111-1　535313

進階可程式邏輯系統設計與應用

Advanced Programmable Logic System Design and Application

實驗編號 : 期末專題

實驗名稱 : D5M 相機和不簡單的外部記憶體使用

結報完成日期 : 2022.10.16

姓名 : 王語

系級 : 機械四

學號 : 0811127

一、 實驗目的

考驗學生查找資源以及閱讀開源文件之能力，此外這也是較有規模的專案開發，是一個策畫架構的好練習。

此實驗涉及影像捕捉以及 VGA 顯示原理，需熟悉 bayer code 如何轉換成 RGB888 格式，並且配置相關記憶體區間，確保每一次顯示畫面都是完整的。

二、 Verilog 程式碼

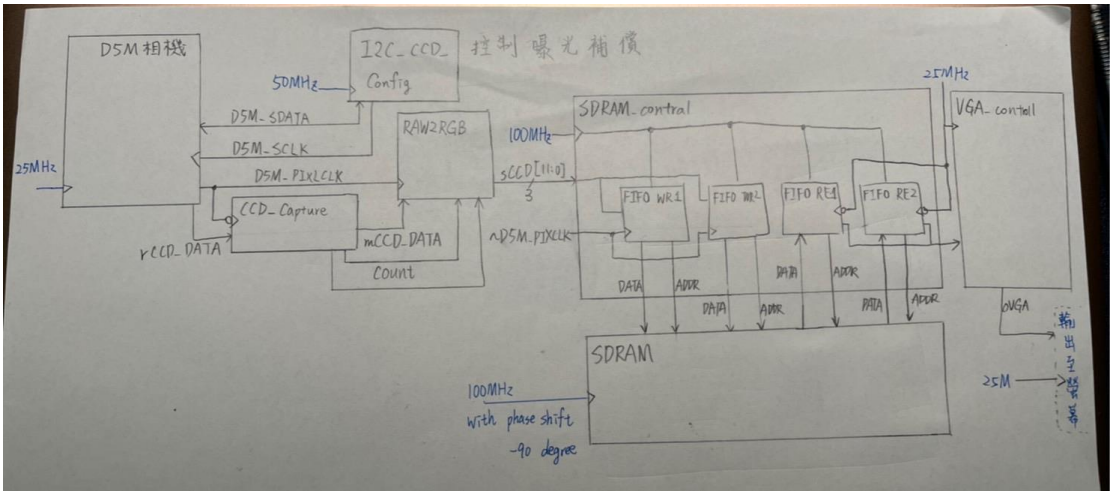
我是使用 DEMO code 改寫完成專題的，源碼連結:
http://mail.terasic.com.tw/~bingxia/DE1_SoC_CAMERA.zip
因為最後儲存三秒畫面的部分沒有完成，只貼上 RBG 轉換的部分，以節省版面，其他程式碼會在問題與討論附上。

我將原本的賦值註解，以一個 casex 完成:

