數位電路期末作業・技術手冊

智慧紅綠燈系統

指導老師：簡韶逸老師、吳柏辰助教

製作組別：第六組

B01901015　姚君翰

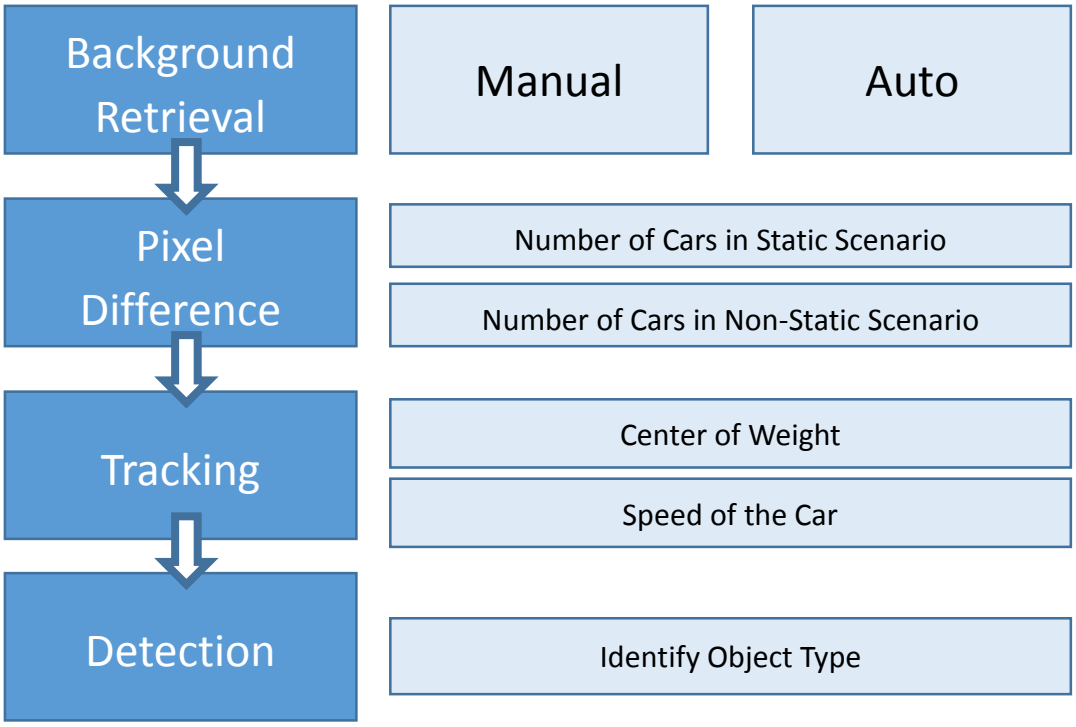
B01901040　劉承曄

B01901171　宋昀蓁

目　　錄

一、基本介紹

本系統目標是利用安裝於紅綠燈上的相機以影像處理方式進行道路車流量偵測，供紅綠燈系統設置各個亮燈模式周期。利用所儲存的背景畫面與目前截取畫面比較，計算不同的像素點，以像素點總數、位置來推得道路上車輛數目、速度。系統運作流程與功能如下圖所示：



