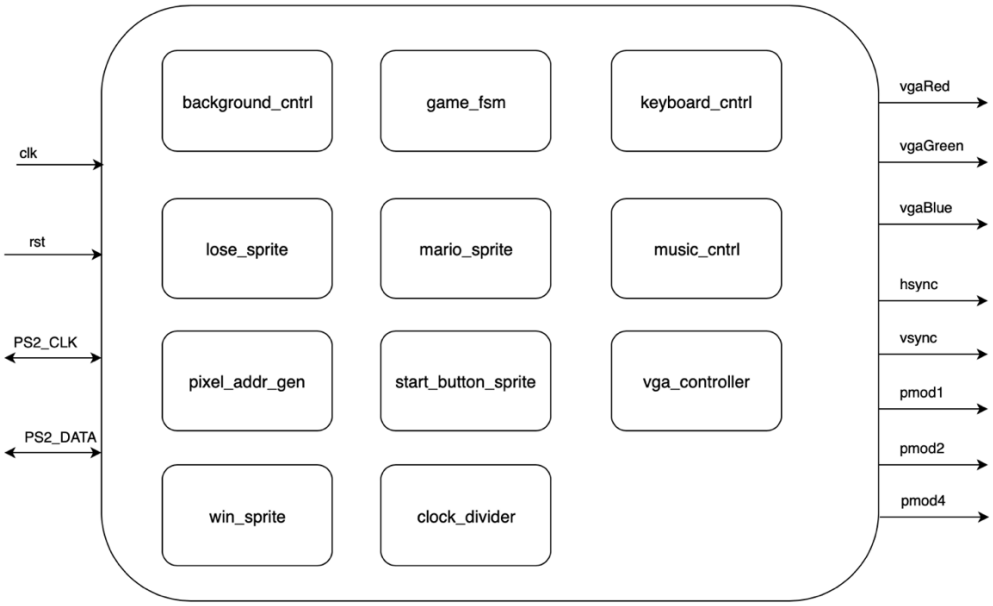
Final Project Report
Team 6

1. 發想過程:
   在 11/30 開始進行初步討論,一開始是在到底要製作遊戲還是要製作夾娃娃機在做討論.後來我們查了一下夾娃娃機的相關資料,在網路上並沒有人用 fpga 為基底來製作夾娃娃機,不過有看到一個對 Arduino 來做設計的夾娃娃機,這變成一個我們一個很重要的參考資料.不過後來我們考量到器材,Fpga 的限制,以及外接馬達等等困難,決定來試試製作遊戲.
   就目前我們所學,我們可以運用的器材是喇叭,滑鼠,鍵盤以及螢幕.而能用 fpga 來製作的遊戲也十分有限,尤其是我們 fpga 限制較多,例如不能同時接兩個 usb 對我們來說就是一個很大的困難.雖然我們有考慮申請較進階的 Fpga,不過目前還是用目前有的 fpga basys3 artix-7 來做 project.
   在眾多遊戲中,較簡單且較多相關資料的不外乎俄羅斯方塊,ping pong,太鼓達人,以及馬力歐.在這些遊戲中,馬力歐的製作難度很明顯的較其他遊戲高.不但要多寫很多腳色以及地圖設計,操縱方面也較複雜.不過相對的,馬力歐這款遊戲可塑性高,很容易從原始的遊戲慢慢發想,增加屬於我們的創意.於是我們選了這款難度較高但易於發想的遊戲來當作我們 Project 的主題.

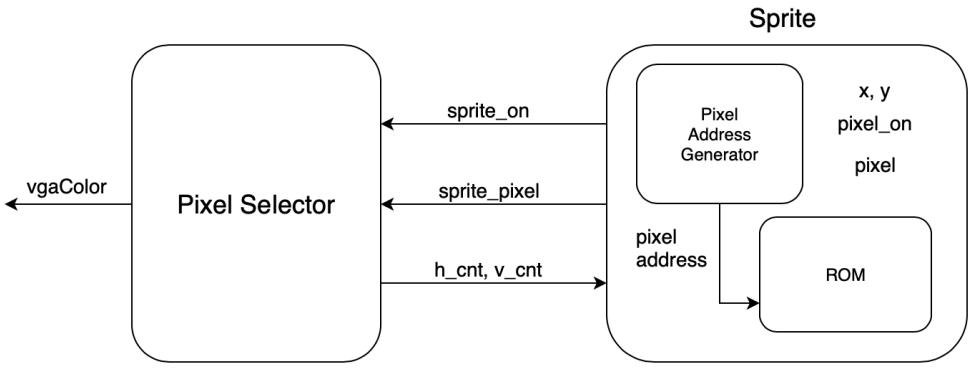
2. 製作過程:
   在最一開始,我們是將 lab6 為模板做更改,並慢慢加上多張圖片及音樂.一開始很掙扎到底要不要畫大地圖,利用部分顯示來做出移動的感覺,後來還是選用讓背景往後的方式來讓馬力歐感覺是前進的.另外,馬力歐的遊戲正常來說不能往回走,所以磚塊並不會因為馬力歐往回走而回來.

