

CPU设计汇报

物联1601 刘小芬 201608010107

规格说明

地址总线16位
数据总线8位

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

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

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

设计方案



指令执行过程分为取指、译码、
执行三个阶段



取指包括三个状态，FETCH1，
FETCH2，FETCH3

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

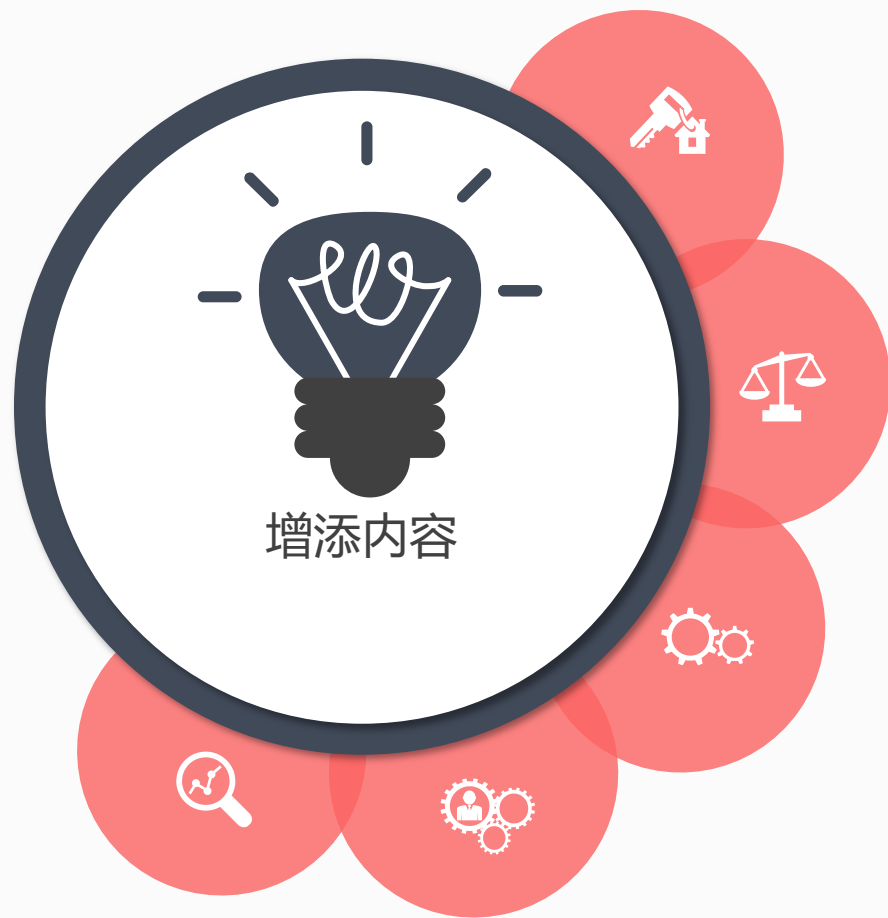


执行根据指令的具体操作分为
若干状态

执行的最后一个状态转移到
FETCH1状态



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



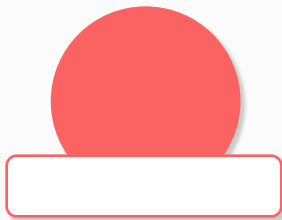
● 增加了FETCH4状态一共四个状态。取址包括四个状态，而译码体现为从FETCH4状态到各指令执行状态序列的第一个状态。执行根据指令的具体操作分为不同的状态，并且其最后一个状态转移到FETCH1状态。

```
when fetch1=>--ar<=pc
```

```
when fetch2=>--dr<=m  pc<=pc+1
```

```
when fetch3=>--ir<=dr
```

```
when fetch4=>--ar<=pc
```



在cpu.vhd文件中，声明了cpu的内部总线、寄存器、控制信号和所有状态所对应的编码，以及对alu相关运算进行设计，并声明了所有会执行的指令。在时钟信号的上升沿，对信号0、1时刻下对应的操作做了规定。在进程for_nextstate中判断所处状态的下一个状态是什么，在另一个进程中gen_controls规定所处状态下所有控制信号的0、1状态。

```

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;
signal thebus: std_logic_vector(15 downto 0);

if(rising_edge(clk)) then
    --update registers
    if(acreset='1') then
        ac<="00000000";
        z<='0';
    end if;
    if(reset='1') then
        pc <= "0000000000000000";
        state <= fetch1;
    else
        state <= nextstate;
    end if;
    if(pcinc='1')then
        pc<=std_logic_vector(unsigned(pc) + 1);
    end if;
    if(acinc='1')then
        ac<=std_logic_vector(unsigned(ac) + 1);
        if(ac="1111111")then
            z<='1';
        else
            z<='0';
        end if;
    end if;
end if;

