Cpu 设计报告

计算机设计与实践

于璇
1110310521

目录

一、实验目的

1. 掌握 xilinx ISE 集成开发环境和 Modelsim 仿真工具的使用方法
2. 掌握 VHDL 语言
3. 掌握 FPGA 编程方法及硬件调试手段
4. 深刻理解处理器结构和计算机系统的整体工作原理

二、实验环境

Xilinx ISE 9.1i 集成开发环境，modelsim xe 仿真工具盒 cop2000 系统实验平台

三、设计思想

1、接口信号定义

信号	位数	方向	来源/去向	意义
RST	1	IN	处理器板	低电平复位
CLK	1	IN	处理器板	系统时钟
ABUS	16	OUT	主存储器	地址总线
DBUS	16	INOUT	主存储器	数据总线
NMREQ	1	OUT	主存储器	存储器片选
NRD	1	OUT	主存储器	存储器读
NWR	1	OUT	主存储器	存储器写
NBHE	1	OUT	主存储器	高字节访问允许
NBLE	1	OUT	主存储器	低字节访问允许

2、处理器设计方案

a. 指令格式设计

序号	伪指令	二进制	含义
1	Mov Rx,X	00000 Ri X	X->Rx
2	Mov Rx,Ry	00001 Rx 00000 Ry	[X]->Rx
3	Mov [X],Ry	00010 Ry X	Ry->[X]
4	Mov Rx,[X]	00011 Rx X	[X]->Rx
5	Adc Rx,X	00100 Rx X	Rx+X+Cy->Rx
6	Adc Rx,Ry	00101 Rx 00000 Ry	Rx+Ry+Cy->Rx
7	Sbb Rx,X	00110 Rx X	Rx-X-Cy->Rx
8	Sbb Rx,Ry	00111 Rx 00000 Ry	Rx-Ry-Cy->Rx
9	And Rx,X	01000 Rx X	Rx and X ->Rx
10	And Rx,Ry	01001 Rx 00000 Ry	Rx and Ry ->Rx
11	Or Rx,X	01010 Rx X	Rx or X ->Rx
12	Or Rx,Ry	01011 Rx 00000 Ry	Rx or Ry ->Rx

13	Clc	01100 00000000000	0->Cy
14	Stc	01101 00000000000	1->Cy
15	Jmp X	01110 000 X	R7//X->PC
16	(间指)Jmp [Rx]	01111 Rx 00000000	R7//Rx->PC
17	Jz sign	10000 000 sign	Z=1:PC+1+sign->PC
18	Jc sign	10001 000 sign	C=1:PC+1+sign->PC
19	(变址)Mov Rx,[R7+X]	10010 Rx X	[R7+X]->Rx

b. 微操作定义

i. 取值：

pc->MAR

1->upc

MDR->ir

ii. 运算:

1. Mov Rx,X：X->Rtemp Rx->raddr
2. Mov Rx,Ry: (Ry)-> Rtemp Rx->raddr
3. Mov [X],Ry: (Ry)-> Rtemp 1->wr (R7)//x->addr
4. Mov Rx,[X]: Rx->raddr 1->rd (R7)//x->addr
5. Adc Rx,X: (Rx)+ X+cy ->Rtemp Rx->raddr
6. Adc Rx,Ry: (Rx)+ cy+Ry ->Rtemp Rx->raddr
7. Sbb Rx,X: (Rx)-X-cy ->Rtemp Rx->raddr
8. Sbb Rx,Ry: (Rx)- cy-Ry ->Rtemp Rx->raddr
9. And Rx,X: (Rx) and X ->Rtemp Rx->raddr
10. And Rx, Ry: (Rx) and (Ry) ->Rtemp Rx->raddr
11. Or Rx,X: (Rx) or X ->Rtemp Rx->raddr
12. Or Rx, Ry: (Rx) or (Ry) ->Rtemp Rx->raddr
13. Clc: 0->cy
14. Stc: 1->cy
15. Jmp X: (R7)//X->PCtemp
16. (间指)Jmp [Rx] : (R7)//00000000->PCtemp (R7)//(Rx)->addr
    1->rd
17. Jz sign: PC+1+sign->PCtemp
18. Jc sign: PC+1+sign->PCtemp
19. (变址)Mov Rx,[R7+X]: Rx->raddr 1->rd
    (R7)//[(R7)+x]->addr

iii. 访存:

rd->nRD

wr->nWr addr->MAR

data->MDR

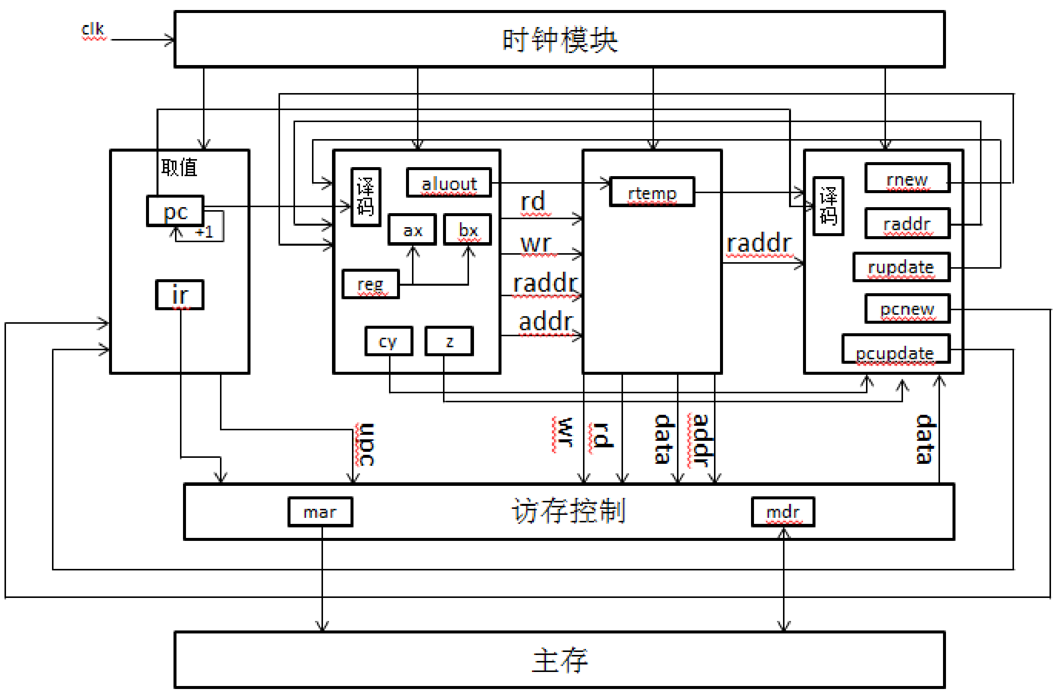
mdr->datat

iv. 回写:

