**计 算 机 体 系 结 构（研 讨 课）实 验 报 告**

实验项目 prj5 小组编号 28 组员姓名 刘景平、张钰堃、付博宇

1. **实验目标**

在exp13实现的CPU基础上完成：

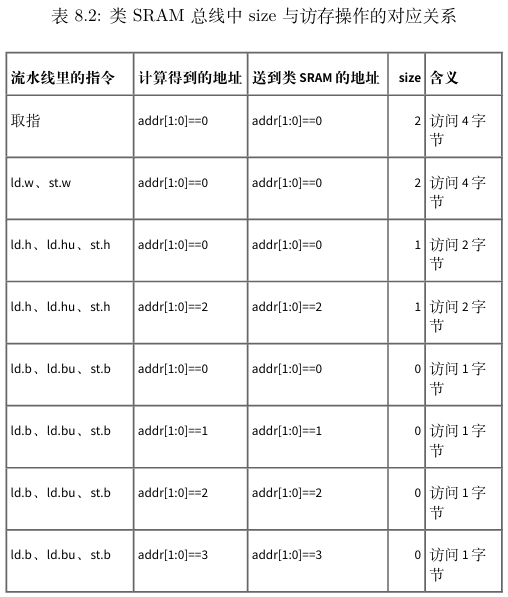
1. 将CPU对外接口修改为类SRAM总线接口。
2. 在采用握手机制的blockRAM的SoC验证环境中完成exp14对应func的随机延迟功能验证。



由讲义可知，与原有的SRAM接口信号相比，req对应en，wr对应|wen，wstrb对应wen，

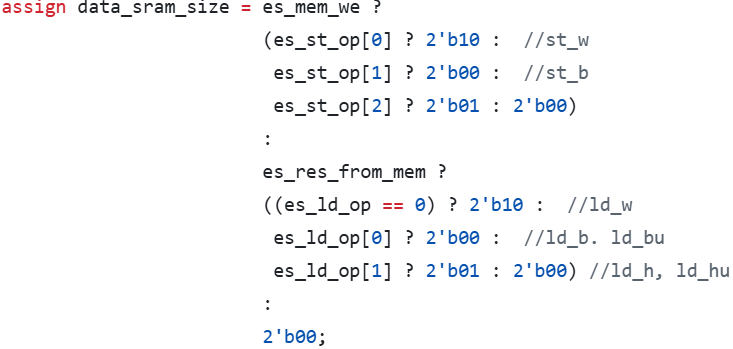
最主要的变化是新增了三个信号size、addr\_ok、data\_ok。

1. **逻辑电路结构与仿真波形相关说明**
2. size信号



根据讲义中表8.2的描述，根据请求的属性直接生成，代码如下：

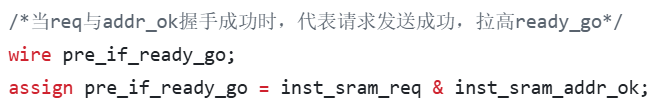




（二）IF阶段调整

1.考虑ready\_go

·pre-IF阶段：目前cpu设计令取指请求由preIF发出，指令返回在IF完成。不同于之前preIF发出的请求总是能被接受，ready\_go可以恒为1；从类SRAM总线反馈回来的addr\_ok不时刻为1，会有取指地址请求没有被接收的情况发生，所以我们需要对ready\_go进行修改。仅当req & addr\_ok 为1的时候ready\_go置1.



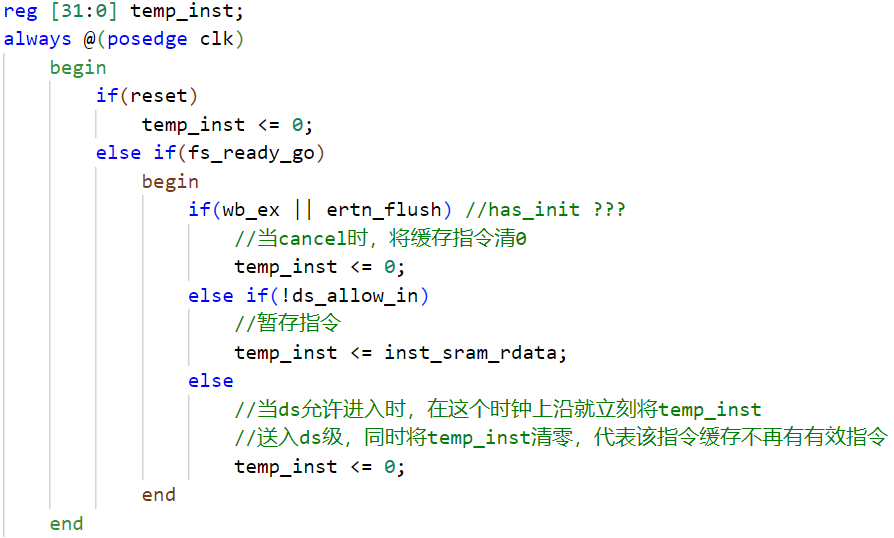
·IF阶段：原本ready\_go可以恒为1。引入类SRAM总线后，只有data\_ok返回1的时候指令码才真正出现在接口上，只在此时IF的ready\_go可以置1.



