相对简单CPU的设计

201608010627 任小禹

目录

01 CPU设计概述

02 遇到的问题及分析

03 测试结果

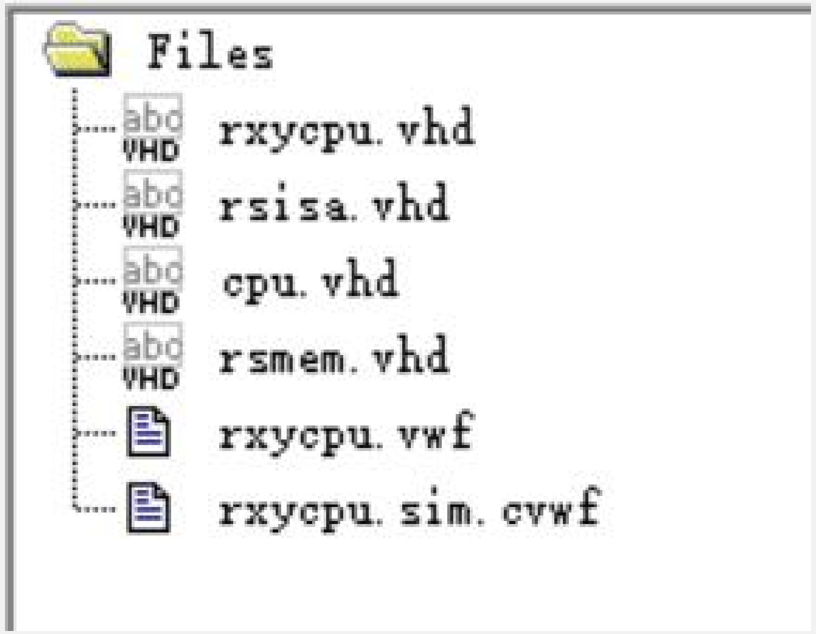
第一部分

Part One

CPU设计概述

CPU设计概述

○ 测试环境：Quartus II，四个vhdl文件



Files
rxycpu.vhd
rsisa.vhd
cpu.vhd
rsmem.vhd
rxycpu.vwf
rxycpu.sim.cvwf

○ 将CPU模块和内存连接起来

○ 将我们CPU的指令集打包成库，方便调用

○ CPU的具体实现

○ 内存读写的实现及输入激励

CPU设计概述

- 指令执行过程分为取指、译码、执行三个阶段
- 取指包括四个状态:

FETCH1,FETCH2,FETCH3，FETCH4

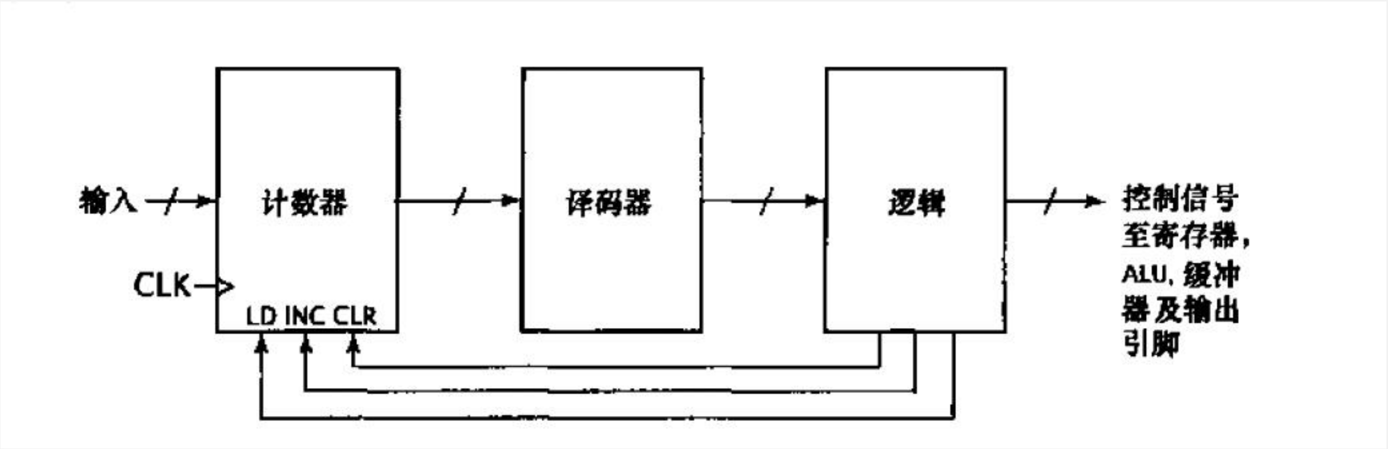
  - FETCH1：AR←PC
  - FETCH2：DR←M，PC←PC＋1
  - FETCH3：IR←DR，
  - FETCH4：AR←PC

- 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
- 这里将取分为四个阶段是因为防止还有译码成功，状态就已经改变

