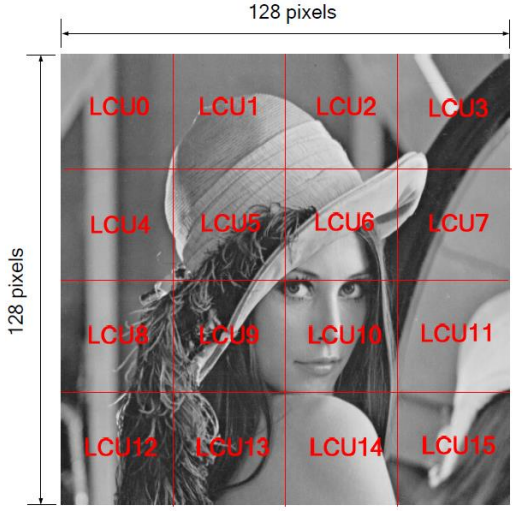
2021 ICD Project – Digital IC Design

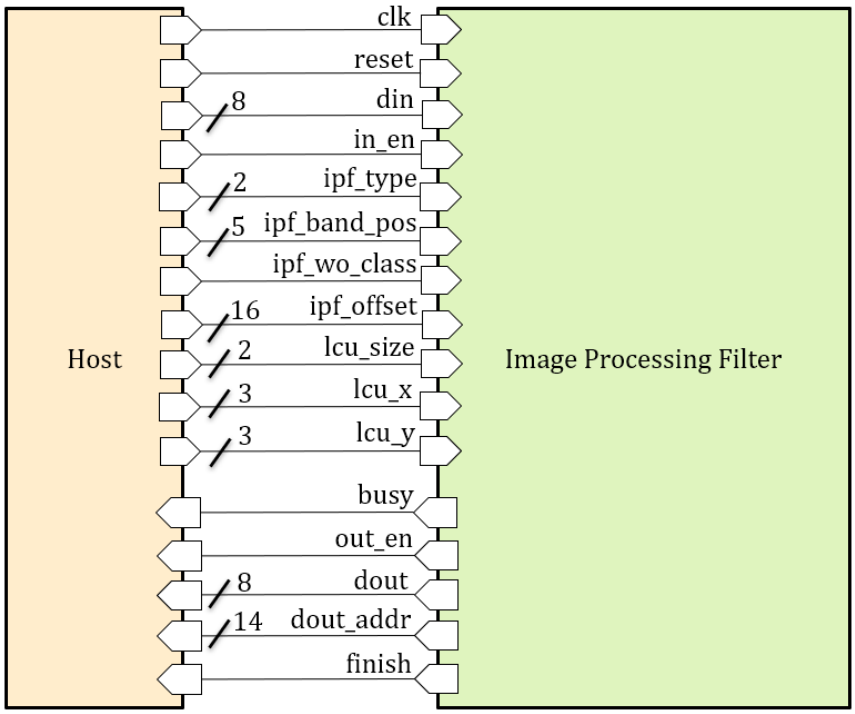
Image Processing Filter

1. 問題描述

請完成一 Image Processing Filter (後文以 IPF 表示)的電路設計，輸入為一已分成多個 LCUs(如圖一所示)的影像，影像大小固定為 128x128 Pixels，LCU Size 有 16x16、32x32、64x64 三種可能。16x16 與 64x64 為公開測資，32x32 為隱藏測資，IPF 對每一 LCU 各自進行獨立運算，最後把整張影像處理完後，將 finish 訊號拉為 High，系統會自動進行比對整張影像資料的正確性。有關 LCU 的定義與 IPF 詳細的運算方式，將描述於後。



