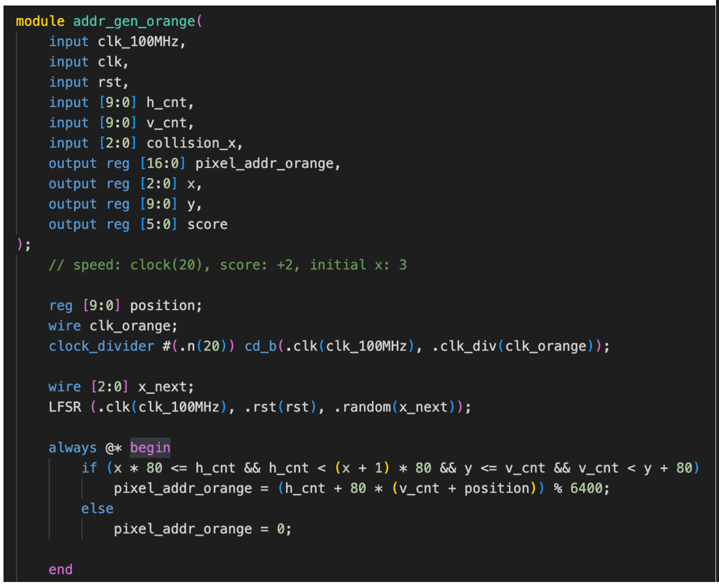
Final Project

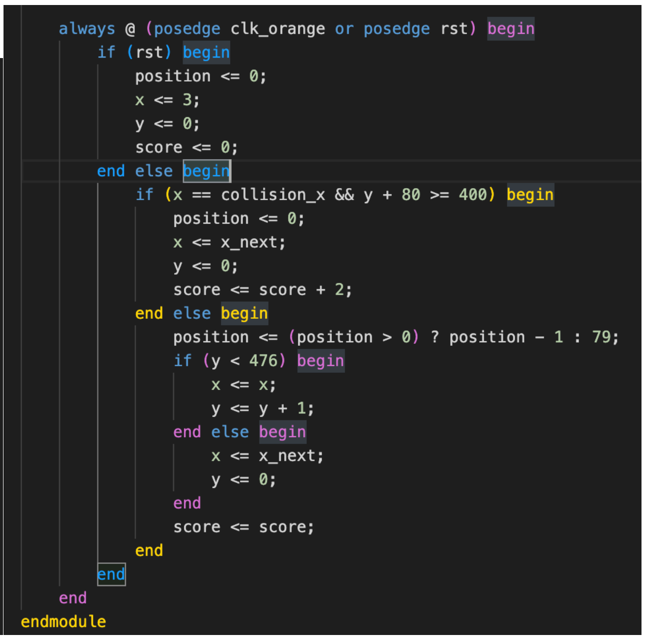
學號: 108062174 姓名: 吳竹婷

A. Lab Implementation

1. 以 LFSR 實作掉落物隨機降落的位置。
2. 每個掉落物有自己的屬性，(x, y) 座標、分數、掉落用的 clock、隨機產生 module。第一個 combinational block 利用座標產生遮罩，讓畫面看起來只有一個掉落物；sequential block 則設定初始位置、更新 y 座標、判斷碰撞並更新 x 座標、更新分數。最後會輸出 pixel address、座標、分數。

