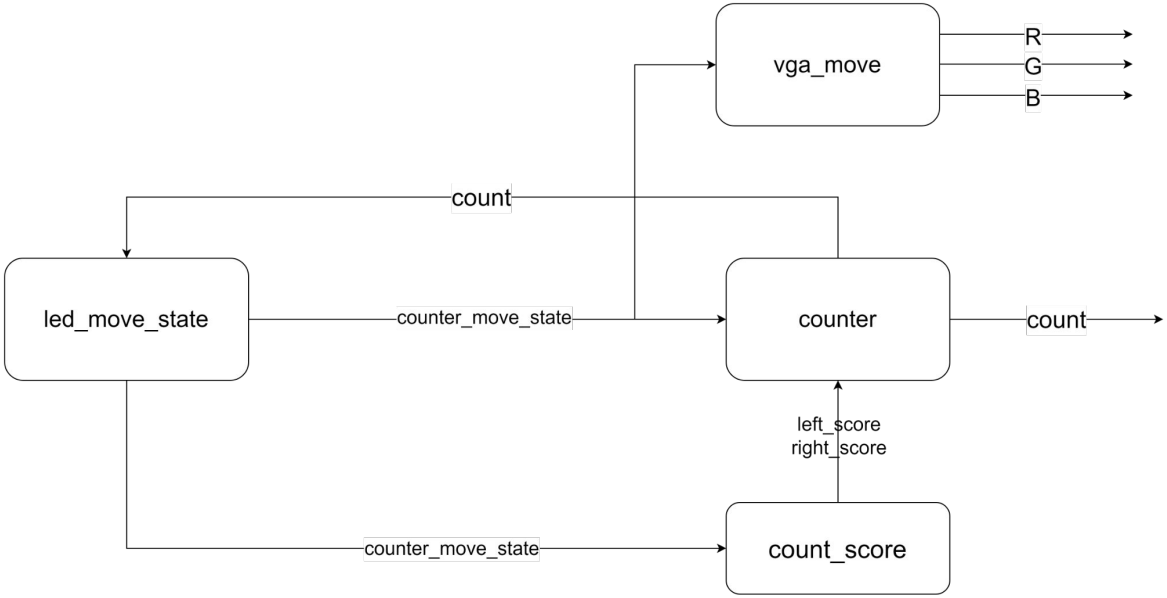
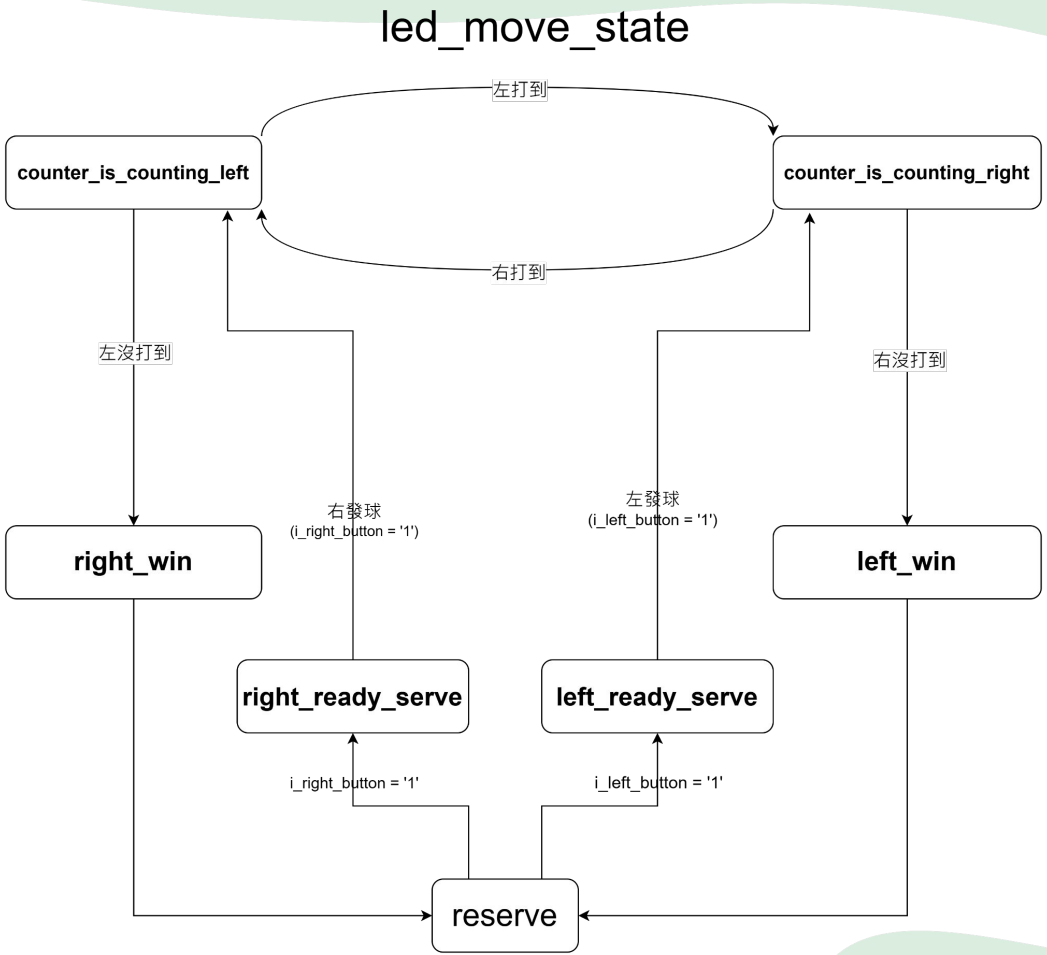
vga+pingpong

