



DART GAME

Logic Design Lab Final Project



JANUARY 14, 2022

TEAM9

108062213 顏浩昀 106062304 黃鈺舒

I. Introduction

本次 project 目的是應用這學期上課所學，做出結合 fpga 的飛鏢射擊遊戲。規則參考飛鏢遊戲的 01game 與輪流計分，而射擊區域則以難易度分類，由中心往外分為紅心·藍心·一倍·兩倍·三倍區，實作出的主要規則如下：

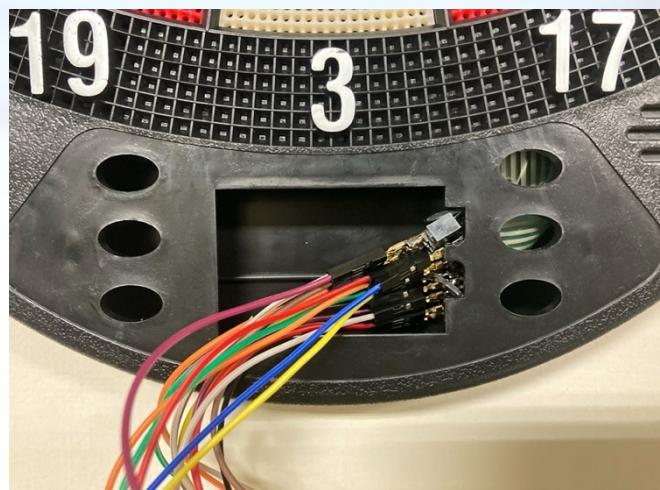
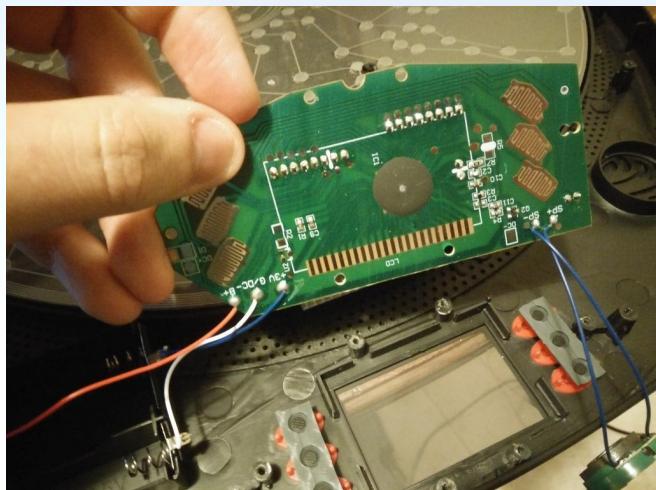
- 1.01game: 最後射擊要使分數剛好歸零，如果射擊分數超過目前剩餘分數，則分數不變。
2. 輪流射擊：雙方玩家一開始有相同初始分數，一人一回合射擊三次換下一人，最先使分數歸零的玩家獲勝。
3. 倍數區域：射擊得分根據難易度分為若干區塊，擊中規定區域之內則得到該分數。

II. Motivation

1. 研究動機：經過一學期的邏輯設計實驗課程後，我們希望能在期末繳出一份有趣且充滿互動的成果發表。參考過往未完成的設計構想，我們選擇投擲類遊戲作為主軸，並且因為在網路上有成功透過其他方式自製飛鏢靶的設計經驗，我們選擇飛鏢遊戲來作為期末報告的主題。
2. 預期目標：完成一套完整的雙人飛鏢遊戲流程。其中功能包含：計分、切換玩家、顯示目前玩家、判定勝利方並顯示勝利畫面。