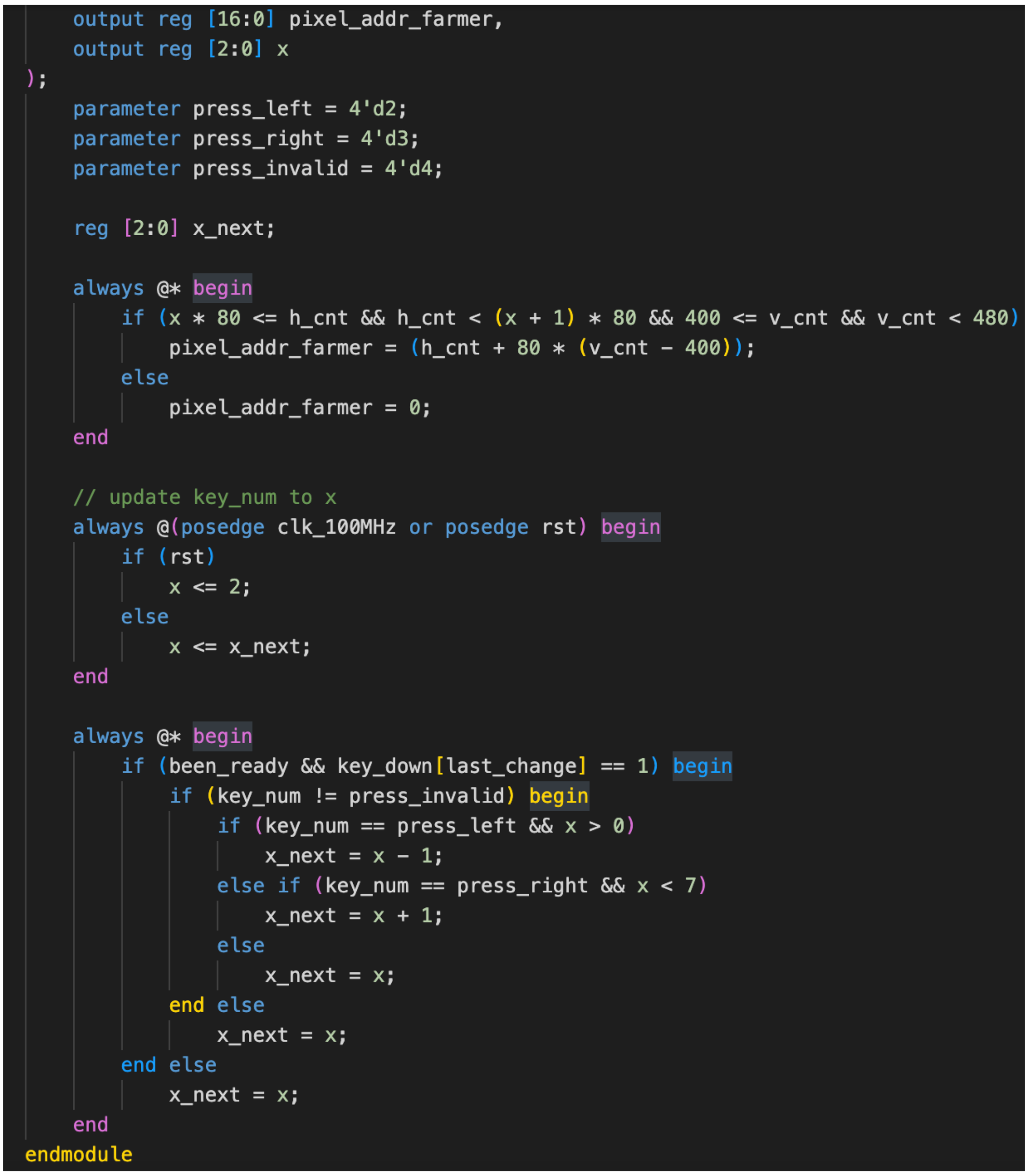


```
module addr_gen_orange(
    input clk_100MHz,
    input clk,
    input rst,
    input [9:0] h_cnt,
    input [9:0] v_cnt,
    input [2:0] collision_x,
    output reg [16:0] pixel_addr_orange,
    output reg [2:0] x,
    output reg [9:0] y,
    output reg [5:0] score
);
    // speed: clock(20), score: +2, initial x: 3

    reg [9:0] position;
    wire clk_orange;
    clock_divider #(.n(20)) cd_b(.clk(clk_100MHz), .clk_div(clk_orange));

    wire [2:0] x_next;
    LFSR (.clk(clk_100MHz), .rst(rst), .random(x_next));

    always @* begin
        if (x * 80 <= h_cnt && h_cnt < (x + 1) * 80 && y <= v_cnt && v_cnt < y + 80)
            pixel_addr_orange = (h_cnt + 80 * (v_cnt + position)) % 6400;
        else
            pixel_addr_orange = 0;

    end
```



