Integrated Circuit Design

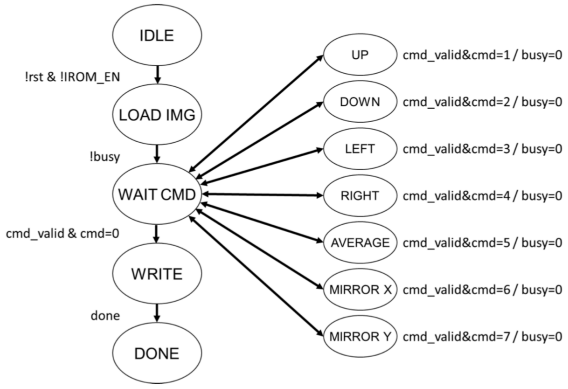
# HW3 Image Display Control
# Report

電機三 B05901030 陳欽安

## 一、有限狀態機 Finite-state Machine

- *States for Imager Display Controller*: 參考pdf

  - IDLE
  - LOAD_IMG
  - WAIT_CMD
    - SHIFT_UP
    - SHIFT_DOWN
    - SHIFT_LEFT
    - SHIFT_RIGHT
    - AVERAGE
    - MIRROR_X
    - MIRROR_Y
  - WRITE
  - DONE




## 二、Registers and Wires

① *For States*

reg [2:0] state_ctrl, state_ctrl_nxt: 紀錄Image Display Controller的states

reg [2:0] state_cmd: 紀錄Command的states

wire [2:0] state_cmd_nxt: 用來紀錄當下輸入的command

② *For Data Memorization*: 紀錄8x8大小的Image Data

```
//image data 64 * 8bits reg wire array
reg [7:0] DATA_TABLE [0:63]
reg [7:0] DATA_TABLE_nxt [0:63]
```

③ *For Output*

```
//register for output
```

reg [7:0] IRB_D_r, IRB_D_nxt: 紀錄 DATA_TABLE[address]的值

reg [6:0] IRB_A_r, IRB_A_nxt: 紀錄輸出DATA的address

reg [6:0] IROM_A_r, IROM_A_nxt: 紀錄跟Image ROM 要的DATA的address

reg IROM_EN_r, IROM_EN_nxt: 紀錄IROM是否可以被讀取

reg done_r, done_nxt: 紀錄整個operation 是否完成了

reg busy_r, busy_nxt: 紀錄Image Display Controller是否正在執行command

reg IRB_RW_r, IRB_RW_nxt: 紀錄Image Register Bank是否可以被寫入

```
//wires for output
```

④ *For Operation*

```
//reg for the 4x4 grid ready to be changed left->right top->down 0~3
```

reg [11:0] GRID [0:3], GRID_nxt [0:3]:  紀錄operation point周圍四格中的DATA

```
//reg for operation point
```

reg [5:0] x, x_nxt: 紀錄x座標

reg [5:0] y, y_nxt: 紀錄y座標

//wire x,y: 為了取的GRID中四個DATA的address值
wire [5:0] y_up: 紀錄 (y-1)*8
wire [5:0] y_down: 紀錄 y*8
wire [5:0] x_left: 紀錄 x-1

//wire averge number
wire [7:0] avg_num: 紀錄目前GRID周圍四個DATA值的平均值
wire [9:0] avg_num_tmp: 因為4個8 bits相加最多會到10bits所以先用10bits紀錄再除以4(右shift2) assign給 avg_num

## 三、問題與解決方式

① *States*：

這次的實作依照pdf可分為12個states，如果全部放在同一個combinational circuit裡面會太複雜增加debug難度，因次我將states分為兩個combinational circuits來implement：處理ctrl的以及處理cmd的。

**- Circuit for ctrl:**

處理的States: IDLE, LOAD_IMG, WAIT_CMD, WRITE, DONE

處理的registers: state_ctrl_nxt, IRB_A_nxt, IRB_D_nxt, IRB_RW_nxt, busy_nxt, done_nxt, IROM_EN_nxt

**-Circuit for cmd**: (會改到DATA_TABLE, GRID_nxt之值的)

處理的States:
LOAD_IMG, WAIT_CMD, SHIFT_UP, SHIFT_DOWN, SHIFT_LEFT, SHIFT_RIGHT, AVERAGE, MIRROR_X, MIRROR_Y

處理的registers:
DATA_TABLE_nxt, GRID_nxt, x_nxt, y_nxy, IROM_A_nxt

② *busy signal* :

由於一開始我把處理busy的過程寫在Circuit for cmd內，在每次進到要處理cmd前先把busy拉高，然後處理完cmd後把busy拉回low；然而因為combinational circuit會平行處理，所以將會使得busy訊號永遠是低的與題目不符。

因此我將處裡busy的過程放到Circuit for ctrl中，在LOAD_IMG要進到WAIT_CMD前先把busy調低以等待cmd輸入，接著在WAIT_CMD時分別判斷兩個條件：

I. **if(cmd_valid && !busy_r)**：表示現在可以接受cmd，下一個clk 開始執行cmd operation，故把busy_nxt調高。
II. **if(!cmd_valid || busy_r)**：表示現在cmd不被接受或者現在有cmd operation正在執行，由於這次的cmd operation都能在一個clk裡面處理完，因此表示在下一個clk ctrl將會無事可做繼續等待cmd，故把busy_nxt調低。

③ *AVERAGE overflow*:

一開始在驗證tb_2時，我發現即使operation point四周的DATA值很大，avg_num卻異常的小，才發現到addition有overflow的問題要處理。因為這次DATA都是8bits，所以四個8bits相加最多會到10bits，於是我的處理方法是先另一個10bits的wire: avg_num_tmp紀錄operation point周圍的四個點相加的值，再把此值right shift 2 bits(除以4) assign 給原本的avg_num，如此便能夠正常操作AVERAGE operation。