III. System specification

1. Hardware implementation



(1) 將電子飛鏢靶拆解，將內部的薄膜感應電路由原本的 PCB 板上解焊，並重新與杜邦線焊接，並藉由 Pmod 接孔與 FPGA 板連接。

(2) 經過測試，訊號傳遞的規則如下：

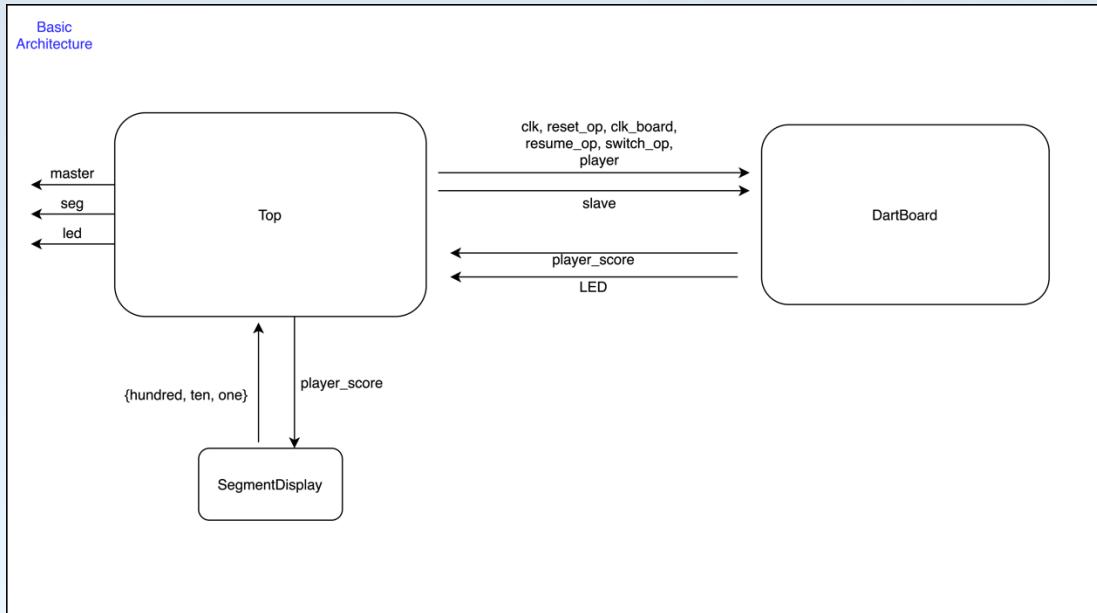
飛鏢靶總共有 10×2 個扇形區塊與內、外紅心。我們透過 master 與 slave 兩個 array 判斷射中的區域。master 為 10bits output array，每個 bit 可判斷的分數區塊為兩組扇形（其中兩個 master 與紅心控制相關），將所有的 master 設為高電位(1)，並輪流將不同 bit 設為低電位(0)來給予 slave 變化。slave 則對應到每兩個分數區塊的子區塊，一個區塊有三塊設有感應器（分別為一倍、兩倍、三倍區），另外加上正中心的區塊，slave 感應的區域總共有七塊，在被 master 觸碰到時，會因為電位差異，產生相對應的 slave array(7 bits input array)。透過實驗與測試 master 與 slave 的對應關係，我們可以知道不同 master 值與 slave 值組合判斷飛鏢擊中之區域，再去計算分數。

分數	slave[0]=0	slave[1]=0	slave[2]=0	slave[3]=0	slave[4]=0	slave[5]=0	slave[6]=0
master [0] = 0	7 三倍	7 雙倍	7 單倍		1 單倍	1 雙倍	1 三倍
master [1] = 0	19 三倍	19 雙倍	19 單倍		18 單倍	18 雙倍	18 三倍
master [2] = 0	3 三倍	3 雙倍	3 單倍		4 單倍	4 雙倍	4 三倍
master [3] = 0	17 三倍	17 雙倍	17 單倍		13 單倍	13 雙倍	13 三倍
master [4] = 0	2 三倍	2 雙倍	2 單倍		6 單倍	6 雙倍	6 三倍
master [5] = 0	15 三倍	15 雙倍	15 單倍		10 單倍	10 雙倍	10 三倍
master [6] = 0	16 三倍	16 雙倍	16 單倍		20 單倍	20 雙倍	20 三倍
master [7] = 0	8 三倍	8 雙倍	8 單倍		5 單倍	5 雙倍	5 三倍

master [8] = 0	11 三倍	11 雙倍	11 單倍	25 (藍心)	12 單倍	12 雙倍	12 三倍
master [9] = 0	14 三倍	14 雙倍	14 單倍	50 (紅心)	9 單倍	9 雙倍	9 三倍

