Summary

Hardware			
	RTL All Pass(∨/X)	SYN All Pass(∨/X)	
Bicubic Resize	V	V	
	RTL Number of Errors	Synthesis Number of Errors	
Pattern 1	0	0	
Pattern 2	0	0	
Pattern 3	0	0	
Pattern 4	0	0	
Pattern 5	0	0	
Pattern 6	0	0	
Grade Class			
Total Number of Errors	Class (A/B/C/D/E)		
0	A		
Synthesis result			
Area	P6 Simulation time (ps)	Area * P6 Simulation time (ps)	
11665	3961251629	46208000252285	
Superlint(number of inline messages)			
Total lines	Warning	Error	coverage(%)
421	30	0	92.8%

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

Deliverables

1) All Verilog codes including testbenches, .bmp and .txt 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.

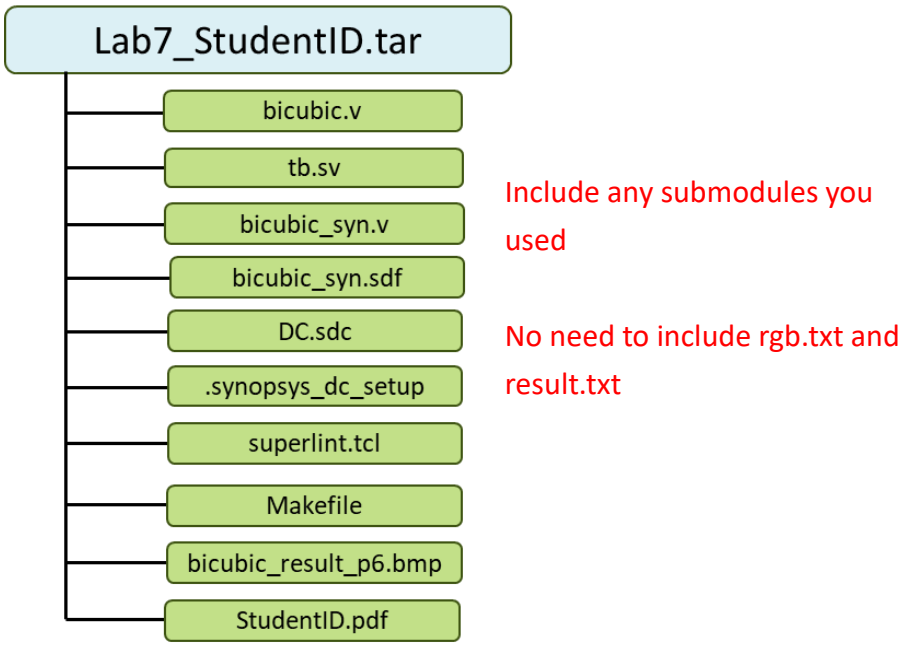


Fig.1 File hierarchy for Homework submission

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

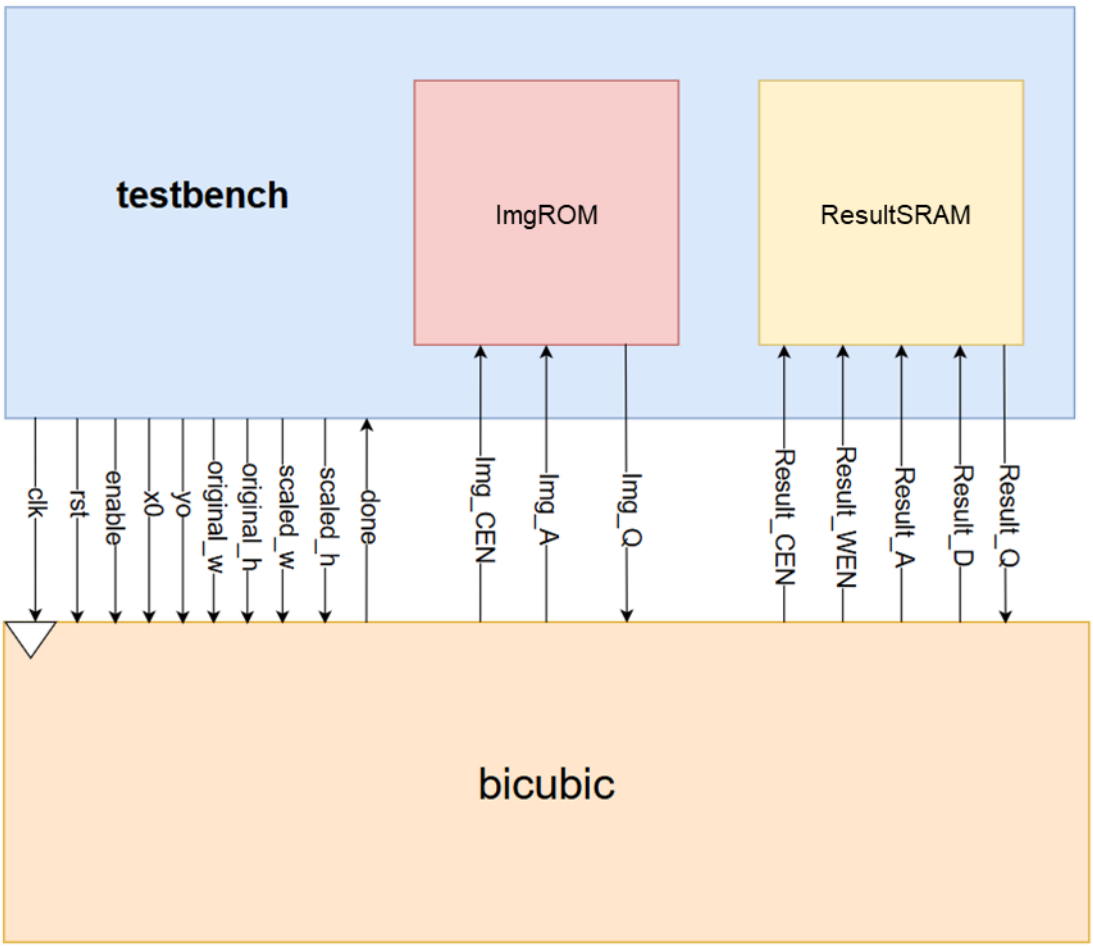


Fig2. The block diagram of bicubic circuit

➢ Port list of Bicubic Resize Engine:

Signal	Type	Bits	Description
clk	input	1	Positive-edged triggered
rst	input	1	Synchronous active high
enable	input	1	Active high enable signal to start processing
Img_CEN	output	1	Active low read enable signal for ImgROM
Img_A	output	14	14 bits address for ImgROM
Img_Q	input	32	Read 32 bits unsigned data from ImgROM
Result_WEN	output	1	(Low write/high read) enable signal for ResultSRAM
Result_A	output	16	16 bits address for ResultSRAM
Result_D	output	8	Write 8 bits unsigned data to ResultSRAM
Result_Q	input	8	Read 8 bits unsigned data from ResultSRAM
Result_CEN	output	1	Active low chip enable signal for ResultSRAM
x0	input	8	The horizontal coordinate value at the top-left corner of the region to be processed
y0	input	8	The vertical coordinate at the top-left corner of the region to be processed
original_w	input	8	The horizontal width of the region to be processed
original_h	input	8	The vertical height of the region to be processed
scaled_w	input	8	The horizontal width of the region after being enlarged
scaled_h	input	8	The vertical height of the region after being enlarged
done	output	1	Active high finish signal