圖一、將一張影像切割成多個 LCUs 之範例



圖二、系統方塊圖

2. 設計規格

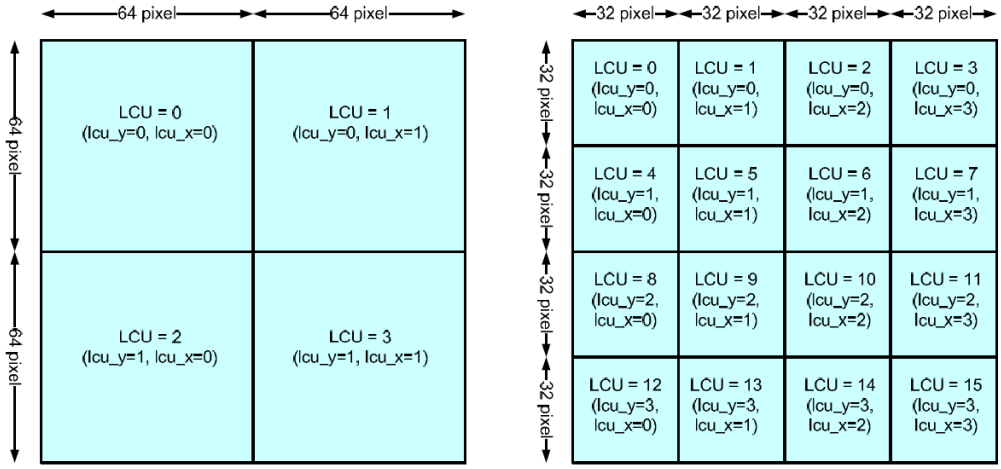
信號名稱	輸出/入	位元寬度	說明
clk	*Input*	1	時脈信號，本系統為同步於時脈正緣設計
reset	*Input*	1	高位準非同步(active high asynchronous)之系統重置訊號
din	*Input*	8	IPF 資料輸入的匯流排。Host 端會透過此匯流排將整張完整影像的訊號進行輸入。每一個週期僅能輸入一個 Pixel 值，且已輸入過的 Pixel 值將無法再重複輸入。輸入順序請參考 IPF 運算之輸入方式。
in_en	*Input*	1	資料輸入致能控制訊號。當 Host 偵測到 busy 訊號為 Low，din 便開始輸入訊號，此時 in_en 開始為 High， din 訊號輸入過程中，無論 busy 訊號為何，in_en 一直維持為 High，直到整張影像輸入完畢，in_en 才會為 Low。當 Host 端所有資料送完後，該訊號到模擬結束前將永遠維持為 Low。
ipf_type	*Input*	2	指定目前輸入 LCU 之 IPF 運算型式，包括：Type=2'd0：OFF 運算、Type=2'd1：PO 運算、Type=2'd2：WO 運算。
ipf_band_pos	*Input*	5	指定目前輸入 LCU 進行 PO 運算所需之 band index 值，該值範圍為：0~31。ipf_band_pos 僅有當 ipf_type=2'd1 時有效；其餘 ipf_type 不使用此值。
ipf_wo_class	*Input*	1	指定目前輸入 LCU 進行 WO 運算之兩種類別，包括水平方向濾波(1'b0)及垂直方向濾波(1'b1)。 ipf_wo_class 僅有當 ipf_type=2'd2 時有效；其餘 ipf_type 不使用 ipf_wo_class。
ipf_offset	*Input*	16	指定目前輸入 LCU 進行 PO 或 WO 運算之 Offset 值。 offset_0= ipf_offset[15:12], offset_1= ipf_offset[11:8] offset_2= ipf_offset[7:4], offset_3= ipf_offset[3:0] ipf_offset 僅有當 ipf_type=2'd1 或 2'd2 時有效。
lcu_size	*Input*	2	指定整張影像訊號切割成多個 LCU 之尺寸大小。 lcu_size = 0/1/2，每一塊 LCU 大小為 16x16/32x32/64x64 個 pixels。
lcu_x	*Input*	3	目前輸入的 LCU 是座落於一張影像的哪個水平方位。 例如：一張影像 128x128 pixels，一個 LCU 為 16x16，因此水平方向最多有 128/16=8 個 LCU。 => lcu_x 合理範圍為：0 ~ 7。
lcu_y	*Input*	3	目前輸入的 LCU 是座落於一張影像的哪個垂直方位。 例如：一張影像 128x128 pixels，一個 LCU 為 16x16，因此垂直方向最多有 128/16=8 個 LCU。 => lcu_y 合理範圍為：0 ~ 7。

busy	*Output*	1	IPF 忙碌之控制訊號。當為 High 時，表示系統正處於忙碌階段，告知 Host 端，暫停 din 資料的輸入；反之，當為 Low 時，表示告知 Host 端可繼續由 din 輸入資料。
out_en	*Output*	1	當 out_en 為 High 時，dout 與 dout_addr 為有效訊號
dout	*Output*	8	經 IPF 運算後在 dout_addr 位址所對應的 Pixel 值
dout_addr	*Output*	14	經 IPF 運算後 Pixel 值 dout 的位址
finish	*Output*	1	IPF 運算完畢之通知訊號。當所有的 LCU 經過個別運算完畢後，需將 finish 訊號拉為 High，以通知 Host 端開始進行所有影像訊號之比對。

