

Final Project Report

Team04 : B02901093 吳岳, B0290197 張哲銘, B02901098 宇德原

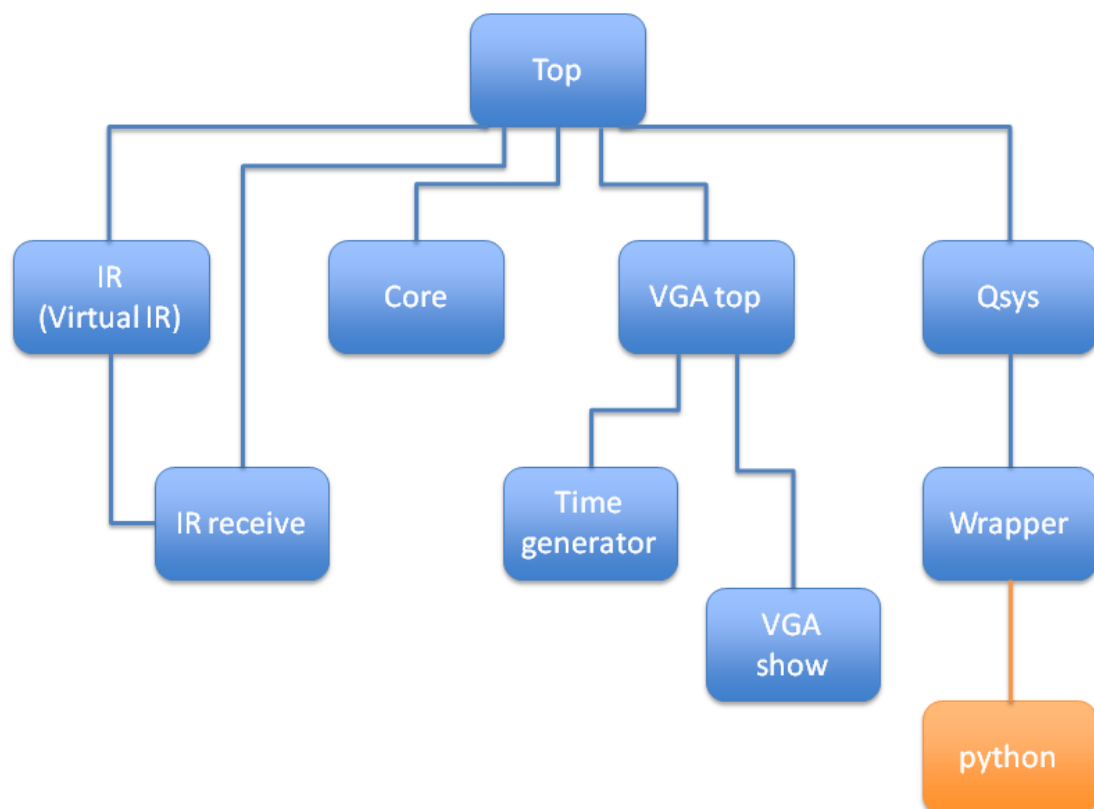
Introduction

我們 Final Project 的主題是利用 FPGA 以及其周邊的 I/O 介面來實現「視力檢測系統」。這個產品的功能，是讓使用者用紅外線遙控器，並照著 VGA 上面顯示的資訊，來自我檢測視力以及有無散光或色盲。除此之外，檢測的結果會和使用者用紅外線遙控器在一開始設定的 ID，一起利用 RS232 傳送到電腦儲存起來。而電腦端會將使用者的 ID、視力，以及散光或色盲與否的結果，搭配當時測試的日期與時間一併建檔。這樣的功能不僅能讓單一使用者在日後了解先前的測量狀況，功能深化後也可套用在眼科量測中心，便於管理和追蹤不同客戶的視力資料。

Equipment

1. DE2-115 FPGA 板
2. VGA 顯示器
3. 電腦
4. RS232 線一條
5. Altera 紅外線遙控器

System Structure

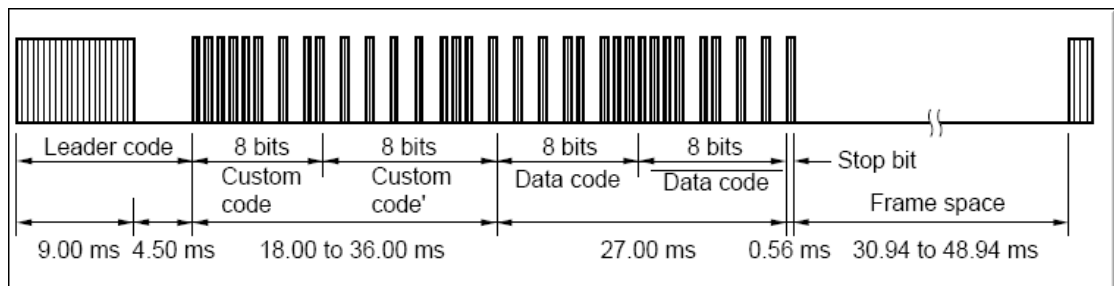


Introduction of different modules

1. IR Receive and Virtual IR

這兩個模組的工作主要是接收紅外線遙控器傳來的訊號、判讀使用者所按的按鍵，再將其轉換為適當的訊號來傳給 Core、VGA、Wrapper 這些模組。

首先，IR Receive 這個模組的作用是接收紅外線訊號並判讀使用者所按的按鍵。由於 DE2-115 開發版所附的 IR 遙控系統使用的是 NEC 協定，所以這裡先簡單講解一下 NEC 協定的內容。以下便是 NEC 協定的 IR 發射訊號示意圖：



