计算机设计与实践

——CPU 设计实验报告

学号：1130310128

学院：计算机科学与技术学院

姓名：杨尚斌

指导老师：罗丹彦

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 WHDL 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整体工作原理。

实验环境

开发软件

Xilinx ISE 9.1

ModelSIM 6.5

开发板

COP2000+实验台

实验设计

CPU 接口信息定义

信号名	位数	方向	来源/去向	意义
RST	1	in	外部复位信号	系统复位使能端
CLK	1	in	外部时钟	系统时钟
ABUS	16	out	存储器	地址总线
DBUS	16	inout	存储器	数据总线
nMREQ	1	out	存储器	存储器片选
nRD	1	out	存储器	存储器读
nWR	1	out	存储器	存储器写
nBHE	1	out	存储器	存储器高位有效
nBLE	1	out	存储器	存储器低位有效

CPU 设计方案

1. 指令格式设计

指令由操作码和地址码两部分组成，在指令系统中所有的指令都是二进制指令。

通用寄存器有 8 个，所以需要 3 为地址与之对应。

访存的时候形式地址为 8 位。

定义指令的高 5 位为操作码。

因此，做下面的定义：

OP(5)	AD1(3)		AD2(3)

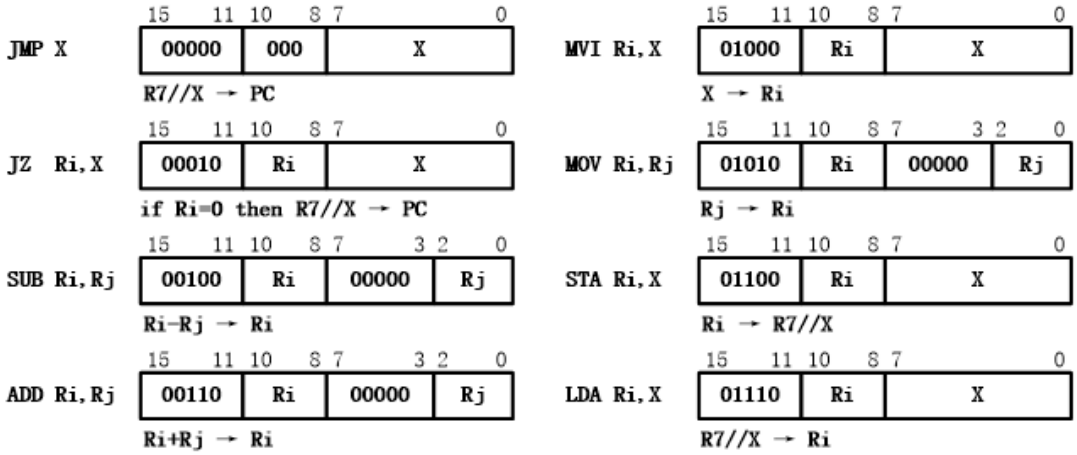
（寄存器-寄存器型指令）

OP（5）	AD1（3）	AD2(8)

（其他类型指令）

2. 微操作定义

指令名称	助记符	二进制操作码
加法操作	ADD	00000
减法操作	SUB	00001
寄存器传送	MOV	00010
立即数传送	MVI	10010
取数操作	LDA	11011
存数操作	STA	11000
无条件跳转操作	JMP	10001
条件跳转	JZ	10000

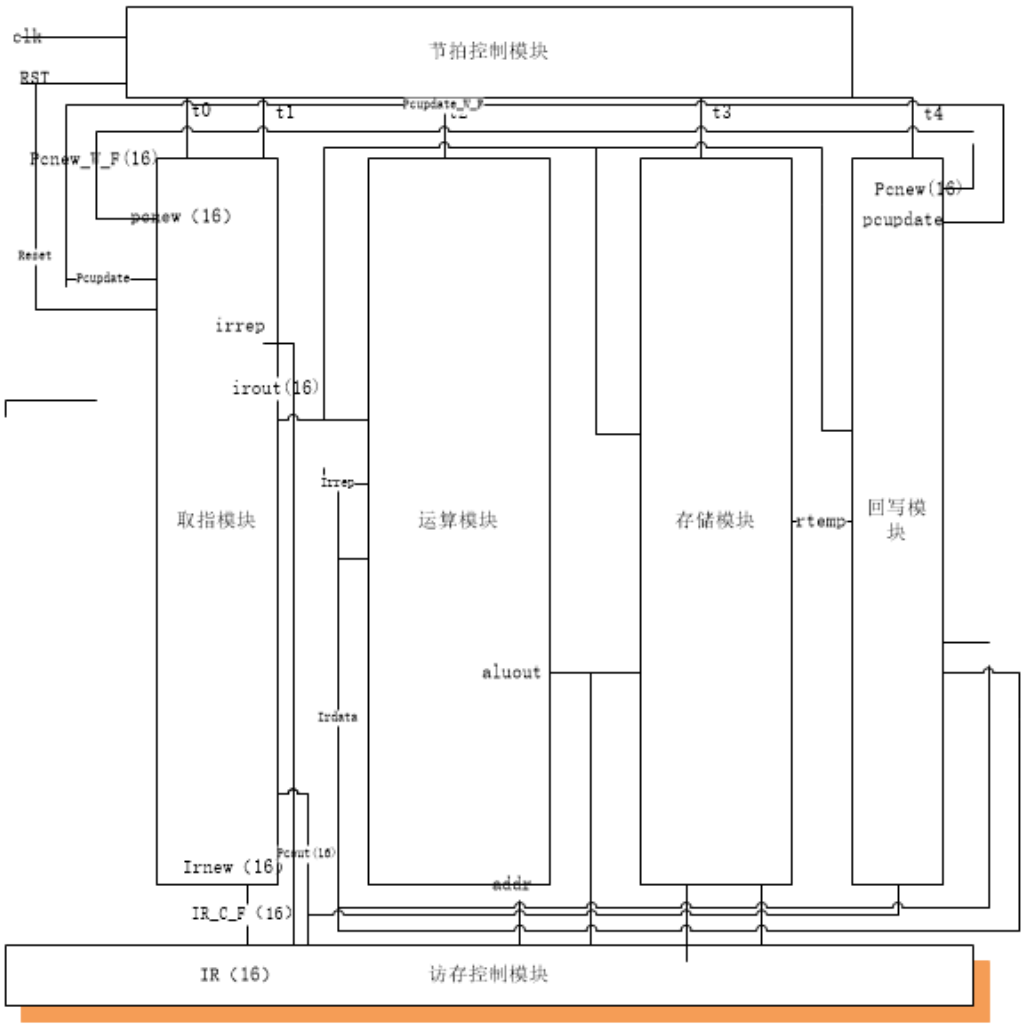


3. 节拍划分

共有 5 个节拍：

T0	取指
T1	PC+1
T2	运算器模块
T3	存储器模块
T4	回写

4. 处理器结构设计框图和功能描述



5. 各模块结构信号定义

a） 节拍计数器

模块说明：此模块为时钟模块，是一个 5 节拍的计数器。当 clk 信号产生一个脉冲的时候，t 就会循环的改变第 i 位的值，最后达到节拍的效果。

接口说明：

信号名	位数	方向	来源/去向	意义
clk	1	in	系统时钟	外接系统时钟
reset	1	in	系统复位	外接系统复位
t	5	out	各模块节拍	对外输出节拍（5 节拍）

b） 取指模块

模块说明：此模块为取指模块，负责取指令阶段，并且在取到指令之后对其他模块发出取到的指令 IR 和地址 PC，除此之外还接收回写模块发出的 pcupdate 和 pcnew 指令，用来回写 pc。此模块分为两个节拍进行，在 t0 = 1 的时候完成取指和 pc 的回写，在 t1 = 1 的时候完成 pc+1 的操作。

接口说明：

信号名	位数	方向	来源/去向	意义
Irnew:	16	In	访存控制 IR	接受从存储器中读取的新指令。
pcnew:	16	in	回写模块 PCnew	接受从回写模块发出的 pc 复写值。
clk：	1	In	系统时钟	接受系统时钟。
pcupdate：	1	In	回写模块 Pcupdate	pc 复写使能端。
reset：	1	In	系统复位	接受复位信号，置 pc=0。
t0、t1：	1	In	时钟模块 t	节拍信号。
irout：	16	Out	各模块 ir	对外输出指令 ir。
pcout：	16	Out	访存控制 PCout、回写模块 PCin	对外输出 pc 值。
irreq：	1	Out	访存控制 irreq	发出读取 ir 的使能信号。

c） 运算模块

模块说明：此模块为运算器模块 ALU，负责各种计算任务。当回写使能信号 enable_wb 为 1 的时候，寄存器 ir 按照回写模块传送过来的 reg_wb 进行更新。当 enable_t(t(2))有效的时候，开始运行计算任务，并且把最后计算得出的值传给 sig_reg7aluout，如果计算内容和地址有关系，则传 R7 与地址并起来给访存控制模块。

接口说明：

信号名	位数	方向	来源/去向	意义
enable_t	**1**	**In**	**时钟模块**	**驱动当前模块**
Ir	16	In	取指模块	判断当前的运算方法以及需要的值
Sig_reg7aluout	16	Out	访存控制模块和存储模块	暂时寄存器
Sig_reg7addrout	16	Out	访存控制模块	和 R7 并在一起，输入至访存控制模块
Enable_tb	1	In	回写模块	回写寄存器使能信号
Reg_wb	8	In	回写模块	要回写的寄存器的值

d) 存储模块

模块说明：此模块为存储管理模块，主要负责运算模块输出的值和取数时访存控制的值，用来向回写模块提供寄存器要回写的值，在 t3 = 1 的时候进行驱动。

接口说明：

信号名	位数	方向	来源/去向	意义
Aluout	7	In	运算模块的 Sig_reg7aluout，取低八位	接收 alu 传来的值。
IR	16	In	取指模块	接收取指模块传出的 ir。
Data	8	In	访存控制	接收取数时访存控制中的数据。
T	1	In	时钟模块 t	接收节拍。
Rtemp	8	Out	回写模块 Rtemp	对回写模块输出要回写的数值
nMRD	1	Out	访存控制模块	读标志
nMWR	1	Out	访存控制模块	写标志

e） 回写模块

模块说明：此模块为回写模块，主要用来对 Pc 和寄存器进行回写，当 t4 = 1 的时候开始执行。顺便对 JZ 操作在这里进行相关的判断。

信号说明：

信号名	位数	方向	来源/去向	意义
PCin	16	In	取指模块	接收取指模块传来的 PC
T	1	In	时钟模块	节拍控制
Rtemp	8	In	存储模块	接收存储模块的寄存器的值
Cy	1	In	运算模块	接收 ALU 传来的进位
Rupdate	1	Out	运算模块	更新寄存器使能信号
Rdata	8	Out	运算模块	寄存器要更新的值
PCupdate	1	Out	取指模块	Pc 更新使能信号
PCnew	16	Out	取指模块	PC 要更新的值

f） 访存控制模块

模块说明：此模块为访存控制模块，是 CPU 设计的核心模块。主要设计三部分，取指，取数，存数，分别由 irrep, nMRD, nMWR 驱动。当 Irrep = 1 时,主存片选有效，对主存 ABUS 输出地址，IR 得到 DBUS 的数据； 当 nMRD = 0 时，开始进行读数操作，主存片选有效，ABUS 输出地址，data 输出 DBUS 的低 8 位值；当 nMWR = 0 时，开始写数才做，主存片选有效，主存 ABUS 输出地址，DBUS 输出要写入的值。

信号说明：

信号名	位数	方向	来源/去向	意义
IRreq	1	In	取指模块	IR 使能
IR	16	Out	取指模块	对取指模块输出 IR
PCout	16	In	取指模块	接收取指指令
ALUOUT	8	In	运算模块	写数时使用

Addr	16	In	运算模块	读数时使用
ABUS	16	Out	主存储器	对主存输出地址
DBUS	16	Ino ut	主存储器	数据总线
nWR	1	Out	主存储器	写主存使能信号
nRD	1	Out	主存储器	读主存使能信号
nMREQ	1	Out	主存储器	片选信号
NBHE	1	Out	主存储器	高字节允许访问
nBLE	1	Out	主存储器	低字节允许访问
nMWR	1	In	存储模块	写数使能
nWRD	1	In	存储模块	读数使能
Data	8	Out	存储模块	对存储模块输出取到的数据

g） 总体设计

信号说明:

信号名	位数	方向	来源/去向	意义
RST	1	in	外部复位信号	系统复位使能端
CLK	1	in	外部时钟	系统时钟
ABUS	16	out	存储器	地址总线
DBUS	16	inou t	存储器	数据总线
nMREQ	1	out	存储器	存储器片选
nRD	1	out	存储器	存储器读
nWR	1	out	存储器	存储器写
nBHE	1	out	存储器	存储器高位有效
nBLE	1	out	存储器	存储器低位有效

元件例化代码：

```
signal t : STD_LOGIC_VECTOR(4 downto 0);          --正常节拍
signal IR_C_F : STD_LOGIC_VECTOR(15 downto 0);    --  取指模块取出的 ir
signal PCout_F_CW : STD_LOGIC_VECTOR(15 downto 0);   -- PC 送往访存控制取指，送回写模块
signal PCnew_W_F : STD_LOGIC_VECTOR(15 downto 0);      -- 跳转的时候要更新的 PC
signal PCupdate_W_F : STD_LOGIC;                       -- 跳转更新 PC 使能信号
signal irout_F_ASW : STD_LOGIC_VECTOR(15 downto 0);   --取指模块取到的 Ir,会送往 ALU 存储 和回写
```

```
signal irreq_F_C : STD_LOGIC;                    -- 取指送往访存控制，告诉要取指令了
signal ALUOUT_A_CS : STD_LOGIC_VECTOR(15 downto 0);    ---ALU 送往其他模块的 aluout
signal Addr_A_C : STD_LOGIC_VECTOR(15 downto 0);       --- ALU 送往访存的 addr
signal Rupdate_W_A : STD_LOGIC;                        ---回写模块送往 ALU 的更改寄存器使能信号
signal Rdata_W_A : STD_LOGIC_VECTOR(7 downto 0);       ----回写模块输出的要更新的寄存器的值
signal data_C_S : STD_LOGIC_VECTOR(7 downto 0);        -- 取数的时候使用
signal nMWR_S_C : STD_LOGIC;                           --写数使能
signal nMRD_S_C : STD_LOGIC;                           --读数使能
signal Rtemp_S_W : STD_LOGIC_VECTOR(7 downto 0);       -- 存储模块向回写模块
signal cy_A_W : STD_LOGIC;                             --进位

    u1: clock port map(CLK, RST, t);
  u2: fetch port map(IR_C_F, PCnew_W_F, CLK , PCupdate_W_F, RST, t(0), t(1), irout_F_ASW, PCout_F_CW, irreq_F_C);
  u3: ALU port map(t(2), irout_F_ASW, ALUOUT_A_CS, Addr_A_C, Rupdate_W_A, Rdata_W_A, cy_A_W);
  u4: control port map(irreq_F_C, IR_C_F, PCout_F_CW, ALUOUT_A_CS(7 downto 0), Addr_A_C, ABUS, DBUS, nWR, nRD, nMREQ, nBHE, nBLE, nMWR_S_C, nMRD_S_C , data_C_S);
  u5: save port map(t(3), ALUOUT_A_CS(7 downto 0), data_C_S, nMWR_S_C, irout_F_ASW, nMRD_S_C, Rtemp_S_W);
  u6: write_back port map(PCout_F_CW, t(4), Rtemp_S_W, irout_F_ASW, cy_A_W, Rupdate_W_A, Rdata_W_A, PCupdate_W_F, PCnew_W_F);
```

6. 波形仿真

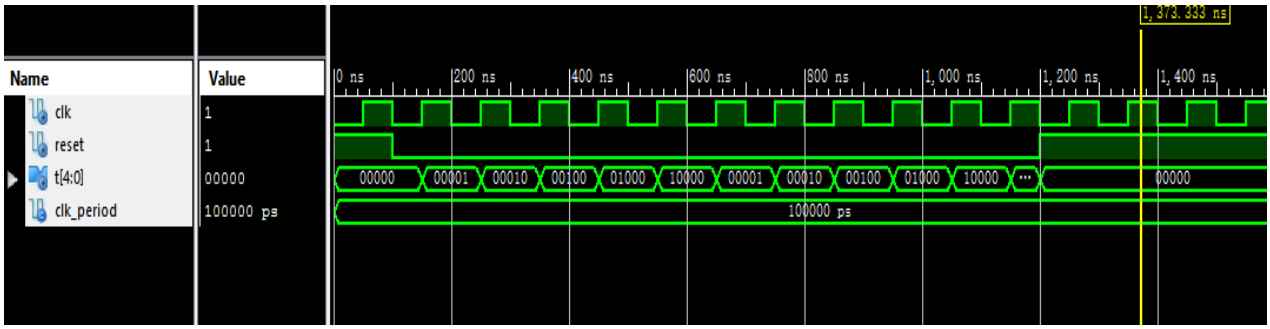
a） 节拍计数器

核心代码：

```
entity clock is
    Port ( clk : in    STD_LOGIC;
           reset : in    STD_LOGIC;
           t : out    STD_LOGIC_VECTOR (4 downto 0));
end clock;
process(clk, reset)
    variable tep : integer range 0 to 6 := 0;
    begin
```

```
        if(reset = '1') then
            t <= "00000";
            tep := 0;
        elsif (clk = '1' and clk' event) then
            tep := tep + 1;
            if tep = 6 then
                tep := 1;
            end if;
            case tep is
                when 1 => t <= "00001";
                when 2 => t <= "00010";
                when 3 => t <= "00100";
                when 4 => t <= "01000";
                when 5 => t <= "10000";
                when others => NULL;
            end case;
        end if;
    end process;
```

