CPU设计实验报告

201608010713张杨康
智能1602

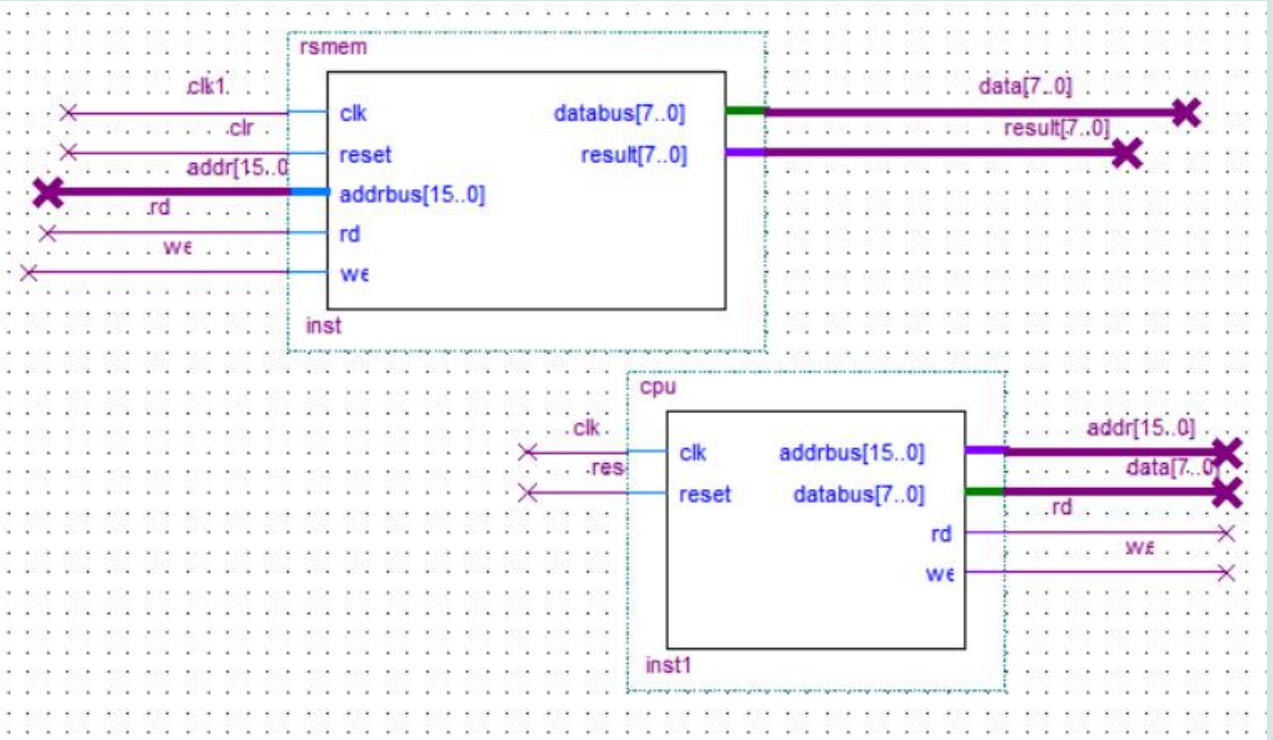
2019/1/3

CONTENT

设计思路

1）顶层文件

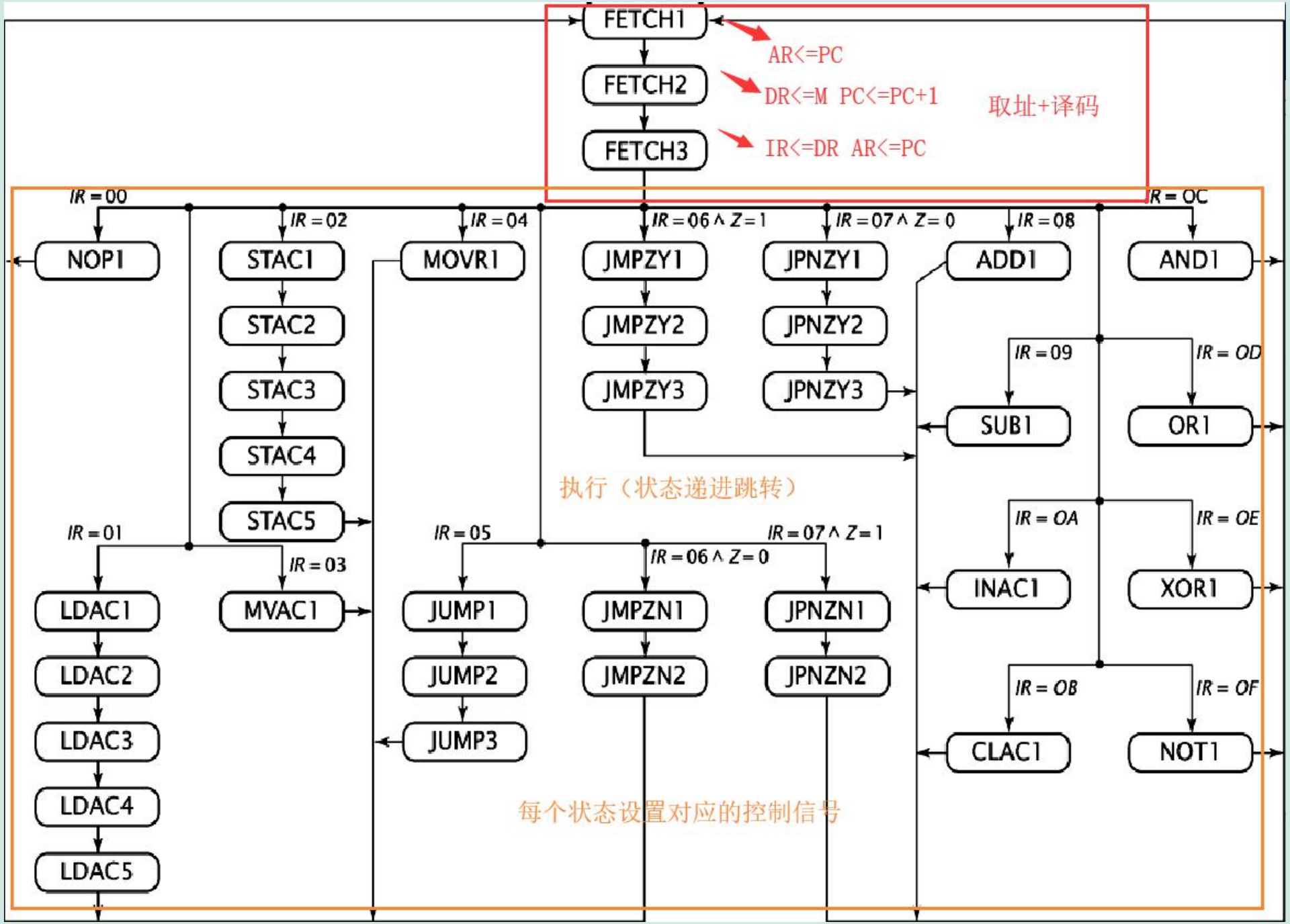
```
component rscpu is
        port(
                clk: in std_logic;
                reset: in std_logic;
                addrbus: out std_logic_vector(15 downto 0);
                databus: inout std_logic_vector(7 downto 0);
                rd: out std_logic;
                wr: out std_logic
            );
end component;

component 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);
                rd: in std_logic;
                wr: in std_logic
            );
```