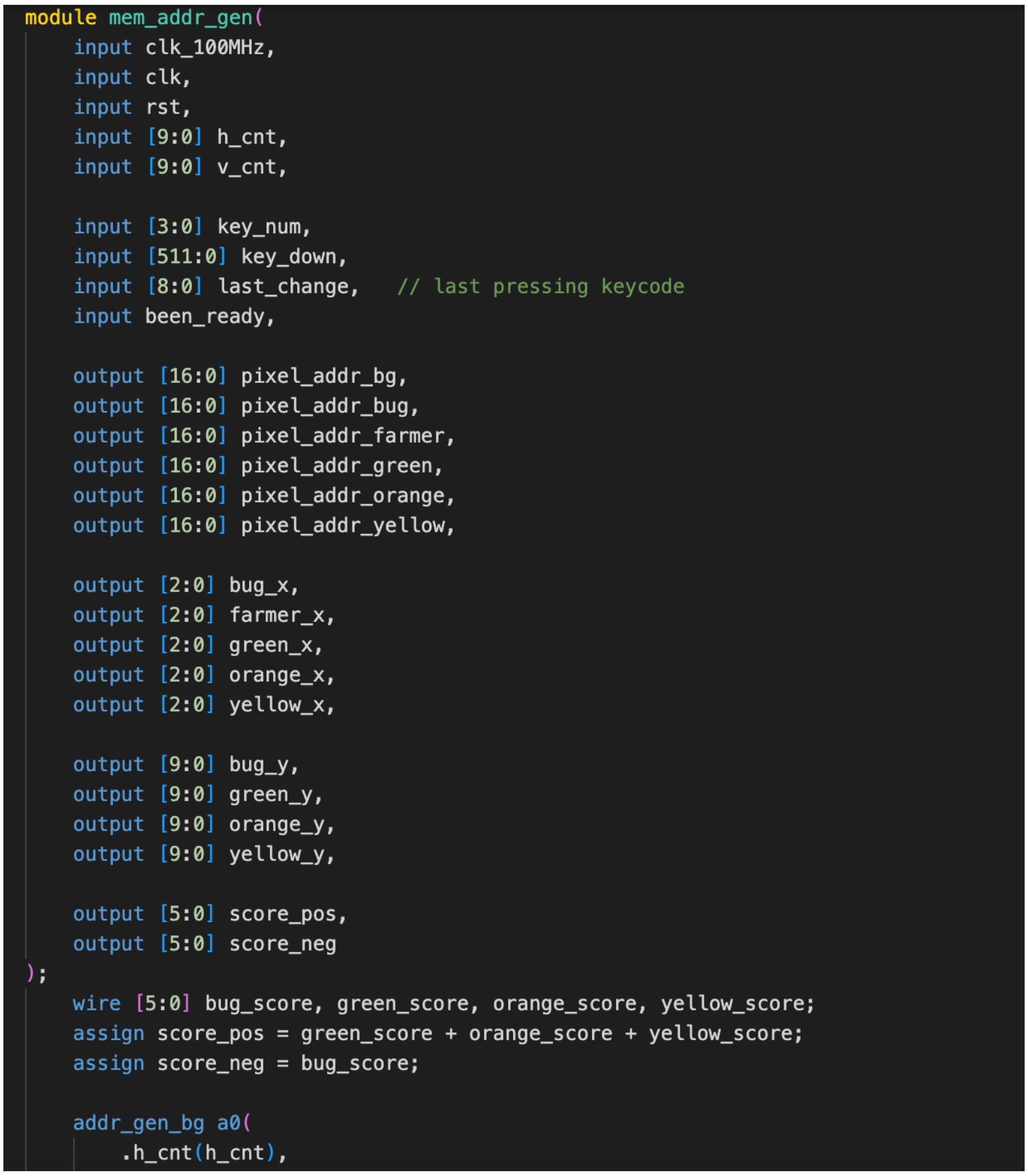
```
    always @ (posedge clk_orange or posedge rst) begin
        if (rst) begin
            position <= 0;
            x <= 3;
            y <= 0;
            score <= 0;
        end else begin
            if (x == collision_x && y + 80 >= 400) begin
                position <= 0;
                x <= x_next;
                y <= 0;
                score <= score + 2;
            end else begin
                position <= (position > 0) ? position - 1 : 79;
                if (y < 476) begin
                    x <= x;
                    y <= y + 1;
                end else begin
                    x <= x_next;
                    y <= 0;
                end
                score <= score;
            end
        end
    end
endmodule
```