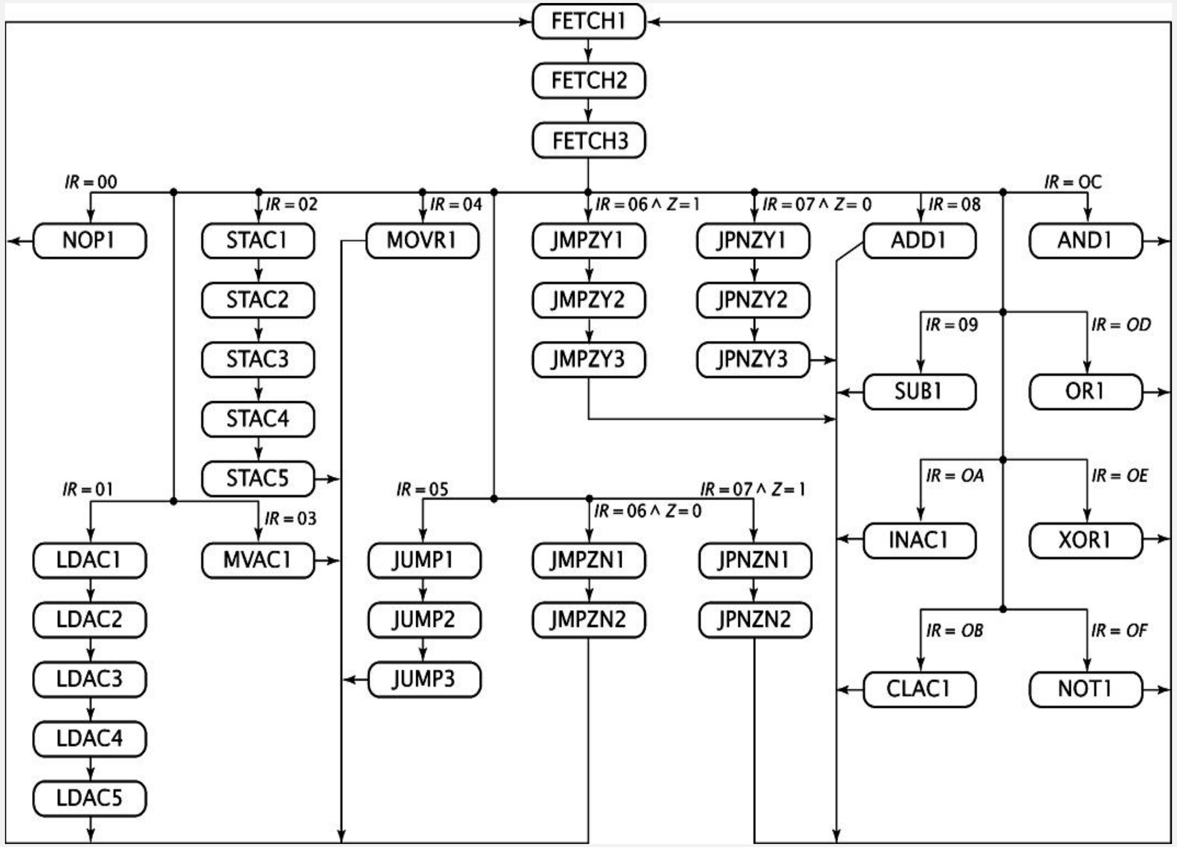
CPU具体实现主要是CPU控制单元的设计

- 硬连线控制
  - 时序逻辑和组合逻辑产生控制信号
- 微程序控制
  - 存储器查表输出控制信号

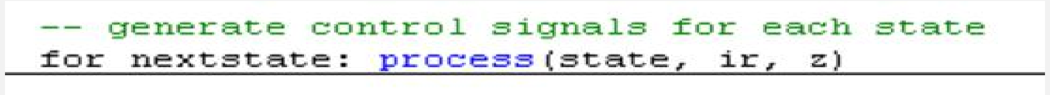
我们采用硬连线控制



CPU设计概述

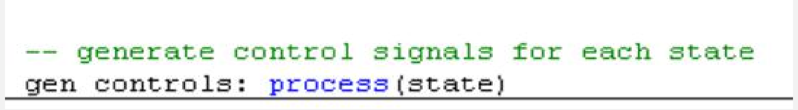


- 我们采用硬连线控制，列出整个CPU的状态图



```
-- generate control signals for each state
for nextstate: process(state, ir, z)
```

