2024 Spring IC Design

Final Project

Bilinear Resize Engine

1. 問題描述

請完成 ***Bilinear Resize Engine***(**後文以 Bilinear 表示**)功能的電路設計，本電路可將選取的 2 維矩陣資料放大到要求的尺寸大小，常應用在影像圖片的選取區域放大功能。

有關 Bilinear 詳細規格將描述於後。表一為本電路各輸入、輸出信號的功能說明。各組必須依照下一節所指定的設計規格進行設計及提供得的 testbench 完成設計驗證。

2. 設計規格

2.1 系統方塊圖

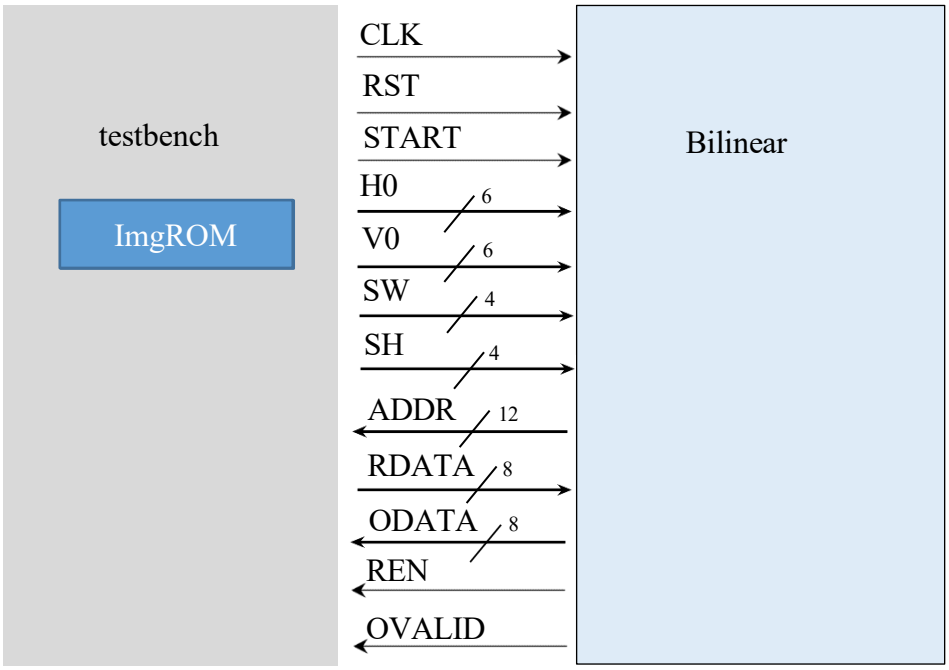


圖 1、系統方塊圖

2.2 輸入/輸出介面

Signal Name	I/O	Width	Simple Description
CLK	I	1	系統工作時脈，本系統為同步於時脈正緣之同步設計。
RST	I	1	系統重置訊號 reset signal (active high)。由 testbench 提供，拉高 2 cycle 後恢復為 low。
START	I	1	系統開始訊號，收到訊號後傳送H0,V0,SW,SH四個輸入，
H0	I	6	Bilinear 電路欲處理區域的左上角座標的 H 座標值，介於 0 到 63。
V0	I	6	Bilinear電路欲處理區域的左上角座標的 V 座標值，介於 0 到 63。
SW	I	4	Bilinear電路欲處理區域的水平方向寬度。該值加上 H0 不會超出原圖範圍。 H0+SW< 63
SH	I	4	Bilinear電路欲處理區域的垂直方向高度。該值加上 V0 不會超出原圖範圍。 V0+SH < 63
ADDR	O	12	儲存 Input map 之 ROM 的地址。輸出欲讀取之地址後，可在得到儲存於該位子的資料
R_DATA	I	8	儲存 Input map 之 ROM 的資料。根據輸出的地址，回傳儲存於該位子的資料
O_DATA	O	8	經過 Resize 計算後的資料。輸出時應按照順序從左至右，上至下輸出
REN	O	1	Active low，欲讀取 ROM 時，須將 REN 降至 low。testbench會在下個cycle將存於ADDR之資料傳到R_DATA
O_VALID	O	1	欲輸出計算完的資料時，須將 O_VALID 升至 high。Testbench 僅會檢查 O_VALID 為 1 時的 O_DATA

