**Final project**

**提供遠端操作功能之多功智能洗烘衣機**

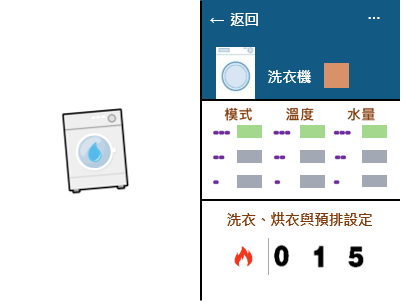
**110060027朱豐蔚、109061116 徐瑞澤**

**壹、專題內容簡介**

在現代忙碌的生活中，洗烘衣機兩用合一的特性使其已然成為使用者不可或缺的3C家電，再者，配合如今的互聯網發展，越來越多的洗烘衣機都有搭配能夠與手機APP連結的功能，使消費者不再需要點擊洗烘衣機上的按鈕來操作，可以透過APP上的按鍵，實現遠端操作，甚至是進一步搭載預先排程(以下簡稱為預排)功能，能夠事先安排洗衣、烘衣的時程，再預定的時間開始進行洗烘衣，以解決現代人因忙碌而來不及回家處理家事的問題。

本期末專題將設計與實現提供遠端操作功能之多功智能洗烘衣機。在專題中，我們以FPGA本身看作洗衣機，FPGA上的按鈕、開關即代表實際上在洗衣機上的諸多按鍵，而相對的，我們以鍵盤上的按鍵模擬手機APP的操作。七段顯示器用來進行倒數時間、洗衣時間、水量設定、烘衣時間、溫度設定、排程時間的交替顯示，LED燈則代表洗衣機各個細節的狀態。

透過VGA輸出(如下圖)，螢幕的左半部分將顯示目前洗衣機的狀態(等待、洗衣中、烘衣中、暫停、開門)，而右半則模擬手機APP介面，呈現現在洗衣機的狀態與相關設定，右半部的呈現為上述LED與SSD的綜合呈現。

