作业20241026

一、三人表决器

源文件

srbj.v tb_srbj.v

C: > Users > zhan > vivado-repo > srbj > srbj.srcs > sources_1 > new > srbj.v

```
1  `timescale 1ns / 1ps
2
3
4  module srbj( input a,
5    input b,
6    input c,
7    output d
8     );
9      assign d=a&b|a&c|b&c;
10
11 endmodule
12
```

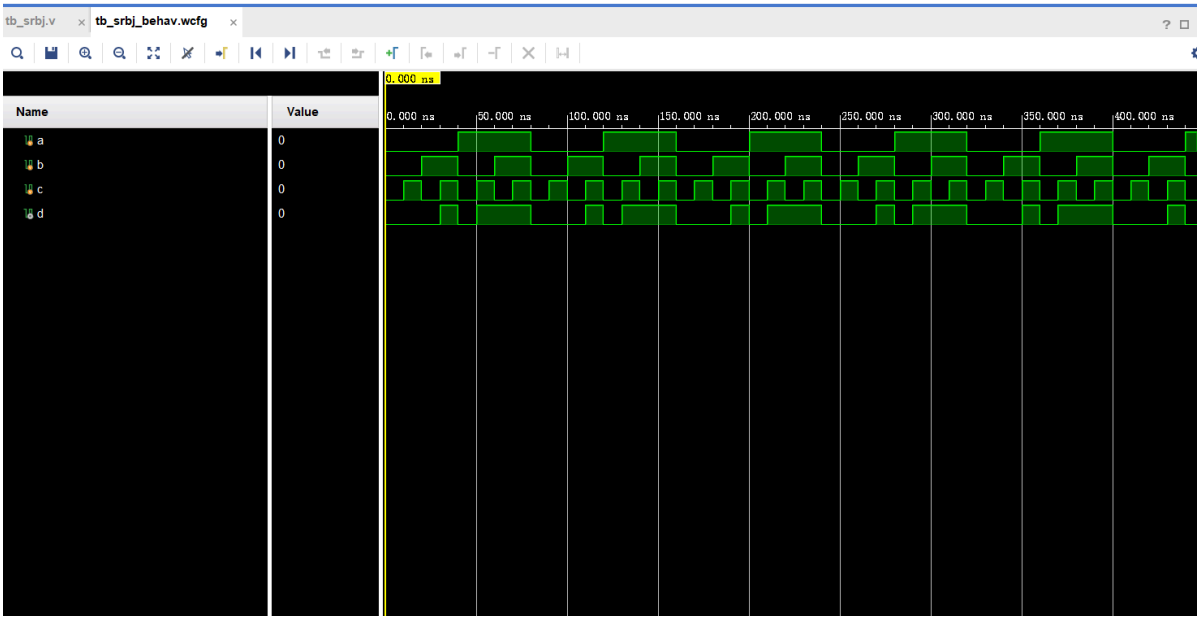
test文件

srbj.v　tb_srbj.v

C: > Users > zhan > vivado-repo > srbj > srbj.srcs > sim_1 > new > tb_srbj.v

```
1  `timescale 1ns / 1ps
2
3
4  module tb_srbj(
5      );
6      reg a,b,c;
7      wire d;
8      srbj sl(a,b,c,d);
9      initial
10     begin
11     a=0;b=0;c=0;
12     end
13     always #10 {a,b,c}={a,b,c}+1;
14 endmodule
15
```

波形图



二、四位加法器

源文件

adder4.v ● test_adder4.v

C: > Users > zhan > vivado-repo > adder4 > adder4.srcs > sources_1 > new > adder4.v

```
1  `timescale 1ns / 1ps
2
3  module adder4(cout,sum,ina,inb,cin);
4      output[3:0] sum;
5      output cout;
6      input[3:0] ina,inb;
7      input cin;
8      assign {cout,sum}=ina+inb+cin;
9  endmodule
10
```

