**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………………………………………………...……P3**

**2. 遊戲架構……………………....…………………….……….P3**

**3. Module…………………………………………...…...…….P3**

**4. 遊戲功能…………………………………………………P4**

**5. 遊戲畫面…………………………………………….……..P7**

**6. 工作分配…………………………………………………P7**

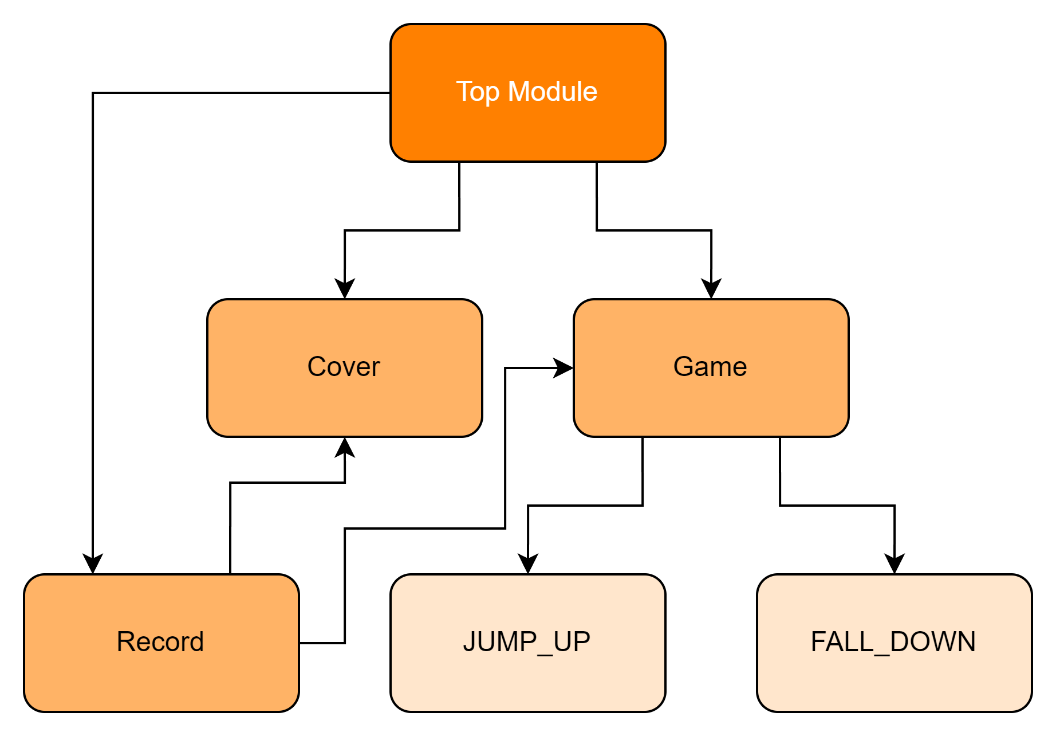
**7. Top Module Block Diagram……………………………....P8**

