**实验11报告**

学号 2017K8009929032 2017K8009929034

姓名 杨程远 杨宇恒

箱子号 15

一、实验任务（10%）

CPU 顶层修改为 AXI 接口，使其对外只有一个 AXI 接口，并内部完成取指和数据访问的仲裁。完成仿真和上板随机延迟测试。

二、实验设计（40%）

（一）总体设计思路

先将原来的顶层模块修改为类SRAM接口，之后使用新的顶层模块调用这个类SRAM接口的CPU和类SRAM-AXI转接桥。

对于内部结构中的取指访存和load、store指令访存进行修改，并修改原来对于访存、取指都可以单周期返回的假设。

（二）重要设计1：取指级修改

1. 功能需求

1）在译码级前递正确的nextpc后，储存nextpc直到发出inst\_req。2）接收到inst\_data并储存直到译码级allowin。3）在flush的时候如果有没有返回的inst\_req，要在它返回的时候忽略。

1. 约定

只有取指级和译码级握手后，才发出下一inst\_req。inst\_req可以被桥片一拍接受（CPU无视addr\_ok）。这样，inst\_req拉高的那一拍与上一条指令，这样，图1中的取指到译码级握手成功这一个周期都对应一个inst\_req。另外，我们认为译码级如果执行完，等候新的指令时不会发出有效的分支next\_pc。

