实验三 简易模型机中组合部件的实现

班级 计科 1907 姓名 杨杰 学号 201908010705

一、实验目的

1．了解简易模型机的内部结构和工作原理。
2．分析模型机的功能，设计 ALU 和移位逻辑。
3．分析模型机的工作原理，设计模型机控制信号产生逻辑

二、实验内容

1．用 VHDL 语言设计模型机的 ALU 模块;
2．用 VHDL 语言设计模型机的移位模块;
3．用 VHDL 语言设计模型机的控制信号产生逻辑。

三、实验方法

1、

实验方法

采用基于 FPGA 进行数字逻辑电路设计的方法。
采用的软件工具是 Quartus II。

2、

实验步骤

1、 新建，编写源代码。
(1).选择保存项和芯片类型：【File】-【new project wizard】-【next】-【next】（-【properties】（type=AHDL）-【next】（family=FLEX10K; name=EPF10K10TI144-4）-【next】-【finish】
(2).新建：【file】-【new】（第二个 AHDL File）-【OK】
2、写好源代码，保存文件
3、编译与调试。确定源代码文件为当前工程文件，点击【processing】-【start compilation】进行文件编译，编译成功。
4、波形仿真及验证。新建一个 vector waveform file。
5、时序仿真或功能仿真。
6、查看 RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

四、实验过程

ALU：

1、编译过程

a）源代码如图（VHDL 设计）

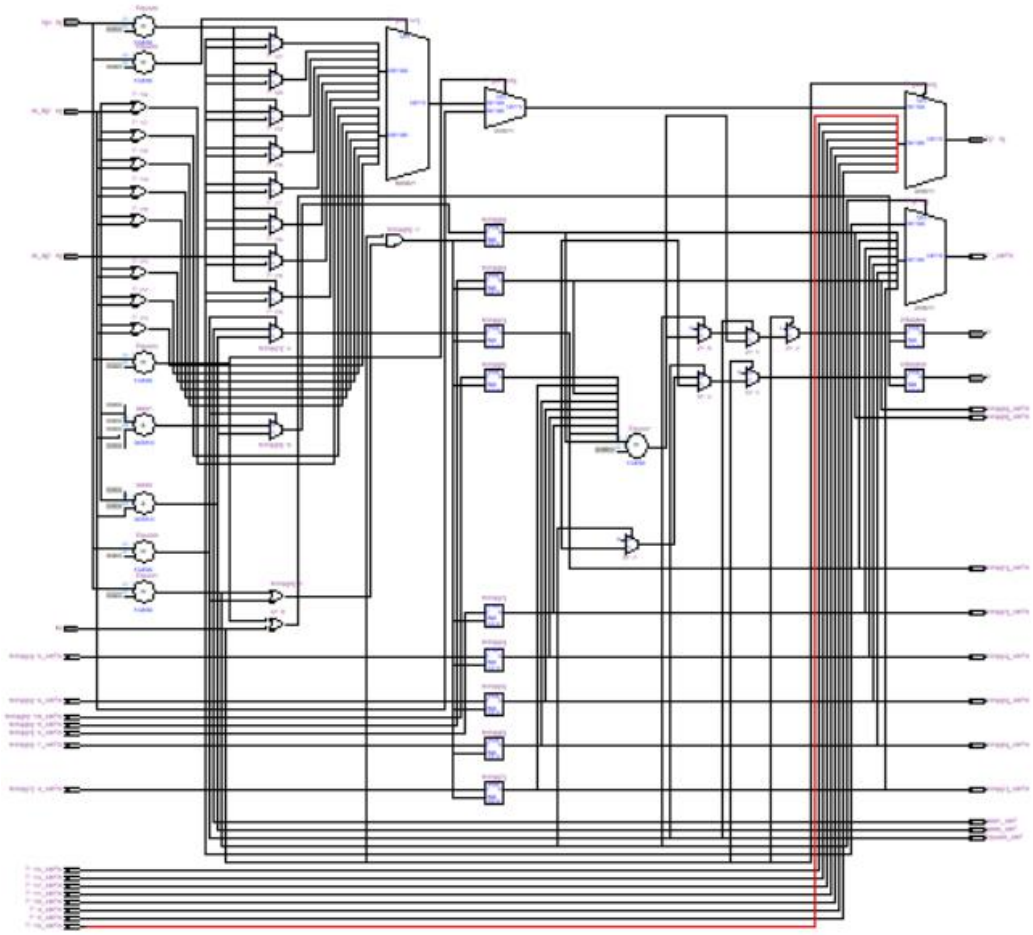
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity ALU is
    port (in_A,in_B:in std_logic_vector(7 downto 0);
          S:in std_logic_vector(3 downto 0);
          M:in std_logic;
          Cf,Zf:out std_logic;
          T:out std_logic_vector(7 downto 0));
end ALU;
architecture ALU_arc of ALU is
signal temp,x1,x2:std_logic_vector(8 downto 0):="000000000";
    begin
        x1<='0' & in_A;
        x2<='0' & in_B;
       process(in_A,in_B,S,M)
       begin
        if(M='0') then
            if(S="1001")then--add
                temp<=x1+x2;
                Cf<=temp(8);
                T<=temp(7 downto 0);
                if(temp="000000000") then Zf<='1';
                else Zf<='0';
                end if;
            elsif(S="0110")then--sub
                temp<=x1-x2;
                Cf<=temp(8);
                T<=temp(7 downto 0);
                if(temp="000000000") then Zf<='1';
                else Zf<='0';
                end if;
```