```

```

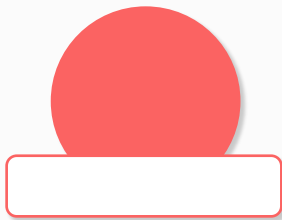
case state is
  when fetch1=>
    nextstate<=fetch2;
  when fetch2=>
    nextstate<=fetch3;
  when fetch3=>
    nextstate<=fetch4;
  when fetch4=>
    case ir is
      when RSCLAC=>
        nextstate<=clac1;
      when RSINAC=>
        nextstate<=incac1;
      when RSADD=>
        nextstate<=add1;
      when RSSUB=>
        nextstate<=sub1;
      when RSAND=>
        nextstate<=and1;
      when RSOR=>
        nextstate<=or1;
      when RSXOR=>
        nextstate<=xor1;

```

```

case state is
  when fetch1=>--ar<=pc
    arload<='1';pcbus<='1';pcinc<='0';
    drload<='0';membus<='0';irload<='0';
    acreset<='0';acinc<='0';rbus<='0';
    s<="000";acload<='0';rload<='0';
    acbus<='0';arinc<='0';trbus<='0';
    drbus<='0';trload<='0';read<='0';
    write1<='0';write<='0';pcload<='0';
  when fetch2=>--dr<=m pc<=pc+1
    arload<='0';pcbus<='0';pcinc<='1';
    drload<='1';membus<='1';irload<='0';
    acreset<='0';acinc<='0';rbus<='0';
    s<="000";acload<='0';rload<='0';
    acbus<='0';arinc<='0';trbus<='0';
    drbus<='0';trload<='0';read<='1';
    write1<='0';write<='0';pcload<='0';

```



在rsisa.vhd文件中，对FETCH4所要进行的下一个状态进行了指令的一个编码，并且给它一个相关的变量名，这是为了在CPU.vhd中对变量名的判断从而执行下一个状态。

```
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";
```

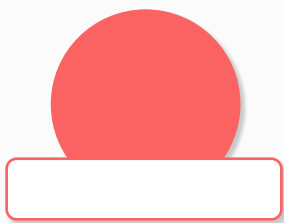


在mem.vhd文件中，对内存大小进行了规定为0到65535，之前的4096太小了会溢出。对内存所要执行的东西进行了设计，可以从1到n进行加法运算，n暂时设为8，result变量对应没进行一次运算的结果，可以很直观的看出结果是否正确。

```
begin
    if(falling_edge(clk)) then
        if(reset='1') then
            addr <= (others=>'0');
        else
            addr <= addrbus;
        end if;

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

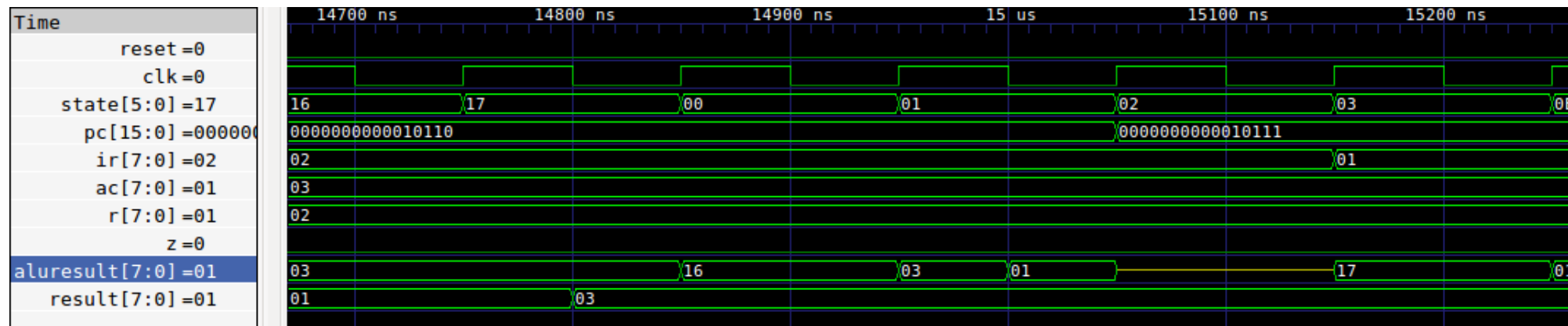
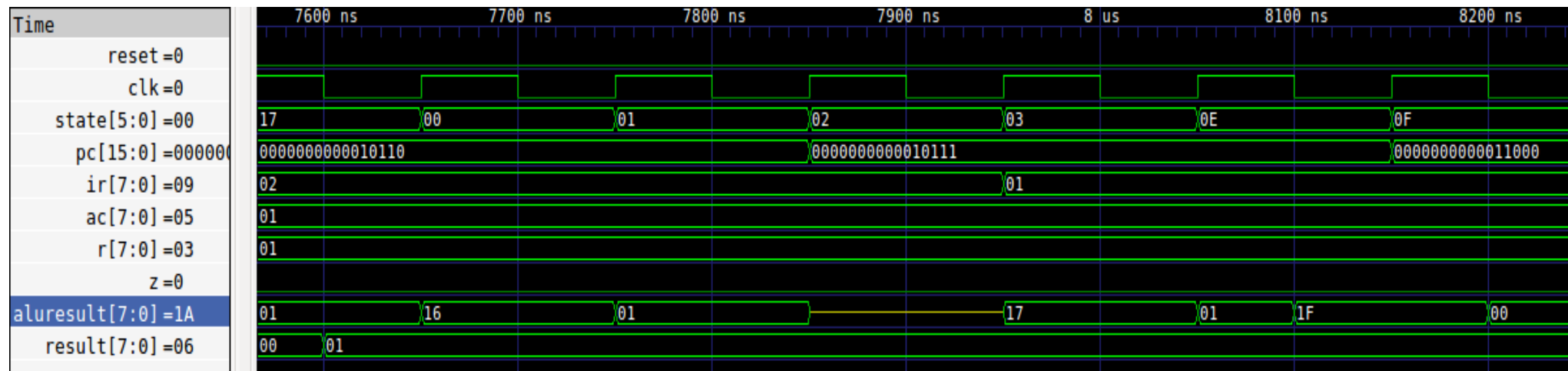
databus <= memdata(to_integer(unsigned(addr))) when (write='0') else "ZZZZZZZZ";
result<=memdata(29);
```