本洗烘衣機提供基本的洗衣、烘衣、暫停等功能，加入多樣化的水量、溫度、時間設定，而預排功能則是此次專題的重點，讓洗衣機能在預定的時程進行洗烘衣的程序。

圖一、VGA呈現畫面示意圖

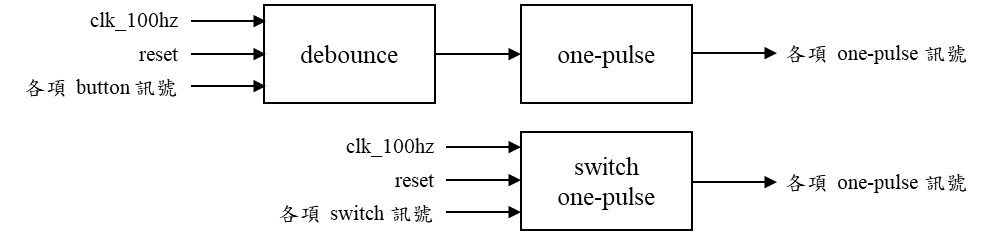
**貳、I/O pins assignment**

|  |  |  |
| --- | --- | --- |
| **Input / Output** | **Pin** | **Function** |
| Button Up | T18 | 上下調整控制項目 |
| Button Down | U17 |
| Button Left | W19 | 調整溫度大小、水量大小、時間多寡 |
| Button Right | T17 |
| Button Mid | U18 | 啟動、暫停、長按取消 |
| DIP switch 1 | V17 | 電源開關(等同於reset) |
| DIP switch 2 | W16 | 模擬洗烘衣機門的開啟與關閉 |
| DIP switch 3 | R3 | 預選方案(細緻洗衣):  洗衣30分、少水、烘衣30分、低溫 |
| DIP switch 4 | W2 | 預選方案(快速洗衣):  洗衣15分、中水、烘衣15分、中溫 |
| DIP switch 5 | U1 | 預選方案(厚重洗衣):  洗衣60分、多水、烘衣60分、高溫 |
| Keypad Up | 同Lab 8 | (模擬手機APP) 上下調整控制項目 |
| Keypad Down |
| Keypad Left | (模擬手機APP)  調整溫度大小、水量大小、時間多寡 |
| Keypad Right |
| Keypad Enter | 啟動、暫停、長按取消 |
| LED light D1 | U16 | 若為洗衣中，燈亮 |
| LED light D2 | E19 | 若為烘衣中，燈亮 |
| LED light D3 | V19 | 若為少水，燈亮 |
| LED light D4 | W18 | 若為中水，燈亮 |
| LED light D5 | U15 | 若為多水，燈亮 |
| LED light D6 | V14 | 若為低溫，燈亮 |
| LED light D7 | V13 | 若為中溫，燈亮 |
| LED light D8 | V3 | 若為高溫，燈亮 |
| LED light D9 | U3 | 若為細緻洗衣，燈亮 |
| LED light D10 | P3 | 若為快速洗衣，燈亮 |
| LED light D11 | N3 | 若為厚重洗衣，燈亮 |
| 7-segment digit 0 | U2 | 倒數時間、洗衣時間、水量設定、烘衣時間、溫度設定、排程時間的交替顯示 |
| 7-segment digit 1 | U4 |
| 7-segment digit 2 | V4 |
| 7-segment digit 3 | W4 |
| LCD Display | 同Lab 9 | 顯示洗衣機目前狀態與遠端控制手機APP介面 |
| Speaker | 同Lab 7 | 洗烘衣完成後的鈴聲 |

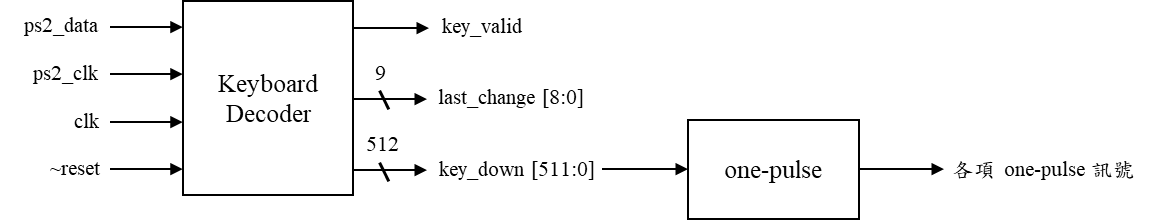
**參、各項前處理**

一、所需除頻器的統整：

在這部分，先將之後所有需要的頻率都先生成出，包括輸出聲音所需要的audio\_lrck、audio\_mclk、audio\_sck，還有更新後續FSM所需的clk\_100h、更新VGA的clk\_25Mhz、倒數計時用的clk\_1hz，最後還有更新SSD顯示digit的refresh。

二、按鈕、開關的one-pulse統整：

在這部分中，將各項輸入的訊號進行one-pulse處理，以方便進行後續各種FSM的判定，此外，switch也使用one-pulse的方式生成打開、關閉兩種one-pulse訊號，以進行開關洗烘衣機門、選用方案等的判定。

三、鍵盤訊號的統整：

在本次專題中，僅用到上下左右以及Enter五個按鍵，故針對此五個按鍵的keydown進行判定，並使用one-pulse進行處理，以方便進行後續各種FSM的判定與button的統整。

**肆、洗烘衣機設定與SSD的顯示切換**

一、設計細節：

在這一個大章節中，將會描述SSD中六種呈現：1.倒數時間、2.洗衣時間、3.水量設定、4.烘衣時間、5.溫度設定、6.排程時間各自的設定細節與切換方式。

在設計中，我們使用鍵盤上下鍵與按扭上下鍵去切換SSD此時要顯示1. ~ 6.的哪一個，我們採用ring counter的設計，即1.與6.之間是互通的。

備註：reset過後預設為顯示1.。

二、水量、溫度的設定FSM：

當在3.狀態下，此時SSD會呈現水量的設定，僅有在此狀態下，左右鍵才會影響水量多寡；當在5.狀態下，此時SSD會呈現溫度的設定，僅有在此狀態下，左右鍵才會影響溫度高低。