```
            else
                T<=in_A;
                Zf<='0';
                Cf<='0';
            end if;
        else
            if(S="1010") then
              T<=in_B;
            elsif(S="1011")then--or
                T<=in_A or in_B;
                Zf<='0';
                Cf<='0';
            elsif(S="0101")then--not
                T<= not in_A;
                Zf<='0';
                Cf<='0';
            else
                T<=in_A;
                Zf<='0';
                Cf<='0';
            end if;
        end if;
        end process;
end ALU_arc;
```

b)编译、调试过程

编译成功

c) RTL 视图

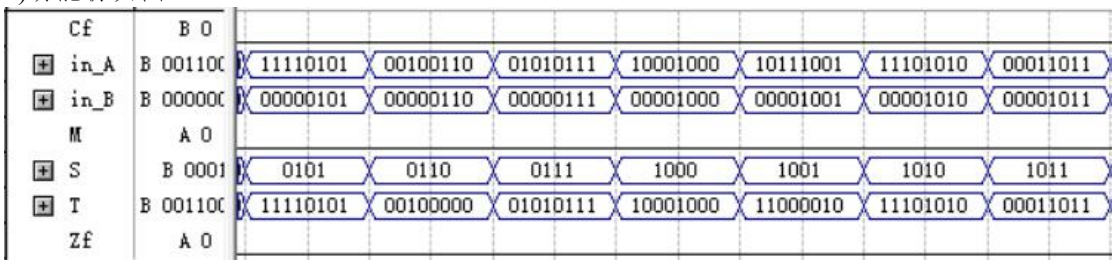


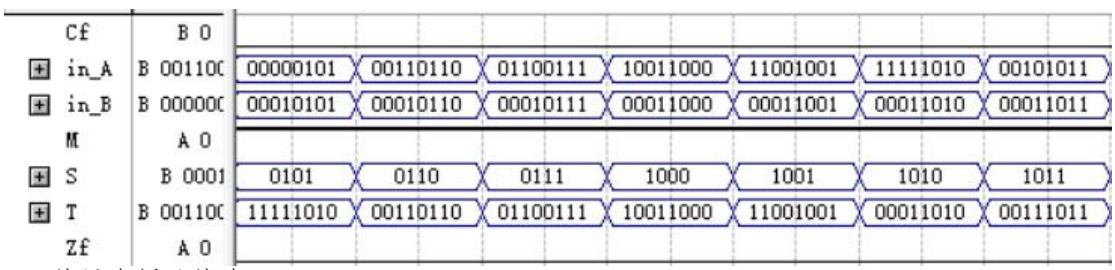
2、 功能仿真

a) 功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

b)功能仿真图





c) 结果分析及结论

当 M=0 时，进行算术运算或者直通

S=1001 执行加法运算 in_A=11001001，in_B=00011010，输出 T=11001001，Cf=0，Zf=0;
S=0110 执行减法运算 in_A=00110110，in_B=00010110，输出 T=00110110，Cf=0，Zf=0;
S 的其余情况为直通，T=in_A

当 M=1 时，进行逻辑运算

S=1011 执行或运算 in_A=00101011，in_B=00011011，输出 T=00111011，Cf=0，Zf=0;
S=0101 执行非运算 in_A=00000101，in_B=00010101，输出 T=11111010，Cf=0，Zf=0;
S=1010 T=in_B
其余情况 T=in_A;

结果正确

移位逻辑：

1、 编译过程

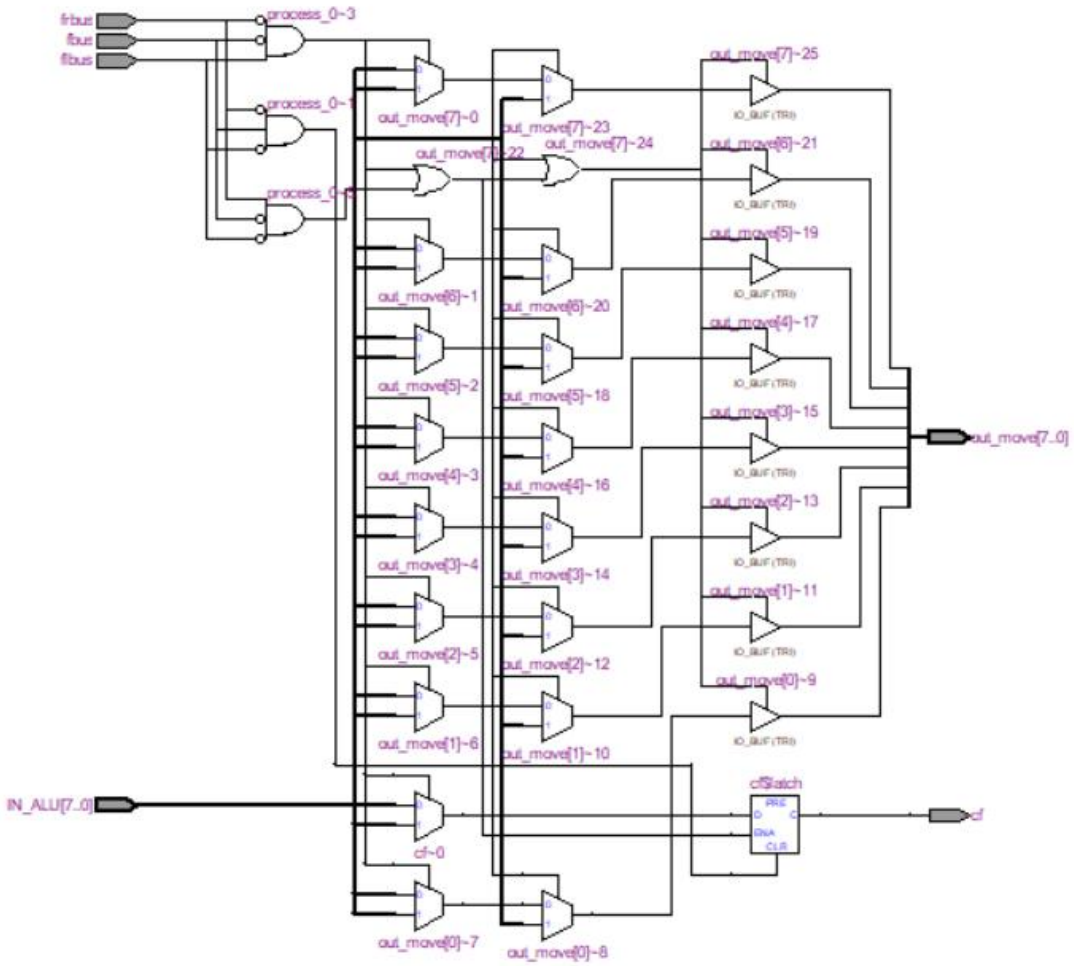
a）源代码如图（VHDL 设计）