仿真波形：



b）取指模块

核心代码：

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

```
entity fetch is
    Port ( irnew : in    STD_LOGIC_VECTOR (15 downto 0);          --访存模块输入的
IR
           pcnew : in    STD_LOGIC_VECTOR (15 downto 0);          --回写模块，
更新 PC
           clk : in    STD_LOGIC;                                 --节拍
           pcupdate : in    STD_LOGIC;                           --告诉要更新
PC 了
           reset : in    STD_LOGIC;                              --复位
           t0 : in    STD_LOGIC;
           t1 : in    STD_LOGIC;
           irout : out    STD_LOGIC_VECTOR (15 downto 0);        --输出的 IR
           pcout : out    STD_LOGIC_VECTOR (15 downto 0);
           irrep : out    STD_LOGIC);                            --使能
end fetch;

architecture Behavioral of fetch is
signal pc : STD_LOGIC_VECTOR (15 downto 0);
begin
    process(clk, reset, t0, t1, pcupdate)
        begin
            if reset = '1' then
                irrep <= '0';
                pc <= "0000000000000000";
            elsif t0 = '1' then
                irrep <= '1';
                irout <= irnew;
            elsif t1 = '1' then
                irrep <= '0';
                if (clk = '1' and clk' event) then
                    pc <= pc + 1;
                end if;
            elsif pcupdate = '1' then
                pc <= pcnew;
                irrep <= '0';

            end if;
            pcout <= pc;
    end process;
end Behavioral;
```

仿真波形：


95.500 ns
Name
Value
irnew[15:0]
0000000000000000
pcnew[15:0]
0000000000000000
clk
1
pcupdate
0
reset
0
t0
0
t1
0
irout[15:0]
pcout[15:0]
irrep
clk_period
100000 ps
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
348.243 ns
250 ns
300 ns
350 ns
400 ns
450 ns
500 ns
1111000011110000
0000000011111111
0000000000000000
00000000
100000 ps
584.353 ns
500 ns
550 ns
600 ns
650 ns
700 ns
750 ns
800 ns
850 ns
1111000011110000
0000000011111111
0000000000000001
0000000011111111
0000000000000000
100000 ps

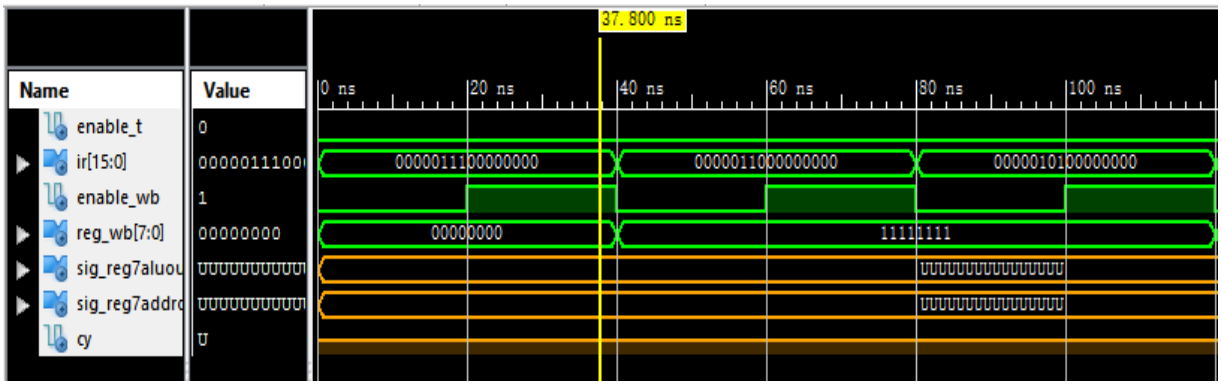
c） 运算模块

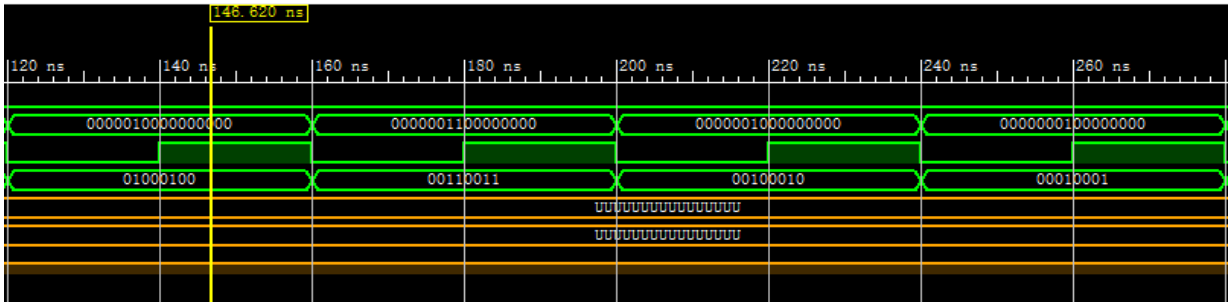
核心代码：

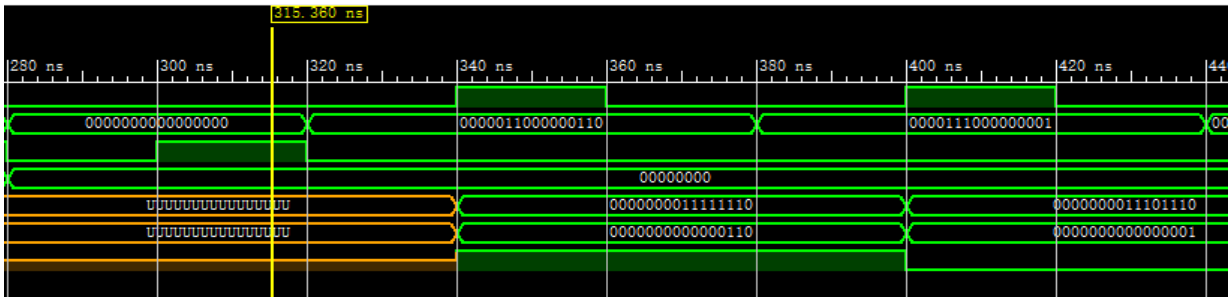
```
entity ALU is
port(
        -- 实现准备和运算功能
        enable_t : in std_logic ; -- 准备和运算功能使能信号
        ir : in std_logic_vector(15 downto 0);

        -- 向访存控制模块输出
        sig_reg7aluout : out std_logic_vector ( 15 downto 0 ); -- 暂存器输出端口
        sig_reg7addrout : out std_logic_vector ( 15 downto 0 ); -- 8 位地址输出端
口
        --reg7_out : out std_logic_vector ( 7 downto 0 );

        -- 实现回写功能
        enable_wb : in std_logic ; -- 回写功能使能
        reg_wb : in std_logic_vector (7 downto 0 );   -- 回写接收端口

        -- 进位标志
        cy : out std_logic
        );

end ALU;

architecture Behavioral of ALU is

type registers_8 is array (7 downto 0) of std_logic_vector(7 downto 0);
signal reg : registers_8; -- 数组型 8 个 8 位寄存器
signal addr : std_logic_vector ( 7 downto 0 ); -- 暂存器
begin
get_ready : process (enable_t,addr)
variable a,b : std_logic_vector ( 7 downto 0 );
variable tempa , tempb ,tempsum: std_logic_vector (8 downto 0 ); -- 进位标志计
算

begin
a := reg(conv_integer(ir(10 downto 8)));
b := reg(conv_integer(ir(2 downto 0)));
addr <= ir( 7 downto 0 );
tempa := '0'&a;
tempb := '0'&b;
      if enable_t = '1'   then
            case ir(15 downto 11) is
                  when "00000"=>   tempsum := tempa + tempb ;              --ADD
                  when "00001"=>     tempsum := tempa - tempb ;              --SUB
```

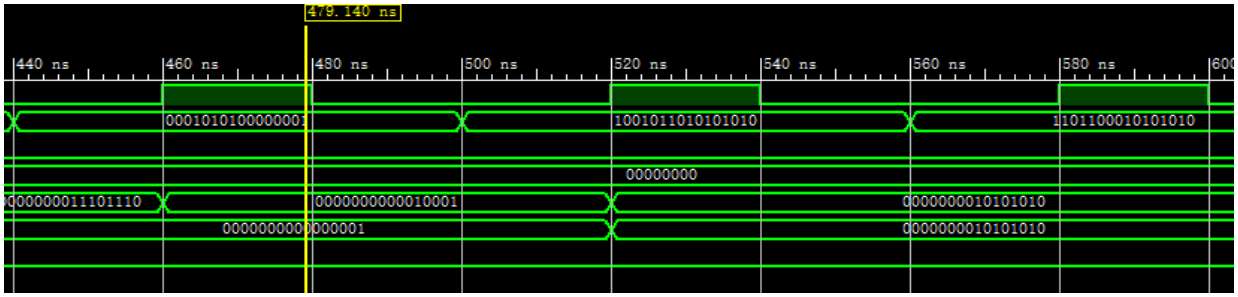
```
                when "00010"=>      tempsum := tempb;                    --
MOV
                when "10010"=>    tempsum := '0'&addr;                 --MVI
                when "11011"=>      tempsum := '0'&addr;                 --LDA
                when "11000"=>      tempsum := tempa;                    --STA
                when "10001"=>      tempsum := '0'&addr;                 --JMP
                when "10000"=>    tempsum := tempa;                    --JZ
                --when "11111"=>   tempsum := '0'&addr;                 --IN
                --when "11100"=>   tempsum :=tempa;                     --
OUT
                when others=>       tempsum :="ZZZZZZZZZ";
            end case;
        sig_reg7aluout <= reg(7)&tempsum ( 7 downto 0 );
        cy <= tempsum (8) ;
        sig_reg7addrout <= reg(7)&addr;
        end if ;
end process;

write_back : process (reg_wb,enable_wb)
begin
        if enable_wb = '1' then
            reg(conv_integer(ir(10 downto 8))) <= reg_wb;
        end if;
end process;

end Behavioral;
```

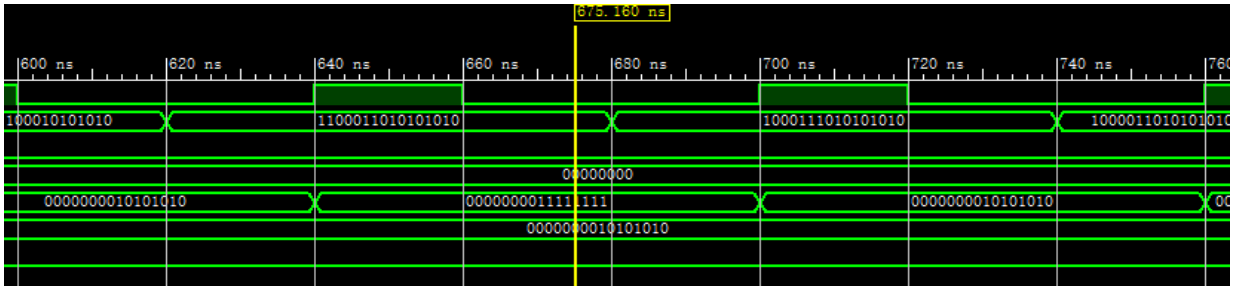
仿真波形

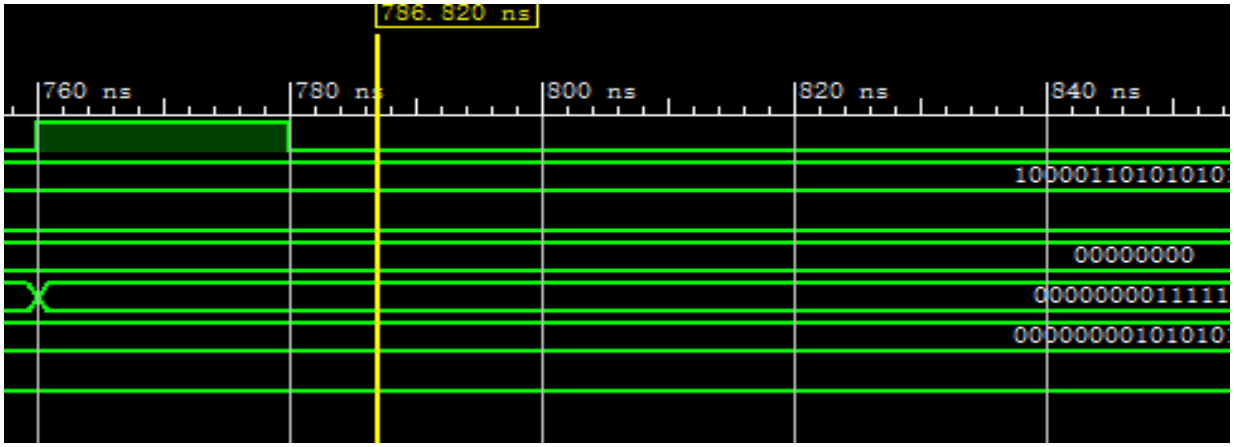



146.620 ns
120 ns
140 ns
160 ns
180 ns
200 ns
220 ns
240 ns
260 ns
01000100
00110011
00100010
00010001


315.360 ns
280 ns
300 ns
320 ns
340 ns
360 ns
380 ns
400 ns
420 ns
00000000


479.140 ns
440 ns
460 ns
480 ns
500 ns
520 ns
540 ns
560 ns
580 ns
00000000


675.160 ns
600 ns
620 ns
640 ns
660 ns
680 ns
700 ns
720 ns
740 ns
00000000

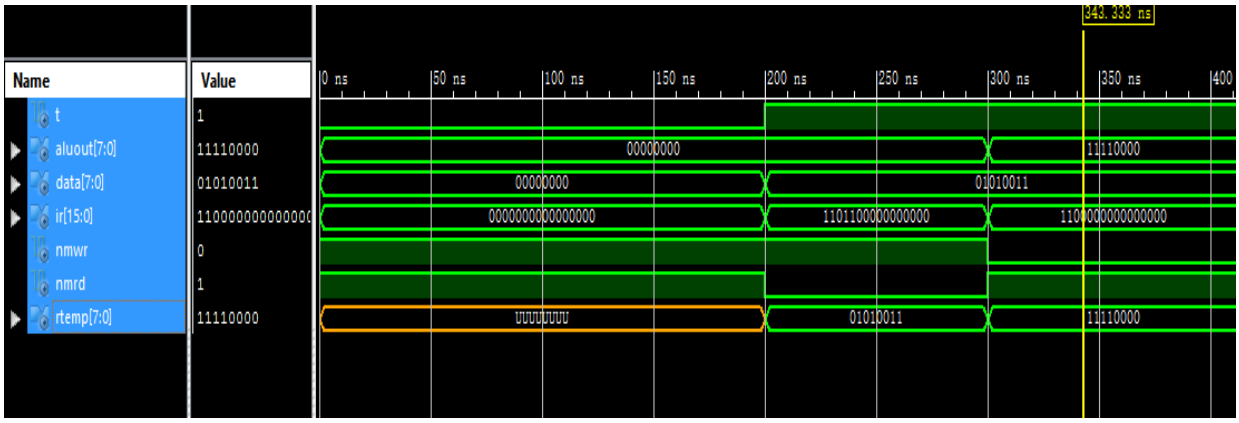

786.820 ns
760 ns
780 ns
800 ns
820 ns
840 ns
00000000

d) 存储模块

核心代码：

```
entity save is
    Port (
        t : in    STD_LOGIC;
        ALUOUT : in std_logic_vector(7 downto 0);          --- 取数的时候用
        data : in std_logic_vector(7 downto 0);            --- 接收区属的时候访存控制的数据
        nMWR : out std_logic;
      IR : in    STD_LOGIC_VECTOR (15 downto 0);
      nMRD : out    STD_LOGIC;
        Rtemp : out std_logic_vector(7 downto 0));       --- 回写模块要输
end save;

architecture Behavioral of save is
begin
    process(t,data,ALUOUT,IR)
    begin
        if t = '1' then -- LDA 与 STA
            case IR(14 downto 12) is
                when "100" =>                --- STA     存数  11000
                    nMWR <= '0';
                    nMRD <= '1';
                    Rtemp(7 downto 0) <= ALUOUT(7 downto 0);
                when "101" =>                --- LDA      取数    11011
                    nMWR <= '1';
                    nMRD <= '0';
                    Rtemp(7 downto 0) <= data(7 downto 0);
                when others =>
                    nMWR <= '1';
                    nMRD <= '1';
                    Rtemp(7 downto 0) <= ALUOUT(7 downto 0);
            end case;
        else
            nMWR <= '1';
            nMRD <= '1';

        end if;

    end process;
end Behavioral;
```

