实验报告

班级：智能 1602

姓名：冶占清

学号：201608010629

一、实验名称

相对简单 CPU 电路设计

二、实验目的

利用 VHDL 设计相对简单 CPU 的电路并验证

三、实验要求

采用 VHDL 描述电路及其测试平台

采用时序逻辑设计电路

采用从 1 累加到 n 的程序进行测试

四、实验内容

相对简单 CPU 的规格说明

地址总线 16 位，数据总线 8 位

有一个 8 位累加寄存器 AC，一个 8 位通用寄存器 R，一个 1 位的零标志

有一个 16 位 AR 寄存器，一个 16 位程序计数器 PC，一个 8 位数据寄存器 DR，一 个

8 位指令寄存器 IR，一个 8 位临时寄存器 TR

有 16 条指令，每条指令 1 个或 3 个字节，其中操作码 8 位。3 字节的指令有 16 位的地址

相对简单 CPU 设计方案

1. 指令执行过程分为取指、译码、执行三个阶段
2. 取指包括三个状态，FETCH1，FETCH2，FETCH3
3. 译码体现为从 FETCH3 状态到各指令执行状态序列的第一个状态

4. 执行根据指令的具体操作分为若干状态
5. 执行的最后一个状态转移到 FETCH1 状态
6. 控制器根据每个状态需要完成的操作产生相应的控制信号

A. rscpu.vhdl

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

use work.rsisa.all;

entity rscpu is
    port(
            clk: in std_logic;
            reset: in std_logic;
            addrbus: out std_logic_vector(15 downto 0);
            databus: inout std_logic_vector(7 downto 0);
            en_read: out std_logic;
            en_write: out std_logic
        );
end entity;

architecture rscpu_behav of rscpu is
    --reg
    signal pc: std_logic_vector(15 downto 0);
    signal ac: std_logic_vector(7 downto 0);
    signal r: std_logic_vector(7 downto 0);
    signal ar: std_logic_vector(15 downto 0);
    signal ir: std_logic_vector(7 downto 0);
    signal dr: std_logic_vector(7 downto 0);
    signal tr: std_logic_vector(7 downto 0);
    signal z: std_logic;

    -- control signal
    signal alu_select: std_logic_vector(2 downto 0);
    signal arload, arinc, pcload, pcinc, drload, trload, irload, rload,
acload, zload: std_logic;
    signal pcbus, drhbus, drlbus, trbus, rbus, acbus, membus:
std_logic;
    signal bus_enable: std_logic_vector(6 downto 0);

    --other signal
    signal alu_result: std_logic_vector(7 downto 0);
    signal main_bus: std_logic_vector(15 downto 0);
signal acinc, accl, write_tmp:std_logic;

--defind state
signal state:std_logic_vector(5 downto 0);
signal nextstate:std_logic_vector(5 downto 0);

constant fetch1:std_logic_vector(5 downto 0) := "000000";
constant fetch2:std_logic_vector(5 downto 0) := "000001";
constant fetch3:std_logic_vector(5 downto 0) := "000010";
constant ldac1:std_logic_vector(5 downto 0) := "000100";
constant ldac2:std_logic_vector(5 downto 0) := "000101";
constant ldac3:std_logic_vector(5 downto 0) := "000110";
constant ldac4:std_logic_vector(5 downto 0) := "000111";
constant ldac5:std_logic_vector(5 downto 0) := "001000";
constant stac1:std_logic_vector(5 downto 0) := "001001";
constant stac2:std_logic_vector(5 downto 0) := "001010";
constant stac3:std_logic_vector(5 downto 0) := "001011";
constant stac4:std_logic_vector(5 downto 0) := "001100";
constant stac5:std_logic_vector(5 downto 0) := "001101";
constant mvac1:std_logic_vector(5 downto 0) := "001110";
constant movr1:std_logic_vector(5 downto 0) := "001111";
constant jump1:std_logic_vector(5 downto 0) := "010000";
constant jump2:std_logic_vector(5 downto 0) := "010001";
constant jump3:std_logic_vector(5 downto 0) := "010010";
constant jmpzn1:std_logic_vector(5 downto 0) := "010011";
constant jmpzn2:std_logic_vector(5 downto 0) := "010100";
constant jpnzn1:std_logic_vector(5 downto 0) := "010101";
constant jpnzn2:std_logic_vector(5 downto 0) := "010110";
constant jmpzy1:std_logic_vector(5 downto 0) := "010111";
constant jmpzy2:std_logic_vector(5 downto 0) := "011000";
constant jmpzy3:std_logic_vector(5 downto 0) := "011001";
constant jpnzy1:std_logic_vector(5 downto 0) := "011010";
constant jpnzy2:std_logic_vector(5 downto 0) := "011011";
constant jpnzy3:std_logic_vector(5 downto 0) := "011100";
constant add1:std_logic_vector(5 downto 0) := "011101";
constant sub1:std_logic_vector(5 downto 0) := "011110";
constant inac1:std_logic_vector(5 downto 0) := "011111";
constant clac1:std_logic_vector(5 downto 0) := "100000";
constant and1:std_logic_vector(5 downto 0) := "100001";
constant or1:std_logic_vector(5 downto 0) := "100010";
constant xor1:std_logic_vector(5 downto 0) := "100011";
constant not1:std_logic_vector(5 downto 0) := "100100";
constant nop1:std_logic_vector(5 downto 0) := "100101";
```

