HW Lab Final Project

雷射打地鼠：精準射擊挑戰

組長: 109034049 邱向辰　　成員: 111034005 曾柏勳

A. 架構圖

我們的設計包含了兩大 Components，其中包含左邊的 Gun(laser)以及右邊的 Target(Receiver)，結合這兩個 Components 以組成一個似夜市射氣球的遊戲，於下一章節的架構細節將會分別從兩個 components 切入，解釋我們是如何設計出這項 Final Project。

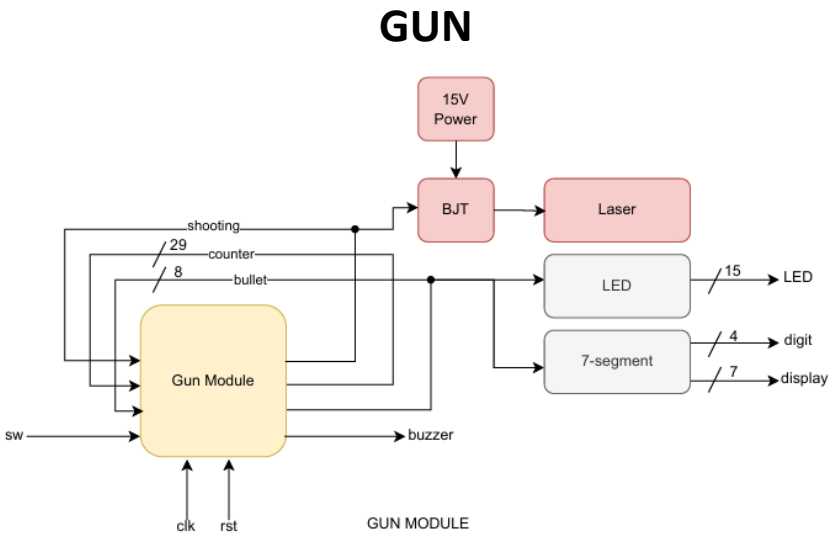
首先列出 Gun 及 Target 有使用到的零組件及 Basys3 上的功能:

Gun	Target
Laser Diode	IR Receivers
Bipolar Junction Transistor (BJT)	VGA
15V Battery	PMod Audio
7-segment	External LED
LED	N/A
Switch	N/A
Buzzer	N/A
Button	N/A

遊戲流程:

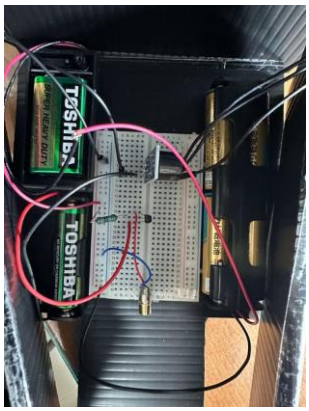
1. 按下開始按鈕以啟動遊戲。
2. 遊戲將進行 3 秒倒數。
3. 當 LED 燈亮起時，開始射擊目標，總共有六回合。
4. 遊戲中最多可錯失 3 回合，超過則遊戲結束。
5. 遊戲將根據你的表現決定勝負結果並顯示勝利/失敗畫面。

B. 架構細節



在 gun 中，我們利用一個 register(sequential block)紀錄所需要儲存的 value，並在每個 cycle 根據不同條件給定 signal。

首先是 SW signal，這是一個 Switch(SW0)的 input signal，作為發射雷射的 trigger，若為 1 時，buzzer 也會被 triggered high，同時 set shooting signal high 用來作為 BJT 的開關，並利用額外的電源去推動雷射光發射。



在 SW 拉為 1 的同時，也會利用 counter 計算剩餘的子彈數(也可以理解為可發射雷射的能量)，在 100Mhz/2^22 的 clock cycle 上更新 counter 及 bullet (100~0)，根據 bullet 決定 LED 及 7-segment 的行為。

- **LED**

```
if(SW) begin
    if(bullet>0) begin
        LED<=16'b1111_1111_1111_1111;
        buzzer<=0;
    end else begin
        LED<=16'b0000_0000_0000_0000;
        buzzer<=1;
    end
    LED_counter<=0;
    first_into_SW0<=1;
end else begin
```

