Final Project Report

-Night Market Pinball-

組別：Team33　學號：110062123、110062231
姓名：陳樂穎、鐘萱容

I. Introduction

Pinball，即彈珠台。最早期的美式彈珠台為透過側邊的彈簧彈出一顆金屬小球，讓小球進入矩任意彈跳，並依據經過的地點得分。當球往下掉的時候，玩家可以操控最底端的板子將小球打回去、繼續彈跳，也是我們起初要做的樣子。

但因為重力和摩擦力等問題，我們決定改成呈現出台灣夜市彈珠台的形式，由側邊發射出金屬球，經過場面上和釘子的碰撞到最後掉入的洞口決定得分與球數。

II. Motivation

身為夜市小遊戲愛好者一定都夢想過家裡就有一台遊戲機隨時都可以玩，綜觀所有遊戲機台，我們認為彈珠台最能和我們所學"FSM"的概念結合，於是我們就往這個方向尋找靈感。

起初我們的構想是可以實體化還原以前舊電腦太空彈珠台內建遊戲(圖一)，但後來發現重力會讓球掉落速度太快、還有底部板子施力不夠、兩側障礙物彈力不足等問題，所以我們轉往實體的彈珠台。第二個構想是還原夜市的彈珠台(圖二)，但是後來也發現不太可行因為這種機台會一直掉出新的球，考慮我們沒有無限顆球，所以最後決定更改一點遊戲規則：購買沒有任何功能的彈珠台底座(圖三)，由我們自己增加功能，亂數指定的洞口由原本掉出新球改成增加分數，加上音效、計分、亂數的功能。

圖一

圖二

圖三

III. System Specification

遊戲規則：

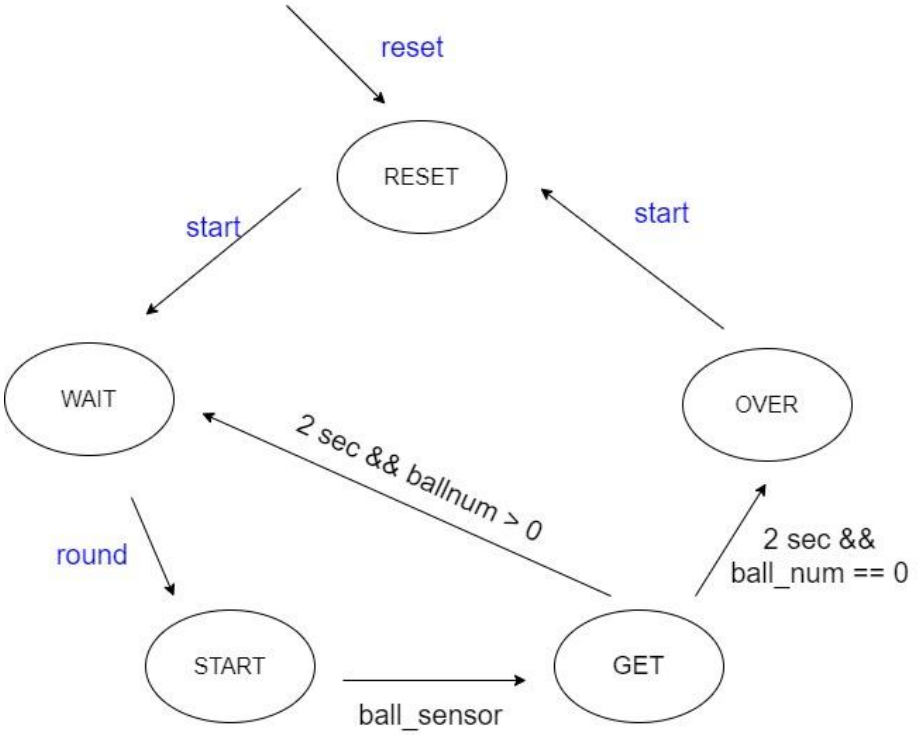
1. 初始狀態為RESET，按下start進入遊戲
2. 進入遊戲後LED燈亂數閃爍，直到按下round進入發射彈珠階段(WAIT)
3. 按下round後LED停止亂數並恆亮指定洞口，只有通過指定洞口才有得分(START)
4. 球滾過後(GET)-按下round或等待兩秒回到發射階段(WAIT)->回到步驟二
   -若球數歸零，則遊戲結束至OVER
5. 若得分超過100為勝利，反之為失敗。