3. 農夫和其他掉落物較為不同，因 y 座標固定故不另外用變數記，x 座標的更新則是 based on 鍵盤的行為，故會從 top module 將鍵盤的變數一路從外面傳入到農夫的 module，若按下的按鍵是合法的，且位置無超出範圍，則更新座標。



```verilog
    output reg [16:0] pixel_addr_farmer,
    output reg [2:0] x
);
    parameter press_left = 4'd2;
    parameter press_right = 4'd3;
    parameter press_invalid = 4'd4;

    reg [2:0] x_next;

    always @* begin
        if (x * 80 <= h_cnt && h_cnt < (x + 1) * 80 && 400 <= v_cnt && v_cnt < 480)
            pixel_addr_farmer = (h_cnt + 80 * (v_cnt - 400));
        else
            pixel_addr_farmer = 0;
    end

    // update key_num to x
    always @(posedge clk_100MHz or posedge rst) begin
        if (rst)
            x <= 2;
        else
            x <= x_next;
    end

    always @* begin
        if (been_ready && key_down[last_change] == 1) begin
            if (key_num != press_invalid) begin
                if (key_num == press_left && x > 0)
                    x_next = x - 1;
                else if (key_num == press_right && x < 7)
                    x_next = x + 1;
                else
                    x_next = x;
            end else
                x_next = x;
        end else
            x_next = x;
    end
endmodule
```