```
library ieee;
use ieee.std_logic_1164.all;

entity Logic_move is
    port (
        fbus,frbus,flbus : in std_logic;
        IN_ALU: in STD_LOGIC_VECTOR(7 downto 0);
        out_move: out STD_LOGIC_VECTOR(7 downto 0);
        cf : out std_logic:='0'
    );
end entity Logic_move;

architecture Logic_move_arc of Logic_move is
begin
process(fbus,frbus,flbus)
    begin
    if(fbus='1' and flbus='0' and frbus='0') then
        out_move<=IN_ALU;
        cf<='0';
    elsif(fbus='0' and flbus='1' and frbus='0') then
            out_move<=IN_ALU(6 downto 0)&IN_ALU(7);
            cf<=IN_ALU(7);
    elsif(fbus='0' and flbus='0' and frbus='1') then
            out_move<=IN_ALU(0) & IN_ALU(7 downto 1);
            cf<=IN_ALU(0);
    else
        out_move<="ZZZZZZZZ";
    end if;
end process;
end architecture Logic_move_arc;
```

b)编译、调试过程

编译成功

c) RTL 视图

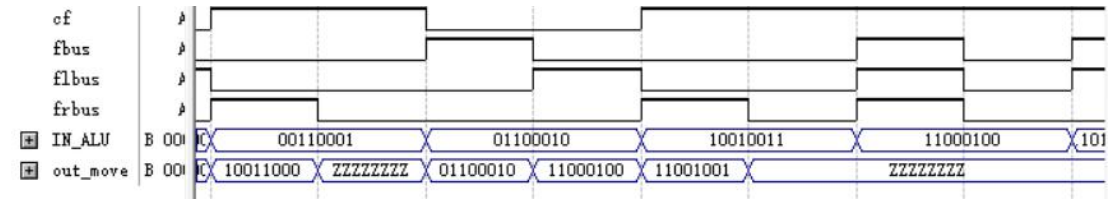


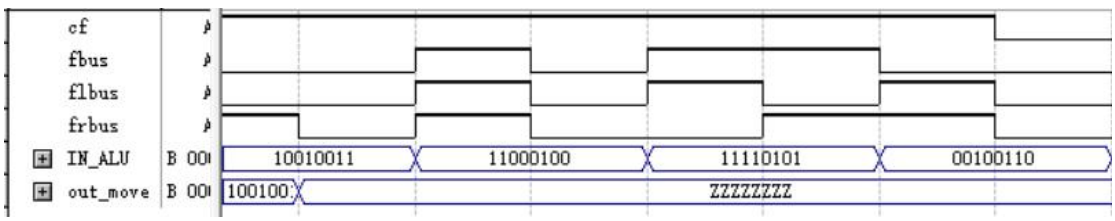
2、 功能仿真

a) 功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

b)功能仿真图





c) 结果分析及结论

Fbus、flbus、frbus 只有一个为 1 时，才有效
当 fbus=1，flbus=0，frbus=0,时为直通，IN_ALU=01100010，out_move=01100010
当 fbus=0，flbus=1，frbus=0,时左移，IN_ALU=01100010,out_move=11000100
当 fbus=0，flbus=0，frbus=1,时右移，IN_ALU=00110001,out_move=10011000
其余情况均为无效状态，输出为高阻“zzzzzzzz”
结果正确

控制信号产生逻辑：

1、 编译过程

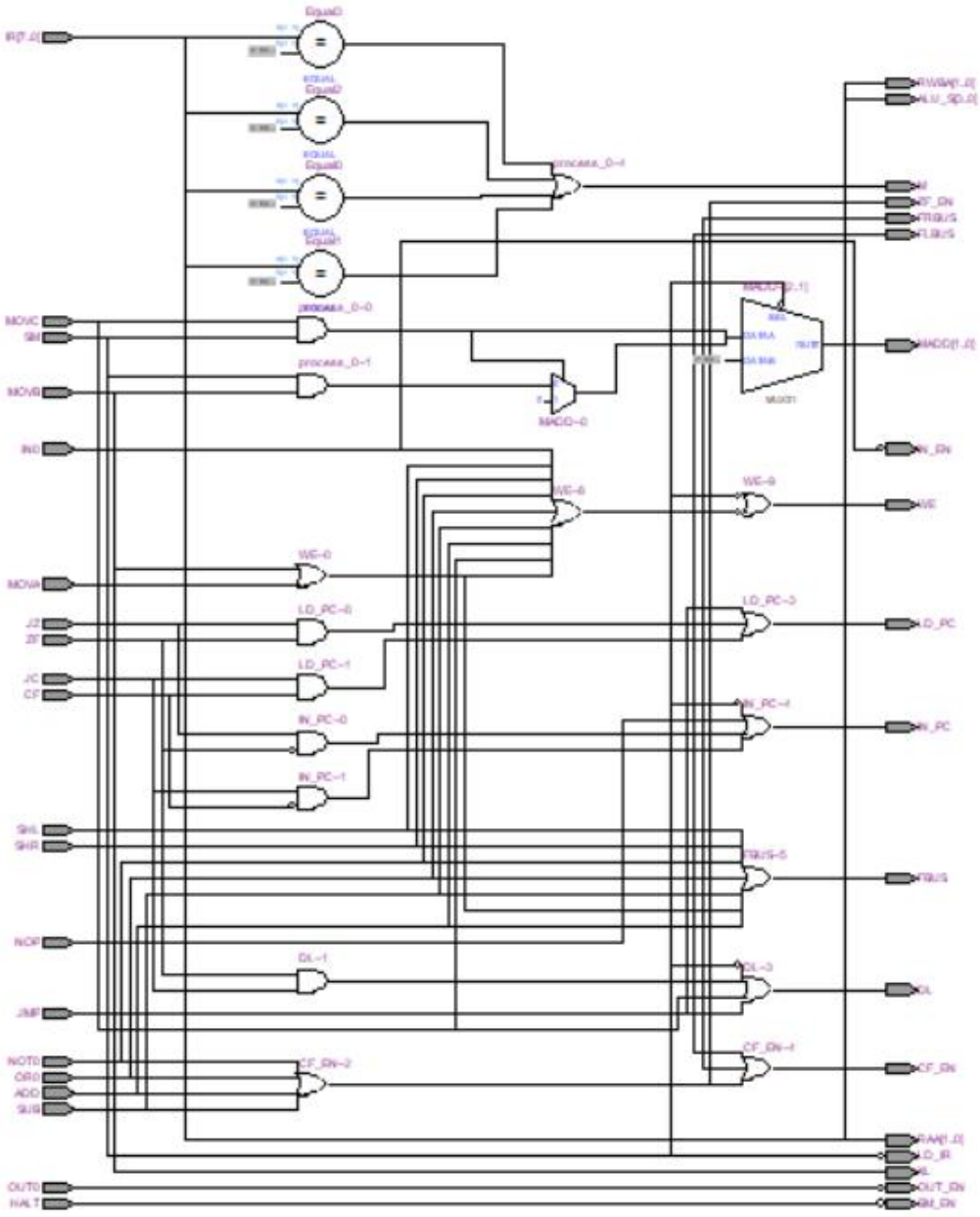
```
library ieee;
use ieee.std_logic_1164.all;

