National Cheng Kung University

Department of Electrical Engineering

Introduction to VLSI CAD (Spring 2024)

Lab Session 6

Design of Local Binary Pattern Circuit

Name	Student ID	
游宗謀	E94106151	
Practical	Points	Marks
Lab 6_1	35	
Lab 6_2	65	
Notes		

Due: 15:00 April 17, 2024 @ moodle

Summary

Hardware		
	RTL(∨/X)	Synthesis(∨/X)
LBP	∨	∨
CLBP	∨	∨

Synthesis result	
Area	Simulation time (ps)
LBP: 182.568245um^2 CLBP: 4107.386977um^2	LBP: 77402034ps CLBP: 337872034ps

Superlint(number of inline messages)			
Total lines	Warning	Error	coverage(%)
LBP: 254 CLBP: 405	LBP: 0 CLBP: 19	LBP: 0 CLBP: 0	LBP: 100 CLBP: 95.31

Note: You must complete and fill out this form with your design information!!!

Deliverables

1) All Verilog codes including testbenches, .bmp and .hex for each problem should be uploaded.
2) NOTE: Please **DO NOT** include source code in the paper report!
3) NOTE: 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.

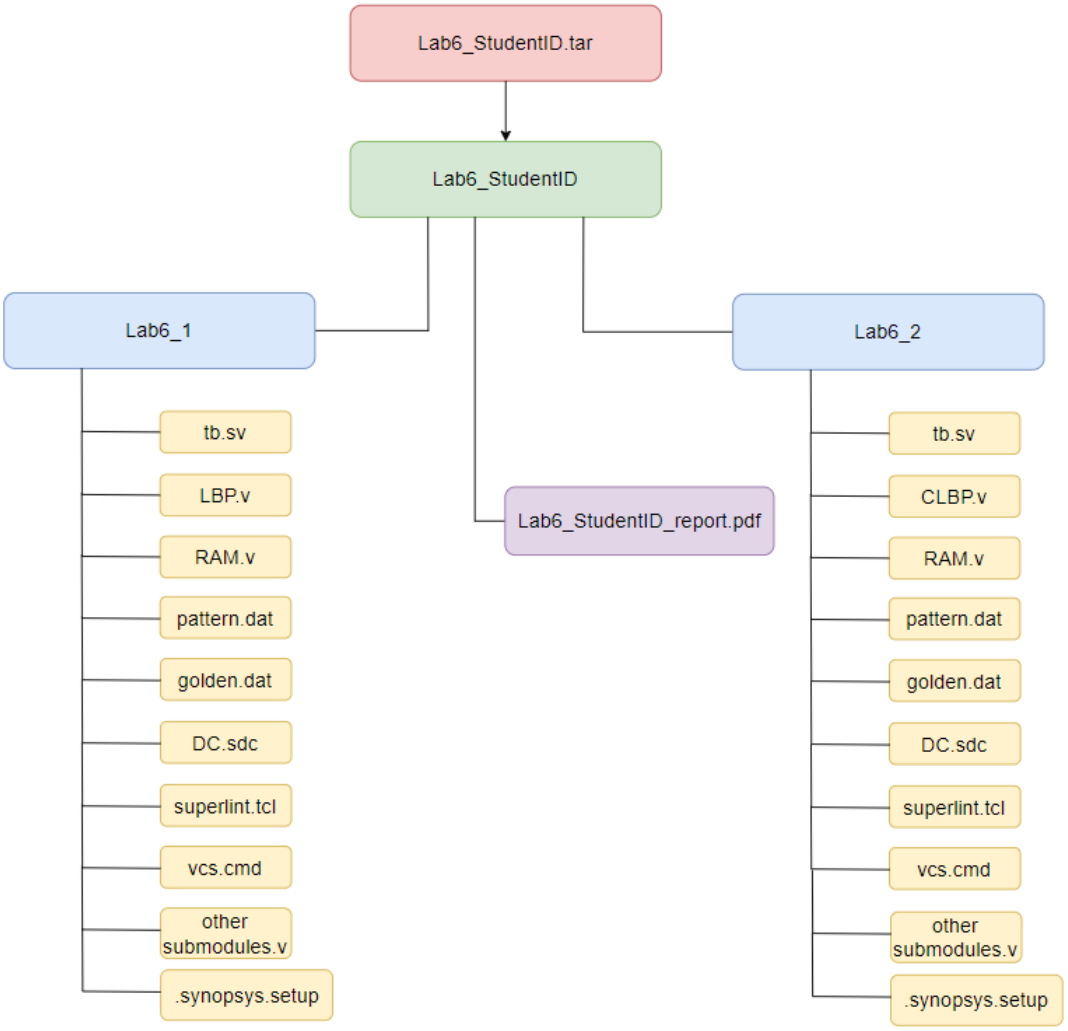


Fig.1 File hierarchy for Homework submission

The design inside the LBP block can be completed by your free will, but do not modify the I/O ports of the LBP block. The block diagram of the testbed-DUT (design under test) system is as shown in **Fig2**.