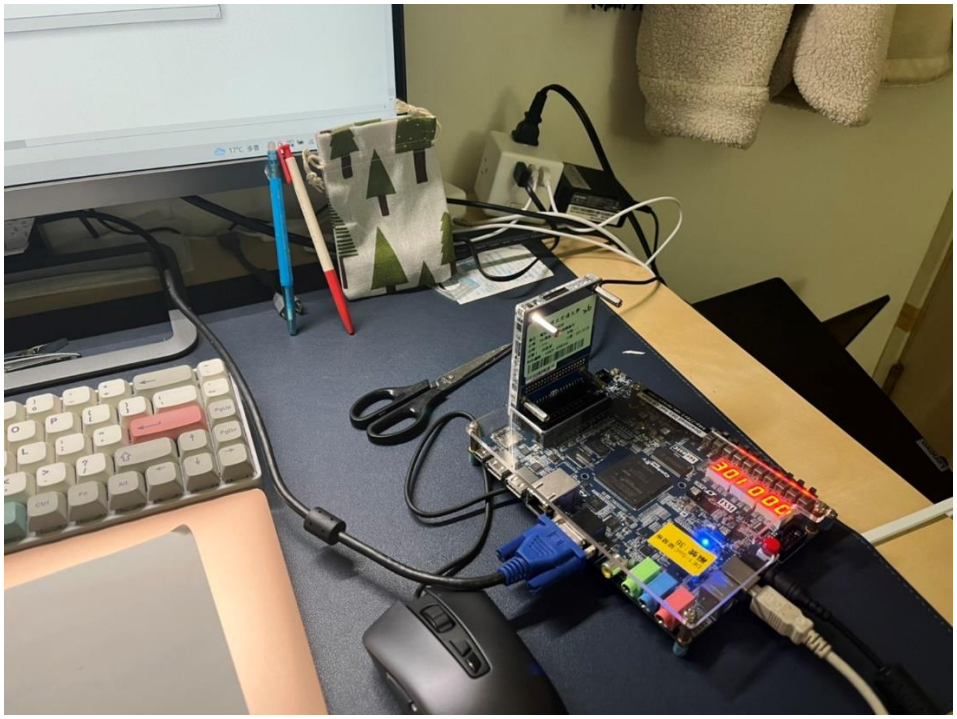
```
////fetch the high 8 bits
//assign  VGA_R = oVGA_R[9:2];
//assign  VGA_G = oVGA_G[9:2];
//assign  VGA_B = oVGA_B[9:2];

always @(*) begin
   casex (SW[3:2])
      2'b1X :begin
         VGA_R = oVGA_B[9:2];
         VGA_G = oVGA_R[9:2];
         VGA_B = oVGA_G[9:2];
      end
      2'b01 : begin
         VGA_R = oVGA_G[9:2];
         VGA_G = oVGA_B[9:2];
         VGA_B = oVGA_R[9:2];
      end
      2'b00 : begin
         VGA_R = oVGA_R[9:2];
         VGA_G = oVGA_G[9:2];
         VGA_B = oVGA_B[9:2];
      end
      default begin
         VGA_R = oVGA_R[9:2];
         VGA_G = oVGA_G[9:2];
         VGA_B = oVGA_B[9:2];
      end
   endcase
end
```

三、 結構圖



四、 實驗結果



鏡頭與環境配置

正常畫面

RGB 轉換

RGB 轉換

五、 Triple buffer 狀態機與 Verilog

設計思路 ：

找出三個不同的記憶體位置區間，每一個區間存放一張 frame，一秒切換一次，當需要回放時，停止寫入新的影響，重複讀取這三個區間的影像作為輸出。

記憶體配置 ：

圖 1 記憶體位置配置

以上三組記憶體位置經過驗證是可以使用的，驗證方式是直接接在 Sdram_Control 的腳位上，可以顯示出正常畫面，不存在破損、顏色失真等問題。

驗證方式 ：

圖 2 驗證記憶體位置 1 範例 verilog

助教可能會好奇為什麼記憶體位置要配置成這樣，不是以 23’h100000 遞增，這部份真的很玄，一時之間我也解釋不出來，像是把 WR/RD1 配置在 23’h100000 ~ 23’h100000 + 640*480 ；WR/RD2 配置在 23’h200000 ~ 23’h200000 + 640*480
，就無法顯示出正常畫面。

經過超級多次的測試，WR/RD 1 的配置一定要是 23’h100000 的 0 2 4 倍，WR/RD 2 的配置一定要是 23’h100000 的 1 3 5 倍，這樣會無法在記憶體容量內找出三個不同的區間，存放三秒個別的畫面，或要以 23’h100000 為間隔配置記憶體，則會使用到 23’h400000 ~ 23’h400000 + 640*480 以及 23’h500000 ~ 23’h500000 + 640*480 ，不過這個位置顯現的畫面完全無法辨識，都是雜訊。連結為顯示畫面 https://pse.is/4pc6js

