一、实验内容

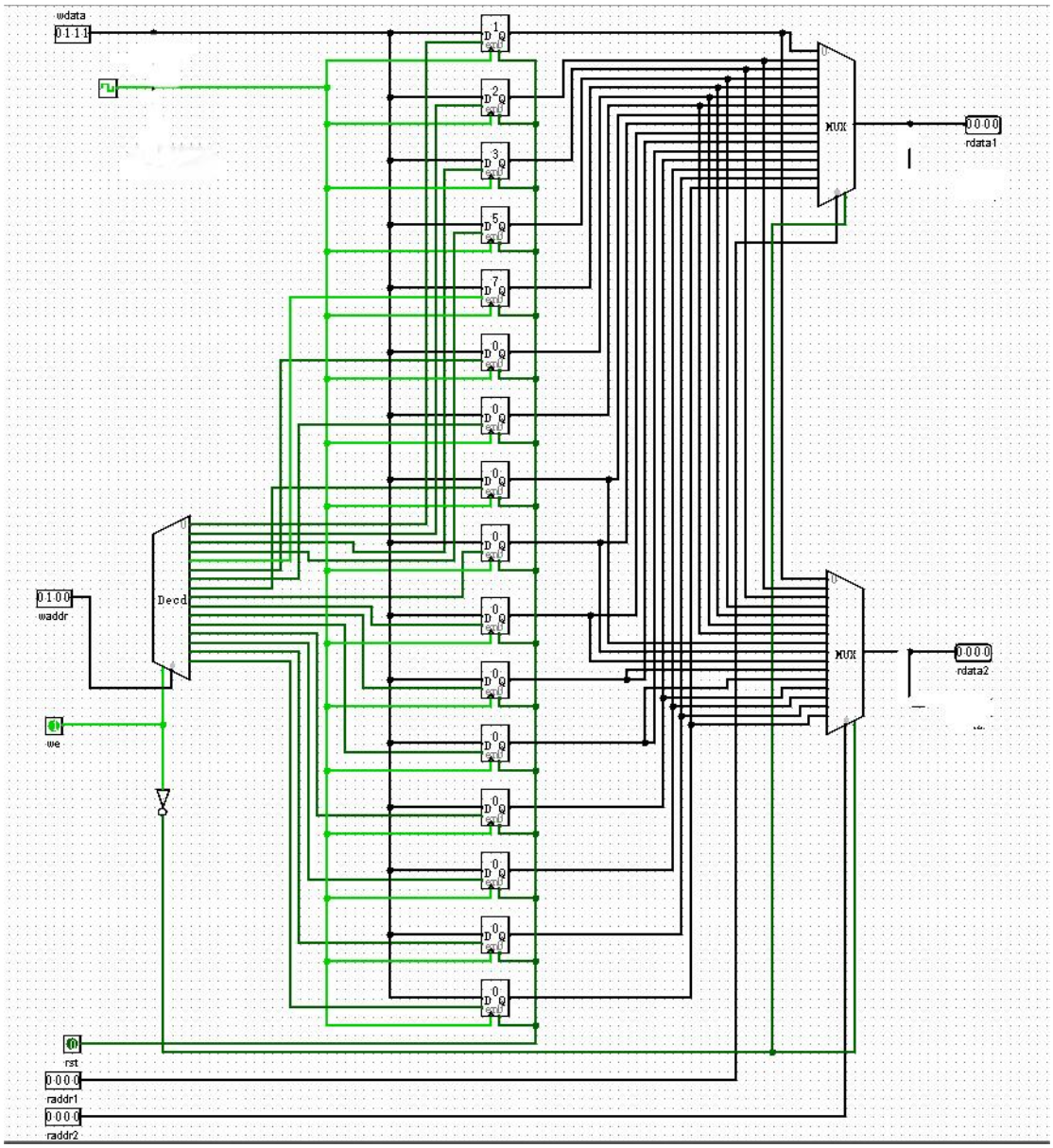
在本次实验中，将使用 Verilog HDL 语言设计实现：

（1）RAM 的设计和仿真

（2）寄存器堆的设计和仿真

二、硬件逻辑图(逻辑验证图在第五部分)

