

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机体系结构 |
| 姓 名： | 汪辉 |
| 学 院： | 计算机科学与技术学院 |
| 系： |  |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3190105609 |
| 指导教师： | 陈文智 |

2021年 9月 14日

**浙江大学实验报告**

课程名称： 计算机体系结构 实验类型： 综合

实验项目名称： **Pipelined CPU supporting RISC-V** **RV32I Instructions**

学生姓名： 汪辉 专业： 计算机科学与技术 学号： 3190105609

同组学生姓名： 王嘉豪 指导老师： 陈文智

实验地点： 曹西301 实验日期： 2021 年 9 月 14 日

1. 实验目的和要求

* **Understand RISC-V RV32I instructions**
* **Master the design methods of pipelined CPU executing RV32I instructions**
* **Master the method of Pipeline Forwarding Detection and bypass unit design**
* **Master the methods of 1-cycle stall of Predict-not-taken branch design**
* **master methods of program verification of Pipelined CPU executing RV32I instructions**

1. 实验内容和原理

* **Design of Pipelined CPU executing RV32I instructions.**
  + Design datapath
  + Design Bypass Unit
  + Design CPU Controller
* **Verify the Pipelined CPU with program and observe the execution of program**

1. 实验过程和数据记录
2. 完善控制单元信号处理

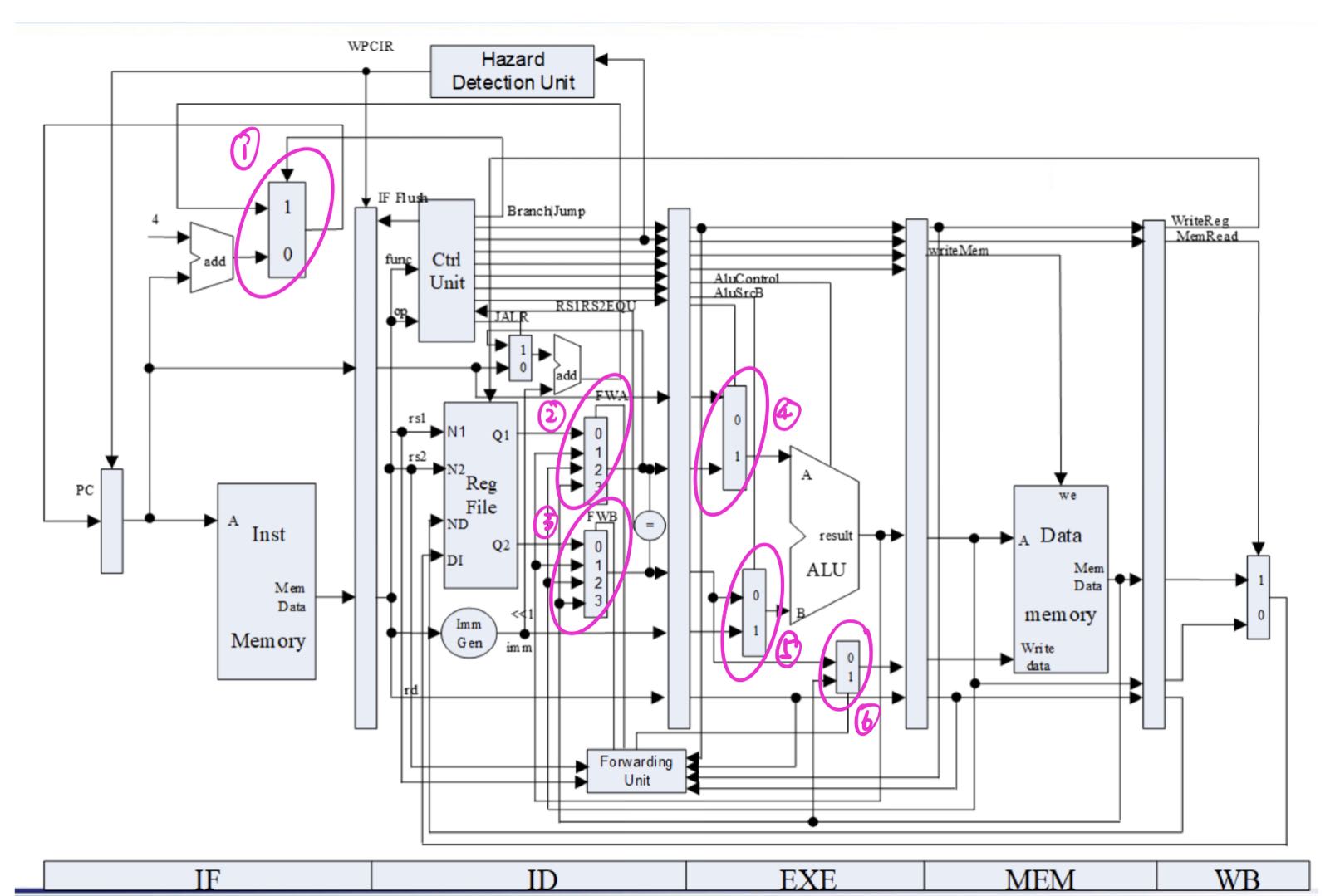
根据各个类型指令的格式完善控制单元内的信号，单元内已定义各个opcode的值，根据opcode和function code完成译码的内容，得到各条指令的信号。例如：

wire BEQ = Bop & funct3\_0; //指令BEQ

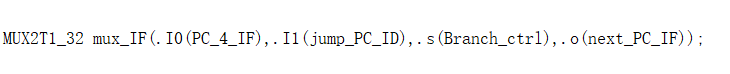
assign rs1use = JALR | B\_valid | I\_valid | S\_valid |L\_valid |R\_valid; //信号rs1use，用于实现HazardDetectionUnit的检测。