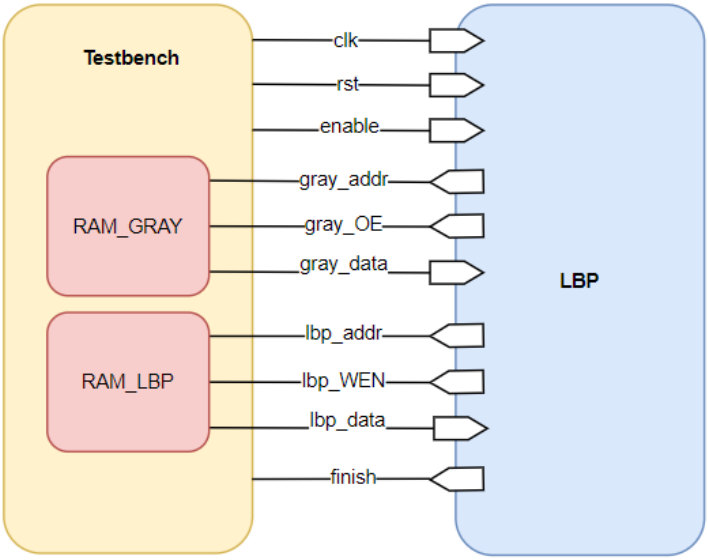


Fig2. The block diagram of local binary pattern circuit

➢ Port list of LBP:

Signal	I/O	Bit-width	Description
clk	I	1	Clock signal
rst	I	1	Reset signal
enable	I	1	Circuit enabling signal
gray_addr	O	12	Address signal connected to RAM_GRAY
gray_OE	O	1	Read enable signal to RAM_GRAY
gray_data	I	8	Read data signal from RAM_GRAY
lbp_addr	O	12	Address signal connected to RAM_LBP
lbp_WEN	O	1	Write enable signal to RAM_LBP
lbp_data	O	8	Write data signal to RAM_LBP
finish	O	1	Indication signal of the circuit is finished

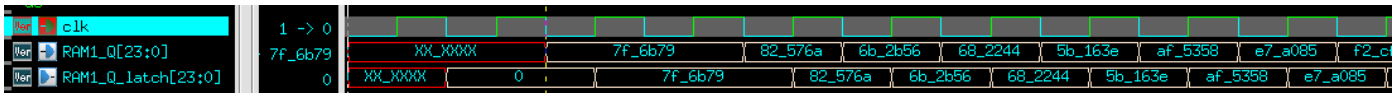


Fig3. example waveform for RAM

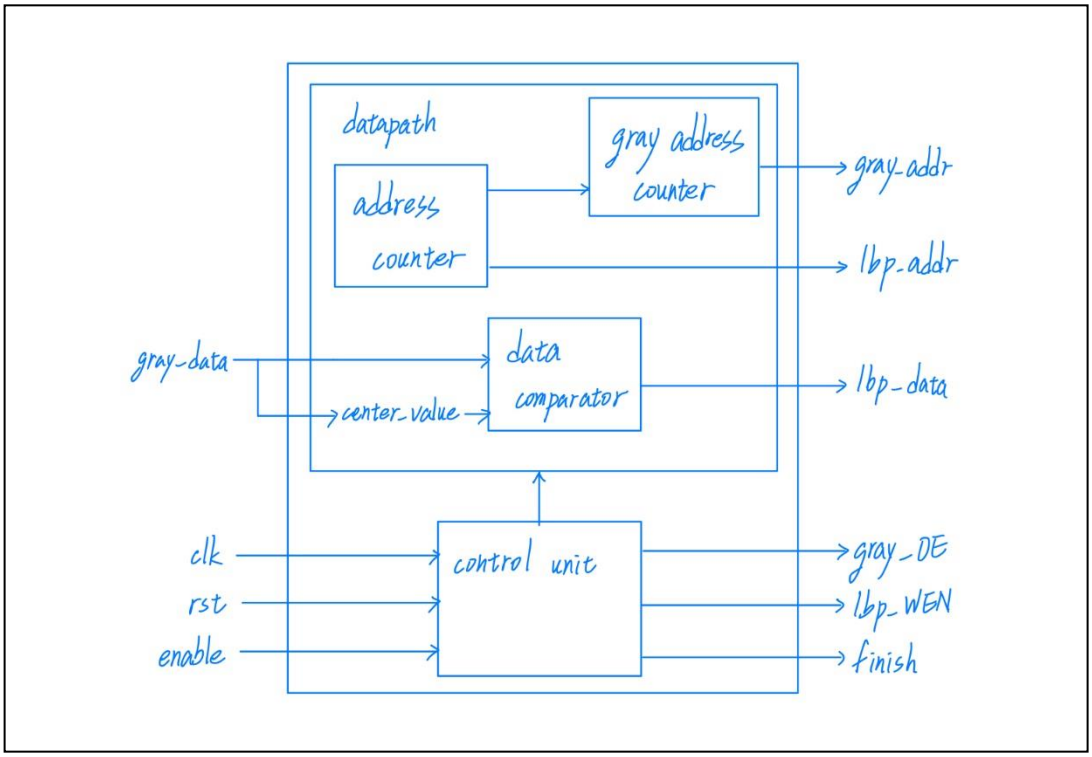
- Understanding the function:

  Once system is initialized, it

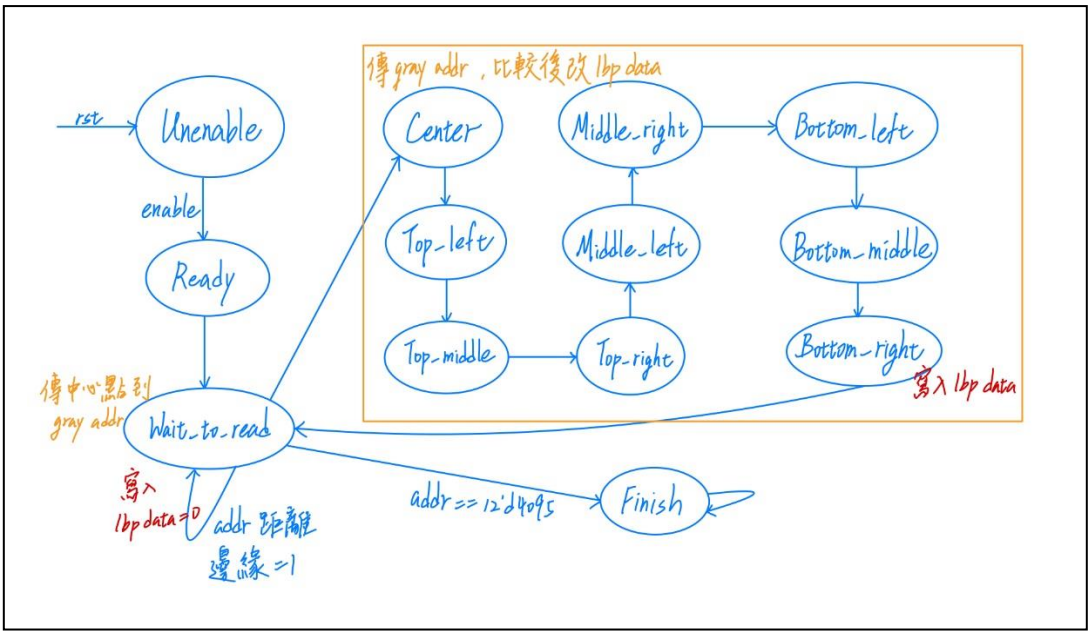
  a) Choose a pixel in the image and select its neighboring pixels.
  b) Construct the mask using threshold function.
  c) Combine the binary values for all neighboring pixels to obtain a binary code for the central pixel and convert it to a decimal value.
  d) Repeat steps a)–c) for each pixel in the image to obtain a binary code for each pixel.

- Know the basic design rules
  - All operations are activated on the positive edge of the clock.
  - Control signals:
    - *RAM_WE*: To store the data into RAM
    - *RAM_OE*: To read data from RAM
    - *finish*: Stop the process

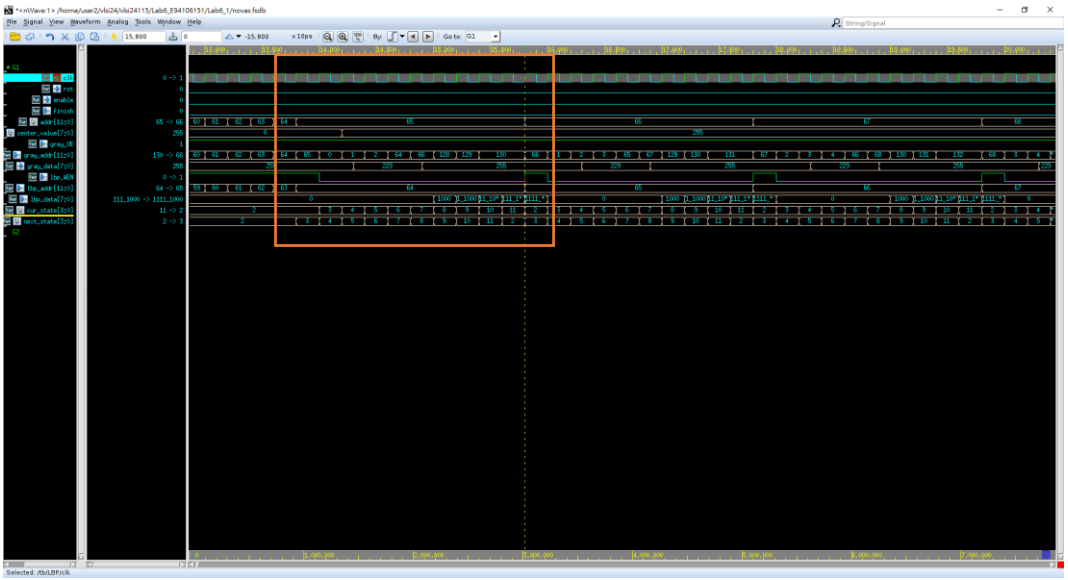
- Describe your design in detail. You can draw internal architecture or block diagram to help elaborate your design, if don't, plain text description is allowed.