entity Controller is
port (
        MOVA,MOVB,MOVC,ADD,SUB,OR0,NOT0,SHR,SHL,JMP,JZ,
        JC,IN0,OUT0,NOP,HALT,SM,CF,ZF: IN STD_LOGIC;
        IR : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        RAA,RWBA,MADD : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
        ALU_S: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        WE,M,FBUS,FRBUS,FLBUS,LD_PC,IN_PC,XL,DL,LD_IR,
        CF_EN,ZF_EN,IN_EN,OUT_EN,SM_EN : OUT STD_LOGIC
    );
end entity Controller;

architecture Controller_arc of Controller is

begin
process(MOVA,MOVB,MOVC,ADD,SUB,OR0,NOT0,SHR,SHL,
    JMP,JZ,JC,IN0,OUT0,NOP,HALT,SM,CF,ZF,IR)
    begin
    LD_IR<=not SM;
    LD_PC<=(JZ AND ZF) OR (JC AND CF) OR JMP;
    IN_PC<=(JZ AND (NOT ZF)) OR (JC AND (NOT CF)) OR NOP OR (NOT SM);
    WE<=not(MOVA or MOVB or MOVC or ADD or SUB or OR0 or
     NOT0 or SHR or SHL or IN0) or not(SM);
    RAA<=IR(1 DOWNTO 0);
    RWBA<=IR(3 DOWNTO 2);
    if SM='0' then
        MADD<="00";
    elsif(SM='1' AND MOVC='1') then
        MADD<="01";
    elsif(SM='1' AND MOVB='1') then
        MADD<="10";
```