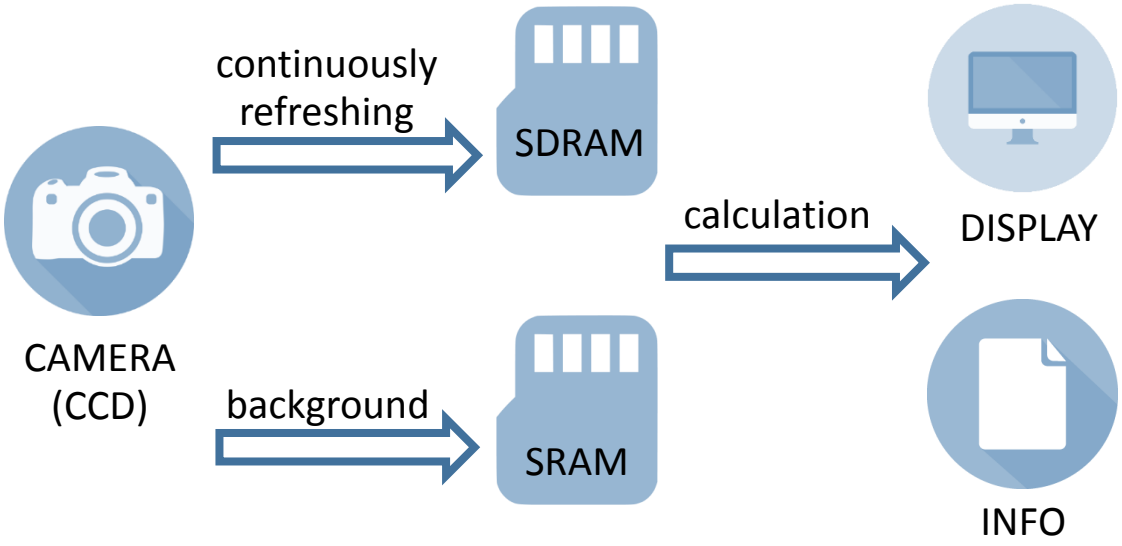
完成初始化後，須先取得背景資訊（Background Retrieval），使用者可以手動或自動的方式儲存背景畫面於靜態記憶體（SRAM）。取得背景資訊後，系統會自動比較背景與目前畫面的差異（Pixel Difference），並根據不同的像素所在位置來更改 diff_flag 標籤。再完成畫素差異計算後，可將畫面中的物件與背景分離，並以此計算物件重心與移動速度，完成物件追蹤（Tacking）功能。追蹤到物件之後，利用汽車方形結構的特性判定物件為車，也就是物件偵測(Detection)功能。

除了初始化以及背景資訊儲存功能需要手動開啟以外，其他幾個功能都是系統自動化完成。而初始化和背景資訊功能僅在一開始安裝時需手動啟動，若實際安裝系統於紅綠燈，僅需在安裝完成後按下初始鈕、背景鈕，而後紅綠燈便可自動運作。相關按鈕設計如下：

按鍵名稱	按鍵位置	功能
一般使用		
重設鈕	KEY[0]	清除可能預先存在記憶體中的雜訊
背景鈕	KEY[1]	自動背景與手動背景模式切換
背景寫入開關	SW[12]	開啟為存取背景，關閉進入車流偵測
移動車輛偵測	SW[17]	開啟移動車輛偵測
除錯模式		
偵測畫面	SW[10]	開啟時顯示物體偵測畫面

背景寫入開關係為手動背景設計，若自動背景功能無法取得滿意的背景，使用者透過手動背景取得更好的背景畫面。而移動車輛偵測在本次版本中需要手動開啟，但若實際安裝於紅綠燈，可將該啟動訊號與紅綠燈的綠燈訊號連接，在綠燈時自動開啟。另外也保留了開啟偵測畫面的功能，方便除錯。

