96 年度大學院校 積體電路設計競賽

標準單元式(Cell-Based)競賽初賽參考解答

研究所/大學組

一、設計結果報告(report.000)

二、暫存器轉換階層(RTL level)設計結果

三、模擬結果

一、設計結果報告(report.000)

隊號(Team number): ***CIC***

----------RTL category----------
使用之 HDL 名稱(Verilog or VHDL): **Verilog**

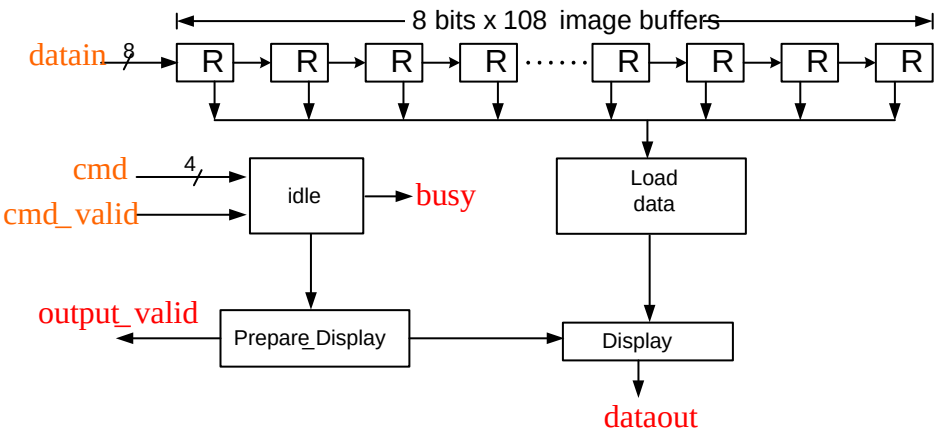
RTL 檔案名稱(RTL Netlist file name): **lcd_ctrl.v**

----------Pre-layout gate-level----------
Gate-Level 檔案名稱: **lcd_ctrl.vg**
Gate-Level simulation SDF: **lcd_ctrl.sdf**
Design Compiler 合成資料庫: **lcd_ctrl.ddc**

Simulation clock period: 6 ns

其他說明事項(Any other information you want to specify:(如設計特點 ...)

1、 此電路設計的方式是使用一個 decoder 分析 cmd 的數值，再分別利用各種 cmd 的不同 index 來由 image memory 中選擇所需要的 16 點 image 的 memory data 輸出。
2、 整體架構如圖一所示，首先 idle state 要先判讀指令使否有效的情況下，依照有效指令的種類去設定其各指令變數值。並且依照其 Zoom In mode 或 Zoom Fit mode 及先前的 cmd 是否使得畫面 rotate，都將影響變數的設定值。並且在輸入有效的指令之後，將 busy 設定為 1，讓系統不再接收其他指令
3、 透過 load data state 連續輸入 108 筆資料，並且將資料存在 memory 中，當 108 筆資料連序輸入完成後將 output_valid 設定為 1，以準備輸出資料。
4、 Prepare Display state 的設計上面，只是依據題目要求，設定將要用來定義輸出 image data 的邊界與 L 軸及 W 軸資料數目的變數值。最後在 Display state 將所定義的變數值透過 Zoom In 及 Zoom Fit 不同的輸出公式來找出相對應的 memory data 輸出。
5、 由於題目在評分標準上面是以上傳時間為主，所以採用最單純直接的方式直接透過選擇特定的 memory data 當作是輸出。



