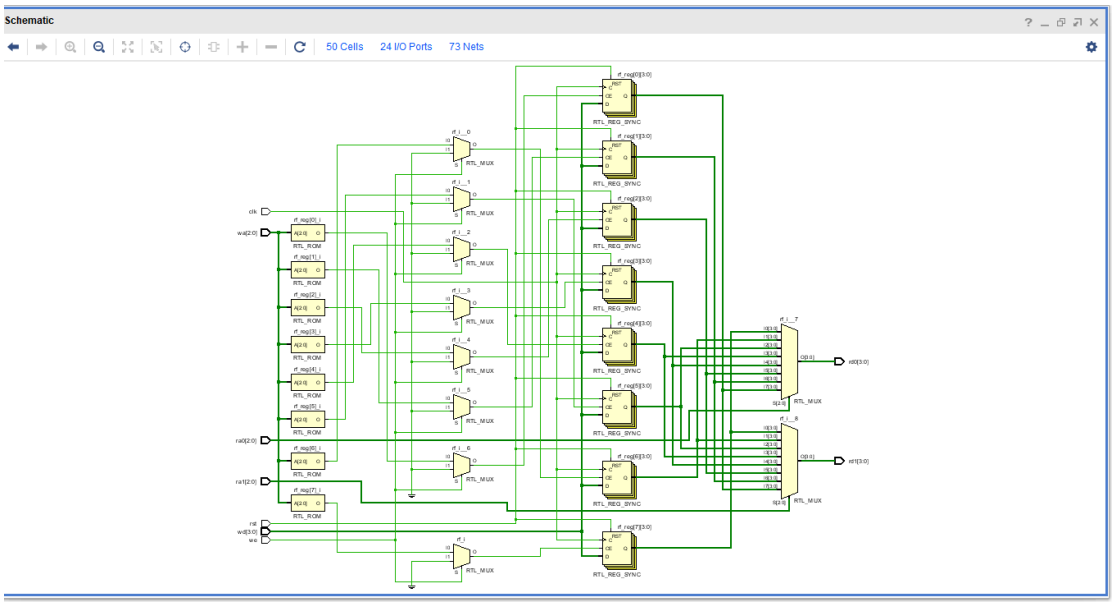
Lab3 寄存器堆与计数器

1. 设计逻辑

(1) 寄存器堆

设计 8*4 的寄存器堆，8 个四位寄存器，wa 为写入地址，wd 为写入数据，ra0 ra1 为读出地址，rd0 rd1 为读出数据，we 为写入使能，we=1 时可以写入，时钟上沿时输入，rst 将寄存器中数据清零。