Raddr->raddr

0/1->rupdate

0/1->pcupdate



Rtemp/datat->rnew

Pctemp->pcnew

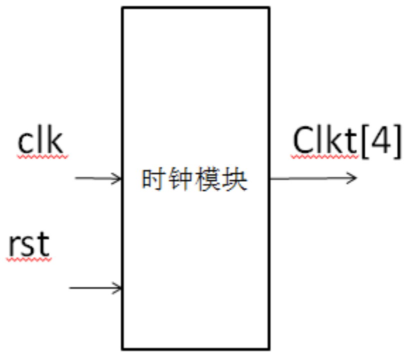
c. 节拍的划分

1. 取值
2. 运算
3. 访存
4. 回写

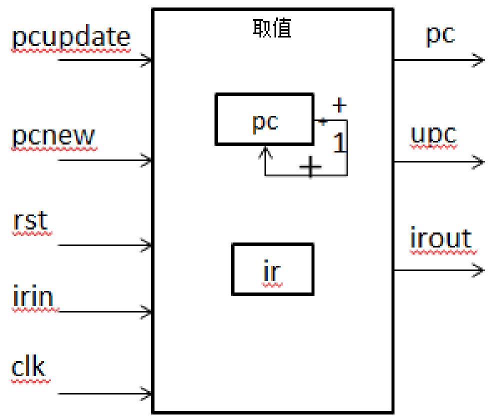
d. 处理器设计框图

e. 各模块设计框图

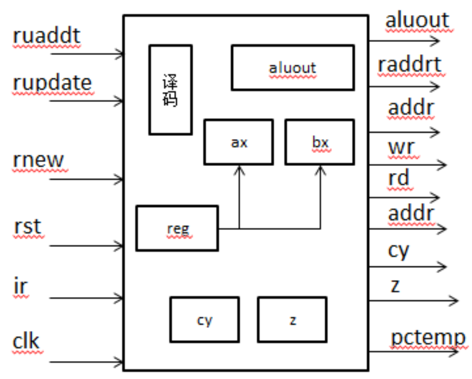
1. 时钟



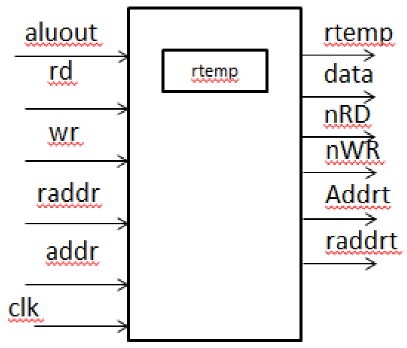
2. 取值



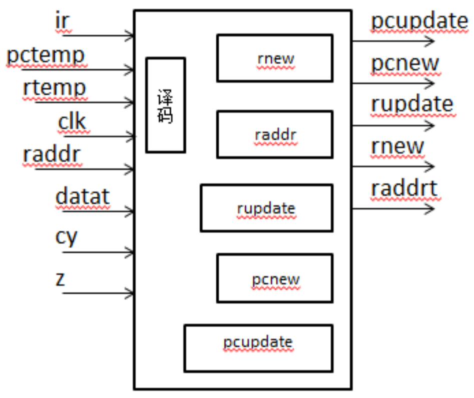
3. 运算模块



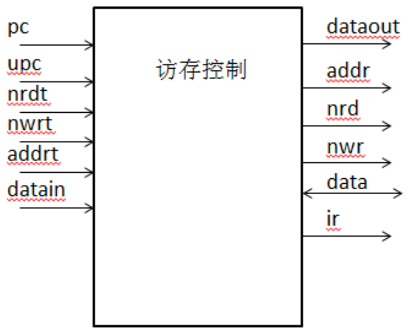
4. 存储管理



5. 回写模块



6. 访存控制



f. 各模块输出管脚定义

1> 时钟模块

功能描述：时钟模块为一节拍发生器，以输入时钟信号作为触发，四个节拍循环往复，当“rst”为 0 时节拍复位。

信号名	位数	方向	来源/去向	意义
clk	1	IN	处理器板	时钟信号输入
rst	1	IN	处理器板	复位
clktemp	4	OUT	各个模块	节拍

2> 取值模块

功能描述：取指模块主要负责取指操作，当复位信号为 0 时，pc 在第四节拍的上升沿加 1，在第一个节拍，将当前 pc 给到访存控制模块，同时送访存请求信号 upc=1（结束后 upc=0）,取得指令；同时将取得的指令送往运算模块、回写模块，pc 送往回写模块。

信号名	位数	方向	来源/去向	意义
clk1	1	IN	时钟	
clk4	1	IN	时钟	
rst	1	IN	处理器板	复位
upc	1	OUT	访存控制	取值请求信号
pcupdate	1	IN	回写	回写 pc 标志
pcdata	16	IN	回写	回写 pc 值
irin	16	IN	访存控制	
irout	16	IN	访存控制	
pc	16	OUT	访存控制	指令地址

3> 运算模块

功能描述：复位信号为 0 时，alutou、addr、Cy、Z、reg 清零；当回写信号为 1 时，将回写的数据回写入寄存器中；在第二节拍完成指令的译码工作，并根据译码结果对相关的指令置好 addr、aluout，wr，raddr，pctemp，和 rd,以及对一些改变运算标志位的指令置好 Cy 和 Z 这两个标志位。

信号名	位数	方向	来源/去向	意义
clk2	1	IN	时钟	
pc	16	IN	取值	指令当前地址
rupdate	1	IN	回写	回写 reg 标志
rnew	8	IN	回写	回写 reg 值
raddrtt	8	IN	回写	回写 reg 地址
ir	16	IN	取值	
aluout	8	OUT	存储管理	运算数值
pctemp	16	OUT	回写	跳转 Pc 下一个地址
rst	1	IN	处理器板	复位
rd	1	OUT	存储管理	读信号标志
wr	1	OUT	存储管理	写信好标志
raddr	3	OUT	存储管理	回写 reg 地址
cy	1	OUT	回写	溢出标志
z	1	OUT	回写	零标志
addr	16	OUT	存储管理	主存读写地址

4> 存储管理模块

功能描述：主要完成从访存控制模块进行不同指令的存数与取数的请求。

信号名	位数	方向	来源/去向	意义
clk3	1	IN	时钟	
clk1	1	IN	时钟	
rst	1	IN	处理器板	复位
rd	1	IN	运算管理	读信号标志
wr	1	IN	运算管理	写信好标志
raddrin	3	IN	运算管理	回写 reg 地址
addrin	16	IN	运算管理	主存读写地址
DATAIN	8	IN	运算管理	运算好的数据
DATA	8	OUT	访存控制	需要写的数据
RDATA	8	OUT	回写	运算好的数据
ADDR	16	OUT	访存控制	主存读写地址
WRD	1	OUT	访存控制	读信号
WWR	1	OUT	访存控制	写信号
raddr	3	OUT	回写	回写 reg 地址

5> 回写模块

功能描述：第四节拍时完成 PC 或者是 Reg 的回写

信号名	位数	方向	来源/去向	意义
data	8	IN	访存控制	主存读回的数据
rtemp	8	IN	存储管理	运算好的数据

cy	1	IN	运算	溢出标志
z	1	IN	运算	零标志
raddr	3	IN	存储管理	回写 reg 地址
raddrt	8	OUT	运算	回写 reg 地址
ir	16	IN	取值	
Clk4	1	IN	时钟	
pcupdate	1	OUT	取值	回写 pc 标志
pcnew	16	OUT	取值	回写 pc 值
pctemp	16	IN	运算	跳转 Pc 下一个地址
rnew	8	OUT	运算	回写 reg 值
rupdate	1	OUT	运算	回写 reg 标志
rst	1	IN	处理器板	复位

6> 访存控制模块

功能描述：当取址模块发出取址请求时，根据 PC 取出指令送给取址模块；根据存储管理模块给的读写请求对存储器进行读写操作。

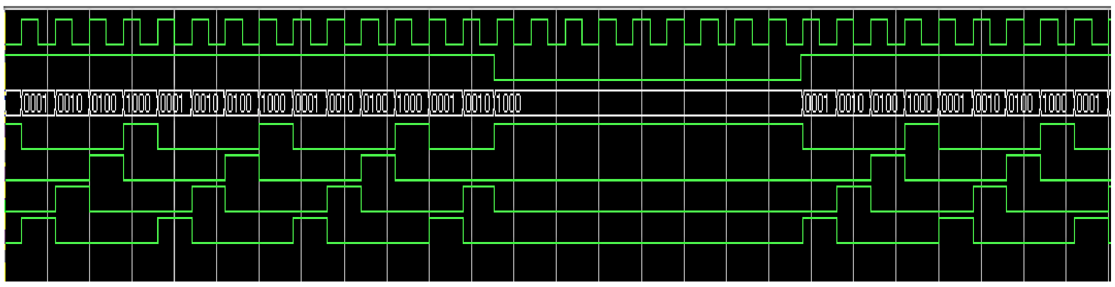
信号名	位数	方向	来源/去向	意义
upc	1	IN	取值	取值请求信号
pc	16	IN	取值	取值地址
wrt	1	IN	存储管理	写信号
rdt	1	IN	存储管理	读信号
datain	8	IN	存储管理	需要写的数值
addrt	16	IN	存储管理	读写地址
mar	16	OUT	处理器板	地址线
ir	16	OUT	取值	指令
mdr	16	OUT	处理器板	数据线
wr	1	OUT	处理器板	写
rd	1	OUT	处理器板	读
dataout	8	OUT	回写	回来的数据