3. 系統功能描述

3.1 IPF 系統架構

IPF 在進行影像處理之前，會先將一張影像切割成多個 LCUs(Large Coding Unit)，本題 IPF 輸入影像大小固定為 128x128 pixels，每個 pixel 為 8bit 灰階，LCU 大小依據 lcu_size 可分成 16x16、32x32、64x64 三種尺寸。如圖一所示，舉例來說，當 LCU Size 為 32x32，水平方向可切割出 128/32=4 個 LCUs，垂直方向也可切割出 4 個 LCUs，因此總共可切割出 16 個 LCUs。在同一張影像的每一個 LCU 都是相同大小。



圖一、將一張影像切割成多個 LCUs 之範例

切割後的每一塊 LCU，都會指定一項 IPF 運算的任務，需要依據每個 LCU 所指定的運算任務，個別完成運算後再輸出至 Host，即完成整個電路設計。IPF 運算有三種，包括(1)OFF、(2)PO、(3)WO，這三種運算方式，將詳細介紹如下。

3.1.1 IPF-OFF 運算方式

當 ipf_type 輸入 2'd0，表示該 LCU 要進行 OFF 運算。所謂 OFF，就是整個 LCU 內的所有 Pixels 數值，保持不變，直接輸出至 Host 端。

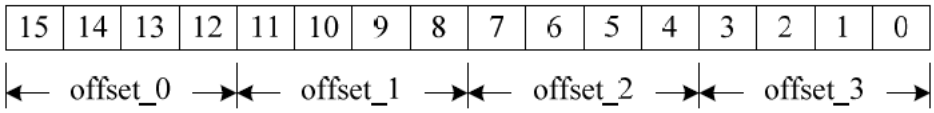
3.1.2 IPF-Pixel Operation(PO)運算方式

當 ipf_type 輸入 2'd1，表示該 LCU 要進行 PO 運算。PO 運算對單一 pixel 做各別處理，一個 8bit pixel 其數值範圍為 0~255，固定分成 32 Bands，即每個 Band 有包含 256/32=8 個不同的 Pixel 值，Band idx 0 為 Pixel 數值 0~7，Band idx 1 為 Pixel 數值 8~15，……，Band idx 31 為 Pixel 數值 248~255。

進行 PO 運算時，Host 端會先透過 ipf_band_pos 訊號輸入待處理的 LCU 其要維持 Pixel 數值的 band index，例如 ipf_band_pos=9，其前後包含自己三個連續 Bands(8、9、10)，落在此三個 Band 的 Pixel 數值保持不變。其餘 Band 則根據其 idx 與其對應 ipf_offset 所指定的 offset 數值相加後，取代原本的 Pixel 值。

ipf_offset 為 4 個 band offset 的合併，如下圖。Band idx 的對應方式為 idx 對 4 取餘數為 0 者(idx=0,4,8,…,28)使用 offset_0，對 4 取餘數為 1 者(idx=1,5,9,…,29)使用 offset_1，以此類推，4 個 band offset 在 PO 運算時的範圍皆為-7~+7。

註：運算後的 Pixel 數值要永遠保持在 0~255 之間，亦即加上 offset 值後，超過 255，就用 255 表示，加上 offset 值後，低於 0，就用 0 表示。



圖二、ipf_offset 對應位置

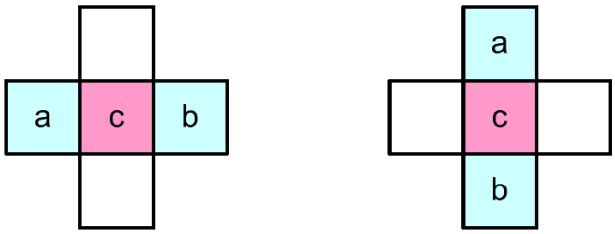
Input	Pixel Value before PO	Pixel Value after PO
ipf_type=2'd1 ipf_offset=16'h73DA =>offset = {+7, +3, -3, -6} ipf_band_pos=5'd11 => Band idx=10, 11, 12 不動	65 (Band idx = 8)	72=65+7(offset_0)
	109 (Band idx = 13)	112=109+3(offset_1)
	20 (Band idx = 2)	17=20-3(offset_2)
	190 (Band idx = 23)	184=190-6(offset_3)
	84 (Band idx = 10)	84