（1）16 个四位寄存器组成的寄存器堆 logisim 原理图



三、模块建模

（1）RAM 的设计和仿真

功能描述：半导体随机读写存储器，简称 RAM，它是数字计算机和其他数字系统的重要存储部件，可存放大量的数据。本设计文件满足时钟信号上升沿时向 ram 内部写入数据、使能信号高电平时存储器才运行，否则输出 z、读写信号高电平为写有效，低电平为读有效，与 ena 同时有效时才可对存储器进行读写的 RAM 功能。

```
module ram (
    input clk,
    input ena,
    input wena,
    input [4:0] addr,
    input [31:0] data_in,
    output reg [31:0] data_out
);
    reg [31:0] mem [31:0]; // 32 个存储位置,每个位置 32bit

        always @(posedge clk) begin
          if(ena)begin
            if(wena)begin//写
                mem[addr] <= data_in;
            end
            else begin//读
                data_out <= mem[addr];
            end
          end
          else begin
            data_out <=32'bz;
          end
        end

        always @(*) begin
          if(ena)begin
            if(wena==0)begin//读
              data_out <= mem[addr];
            end
          end
          else begin
            data_out <=32'bz;
          end
        end
endmodule
```

Inout 型

```
module ram2(
        input clk,
        input wena,
        input [4:0] addr,
        inout [31:0] data,
        input ena
        );
        reg [31:0] mem [31:0]; // 32 个存储位置,每个位置 32bit

                        always @(posedge clk) begin
                           if(ena)begin
                                if(wena)begin//写
                                     mem[addr] <= data;
                                end
                           end
                        end
                        //读操作
                        assign data=(ena==1&&wena==0)?mem[addr]:32'bz;
endmodule
```