**8. 心得………………………………………………………...P8**

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。

1. **遊戲架構**

****

圖一

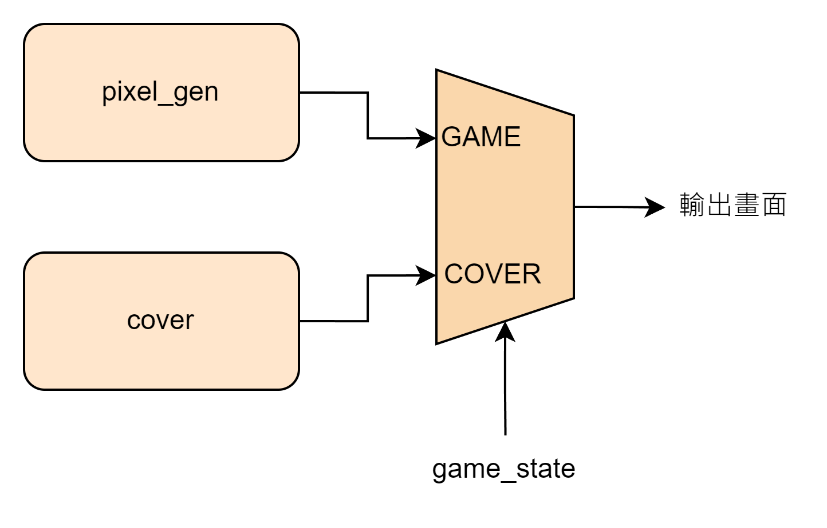
首先我們先將遊戲分為Cover與Game兩種不同的狀態，分別對應到目前應該顯示起始畫面或是目前應該顯示遊戲畫面。在遊戲畫面(Game)的狀態下，依照遊戲主角Slime當前的狀態分為JUMP\_UP與FALL\_DOWN兩種狀態，分別代表Slime目前處於向上跳躍或是向下墜落的狀態。