得分機制：

每次遊戲都會有八顆球，FPGA版上的LED顯示數量即為球數：
四個洞 + 10分、三個洞 + 20分、兩個洞 + 30分、一個洞 + 40分

IV. Design Detail

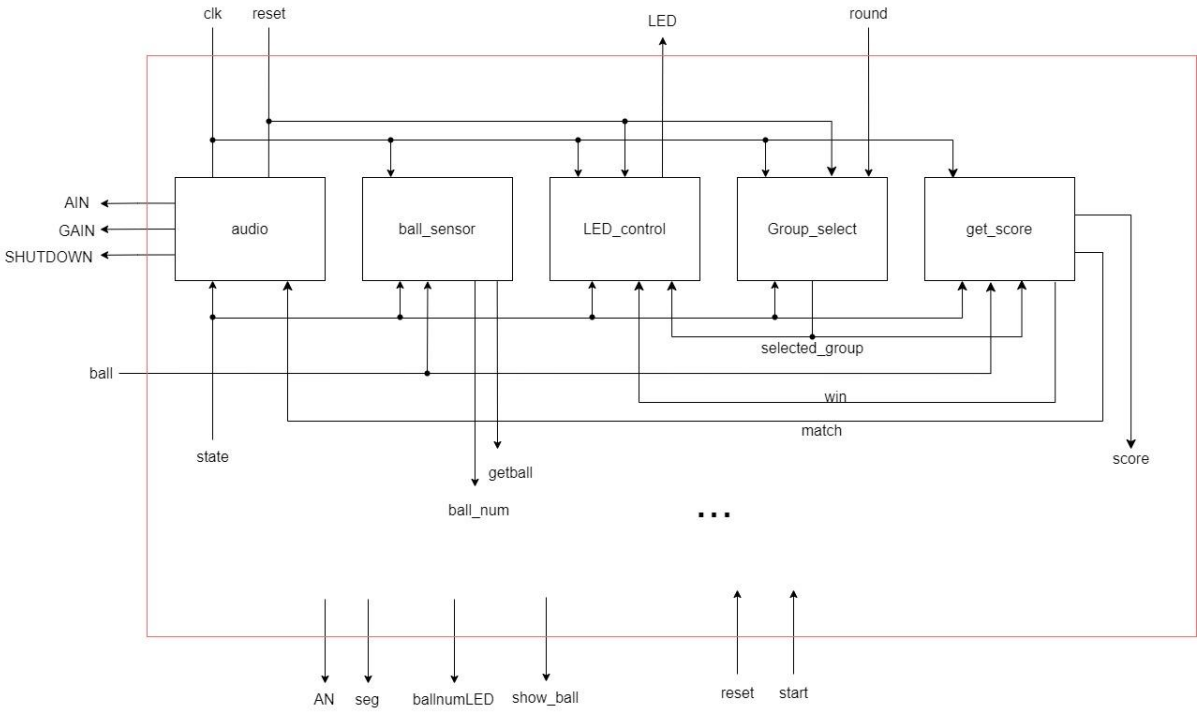
State diagram:

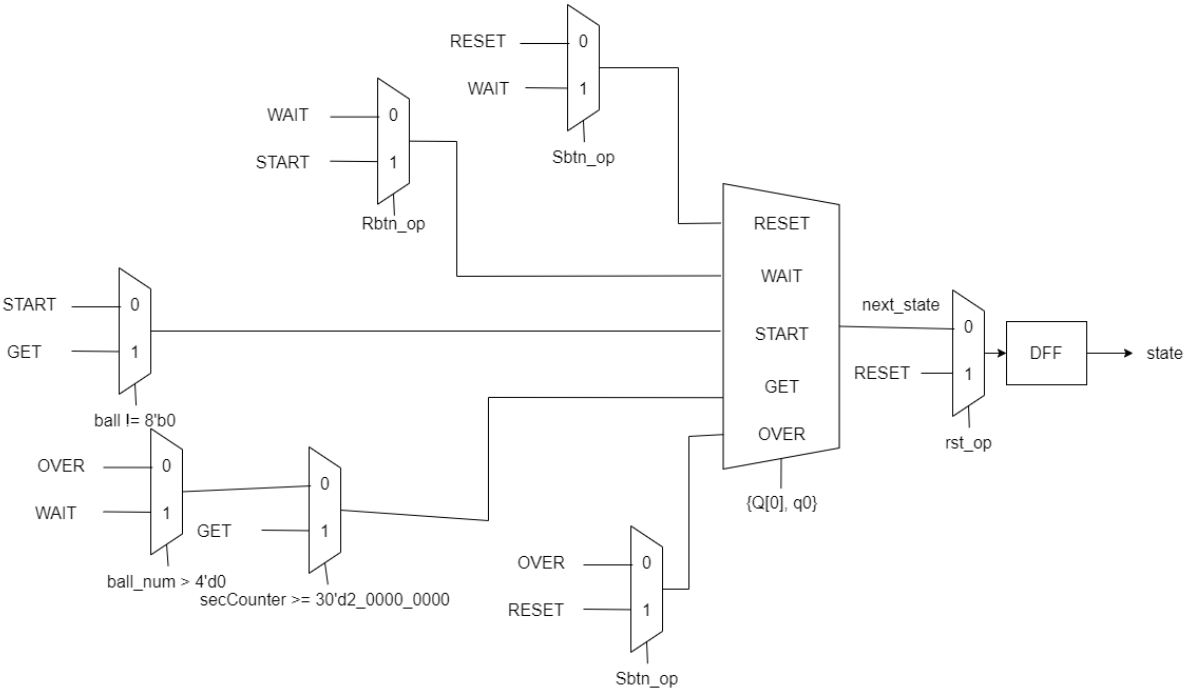


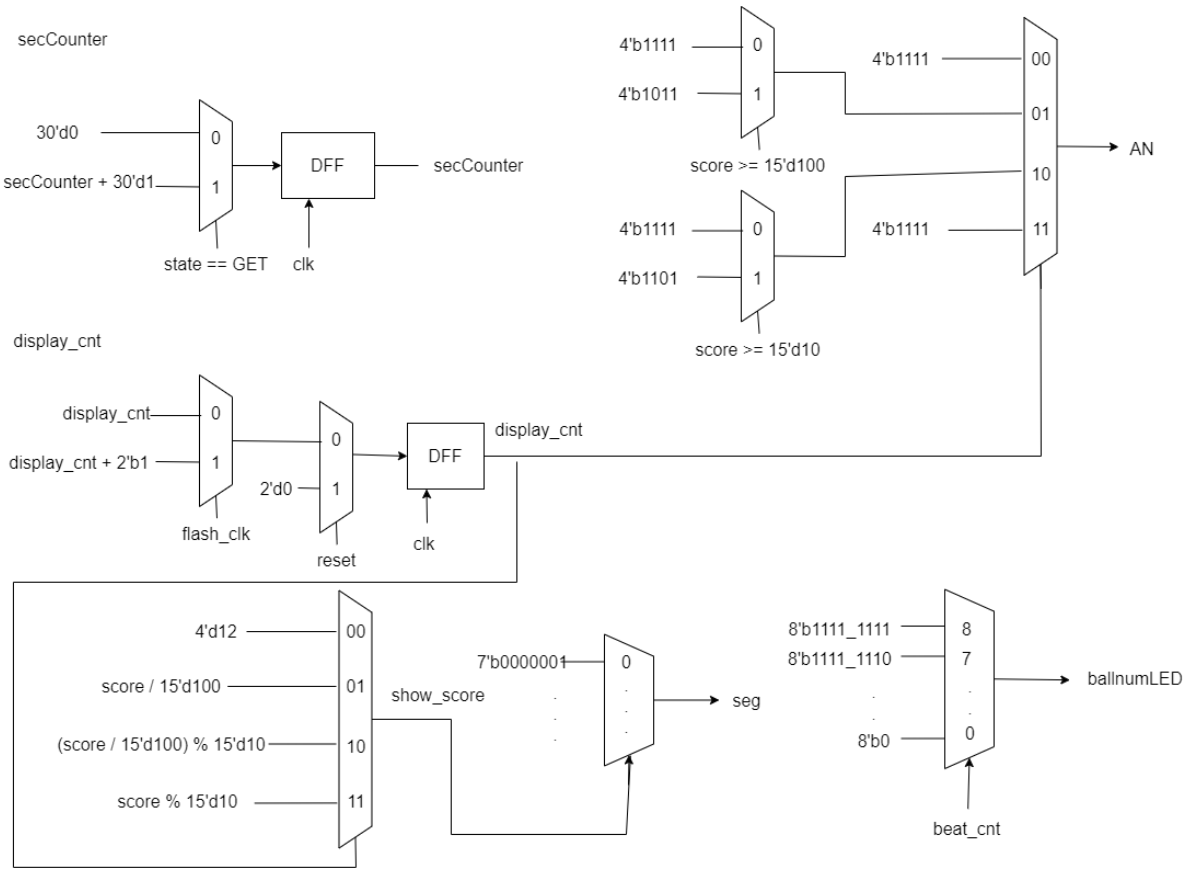
RESET: 初始/按下reset button後會進入，按下start button後進入START。
WAIT: 閃爍，要按下round button才會停止閃爍顯示指定得分洞口。
START: 發射階段，發射球通過洞口後偵測會進入GET。
GET: 判斷剩餘球數是否為零，等兩秒後若沒球了則遊戲結束OVER，反之回到WAIT。
OVER: 遊戲結束，若勝利LED燈會交錯快速閃爍，反之只亮兩顆燈慢速閃。

Block diagram:

1. top module:

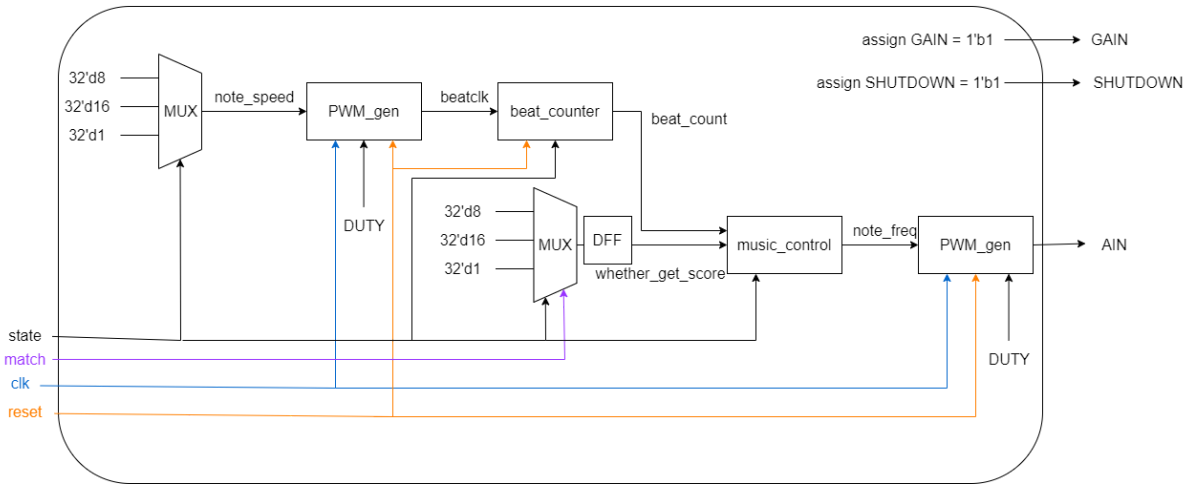

clk
reset
LED
round
AIN
GAIN
SHUTDOWN
audio
ball_sensor
LED_control
Group_select
get_score
selected_group
ball
win
match
state
getball
ball_num
score
AN
seg
ballnumLED
show_ball
reset
start