测试文件

```
1  `timescale 1ns/1ns
2  module test_adder4;
3  wire[3:0] sum;
4  wire cout;
5  reg[3:0] ina,inb;
6  reg cin;
7  adder4 adder4_1(cout,sum,ina,inb,cin);
8  initial begin
9  #0 ina = 4'b0001; inb = 4'b1010; cin = 1'b0;
0  #5 ina = 4'b0010; inb = 4'b1010; cin = 1'b1;
1  #5 ina = 4'b0010; inb = 4'b1110; cin = 1'b0;
2  #5 ina = 4'b0011; inb = 4'b1100; cin = 1'b1;
3  #5 ina = 4'b0111; inb = 4'b1001; cin = 1'b0;
4  #5 ina = 4'b0001; inb = 4'b1100; cin = 1'b1;
5  #5 ina = 4'b0011; inb = 4'b1100; cin = 1'b0;
6  #5 ina = 4'b0111; inb = 4'b1111; cin = 1'b1;
7  #5 $finish;
8  end
9  initial
0  $monitor("At time %t, ina(%b) + inb(%b) + cin(%b) = sum(%b)(%2d),cout(%b)",$time, ina, inb, cin, sum, sum, cout);
1  initial
2  begin
3  $dumpfile("test.vcd");
4  $dumpvars(0,test_adder4);
5  end
6  endmodule
```

波形图

三、触发器

源文件

```
module d_flip_flop(d,clk,q);
    input d;
    input clk;
    output q;
    reg q;
    always @ (posedge clk)
    begin
    q <= d;
    end
endmodule
```

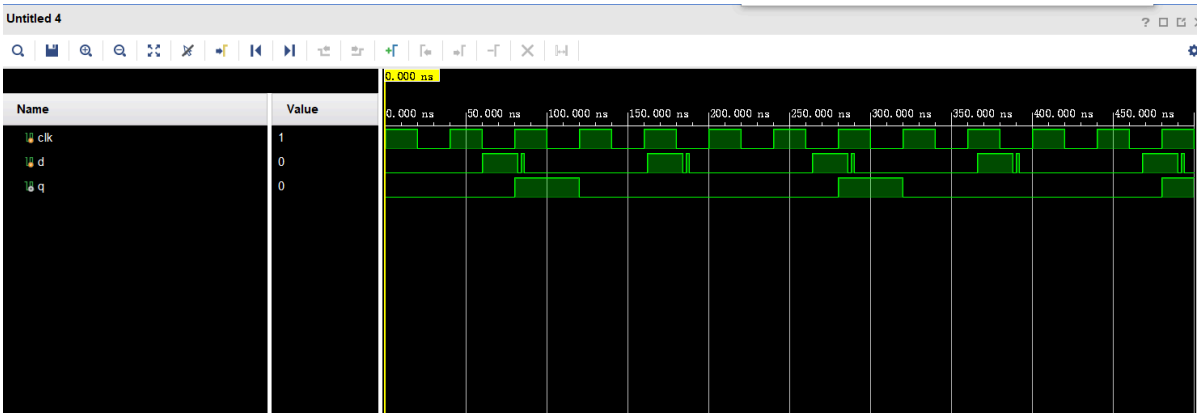
测试文件

d_flip_flop.v × d_flip_flop_tb.v ×

C: > Users > zhan > vivado-repo > d_flip_flop > d_flip_flop.srcs > sim_1 > new > d_flip_flop_tb.v

