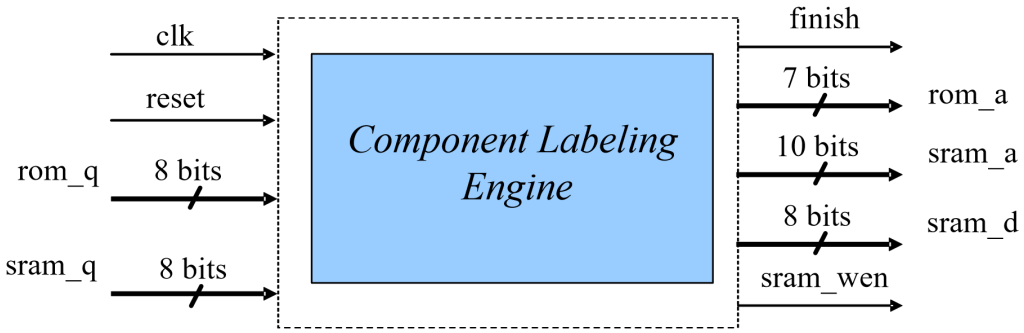
107-1 CVSD – HW3

Component Labeling Engine

1.問題描述

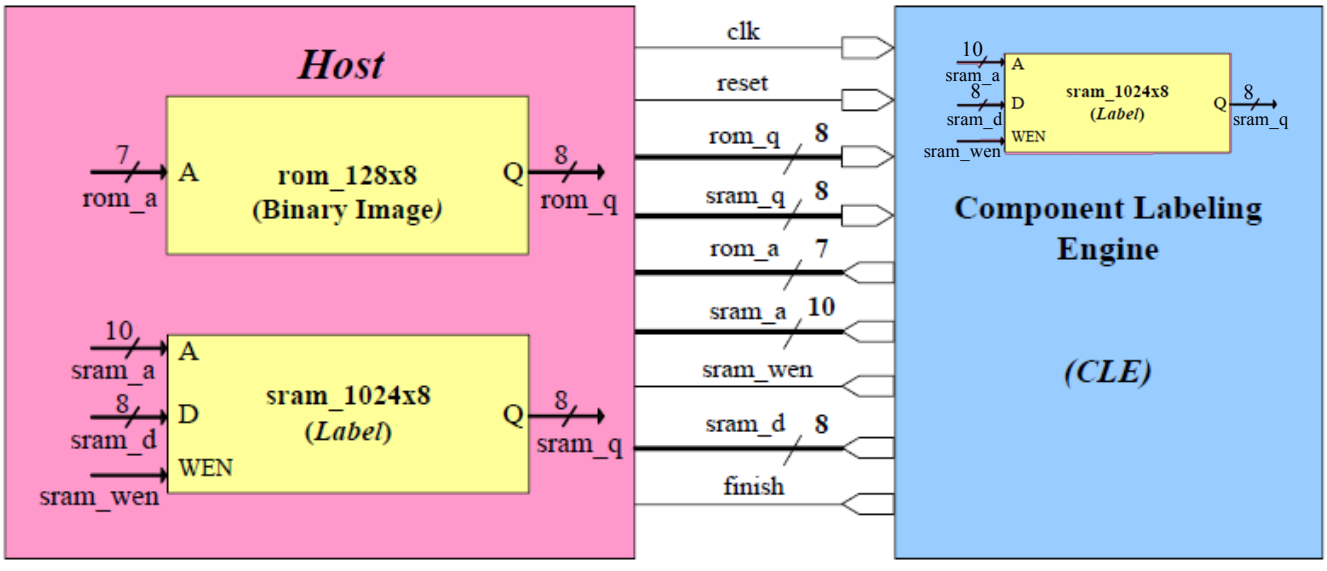
請完成一 Component Labeling Engine(後文以 **CLE** 表示)的電路設計。此電路可將任意 32x32 大小之二元影像(Binary Image)訊號，尋找出該圖片所有前景物件，然後在同一物件上給予相同的編號，完成所有物件的編號後將其儲存於 SRAM 記憶體，即完成 CLE 電路功能。有關 CLE 詳細規格將描述於後。

