同济大学计算机系

数字逻辑课程实验报告

学　　号 2353900

姓　　名 汪嘉晨

专　　业 计算机科学与技术（精英班）

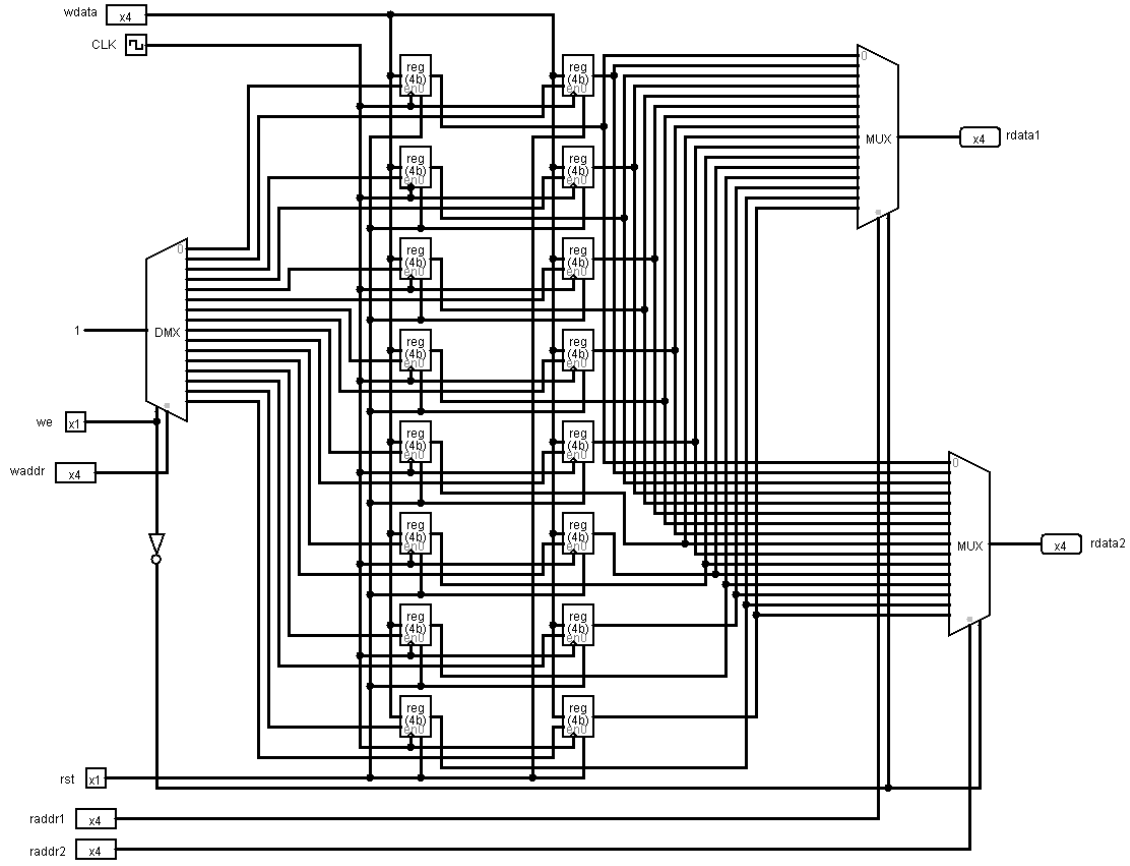
授课老师 郭玉臣

一、实验内容

使用 Verilog HDL 语言实现 RAM 以及寄存器堆的设计和仿真：
1.深入了解 RAM 与寄存器堆的原理。
2.用 Logisim 画出一个包含 16 个寄存器的寄存器堆原理图。
3.学习使用 Verilog HDL 语言设计实现 RAM 以及寄存器堆。

二、硬件逻辑图

包含 16 个寄存器的寄存器堆



三、模块建模

1. 双线输入输出 ram

```
module ram(
        input clk,
        input ena,
        input wena,
        input [4:0] addr,
        input [31:0] data_in,
        output [31:0] data_out
        );
```

```
        reg [31:0] temp[31:0];
        assign data_out = (ena == 1 && wena == 0) ? temp[addr] : {32{1'bz}};
        always @(posedge clk)
            if(ena == 1 && wena == 1)
                temp[addr] = data_in;
    endmodule
```

