**Hardware Lab**

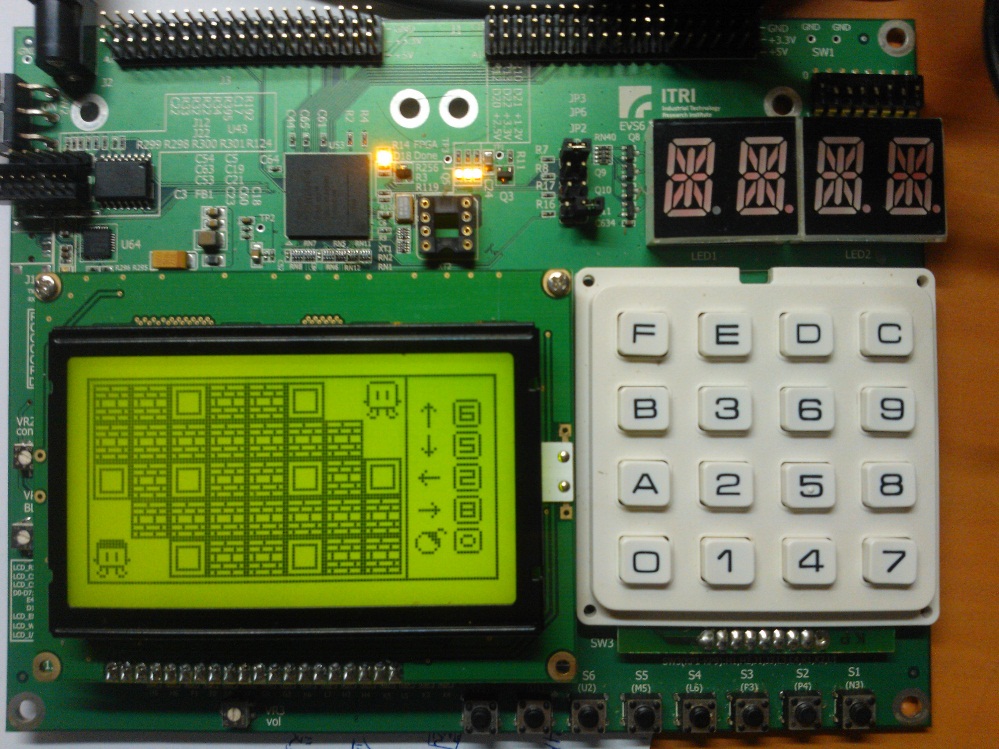
**Final Project Report**

100062238 張凱智

100062241 李青峰

1. **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。



1. **Implementation Details**

**Bomber\_Ctrl**：

PLAYER\_STATE = 0

PLAYER\_STATE = 1

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。

狀態顯示

96 x 60地圖  
每格12 x 12

共有8格 x 5格

* 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

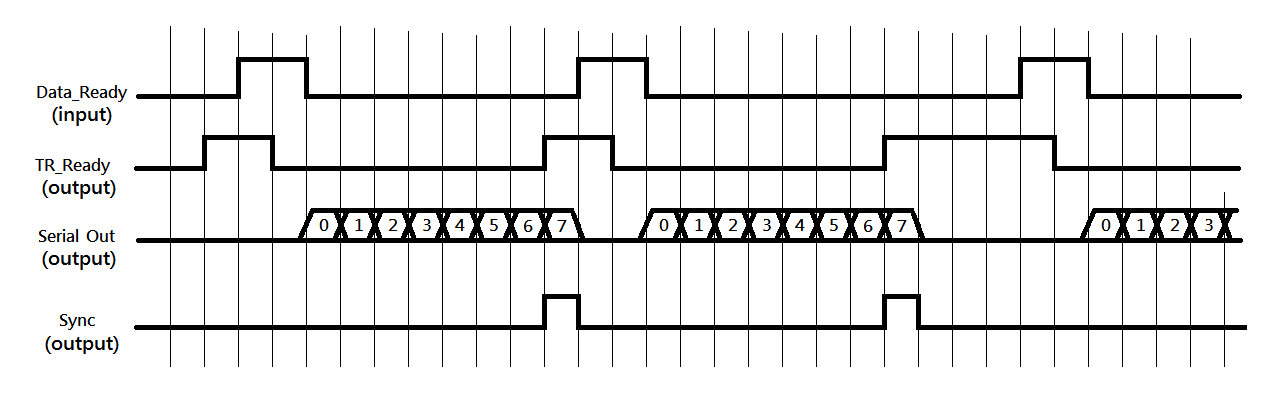
NEW\_COL: 設定COLUMN = 0

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：**

END\_OF\_BLOCK

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。

1. **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)

1. **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。