

## Summary

Hardware		
	RTL All Pass( $\vee / X$ )	SYN All Pass( $\vee / X$ )
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
421	30	0
		coverage(%)
		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.

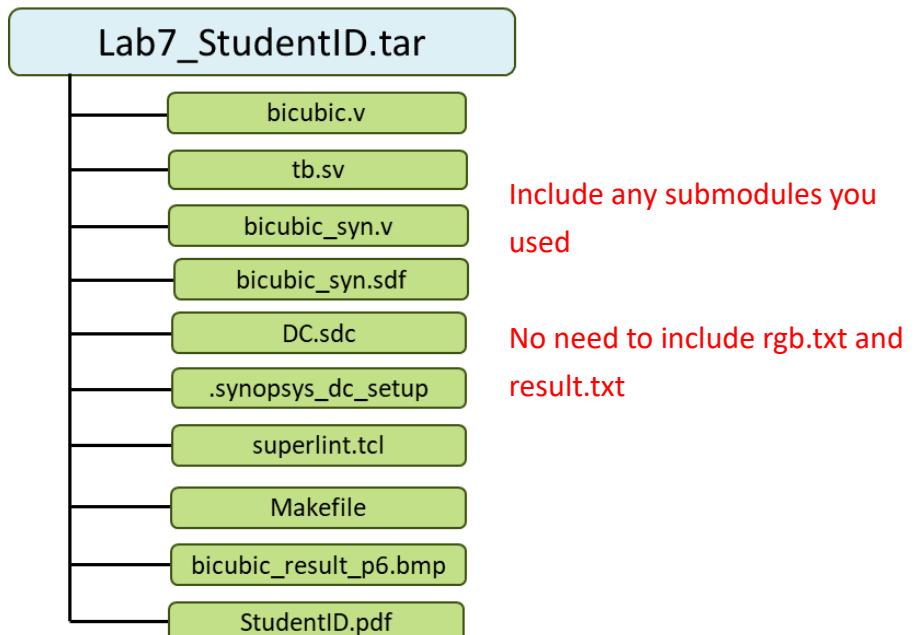


Fig.1 File hierarchy for Homework submission

---

*Lab 7: Bicubic Resize Engine*

---

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**.