2. 单线输入输出 ram

```
    module ram2(
        input clk,
        input ena,
        input wena,
        input [4:0] addr,
        inout [31:0] data
        );

        reg [31:0] temp[31:0];
        assign data = (ena == 1 && wena == 0) ? temp[addr] : {32{1'bz}};
        always @(posedge clk)
            if(ena == 1 && wena == 1)
                temp[addr] = data;
    endmodule
```

3. 寄存器堆

顶层模块

```
        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 [31:0] rdata1,
            output [31:0] rdata2
            );
            wire [31:0] encode_waddr;
            wire [31:0] temp_data[31:0];

            decoder Decoder(.iData(waddr),.iEna(we),.oData(encode_waddr));

            pcreg
        Reg0(.clk(~clk), .rst(rst), .ena(encode_waddr[0]), .data_in(wdata), .data_out(t
        emp_data[0]));
            pcreg
        Reg1(.clk(~clk), .rst(rst), .ena(encode_waddr[1]), .data_in(wdata), .data_out(t
```

```
emp_data[1]));
    pcreg
Reg2(.clk(~clk), .rst(rst), .ena(encode_waddr[2]), .data_in(wdata), .data_out(t
emp_data[2]));
    pcreg
Reg3(.clk(~clk), .rst(rst), .ena(encode_waddr[3]), .data_in(wdata), .data_out(t
emp_data[3]));
    pcreg
Reg4(.clk(~clk), .rst(rst), .ena(encode_waddr[4]), .data_in(wdata), .data_out(t
emp_data[4]));
    pcreg
Reg5(.clk(~clk), .rst(rst), .ena(encode_waddr[5]), .data_in(wdata), .data_out(t
emp_data[5]));
    pcreg
Reg6(.clk(~clk), .rst(rst), .ena(encode_waddr[6]), .data_in(wdata), .data_out(t
emp_data[6]));
    pcreg
Reg7(.clk(~clk), .rst(rst), .ena(encode_waddr[7]), .data_in(wdata), .data_out(t
emp_data[7]));
    pcreg
Reg8(.clk(~clk), .rst(rst), .ena(encode_waddr[8]), .data_in(wdata), .data_out(t
emp_data[8]));
    pcreg
Reg9(.clk(~clk), .rst(rst), .ena(encode_waddr[9]), .data_in(wdata), .data_out(t
emp_data[9]));
    pcreg
Reg10(.clk(~clk), .rst(rst), .ena(encode_waddr[10]), .data_in(wdata), .data_ou
t(temp_data[10]));
    pcreg
Reg11(.clk(~clk), .rst(rst), .ena(encode_waddr[11]), .data_in(wdata), .data_ou
t(temp_data[11]));
    pcreg
Reg12(.clk(~clk), .rst(rst), .ena(encode_waddr[12]), .data_in(wdata), .data_ou
t(temp_data[12]));
    pcreg
Reg13(.clk(~clk), .rst(rst), .ena(encode_waddr[13]), .data_in(wdata), .data_ou
t(temp_data[13]));
    pcreg
Reg14(.clk(~clk), .rst(rst), .ena(encode_waddr[14]), .data_in(wdata), .data_ou
t(temp_data[14]));
    pcreg
Reg15(.clk(~clk), .rst(rst), .ena(encode_waddr[15]), .data_in(wdata), .data_ou
t(temp_data[15]));
    pcreg
```

```
Reg16(.clk(~clk), .rst(rst), .ena(encode_waddr[16]), .data_in(wdata), .data_ou
t(temp_data[16]));
    pcreg
Reg17(.clk(~clk), .rst(rst), .ena(encode_waddr[17]), .data_in(wdata), .data_ou
t(temp_data[17]));
    pcreg
Reg18(.clk(~clk), .rst(rst), .ena(encode_waddr[18]), .data_in(wdata), .data_ou
t(temp_data[18]));
    pcreg
Reg19(.clk(~clk), .rst(rst), .ena(encode_waddr[19]), .data_in(wdata), .data_ou
t(temp_data[19]));
    pcreg
Reg20(.clk(~clk), .rst(rst), .ena(encode_waddr[20]), .data_in(wdata), .data_ou
t(temp_data[20]));
    pcreg
