

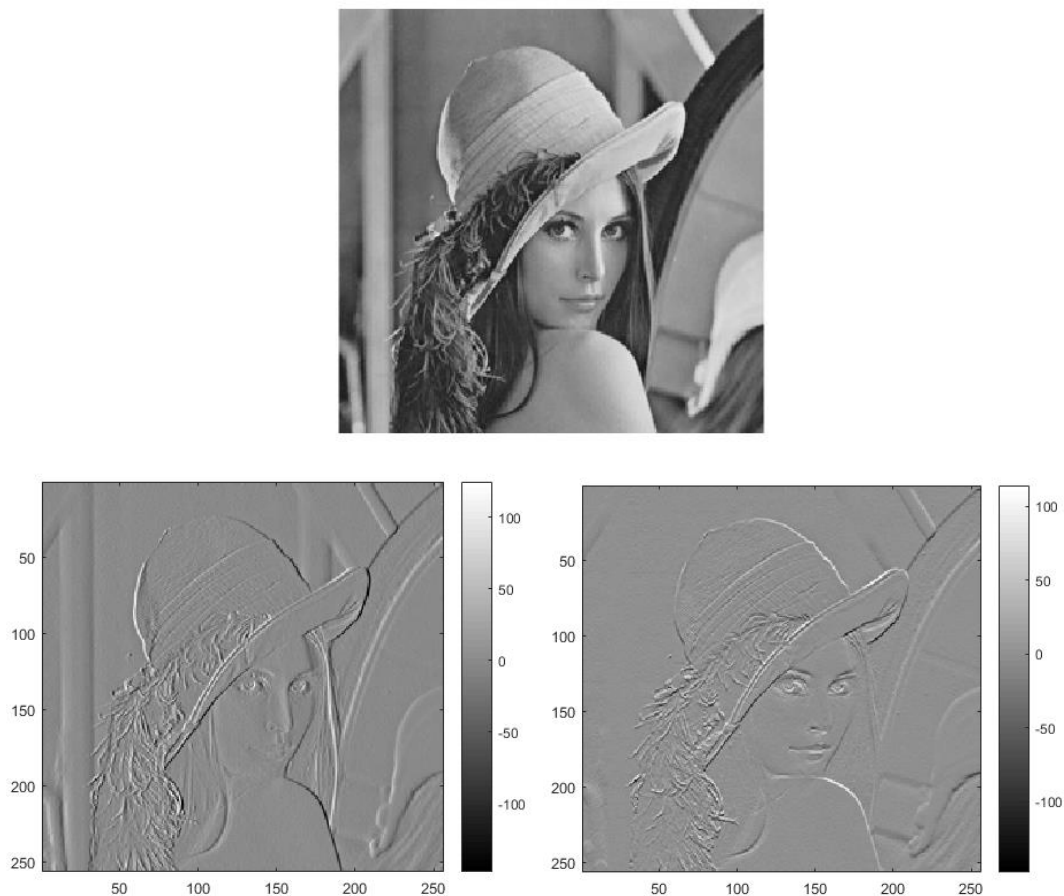
# Integrated Circuit Design

## Homework #3 Image Gradient

Due: 2018/05/09 13:00

### 1. 問題描述

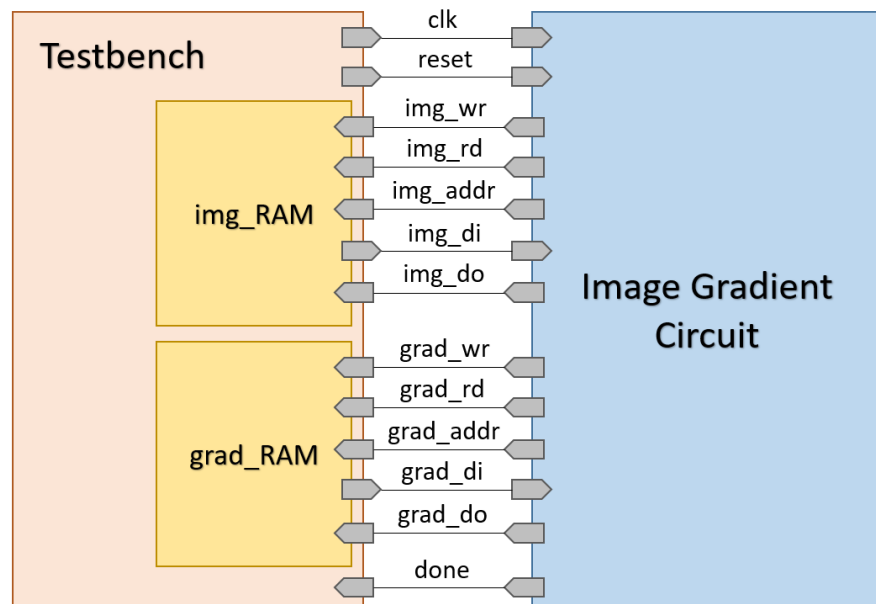
本次作業須完成一 Image Gradient 電路(後文以 IG 電路表示)，其輸入為一張大小 256x256 之 8 bits 灰階圖像，此圖像會儲存在 Host 端的輸入圖像記憶體 (**img\_RAM**) 中。IG 電路需從 **img\_RAM** 中讀取圖像資料，再對此圖像資料進行 Gradient Extraction 運算，並將運算結果寫入 Host 端的輸出結果圖像記憶體 (**grad\_RAM**) 中。運算全部完成後，將輸出訊號 **done** 拉為 High，接著系統會自動比對 **grad\_RAM** 內資料的正確性。



圖一、輸入照片與其 Image Gradient 舉例

左下方為 x 方向 Gradient，右下方為 y 方向 Gradient

## 2. 設計規格



圖二、系統方塊圖

Signal Name	Direction	Width (bit)	Description
clk	Input	1	系統時脈訊號。
reset	Input	1	非同步系統重置訊號。當此訊號為 High 時表示系統重置。
img_rd	Output	1	img_RAM 讀取致能訊號。當此訊號為 High 時表示要向 img_RAM 索取資料。
