实验报告

实验题目：寄存器堆与计数器　　日期：2019 年 4 月 12 日

姓名:廖洲洲　　学号:PB17081504　　成绩:_____

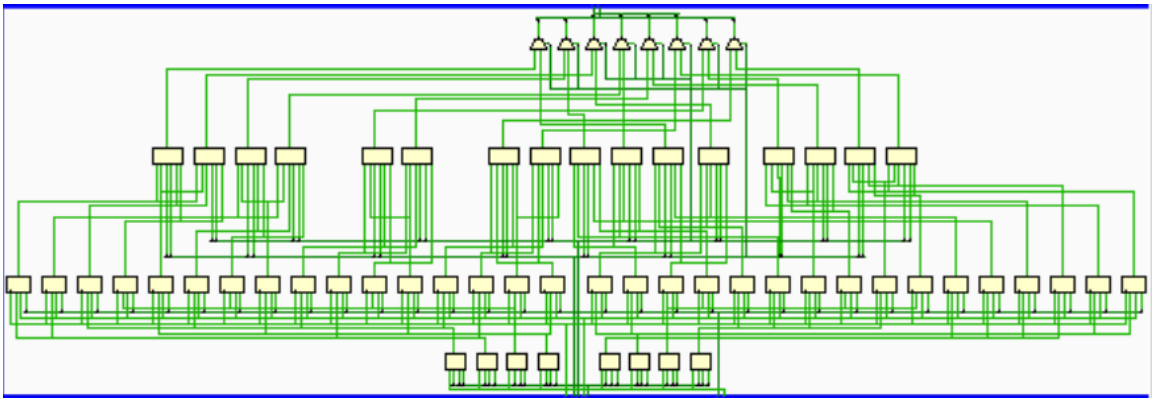
实验目的:1.设计一个寄存器堆

2.利用寄存器堆和适当逻辑实现最大长度为 8 的循环队列

1.寄存器堆

逻辑设计：　:为了能让该寄存器堆适用于 fifo 模块，设计的寄存器堆含 8 个寄存器，每个寄存器数据位为 4 位。该寄存器在写使能 we=1 时进行写操作，同时在该寄存器设计两个异步读端口。

原理图：

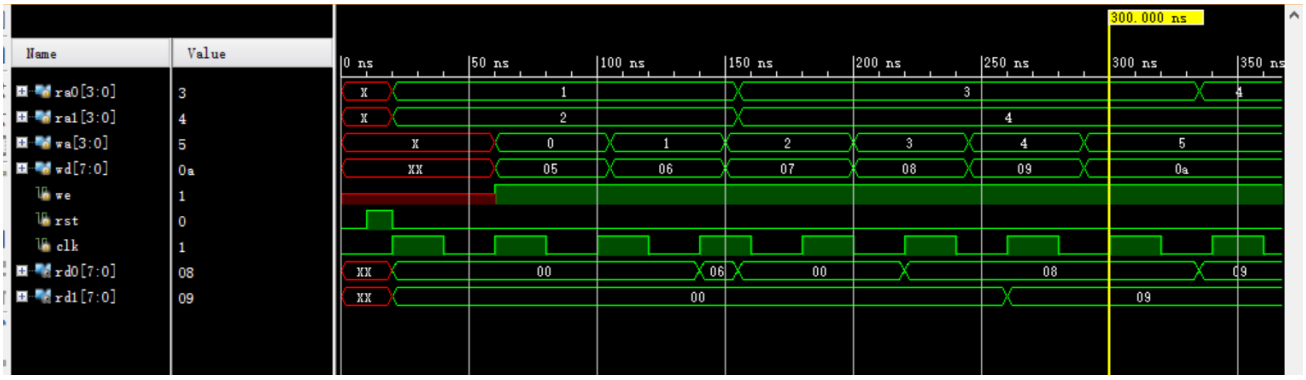


核心代码：

```
always@(posedge rst,posedge clk) begin
    if(rst) begin
        for(i=0;i<8;i=i+1)
            reg_file[i] <= 0;
            end
    else if(we == 1) begin
        reg_file[wa] <= wd;
        end
end

always@(*) begin
   rd0 <= reg_file[ra0];
    rd1 <= reg_file[ra1];
end
```