  - Controller
    - Draw your state diagram in controller and explain it.



1. Complete the LBP module, in the system.
2. Compile the verilog code to verify the operations of this module works properly.
3. Synthesize your *LBP.v* with following constraint:
   - Clock period: no more than 2.0 ns.
   - Don't touch network: clk.
   - Wire load model: N16ADFP_StdCellss0p72vm40c.
   - Synthesized verilog file: *LBP_syn.v*.
   - Timing constraint file: *LBP_syn.sdf*.
4. Please **attach your waveforms** and **specify your operations** on the waveforms.



addr=64:

cur_state=2，傳 gray_addr=addr+1、addr=addr+1，距離邊緣=1，所以輸出 lbp_data=0，lbp_wen=1，next_state=2。

addr=65:

cur_state=2，傳 gray_addr=addr-65，初始化 lbp_data、lbp_wen，next_state=3。

cur_state=3，傳 gray_addr=gray_addr+1，center_value=gray_data，next_state=4。

cur_state=4，傳 gray_addr=gray_addr+1，center_value>gray_data，next_state=5。

cur_state=5，傳 gray_addr=gray_addr+62，center_value>gray_data，next_state=6。

cur_state=6，傳 gray_addr=gray_addr+2，center_value>gray_data，next_state=7。

cur_state=7，傳 gray_addr=gray_addr+62，center_value<=gray_data，lbp_data[3]=1，next_state=8。

cur_state=8，傳 gray_addr=gray_addr+1，center_value<=gray_data，lbp_data[4]=1，next_state=9。

cur_state=9，傳 gray_addr=gray_addr+1，center_value<=gray_data，lbp_data[5]=1，next_state=10。

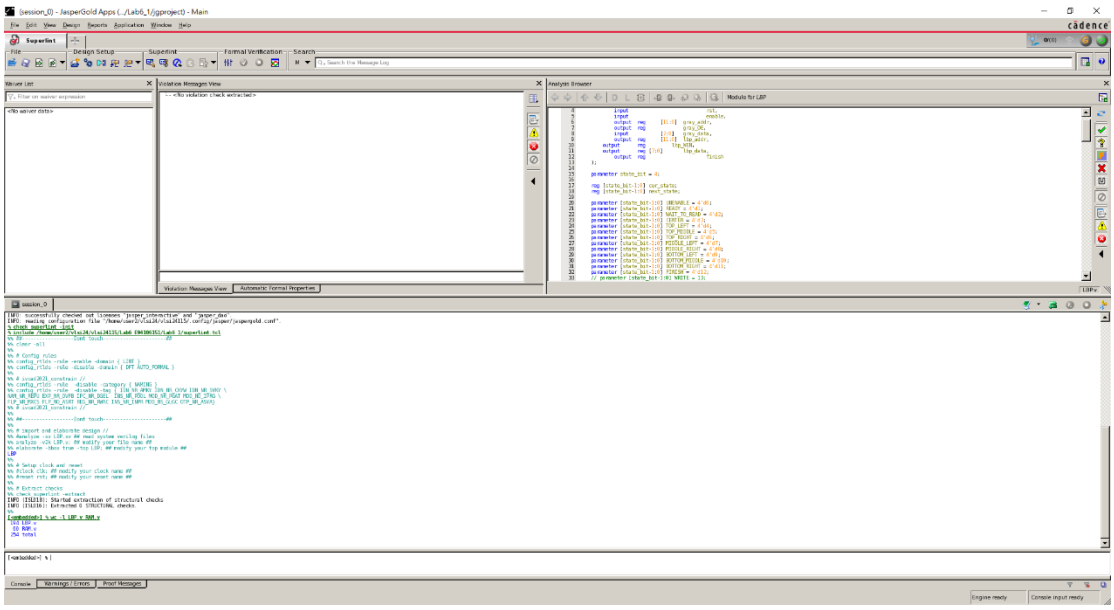
cur_state=10，center_value<=gray_data，lbp_data[6]=1，next_state=11。

cur_state=11，center_value<=gray_data，lbp_data[7]=1，輸出 lbp_addr=addr，lbp_wen=1，addr=addr+1，gray_addr=addr+1，next_state=2。