- 相当于计数器，产生下一状态

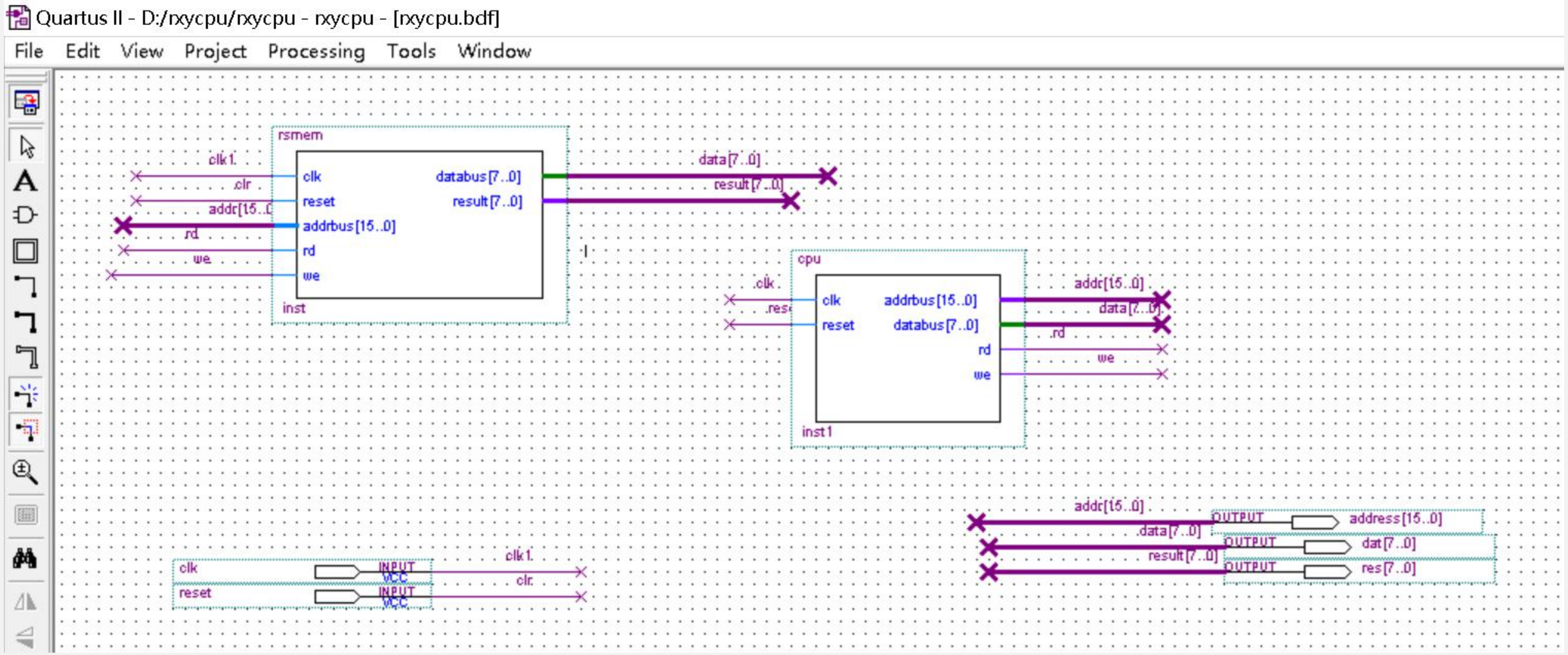


```
-- generate control signals for each state
gen controls: process(state)
```

- 相当于译码器和组合逻辑，根据当前状态产生控制信号

CPU设计概述

- 不用vhdl的元件例化语句，也可以用原理图连线的方法实现

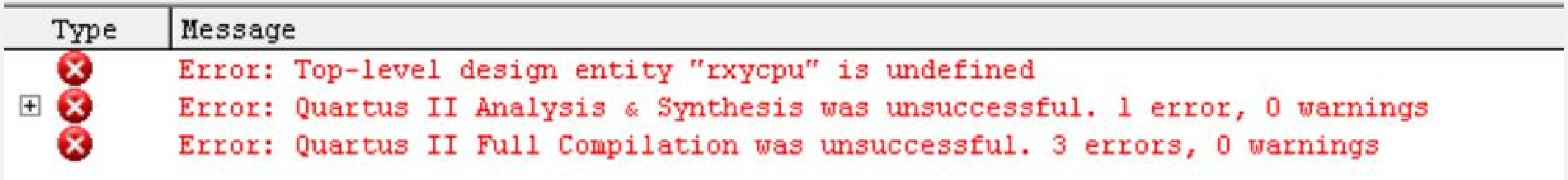


第二部分

Part One

遇到的问题及分析