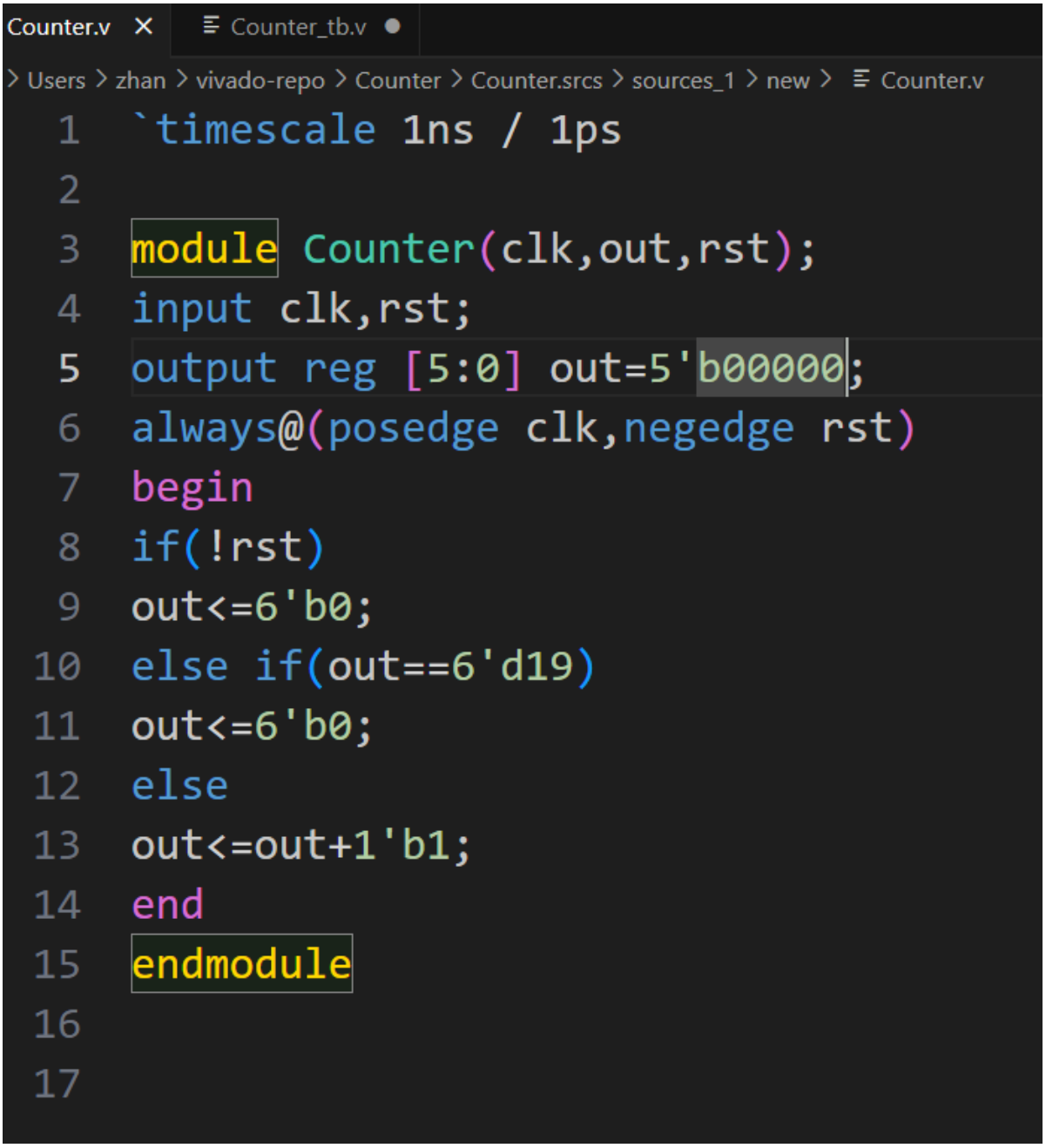
```
1  `timescale 1ns / 1ns
2  module d_flip_flop_tb(
3   );
4
5      reg clk,d=0;
6      wire q;
7      d_flip_flop u1(.d(d),.clk(clk),.q(q));
8      initial
9      begin
10     clk = 1;
11     d <= 0;
12     forever
13     begin
14     #60 d <= 1;
15     #22 d <= 0;
16     #2 d <= 1;
17     #2 d <= 0;
18     #16 d <= 0;
19     end
20     end
21     always #20 clk <= ~clk;
22 endmodule
23
24
```

波形图



四、计数器

源文件


Counter.v
Counter_tb.v
Users > zhan > vivado-repo > Counter > Counter.srcs > sources_1 > new > Counter.v
1 `timescale 1ns / 1ps
2
3 module Counter(clk,out,rst);
4 input clk,rst;
5 output reg [5:0] out=5'b00000;
6 always@(posedge clk,negedge rst)
7 begin
8 if(!rst)
9 out<=6'b0;
10 else if(out==6'd19)
11 out<=6'b0;
12 else
13 out<=out+1'b1;
14 end
15 endmodule
16
17

