Hardware Design and Lab

Final Project

Slime Jump

Team 28

111060013 EECS 26' 劉祐廷

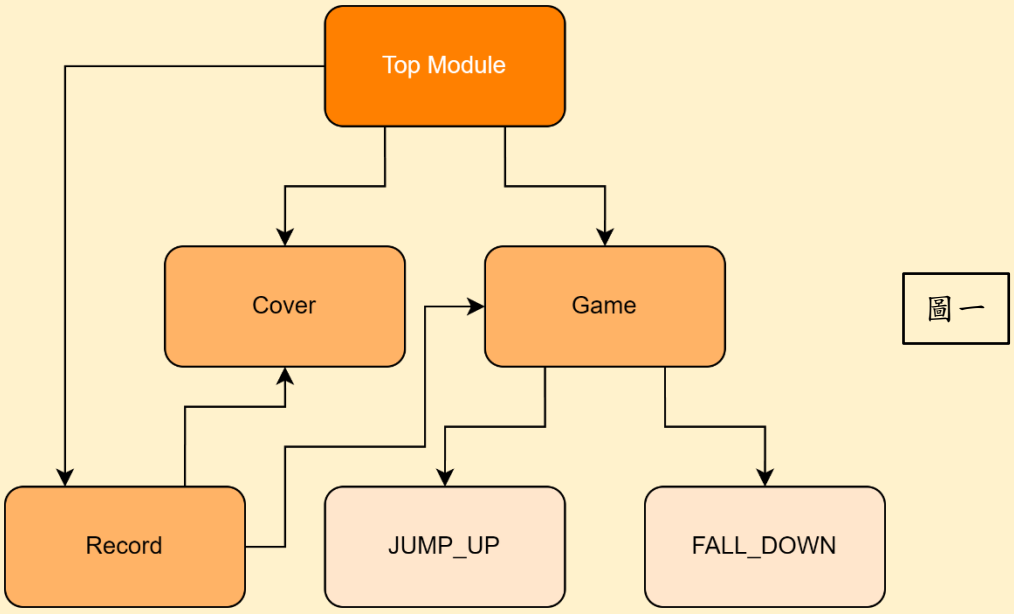
111060002 EECS 26' 李侑霖

Catalog

1. 遊戲介紹

此遊戲靈感來自於手遊《Doodle Jump》，遊戲主角為 Slime，玩家須透過鍵盤操控 Slime 左右移動，讓 Slime 能踏著 floor 慢慢往上跳，直到分數到達 100 分。floor 位置是偽隨機生成的，並且跳每跳 5 次 floor 可以獲得一次二段跳的機會，由空白鍵觸發且不可疊加。遊戲畫面每下移一次即獲得 1 分，遊戲會記錄上一局的分數以及每場遊戲的最高分數，並顯示在 cover 畫面。

2. 遊戲架構



圖一

首先我們先將遊戲分為 Cover 與 Game 兩種不同的狀態，分別對應到目前應該顯示起始畫面或是目前應該顯示遊戲畫面。在遊戲畫面(Game)的狀態下，依照遊戲主角 Slime 當前的狀態分為 JUMP_UP 與 FALL_DOWN 兩種狀態，分別代表 Slime 目前處於向上跳躍或是向下墜落的狀態。