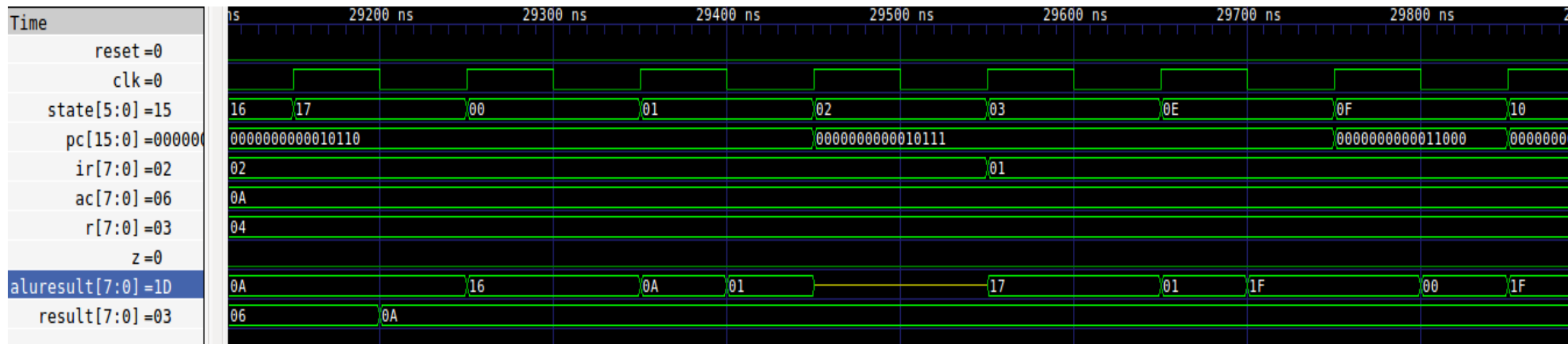
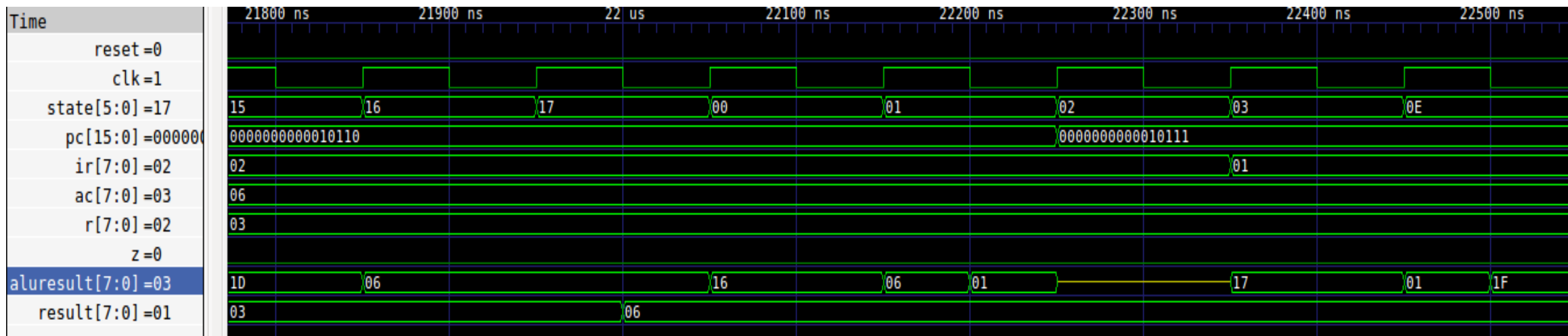