存储器模块 + CPU模块

设计思路

2）整体思路


FETCH1
AR<=PC
FETCH2
DR<=M PC<=PC+1
取址+译码
FETCH3
IR<=DR AR<=PC
IR = 00
NOP1
IR = 02
STAC1
STAC2
STAC3
STAC4
STAC5
IR = 04
MOVR1
IR = 06 ∧ Z = 1
JMPZY1
JMPZY2
JMPZY3
IR = 07 ∧ Z = 0
JPNZY1
JPNZY2
JPNZY3
IR = 08
ADD1
IR = 09
SUB1
IR = OC
AND1
IR = OD
OR1
执行（状态递进跳转）
IR = 01
LDAC1
LDAC2
LDAC3
LDAC4
LDAC5
IR = 03
MVAC1
IR = 05
JUMP1
JUMP2
JUMP3
IR = 06 ∧ Z = 0
JMPZN1
JMPZN2
IR = 07 ∧ Z = 1
JPNZN1
JPNZN2
IR = OA
INAC1
IR = OE
XOR1
IR = OB
CLAC1
IR = OF
NOT1
每个状态设置对应的控制信号

模块设计--存储器模块

1）程序装载

```
0 => RSCLAC,
1 => RSSTAC,--m<=ac    total=0
2 => std_logic_vector(to_unsigned(total_addr, 8)),
3 => X"00",
4 => RSSTAC,--m<=ac    i=0
5 => std_logic_vector(to_unsigned(i_addr, 8)),
6 => X"00",
7 => RSLDAC,  --    ac=i
8 => std_logic_vector(to_unsigned(i_addr, 8)),
9 => X"00",
10 => RSINAC, --ac++
11 => RSSTAC, --i=ac
12 => std_logic_vector(to_unsigned(i_addr, 8)),
13 => X"00",
14 => RSMVAC, --r=ac
15 => RSLDAC, --ac=total
16 => std_logic_vector(to_unsigned(total_addr, 8)),
17 => X"00",
18 => RSADD, --ac=ac+r
19 => RSSTAC, --total=ac
20 => std_logic_vector(to_unsigned(total_addr, 8)),
21 => X"00",
22 => RSLDAC, --ac=n
23 => std_logic_vector(to_unsigned(n_addr, 8)),
24 => X"00",
25 => RSSUB, --ac=ac-r
26 => RSJPNZ,
27 => std_logic_vector(to_unsigned(loop_addr, 8)),
28 => X"00",
29 => X"00",  -- total
30 => X"00",  -- i
31 => X"06",  -- n
others => RSNOP
```

n=6

```
        CLAC
        STAC  total
        STAC  i
Loop:   LDAC  i
        INAC
        STAC  i
        MVAC
        LDAC  total
        ADD
        STAC  total
        LDAC  n
        SUB
        JPNZ  Loop
total:
   i:
```

total = 0 , i = 0

i = i + 1

total = total + i

IF i≠n THEN GOTO Loop

模块设计--存储器模块

2）内存管理（读、写）

```
begin
    -- The process takes addrbus and rd/we 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(we='1') then
                memdata(to_integer(unsigned(addr))) <= databus;
            end if;
        end if;
    end process;

    databus <= memdata(to_integer(unsigned(addr))) when (we='0') else "ZZZZZZZZ";
    result<=memdata(60);
end architecture;
```

写

读

累加结果

模块设计--CPU模块（指令处理）

1）设置初始化寄存器、控制信号、各个状态：

寄存器：

```
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;
```

控制信号：

```
signal pcload: std_logic;
signal arload: std_logic;
signal drload: std_logic;
signal irload: std_logic;
signal acload: std_logic;
signal rload:  std_logic;
signal trload: std_logic;
signal pcbus: std_logic;
signal membus: std_logic;
signal rbus: std_logic;
signal acbus: std_logic;
signal trbus: std_logic;
signal drbus: std_logic;
```

状态标号：

```
constant fetch1:
constant fetch2:
constant fetch3:
constant fetch4:
constant clac1:
constant incac1:
constant add1:
constant sub1:
constant and1:
constant or1:
constant xor1:
constant not1:
constant mvac1:
constant movr1:
constant ldac1:
constant ldac2:
constant ldac3:
constant ldac4:
constant ldac5:
constant stac1:
constant stac2:
constant stac3:
constant stac4:
constant stac5:
```