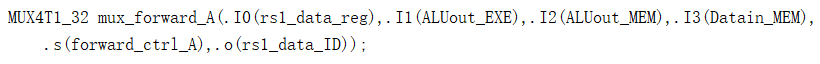
1. 完善core模块各个接口

根据指导中的流水线设计图完成RV32core模块的各个MUX接口。

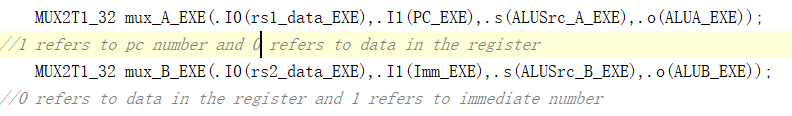


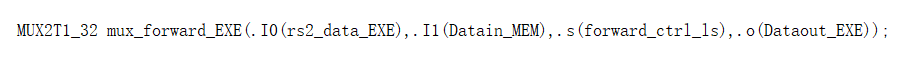
总共需要完善6个MUX的信号：





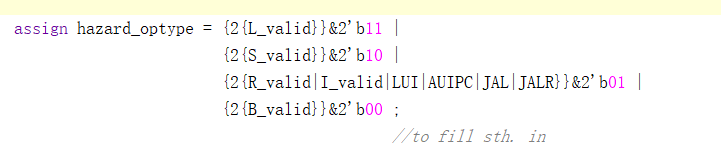




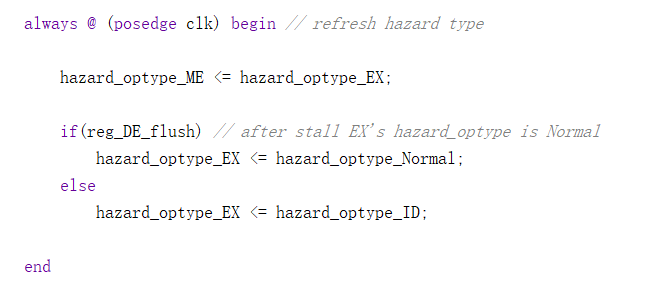


1. 设计完成HazardDetectionUnit模块

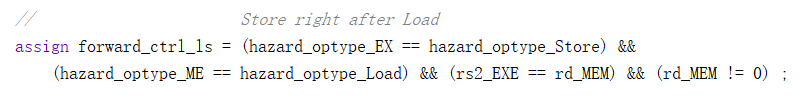
本实验中为了方便对Hazard的处理，将Hazard分为LoadStore、LoadRead和ReadRead三种类型。在CtrlUnit中处理了一个hazard\_optype信号，该信号传递了当前指令的类型，如下：



利用三个寄存器，每周期更新ID、EX、ME三个阶段的hazard\_optype。这样就可以根据指令类型来判断hazard的存在和类型。

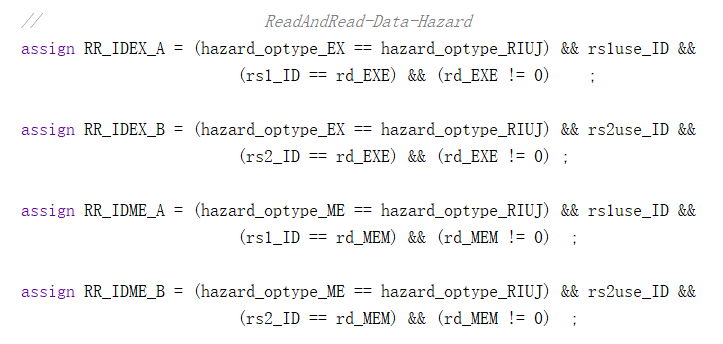


LoadStore类型的处理相对简单，根据第二步接口部分内容第6个Mux的信号，在该类型hazard存在时置信号forward\_ctrl\_ls为1，即直接取内存取出的结果传递给store指令的操作数。

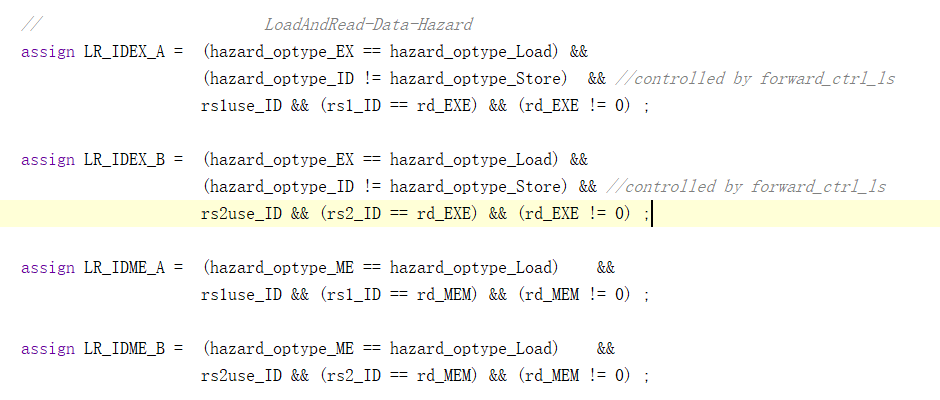


LoadStore类型的hazard只对store的源寄存器的操作数产生影响，ReadRead和LoadRead对两个操作数都可能产生影响，因此需要对rs1和rs2两个源操作数都做处理。

