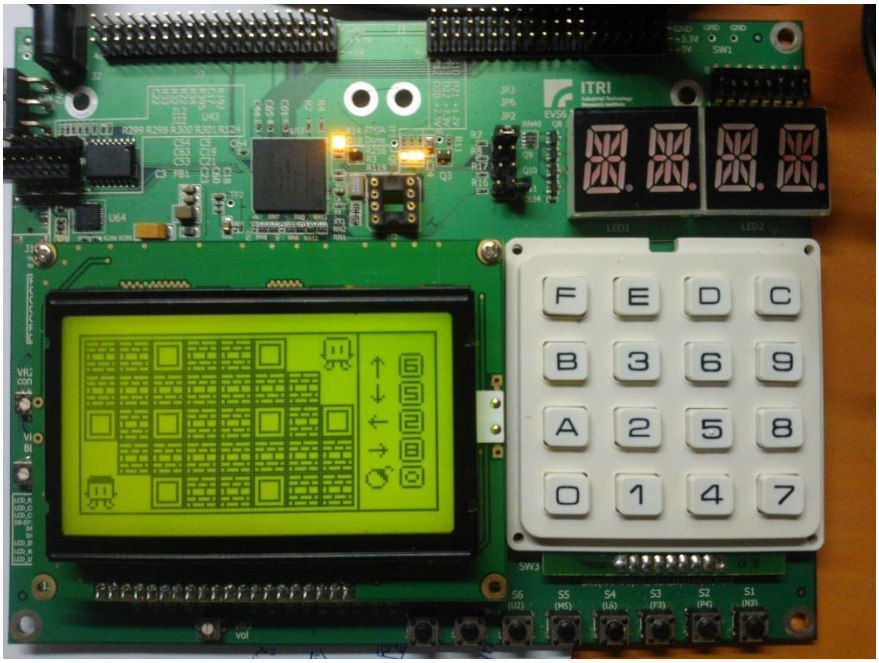
Hardware Lab

Final Project Report

100062238 張凱智
100062241 李青峰

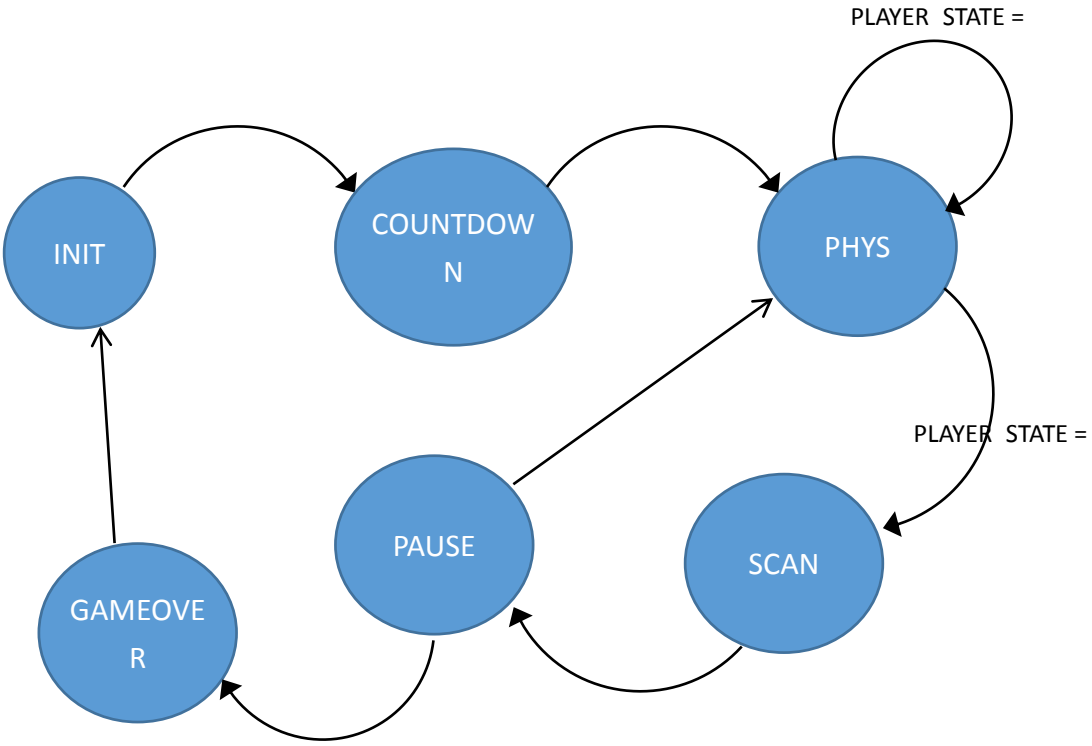
一、 Specification

這是一個 two player 的遊戲，所以我們使用了兩塊 FPGA，並且用一組六條的傳輸線連接。兩台連結後並開機，其開機的先後順序不影響正確性。一開畫面的右方會有操作提示 ”Press 0 to Start”，在這種狀況下按鍵盤的上下左右鍵是不會有任何反應的，按下 0 之後會進入倒數畫面，從 3 倒數到 1 後顯示 ” BATTLE !! ”便宣告遊戲即將開始。遊戲開始後，操作方式會顯示在畫面右方，此時上下左右鍵和 0 便開始有反應。Player 1 的初始位置在畫面右上方而 Player2 則在畫面左下方，阻隔在玩家間的地圖是由不可破壞的石牆 Stone 和可破壞的磚牆 Wall 組成的，玩家可藉由按下 0 放置炸彈，並炸毀 Wall 以製造前進的路。炸彈的火力是除非遇到障礙物，像是 Wall、Stone 和炸彈，否則炸彈的火力會炸到底。而當火力延伸的路徑上遇到另一顆未爆彈時，火力會被阻擋，但也同時會引爆阻礙的炸彈。一次可放置的炸彈顆數不限，以便玩家發揮自己的戰術擊敗對方玩家。當某一方玩家被炸彈炸死時，遊戲便會結束，並宣告被炸死的一方輸了，活下來的一方獲勝。此時按下 0 便可以回到初始畫面，等待下次遊戲的進行。