本電路各輸入輸出信號的功能說明，請參考表一。同學必須根據下一節所給的設計規格及附錄 A 中的測試樣本完成設計驗證。



圖一、Component Labeling Engine 之方塊圖

2.設計規格

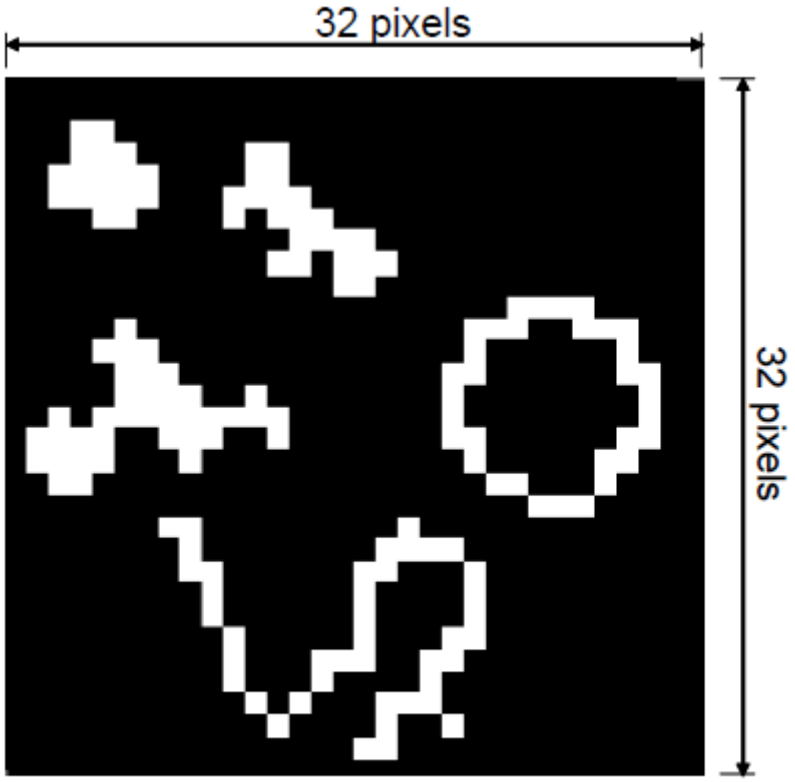


圖二、系統方塊圖

2.2 輸入/輸出介面

表 1 -輸入/輸出訊號

Signal Name	I/O	Width	Simple Description
clk	I	1	本系統為同步於時脈正緣之同步設計。 (註: Host 端採 clk ”正”緣時送資料。)
reset	I	1	高位準非同步(active high asynchronous)之系統重置信號。
rom_a	O	7	CLE 電路至 ROM 讀取資料時會用到的 Address Bus。 CLE 傳送位址訊號至 ROM，就是透過 rom_a 傳遞，該訊號直接與 ROM 的 Address 腳位 A 相連。有關 ROM 的詳細規格說明，可參考 ROM 記憶體附件。 註：本題的 ROM CEN 訊號，在 Host 端已設為 1’b0。
rom_q	I	8	CLE 電路至 ROM 讀取資料時會用到的 Data Bus。 CLE 從 ROM 讀取資料，就是透過 rom_q 傳遞，該訊號直接與 ROM 的輸出訊號腳位 Q 相連。
sram_a	O	10	CLE 電路至 SRAM 存取資料時會用到的 Address Bus。 CLE 傳送位址訊號至 SRAM，就是透過 sram_a 傳遞，該訊號直接與 SRAM 的 Address 腳位 A 相連。有關 SRAM 的詳細規格說明，可參考 SRAM 記憶體附件。 註：本題的 SRAM CEN 訊號，已設為 1’b0
sram_d	O	8	CLE 電路至 SRAM 寫入資料時會用到的 Data Bus。 CLE 若要寫入資料到 SRAM，就是透過 sram_d 傳遞，該訊號直接與 SRAM 的輸入訊號腳位 D 相連。
sram_q	I	8	CLE 電路至 SRAM 讀取資料時會用到的 Data Bus。 CLE 若要從 SRAM 讀取資料，就是透過 sram_q 傳遞，該訊號直接與 SRAM 的輸出訊號腳位 Q 相連。
sram_wen	O	1	CLE 電路對 SRAM 作 Read/Write 的控制訊號。當該訊號為 Low，表示 CLE 要對 SRAM 作寫入，反之，當該訊號為 High，表示 CLE 要對 SRAM 作讀取。該訊號直接與 SRAM 的控制訊號腳位 WEN 相連。
finish	O	1	告知 Host 端，CLE 電路運算完畢，請 Host 端開始檢查 SRAM 的內容值是否運算正確。當為 Low 時，表示 CLE 電路還在運算，Host 端不作檢查；反之，當為 High 時，表示 CLE 電路已運算完畢，Host 端可以開始檢驗 CLE 電路運算結果是否正確。

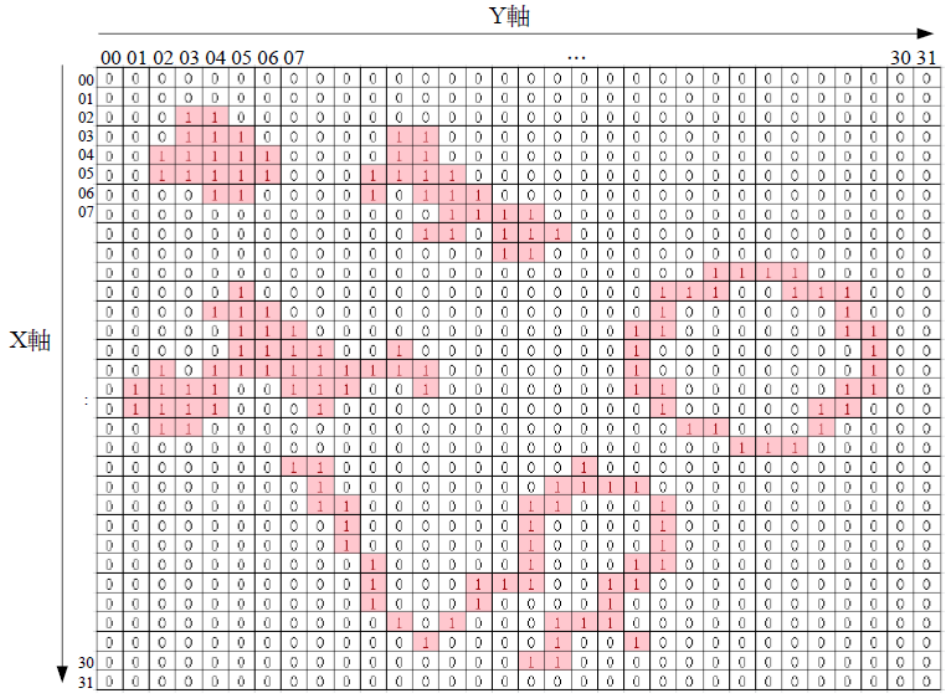


圖三、二元影像訊號

2.3 系統描述

圖三為一張 32x32 大小的二元影像訊號，影像中的每一點訊號稱為 Pixel，每個 Pixel 只有 0 或 1 兩種訊號，0 表示背景(圖中的黑色區域)，1 表示前景物件(圖中的白色區域)。CLE 電路可處理固定影像 32x32 pixels 之圖片，從中尋找出該圖片所有前景物件(即訊號 1 所在處)，然後由參賽者自行判斷，這些訊號 1 的 Pixel 是否有連接在一起(要看到整張完整圖片為主)，有連接在一起的 Pixels，將被視為同一物件，需針對這些物件作處理，同一物件的所有 Pixels 都要被編上一

組相同的編號(編號可自訂)，該編號一旦使用後，便不可再重覆使用於其他物件上，當完成所有物件的編號後，將其結果儲存於 SRAM 記憶體，即完成 CLE 電路功能。



圖四、二元影像各 Pixel 之實際訊號值

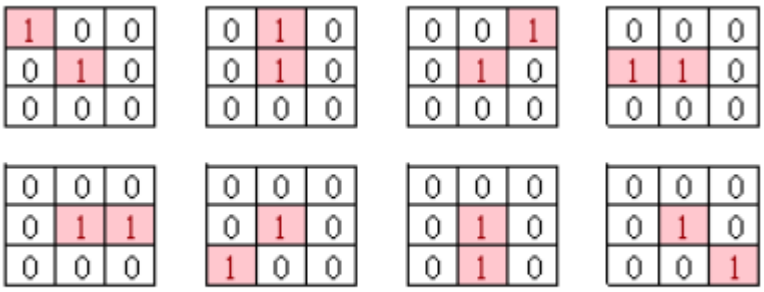
Address	ROM_128x8
0	X軸=00, Y軸=00 ~ 07
1	X軸=00, Y軸=08 ~ 15
2	X軸=00, Y軸=16 ~ 23
3	X軸=00, Y軸=24 ~ 31
4	X軸=01, Y軸=00 ~ 07
5	X軸=01, Y軸=08 ~ 15
126	X軸=31, Y軸=16 ~ 23
127	X軸=31, Y軸=24 ~ 31

圖五、二元影像訊號儲存於 ROM 的方式

2.3.1 CLE 電路的輸入：二元影像訊號已儲存於 ROM

圖四為圖三中二元影像之實際訊號值，該訊號值已存於 128x8 的 ROM 裡，儲存方式如圖五所示，ROM 位址 0，儲存圖五之【X 軸座標 00、Y 軸座標 00~07】，共 8bits 資料，該筆資料的 MSB 為【X 軸座標 00、Y 軸座標 00】，LSB 為【X 軸座標 00、Y 軸座標 07】，ROM 位址 1，儲存圖五之【X 軸座標 00、Y 軸座標 08~15】，…，ROM 位址 127，儲存圖五之【X 軸座標 31、Y

