模型机设计报告

班级 计科 2004 姓名 钟永龙 学号 202008010419

一、设计目的

完整、连贯地运用《电路与电子学》所学到的数电知识，熟练掌握现代 EDA 工具基本使用方法，为后续课程学习和今后从事相关工作打下良好的基础或做下一些铺垫。

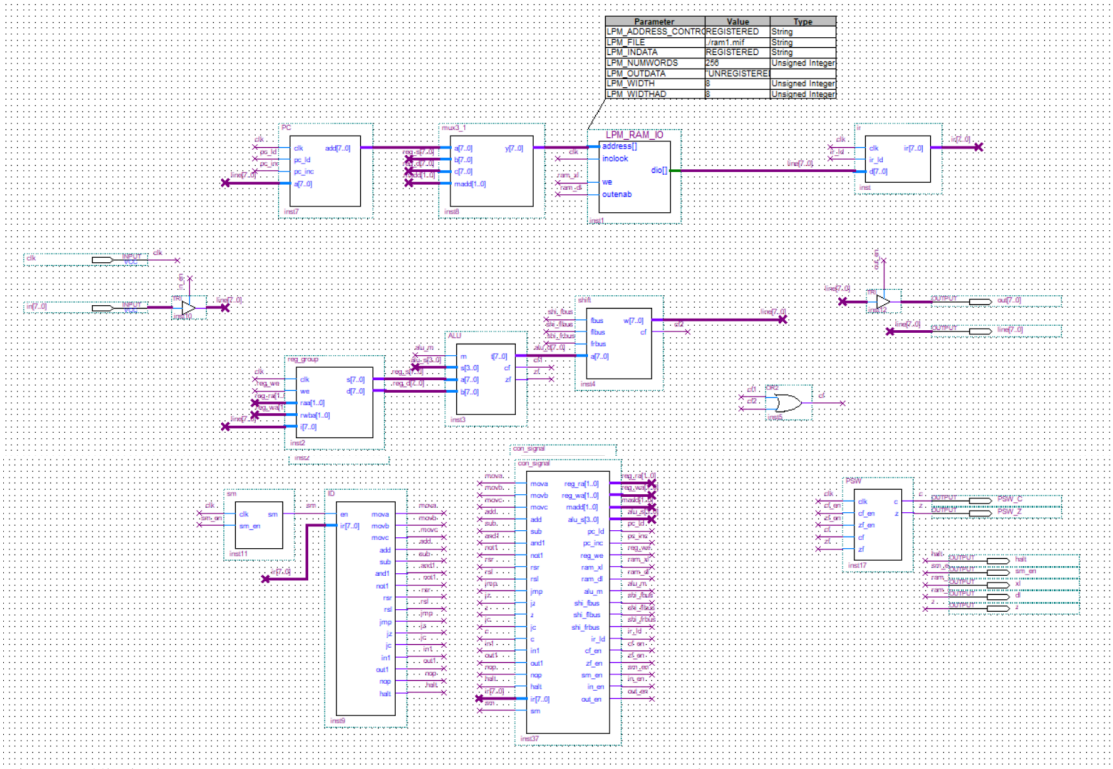
二、设计内容

1. 按照给定的数据通路、数据格式和指令系统，使用 EDA 工具设计一台用硬连线逻辑控制的简易计算机；
2. 对所设计计算机的性能指标进行分析，整理出设计报告

三、详细设计

3.1 设计的整体架构

利用之前实验设计好的模块，创建其符号，利用导线将其连接起来，加入必要的三态门、或门，以解决总线冲突，通过调试，实现正常的功能。如图所示：



涉及的模块有：SM、IR、RAM、PC、ALU、PSW、指令译码器、三选一多路复用器、通用寄存器组、移位逻辑、控制逻辑、

涉及的输入有：CLK 时钟输入、IN 数据输入

涉及的输出有：OUT 数据的输出

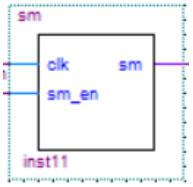
3.2 各模块的具体实现

（1） 时序部分

1) SM

接口设计：输入：CLK，sm_en

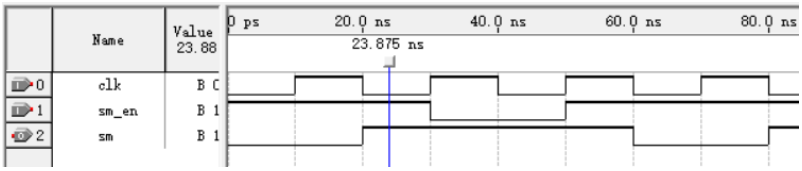
输出：sm



实现如下：

```
module sm(clk, sm_en, sm);
    input clk, sm_en;
    output sm;
    reg sm=1'b_0;

    always @(negedge clk)
    begin
        if(sm_en)
            sm = !sm;
    end
endmodule
```

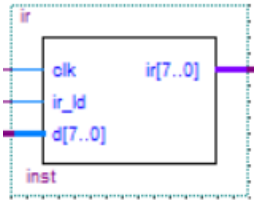
功能仿真如下：



2) ir

接口设计：输入：clk，ir_ld，d

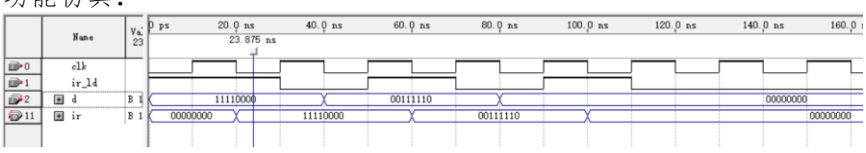
输出：ir



实现如下：

```