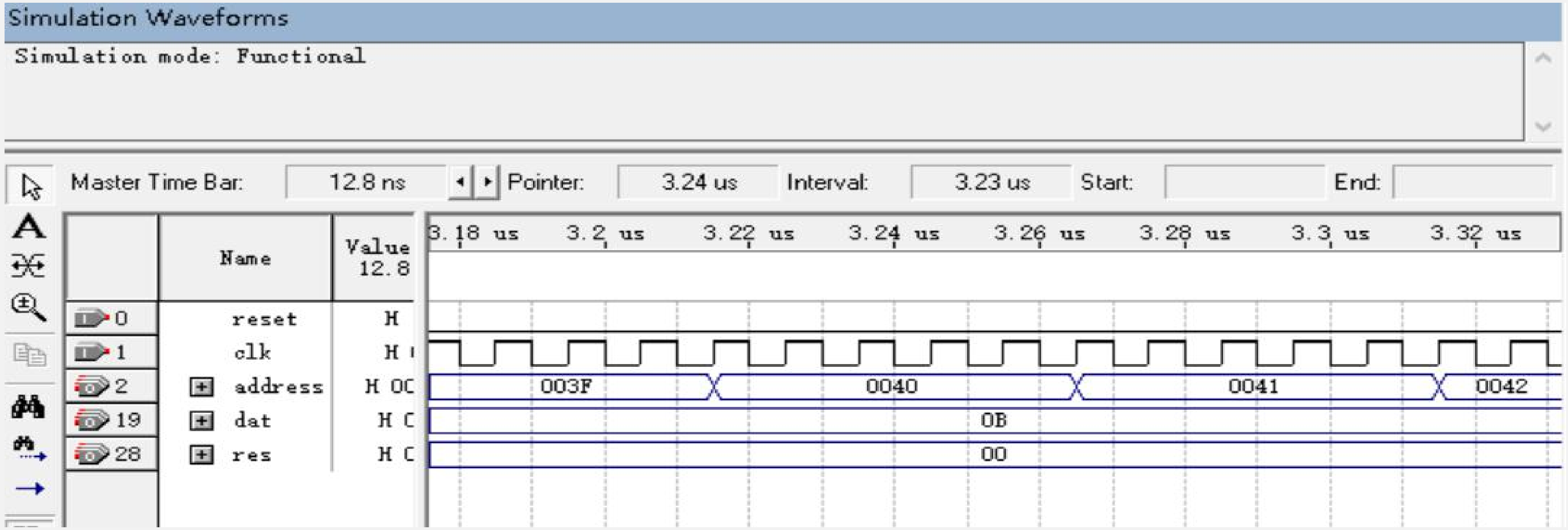
遇到问题及分析1/3



```
Type    Message
        Error: Top-level design entity "rxycpu" is undefined
        Error: Quartus II Analysis & Synthesis was unsuccessful. 1 error, 0 warnings
        Error: Quartus II Full Compilation was unsuccessful. 3 errors, 0 warnings
```