表 1 -輸入/輸出訊號

2.3 系統描述

本 Bilinear 電路設計的目的是對從原始圖像指定區域(左上角(H0,V0)，大小 SW x SH)，放大成17 x 17 大小圖像。(17 代表一行有 17 個資料點，頭尾都包含，因此間距為16)

Bilinear 電路運算以一次內插為基礎，先在原圖上找到內插端點，並利用一次內插的方式得到近似值，若運算點剛好與原圖端點重合，則直接使用原圖資料，不須經過內插。後續會有更詳細的說明。

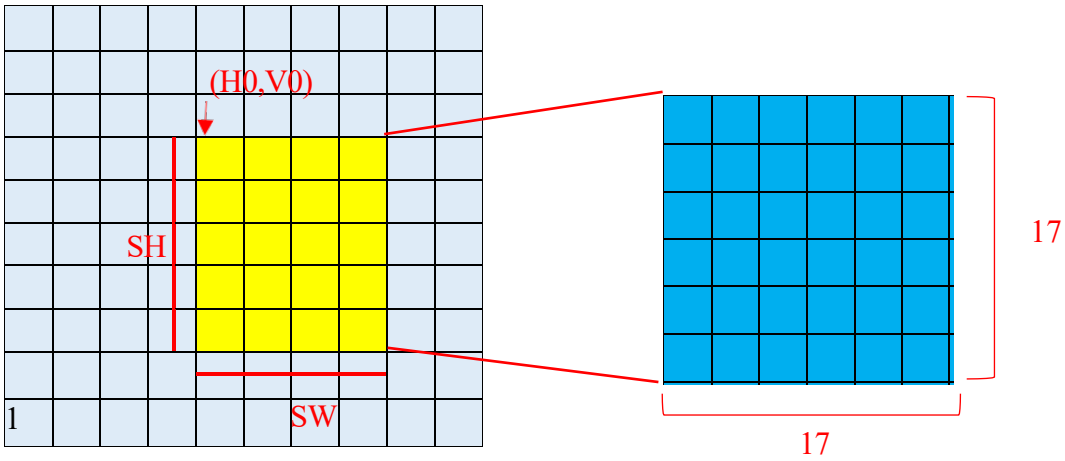


圖 2、Bilinear Resize Engine 工作原理示意圖

原始圖像資料為固定大小 64x64 之圖像，圖像資料存在 ImgROM 內。自系統 reset 後，從輸入 port 取得要進行放大區域的左上角座標(H0,V0)以及尺寸 SW x SH 等資訊，藉由ADDR, R_DATA, REN等 port 讀取ImgROM以取得資料。並藉由 Bilinear 電路將該範圍圖像放大到要求的 17 x 17 尺寸，並將運算結果內容從左到右、從上到下依序輸出。輸出時不一定要連續輸出，只需要確保在想要輸出時，O_VALID 為 high 即可。testbench 會直接讀取輸出內容進行評分。

2.3.1 圖像資料取得

Bilinear 電路所要處理的原始資料是一張 64x64 大小的圖像，每個像素資料為 8bits 正整數，並以一維的方式儲存在 ImgROM。儲存方式從第一條水平列(對應到 V 座標為 0)依序儲存到第 64水平列(對應到 V 座標為 63)