模块设计--CPU模块（指令处理）

2）设置ALU运算：

```
alu<=data(7 downto 0)                                          when s="0000"   else
std_logic_vector(unsigned(ac)+unsigned(data(7 downto 0)))          when s="0001"   else
std_logic_vector(unsigned(ac)-unsigned(data(7 downto 0)))          when s="0010"   else
ac and data(7 downto 0)                                            when s="0011"   else
ac or data(7 downto 0)                                             when s="0100"   else
not ac                                                             when s="0110"   else
ac xor data(7 downto 0)                                            when s="0101"   else
data(7 downto 0);
```

模块设计--CPU模块（指令处理）

3）控制信号控制数据的传输：

```
if(ar_ld='1')    then     ar<=buss;
end if;
if(dr_ld='1')    then     dr<=buss(7 downto 0);
end if;
if(ir_ld='1')    then     ir<=dr;
end if;
if(r_d='1') then     r<=buss(7 downto 0);
end if;
if(tr_ld='1')    then     tr<=dr;
end if;
if(pc_ld='1')    then     pc<=buss;
end if;
```

模块设计--CPU模块（指令处理）

4）设置每个状态对应的控制信号：

```
elsif(state=addl)   then
arload<='0';pcbus<='0';pcinc<='0';drload<='0';membus<='0';irload<='0';acreset<='0';acinc<='0';
rbus<='1';s<="0001";acload<='1';rload<='0';acbus<='0';arinc<='0';trbus<='0';drbus<='0';trload<='0';
rd<='0';writel<='0';we<='0';pcload<='0';
```

```
 if(state=fetchl)     then     --ar<=pc
 arload<='1';pcbus<='1';pcinc<='0';drload<='0';membus<='0';irload<='0';acreset<='0';acinc<='0';
 rbus<='0';s<="0000";acload<='0';rload<='0';acbus<='0';arinc<='0';trbus<='0';drbus<='0';trload<='0';
```

模块设计--CPU模块（指令处理）

5）执行时状态间的切换：

单状态指令：

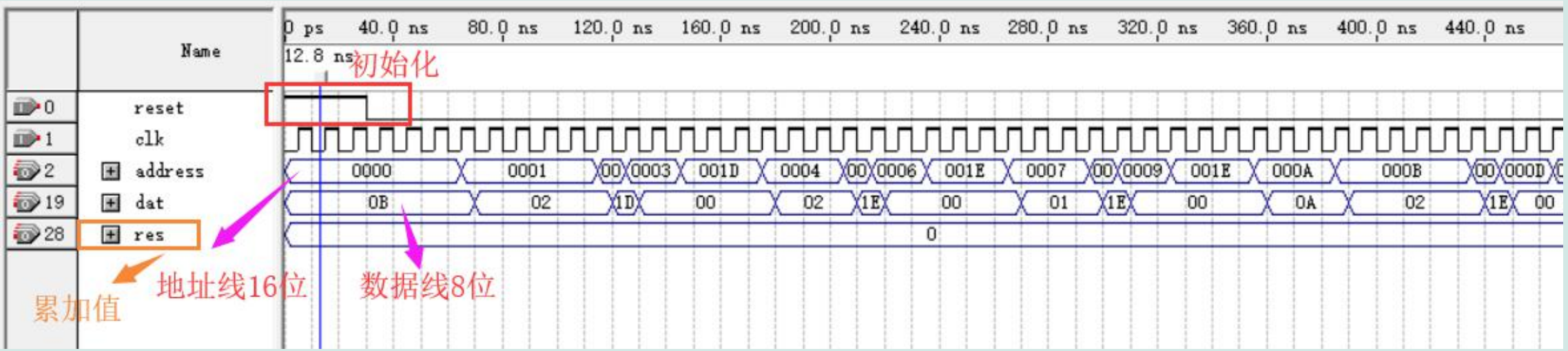
```
elsif(ir=add)          then     nextstate<=add1;
```