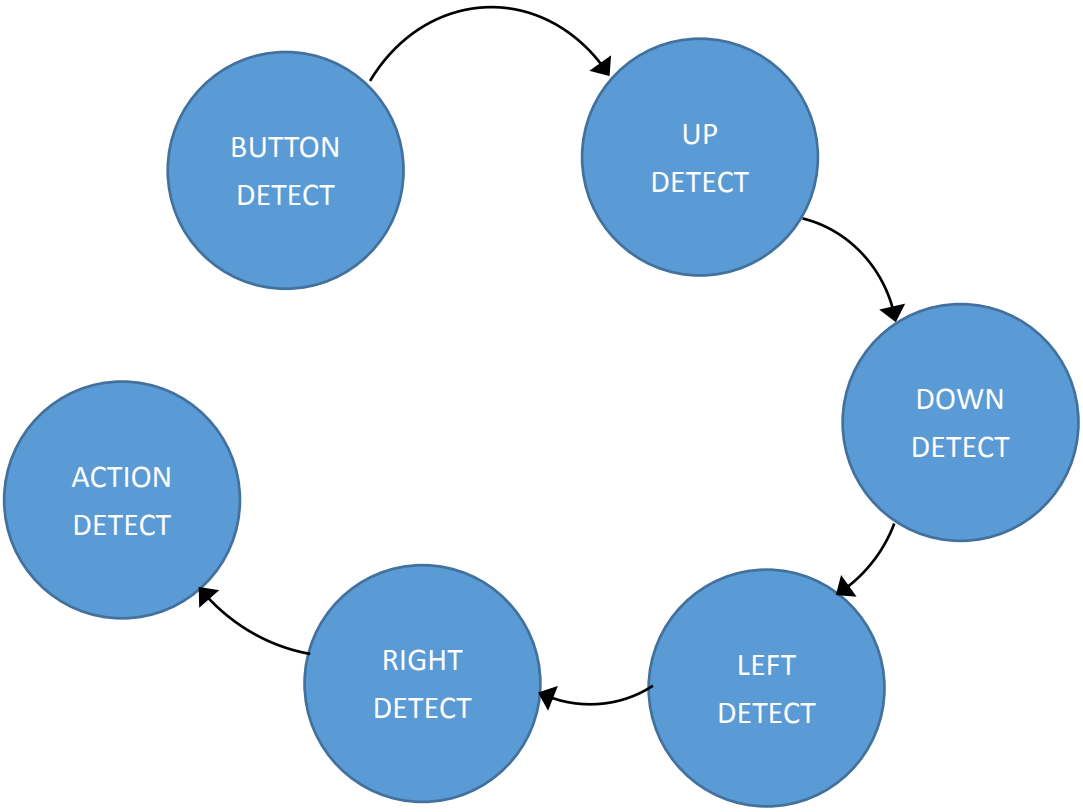
二、 Implementation Details

Bomber_Ctrl：



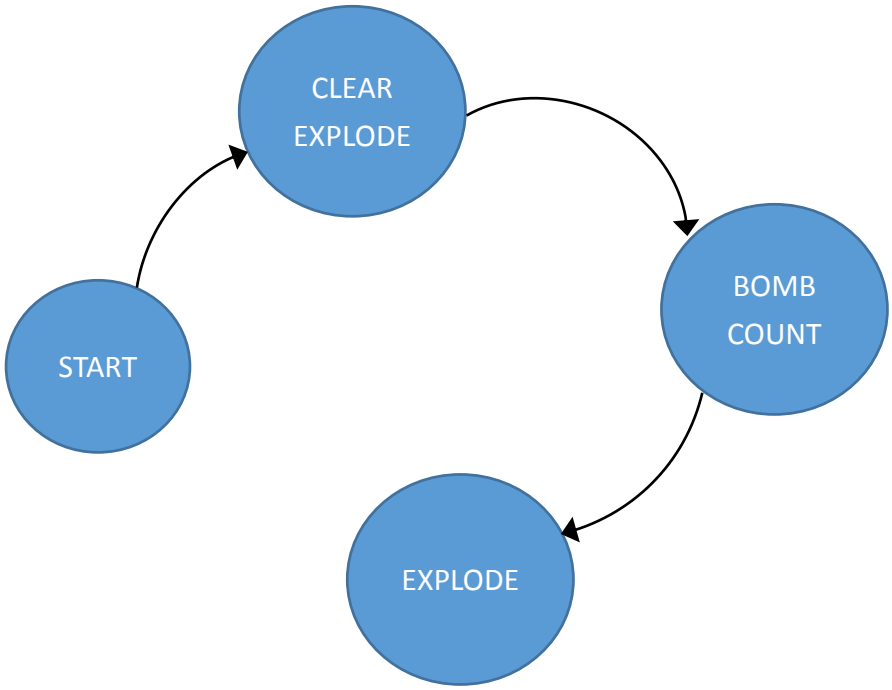
程式一開始在 INIT 裡初始化地圖以及兩位玩家的初始位置設定，當按下 0 時就會進去 COUNTDOWN 裡面，開始就入遊戲前的倒數，當倒數完就進入 PHYS，這裡是用來做角色的移動和碰撞偵測，先將 player 1 偵測一次後再換 player 2 兩位玩家都偵測完以後就會進入 SCAN，這裡再做的事情就是掃描炸彈，並做炸彈倒數，當倒數就讓炸彈爆炸，所以這裡也同時偵測玩家有無被炸彈炸到，若沒有被炸彈炸到的話就進入 PAUSE 停一下然後再回到 PHYS 如此巡迴，但若被炸彈炸到就會就入 GAMEOVER 的 STATE，宣告輸贏畫面，並且當按下 0 時能在回到 INIT 準備下一場遊戲。Bomber_Ctrl 的最下面會持續將遊戲資訊存成一個 16 bit 的 instruction 並且傳出去給 LCD_control。

PHYS：



PHYS 一開始在 button detect 的 state，在這裡會偵測玩家的上下左右鍵有沒有按下去，再來就會分別進入上下左右的偵測，主要是用來作移動和撞牆偵測，以免玩家穿越牆壁會邊界的情形發生。在 PHYS 裡，當偵測到往某一個方向走時，玩家會從一個 12X12 大小的格子移動到另一個格子，但並不會是一次移動到，看起來會像是慢慢走過去。最後的 action detect 是用來偵測有無放炸彈，並且在正確的位置放下炸彈。當 player 1 經過一輪 detect 後就換 player 2，兩位玩家都輪過一變後就會在 action detect 的地方進入 SCAN。