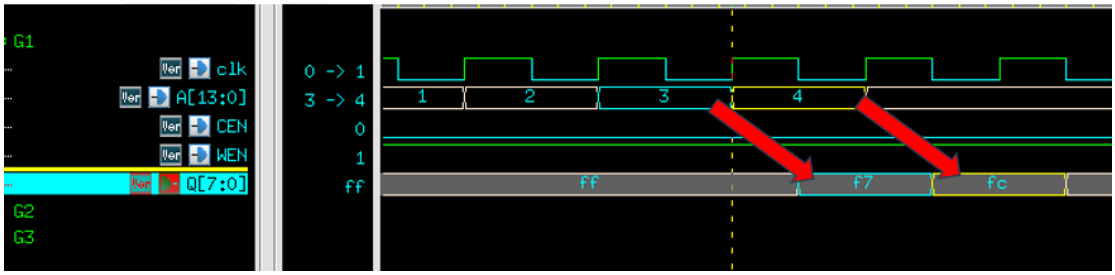


Fig3. example waveform for RAM read operation

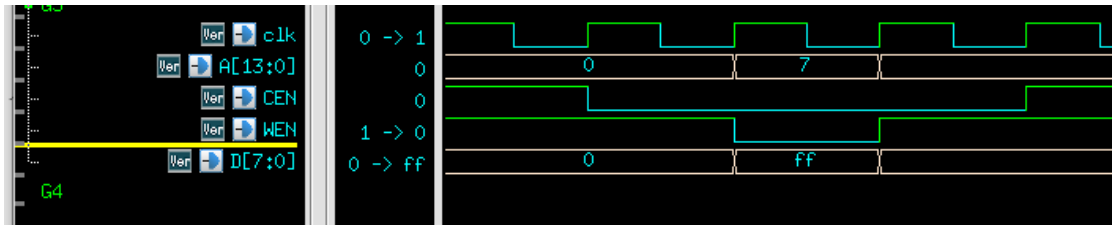


Fig4. example waveform for RAM write operation

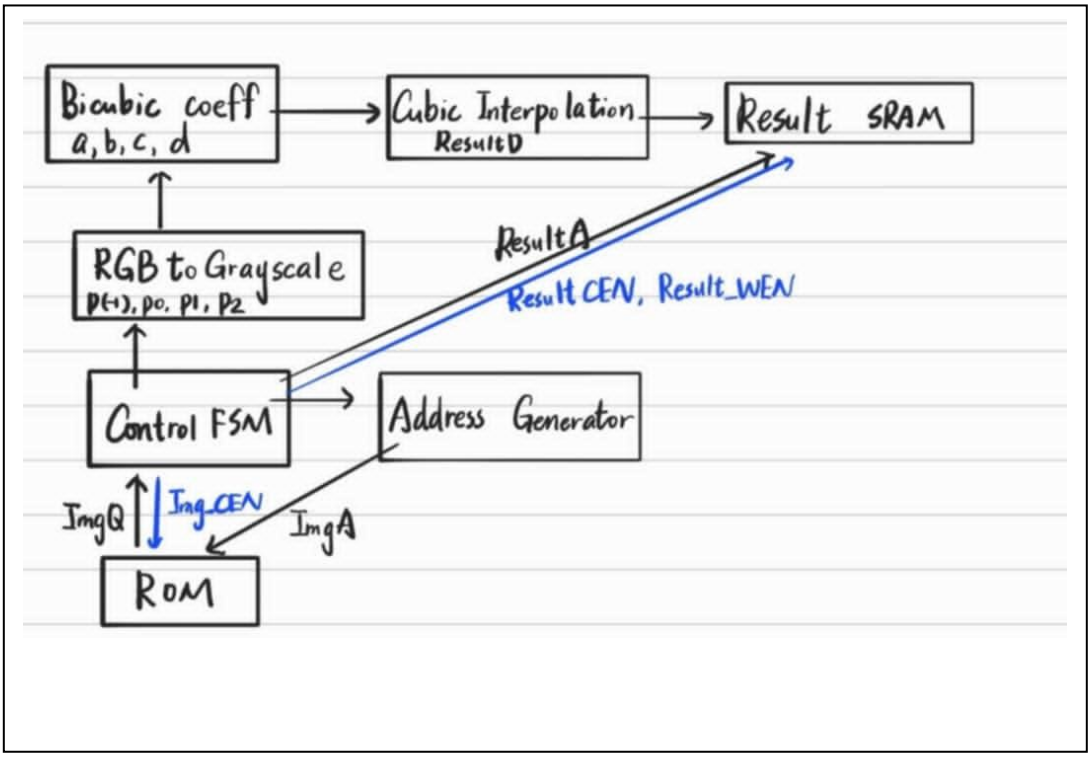
- Understanding the function:

  Once system is initialized, it

  a) Calculate the scaling factor based on original_w and scaled_w.
  b) For a scaled pixel in the scaled picture, find the corresponding original p(0) pixel position by using the calculated scaling factor.
  c) Read the p(-1), p(0), p(1), and p(2) original pixel values from the testbench. Remember to convert the RGB pixels to grayscale.