4. Code implementation

(1) Top module



Top module 用於製作不同的 clock、FPGA 板上按鈕的 debounce 與 onepulse、並連接各個 module。另外，Top module 還負責兩個 player 的轉換，在按下 switch button 後會將 $\text{player} = \sim \text{player}$ ，並且 player 會與分數顯示、LED 燈號顯示、玩家分數計算息息相關。

```
2'b00:begin
    seg = (player0_score==10'd0)? 8'b11110101 : out_0[7:0]; //r
    an = 4'b1110;
end
2'b01:begin
    if(player0_score<10'd10 && player0_score>10'd0) seg = 8'b11111111;
    else if (player0_score==8'd0) seg = 8'b11110011; //l
    else seg = out_0[15:8];
    an = 4'b1101;
end
2'b10:begin
    seg = (player0_score==10'd0)? 8'b11100101 : out_0[23:16]; //c
    an = 4'b1011;
end
```

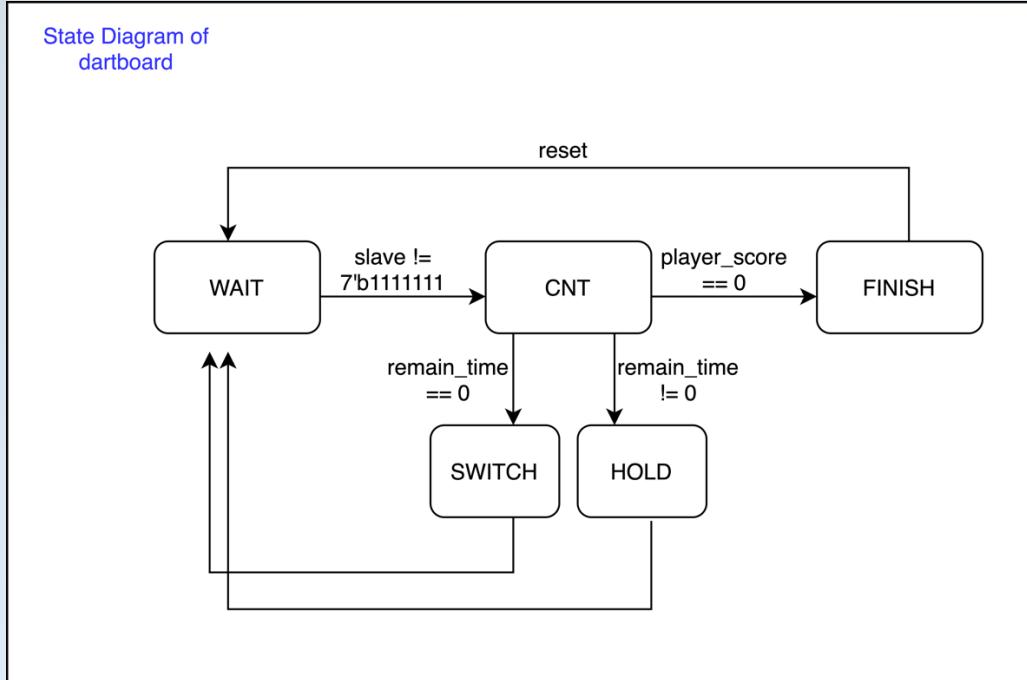
在 7 segment 顯示部分，會透過 segmentDisplay module 將 player score 轉為 7 segment 顯示，並且特別在玩家分數歸零時，以 clr 表示 clear 成為勝利顯示方式，並同時亮起 score 歸零那邊的 player LED。