- quartus顶层文件名必须和工程名保存一致

遇到问题及分析 2/3



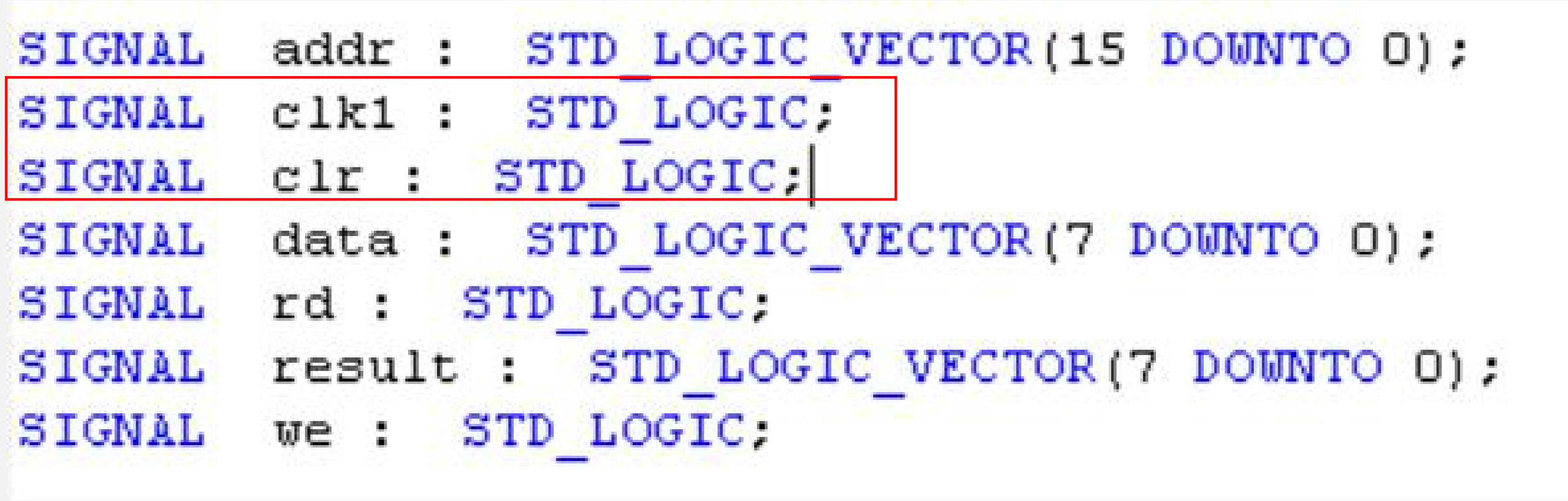
Simulation Waveforms

Simulation mode: Functional

Master Time Bar: 12.8 ns Pointer: 3.24 us Interval: 3.23 us Start: End:

- 数据线数据一直不变，分析原因，可能是元件连接出错

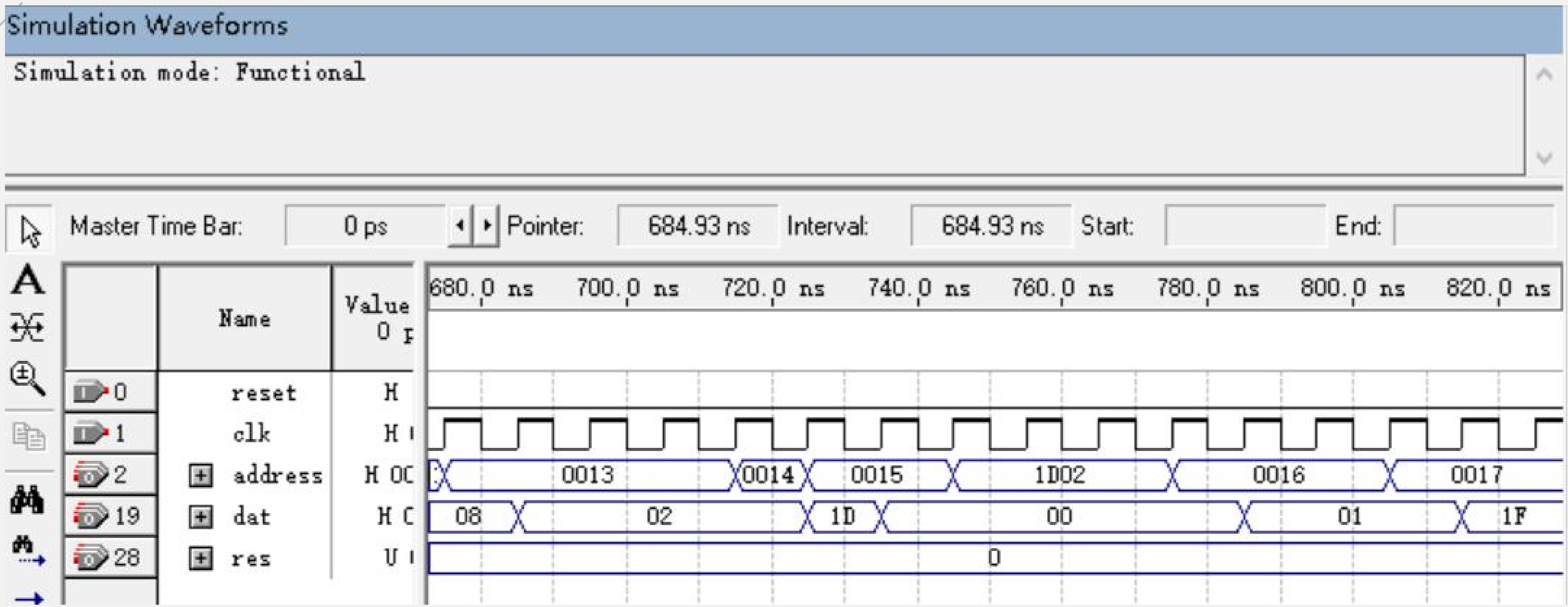
遇到问题及分析 2/3



```
SIGNAL  addr :  STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL  clk1 :  STD_LOGIC;
SIGNAL  clr :  STD_LOGIC;
SIGNAL  data :  STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL  rd :  STD_LOGIC;
SIGNAL  result :  STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL  we :  STD_LOGIC;
```