軸座標 24~31】。參賽者可依電路需求自行至 ROM 讀取資料一次或多次。ROM 的 CEN 訊號，已在 Host 端設定為永遠開啟(即 CEN=1'b0)。



圖六、九宮格範圍內 Pixel 相連的八種情形

2.3.2 CLE 電路的運算方法

CLE 電路運算方式如下：

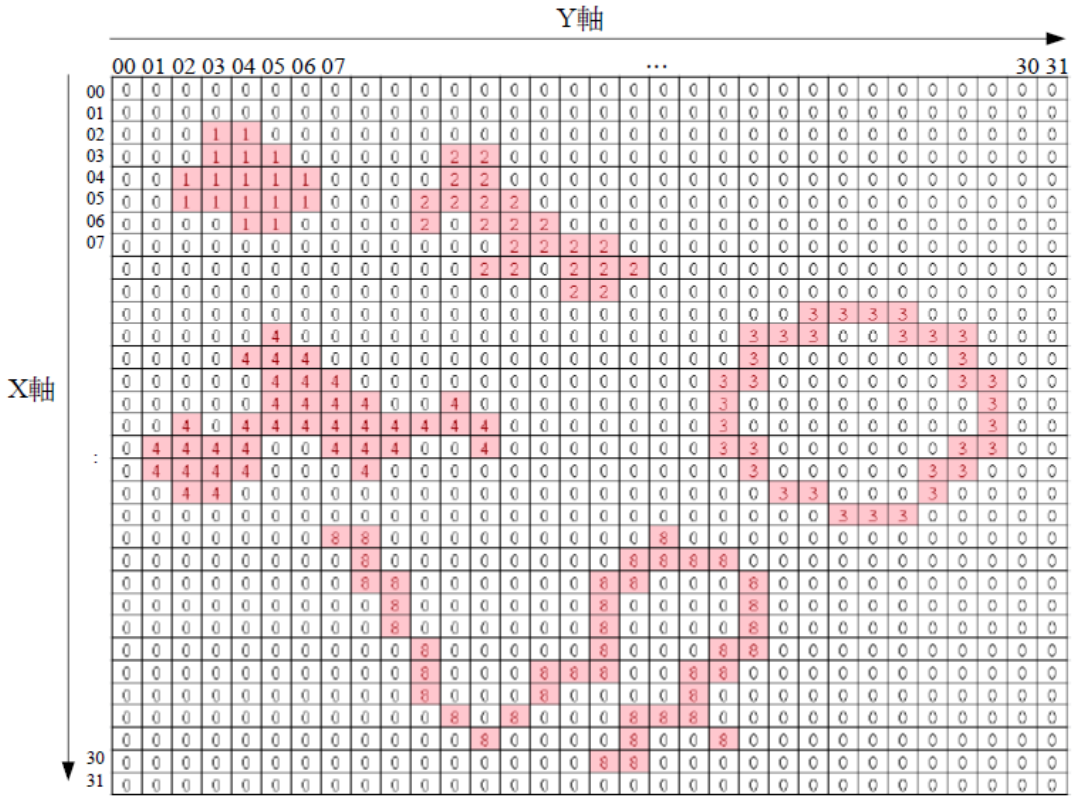
A. 在圖四中尋找 Pixel 訊號值為 1 的點，即找到前景物件所在處，接著自行判斷該 Pixel 於九宮格的範圍內是否有跟其他 Pixel 相連在一起，像圖六這八種可能的情形都稱之為有相連，相連在一起的 Pixel 將被視為同一物件，反之，不相連的 Pixels，將被視為不同物件。

注意：參賽者判斷各 **Pixel** 是否是同一物件，最後要看到整張完整圖片為主。

B. 同一物件中的每個 Pixel 都要編上相同的編號，該編號可自定。

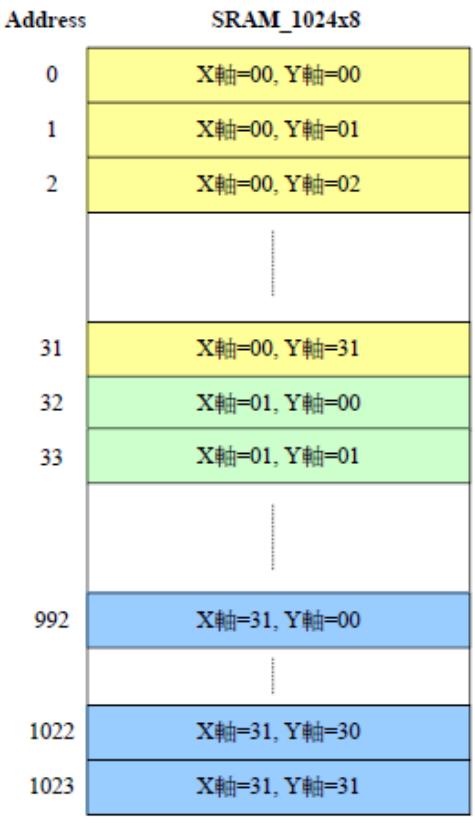
註 1：可用編號範圍：8'h01 ~ 8'hFB。
註 2：禁用編號範圍：8'hFC ~ 8'h FF。(這些編號已作為特殊用途，請勿使用!) 註 3：背景編號數值：8'h00。(前景物件請勿使用此編號)

C. 已使用過的編號不可重複再使用，因此不同物件的編號一定不相同。



圖七、編號後最後儲存在 SRAM 的數值

Y 軸



圖八、編號後的結果每個 Pixel 以 8bits 方式儲存於 SRAM

2.3.3 CLE 電路的輸出：在編號完後自行儲存於 SRAM 即可

依照 2.3.2 運算法則編號後，本範例最後編號完後結果如圖七所示，編號數值僅供參考，從這些編號可以看出，圖三之圖片在 CLE 電路運算後，因僅用了五個編號，因此可判斷出該圖片總共有五個物件，有相連到的 Pixels 確實編號都相同，沒有相連到的 Pixels 確實編號都不同，接著將圖七編號後的結果，包含 A. 背景訊號：編號固定為 8’h00，B. 前景各個物件編號，皆需儲存於 1024x8 的 SRAM 中，儲存方式如圖八所示，SRAM 位址 0，儲存圖七之【X 軸座標 00、Y 軸座標 00】的 Pixel 資料，由於可用編號範圍含背景為 8’h00 ~ 8’hFB，因此 SRAM 一個 Address 剛好存放一個 Pixel 編號後結果，剛好是 8bits，SRAM 位址 1，儲存圖七之【X 軸座標 00、Y 軸座標 01】的 Pixel 資料，…，依此類推。 另外，值得注意的一點是，CLE 電路中的 SRAM 模組，與 Host 中的 SRAM 模組所儲存的值一定相同。

當參賽者確定所有 Pixels 編號後的數值都已填在 SRAM 後，此時將”finish”訊號拉為 High，以告知 Host 端開始檢驗 SRAM 編號完結果，即完成 CLE 電路功能。