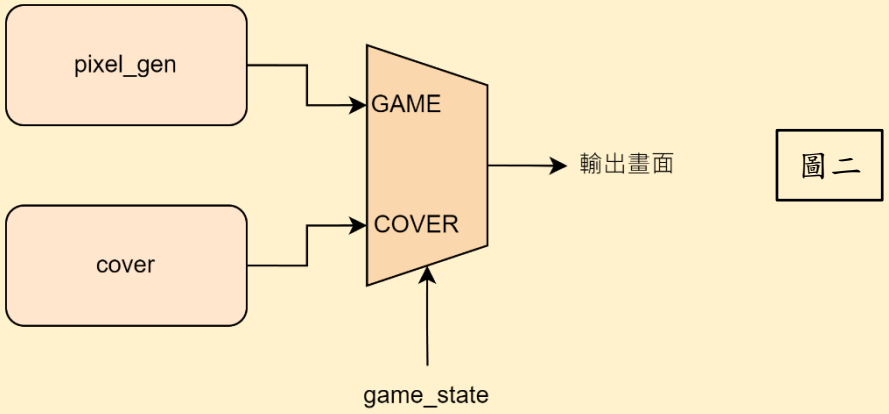
3. Module

module	功能
Top	負責場景的選擇(Game 或是 Cover)與鍵盤輸入的判定，並彙整所有 module 處理連接到終端設備上的輸入輸出。
cover	負責生成遊戲開始畫面(Cover)，將畫面各個 pixel 該顯示的資訊透過 Top module 接線至 vga_controller。
pixel_gen	負責生成遊戲畫面(Game)，處理從 slime_move module 與 floor module 傳進來的遊戲資訊，轉換成由遊戲畫面各個 pixel 的資訊，透過 Top module 接線至 vga_controller。
slime_move	負責遊戲各種功能的實現，包括 Slime 的狀態、位置，遊戲分數，Slime 與 floor 碰撞判定，將遊戲資訊處理好，透過 Top module 接線至 pixel_gen module 生成畫面。

floor	負責生成 floor 的位置，透過 Top module 接線至 slime_move module 與 pixel_gen module，已進行碰撞判定與生成 floor 的畫面。
record	用來記錄上一局遊戲分數與最高分數，在畫面的狀態從 Game 切到 Cover 時會去抓取遊戲內的分數，並判斷是否需要修改最高分數。
clk_div	負責生成 vga_controller module 需要使用的 clock。
clk_vga	負責生成 slime_move module 需要使用的 clock。
clk_floor	負責生成 floor module 需要使用的 clock。
vga_controller	將處理好的 pixel 轉換成 VGA 的訊號輸出。
KeyboardDecoder	解析鍵盤的輸入訊號。
audio	負責生成 Slime 彈跳的音效。

4. 遊戲功能

● 遊戲開始畫面與遊戲畫面的切換

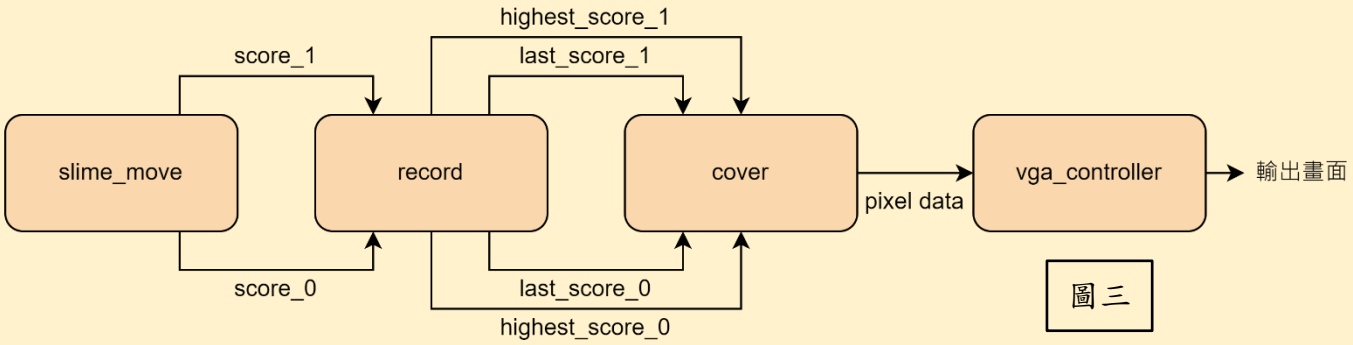


圖二

我們是透過用 mux 去選擇輸出畫面(如圖二)，來達成遊戲場景切換的功能，在遊戲開始畫面時按下 Enter，game_state 會被設成 GAME，進入遊戲畫面，而當 Slime 死亡後，game_state 會再被切回 COVER，回到遊戲開始畫面。在測試的時候有遇到在遊戲開始畫面時碰觸 Key A 與 Key D 後，上局分數被改變的 bug，後來發現原因是因為以我們的設計來說，遊戲畫面與起始畫面是同時運行的，因此在遊戲開始畫面時也能操控 Slime 移動，而去動到上一局的分數，後來我們夠過在鍵盤判定的地方多加了一個以 game_state 當條件的 mux，使 Key A 與 Key D 在遊戲開始畫面時失效以解決這個問題。

