相对简单 CPU 电路设计设计报告

班级 智能 1601 姓名 潘小天 学号 201608010309

一、实验目标

利用 VHDL 设计相对简单 CPU 的电路并验证。

二、实验要求

① 采用 VHDL 描述电路及其测试平台

② 采用时序逻辑设计电路

③ 采用从 1 累加到N的程序进行测试

三、实验内容

3.1 设计的整体架构

这个是本次实验的实验要求的数据通路

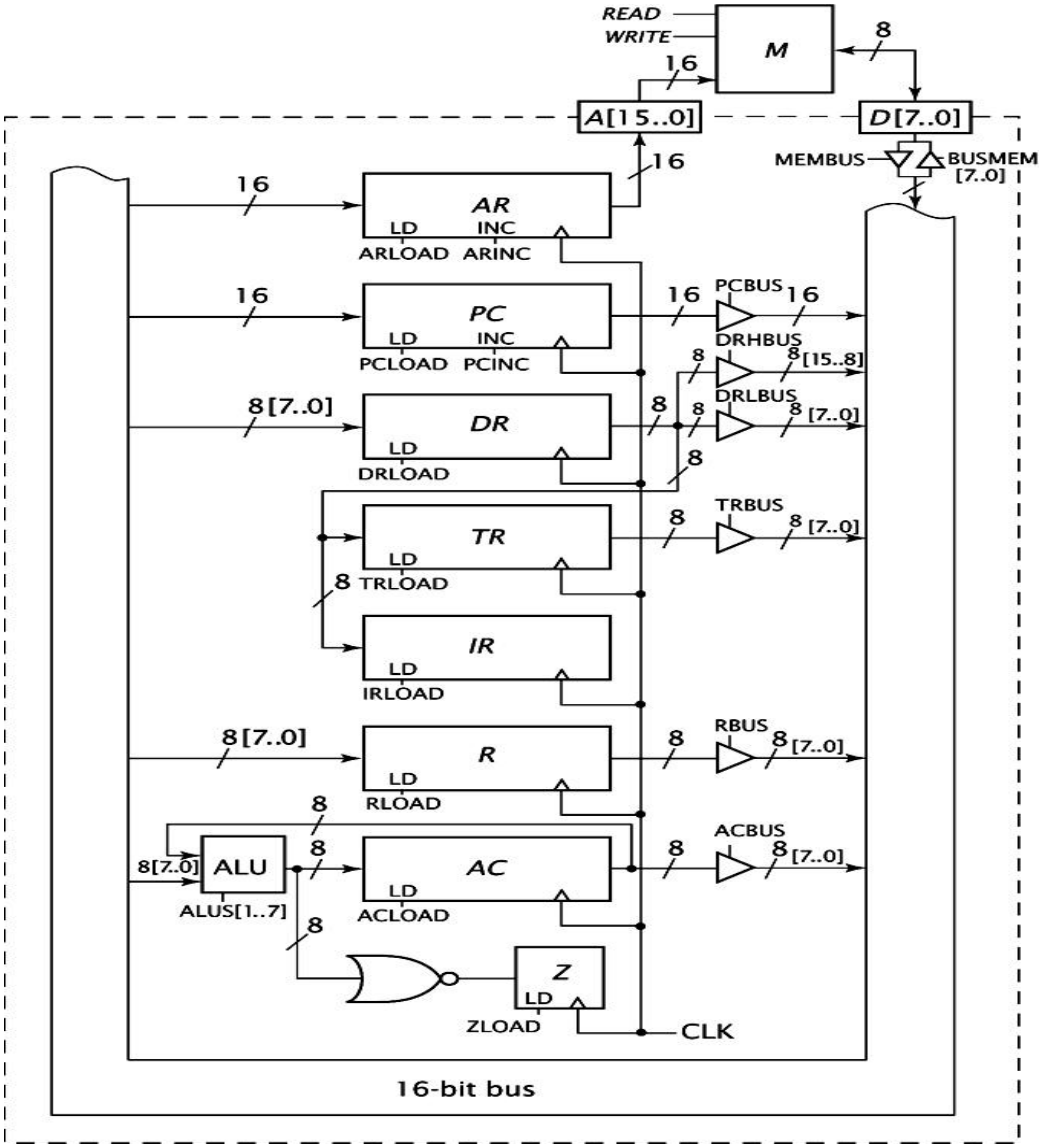


图 1 实验要求数据通路图

相对简单 CPU 的设计需求请详见课件，主要特征如下：

① 地址总线 16 位，数据总线 8 位

② 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

③ 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

④ 有 16 条指令，每条指令 1 个或 3 个字节，其中操作码 8 位 0.3 字节的指令有 16 位的地址

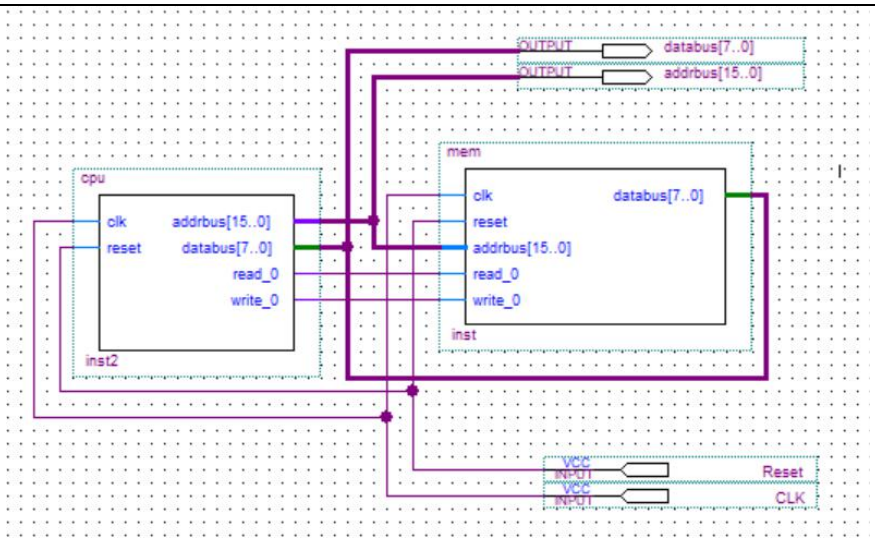


图 2 我设计顶层图

16 条指令对应指令码和操作

指令	指令码	操作
NOP	0000 0000	无
LDAC	0000 0001 Γ	AC←M[Γ]
STAC	0000 0010 Γ	M[Γ]←AC
MVAC	0000 0011	R←AC
MOVR	0000 0100	AC←R
JUMP	0000 0101 Γ	GOTO Γ
JMPZ	0000 0110 Γ	IF（Z＝1） THEN GOTO Γ
JPNZ	0000 0111 Γ	IF（Z＝0） THEN GOTO Γ
ADD	0000 1000	AC←AC＋R，IF（AC＋R＝0） THEN Z←1 ELSE Z←0
SUB	0000 1001	AC←AC－R，IF（AC－R＝0） THEN Z←1 ELSE Z←0