（2）寄存器堆的设计和仿真

功能描述：需要记忆多个字且单个寄存器不够用时，需要使用由多个寄存器组的寄存器堆。

```
module Regfiles(
        input clk,
        input rst,
        input we,
        input [4:0] raddr1,
        input [4:0] raddr2,
        input [4:0] waddr,
        input [31:0] wdata,
        output    reg [31:0] rdata1,
        output reg [31:0] rdata2
        );
        wire [31:0]out[31:0];
        wire [31:0]wa;

        decoder uut(
        .iData(waddr),
        .iEna(we),
        .oData(wa)
        );
```

```
genvar i;
generate
for(i=0;i<32;i=i+1)
begin:bin

      reg32 inst(
    .clk(clk),
    .rst(rst),
    .ena(wa[i]),
    .data_in(wdata),
    .data_out(out[i])
    );
end
endgenerate

selector32-1 select1(
    .iC00(out[0]),
    .iC01(out[1]),
    .iC02(out[2]),
    .iC03(out[3]),
    .iC04(out[4]),
    .iC05(out[5]),
    .iC06(out[6]),
    .iC07(out[7]),
    .iC08(out[8]),
    .iC09(out[9]),
    .iC10(out[10]),
    .iC11(out[11]),
    .iC12(out[12]),
    .iC13(out[13]),
    .iC14(out[14]),
    .iC15(out[15]),
    .iC16(out[16]),
    .iC17(out[17]),
    .iC18(out[18]),
    .iC19(out[19]),
    .iC20(out[20]),
    .iC21(out[21]),
    .iC22(out[22]),
    .iC23(out[23]),
    .iC24(out[24]),
    .iC25(out[25]),
    .iC26(out[26]),
    .iC27(out[27]),
```

```
        .iC28(out[28]),
        .iC29(out[29]),
        .iC30(out[30]),
        .iC31(out[31]),
        .iS(raddr1),
        .ena(~we),
        .oZ(rdata1)
        );

    selector32-1 select2(
        .iC00(out[0]),
        .iC01(out[1]),
        .iC02(out[2]),
        .iC03(out[3]),
        .iC04(out[4]),
        .iC05(out[5]),
        .iC06(out[6]),
        .iC07(out[7]),
        .iC08(out[8]),
        .iC09(out[9]),
        .iC10(out[10]),
        .iC11(out[11]),
        .iC12(out[12]),
        .iC13(out[13]),
        .iC14(out[14]),
        .iC15(out[15]),
        .iC16(out[16]),
        .iC17(out[17]),
        .iC18(out[18]),
        .iC19(out[19]),
        .iC20(out[20]),
        .iC21(out[21]),
        .iC22(out[22]),
        .iC23(out[23]),
        .iC24(out[24]),
        .iC25(out[25]),
        .iC26(out[26]),
        .iC27(out[27]),
        .iC28(out[28]),
        .iC29(out[29]),
        .iC30(out[30]),
        .iC31(out[31]),
        .iS(raddr1),
        .ena(~we),
```

```
            .oZ(rdata1)
            );
endmodule

module Dff(
    input CLK,
    input D, //输入信号 D
    input RST_n, //复位信号，低电平有效
    input ena,
    output reg Q1 //输出信号 Q
    );
    always @(negedge CLK or posedge RST_n)
    begin
        if(RST_n==1)
        begin
            Q1<=0;
        end
        else
        begin
            if(ena==1)
            begin
                Q1<=D;
            end
        end
    end
endmodule

module reg32(
    input clk,
    input rst,
    input ena,
    input [31:0] data_in,
    output [31:0] data_out
    );
    genvar i;
    generate
    for(i=0;i<32;i=i+1)
    begin:bin
        Dff dff(
        .CLK(clk),
        .RST_n(rst),
        .D(data_in[i]),
        .ena(ena),
        .Q1(data_out[i])
```

```
            );
        end
        endgenerate
endmodule

module selector32-1(
        input [31:0] iC00,
        input [31:0] iC01,
        input [31:0] iC02,
        input [31:0] iC03,
        input [31:0] iC04,
        input [31:0] iC05,
        input [31:0] iC06,
        input [31:0] iC07,
        input [31:0] iC08,
        input [31:0] iC09,
        input [31:0] iC10,
        input [31:0] iC11,
        input [31:0] iC12,
        input [31:0] iC13,
        input [31:0] iC14,
        input [31:0] iC15,
        input [31:0] iC16,
        input [31:0] iC17,
        input [31:0] iC18,
        input [31:0] iC19,
        input [31:0] iC20,
        input [31:0] iC21,
        input [31:0] iC22,
        input [31:0] iC23,
        input [31:0] iC24,
        input [31:0] iC25,
        input [31:0] iC26,
        input [31:0] iC27,
        input [31:0] iC28,
        input [31:0] iC29,
        input [31:0] iC30,
        input [31:0] iC31,
        input [4:0] iS,
        input ena,//高电平有效，可以读数据
        output [31:0] oZ
        );
        reg [31:0]temp[31:0];
        reg [31:0]t;
```

```
    always @(*)
    begin
        temp[0]=iC00;
        temp[1]=iC01;
        temp[2]=iC02;
        temp[3]=iC03;
        temp[4]=iC04;
        temp[5]=iC05;
        temp[6]=iC06;
        temp[7]=iC07;
        temp[8]=iC08;
        temp[9]=iC09;
        temp[10]=iC10;
        temp[11]=iC11;
        temp[12]=iC12;
        temp[13]=iC13;
        temp[14]=iC14;
        temp[15]=iC15;
        temp[16]=iC16;
        temp[17]=iC17;
        temp[18]=iC18;
        temp[19]=iC19;
        temp[20]=iC20;
        temp[21]=iC21;
        temp[22]=iC22;
        temp[23]=iC23;
        temp[24]=iC24;
        temp[25]=iC25;
        temp[26]=iC26;
        temp[27]=iC27;
        temp[28]=iC28;
        temp[29]=iC29;
        temp[30]=iC30;
        temp[31]=iC31;
        if(ena==1)
            t<=temp[iS];
        else
            t=32'bz;
    end
    assign oZ=t;
endmodule

module decoder(iData,iEna,oData);
    input [4:0] iData;//{d4，d3，D2,d1,d0}
```

```
    input iEna;
    genvar i;
    generate
    for(i=0;i<32;i=i+1)
    begin:bit
        assign oData[i]=(iData==i)&&(iEna==1)?1'b1:1'b0;
    end
    endgenerate
endmodule
```