狀態機設計 ：

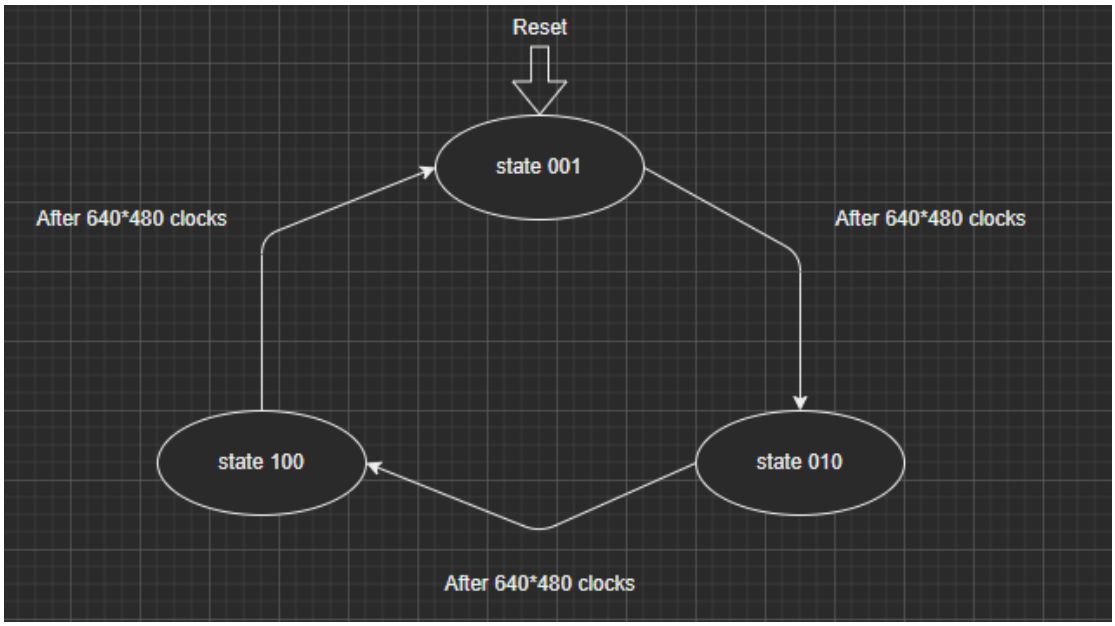


圖 3 狀態機設計

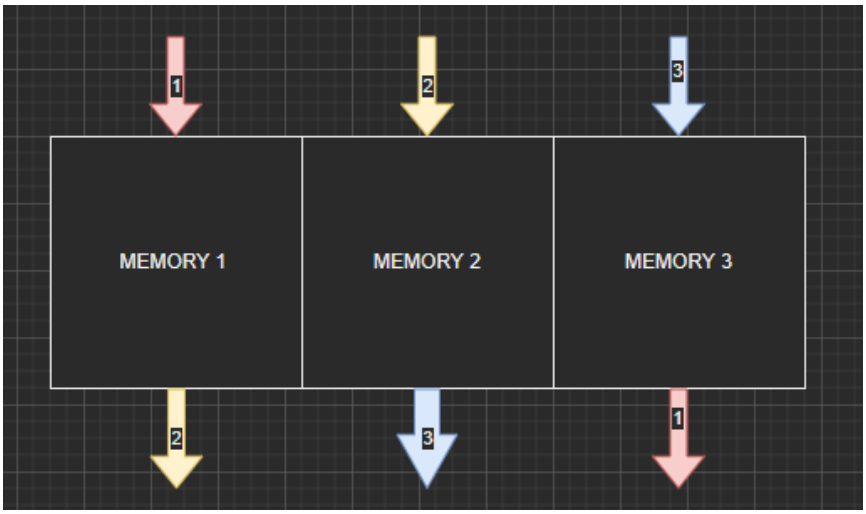
讀取狀態機的 CLK 使用 posedge ~VGA_CTRL_CLK，寫入狀態機的 CLK 使用 posedge ~D5M_PIXLCLK，圖 5 是讀取狀態機的切換 verilog，寫入狀態的切換也同理。

```
391  initial begin
392      read_counter = 50'd0 ;
393      write_counter = 50'd0 ;
394      write_state_of_memory = 3'b001 ;
395      read_state_of_memory = 3'b001 ;
396  end
```

