Final Project

1.Letter Dropping Game

Design Specification

✓ Write down the specification of your design: inputs, outputs and related bit widths. The name should be the same as what you use in the Verilog design files.

```
inout PS2_CLK;
inout PS2_DATA;
input clk;
input rst_n;
output [15:0] life_led;
output [7:0] show;
output [3:0] ssd_ctrl;
output hsync, vsync;
output [3:0] vga_Red;
output [3:0] vgaGreen;
output [3:0] vgaBlue;
```

✓ Draw the block diagram of the design.

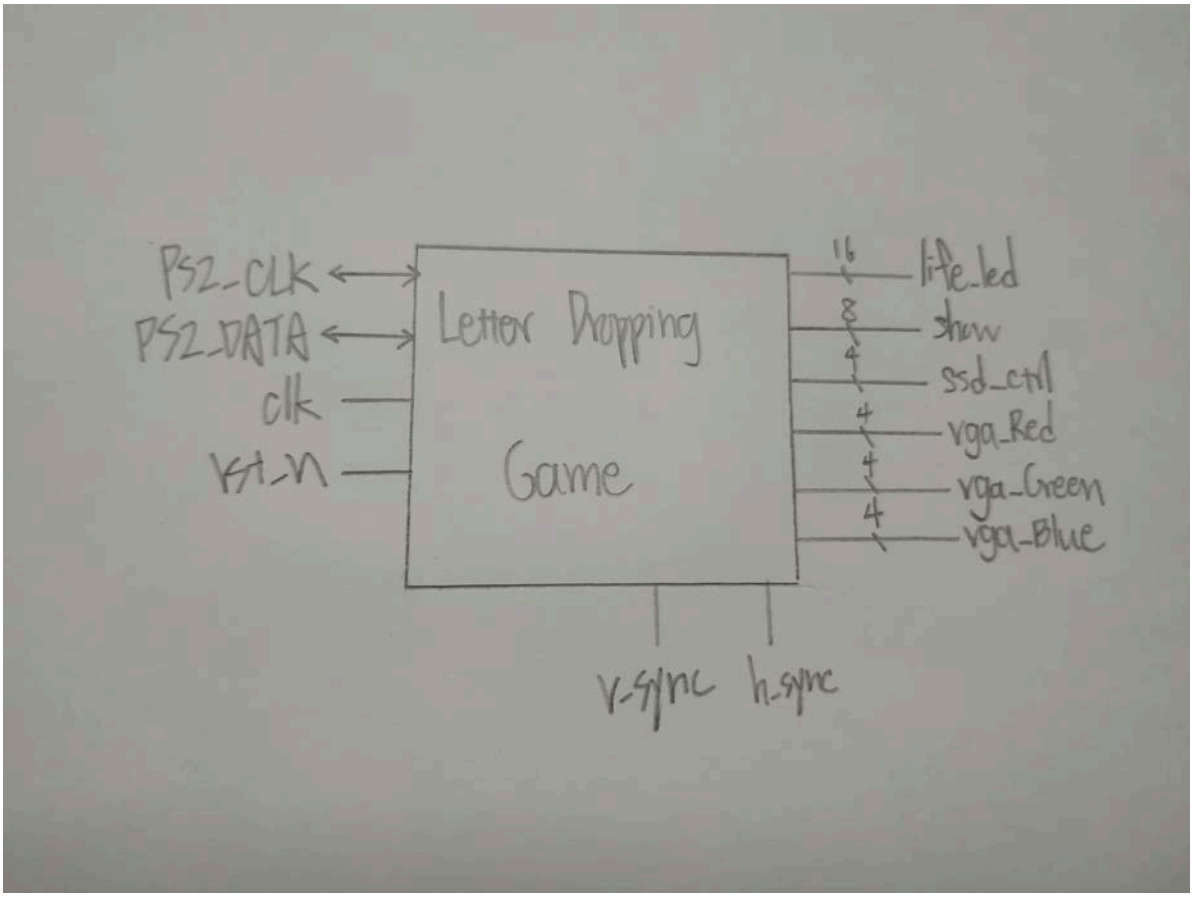


圖 1:letter dropping game block diagram

Design Implementation

✓ Derive the related logic functions ***and/or*** draw the related logic diagram.