img_wr	Output	1	img_RAM 寫入致能訊號。當此訊號為 High 時表示要向 img_RAM 寫入資料。
img_addr	Output	16	img_RAM 位址匯流排。需透過此匯流排向 img_RAM 指示讀取或寫入之圖像資料的位址。
img_di	Input	8	img_RAM 讀取資料匯流排。Host 端利用此匯流排將 img_RAM 內的圖像資料讀取送到 IG 端。
img_do	Output	8	img_RAM 寫入資料匯流排。IG 端利用此匯流排將圖像資料寫入到 Host 端的 img_RAM。
grad_rd	Output	1	grad_RAM 讀取致能訊號。當此訊號為 High 時表示要向 grad_RAM 索取資料。
grad_wr	Output	1	grad_RAM 寫入致能訊號。當此訊號為 High 時表示要向 grad_RAM 寫入資料。

grad_addr	Output	16	grad_RAM 位址匯流排。需透過此匯流排向 grad_RAM 指示讀取或寫入之圖像資料的位址。
grad_di	Input	20	grad_RAM 讀取資料匯流排。Host 端利用此匯流排將 grad_RAM 內的圖像資料讀取送到 IG 端。
grad_do	Output	20	grad_RAM 寫入資料匯流排。IG 端利用此匯流排將圖像資料寫入到 Host 端的 grad_RAM。
done	Output	1	電路計算完成後，將此訊號拉為 High。

表一、訊號說明

### 3. 系統功能描述

當電路的 reset 訊號結束後，系統端會自動將影像存入 img\_RAM 中，此時 img\_RAM 內的資料皆為有效值。請注意，若非必要，**請不要更動 img\_RAM 裡的值**，以免結果錯誤。