```
    begin

        -- address and data bus
        addrbus <= ar;
        databus <= dr when write_tmp='1' else "ZZZZZZZZ";

        main_bus <= pc when pcbus='1' else
                    (dr & tr) when (drhbus='1' and trbus='1') else
                    (X"00" & dr) when drlbus='1' else
                    (X"00" & r) when rbus='1' else
                    (X"00" & ac) when acbus='1' else
                    (X"00" & databus) when membus='1' else
                    "ZZZZZZZZZZZZZZZZ";

        alu_result <= main_bus(7 downto 0) when alu_select="000" else
                      std_logic_vector(unsigned(ac)+unsigned(main_bus(7
downto 0))) when alu_select="001" else
                      std_logic_vector(unsigned(ac)-unsigned(main_bus(7
downto 0))) when alu_select="010" else
                      ac and main_bus(7 downto 0) when alu_select="011"
else
                      ac or main_bus(7 downto 0) when alu_select="100"
else
                      ac xor main_bus(7 downto 0) when alu_select="101"
else
                      not ac when alu_select="110" else
                      main_bus(7 downto 0);

        --update the values of register,bus...
        process_reg:process(clk)
        begin
            if (rising_edge(clk)) then
                --when reset
                if(reset='1') then
                    pc <= X"0000";
                    state <= fetch1;
                else
                    state <= nextstate;
                end if;
                --reg update reffers to control signal
                if(arload = '1') then
                    ar <= main_bus;
                end if;
                if(arinc = '1') then
```

```
                    ar <= std_logic_vector(unsigned(ar)+1);
                end if;
                if(pcload = '1') then
                    pc <= main_bus;
                end if;
                if(pcinc = '1') then
                    pc <= std_logic_vector(unsigned(pc)+1);
                end if;
                if(drload = '1') then
                    dr <= main_bus(7 downto 0);
                end if;
                if(trload = '1') then
                    tr <= dr;
                end if;
                if(irload = '1') then
                    ir <= main_bus(7 downto 0);
                end if;
                if(rload = '1') then
                    r <= main_bus(7 downto 0);
                end if;
                if(acload = '1') then
                    ac <= alu_result;
                end if;
                if(zload = '1') then
                    if((alu_select="001"    or    alu_select="010"    or
alu_select="011"   or   alu_select="100"   or   alu_select="101"   or
alu_select="110") and alu_result=X"00") then
                        z <= '1';
                    else
                        z <= '0';
                    end if;
                end if;
                if(acinc = '1') then
                    ac <= std_logic_vector(unsigned(ac)+1);
                end if;
                if(accl = '1') then
                    ac <= X"00";
                    z <= '1';
                end if;
            end if;
        end process process_reg;

