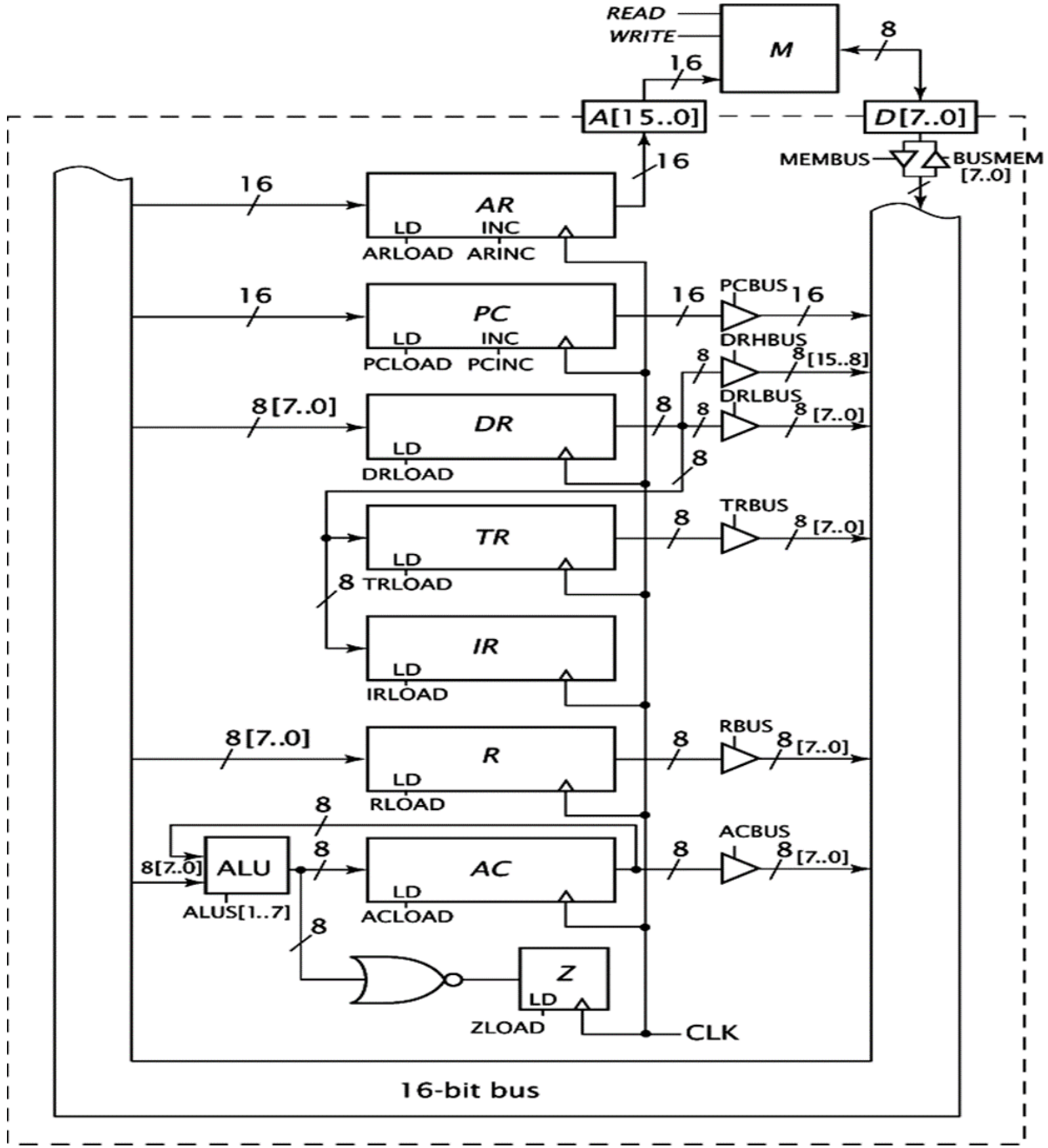
# 简易CPU设计

蒋雨  物联1601  201611020126