圖 4 狀態初始化

```
445      case (read_state_of_memory)
446        3'b001: begin
447          read_state_of_memory <= 3'b010 ;
448        end
449        3'b010: begin
450          read_state_of_memory <= 3'b100 ;
451        end
452        3'b100: begin
453          read_state_of_memory <= 3'b001 ;
454        end
455        default: begin
456          read_state_of_memory <= 3'b001 ;
457        end
458      endcase
```

圖 5 讀取狀態轉移 verilog



STATE 001	寫入位置	記憶體配置 1
	讀取位置	記憶體配置 2
STATE 010	寫入位置	記憶體配置 2
	讀取位置	記憶體配置 3
STATE 100	寫入位置	記憶體配置 3
	讀取位置	記憶體配置 1

不過最後沒能實現完整畫面，我認為是除了狀態切換的機制設計錯誤，尚存在其他問題。

六、 問題與討論

接續討論 Triple buffer 所碰上的問題，若資料的寫入是完整的，讀取也是完整的，則不應該有畫面破碎、色塊分離、超恐怖綠色雜訊的結果，但這些我都遇上了，說明我的設計存在嚴重錯誤，。

狀態之間轉換的機制我自己也抱持懷疑，我認為數 clk 並非一個萬無一失的方法，此外，從結果而言，我數的 CLK 數目也不正確，可能多數或是少數，因此造成畫面被切割 ： https://pse.is/4qeexv