四、实验设计及测试

1. 时钟

```
architecture Behavioral of clock is
signal temp: STD_LOGIC_VECTOR(3 DOWNTO 0):="1000";
begin
PROCESS(clk,rst)
begin
 if rst='0' then
        temp<="1000";
 elsif clk'event and clk='1' then
        temp(3 downto 1)<=temp(2 downto 0);
        temp(0)<=temp(3);
 end if;
end process;
clktemp<=temp;
```

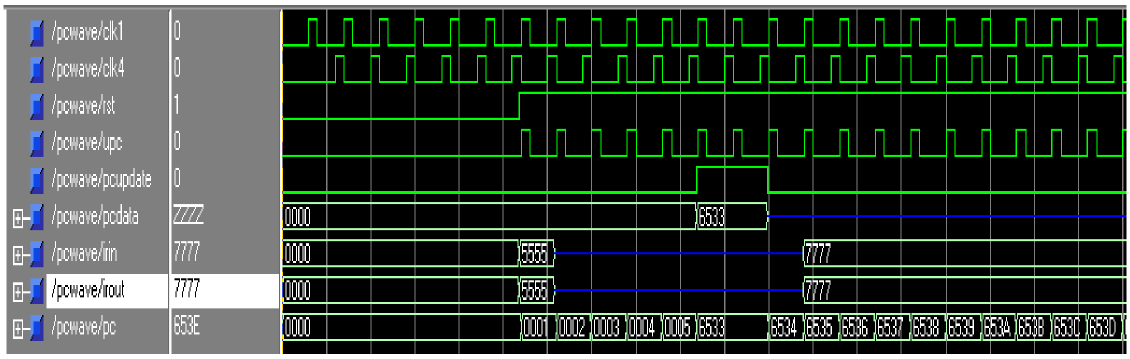
```
end Behavioral;
```



2. 取值模块

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity pcmodel is
port (     clk1 : IN STD_LOGIC;
           clk4 : IN STD_LOGIC;
           rst : IN STD_LOGIC;
           upc : OUT STD_LOGIC;
           pcupdate : IN STD_LOGIC;
           pcdata : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
           irin : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
           irout : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
        pc : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );

end pcmodel;

architecture Behavioral of pcmodel is
signal pctemp:std_logic_vector(15 downto 0):="0000000000000000";
--Variable pctemp:std_logic_vector(15 downto 0);
signal upctemp:std_logic;

begin
process(clk4,rst,pcupdate,pcdata)
begin
 if rst='0' then
       pctemp<="0000000000000000";
 elsif pcupdate='1' then
       pctemp<=pcdata;
 elsif clk4'event and clk4='0' then
       pctemp<=pctemp+1;
 end if;
end process;
process(clk1,rst,pcupdate,pcdata)
begin
```

```
            if rst='0' then
                upctemp<='0';
            elsif clk1='1' then
                    upctemp<='1';
            elsif clk1='0' then
                upctemp<='0';
            end if;
            end process;
            pc<=pctemp;
            upc<=upctemp;
            irout<=irin;
        end Behavioral;
```



3. 运算模块

```
            library IEEE;
            use IEEE.STD_LOGIC_1164.ALL;
            use IEEE.STD_LOGIC_ARITH.ALL;
            use IEEE.STD_LOGIC_UNSIGNED.ALL;

            entity alu is
                Port ( clk2 : in   STD_LOGIC;
                        pc : in   STD_LOGIC_VECTOR (15 downto 0);
                        rupdate : in   STD_LOGIC;
                        rnew : in   STD_LOGIC_VECTOR (7 downto 0);
                        raddrtt : in   STD_LOGIC_VECTOR (7 downto 0);
                        ir : in   STD_LOGIC_VECTOR (15 downto 0);
                        aluout : out   STD_LOGIC_VECTOR (7 downto 0);
                        pctemp : out   STD_LOGIC_VECTOR (15 downto 0);
                        rst : in   STD_LOGIC;
                        rd : out   STD_LOGIC;
                        wr : out   STD_LOGIC;
                        raddr : out   STD_LOGIC_VECTOR (2 downto 0);
                       cy : out   STD_LOGIC;
                       z : out   STD_LOGIC;

            --              seereg0:out        STD_LOGIC_VECTOR        (7        downto
        0):="00000000";
            --          seereg1:out      STD_LOGIC_VECTOR      (7      downto
        0):="00000000";
            --          seereg2:out      STD_LOGIC_VECTOR      (7      downto
        0):="00000000";
            --            seereg3:out     STD_LOGIC_VECTOR      (7      downto
        0):="00000000";
            --          seereg4:out STD_LOGIC_VECTOR (7 downto 0):="00000000";
            --            seereg5:out     STD_LOGIC_VECTOR      (7      downto
        0):="00000000";
            --            seereg6:out     STD_LOGIC_VECTOR      (7      downto
        0):="00000000";
            --            seereg7:out     STD_LOGIC_VECTOR      (7      downto
        0):="00000000";


                        addr : out    STD_LOGIC_VECTOR (15 downto 0));
            end alu;

            architecture Behavioral of alu is

            signal reg0:STD_LOGIC_VECTOR (7 downto 0):="00000000";
            signal reg1:STD_LOGIC_VECTOR (7 downto 0):="00000000";
            signal reg2:STD_LOGIC_VECTOR (7 downto 0):="00000000";
            signal reg3:STD_LOGIC_VECTOR (7 downto 0):="00000000";
            signal reg4:STD_LOGIC_VECTOR (7 downto 0):="00000000";
            signal reg5:STD_LOGIC_VECTOR (7 downto 0):="00000000";
            signal reg6:STD_LOGIC_VECTOR (7 downto 0):="00000000";
            signal reg7:STD_LOGIC_VECTOR (7 downto 0):="00000000";
            signal aluouttemp:STD_LOGIC_VECTOR (8 downto 0):="000000000";
            signal cytemp:STD_LOGIC:='0';
            signal ztemp:STD_LOGIC:='0';

            begin
            process(rupdate,rnew,raddrtt,rst)
            begin
                if rst='0' then
                    reg0<="00000000";
                    reg1<="00000000";
                    reg2<="00000000";
                    reg3<="00000000";
                    reg4<="00000000";
                    reg5<="00000000";
                    reg6<="00000000";
```