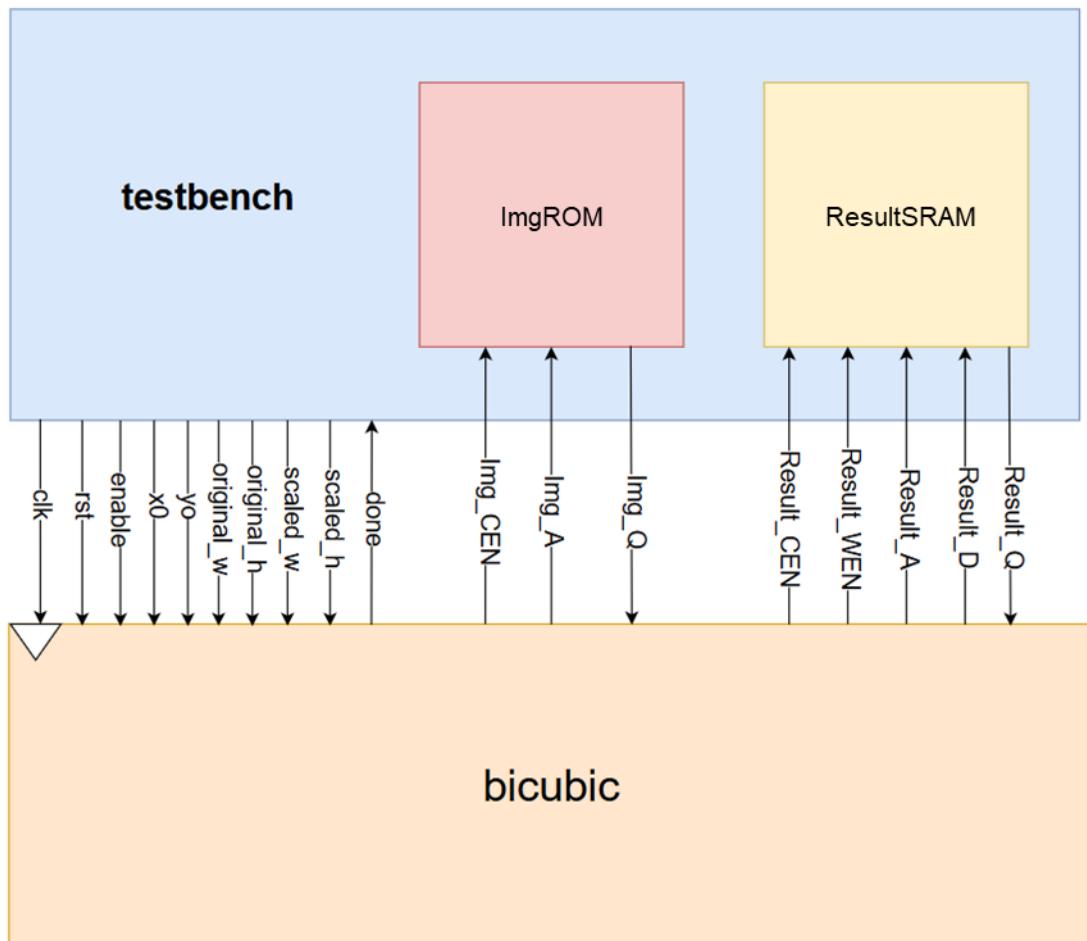


Fig2. The block diagram of bicubic circuit

➤ **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

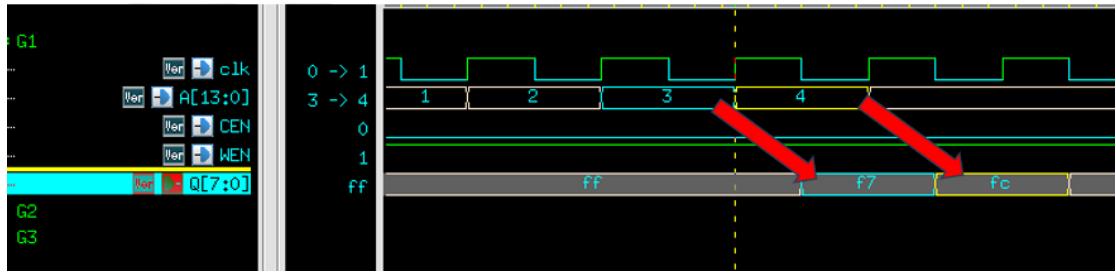


Fig3. example waveform for RAM read operation

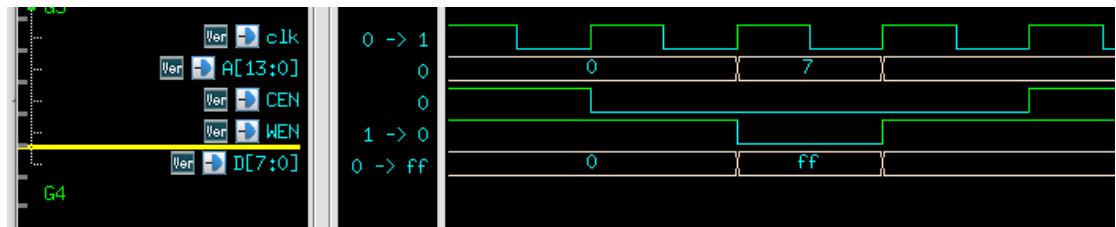
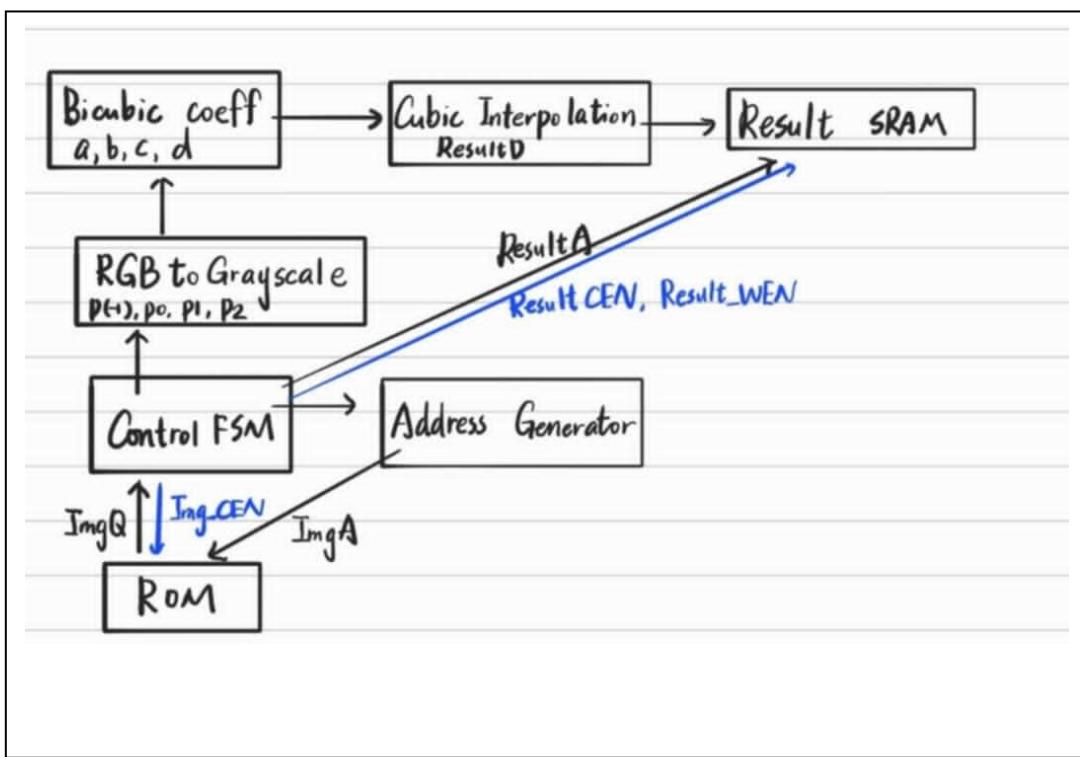


Fig4. example waveform for RAM write operation

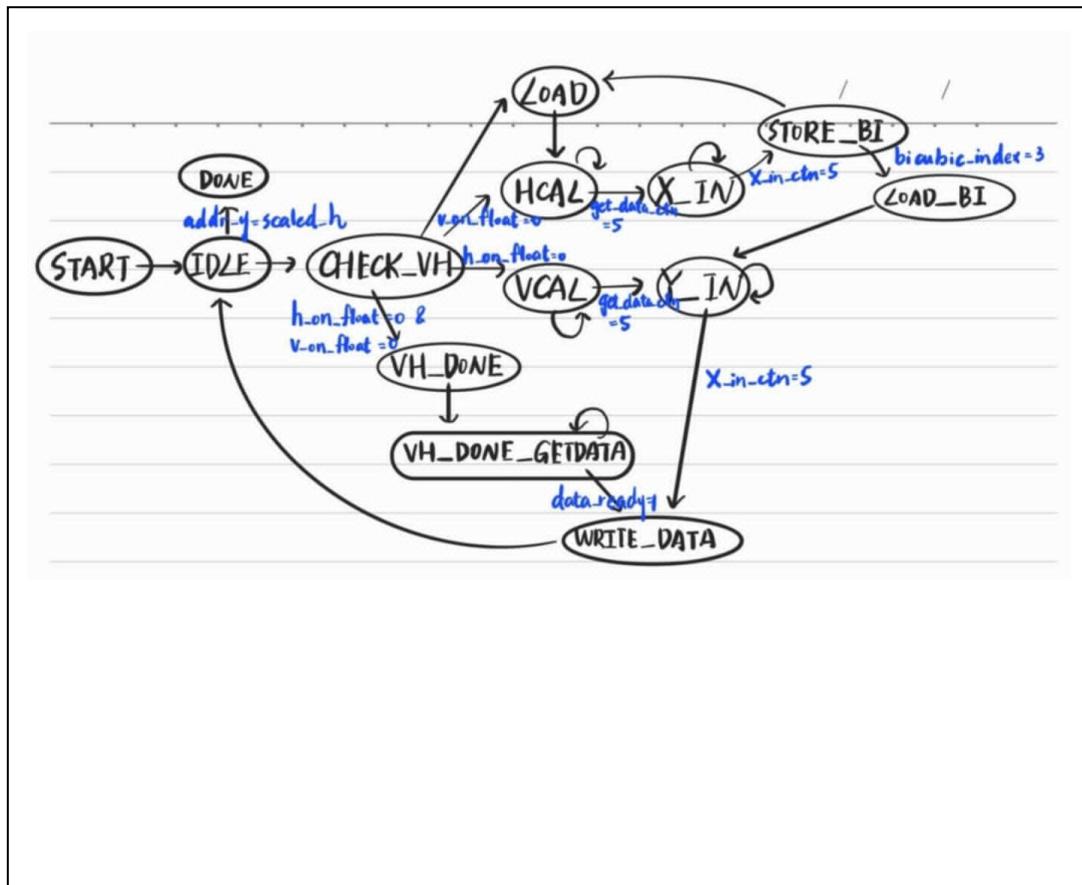
- Understanding the function:  
Once system is initialized, it
  - a) Calculate the scaling factor based on original\_w and scaled\_w.
  - b) For a scaled pixel in the scaled picture, find the corresponding original p(0) pixel position by using the calculated scaling factor.
  - c) Read the p(-1), p(0), 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\_y 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。