四、测试模块建模

（1）RAM 的设计和仿真

```
module ram_tb;
    reg clk;
    reg ena;
    reg wena;
    reg [4:0] addr;
    reg [31:0] data_in;

    wire [31:0] data_out;

    ram uut (
        .clk(clk),
        .ena(ena),
        .wena(wena),
        .addr(addr),
        .data_in(data_in),
        .data_out(data_out)
    );

    initial begin
        clk = 0;
        forever #5 clk = ~clk; // 10ns 周期
    end

    initial begin
        ena = 0;
        wena = 0;
        addr = 0;
        data_in = 0;
        #10;
        // 写操作
```

```
            ena = 1;
            wena = 1;
            addr = 5'b00000; // 地址 0
            data_in = 32'h12345678; // 写入数据
            #10;
            data_in=32'h00000000;
            // 读操作
            ena = 1;
            wena = 0;
            addr = 5'b00000; // 读取地址 0
            #10;
            // 写操作
            ena = 1;
            wena = 1;
            addr = 5'b00100; // 地址 0
            data_in = 32'h66666666; // 写入数据
            #10;
            wena = 0; // 停止写操作
            #5;
            // 读操作
            ena = 1;
            wena = 0;
            addr = 5'b00000; // 读取地址 0
            #5;
            // 读操作
            ena = 1;
            wena = 0;
            addr = 5'b00100; // 读取地址 0
            #10;
            ena=0;
      end
endmodule
```

Inout 型 tb 文件

```
module ram2_tb;
      reg clk;
      reg ena;
      reg wena;
      reg [4:0] addr;
      wire [31:0] data;
      ram2 inst(
            .clk(clk),
            .ena(ena),
            .wena(wena),
            .addr(addr),
```

```
        .data(data)
        );
    integer i;
            initial
            begin
                clk=0;
                for(i=0;i<500;i=i+1)
                begin
                    #5;
                    clk=~clk;
                end
            end
            initial
            begin
                ena=0;
                #20 ena=1;
            end
            initial
            begin
                wena=0;
                #10 wena=1;
                #30 wena=0;
            end
            initial
            begin
                force data=32'habcdef12;
                #30 force data=32'h12345678;
                #10;
            end
            initial
            begin
                addr=5'b01010;
                #30 addr=5'b10101;
                #10 addr=5'b01010;
                #10 addr=5'b10101;
                #10;
            end
endmodule
```