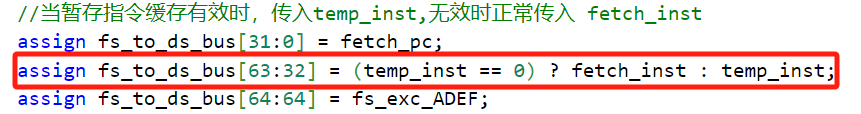
2.考虑allowin

·pre-IF与IF：对于可能出现的pre\_if\_ready\_go = 1， fs\_allowin = 0的情况选择简单的解决方案，仅当IF级allowin为1时pre-IF 级才可以发出地址请求

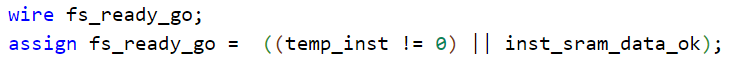
·IF与ID：对于fs\_ready\_go = 1，ds\_allowin = 0的情况，因为不能像pre-IF一样避免，我们采用教材推荐的临时缓存方案，设置一组触发器来暂存（temporary）IF级取回的指令：



即在fs\_ready\_go为1的情况下，若ds\_allow\_in为0，使用该触发器暂存读到的指令inst\_sram\_rdata；一旦ds允许进入，在这个时钟上升沿立刻将暂存的temp\_inst送到ds级同时将该触发器清零，表示该指令缓存不再有有效指令；当该组触发器有有效数据时选择该组触发器保存的数据作为fs取回的指令送往ds：

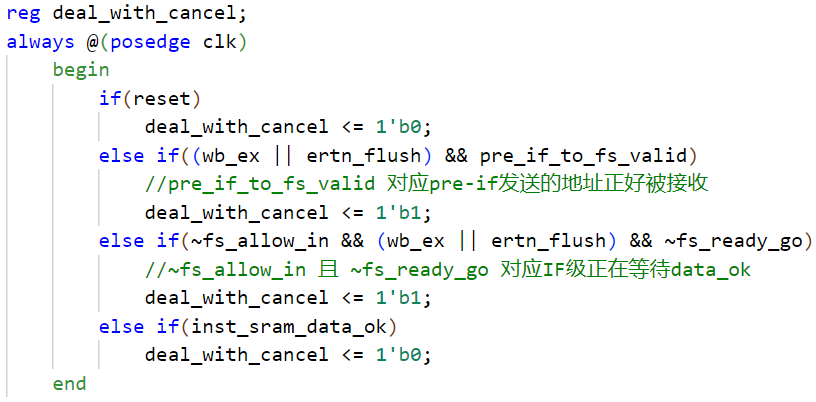


由于新增了temp\_inst之前的ready\_go需要进行修改，判断临时存指令的缓存是否存在有效指令，若存在，ready\_go需要置1，修改如下：



1. 考虑异常清空流水线

在Cancel后，IF级后续收到的第一个返回的指令数据是对当前被Cancel的取指请求的返回。所以后续收到的第一个返回的指令数据需要被丢弃，不能让其流向ID级。讲义中推荐了两种解决方法，这里采用第二种——新增一个触发器：