其他循環也是同一操作

5. Show SuperLint coverage (including all files)

   Coverage = (1-(0/254))*100% = 100%

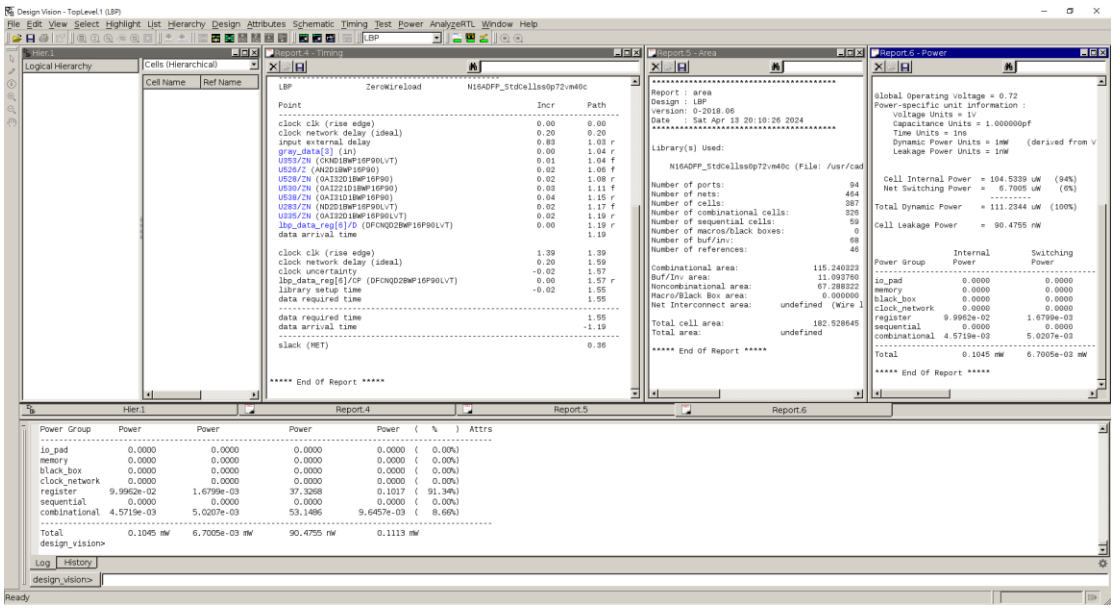


6. Your clock period, total cell area, post simulation time with screenshot.

Clock period: 1.39ns

Total cell area: 182.568245um^2

Post simulation time: 77402034ps



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

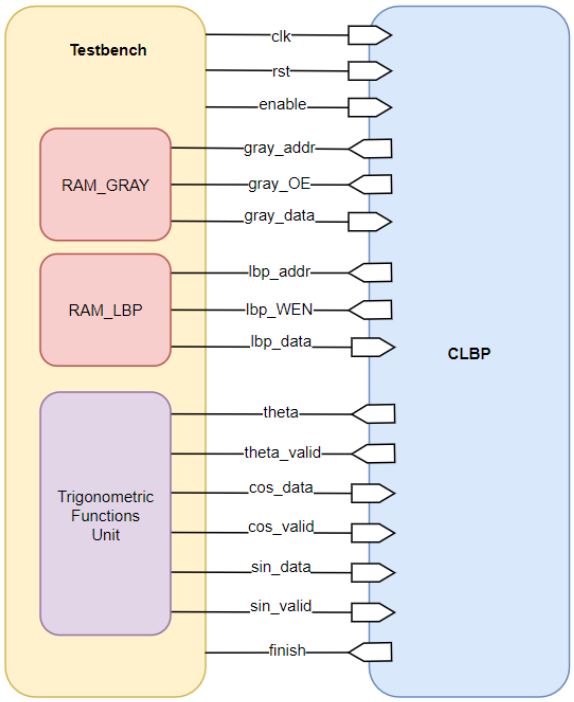
透過共用 addr 以減少 registers 的使用，來降低 cell area。並且在前一個 state 就先輸出 gray_addr 減少等待 RAM_GRAY 回傳 gray_data 的時間，來降低 clk period 的數量。

8. Lessons learned from this lab

這次的 lab 讓我更加熟悉 LBP 演算法相關的題目，並且新學到因為 lbp_data 是二進位，所以可以直接透過更改 lbp_data 某 bit 的值來達成 2 的某次方的加法 (lbp_data[6]<=1'd1)。

Lab 6_2: Circular Local Binary Pattern

Extend the original LBP algorithm to circular one.



▲The block diagram of circular local binary pattern circuit

➢ Port list of top:

Signal	I/O	Bit-width	Description
clk	**I**	**1**	**Clock signal**
rst	**I**	**1**	**Reset signal**
enable	**I**	**1**	**Circuit enabling signal**
gray_addr	**O**	**12**	**Address signal connected to RAM_GRAY**
gray_OE	**O**	**1**	**Read enable signal to RAM_GRAY**
gray_data	**I**	**8**	**Read data signal from RAM_GRAY**
lbp_addr	**O**	**12**	**Address signal connected to RAM_LBP**