本系統在記憶體與其他裝置的使用上如下圖：

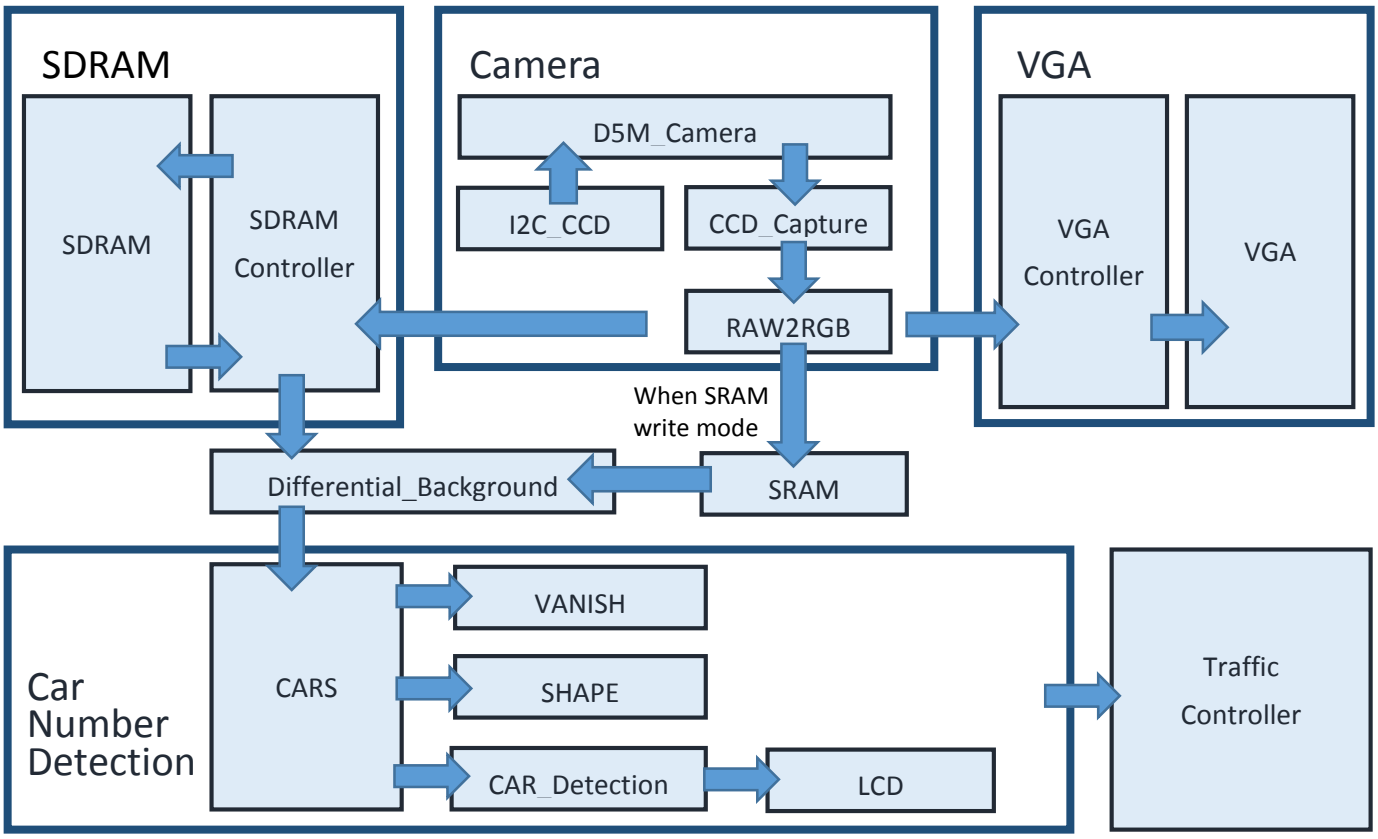


相機取得畫面後，分別儲存於靜態記憶體（SRAM）和同步動態隨機存取記憶體（SDRAM）當中，隨後比較兩者差異並做計算取得交通資訊。而畫面顯示內容則可透過開關調控，一般情況下為目前顯示相機畫面，也就是儲存於 SDRAM 的資訊。若進入偵測模式，也就是開啟除錯開關（SW[10]）畫面僅顯示測得物件。

本次版本的實測是使用側拍的道路影片，在輸入的部分只有時脈訊號（clk）、相機畫面（CCD）與按鈕訊號。若是實際安裝於紅綠燈，可與紅綠燈本身模組的燈號連接，讓本系統隨著紅燈、綠燈開啟不同的偵測模式。甚至可與計時器相連，讓系統在白天與夜晚分別重新儲存一次背景，增加準確度。

二、模組使用介紹

本實驗使用的模組數目眾多，主要可以分成幾大類：相機擷取、螢幕顯示、記憶體存取以及車輛辨識……等四大類。各模組之間的關係如下圖所示：



（一）相機擷取

相機擷取主要由範例程式提供的 CCD_Capture(擷取影像)和 I2C_CCD_Config(初始化 CCD)，在得到影像後，經過範例程式提供的 RAW2RGB，將相機原始檔轉換成 RGB 三原色的 10bits 數值。

（二）螢幕顯示

螢幕顯示主要使用範例程式提供 VGA_Controller，輸入 RGB 三個顏色值，就可以在螢幕上顯示影像。

（三）記憶體存取

記憶體分成兩大部分，SDRAM 和 SRAM，其中 SDRAM 使用範例程式提供的一系列 SDRAM 使用程式，主要 SDRAM 用來存放 CCD 擷取的影像，存進去後再從同

一個地方拿出來，供 VGA 顯示用。並且 VGA 的 clock 和 SDRAM 的 clock 相同，讓他們可以沒有誤差地一個一個畫格輸出並播放。

SRAM 的部分延續實驗三的使用方法，主要可以讓我們將背景存入 SRAM，之後利用與 SDRAM 的影像相減的方式，做接下來的車輛辨識。而儲存背景分成自動背景跟手動控制兩種，將會在接下來的部分作介紹。

（四）車輛辨識

車輛辨識法主要針對靜止和移動兩種不同的情形作辨識，靜止辨識包含加權法(Car_Detect)，形狀辨識法(SHAPE)等。移動辨識主要是利用消失法(VANISH)，這些方法的工作原理將會在接下來的部分作詳細介紹。

