**Lab 9报告**

姓名： 杨泽辰 2021K8009926029

陈振兴 2021K8009925015

江文涛 2021K8009929042

箱号： 44

1. 实验任务
2. 完成Cache模块的设计，通过Cache模块测试。
3. 调整总线接口模块，在CPU中集成ICache，使得ICache能够给正确地CPU提供指令并加速CPU的取指。

二、实验设计

（一）总体设计思路

简要来说，Cache模块利用了访存的时空局部性的特点，对CPU最近访问的数据块进行缓存，从而加快CPU的访存速度。Cache模块位于CPU和主存中间，接受来自CPU的访存请求，并查找这一请求是否落在Cache中。若CPU要访存的数据落在Cache中，则直接访存相应的数据即可。若CPU要访存的数据落不在Cache中，Cache从主存中加载所需数据到Cache中。

从这些功能出发，在结构上，Cache首先要包含一系列Cache块来缓存数据，还要有Tag来标明该Cache块对应哪个内存单元的地址，Valid位来标明是否有效以及Dirty位来标明是否被改写。考虑到内存单元到Cache块的映射，Cache块通常有直接相连、 全相连和组相连这三种索引方式。

在功能上，Cache要能够将CPU的访存地址与Cache块的Tag和Valid位进行对比以确定访存地址是否落在Cache中，根据CPU的请求读写Cache块并返回相应的信号，在Cache不命中时选出Cache块替换为从主存中读取的新块并将脏块写回，以及对Tag、Valid位和Dirty位进行维护。Cache的这些功能可以通过状态机的方式来实现。

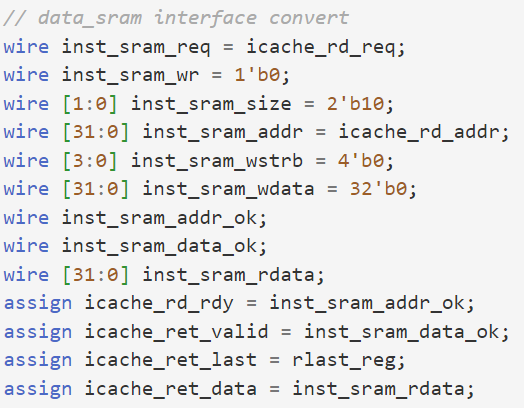
要将ICache集成到我们实现的CPU中，我们需要配合Cache模块修改CPU、类SRAM\_AXI转接桥的访存接口，并与ICache相连接，使得ICache能够接受来自CPU的访存请求，并向类SRAM\_AXI转接桥发起访存请求。此外，我们还要为类SRAM\_AXI转接桥增加burst传输的支持。

1. 重要模块1设计：Cache模块

讲义对此的介绍十分详细，就不在此赘述了。笔者认为，讲义给出的Cache设计有两点比较妙的地方。一是虚Index实Tag的设计，用虚Index查找相比于实Index延迟更少，用实Tag比对避免了Cache中存在同一内存单元的多份拷贝；二是将一个Cache行分为多个bank的设计，减少了读写同一Cache行的时候的冲突。

1. 重要模块2设计：ICache集成

为了将ICache集成进CPU，我们需要修改CPU和类SRAM\_AXI转接桥的访存接口。为了不大面积修改涉及这些要更改的访存接口的代码，我们实际上可以对这些访存接口进行转换。以转接桥为例，由于接口逻辑差不多，因此很简单就可以转换过来，代码如下：



要使转接桥支持burst传输，需要对arlen的赋值逻辑进行修改，读Cache行时置为8’b11，并且对读响应状态机进行相应的修改，一次接收数据直到rlast信号拉高为止。

三、实验过程

（一）实验流水账

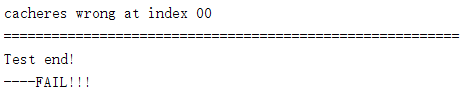
|  |  |  |
| --- | --- | --- |
| 时间 | 行为 | 结果 |
| 2023.12.09 18:00-次日2:00 | (exp20)阅读讲义并完成Cache模块的设计 | 仿真和上板测试通过 |
| 2023.12.17 22:00-次日3:00 | (exp21)将ICache集成至CPU中 | 仿真和上板测试通过 |