SCAN：



SCAN 要做的事情就是去偵測整張地圖上的炸彈，再 START 的時候會先初始會偵測用的 index 然後再進入 clear explode。在這個 stage 裡會去偵測地圖上所有的爆炸火花，並且將其清掉變回一無所有的 AIR 也就是可行走的道路。全檢查完後會進入 bomb count。這裡會偵測所有的炸彈，並將炸彈上的 timer 減一，全部偵測完後就會進入最後的 stage explode。這裡會偵測所有 timer 歸零的炸彈並讓它爆炸，爆炸的火花由四個方位擴散，依上下左右的順序檢察看火花擴散的方向有無石牆、磚牆，有則停止火花，若遭遇玩家則該玩家死亡，偵測完後就進入 PAUSE。在此若先前有玩家死亡則進入 GAMEOVER 結束遊戲，宣告輸贏，並且按下 0 可回到 INIT，若無人死亡則回到 PHYS 如此循環。

Instruction Table：

Item Type	**16-bit**						
player	1’b0	X position 3 bit	X sub pos 3 bit	Y position 3bit	Y sub pos 3 bit	Player 1 bit	Face direct 2 bit
object	1’b1	2’b00	X position 3 bit	Y position 3bit	What object 4 bit	Don’t care 3-bit	

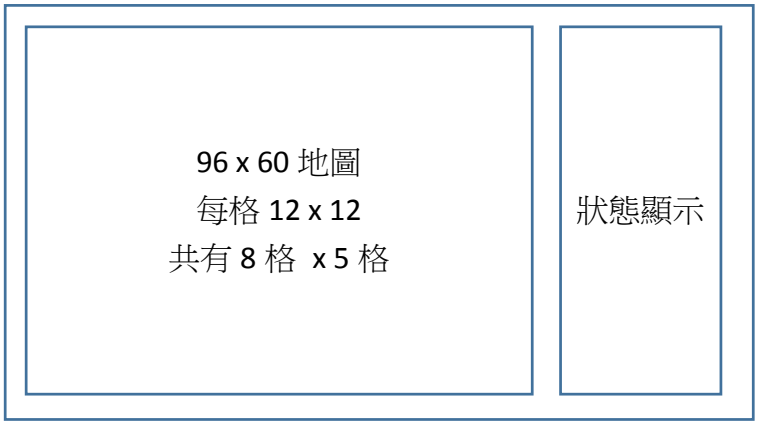
Game State	**16-bit**						
init	1’b1	2’b10	2’b00	Don’t care 11-bit			
Count down	1’b1	2’b10	2’b01	Timer 2 bit	Don’t care 9-bit		
play	1’b1	2’b10	2’b10	Don’t care 11-bit			
Game over	1’b1	2’b10	2’b11	Don’t care 2-bit	Player 1 status 1 bit	Player 2 Status 1bit	Don’t care 7-bit