由上圖我們可以看到，IR 訊號的前端會先輸出一個 9ms 的載波和 4.5ms 的 0 的前置碼，表示遙控器輸出訊息的開始；末端則會有一段結束碼，表示輸出訊號的結束。在這兩個訊號之間，會傳輸 16-bit 的客戶碼和 8-bit 的按鍵碼、8-bit 的反向按鍵碼。然而真正能用來判讀使用者所按按鍵的只有其中那 8-bit 的按鍵碼。

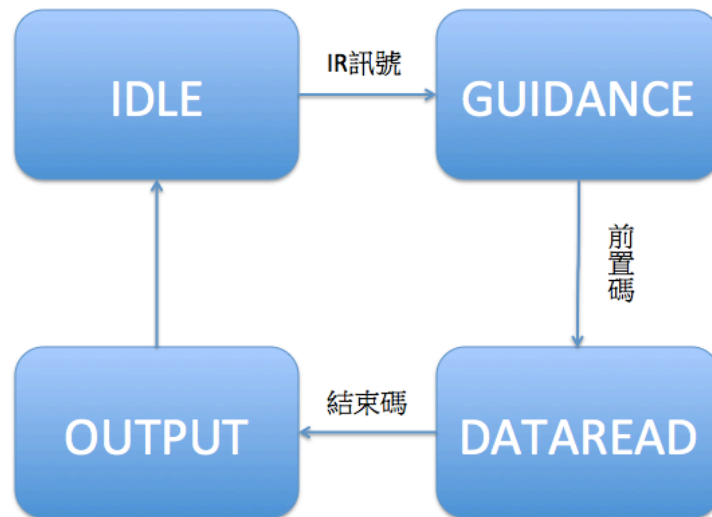
解讀按鍵碼的“1”和“0”的方法，是看每個載波後緊接著的 0 的時間。若是“1”的訊號，是由 560us 的載波和 1680us 的 0

所組成;若是“0”的訊號，則是由 560us 的載波和 560us 的 0 所組成。

如此，解讀出按鍵碼的“1”與“0”的訊號後，我們可以參考按鍵與鍵碼的對應關係表：

遙控按鍵	0	1	2	3	4	5	6
按鍵碼 (8-bit)	00	01	02	03	04	05	06
遙控按鍵	7	8	9	A	B	Power	
按鍵碼 (8-bit)	07	08	09	0F	13	12	

對此，我們的 state machine 會先用三個 state 來接收並判讀此訊號所代表的按鍵，再用一個 state 來輸出給 Virtual IR 做進一步的處理。這部分的 state machine 如下圖所示：



如圖，在 IDLE 這個 state，一收到 IR 訊號就會跳到 GUIDANCE; 在 GUIDANCE 這個 state，若偵測到正確的前置碼後，就會跳到 DATAREAD; 在 DATAREAD 這個 state，會將依序輸入的客戶碼、按鍵碼及反向按鍵碼儲存至暫存器，並藉由按鍵碼和反向按鍵碼判斷輸入訊號是否正確，且當偵測到結束碼時就會跳到 OUTPUT 這個 state; 在 OUTPUT 這個 state，會利用上述所提到的表來判讀輸入的訊號代表哪個按鍵，最後再將此按鍵的訊號輸出給 Virtual IR 這個模組。

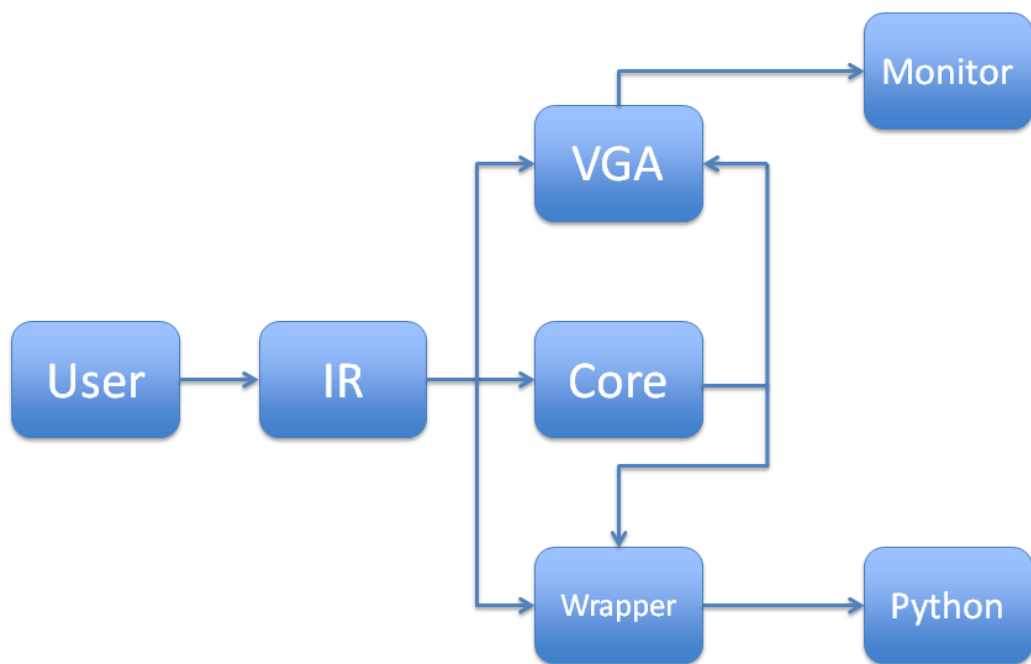
再來要講解的是 Virtual IR 的工作。Virtual IR 從 IR Receive 得到的 input 即代表著使用者按了什麼按鍵。以我們的功能而言，Virtual IR 可能會收到的 input 就只會有 0~9、A、B 和 Power 這幾個按鍵。0~9 這幾個按鍵會被用作輸入的 ID 數字，而其中的 2、8、4、6 又會被用來當作檢測視力時的上、下、左、右之分。