  d) Calculate the value of the scaled pixel with the bicubic formula.
  e) Write the calculated scaled pixel value into the testbench.
  f) Repeat steps b)–e) for each pixel in the scaled image.

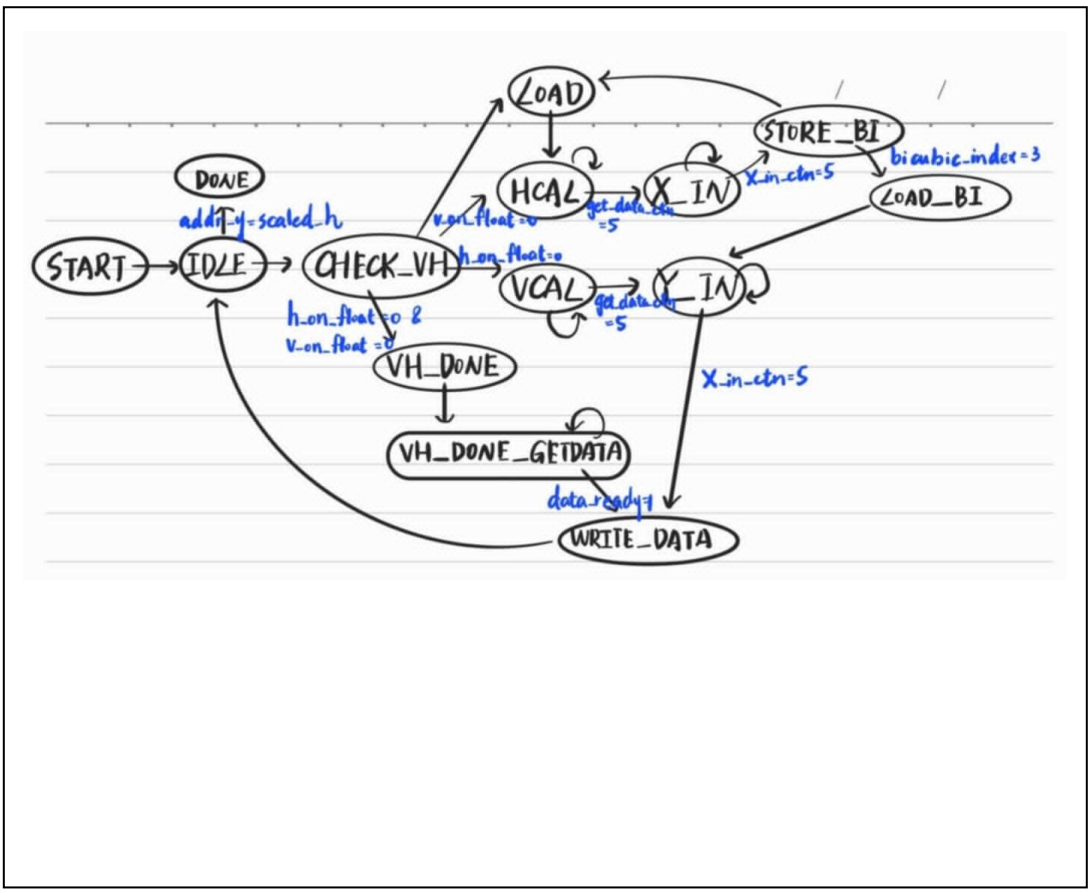
- Know the basic design rules
  - All operations are activated on the positive edge of the clock.
  - Control signals:
    - *Img_CEN*: Active low read enable signal for ImgROM
    - *Result_CEN*: Active low chip enable signal for ResultSRAM
    - *Result_WEN*: (Low write/high read) enable signal for ResultSRAM
    - *done*: 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.