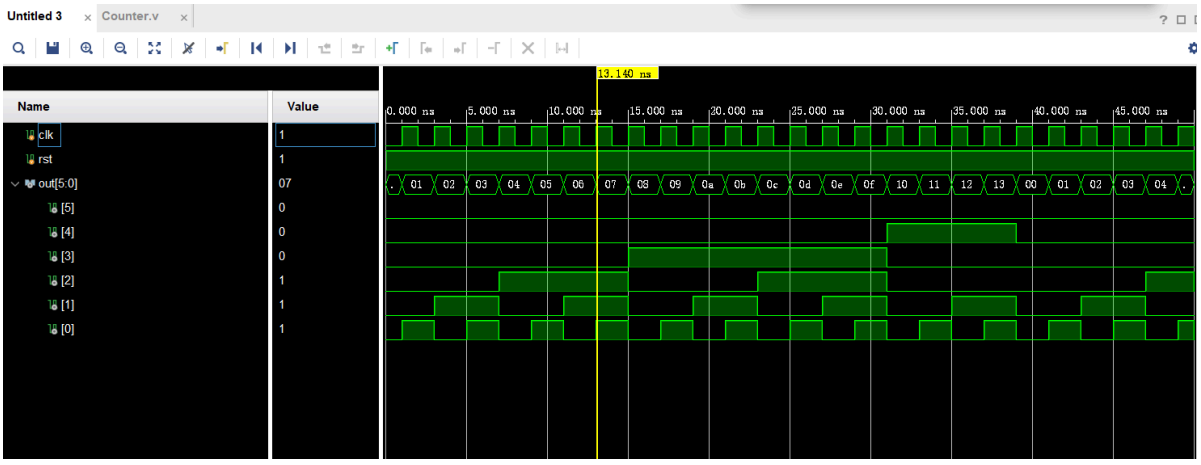
测试文件

Counter.v | Counter_tb.v

C: > Users > zhan > vivado-repo > Counter > Counter.srcs > sim_1 > new > Counter_tb.v

```
1  `timescale 1ns / 1ps
2
3  module Counter_tb( );
4  reg clk,rst;
5  wire [5:0] out;
6  initial
7  begin
8  rst=1;
9  clk=0;
10 #50 rst=0;
11 #30 rst=1;
12 end
13 always #1 clk=~clk;
14 Counter Counter_test(.clk(clk),.rst(rst),.out(out));
15 endmodule
```

波形图



####

五、编码器

源文件

```
module encoder(
    input [7:0] in,
    output reg [2:0] out
);
    always @(*) begin
        case (in)
            8'b00000001: out = 3'b000;
            8'b00000010: out = 3'b001;
            8'b00000100: out = 3'b010;
            8'b00001000: out = 3'b011;
            8'b00010000: out = 3'b100;
            8'b00100000: out = 3'b101;
            8'b01000000: out = 3'b110;
            8'b10000000: out = 3'b111;
            default: out = 3'bxxx;
        endcase
    end
endmodule

