Summary

Hardware			
	RTL(∨/X)		Synthesis(∨/X)
Cubic Resize	V		V
Synthesis result			
Area	Simulation time (ps)		Area * Simulation time (ps)
1851	17694829		32753128479
Superlint(number of inline messages)			
Total lines	Warning	Error	coverage(%)
254	21	0	91.7%

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.

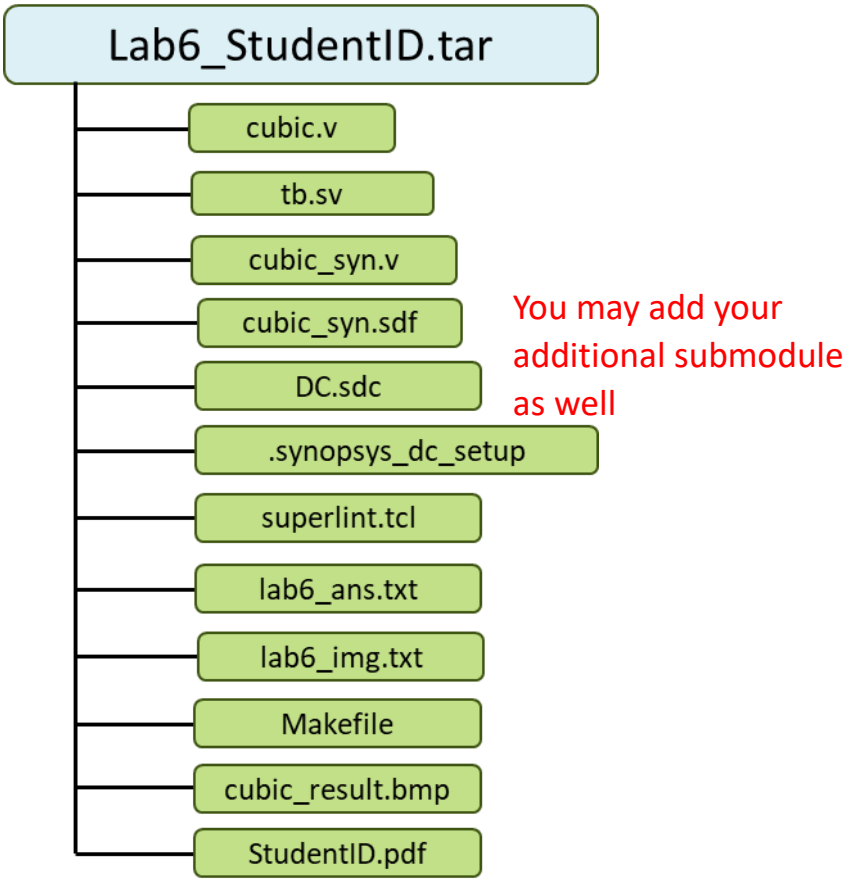


Fig.1 File hierarchy for Homework submission

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

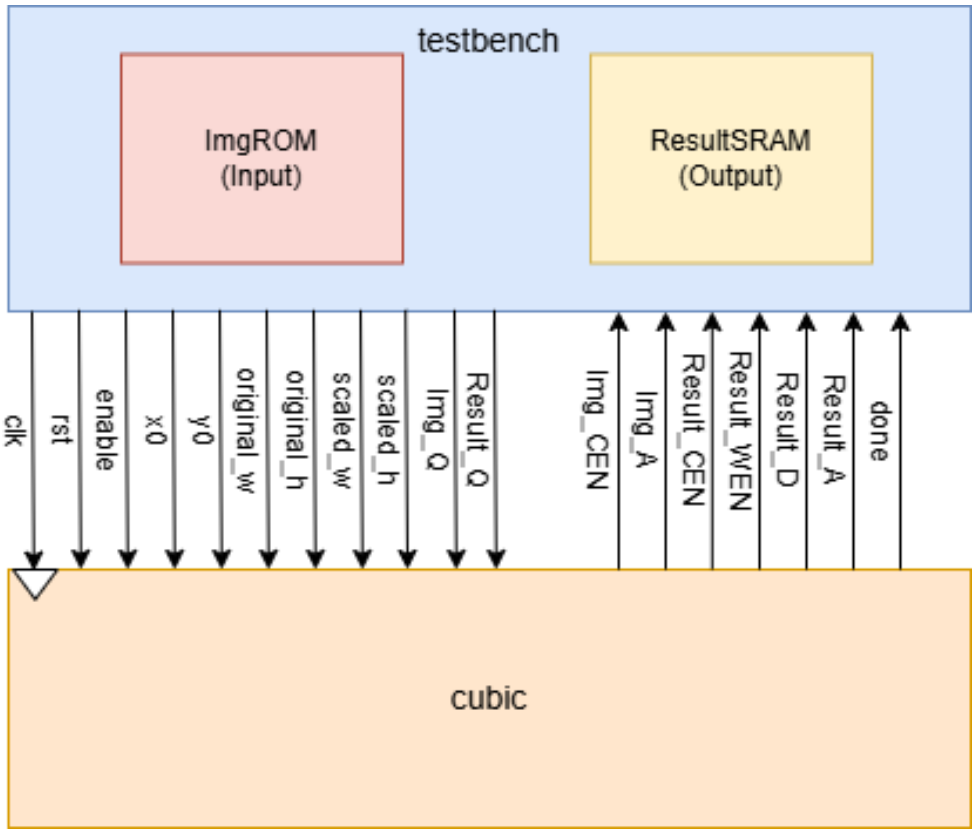


Fig2. The block diagram of cubic circuit

➢ Port list of Cubic 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	8	Read 8 bits unsigned data from ImgROM
Result_WEN	output	1	(Low write/high read) enable signal for ResultSRAM
Result_A	output	14	14 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

