|  |  |
| --- | --- |
| **Lab 6** | |
| 學號: 109021115 | 姓名: 吳嘉濬 |

1. **Lab Implementation**

以下是lab6\_1的block diagram：

一張含有 文字, 圖表, 筆跡, 行 的圖片

自動產生的描述

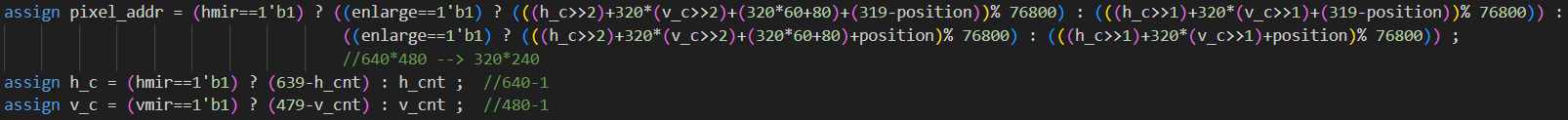
我們主要實作的部分為mem\_addr\_gen的block，基本運作原理是依照需求的功能在給定h\_cnt, v\_cnt的情況下決定你下個cycle的pixel\_addr，再利用blk\_mem\_gen\_0的frame buffer，去找到對應address的data，即pixel顏色的資料，進而顯現在螢幕上。

以下為kernel code：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

每經過clk\_22的clock cycle，根據en和dir值，position會有對應的增減，達到在螢幕上雙向水平移動的效果。

因為lab6\_1形似demo2，所以我的coding style仿造demo2，直接利用assign去決定下個cycle的pixel\_addr：

先看第二個和第三個assign statement，這是處理垂直和水平鏡像時做的變化。

第一個assign考慮是否水平鏡像和是否要放大，所以總共有4種可能的計算。如果要水平鏡像，offset不能是單純的+position，而是要+(319-position)(即offset從左到右變成從右到左)。enlarge的部分，我希望圖案呈現是放大兩倍，而且要放大正中間，所以offset要另外加上(320\*60+80)，如此畫面的左上角才會呈現往正中間放大兩倍後的圖案的左上方。另外，我讓h\_c<<1, v\_c<<1變成h\_c<<2, v\_c<<2，如此，每連續2\*2個pixel都會對應顯示放大前1\*1個pixel的畫面，以達到放大2倍的效果。

以下為lab6\_2的FSM：

一張含有 文字, 字型, 筆跡, 行 的圖片

自動產生的描述

按下rst時，進入SOLVE state，puzzle會回到原本初始被打亂(非隨機)的狀態，當每個區塊對應初始圖片的位置正確且上下皆沒有顛倒時，pass會被assign成1’b1，進入END state，從此不能再交換圖塊或是把某圖塊上下顛倒，直到再次被按下rst。

自己實際上在實作時其實沒有分開成2個state設計，但都有確保機制的正常運作。

以下為lab6\_2的block diagram：

一張含有 文字, 圖表, 筆跡, 方案 的圖片

自動產生的描述

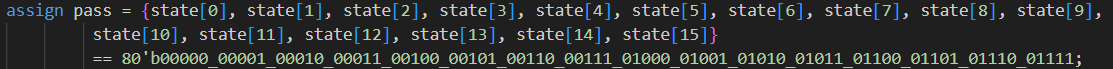
不同於lab6\_1，這次加上了鍵盤的控制，但基本上原理和lab6\_1不會差太多，主要問題在於如何實作兩個區塊互換，以及讓某區塊旋轉180度，還有如何判定puzzle已經解開了。

首先，先說明基本的原理，我讓螢幕依照位置切成4\*4 16塊，從左到右、從上到下，編號0到15。以下為對應code segment：



每一區塊存5個bit的資料，其中最左邊的bit, i.e. state[x][4]，儲存此圖片是否upside down，1’b0表示沒有，1’b1表示圖片是旋轉180度的，剩下4個bit, i.e. state[x][3:0]，表示該區塊正在顯示「未被打亂圖片時，第”state[x][3:0]”個位置」對應的區塊。