Reg21(.clk(~clk), .rst(rst), .ena(encode_waddr[21]), .data_in(wdata), .data_ou
t(temp_data[21]));
    pcreg
Reg22(.clk(~clk), .rst(rst), .ena(encode_waddr[22]), .data_in(wdata), .data_ou
t(temp_data[22]));
    pcreg
Reg23(.clk(~clk), .rst(rst), .ena(encode_waddr[23]), .data_in(wdata), .data_ou
t(temp_data[23]));
    pcreg
Reg24(.clk(~clk), .rst(rst), .ena(encode_waddr[24]), .data_in(wdata), .data_ou
t(temp_data[24]));
    pcreg
Reg25(.clk(~clk), .rst(rst), .ena(encode_waddr[25]), .data_in(wdata), .data_ou
t(temp_data[25]));
    pcreg
Reg26(.clk(~clk), .rst(rst), .ena(encode_waddr[26]), .data_in(wdata), .data_ou
t(temp_data[26]));
    pcreg
Reg27(.clk(~clk), .rst(rst), .ena(encode_waddr[27]), .data_in(wdata), .data_ou
t(temp_data[27]));
    pcreg
Reg28(.clk(~clk), .rst(rst), .ena(encode_waddr[28]), .data_in(wdata), .data_ou
t(temp_data[28]));
    pcreg
Reg29(.clk(~clk), .rst(rst), .ena(encode_waddr[29]), .data_in(wdata), .data_ou
t(temp_data[29]));
    pcreg
Reg30(.clk(~clk), .rst(rst), .ena(encode_waddr[30]), .data_in(wdata), .data_ou
t(temp_data[30]));
```

```
        pcreg
Reg31(.clk(~clk), .rst(rst), .ena(encode_waddr[31]), .data_in(wdata), .data_ou
t(temp_data[31]));

        selector
Rdata1(.iC0(temp_data[0]), .iC1(temp_data[1]), .iC2(temp_data[2]), .iC3(tem
p_data[3]),
                .iC4(temp_data[4]), .iC5(temp_data[5]), .iC6(temp_
data[6]), .iC7(temp_data[7]),
                .iC8(temp_data[8]), .iC9(temp_data[9]), .iC10(temp
_data[10]), .iC11(temp_data[11]),
                .iC12(temp_data[12]), .iC13(temp_data[13]), .iC14(t
emp_data[14]), .iC15(temp_data[15]),
                .iC16(temp_data[16]), .iC17(temp_data[17]), .iC18(t
emp_data[18]), .iC19(temp_data[19]),
                .iC20(temp_data[20]), .iC21(temp_data[21]), .iC22(t
emp_data[22]), .iC23(temp_data[23]),
                .iC24(temp_data[24]), .iC25(temp_data[25]), .iC26(t
emp_data[26]), .iC27(temp_data[27]),
                .iC28(temp_data[28]), .iC29(temp_data[29]), .iC30(t
emp_data[30]), .iC31(temp_data[31]),
                .iS(raddr1), .iEna(~we), .oZ(rdata1));
        selector
Rdata2(.iC0(temp_data[0]), .iC1(temp_data[1]), .iC2(temp_data[2]), .iC3(tem
p_data[3]),
                .iC4(temp_data[4]), .iC5(temp_data[5]), .iC6(temp_
data[6]), .iC7(temp_data[7]),
                .iC8(temp_data[8]), .iC9(temp_data[9]), .iC10(temp
_data[10]), .iC11(temp_data[11]),
                .iC12(temp_data[12]), .iC13(temp_data[13]), .iC14(t
emp_data[14]), .iC15(temp_data[15]),
                .iC16(temp_data[16]), .iC17(temp_data[17]), .iC18(t
emp_data[18]), .iC19(temp_data[19]),
                .iC20(temp_data[20]), .iC21(temp_data[21]), .iC22(t
emp_data[22]), .iC23(temp_data[23]),
                .iC24(temp_data[24]), .iC25(temp_data[25]), .iC26(t
emp_data[26]), .iC27(temp_data[27]),
                .iC28(temp_data[28]), .iC29(temp_data[29]), .iC30(t
emp_data[30]), .iC31(temp_data[31]),
                .iS(raddr2), .iEna(~we), .oZ(rdata2));
        endmodule
```