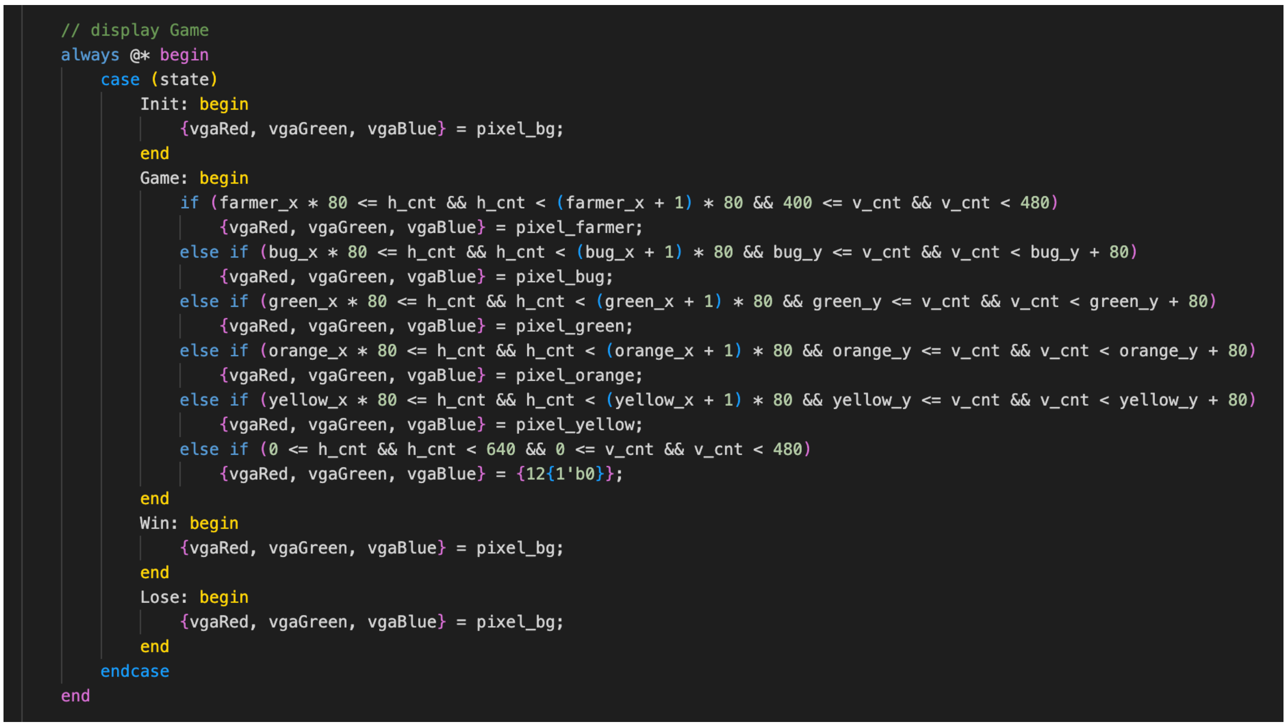
4. Mem_addr_gen module 作為 top module 和各個物件之間的連接，他會將所有物件需要用的變數傳入，例如農夫需要的鍵盤變數；以及將各個物件的輸出做整理後傳回給 top module，像是將個物件的分數分成加分和扣分後加總並回傳。



```
module mem_addr_gen(
    input clk_100MHz,
    input clk,
    input rst,
    input [9:0] h_cnt,
    input [9:0] v_cnt,

    input [3:0] key_num,
    input [511:0] key_down,
    input [8:0] last_change,    // last pressing keycode
    input been_ready,

    output [16:0] pixel_addr_bg,
    output [16:0] pixel_addr_bug,
    output [16:0] pixel_addr_farmer,
    output [16:0] pixel_addr_green,
    output [16:0] pixel_addr_orange,
    output [16:0] pixel_addr_yellow,

    output [2:0] bug_x,
    output [2:0] farmer_x,
    output [2:0] green_x,
    output [2:0] orange_x,
    output [2:0] yellow_x,

    output [9:0] bug_y,
    output [9:0] green_y,
    output [9:0] orange_y,
    output [9:0] yellow_y,

    output [5:0] score_pos,
    output [5:0] score_neg
);
    wire [5:0] bug_score, green_score, orange_score, yellow_score;
    assign score_pos = green_score + orange_score + yellow_score;
    assign score_neg = bug_score;

    addr_gen_bg a0(
        .h_cnt(h_cnt),
```

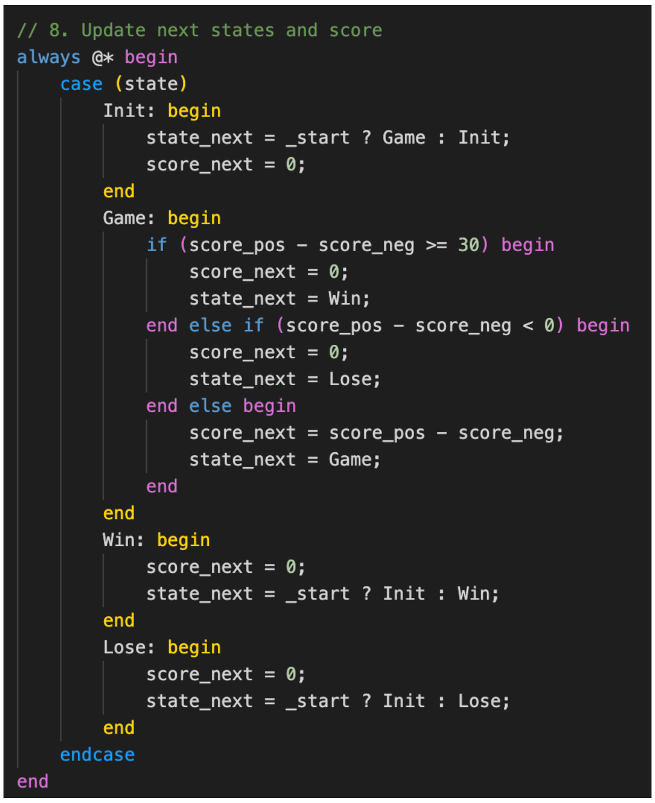
（以下為 top module）

5. 遊戲畫面顯示的概念類似圖層，越上層的物件放在越前面的 if statement，依序分別為農夫 > 扣分的物件 > 分數高的物件 > 分數低的物件 > 背景圖。並依照剛剛回傳的各個物件的座標顯示在螢幕上。