左邊的 code snippet 可以看到，在射擊時(SW==1)，LED 會是全部被 turn on 的狀態，同時 buzzer 會發出聲音。

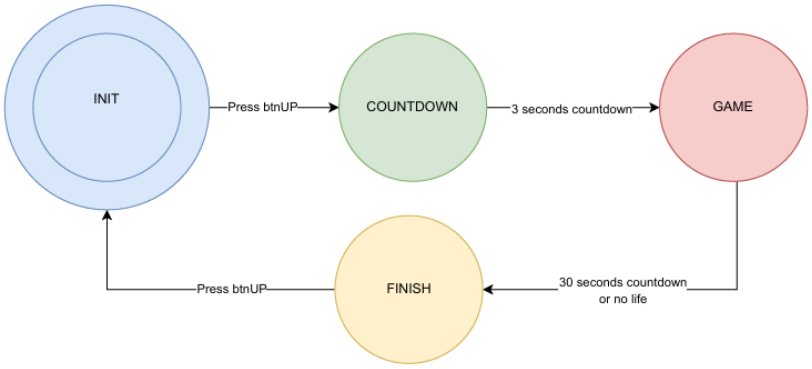
```
end else begin
    buzzer<=1;
    if(first_into_SW0) begin
        if(bullet!=100) begin
            LED<=16'b0000_0000_0000_0111;
            first_into_SW0<=0;
        end else begin
            LED<=16'b0000_0000_0000_0000;
        end
    end
    if(LED_counter<2**23) begin
        LED_counter<=LED_counter+1;
    end else begin
        LED_counter<=0;
        LED<=LED<<1;
        if(LED==16'b0000_0000_0000_0000 &&bullet!=100) begin
            LED<=16'b0000_0000_0000_0111;
        end
    end
end
```

在充能時，LED 則會有移動的特效。

- **7-segment**

在 7-segment 的部分，則是根據當前的 bullet 數量顯示 0~100 的數字做為剩餘能量的展示。

Target



Target Module State Transition

在 target 中，我們設計了如上圖所展示的 FSM 並在每個 state 之中對於 VGA, Audio(PMod)的控制作不同的處理。

我們以遊戲流程去觀察這個 state transition diagram 並說明在不同的 state 中分別作出不同的信號控制。

- INIT

在 INIT state 中，會重製所有紀錄的信號，並且在收到 btnUP 後 jump 到 COUNTDOWN state。此時螢幕上(VGA output)會顯示出遊戲開始的畫面，並撥放初始音樂。

```
if(current_state==INIT) begin
    countdown<=9'd6;
    counter<=0;
    light1<=0;
    light2<=0;
    light3<=0;
    light1_lock<=0;
    light2_lock<=0;
    light3_lock<=0;
    sequence<=0;
    life<=3;
    life_sound<=0;
    COUNTDOWN_count<=0;
    COUNTDOWN_time<=3;
```

在 INIT state reset 所有會用到的信號，其中 light1, light2, light3 為 target(receiver + led 的組合)，sequence 為遊戲回合數，life 為記錄生命的 reg (總共可以 miss 3 回合)