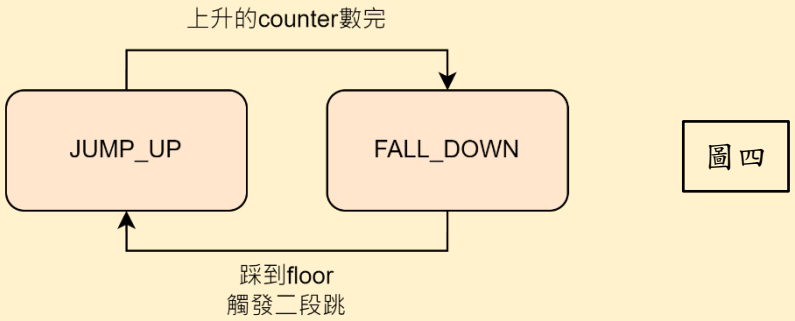
● 紀錄上局分數與最高分數

上局分數與最高分數是由 record module 在 game_state 從 GAME 切換到 COVER 時，去抓從 slime_move 接出來的分數(score_1, score_0)，經過比較後處理成 highest_score_1, highest_score_0, last_score_1, last_score_0 並接入 cover module，編號 1 的 wire 代表十位數，編號 0 的代表百位數，經由 decoder 將這四個數字轉換成對應的 pixel 資訊，再經由 vga_controller 將畫面輸出。(參考圖三)



圖三

- **Slime 跳躍與落下**
  - **定義 State**



圖四

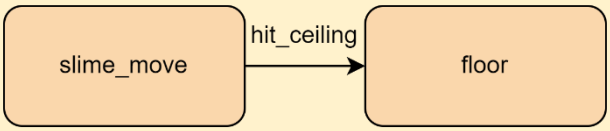
在 slime_move module 內，我們依照 Slime 的狀況將遊戲分為兩個 state，分別為 JUMP_UP 與 FALL_DOWN，在 JUMP_UP 時，Slime 是可以穿越板子的，在 FALL_DOWN 時才會判定使否有踩到板子。

  - **模擬重力加速度**

當 Slime 處於 JUMP_UP 的狀態時，Slime 會以一個初速度向上跳躍，並逐漸變慢，直到某固定高度時，state 會變成 FALL_DOWN，此時 Slime 會向下加速移動，為了避免在下落時穿越 floor 的問題，我們採取的作法是將跳躍的時間(time_gap)分成三段，第一段時每 1 個 clock cycle 移動一個 pixel，第二段每 4 個 clock cycle 移動一個 pixel，第三段每 8 個 clock cycle 移動一個 pixel，以此盡可能的模擬出重力加速度的效果。

  - **視野畫面上移**

為了避免 Slime 向上跳出螢幕範圍的問題，我們設定當 Slime 落下時踩到的板子位於螢幕的上半部的話，要將 slime_move module 接到 floor module 的 hit_ceiling 設為 1'b1 直到 Slime 的狀態再次切回 JUMP_UP，當 hit_ceiling 為 1'b1 時，floor module 會按照 Slime 跳躍的運動方式將所有的 floor 向下移動。



圖五

■ Slime 二段跳

當 Slime 踩了 5 次 floor 之後，Slime 會變成紅色並可以獲得一次二段跳的機會，可以使用空白鍵觸發，若是在 FALL_DOWN 的狀態下觸發，則作法與碰撞到 floor 是一樣的(如圖四)，而在 JUMP_UP 的狀態下觸發，則會將 time_gap (counter)歸零，已達成重新以初速度向上跳躍的功能。