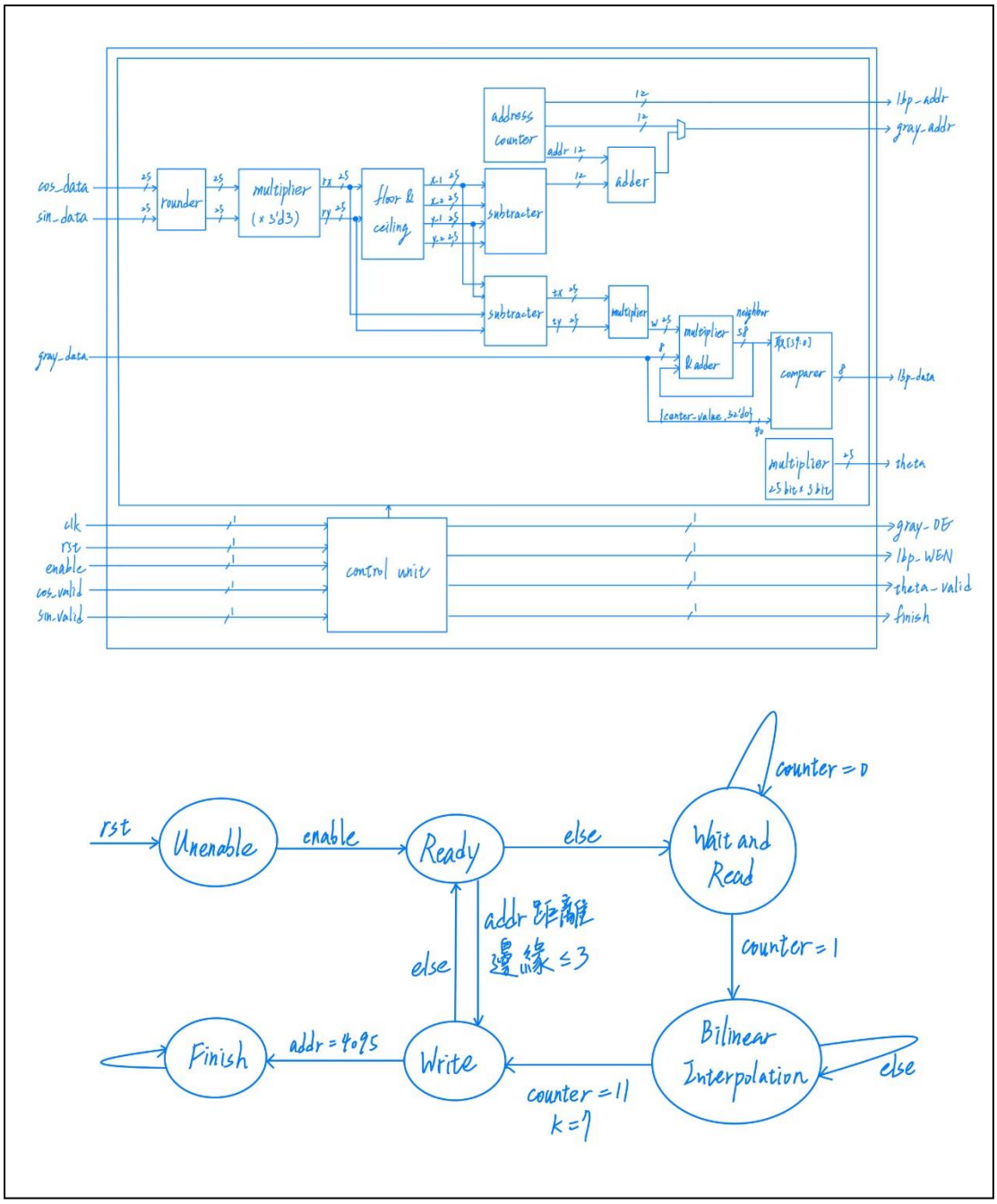
lbp_WEN	**O**	**1**	**Write enable signal to RAM_LBP**
lbp_data	**O**	**8**	**Write data signal to RAM_LBP**
theta	**O**	**25(fixed-point)**	**Current neighbor's angle signal(unit is in radian)**
theta_valid	**O**	**1**	**Indication signal of current neighbor's angle is valid**
cos_data	**I**	**25(fixed-point)**	**Cosine value of the theta(from testbench)**
cos_valid	**I**	**1**	**Indication signal of cosine value is valid**
sin_data	**I**	**25(fixed-point)**	**Sine value of the theta(from testbench)**
sin_valid	**I**	**1**	**Indication signal of sine value is valid**
finish	**O**	**1**	**Indication signal of the circuit is finished**

- Understanding the function:

  Once system is initialized, it

  a) Choose a pixel(center) in the image and select its neighboring pixels.
  b) Calculate bilinear interpolation:
     1) Determine r_x & r_y.
     2) Determine x_1, x_2, y_1, y_2.
     3) Determine t_x, t_y.
     4) Determine w1,w2, w3,w4.
     5) Determine $f(0,0)$, $f(0,1)$, $f(1,0)$, $f(1,1)$.
     6) Determine neighbor.
  c) Repeat b) to calculate all neighbors' values.
  d) Construct the mask using threshold function.
  e) Combine the binary values for all neighboring pixels to obtain a binary code for the central pixel and convert it to a decimal value.
  f) Repeat steps a)–e) for each pixel in the image to obtain a binary code for each pixel.

- Draw your state diagram and explain your design. You can draw internal architecture to describe your design.