表一、PO 範例

3.1.3 IPF-Window Operation(WO)運算方式

當 ipf_type 輸入 2'd2，表示該 LCU 要進行 WO 運算。WO 運算有分成(1) ipf_wo_class=1'b0，水平方向濾波、(2) ipf_wo_class =1'b1，垂直方向濾波兩種方法，每個 LCU 作 WO 運算時，只會限定水平或垂直之其中一種 WO Class，兩種方法都會依據表二所列之 Condition，判斷 c 與相鄰 a、b 兩個 Pixels 之間的關係做分類，若屬於 category 0，便將原始的 Pixel 值+offset_0 後取代原始的 Pixel 值，依此類推，假若表二之四個 Condition 都沒滿足，即為 category 4，表示原始的 Pixel 值不做任何改變。Offset_0, Offset_1 在 WO 運算時的範圍為 0~+7，Offset_2, Offset_3 為-7~0。

註 1：a、b、c 之 Pixel 數值，僅可使用原始影像 Pixel 值，請勿使用運算後之 Pixel 值。
註 2：LCU 邊緣的 Pixels，不作計算，所以當執行水平方向濾波時，LCU 最左及最右邊的 Pixel 值保持不變，當執行垂直方向濾波時，LCU 最上及最下面的 Pixel 值保持不變。
註 3：運算後的 Pixel 數值要永遠保持在 0~255 之間，亦即加上 offset 值後，超過 255，就用 255 表示，加上 offset 值後，低於 0，就用 0 表示。



圖三、左圖為 ipf_wo_class=1'b0 所考量之 a, b, c 位置，右圖為 ipf_wo_class=1'b1 之情形

Category	Condition	Offset
0	$c < \min(a,b)$	0
1	$\min(a,b) \le c \le \max(a,b)$ 且 $c < \frac{a+b}{2}$	1
2	$\min(a,b) \le c \le \max(a,b)$ 且 $c > \frac{a+b}{2}$	2
3	$c > max\ (a,b)$	3
4	None of the above	None

表二、WO 運算對應類別

Input	a	b	c	Category	c after WO
ipf_type=2'd2 ipf_offset=16'h73DA =>offset = {+7, +3, -3, -6}	35	38	30	0	37=30+7
	34	27	30	1	33=30+3
	32	24	30	2	27=30-3
	29	26	30	3	24=30-6
	30	30	30	4	30

表三、WO 範例

3.2 IPF 系統輸入方式

若要將一張 8bits 128x128 影像儲存在 1D 的記憶體，其儲存方式如圖四所示，注意圖四的編號為記憶體位址值，因此一個 16x16 的 LCU 其儲存的影響 Pixel 值並非為連續記憶體位址儲存而成，而是例如 LCU 0 是由 SRAM Address 0~15、128~143、…、1920~1935 共計 16x16=256 個位址，256 個 Pixels 值所構成，為簡化其複雜度，本題影像的輸入訊號 din 送入 IPF 電路的順序為 Raster Scan (如圖四之 LCU size 16x16 為例，順序為 LCU 0 到 LCU 63，其中 LCU 0 的順序為 0~15、128~143、…、1920~1935，其餘 LCU 1~63 也依此順序類推)，另外，每當一個 LCU 資料 din 開始輸入時，ipf_type, ipf_band_pos, ipf_wo_class, ipf_offset, lcu_size, lcu_x, lcu_y 也將跟著進入 IPF 電路，四個 ipf 相關參數不見得每一種 ipf_type 運算都會使用到。