```

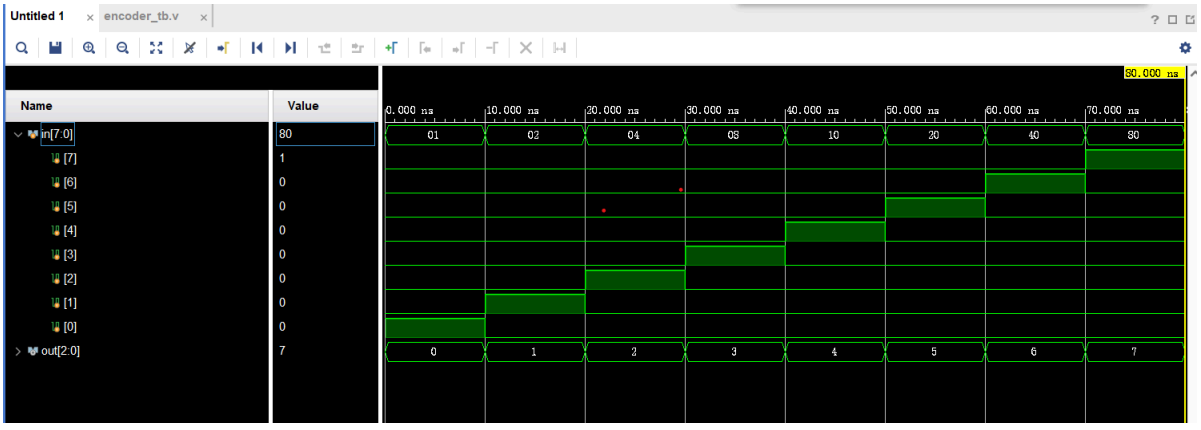
测试文件

```
module encoder_tb;
    reg [7:0] in;
    wire [2:0] out;

    encoder uut (
        .in(in),
        .out(out)
    );

    initial begin
        $monitor("Input = %b, Output = %b", in, out);
        in = 8'b00000001; #10;
        in = 8'b00000010; #10;
        in = 8'b00000100; #10;
        in = 8'b00001000; #10;
        in = 8'b00010000; #10;
        in = 8'b00100000; #10;
        in = 8'b01000000; #10;
        in = 8'b10000000; #10;
        $finish;
    end
endmodule
```

波形图



六、译码器

源文件

```
module decoder(
    input [2:0] in,
    output reg [7:0] out
);
    always @(*) begin
        out = 8'b00000000; // 默认输出为0
        case (in)
            3'b000: out = 8'b00000001;
            3'b001: out = 8'b00000010;
            3'b010: out = 8'b00000100;
            3'b011: out = 8'b00001000;
            3'b100: out = 8'b00010000;
            3'b101: out = 8'b00100000;
            3'b110: out = 8'b01000000;
            3'b111: out = 8'b10000000;
            default: out = 8'b00000000;
        endcase
    end
endmodule

```

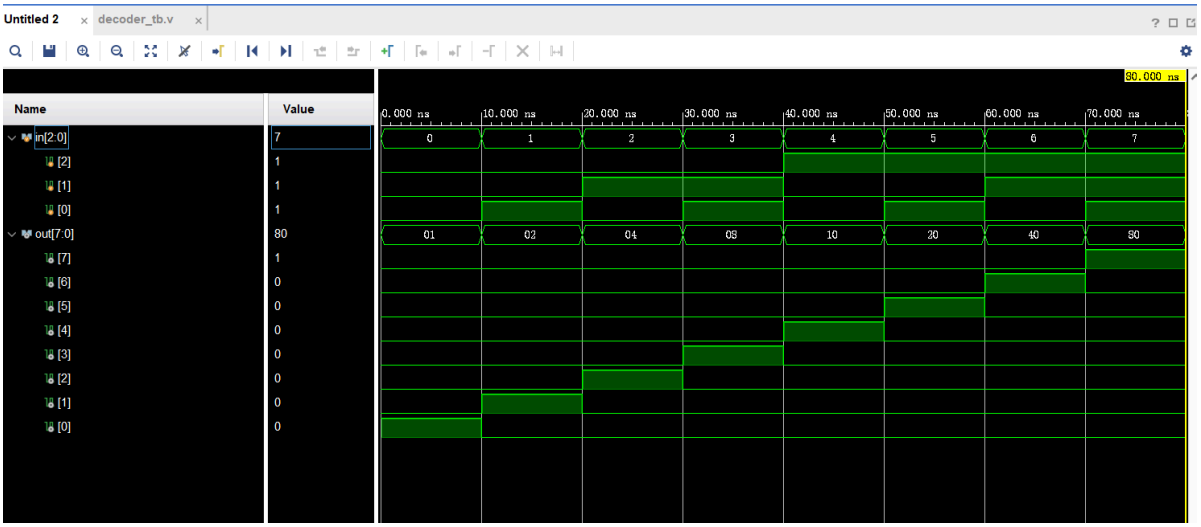
测试文件

```
module decoder_tb;
    reg [2:0] in;
    wire [7:0] out;

    decoder uut (
        .in(in),
        .out(out)
    );

    initial begin
        $monitor("Input = %b, Output = %b", in, out);
        in = 3'b000; #10;
        in = 3'b001; #10;
        in = 3'b010; #10;
        in = 3'b011; #10;
        in = 3'b100; #10;
        in = 3'b101; #10;
        in = 3'b110; #10;
        in = 3'b111; #10;
        $finish;
    end
endmodule
```