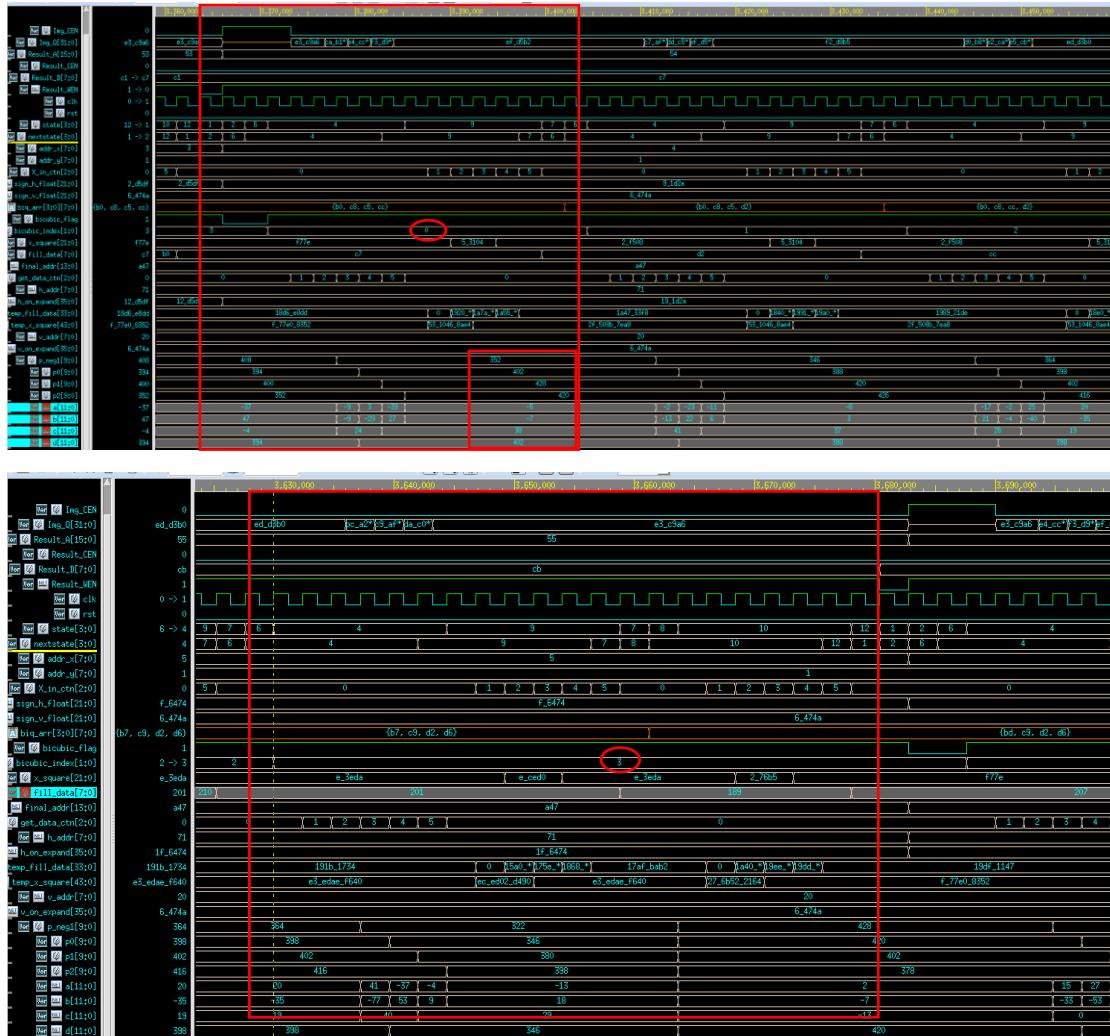
HCAL 狀態計算水平內插，此狀態會得到 p(-1),p(0),p(1),p(2)，並同時得到 a, b, c, d 值，再進入 X\_IN。X\_IN 會帶入 x 值，做完後透過 bicubic\_flag 判斷是否直接進入 WRITE\_DATEA。如果 bicubic\_flag 為 1 代表須將結果存入，進入 STORE\_BI，若 bicubic\_flag 為 0 則直接寫入 memory。