```
                reg7<="00000000";
        elsif rupdate='1' then
                if raddrtt(0)='1' then
                        reg0<=rnew;
                elsif raddrtt(1)='1' then
                        reg1<=rnew;
                elsif raddrtt(2)='1' then
                        reg2<=rnew;
                elsif raddrtt(3)='1' then
                        reg3<=rnew;
                elsif raddrtt(4)='1' then
                        reg4<=rnew;
                elsif raddrtt(5)='1' then
                        reg5<=rnew;
                elsif raddrtt(6)='1' then
                        reg6<=rnew;
                elsif raddrtt(7)='1' then
                        reg7<=rnew;
                end if;
        end if;
end process;

process(ir,clk2,rst)
variable result:STD_LOGIC_VECTOR (8 downto 0);
variable cyy:STD_LOGIC;
variable ax:STD_LOGIC_VECTOR (8 downto 0):="000000000";
variable bx:STD_LOGIC_VECTOR (8 downto 0):="000000000";
begin
 if rst='0' then
        cytemp<='0';
        ztemp<='0';
        aluouttemp<="ZZZZZZZZZ";
 elsif clk2'event and clk2='1' then
        if ir(15 downto 11)="00000" then
                aluouttemp(7 downto 0)<=ir(7 downto 0);
        elsif ir(15 downto 11)="00001" then
                case ir(2 downto 0) is
                        when "000" => aluouttemp(7 downto 0)<=reg0;
                        when "001" => aluouttemp(7 downto 0)<=reg1;
                        when "010" => aluouttemp(7 downto 0)<=reg2;
                        when "011" => aluouttemp(7 downto 0)<=reg3;
                        when "100" => aluouttemp(7 downto 0)<=reg4;
                        when "101" => aluouttemp(7 downto 0)<=reg5;
                        when "110" => aluouttemp(7 downto 0)<=reg6;
                        when "111" => aluouttemp(7 downto 0)<=reg7;
                        when others => NULL;
                end case;
        elsif ir(15 downto 11)="00010" then
                case ir(10 downto 8) is
                        when "000" => aluouttemp(7 downto 0)<=reg0;
                        when "001" => aluouttemp(7 downto 0)<=reg1;
                        when "010" => aluouttemp(7 downto 0)<=reg2;
                        when "011" => aluouttemp(7 downto 0)<=reg3;
                        when "100" => aluouttemp(7 downto 0)<=reg4;
                        when "101" => aluouttemp(7 downto 0)<=reg5;
                        when "110" => aluouttemp(7 downto 0)<=reg6;
                        when "111" => aluouttemp(7 downto 0)<=reg7;
                        when others => NULL;
                end case;
        elsif ir(15 downto 11)="00100" or ir(15 downto 11)="00110" or ir(15
downto 11)="00101" or ir(15 downto 11)="00111" or ir(15 downto 11)="01000" or
ir(15 downto 11)="01001" or ir(15 downto 11)="01010" or ir(15 downto 11)="01011"
then
                case ir(10 downto 8) is
                        when "000" => ax:='0' & reg0;
                        when "001" => ax:='0' & reg1;
                        when "010" => ax:='0' & reg2;
                        when "011" => ax:='0' & reg3;
                        when "100" => ax:='0' & reg4;
                        when "101" => ax:='0' & reg5;
                        when "110" => ax:='0' & reg6;
                        when "111" => ax:='0' & reg7;
                        when others => NULL;
                end case;
                if ir(15 downto 11)="00100" or ir(15 downto 11)="00110" or ir(15
downto 11)="01000" or ir(15 downto 11)="01010" then
                        bx:='0' & ir(7 downto 0);
                elsif ir(15 downto 11)="00101" or ir(15 downto 11)="00111" or ir(15
downto 11)="01001" or ir(15 downto 11)="01011" then
                        case ir(2 downto 0) is
                            when "000" => bx:='0' & reg0;
                            when "001" => bx:='0' & reg1;
                            when "010" => bx:='0' & reg2;
                            when "011" => bx:='0' & reg3;
                            when "100" => bx:='0' & reg4;
                            when "101" => bx:='0' & reg5;
                            when "110" => bx:='0' & reg6;
                            when "111" => bx:='0' & reg7;
```

```
                        when others => NULL;
                    end case;
                end if;
                if ir(15 downto 11)="00100" or ir(15 downto 11)="00110" or ir(15
        downto 11)="00101" or ir(15 downto 11)="00111" then
                    cyy:=cytemp;
                    if ir(15 downto 11)="00100" or ir(15 downto 11)="00101" then
                        aluouttemp<=ax+bx+cyy;
                        result:=ax+bx+cyy;
                    elsif ir(15 downto 11)="00110" or ir(15 downto 11)="00111" then
                        aluouttemp<=ax-bx-cyy;
                        result:=ax-bx-cyy;
                    end if;
                    if result="00000000" then
                        ztemp<='1';
                    else
                        ztemp<='0';
                    end if;
                    if result(8)='0' then
                        cytemp<='0';
                    else
                        cytemp<='1';
                    end if;
                elsif ir(15 downto 11)="01000" or ir(15 downto 11)="01001" or ir(15
        downto 11)="01010" or ir(15 downto 11)="01011" then
                    if ir(15 downto 11)="01000" or ir(15 downto 11)="01001" then
                        aluouttemp<=ax and bx;
                    elsif ir(15 downto 11)="01010" or ir(15 downto 11)="01011" then
                        aluouttemp<=ax or bx;
                    end if;
                end if;
            elsif ir(15 downto 11)="01100" then
                cytemp<='0';
            elsif ir(15 downto 11)="01101" then
                cytemp<='1';
            end if;
        end if;
        end process;

        process(rst,clk2,ir)
        begin
        if rst='0' then
            raddr<="ZZZ";
        elsif clk2'event and clk2='1' then
```

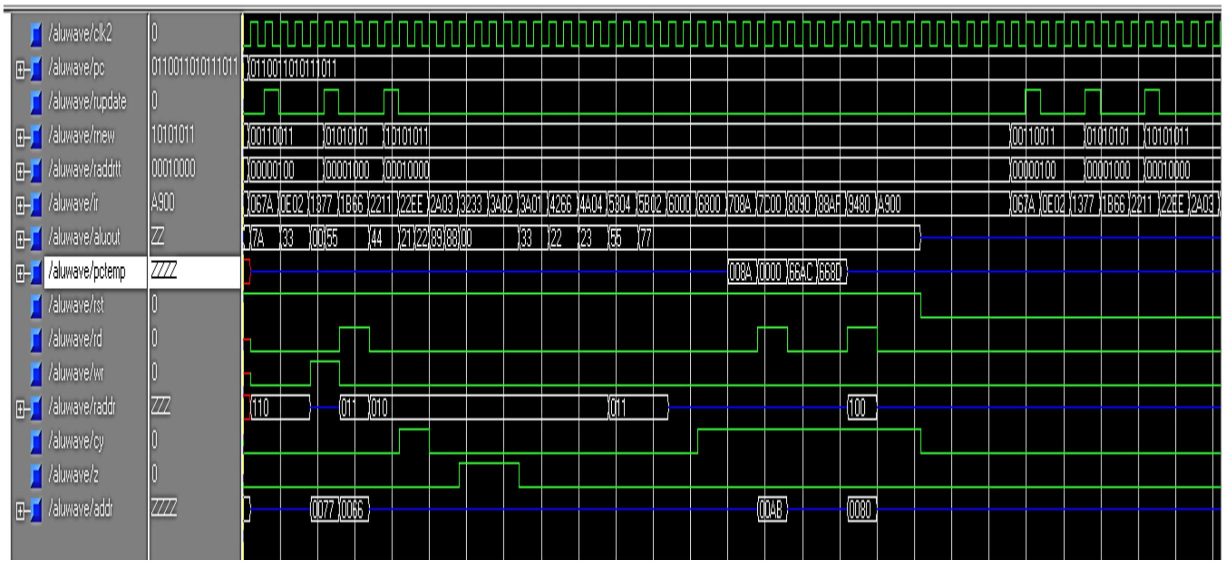
```
          if ir(15 downto 11)="00000" or ir(15 downto 11)="00001" or ir(15 downto 11)="00011" or ir(15 downto 11)="00100" or ir(15 downto 11)="00110" or ir(15 downto 11)="00101" or ir(15 downto 11)="00111" or ir(15 downto 11)="01000" or ir(15 downto 11)="01001" or ir(15 downto 11)="01010" or ir(15 downto 11)="01011" or ir(15 downto 11)="10010" then
               raddr<=ir(10 downto 8);
          else
               raddr<="ZZZ";
          end if;
     end if;
     end process;