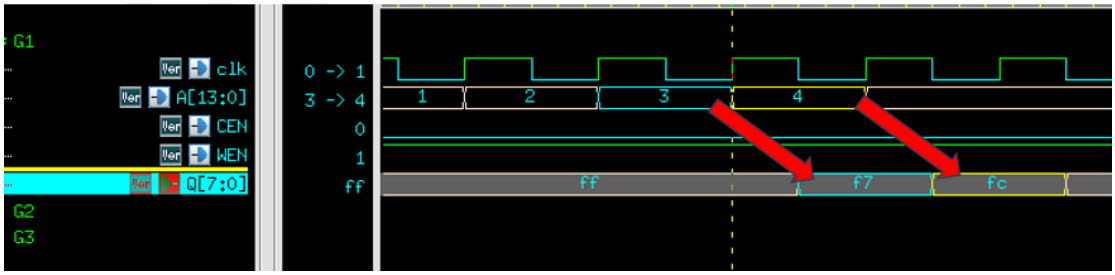


Fig3. example waveform for RAM read operation

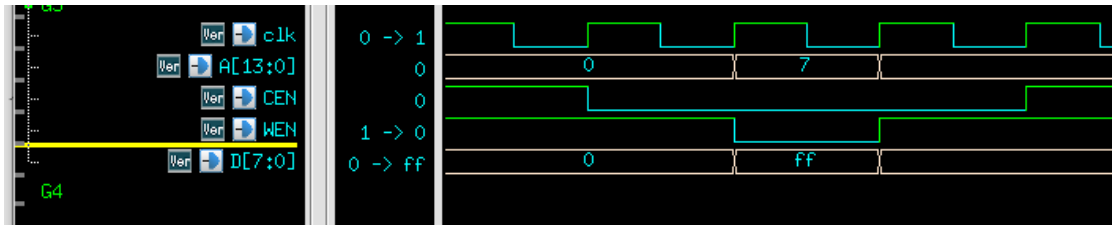


Fig4. example waveform for RAM write operation

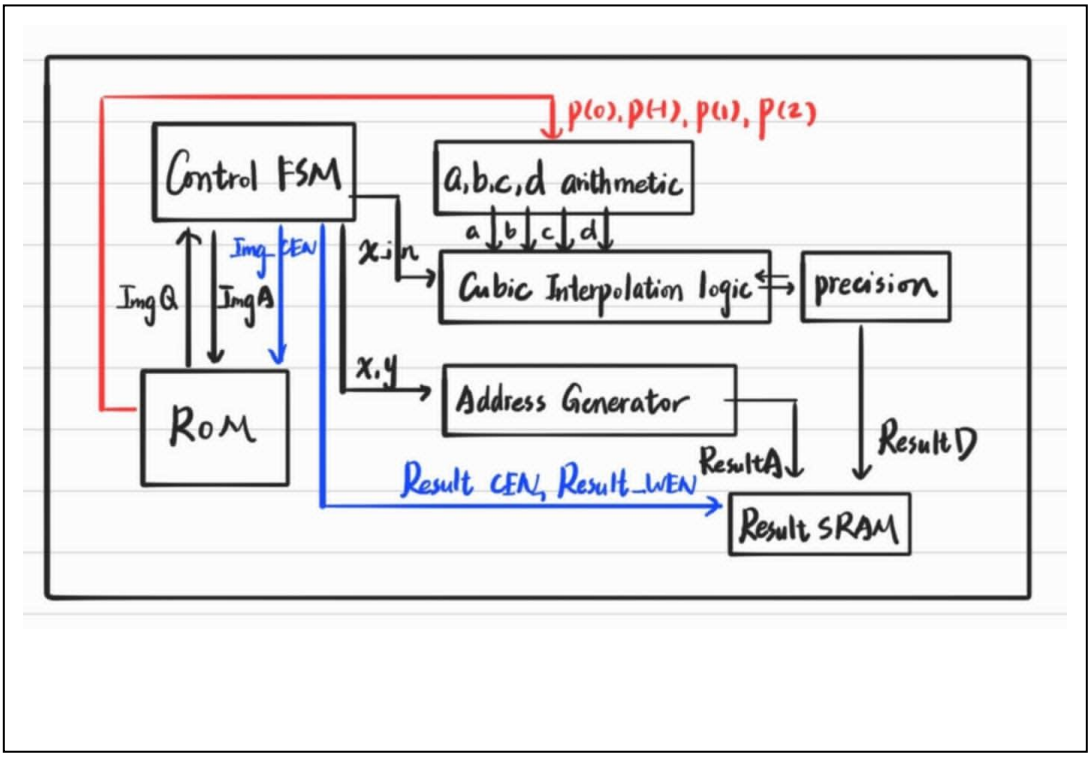
- Understanding the function:

  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.
  d) Calculate the value of the scaled pixel with the cubic 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