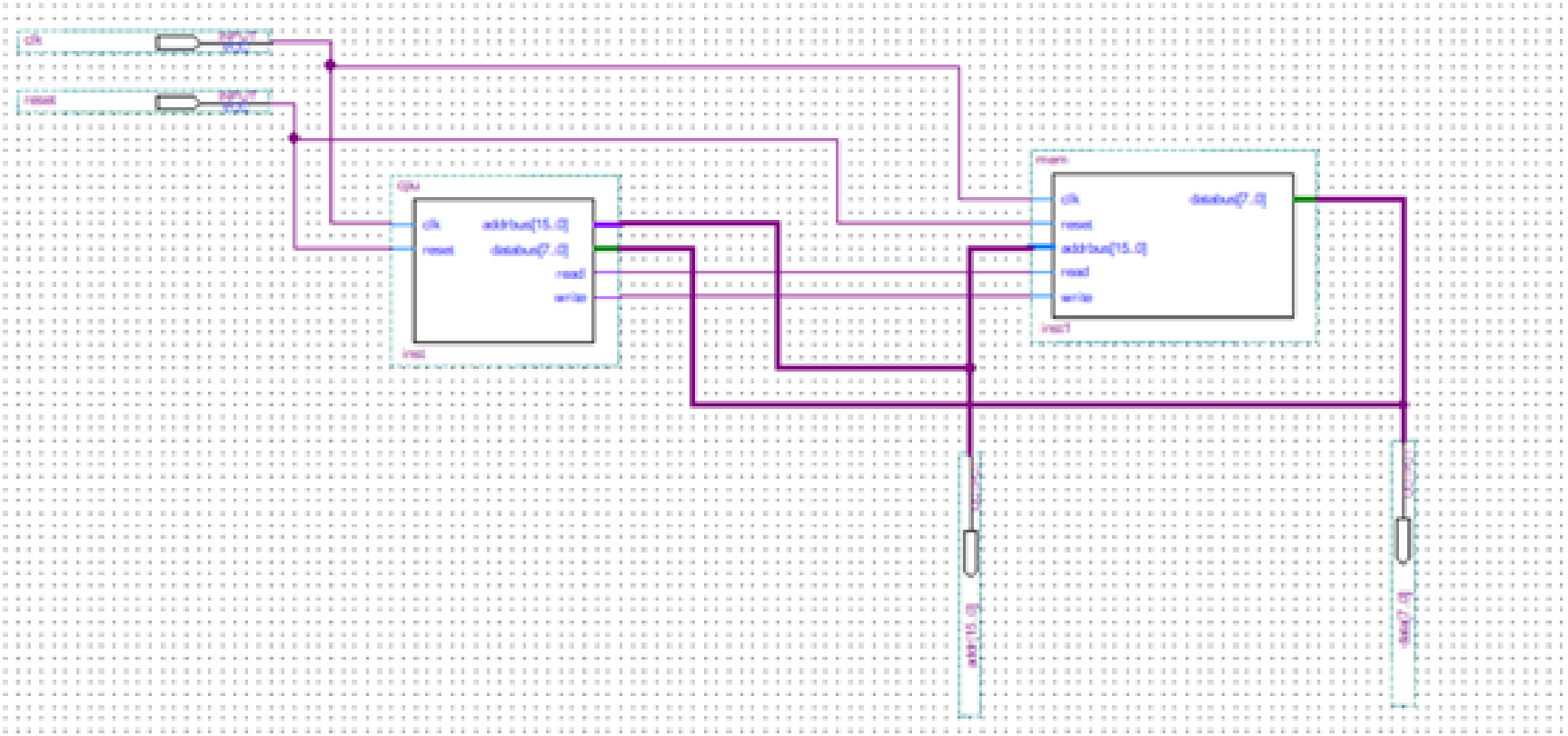
# 解决方法

直接从BUS上把数据给IR



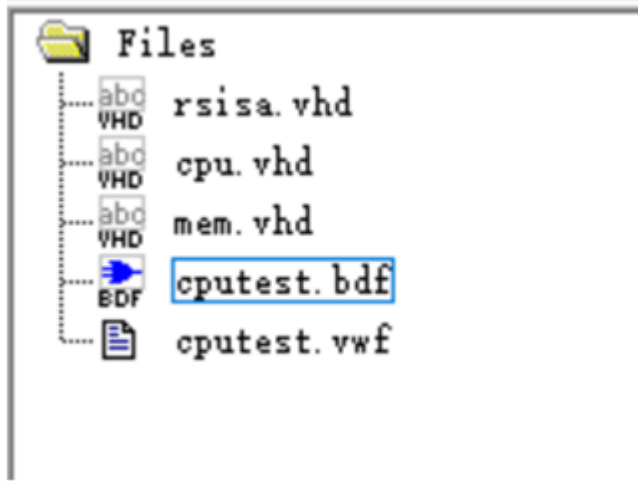