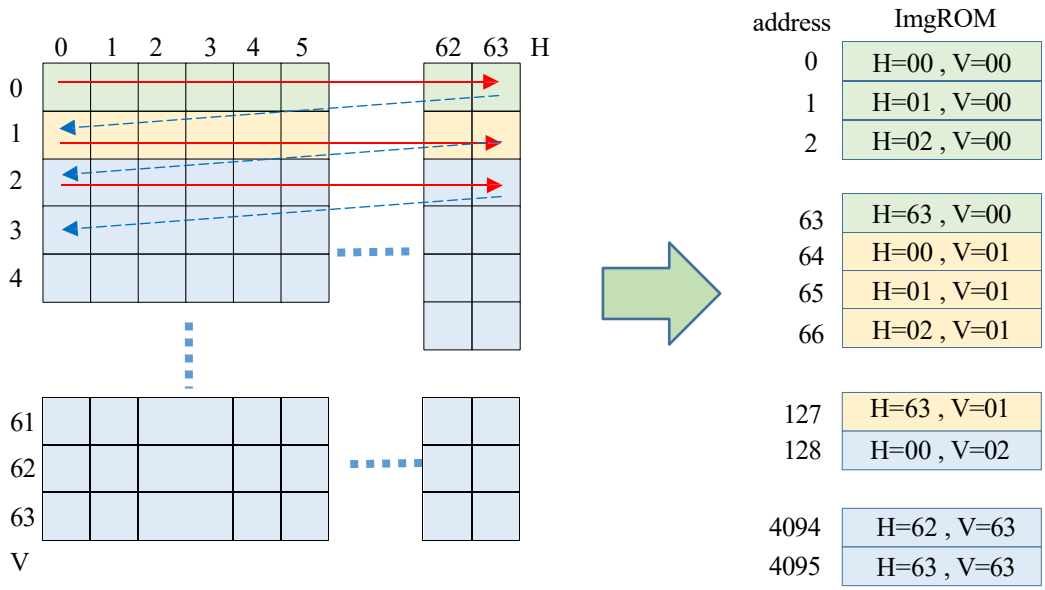


圖 3、二維原始資料儲存於 ImgROM 的方式

2.3.2 決定內插端點

考慮將在 1D(一維空間)上將一 1x7 圖像 P 放大在到 1x10 圖像 Q 的案例，目標是找出 Q 圖所有數值。想像將 Q 圖壓縮到與 P 圖相同長度時，兩圖最前端點重合，最後端點也重合，因此 Q(0)=P(0)，Q(9)=P(6)。

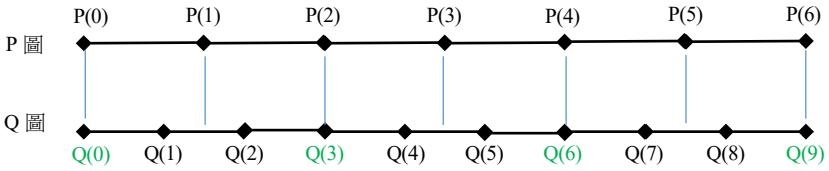


圖 4、尋找內插端點

部分中間點可能互相重合，可直接套用數值，在本案例，Q(3)=P(2)，Q(6)=P(4)。
其餘未重合點無法直接取得數值，必須從 P 圖尋找最接近端點，再用內插方法找出該點數值。在本案例，Q(1)的內插端點為 P(0)與 P(1)，Q(2)的內插端點為 P(1)與 P(2) ，Q(4)的內插端點為 P(2)與 P(3)，其餘類推。

尋找端點的方法之一：

$$Q * \frac{7-1}{10-1} = P$$

EX: $1 * \frac{6}{9} = \frac{2}{3}$, Q(1) 端點為 P(0), P(1)

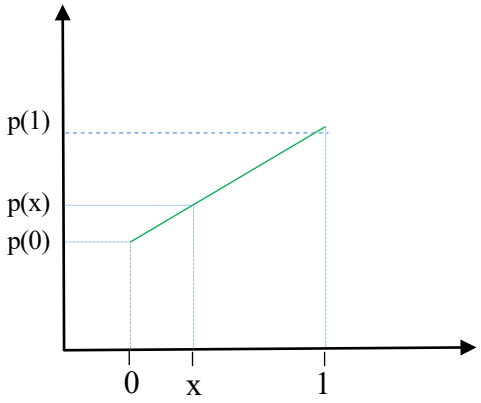
EX: $2 * \frac{6}{9} = 1\frac{1}{3}$, Q(2) 端點為 P(1), P(2)

2.3.3 一維內插

首先介紹在一維做 linear interpolation ，以方便後續延伸到二維 Bilinear interpolation 的說明。

2.3.3.1 Linear interpolation (線性內插)

已知 p(0)及 p(1)兩點數值，求兩點之間的點 p(x)， 0≤ x ≤1，p(x)線性內插公式為



$$p(x) = p(0) + \frac{x-0}{1-0}(p(1) - p(0))$$

$$= p(0) + \quad x(p(1) - p(0))$$

由於 p(0)和 p(1)水平方向相差 1，

x 剛好代表在 0 和 1 間的位置比例

圖 5、線性內插

以圖 4 為例，Q(2)落在 P(1)與 P(2)間 1/3 的位置，可得$Q(2) = P(1) + 1/3(P(2) - P(1))$

2.3.4 二維 Bilinear interpolation (雙一次內插)