译码器

```
        module decoder(
            input [4:0] iData,
```

```
        input iEna,
        output reg [31:0] oData
        );

        always @(*)
        begin
            oData = {32{1'b0}};
            if(iEna == 1)
                oData[iData] = 1'b1;
        end
    endmodule
```

32 位寄存器

```
    module pcreg(
        input clk,
        input rst,
        input ena,
        input [31:0] data_in,
        output reg [31:0] data_out
        );

        wire [31:0] data_tmp;
        Asynchronous_D_FF
    b0(.CLK(clk), .D(data_in[0]), .RST_n(~rst), .Q1(data_tmp[0]));
        Asynchronous_D_FF
    b1(.CLK(clk), .D(data_in[1]), .RST_n(~rst), .Q1(data_tmp[1]));
        Asynchronous_D_FF
    b2(.CLK(clk), .D(data_in[2]), .RST_n(~rst), .Q1(data_tmp[2]));
        Asynchronous_D_FF
    b3(.CLK(clk), .D(data_in[3]), .RST_n(~rst), .Q1(data_tmp[3]));
        Asynchronous_D_FF
    b4(.CLK(clk), .D(data_in[4]), .RST_n(~rst), .Q1(data_tmp[4]));
        Asynchronous_D_FF
    b5(.CLK(clk), .D(data_in[5]), .RST_n(~rst), .Q1(data_tmp[5]));
        Asynchronous_D_FF
    b6(.CLK(clk), .D(data_in[6]), .RST_n(~rst), .Q1(data_tmp[6]));
        Asynchronous_D_FF
    b7(.CLK(clk), .D(data_in[7]), .RST_n(~rst), .Q1(data_tmp[7]));
        Asynchronous_D_FF
    b8(.CLK(clk), .D(data_in[8]), .RST_n(~rst), .Q1(data_tmp[8]));
        Asynchronous_D_FF
    b9(.CLK(clk), .D(data_in[9]), .RST_n(~rst), .Q1(data_tmp[9]));
        Asynchronous_D_FF
    b10(.CLK(clk), .D(data_in[10]), .RST_n(~rst), .Q1(data_tmp[10]));
        Asynchronous_D_FF
```

```
b11(.CLK(clk), .D(data_in[11]), .RST_n(~rst), .Q1(data_tmp[11]));
    Asynchronous_D_FF
b12(.CLK(clk), .D(data_in[12]), .RST_n(~rst), .Q1(data_tmp[12]));
    Asynchronous_D_FF
b13(.CLK(clk), .D(data_in[13]), .RST_n(~rst), .Q1(data_tmp[13]));
    Asynchronous_D_FF
b14(.CLK(clk), .D(data_in[14]), .RST_n(~rst), .Q1(data_tmp[14]));
    Asynchronous_D_FF
b15(.CLK(clk), .D(data_in[15]), .RST_n(~rst), .Q1(data_tmp[15]));
    Asynchronous_D_FF
b16(.CLK(clk), .D(data_in[16]), .RST_n(~rst), .Q1(data_tmp[16]));
    Asynchronous_D_FF
b17(.CLK(clk), .D(data_in[17]), .RST_n(~rst), .Q1(data_tmp[17]));
    Asynchronous_D_FF
b18(.CLK(clk), .D(data_in[18]), .RST_n(~rst), .Q1(data_tmp[18]));
    Asynchronous_D_FF
b19(.CLK(clk), .D(data_in[19]), .RST_n(~rst), .Q1(data_tmp[19]));
    Asynchronous_D_FF
b20(.CLK(clk), .D(data_in[20]), .RST_n(~rst), .Q1(data_tmp[20]));
    Asynchronous_D_FF
b21(.CLK(clk), .D(data_in[21]), .RST_n(~rst), .Q1(data_tmp[21]));
    Asynchronous_D_FF
b22(.CLK(clk), .D(data_in[22]), .RST_n(~rst), .Q1(data_tmp[22]));
    Asynchronous_D_FF
b23(.CLK(clk), .D(data_in[23]), .RST_n(~rst), .Q1(data_tmp[23]));
    Asynchronous_D_FF
b24(.CLK(clk), .D(data_in[24]), .RST_n(~rst), .Q1(data_tmp[24]));
    Asynchronous_D_FF
b25(.CLK(clk), .D(data_in[25]), .RST_n(~rst), .Q1(data_tmp[25]));
    Asynchronous_D_FF
b26(.CLK(clk), .D(data_in[26]), .RST_n(~rst), .Q1(data_tmp[26]));
    Asynchronous_D_FF
b27(.CLK(clk), .D(data_in[27]), .RST_n(~rst), .Q1(data_tmp[27]));
    Asynchronous_D_FF
b28(.CLK(clk), .D(data_in[28]), .RST_n(~rst), .Q1(data_tmp[28]));
    Asynchronous_D_FF
b29(.CLK(clk), .D(data_in[29]), .RST_n(~rst), .Q1(data_tmp[29]));
    Asynchronous_D_FF
b30(.CLK(clk), .D(data_in[30]), .RST_n(~rst), .Q1(data_tmp[30]));
    Asynchronous_D_FF
b31(.CLK(clk), .D(data_in[31]), .RST_n(~rst), .Q1(data_tmp[31]));

    always @(*)
        if(rst == 1 && (ena == 1 || ena == 0))
```

```
                data_out = 0;
            else if(rst == 0 && ena == 1)
                data_out = data_tmp;
            else if(rst == 0 && ena == 0)
                ;
            else
                data_out = {32{1'bx}};
    endmodule
```