ReadRead类型又分为ID、EX的冲突和ID、ME的冲突，具体信号处理：



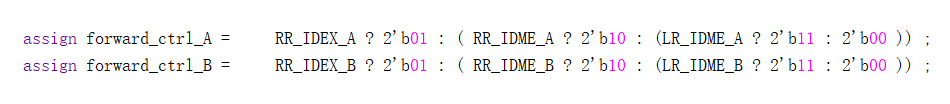
LoadRead的处理原理类似：



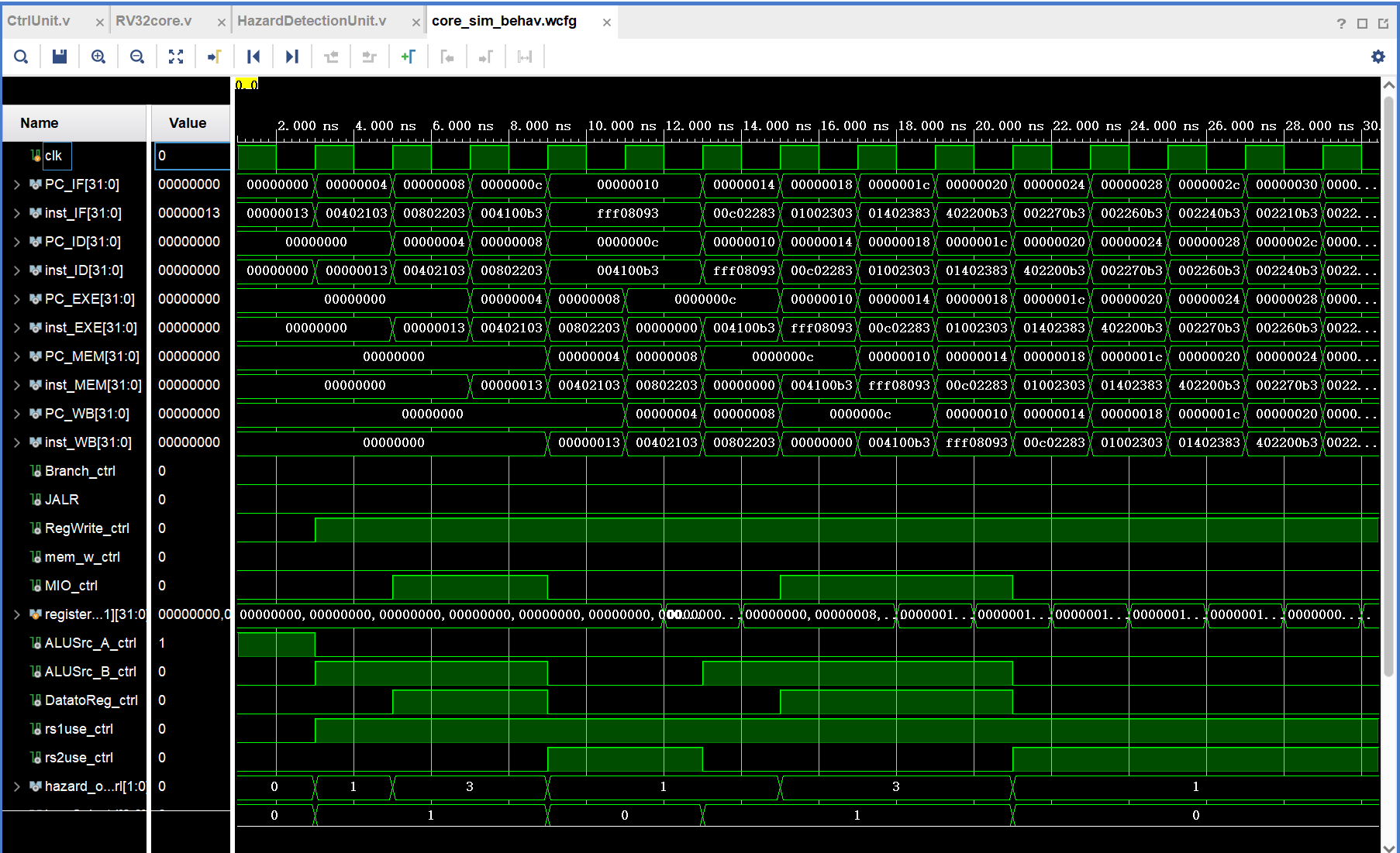
LoadRead有一种特殊情形，当译码阶段取出一条Read类型（RIUJ）的指令例如addi x1，x10，100而前一条指令为lw x10，8（x0）时，由于Load的结果直到ME阶段才得到而addi的结果在EX阶段就已经计算出，为了得到正确的结果不得不插入一条stall：

