95 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

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**

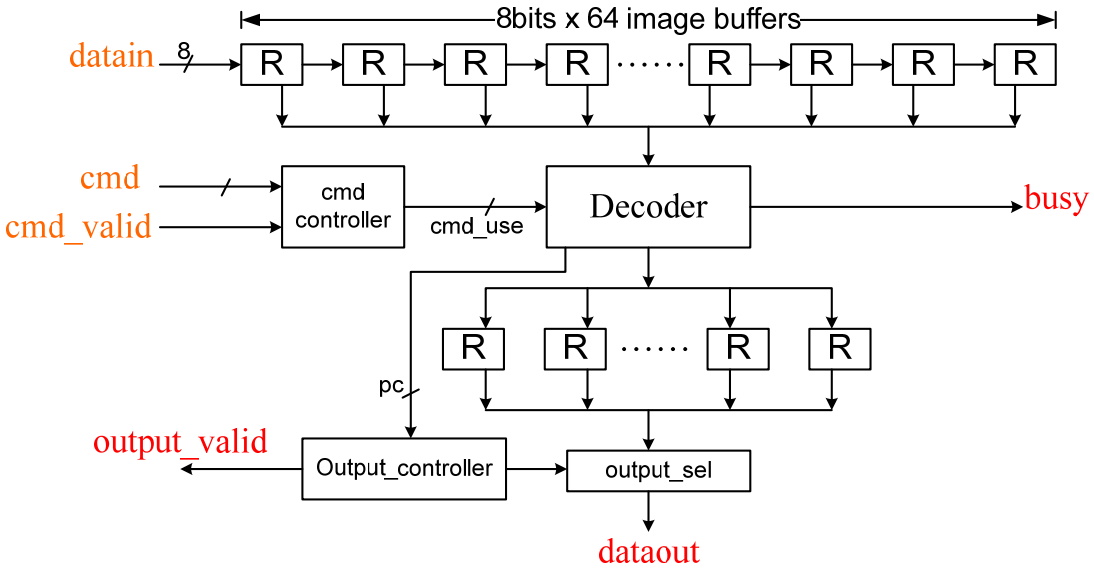
RTL 檔案名稱(RTL Netlist file name): **lcd_ctrl.v**

----------Pre-layout gate-level----------
Gate-Level 檔案名稱: **lcd_ctrl.vmd**
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 的數值再透過 multiplexer 直接選擇所需要的 4x4 image 的 register 輸出。
2、 整體架構如圖一所示，首先 cmd controler 要先判讀指令使否有效的情況下，在執行 cmd 的時候將 cmd 的參數保持在 cmd_use。並且和先前的 cmd 比較分析是否該輸入的 cmd 是否有效。這影響到使用 zoom out 模式的時候，執行 shirt 模式的時候的問題。
3、 透過 decoder 決定指令執行的動作，並且在輸入有效的指令之後，將 busy 設定為 1 並啟動 pc 這個內部參數計數決定何時輸出的 dataout 和輸出數值是否有效。
4、 decoder 的設計上面，只是單純根據題目的輸出要求，將選擇所需要的輸入的影像數值資料到輸出的 output_buf 中。所以在 RTL code 的撰寫上面會有很多 if else 的寫法，為了不產生 latch，所以 code 的長度會很長，但是運算結果卻是很簡單的。
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輸入影像暫存器的結果當作是輸出。



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   [2:0]  cmd;
input         cmd_valid;

output  [7:0]  dataout;
output        output_valid;
output        busy;

`define T_REG_DELAY 0.5

parameter [0:0]  //synopsys enum state_info
        RST = 1'b0,
        CMD_MODE = 1'b1;

reg  current_state,
    next_state;

reg output_valid,
   busy;
reg magnifi,
   zoomout;
reg [2:0] x_addr,
        y_addr,
        cmd_use;
reg [6:0] pc;
reg [7:0] dataout;

reg [7:0] image_buf[0:63];

reg [7:0] output_buf[0:15];

integer i;


always@(posedge clk or posedge reset)
begin
  //synopsys translate_off
  #(`T_REG_DELAY);
    //synopsys translate_on
```

```
     if(reset==1'b1)
  begin
    current_state <= RST;
  end
  else
  begin
    current_state <= next_state;
  end
end

always@(reset or current_state or cmd_valid)
begin
  case(current_state)
    RST:
    begin
      if(cmd_valid == 1'b0)
        next_state = current_state;
      else
        next_state = CMD_MODE;
    end