A 和 B 兩個按鍵會被用來當作 “YES” 、 “NO” 之分。而 Power 則是用作 reset 的功能。

為了要將這些按鍵的訊號轉換成適當的訊號再傳給其他模組，我們還必須把 Virtual IR 的 state machine 分為兩個部分。第一個部分是等待 ID 的四個數字輸入，且每接收到一個數字便會用回授的方式維持其值，以利 VGA 能持續地在螢幕上輸出 ID 數字。直到四個數字都輸入完後，再接收到 “YES” 的訊號，才會跳往第二個部分。而第二個部分是單純地將從 IR Receive 接收到的訊號，轉換成上、下、左、又，或 “YES” 、 “NO” 等訊號，再輸出給 Core 做判斷的處理。

2. Core

這個模組的工作是控制整個系統在不同情況下該做的事情，並且給予其他模組所需要的 input 值，在我們設計的系統下，Core 必須和 Virtual IR, VGA, Wrapper 來做互動，下面就分別詳述互動的方式及過程，還有系統不同 module 之間的互動圖。



首先，Core 必須從 Virtual IR 那邊獲取三個 input，第一個是代表使用者透過紅外線遙控器上面的數字來回答他所看到的方向，第二個是指示使用者是否有按下遙控器上 Power 的鈕，這個按鍵是讓使用者重置自己目前量測視力的結果，並且重新量測，此外，在我們系統設計的三個測驗都完成後，讓使用者確認結束，並且重新回到初始 ID 畫面的按鍵。最後一個 input，用來表示使用者在回答散光和色盲測驗對與否的按鍵，比如說使用者看到我們的散光圖後，覺得線條粗細並不一致，那麼他就可以按下紅外線遙控器上面的 B 來回答 “ 否 ”，反之亦然，因此這個 input 的功用就是把這個結果通知 Core，另外，它同時也是用來讓使用者確認自己的 ID 輸入完畢的按鍵。

接著，Core 在知道使用者所按下的按鍵後，換句話說就是使用者針對不同測驗的回答後，必須判別它的正確性，然後做出適當的反應。由於使用者所能看到的是 VGA 上面資訊和圖像，因此 Core 必須給予 VGA module 適當的 input 來呈現目前的狀態，而這些 input 包括：視力檢測的圖形要顯示的方向，視力檢測圖形的大小，Core 目前的 state，散光測驗的結果，色盲測驗的結果，輸出視力值的確認訊號。而這些訊號的給定方式就是透過 Core 內部 state 的變換來達成，以我們自己設計的系統來說，在使用者輸入 ID 後，一開始會要使用者測驗他的視力，而對於同一個視力的範圍，使用者必須連續答對 Core 隨機跑出的兩個缺口方向，才算擁有這個範圍下的視力水準，我們才會讓他跳到下一個範圍的視力，例如說使用者在 0.9 連續答對兩個後，接下來 Core 就會指示 VGA 輸出 1.0 下的視力測驗圖形，如果使用者在這個視力水準下無法連續答對兩個缺口的方向，那麼最後他的視力測驗結果就會是 0.9。

完成視力測驗後，Core 會指示 VGA 輸出視力測驗的結果讓使用者知道，接下來使用者只要按下遙控器上的 A 進行確認後，在 VGA 上面就會顯示散光的測驗圖形，使用者同樣只要按下 A 或 B 的鍵來進行回答後，Core 就會再度指示 VGA 輸出使用者是否

有散光，最後的色盲測驗也是一樣，而當所有的測驗都完成後，Core 會將所有的資料，包括：使用者的視力水準，散光測驗結果，色盲測驗結果傳給 Wrapper 這個模組，並且送出一個 bit 的 start_to_send 訊號讓 Wrapper 知道它可以開始把使用者的測驗資料傳送到電腦做建檔，最後，我們設計中比較特別的是，User ID 的資料並不是透過 Core 這個模組來傳送給 Wrapper，而是由 Virtual IR 的模組直接紀錄使用者輸入的 ID 後把資料傳送給 Wrapper 暫存，然後等到使用者把所有的測驗都跑完後，由 Core 統一將剩餘的資料丟給 Wrapper 並且指示開始傳送資料到電腦。