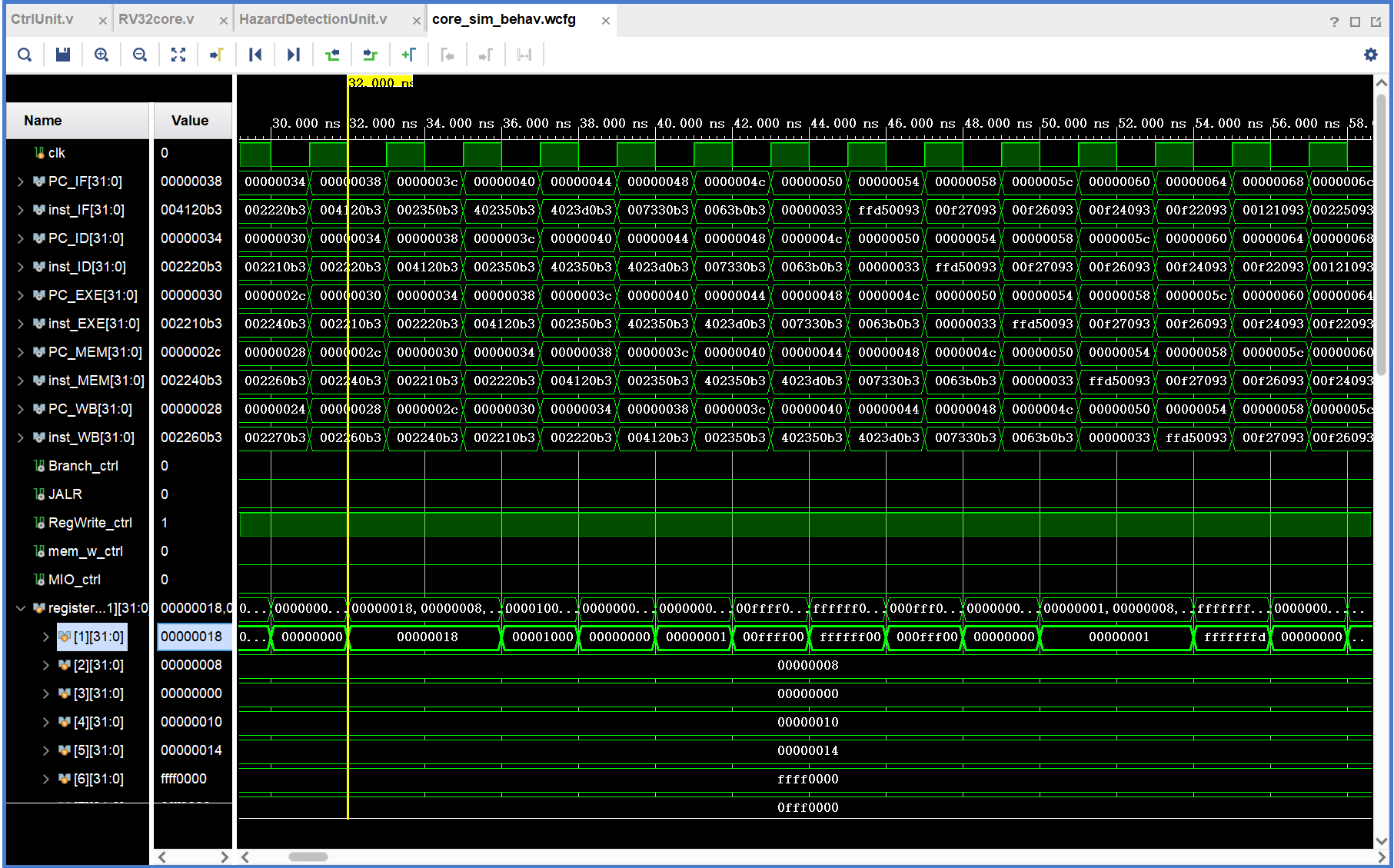


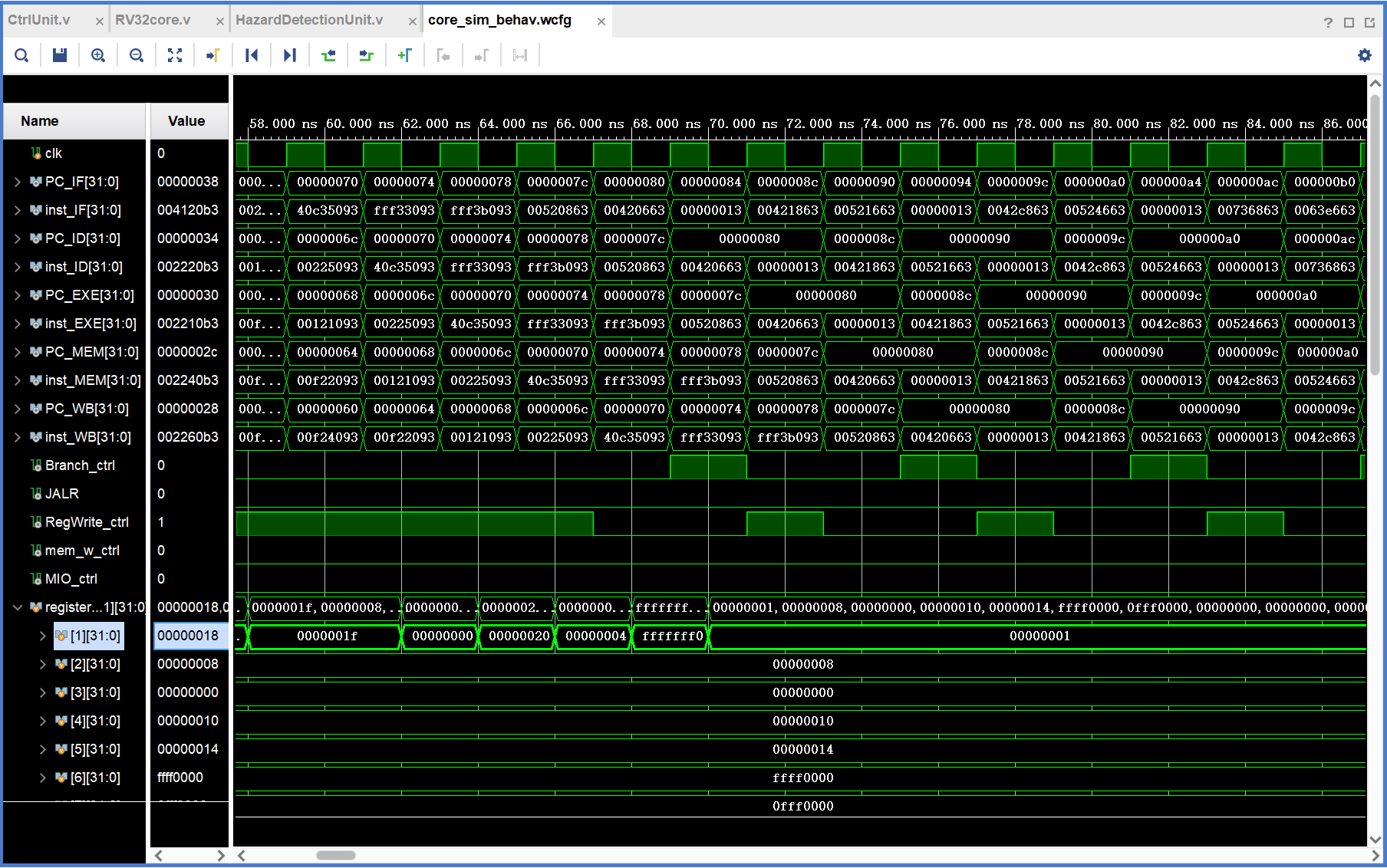
在其余的情况下，则根据ID阶段的2个4to1Mux选择rs1和rs2的正确内容，当ReadRead的hazard为ID、EX的类型时，ctrl信号置为1，选择当前ALU的结果为rs的数据；而为ID、ME类型时，则ctrl信号置为2，选择上一个周期ALU的结算结果；而当Hazard的情况是LoadRead的ID、ME类型时，ctrl信号置为3，选择内存取得的内容为rs的数据。