本次電路將會使用到兩塊 RAM，分別是存放灰階影像的 img\_RAM，以及存放電路計算結果的 grad\_RAM。其中 img\_RAM 的大小為 256 x 256 x 8 bits，而 grad\_RAM 大小為 256 x 256 x 20 bits，兩種 RAM 的解讀方式將於作業說明第 4.5 點詳述。兩塊 RAM 皆具有隨機存取記憶體(Random Access Memory, RAM)的特性，使用者可以存取任意位置，且可重複讀取或寫入同一位置數次。需要注意此 RAM 為 Single-Port SRAM，表示**一個 cycle 內，RAM 只能選擇讀取或是寫入其中一種動作**，無法同一 cycle 內同時執行寫入與讀取兩個動作。

以下說明 Image Gradient 的計算方式，本次電路為降低設計難度，將採用最簡單的 Image Gradient 計算方式，此方式可能與常用的方式不同，請各位多加注意。

首先我們定義影像最左上角的座標為(0,0)，**右方為 X 軸的正方向，下方為 Y 軸的正方向**，影像座標 $(x_0, y_0)$ 點的像素數值將以 $I(x_0, y_0)$ 表示，其右方座標 $(x_0 + 1, y_0)$ 的像素數值以 $I(x_0 + 1, y_0)$ 表示，以此類推。

而影像的 Gradient 將分成 x 與 y 方向，分別以 $G_x$ 和 $G_y$ 表示，以下為 $G_x$ 和 $G_y$ 的計算方式：

$$G_x(x_0, y_0) = I(x_0 + 1, y_0) - I(x_0, y_0)$$

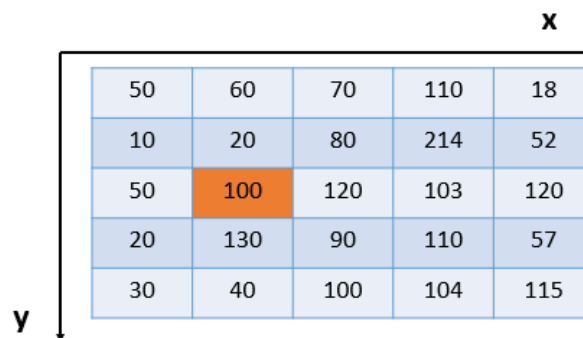
$$G_y(x_0, y_0) = I(x_0, y_0 + 1) - I(x_0, y_0)$$

若在影像最右方的邊界，因沒有右邊的數值，因此 x 方向的 Gradient 無法計算，同理在影像最下方的邊界，y 方向的 Gradient 也無法計算，因此右方及下方邊界的 Gradient 在最後檢查時會被忽略，也就是此區域在 grad\_RAM 內為任意值皆可通過檢查，**各位不必對此邊界進行特別處理**。下方圖示舉例說明：

Ex: 假設輸入影像大小為 5x5，其值如表格內所示，則對橘色點而言

$$G_x = 120 - 100 = 20$$


$$G_y = 130 - 100 = 30$$



	50	60	70	110	18
	10	20	80	214	52
	50	100	120	103	120
	20	130	90	110	57
	30	40	100	104	115

圖三、Image Gradient 計算舉例

經計算後可以得到 Gradient 如下圖，圖中標示為綠色處，因其右方或下方的 Gradient 無法計算，因此這些點最後並不會被檢查，此區域在 grad\_RAM 中放置任意值皆可通過檢查。