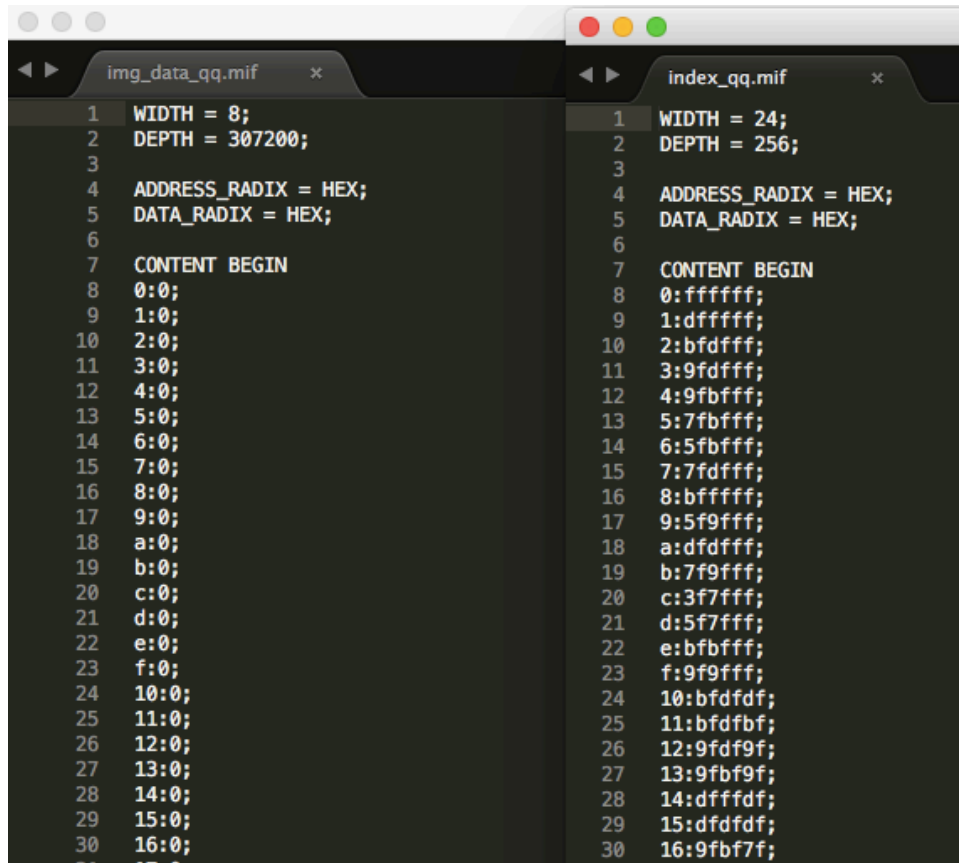
3. VGA

我們使用的是 640*480 Pixel 的解析度，以 60HZ 的頻率更新螢幕，每秒提供 25MHz 的 Clock 給 VGA，是做部分有參考 System CD 裡面音樂合成器(Synthesizer)跟 DE2_115 這兩個範例，分別處理底圖(色盲圖)的輸出跟塑刻形狀較為規律的視力表(E-Chart)和散光圖。

3.1 底圖(色盲圖)讀取

我們發現底圖的資訊是由 mif 檔輸出的，分別是 index.mif 跟 data.mif。index.mif 負責將 256 色的 RGB code 對應到 1~256

的 index，而 data 則設定在不同的座標(共 $480 \times 640 = 307200$ 個座標點)下的 index, 下圖是 mif 檔的範例。