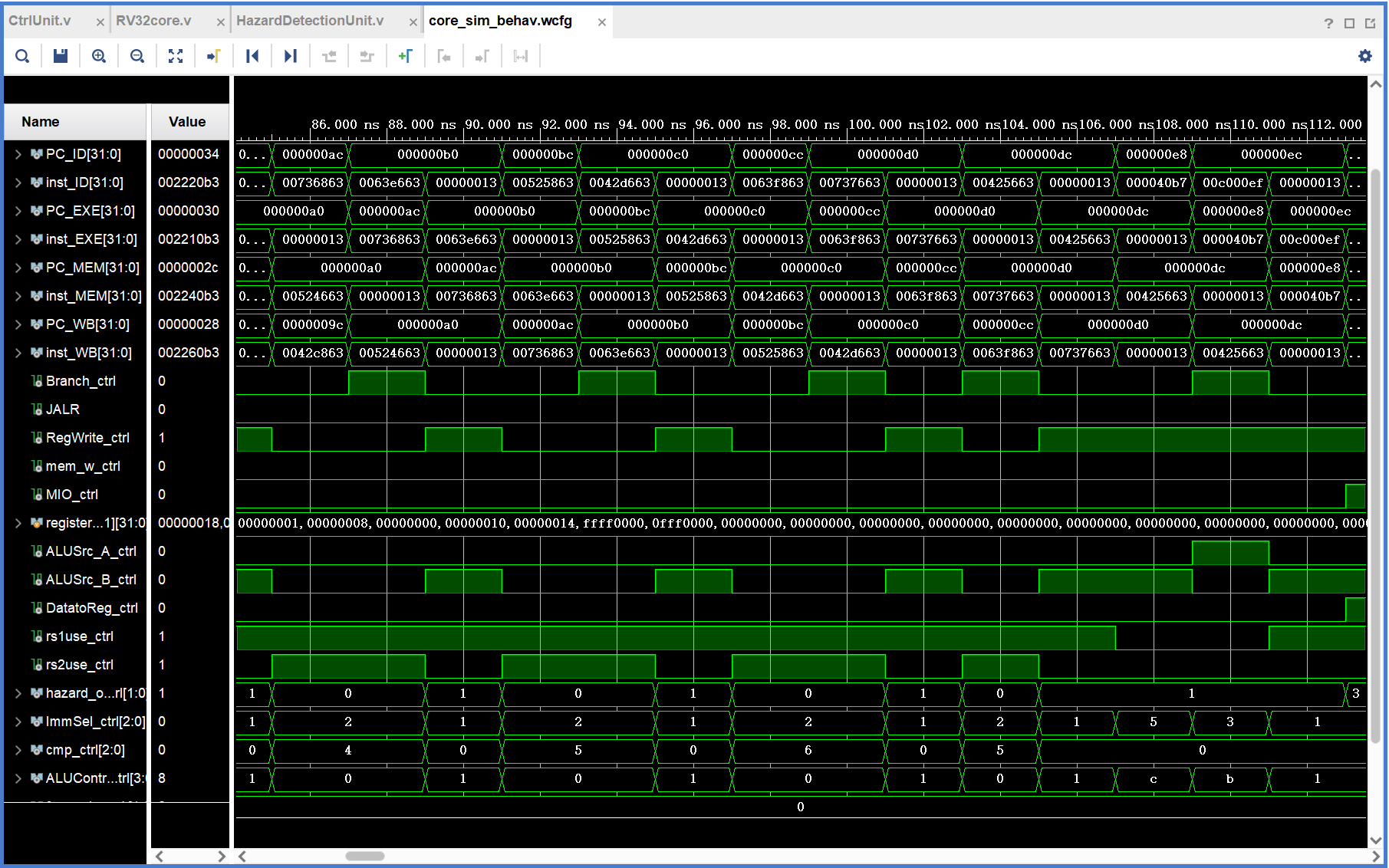


1. 实验结果分析
2. 仿真波形

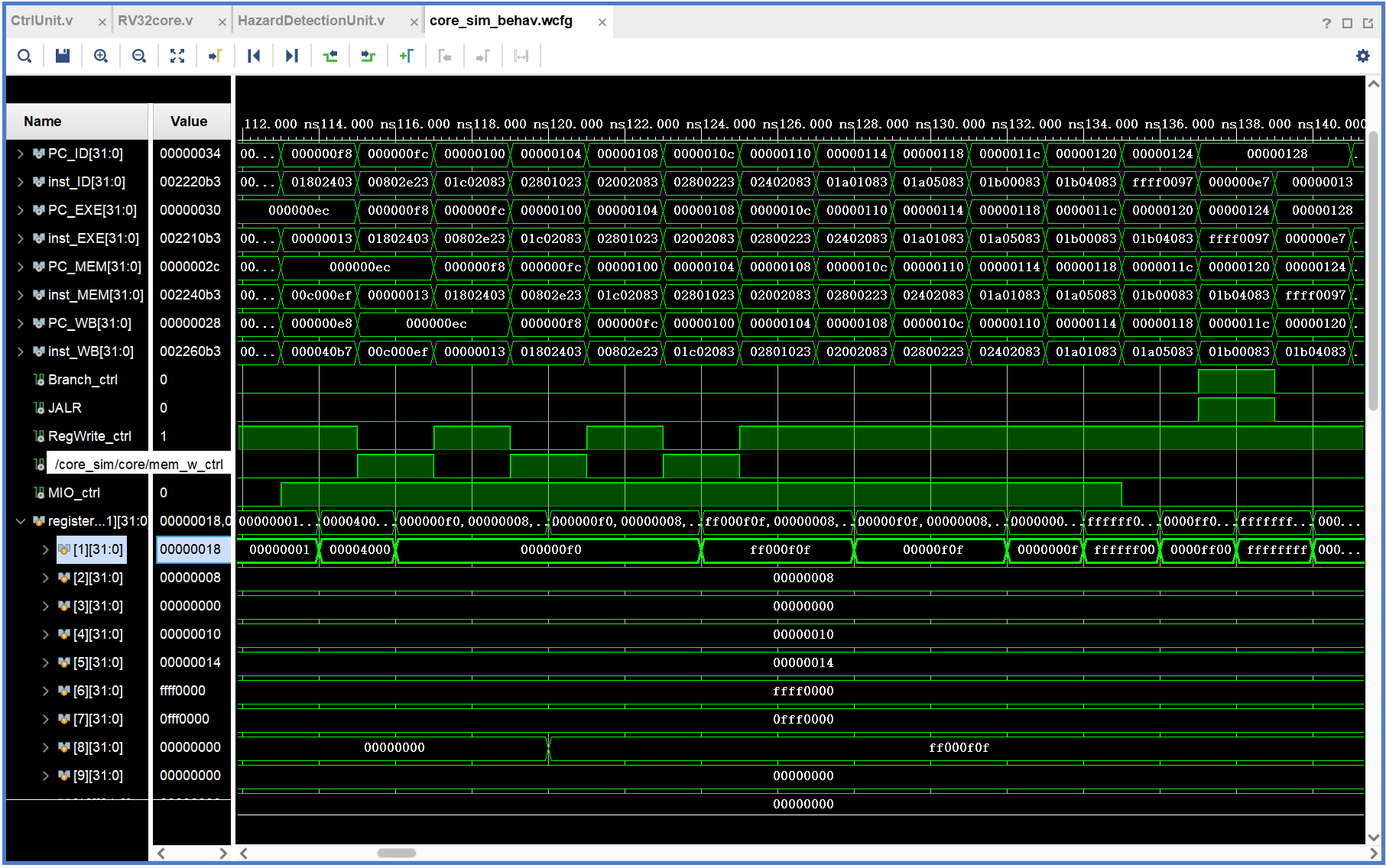