仿真图：



下载照片：

给 0 号写了 3 读出了 3

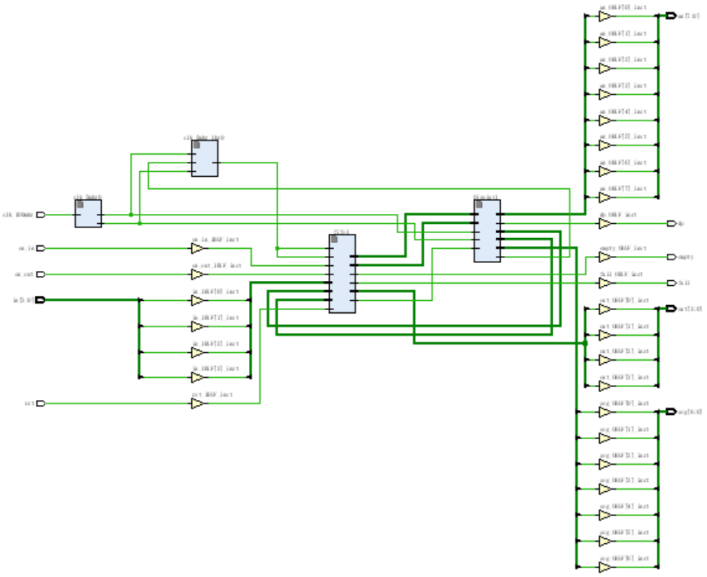
给 1 号写 2 读出了 2

2. FIFO

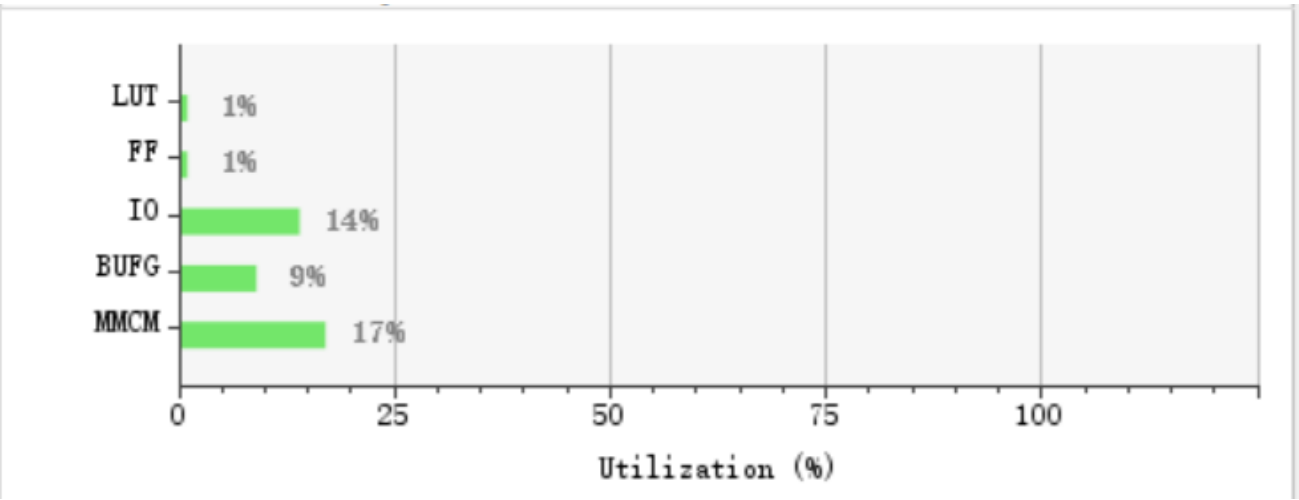
设计逻辑：设计了一个头指针 sel_out，一个尾指针 sel_in。当进使能 en_in=1 且队列不满时时，尾指针加 1，将要放进队列的数 in 放入寄存器堆中地址 sel_in 的寄存器中，标志寄存器堆中地址为 sel_in 的寄存器为 1（即该寄存器含值）。当出使能 en_out=1 且队列不为空时，头指针加 1，标志寄存器堆中地址为 sel_out 的寄存器为 0（即该寄存器逻辑上不含值，也就是之后不会在七段数码管上显示）。由于设计了 flag 标志寄存器是否存值，则当 8 个寄存器都被标志 1 时，说明队列满，给 full 赋 1，当 8 个寄存器都被标志 0 时，说明寄存器为空，给 empty 赋 0。