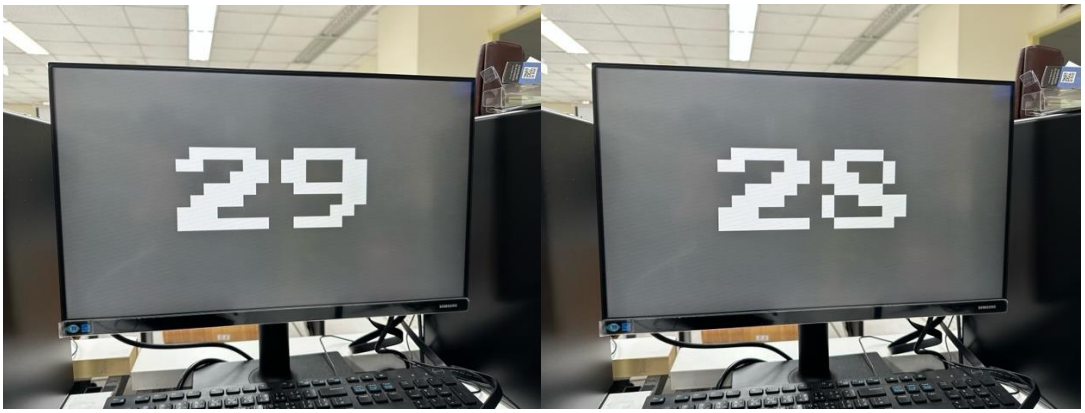
- COUNTDOWN

在 COUNTDOWN state 中，則會倒數三秒，結束後跳至 GAME state，在這個 state 之中也會有獨立的 VGA 畫面以及不同的音樂倒數。

下圖為倒數三秒時的部分 beat signals。

```
if(COUNTDOWN_time==3) begin
    case(ibeatNum_COUNTDOWN)
        12'd0: begin freqL = `a3; freqR = `a3; end
        12'd1: begin freqL = `a3; freqR = `a3; end
        12'd2: begin freqL = `a3; freqR = `a3; end
        12'd3: begin freqL = `a3; freqR = `a3; end
        12'd4: begin freqL = `a3; freqR = `a3; end
        12'd5: begin freqL = `a3; freqR = `a3; end
        12'd6: begin freqL = `a3; freqR = `a3; end
        12'd7: begin freqL = `a3; freqR = `a3; end
        12'd8: begin freqL = `a3; freqR = `a3; end
        12'd9: begin freqL = `a3; freqR = `a3; end
end else if(COUNTDOWN_time==2) begin
    case(ibeatNum_COUNTDOWN)
        12'd0: begin freqL = `a3; freqR = `a3; end
        12'd1: begin freqL = `a3; freqR = `a3; end
        12'd2: begin freqL = `a3; freqR = `a3; end
        12'd3: begin freqL = `a3; freqR = `a3; end
        12'd4: begin freqL = `a3; freqR = `a3; end
        12'd5: begin freqL = `a3; freqR = `a3; end
        12'd6: begin freqL = `a3; freqR = `a3; end
        12'd7: begin freqL = `a3; freqR = `a3; end
        12'd8: begin freqL = `a3; freqR = `a3; end
        12'd9: begin freqL = `a3; freqR = `a3; end
end else if(COUNTDOWN_time==1) begin
    case(ibeatNum_COUNTDOWN)
        12'd0: begin freqL = `a4; freqR = `a4; end
        12'd1: begin freqL = `a4; freqR = `a4; end
        12'd2: begin freqL = `a4; freqR = `a4; end
        12'd3: begin freqL = `a4; freqR = `a4; end
        12'd4: begin freqL = `a4; freqR = `a4; end
        12'd5: begin freqL = `a4; freqR = `a4; end
        12'd6: begin freqL = `a4; freqR = `a4; end
        12'd7: begin freqL = `a4; freqR = `a4; end
        12'd8: begin freqL = `a4; freqR = `a4; end
        12'd9: begin freqL = `a4; freqR = `a4; end
```

- GAME



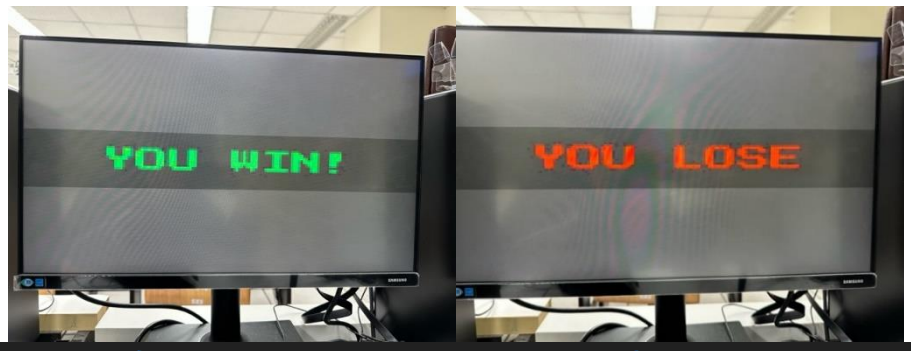
在 GAME state 中，三組 Receivers + External LED 的組合會以我們設計好的順序，展示總共 5 秒*6 回合總共 30 秒的遊戲時間，同時間 VGA 也會顯示從 30 秒開始倒數的畫面。

```
if(counter<2**29) begin // < 4sec
    counter<=counter+1;
    if(sequence==0) begin
        if(light1_lock==0) begin
            light1<=0;
        end
        if(light2_lock==0) begin
            light2<=0;
        end
        if(light3_lock==0) begin
            light3<=1;
        end
    end else if(sequence==1) begin
        if(light1_lock==0) begin
            light1<=1;
        end
        if(light2_lock==0) begin
            light2<=0;
        end
        if(light3_lock==0) begin
            light3<=0;
        end
```

```
end else if(sequence==2) begin
    if(light1_lock==0) begin
        light1<=1;
    end
    if(light2_lock==0) begin
        light2<=1;
    end
    if(light3_lock==0) begin
        light3<=0;
    end
end else if(sequence==3) begin
    if(light1_lock==0) begin
        light1<=1;
    end
    if(light2_lock==0) begin
        light2<=1;
    end
    if(light3_lock==0) begin
        light3<=1;
    end
```

