Final project

第 21 組 邱仁緯 張耀中

Introduction

這款遊戲就像是 Google 的離線小恐龍一樣用跳躍鍵避開障礙物，其中有地面上的仙人掌和天上的翼手龍。除了這些還有自製的音樂以及用七段顯示器來計分。

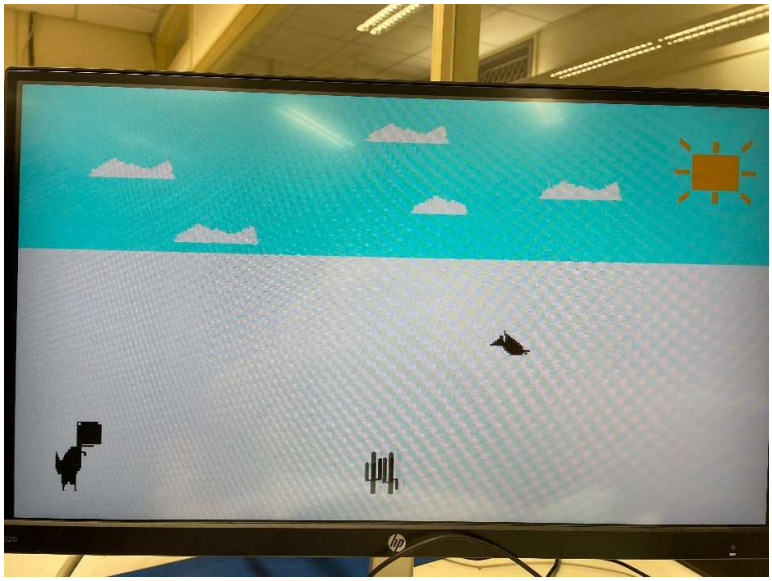
Motivation

一開始我就想要做一個小遊戲但是不知道要選哪一個，但是由於網路上已經很多像是貪吃蛇、推箱子、小朋友下樓梯的影片了所以最後我選了做類似 google chrome 斷線後可以玩的小恐龍遊戲。