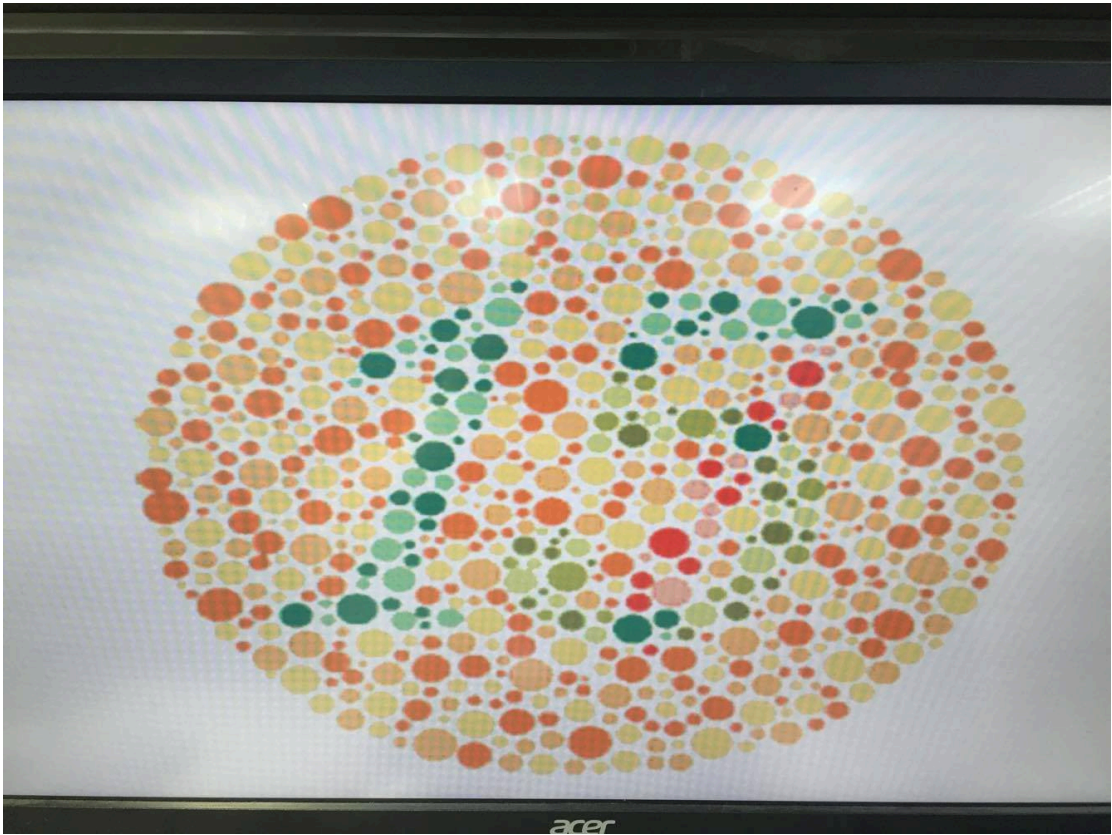
```
1 WIDTH = 8;
2 DEPTH = 307200;
3
4 ADDRESS_RADIX = HEX;
5 DATA_RADIX = HEX;
6
7 CONTENT BEGIN
8 0:0;
9 1:0;
10 2:0;
11 3:0;
12 4:0;
13 5:0;
14 6:0;
15 7:0;
16 8:0;
17 9:0;
18 a:0;
19 b:0;
20 c:0;
21 d:0;
22 e:0;
23 f:0;
24 10:0;
25 11:0;
26 12:0;
27 13:0;
28 14:0;
29 15:0;
30 16:0;
```

```
1 WIDTH = 24;
2 DEPTH = 256;
3
4 ADDRESS_RADIX = HEX;
5 DATA_RADIX = HEX;
6
7 CONTENT BEGIN
8 0:ffffff;
9 1:dfffff;
10 2:bfdfff;
11 3:9dfff;
12 4:9bfff;
13 5:7bfff;
14 6:5bfff;
15 7:7dfff;
16 8:bfffff;
17 9:5f9fff;
18 a:dfdfff;
19 b:7f9fff;
20 c:3f7fff;
21 d:5f7fff;
22 e:bfbfff;
23 f:9f9fff;
24 10:bfdfdf;
25 11:bfdfbf;
26 12:9fd9f;
27 13:9fb9f;
28 14:dfdfdf;
29 15:dfdfdf;
30 16:9fbf7f;
```

但現在有另一個問題，我們要如何將圖檔轉成 mif 檔呢？請教過有經驗的同學之後，我們決定將圖檔先轉成 bmp 點陣圖檔，再將 bmp 轉成 mif 檔。png 或 jpg 檔轉 bmp 線上有轉換器，bmp 轉 mif 有現成的 python code 作轉換。

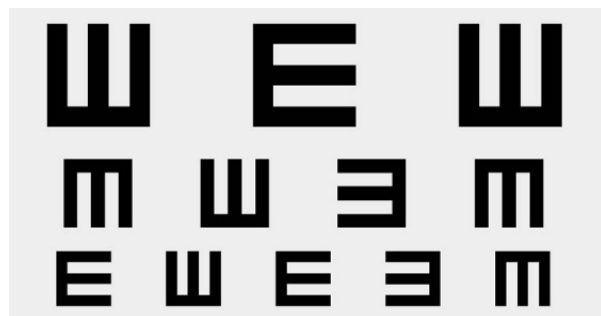
轉換完 mif 檔之後，就可以將讀取 mif 檔的.v 檔放進 project 裡面，配合 qsys 將 mif 的資料存入記憶體裡面，但要額外留意，此規格的 VGA 只能用 25MHz 的 Clk，因此需設定 PLL 將 50MHz 轉為 25 MHz。

以下是我們產生的色盲圖：



3.2 視力表原理

在找過相關資料後，
我們決定用 E-chart 做為
顯示在 VGA 判別度數的
方法，而為了得出精準的



數據，在 E-chart 的長度上需十分考究，標準規格是將視角 5 分
度(5 minutes of arc)的大小定義為「判斷視力 1.0」的 E-chart，
對應 10 倍大是「判斷視力 0.1」的 E-chart，也就是最大的 E-Chart。

有個最大的 E-chart 之後，可以此類推相應度數的 E-chart 大小，舉例來說，若受測者離 E-chart 6m，0.1 的 E-chart 圖換算後為 88.6mm (0.2 的 E-chart 圖換算後為 44.3mm...)。更直觀的換算可以列成以下公式：

$$0.1 \text{ E-Chart 之長度} * 68.75 = \text{受測者跟量測表的距離}$$

最後，我們決定用 6m 的規格做為視力量測標準，下表是視力對應的 Pixel 數：

<i>Vision</i>	Pixel	<i>Vision</i>	Pixel
0.1	120	0.7	17
0.2	60	0.8	15
0.3	40	0.9	13
0.4	30	1.0	12
0.5	24	1.2	10
0.6	20	1.5	8
		2.0	6