1. Complete the CLBP module.
2. Compile the verilog code to verify the operations of this module works properly.
3. Synthesize your *CLBP.v* with following constraint:
   - Clock period: no more than 2.0 ns.
   - Don't touch network: clk.
   - Wire load model: N16ADFP_StdCellss0p72vm40c.
   - Synthesized verilog file: C*LBP_syn.v*.
   - Timing constraint file: C*LBP_syn.sdf*.
4. Please **attach your waveforms** and **specify your operations** on the waveforms.

addr=231:

cur_state=4, then output lbp_data, addr+1, next_state=1

addr=232:

cur_state=1, then initial output reg, next_state=2

cur_state=2, counter=0, then counter+1, next_state=2

cur_state=2, counter=1, then counter=0, center_value=gray_data=10011100, next_state=3

cur_state=3, counter=0, then counter+1, output theta=0

cur_state=3, counter=1, then counter+1

cur_state=3, counter=2, then counter+1, get cos_data and sin_data: rx=cos_data=65536, ry=sin_data=0

cur_state=3, counter=3, then counter+1, multiply R(3): rx=rx*3, ry=ry*(-3)

cur_state=3, counter=4, then counter+1, get x_1, y_1, x_2, y_2 by floor and ceiling of rx, ry: x_1=196608, x_2=196608, y_1=0, y_2=0

cur_state=3, counter=5, then counter+1, calculate tx, ty, output gray_addr=addr+x_1+y_1

cur_state=3, counter=6, then counter+1, output gray_addr=addr+x_1+y_2, w=(1-tx)*(1-ty)

cur_state=3, counter=7, then counter+1, output gray_addr=addr+x_2+y_1, w=tx*(1-ty), neighbor=neighbor+gray_data*w

cur_state=3, counter=8, then counter+1, output gray_addr=addr+x_2+y_2, w=(1-tx)*ty, neighbor=neighbor+gray_data*w

cur_state=3, counter=9, then counter+1, neighbor=neighbor+gray_data*w, w=tx*ty

cur_state=3, counter=10, then counter+1, neighbor=neighbor+gray_data*w

cur_state=3, counter=11, then counter+1, neighbor[39:0]<={center_value, 32'd0}, lbp_data[0]=0