INAC	0000 1010	AC←AC＋1，IF（AC＋1＝0） THEN Z←1 ELSE Z←0
CLAC	0000 1011	AC←0，Z←1
AND	0000 1100	AC←AC∧R，IF（AC∧R＝0） THEN Z←1 ELSE Z←0
OR	0000 1101	AC←AC∨R，IF（AC∨R＝0） THEN Z←1 ELSE Z←0
XOR	0000 1110	AC←AC⊕ R，IF（AC⊕ R＝0） THEN Z←1 ELSE Z←0
NOT	0000 1111	AC←AC’，IF（AC’＝0） THEN Z←1 ELSE Z←0

3.2 相对简单 CPU 设计方案

相对简单 CPU 的设计方案请详见课件，主要思路如下：

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，FETCH4
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

mem 对应代码：

```
signal memdata: memtype(4095 downto 0) := (
0 => RSCLAC,
1 => RSSTAC,--m[total_total]<=ac    total=0
2 => std_logic_vector(to_unsigned(total_addr, 8)),
3 => X"00",
4 => RSSTAC,--m[i_addr]<=ac    i=0
5 => std_logic_vector(to_unsigned(i_addr, 8)),
6 => X"00",
7 => RSLDAC,  -- loop    --ac<=m[i_addr]    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)),
```

