Final project: 打地鼠

Design functions overview:

螢幕顯示 9 個地鼠洞，可由鍵盤右側數字鍵 1～9 的按鍵對應到各個螢幕上的洞，地鼠出現時若按下相對應的鍵盤位置則加一分，若有按錯則扣一分，最多扣到 0 分為止，七段顯示器可即時顯示已得分數。配有紀錄最近一次得分紀錄以及至今最高分紀錄功能，透過一個 push bottom 切換要顯示何者，並顯示於七段顯示器上。此外，也可以設定一次出現幾隻地鼠(1 或 2 或 3 隻)、地鼠出現總次數(15 或 20 或 30 次)，以及地鼠的出現速度快慢(慢、中等、快速)共 27 種模式，設定時使用的是右側數字鍵 2(下)、4(左)、6(右)、8(上)當作上下左右鍵來控制，確定鍵為 enter 鍵。所有遊戲畫面與設定皆顯示於螢幕上，除了螢幕顯示之外，在 FPGA 板上也可以透過 9 個 LED 來顯示地鼠位置，以及 3 個 LED 顯示目前遊戲的狀態。

Input/Output Table

input	function
右側數字鍵 1	對應螢幕上左下的地鼠位置
右側數字鍵 2	對應螢幕上中下的地鼠位置 / 方向鍵(下)
右側數字鍵 3	對應螢幕上右下的地鼠位置
右側數字鍵 4	對應螢幕上左中的地鼠位置 / 方向鍵(左)
右側數字鍵 5	對應螢幕上正中的地鼠位置
右側數字鍵 6	對應螢幕上右中的地鼠位置 / 方向鍵(右)
右側數字鍵 7	對應螢幕上左上的地鼠位置
右側數字鍵 8	對應螢幕上中上的地鼠位置 / 方向鍵(上)
右側數字鍵 9	對應螢幕上右上的地鼠位置
U18 push bottom(正中間)	顯示最近一次得分紀錄(若沒按下則顯示最高分紀錄)
enter 鍵	各項功能的確定按鍵
V17 DIP switch(最右邊)	reset

output	function
vgaRed、vgaBlue、vgaGreen	所有遊戲畫面皆顯示於 VGA 螢幕上
七段顯示器	顯示分數
LED0～8(最左邊開始 9 個)	在 FPGA 板上也顯示地鼠位置
LED9～11(最右邊開始 3 個)	顯示當前遊戲狀態

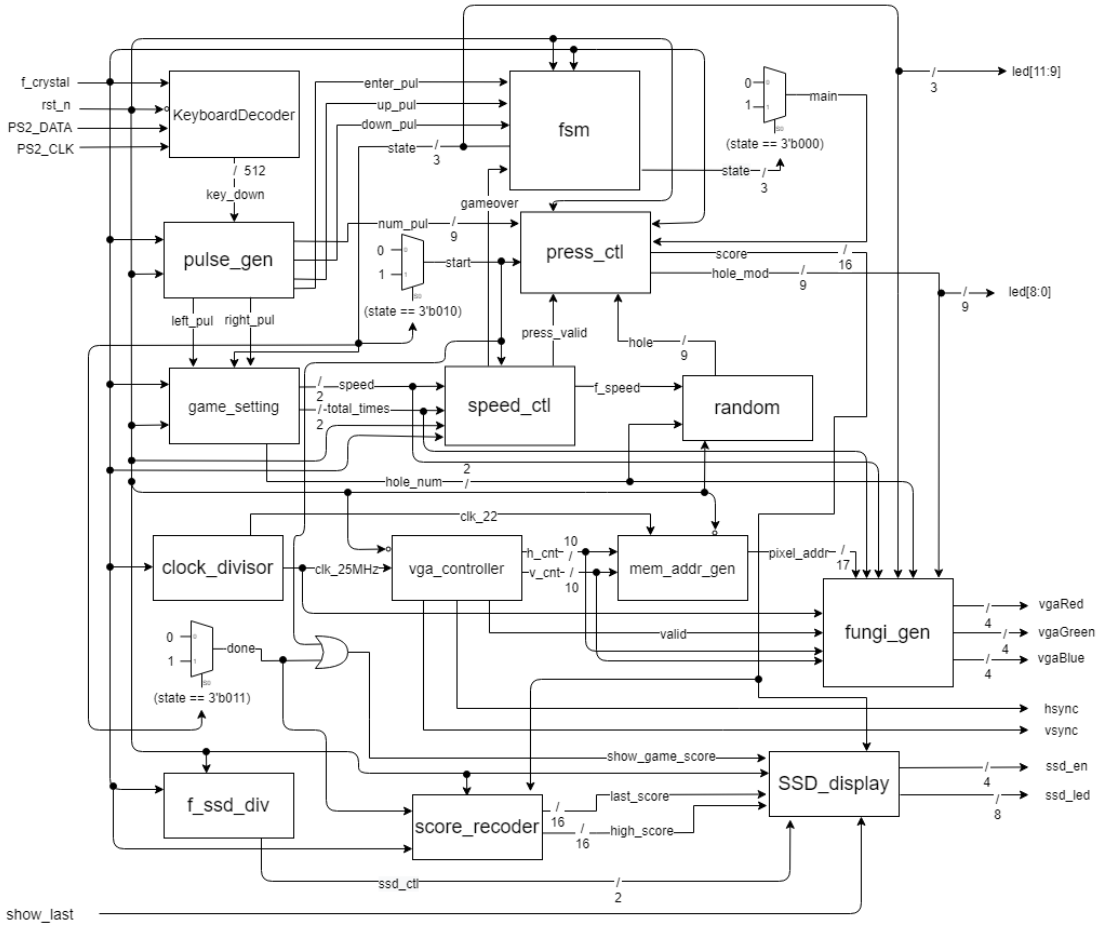
IO:

Inout: PS2_DATA, PS2_CLK

Input: f_crystal, rst_n, show_last

Output: [3:0]vgaRed, [3:0]vgaGreen, [3:0]vgaBlue, hsync, vsync,
[11:0]led, [7:0]ssd_led, [3:0]ssd_en

Block diagram:



1. KeyboardDecoder: 此 module 將鍵盤訊號轉換成可以得知哪些按鍵目前被按著的訊號(keydown)、最近一個被操作的按鍵訊號(last_change)以及按下或放開任一按鍵時的通知訊號(key_valid)。

2. pulse generator: 將 KeyboardDecoder 中產生的 keydown 作為輸入，把右側數字鍵以及 enter 鍵按下時產生的 keydown 訊號化為一個 one pulse 訊號並輸出。

3. finite state machine (fsm): 透過鍵盤按鍵改變當前狀態，整個遊戲共分成 8 個狀態，分別為:(1)主畫面中的等待確認開始畫面 (2)遊戲進行中 (3)遊戲結束並顯示分數 (4)主畫面中的等待確認設定畫面 (5)設定地鼠數量中 (6)設定地鼠出現總次數中 (7)設定地鼠出現速度中 (8)設定結束，準備回到主畫面。

4. press control(press_ctl): 內含:(1)判斷按鍵是否有按到對應的地鼠位置。
(2)4-digit 的 counter，按對 counter + 1(加一分)，按錯則 counter – 1(扣一分)，輸

出遊戲過程中當前累積的分數讓七段顯示器顯示。
(3)此 module 也輸出修正過後的地鼠位置，意思是當按下對應位置的地鼠後，及時將這個洞修正為沒有地鼠，如此才符合常理並且可以避免連續按下同一個位置導致不斷加分的情形。