1. **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。 |

1. **遊戲功能**

* **遊戲開始畫面與遊戲畫面的切換**

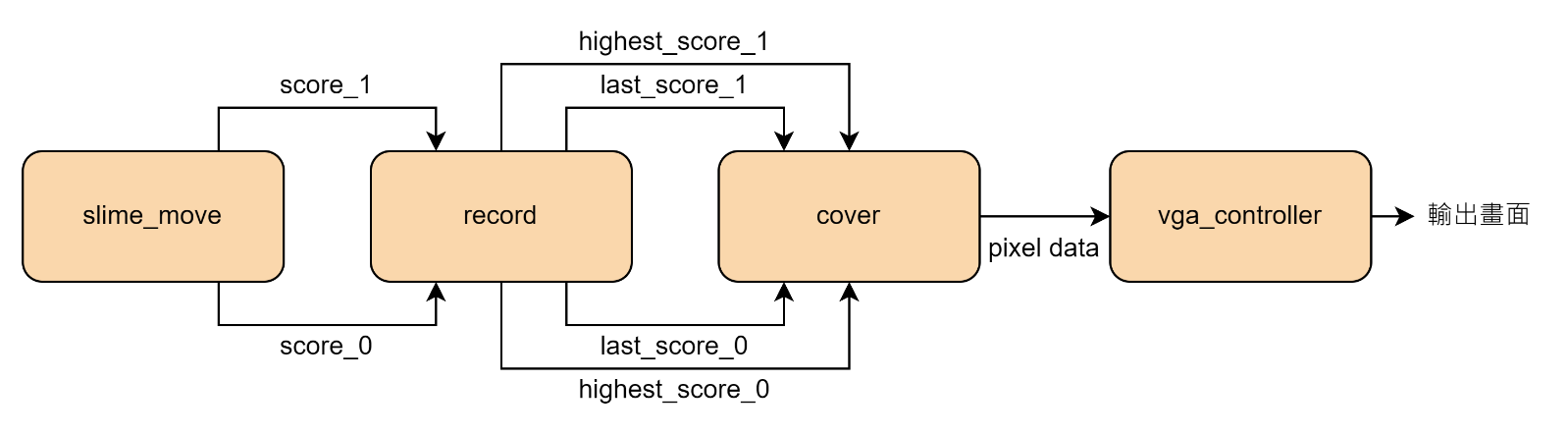


圖二

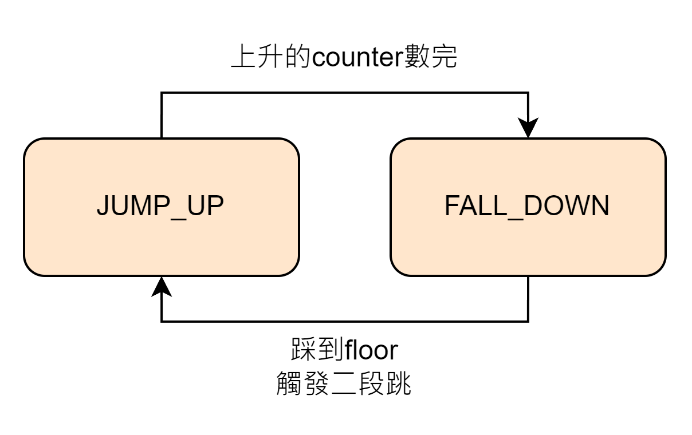
我們是透過用mux去選擇輸出畫面(如圖二)，來達成遊戲場景切換的功能，在遊戲開始畫面時按下Enter，game\_state會被設成GAME，進入遊戲畫面，而當Slime死亡後，game\_state會再被切回COVER，回到遊戲開始畫面。在測試的時候有遇到在遊戲開始畫面時碰觸Key A與Key D後，上局分數被改變的bug，後來發現原因是因為以我們的設計來說，遊戲畫面與起始畫面是同時運行的，因此在遊戲開始畫面時也能操控Slime移動，而去動到上一局的分數，後來我們夠過在鍵盤判定的地方多加了一個以game\_state當條件的mux，使Key A與Key D在遊戲開始畫面時失效以解決這個問題。

* **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

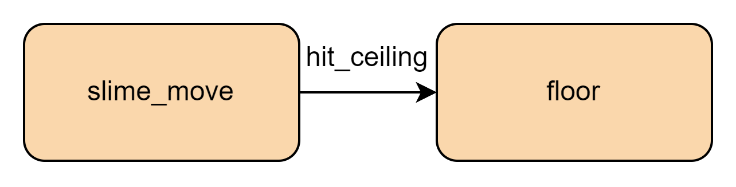
在slime\_move module內，我們依照Slime的狀況將遊戲分為兩個state，分別為JUMP\_UP與FALL\_DOWN，在JUMP\_UP時，Slime是可以穿越板子的，在FALL\_DOWN時才會判定使否有踩到板子。

* + **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**

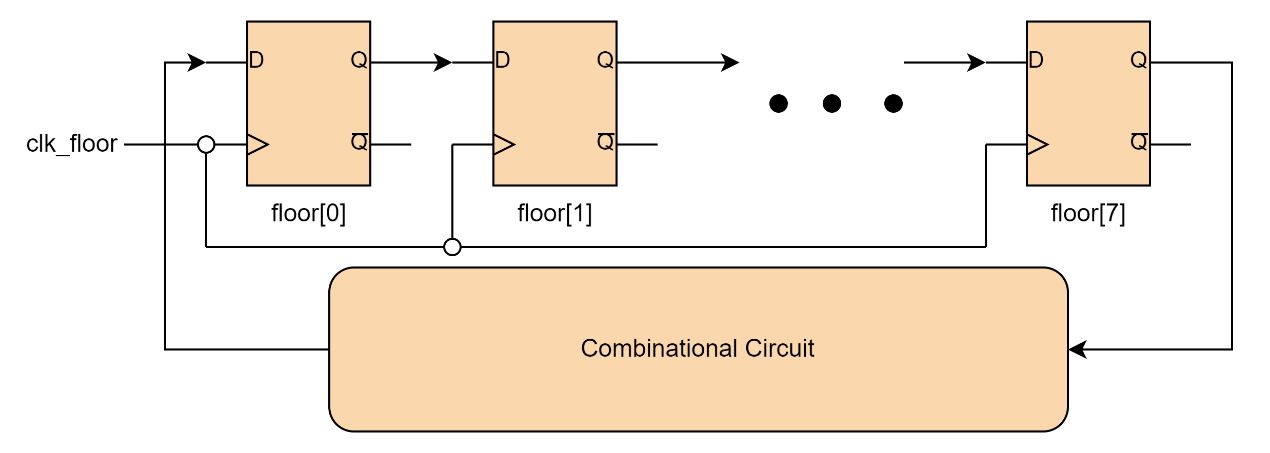
使用A鍵與D鍵選擇移動方向，固定以2個clk\_vga cycle移動1個pixel，若Slime移動到螢幕的左右邊界時，則會從另一邊出現。