這邊展示前四個回合時，三組 Receivers 會顯示出應該要射擊的順序分別為(0, 0, 1), (1, 0, 0), (1, 1, 0), 及(1, 1, 1)

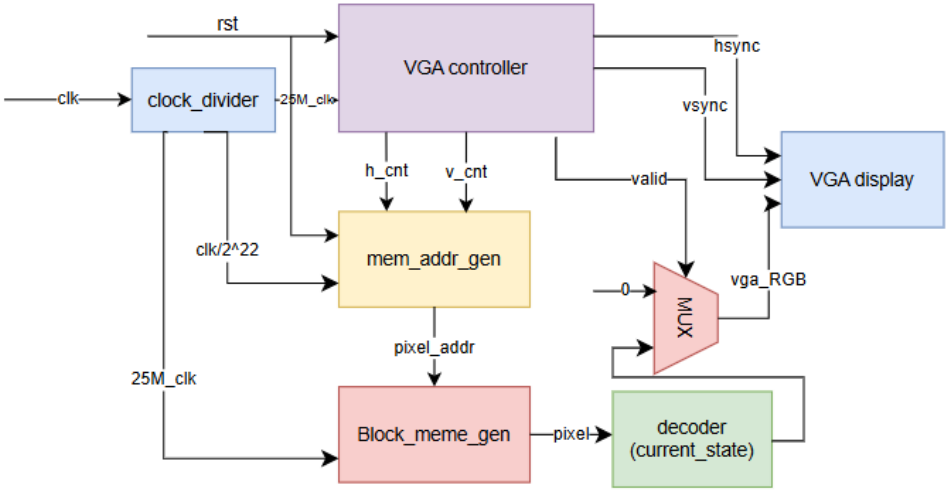
- FINISH




```
end else if (current_state==FINISH && life!=0) begin
    {vgaRed, vgaGreen, vgaBlue} = (valid==1'b1) ? pixel_win:12'h0;
    //{vgaRed, vgaGreen, vgaBlue} = (valid==1'b1) ? 12'h0f0:12'h0;
end else if (current_state==FINISH && life==0) begin
    {vgaRed, vgaGreen, vgaBlue} = (valid==1'b1) ? pixel_lose:12'h0;
```

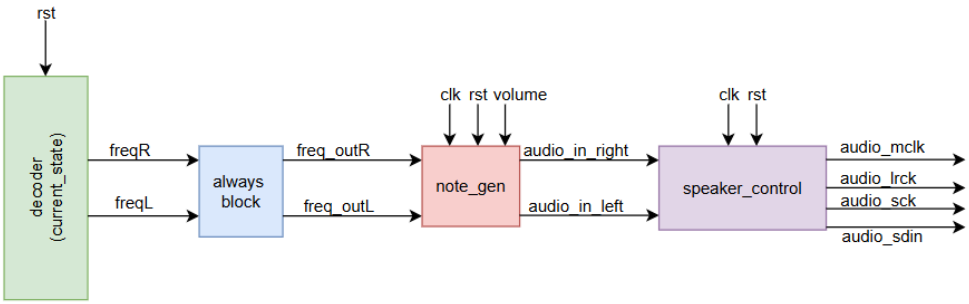
在 FINISH state 中根據 life 判斷玩家是否贏得遊戲，並展示出不同的結算畫面。