        --generate next state
        for_nextstate: process(state, ir)
```

```
begin
    case state is
        when fetch1 =>
            nextstate <= fetch2;
        when fetch2 =>
            nextstate <= fetch3;
        when fetch3 =>
            case ir is
                when RSNOP =>
                    nextstate <= nop1;
                when RSLDAC =>
                    nextstate <= ldac1;
                when RSSTAC =>
                    nextstate <= stac1;
                when RSMVAC =>
                    nextstate <= mvac1;
                when RSMOVR =>
                    nextstate <= movr1;
                when RSJUMP =>
                    nextstate <= jump1;
                when RSJMPZ =>
                    if(z='1') then
                        nextstate <= jmpzy1;
                    elsif(z='0') then
                        nextstate <= jmpzn1;
                    end if;
                when RSJPNZ =>
                    if(z='1') then
                        nextstate <= jpnzn1;
                    elsif(z='0') then
                        nextstate <= jpnzy1;
                    end if;
                when RSADD =>
                    nextstate <= add1;
                when RSSUB =>
                    nextstate <= sub1;
                when RSINAC =>
                    nextstate <= inac1;
                when RSCLAC =>
                    nextstate <= clac1;
                when RSAND =>
                    nextstate <= and1;
                when RSOR =>
                    nextstate <= or1;
        when RSXOR =>
            nextstate <= xor1;
        when RSNOT =>
            nextstate <= not1;
        when others =>
            nextstate <= fetch1;
    end case;
--ldac
when ldac1 =>
    nextstate <= ldac2;
when ldac2 =>
    nextstate <= ldac3;
when ldac3 =>
    nextstate <= ldac4;
when ldac4 =>
    nextstate <= ldac5;
--stac
when stac1 =>
    nextstate <= stac2;
when stac2 =>
    nextstate <= stac3;
when stac3 =>
    nextstate <= stac4;
when stac4 =>
    nextstate <= stac5;
--jump
when jump1 =>
    nextstate <= jump2;
when jump2 =>
    nextstate <= jump3;
--jmpzn
when jmpzn1 =>
    nextstate <= jmpzn2;
--jmpzy
when jmpzy1 =>
    nextstate <= jmpzy2;
when jmpzy2 =>
    nextstate <= jmpzy3;
--jpnzn
when jpnzn1 =>
    nextstate <= jpnzn2;
--jpnzy
when jpnzy1 =>
    nextstate <= jpnzy2;
```

```
            when jpnzy2 =>
                nextstate <= jpnzy3;
            when others =>
                nextstate <= fetch1;
        end case;
    end process for_nextstate;

    --generate control signal
    control_signal:process(state)
    begin
        case state is
            when fetch1 =>      -- ar<-pc
                arload <= '1'; arinc <= '0';  pcload   <=   '0';
pcinc <= '0';   drload <= '0'; alu_select <= "000";
                trload <= '0'; irload <= '0'; rload    <=   '0';
acload <= '0';  zload <= '0';
                pcbus <= '1';   drhbus <= '0'; drlbus   <=   '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl    <=   '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when fetch2 =>      -- ir<-M    pc++
                arload <= '0'; arinc <= '0';  pcload   <=   '0';
pcinc <= '1';   drload <= '0'; alu_select <= "000";
                trload <= '0'; irload <= '1'; rload    <=   '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0'; drlbus   <=   '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl    <=   '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when fetch3 =>      -- ar<-pc
                arload <= '1'; arinc <= '0';  pcload   <=   '0';
pcinc <= '0';   drload <= '0'; alu_select <= "000";
                trload <= '0'; irload <= '0'; rload    <=   '0';
acload <= '0';  zload <= '0';
                pcbus <= '1';   drhbus <= '0'; drlbus   <=   '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl    <=   '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when nop1 =>        -- do nothing
                arload <= '0'; arinc <= '0';  pcload   <=   '0';
pcinc <= '0';   drload <= '0';
                trload <= '0'; irload <= '0'; rload    <=   '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0'; drlbus   <=   '0';
```

```
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when ldac1 =>       -- dr<-M    pc++    ar++
                arload <= '0';  arinc <= '1';   pcload    <=    '0';
pcinc <= '1';   drload <= '1';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when ldac2 =>       -- tr<-dr   dr<-M   pc++
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '1';   drload <= '1';  alu_select <= "000";
                trload <= '1';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when ldac3 =>       -- ar<-dr&tr
                arload <= '1';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '1';  drlbus    <=    '0';
trbus <= '1';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when ldac4 =>       -- dr<-M
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '1';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';
            when ldac5 =>       -- ac<-dr
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '1';  zload <= '0';
```

```
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '1';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';
            when stac1 =>       -- dr<-M    pc++    ar++
                arload <= '0';  arinc <= '1';   pcload    <=    '0';
pcinc <= '1';   drload <= '1';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when stac2 =>       -- tr<-dr   dr<-M   pc++
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '1';   drload <= '1';  alu_select <= "000";
                trload <= '1';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when stac3 =>       -- ar<-dr&tr
                arload <= '1';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '1';  drlbus    <=    '0';
trbus <= '1';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when stac4 =>       -- dr<-ac
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '1';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '1';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when stac5 =>       -- M<-dr
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
```