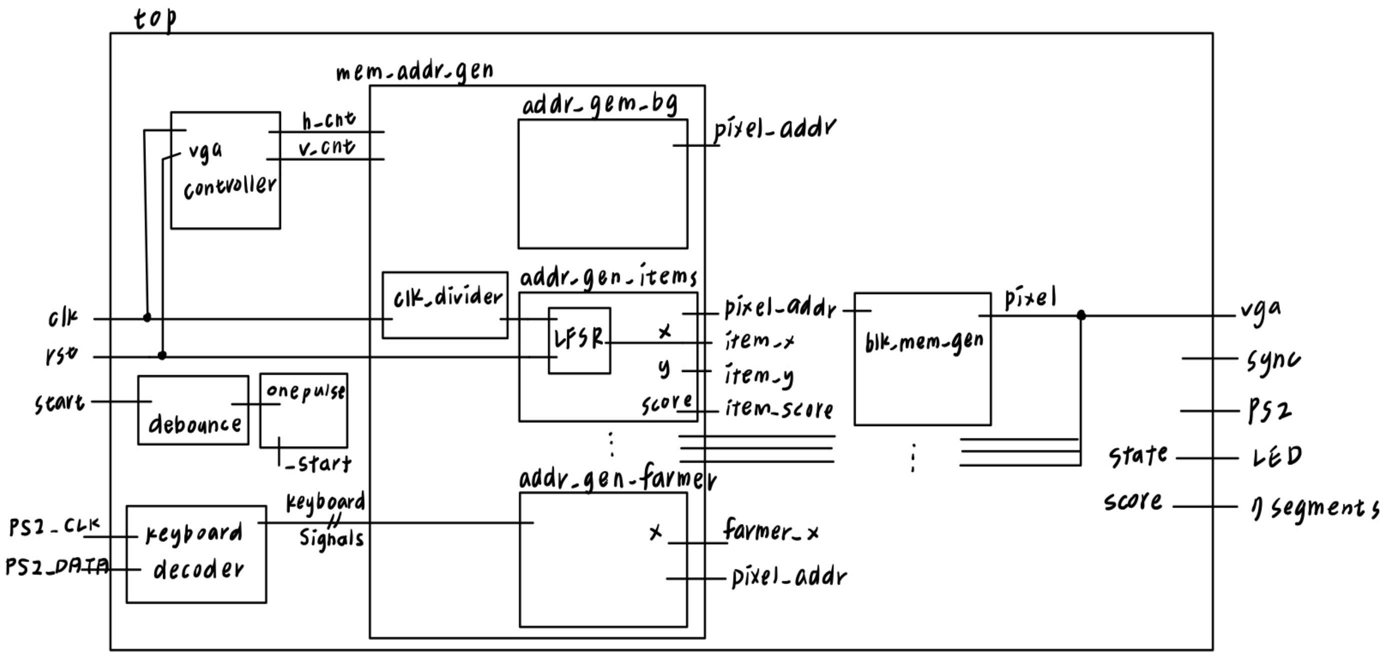
```verilog
// display Game
always @* begin
    case (state)
        Init: begin
            {vgaRed, vgaGreen, vgaBlue} = pixel_bg;
        end
        Game: begin
            if (farmer_x * 80 <= h_cnt && h_cnt < (farmer_x + 1) * 80 && 400 <= v_cnt && v_cnt < 480)
                {vgaRed, vgaGreen, vgaBlue} = pixel_farmer;
            else if (bug_x * 80 <= h_cnt && h_cnt < (bug_x + 1) * 80 && bug_y <= v_cnt && v_cnt < bug_y + 80)
                {vgaRed, vgaGreen, vgaBlue} = pixel_bug;
            else if (green_x * 80 <= h_cnt && h_cnt < (green_x + 1) * 80 && green_y <= v_cnt && v_cnt < green_y + 80)
                {vgaRed, vgaGreen, vgaBlue} = pixel_green;
            else if (orange_x * 80 <= h_cnt && h_cnt < (orange_x + 1) * 80 && orange_y <= v_cnt && v_cnt < orange_y + 80)
                {vgaRed, vgaGreen, vgaBlue} = pixel_orange;
            else if (yellow_x * 80 <= h_cnt && h_cnt < (yellow_x + 1) * 80 && yellow_y <= v_cnt && v_cnt < yellow_y + 80)
                {vgaRed, vgaGreen, vgaBlue} = pixel_yellow;
            else if (0 <= h_cnt && h_cnt < 640 && 0 <= v_cnt && v_cnt < 480)
                {vgaRed, vgaGreen, vgaBlue} = {12{1'b0}};
        end
        Win: begin
            {vgaRed, vgaGreen, vgaBlue} = pixel_bg;
        end
        Lose: begin
            {vgaRed, vgaGreen, vgaBlue} = pixel_bg;
        end
    endcase
end
```