Game screen shot

初始畫面

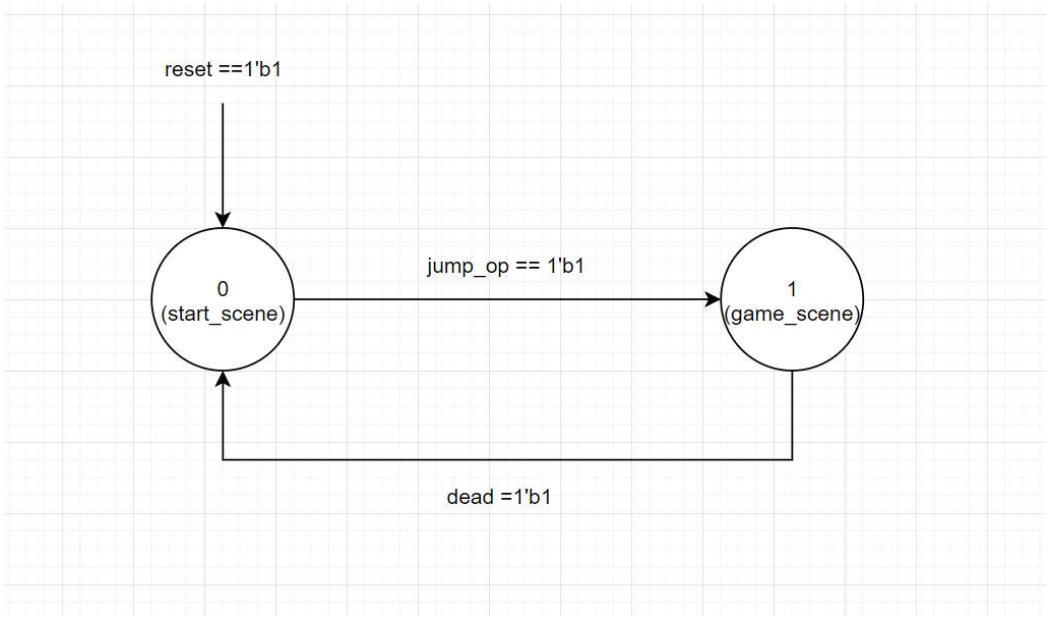
遊戲畫面



Equipment

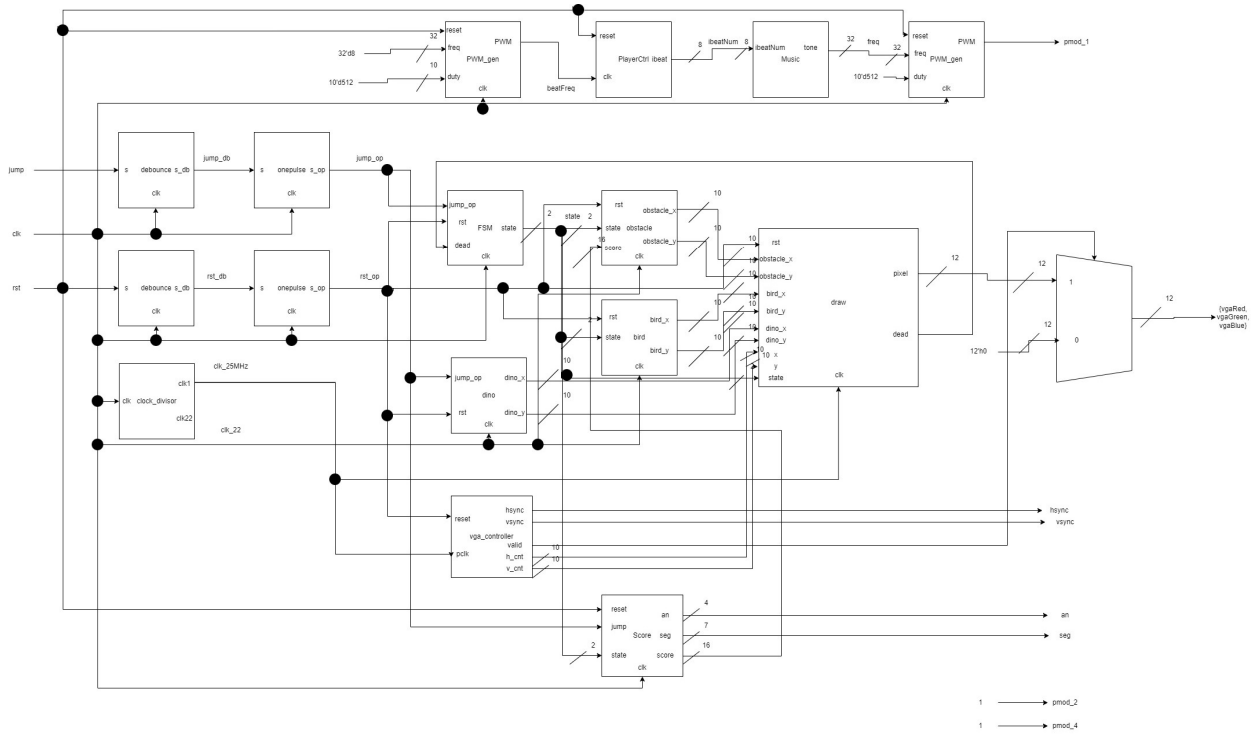
這個遊戲有用到 VGA 線、音源裝置和 FPGA 板。

State diagram



Block diagram:

黑色點表示該處有分支



Details

Top module 的 input 有 clk,rst 和 jump 其中 jump 就是板子上的 up button 而 rst 則是 center button，而 output 則是有各 4 個 bits 的 RGB 訊號、hsync 和 vsync 用來顯示螢幕中的座標、pmod 1、pmod2、pmod4 用來控制音樂、最後 an[3:0]和 seg[6:0]用來控制七段顯示器。在其中有許多 module 而簡單可以分為遊戲、音樂和計分三大部分(並非完全獨立)。因此以下將分別用這三大部分介紹。

一. 遊戲

這部分占最多並且最重要，分別有 FSM、dino、obtacle、bird、draw、vga_controller，先簡單介紹一些 module 的功能。FSM 負責處理 State 的轉換、dino 負責計算恐龍目前座標、obtacle 負責計算仙人掌目前座標、bird 負責計算翼手龍目前座標、vga_controller 負責輸出現在螢幕顯示的像素座標、draw 則是將所有物體座標以及背景根據現在的 h_cnt & v_cnt 輸出目前這個座標的像素顏色。以下詳細介紹幾個比較重要 module 的細節。

(1) FSM:

就是 Finite state machine，其中 input 有 jump_op 和 dead 而 output 是 state。State 有兩個，0 代表開始畫面而 1 代表遊戲畫面。一開始想當然 state = 0 而 jump_op=1’b1 時就會進入遊戲畫面也就是 state = 1，而如果 state == 1 時 dead ==1’b1 的話 state 則會從 1 變回 0 回到開始畫面，其中 dead 這個訊號是由之後的 module draw 負責輸出。