module ir(clk, ir_ld, d, ir);
    input clk, ir_ld;
    input [7:0] d;
    output [7:0] ir;
    wire [7:0] d;
    reg [7:0] ir;

    always @(negedge clk)
    begin
        if(ir_ld)
            ir = d;
    end
endmodule
```

功能仿真：

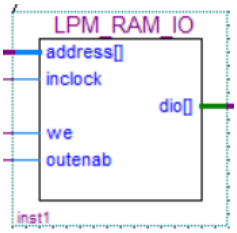


3) RAM

接口设计：调用参数化的元件，通过调节设计如下：

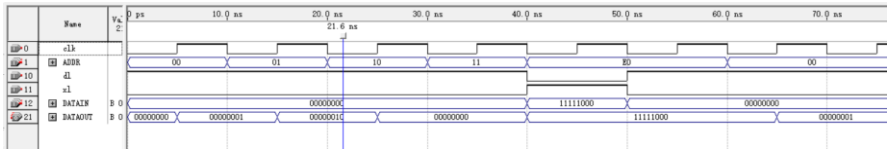
输入：address、inclock、we、outenab、dio

输出：dio



实现：调用参数化元件

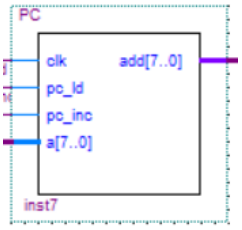
功能仿真如下：



4) PC

接口设计：输入：clk，pc_ld、pc_inc、a

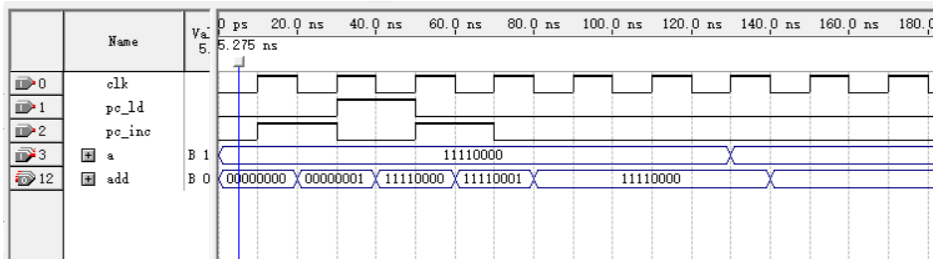
输出：add



实现如下：

```
module PC(clk, pc_ld, pc_inc, a, add);
    input clk, pc_ld, pc_inc;
    input [7:0] a;
    output [7:0] add;
    wire [7:0] a;
    reg [7:0] add;

    initial
    begin
        add = 8'b_0000_0000;
    end

    always @ (negedge clk)
    begin
        if(pc_inc    && !pc_ld)
            add = add + 1'b1;
        else if(!pc_inc && pc_ld)
            add = a;
    end
endmodule
```