3. Design Specification:
   a. Top module: game_top
      i. Block Diagram



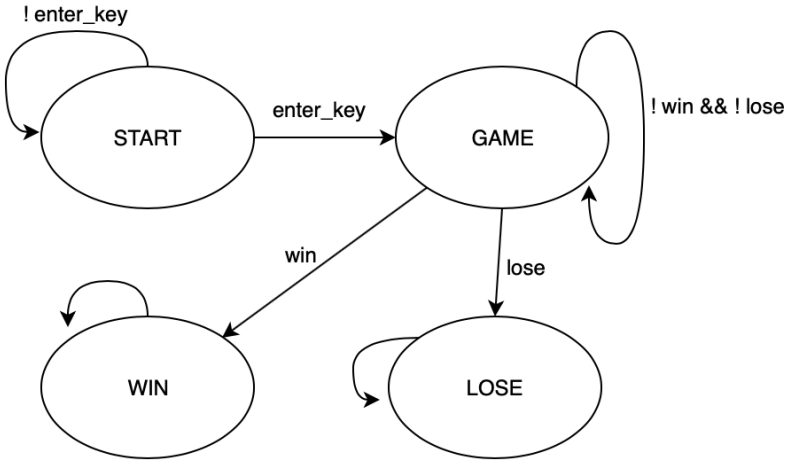
ii. Explanation:
    1. This module serves as the top module and a pixel selector.
    2. The pixel selector selects which pixel to be output.

b. The display mechanism:
    i. Block Diagram:



    ii. Sprite:
        1. A sprite is an object that can be displayed.
        2. Each sprite contains a pixel address generator and a ROM.
        3. A sprite uses a block memory as a ROM to read the pixels.
    iii. Pixel Address Generator:
        1. A pixel address generator takes in the position of the sprite and (h_cnt, v_cnt), and outputs signal *pixel_on* and the pixel address.
        2. Signal *pixel_on* tells the pixel selector whether the pixel should be displayed or not.
    iv. Pixel Selector:
        1. Selects the pixel to be shown from all the sprites.
        2. It goes through all the *spriteName_on* signals to check if the current position VGA is pointing at (h_cnt and v_cnt) is in the region of any sprite, if yes, output the pixel of the sprite.
        3. If no sprite needs to be displayed, output the default pixel.