波形图



七、多路选择器

源文件

```
`timescale 1ns / 1ps
module mux_3
(
    input wire in0,
    input wire in1,
    input wire in2,
    //3个输入端
    input wire [1:0] sel,//选择端
    output reg out//输出端
);
always@(*)
 case(sel)
 2'b00 : out = in0;//由于 always 块中的赋值语句是针对 寄
 2'b01 : out = in1;//out不能为wire类型
 default : out = in2;
endcase
/* 也可使用 assign 语句实现三路选择器（更简洁）
    assign out = (sel == 2'b00) ? in0 :
                 (sel == 2'b01) ? in1 :
                 in2; // 默认选择 in2
*/
endmodule
```

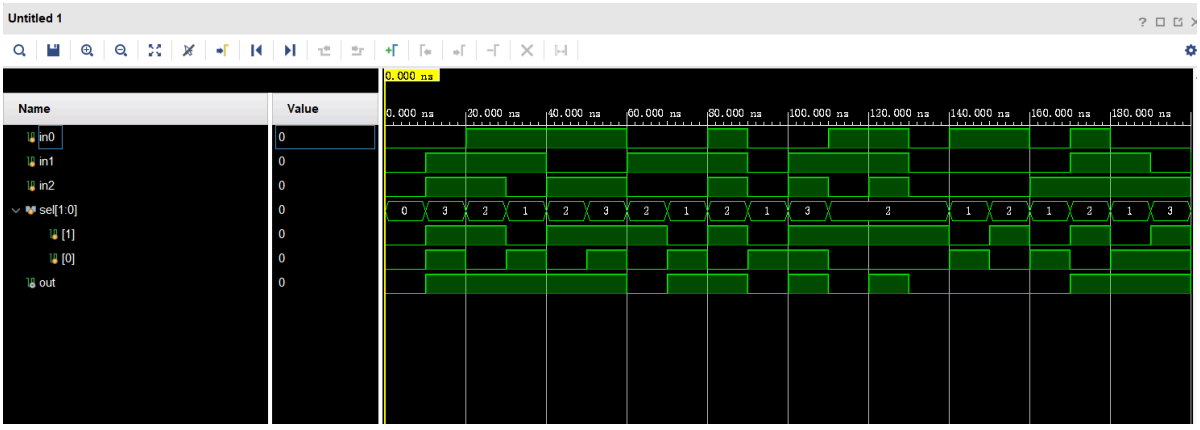
测试文件

```
`timescale 1ns / 1ps
module mux_3_tb( );
 reg in0;
 reg in1;
 reg in2;
 reg [1:0] sel;
 wire out;//在 Verilog 中，如果信号类型不匹配，例如，模块端口定义为 wire 类型，但在 Testb
 initial
 begin
 in0 = 1'b0;
 in1 = 1'b0;
 in2 = 1'b0;
 sel = 2'b00;
 end
 // 随机生成输入信号
 always #10 in0 = $random % 2;
 always #10 in1 = $random % 2;
 always #10 in2 = $random % 2;
 always #10 sel = $random % 4;
 initial begin
```

```
initial begin
//显示时间格式
$timeformat(-9, 0, "ns", 6); // 设置时间格式为 ns，精度为 6
$monitor("@time %t: in0=%b in1=%b in2=%b sel=%b out=%b",$time,in0,in1,in2,sel,out);
end

//------------------------mux_3_inst
mux_3 mux_3_inst
(
.in0(in0), //input in0
.in1(in1), //input in1
.in2(in2), //input in2
.sel(sel), //inputsel
.out(out) //output out
);
endmodule
```