     process(rst,clk2,ir)
     variable pctmp:STD_LOGIC_VECTOR (15 downto 0);
     begin
     if rst='0' then
          pctemp<="ZZZZZZZZZZZZZZZZ";
     elsif clk2'event and clk2='1' then
          if ir(15 downto 11)="01110" then
               pctemp<=reg7 & ir(7 downto 0);
          elsif ir(15 downto 11)="01111" then
               pctemp<=reg7 & "00000000";
          elsif ir(15 downto 11)="10000" or ir(15 downto 11)="10001" then
               pctmp:="000000000" & ir(6 downto 0);
               if ir(7)='0' then
                    pctemp<=pc+pctmp+1;
               else
                    pctemp<=pc-pctmp+1;
               end if;
          else
               pctemp<="ZZZZZZZZZZZZZZZZ";
          end if;
     end if;
     end process;

     process(rst,clk2,ir)
     begin
     if rst='0' then
          addr<="ZZZZZZZZZZZZZZZZ";
          rd<='0';
          wr<='0';
     elsif clk2'event and clk2='1' then
          if ir(15 downto 11)="00010" then
               wr<='1';
```

```
            rd<='0';
            addr<=reg7 & ir(7 downto 0);
        elsif ir(15 downto 11)="00011" then
            wr<='0';
            rd<='1';
            addr<=reg7 & ir(7 downto 0);
        elsif ir(15 downto 11)="01111" then
            wr<='0';
            rd<='1';
            addr(15 downto 8)<=reg7;
            case ir(10 downto 8) is
                when "000" => addr(7 downto 0)<=reg0;
                when "001" => addr(7 downto 0)<=reg1;
                when "010" => addr(7 downto 0)<=reg2;
                when "011" => addr(7 downto 0)<=reg3;
                when "100" => addr(7 downto 0)<=reg4;
                when "101" => addr(7 downto 0)<=reg5;
                when "110" => addr(7 downto 0)<=reg6;
                when "111" => addr(7 downto 0)<=reg7;
                when others => NULL;
            end case;
        elsif ir(15 downto 11)="10010" then
            wr<='0';
            rd<='1';
            addr(15 downto 8)<=reg7;
            addr(7 downto 0)<=reg0+ir(7 downto 0);
        else
            wr<='0';
            rd<='0';
            addr<="ZZZZZZZZZZZZZZZZ";
        end if;
    end if;
end process;
aluout<=aluouttemp(7 downto 0);
cy<=cytemp;
z<=ztemp;

--seereg0<=reg0;
--seereg1<=reg1;
--seereg2<=reg2;
--seereg3<=reg3;
--seereg4<=reg4;
--seereg5<=reg5;
--seereg6<=reg6;
```

```
        --seereg7<=reg7;

    end Behavioral;
```

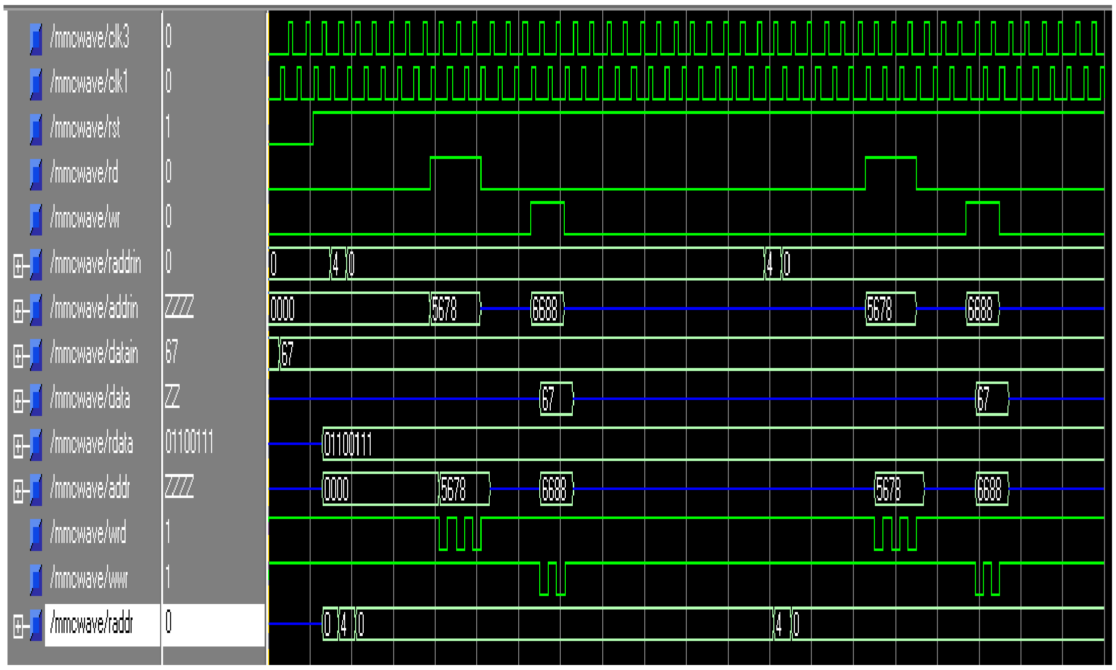
4. 存储管理

```
        library IEEE;
```



```
        use IEEE.STD_LOGIC_1164.ALL;
        use IEEE.STD_LOGIC_ARITH.ALL;
        use IEEE.STD_LOGIC_UNSIGNED.ALL;

        entity mmc is
        port (    clk3 : IN STD_LOGIC;
                  clk1 : in    STD_LOGIC;
                  rst : IN STD_LOGIC;
                  rd    :      IN STD_LOGIC;
                  wr    :      IN STD_LOGIC;
                  raddrin:  IN STD_LOGIC_VECTOR(2 DOWNTO 0);
                  addrin    :      IN STD_LOGIC_VECTOR(15 DOWNTO 0);
                  DATAIN :         IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                  DATA      :      OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
                  RDATA  :         OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
                  ADDR      :      OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
                  WRD       :      OUT STD_LOGIC;
                  WWR       :      OUT STD_LOGIC;
                  raddr     :      OUT STD_LOGIC_VECTOR(2 DOWNTO 0));

        end mmc;

        architecture Behavioral of mmc is
        begin
        process(rst,clk3,clk1)
```