    CMD_MODE:next_state = current_state;
  endcase
end


always@(posedge clk)
begin
  //synopsys translate_off
  #(`T_REG_DELAY);
     //synopsys translate_on
     case(current_state)
     RST:
     begin
       x_addr <= 3'b0;
       y_addr <= 3'b0;
       pc <= 7'h01;
       busy <= 1'b0;
    cmd_use <= cmd;
       zoomout <= 1'b0;
    dataout <= 8'h0;
       magnifi <= 1'b0;

           for(i=0;i<64;i=i+1)
             image_buf[i] <= 8'b0;

           for(i=0;i<16;i=i+1)
             output_buf[i]<= 8'b0;
  end

  CMD_MODE:
  begin
    if(cmd_valid == 1'b1)
       begin
         if(zoomout == 1'b1 && cmd[2] == 1'b1)
           cmd_use <= 3'b011;
         else if(cmd_use == 3'b011 && cmd[2]==1'b1)
           cmd_use <= 3'b011;
         else
       cmd_use <= cmd;
       end
       else
         cmd_use <= cmd_use;

       if(cmd_use == 3'b011 && cmd[2:1] != 2'b01)
         zoomout <= 1'b1;
       else if(cmd_use == 3'b010)
         zoomout <= 1'b0;
       else
         zoomout <= zoomout;


    if(cmd_use == 3'b001)
    begin
     case(pc)
       7'h42:dataout <= output_buf[0];
       7'h43:dataout <= output_buf[1];
```

```
    7'h44:dataout <= output_buf[2];
    7'h45:dataout <= output_buf[3];
    7'h46:dataout <= output_buf[4];
    7'h47:dataout <= output_buf[5];
    7'h48:dataout <= output_buf[6];
    7'h49:dataout <= output_buf[7];
    7'h4a:dataout <= output_buf[8];
    7'h4b:dataout <= output_buf[9];
    7'h4c:dataout <= output_buf[10];
    7'h4d:dataout <= output_buf[11];
    7'h4e:dataout <= output_buf[12];
    7'h4f:dataout <= output_buf[13];
    7'h50:dataout <= output_buf[14];
    7'h51:dataout <= output_buf[15];
    default:dataout<= dataout;
endcase
end
else
begin
  case(pc)
    7'h03:dataout <= output_buf[0];
    7'h04:dataout <= output_buf[1];
    7'h05:dataout <= output_buf[2];
    7'h06:dataout <= output_buf[3];
    7'h07:dataout <= output_buf[4];
    7'h08:dataout <= output_buf[5];
    7'h09:dataout <= output_buf[6];
    7'h0a:dataout <= output_buf[7];
    7'h0b:dataout <= output_buf[8];
    7'h0c:dataout <= output_buf[9];
    7'h0d:dataout <= output_buf[10];
    7'h0e:dataout <= output_buf[11];
    7'h0f:dataout <= output_buf[12];
    7'h10:dataout <= output_buf[13];
    7'h11:dataout <= output_buf[14];
    7'h12:dataout <= output_buf[15];
    default:dataout<= dataout;
endcase
end


if((cmd_use == 3'b001) && pc <= 7'h51)  //LOADDATA
begin
  if(pc == 7'h51)
  begin
    pc <= 7'b0;
    busy<= 1'b0;
  end
  else
  begin
    pc <= pc + 1'b1;
    busy <= 1'b1;
  end

  if(pc >= 7'h42)
    output_valid <= 1'b1;
  else
    output_valid <= 1'b0;

  x_addr <= x_addr;
  y_addr <= y_addr;

  magnifi <= 1'b0;

  if(pc <= 7'h40 && pc >=7'h01 )
  begin
    image_buf[63] <= datain;
    for(i=62;i>=0;i=i-1)
      image_buf[i] <= image_buf[i+1];
  end
  else
    for(i=0;i<64;i=i+1)
      image_buf[i] <= image_buf[i];