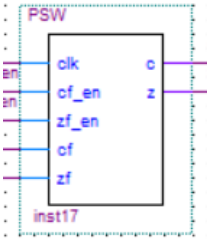
功能仿真：



5) PSW

接口设计：输入：clk、cf_en、zf_en、cf、zf

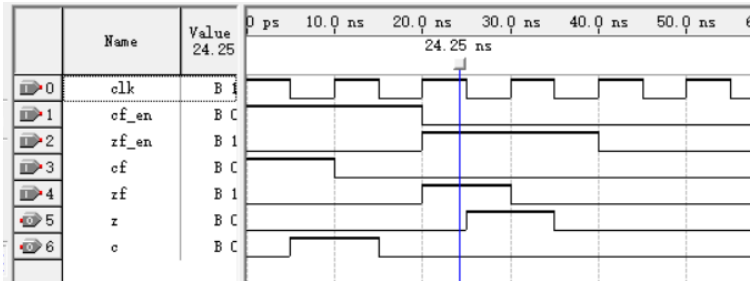
输出：c、z



实现：

```
module PSW(clk, cf_en, zf_en, cf, zf, c, z);
    input clk, cf_en, zf_en;
    input cf, zf;
    output c, z;
    reg c, z;

    always @ (negedge clk)
    begin
        if(cf_en)
            c <= cf;
        if (zf_en)
            z <= zf;
    end
endmodule
```