水量與溫度是使用相同的FSM去做控制，各自都分為低、中、高三種狀態，彼此之間的切換由鍵盤左右鍵與按紐左右鍵的one-pulse來做切換。

備註：低、中、高分別以中橫線呈現於digit\_0、digit\_1、digit\_2。

三、洗衣時間、烘衣時間的設定counter：

當在2.狀態下，此時SSD會呈現洗衣時間的設定，僅有在此狀態下，左右鍵才會影響洗衣時間；當在4.狀態下，此時SSD會呈現烘衣時間的設定，僅有在此狀態下，左右鍵才會影響烘衣時間。

洗衣時間與烘衣時間是使用相同的counter去做控制，而counter又進一步分成處理小時與分鐘的，大體上寫法與Lab\_6類似，每當按鍵盤右鍵或按紐右鍵時，分鐘的counter皆會 +5，而反之，左鍵則是 -5，分鐘的進位與退位會分別讓小時的counter +1與 -1。

會採用一次調整5分鐘的counter，是為了更加貼合現實中的設定，通常洗烘衣時間的微調都是以5分鐘為單位去進行。

備註：此兩個counter不會隨著時間秒數過去而減少，因為在此刻，這兩個counter代表的是之後進入洗烘衣狀態後各自所需要跑的時間，僅是設定，不是之後實際倒數的ocunter。

四、預排時間的設定counter：

當在6.狀態下，此時SSD會呈現預排時間的設定，僅有在此狀態下，左右鍵才會影響預排時間。

預排時間是使用counter去做控制，而counter又進一步分成處理小時、分鐘與秒的，大體上寫法與Lab\_6類似，每當按鍵盤右鍵或按紐右鍵時，分鐘的counter皆會 +1，而反之，左鍵則是 -1，分鐘的進位與退位會分別讓小時的counter +1與 -1。

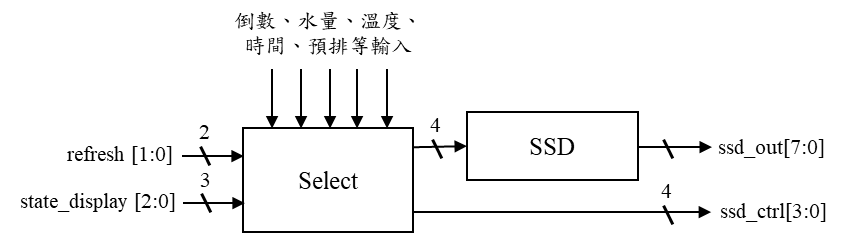
而與洗烘衣設定時間的counter不同，考慮到現實中使用時，預排大多是以小時為大單位調整，所以加入了長按功能，當長按右鍵會一次增加一小時，長按左鍵會一次減少一小時，最大提供了9小時59分的預排時間。

備註：與洗烘衣設定時間的counter不同，當counter不為0時0分0秒時(reset完會為0時0分0秒)，此counter會隨著時間秒數過去而進行倒數，減少到0即會開始進行洗烘衣。換句話說，當在6.狀態下，一點擊右鍵增加任何分鐘、小時數字，都會開始進行倒數。

五、倒數時間的顯示：

當在1.狀態下，此時SSD會呈現目前進行洗烘衣狀態時間的倒數(與現實中洗烘衣同，顯示小時與分鐘數)，在此狀態下，左右鍵並不會影響任何設定。有關於倒數的細節與值將會於下一大章節中進行討論。

備註：在洗烘衣機並未進行任何洗烘衣時，倒數都會顯示為0時0分。

六、SSD控制：

同設計細節中所述，select將會根據state\_display [2:0]來控制最終輸出的四個digits為何，並使用在前述所輸出的refresh來做為SSD的更新訊號，最終將結果輸出於SSD上，以模擬洗衣機上各項燈號、面板的輸出情況。