异步复位 D 触发器

```
    module Asynchronous_D_FF(
        input CLK,
        input D,
        input RST_n,
        output reg Q1,
        output reg Q2
        );

        always @(posedge CLK or negedge RST_n)
            if(RST_n == 0)
                {Q1, Q2} = 2'b01;
            else if(RST_n == 1)
                case(D)
                    0: {Q1, Q2} = 2'b01;
                    1: {Q1, Q2} = 2'b10;
                    default: {Q1, Q2} = 2'bxx;
                endcase
            else
                {Q1, Q2} = 2'bxx;
    endmodule
```

32 位选择器

```
    module selector(
        input [31:0] iC0,
        input [31:0] iC1,
        input [31:0] iC2,
        input [31:0] iC3,
        input [31:0] iC4,
        input [31:0] iC5,
        input [31:0] iC6,
        input [31:0] iC7,
        input [31:0] iC8,
        input [31:0] iC9,
        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 iEna,
    output reg [31:0] oZ
);

    always @(*)
    begin
        if(iEna == 1)
            case(iS)
                5'b00000: oZ = iC0;
                5'b00001: oZ = iC1;
                5'b00010: oZ = iC2;
                5'b00011: oZ = iC3;
                5'b00100: oZ = iC4;
                5'b00101: oZ = iC5;
                5'b00110: oZ = iC6;
                5'b00111: oZ = iC7;
                5'b01000: oZ = iC8;
                5'b01001: oZ = iC9;
                5'b01010: oZ = iC10;
                5'b01011: oZ = iC11;
                5'b01100: oZ = iC12;
                5'b01101: oZ = iC13;
                5'b01110: oZ = iC14;
                5'b01111: oZ = iC15;
```

```
                5'b10000: oZ = iC16;
                5'b10001: oZ = iC17;
                5'b10010: oZ = iC18;
                5'b10011: oZ = iC19;
                5'b10100: oZ = iC20;
                5'b10101: oZ = iC21;
                5'b10110: oZ = iC22;
                5'b10111: oZ = iC23;
                5'b11000: oZ = iC24;
                5'b11001: oZ = iC25;
                5'b11010: oZ = iC26;
                5'b11011: oZ = iC27;
                5'b11100: oZ = iC28;
                5'b11101: oZ = iC29;
                5'b11110: oZ = iC30;
                5'b11111: oZ = iC31;
                default: oZ = {32{1'bz}};
            endcase
        else
            oZ = {32{1'bz}};
    end
endmodule
```

四、测试模块建模

1.双线输入输出 ram

```
`timescale 1ns / 1ps
module ram_tb
    #(
        parameter data_route = "E:/vivado_project/project_8/project_8.srcs/sim_1/new/data.txt"
    );
    reg clk, ena, wena;
    reg [4:0] addr;
    reg [5:0] i;
    reg [31:0] data_reset[31:0];
    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
```

```
        $readmemb(data_route, data_reset);
    initial
        clk = 1;

    always #10 clk = ~clk;
    initial
    begin
        ena <= 0; wena <= 1;
        #23 ena <= 1;
        for (i = 0; i < 32; i = i + 1)
        begin
            addr <= i[4:0]; data_in <= data_reset[i[4:0]];
            #20 ;
        end

        wena <= 0;
        #11 addr = 5'b10101;
        #22 addr = 5'b01011;
        #110 ena = 0;
    end
endmodule
```