除此之外，我認為是因為沒有個別考慮寫入與讀取的兩個 FIFO，才致使紅色與藍色錯位，我可能還要重新設計狀態機。

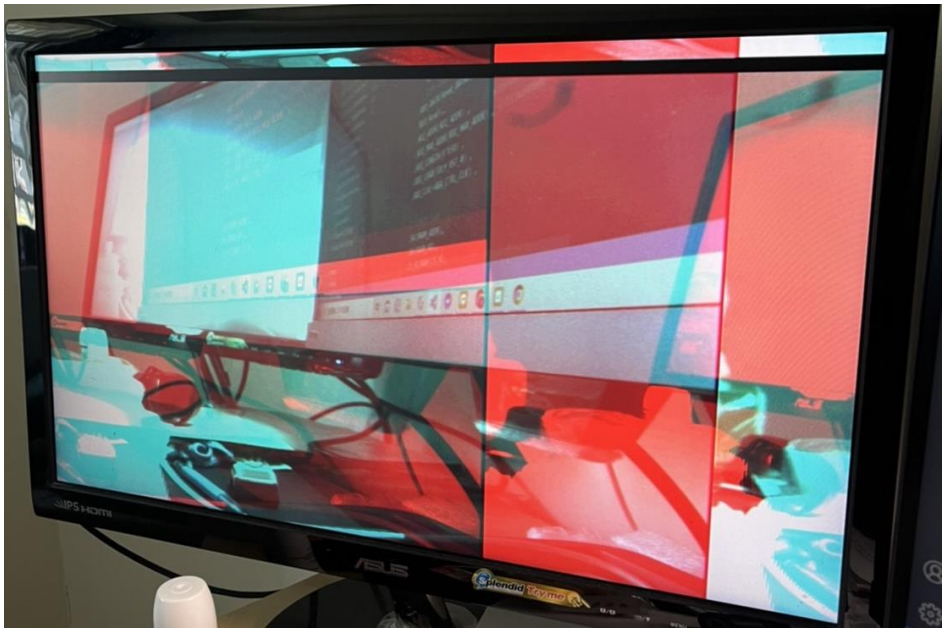


圖 6 色塊分離以及畫面切割說明圖

而每當讀取位置切換到記憶體位置 3 的時候，都會出現大量雜訊，按下 key[0] RESET 也沒辦法解決，問題成因 ：

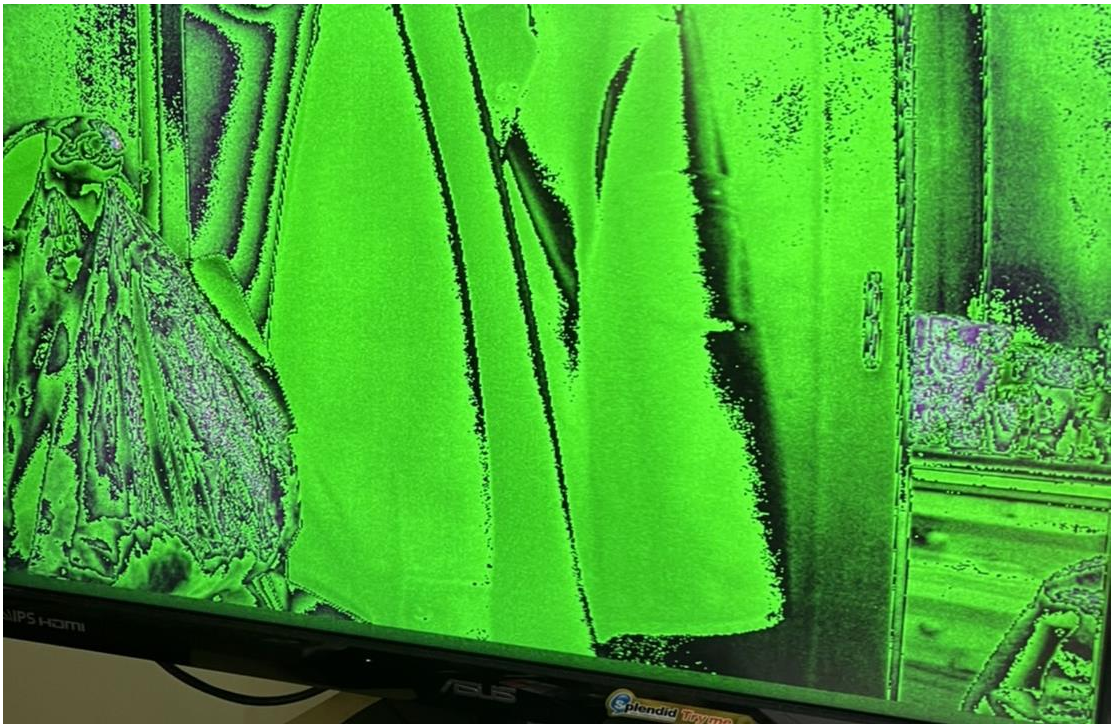


圖 7 超恐怖綠色雜訊

最後是測試中無意間發現的 BUG，我將所有狀態的記憶體讀取、寫入位置配置成記憶體位置 1，但是畫面卻會出現訊，我認為這樣的做法和直接固定記憶體位置是一樣的，但是卻會造成雜訊，很可能是切換過程中，遺漏了其中一個 FIFO 的資料。

```
512  //READ
513  always @(*) begin
514     // read_state_of_memory = 2'b01 ;
515     case (read_state_of_memory)//write_state_of_memory) // read_state_of_memory {SW[5:3]}
516        3'b010 : begin
517           RD1_ADDR = 0 ;
518           RD1_MAX_ADDR = 640*480  ; // + 23'd307200 = 640*480
519           RD2_ADDR = 640*480+10 ; // RD1_ADDR + 23'h100000
520           RD2_MAX_ADDR = 640*480+10+640*480 ;// blocking
521        end
522        3'b100 : begin
523           RD1_ADDR = 0 ;
524           RD1_MAX_ADDR = 640*480  ; // + 23'd307200 = 640*480
525           RD2_ADDR = 640*480+10 ; // RD1_ADDR + 23'h100000
526           RD2_MAX_ADDR = 640*480+10+640*480 ;// blocking
527        end
528        3'b001 : begin
529           RD1_ADDR = 0 ;
530           RD1_MAX_ADDR = 640*480  ; // + 23'd307200 = 640*480
531           RD2_ADDR = 640*480+10 ; // RD1_ADDR + 23'h100000
532           RD2_MAX_ADDR = 640*480+10+640*480 ;// blocking
533        end
534        default: begin
535           RD1_ADDR = 0 ;
536           RD1_MAX_ADDR = 640*480  ; // + 23'd307200 = 640*480
537           RD2_ADDR = 640*480+10 ; // RD1_ADDR + 23'h100000
538           RD2_MAX_ADDR = 640*480+10+640*480 ;// blocking
539        end
540     endcase
541  end
```

```
468   // write
469   always @(*) begin // 4 frame
470       // read_state_of_memory = 2'b10 ;