**伍、進程FSM與倒數設計**

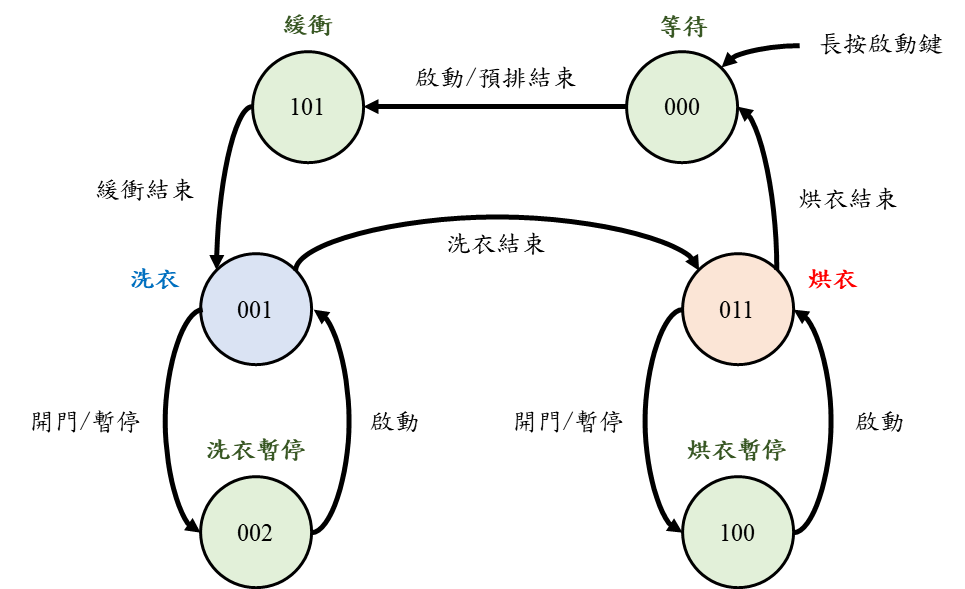
一、進程FSM：

這一部分的設計目的也就是整台洗衣機最核心的部分：進程，負責通過接受各項指令，例如啟動、倒數結束，來控制洗衣機的狀態切換，進一步影響LED燈號、SSD與VGA輸出的呈現。

首先先針對我們所需的state畫出簡易的state diagram，如下頁圖二。如圖，如同正常洗烘衣機，我們需要6種state來分別描述等待階段、緩衝階段、洗衣階段、洗衣暫停階段、烘衣階段和烘衣暫停階段。

在等待階段中，水量、溫度的LED將不會更新，表示階段顯示的LED將不會亮，SSD中的倒數將呈現0時0分，VGA左半部分將根據開門與否呈現開門、關門兩種圖案。在此階段中，將會持續等到使用者按下啟動，或是預排時間倒數結束才會切換到緩衝階段，正式開始洗烘衣。

備註：reset後將會從等待階段開始，在其他任意階段透過長按啟動鍵也能回到等待階段。當開門狀態時，是沒辦法啟動洗烘衣的。

圖二、進程FSM之state diagram簡易圖

進入緩衝階段後，水量、溫度的LED將更新為啟動洗烘衣時當下所設定的水量和溫度，也會根據設定載入洗衣時間與烘衣時間供後續倒數使用，表示階段顯示的LED此時仍不會亮，SSD中的倒數將還是呈現0時0分，VGA左半部分呈現關門圖案。在此階段中，將會空倒數2秒才正式進入衣階段，這樣的設計是為了與現實中的洗衣機更加相似，能夠給予使用者緩衝時間去取消啟動的動作，去預防誤觸等問題。

從進入洗衣階段後，水量、溫度的LED將不會再受外界設定所影響，換句話說，不能因為預排的設定而讓正在洗烘衣的設定產生變化，表示洗衣階段顯示的LED燈會亮，VGA左半部分會呈現洗衣圖案(透過clk\_1hz，我們讓兩張圖片來回切換，進而產生動畫的效果)，且SSD中的倒數將呈現洗衣時間的倒數，倒數結束後會進入烘衣階段。