1.mem_addr_gen block

功能:藉由螢幕當前位址以及鍵盤的輸入，判斷螢幕上該顯示什麼圖像。此外，裡面邏輯中有一組在這一個遊戲中扮演重要角色的 shift register，主要目的為儲存每一列代表字母的數值、以及那一列字母的存在行數。

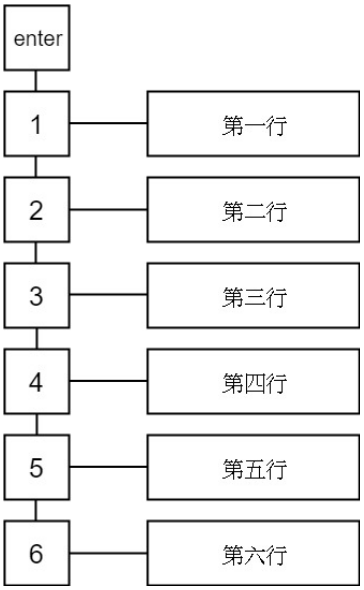


圖:字母下降概念圖

(1)概念圖左行: 6 層樓的 shift register

主要目的為儲存每一列代表字母的數值、以及那一列字母的存在行數。而代表字母的數值、存在行數會傳送給決定每一列顯示影像的 code。

(2)概念圖右行:決定每一列顯示影像的 code

如上圖，每一列儲存的值傳送給對應的 code，而 code 的功用為將字母顯現在螢幕上。

```
if((h_cnt>=(column_1st*50+120))&&(h_cnt<(column_1st*50+170))&&(v_cnt>=(step*5))&&(v_cnt<(step*5+50)))
begin
    if(alphabet_1st= =5'd0)
        pixel_addr = 1;
    else
        pixel_addr = (alphabet_1st-1)*2500+((v_cnt-5*step)*50+(h_cnt-120-column_1st*50));
```

上述 code 中有幾個重要變數:column、alphabet_1st、step，column 和 alphabet 代表某一列隨機產生的行數以及哪一個字母。至於 step，舉第一列為例，當 step 為 0 時，字母位置在 0~50 區間，當 step 為 1 時，字母位置在 5~55 區間，以此類推。當 step 從 9 跑到 0 時，第一列的字母、行數也剛好傳到第二列，因此有順順地往下傳的效果。

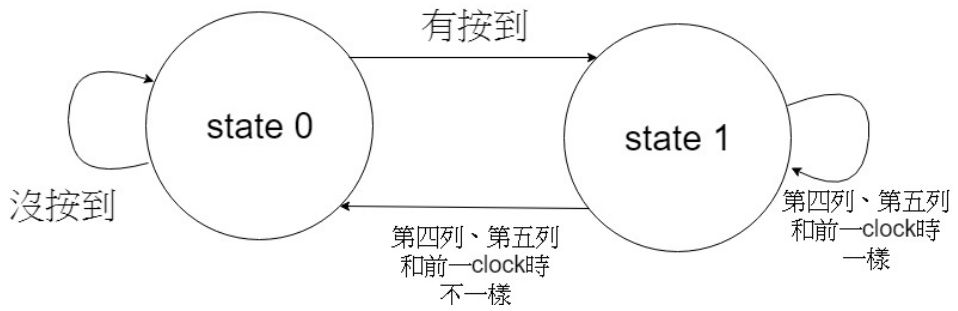


圖:判斷有無擊中字母的 FSM

有沒有按到字母的方式為以 last_change 和 alphabet_6th 判斷有沒有一樣，當原本 state 為 0 代表未得分且扣命，只要一被按到，那麼 state 便會變成 1，代表有得分。state1 回到 state0 時，條件必須為 alphabet_4th、alphabet_5th 必須和前一刻不同(類似 one_pulse_generator，當前一刻與當下值不同時執行指令)。

這一塊 block 最後就是命、分數的 code，命、分數初始值分別為 16、0，並且兩著皆會傳到外部。

2.level counter

傳入除頻器所製造出來的 1Hz clock，用來數現在遊戲開始幾秒鐘了，第一關會持續 21 秒，第 21 到 40 秒為第二關，第 40 秒後為第三關，速度持續增快。所以這個 module 會依照現在的秒數傳出一個 2-bit 訊號，2'b00 代表第一關，2'b01 代表第二關，2'b10 代表第三關，用來讓其他需要關數做判斷的 module 知道現在是第幾關。