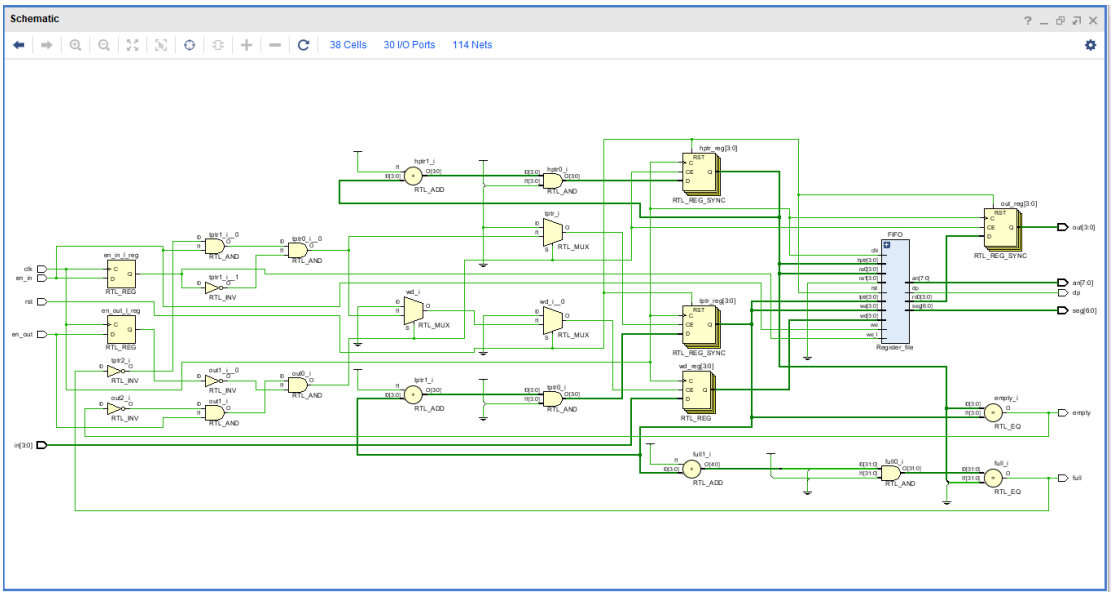
数据通路

(2) FIFO 循环队列

设计长度为 8 的 FIFO 循环队列，使用头指针 hptr 和尾指针 tptr 标记队头和队尾元素，当队空时，hptr=tptr，当队满时，hptr=（tptr+1）%8.由于时钟分频只用来控制数码管，于是使用两个变量 en_in_l 和 en_out_l 来表示上一个时钟周期时 en_in 和 en_out 的值，当队列没有满且 en_in=1,en_in_l=0 时，元素入队，当队列没有空且 en_out=1，en_out_l=0 时出队。rst 清空寄存器堆中值。同时用 hptr 和 tptr 的相对位置来判断 8 个七段数码管哪一个亮。即头指针尾指针中间的元素让七段数码管亮。把

头尾指针扩展成 4 位实现循环队列，写入时，hptr+1，且 hptr=hptr%8，

读出时，tptr+1，且 tptr=tptr%8.



数据通路。

2. 核心代码

```
module Register_file(
    input clk,
    input rst,
    input we,
    input [2:0]ra0,  //m=3  寄存器数量=8
    input [2:0]ra1,
    input [2:0]wa,
    input [3:0]wd,  //n=4 位数
    output [3:0]rd0,
    output [3:0]rd1
    );
    reg [3:0]rf[7:0];
    integer i;

    assign rd0=rf[ra0];
    assign rd1=rf[ra1];

    always @(posedge clk)
    begin
        if(rst)
        begin
            for(i=0;i<8;i=i+1)
                rf[i]<=4'b0;
        end
        else if(we)
        begin
            rf[wa]<=wd;
        end
    end
endmodule
```

寄存器堆，使用二维数组，rf[wa]<=wd 将值写入寄存器堆

```
assign empty=(hptr==tptr);
assign full=((tptr+1)%8==hptr);

always @(posedge clk)  //输出时序
begin
    en_in_l<=en_in;
    en_out_l<=en_out;
    if(rst)
    begin
        out<=0;
        hptr=0;
        tptr=0;
    end
    else if(!empty && en_out && !en_out_l)
    begin
        out<=rd0;
        hptr=(hptr+1)%8;  //队首读出
    end
    else if(!full && en_in && !en_in_l)
    begin
        wd<=in;
        tptr=(tptr+1)%8;  //队尾写入
    end
end

Register_file FIFO(.clk(clk),.we(en_in),.we_l(en_in_l),.rst(rst),.ra0(hptr),.hptr(hptr),.tptr(tptr),.wa(tptr),.wd(wd),.rd0(rd0),.seg(seg),.an(an),.dp(dp));
```

FIFO 队列逻辑，rst 时将 out 和头尾指针清零，并在相应条件下写入和读出数据

```
reg [3:0]seg_in;
reg [7:0]seg_sel;
seg_ctrl seg_ctrl(seg_in,seg_sel,seg,an);
reg [23:0]cnt;
always @(posedge clk)
begin
    if (cnt==24'd4999999)
    begin
        cnt=0;
        clk1=~clk1;
    end
    else
    begin
        cnt=cnt+1;
        dp=1;
        if(rst)
        begin
            seg_sel<=8'b01111111;
            dp=0;
            seg_in<=4'b1111;
        end
        else
        begin
            case(cnt[15:13])
```