(2) dartboard module

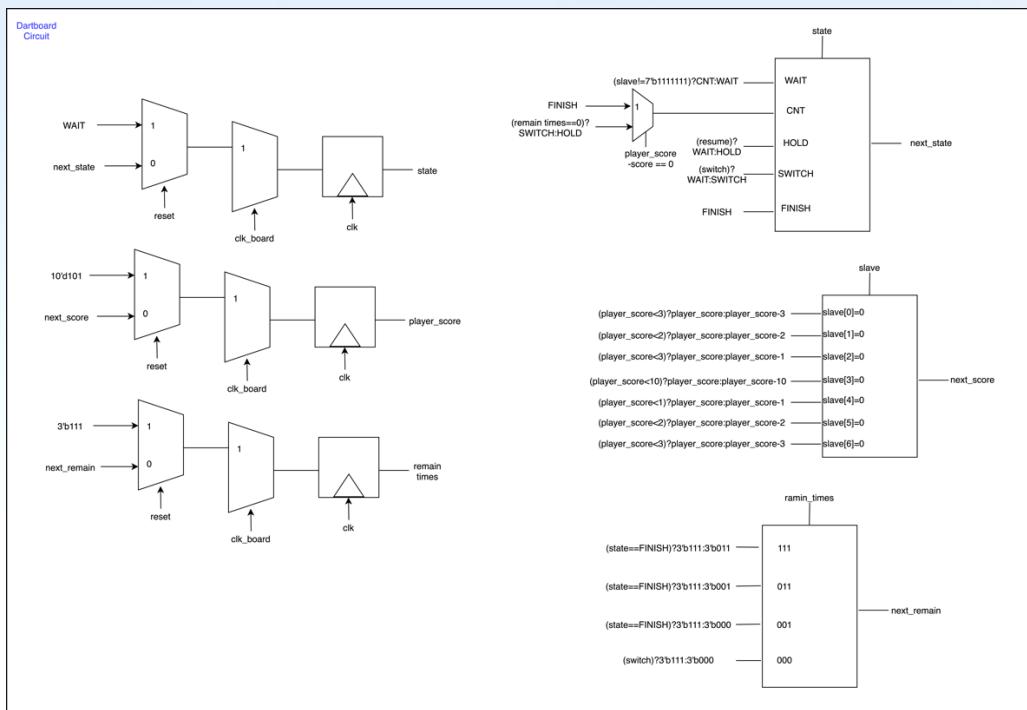
這個 module 是負責 dartboard 感應以及分數計算。

```
dartboard player_0( .clk(clk), .clk_board(clk22), .reset(rst_op), .switch(switch_op), .resume(resume_op), .slave(slave), .player(player),
                    .player_score(player0_score), .remain_times(led_tmp[2:0]) );
dartboard player_1( .clk(clk), .clk_board(clk22), .reset(rst_op), .switch(switch_op), .resume(resume_op), .slave(slave), .player(~player),
                    .player_score(player1_score), .remain_times(led_tmp[5:3]) );
```

我們將每個 player 的分數與任何操作獨立出來，因此在傳入此 module 的 input 有 player 這個 signal，並且在 top module 中會使用兩個 dartboard module，完成比較有系統性的架構。



Dartboard module 實作中，總共有五個 state，分別為：WAIT、CNT、HOLD、SWITCH、FINISH。



WAIT state 是在等待飛鏢射出的 state，我們透過判斷 slave 有任一 bit 變為 0 時，得知有飛鏢碰觸到標靶，並判定該鏢分數，並進入 CNT state。

```

if(player) begin
    case(state)
        WAIT:begin
            next_score = player_score;
            next_remain = remain_times;
            if(slave==7'b1011111)begin//5
                next_state = CNT;
                score = 10'd2;
            end
            else if(slave==7'b1111101)begin//1
                next_state = CNT;
                score = 10'd2;
            end
            else if(slave==7'b1101111)begin//4
                next_state = CNT;
                score = 10'd1;
            end
            else if(slave==7'b1111011)begin//2
                next_state = CNT;
                score = 10'd1;
            end
            else if(slave==7'b1110111)begin//3
                next_state = CNT;
                score = 10'd10;
            end
            else if(slave==7'b1111111)begin//6
                next_state = CNT;
                score = 10'd3;
            end
            else if(slave==7'b1111110)begin//0
                next_state = CNT;
                score = 10'd3;
            end
            else begin
                next_state = WAIT;
                score = 10'd0;
            end
        end
    end
end