三、背景資訊存取：靜態記憶體(SRAM)模組

此模組負責靜態記憶體的存取，由以下變數控制。

變數名稱	使用說明
input Read	讀出資料時設為 1。
input Write	寫入資料時設為 1。
input [19:0] ADDR	指定記憶體位址。
input [15:0] iDATA	欲寫入記憶體的資料。
output [19:0] SRAM_ADDR	指定記憶體位址。
output [15:0] oDATA	由記憶體讀出的資料。
output SRAM_CE_N	Chip enable，本實驗中都設為 0。
output SRAM_OE_N	Output enable，讀出資料時設為 0。
output SRAM_WE_N	Write enable，寫入資料時設為 0。
output SRAM_UB_N	Upper byte control，本實驗中都設為 0。
output SRAM_LB_N	Lower byte control，本實驗中都設為 0。
inout [15:0] SRAM_DQ	記憶體讀出或寫入的資料。

另外，在主模組中與靜態記憶體存取相關的其他變數如下。

變數名稱	使用說明
wire [15:0] sram_iDATA	欲寫入記憶體的資料。
wire [15:0] sram_oDATA	由記憶體讀出的資料。
wire sram_iRead	手動操控讀出資料與否，由 SW[11]控制。
wire sram_iWrite	手動操控寫入資料與否，由 SW[12]控制。
wire Read	由 VGA 操控，Read 為 1 時將資料讀出。
wire sCCD_DVAL	由 CCD 操控，sCCD_DVAL 為 1 時將資料寫入。
reg sram_Read	SRAM 模組的輸入值，sram_iRead 和 Read 同為 1 時將其設為 1。
reg sram_Write	SRAM 模組的輸入值，sram_iWrite 和 sCCD_DVAL 同為 1 時將其設為 1。
reg [19:0] sram_rADDR	記憶體的讀出位址。
reg [19:0] sram_wADDR	記憶體的寫入位址。

本實驗中，靜態記憶體的主要功能是儲存背景畫面。 將沒有移動物體時的畫面經 sram_iDATA 寫入記憶體中，再由 sram_oDATA 讀出，並與即時畫面做比較，即可判斷車流量、車速等交通資訊。

由於靜態記憶體一個位址的空間大小為 16 位元，然而相機讀取的每個像素資訊有 24 位元(RGB 三個顏色各 8 位元)，因此我們將三個顏色的強度相加，再存入靜態記憶體中。 換言之，靜態記憶體中儲存的畫面是灰階的。

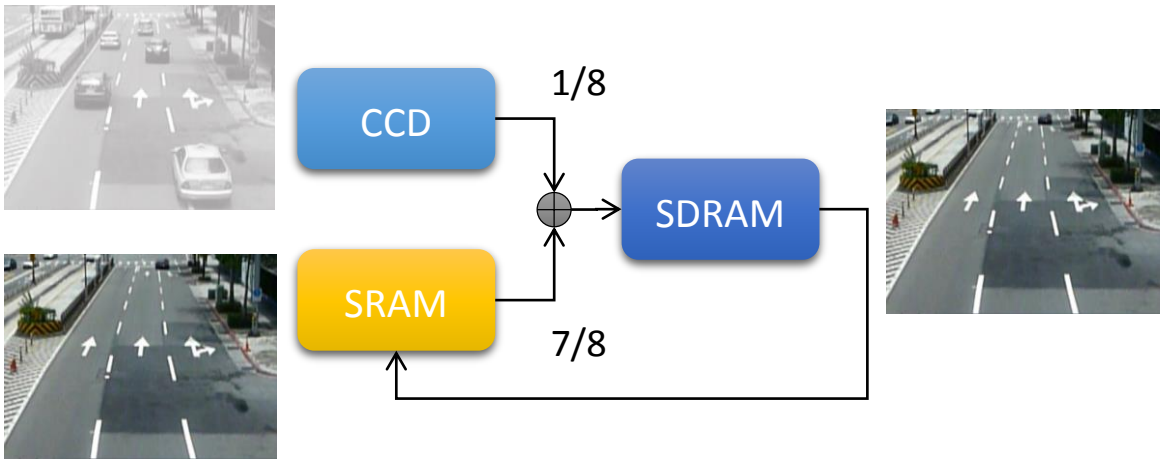
背景畫面的取得分為兩種方式，一是手動擷取，二是自動平均。 若欲手動擷取背景畫面，將變數 sram_iWrite 設為 1，變數 sram_iRead 設為 0，此時由相機讀取的畫面(sCCD_R, sCCD_G, sCCD_B)將寫入靜態記憶體中。 畫面穩定後再將 sram_iWrite 設為 0，sram_iRead 設為 1，即完成背景畫面的儲存。 此模式中，讀出記憶體位址(sram_rADDR)與 VGA 的 X,Y 座標(VGA_CorX, VGA_CorY)同步，寫入記憶體位址(sram_wADDR)則與相機(X_Cont, Y_Cont)同步。