Lcd_control 就是藉由上表的 instruction 來畫出所需的東西。

Lcd_Control：

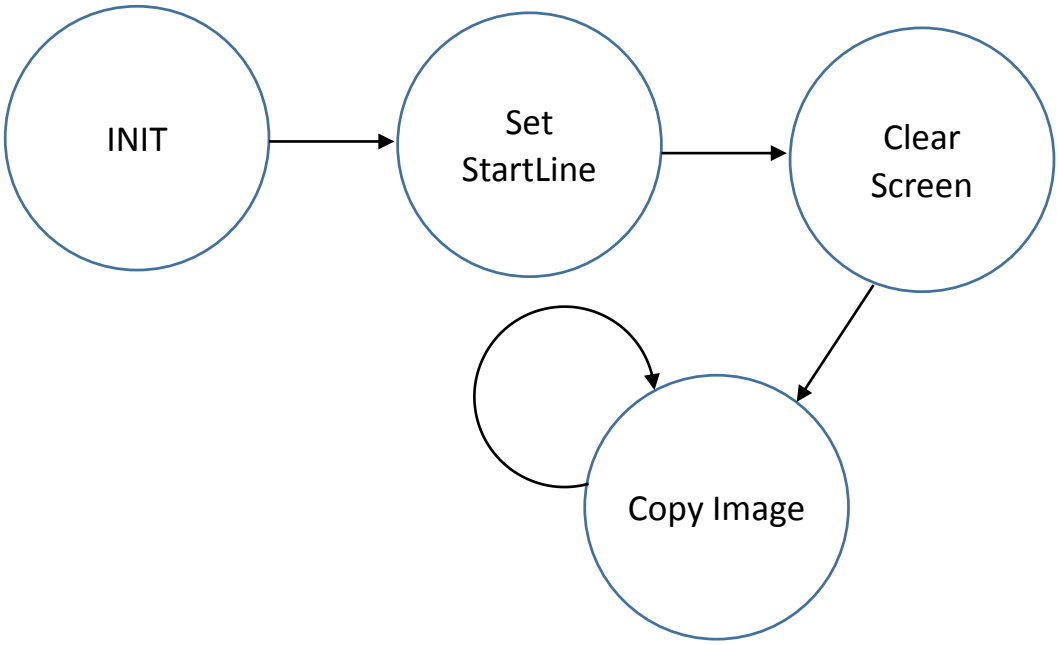
LCD_Control 主要負責將來自 Bomber_Contrl 的指令即時轉換為螢幕上的圖形。主要分為三個部分：

- 依照指令重建出 Bomber_Contrl 內部的地圖、狀態等 (LINK_CLK 驅動)
  此部分接受外部輸入的指令，並維護一組顯示所需的完整遊戲狀態，以提供內部其他電路對其狀態的讀取。



- 依照地圖及遊戲狀態轉換成圖形的 combinational circuit
  此部分儲存了所有需要的遊戲圖形元素，並接受 X_PAGE, INDEX, 以及地圖及遊戲的狀態輸入，再依各類元素先後順序進行疊圖處理，以輸出對應的(疊圖後的)PATTERN(8-bit)。

- 與 LCD 互動，使得 LCD 可以顯示出想要的圖形 (100KHz 驅動)