最后使用 Display 模块将被标志了 1 的寄存器的值在相应的数码管显示出来，被标志为 0 的寄存器对应的数码管则不显示。同时将头指针 sel_out 引出，将其对应的数码管的点点亮。

原理图：



资源使用：



核心代码：

//fifo

```
    assign full=flag[7]&flag[6]&flag[5]&flag[4]&flag[3]&flag[2]&flag[1]&flag[0];
    assign empty=~flag[7]&~flag[6]&~flag[5]&~flag[4]&~flag[3]&~flag[2]&~flag[1]&~flag[0];
    assign clk2=~clk;
    regfile
regfile1(.ra0(ra0),.wa(wa),.wd(wd),.we(we),.clk(clk2),.rst(rst),.rd0(out),.display(display))
;
    always@(posedge rst,posedge clk) begin
        we =0;
```

```
        if(rst) begin
           ra0 <= 0;
           flag <= 8'b00000000;
           sel_out <= 0;
           sel_in <= 0;
//           num <= 0;
           end
        else if (en_out & empty == 0) begin //en_out 相当于头指针
            flag[sel_out] = 0;
            ra0 =sel_out;
            sel_out = (sel_out +1) % 8;//如果队列不空，一个数出队列，则指针加一
     //       num = num-1;
            end
        else if (en_in & full ==0) begin  //en_in 相当于尾指针
            we = 1;
        //    flag[sel_in]=1;
            wa <= sel_in;
            wd <= in;
            flag[sel_in]=1;
            sel_in = (sel_in +1) % 8;
    //        num = num +1;
            end   //如果队列不满，一个数入队列，指针数加一
        end
```

//display

```
    seg_ctrl seg_ctrl(
        .x              (seg_din),
        .sel            (seg_sel),
        .an             (an),
        .seg            (seg),
        .sel_out        (sel_out),
        .flag           (flag),
        .num          (num),
        .dp             (dp)
            );
```

```
  reg [22:0] cnt;
  always@(posedge clk5mhz or negedge locked)
  begin
     if(~locked)
         cnt <= 23'h0;
     else if(cnt<23'd5000000)
         cnt <= cnt + 1;
     else
         cnt <= 23'h0;
  end

  always@(posedge clk5mhz or negedge locked)
  begin
//    flag_reg=flag;
   if(~locked)
  begin
      seg_sel <= 8'b11111111;
      seg_din <= 4'b0;
  end
  else case(cnt[14:12])
      3'b000:
      begin
          num<=0;
          seg_sel <= 8'b1111_1110;
    //      seg_din <= {flag[0],display[3:0]};
          seg_din <= display[3:0];
      end
      3'b001:
      begin
      num<=1;
          seg_sel <= 8'b1111_1101;
  //        seg_din <= {flag[1],display[7:4]};
  seg_din <= display[7:4];
      end
      3'b010:
```

```
begin
num<=2;
    seg_sel <= 8'b1111_1011;
   // seg_din <= {flag[2],display[11:8]};
   seg_din <= display[11:8];
end
3'b011:
begin
num<=3;
    seg_sel <= 8'b1111_0111;
 //   seg_din <= {flag[3],display[15:12]};
 seg_din <= display[15:12];
end
3'b100:
begin
num<=4;
    seg_sel <= 8'b1110_1111;
  //  seg_din <= {flag[4],display[19:16]};
  seg_din <= display[19:16];
end
 3'b101:
 begin
 num<=5;
    seg_sel <= 8'b1101_1111;
//     seg_din <= {flag[5],display[23:20]};
  seg_din <= display[23:20];
  end
  3'b110:
  begin
  num<=6;
    seg_sel <= 8'b1011_1111;
   //  seg_din <= {flag[6],display[27:24]};
  seg_din <= display[27:24];
  end
 default
 begin
```

```
        num<=7;
            seg_sel <= 8'b0111_1111;
          //  seg_din <= {flag[7],display[31:28]};
        seg_din <= display[31:28];
        end
        endcase
end
```