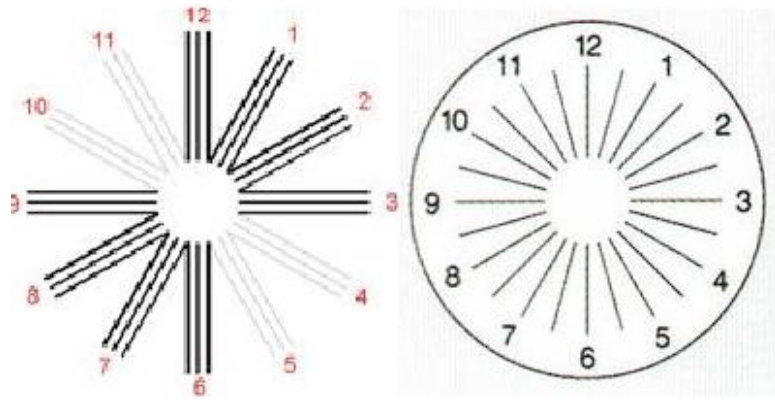
為何會設定 6m？若 pixel 設定的太大，受測者跟量測表的距離將超過紅外線遙控器的使用範圍。若設定太小，在 E-chart 的顯示上就會有較多誤差，以 6m 在視力 0.7 的狀況來論，正確的 E-chart

大小應該是 $120/7 = 17.143$, 但四捨五入之下, 就只能設定為 17 pixel

3-3 散光圖

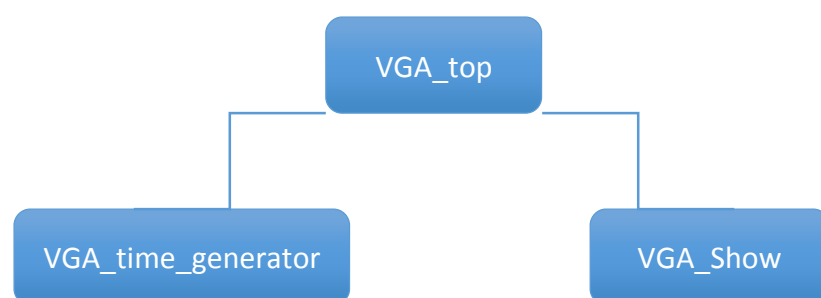
散光量測的詮釋我們用最基本的方法, 以下是兩種常見的圖型,

潛在散光的受試者會認為垂直跟水瓶的線條粗細是不均勻的, 但實



際上任何方向粗細都相同。而考量到塑刻在 VGA 上複雜度, 我們選擇較簡單又不失精準的左圖。

3-4 視力、散光在 VGA 架構上的整合



我們先用 VGA_time_generator 去處理 VGA 頂層輸出的參數, 不同規格的解析度會有不同的設定要預先處理, 以 480×640 pixel 來說, front porch, sync 跟 back porch 的時間就有規定, 未參照的話將無

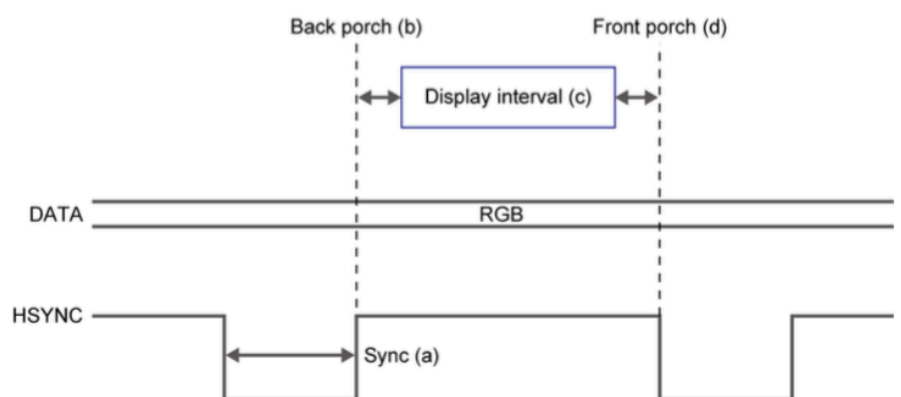
法顯示。另外, 此 Module 也負責 output 出 x, y 座標, 跟 VGA_show 的顯示功能對應。

VGA mode		Horizontal Timing Spec				
Configuration	Resolution(HxV)	a(us)	b(us)	c(us)	d(us)	Pixel clock(MHz)
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25

```

vga_time_generator vga0(
    .pixel_clk(i_clk),
    .h_disp   (640),
    .h_fporch (16),
    .h_sync   (96),
    .h_bporch (48),
    .v_disp   (480),
    .v_fporch (10),
    .v_sync   (2),
    .v_bporch (33),
    .vga_hs   (vga_h_sync),
    .vga_vs   (vga_v_sync),
    .vga_blank(inDisplayArea),
    .CounterY(CounterY),
    .CounterX(CounterX)
);