（二）实践任务 20：Cache 模块设计

1、错误1：(应该是)testbench的判断结果错误时忽略了请求尚未握手成功的情况

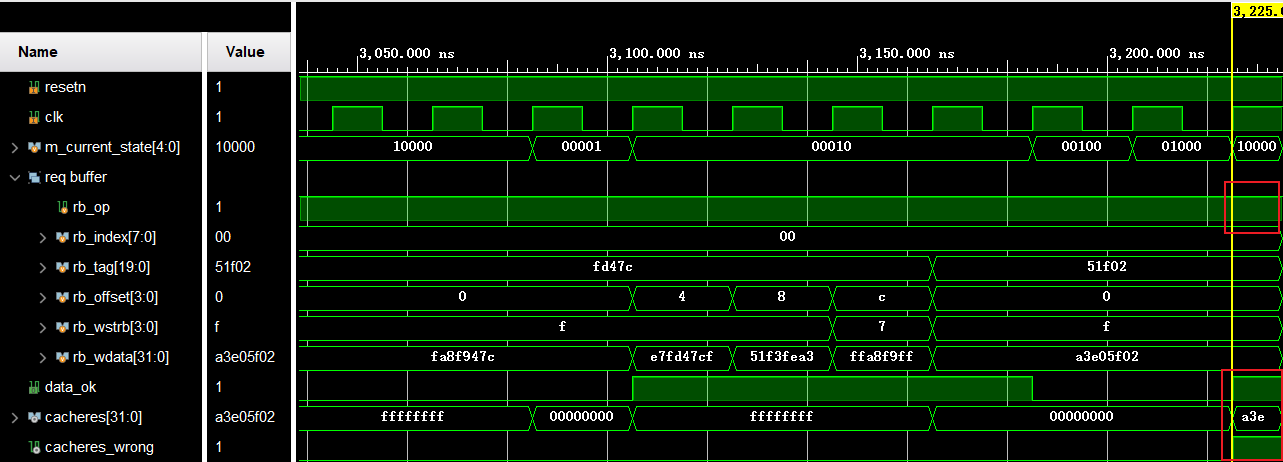
（1）错误现象

仿真时报错，返回值错误：



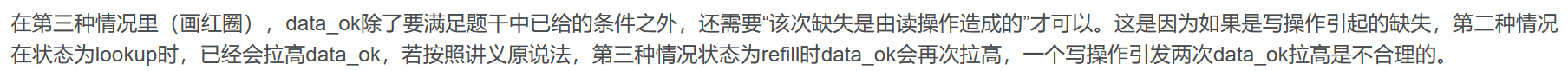
1. 分析定位过程

当时在debug时看req buffer以了解当前处理的请求，然后cacheres\_wrong拉高说明读取值错误，然后查看data\_ok信号拉高说明当前返回数据有效。此时尚在处理写请求，因此认为是写请求MISS时data\_ok会拉高两次导致的。当时去掉了LOOKUP状态且处理的是写操作拉高data\_ok的逻辑。



写报告时重新看来一下之前的波形，结果发现实际上这个读请求还没握上手。

关于这个问题，piazza上有相关的讨论(https://piazza.com/class/lm4n3gc0zlc3kp/post/80):



但是实际上第二次拉高是没有问题的，只要之前addr\_ok没有握上手，不过提出该问题的同学的处理比我好…

查看testbench，在进入读状态后，只要data\_ok拉高就进行判断，而没有考虑有没有握上手。





我的波形如下，可以看到下面波形的addr\_ok信号在testbench要进行读测试(round\_state==2’b10)时并没有被拉高，请求还没握上手，而caches\_wrong就拉高了。



1. 错误原因

testbench的判断结果错误时忽略了请求尚未握手成功的情况。

（4）修正效果

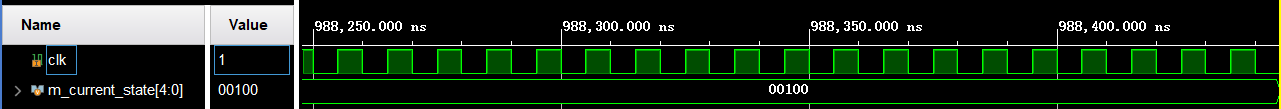
当时靠着去掉了LOOKUP状态且处理的是写操作拉高data\_ok的逻辑的方法通过了仿真和上板测试…