c. Finite state machine of the game: game_fsm
    i. State transition diagram

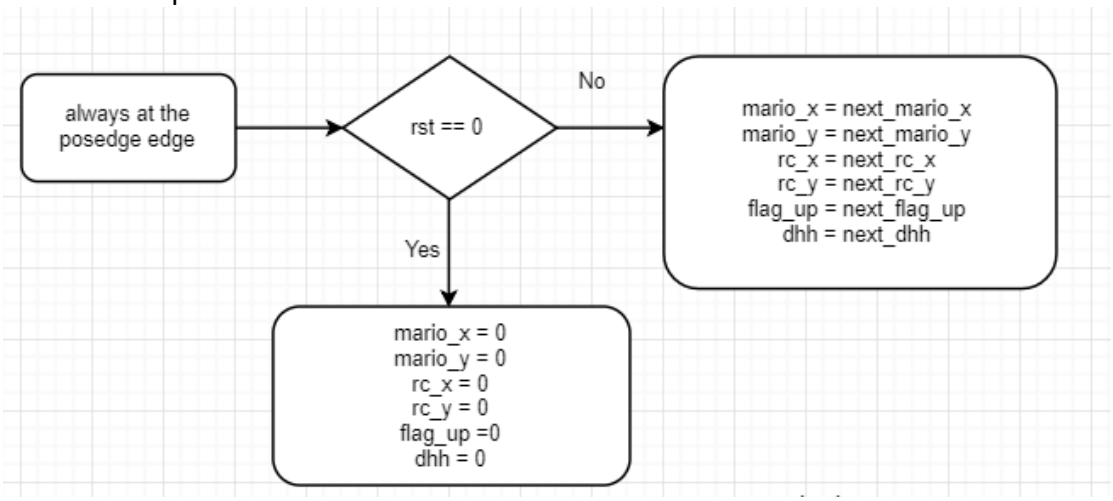


    ii. Explanation:
        1. The module controls the state of the game.
        2. State START: the state that displays the start button. When enter key is pressed, state transitions to GAME. If not, it stays at START.
        3. State GAME: the state of the game play process. If signal *win* or signal *lose* is invoked, state transitions to the corresponding state, if not, it stays at GAME.
        4. State WIN and State LOSE: the state when the game ends and displays an icon to tell the player wins or loses. Once in the state, it never gets out.
        5. Side note: the original design transitions from WIN or LOSE to START at a press of the enter key, however, it causes some inconceivable bugs and is therefore revised into a bug-free, but dumber state transition.

d. Modules that use code from labs throughout this semester:
    i. Keyboard controller:
        1. Reference: keyboard from lab 5.
        2. Explanation: provides the enter key and keys for moving the character.
    ii. Music controller:
        1. Reference: music box from lab 5.
        2. Explanation: rewritten to play the Mario theme song.
    iii. Background controller:
        1. Reference: pixel address generator from lab 6.
        2. Explanation: originally designed to display backgrounds that corresponds to the situation. Due to limitations of hardware and the author's capability, the module is remains open to extension.

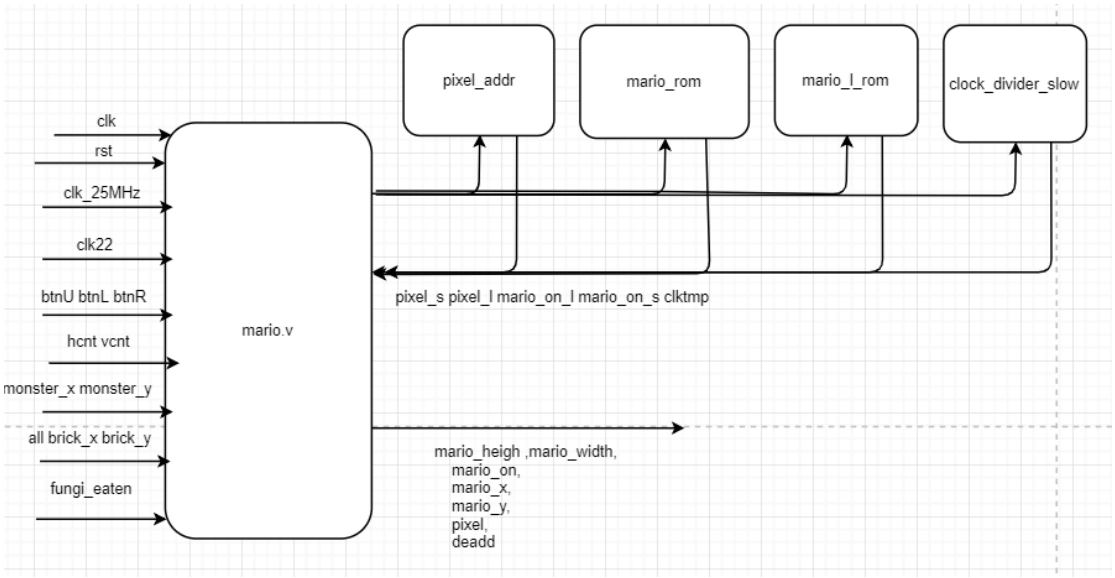
iv. VGA controller: VGA controller from lab 6.
v. Clock divider: clock divisor from lab 6.

e. Mario.v
   i. Sequential Circuit:



Mario_x 和mario_y 是指mario 的x y 座標, rc_x 和rc_y 是在跳躍的時候記錄用的, 記住一開始跳的位置,後來再回去原本的高度,flag_up 是紀錄現在在跳躍的哪個階段, dhh 是拿來判斷死掉後的彈跳是往上還是往下

   ii. More information:

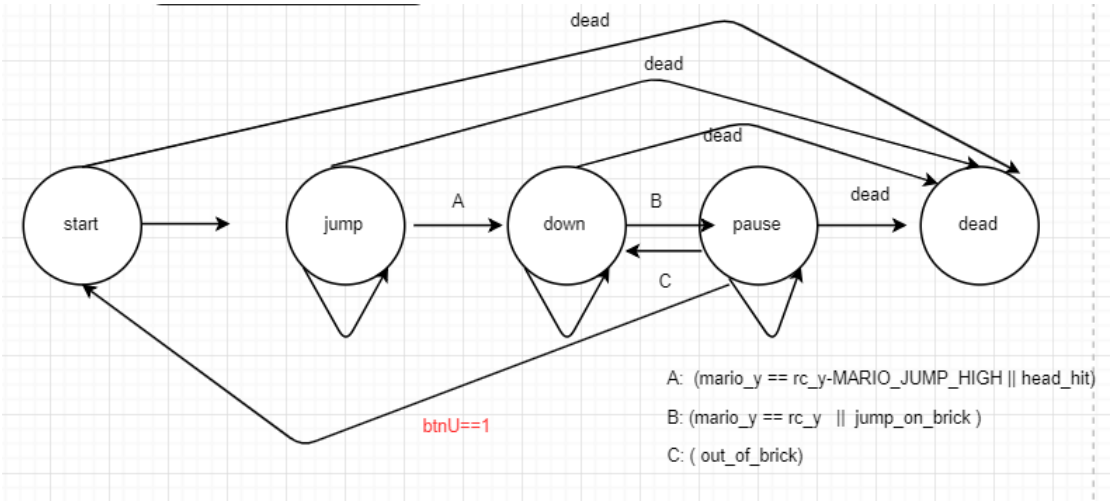


在mario.v裡面,需要知道現在怪獸的座標,磚塊的座標以及吃到香菇了沒,再外接 pixel_addr, Mario_rom ,Mario_l_rom 來做出圖片.在這裡需要兩個rom是因為要產生大圖和小圖,如果 (fungi_eaten==1)的話就會讓大圖顯示.clock_divider_slow 是為了要讓他能夠正常的移動,如果沒有加這個module 的話mario在移動的時候會出現殘影.

在mario的移動中,需要注意一些部分:

1. mario位置的限制:
   mario的位置不能低於地面,mario在向右走的時候不會走到螢幕邊邊,而是會卡在一個邊界上,因為本來的遊戲就不會讓他到底端(除非走到地圖盡頭),在彈跳的過程中,Mario 會被磚塊們擋住

   這邊是mario跳躍的state transition diagram:

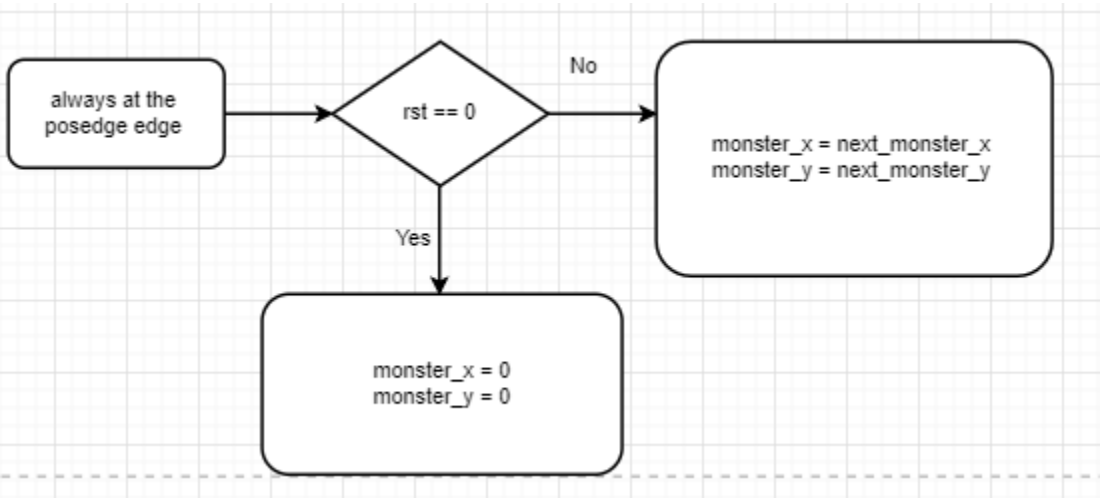


2. mario 吃香菇
   Mario 要吃香菇需要先撞一下方塊que,然後看到香菇出來,要跳上去吃,吃完後他會變大
3. mario 與怪獸
   Mario 在還沒吃香菇前 (fungi_eaten ==0)如果中途碰到怪獸會直接原地跳起然後往下掉,遊戲結束.如果mario是吃了香菇後才碰到怪獸,則不會遊戲結束.不管有無吃香菇mario都可以從怪獸頭上踩下去,怪獸就會消失.
      1. 名詞補充:

Brick_x_long : 為了處理磚塊到x小於50而需要減50可能產生負號,用這個來當作絕對值.
Mario_hit_left : mario撞到左邊磚塊
Mario_hit_right: Mario撞到右邊磚塊
Head_hit :Mario 撞到磚塊下面
Jump_on_brick: mario在磚塊範圍內跳
Out_of_brick: Mario 在磚塊外面且高度在磚塊外,代表從磚塊上往外面跳
Hit_monster: 撞到怪獸
In_renge_x :在磚塊的x範圍內
In_renge_y :在磚塊的y範圍內

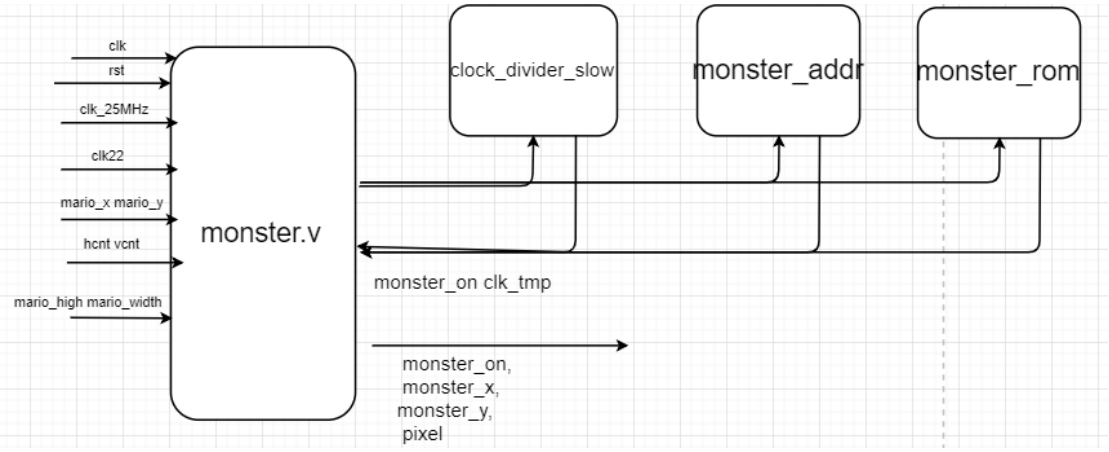
f. Monster.v:

i. Sequential Circuit:



Monster_x 和 monster_y 就是怪物的座標

ii. more information:



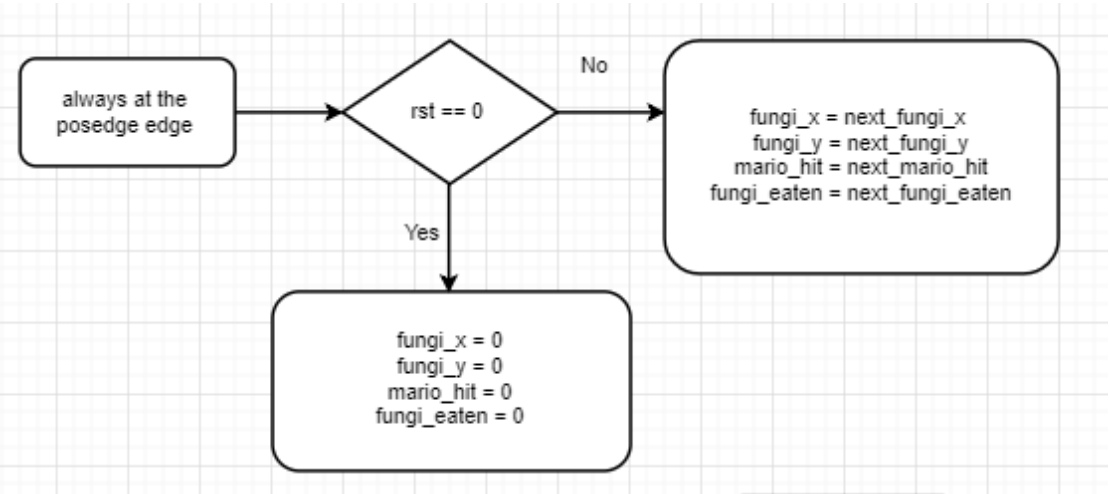
Monster.v 會讀進 mairo 的座標和高度(高度有可能會因為吃了香菇而變,因此不能用成參數而要用變數) ,利用 monster_addr 和 monster_rom 來寫入圖片,最後會輸出怪物的位置