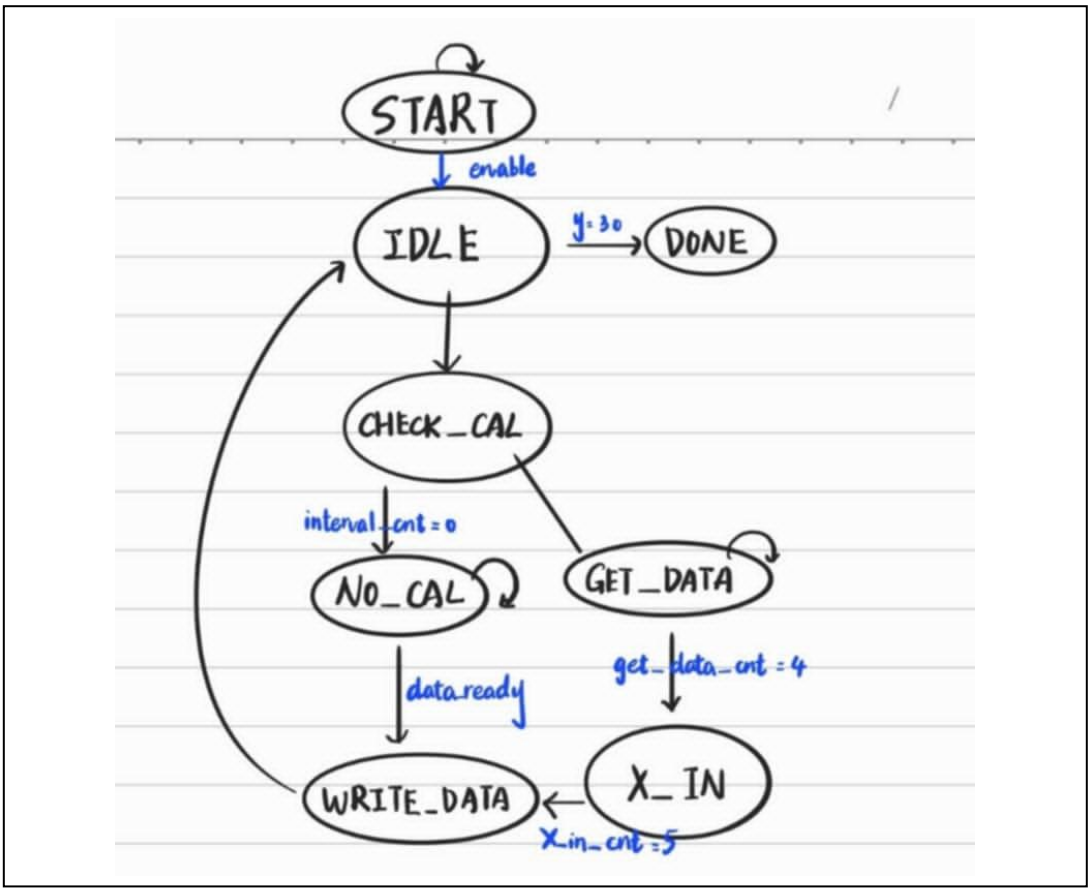
- Describe your design in detail. You can draw internal architecture or block diagram to help elaborate your design, if don't, plain text description is allowed.



Control FSM 為控制訊號，會透過 ImgA,Img_CEN 取得 p(0),p(-1),p(1),p(2)並透過 bitwise operation 取得 a,b,c,d 的值，在進入 CUBIC INTERPOLATION LOGIC 帶入 x_in 計算值，並在 PRESISION 取到小數後 16 位或四捨五入後，最後填入

透過 address generator 獲得的 address，並填入 SRAM。