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%

```

396      always @(*) begin
397          if(RGB_CEll == 1'b0) begin
398              productB = 8 * 6'0001001;
399              productG = 8 * 6'0100011;
400              productR = 8 * 6'0000011;
401              RGB_to_Gray = productR + productG + productB;
402          if(RGB_to_Gray[4] == 1'b1) begin
403              if(RGB_to_Gray[3:0] == 4'd10) begin
404                  if(RGB_to_Gray[12:5] == 10'd0) rounded_RGB_to_Gray = RGB_to_Gray;
405                  else rounded_RGB_to_Gray = RGB_to_Gray + 15'd000000000100000;
406              end
407              else rounded_RGB_to_Gray = RGB_to_Gray + 15'd000000000100000;
408          end
409          else rounded_RGB_to_Gray = RGB_to_Gray;
410          pixel_Gray = rounded_RGB_to_Gray[12:5];
411      end
412      else begin
413          pixel_Gray = 8'd0;
414          productB = 14'd0;
415          products = 14'd0;
416          productG = 14'd0;
417          RGB_to_Gray = 15'd0;
418          rounded_RGB_to_Gray = 15'd0;
419      end
420  end
421 endmodule

```

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

add_303/UBMP9LVT	(UG254/ZN)	0.01	3.57
add_303/SWMP9LVT	(UG254/ZN)	0.00	3.59
UG254/ZN	(OAZ1D1BMP16P99LVT)	0.02	3.59
UG255/ZN	(OAZ1D1BMP16P99LVT)	0.01	3.59
temp_fill_data_req[27]:0	(DFQD2BMP16P90LVT)	0.00	3.59
data arrival time			
Clock CLK (rise edge)		3.60	3.60
clock network delay (ideal)		0.00	3.60
temp_fill_data_req[27]:CP	(DFQD2BMP16P90LVT)	0.00	3.60
library setup time		-0.01	3.59
data required time			3.59
data required time			3.59
data arrival time			-3.59
slack (MET)			0.00