● Slime 左右移動

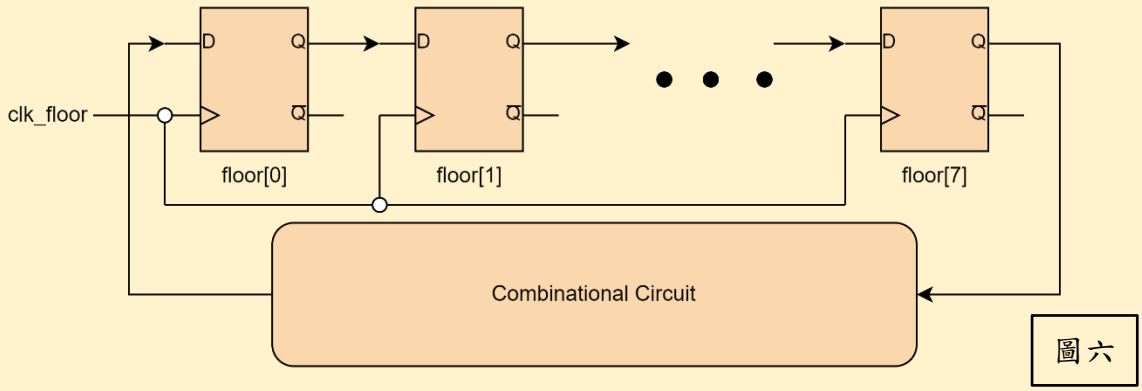
使用 A 鍵與 D 鍵選擇移動方向，固定以 2 個 clk_vga cycle 移動 1 個 pixel，若 Slime 移動到螢幕的左右邊界時，則會從另一邊出現。

● Slime 與 floor 的碰撞判定

為了簡化判定的電路(少作一些加減法)，Slime 的位置基準點是 Slime 的左下角，由基準點向右向上畫出 Slime，而 floor 的位置基準點是 floor 的左上角，由基準點向右向下畫出 floor，因此只要當 Slime 位置的 y 座標為 floor 的 y 座標-1 (Slime 底邊與 floor 頂邊有接觸)，則判定為有碰撞，Slime 的狀態會從 FALL_DOWN 切回 JUMP_UP。

● floor 生成與位置計算

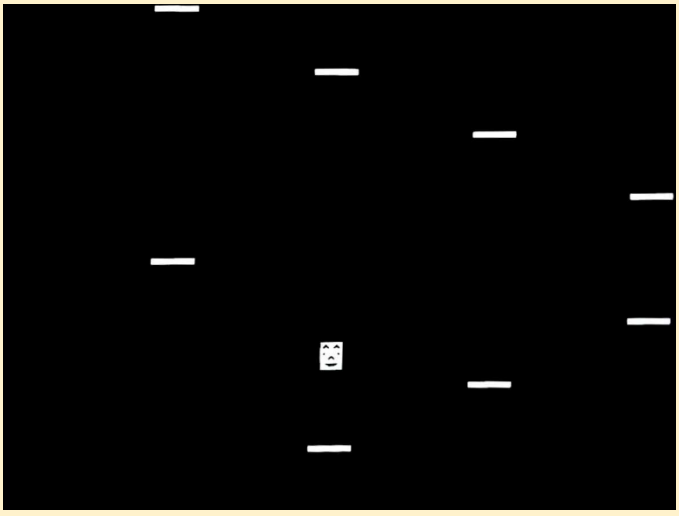
遊戲畫面水平切割成 8 層，每一層又垂直切割成 16 區，每層固定有 1 個 floor，分布在 16 區中的其中一區，除了一開始剛進遊戲的 floor 位置是固定的，後續生成的 floor 的 x 座標會是偽隨機的，主要是使用一個以 100MHz 的 clk 觸發，範圍為 0~30 的 counter，並擷取其[3:0]所表示的數字，乘以 floor 的寬度 40 換算成 floor 應該生成的 x 座標，當 floor 向下移動超過螢幕下界時，就會將 floor 的 y 座標設回 0，x 座標則以上述方式產生。