```
    else MADD<="00";
    end if;

    ALU_S<=IR(7 DOWNTO 4);
    XL<=MOVB;
    DL<=MOVC OR JMP OR (ZF AND JC) OR (NOT SM);

    if((IR(7 DOWNTO 4)="1001" ) OR (IR(7 DOWNTO 4)="0110" )
        OR (IR(7 DOWNTO 4)="1011" ) OR (IR(7 DOWNTO 4)="0101" )) then
        M<='1';
    else M<='0';
    end if;
    FBUS<=MOVA or MOVB or ADD or SUB or OR0 or NOT0 or SHR or SHL;
    FLBUS<=SHL;
    FRBUS<=SHR;
    CF_EN<=ADD or SUB or OR0 or NOT0 or SHR or SHL;
    ZF_EN<=ADD or SUB or OR0 or NOT0;
    SM_EN<= NOT HALT;
    IN_EN<= NOT IN0;
    OUT_EN <= NOT OUT0;
    end process;
end architecture Controller_arc;
```

b)编译、调试过程

编译成功

c) RTL 视图

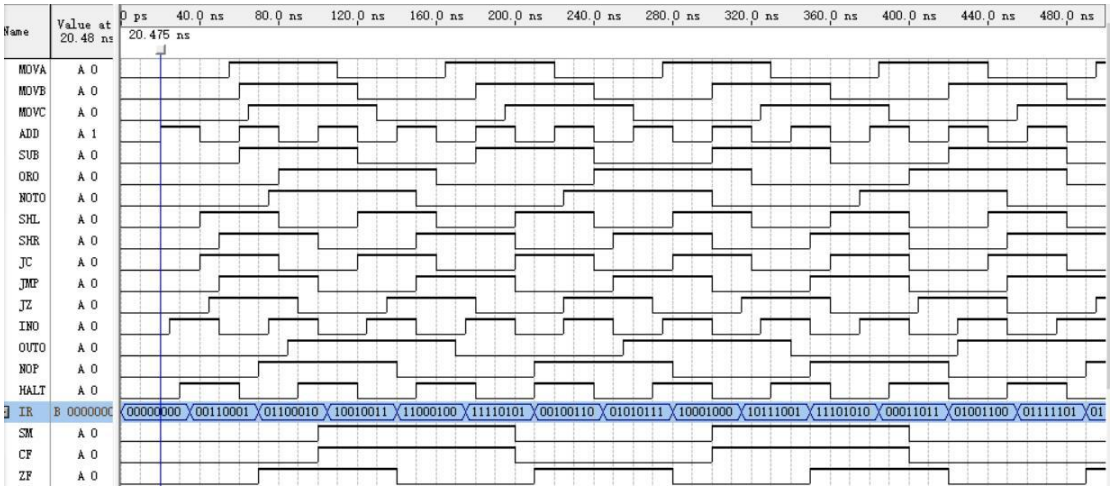


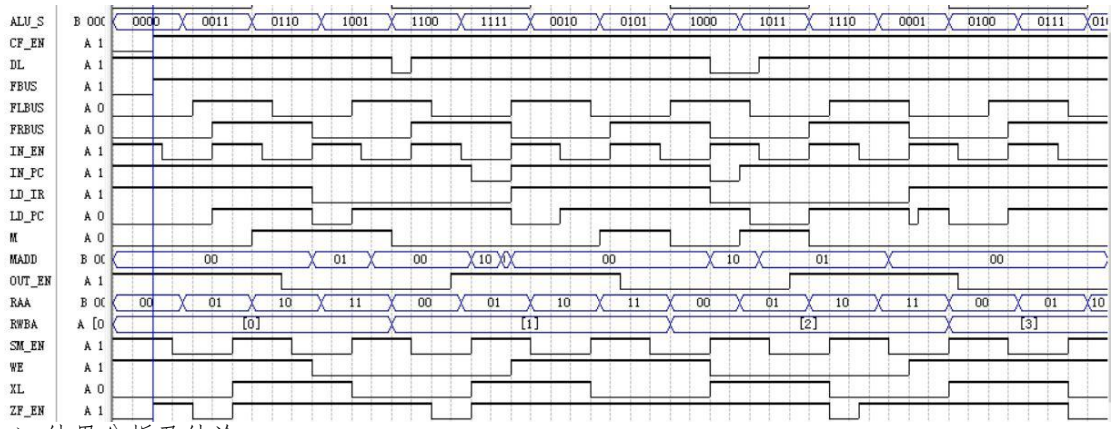
2、 功能仿真

a) 功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

b) 功能仿真图





c) 结果分析及结论

当执行指令 MOVA 时：把数据从 R2 移到 R1

取址周期 SM=0：

DL=1,XL=0，时钟上升沿 RAM 读出指令

时钟处于下降沿，指令寄存器 IR 载入数据，IR=10010001

执行周期 SM=1：

WE=0，RAA=01，RWBA=00，将数据送入 ALU，ALU=1001，R1+R2

时钟处于下降沿，R1 寄存器载入输入数据,产生进位 cf=1

指令执行完成.

当执行指令 MOVB 时：把寄存器中的值送入到 RAM 中

取址周期 SM=0：

DL=1,XL=0，时钟上升沿 RAM 读出指令

时钟处于下降沿，指令寄存器 IR 载入数据，IR=11111111

执行周期 SM=1：

WE=1，RAA=11,RWBA=11，读出寄存器 A 对应数据，DL=0,XL=1 加载 RAM 中，RAM[address]=00000000

指令执行完成.

当执行指令 MOVC 时：把 RAM 中数据移动到寄存器中,其中 M 是 C 寄存器中存储的地址

取址周期 SM=0：

DL=1,XL=0，时钟上升沿 RAM 读出指令

时钟处于下降沿，指令寄存器 IR 载入数据，IR=11110011
执行周期 SM=1：
WE=0，RAA=11，RWBA=00，RAM 读出 00000000 对应数据加载到 A 寄存器中
时钟为 8.0ns，时钟处于下降沿，A 寄存器载入总线数据，
指令执行完成.
当执行指令 ADD 时：将寄存器 R1R2 中的数据做算数相加载入 R1 寄存器
DL=1,XL=0 时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=10010001
执行周期 SM=1：