* **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**

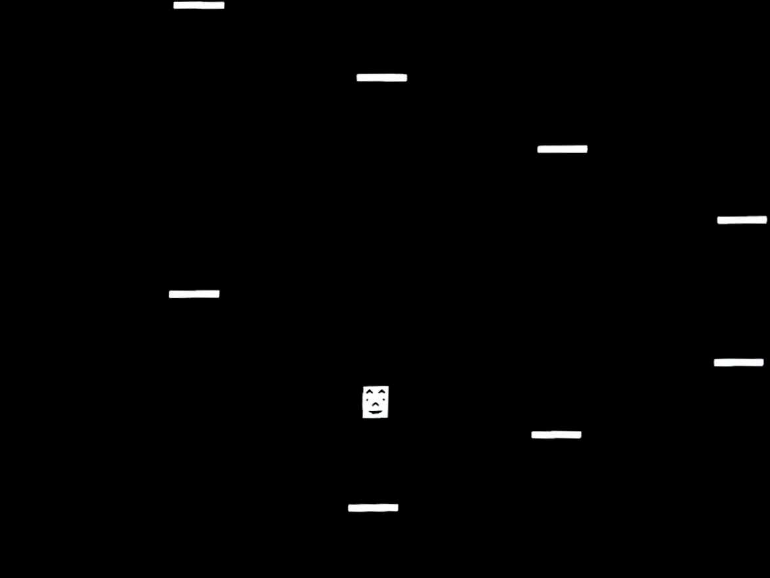
遊戲畫面水平切割成8層，每一層又垂直切割成16區，每層固定有1個floor，分布在16區中的其中一區，除了一開始剛進遊戲的floor位置是固定的，後續生成的floor的x座標會是偽隨機的，主要是使用一個以100MHz的clk觸發，範圍為0~30的counter，並擷取其[3:0]所表示的數字，乘以floor的寬度40換算成floor應該生成的x座標，當floor向下移動超過螢幕下界時，就會將floor的y座標設回0，x座標則以上述方式產生。