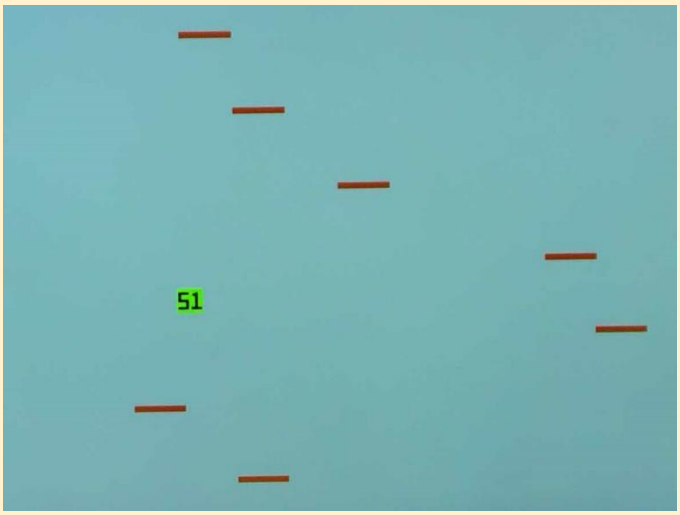


圖六

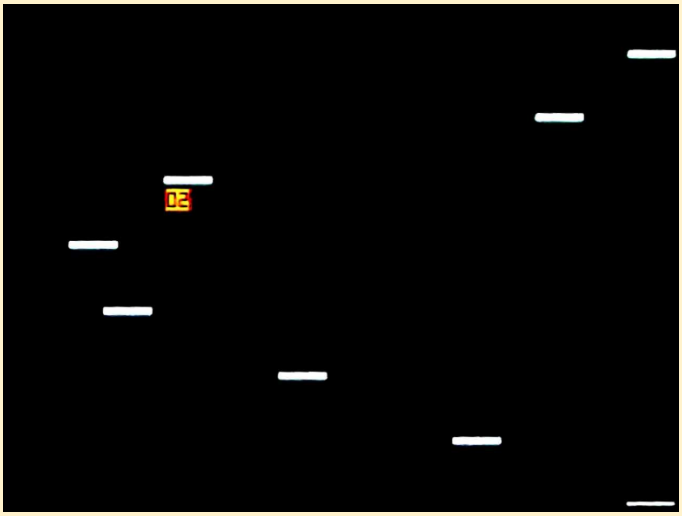
由於 8 個 floor 的 combinational circuit 是一樣的，為了節省邏輯閘的使用量，我們在 floor module 裡面，將儲存這 8 個 floor 位置的 DFF 做成了 FIFO Queue 的形式(如圖六)，並將原本 clk_vga 的 clock cycle 切成 8 段做成 clk_floor，每個 clk_floor 的 cycle 會處理一個 floor 的位置並塞回 FIFO Queue，做 8 次之後剛好可以與原本由 clk_vga 當 divided clock 的其他功能同步。