WE=0，RAA=01,RWBA=00，将数据送入 ALU，ALU=1001，R1+R2
时钟为 12.0ns，时钟处于下降沿，R1 寄存器载入输入数据产生进位 cf=1
指令执行完成
当执行指令 SUB 时：将寄存器 R1R2 中的数据做算数相减输入载入 R1 寄存器
取址周期 SM=0：
DL=1,XL=0,时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=01100001
执行周期 SM=1：
WE=0，RAA=01，RWBA00，将数据送入 ALU，ALU=0110，R1-R2
时钟处于下降沿，R1 寄存器载入输入数据，结果为 00000000，zf 标志为 1.
指令执行完成.
当执行指令 OR0 时：将寄存器 R1R2 中的数据或操作后存入 R1
取址周期 SM=0：
DL=1,XL=0,时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=10110001
执行周期 SM=1：
WE=0，RAA=01，RWBA=00，R1 or R2=11110011
时钟处于下降沿，R1 寄存器载入总线数据
指令执行完成.
当执行指令 NOT0 时：将寄存器 R1 中的数据取反后存入 R1
取址周期 SM=0：
DL=1,XL=0 时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=01010000
执行周期 SM=1：
WE=0，RAA=00，RWBA=00，not R1=11110011
时钟处于下降沿，R1 寄存器载入总线数据
指令执行完成.
当执行指令 SHR 时：将寄存器 R1 中的数据右移后存入 R1
取址周期 SM=0：
RAM_DL=1,RAM_XL=0 时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=10100000
执行周期 SM=1：
WE=0，RAA=00，RWBA=00，R1 右移
时钟处于下降沿，R1 寄存器载入总线数据指令执行完成.
当执行指令 SHL 时：将寄存器 R1 中的数据左移后存入 R1

取址周期 SM=0：
DL=1XL=0,时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=00010000
执行周期 SM=1：
WE=0，RAA=00，RWBA=00，R1 左移
时钟处于下降沿，R1 寄存器载入总线数据
指令执行完成.
当执行指令 JC 时：PC 寄存器转移到指定地址
取址周期 SM=0：
DL=1,XL=0 时钟上升沿 RAM 读出指令
时钟为 13.0ns 时，时钟处于下降沿，指令寄存器 IR 载入数据
执行周期 SM=1：
WE=1，RAA=10，RWBA=00，DL=1,XL=0，从 RAM 读取一条地址
时钟处于下降沿，PC 寄存器载入输入数据
指令执行完成.
当执行指令 JZ 时：PC 储存器转移到指定地址
取址周期 SM=0：
DL=1,XL=0，时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据
WE=1，RAA=10，REBA=00，DL=1,XL=0，从 RAM 读取一条地址
时钟处于下降沿，PC 寄存器载入输入数据指令执行完成.
当执行指令 JMP 时：将寄存器指向指定的地址
取址周期 SM=0：
DL=1,XL=0，时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据
