Final project report
106061151 劉安得 106061227 林柏辰

規則說明：

遊戲開始兩名玩家各有 50 分（七段顯示器顯示），射出飛鏢後扣掉飛鏢射中的得分，在七輪遊戲內先將分數歸零者獲勝。射飛鏢的方法為用鍵盤進行上（W）下（S）左（A）右（D）的瞄準（標靶顯示於螢幕），接著按住按鈕開始蓄力（LED 燈顯示力道大小），放開時飛鏢射出。一名玩家射完後按按鈕切換玩家。

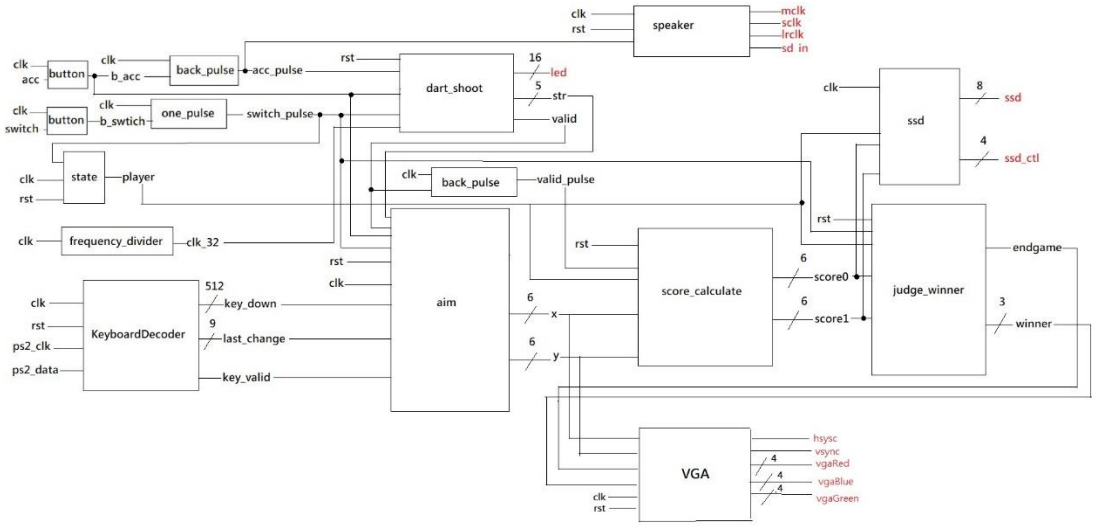
分工：

劉安得: VGA 顯示部分
林柏辰: 七段顯示器、LED、按鈕、鍵盤及提供變數給 VGA 判斷

1. Design Specification

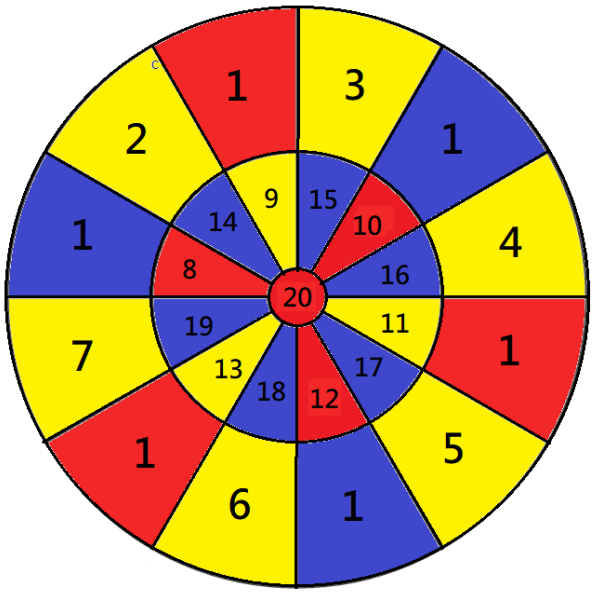
Input:
clk,
rst,
switch,// button 切換玩家
acc// button 蓄力
Inout:
ps2_data,
ps2_clk
Output:
[15:0] led,// LED 顯示蓄力條
[7:0] ssd,//7-segment display 顯示玩家目前分數
[3:0] ssd_ctl,
[3:0] vgaRed,// VGA 顯示標靶及瞄準/命中處
[3:0] vgaGreen,
[3:0] vgaBlue,
hsync,
vsync,
mclk,// speaker 播放飛鏢命中音效
lrclk,
sclk,
sd_in