1. 实践任务 21：在 CPU 中集成 ICache

1、错误1：wdy接为0导致Cache主状态机在MISS状态不动

（1）错误现象

Cache主状态机在MISS状态不动：

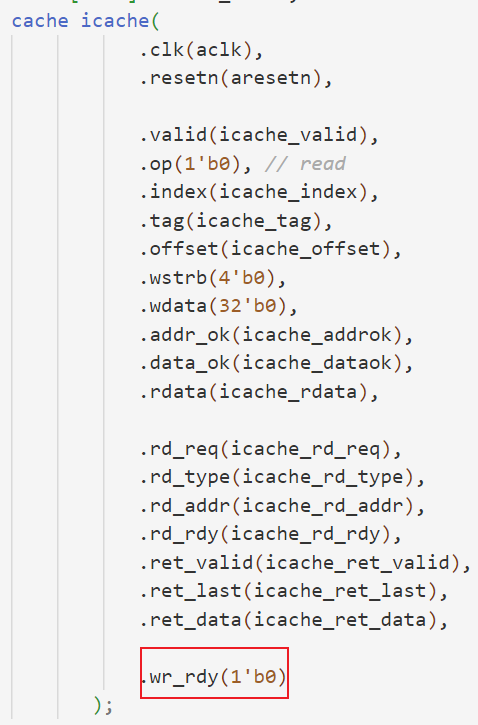


1. 分析定位过程

查看Cache主状态机的MISS状态的状态转移逻辑，需要wr\_rdy为1才能转移到REPLACE状态。

1. 错误原因

实例化ICache时将wr\_rdy接到了1’b0，认为ICache不需要写回从而忽视了状态转移需要wr\_rdy为1的条件。



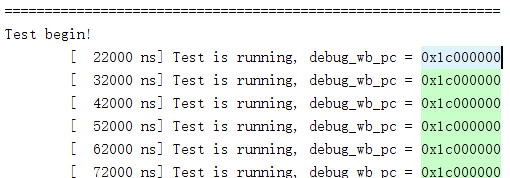
1. 修正效果

修改后遇到下一个BUG。

2、错误2：转接桥的读相应状态机未考虑burst传输

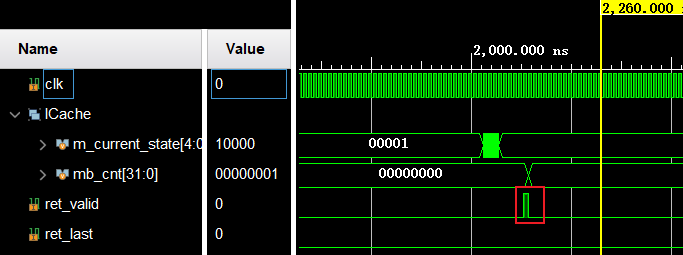
（1）错误现象

debug\_pc停在0x1c000000不动:



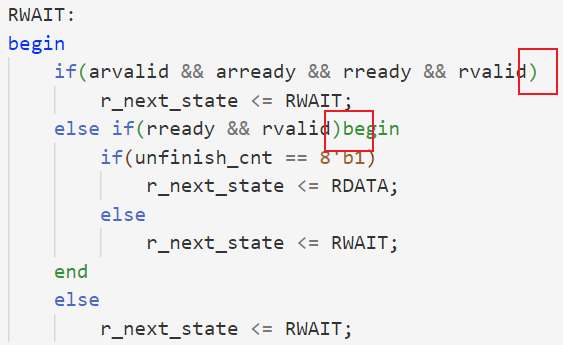
1. 分析定位过程

查看Cache主状态机，卡在REFILL状态。查看到转接桥只返回了一个值。



1. 错误原因

原来的转接桥设计无需支持burst传输。我在添加burst传输支持时忘记修改读响应状态机，导致其在收到一个数据以后状态就转移走了，然后rready拉低不接受数据了。红框处应该与上rlast。



1. 修正效果

修改后通过仿真和上板测试。

1. 实验总结

Cache模块部分讲义写的十分的详细，详细到了把讲义写的文字翻译成verilog代码就可以了的程度，对于新手还是很友好的。笔者在实验时遇到的问题是搞混了同步RAM和Regfile，开始时忘记了同步RAM在下一拍才返回，因此导致了开始时看的比较迷糊，后面动手实验定制同步RAM时突然想起来这一问题。此外，Cache模块的代码量也是有些大的。

而将ICache集成到CPU的实验，尽管遇到了些bug，但是都挺好找的。集成ICache还是挺简单的。