（2）寄存器堆的设计和仿真

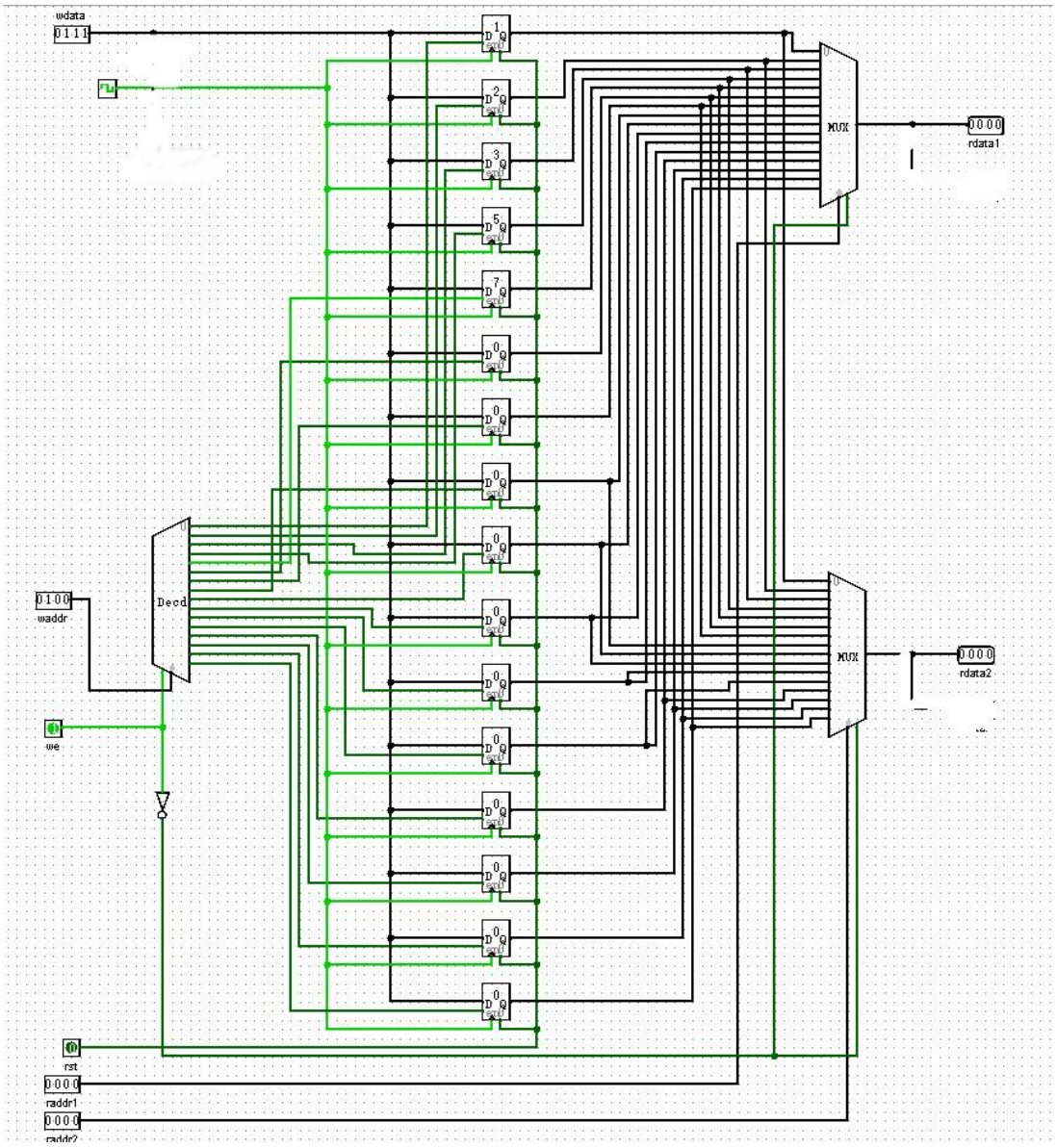
```
module Regfiles_tb;
reg clk;
    reg rst;
```

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

Regfiles uut (
        .clk(clk),
        .rst(rst),
        .we(we),
        .raddr1(raddr1),
        .raddr2(raddr2),
        .waddr(waddr),
        .wdata(wdata),
        .rdata1(rdata1),
        .rdata2(rdata2)
);

initial begin
        clk = 0;
        forever #5 clk = ~clk; // T=10;
end

initial begin
        rst = 1; //高电平复位
        we = 0;
        raddr1 = 0;
        raddr2 = 0;
        waddr = 0;
        wdata = 0;

        #10;
        rst = 0;