Number of combinational cells:	37114		
Number of sequential cells:	226		
Number of macros/black boxes:	0		
Number of buf/inv:	6238		
Number of references:	396		
Combinational area:	11454.670302		
Buf/Inv area:	1892.522919		
Non combinational area:	210.000023		
Macro/Black Box area:	0.000000		
Net Interconnect area:	undefined (Wire load has zero net area)		
Total cell area:	11665.555425		
Total area:	undefined		
***** End Of Report *****			

```

`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

```

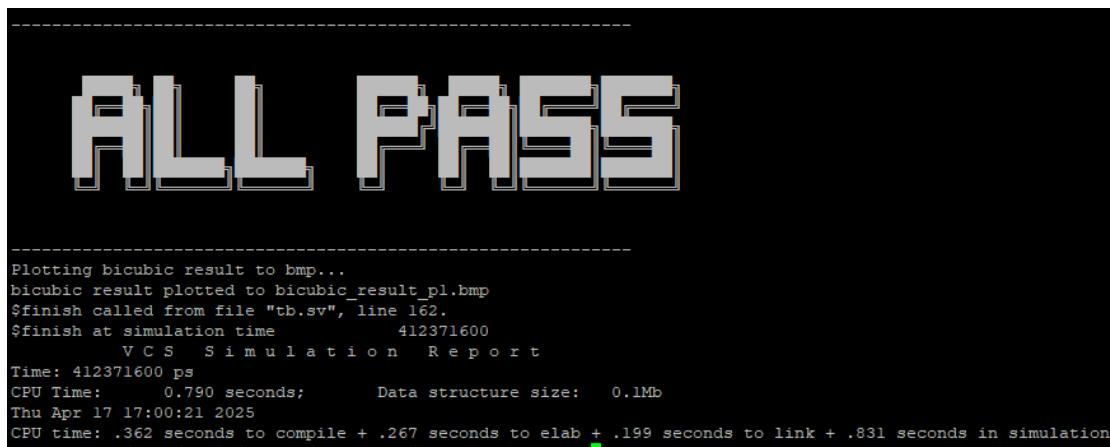
Clock period: 2.4 ns

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:

```
-----  
ALL PASS  
-----  
Plotting bicubic result to bmp...  
bicubic result plotted to bicubic_result_p1.bmp  
$finish called from file "tb.sv", line 162.  
$finish at simulation time 412371629  
V C S S i m u l a t i o n R e p o r t  
Time: 412371629 ps  
CPU Time: 157.170 seconds; Data structure size: 9.4Mb  
Thu Apr 17 17:02:14 2025  
CPU time: 11.617 seconds to compile + 1.752 seconds to elab + .693 seconds to link + 157.255 seconds in simulation
```

Pattern 2 RTL Terminal Result:

```
-----  
ALL PASS  
-----  
Plotting bicubic result to bmp...  
bicubic result plotted to bicubic_result_p2.bmp  
$finish called from file "tb.sv", line 162.  
$finish at simulation time 975051600  
V C S S i m u l a t i o n R e p o r t  
Time: 975051600 ps  
CPU Time: 1.190 seconds; Data structure size: 0.1Mb  
Thu Apr 17 17:11:42 2025  
CPU time: 1.232 seconds in simulation
```

Pattern 2 SYN Terminal Result:

```
-----  
ALL PASS  
-----  
Plotting bicubic result to bmp...  
bicubic result plotted to bicubic_result_p2.bmp  
$finish called from file "tb.sv", line 162.  
$finish at simulation time 975051629  
V C S S i m u l a t i o n R e p o r t  
Time: 975051629 ps  
CPU Time: 349.070 seconds; Data structure size: 9.4Mb  
Thu Apr 17 17:10:06 2025  
CPU time: 8.627 seconds to compile + 1.478 seconds to elab + .677 seconds to link + 349.214 seconds in simulation
```

Pattern 3 RTL Terminal Result:

```
ALL PASS

-----
Plotting bicubic result to bmp...
Bicubic result plotted to bicubic_result_p3.bmp
$finish called from file "tb.sv", line 162.
$finish at simulation time      400189200
          V C S   S i m u l a t i o n   R e p o r t
Time: 400189200 ps
CPU Time:    0.760 seconds;      Data structure size:   0.1Mb
Thu Apr 17 17:11:58 2025
CPU time: .397 seconds to compile + .282 seconds to elab + .211 seconds to link + .805 seconds in simulation
```