圖四、將一張 128x128 影像訊號儲存於記憶體之範例

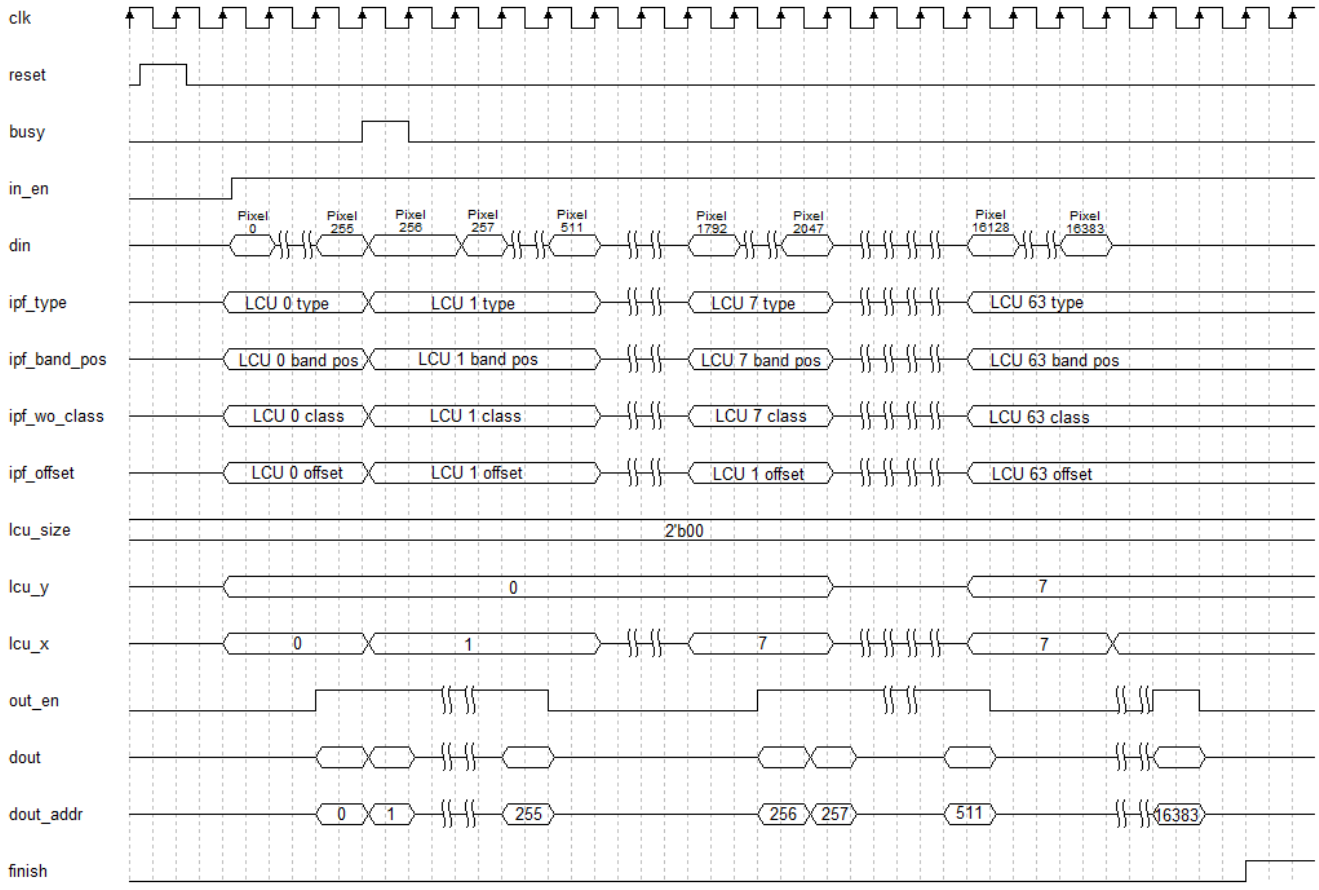
註 1：不要使用 register 2D array 把一整個 LCU 的值都先存下來，頂多存 3 個 row，不然後面合成/APR 會非常耗時。

註 2：已輸入至 ipf 電路的訊號 din、ipf、lcu 相關參數，無法再輸入第二次，倘若輸入的訊號暫時不想接收，可以用 busy 訊號拉為 High，即可暫停資料輸入，待處理完後，再將 busy 訊號拉為 Low，即可繼續輸入剩餘的資料。

3.3 IPF 系統輸出方式

IPF 運算後，須自己找適當時間將運算完結果傳輸至 Host 端的記憶體，其算完後擺放的位址，會與圖四記憶體擺放順序相同，以 LCU Size 16x16 為例，意即 LCU 0 運算完的結果，不會連續儲存 256 個位址，而是要自行儲存在 0~15、128~143、…、1920~1935 共計 256 個位址裡，其餘 LCU 也依此類推，請注意！最後所有的 LCU 都運算完後，請務必將 finish 訊號拉為 High，以通知 Host 端開始進行整張影像比對。