5. 遊戲畫面





超過 50 分畫面會變成彩色的

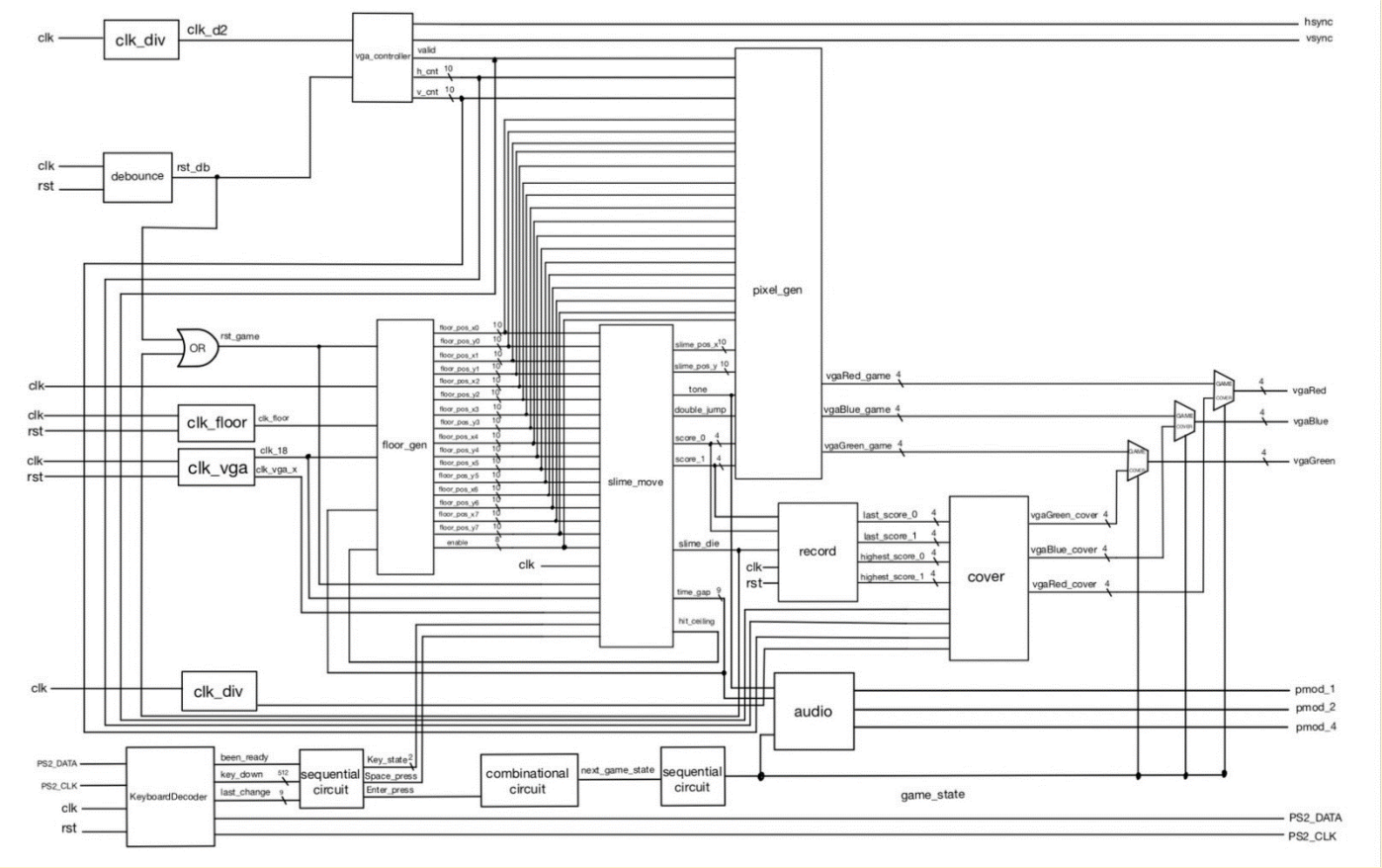


獲得二段跳的機會時 Slime 會變成紅色

6. 工作分配

整份 project 的所有東西(包含 proposal、code、report)均為共同討論完成的，貢獻度分別為 50%。

7. Top Module Block Diagram



8. 心得

- **劉祐廷：**

  在 Lab5 與 Lab6 的時候，由於時間很靠近期末了，課業壓力逐漸繁重，老實講其實對於 vga、keyboard 這些外接設備的 code 都只是讓他會跑就好，並沒有好好了解這些 module 在座什麼事情，因此在剛開始做 final project 時，重新好好的 trace 了所有的 code，對於外接設備的使用更加了解。而在 floor 的 FIFO queue 這個設計讓我收穫良多，原本只是遇到了一些 bug 以為是板子的邏輯閘不夠，因此突發奇想想到了這個方法，突然就理解了老師上課說的：「可以不用在一個 clock cycle 就把所有事情做完」，成功實作出這個功能時還挺有成就感的。

- **李侑霖：**