        // 写操作
        #10;
        we = 1;
        waddr = 5'b00010;
        wdata = 32'h12345678; // 写数据
        #10;
        we = 0; // 停止写操作
        wdata=32'h00000000;
```

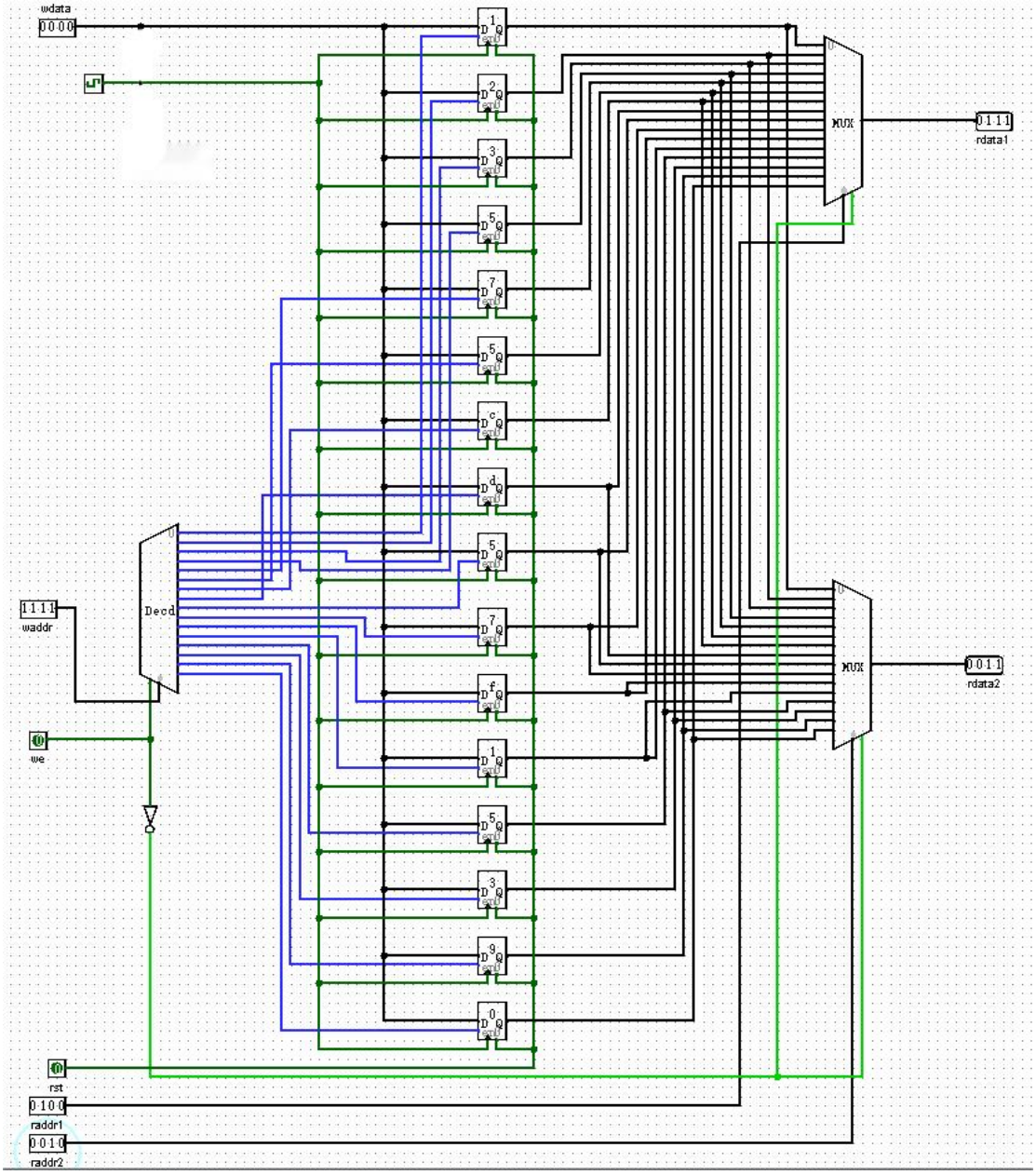
```
        // 读操作
        #5;
        raddr1 = 5'b00010; // 读取\
        #10;
        //观察 dataout