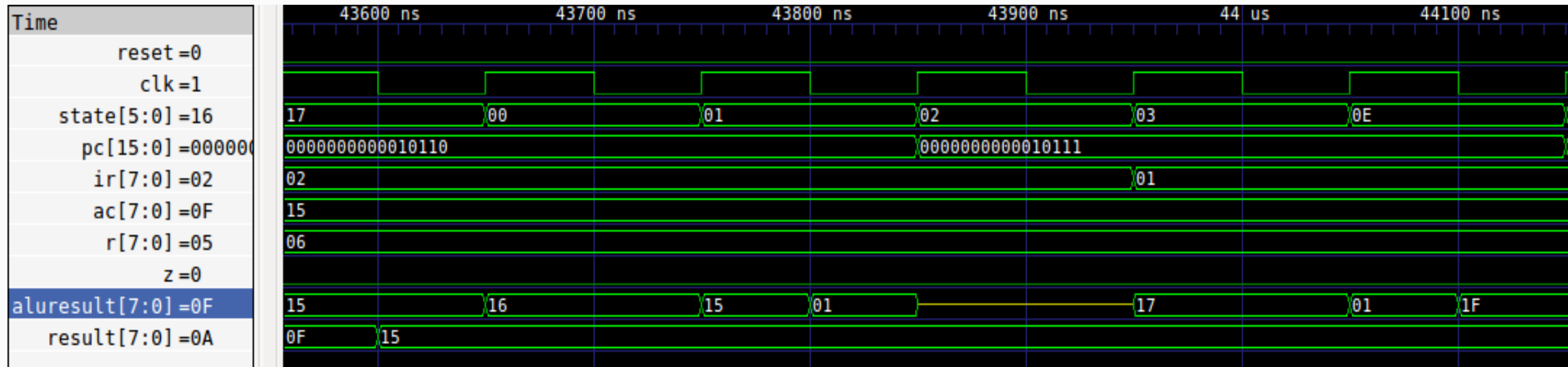
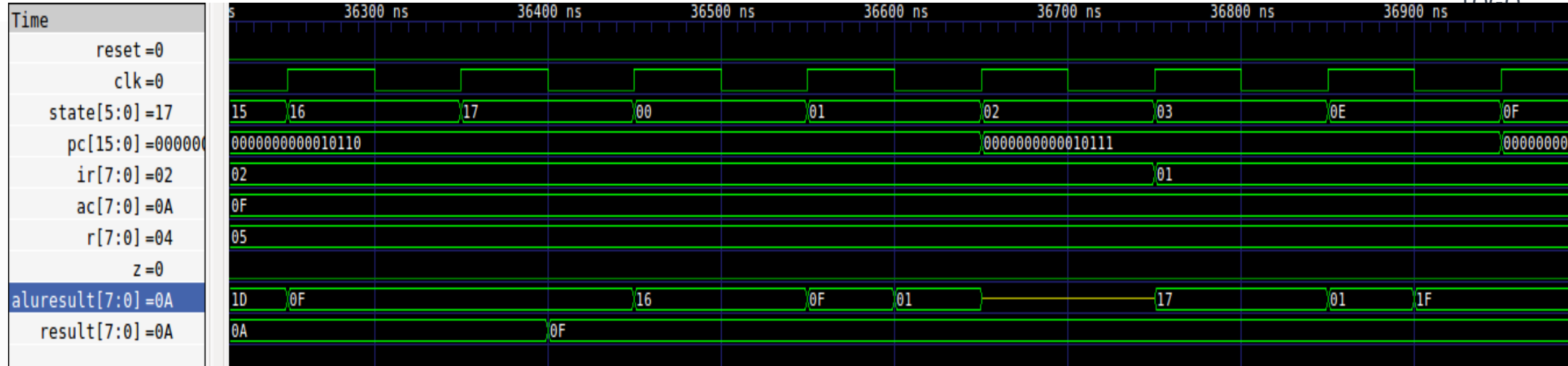
```
reset <= '1', '0' after 300 ns;
clk <= not clk after 50 ns;
```