2.单线输入输出 ram

```
module ram2_tb;
    reg clk;
    reg ena;
    reg wena;
    reg [4:0] addr;
    reg [31:0] data_in;
    wire [31:0] data;

    // Instantiate the RAM module
    ram2 uut (
        .clk(clk),
        .ena(ena),
        .wena(wena),
        .addr(addr),
        .data(data)
    );

    // Drive the bidirectional data bus
    assign data = (wena) ? data_in : 32'bz;

    // Clock generation
    always #5 clk = ~clk;
```

```
        initial begin
            // Initialize signals
            clk = 0;
            ena = 0;
            wena = 0;
            addr = 0;
            data_in = 0;

            // Test sequence
            #10;
            ena = 1;
            wena = 1;
            addr = 5;
            data_in = 32'hA5A5A5A5; // Write data to address 5
            #10;
            wena = 0; // Read data from address 5
            #10;
            $display("Read data: %h", data); // Should display A5A5A5A5

            #10;
            addr = 10;
            wena = 1;
            data_in = 32'h5A5A5A5A; // Write data to address 10
            #10;
            wena = 0; // Read data from address 10
            #10;
            $display("Read data: %h", data); // Should display 5A5A5A5A

            #10;
            $stop;
        end
    endmodule
```

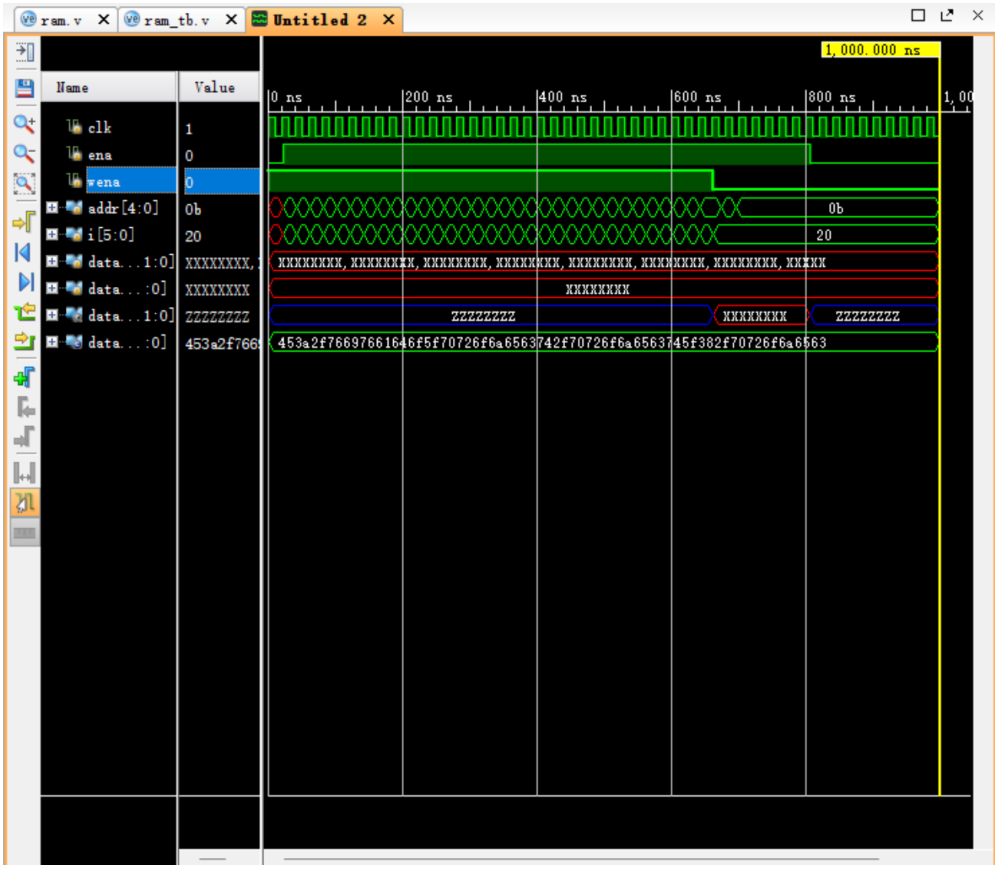
3.寄存器堆

```
    `timescale 1ns / 1ps
    module Regfiles_tb
        #(
            parameter                               data_route                              =
    "E:/vivado_project/project_8/project_8.srcs/sim_1/new/data.txt"
        );
        reg clk, rst, we;
        reg [4:0] raddr1, raddr2, waddr;
        reg [5:0] i;
        reg [31:0] data_reset[31:0];
```