(2) dino:

這個 module 的 input 為 clk,rst 和 jump_op，output 為 dino_x[9:0]、dino_y[9:0]，我用 dino_x 和 dino_y 當作恐龍的左上角座標並在之後的 draw module 中畫出來。

恐龍在 jump_op == 1’b1 要跳躍也就是讓 dino_y 減少再增

加(y 減少表示在螢幕上往上)，為了達到此目的我先令兩個變數 jump、updown 分別代表 jump = 1 時表示正在跳躍中，updown =0 時表示正在上升而 updown=1 時表示正在下降。

```
always @(*)begin
    if(jump_op == 1'b1 && dino_y>=ground)begin
        next_jump = 1'b1;
        next_updown = 1'b0;
    end else begin
        if(dino_y <= minh)begin//reach the hightest point
            next_jump  = 1'b1;
            next_updown = 1'b1;
        end else begin
            if(dino_y >= ground && updown ==1'b1)begin// falling on the ground
                next_jump  = 1'b0;
                next_updown = 1'b0;
            end else begin//else remain
                next_jump  = jump;
                next_updown = updown;
```

此上面的 code 可以解釋為若 jump_op =1 且恐龍在地面時(若不設此條件則恐龍可以在空中連跳)則開始往上跳，而若達到最大高度時則 updown = 1 即往下掉的意思，最後往下掉到地面時 jump 和 updown 都回到初始狀態。

我用 move 來表示 dino_y 的更新頻率，在 count[21]==1’b1 時讓 move =1’b1 也就是 dino_y 可以更新了。以下為部分 code。

```
always @(posedge clk) begin
  if(count[21] == 1'b1)begin//update rate of jumping
        move = 1'b1;
        count <= 21'd0;
  end else begin
        move = 1'b0;
        count <= count +1'b1;
  end
end
```

最後根據計算出的 jump、updown 和 move 就能計算出 dino_y 了，當 move ==1’b1 時若 jump = 1’b1 時，updown ==1’b0 則往上也就是減少 dino_y 反之則增加 dino_y。以下為部分的 code。

```
always @(*)begin
    next_dino_x = dino_x;
    if(move == 1'b1 && jump == 1'b1)
        if(updown ==1'b0)
            next_dino_y = dino_y - 10'd5;
        else
            next_dino_y = dino_y + 10'd4;
    else
        next_dino_y = dino_y;
end
```

(3) obstacle(仙人掌)、bird(翼手龍):

這兩個 module 基本上大同小異所以一起講，與上面的 dino 都是用輸出物體的 x,y 座標只不過它們只有座標 x 會隨時間改變，比較特別的是 obtacle 中多了一個 input score 之後介紹的 module score 中的

output 也就是目前的分數，仙人掌會根據分數越高而變得更快而遊戲也會變得更難。

(4) vga_controller:

這部分就是之前給的 sample code 的部分，主要就是提供現在螢幕輸出的像素的座標，為了方便我在接下來的 draw 中將 h_cnt, v_cnt 分別叫做 x,y。

(5) draw:

這個 module 的 input 就是所有物體的左上角的那一點的 x,y 座標、當前螢幕輸出的像素的座標的 x,y 和 state 而 output 就是該 x,y 該輸出的顏色 pixel 共 12bits(RGB 分別各有 4bits)和 dead 訊號。這個 module 要負責的事情有畫出物體和處理物體碰撞，以下將說明如何處理。

(一) 物體碰撞

有了每個物體的左上角的 x,y 座標後只要令好每個物體的長、寬就能用矩形碰撞區域來知道物體在的區域，如下圖中用 isdino 來表示若為 1 代表這是恐龍的身體

```
35 assign isobtacle = ( x>obtacle_x && x<(obtacle_x+obtacle_w) && y>obtacle_y && y<(obtacle_y+
36 assign isdino = ( x>dino_x && x<(dino_x+dino_w) && y>dino_y && y<(dino_y+dino_h));
37 assign isbir= ( x>bird_x && x<(bird_x+bird_w) && y>bird_y && y<(bird_y+bird_h));
```

最後用下面這行代表若恐龍與翼手龍 or 仙人掌重疊的畫就會讓 dead = 1’ b1 並讓之前的 FSM module 的 input 收到進

而改變 state 到初始畫面。

```
38  assign dead  = (isobtacle&&isdino&&(state ==2'b01)||isbird&&isdino&&(state ==2'b01));
```