5. 遊戲設定(game setting): 透過目前 finite state machine 的狀態，判斷現在要設定地鼠數量、地鼠出現總次數，還是地鼠出現速度等等的功能。

6. speed control(speed_ctl): 讀取遊戲設定中設定好的地鼠出現速度與地鼠出現總次數，將頻率除頻成相對應的頻率快慢並輸出，也透過此頻率與地鼠出現總次數計算出何時遊戲應該結束，並在遊戲應結束時輸出 game over 訊號通知其它 module。

7. 產生隨機數字(random): 利用教授提供的方法產生隨機的數字，不過如果只照著這樣的方法，會導致地鼠不會連續兩次出現在同一個位置，為了改善這樣的情形，我們稍微修改了一些地方，使得地鼠有可能連續兩次出現在同一個位置，增加遊戲豐富度。

8. 除頻器(clock divisor): 將 crystal frequency 除頻成 clk1 與 clk22，提供給 mem_addr_gen、fungi_gen，以及 vga_controller 三個處理 VGA 畫面的模組使用。

9. vga_controller: 處理 VGA 顯示的通訊協定，輸出 h_cnt、v_cnt 給 fungi_gen 和 mem_addr_gen；輸出 valid 給 fungi_gen；以及輸出 hsync、vsync 訊號給螢幕。

10. mem_addr_gen: 將畫質降低成 240p(把 1 個 pixel 當作 4 個用)，以免超出 RAM 容量，並輸出位址給 fungi_gen 使用。

11. fungi_gen: 控制所有 VGA 畫面的 module，分為下列四種畫面: (1)主畫面 → 可以選擇進入遊玩或是進入設定 (2)遊玩畫面 → 地鼠(香菇)會隨機從九個洞出現 (3)結算畫面 → 有一張感謝遊玩的圖片 (4)設定畫面 → 能夠設定遊戲模式的畫面。

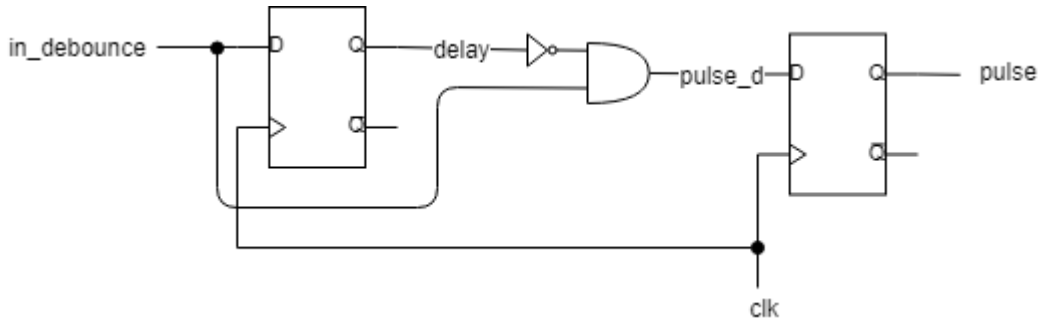
12. ssd control enable generator(f_ssd_div): 產生 2bit 的 ssd control enable，作為七段顯示器的 scan control 控制項。

13. 分數記錄(score_recoder): 將 press control 計算的遊戲得分儲存下來，並比較此得分是否要取代掉最高得分，然後輸出最高得分和最近一次得分給七段顯示器顯示。

14. 七段顯示解碼器(SSD_display): 透過當前狀態判斷要顯示遊戲進行時的即時分數、目前最高分紀錄，或是最近一次的遊戲得分，並藉由 2bit 的 ssd control enable 控制項，快速輪流顯示四個七段顯示器，使得七段顯示器可以同一時間顯示四位數得分。

Logic diagram & Discussion:

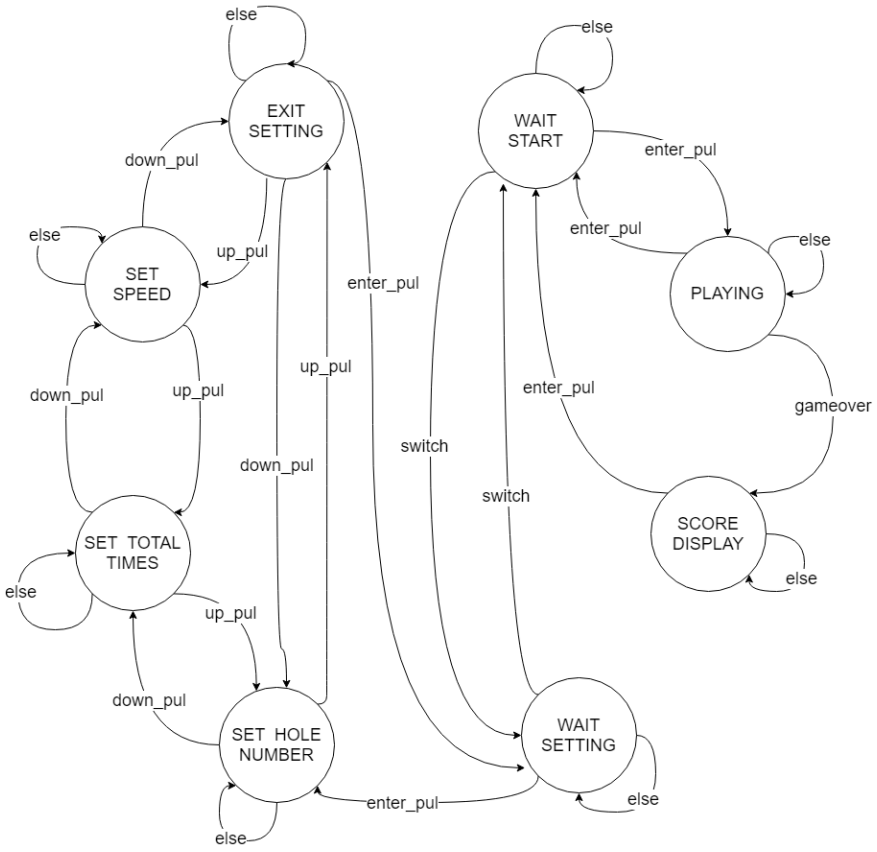
pulse generator: 將 KeyboardDecoder 中產生的 keydown 作為輸入，把右側數字鍵以及 enter 鍵按下時產生的 keydown 訊號化為一個 one pulse 訊號並輸出。



將數字鍵 1～9 以及 enter 鍵各自對應到的 key down 訊號各自連接到 in_debounce 裡，產生各自的 pulse 訊號並輸出。

finite state machine (fsm): 透過鍵盤按鍵改變當前狀態，整個遊戲共分成 8 個狀態，分別為:(1)主畫面中的等待確認開始畫面 (2)遊戲進行中 (3)遊戲結束並顯示分數 (4)主畫面中的等待確認設定畫面 (5)設定地鼠數量中 (6)設定地鼠出現總次數中 (7)設定地鼠出現速度中 (8)設定結束，準備回到主畫面。

```
switch = up_pul | down_pul;
```



※switch: 有按上鍵或下鍵

(1) 主畫面中的等待確認開始畫面 WAIT_START:
若按上或下鍵 → 主畫面中的等待確認設定畫面 WAIT_SETTING。
若按 enter 鍵 → 開始遊戲 PLAYING。