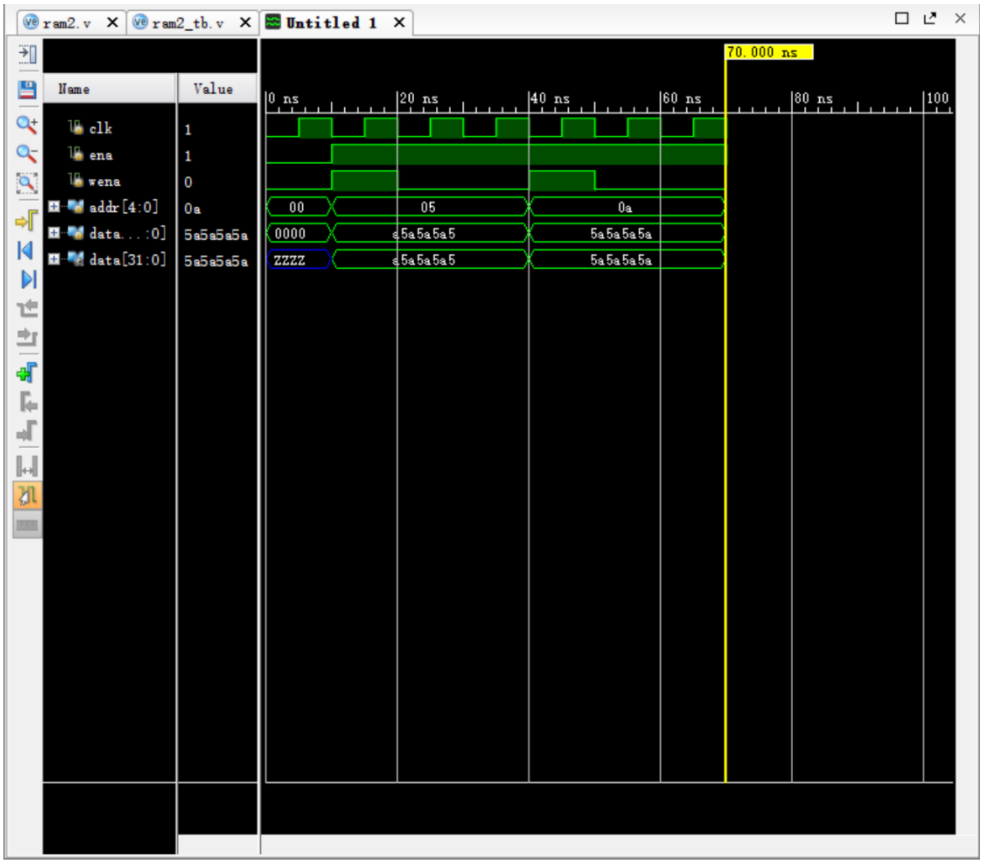
```
    reg [31:0] wdata;
    wire [31:0] rdata1, rdata2;

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

    initial
        $readmemb(data_route, data_reset);
    initial
        clk = 1;
    always #10 clk = ~clk;
    initial
    begin
        rst <= 1; we <= 1;
        #5
        rst <= 0; we <= 0;
        raddr1 <= 5'b10101; raddr2 <= 5'b01010;
        #10
        we = 1;
        for (i = 0; i < 32; i = i + 1)
        begin
            waddr <= i[4:0]; wdata <= data_reset[i[4:0]];
            #20 ;
        end

        #27 we <= 0;
        #50 rst <= 1; raddr1 <= 5'b10000; raddr2 <= 5'b00001;
    end
endmodule
```