# 设计思路

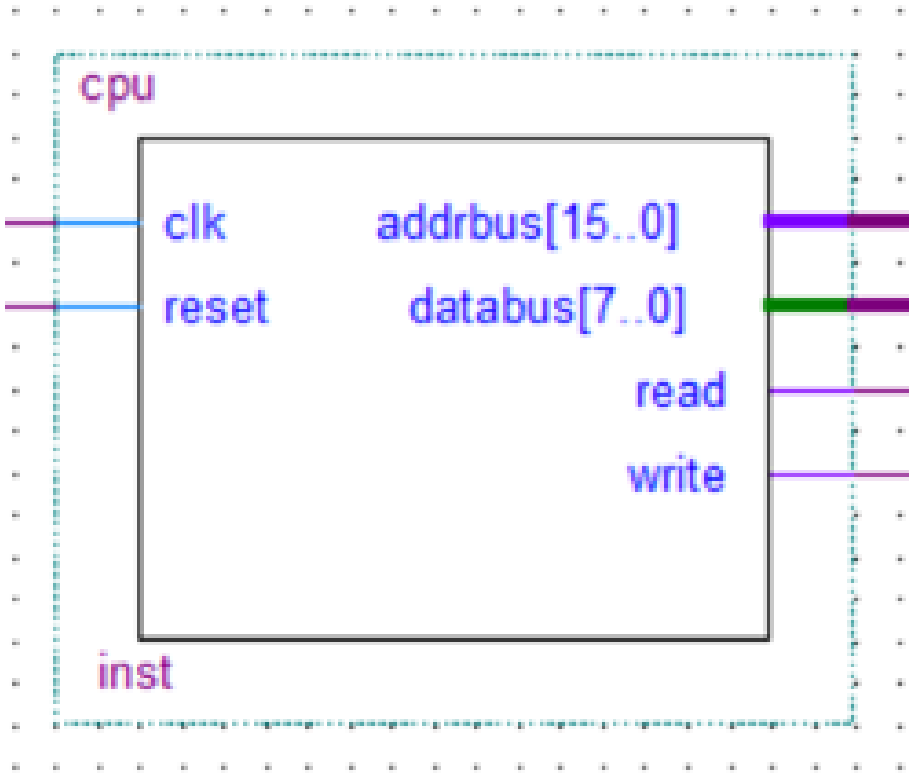
- 根据数据通路和指令的每一个状态设计在相应的部件里，把这些寄存器当作CPU内部的部件，MEM是它唯一访问的外存