波形图



八、寄存器堆

源文件

```
module regfile(
  input         clk,
  input  [4:0]  raddr1,
  output [31:0] rdata1,
  input  [4:0]  raddr2,
  output [31:0] rdata2,
  input         we,//寄存器堆写使能
  input  [4:0]  waddr,//寄存器堆写地址
  input  [31:0] wdata//寄存器堆写数据
);
//32个32位寄存器
reg [31:0] rf[31:0];
  // 初始化寄存器堆
  integer i;
    initial begin
        for (i = 0; i < 32; i = i + 1) begin
            rf[i] = 32'b0;  // 默认初始化所有寄存器为 0
        end
    end
// WRITE
```

```
always @(posedge clk) begin
          if (we) begin
            rf[waddr] <= wdata;//写入数据
        end
end
// READ OUT 1
assign rdata1 = rf[raddr1];
// READ OUT 2
assign rdata2 = rf[raddr2];
endmodule
```

测试文件

```
`timescale 1ns / 1ps

module tb_top();

reg         clk;

reg  [ 4:0] raddr1;
wire [31:0] rdata1;
reg  [ 4:0] raddr2;
wire [31:0] rdata2;
reg         we;
reg  [ 4:0] waddr;
reg  [31:0] wdata;

reg  [ 3:0] task_phase;

regfile u_regfile(
    .clk       (clk       ),
    .raddr1    (raddr1    ),
    .rdata1    (rdata1    ),
    .raddr2    (raddr2    ),
    .rdata2    (rdata2    ),
    .we        (we        ),
```

```
    .waddr     (waddr      ),
    .wdata     (wdata      )
);

//clk生成时钟信号
initial
begin
    clk = 1'b1;
end
always #5 clk = ~clk;

//测试过程
initial
begin
    //初始化信号
    raddr1 =  5'd0;
    raddr2 =  5'd0;
    waddr  =  5'd0;
    wdata  = 32'd0;
    we     =  1'd0;
    task_phase =  4'd0;
    #2000;
```

```
$display("============================");
$display("Test Begin");
#1;
// Part 0 Begin
#10;
task_phase = 4'd0;
we         = 1'b0;
waddr      = 5'd1;
wdata      = 32'hffffffff;
raddr1     = 5'd1;
#10;
we         = 1'b1;
waddr      = 5'd1;
wdata      = 32'h1111ffff;
#10;
we         = 1'b0;
raddr1     = 5'd2;
raddr2     = 5'd1;
#10;
raddr1     = 5'd1;
```

```
#200;
// Part 1 Begin
#10;
task_phase = 4'd1;
we         = 1'b1;
wdata      = 32'h0000ffff;
waddr      =  5'h10;
raddr1     =  5'h10;
raddr2     =  5'h0f;
#10;
wdata      = 32'h1111ffff;
waddr      =  5'h11;
raddr1     =  5'h11;
raddr2     =  5'h10;
#10;
wdata      = 32'h2222ffff;
waddr      =  5'h12;
raddr1     =  5'h12;
raddr2     =  5'h11;
#10;
wdata      = 32'h3333ffff;
```

```
waddr       =  5'h13;
raddr1      =  5'h13;
raddr2      =  5'h12;
#10;
wdata       = 32'h4444ffff;
waddr       =  5'h14;
raddr1      =  5'h14;
raddr2      =  5'h13;
#10;
raddr1      =  5'h15;
raddr2      =  5'h14;
#10;

#200;
// Part 2 Begin
#10;
task_phase = 4'd2;
we          = 1'b1;
raddr1      =  5'h10;
raddr2      =  5'h0f;
```

```
    #10;
    raddr1      =   5'h12;
    raddr2      =   5'h11;
    #10;
    raddr1      =   5'h13;
    raddr2      =   5'h12;
    #10;
    raddr1      =   5'h14;
    raddr2      =   5'h13;
    #10;

    #50;
    $display("TEST END");
    $finish;
end

    initial begin
```

```
        $dumpfile("rf.vcd");
        $dumpvars(0, u_regfile);
    end


endmodule
```

波形图