cpu 对应代码：

```
constant fetch1:    std_logic_vector(5 downto 0) := "000000";-- ar<=pc
constant fetch2:    std_logic_vector(5 downto 0) := "000001";-- dr<=m  pc<=pc+1
constant fetch3:    std_logic_vector(5 downto 0) := "000010";-- ir<=dr
constant fetch4:    std_logic_vector(5 downto 0) := "000011";-- ar<=pc
constant clac1:     std_logic_vector(5 downto 0) := "000100";--ac<=0 z<=1
constant incac1:    std_logic_vector(5 downto 0) := "000101";--ac++ z change
constant add1:      std_logic_vector(5 downto 0) := "000110";--ac=ac+r z change
constant sub1:      std_logic_vector(5 downto 0) := "000111";--ac=ac-r z change
constant and1:      std_logic_vector(5 downto 0) := "001000";--ac=ac&r z change
constant or1:       std_logic_vector(5 downto 0) := "001001";--ac=ac|r z change
constant xor1:      std_logic_vector(5 downto 0) := "001010";--ac=ac xor r z change
constant not1:      std_logic_vector(5 downto 0) := "001011";--ac=not ac z change
constant mvac1:     std_logic_vector(5 downto 0) := "001100";--r<=ac
constant movr1:     std_logic_vector(5 downto 0) := "001101";--ac<=r
constant ldac1:     std_logic_vector(5 downto 0) := "001110";--dr<=m    pc=pc+1    ar=ar+1
constant ldac2:     std_logic_vector(5 downto 0) := "001111";--tr<=dr   dr<=m      pc=pc+1
constant ldac3:     std_logic_vector(5 downto 0) := "010000";--ar<=dr, tr
constant ldac4:     std_logic_vector(5 downto 0) := "010001";--dr<=m
constant ldac5:     std_logic_vector(5 downto 0) := "010010";--ac<=dr
constant stac1:     std_logic_vector(5 downto 0) := "010011";--dr<=m  pc++ ar++
constant stac2:     std_logic_vector(5 downto 0) := "010100";--tr<=dr dr<=m pc++
constant stac3:     std_logic_vector(5 downto 0) := "010101";--ar<=dr,tr
constant stac4:     std_logic_vector(5 downto 0) := "010110";--dr<=ac
constant stac5:     std_logic_vector(5 downto 0) := "010111";--m<=dr
```

```
if(state=fetch1)    then
arload<='1';pcbus<='1';pcinc<='0';drload<='0';membus<='0';irload<='0';acreset<='0';acinc<='0';--ar<=pc
rbus<='0';s<="0000";acload<='0';rload<='0';acbus<='0';arinc<='0';trbus<='0';drbus<='0';trload<='0';
read_0<='0';write_1<='0';write_0<='0';pcload<='0';
elsif(state=fetch2) then
arload<='0';pcbus<='0';pcinc<='1';drload<='1';membus<='1';irload<='0';acreset<='0';acinc<='0';-- dr<=m  pc<=pc+1
rbus<='0';s<="0000";acload<='0';rload<='0';acbus<='0';arinc<='0';trbus<='0';drbus<='0';trload<='0';
read_0<='1';write_1<='0';write_0<='0';pcload<='0';
elsif(state=fetch3) then
arload<='0';pcbus<='0';pcinc<='0';drload<='0';membus<='0';irload<='1';acreset<='0';acinc<='0';--ir<=dr
rbus<='0';s<="0000";acload<='0';rload<='0';acbus<='0';arinc<='0';trbus<='0';drbus<='0';trload<='0';
read_0<='0';write_1<='0';write_0<='0';pcload<='0';
elsif(state=fetch4) then
arload<='1';pcbus<='1';pcinc<='0';drload<='0';membus<='0';irload<='0';acreset<='0';acinc<='0';--ar<=pc
rbus<='0';s<="0000";acload<='0';rload<='0';acbus<='0';arinc<='0';trbus<='0';drbus<='0';trload<='0';
read_0<='0';write_1<='0';write_0<='0';pcload<='0';
elsif(state=clac1)  then
arload<='0';pcbus<='0';pcinc<='0';drload<='0';membus<='0';irload<='0';acreset<='1';acinc<='0';--ac<=0   z<=1
rbus<='0';s<="0000";acload<='0';rload<='0';acbus<='0';arinc<='0';trbus<='0';drbus<='0';trload<='0';
read_0<='0';write_1<='0';write_0<='0';pcload<='0';
```

```
if(state=fetch1)     then    nextstate<=fetch2;
elsif(state=fetch2)  then    nextstate<=fetch3;
elsif(state=fetch3)  then    nextstate<=fetch4;
elsif(state=fetch4)  then
    if(ir=RSCLAC)        then    nextstate<=clac1;
    elsif(ir=RSINAC)     then    nextstate<=incac1;
    elsif(ir=RSADD)      then    nextstate<=add1;
    elsif(ir=RSSUB)      then    nextstate<=sub1;
    elsif(ir=RSAND)      then    nextstate<=and1;
    elsif(ir=RSOR)       then    nextstate<=or1;
    elsif(ir=RSXOR)      then    nextstate<=xor1;
    elsif(ir=RSNOT)      then    nextstate<=not1;
    elsif(ir=RSMVAC)     then    nextstate<=mvac1;
    elsif(ir=RSMOVR)     then    nextstate<=movr1;
    elsif(ir=RSLDAC)     then    nextstate<=ldac1;
    elsif(ir=RSSTAC)     then    nextstate<=stac1;
    elsif(ir=RSJUMP)     then    nextstate<=jump1;
```