RESET
WAIT
Sbtn_op
WAIT
START
Rbtn_op
RESET
WAIT
START
GET
OVER
{Q[0], q0}
next_state
RESET
rst_op
DFF
state
START
GET
ball != 8'b0
OVER
WAIT
ball_num > 4'd0
GET
secCounter >= 30'd2_0000_0000
OVER
RESET
Sbtn_op



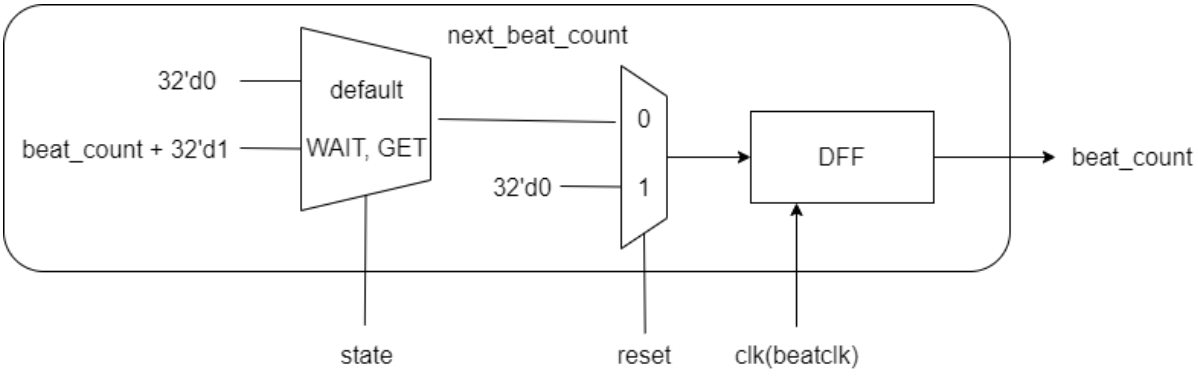
2. audio

依照state和match output出audio需要的AIN, GAIN, SHUTDOWN。先根據不同state製造不同的節拍note_speed，再透過PWM_gen製造出該速度的節拍beatclk，接著透過beat_counter產生beat_countr(對照樂譜第幾個note)，再用music_control得到對應state和beat_count所產生出的音符頻率note_freq，最後再用另一個PMW_gen獲得該音波AIN，其中PWM_gen為助教提供的template module, DUTY設為32'd512。
其中whether_get_score為將有沒有match的結果(可能只有一瞬間)保留整個state的time slot，以便整個GET state都能完整撥放正確的樂譜。



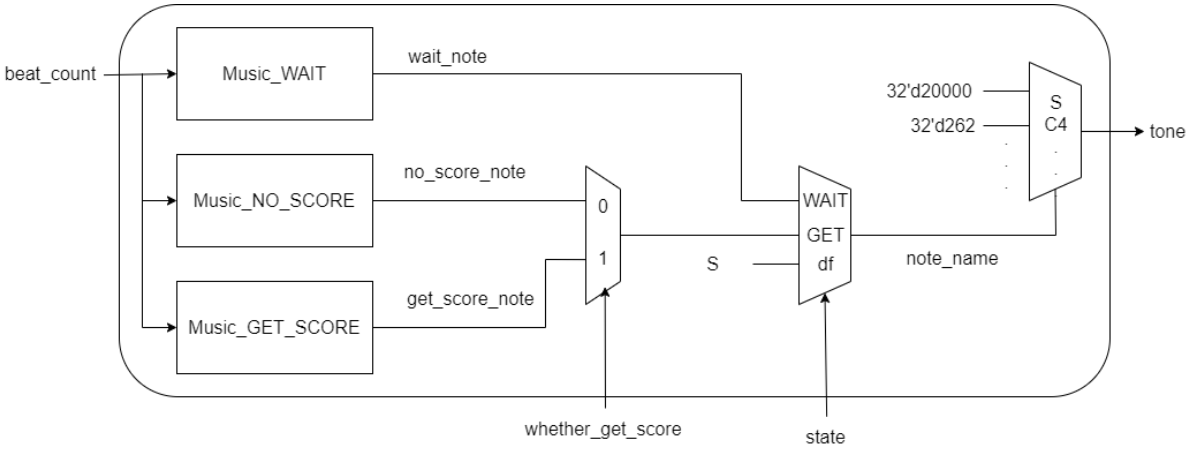
3. beat_counter

因為剛好有需要有音樂的state是WAIT和GET，前後中間都有夾其他state，我們就利用其他state時beat_count歸零、WAIT和GET時beat_count每個beatclk加一達成計算音符序的目的。