(2) 遊戲進行中 PLAYING:
若遊戲結束(gameover 訊號) → 遊戲結束並顯示分數 SCORE_DISPLAY。
若按 enter 鍵 → 跳出遊戲並回到主畫面 WAIT_START。

(3) 遊戲結束並顯示分數 SCORE_DISPLAY:
若按 enter 鍵 → 回到主畫面 WAIT_START。

(4) 主畫面中的等待確認設定畫面 WAIT_SETTING:
若按上或下鍵 → 主畫面中的等待確認開始畫面 WAIT_START。
若按 enter 鍵 → 設定地鼠數量中 SET_HOLE_NUM。

(5) 設定地鼠數量中 SET_HOLE_NUM:
若按下鍵 → 設定地鼠出現總次數中 SET_TOTAL_TIMES。
若按上鍵 → 設定結束，準備回到主畫面 EXIT_SETTING。

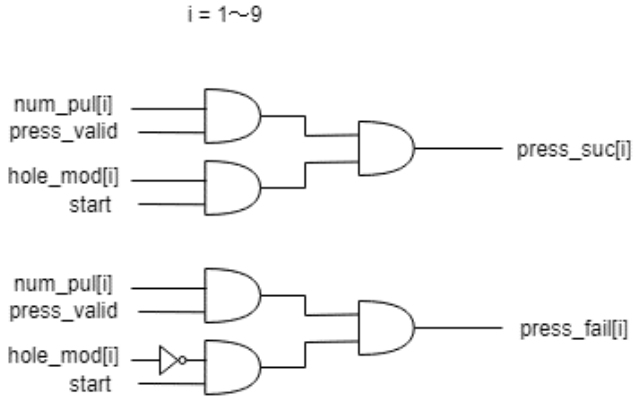
(6) 設定地鼠出現總次數中 SET_TOTAL_TIMES:
若按下鍵 → 設定地鼠出現速度中 SET_SPEED。
若按上鍵 → 設定地鼠數量中 SET_HOLE_NUM。

(7) 設定地鼠出現速度中 SET_SPEED:
若按下鍵 → 設定結束，準備回到主畫面 EXIT_SETTING。
若按上鍵 → 設定地鼠出現總次數中 SET_TOTAL_TIMES。

(8) 設定結束，準備回到主畫面 EXIT_SETTING:
若按 enter 鍵 → 回到主畫面中的等待確認設定畫面 WAIT_SETTING。
若按下鍵 → 設定地鼠數量中 SET_HOLE_NUM。
若按上鍵 → 設定地鼠出現速度中 SET_SPEED。

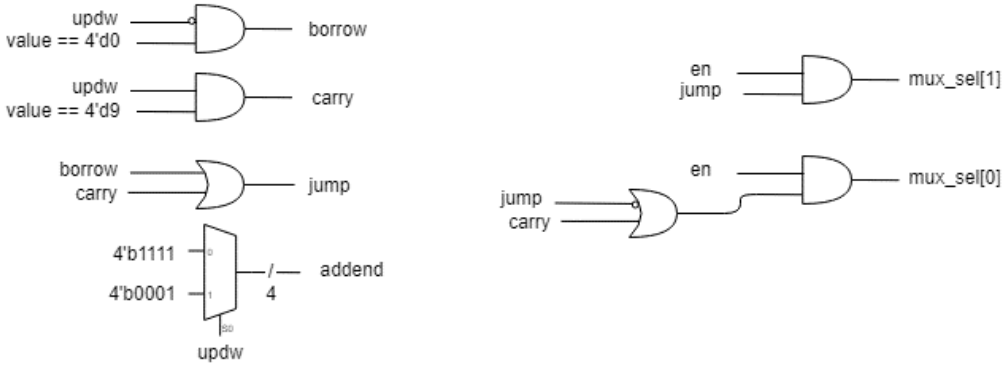
press control(press_ctl): 內含:(1)判斷按鍵是否有按到對應的地鼠位置。
(2)4-digit 的 counter，按對 counter + 1(加一分)，按錯則 counter – 1(扣一分)，輸出遊戲過程中當前累積的分數讓七段顯示器顯示。
(3)此 module 也輸出修正過後的地鼠位置，意思是當按下對應位置的地鼠後，及時將這個洞修正為沒有地鼠，如此才符合常理並且可以避免連續按下同一個位置導致不斷加分的情形。

(1)判斷按鍵是否有按到對應的地鼠位置:
※start 在遊戲進行中時會等於 1

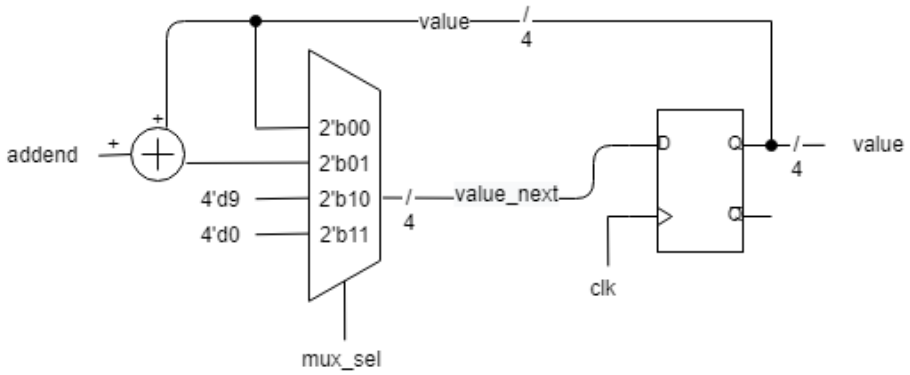


press_suc[i]判定各個按鍵是否有對應到地鼠出現的位置；press_fail[i]判定按鍵是否按錯位置。
讓 plus1 =
press_suc[9]|press_suc[8]|press_suc[7]|press_suc[6]|press_suc[5]|press_suc[4]|press_suc[3]|press_suc[2]|press_suc[1]
讓 minus1 =
press_fail[9]|press_fail[8]|press_fail[7]|press_fail[6]|press_fail[5]|press_fail[4]|press_fail[3]|press_fail[2]|press_fail[1]
透過 plus1&(~minus1)的形式連接進 counter 裡面，因此按對時此訊號為 1；按錯時此訊號為 0。

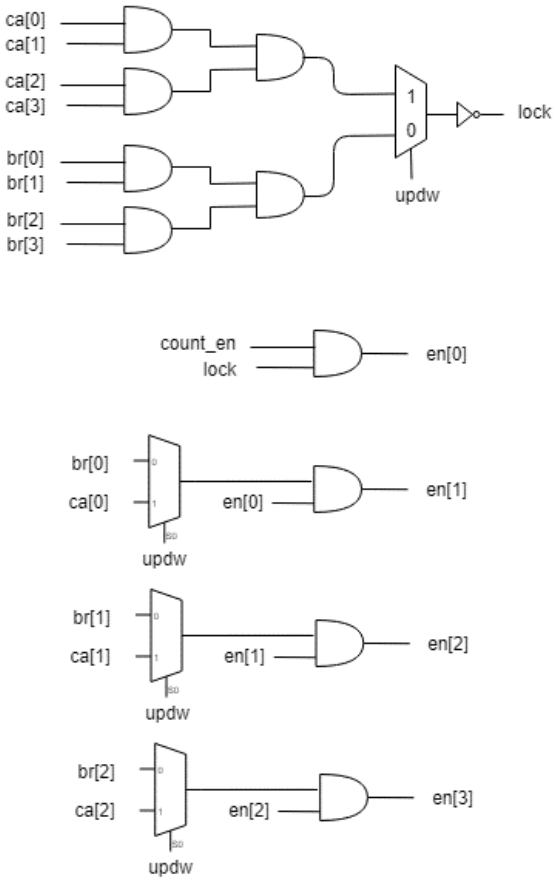
(2)4-digit 的 counter，按對 counter + 1(加一分)，按錯則 counter – 1(扣一分)，輸出遊戲過程中目前累積的分數讓七段顯示器顯示。