一維 linear interpolation 是用 2 個點的數值求得內插數值，二維 Bilinear interpolation 則是用 4 個點求得圖像內插數值。以下圖為例，要取得綠點的內插值，需要周圍 abcd 四點數值；做法是將直行(ab, cd)各自算出一維內插結果，再使用這 2 個結果算出橫排(綠線)的一維內插。

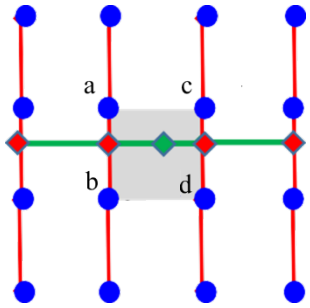
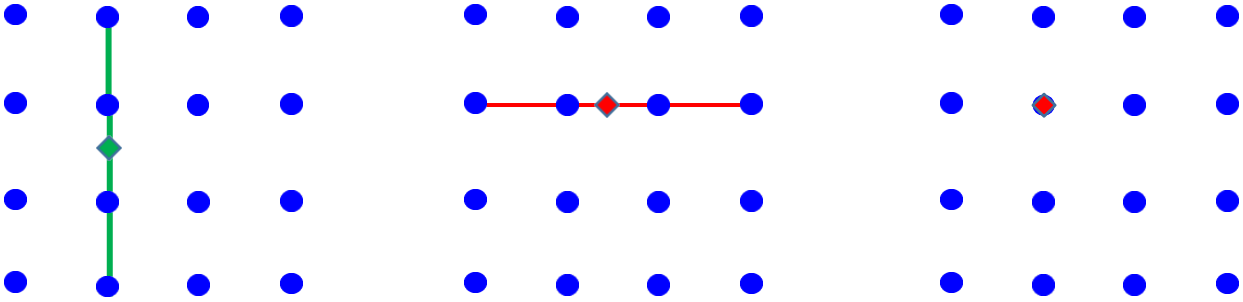


圖 6、二維雙一次內插

若直行或橫排剛好和原圖重合，則直接使用原圖數值。



垂直 V 方向和原圖重合　　水平 H 方向和原圖重合　　水平垂直方向皆重合

圖 7、水平方向或垂直方向重合

✧注意事項✧

1. Linear 每次運算結果請無條件捨去小數部分 (包含 2 個垂直內插以及 1 個水平內插)。
2. Linear 內插結果可能會超出 8bit 值域範圍，若內插結果大於 255，請將結果設為 255；若小於 0，請將結果設為 0 (包含 2 個垂直內插以及 1 個水平內插)。
3. 若先做水平內插再做垂直內插也可以得到結果，但因取概數原因結果可能會略為不同，因此請注意在計算二維內插時，務必先做垂直內插再做水平內插。

2.3.5 結果輸出

Bilinear 處理完的結果是一張 17 x 17 大小的圖像，每個像素資料為 8bits 正整數，結果需依序輸出到 testbench。由第一條水平列開始 (對應到 V 座標為 0)，直到最後一條水平列(對應到 V 座標為 16) 。在輸出時不一定要連續輸出 289 個 cycle，中間可以中斷不輸出，testbench 只會在 O_VALID 為 1 時判斷資料正確性，因此 O_VALID 為 0 時O_DATA可不必處理，只需要確保在O_VALID 為 1 輸出的資料有按照順序及可。