若欲進行自動平均，則將靜態記憶體(SRAM)與動態記憶體(SDRAM)的讀寫反相進行。 當相機讀取即時畫面(sCCD_DVAL == 1)時，靜態記憶體讀出資料，並與即時畫面進行加權平均(weighted average)，再將平均結果寫入動態記憶體中。 當 VGA 欲輸出螢幕畫面(Read == 1)時，動態記憶體讀出資料，寫入靜態記憶體中。 此即完成一次背景畫面的更新。此模式中，讀出記憶體位址(sram_rADDR)與相機同步，寫入記憶體位址(sram_wADDR)則與 VGA 的 X,Y 座標同步。 另外，自動平均背景的結果與加權係數有相當大的關係，當即時畫面佔的比重越低，背景畫面的更新速度越慢。

目前自動背景所使用的計算公式如下：

$$B\_avg \leftarrow (1-\alpha) * B\_avg + \alpha * B\_sampled$$

根據實驗結果，最佳的 α 值為 1/8。其示意圖如下：



手動擷取

if (sCCD_DVAL && sram_iWrite)

sram_Read ← 0

sram_Write ← 1

sram_ADDR ← sram_wADDR

iSDRAM_R ← sCCD_R[11:2]

iSDRAM_G ← sCCD_G[11:2]

iSDRAM_B ← sCCD_B[11:2]

sram iDATA ← sCCD R[11:2] + sCCD G[11:2] + sCCD B[11:2]

if (Read && sram_iRead)

sram_Read ← 1

sram_Write ← 0

sram_ADDR ← sram_rADDR

自動平均

if (sCCD_DVAL)

sram_Read ← 1

sram_Write ← 0

sram_ADDR ← sram_rADDR

iSDRAM_R ← (sCCD_R[11:2] + {sram_oDATA[14:10], 5'd0}*7) / 8

iSDRAM_G ← (sCCD_G[11:2] + {sram_oDATA[9:5], 5'd0}*7) / 8

iSDRAM_B ← (sCCD_B[11:2] + {sram_oDATA[4:0], 5'd0}*7) / 8

if (Read)