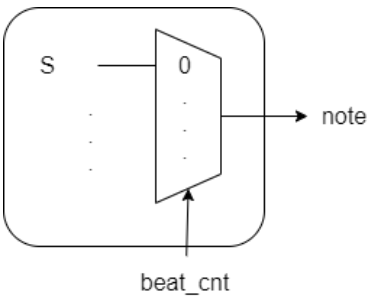
4. music_control

根據state, GET是否有get score選擇該樂譜的output音名，選擇要output的音名(note_name)，再轉換為相對頻率tone。



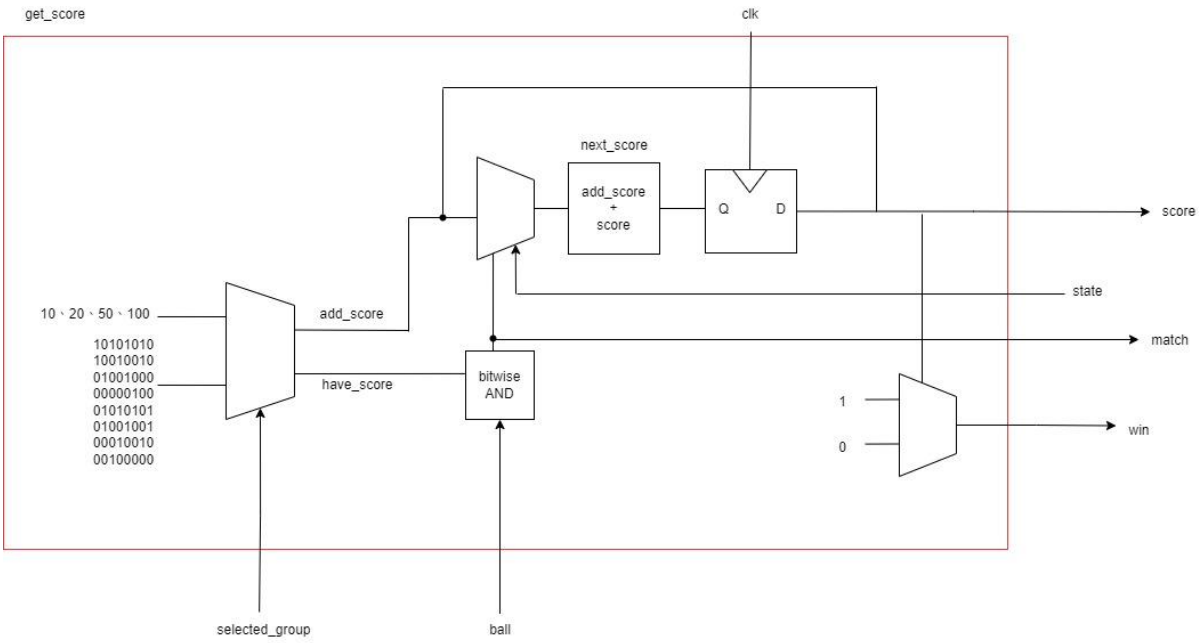
5. Music_WAIT/Music_NO_SCORE/Music_GET_SCORE

根據beat_cnt刻出每拍對應的音名note



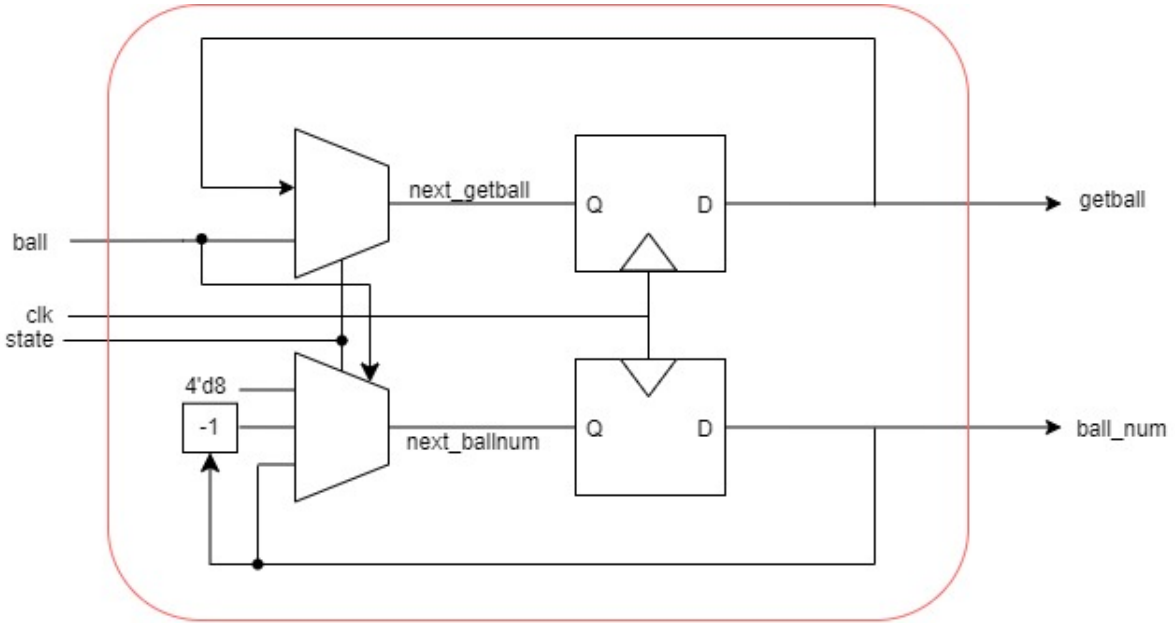
6. Get_score: 依據偵測進球的洞和亂數得分洞口來更新分數

依據selected_group決定要加10/20/50/100分，和決定哪幾個洞會得分到have_score與ball sensor做bitwise AND看是否成功match可以得分。