用 updw 連接上述的 plus1&(~minus1)訊號，再用 updw 的值判斷被加數 attend 的值，當 updw = 1 → 被加數 = 4’b0001；當 updw = 0 → 被加數 = 4’b1111。



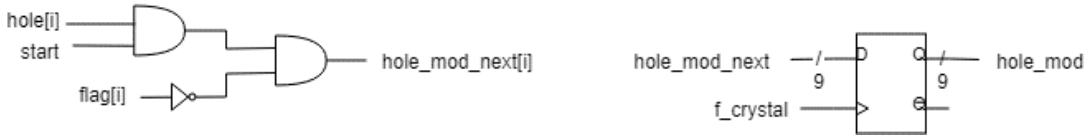
由於扣分時的被加數等於 4’b1111，因此自然可以達成減 1 的效果。
當進位時，下一個 value 值 = 4’d0，並通知下一級要進位；當借位時，下一個 value 值 = 4’d9，並通知上一級要借位。



在這個 module 將四位數的 counter 連接起來，形成四個互相進位、借位的加法器。

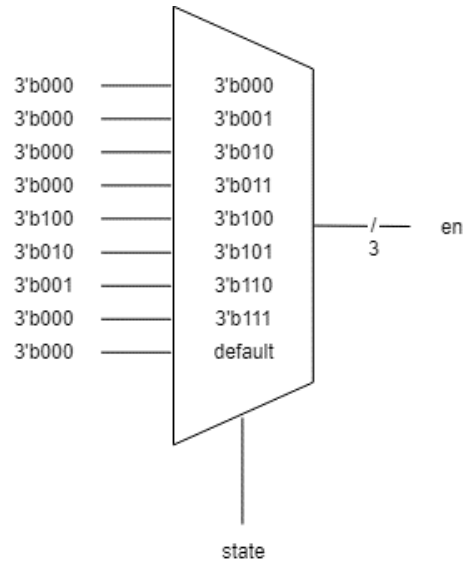
(3)輸出修正過後的地鼠位置，意思是當按下對應位置的地鼠後，及時將這個洞修正為沒有地鼠，如此才符合常理並且可以避免連續按下同一個位置導致不斷加分的情形。

使用 flag 來標記每個地鼠位置是否在出現地鼠時被按對了，若按對，則 flag 變成 1，直到下一次刷新地鼠位置才變回 0。



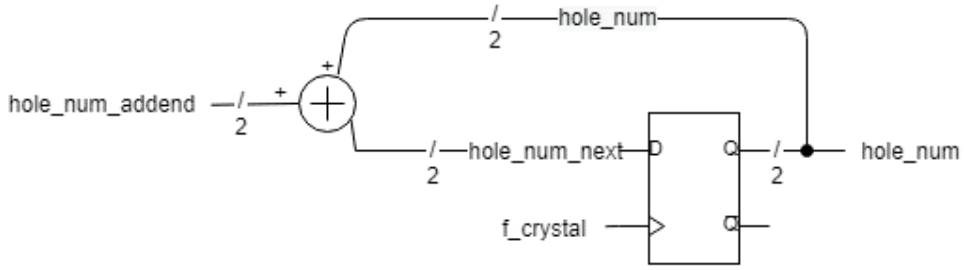
把原本隨機產生好的地鼠位置做即時修正，若有按對則去除那個位置的地鼠，並將修正後的地鼠位置訊號輸出給 fungi_gen 處理後同時修正螢幕上顯示的地鼠位置。

<u>遊戲設定(game setting):</u> 透過目前 finite state machine 的狀態，判斷現在要設定地鼠數量、地鼠出現總次數，還是地鼠出現速度等等的功能。



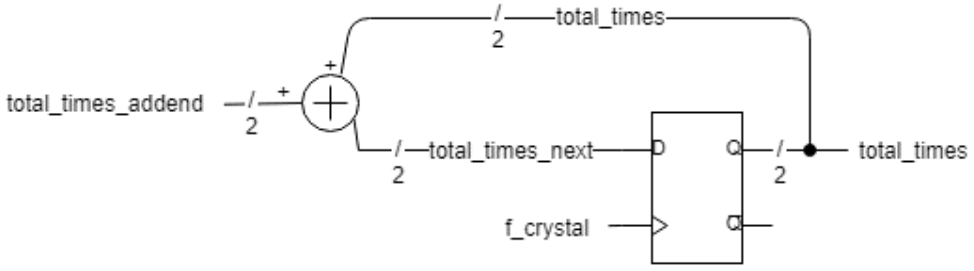
State = 3'b100 時 → 當前狀態為設定地鼠數量中，讓 en[2] = 1。
State = 3'b101 時 → 當前狀態為設定地鼠出現總次數中，讓 en[1] = 1。
State = 3'b110 時 → 當前狀態為設定地鼠出現速度中，讓 en[0] = 1。

(1)設定地鼠數量:



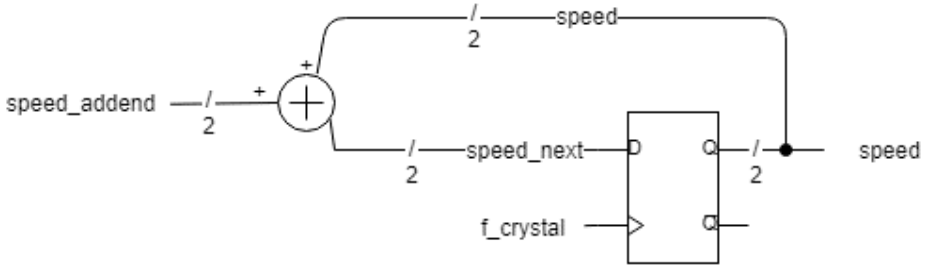
hole_num_addend: 當 en[2]等於 1 且當前地鼠數量未達上限時，按下右鍵使得 hole_num_addend 等於 2'b01，讓 hole_num + 1；未達下限時，按下左鍵使得 hole_num_addend 等於 2'b11，讓 hole_num + 2'b11，也就是減 1 的意思。

(2)設定地鼠出現總次數:



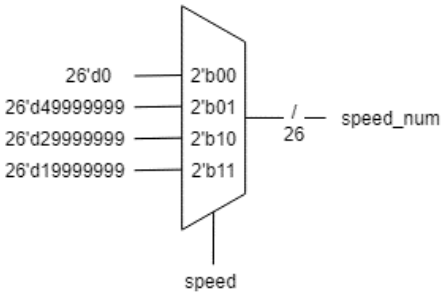
total_times_addend: 當 en[1]等於 1 且當前地鼠數量未達上限時，按下右鍵使得 total_times_addend 等於 2'b01，讓 total_times + 1；未達下限時，按下左鍵使得 total_times_addend 等於 2'b11，讓 total_times + 2'b11，也就是減 1 的意思。

(3)設定地鼠出現速度:

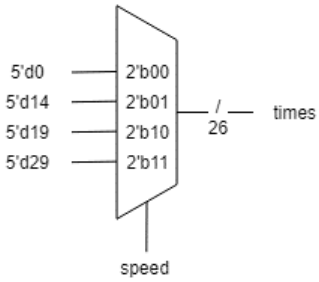


speed_addend: 當 en[1]等於 1 且當前地鼠數量未達上限時，按下右鍵使得 speed_addend 等於 2'b01，讓 speed + 1；未達下限時，按下左鍵使得 speed_addend 等於 2'b11，讓 speed + 2'b11，也就是減 1 的意思。

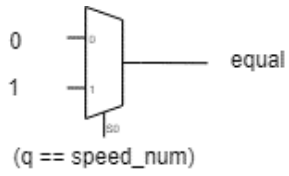
<u>speed control(speed_ctl):</u> 讀取遊戲設定中設定好的地鼠出現速度與地鼠出現總次數，將頻率除頻成相對應的頻率快慢並輸出，也透過此頻率與地鼠出現總次數計算出何時遊戲應該結束，並在遊戲應結束時輸出 game over 訊號通知其它 module。



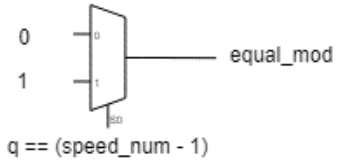
依據遊戲設定中設定好的速度等級來判斷要除頻的除數為多少。若是等級 1(慢) → 將 FPGA 板上的 crystal frequency 除以(五千萬*2)；若是等級 2(中等) → 將 FPGA 板上的 crystal frequency 除以(三千萬*2)；若是等級 3(快) → 將 FPGA 板上的 crystal frequency 除以(兩千萬*2)。



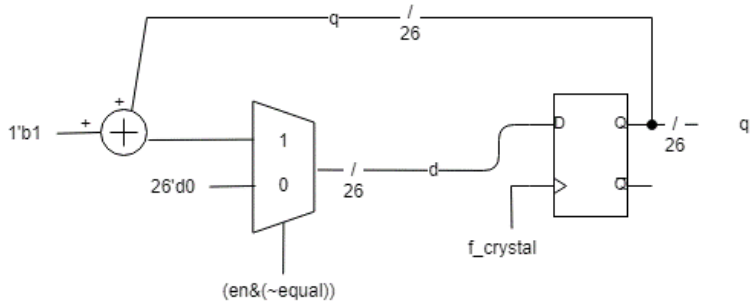
依據遊戲設定中設定好的總次數等級來判斷地鼠出現總次數。若是等級 1(少) → times = 15 次；若是等級 2(中等) → times = 20 次；若是等級 3(多) → times = 30 次。

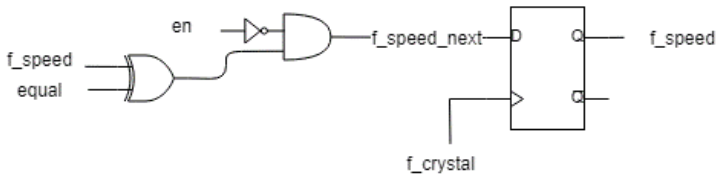


equal 在 q 數到要除頻的除數(speed num)時會等於 1。

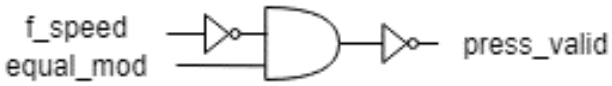


equal_mod 在 q 數到要除頻的除數減 1(speed num - 1)時會等於 1。

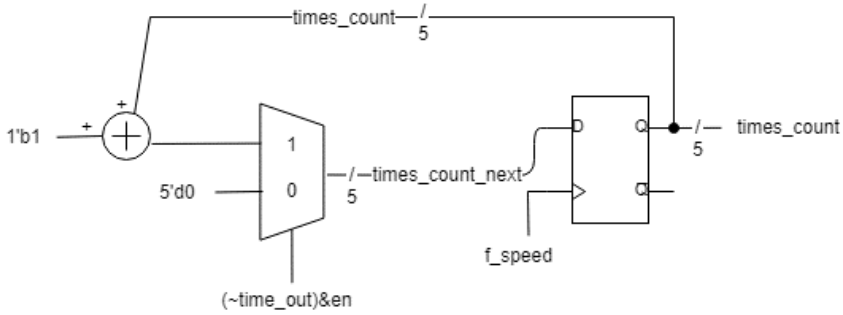




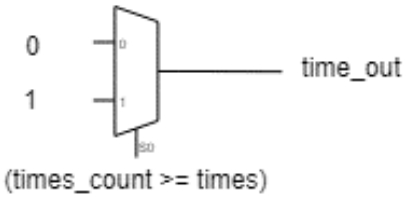
使用一個數到除數的上數器，搭配一個會 toggle 的 f_speed，除頻出遊戲設定好的相對應的快慢頻率。



press_valid 維持 1，只在要刷新地鼠位置前一刻掉到 0，使得所有按鍵均為無效，確保其它模組功能有資料變化前的準備時間。

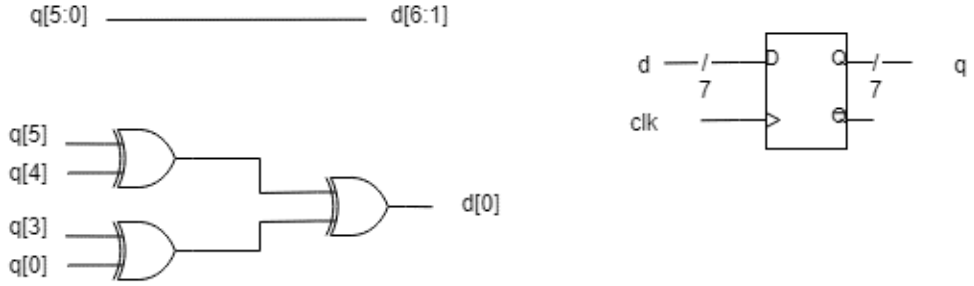


使用 times_count 計算是否已經到達地鼠出現的總次數了。



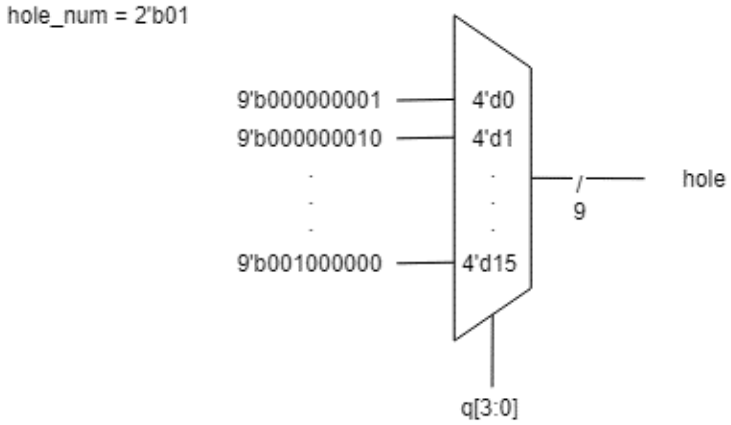
當次數大於等於總次數時，讓 time_out = 1，再將此訊號輸出到產生 one_pulse 的模組裡，產生一個 gameover 的 pulse 訊號。