Control FSM 為控制訊號，先將 addr_x, addr_傳入 Address Generator 得到 ImgA，再透過 ImgA, Img_CEN 取得 ImgQ 並進行 RGB to Gray 轉換得到 p(-1),p(0),p(1),p(2)，藉由組合電路取得 a,b,c,d 的值。進入 CUBIC INTERPOLATION 帶入 x 計算，透過 Control FSM 控制 ResultA(每次寫入後都加 1) 和 Result_WEN, Result_CEN 等控制訊號，寫入 SRAM。

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



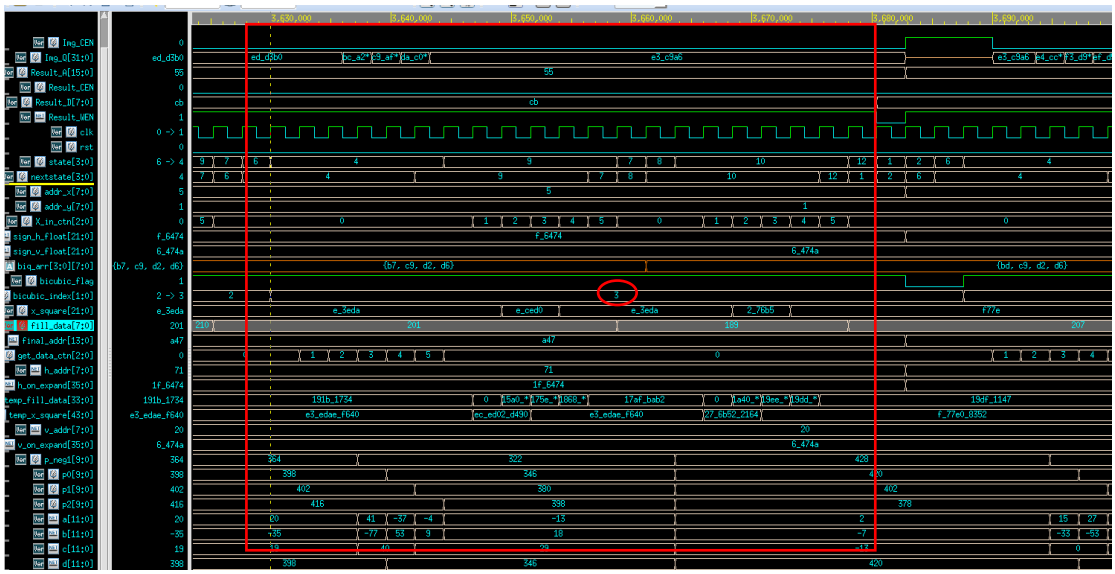
由 START 開始，IDLE 時會檢查 addr_y 是否等於(scaled_h)，相等代表已完成所有 pixel 處理進入 DONE。
CHECK_VH 檢查該 pixel 位置，若恰落在原圖點上，進入 VH_DONE，若恰落在原圖橫線上，代表只需計算水平內插，進入 HCAL，若恰落在原圖直線上，進入 VCAL，都不在上面則須計算 bicubic，進入 LOAD，LOAD 狀態會將 bicubic_flag 設為 1，以便在做完 HCAL 後檢查是否進入 WRITE_DATEA，LOAD 會進入 HCAL。
HCAL 狀態計算水平內插，此狀態會得到 p(-1),p(0),p(1),p(2)，並同時得到 a, b, c, d 值，再進入 X_IN。X_IN 會帶入 x 值，做完後透過 bicubic_flag 判斷是否直接進入 WRITE_DATA。如果 bicubic_flag 為 1 代表須將結果存入，進入 STORE_BI，若 bicubic_flag 為 0 則直接寫入 memory。

STORE_BI 會在做完四次後跳入 LOAD_BI，將剛剛儲存在陣列內結果賦值到 p(-1),p(0),p(1),p(2)，最後進入 Y_IN 將 y 值帶入計算。若還沒有做到第四次，則會跳回 LOAD 繼續做水平內插。

1. Complete the bicubic module, in the system.
2. Compile the verilog code to verify the operations of this module works properly.
3. Synthesize your *bicubic.v* with following constraint:
   - Clock period: no more than 10.0 ns.
   - Don't touch network: clk.
   - Wire load model: N16ADFP_StdCellss0p72vm40c.
   - Synthesized verilog file: *bicubic_syn.v*.
   - Timing constraint file: *bicubic_syn.sdf*.

4. Please **attach the resulting final bicubic_result_p6.bmp** image generated from the simulation. You may also attach a non-final bicubic_result.bmp together as well to highlight your progression to a perfect output (not compulsory).