sram_Read ← 0

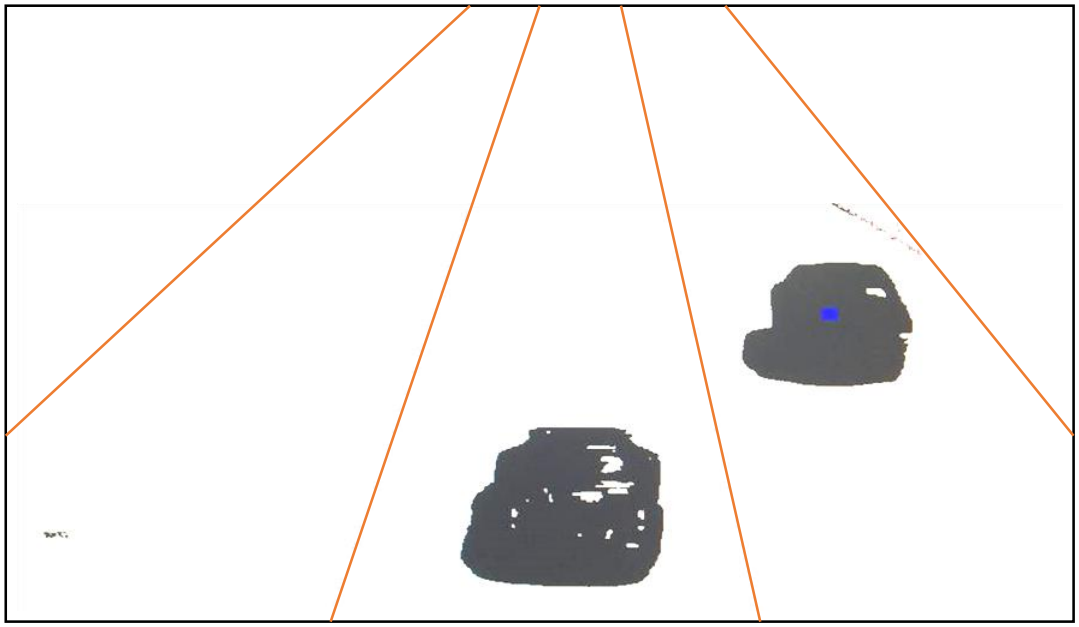
sram_Write ← 1

sram_ADDR ← sram_wADDR

sram_iDATA ← {1'b0,Read_DATA2[9:5],Read_DATA1[14:10],Read_DATA1[9:5]}

四、物件重心與靜態車輛偵測：CARS 模組

在取得背景與即時影像的相差之後，我們將影像二值化，超過一定的差值以上我們讓他變成全黑，反之則為全白。藉此，我們便能在給定一個特定的區域之後，判斷這個區域中黑色的畫格的個數，同時我們也可以藉由每個黑色的畫格的 XY 座標，判斷黑色畫格的平均重心座標。而我們將這個中心座標附近用藍色表示，此藍色方塊便是畫面中黑色畫格的中心點。如下圖所示：



在影像中，同樣大小的物體，越靠近相機所佔面積越大。以本次測資畫面而言便是同樣大小物體，Y 坐標越大，所佔像素越多。因此系統在計算目前畫格與背景差異時，除了將與背景相同色彩之像素 diff_flag 設為 0，也就是 CARS 模組輸入 Black_intensity 為 0，也會根據 Y 坐標將不同之像素的 diff_flag 加權。

而藉由此一藍色方塊的座標位移的差值以及知道每一畫格數持續的時間 (60Hz = 1/60 秒)，我們便能得到物體大約的移動速度。而為了得到更好的結果，我們將數個畫格得到的結果取平均後輸出。而取多少畫格再平均，則由電路內部參數控制。

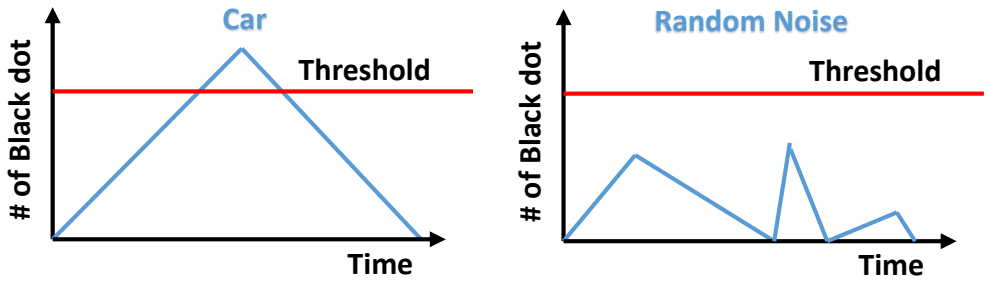
CARS 模組相關變數則如下頁表格所示：

變數類型	變數名稱	使用說明
Input	clk	時間訊號。
Input	reset	重置訊號。
Input[3:0]	Black_intensity	由外部模組產生，告知是否為黑色畫格。由於包含靜止法的不同位置的加權，此一參數有四個位元，對 Y 值座標越高的黑畫格此值越高，反之則越低。
Input	Read	表示 SDRAM 是否為讀取狀態，若非，則不做事。
Input[9:0]	iCorX	輸入畫格的 X 座標。
Input[9:0]	iCorY	輸入畫格的 Y 座標。
Input	en	由外部 Switch 控制，打開時，輸出變為即時輸入而不取平均，反之則為平均後再輸出。
Output[9:0]	oCorX	輸出的藍色方框重心 X 座標。
Output[9:0]	oCorY	輸出的藍色方框重心 Y 座標。
Output[9:0]	speedX	輸出的 X 方向速度。
Output[9:0]	speedY	輸出的 Y 方向速度。
Output[25:0]	B_total	輸出的黑色畫格總個數。
Output	valid	告知外部模組，內部判斷已經完成。