7. ball_sensor: 根據偵測到球通過來更新目前球數和記下來球是通過哪個洞

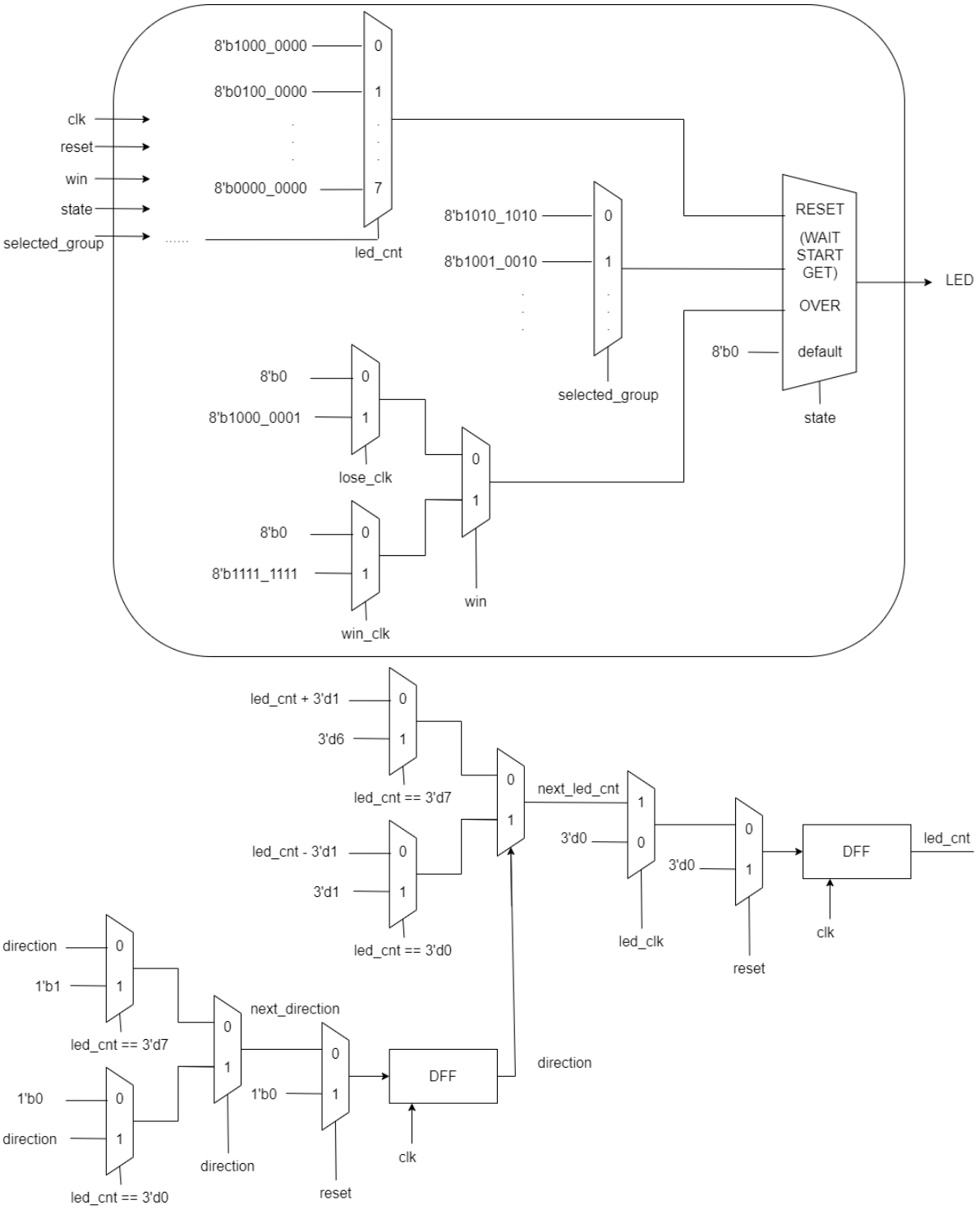
由目前的state和球的數量決定是否要-1顆球，若偵測到球通過和當下是START則減一顆球，反之不變；還有以getball記下球是通過哪個洞。



8. LED_control

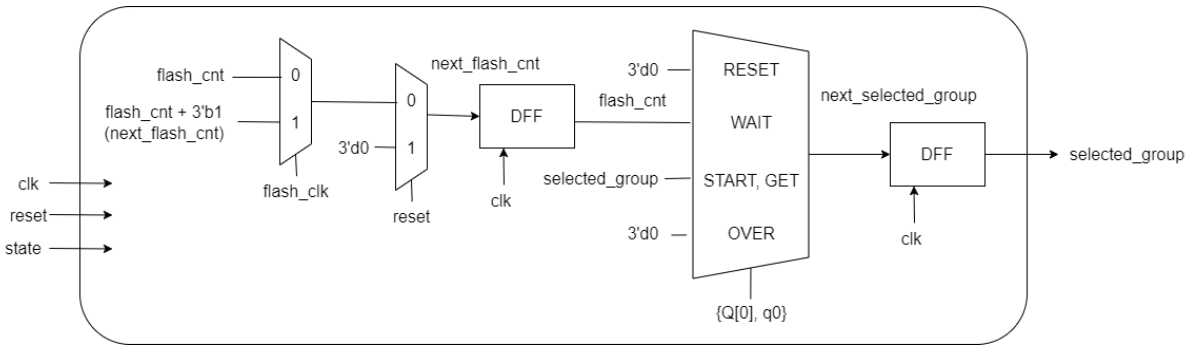
根據不同state output 8bit的LED。RESET使用的led_cnt為輪流從最左邊的LED燈閃到右邊、再從右邊閃回來，一直重複；這邊用的divided clk led_clk為每個clk cyc只有一小個原clk長度的1，因此做為DFF前的enable；OVER時為用不同頻率(win_clk, lose_clk)閃爍的LED，由於只有亮和不亮的變化，便將divided clk做成duty 50%的wave，作為直接的enable。