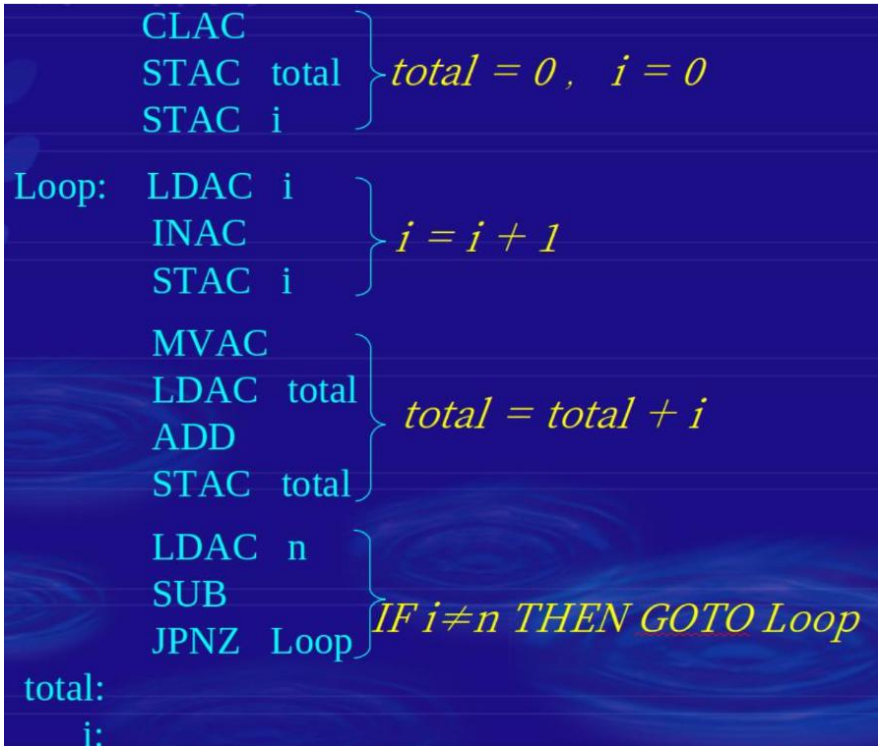
四、注：实验步骤如同控制台系统测试

4.1 测试环境

软件：QUARTUS II 9.0

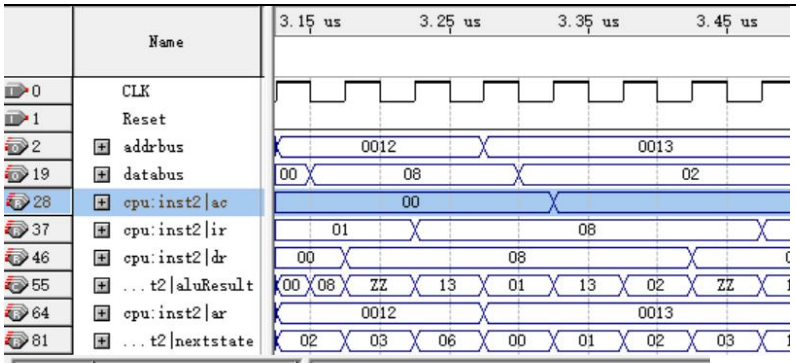
仿真芯片：EP2C5F256C6

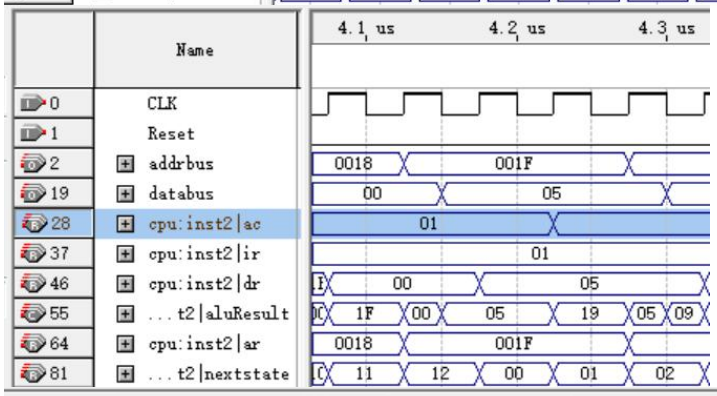
4.2 测试输入