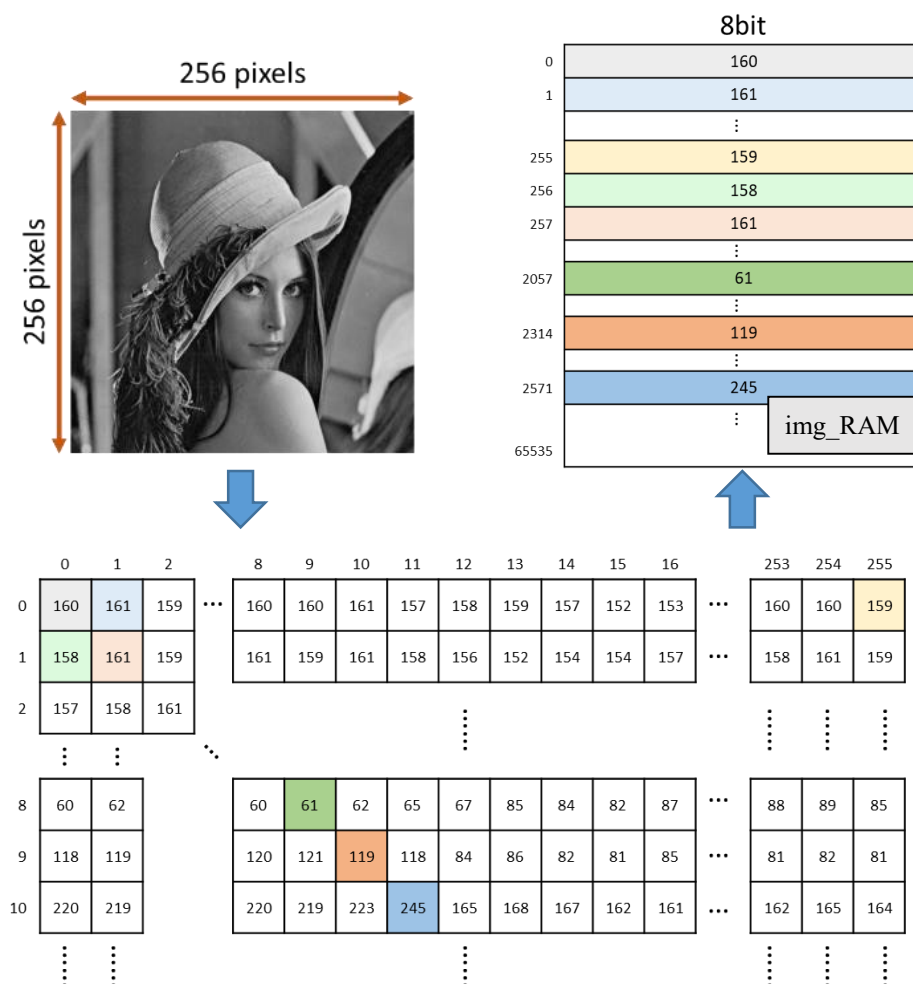


	10,-40	10,-40	40,10	-92,104	-
	10,40	60,80	134,40	-162,-111	-
	50,-30	20,30	-17,-30	17,7	-
	110,10	-40,-90	20,10	-53,-6	-
	-	-	-	-	-

圖四、Image Gradient 不檢查區域

#### 4. 輸入影像記憶體(img\_RAM)對應方式及時序規格

輸入灰階影像之大小固定為 256x256 pixels，每個 pixel 為 8 bits 資料，範圍為 0 到 255；而 img\_RAM 有 256x256 共 65536 個位址，每個位址內儲存 8 bits 資料，因此每個位址剛好可以存放 1 個 pixel 的影像資料。影像資料放置順序由左至右、由上至下，如下圖所示。