五、動態車輛偵測：VANISH 模組

在現實世界中，綠燈時的車子都在移動，移動中的物件使用之前的靜止偵測法效果便不太顯著，因此我們也提供另一個專門用於綠燈時、動態車輛的判斷法，稱為消失偵測法。

此一偵測法主要針對一個特定的區域，我們首先把判斷黑色畫格個數的區域縮小，只針對影像中道路開始的位置或道路消失的位置做判斷，由於所有車子都會經過這個地方，所以經過這個區塊的時候，影像中的黑色畫格數目會變多，離開時會變少，我們判斷這個上升在下降的過程，若有超過一定的閾值的話，便判斷剛剛有一台車經過。此方法也可避免因為影子或行人所造成的誤判，因兩者所造成的瞬間像素差異通常不超過閾值，如下圖所示：



此一方法主要由以下電路實現：

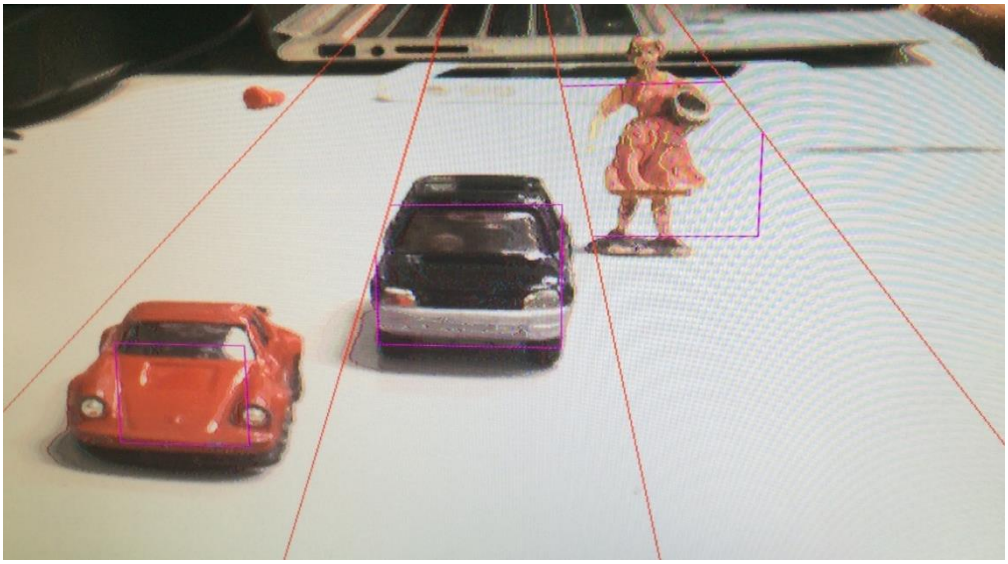
變數類型	變數名稱	使用說明
Input	iclk	時間訊號。
Input	reset_n	重置訊號，可將輸出數字歸零。
Input[19:0]	iGary	由外部模組產生，告知區域內的黑色畫格個數。
Input[7:0]	onum	累計這段時間內的車子總數。

六、物件偵測：SHAPE 模組

由於現實世界中，馬路上會出現的移動物體大多為汽車、機車、腳踏車與行人，而我們的目標就是把汽車辨識出來，其他移動物件則忽略（因為汽車數量往往才是影響交通的關鍵），巧合的是，除了汽車為方形外，其他移動物件都是長形的，所以我們的目標就變成在取得一個物件後，判斷是否為正方形，再取得藍色方框後，我們藉由以下演算法，判斷此一物件是否為正方形：

1. 從藍色方塊的正中心，取一個初始邊長，以此正中心為中心點，畫一個正方形。
2. 將此正方形縮小，邊長減少一個固定的差值，若此一方塊邊長小於最低臨界值，則變回初始邊長。
3. 判斷上次的正方形中的黑色畫格數，和這次的黑色畫格數的差值，差值若大於一個閾值，則進入步驟四，反之回到步驟二。
4. 判斷方塊中的黑色畫格佔方塊中總畫格的比例，若比例超過一個閾值，則判斷是車子，若非，則判斷不是車子。

藉由此演算法，我們可以判斷道路上的物件是汽車還是其他物件。下圖為實際實測畫面，如圖所示，若不是車輛，而是行人或其他物件，便無法大致佔滿整個方框。而在目前版本的程式碼，若經由此方法判別為非車輛，便不會顯示紫色方框。