```



VGA_top 統一整理來自 Core 或是 IR 的 input 資料(ex: E-Chart的大小、E-Chart的方向、State、視力偵測結果、散光結果...) , 傳給 VGA_show 去顯示, 但由於這部分只用到黑白兩色, VGA_show 僅需整合 VGA_time_generator 傳給它的 x, y 座標, 將對應做標下的 1(白色)或 0(黑色)回傳, 再傳至 VGA_top 轉成 RGB 資料

3-5 黑白(視力&散光)跟彩色(色盲底圖)的整合

色盲圖是後來才加的, VGA 控制的方法跟視力及散光的方法不太一樣, 不可用原本的

VGA_time_generator。因此在整合上, 我們用範例的

VGA_controller 稍加修改, 並額

外新增 VGA_transfer 這個 Module, 唯有在需要顯示色盲圖的 state

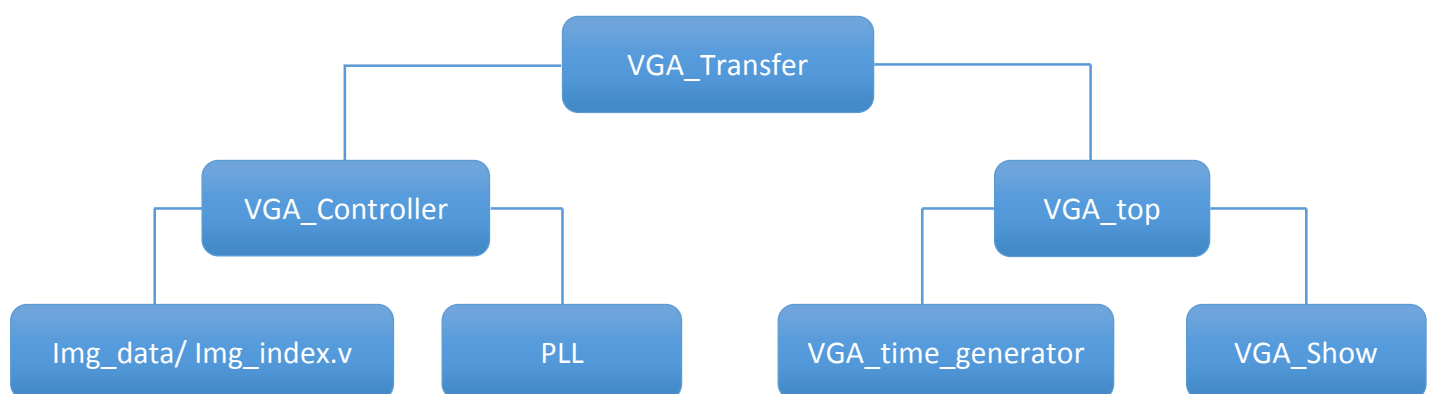
之下, 才從 VGA_controller 那邊擷取資料, 其他時候都從

VGA_transfer 那裏讀值。

```
if(state_to_vga == COLOR_BLIND_STATE) begin
    VGA_B_w = VGA_B_2;
    VGA_G_w = VGA_G_2;
    VGA_R_w = VGA_R_2;
    VGA_HS_w = VGA_HS_2;
    VGA_VS_w = VGA_VS_2;
    VGA_BLANK_N_w = VGA_BLANK_N_2;
    VGA_SYNC_N_w = VGA_SYNC_N_2;
end
else begin
    VGA_B_w = VGA_B_1;
    VGA_G_w = VGA_G_1;
    VGA_R_w = VGA_R_1;
    VGA_HS_w = VGA_HS_1;
    VGA_VS_w = VGA_VS_1;
    VGA_BLANK_N_w = VGA_BLANK_N_1;
    VGA_SYNC_N_w = VGA_SYNC_N_1;
end
```

(for 色盲)

(for 視力、散光)



4. Wrapper

這裡 Wrapper 的操作方法跟 Lab 2 的 Wrapper 差不多, 只是插在透過 Wrapper 額外傳 RS232 那些需要寫檔的資料(視力、散光、色盲檢測結果)。

```
module Rsa256Wrapper(  
    input avm_rst,  
    input avm_clk,  
    output [4:0] avm_address,  
    output avm_read,  
    input [31:0] avm_readdata,  
    output avm_write,  
    output [31:0] avm_writedata,  
    input avm_waitrequest,  
    input [3:0] i_ID1,  
    input [3:0] i_ID2,  
    input [3:0] i_ID3,  
    input [3:0] i_ID4, ←  
    input [3:0] i_size,  
    input i_astigmatism_result,  
    input i_start_to_send,  
    input color_test_result  
);
```