仿真波形：



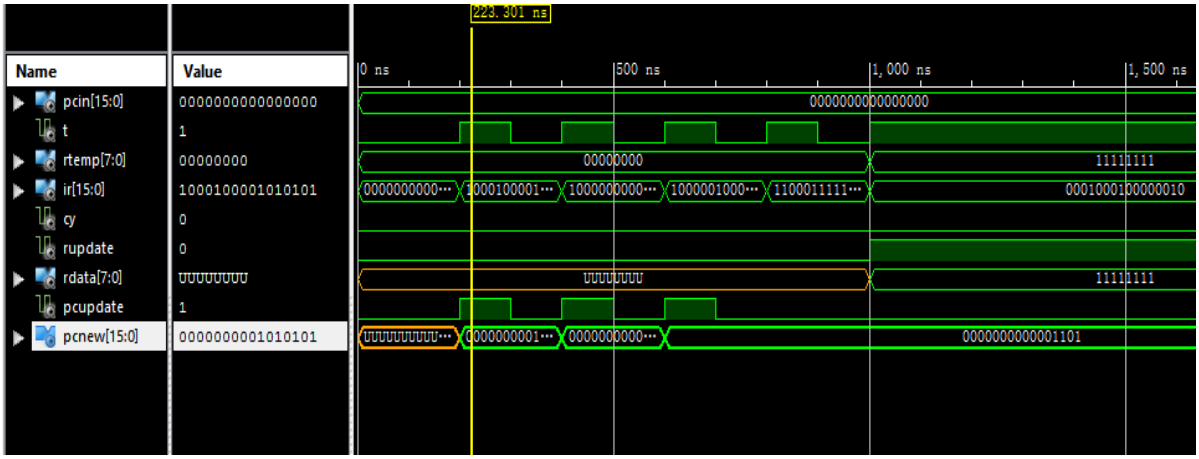
e） 回写模块

核心代码：

```
entity write_back is
Port (
      PCin:in std_logic_vector(15 downto 0);                --接收取指模块传出的
PC，用于 0 跳转和直接跳转
      t : in    STD_LOGIC;                                    -- 回写使能
    Rtemp : in    STD_LOGIC_VECTOR (7 downto 0);           -- 接收来自存储管理
模块的寄存器
      IR : in    STD_LOGIC_VECTOR (15 downto 0);           -- 接收取指模块传出
的 IR
    --z:in STD_LOGIC;                                       --接收 ALU 传出的 z
    cy:in STD_LOGIC;                                       --接收 ALU 传出的进
位
      Rupdate : out    STD_LOGIC;                           -- 寄存器回写使能
信号
    Rdata : out    STD_LOGIC_VECTOR (7 downto 0);          -- ALU 输出的寄存器回
写数据
    PCupdate : out    STD_LOGIC;                           -- PC 回写使能型号
      PCnew : out    STD_LOGIC_VECTOR (15 downto 0)        --输出 PC 回写的值
);
end write_back;

architecture Behavioral of write_back is
--signal tempa:std_logic_vector(15 downto 0);
--signal tempb:std_logic_vector(15 downto 0);
begin
  -- tempa<="00000000"&(IR(7 downto 0));
  process(t, cy, IR)
  begin
       if t='1' then
```

```
        case IR(15 downto 11) is
            when "10001" =>
                Rupdate <= '0';    --jmp
               PCupdate <= '1';
               PCnew <= "00000000"&(IR(7 downto 0));
            when "10000" =>          --jz
                Rupdate <= '0';
                if (Rtemp(7 downto 0) = "00000000") then
                --if    z='1' then
                        PCnew <= "00000000"&(IR(7 downto 0));
                        PCupdate <= '1';
             else
                        PCupdate<='0';

               end if;
            when "11000" =>null;--STA
            when others =>
                Rupdate<='1';
               PCupdate <= '0';
                Rdata(7 downto 0)<= Rtemp(7 downto 0);
        end case;
    else PCupdate<='0';Rupdate<='0';
    end if;
end process;
end Behavioral;
```

仿真波形：



f） 访存控制模块

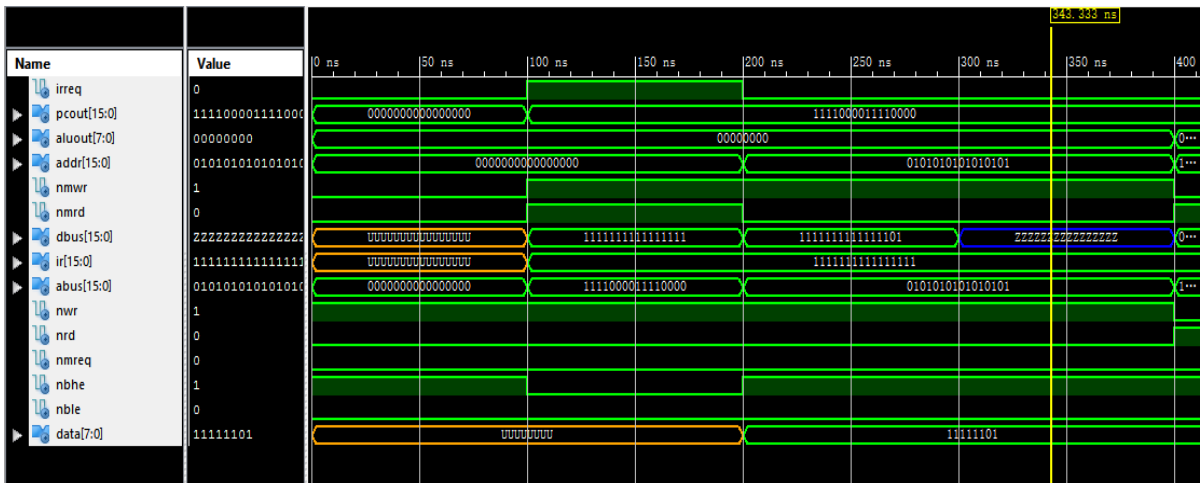
核心代码：

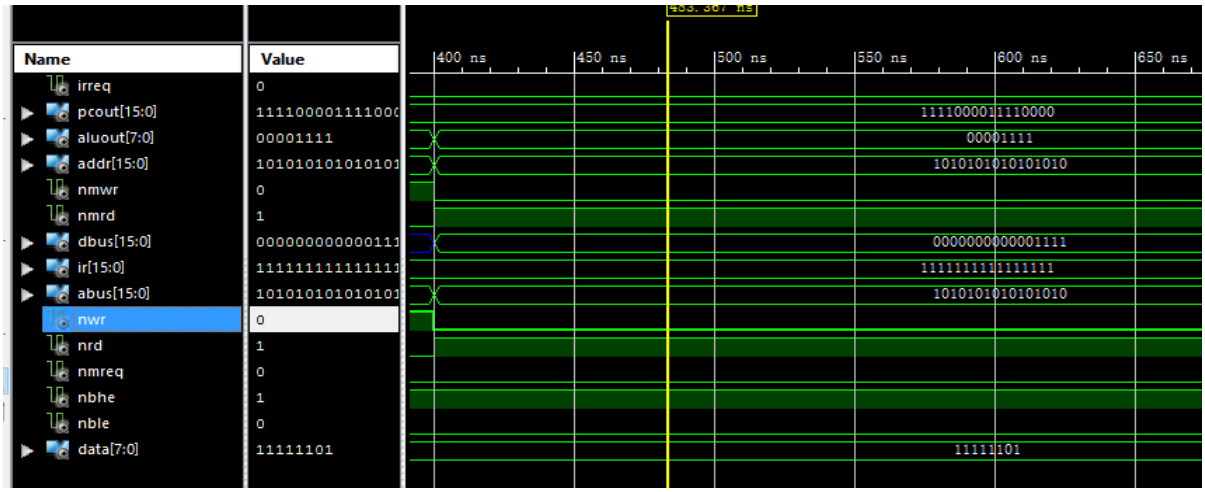
```
entity control is
```

```
port(
    IRreq :in STD_LOGIC;                                    --ir 使能
    IR:out    STD_LOGIC_VECTOR (15 downto 0);               --对取指模块输出 ir
    PCout: in    STD_LOGIC_VECTOR (15 downto 0);            --接收取指指令

    ALUOUT : in    STD_LOGIC_VECTOR (7 downto 0);           --运算模块
    Addr : in    STD_LOGIC_VECTOR (15 downto 0);            --运算模块

    ABUS : out    STD_LOGIC_VECTOR (15 downto 0);           --对主存输出地址
    DBUS : inout    STD_LOGIC_VECTOR (15 downto 0); --数据总线

    --给主存发
    nWR : out    STD_LOGIC;                                 --写主存使能
    nRD : out    STD_LOGIC;                                 --读主存使能
    nMREQ : out    STD_LOGIC;                               --主存片选信号
    nBHE : out    STD_LOGIC;                                --主存高八位控制信号
    nBLE : out    STD_LOGIC;                                --主存低八位控制信号
    --运算模块/取指模块给出，要不要访内存
    nMWR : in    STD_LOGIC;                                 --ALU 写数使能
    nMRD : in    STD_LOGIC;                                 --ALU 取数使能
    --来自存储模块
    data : out    STD_LOGIC_VECTOR (7 downto 0)             --对存储控制输出取到的
数据。
);
end control;
architecture Behavioral of control is

begin
    --IR <= DBUS;
     --DBUS<="00000000"&ALUOUT when nMWR='0' else "ZZZZZZZZZZZZZZZZ";
     --data <= DBUS(7 downto 0) when nMRD = '0' else "ZZZZZZZZ";
     --ABUS<=PCout when IRreq='1' else Addr;
process(IRreq,nMRD,nMWR)
     begin
          DBUS<="ZZZZZZZZZZZZZZZZ";
        if IRreq ='1' then      --取指模块
                nBHE <= '0';
                nBLE <= '0';--高低位
                nMREQ <= '0';
                nWR <= '1';
                nRD <= '0';--读有效，低电位有效
                IR <= DBUS;
            --DBUS<="ZZZZZZZZZZZZZZZZ";
                ABUS<=PCout;
```

```
        elsif nMRD = '0' then --需要读内存(运算模块)---读取使能，低电平有效
            nBHE <= '1';
            nBLE <= '0';
            nMREQ <= '0';
            nRD <= '0';      --?劣行?
            nWR <= '1';
            ABUS <= Addr;
            --data <= "00100101";
            data<=DBUS(7 downto 0);
            --DBUS<="ZZZZZZZZZZZZZZZZ";
        elsif nMWR = '0' then --写内存(运算模块)
            nBHE <= '1';
            nBLE <= '0';
            nMREQ <= '0';
            nRD <= '1';
            nWR <= '0';--写有效
             DBUS<="00000000"&ALUOUT;
             ABUS<=Addr;
        else
            nBHE <= '1';
            nBLE <= '1';
            nMREQ <= '1';
            nRD <= '1';
            nWR <= '1';
             data<="ZZZZZZZZ";
             DBUS<="ZZZZZZZZZZZZZZZZ";
    end if;
end process;
end Behavioral;
```

仿真波形：





g） 总体波形仿真

核心代码：

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity CPU is
    Port ( RST : in    STD_LOGIC;
           CLK : in    STD_LOGIC;
           ABUS : out    STD_LOGIC_VECTOR (15 downto 0);
           DBUS : inout    STD_LOGIC_VECTOR (15 downto 0);
           nMREQ : out    STD_LOGIC;
           nRD : out    STD_LOGIC;
           nWR : out    STD_LOGIC;
           nBHE : out    STD_LOGIC;
           nBLE : out    STD_LOGIC);
end CPU;

architecture Behavioral of CPU is
    component clock is
        port (
            clk : in    STD_LOGIC;
```

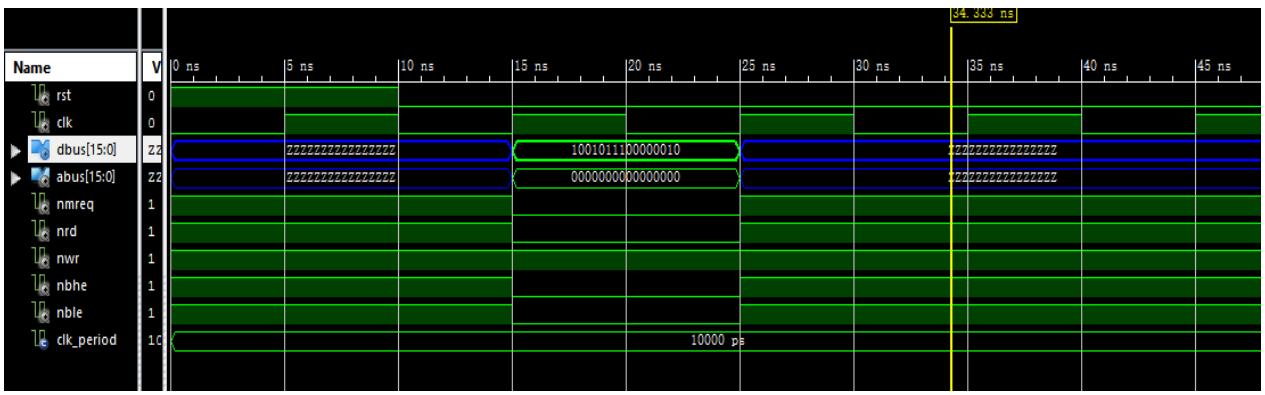
```
            reset : in    STD_LOGIC;
            t : out    STD_LOGIC_VECTOR (4 downto 0)
           );
    end component;

    component fetch is
        port (
            irnew : in    STD_LOGIC_VECTOR (15 downto 0);          --访存模
块输入的 IR
            pcnew : in    STD_LOGIC_VECTOR (15 downto 0);          --回写模
块，更新 PC
            clk : in    STD_LOGIC;                                 --节拍
            pcupdate : in    STD_LOGIC;                            --告诉要更
新 PC 了
            reset : in    STD_LOGIC;                               --复位
            t0 : in    STD_LOGIC;
            t1 : in    STD_LOGIC;
            irout : out    STD_LOGIC_VECTOR (15 downto 0);        --输出的 IR
            pcout : out    STD_LOGIC_VECTOR (15 downto 0);
            irrep : out    STD_LOGIC
           );
    end component;

    component ALU is
        port (
           -- 实现准备和运算功能
           enable_t : in std_logic; -- 准备和运算功能使能信号
            ir : in std_logic_vector(15 downto 0);     --16 位的 IR 信号

            -- 向访存控制模块输出
            sig_reg7aluout : out std_logic_vector ( 15 downto 0 ); -- 暂存器输
出端口
            sig_reg7addrout : out std_logic_vector ( 15 downto 0 ); -- 8 位地址
输出端口
            --reg7_out : out std_logic_vector ( 7 downto 0 );

            -- 实现回写功能
            enable_wb : in std_logic ; -- 回写功能使能
            reg_wb : in std_logic_vector (7 downto 0 );    -- 回写接收端口

            -- 进位标志
            cy : out std_logic
        );
    end component;
```

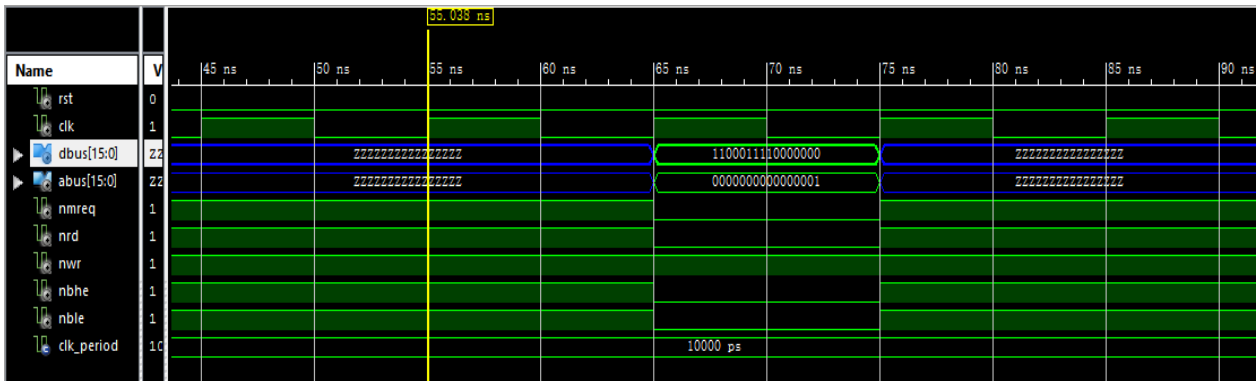
```
    component control is
        port (
            IRreq :in STD_LOGIC;                                    --ir 使能
            IR:out    STD_LOGIC_VECTOR (15 downto 0);           --对取指模块
输出 ir
            PCout: in    STD_LOGIC_VECTOR (15 downto 0);       --接收取指指
令

            ALUOUT : in    STD_LOGIC_VECTOR (7 downto 0);     --运算模块
            Addr : in    STD_LOGIC_VECTOR (15 downto 0);       --运算模块

            ABUS : out    STD_LOGIC_VECTOR (15 downto 0);     --对主存输出
地址
            DBUS : inout    STD_LOGIC_VECTOR (15 downto 0); --数据总线