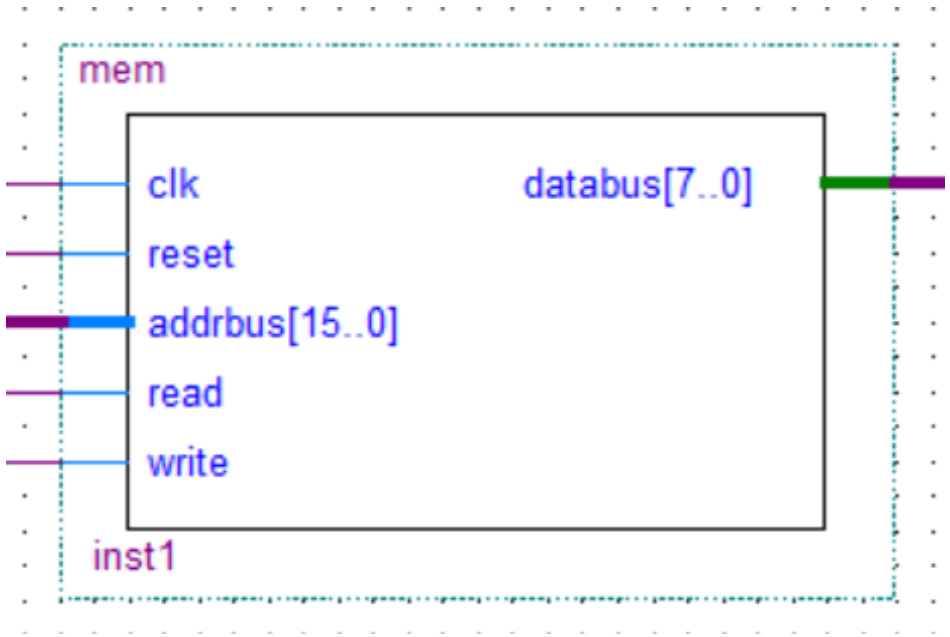
# 设计思路

- 工程由5个文件组成，而rsisa声明每条指令对应的变量名，cpu文件是完成CPU的内部组成、cpu可能达到的各个状态，和cpu处于各个状态下采取的动作。mem声明内存的大小、初始化内存，并规定read、write有效时内存相应的动作，cputest.bdf文件就是顶层文件，把cpu和mem连接起来，进行相应的操作，vwf文件是用于验证结果是否正确的





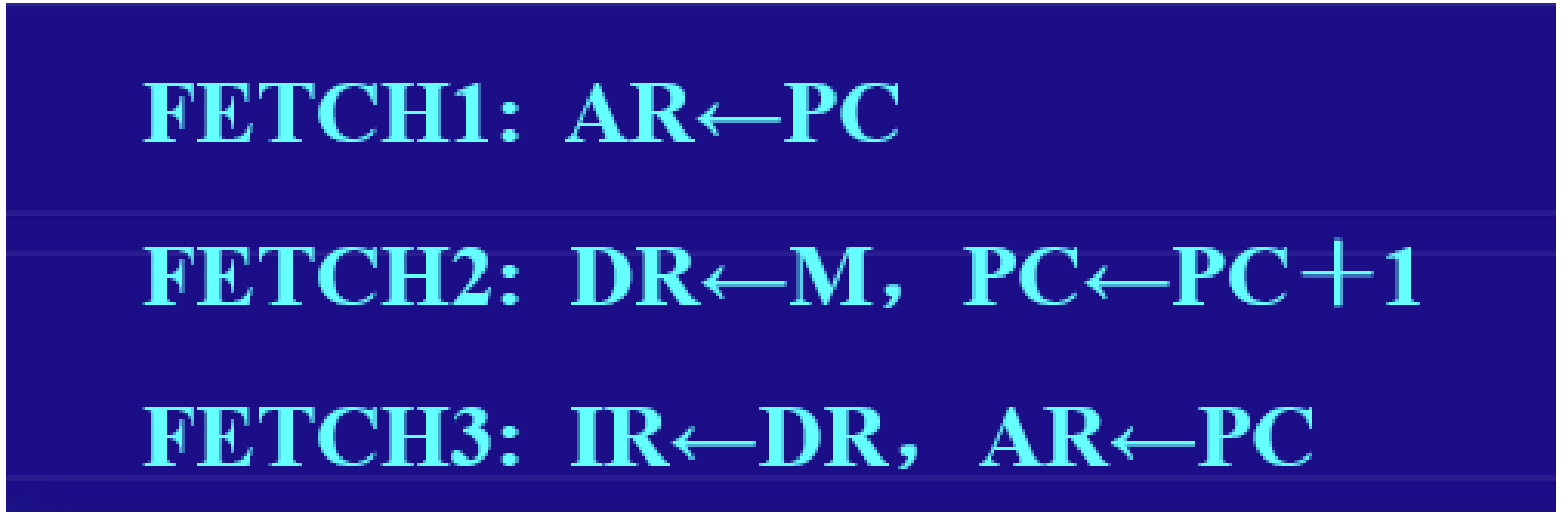

1.首先是各个寄存器，在这里就把它定义为CPU内的信号，进行相应的赋值
2.第三步是给相应的微操作赋予相应的操作码
3. CPU设计采用的是多进程的模式，下面三个进程就分别描述了时钟上升沿要进行的操作还有根据当前状态确定下一步，以及根据当前的状态使相应的控制信号有效