对 100MHz 时钟分频以满足七段数码管使用频率，并在 rst 时使得最左端的小数点亮

```
3'b000:
if ((hptr<=tptr & (7<hptr | 7>=tptr))|(hptr>tptr &(tptr<=7 & 7<hptr))) seg_sel<=8'b11111111;else
begin
    if (hptr==7) dp=0;
    seg_sel<=8'b11111110;
    seg_in<=register[7];
end
3'b001:
if ((hptr<=tptr & (6<hptr | 6>=tptr))|(hptr>tptr &(tptr<=6 & 6<hptr))) seg_sel<=8'b11111111;else
begin
    if (hptr==6) dp=0;
    seg_sel<=8'b11111101;
    seg_in<=register[6];
end
3'b010:
if ((hptr<=tptr & (5<hptr | 5>=tptr))|(hptr>tptr &(tptr<=5 & 5<hptr))) seg_sel<=8'b11111111;else
begin
    if (hptr==5) dp=0;
    seg_sel<=8'b11111011;
    seg_in<=register[5];
end
3'b011:
if ((hptr<=tptr & (4<hptr | 4>=tptr))|(hptr>tptr &(tptr<=4 & 4<hptr))) seg_sel<=8'b11111111;else
begin
    if (hptr==4) dp=0;
    seg_sel<=8'b11110111;
    seg_in<=register[4];
end

  3'b101:
  if ((hptr<=tptr & (2<hptr | 2>=tptr))|(hptr>tptr &(tptr<=2 & 2<hptr))) seg_sel<=8'b11111111;else
  begin
      if (hptr==2) dp=0;
      seg_sel<=8'b11011111;
      seg_in<=register[2];
  end
  3'b110:
  if ((hptr<=tptr & (1<hptr | 1>=tptr))|(hptr>tptr &(tptr<=1 & 1<hptr))) seg_sel<=8'b11111111;else
  begin
      if (hptr==1) dp=0;
      seg_sel<=8'b10111111;
      seg_in<=register[1];
  end
  3'b111:
  if ((hptr<=tptr & (0<hptr | 0>=tptr))|(hptr>tptr &(tptr<=0 & 0<hptr))) seg_sel<=8'b11111111;else
  begin
      if (hptr==0) dp=0;
      seg_sel<=8'b01111111;
      seg_in<=register[0];
  end
  endcase
```

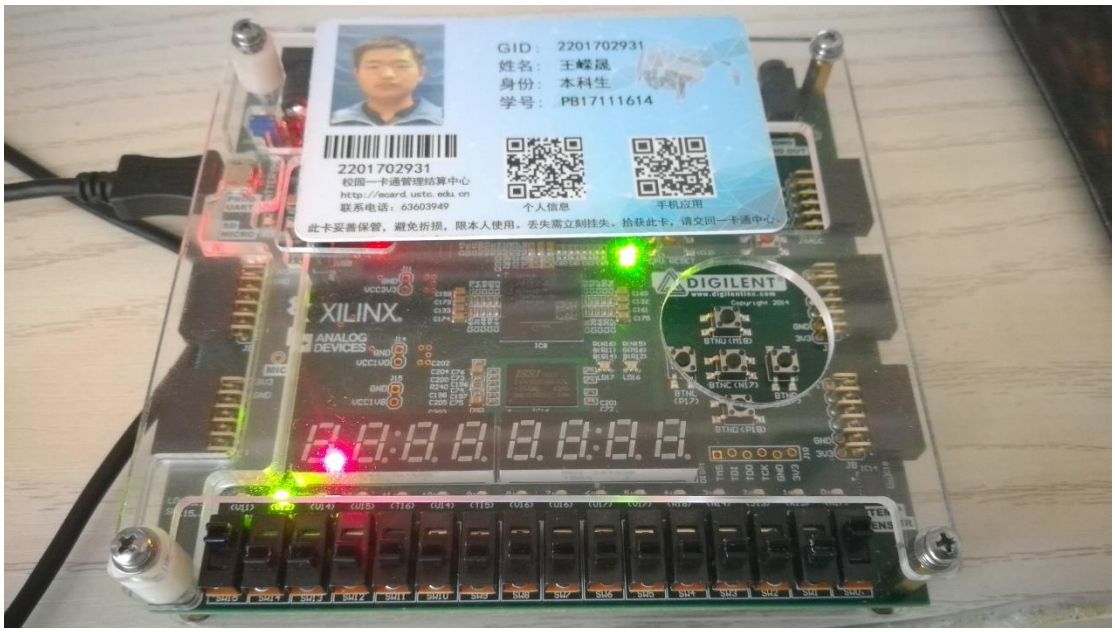
判别哪个七段数码管亮的代码

3. 仿真结果与下载结果:

下载结果:

Wa=010,wd=1110,ra0=010,rd0 如 led 显示所示

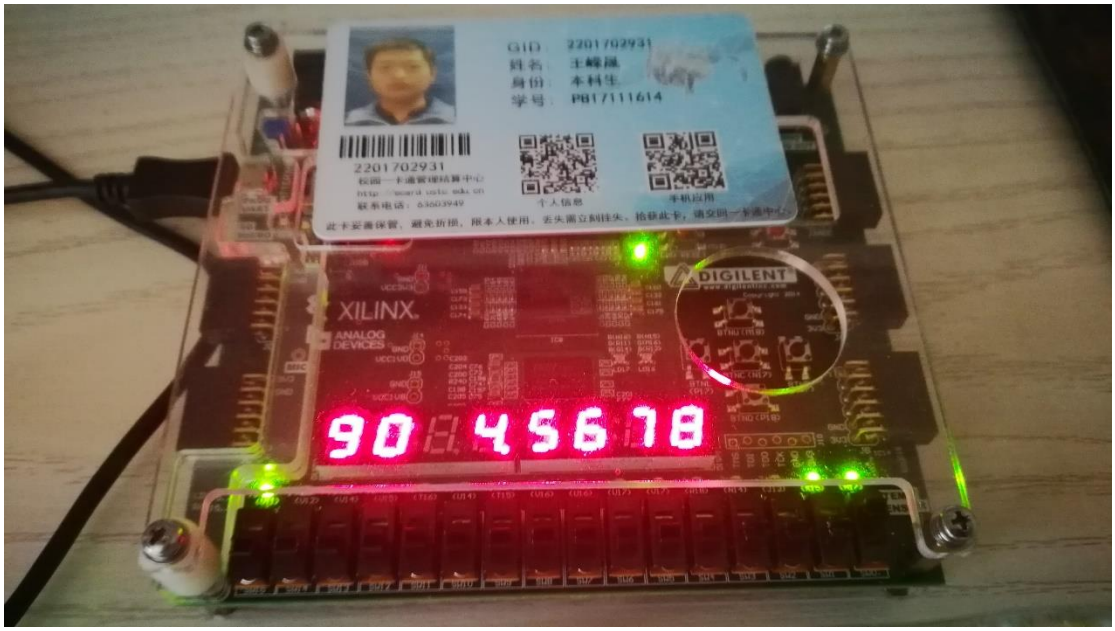
Wa=011 wd=1011 ra0=011 rd0 如 led 显示所示



Rst，队空，只亮小数点

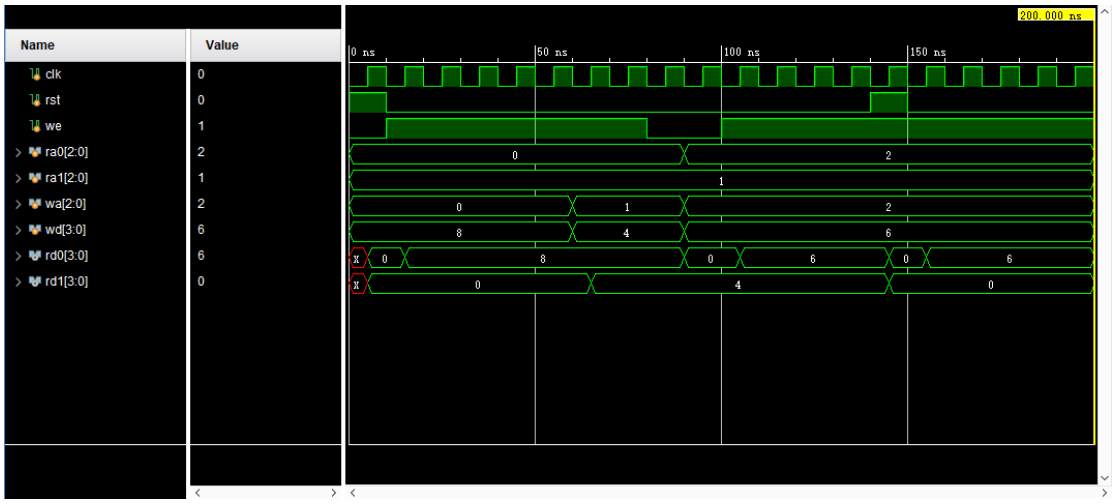
入队 1234567，队满，1 位队头

1 出队，2 为队头

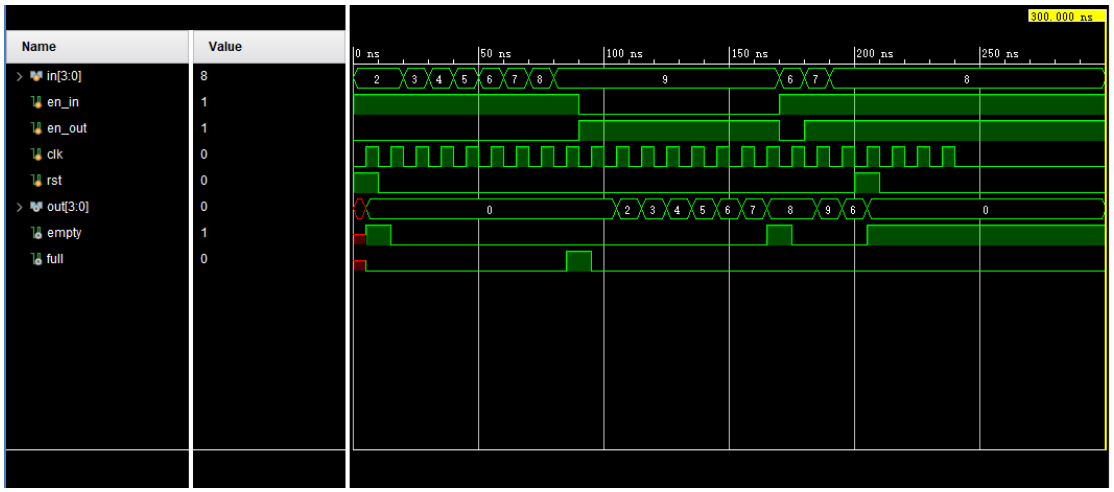


验证为循环队列

仿真截图:



寄存器堆



FIFO 循环队列

4. 仿真代码:

```
Register_file LUT(.clk(clk),.we(we),.rst(rst),.ra0(ra0),.ra1(ra1),.wa(wa),.wd(wd),.rd0(rd0),.rd1(rd1));

initial
begin
    rst=1; we=0; ra0=0; ra1=1; wa=0; wd=8; clk=0; #5
    clk=1; #5
    rst=0; we=1; clk=0; #5
    clk=1; #5
    clk=0; #5
    clk=1; #5
    clk=0; #5
    clk=1; #5
    rst=0; clk=0; #5
    clk=1; #5
    clk=0; #5
    clk=1; #5
    wa=1; wd=4; clk=0; #5
    clk=1; #5
    clk=0; #5
    clk=1; #5
    we=0; clk=0; #5
    clk=1; #5
    wa=2; wd=6; ra0=2; clk=0; #5
    clk=1; #5
    we=1; clk=0; #5
    clk=1; #5
    clk=0; #5
    clk=1; #5
    clk=0; #5