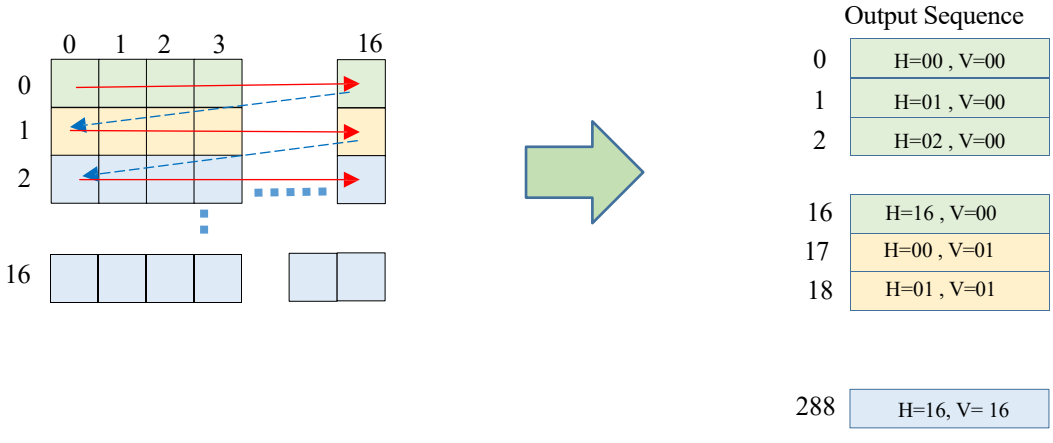


圖 8、影像輸出的方式

2.4 Bilinear 電路時序規格

輸出對應的時序波形圖如圖 所示，並請對照下面說明閱讀。

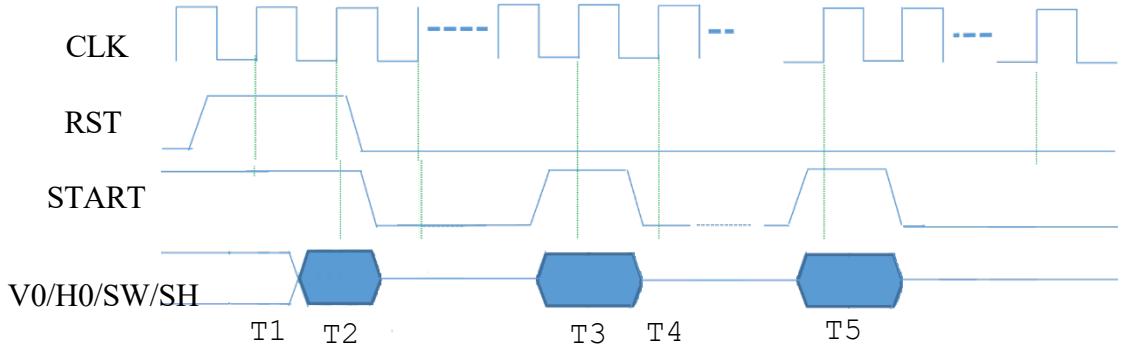


圖 9、Bilinear 電路時序圖

1. T1 時間點，進行起 2 週期 RST，並將 START訊號升至 high。
2. T2 時間點，開始輸入 V0、H0、SW、SH等資料，並持續一個
3. T2~ T3，藉由 ADDR, REN 讀取 ImgROM 內的資料，並將運算完的資料藉由 O_DATA, O_VALID 傳出。
4. T3-1 時間點，Bilinear 完成所有運算，並輸出第 289 筆資料後，testbench 完成比對所有輸出資料。
5. T3 時間點，START 訊號再次升至 high後，送入第二個 pattern。
6. T4 時間點，START 訊號再次降至 low
7. T5-1 時間點，Bilinear 完成第二個 pattern。
8. T5 時間點， START 訊號再次升至 high後，送入第三個 pattern。
9. 以此類推直到最後一個 pattern 結束

testbench 預設 pattern simulation 最多耗費 100000 週期，會在 100000 週期處發出$finish強制停止 simulation 。若有需要更多的周期進行 simulation ，請自行修改 tb.v 的 MAX_CYCLE 值。

2.4.1 ROM 之時序圖

ImgROM (Address 12bit , Data 8bits)，底下提供記憶體讀取時序圖。由於 Reset 期間，記憶體的輸入訊號可能尚未初始化，模擬時可能會出現 timing violation，因此在 Reset 期間出現的 timing violation 可忽略不管。

