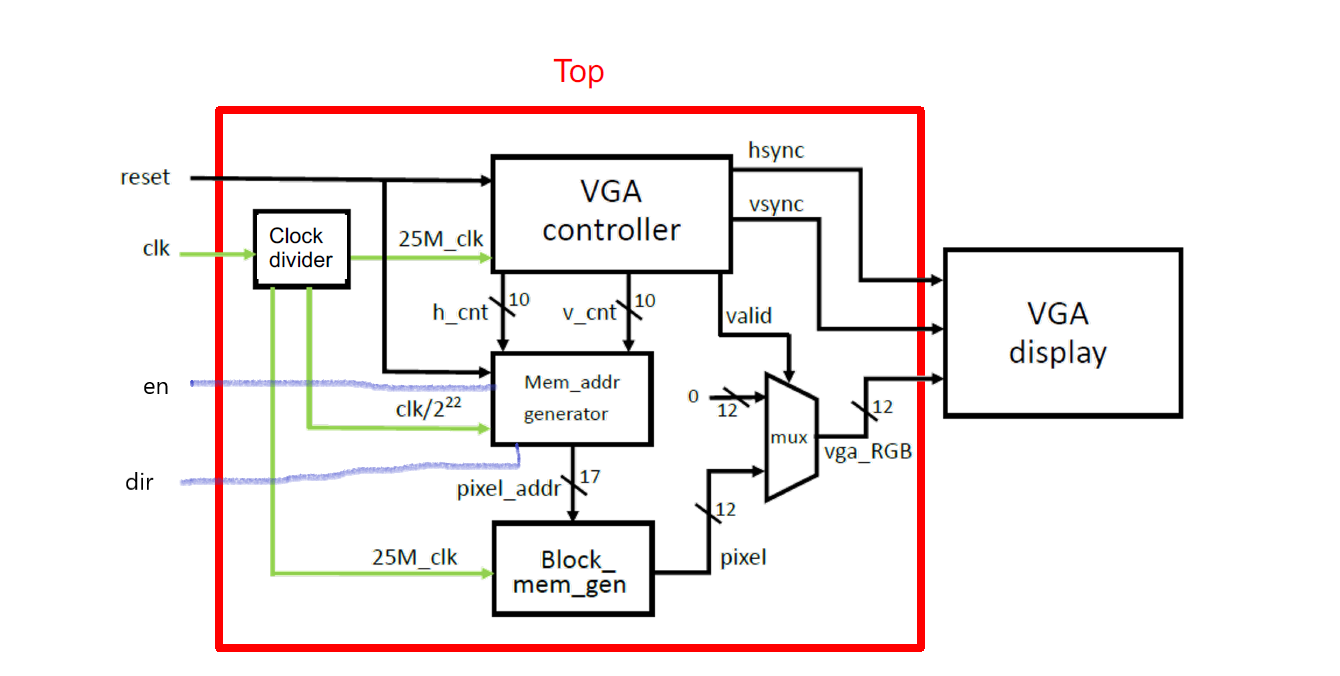
|  |  |
| --- | --- |
| **Lab 7** | |
| 學號: 109062318 | 姓名: 簡弘哲 |

1. 實作過程

7-1:



因為7-1跟上課講義裡的demo2差不多所以我就將demo2的code拿來做小修改，將mem\_addr\_gen這個module加上兩個input signal en, dir就完成了。

Scrolling:

一張含有 文字 的圖片

自動產生的描述

根據dir的值決定是往上還是往下，判斷position的值即可，僅須注意邊界239,1。

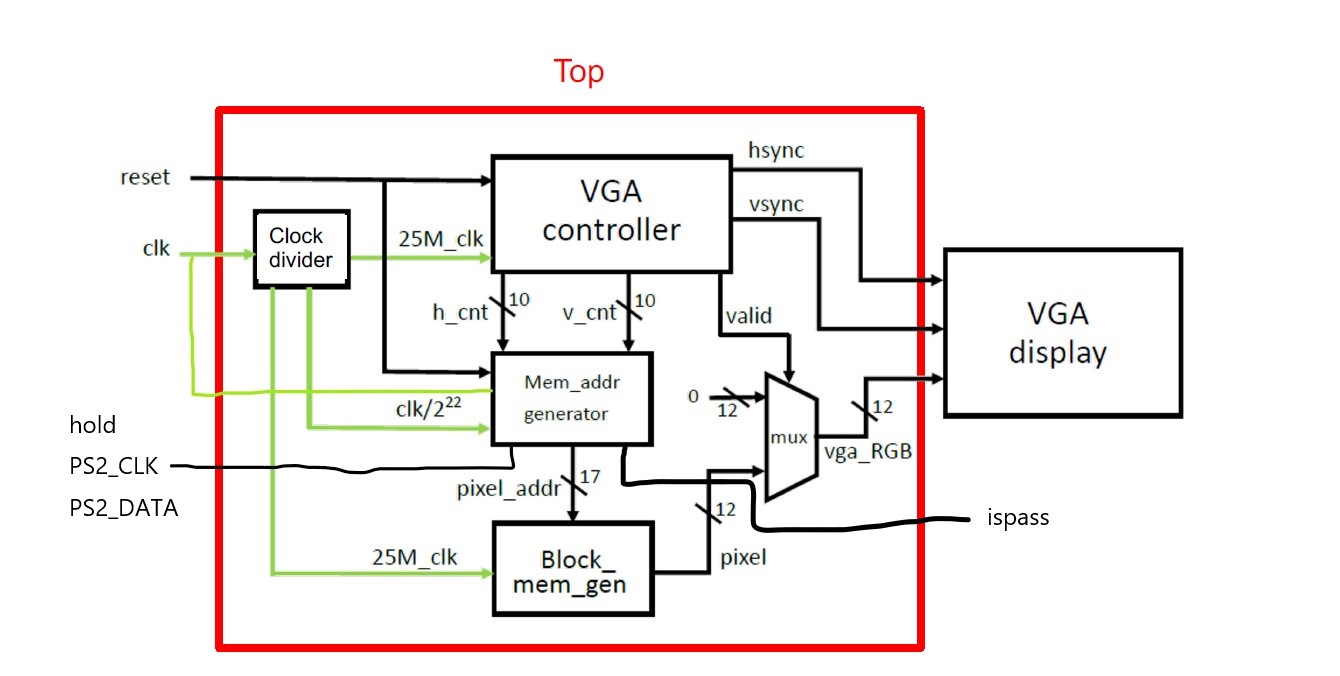
Negative film:

一張含有 文字 的圖片

自動產生的描述

Negative film非常簡單，用bit complement(~)就可完成。

7-2:



從7-1修改而來，在mem\_addr\_gen加一些input(hold, PS2\_CLK, PS2\_DATA), output(ispass)，因為我在mem\_addr\_gen裡有放keyboardDecoder，所以需要傳入PS2\_CLK, PS2\_DATA，檢查遊戲完成的機制也是實作在該module裡面，因此需要一個ispass的output信號。

變數意義解釋:

cnt是一個長度為12的陣列，每個元素皆為2bit，我定義cnt[i]==0就是不轉(0度)、1是順時針轉90度、2是轉180、3是逆時針轉90。

Index 0記錄圖片左上角那格的方向

3右上角

9左下角

11右下角

一張含有 文字 的圖片

自動產生的描述

上圖是實作逆時針旋轉的code，以左上角那格為例子，每次逆時針轉90度，cnt[0]的值就-1，然後根據該格目前的擺放狀態決定下一個該呈現在螢幕上的角度公式，例如: 目前圖片是擺正的(cnt[0]==0)，那逆時針旋轉後就要變成270度，4條公式都是預先算好的。

Hold, pass, 順時針旋轉:

一張含有 文字 的圖片

自動產生的描述

因為hold跟pass的效果都是顯示原圖，所以可以把他們寫在一起。Else的部分則是遊戲還在進行中，以左上角那格為例，每次順時針旋轉cnt[0]就+1，然後再根據目前圖片的狀態決定旋轉後該呈現在螢幕上的公式。

判斷遊戲結束:

一張含有 文字 的圖片

自動產生的描述

因為我怕用loop + if寫可能會造成latch的問題，所以我就直接列出12個cnt的值，當所有值都是0，也就是每一格都是擺正的，才可以宣告遊戲結束。

當沒有按鍵按下的時候:

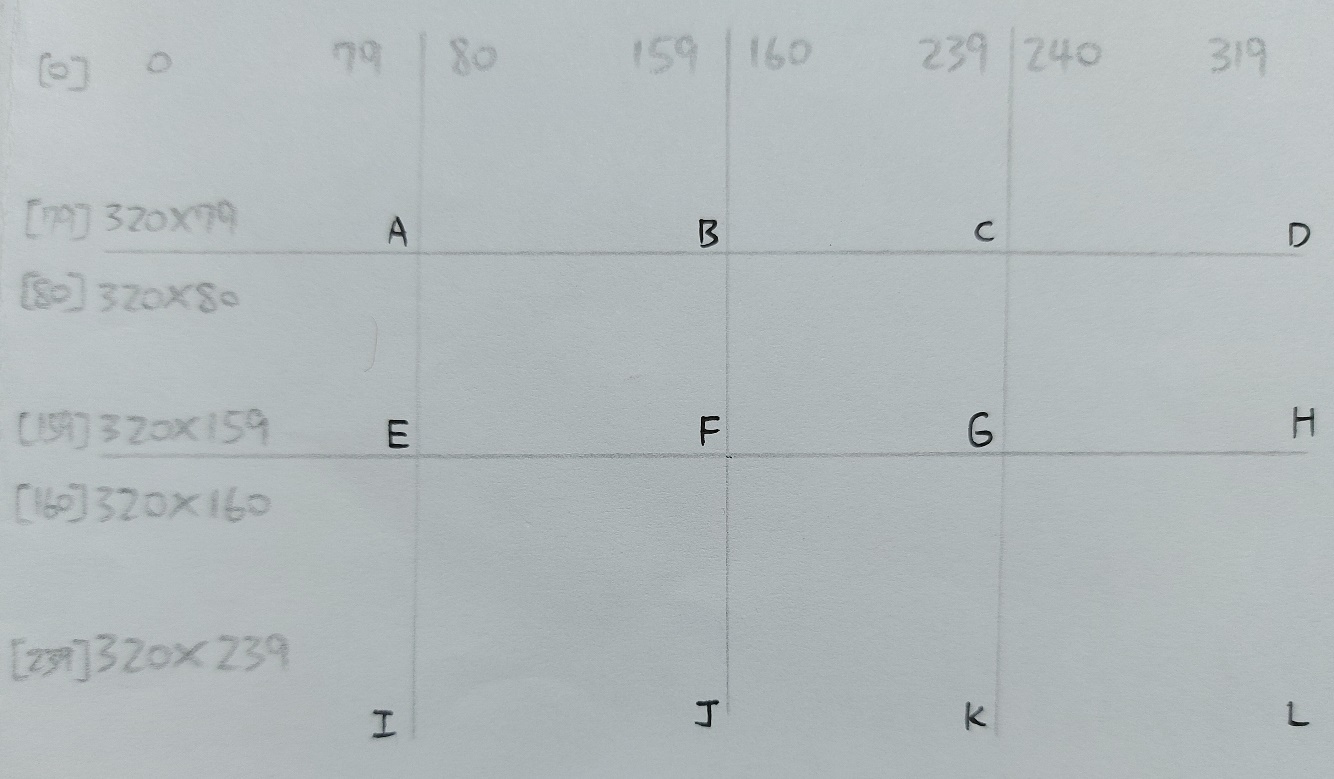
一張含有 文字 的圖片

自動產生的描述

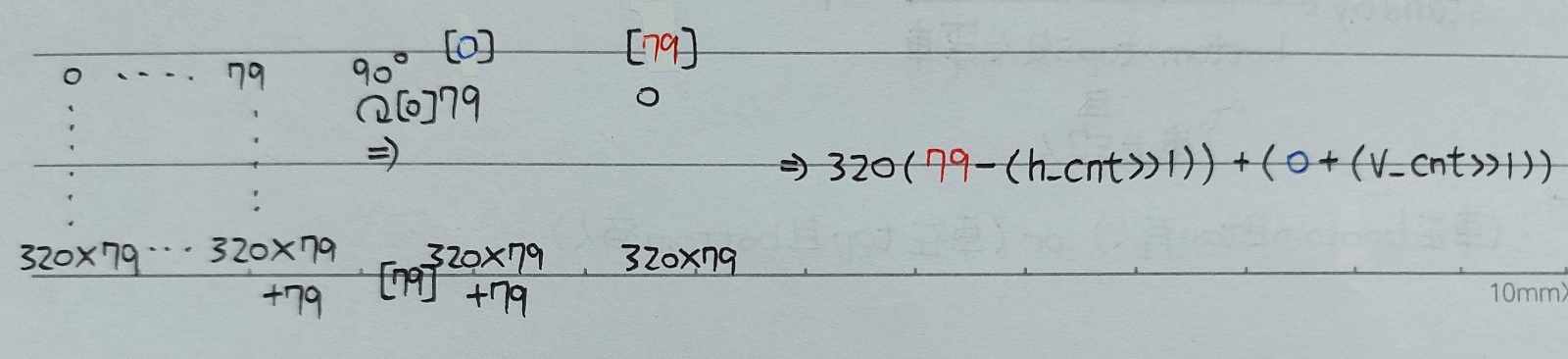
根據v\_cnt與h\_cnt的所在位置判斷目前vga.v在畫哪一格的pixel，以左上角那格為例子，如果按下hold或是遊戲結束就顯示原圖，否則的話就看該格目前的狀態並將公式寫進去，像是該格目前是順時針90度(cnt[0]==1)，那就將90度的公式套進去。