```
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '1';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '0'; en_write <= '1';    write_tmp <= '1';
            when mvac1 =>       -- r<-ac
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload     <=    '1';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '1';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when movr1 =>       -- ac<-r
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload     <=    '0';
acload <= '1';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '1';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jump1 =>       -- dr<-M    ar++
                arload <= '0';  arinc <= '1';   pcload    <=    '0';
pcinc <= '0';   drload <= '1';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload     <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jump2 =>       -- tr<-dr   dr<-M
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '1';  alu_select <= "000";
                trload <= '1';  irload <= '0';  rload     <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=     '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jump3 =>       --pc<-dr&tr;
                arload <= '0';  arinc <= '0';   pcload    <=    '1';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
```

```
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '1';  drlbus    <=    '0';
trbus <= '1';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jmpzn1 =>      -- pc++
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '1';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';
            when jmpzn2 =>      -- pc++
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '1';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jpnzn1 =>      -- pc++
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '1';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jpnzn2 =>      -- pc++
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '1';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jmpzy1 =>      -- dr<-M    ar++
                arload <= '0';  arinc <= '1';   pcload    <=    '0';
```

```
pcinc <= '0';   drload <= '1';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jmpzy2 =>      -- tr<-dr   dr<-M
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '1';  alu_select <= "000";
                trload <= '1';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jmpzy3 =>      -- pc<-dr&tr;
                arload <= '0';  arinc <= '0';   pcload    <=    '1';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '1';  drlbus    <=    '0';
trbus <= '1';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jpnzy1 =>      -- dr<-M    ar++
                arload <= '0';  arinc <= '1';   pcload    <=    '0';
pcinc <= '0';   drload <= '1';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jpnzy2 =>      -- tr<-dr   dr<-M
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '1';  alu_select <= "000";
                trload <= '1';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '1';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when jpnzy3 =>      -- pc<-dr&tr;
```

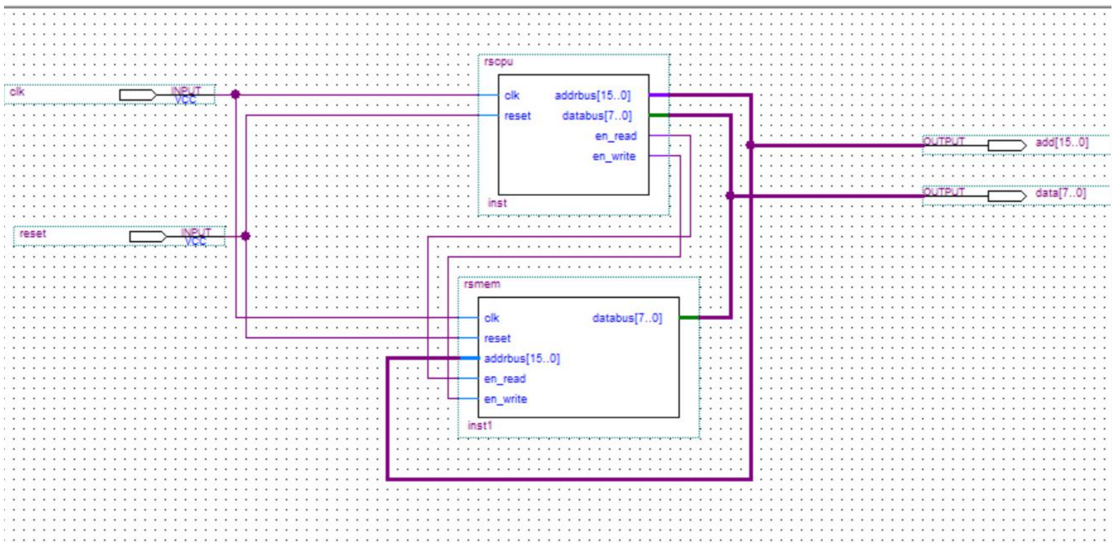
```
                arload <= '0';  arinc <= '0';   pcload    <=    '1';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '1';  drlbus    <=    '0';
trbus <= '1';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when add1 =>        --
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "001";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '1';  zload <= '1';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '1';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when sub1 =>
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "010";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '1';  zload <= '1';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '1';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when inac1 =>
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '1';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '1';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when clac1 =>
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "000";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '1';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '1';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
```