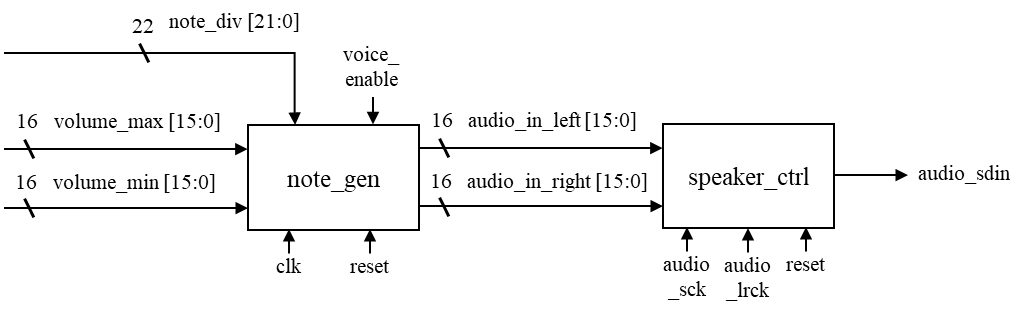
備註：當在洗衣階段下，按下暫停(即啟動鍵)或是開門，皆會使洗衣暫停，此時暫停倒數，必須待到將門關上後，再重新點擊啟動鍵才可回到洗衣階段。若將洗衣時間設定為0時0分，與緩衝時間相同，我們仍會讓洗衣狀態空跑2秒才進烘衣階段，以提供給使用者時間來做取消的動作。

烘衣階段、烘衣等待階段與上述洗衣皆雷同，因此不再重複描述。

二、倒數counter的設計：

從上述FSM的設計，可以看出倒數counter在本次專案的重要性以及廣泛性，為求方便與統一，我們統一都是使用一個counter (包含時、分、秒的子counter)來進行緩衝、洗衣、烘衣時間的倒數。

由於採用同一個counter，我們使用load counter的設計，並附上load訊號，讓counter能夠根據不同state下來load所設定之倒數時間。至於完成倒數的判斷，我們採用one-pulse的方式，當時分秒皆歸0時，downtime\_complete將輸出1，其餘皆輸出0，而透過switch one-pulse，就可以將downtime\_complete的one-pulse訊號取出，供進程FSM使用。

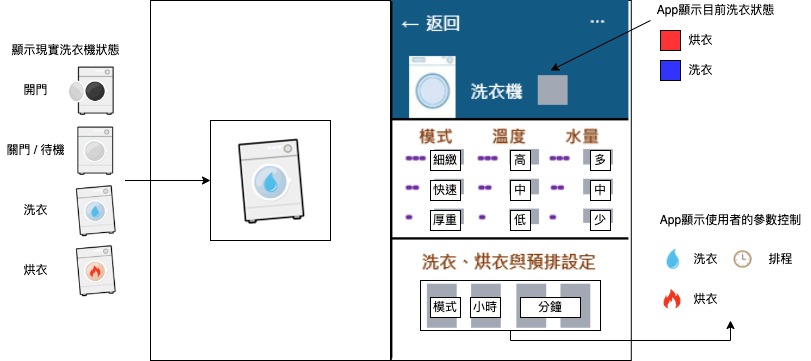
**陸、聲音輸出控制**

在本次專案中，為了還原洗衣機在結束洗衣、烘衣會有的音效，我們使用了speaker來做聲音的輸出。

由於不須調控音量，於是我們直接在assign volume­\_max (16'b0011\_1111\_1111\_1111)、volume­\_min (16'b1100\_0000\_0000\_0000)的值。而至於音高，我們將洗衣完成後的聲音定為Mi (note\_div = 22'd151515)、烘衣完成的聲音定為Do (note\_div = 22'd191571)。

而控制要不要發出聲音的訊號為voice\_enable，當洗衣、烘衣各自結束後的一個clk\_1hz，輸出為voice\_enable，即可達成洗衣、烘衣結束時的音效播放。