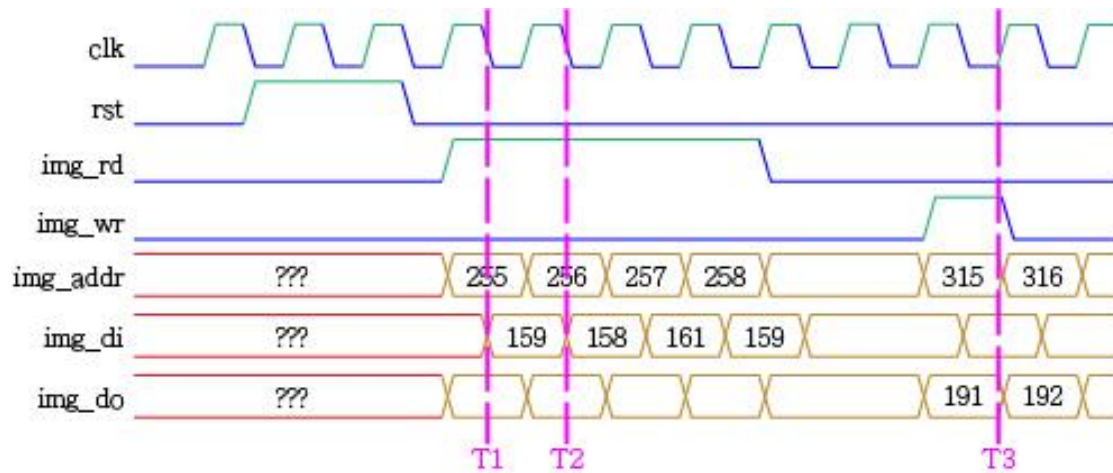


圖五、img\_RAM 資料儲存範例

當負緣時脈訊號時，img\_RAM 會判斷是否要進行讀取的動作，若 img\_rd 訊號為 High(如下圖的 T1 及 T2 時間)，進行記憶體資料讀取，將 img\_addr 訊號所指定之位址的資料，由 img\_di 匯流排送出；若 img\_rd 為 Low，則 img\_RAM 將不會進行任何動作。

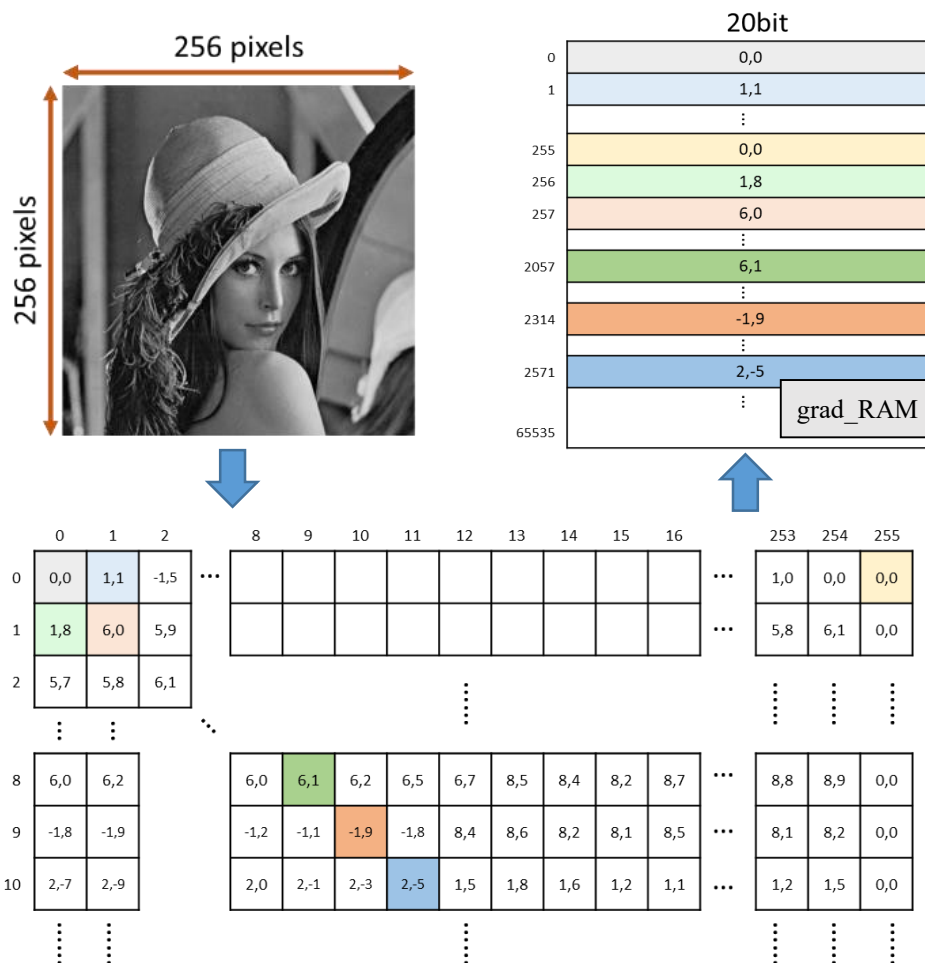
當正緣時脈訊號時，img\_RAM 會判斷是否要進行寫入的動作，若 img\_wr 訊號為 High(如下圖的 T3 時間)，則此時在 img\_do 匯流排上的資料就會被寫入到 img\_addr 所指定的位址內；如果 img\_wr 為 Low，則 img\_RAM 將不會進行任何動作。