2. Block diagram



(紅字為 output)

3. Design Implementation



（本次使用的飛鏢圖）

這次 Final project 所需的 module 為：

基本輸入處理：

button: 為 button 訊號的 debouncer，產生的 output 有 b_switch, b_acc

one_pulse: 將 debounce 過的 button 訊號轉成一個 clock cycle 的 pulse，為按下按鈕時產生，產生的 output 有 switch_pulse

back_pulse: 類似 one_pulse 但在按鈕放開時產生 pulse，產生的 output 有

acc_pulse, valid_pulse

frequency_divider: 產生 32Hz 的 clk，此為 LED 燈變動的速度

KeyboardDecoder: 鍵盤輸入訊號處理，是之前老師給的 source code

state: FSM，指示目前玩家

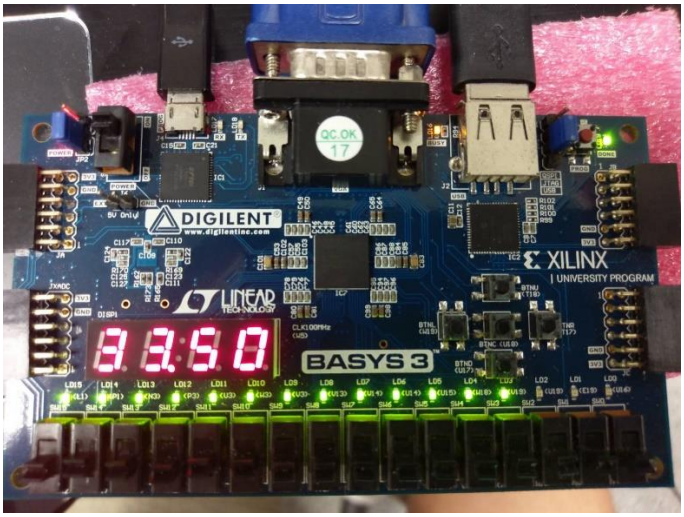
主要功能:

dart_shoot: 蓄力射飛鏢的 module。產生代表蓄力條的 LED output、相對應的力道大小 str(值為 0-16)，以及代表瞄準(1)/射出(0)狀態的 valid

aim: 產生瞄準時的預計落點已及射出後的命中位置。x 座標由鍵盤秒準的結果以及亂數產生-2~2 的誤差決定；y 座標由鍵盤瞄準結果以及蓄力的結果決定。標靶為半徑 10 單位的正圓，鍵盤每次可往上/下調整一單位，或是左/右調整兩單位。

score_calculate: 將(x, y)座標對應到相應的得分，產生的 output 為兩位玩家目前的剩餘分數

ssd: 將兩位玩家目前的分數顯示於七段顯示器，左邊兩個 digits 為玩家 1；右邊兩個 digits 為玩家 2。另外，分數右下角的點如果亮起代表輪到該玩家