            --给主存发
            nWR : out    STD_LOGIC;                                  --写主存使
能
            nRD : out    STD_LOGIC;                                  --读主存使
能
            nMREQ : out    STD_LOGIC;                                --主存片选
信号
            nBHE : out    STD_LOGIC;                                 --主存高八
位控制信号
            nBLE : out    STD_LOGIC;                                 --主存低八
位控制信号
            --运算模块/取指模块给出，要不要访内存
            nMWR : in    STD_LOGIC;                                   --ALU 写
数使能
            nMRD : in    STD_LOGIC;                                   --ALU 取数
使能
            --来自存储模块
            data : out    STD_LOGIC_VECTOR (7 downto 0)        --对存储控制
输出取到的数据。
        );
    end component;

    component save is
        port (
            t : in    STD_LOGIC;
            ALUOUT : in std_logic_vector(7 downto 0);         --- ALU 输出的值
            data : in std_logic_vector(7 downto 0);            ---  接收区属的时
候访存控制的数据
```

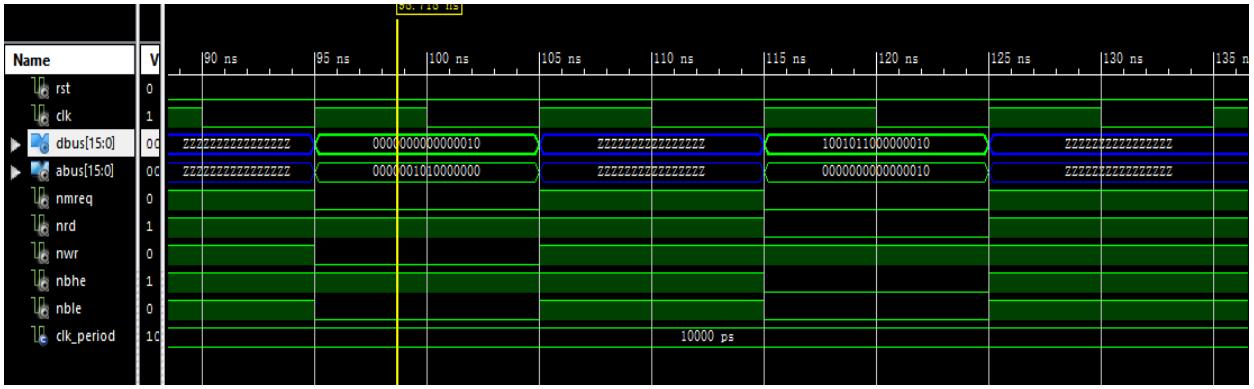
```
            nMWR : out std_logic;
            IR : in    STD_LOGIC_VECTOR (15 downto 0);
            nMRD : out    STD_LOGIC;
            Rtemp : out std_logic_vector(7 downto 0)
        );
    end component;

    component write_back is
        port (
            PCin:in std_logic_vector(15 downto 0);                --接收取指模块传出的 PC，用于 0 跳转和直接跳转
            t : in    STD_LOGIC;                                  -- 回写使能
            Rtemp : in    STD_LOGIC_VECTOR (7 downto 0);          -- 接收来自存储管理模块的寄存器
            IR : in    STD_LOGIC_VECTOR (15 downto 0);            -- 接收取指模块传出的 IR
            --z:in STD_LOGIC;                                     --接收 ALU 传出的 z
            cy:in STD_LOGIC;                                      --接收 ALU 传出的进位
            Rupdate : out    STD_LOGIC;                           -- 寄存器回写使能信号
            Rdata : out    STD_LOGIC_VECTOR (7 downto 0);         -- ALU 输出的寄存器回写数据
            PCupdate : out    STD_LOGIC;                          -- PC 回写使能型号
            PCnew : out    STD_LOGIC_VECTOR (15 downto 0)         --输出 PC 回写的值
        );
    end component;

signal t : STD_LOGIC_VECTOR(4 downto 0);          --正常节拍
signal IR_C_F : STD_LOGIC_VECTOR(15 downto 0);    --    取指模块取出的 ir
signal PCout_F_CW : STD_LOGIC_VECTOR(15 downto 0);    -- PC 送往访存控制取指，送回写模块
signal PCnew_W_F : STD_LOGIC_VECTOR(15 downto 0);      -- 跳转的时候要更新的 PC
signal PCupdate_W_F : STD_LOGIC;                       -- 跳转更新 PC 使能信号
signal irout_F_ASW : STD_LOGIC_VECTOR(15 downto 0);    --取指模块取到的 Ir,会送往 ALU 存储 和回写
signal irreq_F_C : STD_LOGIC;                          -- 取指送往访存控制，告诉要取指令了
```

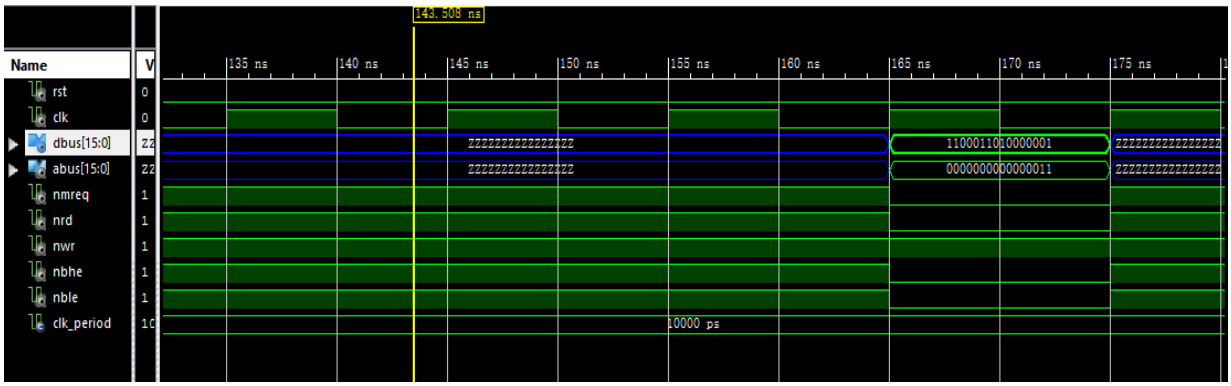
```
signal ALUOUT_A_CS : STD_LOGIC_VECTOR(15 downto 0);      ---ALU 送往其他模块的 aluout
signal Addr_A_C : STD_LOGIC_VECTOR(15 downto 0);         --- ALU 送往访存的 addr
signal Rupdate_W_A : STD_LOGIC;                          ---回写模块送往 ALU 的更改寄存器使能信号
signal Rdata_W_A : STD_LOGIC_VECTOR(7 downto 0);         ----回写模块输出的要更新的寄存器的值
signal data_C_S : STD_LOGIC_VECTOR(7 downto 0);          -- 取数的时候使用
signal nMWR_S_C : STD_LOGIC;                             --写数使能
signal nMRD_S_C : STD_LOGIC;                             --读数使能
signal Rtemp_S_W : STD_LOGIC_VECTOR(7 downto 0);         -- 存储模块向回写模块
signal cy_A_W : STD_LOGIC;                               --进位

begin

    u1: clock port map(CLK, RST, t);
    u2: fetch port map(IR_C_F, PCnew_W_F, CLK , PCupdate_W_F, RST, t(0), t(1), irout_F_ASW, PCout_F_CW, irreq_F_C);
    u3: ALU port map(t(2), irout_F_ASW, ALUOUT_A_CS, Addr_A_C, Rupdate_W_A, Rdata_W_A, cy_A_W);
    u4: control port map(irreq_F_C, IR_C_F, PCout_F_CW, ALUOUT_A_CS(7 downto 0), Addr_A_C, ABUS, DBUS, nWR, nRD, nMREQ, nBHE, nBLE, nMWR_S_C, nMRD_S_C , data_C_S);
    u5: save port map(t(3), ALUOUT_A_CS(7 downto 0), data_C_S, nMWR_S_C, irout_F_ASW, nMRD_S_C, Rtemp_S_W);
    u6: write_back port map(PCout_F_CW, t(4), Rtemp_S_W, irout_F_ASW, cy_A_W, Rupdate_W_A, Rdata_W_A, PCupdate_W_F, PCnew_W_F);