由於版面不夠其中的led_cnt diagram在下方！



9. Group_select

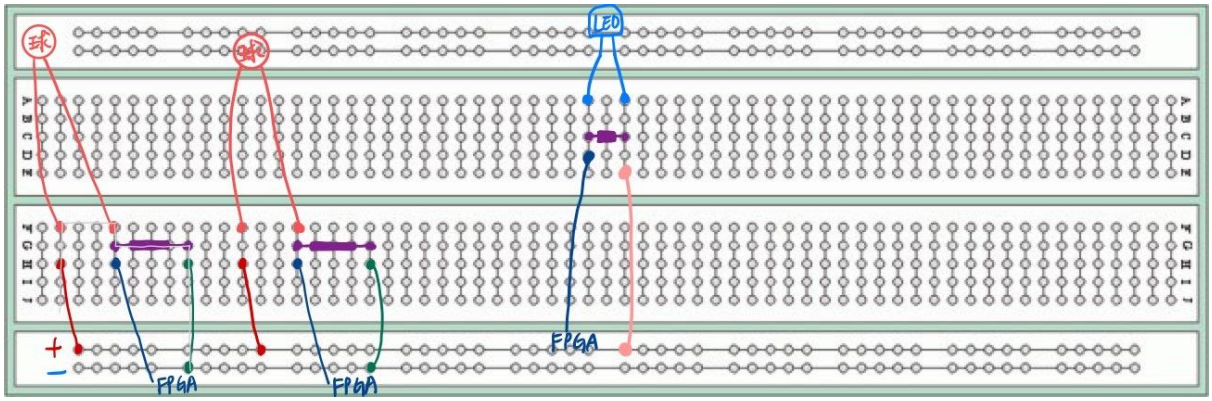
將洞的選擇分成8個組合，因此用一個3bit的flash_cnt表示選擇的組別，雖說亂數但其實是快速輪流閃8個組別，最後當state從WAIT到START時保留剛才跑的最後一組，看起來就像在快速亂數選擇一種洞的組合。