本電路模擬時，Host 端檢驗 SRAM 的內容值，正確與否的結果顯示方式如下：

A. 先將參賽者的 SRAM 編號後的資料，以 Log Message 顯示，其結果如圖九所示。

```
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_01_01_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_01_01_01_00_00_00_00_00_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_01_01_01_01_01_00_00_00_00_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_01_01_01_01_01_00_00_00_02_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_01_01_00_00_00_00_02_00_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_02_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_02_02_00_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_03_03_00_00_00_00_00_
00_00_00_00_00_04_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_03_00_00_03_03_03_00_00_00_
00_00_00_00_04_04_04_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_00_00_00_00_00_00_03_00_00_00_
00_00_00_00_00_04_04_04_00_00_00_00_00_00_00_00_00_00_00_00_03_03_00_00_00_00_00_00_03_03_00_00_
00_00_00_00_00_04_04_04_04_00_00_04_00_00_00_00_00_00_00_00_03_00_00_00_00_00_00_00_00_03_00_00_
00_00_04_00_04_04_04_04_04_04_04_04_04_00_00_00_00_00_00_00_03_00_00_00_00_00_00_00_00_03_00_00_
00_04_04_04_04_00_00_04_04_04_00_00_04_00_00_00_00_00_00_00_03_03_00_00_00_00_00_00_03_03_00_00_
00_04_04_04_04_00_00_00_04_00_00_00_00_00_00_00_00_00_00_00_00_03_00_00_00_00_00_03_03_00_00_00_
00_00_04_04_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_00_00_00_03_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_03_00_00_00_00_00_
00_00_00_00_00_00_00_08_08_00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_00_00_08_08_08_08_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_08_08_00_00_00_00_00_00_08_08_00_00_00_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_08_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_08_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_08_00_00_00_08_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_08_00_00_00_08_08_08_00_00_08_08_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_08_00_00_00_08_00_00_00_08_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_08_00_08_00_00_00_08_08_08_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_08_00_00_00_00_08_00_00_08_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_08_08_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
```

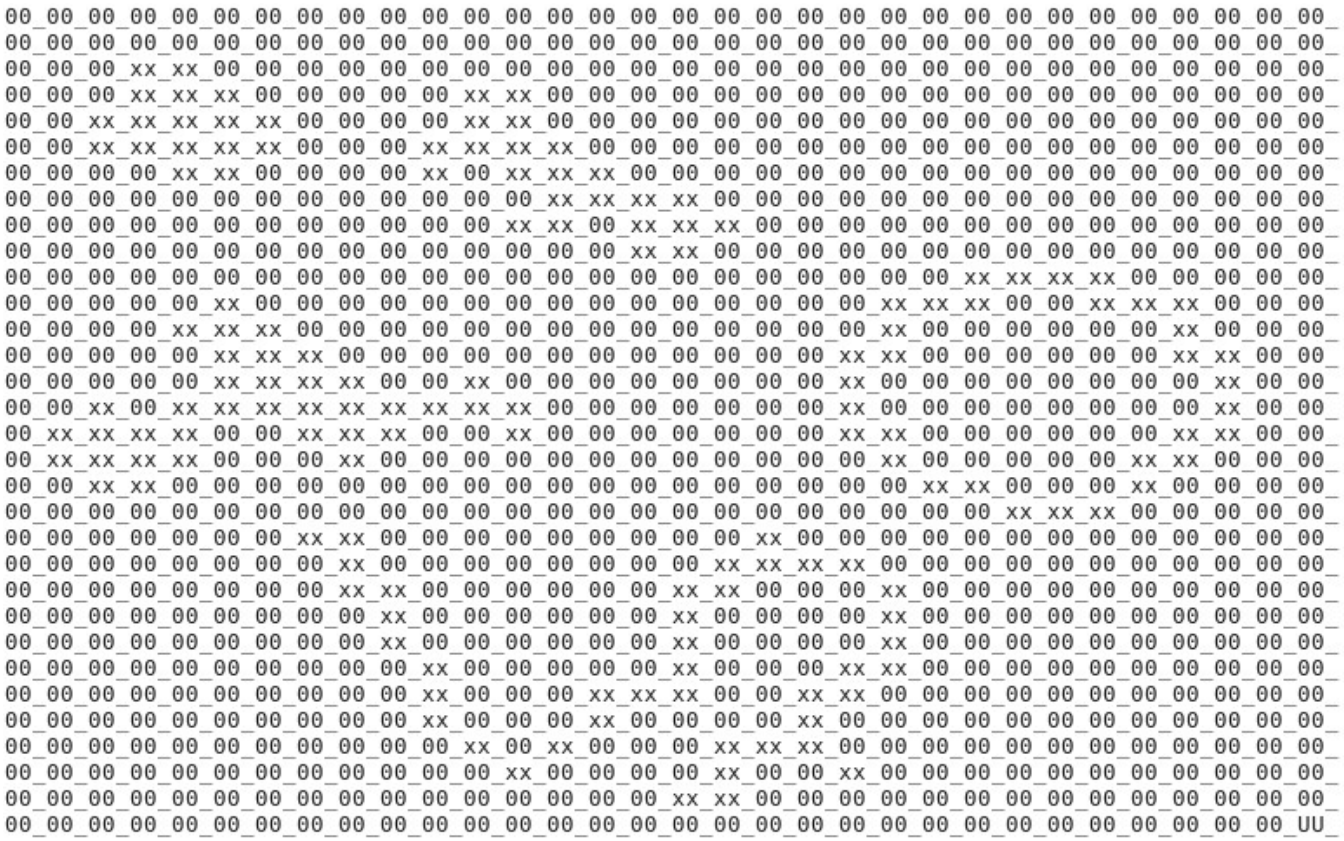
圖九、Function Simulation 時系統會將 SRAM 的資料以 Log Message 先秀出

註：圖九的數值，採 16 進制顯示，而每個 Pixel 有 8bits，因此每個 Pixel 會以 2 位數顯示。

表 2、Log Message 顯示之特殊符號解說

特殊符號	解說
XX	該 Pixel 是屬於背景訊號，但參賽者卻對此 Pixel 編號錯誤
UU	該 Pixel 是屬於背景訊號，但參賽者卻對此 Pixel 編號為 Unknow 訊號
xx	該 Pixel 是屬於前景物件，但參賽者卻對此 Pixel 編號錯誤
uu	該 Pixel 是屬於前景物件，但參賽者卻對此 Pixel 編號為 Unknow 訊號