6. 用分數判斷 states 的轉換。

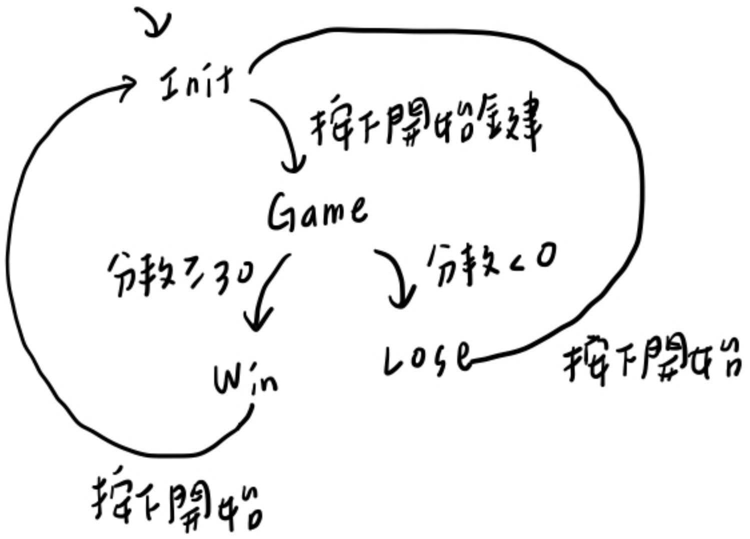


```verilog
// 8. Update next states and score
always @* begin
    case (state)
        Init: begin
            state_next = _start ? Game : Init;
            score_next = 0;
        end
        Game: begin
            if (score_pos - score_neg >= 30) begin
                score_next = 0;
                state_next = Win;
            end else if (score_pos - score_neg < 0) begin
                score_next = 0;
                state_next = Lose;
            end else begin
                score_next = score_pos - score_neg;
                state_next = Game;
            end
        end
        Win: begin
            score_next = 0;
            state_next = _start ? Init : Win;
        end
        Lose: begin
            score_next = 0;
            state_next = _start ? Init : Lose;
        end
    endcase
end
```

B. Block Diagram



C. Finite State Machine



D. Problem Encountered

1. 一直遇到沒有 error message 的 synthesis failed，確定 code 邏輯沒有錯；之後重開專案檔放上一樣的 code 又好了，推斷可能是 Vivado 的問題。
2. 水果的圖片需要能整除螢幕尺寸，否則會出現掉落時遮罩擷取錯誤的圖片範圍。
3. 依圖片順序放上圖片後想在最後底下放背景圖，但是放入後顏色會呈深紫色，最後捨棄背景圖設背景為全黑。