說明



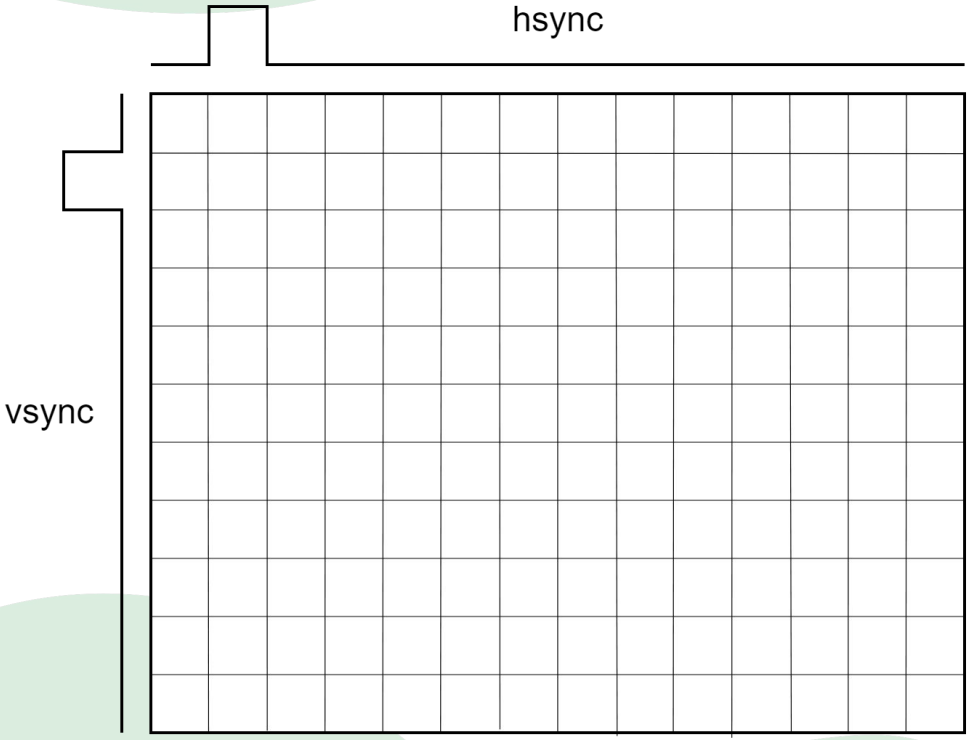
- Led_move_state從led的狀態，更新 counter_move_state傳給 counter和counter_score
- counter是用來位移 led
- counter負責計分假如左邊獲勝左邊就會加分再傳給 counter來顯示
- vga_move用來跟新螢幕上球的位子