```
            when and1 =>
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "011";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '1';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '1';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when or1 =>
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "100";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '1';  zload <= '1';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '1';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when xor1 =>
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "101";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '1';  zload <= '1';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '1';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when not1 =>
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';  alu_select <= "110";
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '1';  zload <= '1';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
en_read <= '1'; en_write <= '0';    write_tmp <= '0';
            when others =>
                arload <= '0';  arinc <= '0';   pcload    <=    '0';
pcinc <= '0';   drload <= '0';
                trload <= '0';  irload <= '0';  rload    <=    '0';
acload <= '0';  zload <= '0';
                pcbus <= '0';   drhbus <= '0';  drlbus    <=    '0';
trbus <= '0';   rbus <= '0';    membus <= '0';
                acbus <= '0';   acinc <= '0';   accl     <=    '0';
```

```
en_read <= '0'; en_write <= '0';    write_tmp <= '0';
        end case;
    end process control_signal;

end;
```
B. rsisa.vhdl

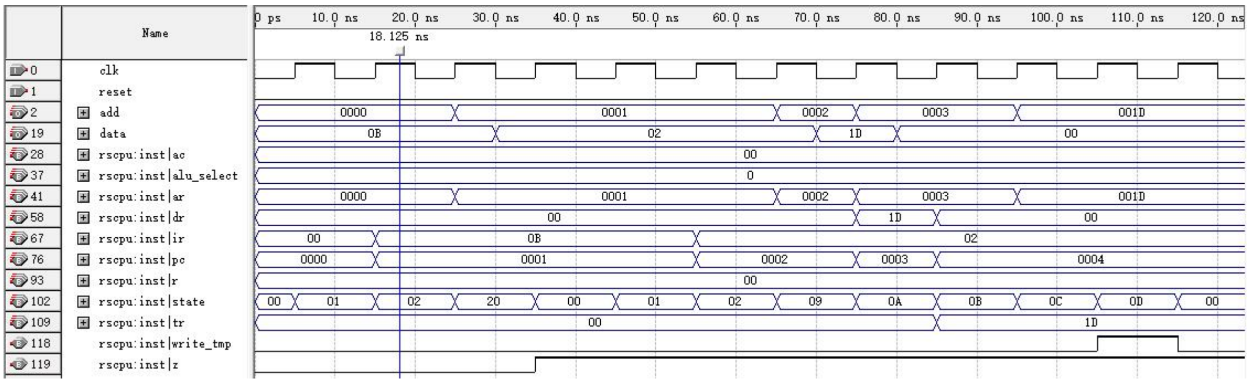
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


package rsisa is

    -- RS prefix is used to avoid tautonym such like AND, OR, XOR, NOT
    constant RSNOP: std_logic_vector(7 downto 0) := "00000000";
    constant RSLDAC: std_logic_vector(7 downto 0) := "00000001";
    constant RSSTAC: std_logic_vector(7 downto 0) := "00000010";
    constant RSMVAC: std_logic_vector(7 downto 0) := "00000011";
    constant RSMOVR: std_logic_vector(7 downto 0) := "00000100";
    constant RSJUMP: std_logic_vector(7 downto 0) := "00000101";
    constant RSJMPZ: std_logic_vector(7 downto 0) := "00000110";
    constant RSJPNZ: std_logic_vector(7 downto 0) := "00000111";

    constant RSADD: std_logic_vector(7 downto 0) := "00001000";
    constant RSSUB: std_logic_vector(7 downto 0) := "00001001";
    constant RSINAC: std_logic_vector(7 downto 0) := "00001010";
    constant RSCLAC: std_logic_vector(7 downto 0) := "00001011";
    constant RSAND: std_logic_vector(7 downto 0) := "00001100";
    constant RSOR: std_logic_vector(7 downto 0) := "00001101";
    constant RSXOR: std_logic_vector(7 downto 0) := "00001110";
    constant RSNOT: std_logic_vector(7 downto 0) := "00001111";

end package;
```
C. rsmem.vhdl

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

use work.rsisa.all;

entity rsmem is
    port(
            clk: in std_logic;
            reset: in std_logic;
            addrbus: in std_logic_vector(15 downto 0);
            databus: inout std_logic_vector(7 downto 0);
            en_read: in std_logic;
            en_write: in std_logic
        );
end entity;

architecture rsmem_behav of rsmem is
    signal addr: std_logic_vector(15 downto 0);

    type  memtype  is  array(natural  range<>)  of  std_logic_vector(7
downto 0);
    constant total_addr : integer := 29;
    constant i_addr : integer := 30;
    constant n_addr : integer := 31;
    constant loop_addr : integer := 7;