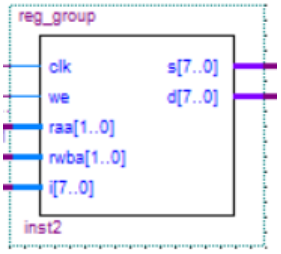
功能仿真：



6) 通用寄存器组

接口设计：输入：clk、we、raa、rwba、i

输出：s、d

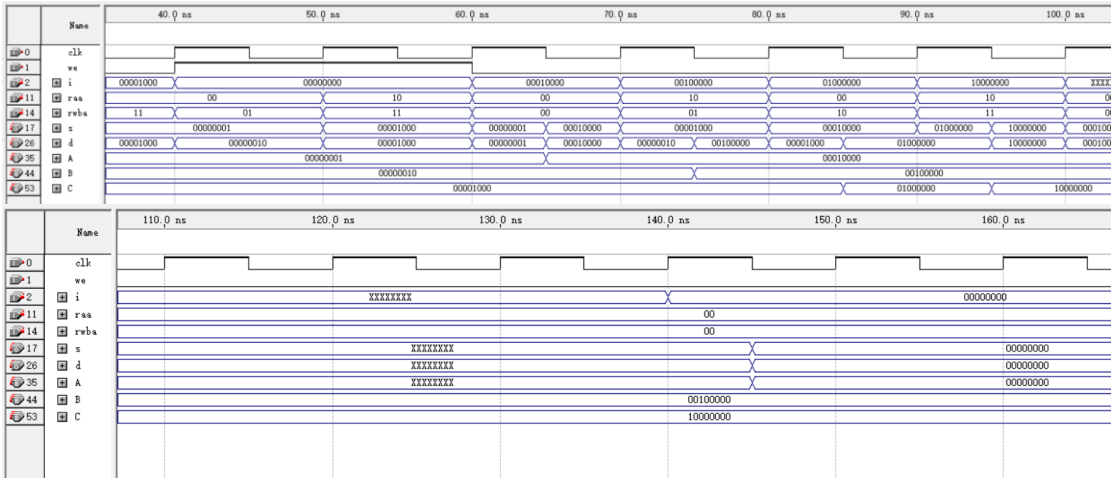


实现：

```
module reg_group(clk, we, raa, rwba, i, s, d);
    input clk, we;
    input [7:0] raa, rwba, i;
    output [7:0] s, d;
    wire [1:0] raa, rwba;
    wire [7:0] i;
    reg [7:0] s, d;
    reg [7:0] A, B, C;

    initial
    begin
```

```
            A = 8'b_0000_0000;
            B = 8'b_0000_0000;
            C = 8'b_1000_0000;
        end

        always @(negedge clk)
        begin
            if(!we)
            begin
                case(rwba)
                    2'b00: A <= i;
                    2'b01: B <= i;
                    default C <= i;
                endcase
            end
        end

        always @(raa or rwba or A or B or C)
        begin
            case(raa)
                2'b00: s = A;
                2'b01: s = B;
                default s = C;
            endcase
            case(rwba)
                2'b00: d = A;
                2'b01: d = B;
                default d = C;
            endcase
        end
    endmodule
```

功能仿真：



（2） 组合部分

1) ALU

接口设计：输入：m、s、a、b

输出：t、cf、zf

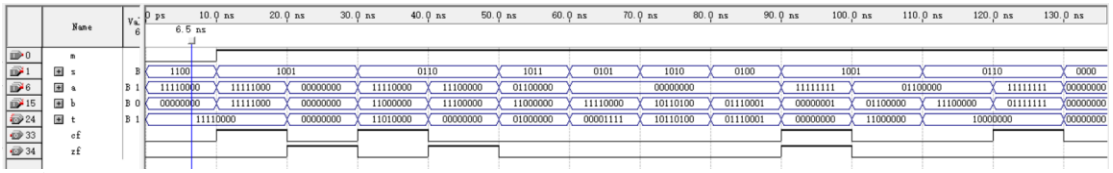
实现：

```
module ALU(m, s, a, b, t, cf, zf);
    input m;
    input [3:0] s;
    input [7:0] a;
```

```
    input [7:0] b;
    output [7:0] t;
    output cf;
    output zf;
    reg [7:0] t;
    reg cf, zf;

always @(m, s, a, b, t, cf, zf)
begin
    cf = 0;
    zf = 0;
    t = 8'b00000000;
    if(m==0)
    begin
        t=a;
    end
    else
        case(s)
            4'b1001:begin
                {cf, t} = {1'b0, a} + {1'b0, b};
                if(t == 0)
                begin
                    zf = 1;
                end
                else
                begin zf = 0; end
            end
            4'b0110:begin
                {cf, t} = {1'b1, b} - {1'b0, a};
                cf = !cf;
                if(t == 0)
                begin
                    zf = 1;
                end
                else
                begin zf = 0; end
            end
            4'b1011:begin
                t = a & b;
            end
            4'b0101:begin
                t = ~b;
            end
            4'b1010:begin
                t = b;
            end
            4'b0100:begin
                t = b;
            end
            4'B1100:begin
                t = a;
            end
            default:begin
                t = 0;
                cf = 0;
                zf = 0;
            end
```

```
        endcase
end
endmodule
```