INIT: 啟動 LCD

Set_StartLine: 將 StartLine 設為 0。

Clear_Screen: 將 LCD 的螢幕全部設為白色(0)。

CopyImage: 將 PATTERN 依次複製到螢幕上。

其中

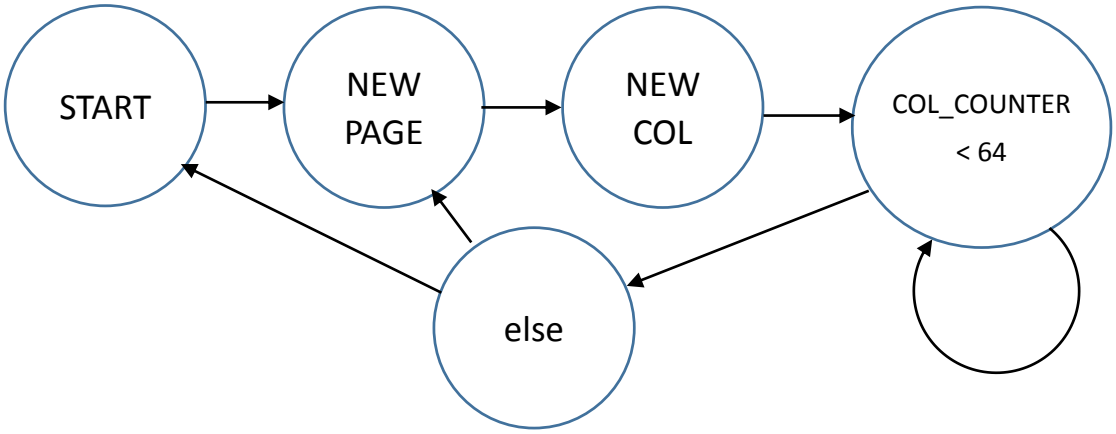
START: 開啟左半邊螢幕，初始化必要值。

NEW_PAGE: 設定 PAGE = X_PAGE

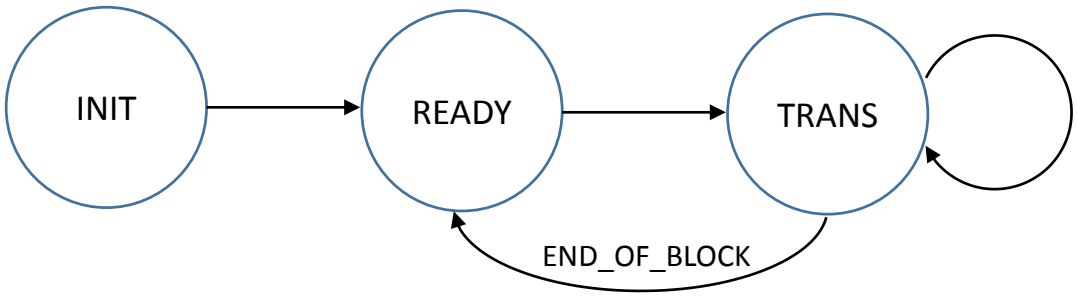
NEW_COL: 設定 COLUMN = 0

COL_COUNTER < 64: 依次複製該 PAGE 的資料

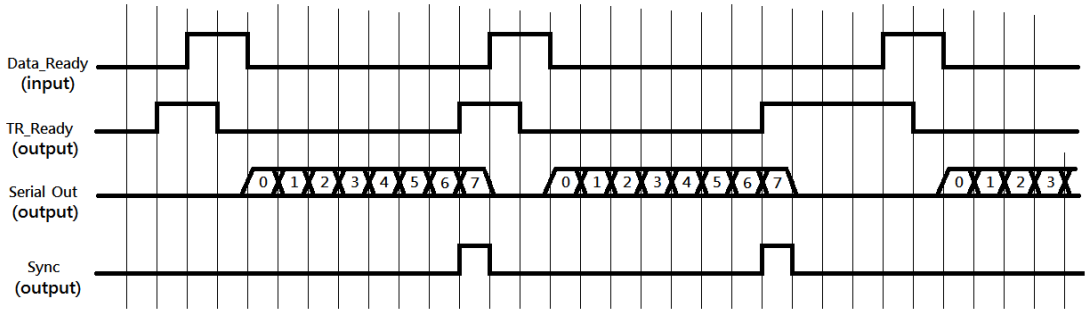
else: 如果現在開啟的是左螢幕，切換至右螢幕，並轉換狀態至 NEW_PAGE；如果現在開啟的是右螢幕但尚未到達最後一個 PAGE，切換至下一個 PAGE，並轉換狀態至 NEW_PAGE；如果現在開啟的是右螢幕且到達最後一個 PAGE，直接轉換狀態至 START。



transmitter：



Timing Diagram:



Transmitter 主要功能是負責兩塊 FPGA 版之間的傳輸，也就是將原本的一些內部平行訊號，轉換成 Serial Data 在線路上傳送。傳送的過程需要兩條傳輸訊號，一條是 Serial Out 也就是資料流；另一條是 Sync 也就是同步訊號，會在 transmitter 傳送到一個 BLOCK 的結尾時變為 1，避免傳輸過程的 clock signal loss 造成接收端資料還原錯誤的情形。

其中 transmitter 在可以接收新資料時，TR_READY = 1。而想要傳輸資料的一方則可以在準備好資料以後，設定 Data_Ready = 1，隨後在 transmitter 開始傳送時 TR_READY 會回到 0。

在這次的實作中，master(主機)上的 transmitter 為 49-bit 的 Block size，用來傳送 LCD 顯示的 instruction；slave(副機)上的 transmitter 為 16-bit 的 Block size，用來傳送鍵盤訊號至主機端(master)處理。