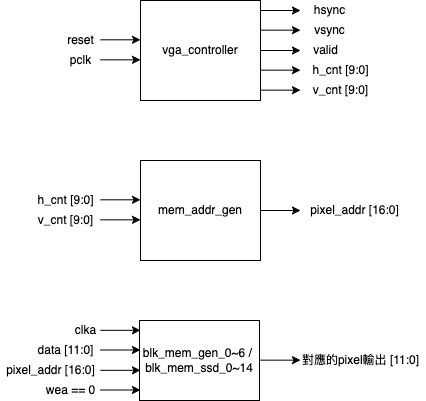
**柒、VGA輸出控制**

圖案對照功能之示意圖如下方所示：

圖三、VGA螢幕輸出圖案對照功能圖

而以下部分為VGA控制的整體架構說明(部分codes位於main.v中) ：

1. VGA\_controller：



此Module為負責VGA控制訊號的輸出(vsync, hsync)，以及輸出掃描點目前的位置(v\_cnt, h\_cnt)。

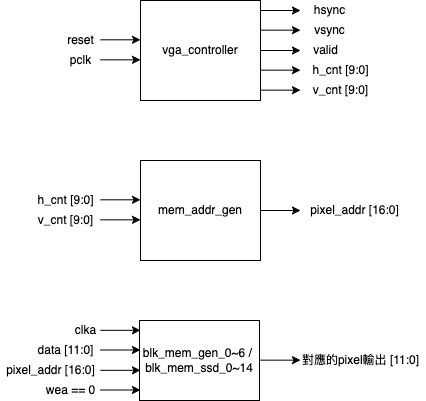
我們透過兩個counter產生螢幕掃描的水平(hsync)與垂直(vsync)訊號。而hsync 和vsync的輸出值預設為1，只有當counter在sync pulse的範圍(範圍如下頁表一所示)中，vsync和hsync的值才會 = 0。

一張含有 桌 的圖片

自動產生的描述表一、VGA規範表

h\_cnt和v\_cnt的功能則是：當counter的值位於visible area內(0 ~ 639 / 0 ~ 479)時，兩者才會輸出相對應的行列數。另外，valid則是用來確認目前的掃描位置是否位於visible area內，只有當掃描點在裡面時，數值才 = 1。