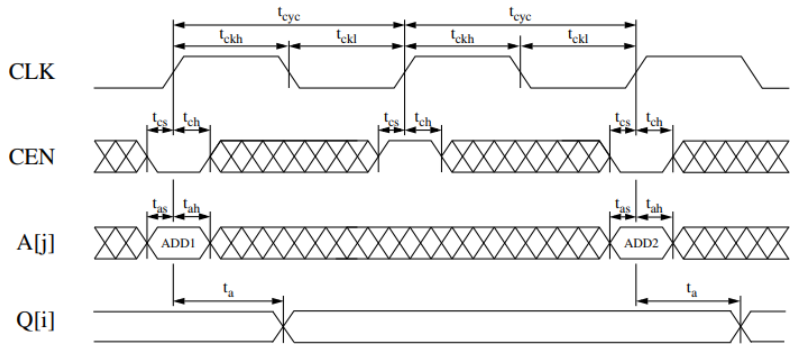


圖 10、Synchronous ROM single read-cycle timing

3. 檔案說明

資料夾	檔案	說明
00_TESTBED	tb.v	測試樣本檔，可以根據需求調整 clock cycle。
	*.dat	模擬時，會用到的測資。
01_RTL	01_run.sh	用於執行 RTL 模擬的腳本，請勿更改。
	rtl_01.f	RTL 模擬會使用的各種檔案路徑，若有新增.v 檔的話，必須將路徑加入此檔案。
	interpolation.v	本專題的設計檔，請勿更改輸入/出宣告，同學請於此檔案內作設計。
02_SYN	02_run.sh	用於執行 SYN 模擬的腳本，請勿更改。
	syn_02.f	SYN 模擬會使用的各種檔案路徑，若有新增.v 檔的話，必須將路徑加入此檔案。
	synthesis.tcl	合成用的 design constraint cycle 可依自己設定來更動，但其餘部分請勿更改。
	tsmc13.enc	解密方式同作業三，Gate level simulation 所使用的 process file(已加密)
	synopsys_dc.enc	解密方式同作業三，design vision 所需的設定檔(已加密)，請置於 02_SYN 資料夾，並在此資料夾開啟 dv 。注意此為隱藏檔
03_APR	03_run.sh	用於執行 APR 模擬的腳本，請勿更改。
	apr_03.f	APR 模擬會使用的各種檔案路徑，若有新增.v 檔的話，必須將路徑加入此檔案。
	library/*	APR 用的檔案，請勿更改。
	layout/*	APR 用的檔案，請勿更改(interpolation_apr.sdc 中的 cycle 除外) 。
	syn/tsmc13_neg.enc	解密方式同作業四，Transistor level simulation 所使用的 process file(已加密)
.	design.spec	填寫各個階段的資訊，會依據填寫 cycle time 來執行 testbench。單位請和檔案一致。

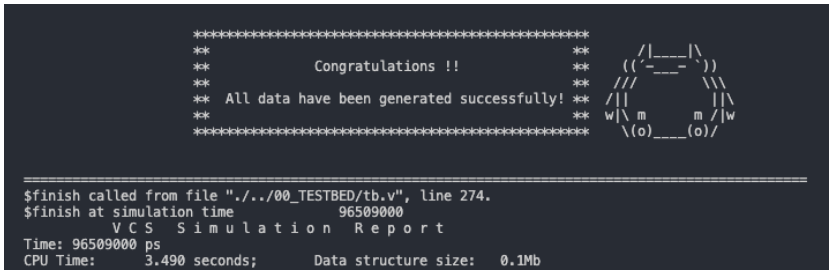
	report.docx	report template，記錄過程 完成後請轉成 pdf 並將檔名取為 bxx901xxx_report.pdf

4. 模擬指令

- **RTL Simulation:**
  $ cd 01_RTL
  $ bash 01_run.sh # include 3 different patterns
- **Gate level Simulation:**
  $ cd 02_SYN
  $ bash 02_run.sh # include 3 different patterns
- **Transistor Level Simulation:**
  $ cd 03_APR
  $ bash 03_run.sh # include 3 different patterns
-

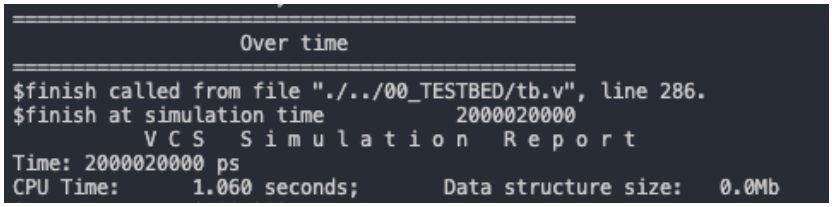
5. 模擬結果

- **模擬成功**



- **如果有錯誤出現**

- **超出時間**



6. 評分標準

Baseline (60%) + Performance (30%) + Report (10%)

共有三組公開測資+一組隱藏測資

a. 通過 RTL simulation. (30%)

b. 通過 gate level simulation (with no latch) (20%)

c. 通過 post layout simulation (10%)