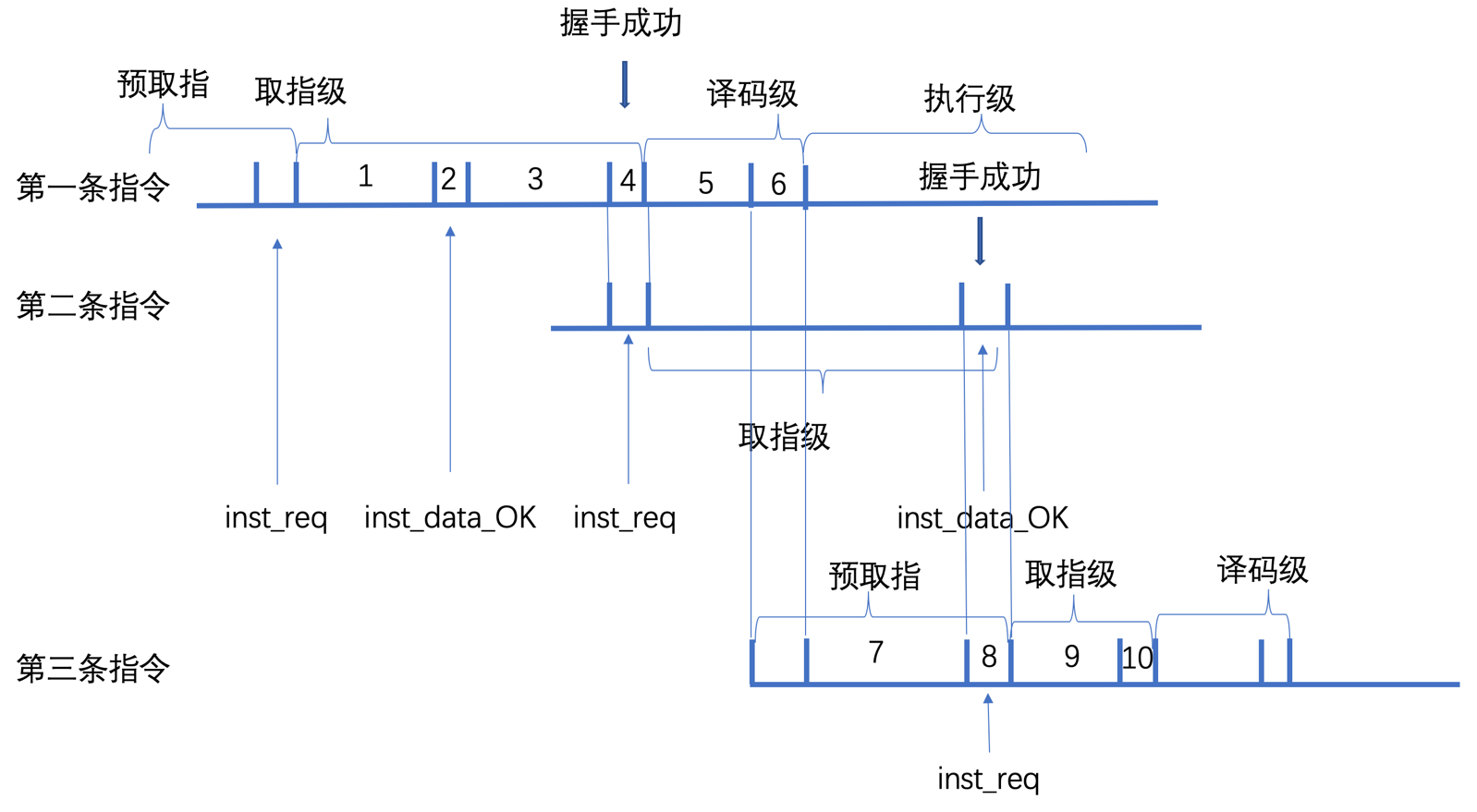


图1 预取指级和取指级中，类SRAM请求时序

1. 功能实现

结合图1对每个功能需求进行如下详细分析，并验证代码满足这些需求。

对于第二个储存指令的功能需求，我们参考第一条指令执行：在取指级过程中的2号周期，得到inst\_data\_OK，这时需要存下指令到buf\_inst、并在下一拍开始标记buf\_inst\_valid。等到4号周期，译码级ds\_allowin拉高，指令从5号时间段开始就存入了译码级寄存器。注意到此时下一条inst\_req才发出，不会在同一周期收到inst\_data\_OK，于是可以拉低buf\_inst\_valid。另一种情况如第二条指令：ds\_allowin先于或同时于inst\_data\_OK拉高，这就不需要在buf\_inst中保存，因此代码中有ifelse的优先级。

always @(posedge clk) begin

if (reset) begin

buf\_inst\_valid <= 0;

end

else if (fs\_to\_ds\_valid && ds\_allowin || flush) begin

buf\_inst\_valid <= 0;

end

else if (inst\_sram\_data\_ok\_after\_ignore /\*&& !flush\*/) begin

buf\_inst\_valid <= 1;

end

if (inst\_sram\_data\_ok\_after\_ignore) begin

buf\_inst <= inst\_sram\_rdata;

end

end

对于第一个储存nextpc的功能需求，我们参考第三条指令：其pc值有第一条指令的分支情况确定（第二条为延迟槽），考虑到数据前递中load指令需要阻塞一拍，译码级只有发出ready\_go的时候，即 6号周期，才可以得到正确的下一条指令地址，这时存到buf\_npc中，正如else if (!buf\_npc\_valid && ds\_allowin)条件。

always @(posedge clk) begin

if (reset) begin

buf\_npc\_valid <= 0;

end

else if (to\_fs\_valid && fs\_allowin || flush) begin

buf\_npc\_valid <= 0;

end

else if (!buf\_npc\_valid && ds\_allowin) begin

buf\_npc\_valid <= 1;

end

if (!buf\_npc\_valid && ds\_allowin) begin

buf\_npc <= nextpc;

end

end

对于第三个清流水线的需求，我们结合三条指令、任意发出flush的周期分析。对于第一条指令（inst\_data\_OK先于ds\_allowin），如果时间段1flush，需要记录无视下一个inst\_data\_OK；如果周期2flush，需要阻止buf\_inst\_valid下一周期拉高（下一周期不会返回例外处理/epc指令）；如果时间段3flush，需要下一周期拉低buf\_inst\_valid；如果周期4flush，需要逻辑控制当前周期fs\_ready\_go为低，并且下一周期拉低buf\_inst\_valid。对于第二条指令（ds\_allowin先于或同时于inst\_data\_OK），如果周期4flush，直接更改当前周期inst\_req地址，并发起inst\_req请求；如果时间段5/6/7flush，记录无视下一个inst\_data\_OK；如果周期8flush，需要逻辑控制当前周期fs\_ready\_go为低，并且阻止buf\_inst\_valid下一周期拉高。对于第三条指令，如果时间段4-7flush，只需下一周期转到等候pc的状态，即拉低buf\_npc\_valid并在下一周期得到pc。

（三）重要设计2：数据访存修改

1. 功能需求

保证对于访存延时的支持。

1. 约定

对于store指令，访存级收到data\_OK后在继续，简化设计。

1. 功能描述

与预取指级和取指级的修改类似。

（二）重要设计3：除法器模块结果暂存

1. 工作原理

在原来的除法器中，默认结果可以立即被下一流水级接受，但现在需要保存输出结果直至mem级可以进入。

1. 功能描述

原来除法器模块已经进行了握手封装，只不过out\_ready始终为1，只需将out\_ready设为ms\_allowin即可。

三、实验过程（50%）

（一）实验流水账

11.25 18:00-24:00 进行设计和取指级的修改。

11.26 12:00-18:00 完成代码和测试。

（二）错误记录

1、错误1：EPC错误

（1）错误现象

EPC错误

（2）分析定位过程

EPC因该是延迟槽前的一条指令但现在是延迟槽指令，于是检查bd的产生逻辑。

（3）错误原因

原来的设计中，直接将译码级的指令是否为分支传回取指级，但现在由于译码级可能流空，需要在取指级像保存nextpc那样将是否为分支的判断保存，作为之后的bd判断。

（4）修正效果

通过这一测试点。