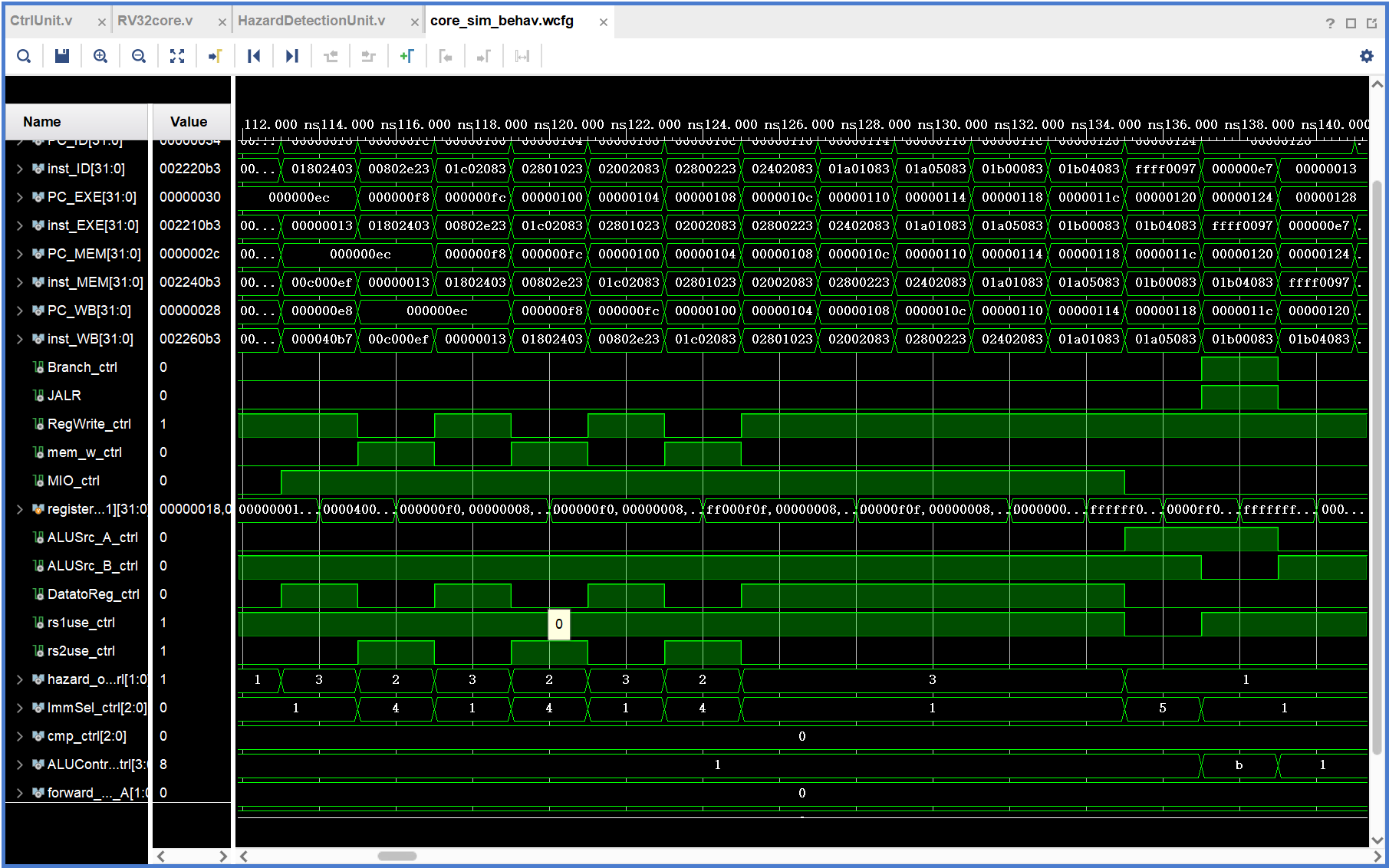




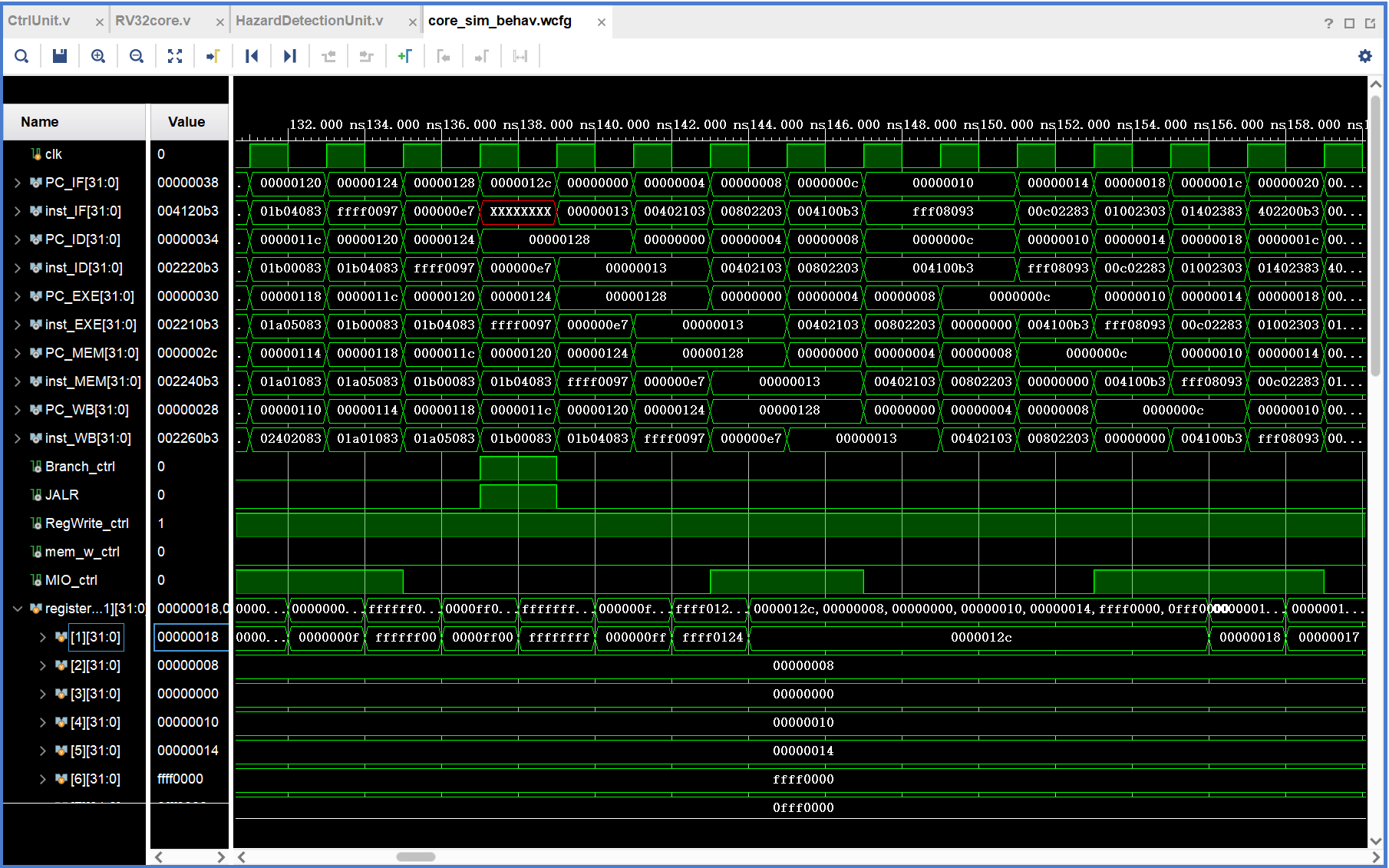
最后部分load指令的x1结果：



该部分的控制信号：



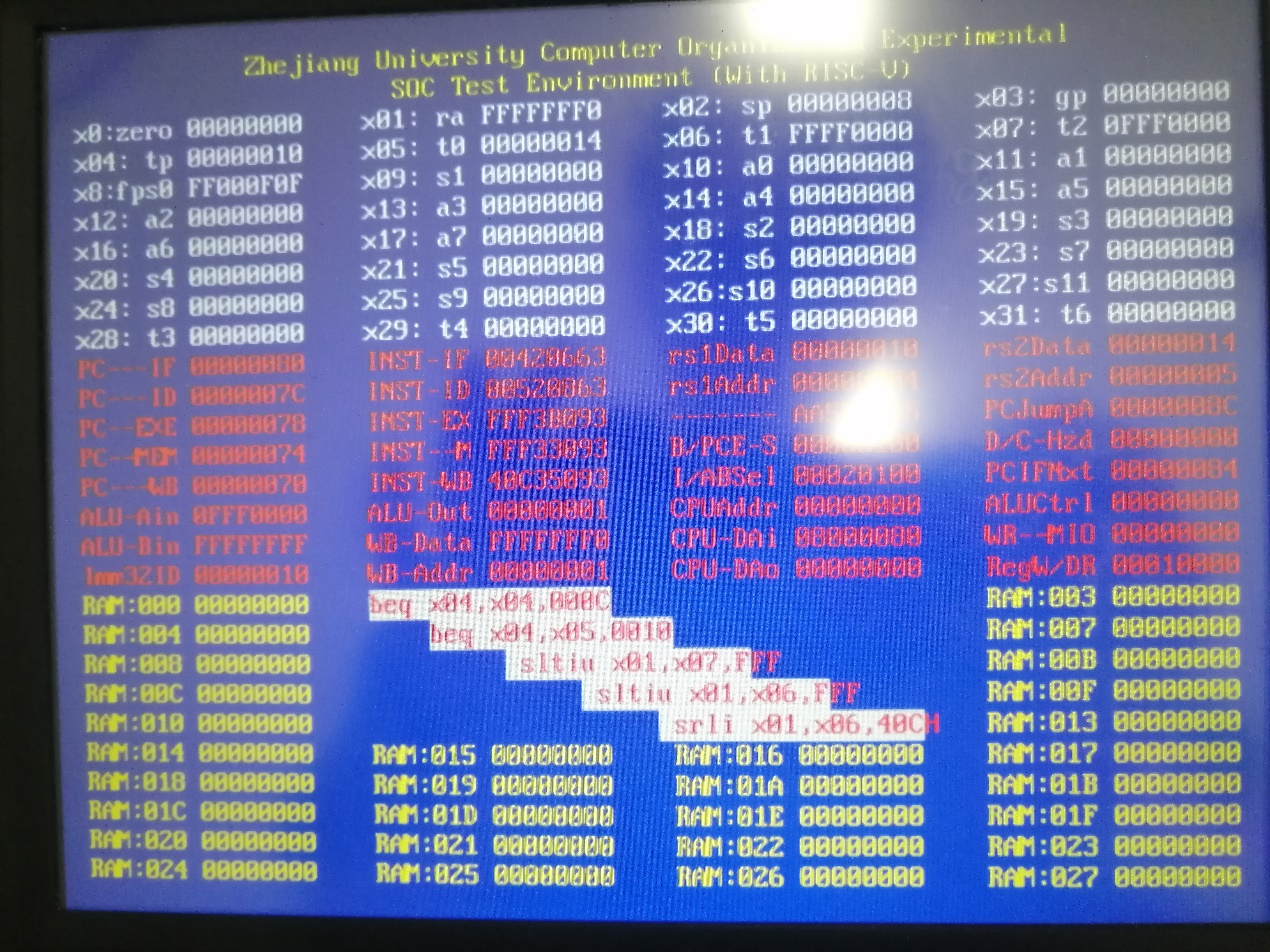