3.life display

我們設定總共 16 條命，分別顯示在 16 個 LED 上，每答錯一次命就減一條，也就是最左邊的 LED 燈會暗掉。因此，本 module 的主要功能就是依照現在的[4:0] life 訊號(5-bit 的原因是因為有 16 條命)，轉換成燈的亮暗情形，傳出 16-bit 的 LED 燈號訊號。比如，若現在有 11 條命，則這個 16 bit 的輸出就是 16'b0000_0111_1111_1111，依此類推。

4.score display

此 module 的主要功能就是將現在的分數顯示到七段顯示器上，現在的分數是一個 10bit 的訊號，運用先前 lab 的技巧分離出千位數、百位數、十位數個位數，並配合除頻器傳出來的 2-bit 快速變動的 clock 輸出到七段顯示器上。

5.background music

先引入之前 lab 的 speaker module。之後，我拉入了除頻器的 2Hz 的 clock，製作一個 counter，為了要循環播放背景音樂，我取這個 counter 的值除以總音符數目的餘數，去判斷我現在要送哪一個音符進到 note_div_decoder，在游 note_div_decoder 送出我要的[21:0] note_div 進到 speaker 的 module 之中，產生相對應的音高。

6.last change decoder

傳入鍵盤的[8:0] last change，傳出相對應的數字，1 代表 A、2 代表 B……依此類推，讓其他 module 判斷是否正確。

✓ List the I/O pin assignment for your design.

PS2_CLK	PS2_DATA	clk	rst_n	life_led[15]	life_led[14]
C17	B17	W5	V17	L1	P1

life_led[13]	life_led[12]	life_led[11]	life_led[10]	life_led[9]	life_led[8]
N3	P3	U3	W3	V3	V13
life_led[7]	life_led[6]	life_led[5]	life_led[4]	life_led[3]	life_led[2]
V14	U14	U15	W18	V19	U19
life_led[1]	life_led[0]	show[7]	show[6]	show[5]	show[4]
E19	U16	W7	W6	U8	V8
show[3]	show[2]	show[1]	show[0]	ssd_ctrl[3]	ssd_ctrl[2]
U5	V5	U7	V7	W4	V4
ssd_ctrl[1]	ssd_ctrl[0]	hsync	vsync	vga_Red[3]	vga_Red[2]
U4	U2	P19	R19	N19	J19
vga_Red[1]	vga_Red[0]	vgaGreen[3]	vgaGreen[2]	vgaGreen[1]	vgaGreen[0]
H19	G19	D17	G17	H17	J17
vgaBlue[3]	vgaBlue[2]	vgaBlue[1]	vgaBlue[0]		
J18	K18	L18	N18		

Discussion

周沛毅的部分:

1.字母上下跳動的 bug

這一個 bug 是這一次 final project 中最大的 bug,原本我們期望字母方塊會非常順地掉下來，但是我們最初字母方塊會規律地上下跳動。

首先，第一次 debug 之前，控制字母方塊速度的相關 clock : down_velocity 以及 next_picture_delay 在三個關卡有三種頻率。我們採用選擇的方式，也就是利用多功器，選擇已經做好的頻率作為上述兩 clock 的頻率。不過老師說過，考量到電壓穩定問題，clock 理應不該經過任何邏輯判斷直到控制 DFFs。因此，我們改為調整最大值的方式，調整上述兩者的頻率。

接著，字母方塊依舊會上下跳動。我們推斷字母方塊上下跳動原因為 step 和 shiift register 的週期不同步。因此，我們想出使它們週期同步的方法:一起計數。

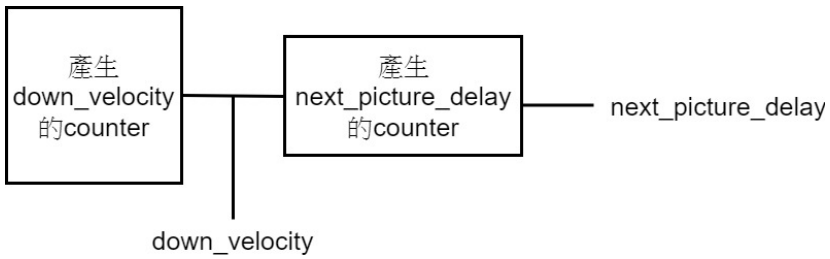


圖:一起計數的關係圖

首先，利用限制 counter 計數最大值，數到最大值時使 down_velocity 從 0 變成 1 或從 1 變成 0。再來，down_velocity 傳送給一個數十次的 counter，每數五次就使 next_picture_delay 從 0 變成 1，從 1 變成 0。如此兩者便能同步，最後 Bug 也因此得以解決。