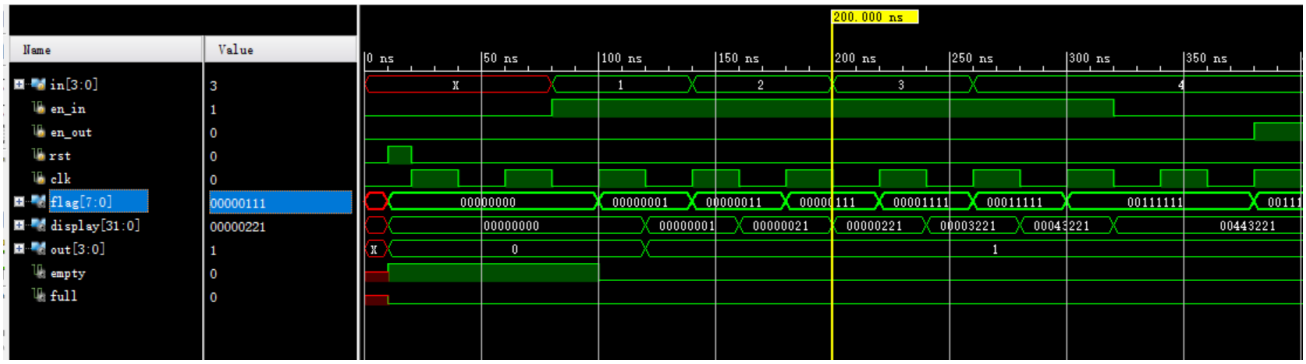
//seg_ctrl

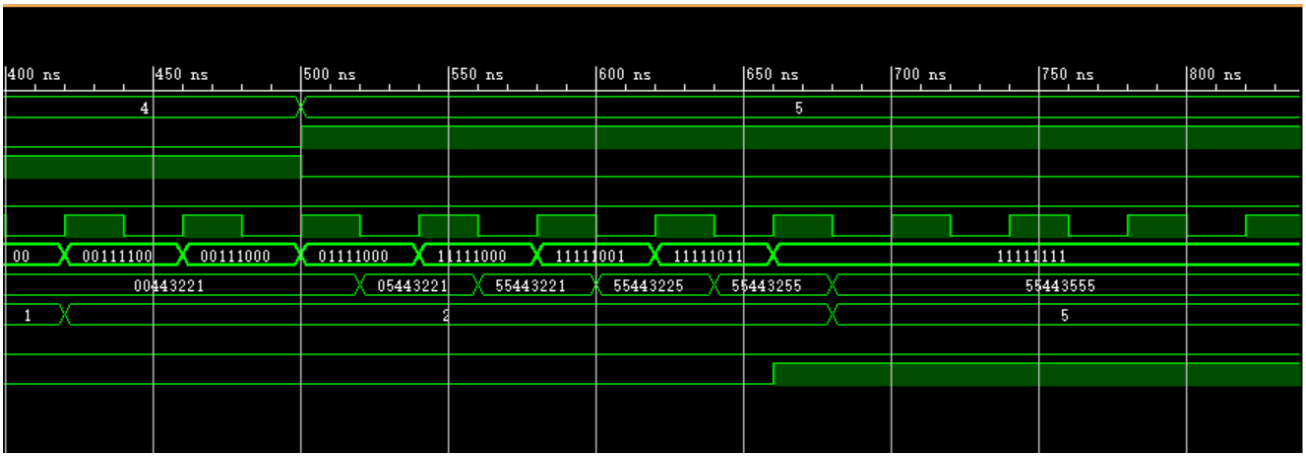
```
   always @(*)
    begin
 //       if(x[4]==0)  seg=8'b11111111;
//        else
        if(num == sel_out)
            dp =0;
        else
            dp = 1;

        if(flag[num] ==0)
            seg = 8'b11111111;
        else
        case(x)
        4'b0000:seg={7'b1000000};//0
        4'b0001:seg={7'b1111001};//1
        4'b0010:seg={7'b0100100};//2
        4'b0011:seg={7'b0110000};//3
        4'b0100:seg={7'b0011001};//4
        4'b0101:seg={7'b0010010};//5
        4'b0110:seg={7'b0000010};//6
        4'b0111:seg={7'b1111000};//7
        4'b1000:seg={7'b0000000};//8
        4'b1001:seg={7'b0010000};//9
        4'b1010:seg={7'b0001000};//a
        4'b1011:seg={7'b0000011};//b
        4'b1100:seg={7'b1000110};//c
        4'b1101:seg={7'b0100001};//d
```

```
4'b1110:seg={7'b0000110};//e
4'b1111:seg={7'b0001110};//f
endcase
end
```