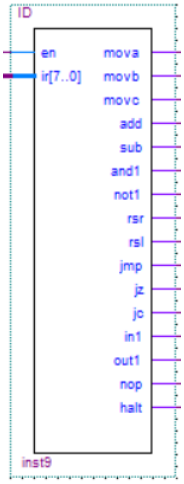
功能仿真：



2) 指令译码器

接口设计：输入：en、ir

输出：mova、movb、movc、add、sub、and1、not1、rsr、rsl、jmp、jz、jc、in1、out1、nop、halt

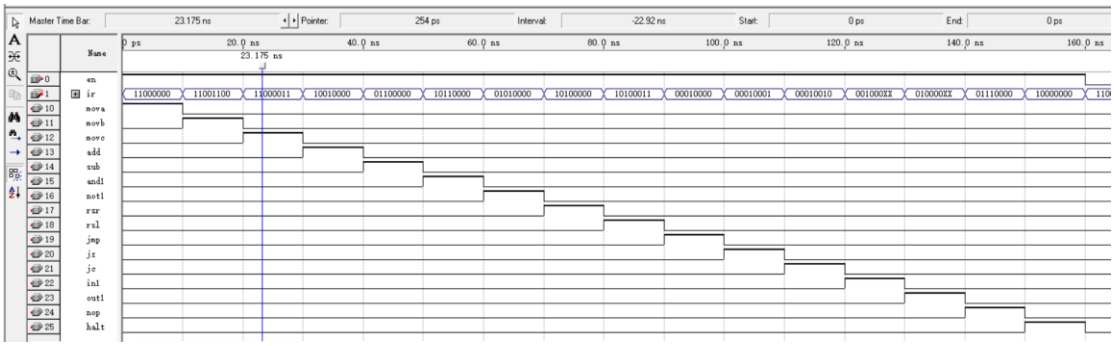


实现：

```
module ID(en, ir, mova, movb, movc, add, sub,
        and1, not1, rsr, rsl, jmp, jz, jc,
        in1, out1, nop, halt);
    input en;
    input [7:0] ir;
    output mova, movb, movc, add, sub,
        and1, not1, rsr, rsl, jmp, jz, jc,
        in1, out1, nop, halt;
    wire [7:0] ir;
    reg mova, movb, movc, add, sub,
        and1, not1, rsr, rsl, jmp, jz, jc,
        in1, out1, nop, halt;


    always @(en, ir, mova, movb, movc, add, sub,
        and1, not1, rsr, rsl, jmp, jz, jc,
        in1, out1, nop, halt)
    begin
        mova=1'b0; movb=1'b0; movc=1'b0; add=1'b0; sub=1'b0;
        and1=1'b0; not1=1'b0; rsr=1'b0; rsl=1'b0; jmp=1'b0;
        jz=1'b0; jc=1'b0; in1=1'b0; out1=1'b0; nop=1'b0; halt=1'b0;
        if(en!=0)
            case(ir[7:4])
                4'b1100:begin
                            if(ir[3:2]==2'b11)
                            begin movb=1'b1;
                            end
```

```
                    else if(ir[1:0]==2'b11)
                    begin movc=1'b1;
                    end
                    else
                    begin mova=1'b1;
                    end
                end
            4'b1010:begin
                    if(ir[1:0]==2'b00)
                    begin rsr=1'b1;
                    end
                    else
                    begin rsl=1'b1;
                    end
                end
                4'b0001:begin
                            if (ir[3:0]==4'b0000)
                            begin jmp=1'b1;
                            end
                            else if(ir[3:0]==4'b0001)
                            begin jz=1'b1;
                            end
                            else
                            begin jc=1'b1;
                            end
                        end
            4'b1001:begin add=1'b1;
                end
            4'b0110:begin sub=1'b1;
                end
            4'b1011:begin and1=1'b1;
                end
            4'b0101:begin not1=1'b1;
                end
            4'b0010:begin in1=1'b1;
                end
            4'b0100:begin out1=1'b1;
                end
            4'b0111:begin   nop=1'b1;
                end
            4'b1000:begin halt=1'b1;
                end
            default:begin
                end
        endcase
    else
    begin
    end
end
endmodule
```

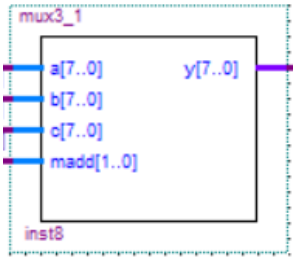
功能仿真：



3) 三选一多路复用器

接口设计：输入：a、b、c、madd

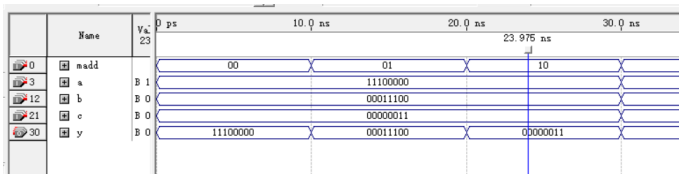
输出：y



实现：

```
module mux3_1(a, b, c, madd, y);
    input [7:0]a;
    input [7:0]b;
    input [7:0]c;
    input [1:0]madd;
    output [7:0]y;
    reg [7:0]y;
    always@(*)
    begin
        case(madd)
            2'b00:y=a;
            2'b01:y=b;
            2'b10:y=c;
            default:y=8'b00000000;
        endcase
    end
endmodule
```