```
      begin
       if rst='0' then
              WRD<='1';
              WWR<='1';
              ADDR<="ZZZZZZZZZZZZZZZZ";
              DATA<="ZZZZZZZZ";
              raddr<="ZZZ";
              RDATA<="ZZZZZZZZ";
              DATA<="ZZZZZZZZ";
       elsif clk1='1' then
              WRD<='1';
              WWR<='1';
       elsif clk3'event and clk3='1' then
              WRD<=not rd;
              WWR<=not wr;
              raddr<=raddrin;
              ADDR<=addrin;
              if wr='1' then
                     DATA<=DATAIN;
              else
                     DATA<="ZZZZZZZZ";
              end if;
              RDATA<=DATAIN;
       end if;
      end process;
end Behavioral;
```



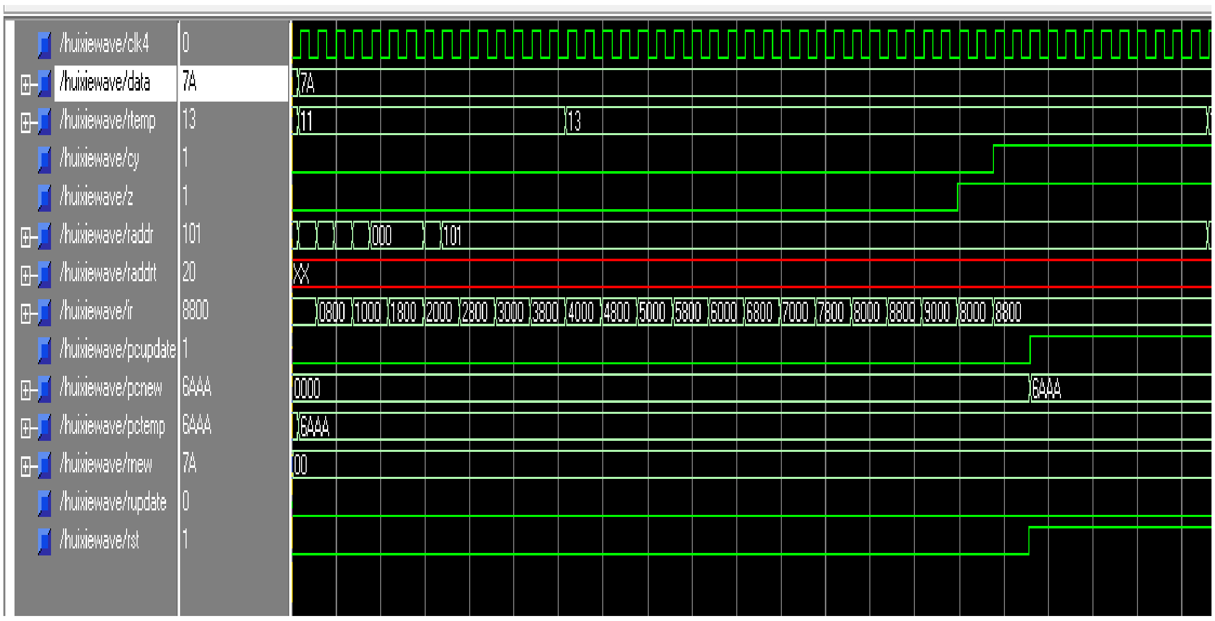
5. 回写模块

```
      library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity huixie is
      Port ( data : in   STD_LOGIC_VECTOR (7 downto 0);
                rtemp : in   STD_LOGIC_VECTOR (7 downto 0);
                cy : in   STD_LOGIC;
                z : in   STD_LOGIC;
                raddr : in   STD_LOGIC_VECTOR (2 downto 0);
                raddrt : out   STD_LOGIC_VECTOR (7 downto 0);
                ir : in   STD_LOGIC_VECTOR (15 downto 0);
                clk4 : in   STD_LOGIC;
                pcupdate : out   STD_LOGIC;
                pcnew : out   STD_LOGIC_VECTOR (15 downto 0);
               pctemp : in   STD_LOGIC_VECTOR (15 downto 0);
                rnew : out   STD_LOGIC_VECTOR (7 downto 0);
                rupdate : out   STD_LOGIC;
               rst      :     in    std_logic);
end huixie;

architecture Behavioral of huixie is

begin
process(clk4,rst)
begin
 if rst='0' then
      pcnew<="0000000000000000";
      pcupdate<='0';
      rnew<="00000000";
      rupdate<='0';
 elsif clk4'event and clk4='1' then
      if ir(15 downto 11)="00000" or ir(15 downto 11)="00001" or ir(15
downto 11)="00100" or ir(15 downto 11)="00101" or ir(15 downto 11)="00110" or
ir(15 downto 11)="00111" or ir(15 downto 11)="01000" or ir(15 downto 11)="01001"
or ir(15 downto 11)="01010" or ir(15 downto 11)="01011"   then
            rnew<=rtemp;
            rupdate<='1';
            pcupdate<='0';
            if raddr="000" then
                  raddrt<="00000001";
            elsif raddr="001" then
                  raddrt<="00000010";
            elsif raddr="010" then
```

```
                raddrt<="00000100";
            elsif raddr="011" then
                raddrt<="00001000";
            elsif raddr="100" then
                raddrt<="00010000";
            elsif raddr="101" then
                raddrt<="00100000";
            elsif raddr="110" then
                raddrt<="01000000";
            elsif raddr="111" then
                raddrt<="10000000";
            end if;
        elsif ir(15 downto 11)="00011" or ir(15 downto 11)="10010" then
            rnew<=data;
            rupdate<='1';
            pcupdate<='0';
            if raddr="000" then
                raddrt<="00000001";
            elsif raddr="001" then
                raddrt<="00000010";
            elsif raddr="010" then
                raddrt<="00000100";
            elsif raddr="011" then
                raddrt<="00001000";
            elsif raddr="100" then
                raddrt<="00010000";
            elsif raddr="101" then
                raddrt<="00100000";
            elsif raddr="110" then
                raddrt<="01000000";
            elsif raddr="111" then
                raddrt<="10000000";
            end if;
        elsif ir(15 downto 11)="01110" then
            pcnew<=pctemp;
            pcupdate<='1';
            rupdate<='0';
        elsif ir(15 downto 11)="01111" then
            pcnew(15 downto 8)<=pctemp(15 downto 8);
            pcnew(7 downto 0)<=data;
            pcupdate<='1';
            rupdate<='0';
        elsif ir(15 downto 11)="10000" then
            if z='1' then
```

```
                    pcnew<=pctemp;
```

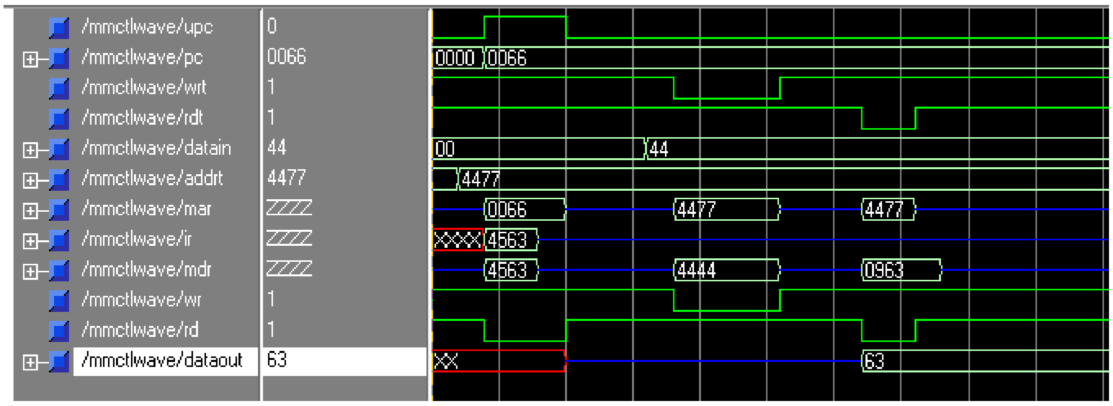