仿真截图：





从仿真截图可以看出，队列依次进了 1 2 2 3 4 4 然后又依次出了 1 2 2，最后再进了 5 个 5，队列满，队列为 55534455

下载照片：

进 1

进 2

进 3

进 4

进满，队列为满

出 1

出 2

出 3 出 4

全出，队列为空

实验总结：

1. 通过本次实验我设计完成了一个寄存器堆，设计的时候较为顺利，仿真的时候发现了一点点问题，发现有时候给寄存器堆写数据的时候写不进去，通过仔细观察后，发现寄存器堆的设计并没有问题，问题原来时出再仿真的设计上，由于将写入数据的间隔时间设计的过小，导致在一个时钟周期内，要写入的数据就发生了变化，于是在时钟的上升沿并没有读取到数据。
2. 在 FIFO 的设计中,刚开始使用 num 记录队列中元素的个数,用于判断空与满，但在下载时出现了错误，故改为使用标志位的逻辑运算来判断空和满。
3. 在设计 7 段数码管的显示时特别容易出现 bug，需要仔细考虑。

源代码：

```
module display_fifo(
    input clk_100mhz,
    input rst,
    input [3:0] in,
    input en_in,en_out,
    output [3:0] out,
    output empty,full,
    output [7:0] an,
    output [6:0] seg,
    output dp
    );

    wire clk_5mhz;
```

```
    wire locked;//将100mhz的时钟转换成5mhz
    clk_wiz_0 clk_5mhz0(.clk_in1(clk_100mhz),.locked(locked),.clk_out1(clk_5mhz));

    wire clk_1hz;//将5mhz时钟转换为1hz
    clk_5mhz_1hz clk_5mhz_1hz0 (clk_5mhz,locked,clk_1hz);

    reg clk;
    wire [7:0] flag;
    wire [2:0] sel_out;
    wire  [31:0] display;//fifo主要模块，它将主要数据以二进制显示出来
    fifo fifo1(in,en_in,en_out,rst,clk_1hz,flag,display,out,empty,full,sel_out);
  //  wire [7:0] seg1;
    //下面是数码管显示模块,将flag与display在数码管上显示
    Display Display1(clk_5mhz,locked,display,flag,sel_out,an,seg,dp);



endmodule

module clk_5mhz_1hz(
    input clk5mhz,
    input locked,
    output clk_1hz
    );

    reg Q;
    reg [23:0] count;
    assign clk_1hz=Q;

       initial
           begin
           count = 0;
           Q = 0;
           end

       always @(posedge clk5mhz or  negedge locked)
```

```
                begin
                    if(~locked)
                       count <= 0;
                    else if(count==2499999)
                        begin
                        count <= 0;
                        Q <= ~Q;
                        end
                    else
                        count <= count + 1;
                 end
endmodule

module fifo(
    input [3:0] in,
    input en_in,
    input en_out,
    input rst,
    input clk,
    output reg [7:0] flag,
    output [31:0] display,
    output [3:0] out,
    output  empty,
    output  full,
    output reg [2:0] sel_out
    );
    reg [2:0] sel_in,ra0,wa;
  //  reg [3:0] num;
//    wire [31:0] display;
    reg [3:0] wd;
    reg we;
    assign full=flag[7]&flag[6]&flag[5]&flag[4]&flag[3]&flag[2]&flag[1]&flag[0];
    assign
empty=~flag[7]&~flag[6]&~flag[5]&~flag[4]&~flag[3]&~flag[2]&~flag[1]&~flag[0];
    assign clk2=~clk;
    regfile
```

```
regfile1(.ra0(ra0),.wa(wa),.wd(wd),.we(we),.clk(clk2),.rst(rst),.rd0(out),.display(display));
    always@(posedge rst,posedge clk) begin
        we =0;
        if(rst) begin
           ra0 <= 0;
           flag <= 8'b00000000;
           sel_out <= 0;
           sel_in <= 0;
//           num <= 0;
           end
        else if (en_out & empty == 0) begin //en_out 相当于头指针
            flag[sel_out] = 0;
            ra0 =sel_out;
            sel_out = (sel_out +1) % 8;//如果队列不空，一个数出队列，则指针加一
     //       num = num-1;
            end
        else if (en_in & full ==0) begin  //en_in 相当于尾指针
            we = 1;
        //    flag[sel_in]=1;
            wa <= sel_in;