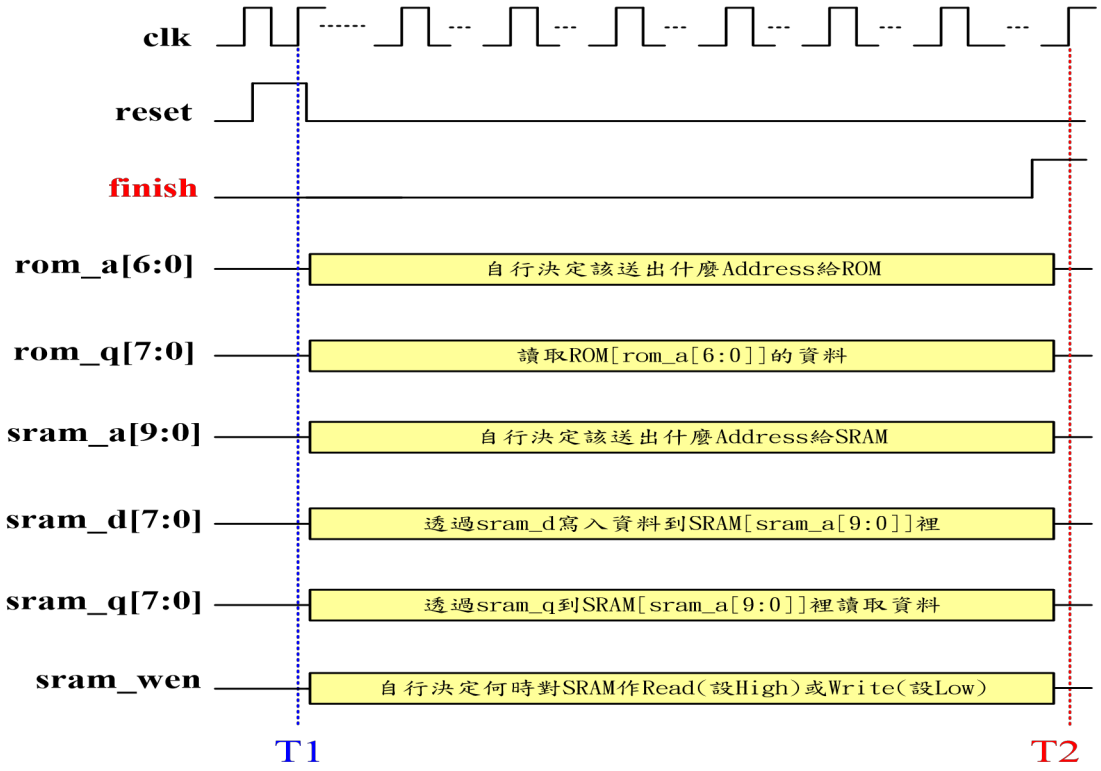
B. 再將參賽者的 SRAM 資料與標準解答作比對，比對結果如果正確，Log Message 的畫面會如圖九相同，但假若比對結果有不同處，會用以下特殊符號表示，如表 2 所示。假設某位參賽同學，對 SRAM 全部編號 8'h00，其顯示出訊息如圖十所示，結果顯示：背景所有 Pixels 全對，以 00 顯示，前景物件全錯以小寫 xx 顯示，最後一個 Pixel 是 Unknow，因隸屬於背景訊號區域，故 Log Message 以大寫 UU 來表示錯誤。



圖十、Function Simulation 時系統會將比對完的結果有錯誤處以特殊符號顯示

2.4 CLE 電路時序規格

2.4.1 CLE 電路之時序圖



圖十一、CLE 電路時序圖

1. CLE 電路初始化，Reset 一個 Cycle 的時間。
2. T1 時間點，reset 後可隨即送出 ROM Address 開始讀取二元影像訊號。二元影像訊號要讀取哪個位址、同一個 Pixel 要重覆讀取幾次與電路規畫有關，可自行決定。讀取二元影像訊號後，要開始怎麼針對物件作編號處理，SRAM 位址要怎麼送、怎麼讀、怎麼寫，也與電路規劃與自行想出的演算法有關，在此皆無硬性規定。
3. 當整張圖片之前景物件編號完成及背景訊號 8'h00 皆寫入 SRAM 後，T2 時間點將 finish 訊號拉為 High，即完成 CLE 電路運作！此時 Host 端會開始自動檢驗 SRAM 裡的資料其正確性，檢驗結果會以 Log Message 秀出。

註：SRAM 資料正確性之檢驗期間，不會花費到任何 simulation time。

2.4.2 ROM_128x8、SRAM_1024x8 之時序圖

有關 ROM_128x8、SRAM_1024x8 記憶體細項規格與記憶體讀寫時序圖，詳如記憶體附件中。

附錄

附錄 A 設計檔(For Verilog)

下表為主辦單位所提供各參賽者的設計檔

表 3、設計檔案說明

檔名	說明
CLE.v	參賽者所使用的設計檔，已包含系統輸/出入埠之宣告。
testfixture_a.v testfixture_b.v testfixture_c.v	本題共計三個 TestBench，每組測試皆為 32x32 pixels 之影像。
rom_128x8_a.v rom_128x8_b.v rom_128x8_c.v	CLE 電路會用到的三組 32x32 pixels 之影像，已擺在 ROM 檔案裡。當模擬軟體看到 rom_128x8_a.v 檔案時，會自動讀取影像資料 rom_128x8_verilog_a.rcf；同理，當模擬軟體看到 rom_128x8_b.v 檔案時，會自動讀取影像資料 rom_128x8_verilog_b.rcf，依此類推! 註：此三檔案已被 include 到 TestBench。
sram_a.dat sram_b.dat sram_c.dat	sram_a.dat 為圖片 a 的 Golden 結果檔案。 sram_b.dat 為圖片 b 的 Golden 結果檔案。 sram_c.dat 為圖片 c 的 Golden 結果檔案。註：此三檔案已被加入到 TestBench。
.synopsys_dc.setup	使用 Design Compiler 作合成或 IC Compiler Layout 之初始化設定檔。參賽者請依 Library 實際擺放位置，自行修改 Search Path 的設定。注意：無論合成或 **APR**，只需使用 **worst case library**。
CLE_DC.sdc	Design Compiler 作合成之 Constraint 檔案。參賽者可依需求自行修改 cycle 的設定。注意：環境相關參數請勿更改。
sram_1024x8_t13.v	1KB SRAM Verilog 模擬用之檔案。於 Memory/sram_1024x8_t13之資料夾當中
main.tcl	助教提供之合成基本指令集。
tsmc13.v	TSMC 0.13um 製程檔，僅提供同學進行 gatelevel 模擬用，請勿外傳於網路上。

請使用 ***CLE.v***，進行 CLE 電路之設計。其模組名稱、輸出/入埠宣告如下所示：

```
`timescale 1ns/10ps module  CLE ( clk, reset, rom_q, rom_a, sram_q, sram_a, sram_d,
sram_wen, finish);
input        clk;
input        reset;
input   [7:0]  rom_q;
output  [6:0]  rom_a;
input   [7:0]  sram_q;
output  [9:0]  sram_a;
output  [7:0]  sram_d;
output       sram_wen;
output       finish;

wire    [7:0] sram_q_internal;

sram_1024x8 u_sram(
                .Q       (sram_q_internal    ),
                .CLK     (clk       ),
                .CEN     (1'b0      ),
                .WEN     (sram_wen  ),
                .A       (sram_a    ),
                .D       (sram_d    )
               );

endmodule
```

1. 本題有三組 Pattern 要測試：testfixture_a.v、testfixture_b.v、testfixture_c.v，這些檔案會用到之相關檔案已設定完成，參賽者只要注意 rom_128x8_a.v、rom_128x8_b.v、rom_128x8_c.v rom_128x8_verilog_a.rcf、rom_128x8_verilog_b.rcf、rom_128x8_verilog_c.rcf、sram_a.dat、sram_b.dat、sram_c.dat 等九個檔案，有擺放在目前目錄即可！

2. 本題所提供的 Test Bench 檔案，有多增加幾行特別用途的敘述如下：

```
`define  SDFFILE    "./CLE_syn.sdf"
`ifdef  SDF
initial  $sdf_annotate(`SDFFILE, u_CLE);
`endif
```