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



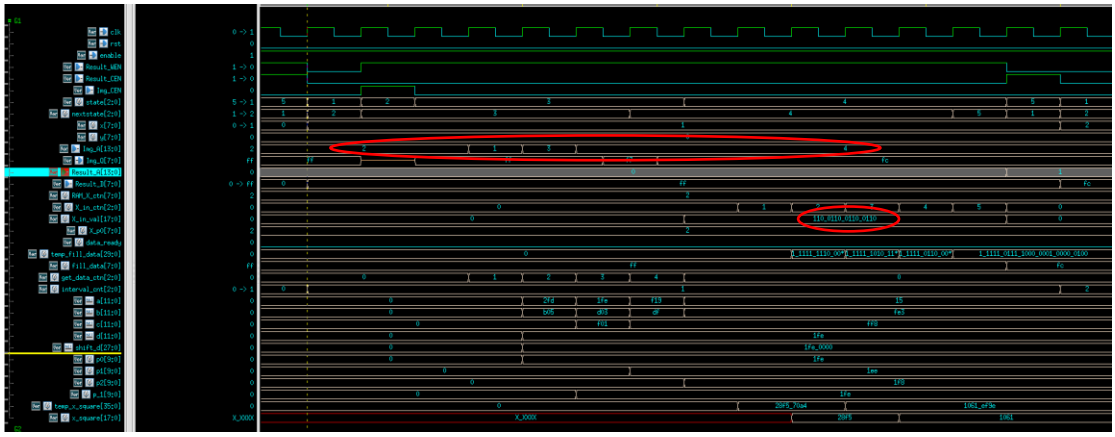
由 START 出發，在接收到 enable 訊號後轉移至 IDLE。判斷 y 是否等於 30，若等於 30 代表完成，會轉移至 DONE。若不等於則轉移至 CHECK_CAL。
CHECK_CAL 判斷此點是否可以直接對應(internal_cnt=0)，如果可以則轉移至 NO_CAL，後取得對應點值，無法對應則需轉移至 GET_DATA 分別取得 p(0),p(-1),p(1),p(2)，取得後轉移至 X_IN。
X_IN 則透過 internal_cnt 判斷應該 x 值應帶入 0.4/0.8/0.2/0.6 的哪一個。此 state 主要計算 ax^3+bx^2+cx+d，並將計算拆成六個 stage 用來縮小 area。
WRITE_DATA 即把資料寫入 memory。