多状态指令：

```
if(state=fetch1)     then     nextstate<=fetch2;
elsif(state=fetch2)  then     nextstate<=fetch3;
elsif(state=fetch3)  then
```

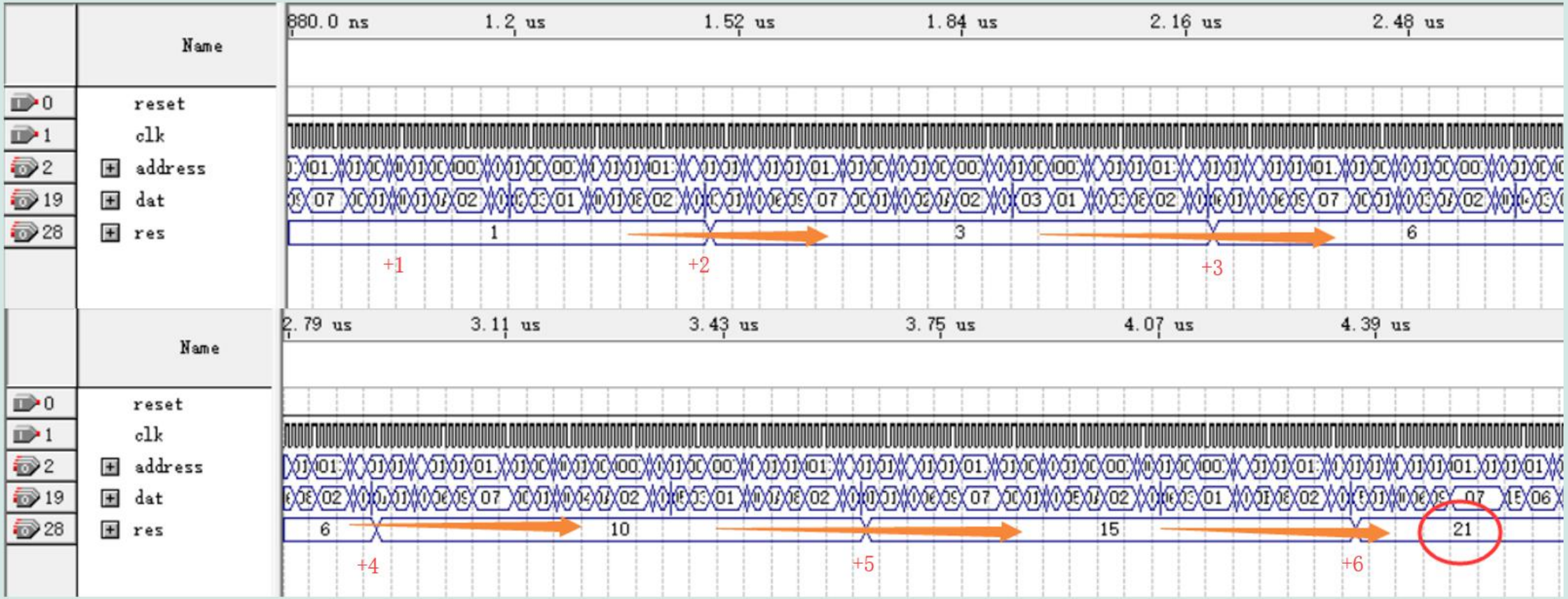
```
elsif(state=jump1)   then     nextstate<=jump2;
elsif(state=jump2)   then     nextstate<=jump3;
elsif(state=jump3)   then     nextstate<=fetch1;
```

仿真结果


Name
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
280.0 ns
320.0 ns
360.0 ns
400.0 ns
440.0 ns
12.8 ns
初始化
0 reset
1 clk
2 address
19 dat
28 res
0000 0001 0003 001D 0004 0006 001E 0007 0009 001E 000A 000B 000D
0B 02 1D 00 02 1E 00 01 1E 00 0A 02 1E 00
0
累加值
地址线16位
数据线8位

仿真结果

（n=6）



经检验，结果正确

Thanks.

感谢观看！