- 往mem里不同的地址写上相应的指令和数据，实现简单的从1加到n系统测试

# 遇到的问题:

原因会是因为这个指令周期内总线要先从DR里先读数据传到IR然后CPU再读IR的指令所以来不及了



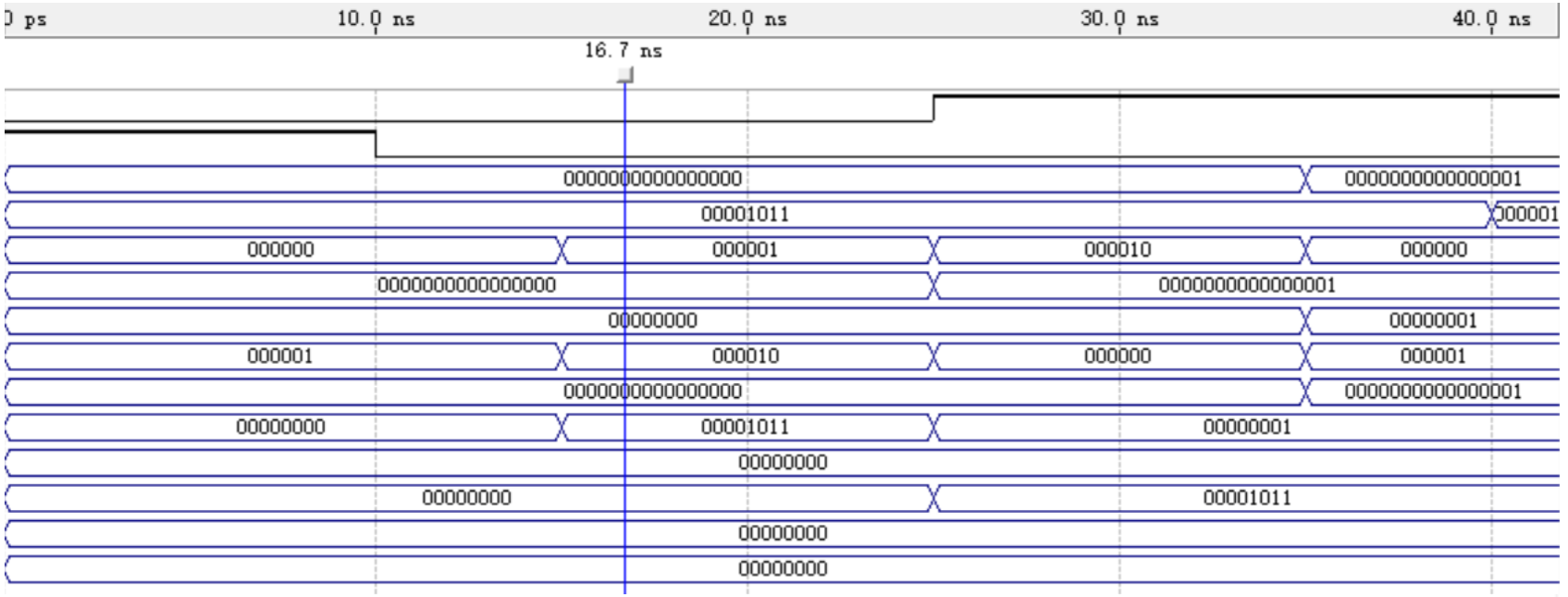
- 在fetch3的时候ir要从总线上读信息，ar也要从总线上读从pc来的信息，会不会混乱
- 最后使用了fetch4进行验证得到的也是相同的结果