圖一　架構圖

二、暫存器轉換階層(RTL level)設計結果 (Verilog HDL)

```
module LCD_CTRL(clk, reset, datain, cmd, cmd_valid, dataout, output_valid, busy);
input           clk;
input           reset;
input   [7:0]   datain;
input   [3:0]   cmd;
input           cmd_valid;
output  [7:0]   dataout;
output          output_valid;
output          busy;

reg [2:0] state;    // define four state for "idle", "load data", "prepare display", "display"
reg output_valid; // the dataout is the effective signal, High enable
reg zoom;
    //zoom=0 fit
    //zoom=1 zoomout
reg [1:0] rotate;
    // rotate=0    no rotate
    // rotate=1    rotate right
    // rotate=2    180 degree
    // rotate=3    rotate left
reg [7:0] mem [107:0];    // store 108 datain signal
reg busy;        // System busy signal, when any function is executed, the busy signal will keep 1. In this condition, any datain is invalid.
reg [3:0] orgl;
reg [3:0] orgw;
reg [7:0] fitindex [15:0];    // Used in Zoom Fit mode to find the value of display point of data memory
reg [7:0] loadcount;    // count of load 108 datain sequentally, count from 0 to 107
wire [3:0] fitmap;
wire [7:0] fitxy;
wire [7:0] zoominxy;
reg [1:0] offsetx;
reg [1:0] offsety;

assign    dataout=(zoom==0)?mem[fitxy]:mem[zoominxy];
                // zoom = 0 : Zoom Fit mode
```

```
                // zoom = 1 : Zoom In mode
assign    fitmap = offsetx + offsety*4;
assign fitxy=fitindex[fitmap];
assign    zoominxy = orgl+offsetx + (orgw+offsety)*12;

always@( negedge reset)       // define the display data index each cycle, only set in reset period
begin
  fitindex[0]=13;
  fitindex[1]=16;
  fitindex[2]=19;
  fitindex[3]=22;
  fitindex[4]=37;
  fitindex[5]=40;
  fitindex[6]=43;
  fitindex[7]=46;
  fitindex[8]=61;
  fitindex[9]=64;
  fitindex[10]=67;
  fitindex[11]=70;
  fitindex[12]=85;
  fitindex[13]=88;
  fitindex[14]=91;
  fitindex[15]=94;
end


always @(posedge clk)
begin
      if(reset==1)          // reset
      begin
            state<=0;
            zoom<=0;
            busy<=0;
            rotate<=0;
      end
      else
      begin
            if(state==0)    ////////////////////////idle state
            begin
                  output_valid=0;
                  if(cmd_valid==1 && busy==0)       // command accepted when busy = 0
                  begin
                        if(cmd==4'd0)                 // cmd==0, goto "load data" stage, initialize parameter of "load data"
                        begin
                              loadcount=0;
                              state<=1;
                              busy<=1;
                        end
                        else
                        begin
```

```
state<=2;
busy<=1;
if((cmd==4'd1)&&(zoom==0)) //Zoom Fit mode + cmd=1, it's rotate left function
    rotate<=rotate-1;       // rotate left
else if((cmd==4'd2)&&(zoom==0)) //rotate right
    rotate<=rotate+1;       // rotate right
else if(cmd==4'd3) // enable zoom In mode
begin
    zoom<=1;
    orgl<=4;
    orgw<=3;
end
else if(cmd==4'd4) // enable zoom Fit mode
begin
    zoom<=0;
    orgl<=4;
    orgw<=3;
end
else if((cmd==4'd5)&&(zoom==1)) //Zoom Fit mode, enable "shift right" function
begin
    if(rotate==0)    // if "no rotate"
    begin
        if(orgl<8)
            orgl<=orgl+1;
    end
    else if(rotate==1)    // else if "rorate right"
    begin
        if(orgw>0)
            orgw<=orgw-1;
    end
    else if(rotate==3)    // else if "rotate left"
    begin
        if(orgw<5)
            orgw<=orgw+1;
    end
end
else if((cmd==4'd6)&&(zoom==1)) //Zoom Fit mode, enable "shift left" function
begin
    if(rotate==0)    // if "no rotate"
    begin
        if(orgl>0)
            orgl<=orgl-1;
    end
    else if(rotate==1)    // else if "rorate right"
    begin
        if(orgw<5)
            orgw<=orgw+1;
    end
    else if(rotate==3)       // else if "rotate left"
    begin
```

```
                        if(orgw>0)
                            orgw<=orgw-1;
                    end
                end
                else if((cmd==4'd7)&&(zoom==1)) //shift up
                begin
                    if(rotate==0)   // if "no rotate"
                    begin
                        if(orgw>0)
                            orgw<=orgw-1;
                    end
                    else if(rotate==1)    // else if "rorate right"
                    begin
                        if(orgl>0)
                            orgl<=orgl-1;
                    end
                    else if(rotate==3)       // else if "rotate left"
                    begin
                        if(orgl<8)
                            orgl<=orgl+1;
                    end
                end
                else if((cmd==4'd8)&&(zoom==1)) //shift down
                begin
                    if(rotate==0)   // if "no rotate"
                    begin
                        if(orgw<5)
                            orgw<=orgw+1;
                    end
                    else if(rotate==1)    // else if "rorate right"
                    begin
                        if(orgl<8)
                            orgl<=orgl+1;
                    end
                    else if(rotate==3)       // else if "rotate left"
                    begin
                        if(orgl>0)
                            orgl<=orgl-1;
                    end
                end
            end
        end
    end
    else if(state==1)     // load data, load the input signal data, total 108 data sequentially
    begin
        output_valid=0;
        if(loadcount==107)    // load data finish !!
        begin
            state<=3;
            busy<=1;
```

```
            output_valid=1;
            orgl<=4;
            orgw<=3;
            offsetx<=0;
            offsety<=0;
            zoom<=0;
            rotate<=0;
        end
        mem[loadcount]=datain;
        loadcount=loadcount+1;
end
else if(state==2)     //prepare display,    define the offsetx and offsety value
begin
        if(rotate==0)    // no rotate
        begin
            offsetx<=0;
            offsety<=0;
        end
        else if(rotate==1) //rotate right
        begin
            offsetx<=0;
            offsety<=3;
        end
        else if(rotate==2) //rotate 180 degree
        begin
            offsetx<=3;
            offsety<=3;
        end
        else if(rotate==3) //rotate left
        begin
            offsetx<=3;
            offsety<=0;
        end
        output_valid=1;
        state<=3;
        busy<=1;
end
else if(state==3)     //display, use the offsetx and offsety to compute the index value of fitxy & zoominxy
begin
        if(rotate==0)       //no rotate
        begin
            if(offsetx==3)
            begin
                if(offsety==3)
                begin
                    state<=0;        // goto idle mode
                    busy<=0;         // disable busy signal, ready to accept another command
                    output_valid=0;    // output signal is invalid.
                end
                offsety<=offsety+1;
```

```
                end
                offsetx<=offsetx+1;
            end
            else if(rotate==1) //rotate right
            begin
                if(offsety==0)
                begin
                    if(offsetx==3)
                    begin
                        state<=0;
                        busy<=0;
                        output_valid=0;
                    end
                    offsetx<=offsetx+1;
                end
                offsety<=offsety-1;
            end
            else if(rotate==3) //rotate left
            begin
                if(offsety==3)
                begin
                    if(offsetx==0)
                    begin
                        state<=0;
                        busy<=0;
                        output_valid=0;
                    end
                    offsetx<=offsetx-1;
                end
                offsety<=offsety+1;
            end
        end
    end
end


endmodule
```