471       // write_state_of_memory = 2'b00 ;
472       case (write_state_of_memory)
473           3'b001 : begin // good
474               WR1_ADDR = 0 ;
475               WR1_MAX_ADDR = 640*480 ; // 23'd307200 = 4B000
476               WR2_ADDR =  640*480+10 ; // WR1_ADDR + 23'h100000
477               WR2_MAX_ADDR = 640*480+10+640*480 ;// blocking
478           end
479           3'b010 : begin // bad
480               WR1_ADDR = 0 ;
481               WR1_MAX_ADDR = 640*480 ; // 23'd307200 = 4B000
482               WR2_ADDR =  640*480+10 ; // WR1_ADDR + 23'h100000
483               WR2_MAX_ADDR = 640*480+10+640*480 ;// blocking
484           end
485           3'b100 : begin
486               // WR1_ADDR = (640*480)*4+4 ;
487               WR1_ADDR = 0 ;
488               WR1_MAX_ADDR = 640*480 ; // 23'd307200 = 4B000
489               WR2_ADDR =  640*480+10 ; // WR1_ADDR + 23'h100000
490               WR2_MAX_ADDR = 640*480+10+640*480 ;// blocking
491           end
492           default: begin
493               WR1_ADDR = 0 ;
494               WR1_MAX_ADDR = 640*480 ; // 23'd307200 = 4B000
495               WR2_ADDR =  640*480+10 ; // WR1_ADDR + 23'h100000
496               WR2_MAX_ADDR = 640*480+10+640*480 ;// blocking
497           end
498       endcase
499   end
500
```

七、 心得

這次專題讓我感觸深切，雖然最後只有拿到八十五分，不過收穫非常非常多，我因為這個專題，我了解了 VGA 顯示原理、SDRAM 以及 DDR3 運作原理、異步以及同步 FIFO 的架構以及操作、三重緩衝的實現……，Quartus II PLL IP 的使用，我覺得我在嵌入系統方面有顯著的進步，如

果時間再多個五天，或許能將三秒影片的回放實現，老實說挫折感也挺重的，社會是結果論，分數也是，即便我投入非常多時間，犧牲很多玩樂、睡眠時間，讓自己延後兩個禮拜放寒假，沒做出來就是沒做出來，除了認清自己能力尚有許多進步空間之外，也認清知自身知識的匱乏，有了這次經驗，再下一次大型專案開發之前，我會先設計好系統的架構圖，並且將時序釐清，還要會一些通靈，期許我能繼續堅持在這條路上，越來越熟稔電路的設計與程式撰寫。

八、 特別感謝

即便功虧一簣，不過依舊非常感謝政權助教，願意投入自己的時間和心力協助專題，也無私的分享許多經驗，好人一生平安。