<u>產生隨機數字(random):</u> 利用教授提供的方法產生隨機的數字，不過如果只照著這樣的方法，會導致地鼠不會連續兩次出現在同一個位置，為了改善這樣的情形，我們稍微修改了一些地方，使得地鼠有可能連續兩次出現在同一個位置，增加遊戲豐富度。



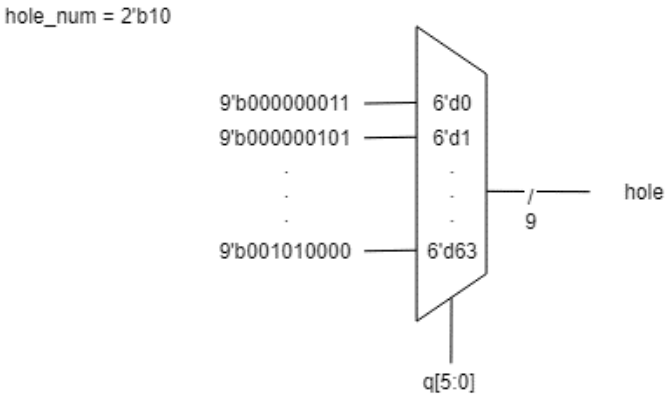
以上產生 7 位數的隨機數字，並用 q 記錄下來。

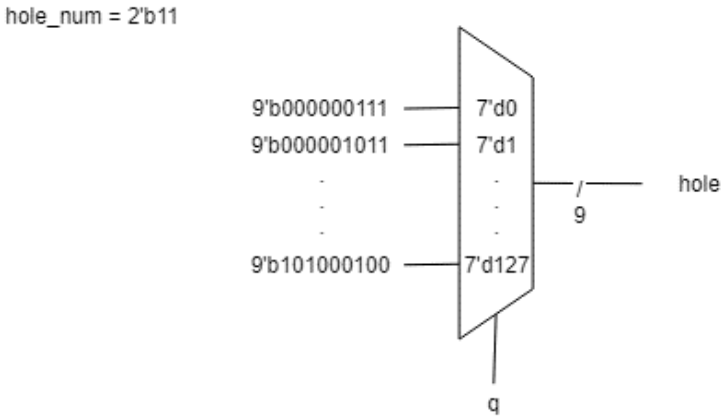
為了使同一個位置可以連續出現地鼠:
(1)當地鼠數量設定為 1 時:



9’b 的 hole 表示第幾個 bit 為 1 則幾號的洞就會有地鼠出現。
讓同一個洞對應到不只一個 q 值，則從這次的 q 值跳到下一個 q 值時，對應到的洞就有可能是一樣的。

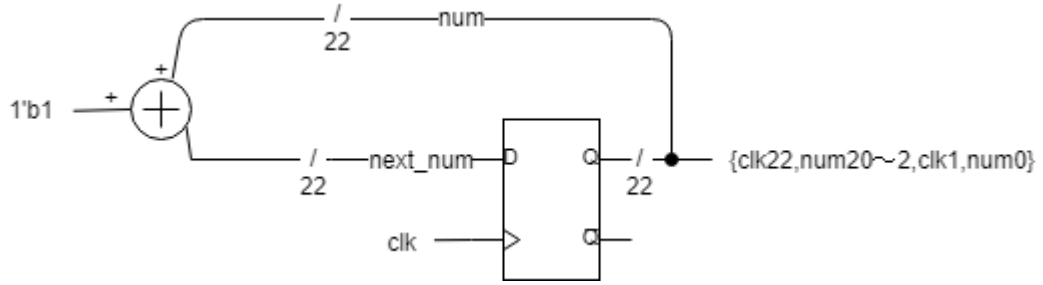
(2)當地鼠數量設定為 2 或 3 時:
以相同的方法讓同兩個或同三個洞對應到不只一個 q 值





如此可以達成讓地鼠有機率連續出現在相同的洞的效果。

除頻器(clock divisor): 將 crystal frequency 除頻成 clk1 與 clk22，提供給 mem_addr_gen、fungi_gen，以及 vga_controller 三個處理 VGA 畫面的模組使用。



用 FPGA 板上的 crystal frequency 當作 clock，每次 clock 使 num + 1，clk22 為 num 的第 21 位數；clk1 為 num 的第 1 位數，將 clk1 與 clk22 提供給 mem_addr_gen、fungi_gen，以及 vga_controller 三個處理 VGA 畫面的模組使用。

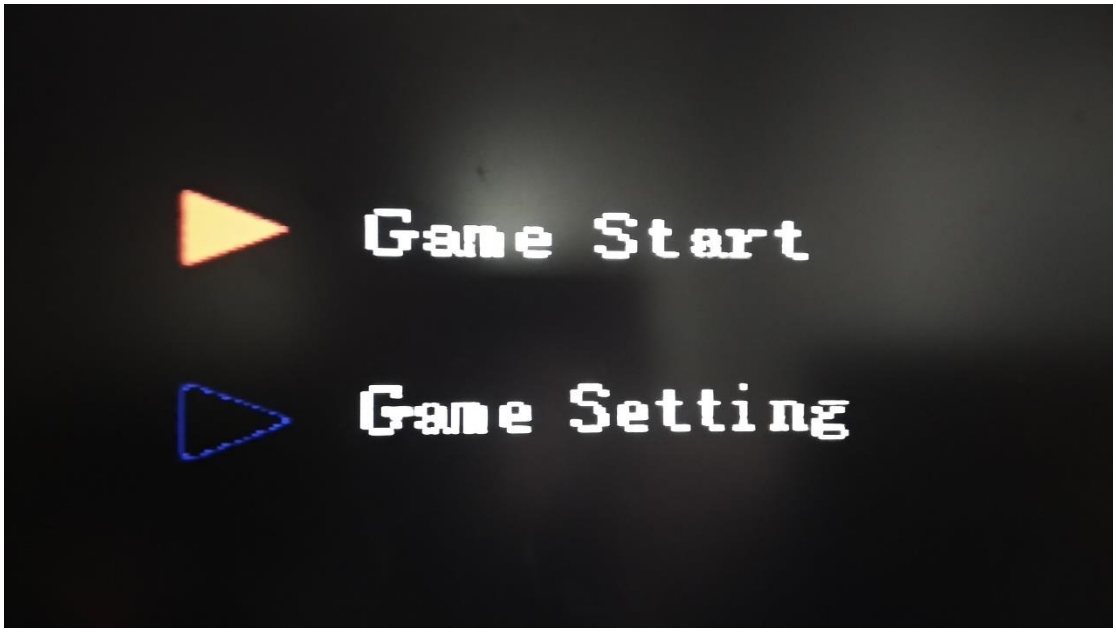
fungi_gen: 控制所有 VGA 畫面的 module，分為下列四種畫面: (1)主畫面 → 可以選擇進入遊玩或是進入設定 (2)遊玩畫面 → 地鼠(香菇)會隨機從九個洞出現 (3)結算畫面 → 有一張感謝遊玩的圖片 (4)設定畫面 → 能夠設定遊戲模式的畫面。

(1)主畫面:

主畫面是一張 180*120 的圖片，經過座標平移後放在螢幕中間的，具體方式如下。

a. 決定圖片在螢幕中的範圍 h_cnt 在[140, 499]區間，v_cnt 在[120, 359]區間。
b. 產生平移值，當 h_cnt 與 v_cnt 在範圍內時，水平平移值(init_shift_x)為 140，垂直平移值(init_shift_y)為 120；在範圍外時，init_shift_x 與 init_shift_y 則直接等於 h_cnt 與 v_cnt(避免 memory 讀取到 invalid 的值而產生花屏)。
c. 生成 memory_addr，將 h_cnt 與 v_cnt 分別減去 init_shift_x 與 init_shift_y，得到 h_cnt_init 與 v_cnt_init，最後用類似老師的作法，得到 init_addr = (h_cnt_init>>1)+(180*(v_cnt_init>>1))。
d. 將 init_addr 帶到生成的 memory 中，得到 12bits 的 init_pixel，交給 MUX 選擇輸出的 vgaRed, vgaGreen, vgaBlue。

主畫面有兩個選項，按 8 或 2 可以進行切換，同時畫面中的紅藍三角形也會在空心與實心狀態切換，實作方法是利用 init_pixel 的顏色判定再用 MUX 做選擇，當 state 為 WAIT_START 時，將偏藍的 pixel 改為黑色，其餘的保留原色，反之當 state 為 WAIT_SETTING 時，將偏紅的 pxiel 改為黑色，但因為外框不夠紅所以保留原色。



(2) 遊玩畫面:

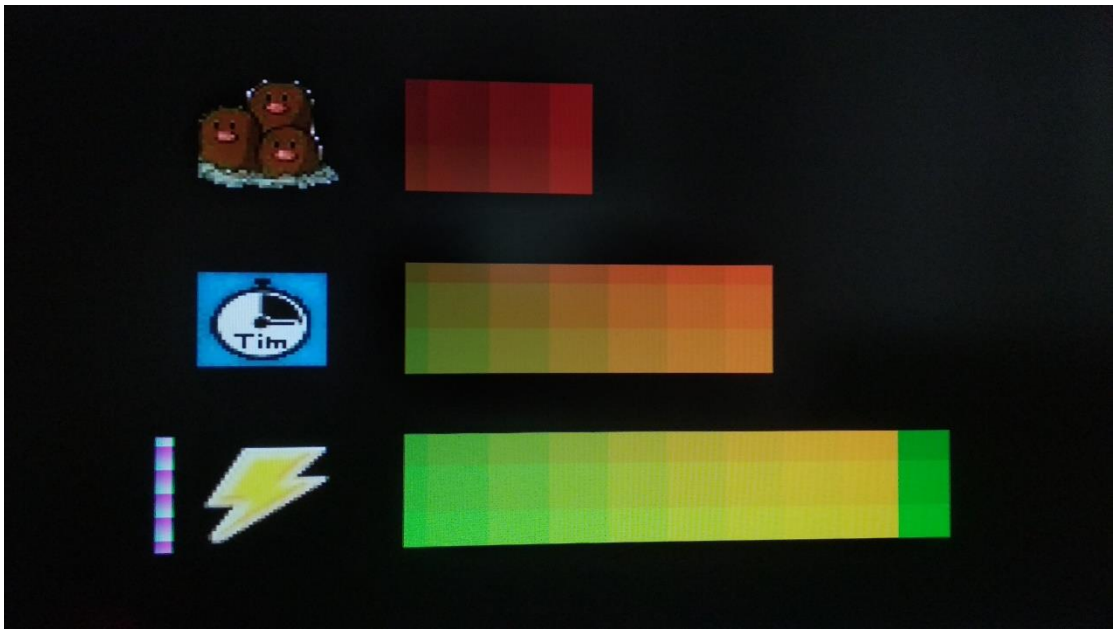
由背景與地鼠(香菇)兩張照片組成，地鼠的出現由 hole_mod 控制。使用與上面相似的做法，將地鼠放置在對應的洞上面，然後地鼠的圖片是白色背景，所以用 MUX 選擇，當地鼠的 memory 輸出白色的時候，vgaRed, vgaGreen, vgaBlue 選擇背景圖片的 pxiel。因為畫面是用掃描式的，所以隨著 h_cnt、v_cnt 改變 addr，用一個 memory block，就可以完成遊玩畫面的顯示。

(3) 結算畫面:

用一張 100*56 的照片 ，作為結算畫面，外框則是利用 h_cnt、v_cnt、pixel_addr 取 32 的餘數，個別代表 RGB 的值，得到的一個有規律的紋路。(用其他小於 32 的數字會產生截然不同的紋路)

(4) 設定畫面:

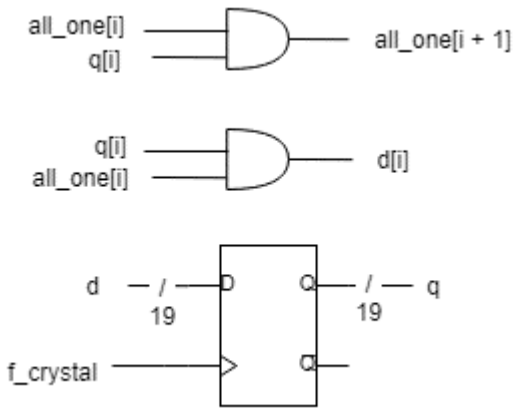
主畫面有三張 40*40 的照片，由上而下分別表示地鼠數量、次數、速度，在閃電左邊的直條代表在設定速度，會根據 state 的不同改變位置，而顏色是用剛才取餘數同樣的方法生成的。在照片的旁邊有三條橫向色帶，色帶的長度由 game_setting 的 hole_num, total_times, speed 三個參數決定，而色條的顏色也是從 h_cnt 與 v_cnt 生成，取其不同的 4bits 作為 RG，B 則設為 0。



ssd control enable generator(f_ssd_div): 產生 2bit 的 ssd control enable，作為七段顯示器的 scan control 控制項。

※首先先將 all_one[0]輸入 1

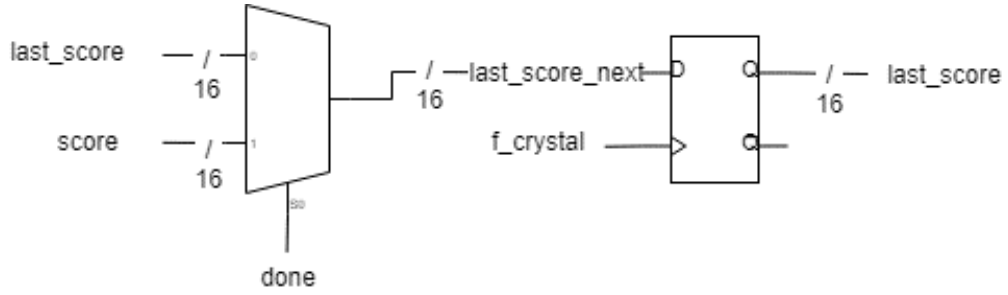
※i 為 0～18 的整數



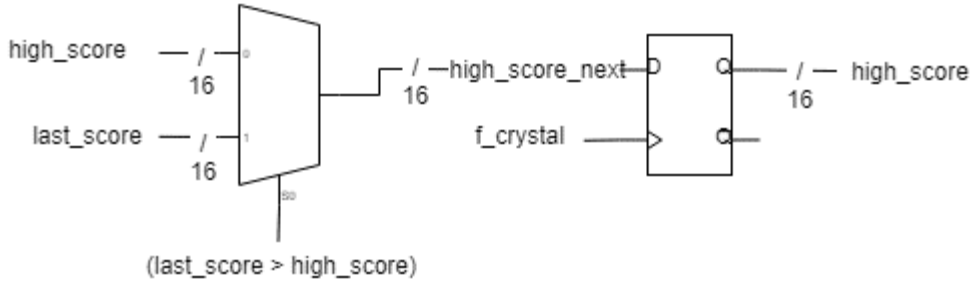
取出 q 的第 17、18bit 當作 scan control 的控制項。

分數記錄(score_recoder): 將 press control 計算的遊戲得分儲存下來，並比較此得分是否要取代掉最高得分，然後輸出最高得分和最近一次得分給七段顯示器顯示。