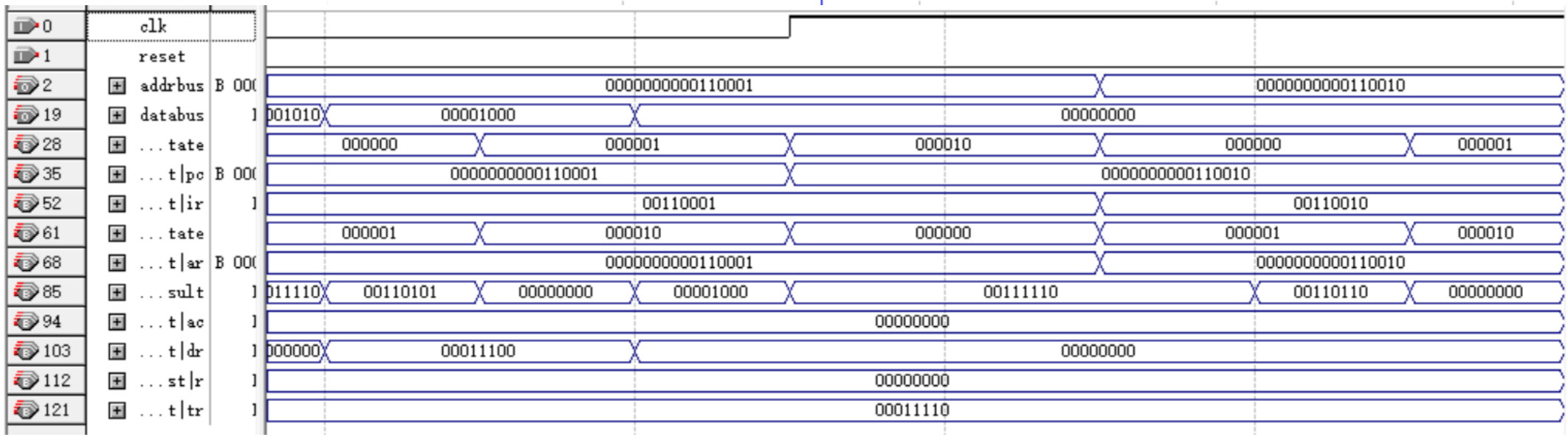
```
constant fetch1:    std_logic_vector(5 downto 0) := "000000";-- ar<=pc

constant fetch2:    std_logic_vector(5 downto 0) := "000001";-- dr<=m  pc<=pc+1

constant fetch3:    std_logic_vector(5 downto 0) := "000010";--ir<=thebus ar<=pc
```

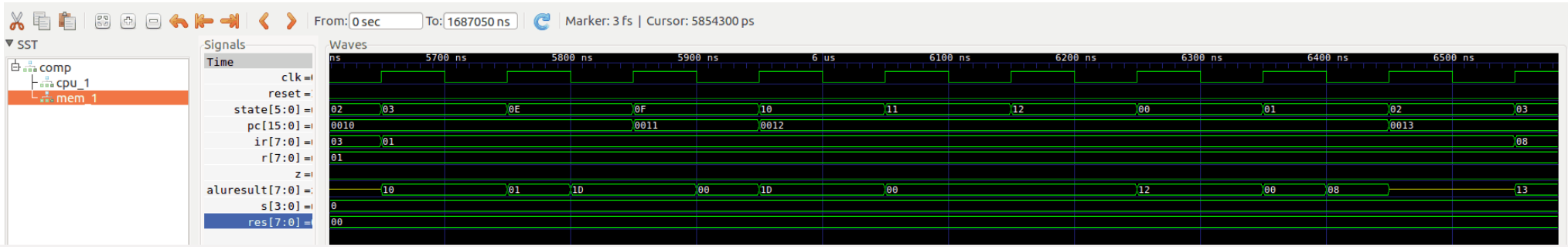
```
constant fetch3:    std_logic_vector(5 downto 0) := "000010";-- ir<=dr

constant fetch4:    std_logic_vector(5 downto 0) := "000011";-- ar<=pc
```