- 元件连接的文件出错，我编写的元件例化语句有问题
- clk和reset均为输入信号，可以直接通过元件例化语句映射，无需中间信号变量

遇到问题及分析3/3



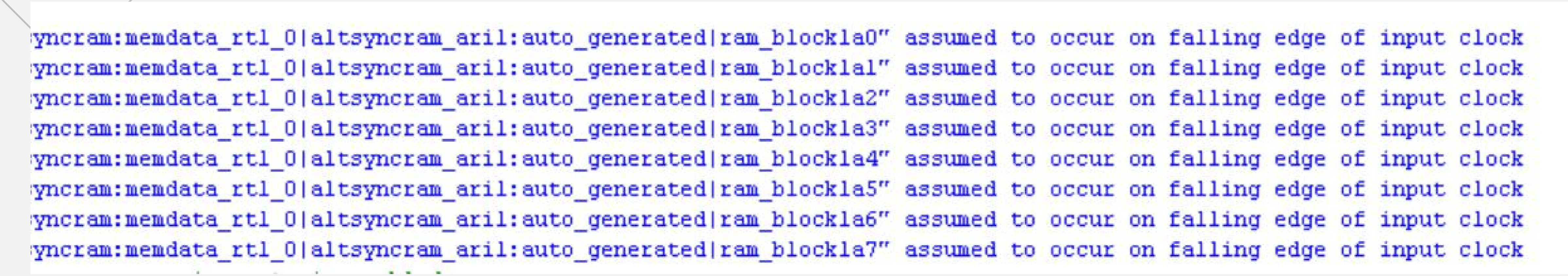
Simulation Waveforms

Simulation mode: Functional

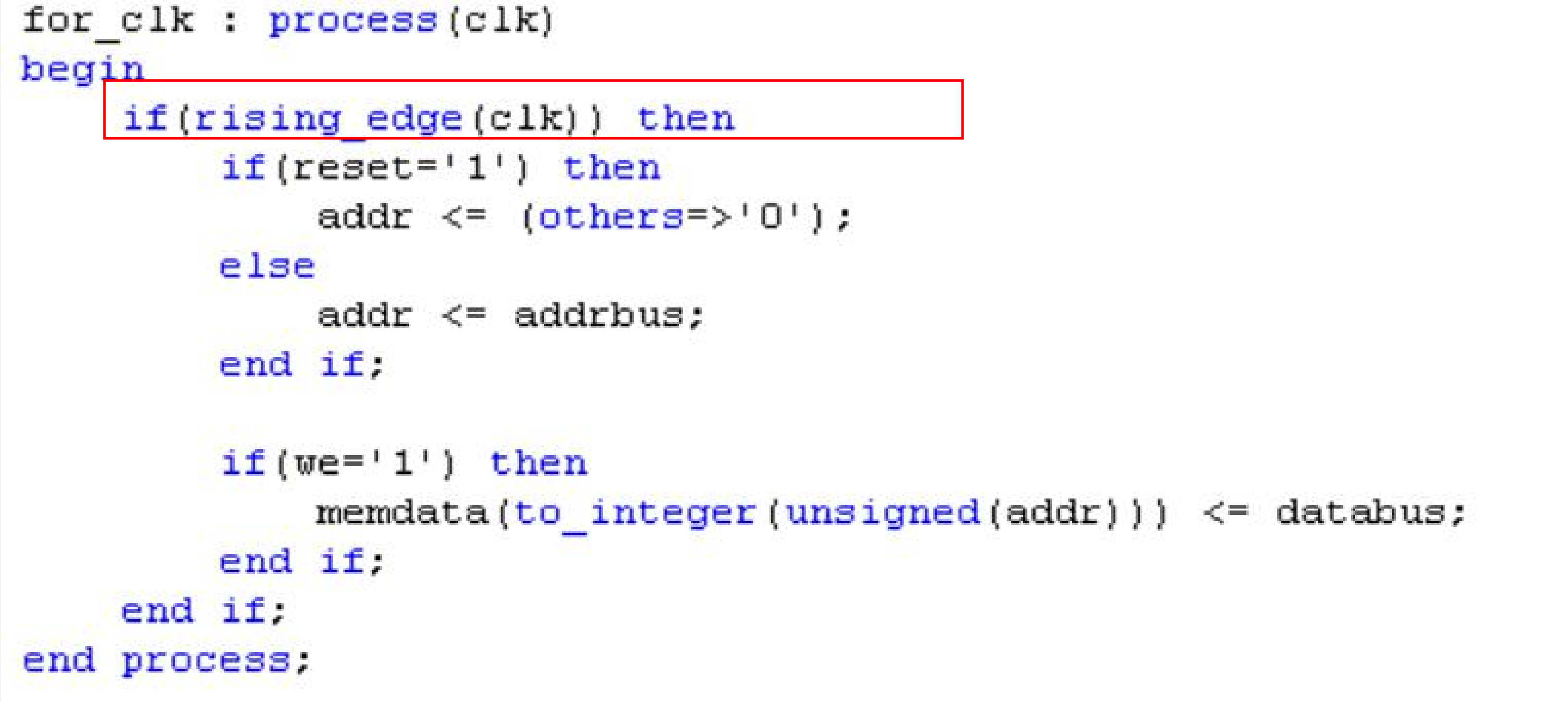
Master Time Bar: 0 ps Pointer: 684.93 ns Interval: 684.93 ns Start: End:

- 数据线有了数据一直不变，但是输出的total值一直为0

遇到问题及分析



```
yncram:memdata_rtl_0|altsyncram_aril:auto_generated|ram_block1a0" assumed to occur on falling edge of input clock
yncram:memdata_rtl_0|altsyncram_aril:auto_generated|ram_block1a1" assumed to occur on falling edge of input clock
yncram:memdata_rtl_0|altsyncram_aril:auto_generated|ram_block1a2" assumed to occur on falling edge of input clock
yncram:memdata_rtl_0|altsyncram_aril:auto_generated|ram_block1a3" assumed to occur on falling edge of input clock
yncram:memdata_rtl_0|altsyncram_aril:auto_generated|ram_block1a4" assumed to occur on falling edge of input clock
yncram:memdata_rtl_0|altsyncram_aril:auto_generated|ram_block1a5" assumed to occur on falling edge of input clock
yncram:memdata_rtl_0|altsyncram_aril:auto_generated|ram_block1a6" assumed to occur on falling edge of input clock
yncram:memdata_rtl_0|altsyncram_aril:auto_generated|ram_block1a7" assumed to occur on falling edge of input clock
```



```
for_clk : process(clk)
begin
    if(rising_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;
```

- 仿真器提示内存读写在下降沿，将上升沿改为下降沿

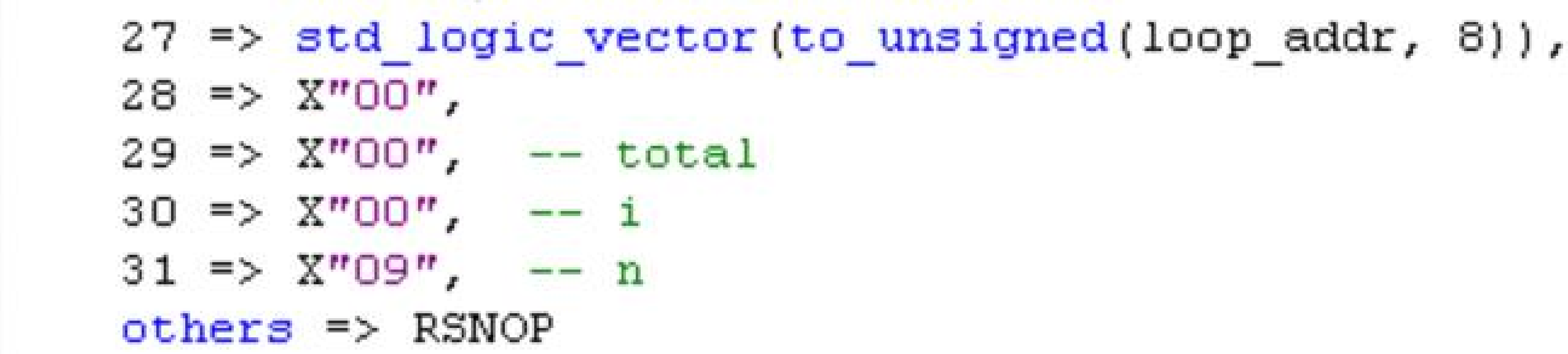
第三部分

Part One

测试结果

测试结果

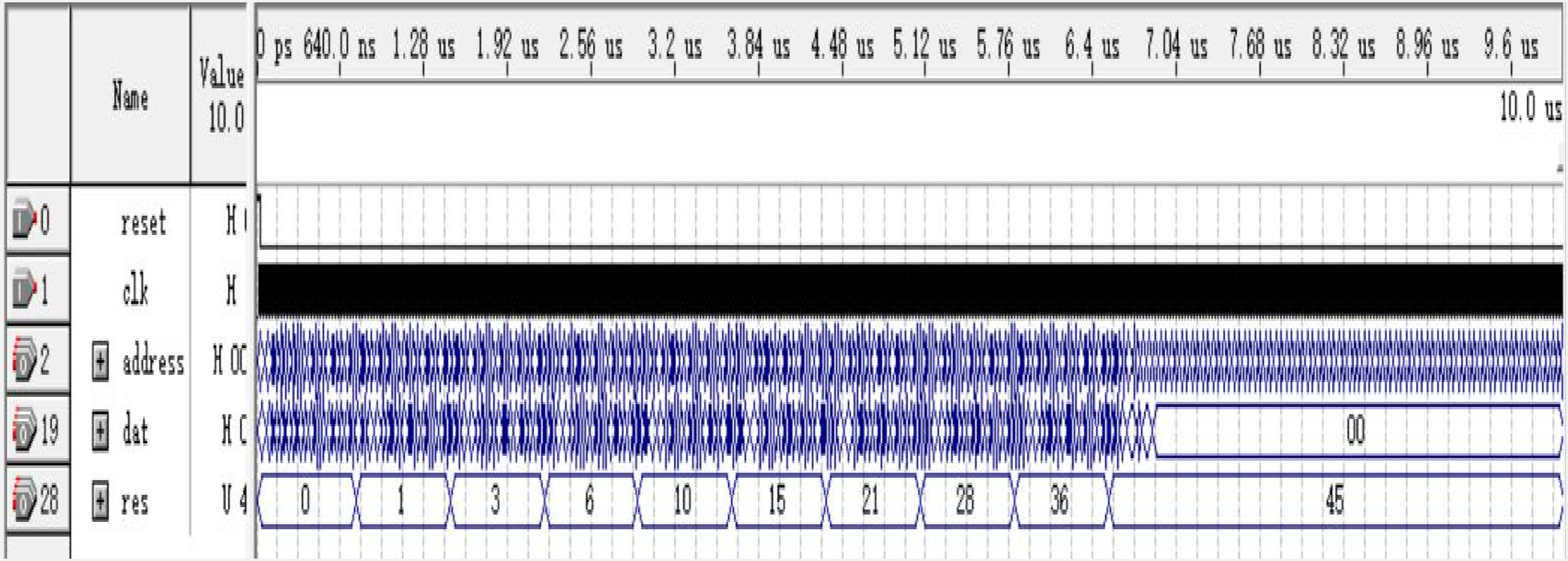
- n设为9



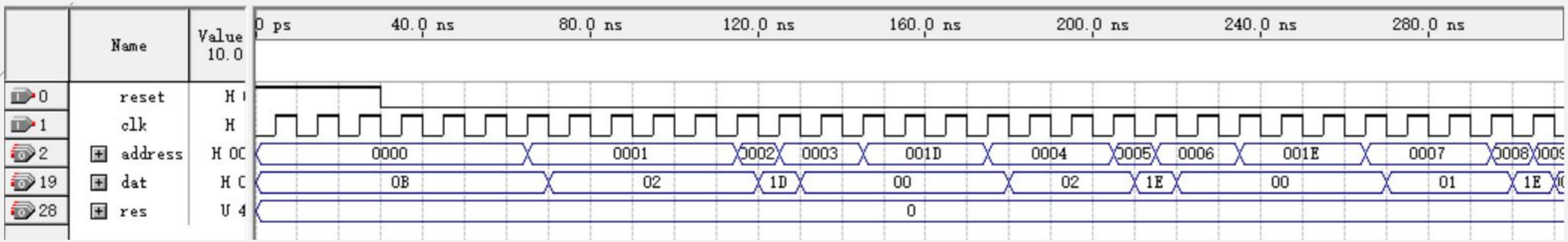
```
27 => std_logic_vector(to_unsigned(loop_addr, 8)),
28 => X"00",
29 => X"00",  -- total
30 => X"00",  -- i
31 => X"09",  -- n
others => RSNOP
```