怪物的移動方式:

怪物在還沒有被 mario 攻擊的情況下是正常的由左跑到右,並會一直重複.如果 mario 從上面往下踩到怪獸,則 monster_on 則會讀到 beat,然後怪獸就會消失.

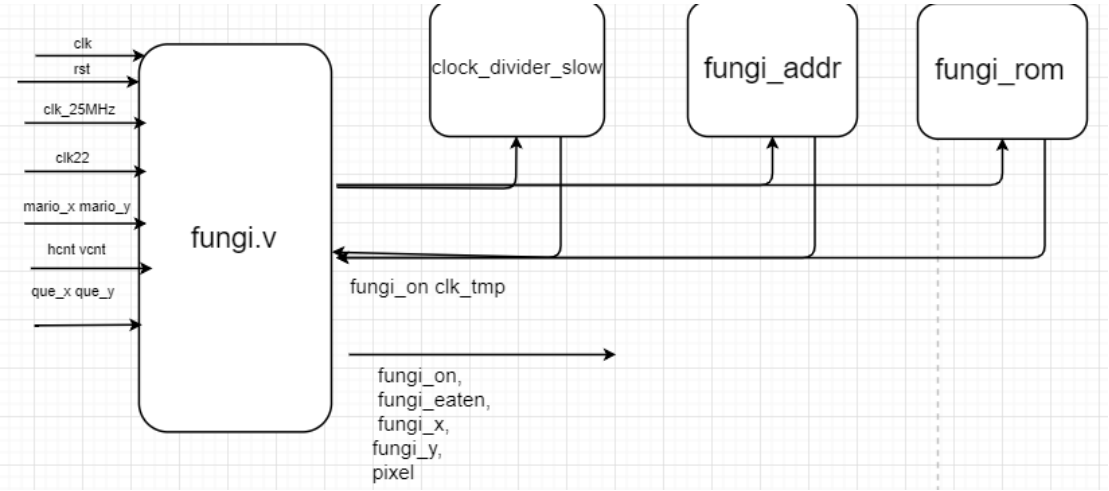
g. fungi.v:

i. Sequential Circuit:



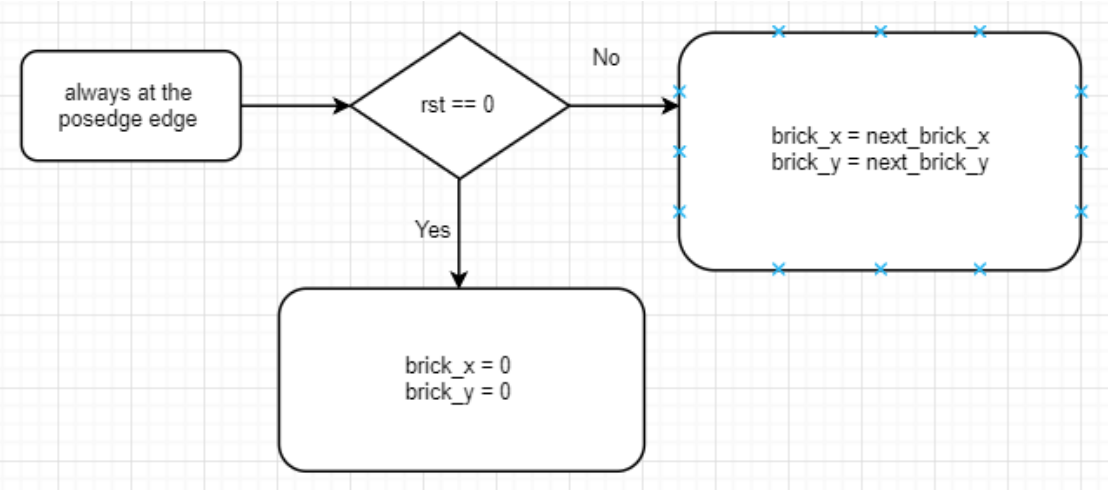
Fungi_x fungi_y 就是香菇的座標,Mario_hit 是判斷是否被 mario 打到,被打到就要往上移動,從磚塊出來,Fungi_eaten 是判斷是不是被 mario 吃到了,若被吃到就不顯示

ii. more information:



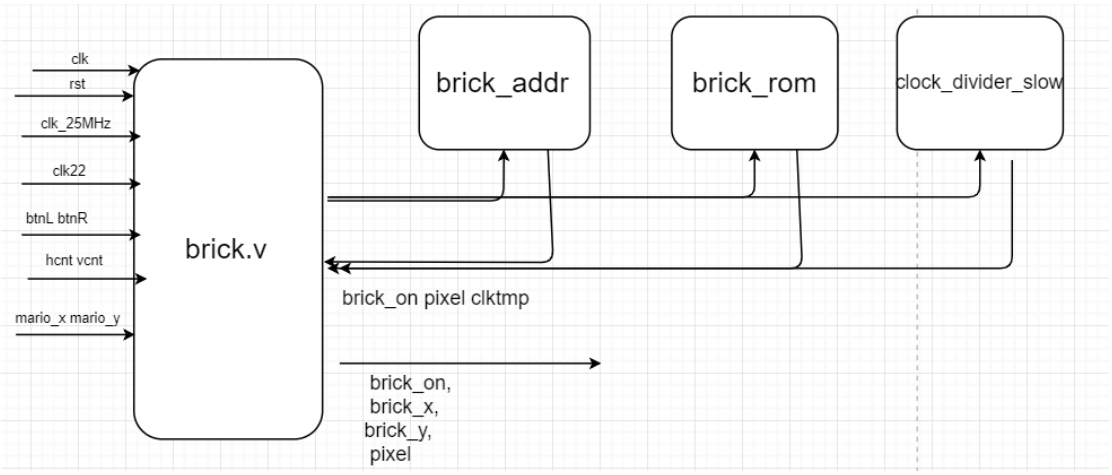
Fungi.v 會讀進 que 磚塊的位置,還有 mario 的位置.讀進 mario 的位置的原因是要判斷 fungi_eaten 是不是為 0 ,並把 fungi_eaten output 出去.如果香菇被吃掉了,香菇就會消失(fungi_on ==0)

4. Brick.v:
    a. Sequential Circuit:



Brick_x brick_y 就是磚塊的位置

    b. more information:



磚塊切成五份,四份普通的磚塊和一個問號磚塊.每個磚塊的位置基本上是寫死的,不過會配合 mario 的位置來往後移動,目的是為了讓 mario 看起來是向前進的.當磚塊移動到左邊盡頭的時候,磚塊會直接消失不見.

5. 遇到的問題:
    i. load 放大圖
    一開始並不知道要怎麼樣讓他吃到香菇後變成大的圖片,後來利用 on 的判斷式,判斷現在的圖片要用大圖片還是小圖片

    ii. 磚塊邊界
    馬力歐在撞到磚塊時要被擋住,所以必須要把磚塊的邊界都弄出來,再來判斷有沒有重疊到

iii. 圖片處理----去背及疊圖

關於疊圖,我們將每張圖片的背景設成某種顏色,並利用產生的 coe 檔找出 rgb, 並利用 rgb 值來判斷哪張圖片在前

6. 心得:

a. 王駿

這次是第一次用 verilog 做 project, 最頭痛的應該就是顯示畫面的部份,因為對這部分比較不熟悉,摸了很久才搞懂.這份 project 也幫我複習整學期所學,尤其是產生圖片及運用的部分,令我受益良多.

b. 陳騰鴻

- Understood how hard collaborating with others is, and how crucial tools like Git Hub is to collaboration code development.
- Felt that you might fail many times if your dream is big, but you can't achieve big things without dreaming big.
- Understood brainstorming requires thinking, expressing yourself, and most important of all, listening to others.
- Realize the importance of designing before implementing (coding).
- Developed skills of debugging.