本此作業使用的記憶體並非真實的記憶體，是由 testbench 所模擬出來的，因此不需考慮讀取延遲的問題。

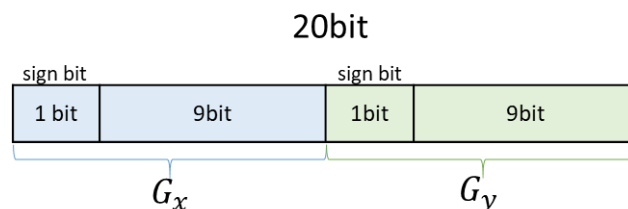


## 5. 輸出結果記憶體(grad\_RAM)對應方式及時序規格

輸出結果之大小為 256x256 pixels，每個 pixel 有 20 bits 的資料，由 10 bits 的 x 方向 gradient 和 10 bits 的 y 方向 gradient 組成；而 grad\_RAM 有 256x256 共 65536 個位址，每個位址內儲存 20 bits 資料，剛好可以存放 1 個 pixel 的兩個方向的 gradient。結果資料放置順序由左到右，由上往下，如下圖所示。



20bits 的 gradient 資料解讀方式如下，由於輸入影像為 8 bits 的資料，其值介於 0~255 之間的整數，經相減運算後的範圍為-510~510 之間的整數，需要使用 10 bits 的空間來儲存，並**統一規定使用 10 bits 2's complement** 來表示，最後將兩個方向的 gradient 訊號如下圖所示合併為 20 bits，放入 grad\_RAM。



grad\_RAM 的時序控制方式和 img\_RAM 一樣，因此不再多加贅述，兩個 RAM 的差別只有每個位址儲存資料的寬度，img\_RAM 為 8 bits，grad\_RAM 為 20 bits。

## 6. 時序規格

本次作業系統不會自動將資料輸入，請同學們依自己電路設計的需要，藉由控制 img\_RAM 的相關訊號(wr, rd, addr, do)，來操作記憶體獲得資料。電路計算完的結果，一樣藉由控制 grad\_RAM 的相關訊號(wr, rd, addr, do)，放置到正確的位置中。

當完成全部的計算後，請自行將 done 訊號拉為 High，讓 testbench 開始進行 grad\_RAM 內的結果比對。

## 7. 運算範例

下圖(a)為一部分的影像資料，下圖(b)運算完的 gradient 數值。

	8	9	10	11	12		8	9	10	11	12
6	156	156	155	154	154	6	0,0	-1,0	-1,0	0,0	2,0
7	156	156	155	154	154	7	0,-1	-1,-1	-1,1	0,2	0,-1
8	155	155	156	156	153	8	0,2	1,1	0,-1	-3,-1	0,0
9	157	156	155	155	153	9	-1,1	-1,1	0,0	-2,-1	1,1
10	158	157	155	154	154	10	-1,0	-2,1	-1,1	0,2	1,1