            wd <= in;
            flag[sel_in]=1;
            sel_in = (sel_in +1) % 8;
    //        num = num +1;
            end   //如果队列不满，一个数入队列，指针数加一
        end
 //   always@(*) begin
 //       if(num == 8) begin
 //           full <= 1;
  //          empty <= 0;
  //          end
   //     else if(num==0) begin
   //          full <= 0;
    //         empty <= 1;
   //          end
```

```
  //      else begin
 //           full <= 0;
  //          empty <= 0;
   //         end
    //    end
endmodule

module regfile (
    input [2:0] ra0,
    input [2:0] ra1,
    input [2:0] wa,
    input [3:0] wd,
    input we,
    input rst,
    input clk,
    output reg [3:0] rd0,
    output reg [3:0] rd1,
    output [31:0] display
    );
    reg [3:0] reg_file[7:0];
    integer i;
assign
display={reg_file[7],reg_file[6],reg_file[5],reg_file[4],reg_file[3],reg_file[2],reg_fi
le[1],
                    reg_file[0]};
    always@(posedge rst,posedge clk) begin
        if(rst) begin
            for(i=0;i<8;i=i+1)
                reg_file[i] <= 0;
                end
        else if(we == 1) begin
            reg_file[wa] <= wd;
            end
    end

    always@(*) begin
```

```
        rd0 <= reg_file[ra0];
         rd1 <= reg_file[ra1];   //仿真时发现如果写成 always@(*) ,当 ra 未变时，如果对应的
reg_file 的值变化了，其未该改变
        end                     //  因此改为线连出
//    assign rd0 = reg_file[ra0];
//    assign rd1 = reg_file[ra1];
endmodule

module Display(
    input clk5mhz,
    input locked,
    input [31:0] display,
    input [7:0] flag,
    input [2:0] sel_out,
    output [7:0] an,
    output [6:0] seg,
    output dp
    );

    reg [7:0] seg_sel;
    reg [3:0] seg_din;
    reg [3:0] num;
//    reg [7:0] flag_reg;
    seg_ctrl seg_ctrl(
        .x              (seg_din),
        .sel            (seg_sel),
        .an             (an),
        .seg            (seg),
        .sel_out        (sel_out),
        .flag           (flag),
        .num          (num),
        .dp           (dp)
            );


    reg [22:0] cnt;
```

```
  always@(posedge clk5mhz or negedge locked)
  begin
     if(~locked)
         cnt <= 23'h0;
     else if(cnt<23'd5000000)
         cnt <= cnt + 1;
     else
         cnt <= 23'h0;
  end

  always@(posedge clk5mhz or negedge locked)
  begin
//    flag_reg=flag;
   if(~locked)
  begin
      seg_sel <= 8'b11111111;
      seg_din <= 4'b0;
  end
  else case(cnt[14:12])
      3'b000:
      begin
          num<=0;
          seg_sel <= 8'b1111_1110;
    //      seg_din <= {flag[0],display[3:0]};
          seg_din <= display[3:0];
      end
      3'b001:
      begin
      num<=1;
          seg_sel <= 8'b1111_1101;
  //        seg_din <= {flag[1],display[7:4]};
  seg_din <= display[7:4];
      end
      3'b010:
      begin
      num<=2;
```

```
    seg_sel <= 8'b1111_1011;
   // seg_din <= {flag[2],display[11:8]};
   seg_din <= display[11:8];
end
3'b011:
begin
num<=3;
    seg_sel <= 8'b1111_0111;
 //   seg_din <= {flag[3],display[15:12]};
 seg_din <= display[15:12];
end
3'b100:
begin
num<=4;
    seg_sel <= 8'b1110_1111;
  //  seg_din <= {flag[4],display[19:16]};