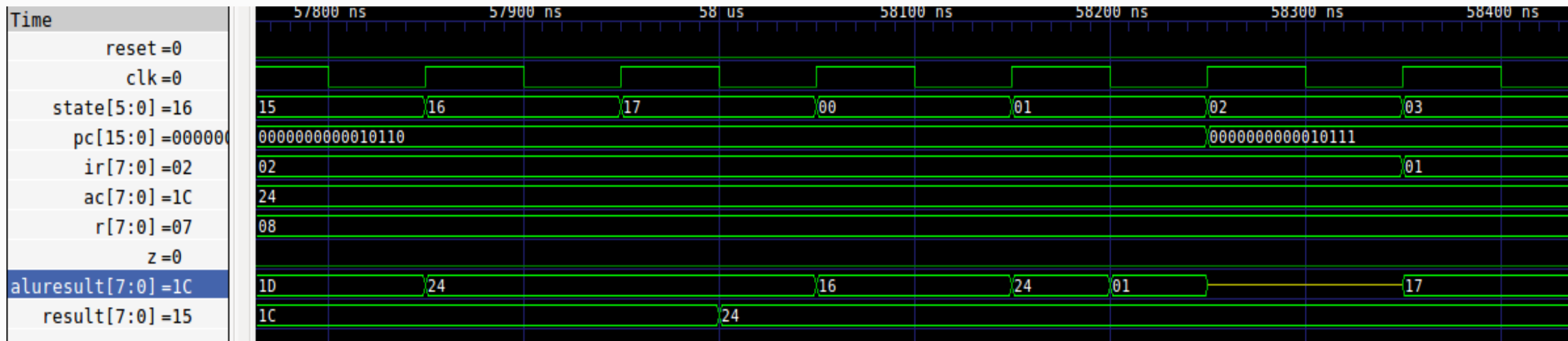
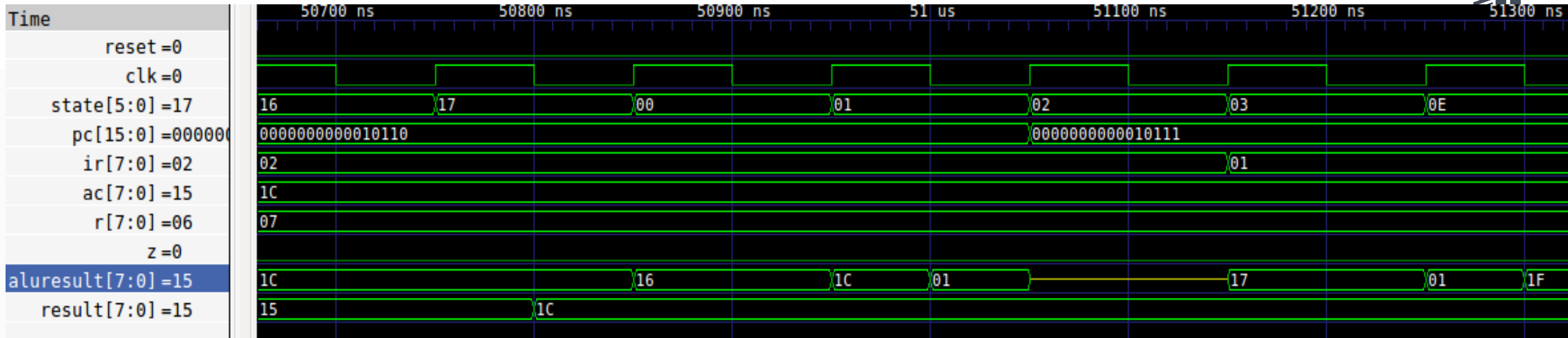
在comp.vhd文件中，对cpu和mem端口进行了连接，并规定了时钟周期和reset的时间大小。

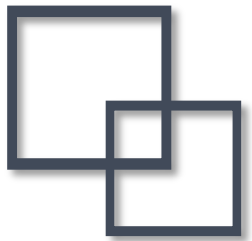
仿真结果











THANK YOU