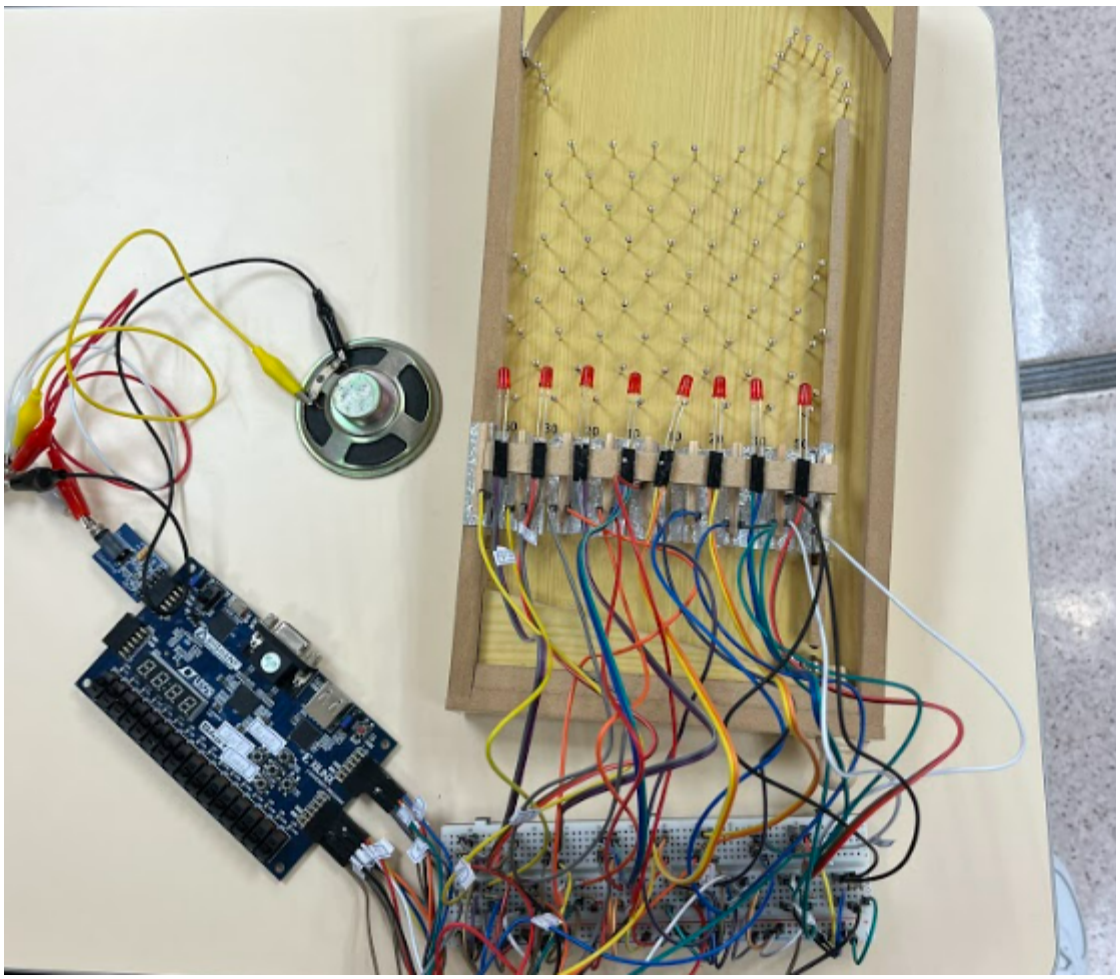


線路圖:

1. 偵測球:正極接出一端給球、另一端回到接地(中間記得加上500歐姆電阻),再接出訊號給板子。
2. LED燈:由板子供訊號給LED,接電阻後另一端接地。

總共八個洞,共八組,所以以上都乘以八,共花費74條線,16個電阻。





V. Experimental Result

照片與影片(說明見資料夾內照片說明):
https://drive.google.com/drive/folders/1_zuFkdWxO9FHw2LmnJE1rtOp_R4shRkF?usp=sharin

曾遇到的問題:

1. 買了微動開關，原本想裝在上面讓彈珠滾過去觸發下壓彈片，結果壓不下去球會卡住滾不下來；後來改成挖洞在下方想讓彈珠滾過壓過去，結果會飛出去；所以最後決定不要用開關，直接把針(鋁箔紙)放在洞口，反正鋼珠可以導電。
2. 原本為了證實有電流就是訊號1，所以直接把線接到板子上的其中一個port測試，結果沒有反應，問了助教說可能是按鈕使用不當的問題，還有需要先加上電阻避免電流過大短路對板子造成損害。
3. 原本認為通電傳入板子訊號會是1，藉此可以感測有球通過，發現通電反而是0會亮板子上的led -> 結果發現是線接錯(接地很重要)。
4. 用鋁箔紙包住洞口接上線來偵測球通過，結果發生球卡住或是過不去的問題->再調整每個針的距離和把底部鋁箔紙距離黏得更近。
5. 測試期間發現有其中一個led燈都不會亮，經過一連串測試，發現是麵包版上其中有一行似乎壞掉，把線換一行接就可以了，硬體問題很多需要一直重新測試...
6. 用鋁箔紙偵測球通過好像不用debounce、onepulse，猜測是因為debounce的clk會比鋼球通過的時間還長。
7. 每次都需要確認一次我們的洞是7:0還是0:7，常常忘記順序會對不上。

8. 做到後面又發現有一個洞永遠偵測不到，已有經驗的我們決定先檢查硬體是否有問題，結果發現真的是有一條線好像壞掉了。
9. 因為最後判斷是否勝利的條件是在OVER判斷分數是否>=100，一開始不知為何就算7segment顯示的分數正確但實際似乎永遠都小於100，推測是因為分數不斷往上加溢位導致判斷出錯，最後的解決方法是多設一個變數給他記勝利就是1。
10. Demo當天好像放著給他待機太久麵包版被燒壞了...
11. 因為有多個module，最好將同一signal不管在哪裡都用一樣的命名，否則打code的時候不小心寫錯他不會告訴你(因為在其他module有宣告過)，然後就要debug超久;D

VI. Conclusion

想到這門課是堂「硬體課」，所以我們決定實作硬體遊戲而不是介面遊戲，嘗試根本不曾接觸過的實體電線、電阻、麵包版。剛開始很徬徨不知從何下手，甚至沒有概念的買了超小的麵包版差點不夠插，只知道麵包版似乎可以供電，但對要插哪幾個洞毫無想法。查了無數的資料，從慢慢知道正極負極接地、電阻電壓的概念才慢慢上手，雖然最後接出的東西很可怕，但看到他有變成通路成功傳送訊號的結果非常有成就感。

經過這次的project，我們更了解到硬體瑕疵問題常常會發生，而且還不像寫軟體可以馬上印出錯誤訊息，需要耐心的一條一條線反覆測試是哪裡接觸不良或壞掉，但這種問題也是最容易排除的，難怪每次有什麼設備有問題網路上都會建議說先看線有沒有接觸不良或是電壓有沒有錯誤再說。

組裝硬體的過程雖然麻煩但也很有趣，一度認為我們修的是美勞課，因為零件都很小、間隔很接近、手指頭太粗，讓我們每次黏鋁箔紙或接線會需要用單隻眼睛對焦加上鑷子輔助，這個過程好像自從國小美術課後就沒再經歷過，但也只有做硬體作品才會有這個過程，可說是難忘又有趣的經驗。

最後，想謝謝教授這學期對我們的照顧和請我們吃披薩，雖然每次寫Lab都寫到快爆肝，但不得不承認是這個過程讓我們更深入了解verilog的奧妙。總而言之，雖然我們期中都爆了，但可以不要當我們冂