因此我們可以判斷出何時pass可以被set成1’b1：



每個區塊正在顯示未被打亂前對應位置的區塊的畫面，並且皆沒有被upside down。

以下為交換兩區塊以及旋轉某區塊180度的kernel code：

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

判斷出第i個區塊對應的key和shift鍵被同時按下時，我們讓next\_state[i]={~state[i][4], state[i][3:0]}，讓最左邊的bit從0變成1或從1變成0，改變旋轉狀態。

判斷出第i個區塊對應的key和第j個區塊對應的key被同時按下時，我們讓next\_state[i]=state[j], next\_state[j]=state[i]，第i個區塊和第j個區塊的資訊交換，達到交換區塊的效果。

最後則是如何利用以上資訊去決定每個cycle的pixel\_addr，以下為對應kernel code：

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

其中ho和vo儲存16個區塊的水平方向offset(horizontal offset)和垂直方向offset(vertical offset)，利用ho和vo的值，可以幫助我們在不同區塊時去計算下個cycle的pixel\_addr(i.e. next\_pa)。

1. **Questions and Discussions**

A. If we want to turn the image with a special effect of the negative film (which means each pixel is complemented in color, like the example shown in below, the word ’POPCAT’ is original white, but after complemented, it is shown in black), how would you modify your design of lab6\_1?

把這一行



改成這一行



即可達到negative film的效果。

因為pixel記錄著在某個cycle時某一格pixel要顯示的顏色，其中每4個bit一組總共三組，分別代表著R(red), G(green), B(blue)的intensity。我們把RGB每個元素各自的intensity變成原本的互補，也又是從4’hx變成4’hf-4’hx (i.e., 15-x)，即可達到負片的效果。(其中x是介於0到f(15)之間的整數)

B. Suppose we want to create a VGA game with animations, how can you design the frame buffer and let the item on the monitor moves? (hint: you can use two frame buffers)

使用兩個frame buffers，我們可以在其中一個frame buffer(called A)在display到螢幕上時，讓另一個frame buffer(called B)可以同時update下一個將要display的畫面，當進入下一幀時，A, B兩者角色互換，換成frame buffer B display畫面，frame buffer A去update下一個畫面，一直這樣反覆下去，如此能達到比較流暢逼真的連續畫面效果。

C. Our FPGA equips with the BRAM of only 1800 Kbits, which a 640×480 image cannot fit in. If we want to implement a video game, apart from storing a smaller image (e.g., 320×240) like we did in this lab, please give at least 2 possible methods to reduce the BRAM usage.

其中一個方法是image tiling，將原始圖片切分成許多小區塊，稱為tile，並把圖片中出現的小區塊組合儲存成一個set，如此一來，顏色一樣但位置不同的色塊不會被重複儲存以達到節省BRAM使用的目的。

另一個方法是使用RLE(run-length encoding)，把連續同個顏色的pixel以顏色及長度資訊來儲存，即可達到節省BRAM使用的目的。例如，有連續100個白色pixel時，我們與其使用1200個bit儲存{12’h0, 12’h0, …, 12’h0}，不如直接儲存{12’h0, 12’d100}(假設使用12 bit去儲存連續相同顏色pixel的長度)。

1. **Problem Encountered**

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

在實作兩個色塊位置對調時，我一開始是使用以上的code，其中next\_state[i]=state[i]; 這一行是為了給每一個色塊default的assignment，但是實際在跑時，我發現欲交換的色塊假設為A, B，理當在交換後會變成B, A，但事實上卻是變成A, A或B, B，於是我重新trace這一段code，發現問題就是出現在next\_state[i]=state[i];這一行出現的位置不對，我應該把它擺在獨立的迴圈內去執行，即上面被comment掉的第2到4行，否則如果在第6行執行，在第5行的迴圈當中，假設在iteration i=x, j=y時，發生了互換，next\_state[x]=state[y]且next\_state[y]=state[x]，但因為沒有break掉迴圈，所以下一個iteration會繼續被執行，此時下一個iteration的next\_state[i]=state[i]這一行可能會讓剛剛交換完state的色塊又被assign成default的情況，導致可能其中一個色塊在交換完state後又被default回原本的state，造成出現A, A或B, B的狀況。

1. **Suggestions**

希望老師在上課時可以更頻繁講笑話，或許對出席率會有幫助。