end Behavioral;
```

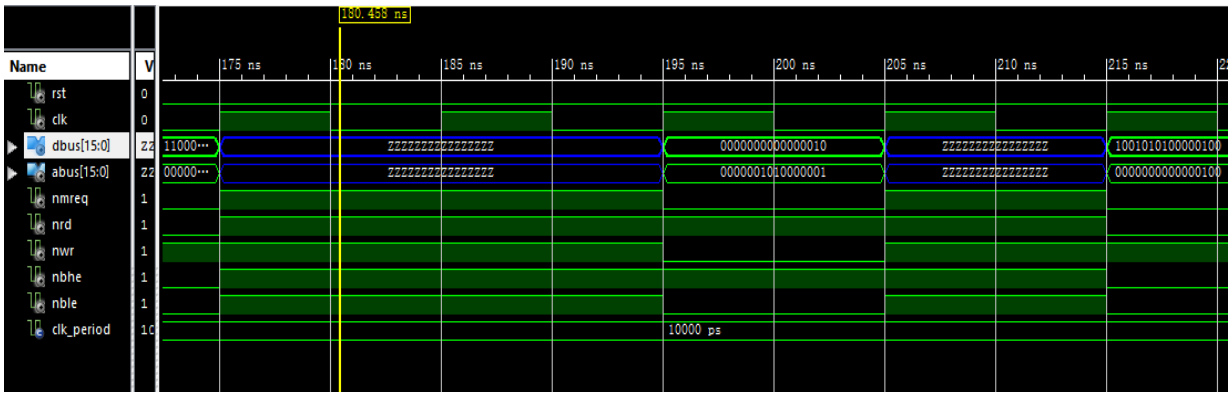
仿真波形：

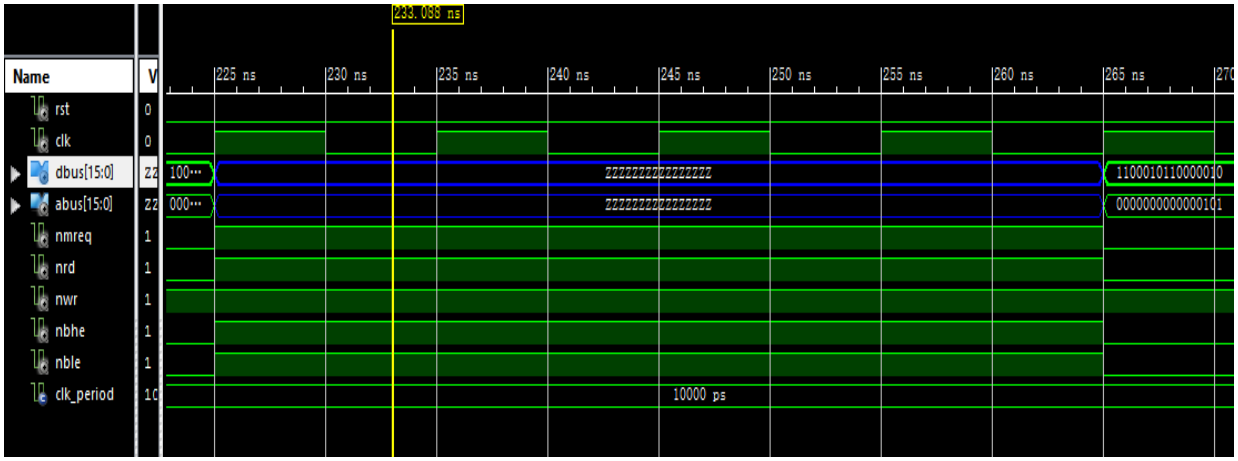


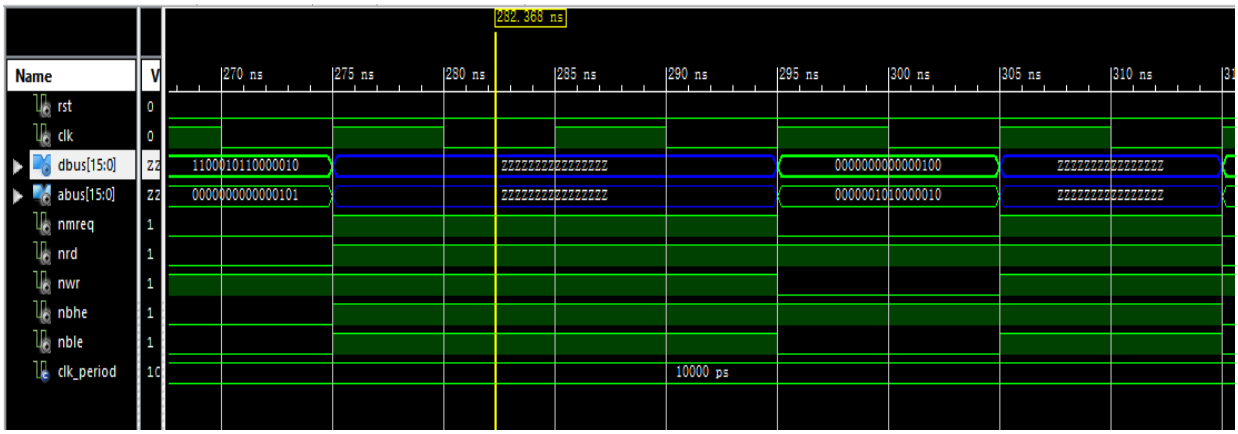

55.038 ns
Name
V
45 ns
50 ns
55 ns
60 ns
65 ns
70 ns
75 ns
80 ns
85 ns
90 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
zzzzzzzzzzzzzzz
1100011110000000
0000000000000001
10000 ps

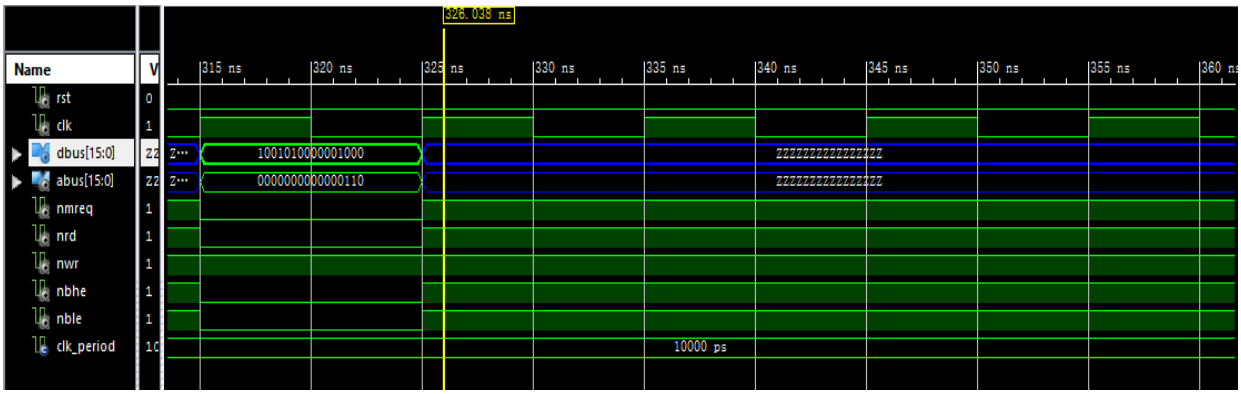

Name
V
90 ns
95 ns
100 ns
105 ns
110 ns
115 ns
120 ns
125 ns
130 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
zzzzzzzzzzzzzzz
0000000000000010
1001011000000010
0000000010000000
10000 ps

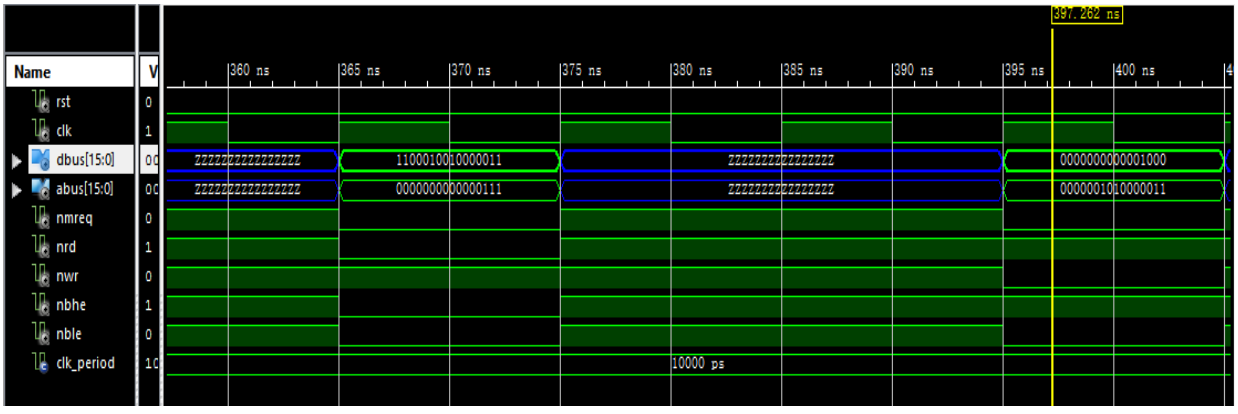

143.508 ns
Name
V
135 ns
140 ns
145 ns
150 ns
155 ns
160 ns
165 ns
170 ns
175 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
zzzzzzzzzzzzzzz
1100011010000001
0000000000000011
10000 ps

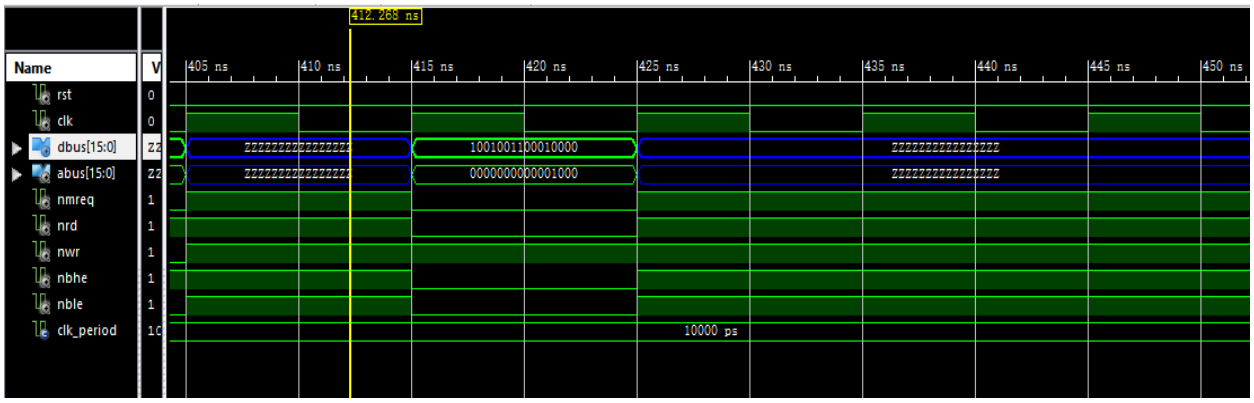

180.438 ns
Name
V
175 ns
180 ns
185 ns
190 ns
195 ns
200 ns
205 ns
210 ns
215 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
11000···
00000···
zzzzzzzzzzzzzzz
0000000000000010
0000001010000001
1001010100000100
0000000000000100
10000 ps

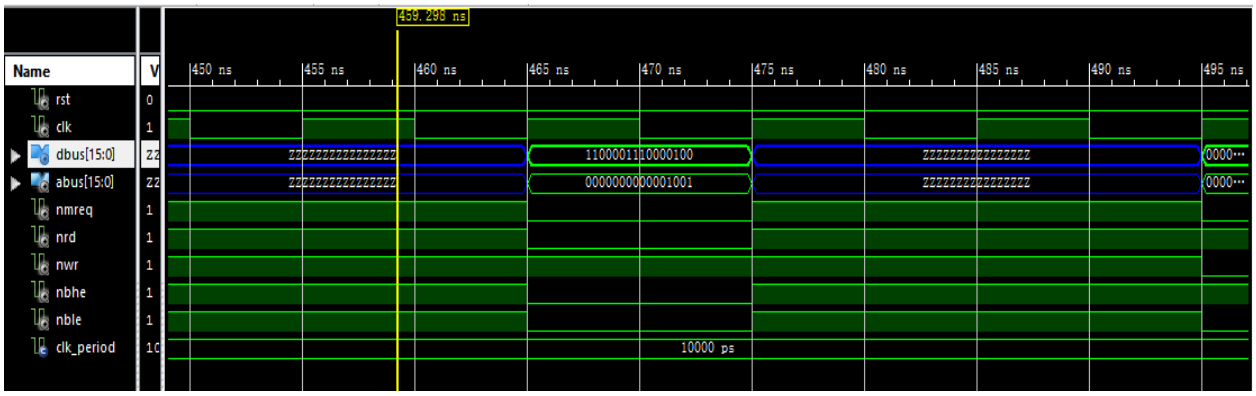

233.086 ns
Name
V
225 ns
230 ns
235 ns
240 ns
245 ns
250 ns
255 ns
260 ns
265 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
10000 ps

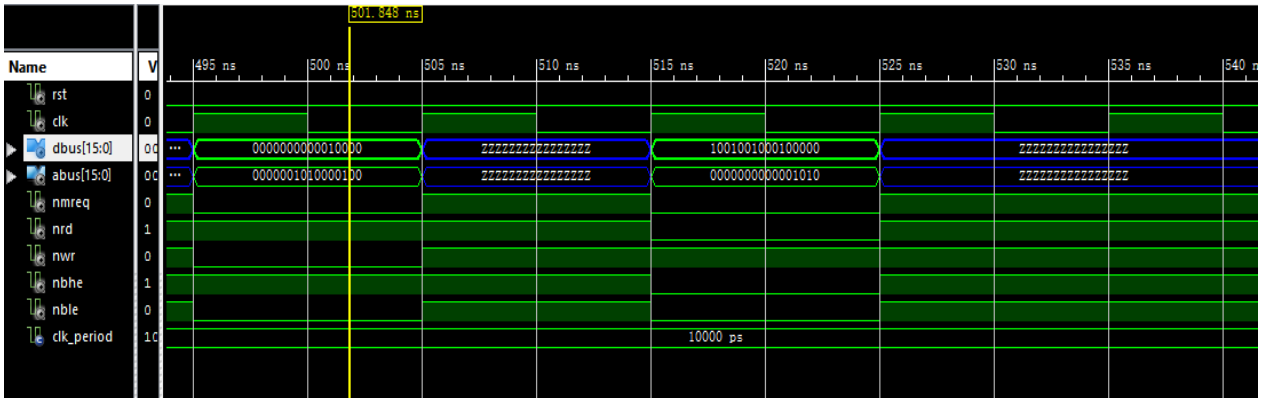

282.368 ns
Name
V
270 ns
275 ns
280 ns
285 ns
290 ns
295 ns
300 ns
305 ns
310 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
10000 ps

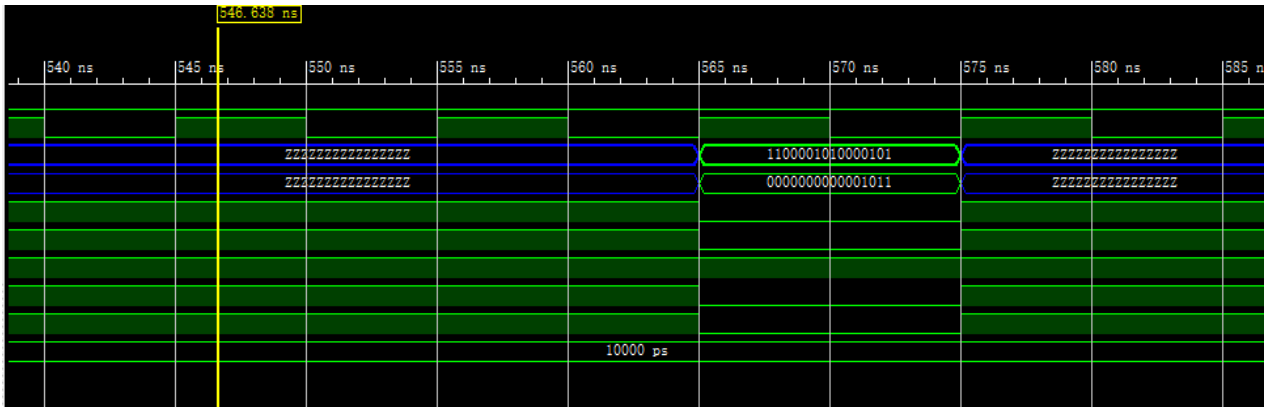

326.038 ns
Name
V
315 ns
320 ns
325 ns
330 ns
335 ns
340 ns
345 ns
350 ns
355 ns
360 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
10000 ps

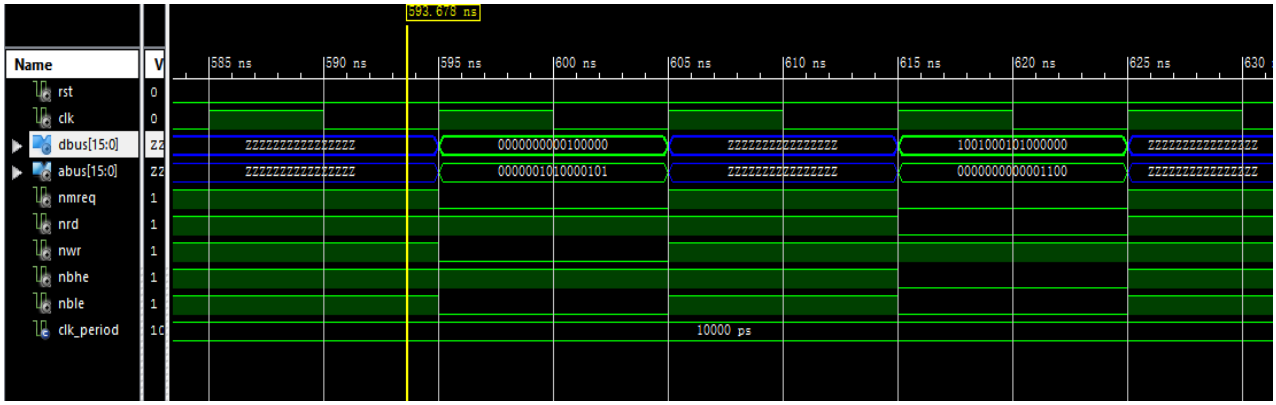

397.262 ns
Name
V
360 ns
365 ns
370 ns
375 ns
380 ns
385 ns
390 ns
395 ns
400 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
10000 ps

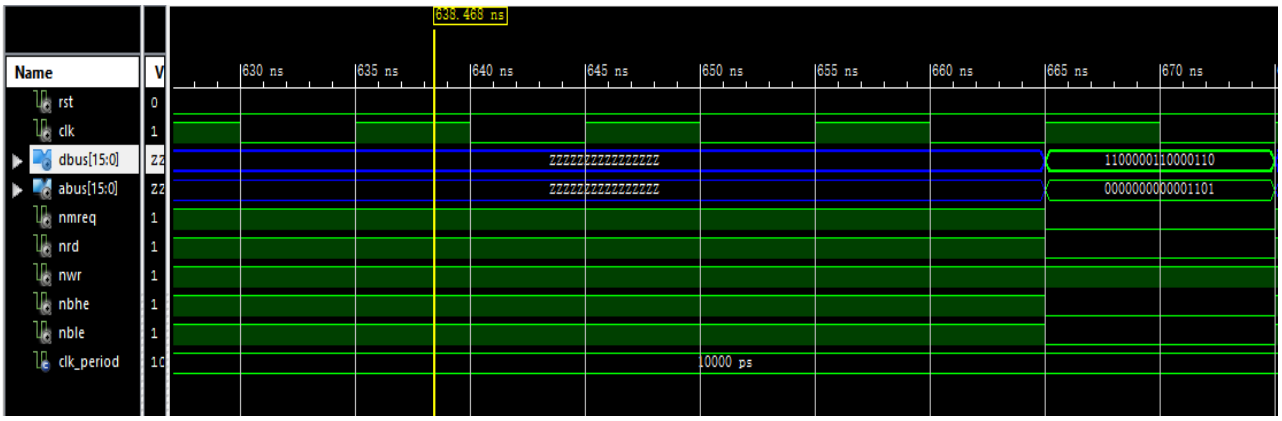

412.268 ns
Name
V
405 ns
410 ns
415 ns
420 ns
425 ns
430 ns
435 ns
440 ns
445 ns
450 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
1001001100010000
0000000000001000
10000 ps

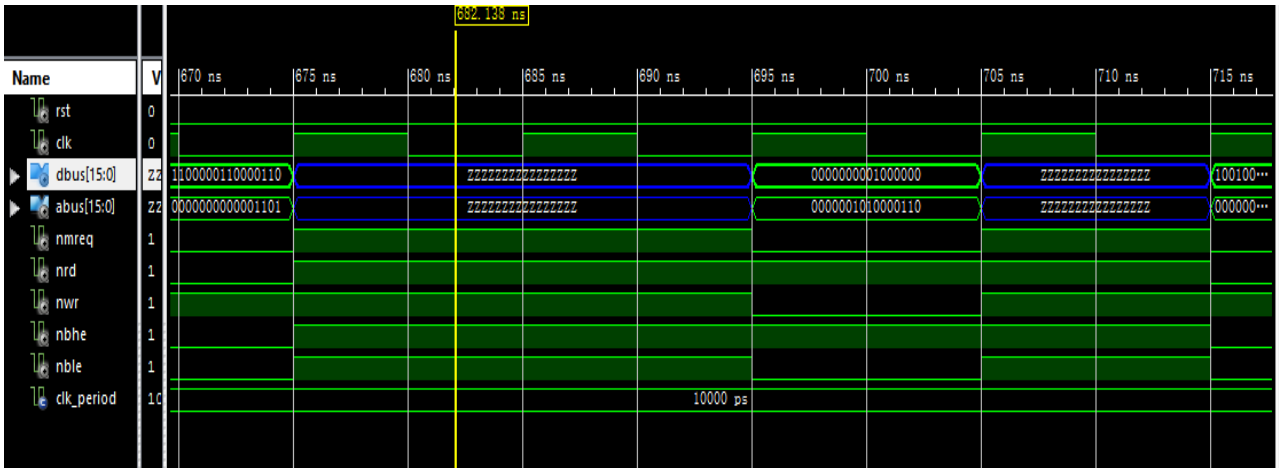

459.298 ns
Name
V
450 ns
455 ns
460 ns
465 ns
470 ns
475 ns
480 ns
485 ns
490 ns
495 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
11000011 0000100
0000000000001001
10000 ps

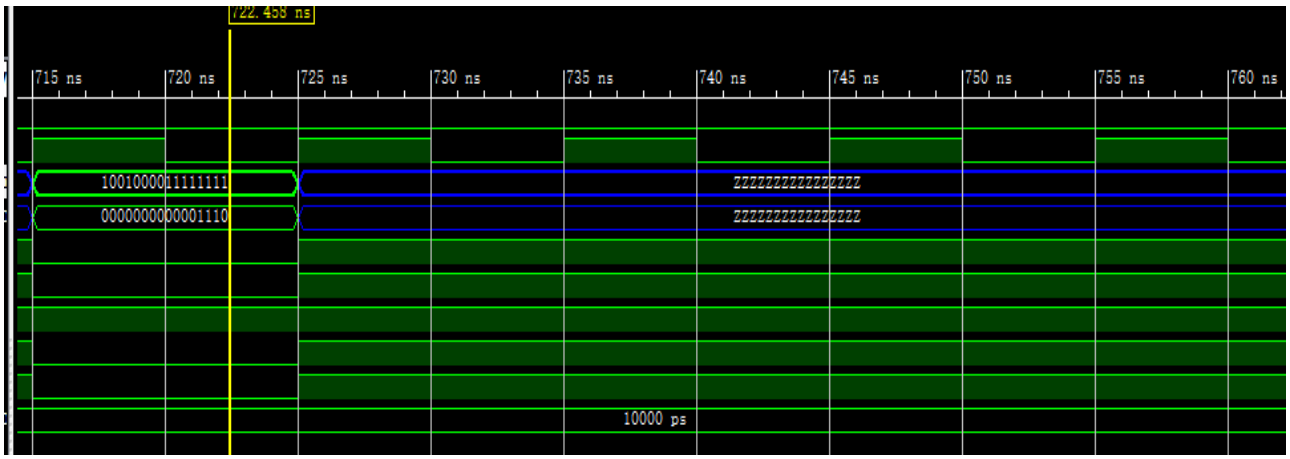

501.848 ns
Name
V
495 ns
500 ns
505 ns
510 ns
515 ns
520 ns
525 ns
530 ns
535 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
0000000000010000
0000001010000100
1001001000100000
0000000000001010
10000 ps

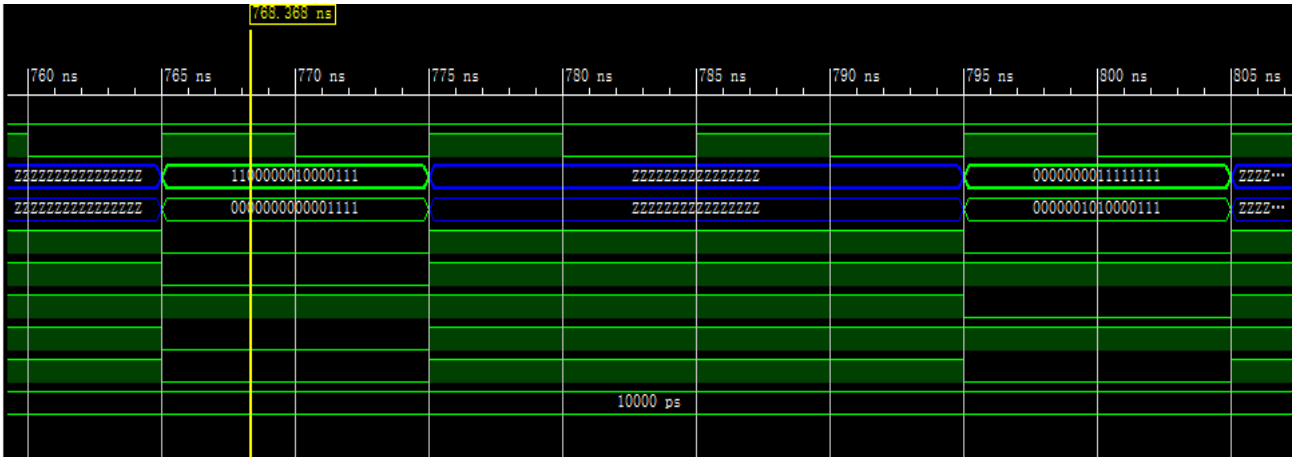

546.638 ns
540 ns
545 ns
550 ns
555 ns
560 ns
565 ns
570 ns
575 ns
580 ns
585 n
1100001010000101
0000000000001011
10000 ps

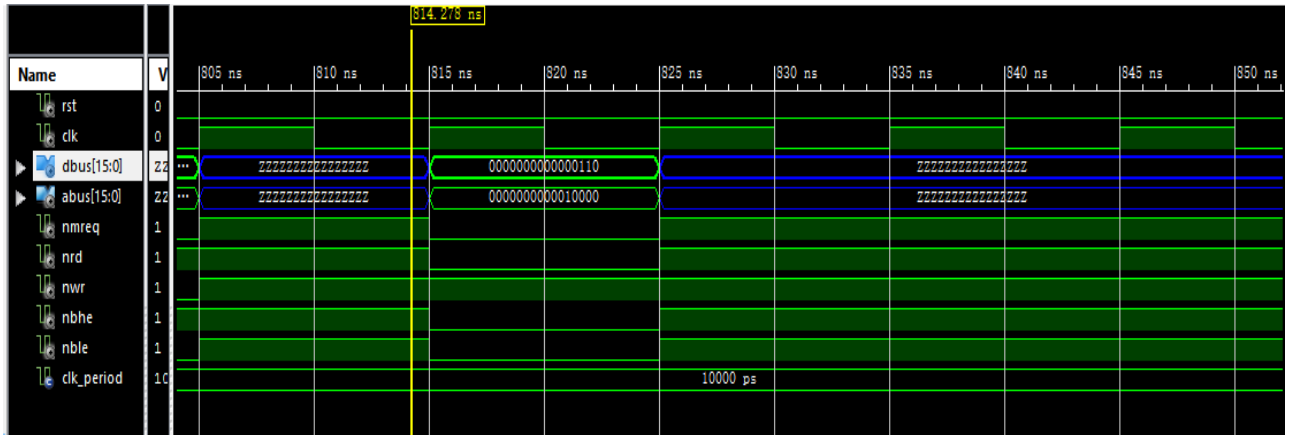

593.678 ns
Name
V
585 ns
590 ns
595 ns
600 ns
605 ns
610 ns
615 ns
620 ns
625 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
10000 ps

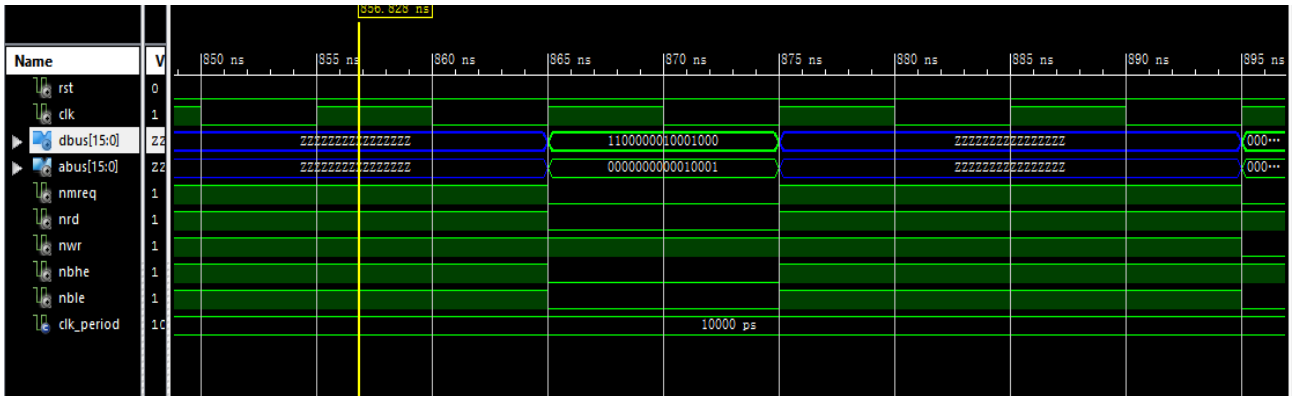

638.468 ns
Name
V
630 ns
635 ns
640 ns
645 ns
650 ns
655 ns
660 ns
665 ns
670 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
10000 ps

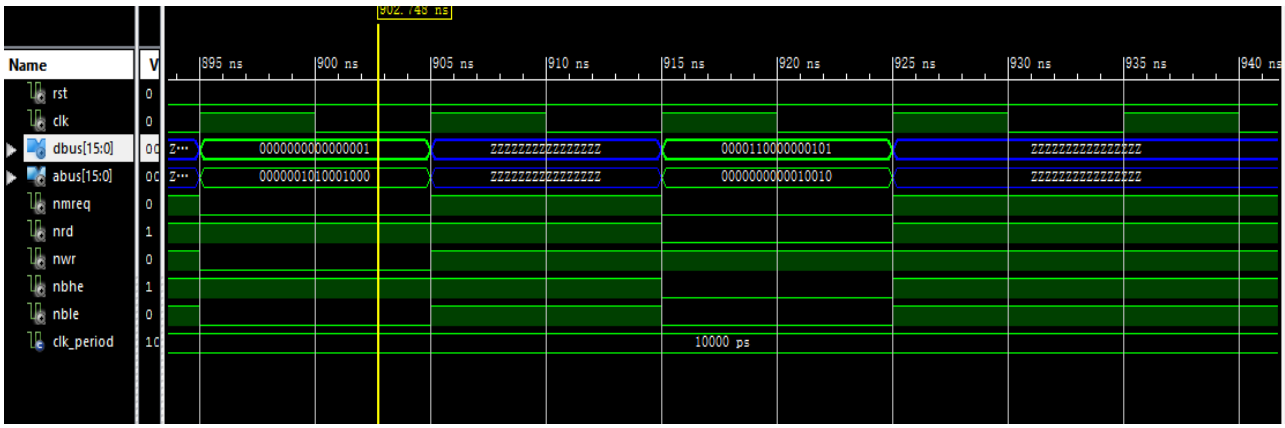

682.138 ns
Name
V
670 ns
675 ns
680 ns
685 ns
690 ns
695 ns
700 ns
705 ns
710 ns
715 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
10000 ps

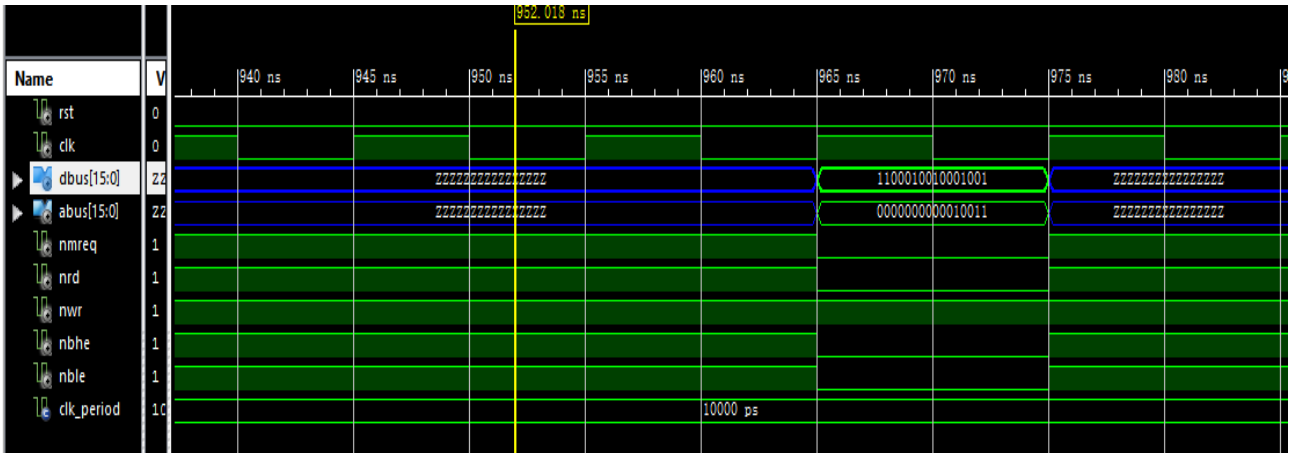

722.458 ns
715 ns
720 ns
725 ns
730 ns
735 ns
740 ns
745 ns
750 ns
755 ns
760 ns
1001000011111111
0000000000001110
zzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzz
10000 ps

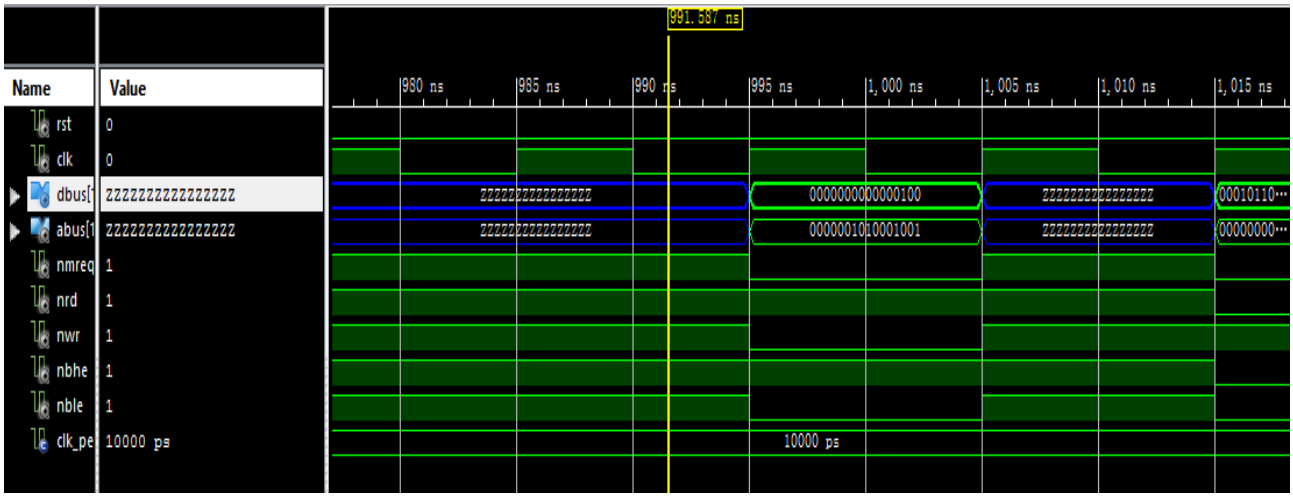

768.368 ns
760 ns
765 ns
770 ns
775 ns
780 ns
785 ns
790 ns
795 ns
800 ns
805 ns
zzzzzzzzzzzzzzzz
1100000010000111
zzzzzzzzzzzzzzzz
0000000011111111
zzzz
zzzzzzzzzzzzzzzz
0000000000001111
zzzzzzzzzzzzzzzz
0000001010000111
zzzz
10000 ps

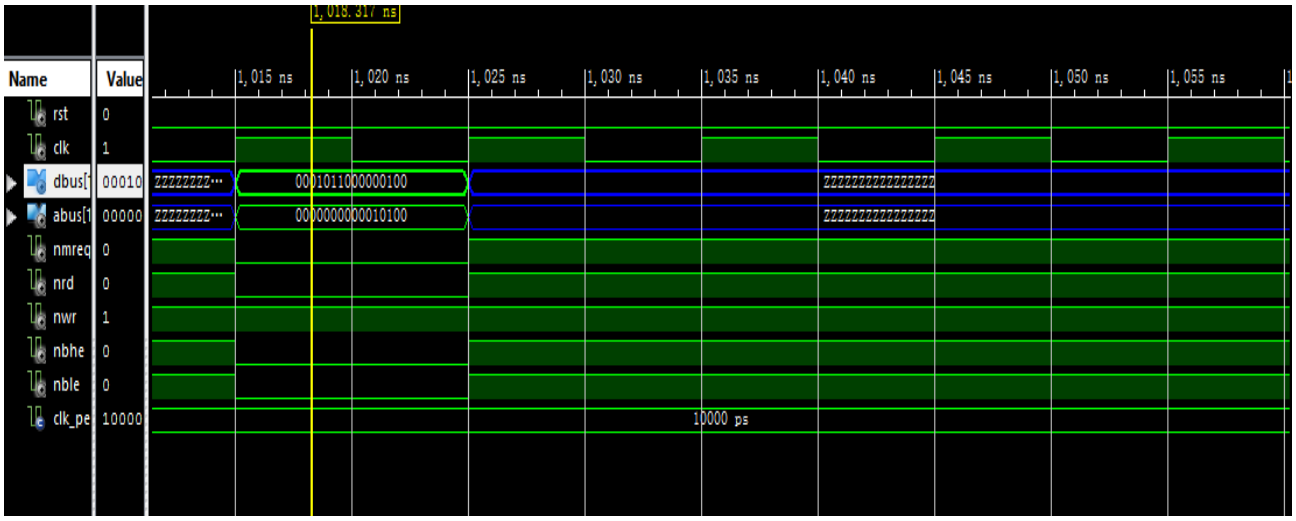

814.278 ns
Name
V
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
805 ns
810 ns
815 ns
820 ns
825 ns
830 ns
835 ns
840 ns
845 ns
850 ns
zzzzzzzzzzzzzzzz
0000000000000110
zzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzz
0000000000010000
zzzzzzzzzzzzzzzz
10000 ps

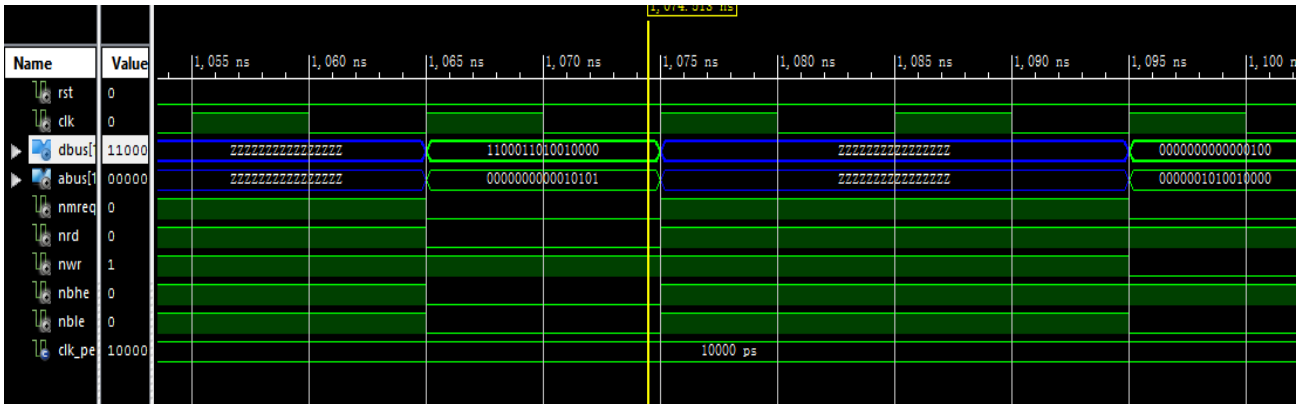

856.828 ns
Name
V
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
850 ns
855 ns
860 ns
865 ns
870 ns
875 ns
880 ns
885 ns
890 ns
895 ns
zzzzzzzzzzzzzzzz
1100000010001000
zzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzz
0000000000010001
zzzzzzzzzzzzzzzz
10000 ps

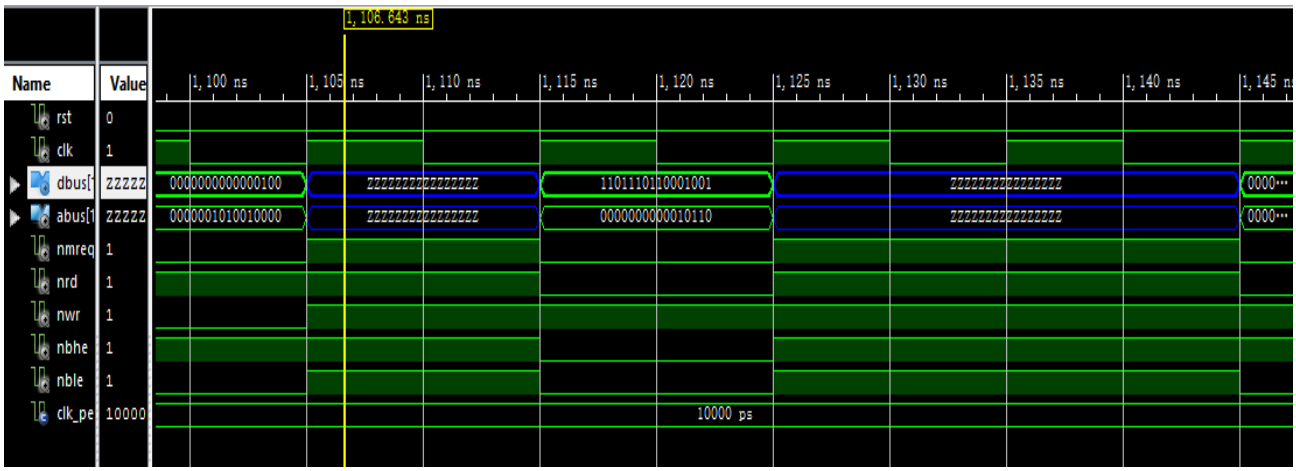

902.748 ns
Name
V
895 ns
900 ns
905 ns
910 ns
915 ns
920 ns
925 ns
930 ns
935 ns
940 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
0000000000000001
zzzzzzzzzzzzzzzz
0000110000000101
zzzzzzzzzzzzzzzz
0000001010001000
zzzzzzzzzzzzzzzz
0000000000010010
zzzzzzzzzzzzzzzz
10000 ps


952.018 ns
Name
V
940 ns
945 ns
950 ns
955 ns
960 ns
965 ns
970 ns
975 ns
980 ns
rst
clk
dbus[15:0]
abus[15:0]
nmreq
nrd
nwr
nbhe
nble
clk_period
zzzzzzzzzzzzzzzz
1100010010001001
zzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzz
0000000000010011
zzzzzzzzzzzzzzzz
10000 ps

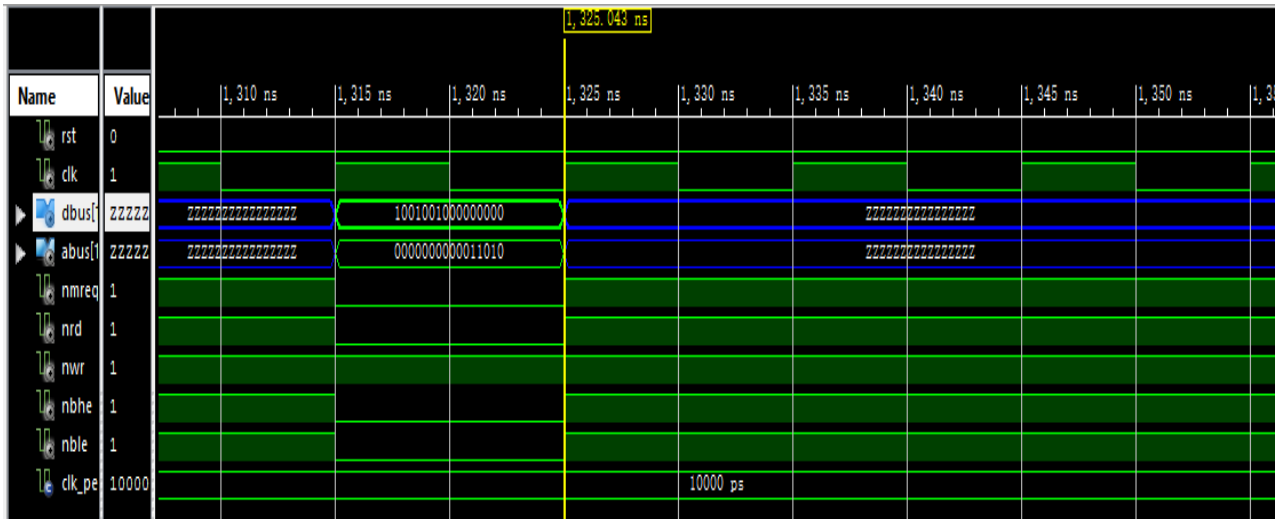

991.587 ns
Name
Value
980 ns
985 ns
990 ns
995 ns
1,000 ns
1,005 ns
1,010 ns
1,015 ns
rst
clk
dbus[1
abus[1
nmreq
nrd
nwr
nbhe
nble
clk_pe
zzzzzzzzzzzzzzzz
0000000000000100
zzzzzzzzzzzzzzzz
00010110···
zzzzzzzzzzzzzzzz
0000001010001001
zzzzzzzzzzzzzzzz
00000000···
10000 ps

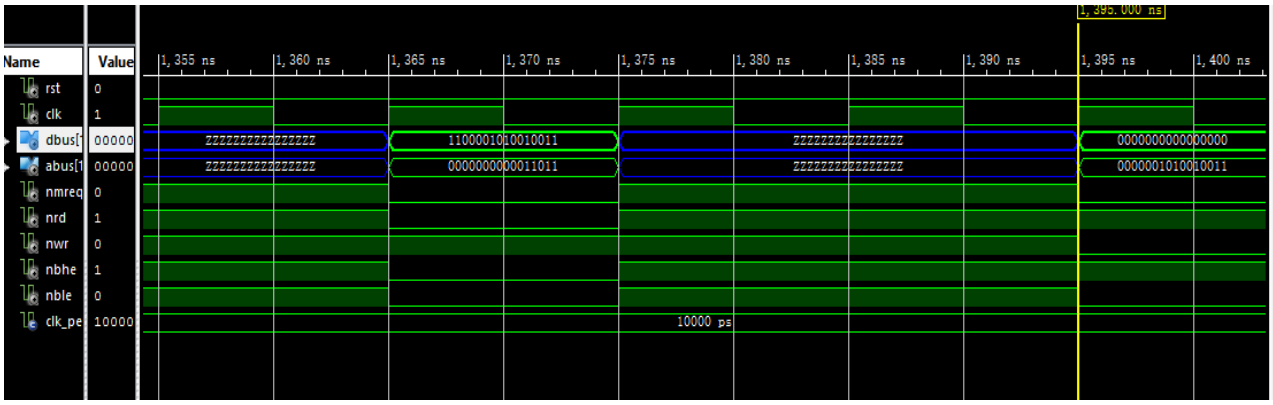

Name
Value
rst
clk
dbus[1
abus[1
nmreq
nrd
nwr
nbhe
nble
clk_pe
00010
00000
10000
10000 ps

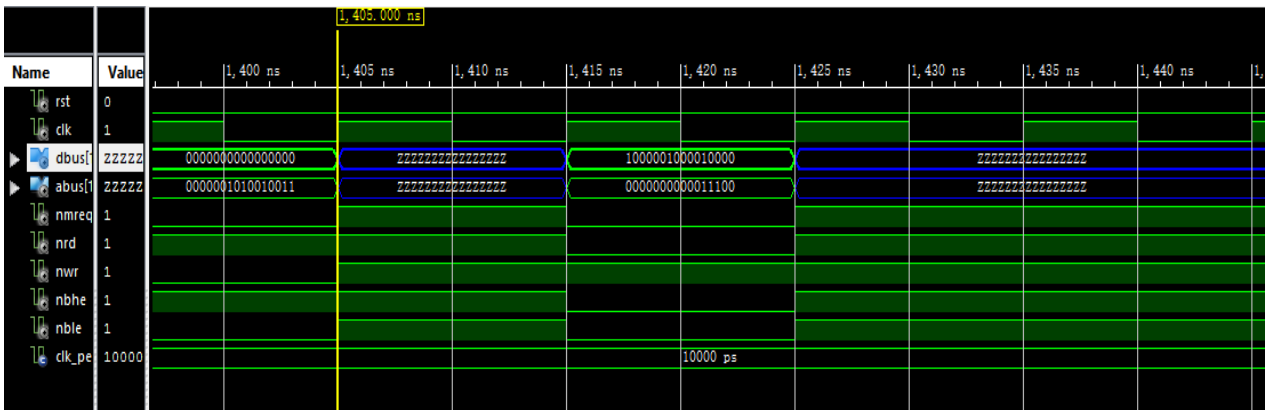

Name
Value
rst
clk
dbus[1
abus[1
nmreq
nrd
nwr
nbhe
nble
clk_pe
11000
00000
10000
10000 ps

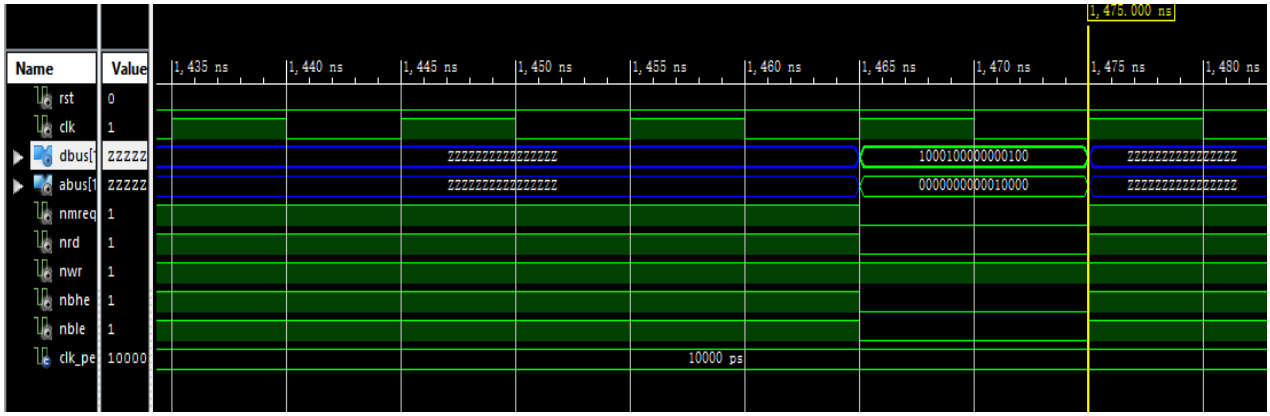

1, 106. 643 ns
Name
Value
rst
clk
dbus[1
abus[1
nmreq
nrd
nwr
nbhe
nble
clk_pe
zzzzz
10000
10000 ps

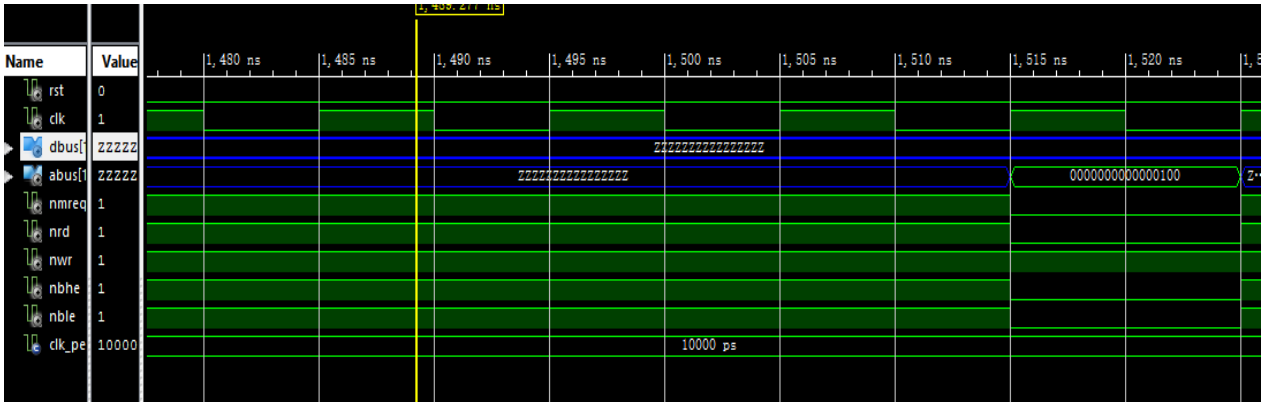

Name
Value
rst
clk
dbus[1
abus[1
nmreq
nrd
nwr
nbhe
nble
clk_pe
00000
00000
0
0
1
1
0
10000
1, 145 ns
1, 150 ns
1, 155 ns
1, 160 ns
1, 165 ns
1, 170 ns
1, 175 ns
1, 180 ns
1, 185 ns
1, 190 ns
0000000011111111
zzzzzzzzzzzzzzzz
1100010110010001
zzzzzzzzzzzzzzzz
0000001010001001
zzzzzzzzzzzzzzzz
0000000000010111
zzzzzzzzzzzzzzzz
10000 ps
1, 190 ns
1, 195 ns
1, 200 ns
1, 205 ns
1, 210 ns
1, 215 ns
1, 220 ns
1, 225 ns
1, 230 ns
zzzzz
1
1
1
1
1
1
1
10000
zzzzzzzzzzzzzzzz
0000000011111111
zzzzzzzzzzzzzzzz
1100011110010010
zzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzz
0000001010010001
zzzzzzzzzzzzzzzz
0000000000011000
zzzzzzzzzzzzzzzz
10000 ps
1, 244.763 ns
1, 230 ns
1, 235 ns
1, 240 ns
1, 245 ns
1, 250 ns
1, 255 ns
1, 260 ns
1, 265 ns
1, 270 ns
zzzzz
zzzzz
1
1
1
1
1
1
10000
zzzzzzzzzzzzzzz
0000000000000010
zzzzzzzzzzzzzzz
1000001000010000
zzzzzzzzzzzzzzz
0000001010010010
zzzzzzzzzzzzzzz
0000000000011001
10000 ps
1, 280.073 ns
1, 270 ns
1, 275 ns
1, 280 ns
1, 285 ns
1, 290 ns
1, 295 ns
1, 300 ns
1, 305 ns
1, 310 ns
zzzzz
zzzzz
1
1
1
1
1
1
10000
1000001000010000
zzzzzzzzzzzzzzz
0000000000011001
zzzzzzzzzzzzzzz
10000 ps


1, 325.043 ns
Name
Value
1, 310 ns
1, 315 ns
1, 320 ns
1, 325 ns
1, 330 ns
1, 335 ns
1, 340 ns
1, 345 ns
1, 350 ns
rst
0
clk
1
dbus[
zzzzz
zzzzzzzzzzzzzzzz
1001001000000000
zzzzzzzzzzzzzzzz
abus[1
zzzzz
zzzzzzzzzzzzzzzz
0000000000011010
zzzzzzzzzzzzzzzz
nmreq
1
nrd
1
nwr
1
nbhe
1
nble
1
clk_pe
10000
10000 ps


1, 395.000 ns
Name
Value
1, 355 ns
1, 360 ns
1, 365 ns
1, 370 ns
1, 375 ns
1, 380 ns
1, 385 ns
1, 390 ns
1, 395 ns
1, 400 ns
rst
0
clk
1
dbus[
00000
zzzzzzzzzzzzzzzz
1100001010010011
zzzzzzzzzzzzzzzz
0000000000000000
abus[1
00000
zzzzzzzzzzzzzzzz
0000000000011011
zzzzzzzzzzzzzzzz
0000001010010011
nmreq
0
nrd
1
nwr
0
nbhe
1
nble
0
clk_pe
10000
10000 ps


1, 405.000 ns
Name
Value
1, 400 ns
1, 405 ns
1, 410 ns
1, 415 ns
1, 420 ns
1, 425 ns
1, 430 ns
1, 435 ns
1, 440 ns
rst
0
clk
1
dbus[
zzzzz
0000000000000000
zzzzzzzzzzzzzzzz
1000001000010000
zzzzzzzzzzzzzzzz
abus[1
zzzzz
0000001010010011
zzzzzzzzzzzzzzzz
0000000000011100
zzzzzzzzzzzzzzzz
nmreq
1
nrd
1
nwr
1
nbhe
1
nble
1
clk_pe
10000
10000 ps


1, 475.000 ns
Name
Value
1, 435 ns
1, 440 ns
1, 445 ns
1, 450 ns
1, 455 ns
1, 460 ns
1, 465 ns
1, 470 ns
1, 475 ns
1, 480 ns
rst
0
clk
1
dbus[
zzzzz
zzzzzzzzzzzzzzzz
1000100000000100
zzzzzzzzzzzzzzzz
abus[1
zzzzz
zzzzzzzzzzzzzzzz
0000000000010000
zzzzzzzzzzzzzzzz
nmreq
1
nrd
1
nwr
1
nbhe
1
nble
1
clk_pe
10000
10000 ps



实验感想

1. **设计过程**：在刚开始的设计中，出现了对整体的把握不是特别严谨的问题，尤其是在访存这块出现了比较严重的问题，没有比较好的理清几个访存操作，致使在写 VHDL 的过程中一直不能比较良好的解决问题。
2. **调试过程**：数据总线采用的是 inout 类型，但是在整个过程中对 inout 的理解不是很好，在老师的辅导和同学的帮助中才慢慢的开始理解他，最后解决了问题。
3. **下载过程**：本以为总的仿真波形没有问题之后可以很快的下载到开发板上，结果就在生成 bit 文件的时候遇到了闭门羹——直接无法生成 bit 文件，遇到了 93 错误，还好在老师的提醒下修改了取指阶段的代码，让每一个 process 负责一个信号，即把 pc 和 ir 分来，放在两个 Process 中。接下来的过程还是遇到了很多奇怪的问题，出现了往主存中写数写不出的错误，最后在老师的帮助下对一些信号接上了等，逐一去查找错误，最后发现问题出现在回写阶段，才比较好的解决了问题。

总的来说：虽然实验台比较老旧，甚至有的设备直接无法工作。但这次实验室特别有用的，自己开始按照自己的一些思路去设计 CPU，比较好的复习了计算机组成原理中的一些东西，也对 CPU 有了更深层次的认识。

附测试代码：

Clock_tb:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY clock_tb IS
END clock_tb;

ARCHITECTURE behavior OF clock_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT clock
    PORT(
         clk : IN  std_logic;
         reset : IN  std_logic;
         t : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal clk : std_logic := '0';
   signal reset : std_logic := '0';

  --Outputs
   signal t : std_logic_vector(3 downto 0);

   -- Clock period definitions
   constant clk_period : time := 100 ns;

BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: clock PORT MAP (
          clk => clk,
          reset => reset,
          t => t
```

```
        );

    -- Clock process definitions
    clk_process :process
    begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;


    -- Stimulus process
    stim_proc: process
    begin
        -- hold reset state for 100 ns.
        reset <= '1';
        wait for 100 ns;
        reset <= '0';
        wait for clk_period*10;
        wait for 100ns;
        reset <= '1';
        -- insert stimulus here

        wait;
    end process;

END;
```