  seg_din <= display[19:16];
end
 3'b101:
 begin
 num<=5;
    seg_sel <= 8'b1101_1111;
//     seg_din <= {flag[5],display[23:20]};
  seg_din <= display[23:20];
  end
  3'b110:
  begin
  num<=6;
    seg_sel <= 8'b1011_1111;
   //  seg_din <= {flag[6],display[27:24]};
  seg_din <= display[27:24];
  end
default
begin
num<=7;
    seg_sel <= 8'b0111_1111;
```

```
          //  seg_din <= {flag[7],display[31:28]};
        seg_din <= display[31:28];
        end
        endcase
end


endmodule

module seg_ctrl(
    input [3:0] x,
    input [7:0] sel,
    input [2:0] sel_out,
    input [7:0] flag,
    input [3:0] num,
    output [7:0] an,
    output reg [6:0] seg,
    output reg dp
    );
    assign an=sel;

 //   reg spot;
 //   always@(*) begin
     //   if(sel[sel_out] == 0)
         //   spot =0;
      //  else
   //         spot =1;
     //       end

    always @(*)
    begin
 //       if(x[4]==0)  seg=8'b11111111;
//        else
        if(num == sel_out)
            dp =0;
        else
```

```
            dp = 1;

        if(flag[num] ==0)
            seg = 8'b11111111;
        else
        case(x)
        4'b0000:seg={7'b1000000};//0
        4'b0001:seg={7'b1111001};//1
        4'b0010:seg={7'b0100100};//2
        4'b0011:seg={7'b0110000};//3
        4'b0100:seg={7'b0011001};//4
        4'b0101:seg={7'b0010010};//5
        4'b0110:seg={7'b0000010};//6
        4'b0111:seg={7'b1111000};//7
        4'b1000:seg={7'b0000000};//8
        4'b1001:seg={7'b0010000};//9
        4'b1010:seg={7'b0001000};//a
        4'b1011:seg={7'b0000011};//b
        4'b1100:seg={7'b1000110};//c
        4'b1101:seg={7'b0100001};//d
        4'b1110:seg={7'b0000110};//e
        4'b1111:seg={7'b0001110};//f
        endcase
        end

endmodule

module fifo_tb_1(

    );
     reg [3:0] in;
     reg en_in,en_out,rst,clk;
     wire [7:0] flag;
     wire  [31:0] display;
     wire  [3:0] out;
     wire empty;
```

```
     wire full;
     wire [2:0] sel_out;
     fifo fifo1(in,en_in,en_out,rst,clk,flag,display,out,empty,full,sel_out);
     initial
        begin
           clk=0;
        forever
           #20 clk=~clk;
        end

    initial
        begin
        en_in=0;en_out=0;
        rst=0;
        #10 rst=1;
        #10 rst=0;
        #60 en_in=1;in=1;
        #60 in=2;
        #60 in=3;
        #60 in=4;
        #60 en_in=0;
        #60 en_out=1;
        #60 en_out=1;
        #60 en_out=0;en_in=1;in=5;
        end
endmodule

module regfile_tb(

    );

    reg [3:0] ra0;
    reg [3:0] ra1;
    reg [3:0] wa;
    reg [7:0] wd;
    reg we;
```

```
    reg rst;
    reg clk;
    wire [7:0] rd0;
    wire [7:0] rd1;

    regfile regfile(ra0,ra1,wa,wd,we,rst,clk,rd0,rd1);

    initial
        begin
           clk=0;
        forever
           #20 clk=~clk;
        end

    initial
        begin
        rst=0;
        #10 rst=1;
        #10 rst=0;ra0=1;ra1=2;
        #40 we=1;wa=0;wd=5;
        #45 wa=1;wd=6;
        #45 wa=2;wd=7;
        #5 ra0=3;ra1=4;
        #45 wa=3;wd=8;
        #45 wa=4;wd=9;
        #45 wa=5;wd=10;
        #45  ra0=4;
        #45  ra0=3;
        end
endmodule
```