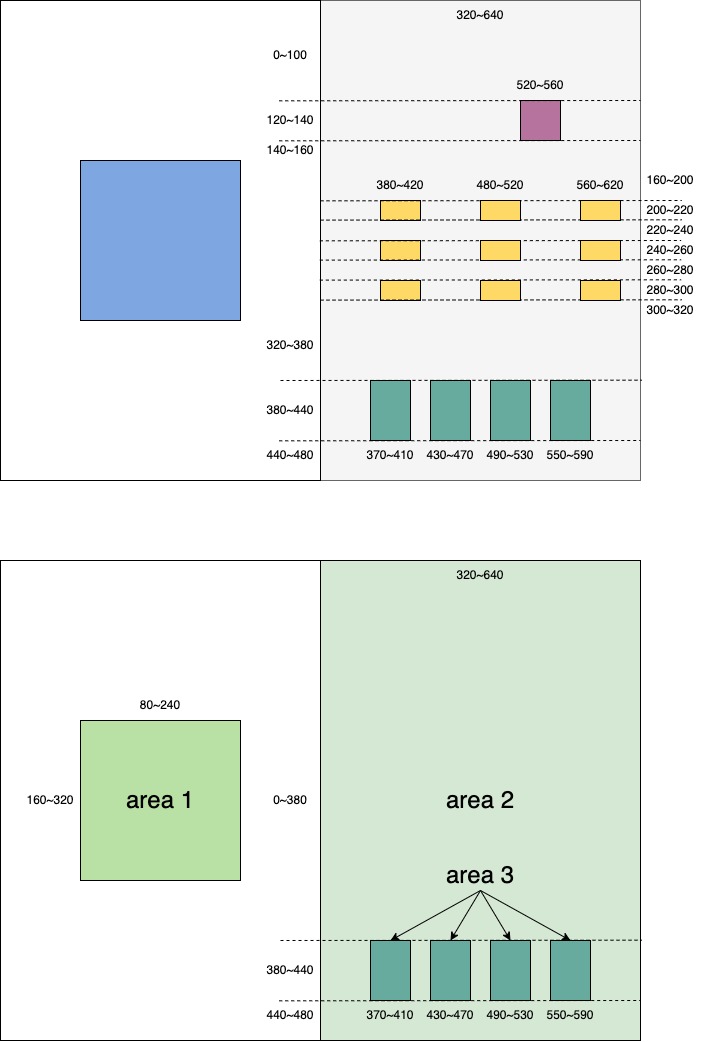
1. mem\_addr\_gen：



此module用來定位照片輸出時，掃瞄照片像素點的位置。

一張含有 文字 的圖片

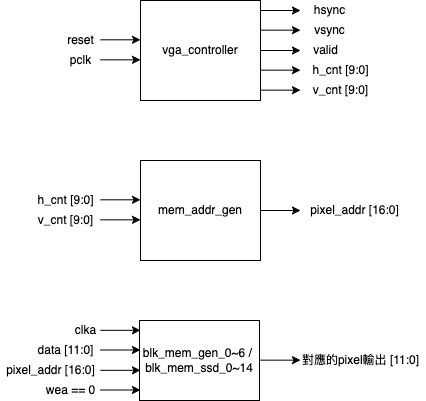
自動產生的描述輸出值pixel\_addr (即指定掃描相片像素位置)的代碼和說明如下：

此外，在此module中，我們透過if-else將整個畫面切割成諸多子畫面，以易於debug跟快速修改參數。切割的範圍如下頁圖四所示：

圖四、畫面分割示意圖ㄧ

其中，area 1用於顯示洗衣機動畫、area 2用於顯示洗衣機App的背景、area 3則用來顯示使用者想設定的參數(水量、溫度、時間等)。實際上，因為繪圖輸出時造成的微小誤差，所以實際範圍會略為加減幾個像素點的位置。

三、block memories



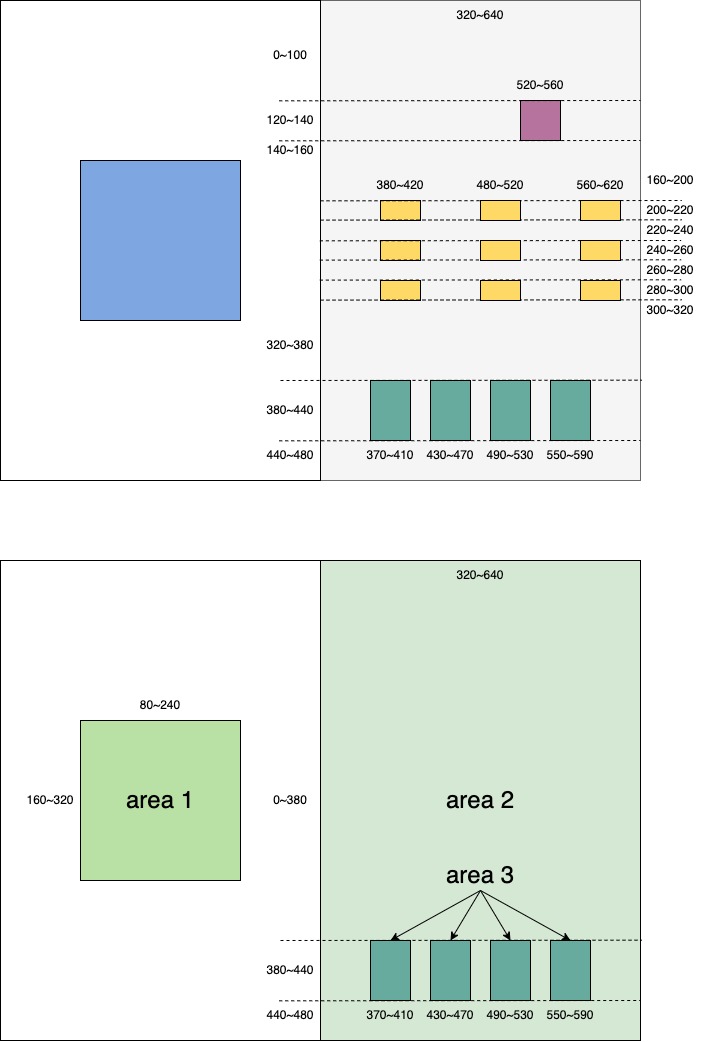
我們使用blk\_mem\_gen\_0~6 / blk\_mem\_ssd\_0~14，儲存我們整個final project所用到的相片檔，並透過pixel\_addr呼叫圖片像素對應的位置，將顏色的輸出值(Ex：pixel\_ssd\_0, pixel\_back等)回傳給我們。