2.屏幕逐漸黯淡

當我在 debug 上述 bug 時，有一次的執行結果為一開始正常，過不久屏幕變黯淡，而且只剩綠色，紅色、藍色都消失了。最後黑屏，螢幕再也開不了，就算是 reset 之後還是開不了。一開始我推斷是接線問題，可能 vgaRed、vgaBlue 那邊有問題，後來接上另外一個螢幕，而且接頭接緊，結果螢幕還是會逐漸黯淡，最後黑屏。因此，我再推斷應該是 code 本身有問題。

最後發現是當時產生 next_picture_delay 的 code 有複雜計算，包含/、%，可能因此造成計算量過大，系統不穩定，最後掛機。之後才想到用上述方式產生正確、同步的 clock。

謝霖泳的部分

在真正打 verilog code 之前，我們先把大概的 block diagram 畫出來，並藉由 block 進行分工。我也先用小畫家畫了一張圖，格視為.jpg 檔，上面是這次所需要用到的圖形，也就是字母 A~Z，為了方便配合 h_cnt 與 v_cnt 的位置顯示相對應的圖片區塊，我將每一個字母的大小都設定成 50*50，在轉為.coe 檔變為 pixel 的一維陣列之後，我們便可以很清楚的知道，第幾個值到第幾個值分別代表哪一個字母，以字母 A 為例，就是.coe 檔中的 0~2499，因為我已經設定好每個字母 50*50 的大小。

由於我們這對分工的方式是以 block 去分工，而這次的除頻器一開始是由我負責，我都用跟之前 lab 相同的方法打，可是這次卻一直出現無法解決的 bug，導致我們樸片再往下滑的時候會往上跳動，研判其可能原因為除頻器出來的頻率雖然都是由同一個 100MHz 的 global clock 所產生，但他們之間並不會完完全全同步，會有一點點非常微小的誤差，又因為我們為了讓圖片下滑的過程看起來比較順暢，因此將時間間格切得非常細，每次只下降 5 pixel，因為時間間格很小，因此時間就會有很多段，原本很小的誤差就會一直被放大，因此產生了我們預期之外的不規則跳動。

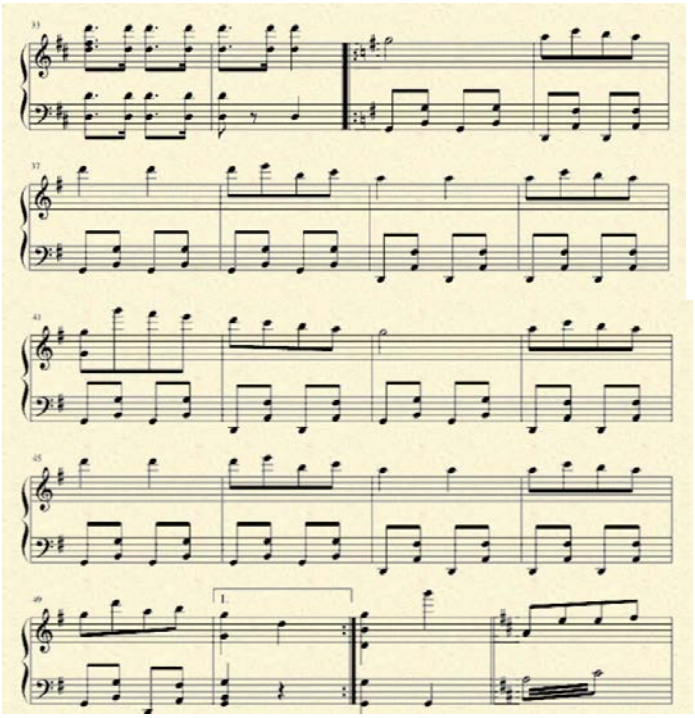
此外，我們的 design 之中，有一快也是由我負責的 level counter module，根據之前所述，該 module 的功能是遊戲開始時開始 count，在不同的秒數後，傳出不同 level 的訊號。此外，這個 module 原本還有一些任務，就是傳出對應該關卡的圖片下降所需的兩個 clock，原本我的作法是，將除頻器製作出來與圖片下降有關的 6 個 clock 全部傳入我的 level counter module，並藉由一組 MUX 去依照我的秒數區間選擇我要的 clock 作為輸出，但因為這樣會一直產生 bug，所以推測可能原因就是之前上課曾經提及的「除頻器出來的 clock 不可再經過其他邏輯元，否則電壓會被減弱，會產生問題。」