judge_winner: 判斷玩家的輸贏。遊戲為玩家 1 先射，判斷規則有：

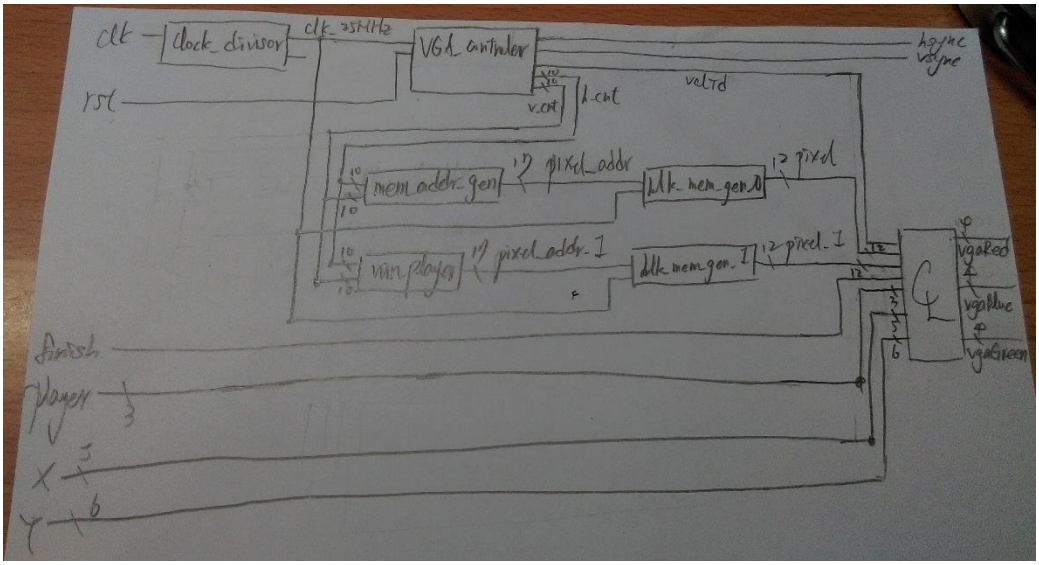
1. 經過七輪遊戲未分出勝負，則剩餘分數較低者獲勝，若一樣則平手。
2. 當玩家 1 射出後分數歸零，玩家 2 剩餘分數在 20 分以上，則宣告玩家 1 獲勝。
3. 當玩家 1 射出後分數歸零，玩家 2 分數小於 20 分，則輪到玩家 2；若玩家 2 射出後分數未歸零，則宣告玩家 1 獲勝；若玩家 2 射出後分數歸零則平手。
4. 若玩家 2 射出後分數歸零，玩家 1 分數不為零，宣告玩

家二獲勝。

speaker: 飛鏢射出後播放命中音效

VGA 部分:

因為似乎放不下兩張 320*240，而且我遊戲結束的圖片有三張，於是我決定把遊戲結束的圖片再縮小 1/2，並垂直疊起來一起讀取變成 160*360，所以我總共讀了一張 320*240 和 160*360



Clock_divisor: 除頻器，提供 25Hz 給 memory IP 使用

Mem_addr_div: 當顯示標靶時，產出要讀取的 address

Win_player: 當遊戲結束時，根據輸贏判斷顯示哪張圖片。當 player 為 001 時代表玩家一勝出，010 代表玩家二勝出，100 代表玩家三勝出

Vga_controler: 產出 h_cnt, v_cnt, hsync, vsync, valid

I/O	Led[15]	Led[14]	Led[13]	Led[12]	Led[11]	Led[10]	Led[9]	Led[8]
Pin	L1	P1	N3	P3	U3	W3	V3	V13

I/O	Led[7]	Led[6]	Led[5]	Led[4]	Led[3]	Led[2]	Led[1]	Led[0]
Pin	V14	U14	U15	W18	V19	U19	E19	U16

I/O	Ssd[7]	Ssd[6]	Ssd[5]	Ssd[4]	Ssd[3]	Ssd[2]	Ssd[1]	Ssd[0]

Pin	W7	W6	U8	V8	U5	V5	U7	V7

I/O	Ssd_ctl[3]	Ssd_ctl[2]	Ssd_ctl[1]	Ssd_ctl[0]
Pin	W4	V4	U4	U2

I/O	vgaRed[3]	vgaRed[2]	vgaRed[1]	vgaRed[0]
Pin	N19	J19	H19	G19

I/O	vgaBlue[3]	vgaBlue[2]	vgaBlue[1]	vgaBlue[0]
Pin	J18	K18	L18	N18

I/O	vgaGreen[3]	vgaGreen[2]	vgaGreen[1]	vgaGreen
Pin	D17	G17	H17	J17

I/O	acc	show	Switch	Rst	clk	hsync	vsync
Pin	T17	V16	T18	V17	W5	P19	R19

I/O	mclk	Lrclk	Sclk	Sd_in	Ps2_clk	Ps2_data
Pin	A14	A16	P19	B16	C17	B17

4. Discussion

我們的螢幕會有些破圖，我們找不到原因，並且重跑 bitstream 後每次破的位置都不同。一開始我們有試著顯示看看，這時候比較沒有破圖，所以我覺得可能跟複雜度有關，當 project 越複雜越容易破圖，但我們還是找不到原因

一開始我們打算顯示四張圖片，但卻發現連顯示兩張都記憶體不夠。最

後，想到縮小再放大的方法，解決記憶體不夠的問題。

5. Conclusion

這次 final project 讓我們學到如何從零開始的做出一個成品。我們要從一開始的規格開始設計，不再像之前的 lab 一開始就有 specifIcation。再過的過程中也會不斷的新增功能或改良，不斷的嘗試。