註：

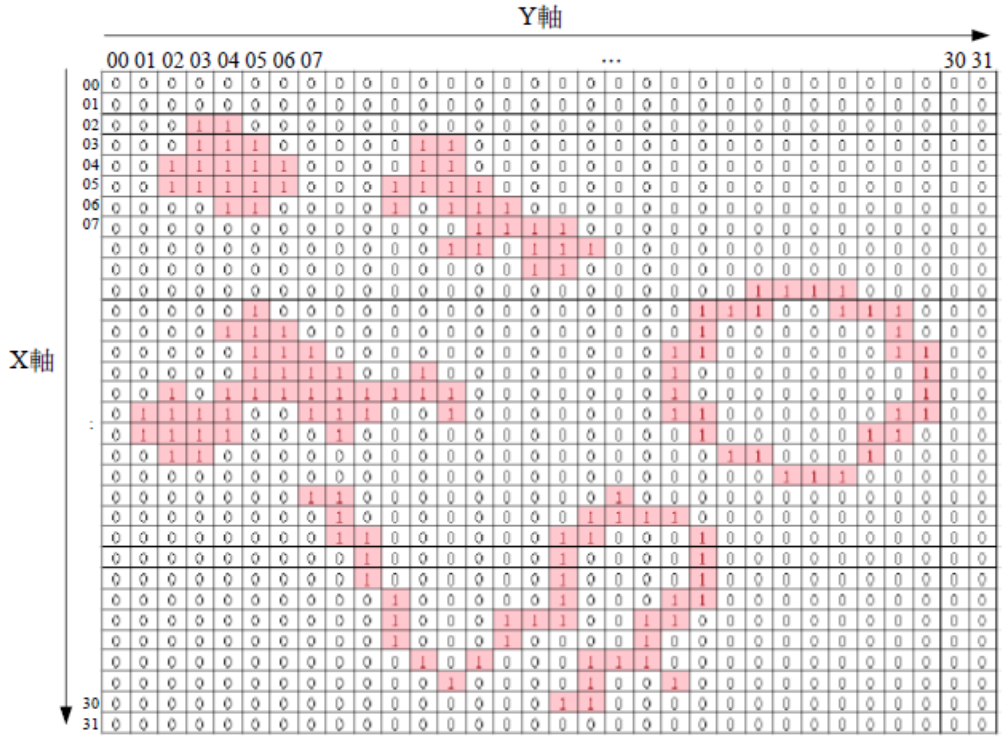
1. SDF 檔案，請自行修改 SDF 實際檔名及路徑後再模擬。
2. 在 Test Bench 中，主辦單位提供`ifdef SDF 的描述，其目的是讓本 Test Bench 可以作為 RTL 模擬、合成後模擬與 Layout 後模擬皆可使用。注意：當參賽者在合成或 Layout 後模擬，請務必多加一個參數"+define+SDF"，方可順利模擬。

例如：當合成後，使用 NC-Verilog 模擬第一組樣本，在 UNIX 下執行下面指令

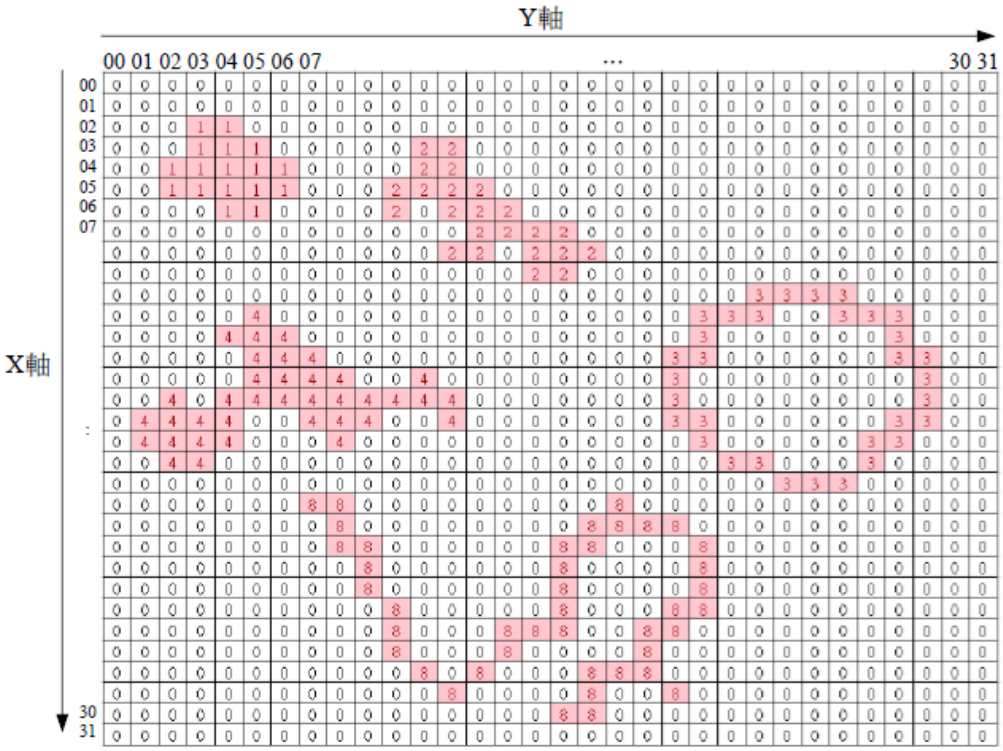
> ncverilog **+ncmaxdelays** testfixture_a.v CLE_syn.v –v tsmc13.v **+define+SDF** +access+rw

附錄 **B** 測試樣本

測試樣本 a: (32x32pixel 影像已存於 ROM，其儲存內容可參考 rom_a.dat，有詳細的座標說明)