(二)　畫出物體

這裡用非常暴力的辦法來畫圖，以 dino 為例我們用 draw_dino 這個訊號表示恐龍要上色的地方。Assign draw_dino 就能得到當前 x,y 是否是恐龍進而輸出它的顏色

如下圖(僅是冰山一角的 code)。

```
51  assign dino_draw =
52  ( (x== (dino_x + 10'd3)) &&(y > (dino_y+10'd35)) &&(y <dino_y+10'd55) )||
53  ( (x== (dino_x + 10'd4)) &&(y > (dino_y+10'd38)) &&(y <dino_y+10'd58) )||
54  ( (x== (dino_x + 10'd5)) &&(y > (dino_y+10'd41)) &&(y <dino_y+10'd61) )||
55  ( (x== (dino_x + 10'd6)) &&(y > (dino_y+10'd44)) &&(y <dino_y+10'd61) )||
56  ( (x== (dino_x + 10'd7)) &&(y > (dino_y+10'd44)) &&(y <dino_y+10'd61) )||
57  ( (x== (dino_x + 10'd8)) &&(y > (dino_y+10'd44)) &&(y <dino_y+10'd71) )||
58  ( (x== (dino_x + 10'd9)) &&(y > (dino_y+10'd41)) &&(y <dino_y+10'd79) )||
59  ( (x== (dino_x + 10'd10)) &&(y > (dino_y+10'd38)) &&(y <dino_y+10'd75) )||
60  ( (x== (dino_x + 10'd10)) &&(y > (dino_y+10'd77)) &&(y <dino_y+10'd79) )||
61  ( (x== (dino_x + 10'd11)) &&(y > (dino_y+10'd35)) &&(y <dino_y+10'd72) )||
62  ( (x== (dino_x + 10'd12)) &&(y > (dino_y+10'd35)) &&(y <dino_y+10'd72) )||
63  ( (x== (dino_x + 10'd13)) &&(y > (dino_y+10'd32)) &&(y <dino_y+10'd69) )||
```

最後用 25MHz 的 always block 中根據當前 x,y、state 和物體來輸出 RGB，下圖為部分的 code。

```
934         end else if(state == 2'b01)begin//game
935             if( obtacle_draw )
936                 pixel <= 12'h000;
937             else if(dino_draw)
938                 pixel <= 12'h000;
939             else if(bird_draw)
940                 pixel <= 12'h000;
```

二. 音樂

這部分並不難，只需要將之前 sample 中 PWM_gen、PlayerCtrl 引入並將 Music 這個 module 中的頻率改成自己的音樂即可，其中它

的 output 為 pmod1,pmod2,pmod4，而且在遊戲中我們自己寫了自創的電子音樂。

三. 計分

這個 module 中負責在七段顯示器上用一秒得一分顯示目前分數，用 counter 從 0 在 100M 的 clk 中加到 100M 後讓 number 加一就能有一秒加一分的效果，在 state = 1 遊戲開始時才會開始計分而死掉後到 state =0 時還會保留分數直到再次進入遊戲。而其他部分如 seg 和 an 在以前 lab 就學過二位數的七段顯示器而推到 4 位數也是一樣的道理。

Contribution

邱仁緯:整體遊戲 && report 中除了 Block diagram 的全部。

張耀中:遊戲中的圖案、填樂譜 && block diagram。

What have I learn

邱仁緯:

雖然說在程設就有做過遊戲了但是和 c++比起來 verilog 又更麻煩了。不像

c++有方便的 class 和 library，說到底 verilog 就是一堆的邏輯電路頂多只能用 behavior modeling 來簡化整個電路，連螢幕輸出和音樂在 c++輕而易舉的事情都要用好幾個 module 才能做出粗糙的成品。雖然說做了一個 c++就能做的遊戲但是我覺得最重要的就是我學到了很多更底層、更硬體的知識，從第一堂的 lab 的用 basic gate 做出一些 module，lab3、lab4 有了 sequential 和 combinational 的概念，到 lab5、lab6 用這些 module 輸出到 vga、audio 讓螢幕有畫面、喇叭有聲音讓我對硬體的知識有更多的了解了，因此我也認為真正的資工系就是該了解整個架構精通軟體、硬體才能寫出更好的程式。

張耀中:

我做的部分屬於勞力活動，填樂譜是用醜小鴨的節奏加一點巧思來改編為電子音樂，另外畫出仙人掌、翼手龍、恐龍、雲朵、太陽、S、T、A、R、T 字樣都是一個一個座標去試、去找他的位置的。