(a) img
(b) grad



## 8. 檔案說明

檔名	說明
testfixture.v	測試樣本檔。此測試樣本檔定義了時脈週期與測試樣本之輸入信號。
IG.v	設計檔，請勿更改輸入輸出宣告，請於此檔案內做設計
./dat/image.dat	灰階影像測試樣本
./dat/gradient.dat	Gradient 結果測試樣本
.cshrc_icd	Tool 環境設定檔

表三、檔案說明

## 9. 模擬指令

### I. ncverilog

- 使用 ncverilog 前須先輸入下列指令  
source /usr/cadence/CIC/incisiv.cshrc
- RTL-level 模擬指令如下  
ncverilog testfixture.v IG.v +access+r

### II. nWave

- 使用 nWave 前須先輸入下列指令  
source /usr/spring\_soft/CIC/verdi.cshrc
- 開啟 nWave 指令  
nWave &

## 10. 模擬結果

如果模擬有錯誤，testbench 會顯示期望的結果跟實際的結果：

```
Gradient y of pixel      0 is wrong! Your gradient y is 001, but expected gradient y is 3ff
Gradient x of pixel      1 is wrong! Your gradient x is 000, but expected gradient x is 3ff
Gradient y of pixel      1 is wrong! Your gradient y is 000, but expected gradient y is 001
Gradient x of pixel      2 is wrong! Your gradient x is 002, but expected gradient x is 000
Gradient x of pixel      3 is wrong! Your gradient x is 3fe, but expected gradient x is 002
Gradient x of pixel      4 is wrong! Your gradient x is 3fd, but expected gradient x is 3fe
Gradient x of pixel      5 is wrong! Your gradient x is 001, but expected gradient x is 3fd
Gradient y of pixel      5 is wrong! Your gradient y is 001, but expected gradient y is 000
Gradient x of pixel      6 is wrong! Your gradient x is 3fd, but expected gradient x is 001
Gradient y of pixel      6 is wrong! Your gradient y is 000, but expected gradient y is 001
Gradient x of pixel      7 is wrong! Your gradient x is 004, but expected gradient x is 3fd
```

```
Gradient of Pixel: 0 ~    999 are wrong ! The wrong pixel reached a total of    1594 or more !
Gradient of Pixel: 0 ~   1999 are wrong ! The wrong pixel reached a total of    3294 or more !
Gradient of Pixel: 0 ~   2999 are wrong ! The wrong pixel reached a total of    5016 or more !
Gradient of Pixel: 0 ~   3999 are wrong ! The wrong pixel reached a total of    6748 or more !
Gradient of Pixel: 0 ~   4999 are wrong ! The wrong pixel reached a total of    8451 or more !
Gradient of Pixel: 0 ~   5999 are wrong ! The wrong pixel reached a total of   10192 or more !
```



```

-----
FAIL! There are      117677 errors at functional simulation !
----- The test result is ..... FAIL -----

```

如果模擬無法在規定 cycle 數完成，會顯示以下結果：

```

-----
Error!!! Running out of time!
There is something wrong with your code!
-----The test result is .....FAIL -----
-----
Simulation complete via $finish(1) at time 100 MS + 0

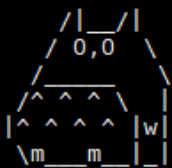
```

如果模擬結果正確，testbench 會顯示以下結果：

```

=====The test result is ..... PASS=====
*****
**                                     **
**          Congratulations !!          **
**                                     **
** All data have been generated successfully! **
**                                     **
*****

```



```

=====

```

## 11. 作業評分

本次作業滿分標準為：

通過 RTL-level 模擬，跑完 testbench 沒有任何 error。

所需繳交的檔案為：

Type	File Name	Description
RTL-level	IG.v	RTL Verilog Code
Report	ICD_HW3_StudentID.pdf	內需包含 RTL-level 模擬結果的截圖。

請將以上檔案壓縮成 YourID#\_HW3\_v\*.tar.gz 並使用 FTP 繳交

[FTP info]

IP: 140.112.48.159

Port: 56666

User name: icd2018

Password: ntueeicd