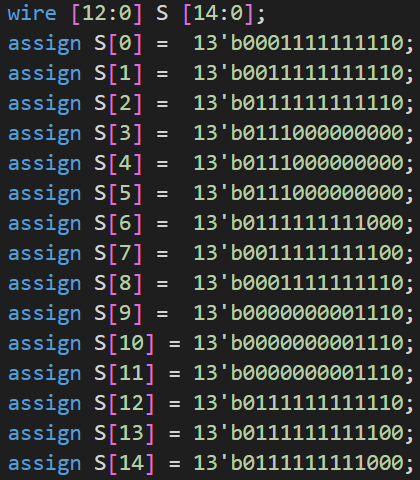
遊戲起始畫面：

此畫面主要實作在cover module裡面，包含遊戲標題、上局分數、最高分數與閃爍的提示字。

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。

閃爍的提示字是在cover module內，透過counter搭配DFF的方式，以固定的間隔與頻率把顏色設為黑色與白色的方式實現。

畫面：這份project裡面所有的畫面都是由預先儲存在module裡面的二維陣列畫出來的(如圖所示)



Slime jump：

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。

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分成三段，第一段時每1個clock cycle移動一個bit，第二段每4個clock cycle移動一個bit，第三段每8個clock cycle移動一個bit，以此盡可能的模擬出重力加速度的效果。

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。

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，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。