算公式:

因為照片的二維資料是存在一維陣列裡面的，所以要推算該去一維陣列的哪個index取出正確的資料。一張照片可以分成12格，如下圖所示，然後我將每一格分別命名為A到L，先寫出邊上pixel在一維陣列裡面的index值以方便計算。



以左上角那格為例，它原本四個角的index分別為0, 79, 320\*79, 320\*79+79，將該格順時針旋轉90度後如下圖所示，四個角所在的index會改變，然後將旋轉過後的index分布用column的index(0,79)去表達出公式，這需要一些時間觀察，最後得出右邊的那條式子，求其他格也是用類似的方法。



到最後歸納出12格在旋轉不同的角度(順時針90,180,270)後推算出pixel address的公式，

其中c是(h\_cnt>>1)、r是(v\_cnt>>1)

m是該格水平範圍的較小值、M是該格水平範圍的較大值

以L格(圖片右下角)來說，該格的水平範圍是240~319，因此m=240,M=319帶入公式即可。

因為我觀察到在分成12格的圖片裡面，位於同一行的格子(ex.ABCD、EFGH、IJKL)他們的旋轉公式都一樣，例如第一行(ABCD)順時針旋轉90度的公式皆為320(M-c)+(m+r)，再套入該格水平範圍的較大、較小值就可得出該格的公式。

一張含有 桌 的圖片

自動產生的描述

1. 學到的東西與遇到的困難
2. 寫7-2時一開始完全不知道怎麼下手，漸漸學會如何算pixel

一開始完全無法理解圖片要怎麼旋轉，所以就往修改vga.v的方向思考，想說應該不再是從左到右、從上到下的掃pixel，但後來覺得要是圖片轉很亂的話根本就不知道下一個該掃的pixel在哪邊，所以就認為應該是修改mem\_addr\_gen，當圖片旋轉時就去推算該出現的pixel address是多少。但我稍微想了想，有12格圖片，除了0度以外不用算，剩下的3個方向都要算，所以總共要算3\*12=36次，這數字讓我覺得有點多，可能還有更好的方法。但再想了一段時間後還是覺得只能乖乖算pixel address，別無他法了。幸好算了一些方向後有發現一些規律才不用36種都算，每次算的時候將翻轉過後pixel分布用column的index表示。

1. [7-2] latch造成reset有問題

一開始寫reset的時候永遠只能reset到最左上角的那格，其他格就重設不到，會維持原來的樣子，這困擾了我好一陣子，我一直在if(rst)那邊打轉但終究不見起色，直到我注意到vivado的warning說有infer latch，把它改掉之後一切就正常了。

1. [7-2] 的鍵盤不靈敏

我仿造lab6的寫法，將鍵盤code的部分搬了過來，但沒想到在測試的時候鍵盤竟然是那麼的不靈敏，lab6都沒有這種問題。這個問題也困擾了我許久，一直盯著code也不知道該改哪邊，正當我絕望之際，我腦中突然閃過了一個想法，”把這塊code block搬到別的地方去會怎樣?”，結果一實驗發現鍵盤就可以正常運作了。看來有時候寫不出來就把code block換個位置，說不定就會正常了。

(因為我一開始的寫法是

If(h\_cnt>>1, v\_cnt>>1的值落在左上角那格){

處理鍵盤的code…

}

後來將它改成=>

處理鍵盤的code…(搬到if外面)

If(h\_cnt>>1, v\_cnt>>1的值落在左上角那格){

其他的code…

}

就正常了，看起來是處理鍵盤的code寫外面會比較好

)

1. 想對老師或助教說的話

在verilog裡的for loop有break這個關鍵字嗎?