※done 為遊戲結束後停留在結算畫面時的 FSM 狀態



當遊戲還沒結束(done = 0) → 直接顯示遊戲當下累積的分數
當遊戲結束時(done = 1) → 將這次遊玩的分數紀錄 last_score 裡

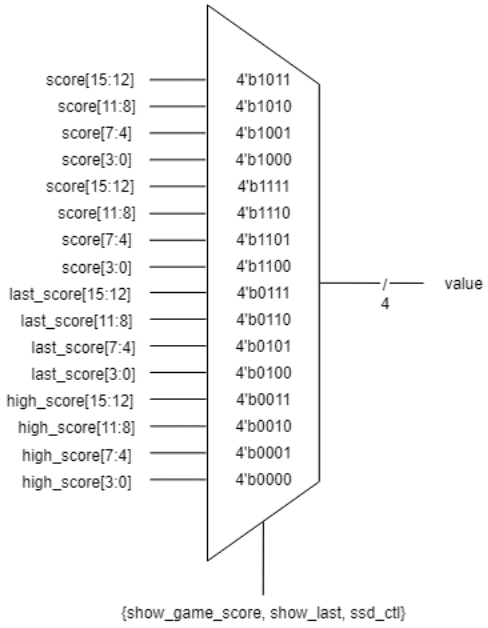


將此次遊玩的分數與先前記錄下來的最高成績做比較，若 last_score > high_score，則將此次分數記錄到 high_score 裡；last_score <= high_score，則 high_score 值不變。

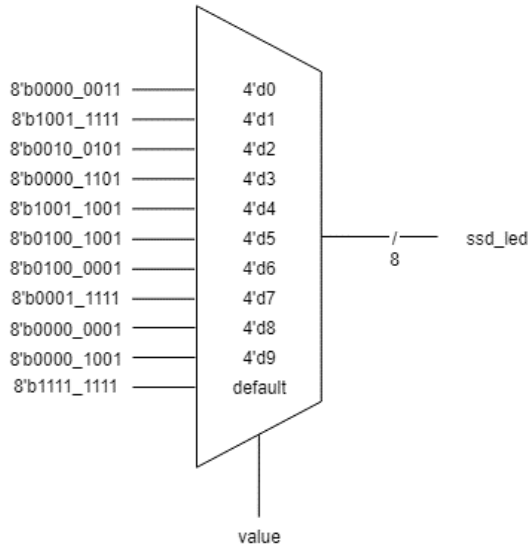
七段顯示解碼器(SSD_display): 透過當前狀態判斷要顯示遊戲進行時的即時分數、目前最高分紀錄，或是最近一次的遊戲得分，並藉由 2bit 的 ssd control enable 控制項，快速輪流顯示四個七段顯示器，使得七段顯示器可以同一時間顯示四位數得分。

ssd_ctl = 00 → ssd_en = 1110
ssd_ctl = 01 → ssd_en = 1101
ssd_ctl = 10 → ssd_en = 1011
ssd_ctl = 11 → ssd_en = 0111
ssd_en 的 4 個 bit 快速輪流為 0(low active)，使得七段顯示器看起來同時顯示。

※show_game_score 在遊戲進行時和遊戲結束的結算畫面時為 1
※show_last 為 FPGA 板上的一個 push bottom



當 show_game_score = 1 → 顯示即時的分數。
當 show_game_score = 0 且 show_last = 1(push bottom 被按下) → 顯示最近一次遊玩分數。
當 show_game_score = 0 且 show_last = 0(push bottom 沒有被按下) → 顯示最高分紀錄。



將各個 value 值所對應到的數字進行解碼，並將分數顯示在七段顯示器上。

****
結合上述所有模組功能，我們完成了一個可以設定地鼠數量、地鼠出現總次數以及地鼠出現速度，並且做出有機率會在相同的洞出現地鼠、可以切換查看最高分紀錄/最近一次遊玩紀錄，並且將整個遊戲呈現在 VGA 螢幕上的打地鼠遊戲期末專題。
****

I/O pin assignment:

Inout:

PS2_CLK	PS2_DATA
C17	B17

Input:

f_crystal	rst_n	show_last
W5	V17	U18

Output:

##7 segment display on FPGA

ssd_led [7]	ssd_led [6]	ssd_led [5]	ssd_led [4]
W7	W6	U8	V8

ssd_led [3]	ssd_led [2]	ssd_led [1]	ssd_led [0]
U5	V5	U7	V7

ssd_en [3]	ssd_en [2]	ssd_en [1]	ssd_en [0]
W4	V4	U4	U2

##LEDs on FPGA

led[11]	led[10]	led[9]
U19	E19	U16

led[8]	led[7]	led[6]	led[5]	led[4]
L1	P1	N3	P3	U3

led[3]	led[2]	led[1]	led[0]
W3	V3	V13	V14

##VGA Connector

vgaRed[3]	vgaRed[2]	vgaRed[1]	vgaRed[0]
N19	J19	H19	G19

vgaBlue[3]	vgaBlue[2]	vgaBlue[1]	vgaBlue[0]
J18	K18	L18	N18

vgaGreen[3]	vgaGreen[2]	vgaGreen[1]	vgaGreen[0]
D17	G17	H17	J17

vsync	hsync
R19	P19

Problems Encountered & Solutions:

除了影片中討論的問題之外，由於本次專題是兩個人一起完成，因此寫 code 時必須要寫的很容易讀懂，排板也要清晰，而在變數的取名上也花了很多功夫，這些都是之前自己做時不一定會如此嚴謹的地方，畢竟之前或許會覺得只要自己看得懂、寫得出來就好了，因此經過這次的期末專題，我們都學習到了很多關於這方面的解決辦法，並且更加了解容易讓人讀懂的 coding 應該長怎樣。

這次期末專題的 bit 資料遠遠多於之前的 lab 作業，每次要測試時，最少也都要燒錄至少 5、6 分鐘，這一點是相當耗時的，因此後來我們決定讓能在 FPGA 板上能夠初步顯示的功能就先不接上螢幕來做 debug，稍微減少了一點時間，不過必須要顯示在螢幕上的畫面依舊耗了不少時間。

我們兩個這次都是第一次自己親自做出一款遊戲，雖然過程中可能會不斷地有挫折，但是每一次成功了某項功能，進度往前多走了一步，就會覺得很有成就感，這次的期末專題把我們之前學過的、甚至是沒學過的螢幕顯示都結合在一起，因此透過之前的 lab 來熟悉邏輯觀念、verilog 的語法功能真的非常重要，透過這次跟隊友的合作經驗，我們都學習到了許多以前不會注意的細節，並完成了一個自己設計出來的遊戲，雖然過程很累，但我們的付出一定會得到更多的收穫。

※分工:
林禹陞—final project proposal、coding、report
張華泰—coding、VGA display coding、report

※參考資料:

iLMS 數位學習系統—邏輯設計實驗課程—產生隨機數字方法
iLMS 數位學習系統—邏輯設計實驗課程—VGA 相關解說
iLMS 數位學習系統—邏輯設計實驗課程—上課教材—VGA Memory Mapping