Pattern 3 SYN Terminal Result:

```
ALL PASS

-----
Plotting bicubic result to bmp...
Bicubic result plotted to bicubic_result_p3.bmp
$finish called from file "tb.sv", line 162.
$finish at simulation time      400189229
          V C S   S i m u l a t i o n   R e p o r t
Time: 400189229 ps
CPU Time:    142.990 seconds;      Data structure size:   9.4Mb
Thu Apr 17 17:14:25 2025
CPU time: 10.217 seconds to compile + 1.505 seconds to elab + .679 seconds to link + 143.138 seconds in simulation
```

Pattern 4 RTL Terminal Result:

```
ALL PASS

-----
Plotting bicubic result to bmp...
Bicubic result plotted to bicubic_result_p4.bmp
$finish called from file "tb.sv", line 162.
$finish at simulation time      400189200
          V C S   S i m u l a t i o n   R e p o r t
Time: 400189200 ps
CPU Time:    0.750 seconds;      Data structure size:   0.1Mb
Thu Apr 17 17:12:20 2025
CPU time: .369 seconds to compile + .302 seconds to elab + .199 seconds to link + .785 seconds in simulation
```

Pattern 4 SYN Terminal Result:

```
-----  
ALL PASS  
-----  
Plotting bicubic result to bmp...  
Bicubic result plotted to bicubic_result_p4.bmp  
$finish called from file "tb.sv", line 162.  
$finish at simulation time 400189229  
V C S S i m u l a t i o n R e p o r t  
Time: 400189229 ps  
CPU Time: 104.570 seconds; Data structure size: 9.4Mb  
Thu Apr 17 17:16:33 2025  
CPU time: 9.396 seconds to compile + 1.531 seconds to elab + .714 seconds to link + 104.663 seconds in simulation  
-----
```

Pattern 5 RTL Terminal Result:

```
-----  
ALL PASS  
-----  
Plotting bicubic result to bmp...  
Bicubic result plotted to bicubic_result_p5.bmp  
$finish called from file "tb.sv", line 162.  
$finish at simulation time 504920400  
V C S S i m u l a t i o n R e p o r t  
Time: 504920400 ps  
CPU Time: 0.820 seconds; Data structure size: 0.1Mb  
Thu Apr 17 17:12:36 2025  
CPU time: .373 seconds to compile + .306 seconds to elab + .198 seconds to link + .859 seconds in simulation  
-----
```

Pattern 5 SYN Terminal Result:

```
-----  
ALL PASS  
-----  
Plotting bicubic result to bmp...  
Bicubic result plotted to bicubic_result_p5.bmp  
$finish called from file "tb.sv", line 162.  
$finish at simulation time 504920429  
V C S S i m u l a t i o n R e p o r t  
Time: 504920429 ps  
CPU Time: 190.230 seconds; Data structure size: 9.4MB  
Thu Apr 17 17:19:13 2025  
CPU time: 8.776 seconds to compile + 1.474 seconds to elab + .688 seconds to link + 190.379 seconds in simulation  
-----
```

### 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  $x_{\text{square}}$  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  $x_0, y_0$ ，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)	<code>make rtl%</code>
RTL Simulation (plotting image values without color)	<code>make rtl%_plot</code>
RTL Simulation (plotting image values with color)	<code>make rtl%_color</code>
Open Design Vision GUI for synthesis (refer Lab 5 for synthesis)	<code>dv &amp;</code>
Post-Synthesis Simulation (without plotting image values)	<code>make syn%</code>
Post-Synthesis Simulation (plotting image values without color)	<code>make syn%_plot</code>
Post-Synthesis Simulation (plotting image values with color)	<code>make syn%_color</code>
View Waveform	<code>make nWave</code>
Open Superlint	<code>make superlint</code>
Delete built files for simulation, synthesis or verification	<code>make clean</code>

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