五、实验结果

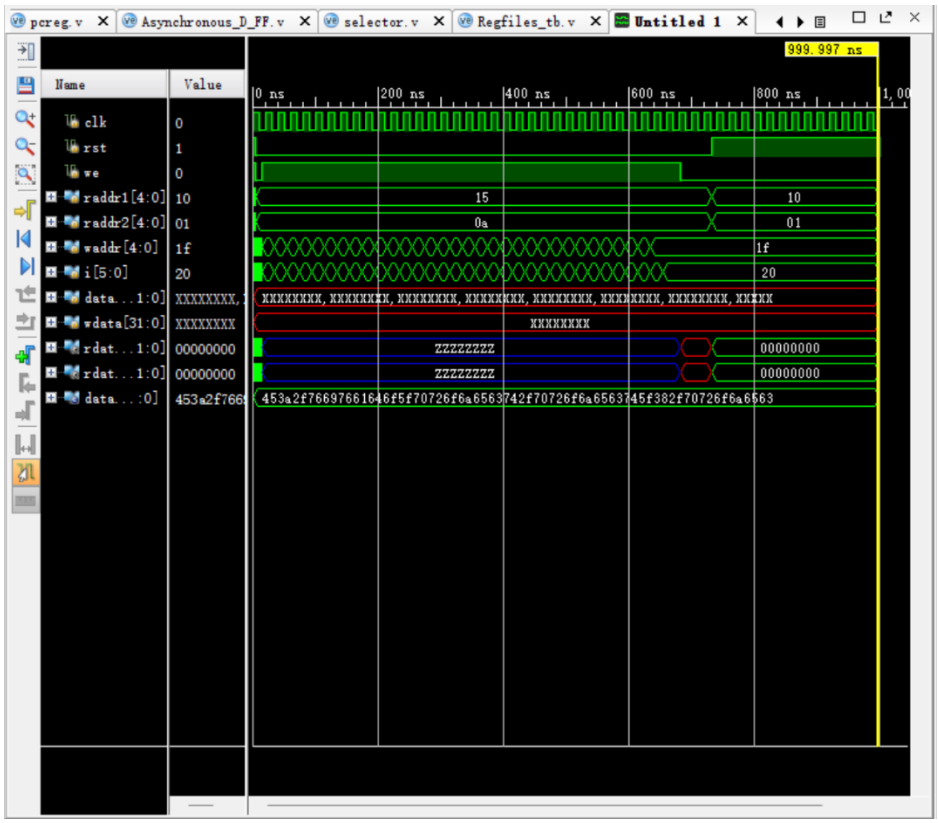
1. 双线输入输出 ram


ram.v
ram_tb.v
Untitled 2
Name
Value
clk
ena
wena
addr[4:0]
i[5:0]
data...1:0]
data...:0]
data...1:0]
data...:0]
1
0
0
0b
20
0b
20
zzzzzzzz
xxxxxxxx
zzzzzzzz
0 ns
200 ns
400 ns
600 ns
800 ns
1,000.000 ns

2. 单线输入输出 ram


ram2.v
ram2_tb.v
Untitled 1
Name
Value
clk
ena
wena
addr[4:0]
data...:0]
data[31:0]
1
1
0
0a
5a5a5a5a
5a5a5a5a
00
05
0a
0000
zzzz
0 ns
20 ns
40 ns
60 ns
80 ns
100
70.000 ns

3. 寄存器堆


pcreg.v
Asynchronous_D_FF.v
selector.v
Regfiles_tb.v
Untitled 1
999.997 ns
Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
clk
0
rst
1
we
0
raddr1[4:0]
10
15
10
raddr2[4:0]
01
0a
01
waddr[4:0]
1f
1f
i[5:0]
20
20
data...1:0]
xxxxxxxx, xxxxxxxx, xxxxxxxx, xxxxxxxx, xxxxxxxx, xxxxxxxx, xxxxxxxx, xxxxx
wdata[31:0]
xxxxxxxx
xxxxxxxx
rdat...1:0]
00000000
zzzzzzzz
00000000
rdat...1:0]
00000000
zzzzzzzz
00000000
data...:0]
453a2f7669766164 6f5f70726f6a6563742f70726f6a6563745f382f70726f6a6563