WE=1，RAA=00，RWBA=00，IN0=0,LD=1,
时钟处于下降沿，PC 寄存器载入总线数据
指令执行完成.
当执行指令 IN0 时：将数据输入载入 R1 储存器
取址周期 SM=0：
DL=1,XL=0,时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=00100000
执行周期 SM=1：
WE=0，RAA=00，RWBA=00，将 IN 线对应数据加载到 R1 寄存器中
时钟处于下降沿，A 寄存器载入输入数据，IN0=11110011
指令执行完成.
当执行指令 OUT0 时：将寄存器 R1 的数据输出
取址周期 SM=0：
DL=1,XL=0，时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=01000000
执行周期 SM=1：
WE=1，RAA=00，RWBA=00，OUT0=1，向外展示数据，LED=00001100.
指令执行完成.
当执行指令 NOP 时：空操作不进行任何实际操作

取址周期 SM=0：
DL=1,XL=0 时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=011100000
执行周期 SM=1：
WE=1，RAA=00，RWBA=00，载入 pc
时钟处于下降沿，不进行任何操作.
指令执行完成.
当执行指令 HALT 时：停机操作，所有使能信号失效，计算机系统不再运作
取址周期 SM=0：
DL=1,XL=0，时钟上升沿 RAM 读出指令
时钟处于下降沿，指令寄存器 IR 载入数据，IR=10000000
执行周期 SM=1：
系统所有使能信号失效，此时所有部件不工作.
指令执行完成.
指令执行完成
结论：仿真结果满足功能要求，设计正确

五、实验结论

1、思考题

1）移位逻辑不工作时，输出应该为何值？为什么？

输出为高阻“ZZZZZZZZ”

因为不工作时即不导通，因此会相应的输出高阻态

2）ALU 的输出 Cf 和 Zf 应该如何处理？

将 CF,ZF 分别传入 CF,ZF 对应的触发器内，当触发器时钟信号上升或下降沿时产生相应的控制信号控制 JC,JZ 执行。

CF 为进位标志，加法时的最高位产生进位或者减法时的最高位出现借位，则 CF=1，否则 CF=0；

ZF 为零标志，运算结果每位都是 0 时，ZF=1，否则 ZF=0；

3）如何产生正确的控制信号以及具体的编程实现？

FBUS，FRBUS，FLBUS：为控制移位寄存器的信号指导移位寄存器进行左右移位，传送以及高阻状态的执行

IR：为控制指令寄存器的信号，指导 IR 进行载入

SM：为控制 SM 时钟发生器的信号，指导其工作

WE，RAA，RWBA：为控制寄存器组的信号，指导寄存器组的读写操。

MADD：为控制选择器的信号，指导控制器进行选择

ALU，M：为控制函数发生器的信号，指导函数发生器工作

LD，PC：为控制指向 PC 寄存器的信号，指导 PC 进行加一和跳转操作

XL，DL：为指向 RAM 的控制信号指导 RAM 进行读写或高阻状态

CF,ZF：为控制 C,Z 寄存器的信号，指导 C，Z 寄存器的工作

INO：为控制输入的信号。

OUTO：为控制输出的信号。

具体的编程实现见上述控制器的实验

2、实验总结与实验心得

本次的实验对我而言有一定的难度，所以其中简单的一部分由自己独立完成,另外一部分求助了同学。在同学的帮助下，我解决了自己的问题。在实验完成之后我对 ALU，移位逻辑，控制信号的产生有了更深入了解，同时对模型机的指令运作过程有了更深入的理解。