寫檔需要的資料

5. Python program

在電腦端 Python 的程式主要是處理 Wrapper 傳送過來的使用者資料建檔的部分, 由於我們是設定 Wrapper 一次要傳送 256 bit 的資料給電腦端, 而我們真正需要的資料的位元數其實只有包含使用者 ID (16 bit), 視力量測結果(4 bit), 散光結果(1 bit), 色盲結果(1 bit), 總共 22bit, 然而為了讓系統更有擴充性我們設定讓重送的資料從 LSB 算起有 24 bit 的長度可以使用, 所以在這樣的協定下, 假設這傳送的 256 bit 的 LSB 是代表 bit 1 的話, 那 bit 24 ~ bit 17 就是用來表示 User ID 的位置, 而 User ID 總共是 4 個十進位數, 所以我們是以 4 個 bit 來表示一個十進位數, 因此 bit 24 ~ bit 21 就是 User ID 的千位數, 同理 bit 20 ~ bit 17 就是 User ID 的百位數, 而後面的位數抑是遵照此規則安排。接著從 bit 8 ~

bit 5 是用來放使用者的視力結果，因為視力結果共有 13 種可能，所以我們就給定 4 個 bit 來儲存，最後 4 個 bit 就是 bit 4 ~ bit 1，這部分由於只需要安放色盲與否以及散光與否的資訊，所以只需要用到兩個 bit，因此，我們將 bit 3 用來代表色盲與否的資訊，bit 1 則用來代表散光與否的資料，這樣就把所有 Core 從使用者那邊得到的資訊都利用 Wrapper 傳送給電腦了。

接下來的問題是在於，python 的程式透過 Serial Read 的方式從 Wrapper 那邊獲得 256 bit 的資訊後，其實會進一步拆解成 8 個 bit 的單位一次讀取後變成十進位數，然後把這 32 個單位的十進位數資料放在類似 array 的資料型態裡面存放，所以我們要做的事情是先取得這個 array 的最後一筆十進位的資料，而這筆資料就是代表 8 個 bit 的傳送資訊，把它轉成二進位後，並且作一些判斷把轉換後缺少的 0 補滿，就可以還原傳送出來的時候的原始資訊了，接著就只要把這 8 個 bit 的資料拆成前面 4 個 bit 來判讀色盲和散光的結果，後面 4 個 bit 來判斷使用者的視力測量值，就可以把資料寫進指定好的檔案裏面。最後就是要處理 array 裡面的另外兩組十進位數的資訊，由於 User ID 是以 16 bit 來表示，因此會被電腦拆成兩個十進位數的資料，但是我們在這邊可以利用 User ID 在這個系統的設定下剛好是 4 個 bit，而且這兩筆 array

裡的資料事實上總共就是代表 16 個 bit，所以我們只要將 array 裡的兩筆十進位數的結果轉成 16 進位就是代表原本使用者輸入的 ID，經過這樣的處理後，並且補上適當數目的 0，我們就可以正確地還原使用者的 ID 資料，並且搭配前面獲得的資訊一併寫入檔案裏面建檔，這部分就完成的 Python 程式的工作。

Future Improvement

1. 距離量測

雖然使用者操作時可以按照 6m 的規格去量測度數，但還要先確定距離是否剛好 6m，這不太方便。另外，距離的換算也會隨螢幕的尺寸而調整，不可能都是 6m。

其實在 Final Project 發想之初，我們有打算用超音波來檢測使用者跟螢幕的距離，隨之改變 E-Chart 的大小，但考量預算以及系上現有的超音波模組沒有辦法支援 50cm 以上的距離。此構想也只好先擱置，等 7, 8 月準備 Altera 創新競賽複賽時再努力。

2. 視力量測的深度跟廣度

- 深度：

1. 散光圖、結果判斷 Yes/No 的圖片可做得更細緻，或者直接

讀圖。

2. 在色盲圖方面，通常量測色盲不可能只用一張圖就能決斷，更何況色盲也不只有紅綠色盲一種，若能讀入更多的色盲圖，也將增加判斷色盲的準確性，但考量到記憶體의存取空間，同時將 5, 6 張的色盲圖燒到 FPGA 裡面根本不可能，若能善用 RS232 或是 USB 模組，需要用到色盲圖時再將圖片 pixel 的 RGB 值讀入。
3. 視力量測上我們只記錄一種結果，但實際上，視力有分成左右眼或裸視配鏡下 4 種結果，甚至戴上幾度配鏡(ex: 150 度眼鏡視力 0.7, 200 度眼鏡視力 1.0)下的視力。
4. 在 Python Code 的設計上還有很多改進的空間，之後可用 sorting 的演算法讀檔將相同的 User ID 放在一起，並照時間先後、度數排列；程式也可做成全自動的，不需每次讀檔時重新手動執行。

- 廣度

除了度數、散光、色盲之外，還有很多視力判斷的指標，(ex: 左右眼焦距)，這是我們之後可在 Altera 比賽時著手的。

Reference

- <https://eewiki.net/pages/viewpage.action?pageId=15925278> (VGA_controller)
- http://bbs.ednchina.com/BLOG_ARTICLE_184836.HTM
(基於 Verilog 的 VGA 顯示控制)
- https://en.wikipedia.org/wiki/Snellen_chart (Snellen Chart)
- https://en.wikipedia.org/wiki/Visual_acuity (Visual Acuity)
- <http://www.allaboutvision.com/eye-test> (The Eye Chart and 20/20 Vision)
- DE2-115_V.3.0.0_SystemCD - DE2_115_Synthesizer 範例
- DE2-115_V.3.0.0_SystemCD - Default 範例
- 邏輯電路設計 DE2-115 實戰寶典(友晶科技)

ISBN 978-986-88073-0-3