1. Complete the cubic module, in the system.
2. Compile the verilog code to verify the operations of this module works properly.
3. Synthesize your *cubic.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: *cubic_syn.v*.
- Timing constraint file: *cubic_syn.sdf*.

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

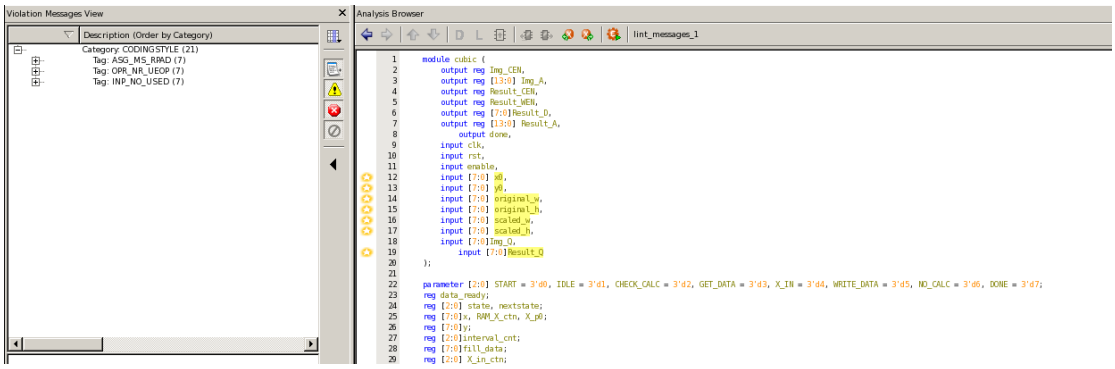
5. Please **attach your waveforms** and **specify your operations** on the waveforms.



State2 時透過 interval_cnt 決定五個區間中的第幾個，決定 ImgA 和 p(0)的 x 座標值，State3 會依序更新 ImgA 取得 p(-1), p(1), p(2)以計算 a, b, c, d。State4 會帶入 x 並將 ax^3 + bx^2 + cx + d 拆成多個 state 計算，最後 State5 會將結果寫回 ResultQ。以圖為例，此時 interval_cnt 為 1，代表是區間的第一個值，因此 p(0)的 x 值會等於 2，再依序取出 x=1, x=3, x=4 的值，分別代表 p(1),p(3),p(4)，多項式中的 x 會帶入 0.04，轉成 2 進位取小數加上 signbit 後 16 位是 0001100110011001l0，最後經運算和精度簡化後存入 fill_data，等一個 clk 後輸入 ResultD。