5. Please **attach your waveforms** and **specify your operations** on the waveforms.

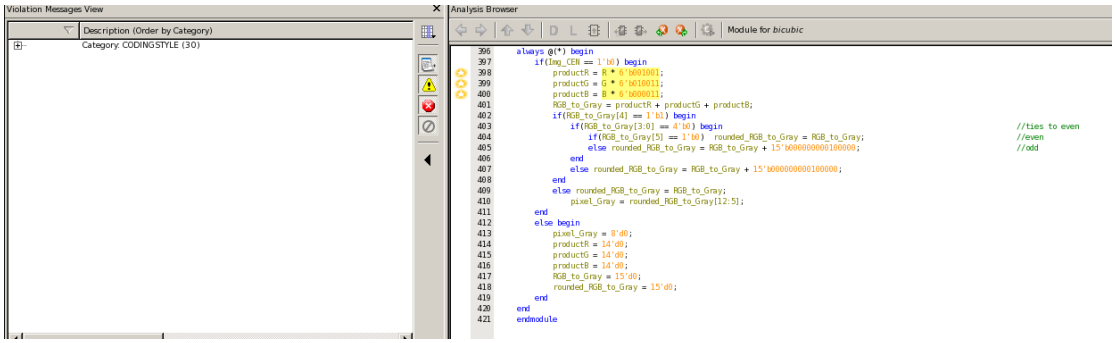


以測資 pic2 為例，add_x=3,add_y=1 做一次 bicubic 過程。

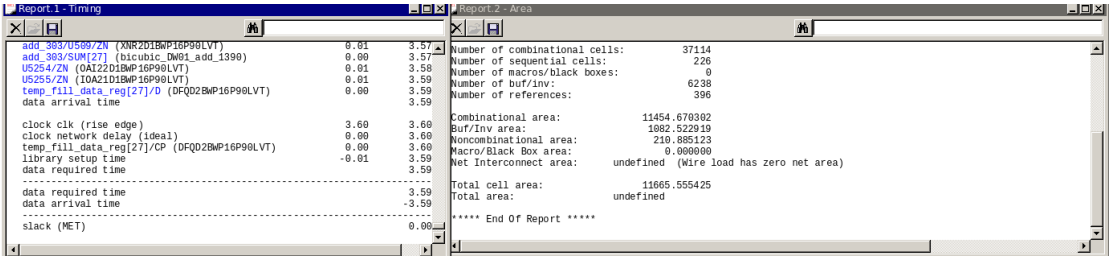
圖 1 因為 h_on_expand[19:0], v_on_expand[19:0]皆不為 0，代表需做 bicubic，因此先進入 LOADstate(6)後計算水平內插 HCALstate(4)，在 HCALstate(4)獲得 p_neg,p0,p1,p2 時同時藉由組合電路獲得 a,b,c,d。接著在 X_INstate(9)帶入 x 值，完成後因為 bicubic_flag 為 1，進入 STORE_BIstate(7)，在 STORE_BIstate 時因為 bicubic_index 不為 3，代表還未做到第四個，因此再次進入 LOADstate(6)。圖 2 差別在進入 STORE_BIstate(7)時，因為 bicubic_index=3，因此進入 LOADBIstate(8)，將存在 bicubic_arr 的結果分別賦值到 p_neg1,p0,p1,p2，最後進入到 Y_INstate(10)，帶入 y 值，最後在 WRITE_DATAstate 將結果存入 memory。

6. Show SuperLint coverage (including all files)

Coverage = 391/421 = 92.8%



7. Your clock period, total cell area, post simulation time, max timing report, simulation results from terminal with screenshot.




```
`timescale 1ns/1ps
`define CYCLE_TIME 2.4 // ns, modify according to your design
`define MAX_CYCLE 150000000    // adjust this if you need more cycles
```