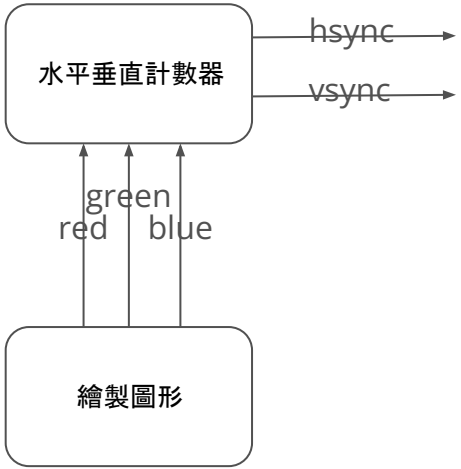


Led_move_state 內部架構圖說明

剛開始狀態從 reserve開始，如果 i_left_button按下進入左發球狀態，如果 i_right_button按下進入右發球狀態，當打出勝負並顯示完分數後會再進入 reserve，以此循環

架構





繪製圖形:在設計的面積，輸出RGB來顯示圖形顏色

水平垂直計數器:當hsync水平掃描完後，vsync會+1hsync會重新再掃描一次，當權部屬完後再重新開始

說明

```
begin
    process(fclk, rst_n)
    begin
        if rst_n = '0' then
            h_count <= 0;
            v_count <= 0;
        elsif rising_edge(fclk) then
            if h_count = 799 then
                h_count <= 0;
                if v_count = 524 then
                    v_count <= 0;
                else
                    v_count <= v_count + 1;
                end if;
            else
                h_count <= h_count + 1;
            end if;
        end if;
    end process;
```

檢查 **h_count** 是否達到 799(水平掃描已經完成)，否則繼續 +1，**h_count** 的計數範圍是從 0 到 799(共 800 個像素周期，包括同步脈衝、前座標、顯示區域和後座標)。當 **h_count** 數到 799 時，該行的顯示完成，準備開始新的一行。

更新 **h_count** 和 **v_count** 兩個計數器，用來生成 VGA 的水平計數和垂直計數。當 **h_count** 數到 799(這代表一個水平掃描線結束時)，就會將 **h_count** 重置為 0，同時 **v_count** +1，當 **v_count** 達到 524(即整個畫面掃描完成)，則重置為 0，以此達到掃描顯示區域的循環

顯示球的位子

```
constant H_SYNC_CYCLES : integer := 96;  -- 水平同步脈寬
constant H_FRONT_PORCH : integer := 16;  -- 水平前座標
```

```
-- 圓形的繪製邏輯
process(fclk, i_rst)
begin
    if i_rst = '0' then
        red   <= "0000";
        green <= "0000";
        blue  <= "0000";  -- 初始為黑色
    elsif rising_edge(fclk) then
        -- 圓心位置 (480, 360)，半徑 15
        if ( (  h_count - x -xplus) * ( h_count - x-xplus) + (v_count - 360) * (v_count - 360) <= 15 * 15 ) then
            red   <= "0000";          -- 紅色為 0000
            green <= "1111";          -- 綠色為 1111
            blue  <= "0000";          -- 藍色為 0000
        else
            red   <= "0100";          -- 默認為黑色
            green <= "1000";          -- 默認為黑色
            blue  <= "0011";          -- 默認為黑色
        end if;
    end if;
end process;
```

