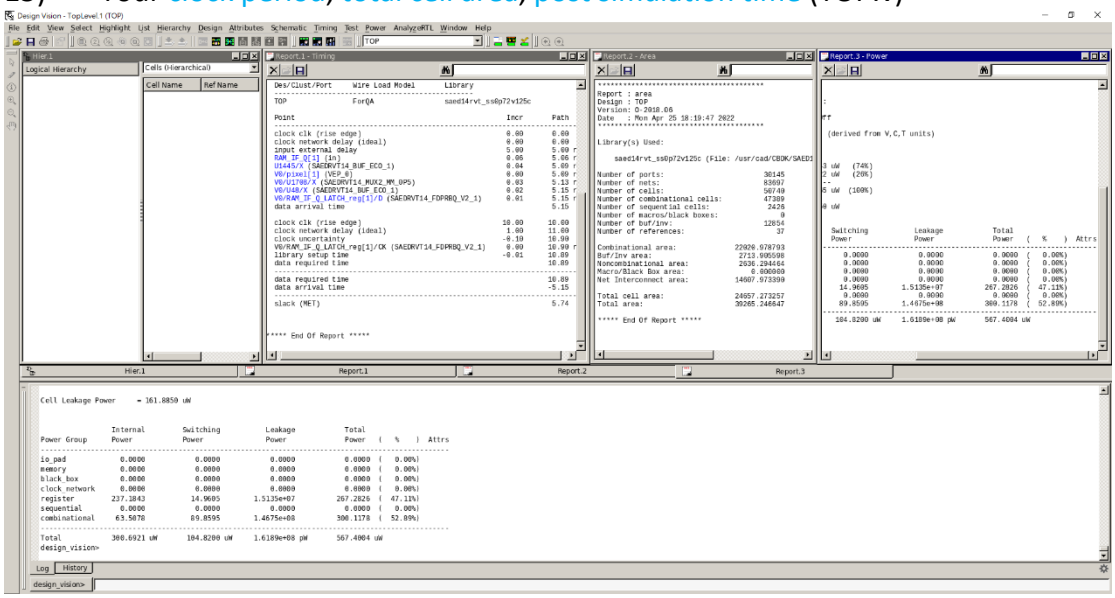
14) Show SuperLint coverage (TOP.v)



99%

都是 unused 的訊號

15) Your clock period, total cell area, post simulation time (TOP.v)



Clock period:10

Total cell area:24657.273257

Post simulation time:1434305ns

➢ Lessons learned from this lab

Please compress all the following files into one compressed file (".tar " format) and submit through Moodle website:

※ NOTE:

1. If there are other files used in your design, please attach the files too and make sure they're properly included.
2. Simulation command

Problem	Command
Lab7_1(pre-sim)	ncverilog top_tb.v +define+X (WEIGHT, RESULT, FULL)
Lab7_1 (pre-sim with waveform)	ncverilog top_tb.v +access+r +define+FSDB+X
Lab7_1(post-sim)	ncverilog top_tb.v +define+syn+X
Lab7_1 (post-sim with waveform)	ncverilog top_tb.v +access+r +define+FSDB+syn+X
Lab7_2(pre-sim)	ncverilog top_tb.v +define+X (WEIGHT, RESULT0, RESULT1, RESULT2, RESULT3, RESULT4, FULL)
Lab7_2 (pre-sim with waveform)	ncverilog top_tb.v +access+r +define+FSDB+X
Lab7_2(post-sim)	ncverilog top_tb.v +define+FSDB+syn+X
Lab7_2 (post-sim with waveform)	ncverilog top_tb.v +access+r +define+FSDB+syn+X