6. Show SuperLint coverage (including all files)

Coverage = 233/254 = 91.7%



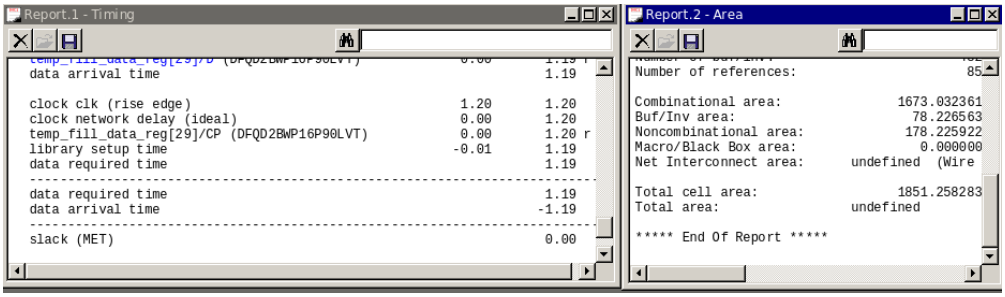
7. Your clock period, total cell area, post simulation time, max timing report with screenshot.

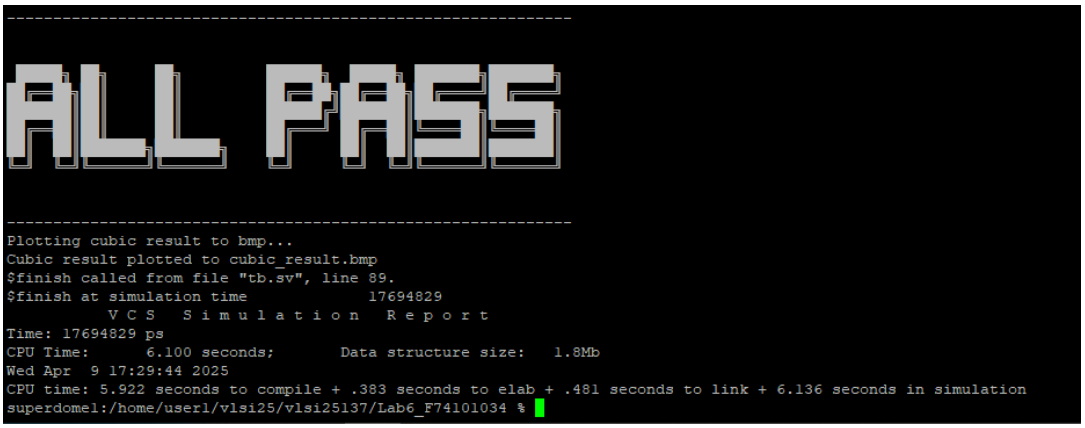
Clock period: 0.8 ns

Total cell area: 1851

Post simulation time: 17694829ps

Max timing report: slack=0






```
`timescale 1ns/1ps
`define CYCLE_TIME 0.8 // ns, modify according to your design
`define MAX_CYCLE 150000    // adjust this if you need more cycles
```

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.

- 用 SHIFT 操作取代乘法器
  我在計算 a, b, c, d 係數和 address 都利用左移、右移來取代乘法器。
- Pipelining、Resource Reuse
  我將計算 ax^3+bx^2+cx+d，拆解成多個 stage，並且在 stage 做完後進行省略，只取到小數後 16 位(原本為 32 位)。例如第一個 stage 計算 x^2，並存入 36bit 變數，第二個 stage 對此變數進行精度省略，並存入 18bit 變數。第三個 stage 將剛剛獲得 18bit x_square 與 x 相乘(為了獲得 x^3)，並存入 stage1 的 36bit 變數，達到 Resource Reuse。

9. Lessons learned from this lab
   這次的 lab 因為要進行有號小數的運算，因此讓我對精度省略、bit 大小設計和有號數運算更加熟悉。也因為此 lab 需要比較 area 和 execution time，所以在設計時也須多考慮如何減縮小面積，而不是用最簡單易懂的方式去撰寫電路。

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