```

```

if(remain_times==3'b111) begin
    next_remain = 3'b011;
    next_state = HOLD;
end
else if(remain_times==3'b011) begin
    next_remain = 3'b001;
    next_state = HOLD;
end
else if(remain_times==3'b001) begin
    next_remain = 3'b000;
    next_state = SWITCH;
end

```

在

CNT state 中，首先要先計算射完該鏢後的剩餘分數，如果分數歸零就會進入 FINISH state，若沒有歸零則遊戲繼續；在分數沒有歸零的情形下，我們又分為兩個情形。根據常見的飛鏢規則，通常每個玩家可以在一輪中投擲三隻飛鏢，因此我們使用 remain times 這個變數來判斷這名 player 還有多少次投擲機會。如果投擲機會還有剩餘，會進入 HOLD state，並且透過 FPGA 板上的 LED 燈顯示剩餘機會；如果三次投擲完畢，LED 燈就會全數熄滅，並且進入 SWITCH state。

HOLD state 的使用是因為 input 的 slave signal 訊號比較不穩定，如果飛鏢靶受到叫強力的碰撞、晃動，很容易造成板子上的方塊壓到薄膜，導致連續得分。為了避免這個情形發生，我們在每次得分後就進入 HOLD state，需要按下一顆 button 才可以回到 WAIT state 繼續遊戲。

SWITCH state 則是三枚飛鏢全部投擲完畢後，準備進行人員更換而準備的 state。在按下指定 button 後 Top module 會改變 player，此時的分數計算已經不再與這邊有所相關，但會重設這名玩家的 remain times。

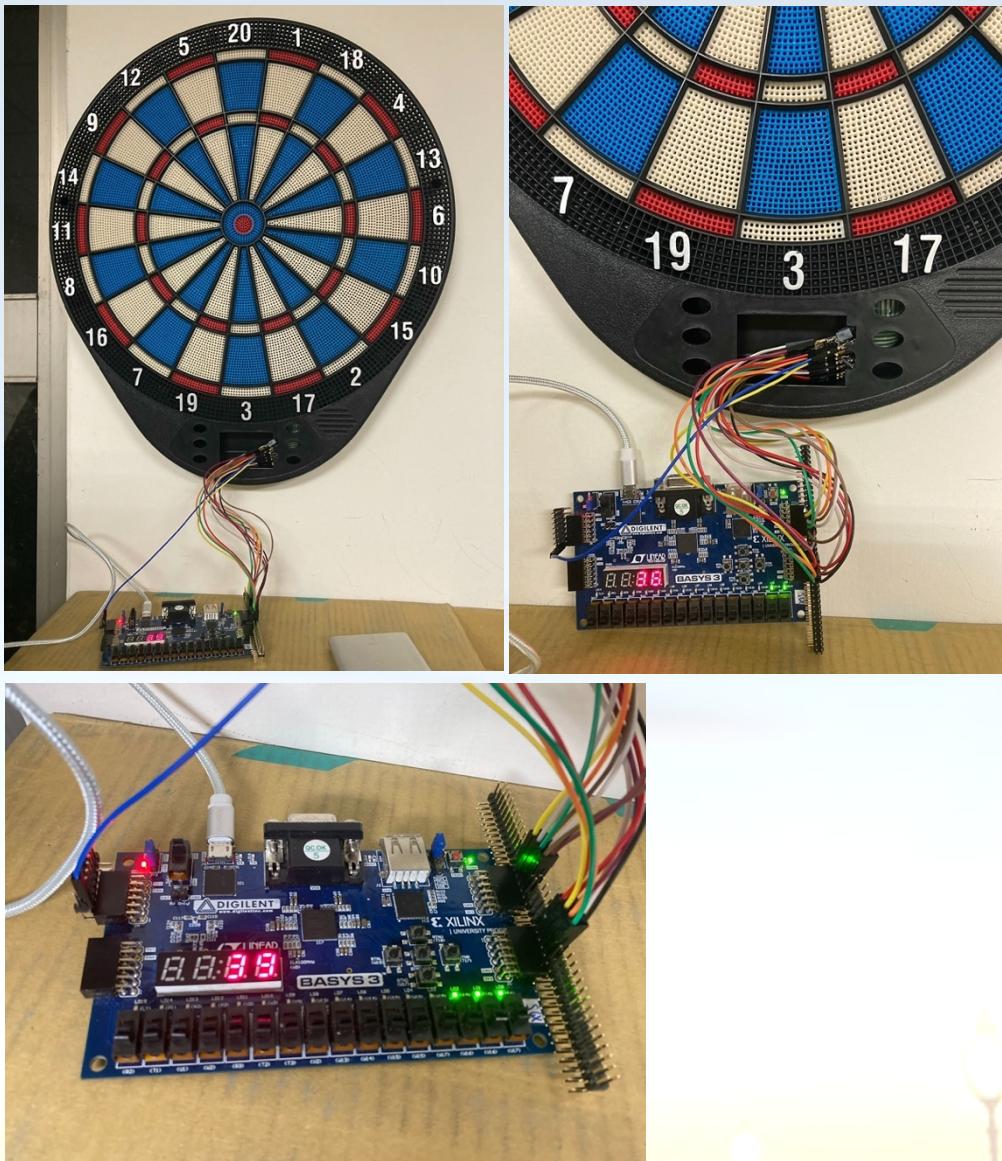
FINISH state 是玩家分數歸零時進入的 state，特別注意的是如果射出的分數超出剩餘分數是不會 FINISH 的，分數會是維持原本不做改變。FINISH state 會將勝利方的 remain times 設為 3'b111，這是因為 LED 燈號顯示是透過 remain times 判斷，為了顯示勝方的 LED 燈特別進行此處理。