4. 系統時序圖



圖五、IPF 電路時序圖

首先會 reset 一個 cycle，IPF 電路初始化結束，Host 端判斷到 busy 訊號為 Low，因此開始輸入資料，in_en 拉為 High，din, ipf_type, ipf_band_pos, ipf_wo_class, ipf_offset, lcu_size, lcu_x, lcu_y 一起送出第一筆資料，開始運算第一個 LCU 0。

經過 Host 端送出 256 筆 Pixels，lcu_x 便加 1，LCU0 未必已完成計算，因此過程中可能會將 busy 訊號拉為 High，例如 Pixel 256 有輸入但是 busy 為 High，因此暫時維持 din 資料一直是 Pixel256，直到 clk 正緣發現 busy 是 Low，才會再送下一筆資料 Pixel 257。往後的 busy 訊號的動作行為，也是依此類推。

LCU1 的 256 筆 Pixels 全數送完後，過程中或送完後還會有其餘運算要作，請隨時使用 busy 訊號作好控制，因為送進來過的 Pixel 訊號就不會再送一次了。完成 LCU1 運算後，lcu_x 又會再加 1，就這樣類似動作一直加到 lcu_x=7 後，lcu_y 便會加 1，最後 lcu_y 加到等於 7，lcu_x 也等於 7，便完成整張影像的輸入。

運算過程中，只要有 Pixel 完成運算，隨時可以將 out_en 拉為 High，並同時輸出計算後的新 Pixel 值 dout，與其對應的位址 dout_addr，可以不用照位址的大小順序輸出。資料全數輸入完成後，in_en 拉為 Low，表示不再有新資料輸入，當結束所有運算後，finish 拉為 High 告知 Host 端開始進行資料比對，比對不會消耗任何模擬時間，整個模擬會立即結束。

5. 檔案規格

檔名	說明
testbench16.v	測試樣本檔，此 testbench 對應的 LCU size 為 16x16
testbench64.v	測試樣本檔，此 testbench 對應的 LCU size 為 64x64
IPF.v	設計檔，請勿更改輸入輸出宣告，同學請於此檔案內做設計
image_16x16/64x64.dat	影像 din 的輸入訊號來源
lcu_16x16/64x64.dat	ipf_type, ipf_band_pos, ipf_wo_class, ipf_offset 之輸入訊號來源，該 Type 不需要用到的訊號，會以 High-Z 訊號表示。
golden_16x16/64x64.dat	經過 IPF 運算後之結果
sythesis.tcl	合成用 design contstraint 資料，可在裡面修改 cycle
.synopsys_dc.setup	合成用 Design compiler 環境設定檔(不須更動)
tsmc13.v	合成模擬用製程檔
./layout/IPF_APR.sdc	APR 用 design contstraint 資料，可在裡面修改 cycle
./layout 其餘檔案&./library	APR 用資料(不須更動)
./syn/tsmc13_neg.v	APR 模擬用製程檔

6. RTL/Synthesis/APR 模擬指令

本次提供兩個 testbench，其模擬相關指令如下。

ncverilog testbench16.v IPF.v

如果要輸出波形，可以使用 *+define+FSDB* 或者是 *+define+VCD* 並且加上 *+access+r*

ncverilog testbench16.v IPF.v +define+FSDB +access+r

ncverilog testbench16.v IPF.v +define+VCD +access+r

Synthesis 合成後 testbench，模擬相關指令如下。

ncverilog testbench16.v IPF_syn.v tsmc13.v +define+SDFSYN

如果要輸出波形，可以使用 *+define+FSDB* 或者是 *+define+VCD* 並且加上 *+access+r*

調整 testbench 中的 cycle time，必須和.tcl 中的 cycle time 相同，如果模擬沒過，可以慢慢提高 testbench 中的 cycle time，直到通過為止。

APR 後的 testbench，模擬相關指令如下。

ncverilog testbench16.v IPF_APR.v -v ./syn/tsmc13_neg.v +define+SDFAPR +ncmaxdelays