三、模擬結果 (Gate-level)

(a) 模擬之 log 檔

```
ncverilog: 06.10-p001: (c) Copyright 1995-2007 Cadence Design Systems, Inc.
TOOL:     ncverilog 06.10-p001: Started on May 22, 2008 at 14:22:15 CST
ncverilog
     testfixture.v
     lcd_ctrl.vg
     -v
     tsmc13_neg.v
     +access+r
file: testfixture.v
     module worklib.test:v
          errors: 0, warnings: 0
file: lcd_ctrl.vg
     module worklib.LCD_CTRL:vg
          errors: 0, warnings: 0
   ...
ncsim> source /usr/cad/cadence/IUS/cur/tools/inca/files/ncsimrc
ncsim> run
Novas FSDB Dumper for Verilog-XL, Release 2007.10 (SOLARIS) 10/02/2007
Copyright (C) 1996 - 2007 by Novas Software, Inc.
*Novas* Create FSDB file 'LCD_CTRL.fsdb'
*Novas* Begin traversing the top scope(test), layer(0).
*Novas* End of traversing the top scope(test)
-----------------------------------------------------

Congratulations! The first  test you have passed!
Congratulations! The second test you have passed!
Congratulations! The third  test you have passed!
Congratulations! The fourth test you have passed!
Congratulations! The fifth test you have passed!
Congratulations! The sixth  test you have passed!
Congratulations! All data have been generated successfully!

------------------------PASS-----------------------

Simulation complete via $finish(1) at time 330551 NS + 0
./testfixture.v:180      $finish;
ncsim> exit
TOOL:     ncverilog 06.10-p001: Exiting on May 22, 2008 at 14:22:36 CST  (total: 00:00:21)
```