水平位置　　垂直位置　　圖形

在 VGA 顯示的畫面上根據當前掃描位置　h_count、v_count確定是否顯示綠色圓形。圓心的位置設置為 (h_count-x-xplus, 360)，半徑設置為 20 像素。當掃描的位置在這個圓形的範圍　內時，顯示出一個綠色的圓形。xplu是固定145(要大於H_SYNC_CYCLES + H_FRONT_PORCH並加上球的直徑) 才能將圖形完整顯示在螢幕上

跟新球的水平座標

```
vga_move :process (i_clk , i_rst)
begin
    if i_rst = '0' then
        x <=320;
    elsif led_clk' event and led_clk = '1' then
        case counter_move_state is
            when counter_is_counting_left =>
                x <= x-65;
            when counter_is_counting_right =>
                x <= x+65;
            when right_win =>
                null;
            when left_win =>
                null;
            when left_ready_serve =>
                x <=65;
            when right_ready_serve =>
                x <=540;
            when others =>
                null;
        end case;
    end if;
end process;
```

藉由不同的counter_move_state狀態對X值加減65

counter_is_counting_left:
x座標水平向左移 65

counter_is_counting_right:
x座標水平向右移 65

right_win和left_win :
皆不動作

left_ready_serve:
設定x座標初始值

right_ready_serve:
設定x座標初始值

跟新counter_move_state狀態

```
led_move_state :process (i_clk , i_rst , i_left_button , i_right_button)
begin
    if  i_rst = '0'  then
            counter_move_state <= reserve;
    elsif i_clk' event and i_clk = '1' then
        prestate <= counter_move_state;
        left_button <=  i_left_button;
        right_button <= i_right_button;
        case counter_move_state is
            when counter_is_counting_left =>
                if (count = "10000000") and (i_left_button = '1') then
                    counter_move_state <= counter_is_counting_right;
                elsif (i_left_button = '0' and count = "00000000") or (count<"10000000" and i_left_button='1') then
                    counter_move_state <= right_win;
                end if;
            when counter_is_counting_right =>
                if (count = "00000001") and (i_right_button = '1') then
                    counter_move_state <= counter_is_counting_left;
                elsif (i_right_button = '0' and count = "00000000") or (i_right_button = '1' and count > "00000001") then
                    counter_move_state <= left_win;
                end if;
            when right_win =>
                if count = (left_score(0)&left_score(1)&left_score(2)&left_score(3)) & right_score then
                    counter_move_state <= reserve;
                end if;
            when left_win =>
                if count = (left_score(0)&left_score(1)&left_score(2)&left_score(3)) & right_score then
                    counter_move_state <= reserve;
                end if;
            when left_ready_serve =>
                if count = "10000000" then
                    counter_move_state <= counter_is_counting_right;
                end if;
            when right_ready_serve =>
                if count = "00000001" then
                    counter_move_state <= counter_is_counting_left;
                end if;
            when reserve =>
```

reset為0時進入決定由誰發球的狀態

```
            if i_left_button = '1' and left_button ='0' then
                counter_move_state <= left_ready_serve;
            elsif i_right_button = '1'and right_button ='0' then
                counter_move_state <= right_ready_serve;
            else
                counter_move_state <= reserve;
            end if;
        when others =>
            null;
    end case;
end if;
end process;
```

按鈕按下那邊發球

Mealey machine
確保不會因為按太久導致球直接發出去,需要放再按才能進下一個狀態

搶拍或是漏接都算輸

顯示分數和實際分數相同時進入決定誰發球狀態
PS)最小位元的再最邊邊

說明

```
    -- 時鐘分頻處理（為了產生fclk信號）
  fd:process(i_clk ,i_rst)
begin
if i_rst = '0' then
    divclk <= (others => '0');
elsif rising_edge(i_clk) then
    divclk <= divclk +1 ;
end if;
end process fd;
led_clk <= divclk(24);
fclk <= divclk(1);
```

將快速clk變成一個較慢的 fclk
使用 divclk 計數器來實現除頻

Thanks