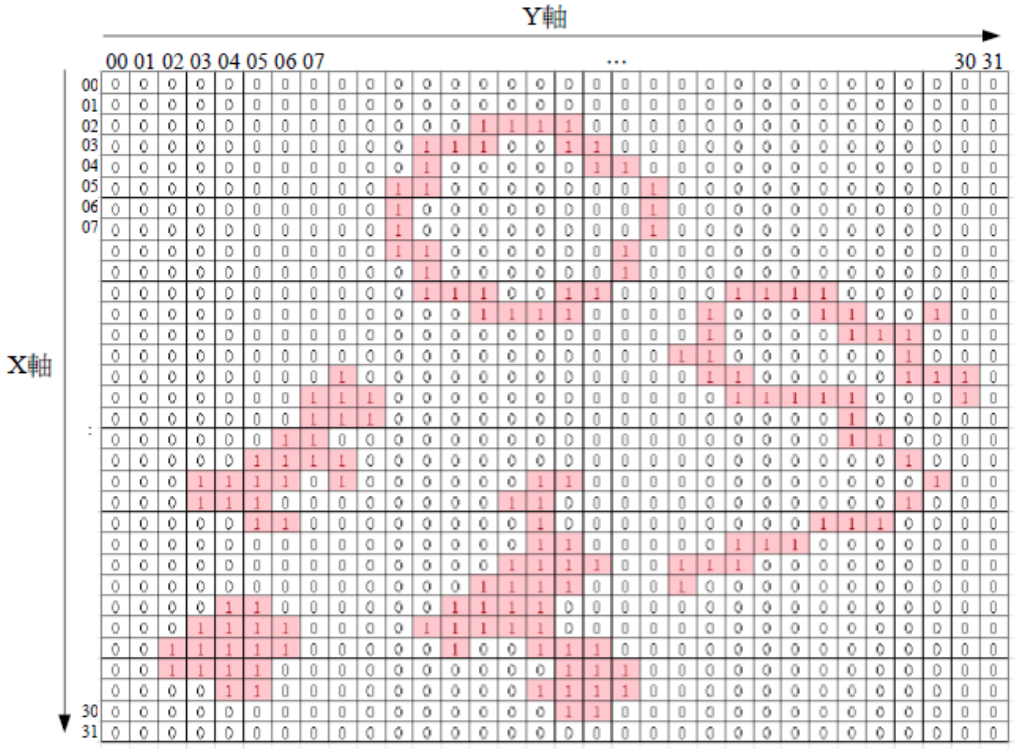


圖十二、測試樣本_a(已儲存於 rom_128x8_verilog_a.rcf)

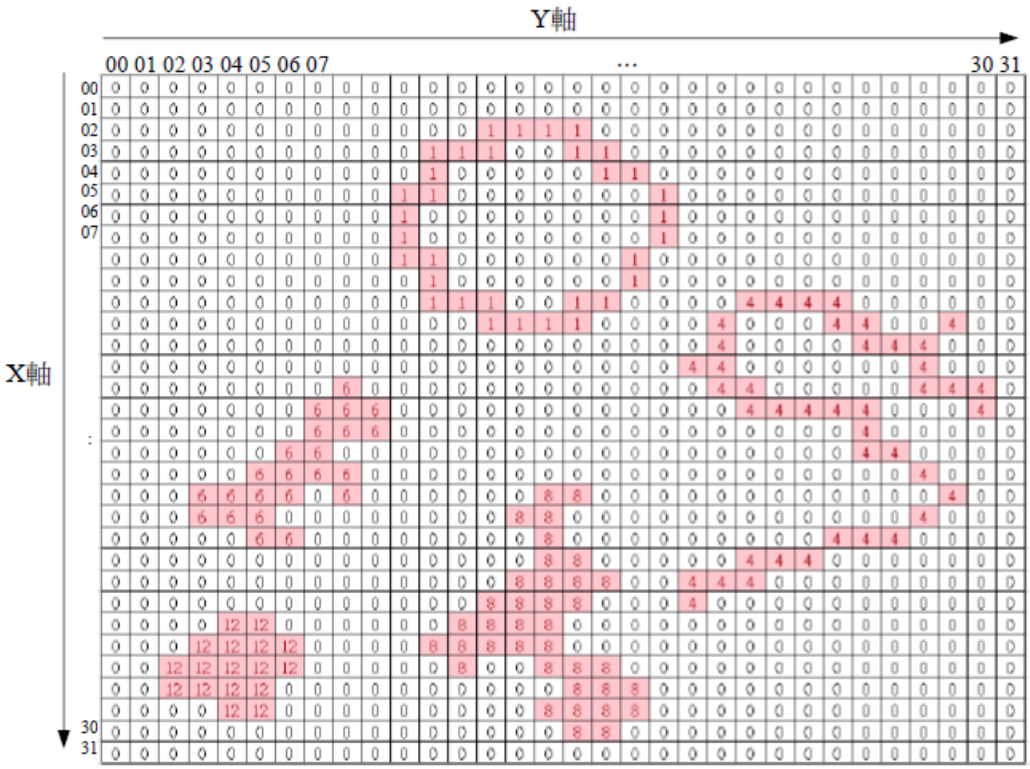


圖十三、測試樣本_a - 編號後的參考解答(已儲存於 sram_a.dat，編號值僅供參考)

測試樣本 b : (32x32pixel 影像已存於 ROM，其儲存內容可參考 rom_b.dat，有詳細的座標說明)

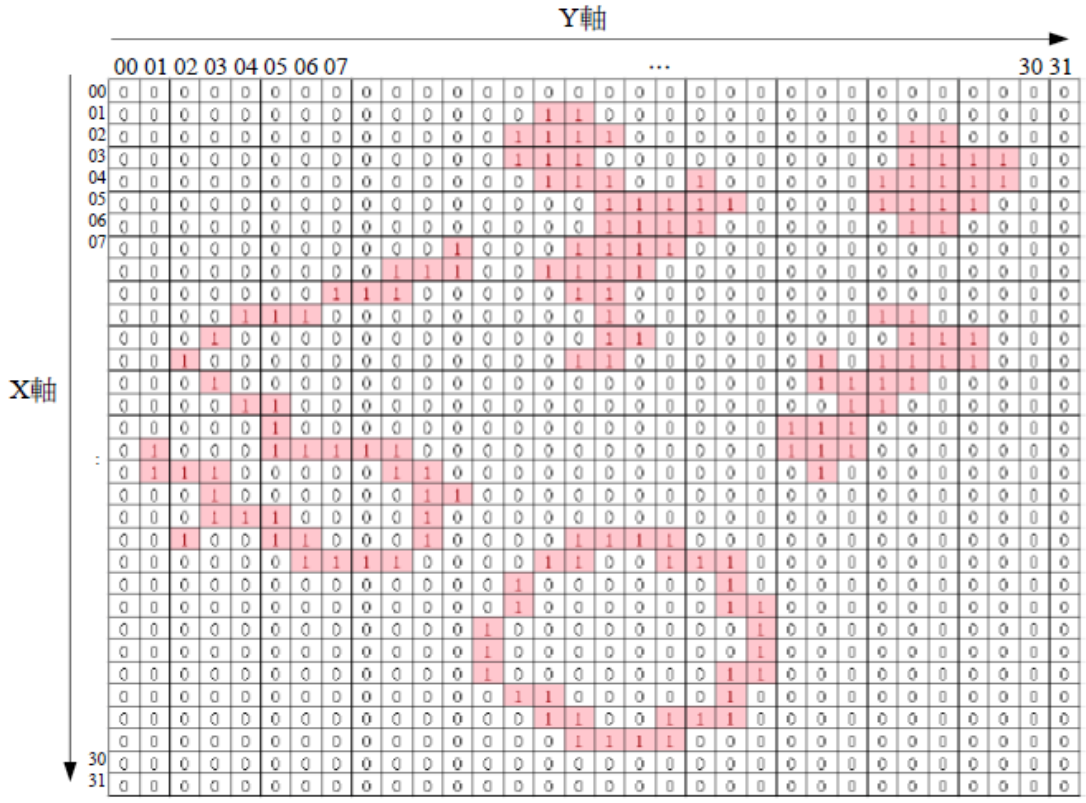


圖十四 、測試樣本_b(已儲存於 rom_128x8_verilog_b.rcf)