  if(pc >7'h3f)
  begin
    output_buf[0] <= image_buf[0] ;
```

```
    output_buf[1] <= image_buf[2] ;
    output_buf[2] <= image_buf[4] ;
    output_buf[3] <= image_buf[6] ;
    output_buf[4] <= image_buf[16];
    output_buf[5] <= image_buf[18];
    output_buf[6] <= image_buf[20];
    output_buf[7] <= image_buf[22];
    output_buf[8] <= image_buf[32];
    output_buf[9] <= image_buf[34];
    output_buf[10]<= image_buf[36];
    output_buf[11]<= image_buf[38];
    output_buf[12]<= image_buf[48];
    output_buf[13]<= image_buf[50];
    output_buf[14]<= image_buf[52];
    output_buf[15]<= image_buf[54];
  end
  else
  begin
    for(i=0;i<16;i=i+1)
      output_buf[i]<=output_buf[i];
  end
end
else
begin
  for(i=0;i<64;i=i+1)
    image_buf[i]<=image_buf[i];

  if((cmd_use == 3'b000) && pc <= 7'h12)   //ReFlash
  begin
    if(pc == 7'h12)
    begin
      pc <= 7'b0;
      busy <= 1'b0;
    end
    else
    begin
      pc <= pc + 1'b1;
      busy <= 1'b1;
    end

    if(pc >= 7'h03)
      output_valid <= 1'b1;
    else
      output_valid <= 1'b0;
    x_addr <= x_addr;
    y_addr <= y_addr;
    magnifi <= magnifi;
    for(i=0;i<16;i=i+1)
      output_buf[i]<=output_buf[i];

  end
  else if((cmd_use[2:1] == 2'b01) && (pc <= 7'h12))  //ZOOM IN / ZOOM OUT
  begin
    if(pc == 7'h12)
    begin
      busy <= 1'b0;
      pc <= 7'b0;
    end
    else
    begin
      busy <= 1'b1;
      pc <= pc + 1'b1;
    end
    if( pc >= 7'h03)
      output_valid <= 1'b1;
    else
      output_valid <= 1'b0;
    magnifi <= ~cmd_use[0];
    if(magnifi == 1'b0)
    begin
      x_addr <= 3'b0;
      y_addr <= 3'b0;
      output_buf[0] <= image_buf[0] ;
      output_buf[1] <= image_buf[2] ;
      output_buf[2] <= image_buf[4] ;
      output_buf[3] <= image_buf[6] ;
      output_buf[4] <= image_buf[16];
```

```
      output_buf[5] <= image_buf[18];
      output_buf[6] <= image_buf[20];
      output_buf[7] <= image_buf[22];
      output_buf[8] <= image_buf[32];
      output_buf[9] <= image_buf[34];
      output_buf[10]<= image_buf[36];
      output_buf[11]<= image_buf[38];
      output_buf[12]<= image_buf[48];
      output_buf[13]<= image_buf[50];
      output_buf[14]<= image_buf[52];
      output_buf[15]<= image_buf[54];
    end
    else
    begin
      x_addr <= 3'b010;
      y_addr <= 3'b010;
      output_buf[0] <= image_buf[18];
      output_buf[1] <= image_buf[19];
      output_buf[2] <= image_buf[20];
      output_buf[3] <= image_buf[21];
      output_buf[4] <= image_buf[26];
      output_buf[5] <= image_buf[27];
      output_buf[6] <= image_buf[28];
      output_buf[7] <= image_buf[29];
      output_buf[8] <= image_buf[34];
      output_buf[9] <= image_buf[35];
      output_buf[10]<= image_buf[36];
      output_buf[11]<= image_buf[37];
      output_buf[12]<= image_buf[42];
      output_buf[13]<= image_buf[43];
      output_buf[14]<= image_buf[44];
      output_buf[15]<= image_buf[45];
    end
  end
  else              //SHIRFT
  begin
    if(pc == 7'h12)
    begin
      busy <= 1'b0;
      pc <= 7'b0;
    end
    else
    begin
      busy <= 1'b1;
      pc <= pc + 1'b1;
    end