        #10 raddr2 = 5'b00100;
        #10;

    end

endmodule
```

五、实验结果

Logisim 逻辑验证图：



该图为写入验证。此时 we 为 1，waddr 为 0100，当时钟上升沿到来时，wdata=0111 的数据写入五号寄存器（从上至下为 1、2、3……号寄存器）。

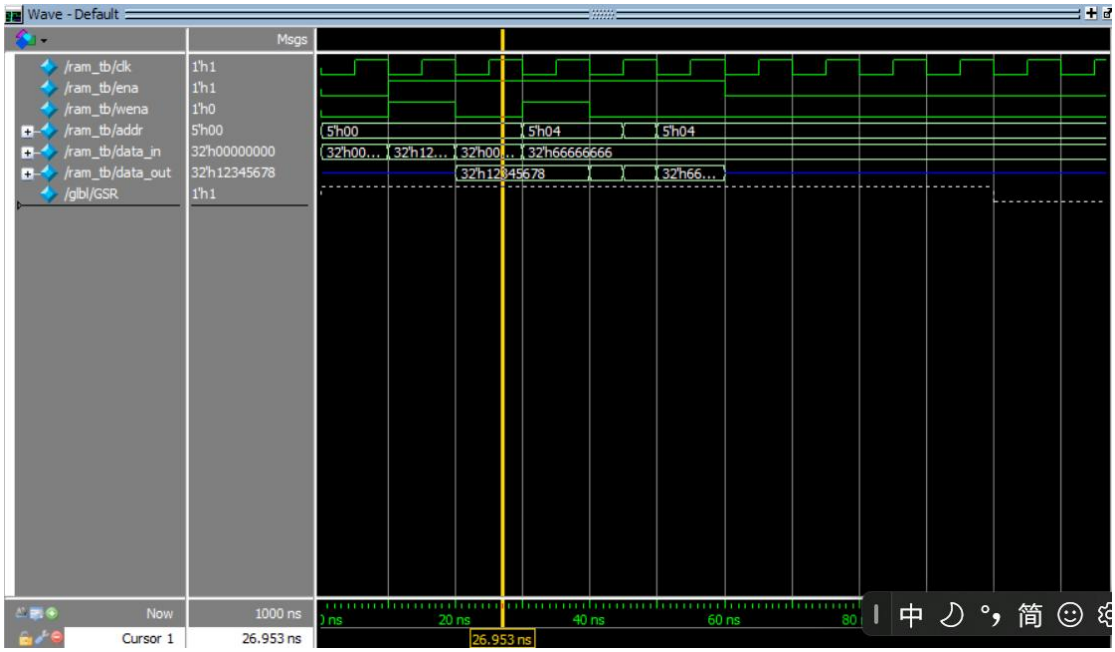


该图为读入验证。当 we 为 0 时，raddr1=0100，使 rada1 读入对应寄存器的值 0111；raddr2=0010，使 rdata2 读入对应的寄存器的值 0011。