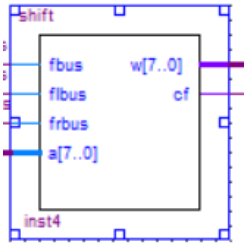
功能仿真：



4) 移位逻辑

接口设计：输入：fbus、flbus、frbus、a

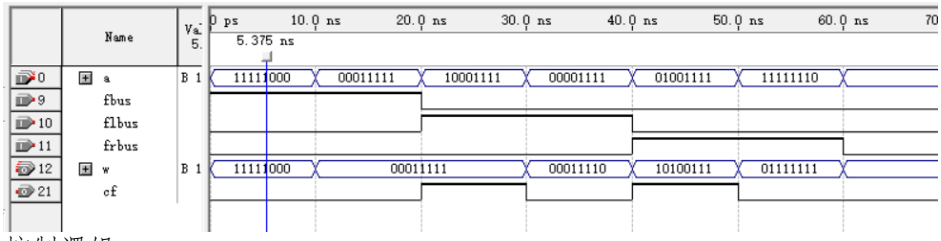
输出：w、cf



实现：

```
module shift(fbus, flbus, frbus, a, w, cf);
    input fbus, flbus, frbus;
    input [7:0]a;
    output [7:0]w;
    output cf;
    reg [7:0]w;
    reg cf;

    always @(*)
    begin
        if(fbus == 1 && flbus == 0 && frbus == 0)
        begin
            w = a;
            cf = 1'bx;
        end
        else if(fbus == 0 && flbus == 1 && frbus == 0)
        begin
            w = {a[6:0], a[7]};
            cf = a[7];
        end
        else if(fbus == 0 && flbus == 0 && frbus == 1)
        begin
            w = {a[0], a[7:1]};
            cf = a[0];
        end
        else
        begin
            w = 8'bzzzzzzzz;
            cf = 1'b0;
        end
    end
endmodule
```

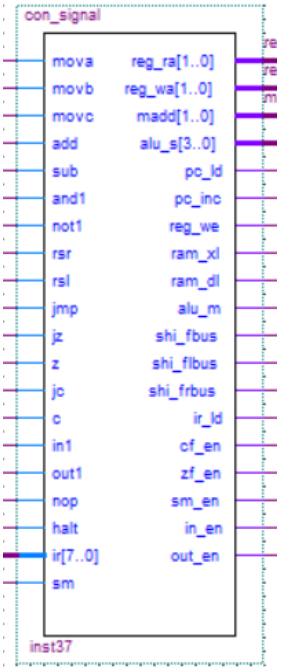
功能仿真：



5) 控制逻辑

接口设计：输入：mova、movb、movc、add、sub、and1、not1、rsr、rsl、jmp、jz、z、jc、c、in1、out1、nop、halt、ir、sm

输出：reg_ra、reg_wa、madd、alu_s、pc_ld、pc_inc、reg_we、ram_xl、ram_dl、alu_m、shi_fbus、shi_frbus、shi_flbus、ir_ld、cf_en、zf_en、sm_en、in_en、out_en