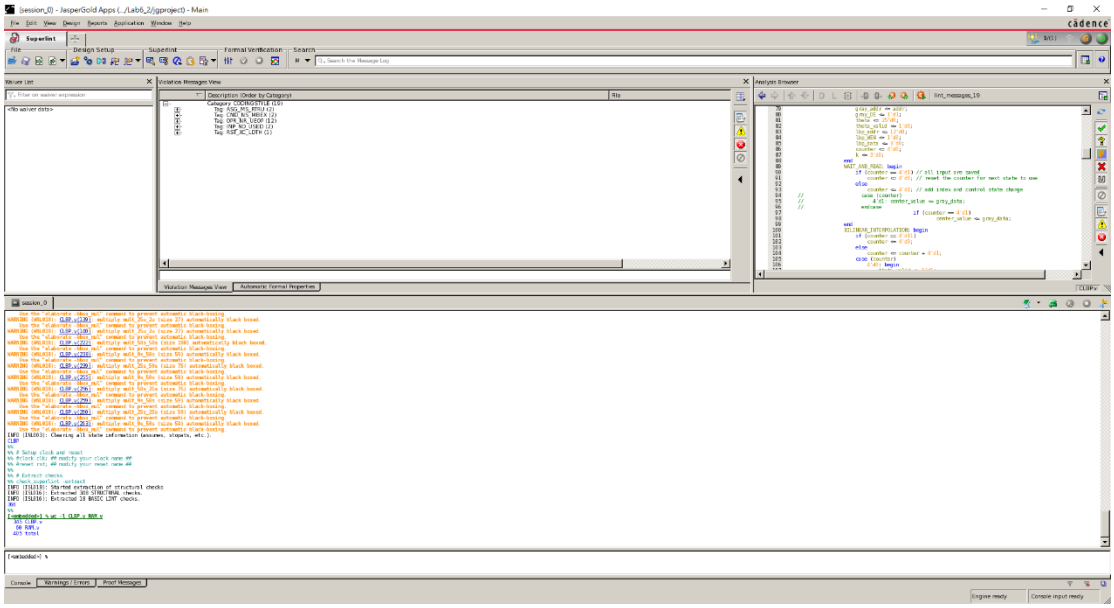
再循環 cur_state=1~3 七次, 算出 lbp_data

cur_state=4, output lbp_data, next_state=1

其他也是一直循環下去

5. Show SuperLint coverage (include all files)

Coverage = (1-(19/405))*100% = 95.31%

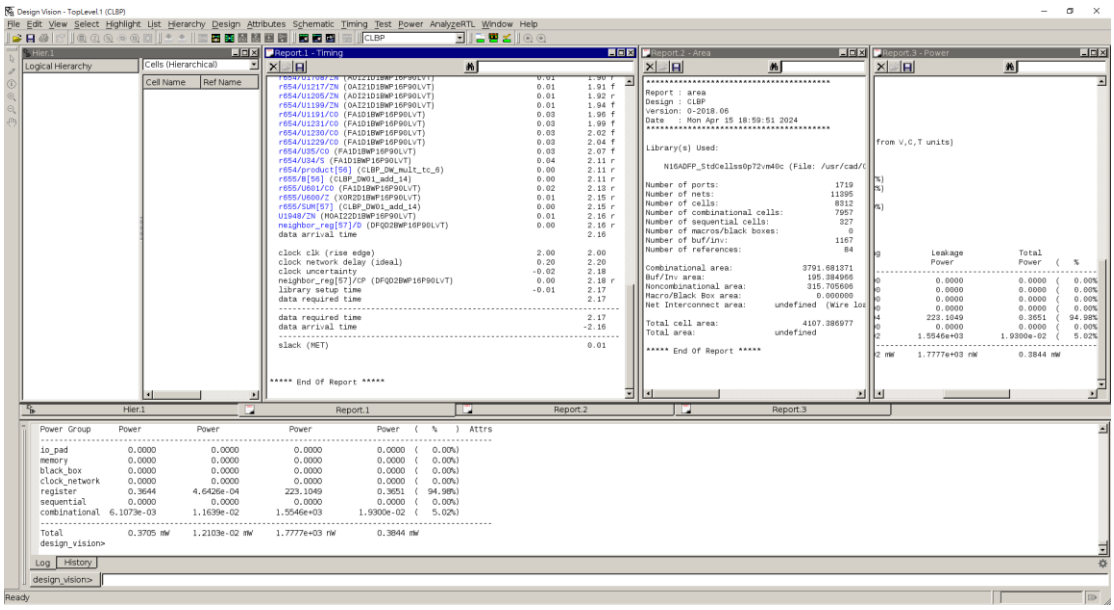


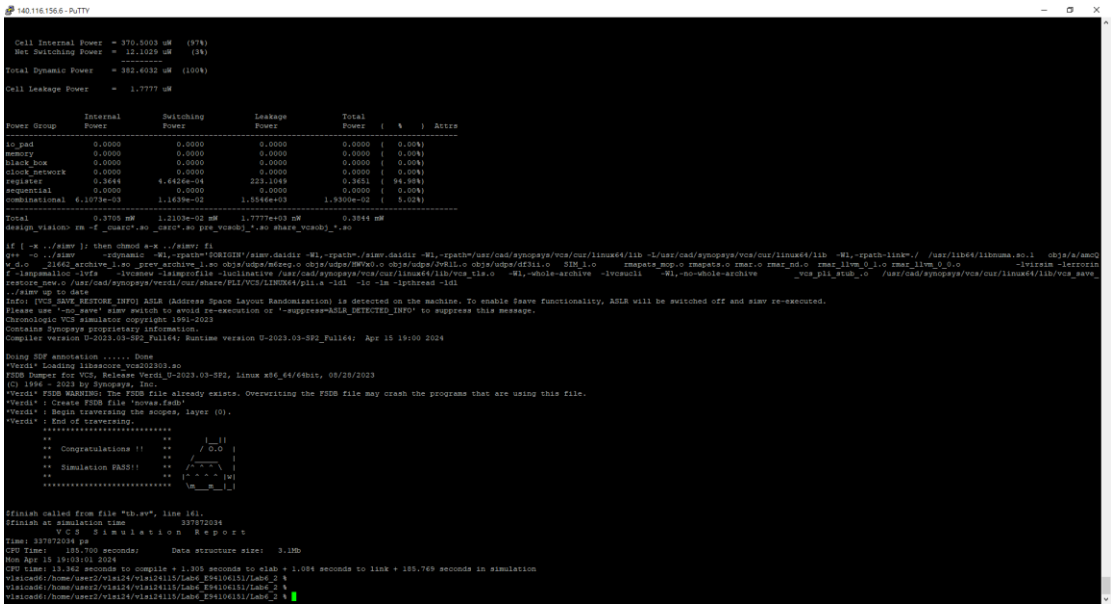
6. Your clock period, total cell area, post simulation time with screenshot

Clock period: 2ns

Total cell area: 4107.386977um^2

Post simulation time: 337872034ps





7. Lessons learned from this lab

這次的 lab 讓我更加熟悉 CLBP 演算法相關的題目，讓我學到在寫 verilog 的時候要檢查 signed 和 unsigned 的加減法，signed 和 unsigned 相加減時要先將負的 signed 轉成正的，然後再扣掉，才會算出正確的值。

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
Lab6_1(pre)	vcs -R -full64 -sverilog tb.sv +access+r +vcs+fsdbon
Lab6_1(post)	vcs -R -full64 -sverilog tb.sv +access+r +vcs+fsdbon +define+SDF+SYN
Lab6_2(pre)	vcs -R -full64 -sverilog tb.sv +access+r +vcs+fsdbon
Lab6_2(post)	vcs -R -full64 -sverilog tb.sv +access+r +vcs+fsdbon +define+SDF+SYN