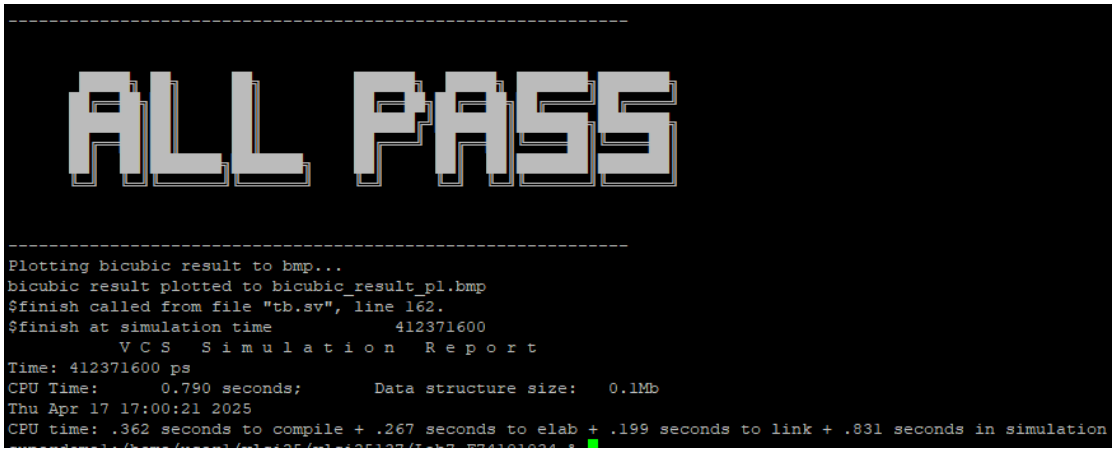
Clock period: 2.4 ns

Total cell area:11665.55

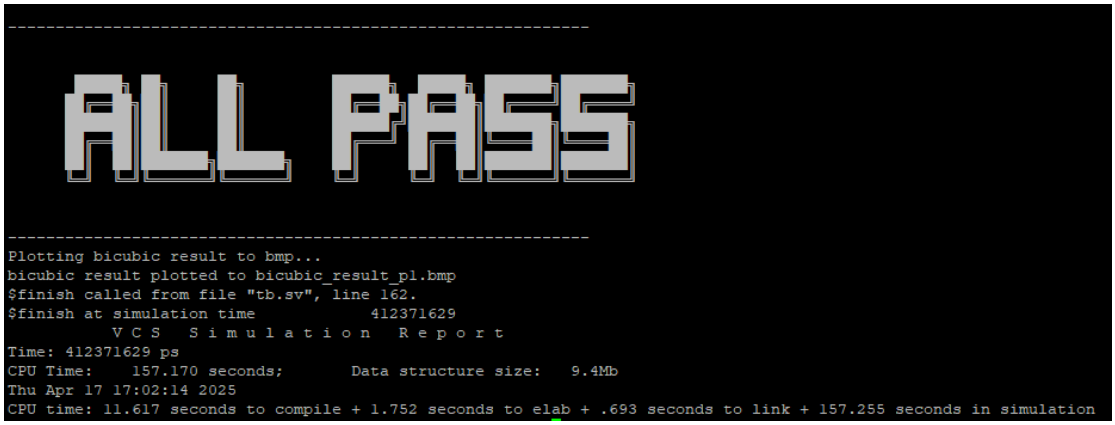
Post simulation time: 3961251629 ps

Max timing report: slack = 0

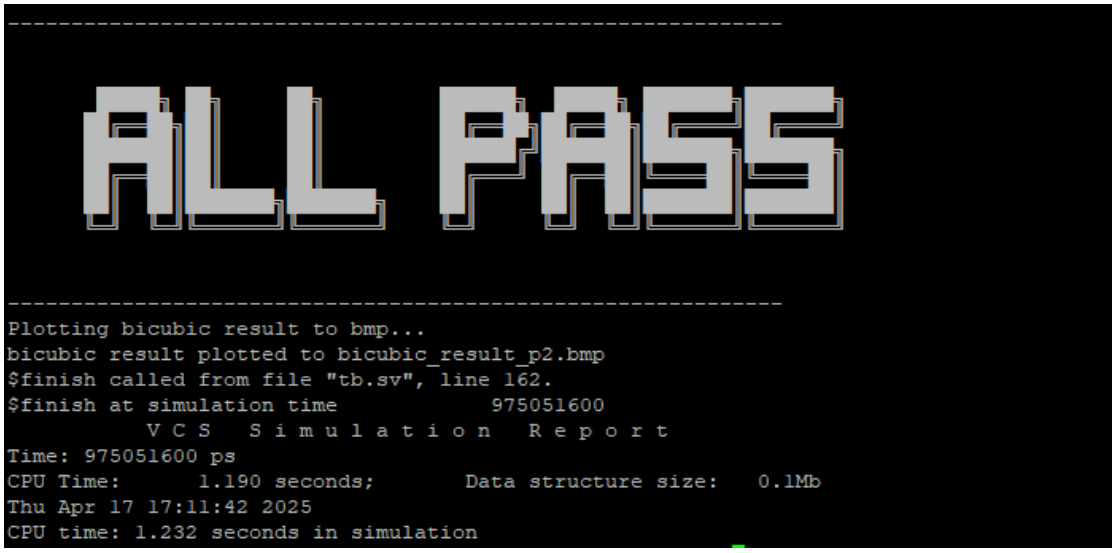
Pattern 1 RTL Terminal Result:



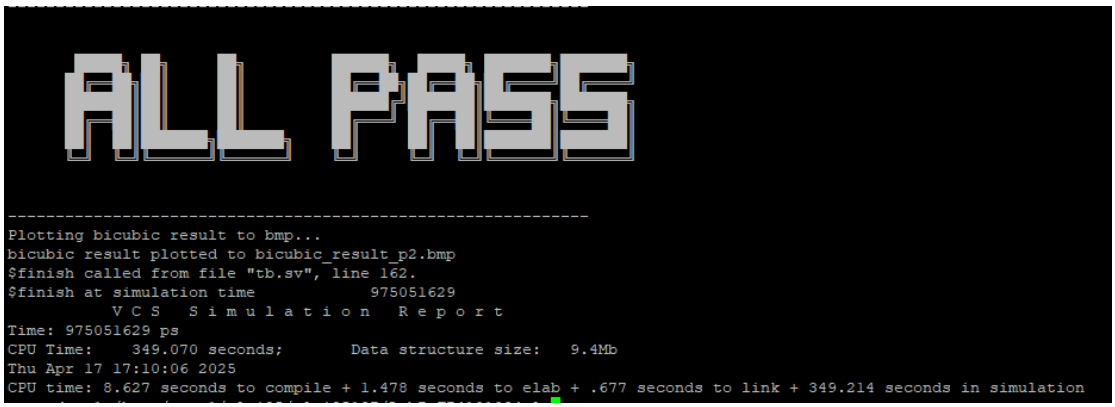
Pattern 1 SYN Terminal Result:



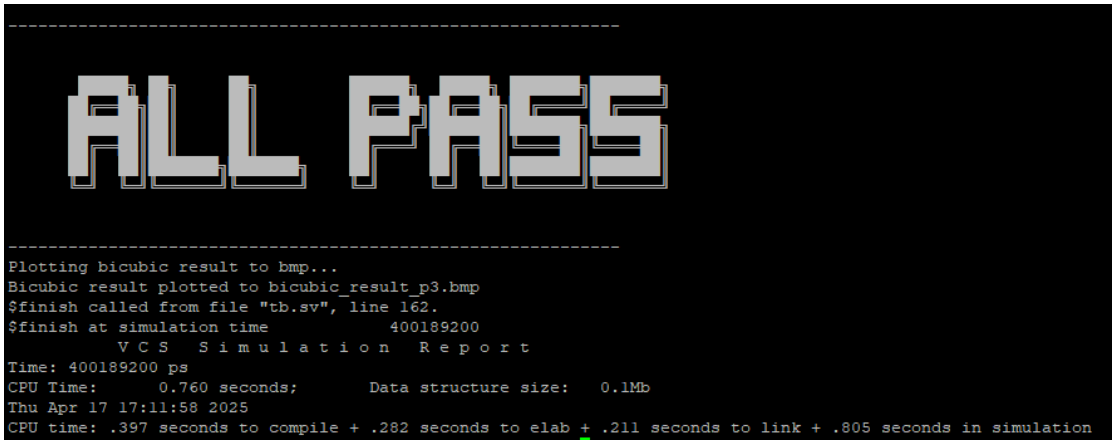
Pattern 2 RTL Terminal Result:



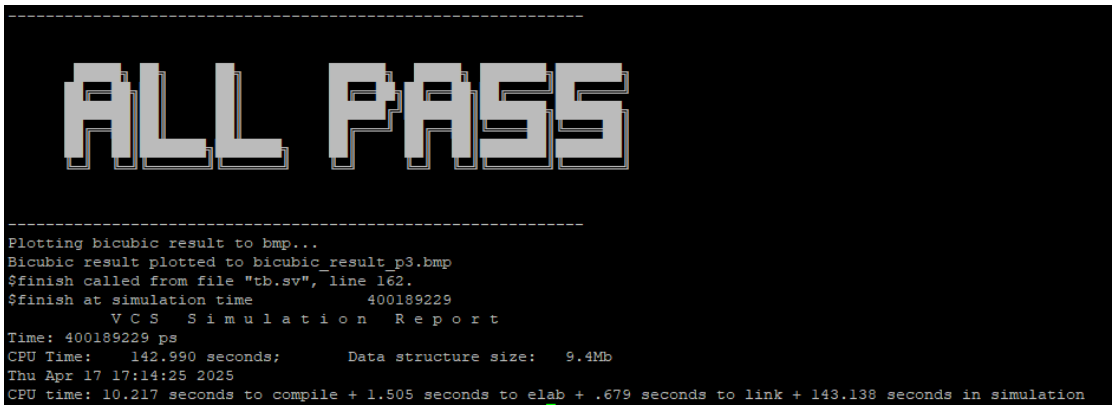
Pattern 2 SYN Terminal Result:



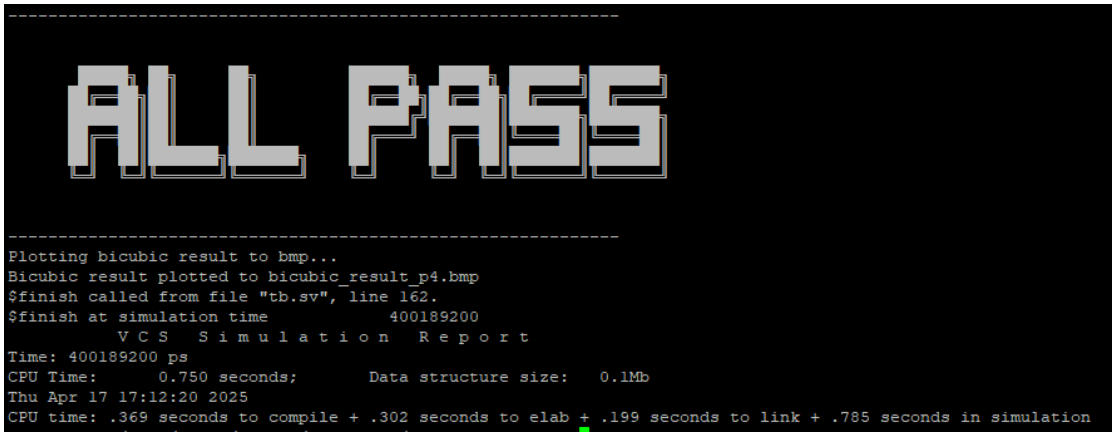
Pattern 3 RTL Terminal Result:



Pattern 3 SYN Terminal Result:

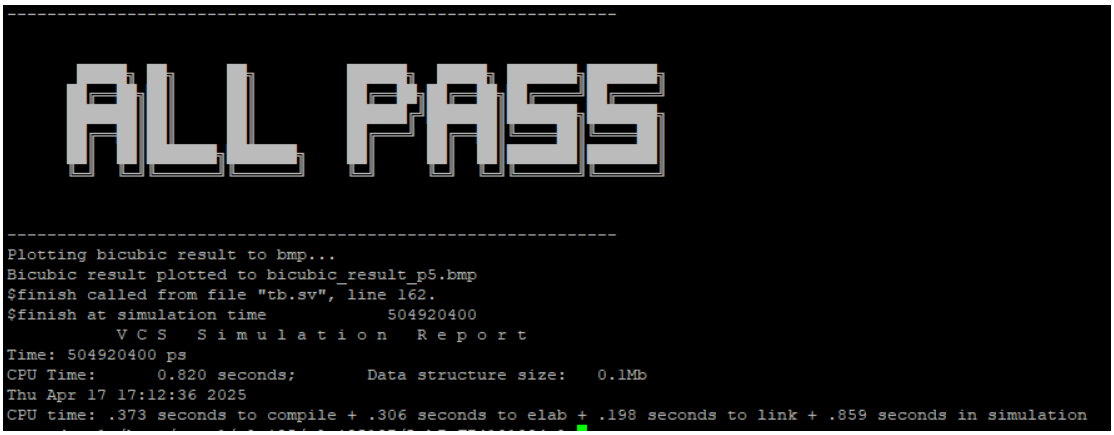


Pattern 4 RTL Terminal Result:

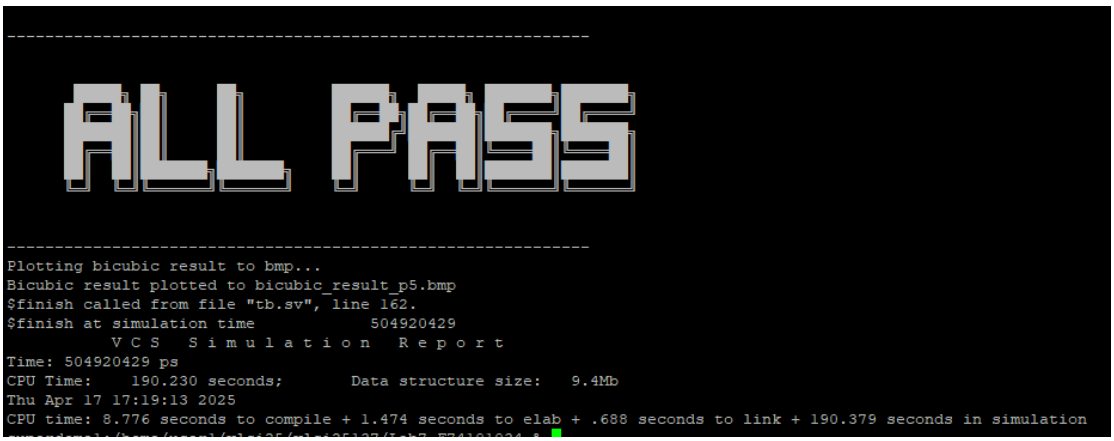


Pattern 4 SYN Terminal Result:

Pattern 5 RTL Terminal Result:



Pattern 5 SYN Terminal Result:



Pattern 6 RTL Terminal Result:

Pattern 6 SYN Terminal Result:

8. 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.

Pipelining、Resource Reuse

我將計算 ax^3+bx^2+cx+d，拆解成多個 stage，並且在每個 stage 做完乘法後進行省略，以縮小暫存器所需單位，只取到小數後 20 位(原本為 40 位)。例如第一個 stage 計算 x^2，並存入 44bit 變數，第二個 stage 對此變數進行精度省略，並存入 22bit 變數。第三個 stage 將剛剛獲得 22bit x_square 與 x 相乘(為了獲得 x^3)，並存入 stage1 的 44bit 變數， 達到 Resource Reuse。

以這此方法，我只需要兩個乘法器即可完成 ax^3+bx^2+cx+d 的計算。

9. Lessons learned from this lab

上次的作業跟這次最大的差別在需要可以接受不同大小的 scale 和起始點 x0, y0，以上次的寫法我只能處理固定大小的圖片，因此需要重新設計判斷 p(-1), p(0), p(1), p(2)位置和該點是否內插的發法。我透過先將處理的位置右移 20bit，在乘(origin-1)並除(scale-1)。並判斷後 20bit 是否等於 0 來看此點有沒有在線上，若等於 0 表示整除在線上，如果沒有就需進行 bicubic interpolation。我將相乘和相除寫在同一行是式子，但是忽略相乘時會有 overflow 的發生，因此在跑 img6 時會出現錯誤，檢查後擴大暫存器避免 overflow。

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

Situation	Command
RTL Simulation (without plotting image values)	make rtl%
RTL Simulation (plotting image values without color)	make rtl%_plot
RTL Simulation (plotting image values with color)	make rtl%_color
Open Design Vision GUI for synthesis (refer Lab 5 for synthesis)	dv &
Post-Synthesis Simulation (without plotting image values)	make syn%
Post-Synthesis Simulation (plotting image values without color)	make syn%_plot
Post-Synthesis Simulation (plotting image values with color)	make syn%_color
View Waveform	make nWave
Open Superlint	make superlint
Delete built files for simulation, synthesis or verification	make clean

Replace % with 1 ~ 6, to simulate pattern 1 ~ 6 respectively (e.g. make rtl1, make syn2_color)