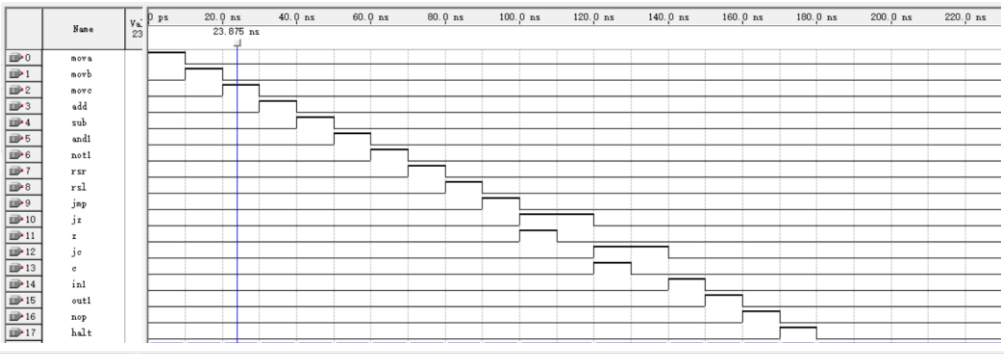


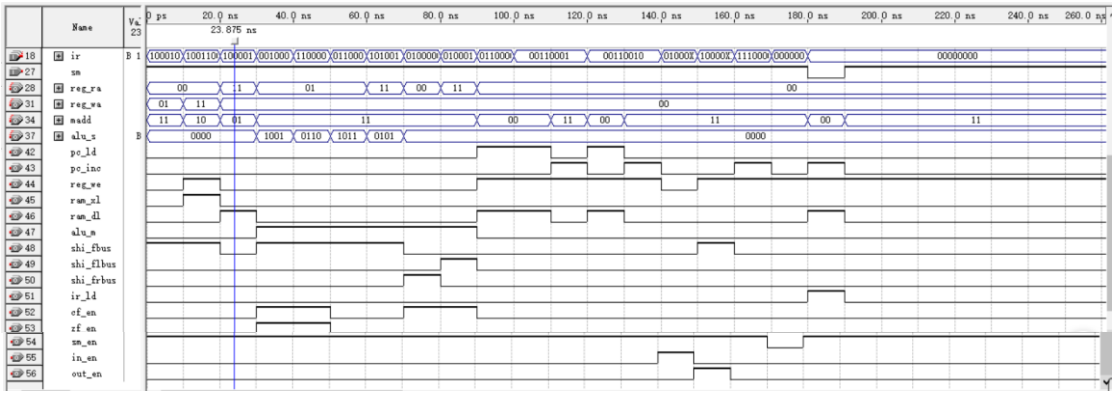
实现：

```
module con_signal (mova, movb, movc, add, sub, and1, not1,
                   rsr, rsl, jmp, jz, z, jc, c, in1, out1,
                   nop, halt, ir, sm, reg_ra, reg_wa, madd,
                   alu_s, pc_ld, pc_inc, reg_we, ram_xl,
                   ram_dl, alu_m, shi_fbus, shi_flbus,
                   shi_frbus, ir_ld, cf_en, zf_en, sm_en,
                   in_en, out_en);
    input mova, movb, movc, add, sub, and1, not1, rsr, rsl,
          jmp, jz, z, jc, c, in1, out1, nop, halt, sm;
    input [7:0]ir;
    output [1:0]reg_ra;
    output [1:0]reg_wa;
    output [1:0]madd;
    output [3:0]alu_s;
    output pc_ld, pc_inc, reg_we, ram_xl, ram_dl, alu_m, shi_fbus,
           shi_flbus, shi_frbus, ir_ld, cf_en, zf_en, sm_en, in_en, out_en;
    reg [1:0]reg_ra;
    reg [1:0]reg_wa;
    reg [1:0]madd;
    reg [3:0]alu_s;
    reg pc_ld, pc_inc, reg_we, ram_xl, ram_dl, alu_m, shi_fbus,
        shi_flbus, shi_frbus, ir_ld, cf_en, zf_en, sm_en, in_en, out_en;
    always @ (*)
    begin
        reg_ra = ir[1:0];
        reg_wa = ir[3:2];
        madd = 2'b00;
        alu_s = ir[7:4];
        pc_ld = 1'b0;
        pc_inc = 1'b0;
        ram_xl = 1'b0;
        ram_dl = 1'b0;
        alu_m = 1'b0;
        shi_fbus = 1'b0;
        shi_flbus = 1'b0;
```

```
            shi_frbus = 1'b0;
            ir_ld = 1'b0;
            cf_en = 1'b0;
            zf_en = 1'b0;
            sm_en = 1'b1;
            in_en = 1'b0;
            out_en = 1'b0;
            reg_we = 1'b1;
//-------------------------------------------------------------
      //reg_we
      reg_we = !(mova || movc || add || sub || and1 || not1 || rsr || rsl || in1);
      //alu_m
      alu_m = (add || sub || and1 || not1 || rsr || rsl || out1);
      //shi_fbus
      shi_fbus =(mova || movb || add || sub || and1 || not1 || out1);
      //shi_flbus
      shi_flbus = (rsl);
      //shi_frbus
      shi_frbus = (rsr);
      //pc_ld
      pc_ld = (jmp || (jz && z) || (jc && c));
      //pc_inc
      pc_inc = (!sm || (jz && !z) || (jc && !c));
      //ram_xl
      ram_xl = (movb);
      //ram_dl
      ram_dl = (!sm || movc || jmp || (jz && z) || (jc && c));
      //madd
      if (!sm || jmp || (jz && z) || (jc && c))
      begin
            madd = 2'b_00;
      end
      else if (movc)
      begin
            madd = 2'b_01;
      end
      else if(movb)
      begin
            madd = 2'b_10;
      end
      //ir_ld
      ir_ld = (!sm);
      //cf_en
      cf_en = (add || sub || rsr || rsl);
      //zf_en
      zf_en = (add || sub);
      //sm_en
      sm_en = !(halt);
      //in_en
      in_en = (in1);
      //out_en
      out_en = (out1);
//-------------------------------------------------------------
      end
endmodule
```