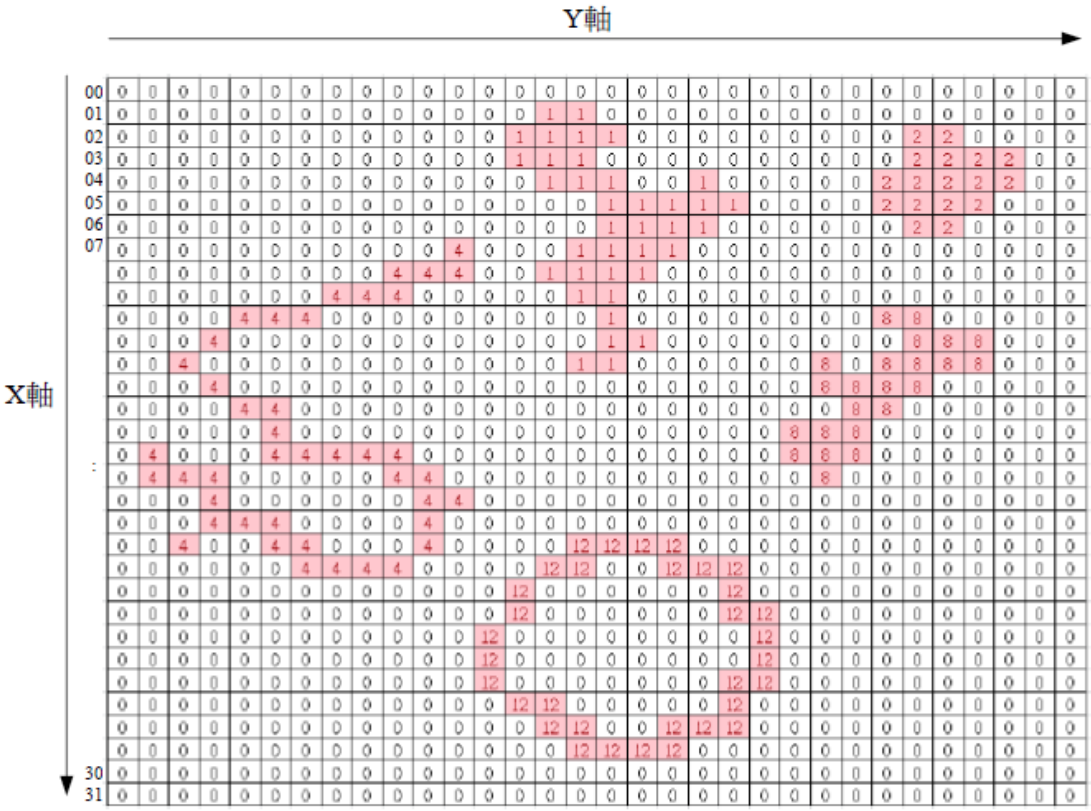


圖十五 、測試樣本_b - 編號後的參考解答(已儲存於 sram_b.dat，編號值僅供參考)

測試樣本 c: (32x32pixel 影像已存於 ROM，其儲存內容可參考 rom_c.dat，有詳細的座標說明)



圖十六、測試樣本_c(已儲存於 rom_128x8_verilog_c.rcf)



圖十七、測試樣本_c - 編號後的參考解答(已儲存於 sram_c.dat，編號值僅供參考)

附錄 C 繳交檔案

RTL Category		
Design Stage	File	Despcription
N/A	StudentID.pdf	Design Report Form
RTL Simulation	CLE.v	Verilog synthesizable RTL code
Gate-level Category		
Design Stage	File	Description
Pre-layout Gate-level Simulation	CLE_syn.v	Verilog gate-level netlist generated by Synopsys Design Compiler
	CLE_syn.sdf	Pre-layout gate-level sdf
	CLE_syn.ddc	Design database generated by Synopsys Design Compiler

Naming convention: **StudentID_HW3_vk.zip**
(k is number of version, k = 1,2,…)

FTP: 只允許上傳/ 無權限下載或刪除

Deadline	2018/11/6 中午12:00
IP	140.112.20.128
Port	1232
Account	CVSD_STUDENT
Password	cvsd2018

附錄 D 助教聯絡方式

陳奕達 edan@access.ee.ntu.edu.tw

有問題請寄信給助教

附錄 E 評分標準

(1) 依是否完成設計第一階段評分(60%)：

即完成下列三項要求。請按照 A、B、C 的要求順序完成。

✧ “完成設計”的三項要求:

A、 RTL通過作業提供的測試樣本A、B模擬。(20%)

B、 RTL通過作業提供的測試樣本C模擬。(20%)

C、 完成Synthesis，且CLE_syn.v的GateLevel Pre-layout Simulation有達到A、B項之要求。(20%)

<Note1> 請確認助教在 RTL 模擬上可以直接跑下列指令。

ncverilog testfixture_a.v CLE.v Memory/sram_1024x8_t13/sram_1024x8_t13.v –v

如果有額外的 module 在其它 .v 檔，請利用 `include 檔案在 CLE.v 之中，並且確認檔案之中，並且確認檔案命名。

在 gate-level 模擬，助教會用以下指令 模擬，助教會用以下指令：

ncverilog +ncmaxdelays testfixture_a.v CLE_syn.v –v tsmc13.v +define+SDF +access+rw

對於 gate-level simulation，請記得更改 SDF 檔案的名稱以及確認位置，與 testbench 中的變數 ”SDFFILE” 一致 。

<Note2>合成環境的設置合成環境的設置：請根據 LAB 中 Synthesis 部分所提供的 “.synopsys_dc.setup”設置 。

target_library	slow.db fast.db Memory/sram_1024x8_t13/sram_1024x8_t13_slow_syn.db
link_library	* $target_library dw_foundation.sldb
symbol_library	generic.sdb
synthetic_library	dw_foundation.sldb

<Note3> 關於合成 sdc constraint 除 clock cycle 之外不可更動。

(2) 依設計的 依設計的 Cost 做第二階評分 (40%)：

助教會將符合第一階段完成設計(三項要求都完成)者，依據下式作第二階段的評分其中 Cost 越低成績越佳。

Cost = Area(um^2) * Total simulation time^2(ns)

1. Area = synthesis report 中的 Total Cell Area (不含 wire load model 面積)
2. Total simulation time = clock period*total clock cycle (ncverilog 總模擬時間)
3. 請將 design compiler 關於 Area 以及 ncverilog 模擬截圖附在報告中以證明，否則此部分無法獲得分數。

A+ Class: ~5% (40 points)
A Class: 6%~10% (35 points)
B Class: 11%~25% (30 points)
C Class: 26%~70% (20 points)
D Class: 71%~100% (10 points)
F Class: 在 gate-level 模擬時有錯誤 (0 points)

- ✧ 注意，因助教測資對於 Stage Analysis 之精準度考量請不要 truncate 此部分的運算。
- ✧ **在此次作業中助教會嚴厲抓抄襲，實際看.v 檔以及合成時的.ddc 檔(工作檔，可知道 contraint 等)，並且檢查報告內容是否與程式相符以確認公平性。任何作弊行為將會導致無法通過這門課。**

(3) 作業報告:

1. 附上數據表:

Area(um^2)	
Clock cycle(ns)	
Total simulation time(ns)	
Cost(um^2*ns)	

2.

Area (total cell area) in Design compiler 截圖

RTL 以及 gate-level ncverilog simulation 截圖

3.

架構實現方法