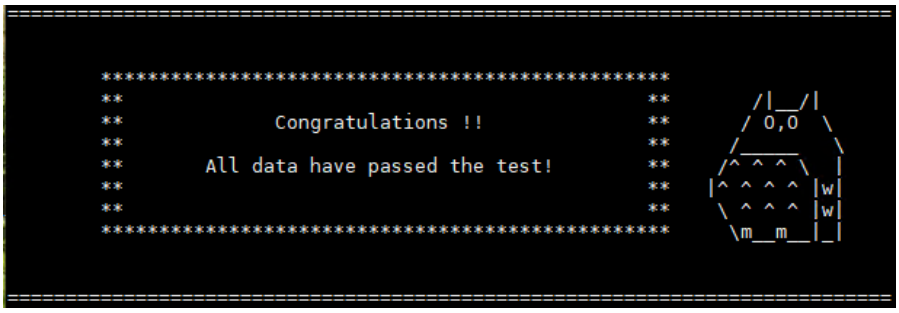
如果要輸出波形，可以使用 *+define+FSDB* 或者是 *+define+VCD* 並且加上 *+access+r*

調整 testbench 中的 cycle time，必須和.sdc 中的 cycle time 相同，如果模擬沒過，可以慢慢提高 testbench 中的 cycle time，直到通過為止。

如果跳一兩個 error 說 IPF_APR.v 中某幾行有 ANTENNA…之類的問題，進入檔案把它們註解掉即可。

7. 模擬結果

如果模擬結果都正確的話，應該可以看到如下圖的結果



圖六、模擬結果正確

有錯誤時，則可能會出現

```
Address:    0 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 0): ab !=expect aa
Address:    1 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 1): 9a !=expect 99
Address:    2 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 2): 9a !=expect 99
Address:    3 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 3): 9a !=expect 99
Address:    4 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 4): ab !=expect aa
Address:    5 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 5): 9a !=expect 99
Address:    6 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 6): 9a !=expect 99
Address:    7 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 7): 9a !=expect 99
Address:    8 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 8): ab !=expect aa
Address:    9 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x= 9): 9a !=expect 99
Address:   10 => LCU(lcu_y= 0, lcu_x= 0): ERROR at Pixel(y= 0, x=10): ab !=expect aa
```

圖七、模擬結果錯誤

8. 作業要求

1. 通過兩個 testbench 的 RTL Level 模擬 (40%)
2. 通過兩個 testbench 的 Synthesis Level 模擬 (20%)
3. 同時通過兩個 testbench+一個隱藏測資的 APR Level 模擬 (10%)
4. 通過 testbench16 的 APR 設計品質 (30%)

   Score = Area*Timing、越小越好

   Area = Area report 中的 total cell area

   Timing = testbench16 所執行的時間 (舉例如下圖為 315ns)

```
Simulation complete via $finish(1) at time 315 NS + 0
```

提示：盡量把 cycle 壓低，slack 接近 0

依照 Score 給分

第 1 名到第 6 名：30、28、26、24、22、20 分

第 7 名到第 16 名：19、18、17、16、15、14、13、12、11、10 分

第 17 名到第 24 名：9、9、8、8、7、7、6、6 分

第 25 名後：5 分

9. 繳交檔案與期限

繳交檔案如下：Project_b0*901***.zip

分類	檔案名稱	描述
RTL	IPF.v	RTL Verilog Code
Synthesis	IPF_syn.v	Synthesis Verilog Code
Synthesis	IPF_syn.sdf	SDF file
Synthesis	IPF_syn.ddc	DDC file
Synthesis	IPF_timing.txt	Timing Report
Synthesis	IPF_area.txt	Area Report
Synthesis	IPF_power.txt	Power Report
APR	IPF_APR.v	Netlist Verilog Code
APR	IPF_APR.sdf	SDF file
Report	b0*901***_report.pdf	填寫 report.doc 存成 pdf

6/30 (三)中午 13:00 以前上傳至 Ceiba

同學如果有任何問題，請先盡量透過 email 詢問助教。剛開始學習大家遇到的問題都會蠻像的，如果要寄 email，請同時寄給兩位助教，記得在信件前加 [積體電路設計] 避免漏信。

助教 林奕憲 d06943006@ntu.edu.tw

助教 葉陽明 d05943006@ntu.edu.tw