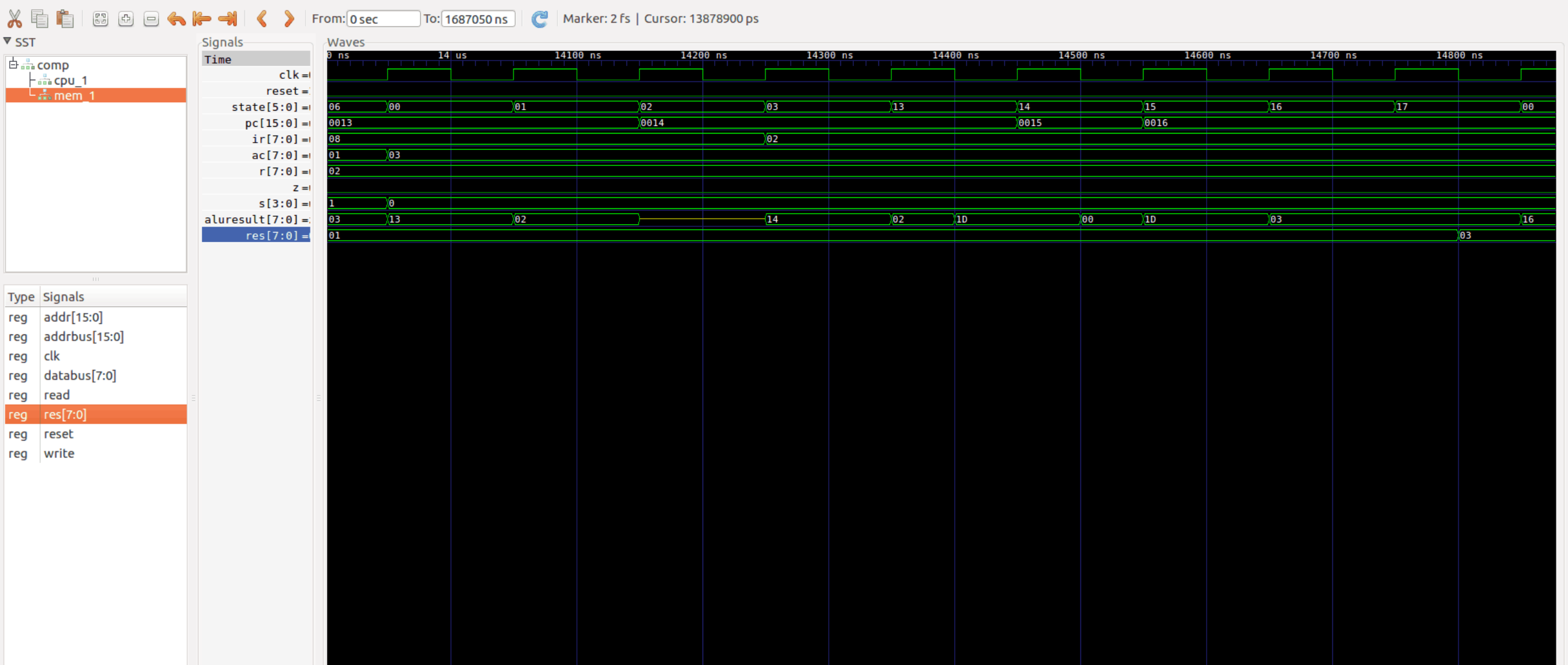
# 结果验证

我的QUARTUS的endtime不能超过1us否则就崩了，所以没办法看到计算的值，继续采用了GHDL的验证




From: 0 sec
To: 1687050 ns
Marker: 3 fs | Cursor: 5854300 ps
SST
comp
cpu_1
mem_1
Signals
Time
clk
reset
state[5:0]
pc[15:0]
ir[7:0]
r[7:0]
z
aluresult[7:0]
s[3:0]
res[7:0]
Waves
ns
5700 ns
5800 ns
5900 ns
6 us
6100 ns
6200 ns
6300 ns
6400 ns
6500 ns


From: 0 sec
To: 1687050 ns
Marker: 2 fs | Cursor: 13878900 ps
SST
comp
cpu_1
mem_1
Signals
Time
clk
reset
state[5:0]
pc[15:0]
ir[7:0]
ac[7:0]
r[7:0]
z
s[3:0]
aluresult[7:0]
res[7:0]
Waves
0 ns
14 us
14100 ns
14200 ns
14300 ns
14400 ns
14500 ns
14600 ns
14700 ns
14800 ns
Type
Signals
reg addr[15:0]
reg addrbus[15:0]
reg clk
reg databus[7:0]
reg read
reg res[7:0]
reg reset
reg write