- APR DRC/LVS no errors
- Power ring: width = 2, ring number ≥ 2
- Power stripe: width = 1, stripe number ≥ 1

d. Report (10%)

如果沒有通過 c (post layout simulation)，則 report 的分數，會根據最後一部分，也就是對於 design 的描述來評分。反之，有通過的話，report 僅需要如 HW3/4，附上相關資料即可獲得這部分得分數。

e. APR performance (30%)

如果沒有通過post layout simulation，

則不會有這部分的分數。

Cost = Area * Time (um^2*ns)
(A design with a lower cost is considered a better design.)
Area = Total area of Core in summaryReport.rpt
Time = Total time of testbench

這部分的分數會根據排名，

由30至0分線性遞減。

如果沒有通過post layout simulation，

則不會有這部分的分數。

範例：如果總共繳交的人數有11人，

Rank	Score
1	30
2	27
3	24
…	
10	3
11	0

7. 繳交規則

1. 繳交檔案如下: FINAL_b**901***.zip

   請將檔案打包進 FINAL_b**901*** 資料夾再進行 zip 壓縮

   參考指令如下:

   ```
   zip -r FINAL_b**901***.zip FINAL_b**901***/
   ```

2. 請檢查交上來的壓縮檔內 FINAL_b**901*** 資料夾中含以下檔案，檔案格式、資料夾階層或命名方式與規定不同將本次作業成績扣 10 分。

   - 檔案和路徑

```
FINAL_b**901***.zip
└→FINAL_b**901***
    ├→interpolation.v
    ├→interpolation_syn.v
    ├→interpolation_syn.ddc
    ├→interpolation_area.txt
    ├→interpolation_timing.txt
    ├→interpolation_power.txt
    ├→interpolation_apr.v
    ├→interpolation_apr.sdf
    ├→summaryReport.rpt
    ├→bxx901xxx_report.pdf
    └→design.spec
```

8. 繳交期限

[FINAL] 6/14(五)14:20 以前上傳至 NTU cool，不接受遲交。

同學如果有任何問題，可以優先在討論區發問，但也請避免包含程式碼的截圖。

要寄 email，請根據問題類型，同時寄給兩位助教，記得在信件前加 [積體電路設計] 避免漏信。

- Verilog 問題

  助教 林克帆 R11943131@ntu.edu.tw

  助教 謝承恩 R11943015@ntu.edu.tw

- 演算法問題

  助教 黃政勛 R12943018@ntu.edu.tw

  助教 林家弘 R12943021@ntu.edu.tw

  助教 曾泓瑜 R12943119@ntu.edu.tw

附錄 **Report**

- Cell area in Synthesis report (design vision)
  report_area > core_area.txt, find Total cell area.

```
Total cell area:          1473.343187
Total area:               14586.741777
```

- Cell area in APR report (Innovus)
  File > Report > Summary > choose Text only, file name: summaryReport.rpt > OK, find Total area of Core.

- Total time (vcs)

```
$finish called from file "tb.v", line 205.
$finish at simulation time              143800000
           V C S   S i m u l a t i o n   R e p o r t
Time: 143800000 ps
```

  You should report “ns’ rather than “ps.”

- Synthesis: No Latch (screenshot)

```
Inferred memory devices in process
        in routine counter line 61 in file
                '/home/raid3_1/user00/r00022/2013ICD/example/counter.v'.
===============================================================================
|    Register Name    |   Type      | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|      count_reg      | Flip-flop   |   2   |  Y  |  N |  N |  Y |  N |  N |  N |
|      count_reg      | Flip-flop   |   2   |  Y  |  N |  Y |  N |  N |  N |  N |
|       num_reg       | Flip-flop   |   2   |  Y  |  N |  N |  Y |  N |  N |  N |
|       num_reg       | Flip-flop   |   2   |  Y  |  N |  Y |  N |  N |  N |  N |
|      state_reg      | Flip-flop   |   2   |  Y  |  N |  Y |  N |  N |  N |  N |
===============================================================================
Presto compilation completed successfully.
Current design is now '/home/raid3_1/user00/r00022/2013ICD/example/counter.db:counter'
Loaded 1 design.
Current design is 'counter'.
design_vision>
Current design is 'counter'.
```

No Latch

- APR: NanoRoute Innovus Result (screenshot)