功能仿真：





四、 系统测试

4.1 测试环境

Windows10

QuartusII version:13 64-Bit family=Cyclone；name=EP2C5T144C8

4.2 测试代码

（使用模型机实现的指令编写一至两个程序测试模型机的正确性。）

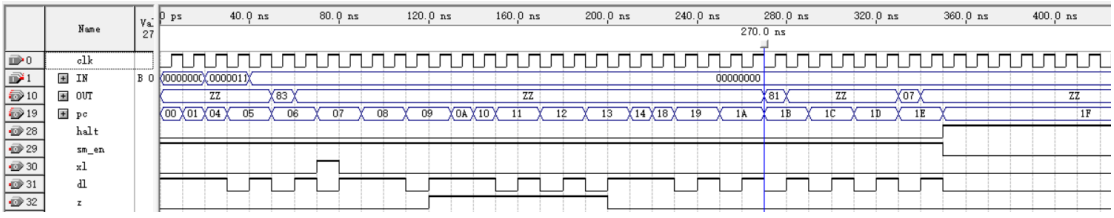
存储在 RAM 中的数据：

00010000	00000100	00000000	00000000	00100000	01000000	11001100	11000111
01100001	00010001	00010000	00010000	00000000	00000000	00000000	00000000
01010000	10110010	10010001	00010010	00011000	00011000	00000000	00000000
01110000	10100000	01000000	10100111	11000001	01000000	10000000	10010001
01000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

IN 输入详见测试结果处的仿真波形

4.3 测试结果

结果如图所示：



4.4 模型机性能分析

Flow Status	Successful - Sun Jan 02 23:43:14 2022
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	top
Top-level Entity Name	top
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	242 / 4,608 (5 %)
Total combinational functions	241 / 4,608 (5 %)
Dedicated logic registers	43 / 4,608 (< 1 %)
Total registers	43
Total pins	32 / 89 (36 %)
Total virtual pins	0
Total memory bits	2,048 / 119,808 (2 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

成功执行指令最小时钟周期 27.948ns

五、 实验总结、必得体会及建议

5.1 从需要掌握的理论、遇到的困难、解决的办法以及经验教训等方面进行总结。

在写各个小模块时存在一定的缺陷，使得在最后合成时存在一点的困难，但是在老师的指导下，修改不完善的地方，最终完成了试验任务。在此过程中最难的一点便是确定错误发生的地方，在发现错误后，修改便是较为简单的事情，尤其是总线冲突较为难发现错误发生的地方。

在此次实验中，我较为熟练地掌握了 quartus 的操作使用，学习了解了许多设计方法与思想，满足了我理论上实现 CPU 的幻想。