為了避免錯誤，所有發送的資料都有 Forward Error Correction 編碼，編碼方式為：{DATA, ~DATA, DATA, (~^DATA)}，最後一碼作為 parity bit，可以在接收並修正資料後再次檢查其正確性。

receiver：

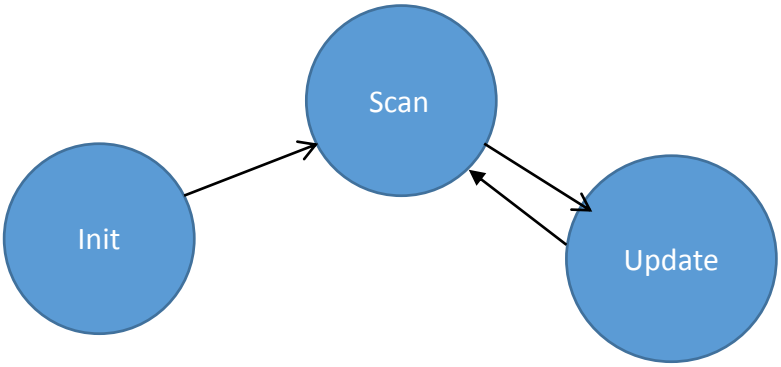
Receiver 主要功能是負責接收來自 Transmitter 的傳輸訊號，並正確將其還原。它的結構較簡單，內部使用 Shift Register 將收到的資料依序暫存，只有在接收到 Sync 訊號時才將資料 update 到 Output Port。而 Receiver 的輸出除了 DATA_BUS 外，還有一條 RECV_OK 用來表示新資料的輸入，每當新資料接收完成後，就會變成 1 並維持一個 LINK_CLK cycle。

經過外部線路收到的資料難免會有錯誤，因此我們在 top.v 中實作了 Error Correction 的檢查(傳輸時冗於資料已預先加了進去)，共有以下幾種 cases：

- 三段資料相同，解碼成功。
- 其中兩段資料相同，並比較 Parity bit 確認，將資料修復並解碼成功。
- 其中兩段資料相同，並比較 Parity bit 發現錯誤，資料嚴重錯誤，解碼失敗。
- 沒有任何兩段相同的資料，資料嚴重錯誤，解碼失敗。

若解碼失敗，該筆資料會被直接丟棄，不會被接受。即便如此，由於本實作中前後 instruction 無相關性，因此即使發生一些小錯誤仍難易察覺，不會造成運作上的嚴重錯誤。

Keypad scanner：



我們使用 lab 時使用過的 keypad scanner 來做修改，因為我們只需要 5 個按鍵。於 init 的時候就會將上下左右跟放炸彈設成 0，也就是沒有按，然後就進入下一個 state。再 scan 裡面我們就掃描每一 row alternatively，當掃描到 3rd row 以後就換到下一個 state。最後再 update 裡面我們就去更新按下去的鍵，並且判斷上下左右和放炸彈有沒有按下去。然後就再回到 scan 重來一次，如此循環。

三、 **Problems**：

1. 由於我們要連線，但卻發現用連接線傳遞訊號時，有可能因為訊號線被干擾而導致資料傳遞錯誤。(**解決：**用Foward Error Correction編碼)
2. 不知道遊戲引擎要如何跟LCD做溝通。(**解決：**使用描述狀態的Instruction)
3. 移動碰撞偵測 還有 炸彈火花擴散的code複雜度太高，所以燒不進FPGA裡。(**解決：**將硬體中重複的動作找出來，分配至不同的clock執行，避免相同的硬體被複製太多份，並且調整Design Goals & Strategies設定，使占用晶片面積盡量減少，並且一些不是非常必要的功能如道具、計時計分則被完全刪去)

四、 **Any Suggestion for this class**

我覺得希望在課堂上能再提供多一點 FPGA 和 LCD 的資訊，像這次我們不知道原來能使用的電晶體數量是有限的，而發生了

There were not enough sites to place all selected components.

This design does not fit into the number of slices available in this device due to the complexity of the design and/or constraints.

這樣的問題，導致我們 program 不進 FPGA 裡，我們的 code 確定是對的，但必須修改到複雜度沒有那麼高才燒得進去，因為耗費了我們很多多餘的時間，所以想說以後可能在課堂上可能要提醒大家說怎樣避免這種情況，因為不是所有人都有辦法像我們這樣把自己打好的 code 全部改掉的。