```
                    pcupdate<='1';
                    rupdate<='0';
                else
                    pcupdate<='0';
                    rupdate<='0';
                end if;
            elsif ir(15 downto 11)="10001" then
                if cy='1' then
                    pcnew<=pctemp;
                    pcupdate<='1';
                    rupdate<='0';
                else
                    pcupdate<='0';
                    rupdate<='0';
                end if;
            else
                pcupdate<='0';
                rupdate<='0';
            end if;
        end if;
        end process;
    end Behavioral;
```

6. 访存控制

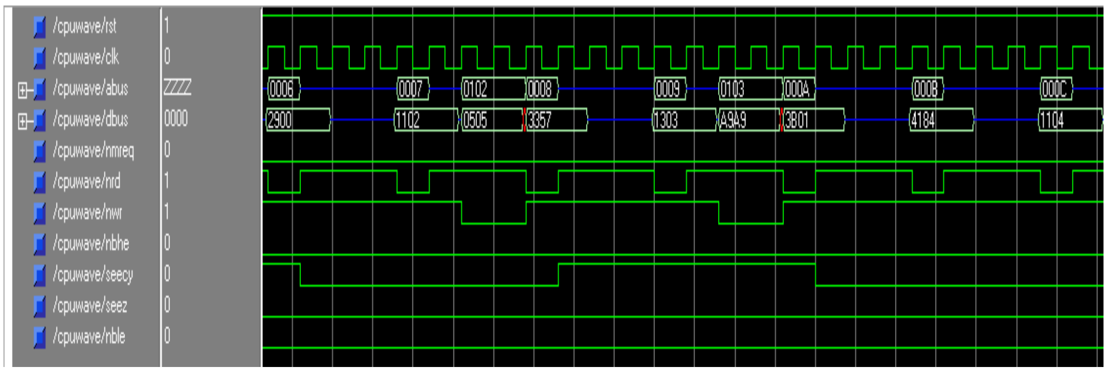
```
        library IEEE;
        use IEEE.STD_LOGIC_1164.ALL;
        use IEEE.STD_LOGIC_ARITH.ALL;
        use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity mmcontrol is
    Port ( upc : in   STD_LOGIC;
              pc : in   STD_LOGIC_VECTOR (15 downto 0);
              wrt : in   STD_LOGIC;
              rdt : in   STD_LOGIC;
              datain : in   STD_LOGIC_VECTOR (7 downto 0);
              addrt : in   STD_LOGIC_VECTOR (15 downto 0);
              mar : out   STD_LOGIC_VECTOR (15 downto 0);
             ir : out   STD_LOGIC_VECTOR (15 downto 0);
              mdr : inout   STD_LOGIC_VECTOR (15 downto 0);
              wr : out   STD_LOGIC;
              rd : out   STD_LOGIC;
              dataout : out   STD_LOGIC_VECTOR (7 downto 0));
end mmcontrol;

architecture Behavioral of mmcontrol is
signal mdrtemp:STD_LOGIC_VECTOR (15 downto 0);
begin
process(wrt,rdt,upc)
begin
 if wrt='0' and upc='0' then
        wr<='0';
 else
        wr<='1';
 end if;
end process;

process(wrt,rdt,upc)
begin
 if rdt='0' then
        rd<='0';
 elsif upc='1' then
        rd<='0';
 else
        rd<='1';
 end if;
end process;

process(wrt,rdt,upc,mdrtemp)
begin
 if upc='1' then
        ir<=mdrtemp;
 else
        dataout<=mdrtemp(7 downto 0);
    end if;
    end process;

    process(wrt,rdt,upc,addrt,pc)
    begin
    if (rdt='0' or wrt='0') and upc='0' then
            mar<=addrt;
    elsif upc='1' then
            mar<=pc;
    else
```

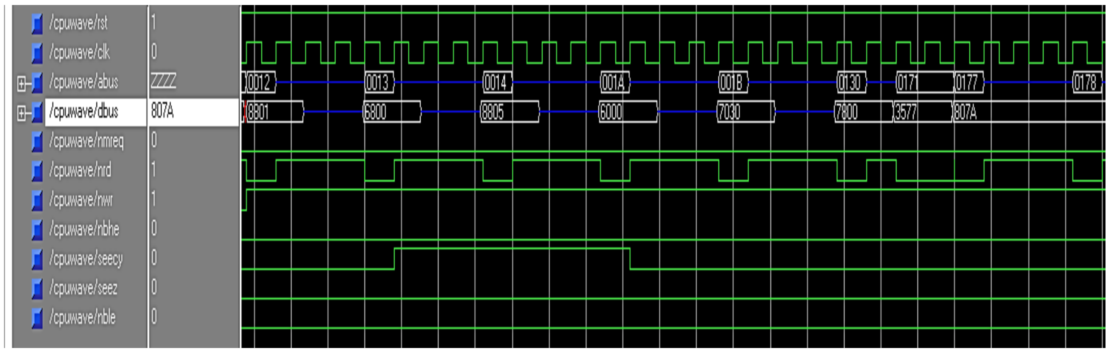


```
            mar<="ZZZZZZZZZZZZZZZZ";
    end if;
    end process;

    process(wrt,rdt,upc,datain)
    begin
    if wrt='0' and upc='0' then
            mdr(7 downto 0)<=datain;
            mdr(15 downto 8)<=datain;
    else
            mdr<="ZZZZZZZZZZZZZZZZ";
    end if;
    end process;

    process(wrt,rdt,upc,mdr)
    begin
    if rdt='0' or upc='1' then
            mdrtemp<=mdr;
    end if;
    end process;