四、顏色輸出（整合在main.v中）：

一張含有 文字 的圖片

自動產生的描述我們透過以下代碼來實現顯示指定顏色的功能：

跟我們在mem\_addr\_gen用到的概念類似，我們先將畫面透過if-else切割成諸多小部分(如圖四) (不同色塊代表不同的area)：

圖五、畫面分割示意圖二

其中，藍色區域為現實生活中洗衣機的狀態、紫色區域為App中顯示洗衣機「目前狀態」(洗衣或烘衣)的位置、黃色區域為顯示洗衣機「目前正在運作」的模式、溫度、水量、綠色區域則顯示使用者在app上「預排」的數值。另外，白色區域單純設定背景為白色、灰色區域則是匯入我們App背景的顏色數值。實際上，因為繪圖輸出時造成的微小誤差，所以實際範圍會略為加減幾個像素點的位置。

此外，因為顯示的圖片、顏色需要跟其他module內的狀態相對應，所以在if-else中，我們另外使用case來指定：當接收到的數值為何時，App或洗衣機的狀態該顯示哪張照片的色塊。除了用來指定該顯示哪個色塊外，case還可以幫助我們實現動畫的效果（透過1hz clock控制），讓洗衣機的狀態可以更活潑生動。

**捌、過程中遇到的困難**

一開始我們構想是將所有照片都以320\*240製作，這樣在寫code的時候就不需要一直對掃描位置跟圖片位置之間的關係。但當我們把好幾張320\*240的照片匯入RAM時，在synthesis就會直接報錯。查了一下Error message後，發現可能是我們memory總共需要的Depth太多，所以Vivado直接不給跑Synthesis。後來我們只好去測試FPGA版能接受的總Depth最大為多少（大概為13、14萬內），並將圖片的部分區改用色塊顯示，減少需要用到的memory depth。

另外，在編寫VGA的顏色輸出時，有一陣子顯示出來的顏色都很奇怪，最後才發現是自己少寫一個else跳出去。由於寫VGA需要用到大量的if-else，因此也是花不少時間一個個將code給註解掉，才發現這個問題。

在最終的Debug階段，我們遇到了由於意外的數值插隊進緩衝時間的倒數，進而產生的無法切換state問題，我們很快就能料想到是在倒數的部分出了差錯，但卻在找實際錯在哪個細節上找了許久，最後才發現是在if-else迴圈中少寫了最終的else。同上，if-else、default等問題常是我們專題出問題的地方，而這樣漏寫else的壞習慣，若不是這次專題的撰寫，我們恐怕不會有那麼深刻的體悟，也感謝這次專題的機會，讓我們上到了寶貴的一課。

**玖、心得與感想**

朱豐蔚：

在製作此次專題的過程中，不像大多數人選擇製作遊戲，我們選擇了能夠完整呈現我們這學期所學的題目，結合鍵盤、speaker、VGA去製作一台功能完整的洗烘衣機。我認為完整性會是我們組最大的優勢，不管是從一開始的發想，到實際對現有洗衣機的詳細考察，還是到最終的Debug，我們幾乎考量到所有問題與使用者體驗上的不足，此點也可以從上述我們對專題的細節說明了解到。然而，由於我是負責洗烘衣機的主架構，在重複製作、燒板的過程中，在還沒有加入VGA之前，其實很難體會到真的做出一台洗烘衣機的真實感。很感謝我的組員在VGA上的製作真的花了很大的心力，能夠在我們最後合併專題時，在一看到洗衣機正常運作、並呈現動畫的剎那，能夠感受到滿滿的成就感。也再次謝謝教授與各位助教這一學期的辛苦指導，很高興能修這門課。

徐瑞澤：

這次我主要負責VGA的部分，因為VGA是全新的部分，個人認為雖然比起音訊的Lab，少了需要一直對clock的部分，但因為這次是牽涉到二維平面上，掃描點的運動，因此理解起來比音訊的部分還要花時間。不過看著什麼都顯示不出來、顏色錯誤、像素點跑掉，到能夠完整運行洗衣機的功能，心裡的成就感還是多了不少。另外也很感謝我的隊友，讓我能夠沒有後顧之憂的專心調整VGA，不然光要調整那個像素點真的很燒眼睛。