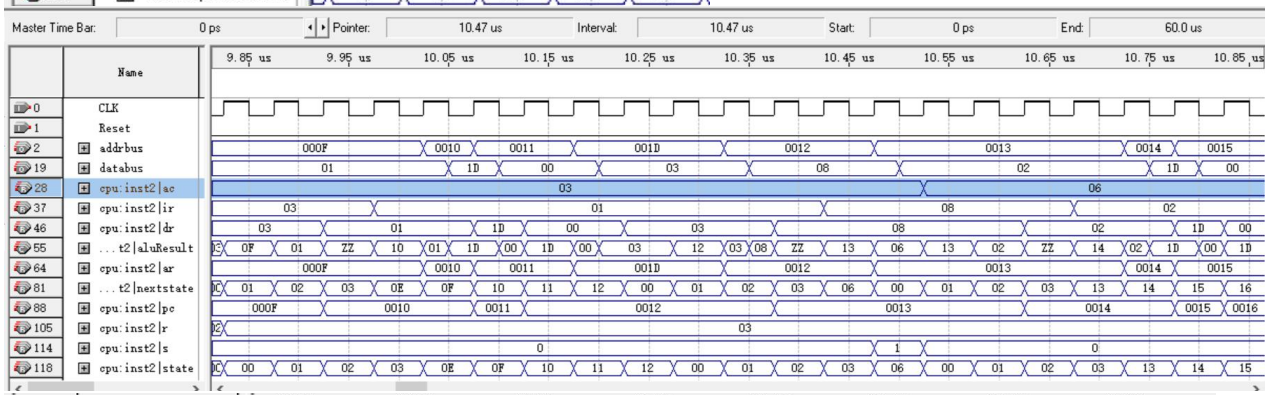


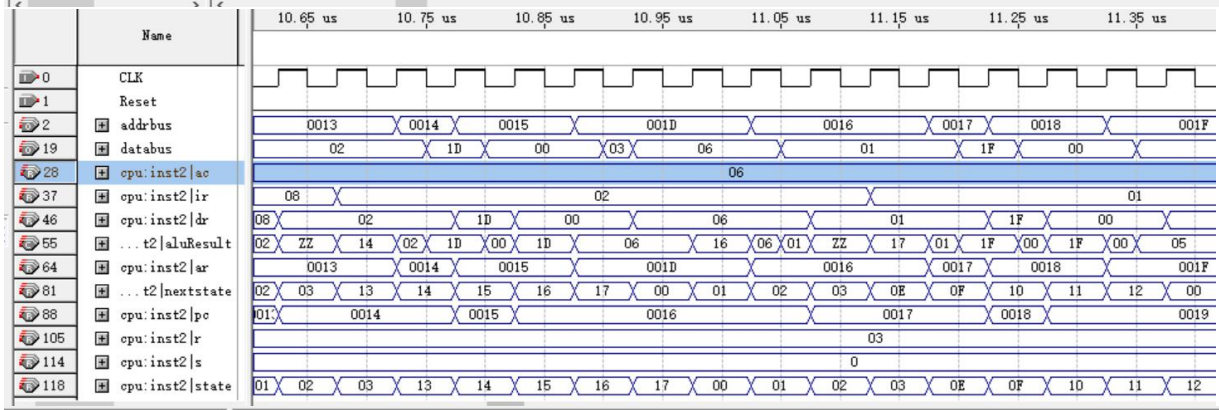
4.3 测试结果

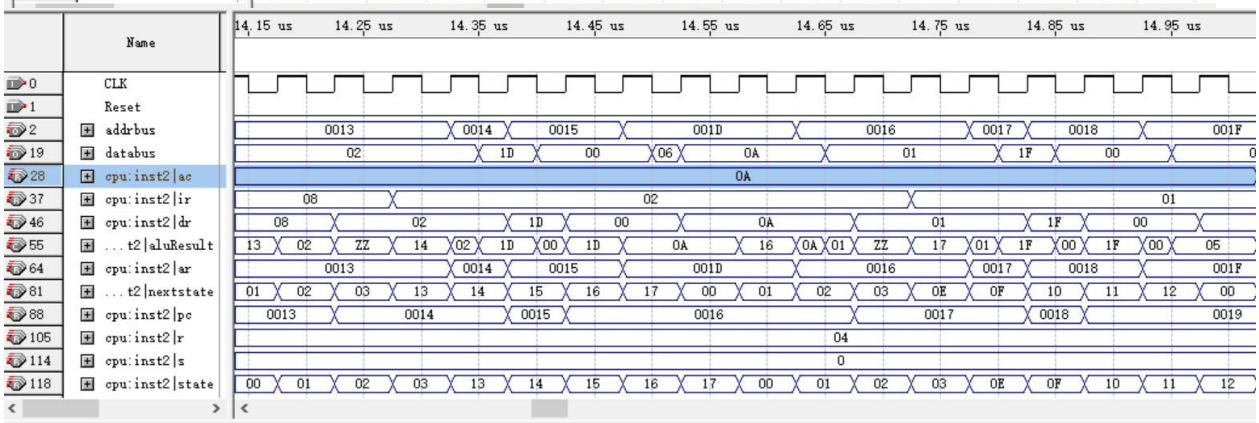
Total 中值为 0，1，3，6，10，15 时对应的寄存器的值

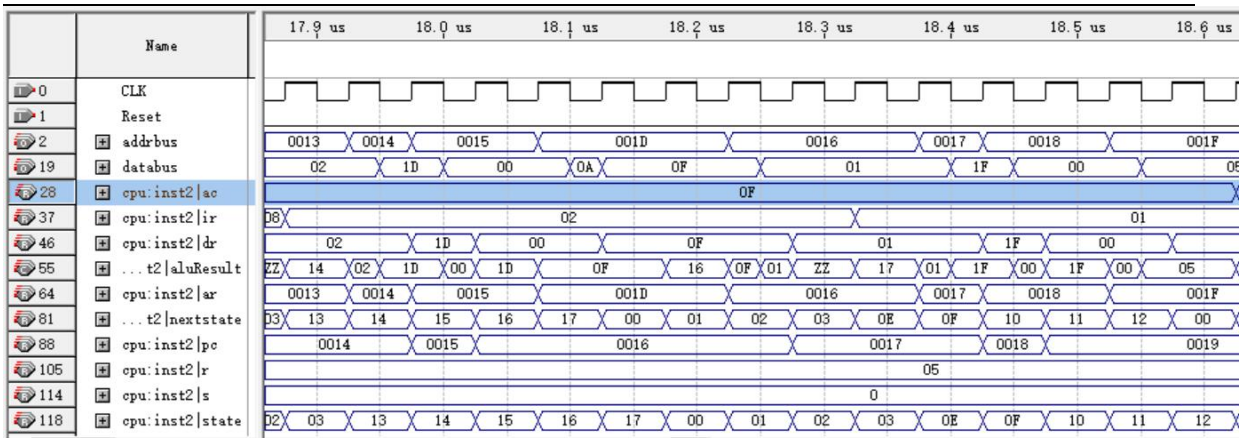

Name
3.15 us
3.25 us
3.35 us
3.45 us
0 CLK
1 Reset
2 addrbus
19 databus
28 cpu:inst2|ac
37 cpu:inst2|ir
46 cpu:inst2|dr
55 ...t2|aluResult
64 cpu:inst2|ar
81 ...t2|nextstate