end Behavioral;
```

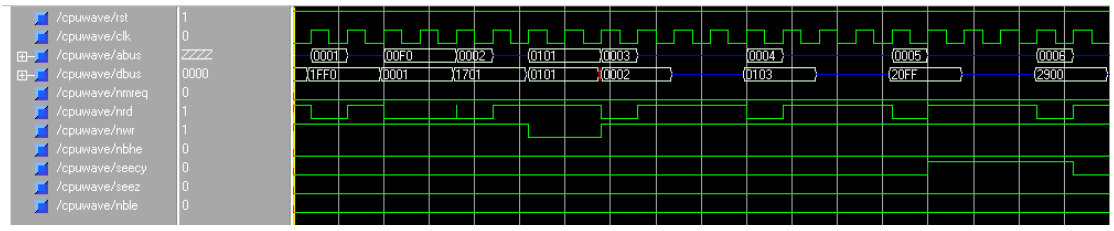
五、 实验测试及结果分析

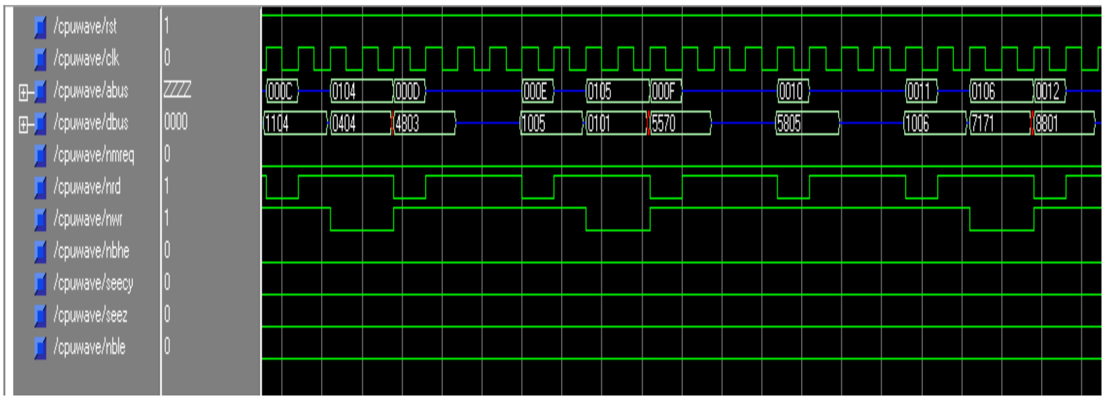


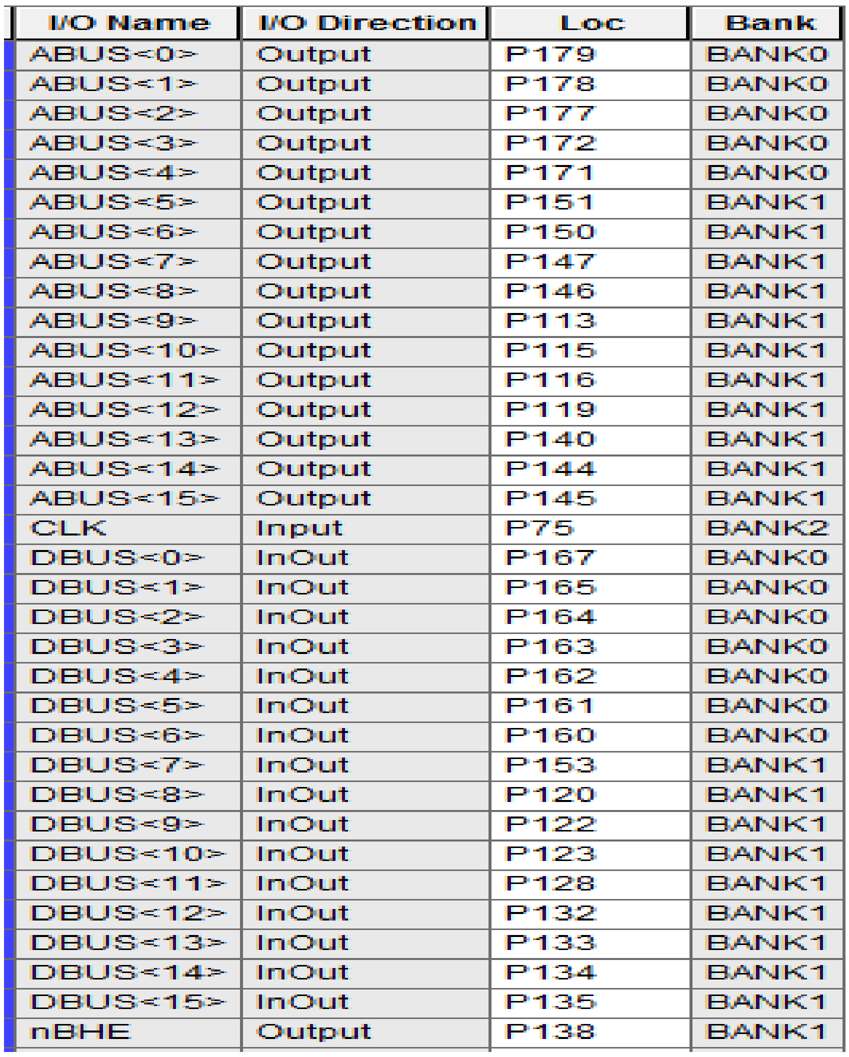
1. 测试波形



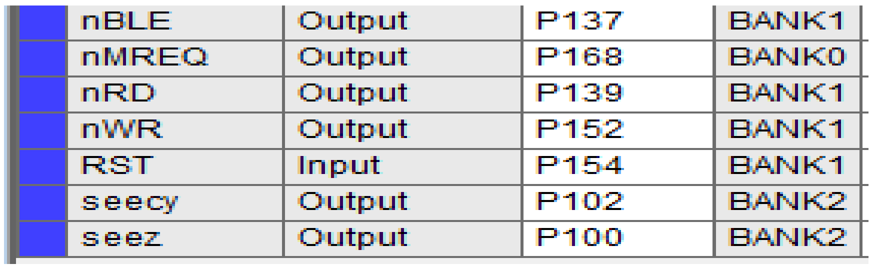
2. 编写.ucf 程序







I/O Name	I/O Direction	Loc	Bank
ABUS<0>	Output	P179	BANK0
ABUS<1>	Output	P178	BANK0
ABUS<2>	Output	P177	BANK0
ABUS<3>	Output	P172	BANK0
ABUS<4>	Output	P171	BANK0
ABUS<5>	Output	P151	BANK1
ABUS<6>	Output	P150	BANK1
ABUS<7>	Output	P147	BANK1
ABUS<8>	Output	P146	BANK1
ABUS<9>	Output	P113	BANK1
ABUS<10>	Output	P115	BANK1
ABUS<11>	Output	P116	BANK1
ABUS<12>	Output	P119	BANK1
ABUS<13>	Output	P140	BANK1
ABUS<14>	Output	P144	BANK1
ABUS<15>	Output	P145	BANK1
CLK	Input	P75	BANK2
DBUS<0>	InOut	P167	BANK0
DBUS<1>	InOut	P165	BANK0
DBUS<2>	InOut	P164	BANK0
DBUS<3>	InOut	P163	BANK0
DBUS<4>	InOut	P162	BANK0
DBUS<5>	InOut	P161	BANK0
DBUS<6>	InOut	P160	BANK0
DBUS<7>	InOut	P153	BANK1
DBUS<8>	InOut	P120	BANK1
DBUS<9>	InOut	P122	BANK1
DBUS<10>	InOut	P123	BANK1
DBUS<11>	InOut	P128	BANK1
DBUS<12>	InOut	P132	BANK1
DBUS<13>	InOut	P133	BANK1
DBUS<14>	InOut	P134	BANK1
DBUS<15>	InOut	P135	BANK1
nBHE	Output	P138	BANK1



nBLE	Output	P137	BANK1
nMREQ	Output	P168	BANK0
nRD	Output	P139	BANK1
nWR	Output	P152	BANK1
RST	Input	P154	BANK1
seecy	Output	P102	BANK2
seez	Output	P100	BANK2

3. 功能测试程序

十六进制指令	汇编指令	二进制指令	结果
1FF0	MOV R7，[F0]	00011 111 F0(16 进制)	R7=01
1701	MOV [01]，R7	00010 000 01(16 进制)	R7=01;[0101]=01
0002	MOV R0，02	00000 000 02(16 进制)	R0=02;
0103	MOV R1，03	00000 001 03(16 进制)	R1=03;
20FF	ADC R0, FF	00100 000 FF	R0=01;CY=1
2900	ADC R1, R0	00101 001 000 0(8	R1=03+01+CY=05;

		个)	CY=0;
1102	MOV [02],R1	00010 001 02(16 进制)	[0102]=05
3357	SBB R3，57	00110 011 57	R3=A9;CY=1
1303	MOV [03],R3	00010 011 03	[0103]=A9
3B01	SBB R3，R1	00111 011 00000 001	R3=A9-05-CY=A3; CY=0
4184	AND R1，84	01000 001 84	R1=04
1104	MOV [04],R1	00010 001 04	[0104]=04
4803	AND R0，R3	01001 000 00000 011	R0=01
1005	MOV [05], R0	00010 000 05	[0105]=01
5570	OR R5，70	01011 101 70	R5=70
5805	OR R0，R5	01011 000 00000 101	R0=71
1006	MOV [06], R0	00010 000 06	[0106]=71
8801	JC 01	10001 000 01	不跳
6800	STC	01101 0(11 个)	CY=1
8805	JC 05	10001 000 05	PC=PC+1+05
8883	JC 83	10001 000 83	PC=PC+1-03
6000	CLC		CY=0
7030	JMP 30	01110 000 30	PC=01(R7)//30
7800	JMP [R0]	01111 000 0(8 个)	PC=01(R7)// [01(R7)//71(R0)]
807A	JZ 7A	10000 000 7A	不跳
3901	SBB R1,R1	00111 001 00000 001	R1=0;Z=1
8006	JZ 06	10000 000 06	PC=PC+1+06
8083	JZ 83	10000 000 83	PC=PC+1-03
9602	MOV R6, [R7//(R0+02)]	10010 110 00000 02	R5=[01(R7)// (17(R0)+02)]=[0173]
1607	MOV [07],R6	00010 110 07	[0107]=(R6)

六、 实验总结

本次的 CPU 设计实验，了解了 vhdl，cpu 时序方式，各指令节拍控制知识，又在此基础上将各方面的理论知识汇总、整理、归纳成清晰的设计思路，最后在几个星期的努力下，完成了最终的 CPU 设计。

同时，本次实验大大加深了我们对 CPU 执行指令的整个工作流程的认识和理解，这是平时的理论学习远远不能及的。

在其中我遇到了很多的困难，特别是 process 里信号量的混乱赋值方面，以及变量和信号的理解。但是在这次实验中，我总结经验，虽然刚开始做实验时做的比较慢，但是后面由于有了经验，渐渐赶了回来，最后在期限之前完成了，很高兴。

最终，由于能力的有限，此次实验要求的指令比较简单，但是还是从中学习到了很多。