EECS 2070 02 Digital Design Labs 2019

Lab 7

學號：107000115　姓名：林珈卉

1) 實作過程
    a) lab07_1
        i. 設計概念
            (1) 需要讓圖片可以左右移動
            (2) 可以使用 en 跟 dir 來控制是否移動以及移動的方向
            (3) 圖片移動時，左右都會連接著同一張圖片一起移動
        ii. 實作方法
            (1) Top module 會呼叫 mem_addr_gen, clock_divisor, blk_mem_gen_0 及 vga_controller
            (2) mem_addr_gen 負責將每個 pixel 應該要放的顏色與圖片對應位置的編號傳給 top module
            (3) 圖片大小為 320 * 240，所以 pixel_addr 會介於 0 和 76800（不含）之間
            (4) 將 top module 的訊號輸入 mem_addr_gen 作為 input wire，以控制圖片移動的方式
            (5) 宣告變數 reg [8:0] position，記錄圖片移動了幾個 pixel
            (6) 設計一個 always block，以 posedge clk or posedge rst 去 trigger
                (a) 若 rst==1，position = 0
                (b) 使用 en 來判斷，若為 0 則 position = position，即不移動
                (c) en 為 1 時，則判斷 dir
                    (i) 若 dir==0，圖片向右移動

1. 若 position < 319，則 position = position +1，因為 position 的範圍是 0 至 319
2. 否則 position = 0，回到圖片沒有位移的狀況

(ii) 若 dir==1，圖片向左移動

1. 若 position > 0，則 position = position - 1
2. 否則 position = 0

(7) pixel_addr 則 assign 為 ((h_cnt>>1) + position) + 320*(v_cnt>>1))

(a) 但須先判斷 ((h_cnt>>1) + position 有沒有超過 319，若有就必須先減去 320

b) lab07_2

i. 設計概念

(1) 設計出兩個模式，shift 及 split

(2) shift 模式可以讓圖片由右而左被黑屏，待全部消失後再由上往下出現

(3) split 模式讓圖片分成四塊，每一塊朝不同的方向向外移動，移動出去後的背景為黑底，直至全部都移動到屏幕外

ii. 實作方法

(1) 由於 shift 與 split 的訊號是用按鈕控制，兩個訊號皆須經過 debounce 及 one_pulse

(2) 由於 shift 的模式會先由右到左消失，再從上到下出現，我宣告了一個變數 appearing 來判斷現在是消失中 (0) 還是出現中 (1)

(3) 設計一個 finite state machine

(a) state 分為三個，init、shifting 及 splitting

(4) 設計三個 always block

(a) 一個 always block 由 posedge clk or posedge rst

trigger

(i) rst 時，state = init，position = 0，appearing = 0

(ii) 否則 state = next_state，position = next_position，next_appearing = appearing

(b) 第二個 always block 則控制 next 的值

(i) 以 case 判斷 state

(ii) init state

1. 判斷是否按下 shift 按鈕，若有則 next_state = shifting，next_position = 319（最右邊開始），next_appearing = 0（先消失）
2. 判斷是否按下 split 按鈕，若有則 next_state = splitting，next_position = 0

(iii) shifting state

1. 判斷 appearing 的值
2. appearing 為 0，代表現在正在由右向左消失
    a. position > 0 的話，next_position = position - 1
    b. 否則 next_appearing = 1，next_position = 0
3. appearing 為 1，代表現在正在由上往下出現
    a. position <319 的話，next_position = position + 1
    b. 否則 next_state 回到 init，next_position = 0

(iv) splitting state

1. 若 position < 159，next_position = position + 1（位移量）
2. 否則 next_state 回到 init，next_position = 0

(c) 宣告 output reg 變數 black，若 black == 1 則代表該 pixel 需顯示黑色

(d) top mdule 修改成 assign {vgaRed, vgaGreen, vgaBlue} = (valid==1'b1 && black==0) ? pixel:12'h0;

(e) 第三個 always block 則控制 output reg 訊號 pixel_addr

(i) init state
1. black = 0
2. pixel_addr = (h_cnt>>1) + 320*(v_cnt>>1)

(ii) shifting state

(iii) 由右向左消失時，appearing == 0
1. 若 h_cnt>>1 > position，black = 1
2. 否則 black = 0，pixel_addr = (h_cnt>>1) + 320*(v_cnt>>1)

(iv) 由上往下出現時，appearing == 1
1. 若 v_cnt>>1 > position，black = 1
2. 否則 black = 0，pixel_addr = (h_cnt>>1) + 320*(v_cnt>>1

(f) splitting state

(i) 根據 h_cnt 及 v_cnt 分為四塊判斷
1. 如果在位移量 position 以內，black = 1
2. 否則改變 pixel_addr 的算式
   a. 左上為 (h_cnt>>1) + 320*((v_cnt>>1) + position)

b. 左下為 (h_cnt>>1) + position + 320*(v_cnt>>1)
c. 右上為 (h_cnt>>1) + 320*(v_cnt>>1) - position
d. 右下為 (h_cnt>>1) + 320*((v_cnt>>1) - position)

2) 學到的東西與遇到的困難
   a) lab07_1 的圖片會往上一個 pixel，且使用 %320 時會出問題，所以需要特別判斷當 ((h_cnt>>1) + position) > 319 時要減去 320
   b) lab07_2 在判斷 splitting state 左上的 pixel_addr 時，會遇到 120-position overfloaw 的問題，必須要先判斷 position >= 120 時， black = 1
3) 想對老師或助教說的話
   a) Sample code 提供的範例圖片讓人看了很心寒（畢竟二退時間還沒過...），下一屆可不可以換一張圖片給學弟妹，才不會打擊太大ฅ_ฅ