Fetch_tb:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY fetch_tb IS
END fetch_tb;

ARCHITECTURE behavior OF fetch_tb IS

    -- Component Declaration for the Unit Under Test (UUT)
```

```
    COMPONENT fetch
    PORT(
            irnew : IN    std_logic_vector(15 downto 0);
            pcnew : IN    std_logic_vector(15 downto 0);
            clk : IN    std_logic;
            pcupdate : IN    std_logic;
            reset : IN    std_logic;
            t0 : IN    std_logic;
            t1 : IN    std_logic;
            irout : OUT    std_logic_vector(15 downto 0);
            pcout : OUT    std_logic_vector(15 downto 0);
            irrep : OUT    std_logic
          );
    END COMPONENT;


   --Inputs
    signal irnew : std_logic_vector(15 downto 0) := (others => '0');
    signal pcnew : std_logic_vector(15 downto 0) := (others => '0');
    signal clk : std_logic := '0';
    signal pcupdate : std_logic := '0';
    signal reset : std_logic := '0';
    signal t0 : std_logic := '0';
    signal t1 : std_logic := '0';

  --Outputs
    signal irout : std_logic_vector(15 downto 0);
    signal pcout : std_logic_vector(15 downto 0);
    signal irrep : std_logic;

    -- Clock period definitions
    constant clk_period : time := 100 ns;