(b) 波形圖(部分)

圖 A：一開始執行 load 的指令，需要花費 108cycle 將資料讀進來

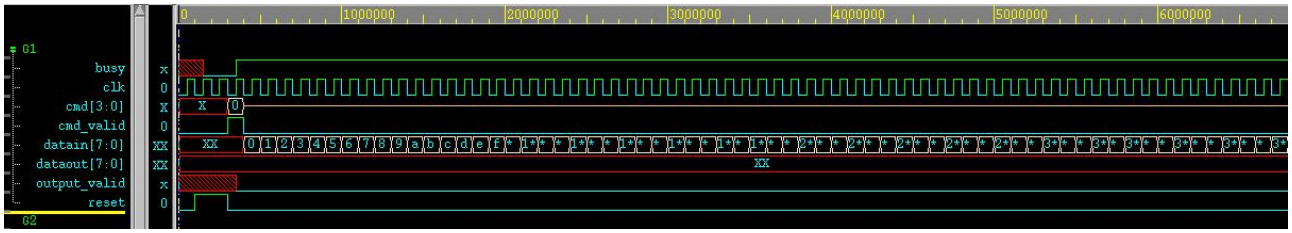


圖 B：執行 load 指令時候，輸出是 zoom out 模式

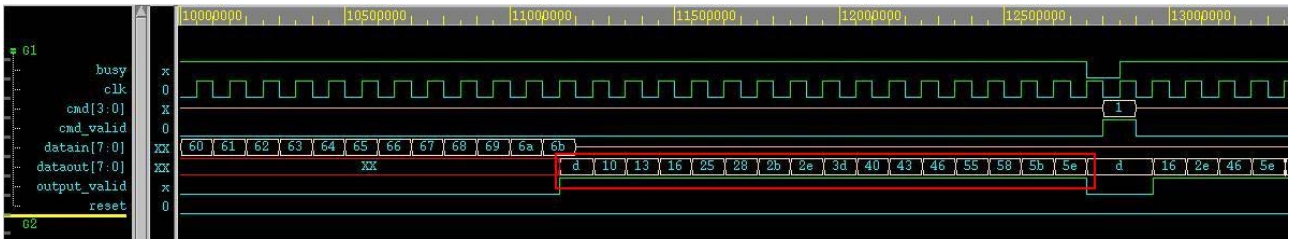


圖 C：正常工作的情況，接受指令之後輸出結果

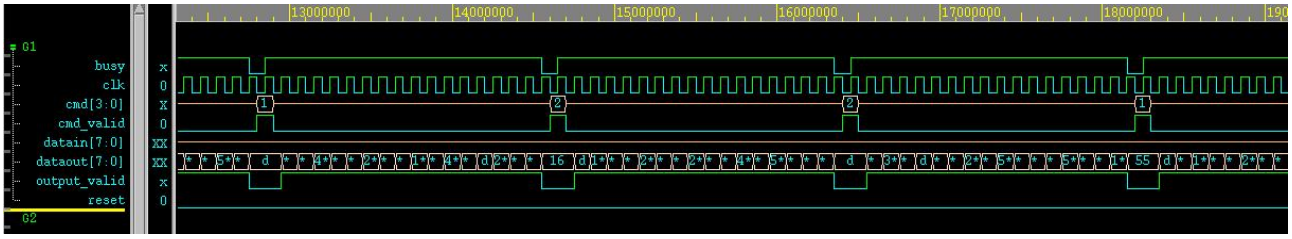


圖 D：在 zoom out 模式接受到 shirt 指令，輸出結果不變