Name
4.1 us
4.2 us
4.3 us
0 CLK
1 Reset
2 addrbus
19 databus
28 cpu:inst2|ac
37 cpu:inst2|ir
46 cpu:inst2|dr
55 ...t2|aluResult
64 cpu:inst2|ar
81 ...t2|nextstate


Master Time Bar:
Pointer:
Interval:
Start:
End:
Name
0 CLK
1 Reset
2 addrbus
19 databus
28 cpu:inst2|ac
37 cpu:inst2|ir
46 cpu:inst2|dr
55 ...t2|aluResult
64 cpu:inst2|ar
81 ...t2|nextstate
88 cpu:inst2|pc
105 cpu:inst2|r
114 cpu:inst2|s
118 cpu:inst2|state


Name
10.65 us
10.75 us
10.85 us
10.95 us
11.05 us
11.15 us
11.25 us
11.35 us
0 CLK
1 Reset
2 addrbus
19 databus
28 cpu:inst2|ac
37 cpu:inst2|ir
46 cpu:inst2|dr
55 ...t2|aluResult
64 cpu:inst2|ar
81 ...t2|nextstate
88 cpu:inst2|pc
105 cpu:inst2|r
114 cpu:inst2|s
118 cpu:inst2|state


Name
14.15 us
14.25 us
14.35 us
14.45 us
14.55 us
14.65 us
14.75 us
14.85 us
14.95 us
0 CLK
1 Reset
2 addrbus
19 databus
28 cpu:inst2|ac
37 cpu:inst2|ir
46 cpu:inst2|dr
55 ...t2|aluResult
64 cpu:inst2|ar
81 ...t2|nextstate
88 cpu:inst2|pc
105 cpu:inst2|r
114 cpu:inst2|s
118 cpu:inst2|state



计算结束后 z 值由 0 变为 1，计算过程结束：

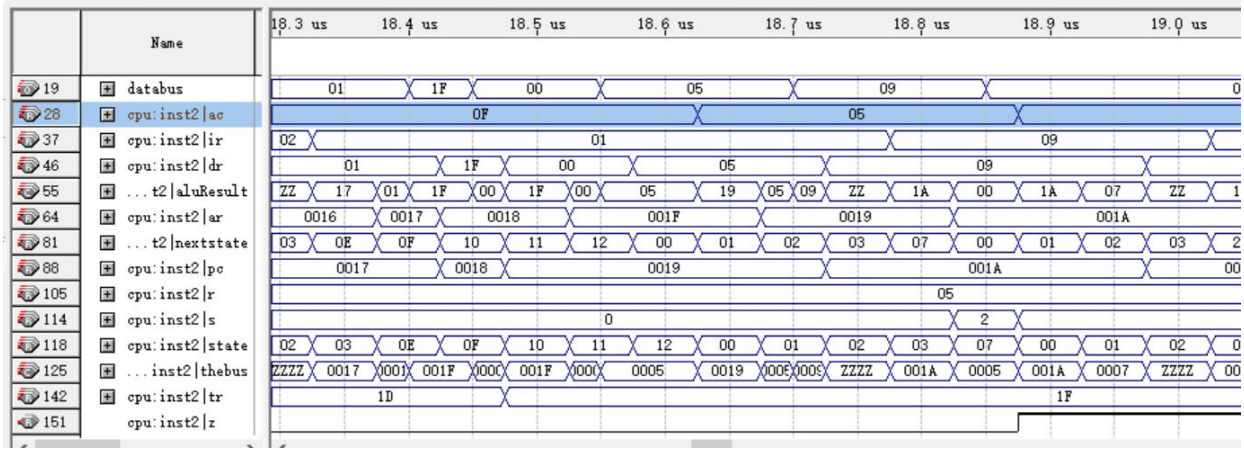


图 12 测试结果原图

仿真结果正确。