此一演算法由以下電路實現：

變數類型	變數名稱	使用說明
Input	Clk	時間訊號。
Input	Reset	重置訊號。
Input	Diff	由外部模組產生，告知是否為黑色畫格。
Input	Read	表示 SRAM 是否為讀取狀態，若非，則不做事。
Input	Shape_inside	由外部產生，告知此一畫格是否在此方塊內 只有在 Read,Diff,Shape_inside 都非零的時候，內部計算黑色畫格個數的電路才會計算此一畫格。
Input	en	由外部 Switch 控制，打開時，停止縮小的方框可以回歸原始大小，再繼續縮小，關閉時則停止後就部會在變化。
Input[9:0]	iCor_Diff	輸入的方框邊長。
output[9:0]	oCor_Diff	輸出的方框邊長，外部電路會把此一輸出結果變成新的方框邊長。
output	Car	若是車，則為一，反之則為零。
output	Valid	告知外部模組，內部判斷已經結束，此一結果是正確的。

七、畫面顯示：Display 模組

Display 模組用於畫面左方顯示的道路資訊，每 50X32 為一行，可顯示 6 位元字母。每一行皆須使用一個 module 操控，視編碼者愈顯示行數操控。Display 模組除了輸入時脈訊號（clk）與重置訊號(rst)外，還為輸入目前畫面的 X 和 Y 座標，藉此計算並回傳該像素是否為文字像素，若為文字像素，將回傳為 1。相關變數如下：

變數類型	變數名稱	使用說明
Input	clk	時脈訊號。
Input	reset	重置訊號。
Input	CorX	畫面 X 座標。
Input	CorY	畫面 Y 座標
Input	b0,b1,b2, b3,b4,b5	分別為第一到第六字元所要顯示的字母編號。
Output	value	若該像素為顯示字母的一部分，輸出 1。

另外也需輸入六個字元所預顯示的字元代碼，其對應如下：

編號	字元	編號	字元	編號	字元	編號	字元
0	0	10	A	20	O	30	#
1	1	11	B	21	P	31	N
2	2	12	C	22	R	32	空白
3	3	13	D	23	S		
4	4	14	E	24	T		
5	5	15	F	25	U		
6	6	16	G	26	V		
7	7	17	H	27	Y		
8	8	18	I	28	?		
9	9	19	L	29	:		

Display 模組目前並不支援所有英文字與符號，若預新增顯示字母，可修改 Display 模組中以 alphabet 為根據的 case 函數，目前空間最多可支援 64 個字母。

六、討論與未來展望

（一）參數自動設定

目前系統當中設定了許多參數，如：道路區域畫分（region1、region2、region3）、像素差異判別係數（THRESHOLD DISAPPEAR）、自動背景係數（iR_avg、iG_avg、iB_avg）、靜態法車輛偵測（num_cars1、num_cars2、num_cars3），這些部分都希望能修改為系統自動調整參數值。

在自動背景參數的部分，可參考以下兩篇論文：

"Automatic threshold decision of background registration technique for video segmentation", Yu-Wen Huang, Shao-Yi Chien, Bing-Yu Hsieh, and Liang-Gee Chen (2002)

"Efficient Moving Object Segmentation Algorithm Using Background Registration Technique", Shao-Yi Chien, Shyh-Yih Ma, and Liang-Gee Chen (2002)

然而若要實作兩篇論文的方法，需要能一次取得整個畫面資訊，以 kernel 方式掃過整個畫面進行運算。這部分或許可搭配 NIOSII 運算，但使用 NIOSII 則有無法即時輸出的問題需解決。

（二）雜訊去除

系統當中目前還有些微雜訊，雖不至於影響偵測準確度，但仍希望能除去。主要的 latch 來源是範例程式當中的 VGA_Controller 以及 SDRAM 相關模組，若能除去這些 latch，可望提升畫值。

（三）聯網、紅綠燈溝通

在我們的企畫書中有提到希望能達到紅綠燈之間互相溝通、交換路況的功能，這部分在此版本中來不及實作。經嘗試確認可用外接 Arduino 藍芽模組的方式傳送資料，由於運算部分仍是由 FPGA 完成，Arduino 的低運算效能應該不會影響整個系統效率。但仍可嘗試使用 RS323、Ethernet 或 WiFi 來傳送，訊號較為穩定、傳送範圍較廣。

（四）準確度提升

目前是簡單以相異畫素數目作為運算基礎。若能加入更準確的形狀或顏色考量，可提升準確度，避開特殊狀況下的誤差，例如成果發表當日同學提到車子顏色與道路相同的情況、我們實測時發現的陰影問題。