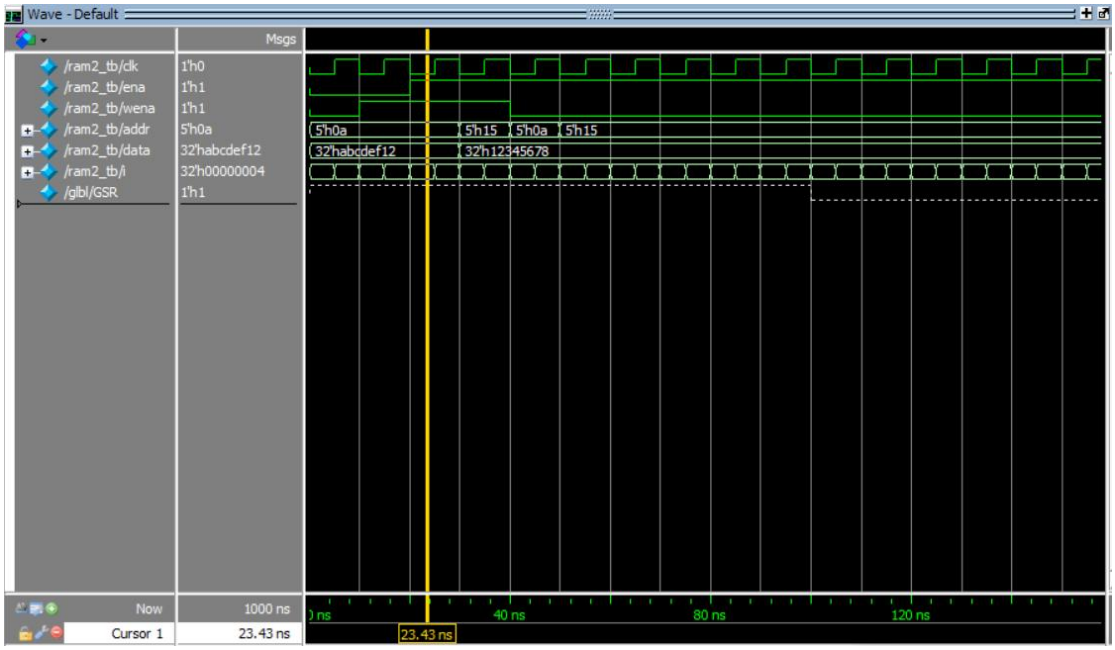
Modelsim 仿真波形图

（1）RAM 的设计和仿真

Ram：

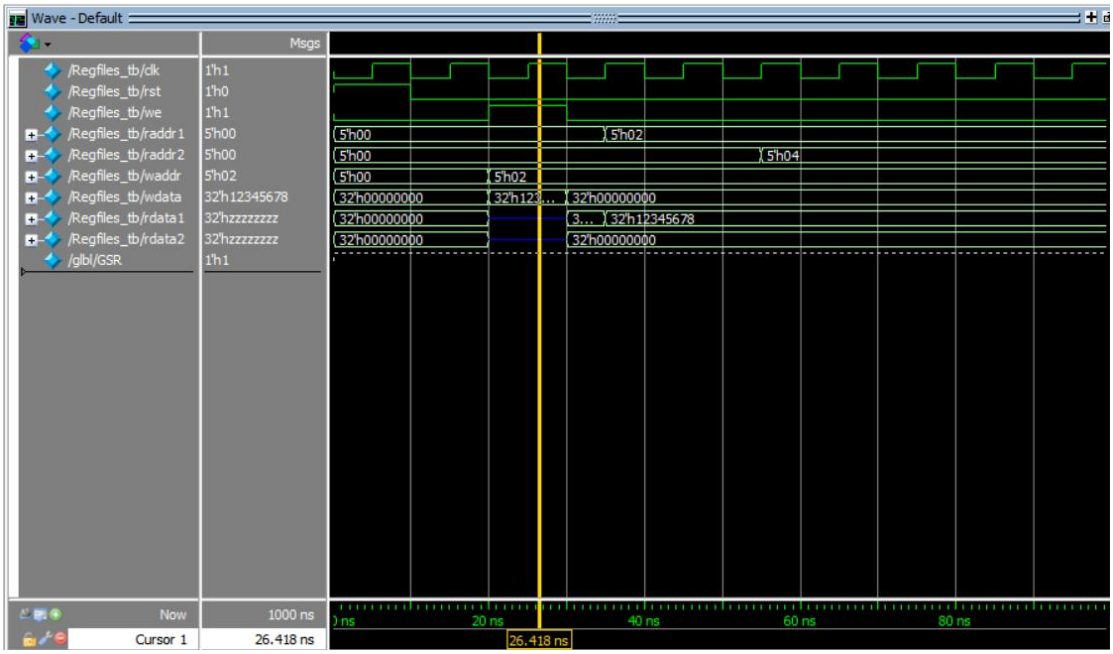


Ram2：



说明：两个 ram 的测试思路均是---以“读操作查写操作”。即某时刻写入数据，待写入后某时刻读取，若能读到，则说明当时写操作执行完成。并且同时验证了读操作的正确性。

（2）寄存器堆的设计和仿真



说明：该寄存器堆的测试思路也是---以“读操作查写操作”。即某时刻写入数据，待写入后某时刻读取，若能读到，则说明当时写操作执行完成。并且同时验证了读操作的正确性。