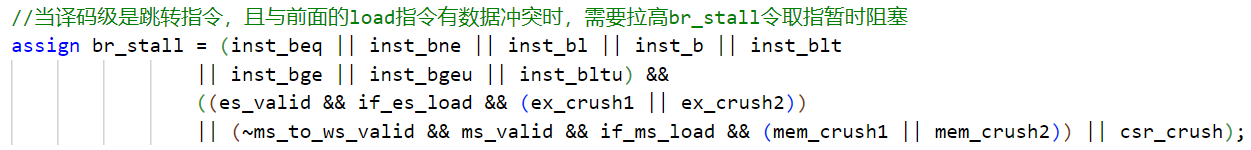


同时修改fs\_ready\_go，即当该触发器为1时，fs\_reday\_go需要抹零。



1. 考虑转移计算未完成的情况

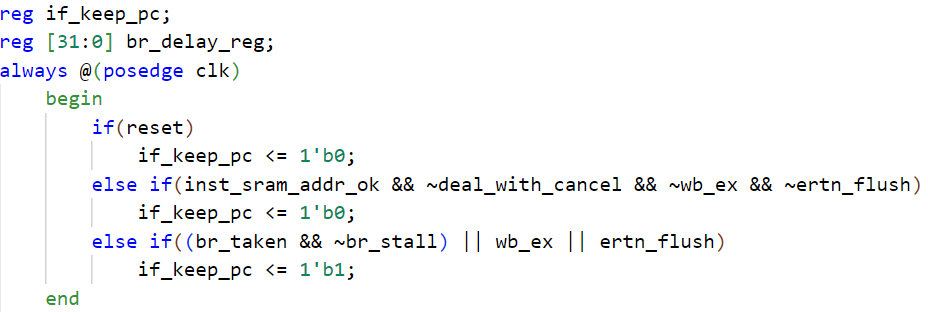
根据讲义提供的思路，在阻塞取指的设计基础上，ds送到fs级的br\_bus可以新增一个控制信号br\_stall,当ds上存在转移指令且处于计算未完成状态时，将br\_stall置1；pre-IF级看到br\_stall为1时将暂停发出取指请求知道br\_stall为0，代码如下：





1. 异常/跳转

当出现异常入口pc、异常返回pc和跳转pc时，信号和pc可能只能维持一拍，但在req收到addr\_ok前需要维持取址地址不变，新增触发器if\_keep\_pc

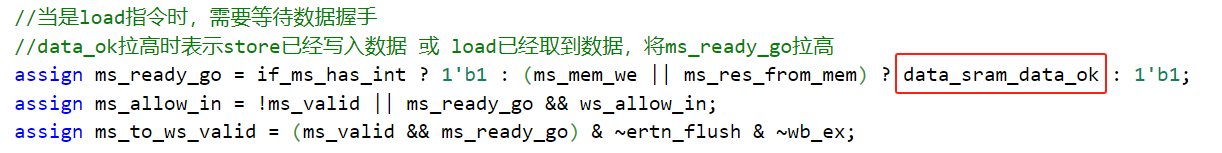




（三）访存设计的考虑

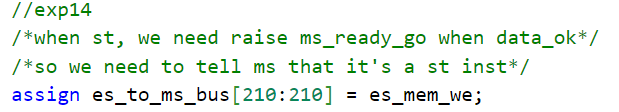
·load指令

根据讲义所述，调整ready\_go，即当data\_ok拉高时表示st已经写入/ld已取到数据，这时将ms\_reday\_go拉高，允许进入下一级流水，修改如下：

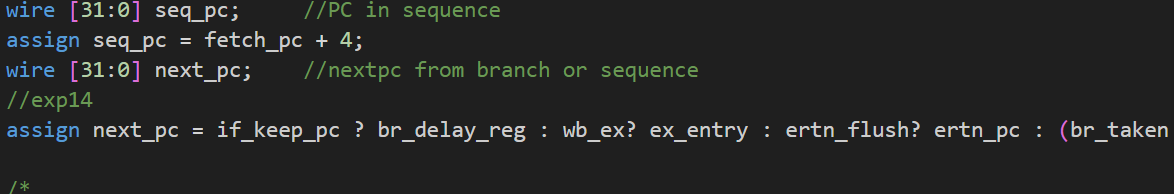


·store指令

由于在data\_ok为1时，我们要拉高ready\_go,我们要在ms时知道是否是st指令，所以对es\_to\_ms\_bus新增es\_mem\_we信号



1. **实验过程中遇到的问题以及debug心得**
2. 需要注意中断（int）和异常（wb）之间的区别，异常是指令本身有误，是程序内在因素导致的；而中断则是来自外部，是外部临时插入的特殊信号。在处理上两者有差别，异常由ID电路自行进行判断，而中断则是直接向ID模块发出一个has\_int信号。所以has\_int信号不应该出现在其他模块的的输入信号中，否则会造成同时有两条指令接收到同一个中断的情况，造成紊乱。在exp13的实验中，我们没有注意到这一点，在IF模块的nextpc判断中加入了一个has\_int，导致连续两条指令被标记中断。



1. 在同一条指令的取指阶段，sram可能会连续发出两个data\_ok信号（没有搞懂产生这种情况的原因）。这种情况本身造成处理器处理结果出错，但是在trace比对的时候可能会导致连续将同一个数据被比对两次。所以采取了相对暴力的解决方法，即在WB模块规定相同的trace不能连续比对两次。
2. **实验分工**

付博宇、张钰堃负责完成exp14的代码及实验报告。