    signal memdata: memtype(4095 downto 0) := (
    0 => RSCLAC,
    1 => RSSTAC,
    2 => std_logic_vector(to_unsigned(total_addr, 8)),
    3 => X"00",
    4 => RSSTAC,
    5 => std_logic_vector(to_unsigned(i_addr, 8)),
    6 => X"00",
    7 => RSLDAC,  -- loop
    8 => std_logic_vector(to_unsigned(i_addr, 8)),
    9 => X"00",
    10 => RSINAC,
    11 => RSSTAC,
    12 => std_logic_vector(to_unsigned(i_addr, 8)),
    13 => X"00",
    14 => RSMVAC,
    15 => RSLDAC,
    16 => std_logic_vector(to_unsigned(total_addr, 8)),
    17 => X"00",
    18 => RSADD,
    19 => RSSTAC,
    20 => std_logic_vector(to_unsigned(total_addr, 8)),
    21 => X"00",
    22 => RSLDAC,
    23 => std_logic_vector(to_unsigned(n_addr, 8)),
    24 => X"00",
    25 => RSSUB,
        26 => RSJPNZ,
        27 => std_logic_vector(to_unsigned(loop_addr, 8)),
        28 => X"00",
        29 => X"00",  -- total
        30 => X"00",  -- i
        31 => X"04",  -- n
        others => RSNOP
    );

    begin
        -- The process takes addrbus and read/write signals at first,
        -- then at the next clock does the data transmission.
        for_clk : process(clk)
        begin
            if(falling_edge(clk)) then
                if(reset='1') then
                    addr <= (others=>'0');
                else
                    addr <= addrbus;
                end if;

                if(en_write='1') then
                    memdata(to_integer(unsigned(addr))) <= databus;
                end if;
            end if;
        end process;

        databus <= memdata(to_integer(unsigned(addr))) when (en_read='1')
else "ZZZZZZZZ";

    end architecture;
```
D. rcomp.bdf

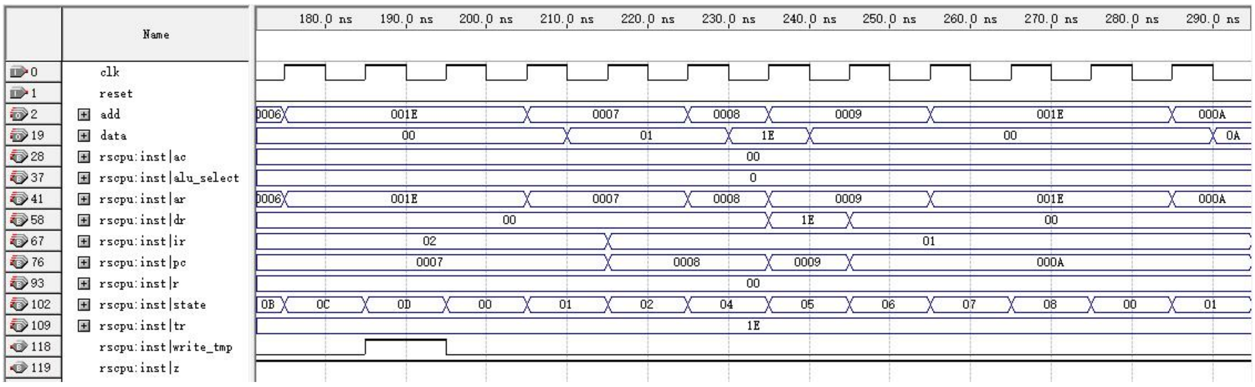


clk
INPUT VCC
reset
INPUT VCC
rscpu
clk
reset
addrbus[15..0]
databus[7..0]
en_read
en_write
inst
rsmem
clk
reset
addrbus[15..0]
en_read
en_write
databus[7..0]
inst1
OUTPUT
add[15..0]
OUTPUT
data[7..0]

五、测试结果



Name
18.125 ns
clk
reset
add
data
rscpu:inst|ac
rscpu:inst|alu_select
rscpu:inst|ar
rscpu:inst|dr
rscpu:inst|ir
rscpu:inst|pc
rscpu:inst|r
rscpu:inst|state
rscpu:inst|tr
rscpu:inst|write_tmp
rscpu:inst|z



Name
clk
reset
add
data
rscpu:inst|ac
rscpu:inst|alu_select
rscpu:inst|ar
rscpu:inst|dr
rscpu:inst|ir
rscpu:inst|pc
rscpu:inst|r
rscpu:inst|state
rscpu:inst|tr
rscpu:inst|write_tmp
rscpu:inst|z