BEGIN

  -- Instantiate the Unit Under Test (UUT)
    uut: fetch PORT MAP (
              irnew => irnew,
              pcnew => pcnew,
              clk => clk,
              pcupdate => pcupdate,
              reset => reset,
              t0 => t0,
              t1 => t1,
```

```
            irout => irout,
            pcout => pcout,
            irrep => irrep
        );

-- Clock process definitions
clk_process :process
begin
    clk <= '0';
    wait for clk_period/2;
    clk <= '1';
    wait for clk_period/2;
end process;


-- Stimulus process
stim_proc: process
begin
    -- hold reset state for 100 ns.
    wait for 100ns;
    reset <= '1';
    t0 <= '0';
    t1 <= '0';
    pcupdate <= '0';
    irnew <= "1111000011110000";
    pcnew <= "0000000011111111";

    wait for 100ns;
    reset <= '0';
    t0 <= '1';
    wait for 100ns;
    t0 <= '0';
    --pcupdate <= '1';
    wait for 150ns;
    pcupdate <= '0';
    t1 <= '1';
    wait for 100ns;
    --下面一行是刚才修改的
    t0 <= '1';
    pcupdate <= '1';
    t1 <= '0';
    wait for 100ns;
    reset <= '1';
```

```
        wait for clk_period*10;


        -- insert stimulus here

        wait;
    end process;

END;
```

ALU_tb:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY ALU_tb IS
END ALU_tb;

ARCHITECTURE behavior OF ALU_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT ALU
    PORT(
        enable_t : IN    std_logic;
            ir : IN std_logic_vector(15 downto 0);
        sig_reg7aluout : OUT    std_logic_vector(15 downto 0);
        sig_reg7addrout : OUT    std_logic_vector(15 downto 0);
        enable_wb : IN    std_logic;
        reg_wb : IN    std_logic_vector(7 downto 0);
        cy : OUT    std_logic
        );
    END COMPONENT;


    --Inputs
    signal enable_t : std_logic := '0';
     signal ir : std_logic_vector(15 downto 0) := (others => '0');
    signal enable_wb : std_logic := '0';
    signal reg_wb : std_logic_vector(7 downto 0) := (others => '0');
```

```
  --Outputs
signal sig_reg7aluout : std_logic_vector(15 downto 0);
signal sig_reg7addrout : std_logic_vector(15 downto 0);
signal cy : std_logic;
-- No clocks detected in port list. Replace <clock> below with
-- appropriate port name


BEGIN

  -- Instantiate the Unit Under Test (UUT)
uut: ALU PORT MAP (
          enable_t => enable_t,
               ir => ir,
          sig_reg7aluout => sig_reg7aluout,
          sig_reg7addrout => sig_reg7addrout,
          enable_wb => enable_wb,
          reg_wb => reg_wb,
          cy => cy
        );

-- Stimulus process
stim_proc: process
begin
        -- 测试回写模
        ir <= "0000011100000000";
        reg_wb <= "00000000";
        enable_wb <= '0';
        wait for 20 ns;
        enable_wb <= '1';
        wait for 20 ns;


        ir <= "0000011000000000";
        reg_wb <= "11111111";
        enable_wb <= '0';
        wait for 20 ns;
        enable_wb <= '1';
        wait for 20 ns;


        ir <= "0000010100000000";
        reg_wb <= "11111111";
        enable_wb <= '0';
```

```
wait for 20 ns;
enable_wb <= '1';
wait for 20 ns;


ir <= "0000010000000000";
reg_wb <= "01000100";
enable_wb <= '0';
wait for 20 ns;
enable_wb <= '1';
wait for 20 ns;


ir <= "0000001100000000";
reg_wb <= "00110011";
enable_wb <= '0';
wait for 20 ns;
enable_wb <= '1';
wait for 20 ns;


ir <= "0000001000000000";
reg_wb <= "00100010";
enable_wb <= '0';
wait for 20 ns;
enable_wb <= '1';
wait for 20 ns;


ir <= "0000000100000000";
reg_wb <= "00010001";
enable_wb <= '0';
wait for 20 ns;
enable_wb <= '1';
wait for 20 ns;


ir <= "0000000000000000";
reg_wb <= "00000000";
enable_wb <= '0';
wait for 20 ns;
enable_wb <= '1';
wait for 20 ns;
enable_wb <= '0';
```

```
-- 测试操作码，同时测试进位标志
--ADD
ir <= "0000011000000110";
enable_t <= '0';
wait for 20 ns ;
enable_t <= '1';
wait for 20 ns ;
enable_t <= '0' ;
wait for 20 ns;

--SUB
ir <= "0000111000000001";
enable_t <= '0';
wait for 20 ns ;
enable_t <= '1';
wait for 20 ns ;
enable_t <= '0' ;
wait for 20 ns;

--MOV
ir <= "0001010100000001";
enable_t <= '0';
wait for 20 ns ;
enable_t <= '1';
wait for 20 ns ;
enable_t <= '0' ;
wait for 20 ns;

--MVI
ir <= "1001011010101010";
enable_t <= '0';
wait for 20 ns ;
enable_t <= '1';
wait for 20 ns ;
enable_t <= '0' ;
wait for 20 ns;

--LDA
ir <= "1101100010101010";
enable_t <= '0';
wait for 20 ns ;
enable_t <= '1';
wait for 20 ns ;
```

```
            enable_t <= '0' ;
            wait for 20 ns;

            --STA
            ir <= "1100011010101010";
            enable_t <= '0';
            wait for 20 ns ;
            enable_t <= '1';
            wait for 20 ns ;
            enable_t <= '0' ;
            wait for 20 ns;

            --JMP
            ir <= "1000111010101010";
            enable_t <= '0';
            wait for 20 ns ;
            enable_t <= '1';
            wait for 20 ns ;
            enable_t <= '0' ;
            wait for 20 ns;

            --JZ
            ir <= "1000011010101010";
            enable_t <= '0';
            wait for 20 ns ;
            enable_t <= '1';
            wait for 20 ns ;
            enable_t <= '0' ;
            wait for 20 ns;

        wait;
    end process;

END;
```

Control_tb:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY control_tb IS
```

```
END control_tb;

ARCHITECTURE behavior OF control_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT control
    PORT(
            IRreq : IN    std_logic;
            IR : OUT    std_logic_vector(15 downto 0);
            PCout : IN    std_logic_vector(15 downto 0);
            ALUOUT : IN    std_logic_vector(7 downto 0);
            Addr : IN    std_logic_vector(15 downto 0);
            ABUS : OUT    std_logic_vector(15 downto 0);
            DBUS : INOUT    std_logic_vector(15 downto 0);
            nWR : OUT    std_logic;
            nRD : OUT    std_logic;
            nMREQ : OUT    std_logic;
            nBHE : OUT    std_logic;
            nBLE : OUT    std_logic;
            nMWR : IN    std_logic;
            nMRD : IN    std_logic;
            data : OUT    std_logic_vector(7 downto 0)
          );
    END COMPONENT;


   --Inputs
   signal IRreq : std_logic := '0';
   signal PCout : std_logic_vector(15 downto 0) := (others => '0');
   signal ALUOUT : std_logic_vector(7 downto 0) := (others => '0');
   signal Addr : std_logic_vector(15 downto 0) := (others => '0');
   signal nMWR : std_logic := '0';
   signal nMRD : std_logic := '0';

    --BiDirs
   signal DBUS : std_logic_vector(15 downto 0);

    --Outputs
   signal IR : std_logic_vector(15 downto 0);
   signal ABUS : std_logic_vector(15 downto 0);
   signal nWR : std_logic;
   signal nRD : std_logic;
   signal nMREQ : std_logic;
```

```
    signal nBHE : std_logic;
    signal nBLE : std_logic;
    signal data : std_logic_vector(7 downto 0);
    -- No clocks detected in port list. Replace <clock> below with
    -- appropriate port name


BEGIN

     -- Instantiate the Unit Under Test (UUT)
    uut: control PORT MAP (
              IRreq => IRreq,
              IR => IR,
              PCout => PCout,
              ALUOUT => ALUOUT,
              Addr => Addr,
              ABUS => ABUS,
              DBUS => DBUS,
              nWR => nWR,
              nRD => nRD,
              nMREQ => nMREQ,
              nBHE => nBHE,
              nBLE => nBLE,
              nMWR => nMWR,
              nMRD => nMRD,
              data => data
            );


    -- Stimulus process
    stim_proc: process
    begin
        -- hold reset state for 100 ns


            --- ??
            wait for 100 ns;
            nMRD <= '1';
            nMWR <= '1';
            DBUS <= "1111111111111111";
            IRreq <= '1';
            PCout <= "1111000011110000";
```

```
        wait for 100 ns;
        IRreq <= '0';
        nMRD <= '0';
        DBUS <= "1111111111111101";
        Addr <= "0101010101010101";

        wait for 100 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ"; --gaozu

        wait for 100 ns;
        nMRD <= '1';
        nMWR <= '0';
        ALUOUT <= "00001111";
        Addr <= "1010101010101010";


    wait;
  end process;

END;
```

Save_tb:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY save_tb IS
END save_tb;

ARCHITECTURE behavior OF save_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT save
    PORT(
        t : IN    std_logic;
        ALUOUT : IN    std_logic_vector(7 downto 0);
        data : IN    std_logic_vector(7 downto 0);
        nMWR : OUT    std_logic;
        IR : IN    std_logic_vector(15 downto 0);
        nMRD : OUT    std_logic;
```

```
            Rtemp : OUT    std_logic_vector(7 downto 0)
          );
    END COMPONENT;


   --Inputs
   signal t : std_logic := '0';
   signal ALUOUT : std_logic_vector(7 downto 0) := (others => '0');
   signal data : std_logic_vector(7 downto 0) := (others => '0');
   signal IR : std_logic_vector(15 downto 0) := (others => '0');

  --Outputs
   signal nMWR : std_logic;
   signal nMRD : std_logic;
   signal Rtemp : std_logic_vector(7 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name


BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: save PORT MAP (
            t => t,
            ALUOUT => ALUOUT,
            data => data,
            nMWR => nMWR,
            IR => IR,
            nMRD => nMRD,
            Rtemp => Rtemp
          );



   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
      t <= '0';
      wait for 100 ns;
      t <= '1';
      IR <= "1101100000000000";      --取数
      data <= "01010011";
```

```
        wait for 100 ns;
        IR <= "1100000000000000";      --存数
        ALUOUT <= "11110000";
        -- insert stimulus here

        wait;
    end process;

END;
```

Write_back_tb:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY write_back_tb IS
END write_back_tb;

ARCHITECTURE behavior OF write_back_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT write_back
    PORT(
        PCin : IN    std_logic_vector(15 downto 0);
        t : IN    std_logic;
        Rtemp : IN    std_logic_vector(7 downto 0);
        IR : IN    std_logic_vector(15 downto 0);
        z : IN    std_logic;
        cy : IN    std_logic;
        Rupdate : OUT    std_logic;
        Rdata : OUT    std_logic_vector(7 downto 0);
        PCupdate : OUT    std_logic;
        PCnew : OUT    std_logic_vector(15 downto 0)
      );
    END COMPONENT;


   --Inputs
   signal PCin : std_logic_vector(15 downto 0) := (others => '0');
   signal t : std_logic := '0';
```

```
    signal Rtemp : std_logic_vector(7 downto 0) := (others => '0');
    signal IR : std_logic_vector(15 downto 0) := (others => '0');
    signal z : std_logic := '0';
    signal cy : std_logic := '0';

   --Outputs
    signal Rupdate : std_logic;
    signal Rdata : std_logic_vector(7 downto 0);
    signal PCupdate : std_logic;
    signal PCnew : std_logic_vector(15 downto 0);
    -- No clocks detected in port list. Replace <clock> below with
    -- appropriate port name


BEGIN

   -- Instantiate the Unit Under Test (UUT)
    uut: write_back PORT MAP (
                PCin => PCin,
                t => t,
                Rtemp => Rtemp,
                IR => IR,
                z => z,
                cy => cy,
                Rupdate => Rupdate,
                Rdata => Rdata,
                PCupdate => PCupdate,
                PCnew => PCnew
            );




    -- Stimulus process
    stim_proc: process
    begin
        -- hold reset state for 100 ns.
        wait for 100 ns;
        t <= '0';

        wait for 100 ns;
        t <= '1'; ---JMP
        IR <= "1000100001010101";
```

```
        wait for 100 ns;
        t <= '0';

        wait for 100 ns;
        t <= '1';
        IR <= "1000000000001111";
        z <= '1';

        wait for 100 ns;
        t <= '0';

        wait for 100 ns;
        t <= '1';
        IR <= "1000001000001101";
        z <= '0';

        wait for 100 ns;
        t <= '0';

        wait for 100 ns;
        t <= '1';
        IR <= "1100011111110000";      -- STA

        wait for 100 ns;
        t <= '0';

        wait for 100 ns;
        t <= '1';
        IR <= "0001000100000010";
        Rtemp <= "11111111";
        -- insert stimulus here

        wait;
    end process;

END;
```

CPU_tb:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;
```

```
ENTITY CPU_tb IS
END CPU_tb;

ARCHITECTURE behavior OF CPU_tb IS

	-- Component Declaration for the Unit Under Test (UUT)

	COMPONENT CPU
	PORT(
		RST : IN	std_logic;
		CLK : IN	std_logic;
		ABUS : OUT	std_logic_vector(15 downto 0);
		DBUS : INOUT	std_logic_vector(15 downto 0);
		nMREQ : OUT	std_logic;
		nRD : OUT	std_logic;
		nWR : OUT	std_logic;
		nBHE : OUT	std_logic;
		nBLE : OUT	std_logic
	      );
	END COMPONENT;


    --Inputs
    signal RST : std_logic := '0';
    signal CLK : std_logic := '0';

     --BiDirs
    signal DBUS : std_logic_vector(15 downto 0);

     --Outputs
    signal ABUS : std_logic_vector(15 downto 0);
    signal nMREQ : std_logic;
    signal nRD : std_logic;
    signal nWR : std_logic;
    signal nBHE : std_logic;
    signal nBLE : std_logic;

    -- Clock period definitions
    constant CLK_period : time := 10 ns;

BEGIN

	-- Instantiate the Unit Under Test (UUT)
```

```
    uut: CPU PORT MAP (
            RST => RST,
            CLK => CLK,
            ABUS => ABUS,
            DBUS => DBUS,
            nMREQ => nMREQ,
            nRD => nRD,
            nWR => nWR,
            nBHE => nBHE,
            nBLE => nBLE
          );

    -- Clock process definitions
    CLK_process :process
    begin
          CLK <= '0';
          wait for CLK_period/2;
          CLK <= '1';
          wait for CLK_period/2;
    end process;


    -- Stimulus process
    stim_proc: process
    begin
        -- hold reset state for 100 ns.
          --复位
          RST <= '1';
          DBUS <= "ZZZZZZZZZZZZZZZZ";

        wait for 10 ns;
          RST <= '0';
          wait for 5 ns;
          DBUS <= "1001011100000010";                    --- MVI R7    00000000
9700
          wait for 10 ns;
          DBUS <= "ZZZZZZZZZZZZZZZZ";
          wait for 40 ns;

          DBUS <= "1100011110000000";                    --- STA R7 80h         c780
          wait for 10 ns;
          DBUS <= "ZZZZZZZZZZZZZZZZ";
          wait for 40 ns;
```

```
        DBUS <= "1001011000000010";                 --- MVI R6    00000010
9600
      wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100011010000001";                  --- STA R6       81h
c681
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1001010100000100";                  --- MVI R5    00000100
9504
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100010110000010";                  --- STA R5    82h
c582
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1001010000001000";                  --- MVI    R4 00001000
9408
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100010010000011";                  --- STA R4 83h
c483
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1001001100010000";                  --- MVI R3    00010000
9310
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100001110000100";                  --- STA R3 84h
c384
```

```
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1001001000100000";                 --- MVI R2    00100000
9220
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100001010000101";                 --- STA R2 85h
c285
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1001000101000000";                 ---MVI R1     01000000
9140
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100000110000110";                 --- STA R1 86h
c186
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1001000011111111";                 --- MVI R0 11111111
90ff
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100000010000111";                 --- STA R0 87h
c087
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "0000000000000110";                 --- ADD R0    R6
11111111 + 10     0006
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
```

```
        wait for 40 ns;

        DBUS <= "1100000010001000";                 --- STA R0 88h
00000000 cy 1      c088
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "0000110000000101";                 --- SUB R4 R5
0c05
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100010010001001";                 --- STA R4 89h
00000100        1000-0100        c489
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <=    "0001011000000100";               --- MOV R6 R4
1604
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100011010010000";                 --- STA R6 90h
0100      c690
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1101110110001001";                 --- LDA R5 10001001
89h      r4        dd89
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 20 ns;
        DBUS <= "0000000011111111";
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 10 ns;

        DBUS <= "1100010110010001";                 --- STA R5 91h
11111111        c591
```

```
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100011110010010";                 --STA    R7 92h
c792
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1000001000010000";                 -- JZ R2 00001000
8210
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1001001000000000";                 -- MVI R2 00000000
9200
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1100001010010011";                 --STA    R2 93h       c293
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1000001000010000";                 -- JZ R2 00010000
8210
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

        DBUS <= "1000100000000100";                 -- JMP    00000100
8804
        wait for 10 ns;
        DBUS <= "ZZZZZZZZZZZZZZZZ";
        wait for 40 ns;

      -- insert stimulus here

      wait;
   end process;
```

```
END;
```