IV. Experiment results

1. 經過實測，電子器材中的 PCB 板訊號能成功 program 至 FPGA 板，兩片薄膜電路也可以成功感測訊號，至於傳輸穩定度則取決於焊接部分的完整程度。

2. master 訊號會影響 slave 訊號，若電路跳動頻繁或在遊戲中震動到標靶內部元件，會影響 FPGA 接收到跳動指令，但事實上並無擊中該區域，因此應固定標靶，並設置按鈕防止 FPGA 多次扣分，影響遊戲體驗。

3. Presentation



1. Review

這次實作我們成功將 FPGA 連結至標靶，了解到硬體內部控制的機制並實做了接線的流程，並將感應結合程式，相較於以往的程式實作，能實際做出硬體是我們最大的收穫，過程中也遭遇了許多困難，從一開始由拆解 PCB 板焊接電路到 FPGA，甚至是還毀掉了第一塊飛鏢靶，到一步一步實驗測試 master 與 slave 薄膜電路的關係，尤其是如何消除雜訊，提升接收訊號的準確度，是我們一直努力的方向。

2. Future

這次比較可惜的是因為 slave 這個 input 訊號太過不穩定，尤其在 master 輪流轉換時很難準確的判斷 slave 與 master 被接觸的組合。雖然多次嘗試 debounce slave 訊號，也曾經參考 keyboard sample code 中 debounce 的方法，但最終都無功而返。為了讓 demo 時有較好的遊戲呈現，我們將 master 改為固定，讓 slave 訊號較為穩定，但就較缺乏計分的完整度，是希望以後有機會能夠改善的地方。

VI. Reference

1. <https://create.arduino.cc/projecthub/ricardo-alves/opendarts-homemade-dartboard-machine-2a2914>
2. <https://digilent.com/reference/basys3/refmanual>
3. https://www.youtube.com/watch?v=lUwSO_UY72Y
4. <https://www.networktechinc.com/ps2-prots.html>