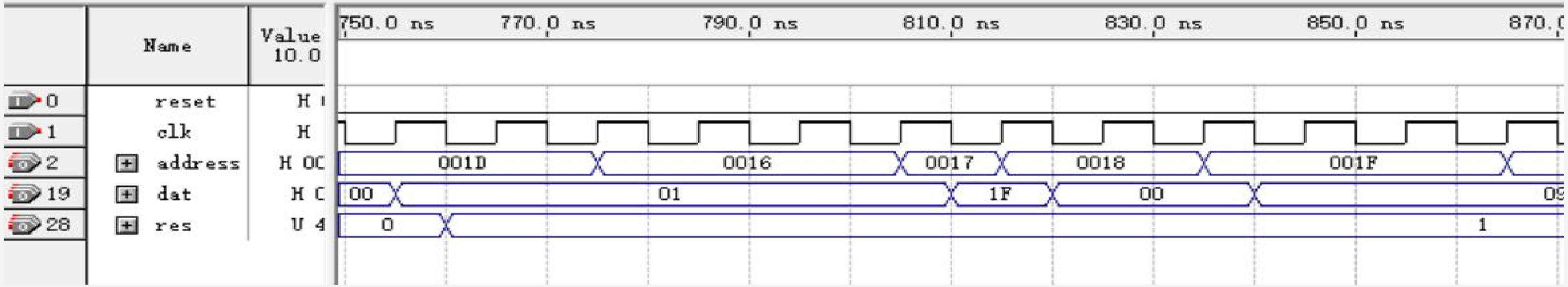
测试结果

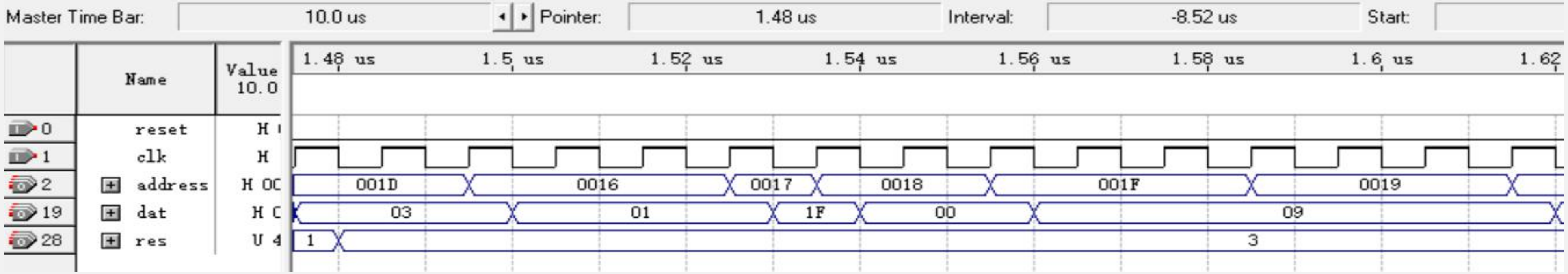
0、1、3、6、10、15、21、28、36、45之后不变实现了1~9的累加



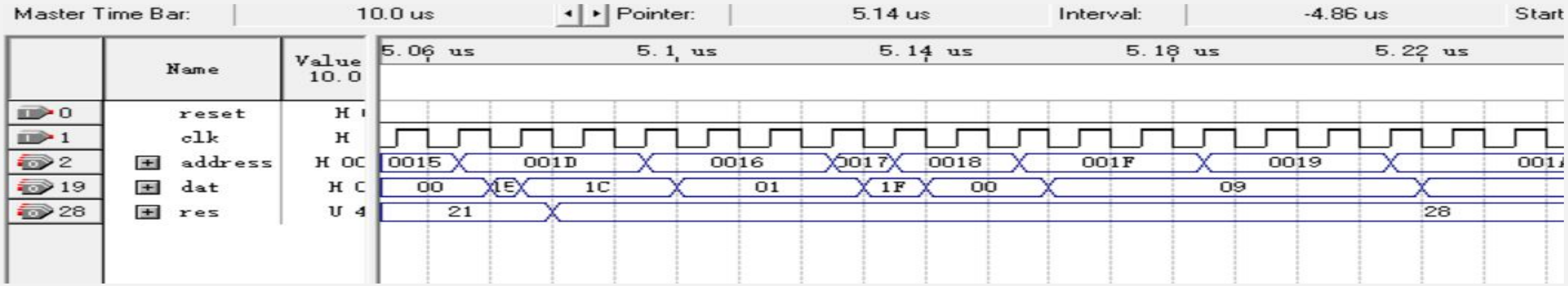
测试结果



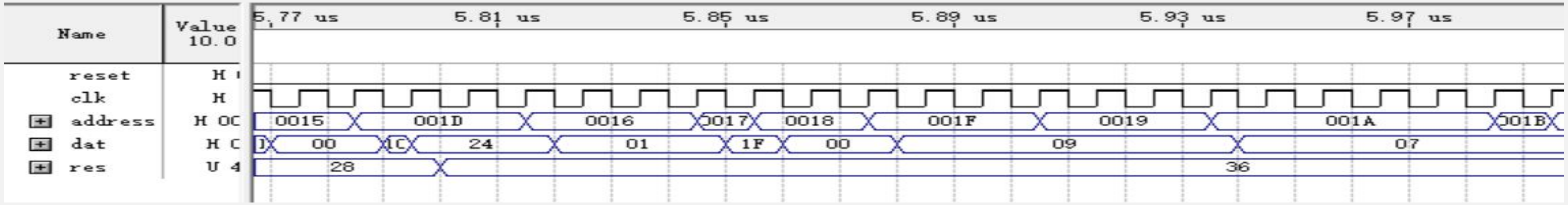


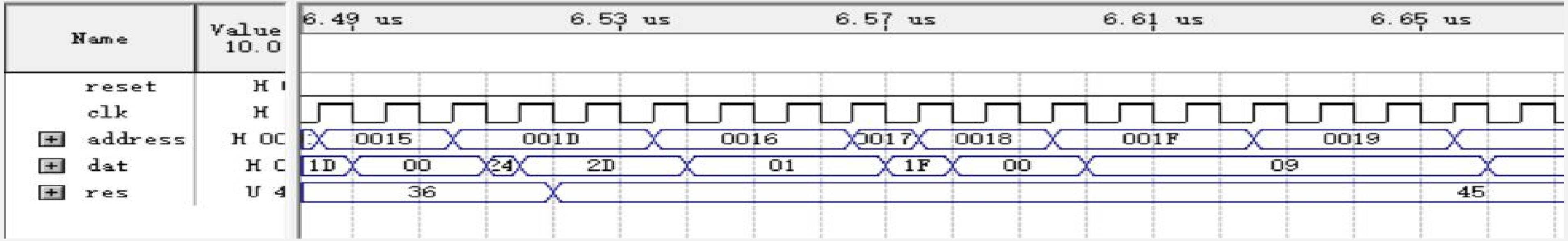

Master Time Bar: 10.0 us
Pointer: 1.48 us
Interval: -8.52 us
Start:

测试结果



Master Time Bar: 10.0 us Pointer: 5.14 us Interval: -4.86 us Start





演示完毕 谢谢欣赏