因此，後來改變方法，不是依照我的 level 去選擇 clock，而是依照我的 level，去選擇兩個相對應的數字，傳入除頻器之中，作為除頻器當中 counter 的依據，也就是讓除頻器的 counter 加到某一特定的數值時，讓他相對應的 output clock 訊號 toggle，這樣就不會產生 clock 經過 MUX 的問題了。

在這次的 final project 之中，還有遇到幾個問題，就是有幾次改完 code 燒上去之後，原本顏色都正常的方塊，過幾秒後顏色突然愈來愈深、愈來愈奇怪，最後甚至整個黑屏，什麼都看不到，我推測其可能原因為我們沒有處理好 VGA demo2 module 中的 valid 訊號的位置判定，讓原本應該是 valid 的位置變成 invalid，因此無法顯示我們在該區的 pixel，而顯示到別區的 pixel，讓顏色看起來非常奇怪，甚至黑屏。

© Ma, Hsi-Pin, Lab for Reliable Computing (LaRC), EE, NTHU.

而整個 design 的背景音樂也是由我負責的，背景音樂的選歌也讓我費盡心思，後來考量到這是一個速度會愈來愈快的遊戲，因此找了一首節奏輕快的歌曲，名為「康康舞曲」，是知名音樂家雅克·奧芬巴赫（Jacques Offenbach，1819~1880）所作輕歌劇《天國與地獄》的序曲，應該對大部分人來說都是耳熟能詳的。為了讓音樂配合遊戲快速的節奏，我使用來播放音樂音符的 clock 是 5Hz，並且在遊戲期間不斷循環播放，所用的方法是時間除以總音符數的餘數，去對應相應的[21:0] note_div，傳入 speaker 的 module 之中。以下在兩個重複記號之間的就是我所取用的歌譜片段。



Conclusion

周沛毅的部分:

對我來說，這一次final project最深刻的意義為學習自己做出一個簡單的小電腦遊戲，並且利用電腦背後判斷的結果，將結果顯示於螢幕上。如何搭起在邏輯與螢幕之間的橋樑是這一次final project和lab 11第三題bonus的重點。

此外，這一次final project另外一個相當深刻的意義為學習如何和自己的隊友溝通。為了進入狀況更容易，我們這一組基本上同一塊module都是同一個人打出來，另外一個人頂多在debug的時候幫忙尋找盲點。另外，我發現我和隊友都有各自相對較強、熟悉的部分，因此在最初分工的時候，我們就依據討論的時候，誰想出來某部分的功能，內部code該怎麼打，去分工。最後，兩個人一起打code效率真的高出許多，如果這一個由系全部都由自己完成，那麼我可能要花上更久的時間尋找我的盲點，以及減少做出錯誤判斷的機率，例如做白工。

謝霖泳的部分：

這次的期末專題讓我學到了很多，除了在 verilog 上的 coding 技巧與該注意的地方之外，也讓我學到了如何與別人合作完成一個複雜的 project，也就是「要走得快，一個人走；要走得遠，大家一起向前走」的道理。其中，有一些組別都有變數未溝通清楚而命名重複，或是不知道別人的變數叫什麼的問題，但我們幾乎沒有遇到這種溝通上的問題，因為早在一開始畫 block diagram 時，就已經先把 top module 中的 input、output、inout 與 reg 和 wire 的名字訂好了，因此比較不會遇到這種溝通上的問題。

© Ma, Hsi-Pin, Lab for Reliable Computing (LaRC), EE, NTHU.

「如果你有心要做一件事情，全世界都會來幫你。」這幾天，我們也在實驗室度過了幾個白天與黑夜,在實驗室中有許多同學都曾經給我們意見和幫助,讓我們又多做了不少遊戲的優化。一個 project 從無到有絕不是一個人三兩下就可以搞出來的,因此必須感謝在宿舍和實驗室每個給過我們意見的同學，也要感謝教授這學期的教導，讓我們可以在 deadline 前順利生出這個 final project。

References

沒有