    clk=1; #5
    clk=0; #5
```

寄存器堆仿真代码

```
FIFO DUT(.in(in),.en_in(en_in),.en_out(en_out),.clk(clk),.rst(rst),.out(out),.empty(empty),.full(full));

initial
begin
    in=2; en_in=1; en_out=0; rst=1; clk=0; #5
    clk=1; #5
    rst=0; clk=0; #5
    clk=1; #5
    in=3; clk=0; #5
    clk=1; #5
    in=4; clk=0; #5
    clk=1; #5
    in=5; clk=0; #5
    clk=1; #5
    in=6; clk=0; #5
    clk=1; #5
    in=7; clk=0; #5
    clk=1; #5
    in=8; clk=0; #5
    clk=1; #5
    in=9; clk=0; #5
    clk=1; #5
    en_in=0; en_out=1; clk=0; #5
    clk=1; #5
    clk=0; #5
    clk=1; #5
    clk=0; #5
    clk=1; #5
    clk=0; #5
    clk=1; #5
    clk=0; #5
```

FIFO 仿真代码

5. 结果分析

仿真和下载结果正确，但是 FIFO 队列有一个状态延迟

6. 实验总结

本次试验学习了寄存器堆的设计以及用寄存器堆实现 FIFO 循环队列。实现 FIFO 循环队列时使用七段数码管，难点在于使用板载时钟分频只能用于七段数码管，如何控制输入输出使能信号在时钟周期中单次有效。FPGA 开发板似乎有一点小问题，不过更改了按键后下载结果勉强可以通过。

7. 意见/建议：

实验要求不是很具体，导致操作过程中可能会有一些意想不到的问题。