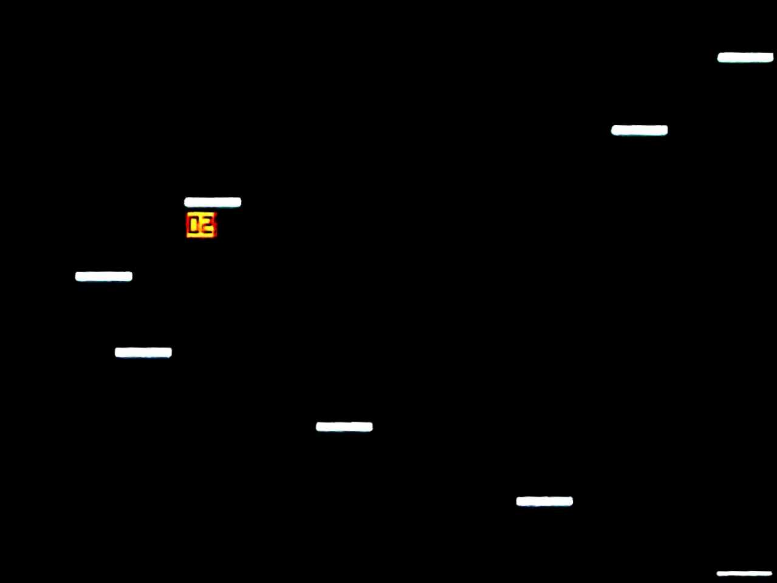


圖六

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。

1. **遊戲畫面**



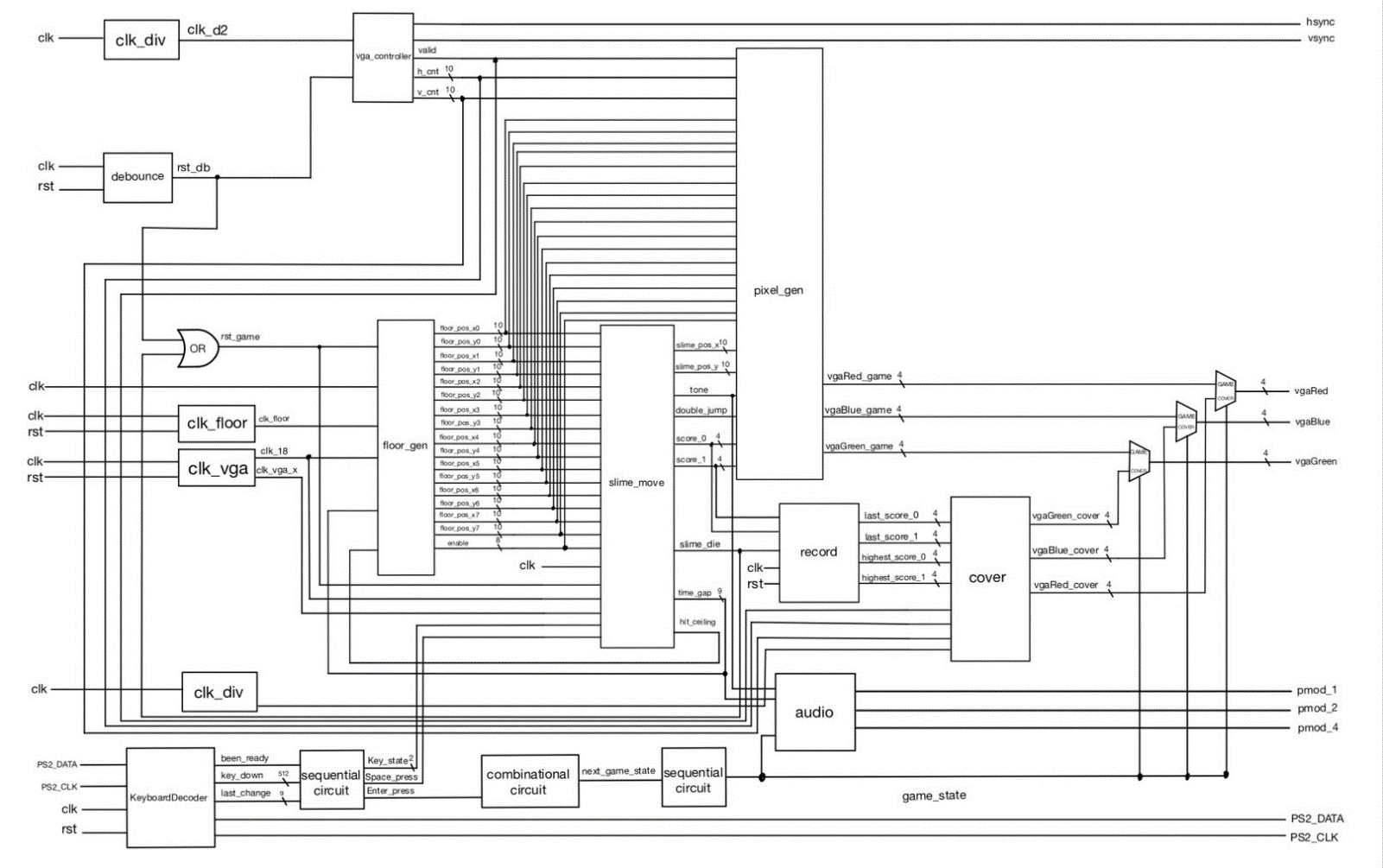


獲得二段跳的機會時Slime會變成紅色

超過50分畫面會變成彩色的

1. **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%。

1. **Top Module Block Diagram**
2. **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：**