    if(pc >= 7'h03)
      output_valid <= 1'b1;
    else
      output_valid <= 1'b0;
    magnifi <= magnifi;
    if(pc == 7'h1 )
    begin
      if(cmd_use[1:0] == 2'b00)
      begin
        if(x_addr == 3'b100)
          x_addr <= x_addr;
        else
          x_addr <= x_addr + 1'b1;
        y_addr <= y_addr;
      end
      else if(cmd_use[1:0] == 2'b01)
      begin
        if(x_addr == 2'b00)
          x_addr <= x_addr;
        else
          x_addr <= x_addr - 1'b1;
        y_addr <= y_addr;
      end
      else if(cmd_use[1:0] == 2'b10)
      begin
        if(y_addr == 2'b00)
          y_addr <= y_addr;
        else
          y_addr <= y_addr - 1'b1;
        x_addr <= x_addr;
```

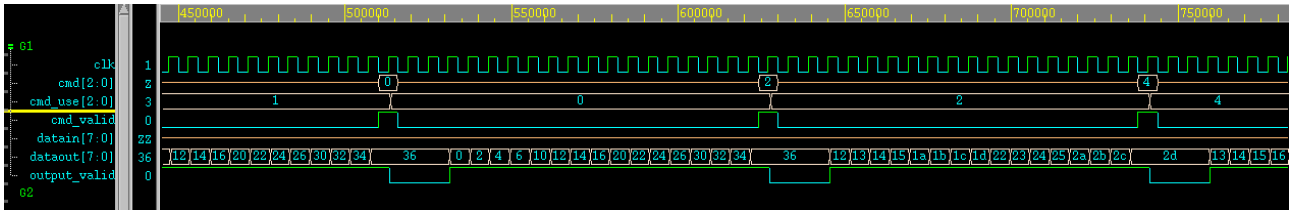
```
        end
        else
        begin
          if(y_addr == 3'b100)
            y_addr <= y_addr;
          else
            y_addr <= y_addr + 1'b1;
          x_addr <= x_addr;
        end
      end
      else
      begin
        x_addr <= x_addr;
        y_addr <= y_addr;
      end

      output_buf[0] <= image_buf[8*y_addr+x_addr];
      output_buf[1] <= image_buf[8*y_addr+x_addr+1];
      output_buf[2] <= image_buf[8*y_addr+x_addr+2];
      output_buf[3] <= image_buf[8*y_addr+x_addr+3];
      output_buf[4] <= image_buf[8*(y_addr+1)+x_addr];
      output_buf[5] <= image_buf[8*(y_addr+1)+x_addr+1];
      output_buf[6] <= image_buf[8*(y_addr+1)+x_addr+2];
      output_buf[7] <= image_buf[8*(y_addr+1)+x_addr+3];
      output_buf[8] <= image_buf[8*(y_addr+2)+x_addr];
      output_buf[9] <= image_buf[8*(y_addr+2)+x_addr+1];
      output_buf[10]<= image_buf[8*(y_addr+2)+x_addr+2];
      output_buf[11]<= image_buf[8*(y_addr+2)+x_addr+3];
      output_buf[12]<= image_buf[8*(y_addr+3)+x_addr];
      output_buf[13]<= image_buf[8*(y_addr+3)+x_addr+1];
      output_buf[14]<= image_buf[8*(y_addr+3)+x_addr+2];
      output_buf[15]<= image_buf[8*(y_addr+3)+x_addr+3];
    end
   end
  end

 endcase
end

endmodule
```

三、 模擬結果 (Gate-level)

(a) 模擬之 log 檔

```
cverilog: 05.50-p004: (c) Copyright 1995-2005 Cadence Design Systems, Inc.
TOOL:     ncverilog 05.50-p004: Started on Mar 29, 2007 at 14:16:55 CST
ncverilog
     -f vlog_gate.f
          ./testfixture.v
          ./lcd_ctrl.vmd
          ./lib/tsmc13_neg.v
          +access+r
Recompiling... reason: file './lcd_ctrl.sdf' is newer than expected.
     expected: Thu Mar 29 14:04:52 2007
     actual:   Thu Mar 29 14:13:27 2007
file: ./testfixture.v
     module worklib.test:v
          errors: 0, warnings: 0
file: ./lcd_ctrl.vmd
     module worklib.LCD_CTRL:vmd
          errors: 0, warnings: 0
file: ./lib/tsmc13_neg.v
          Caching library 'worklib' ....... Done
     Elaborating the design hierarchy:
   ...
Loading snapshot worklib.test:v ..................... Done
ncsim> source /usr/cad/cadence/IUS/cur/tools/inca/files/ncsimrc
ncsim> run


--------------------------------------------

All data have been generated successfully!

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

--------------------------------------------

Simulation complete via $finish(1) at time 6288 NS + 0
./testfixture.v:131      $finish;
ncsim> exit
TOOL:     ncverilog 05.50-p004: Exiting on Mar 29, 2007 at 14:17:12 CST  (total: 00:00:17)
```

(b) 波形圖(部分)


450000
500000
550000
600000
650000
700000
750000
G1
clk
cmd[2:0]
cmd_use[2:0]
cmd_valid
datain[7:0]
dataout[7:0]
output_valid
G2