最后部分的跳转，x1跳转到012c：

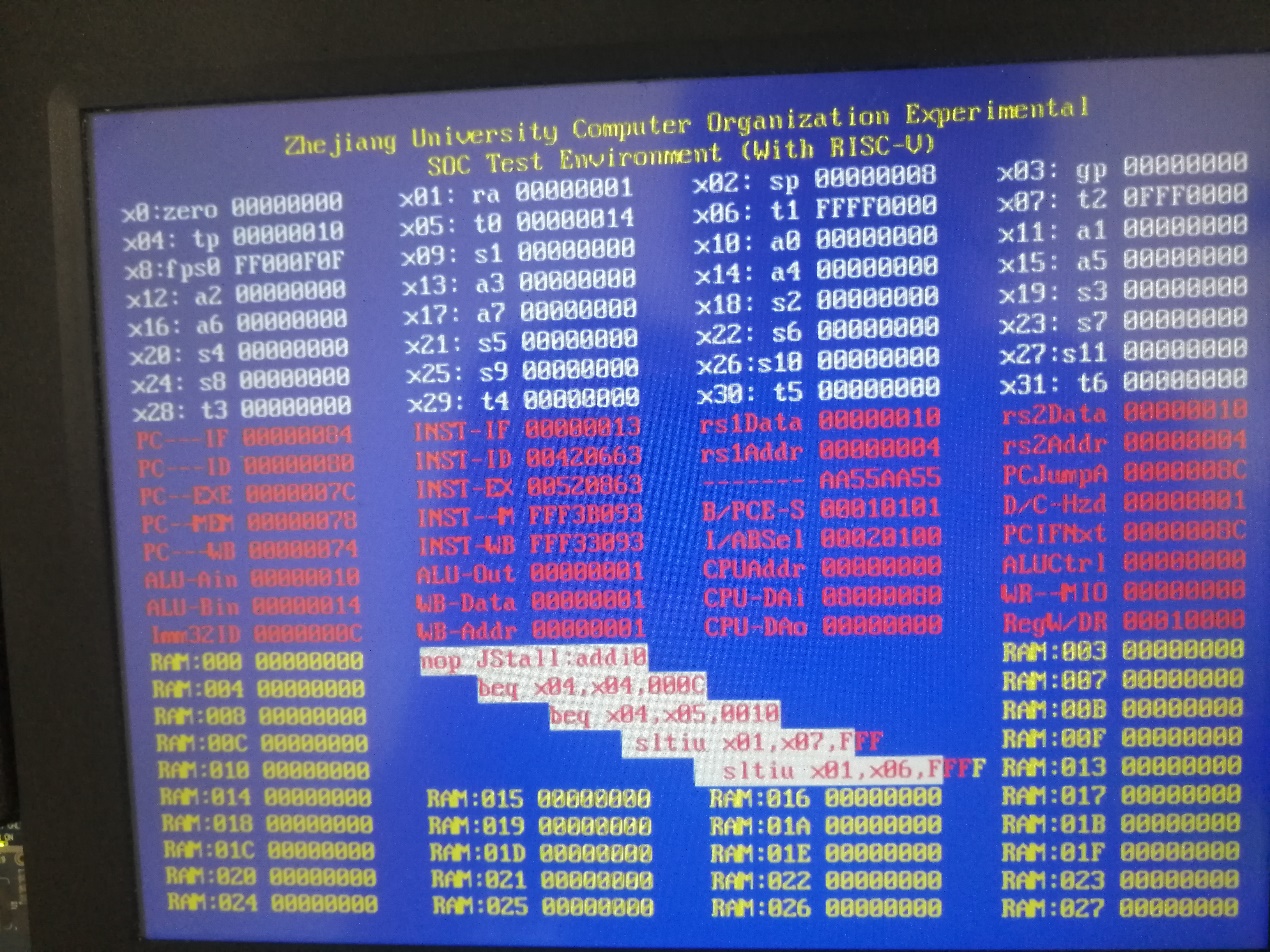


1. 实验箱记录

一些实验照片：







1. 讨论与心得

为了实现跳转的指令，ctrlunit的branch信号对jal、jalr和B型指令和比较结果取或，本实验中对跳转指令（b type）的比较处理都在ID阶段完成，若比较结果为跳转，则会跳跃指令并且插入jstall来中断之前的取指并继续执行，这一点在实验箱调试中可见。