- VGA block diagram



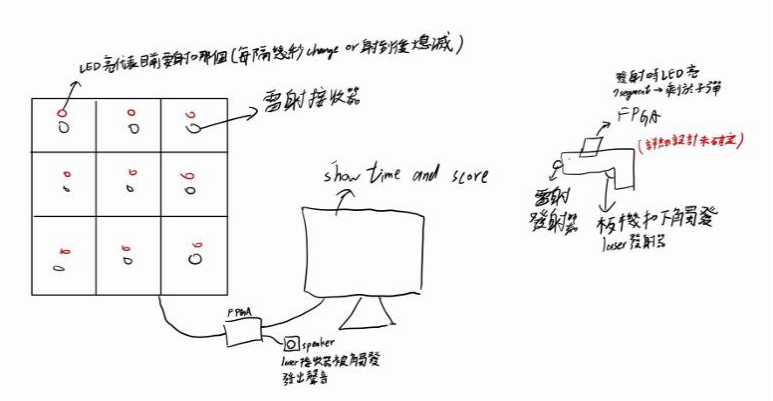
```
end else if (current_state==GAME) begin
    {vgaRed, vgaGreen, vgaBlue} = (valid==1'b1) ? pixel:12'h0;
end else if (current_state==FINISH && life!=0) begin
    {vgaRed, vgaGreen, vgaBlue} = (valid==1'b1) ? pixel_win:12'h0;
    //{vgaRed, vgaGreen, vgaBlue} = (valid==1'b1) ? 12'h0f0:12'h0;
end else if (current_state==FINISH && life==0) begin
    {vgaRed, vgaGreen, vgaBlue} = (valid==1'b1) ? pixel_lose:12'h0;
end else begin
    {vgaRed, vgaGreen, vgaBlue} = (valid==1'b1) ? 12'hfff:12'h0;
end
```

- Audio block diagram




```
end else begin
    case (current_state)
        INIT: begin
            freqL = `silence;
            freqR = `silence;
        end
        COUNTDOWN: begin
            if(COUNTDOWN_time==3) begin
```

C. 實作完程度 & 難易度說明



上圖為我們在 proposal 中想要呈現的大致示意圖，我們除了因為接線複雜度及電源供應問題將 target(IR receiver)的數量減為 3 個以及在槍械的物理結構上做調整外，其餘目標皆有完成。

此外，我們額外新增了 buzze; 不同 state 中客制化的 audio 以及 VGA 輸出; 透過 BJT 以額外電源驅動 laser 模組等規格外作法，考量到上述原因，我們認為這次的 project 具有高完成度及難度:

實作完成度: 95 / 100

難易度: 95 / 100

D. 困難&解決方法

最困難的部分我們認為是在硬體的部分，因為 FPGA 所供的電流不足以讓雷射頭亮，所以我們上網查了許多的解決方法，一開始先試了接一顆 MOS，結果不知道為什麼還是亮不了，後來詢問了其他人才知道還可以接 BJT。接完之後雖然會亮，但還是沒有達到我們期望的亮度，之後利用了 FPGA3.3V output 當作 BJT 的 Switch，串流 15V 的外接電源，就成功解決了這個問題。

還有一個困難的部分是 VGA，在做 GAME state 的時候我們想讓 VGA 呈現 30~1 的倒數畫面，一開始有點不確定應該如何實現，測試過各種方法後我們繪製了一個 5x6 的表格並在每格中央放上數字，最後再用 mapping 的方式將該格在固定的時間之內放到整個 VGA output 的螢幕上。

E. 心得討論

這次的 final project 從題目發想到實作完成，我們經歷了許多挑戰。首先，我們需要規劃一個既能在能力範圍內實現，又能最大化成果的方案。在實作過程中，我們自行摸索了 laser diode、BJT、外部電源以及 IR receivers 的使用方法，並一步步克服了各種技術難題。為了提升整體專案的完成度，我們甚至實現了許多原本不在規劃內的功能。

這段旅程不僅讓我們學到了許多實用技能，也帶給我們極大的成就感。從零開始構築一個創意十足的專案，過程中的每一步都充滿了挑戰與收穫。我們衷心感謝老師與助教這學期的悉心指導，也感謝自己在過去半年中不斷累積的實作經驗，為這次專案的順利完成奠定了基礎。

F. 詳細分工

- 邱向辰: Laser diode + BJT, VGA, LED, Report
- 曾柏勳: Target (IR receiver + LEDs), switch, LED, audio, VGA, 組裝

G. 課程外部分

如先前所述，我們獨自探索了 IR receivers 以及 laser diode + BJT 的組合，後者的部分要感謝組長邱向辰的哥哥提供技術支援，提供了以 FPGA 電壓輸出做為 BJT 開關並串接到 15V 電池做為 laser 驅動的想法，其餘部份我們僅參考這學期 lab1~lab6 的內容並獨力製作完畢，並無使用任何網路來源的程式碼或是先前自己開發過的專案。