DOI:10.16652/j.issn.1004-373x.2008.04.029

解决CPU流水线冲突技术的设计与实现

张大发，曹庆华，傅翠娇
（北京航空航天大学 计算机学院 北京 100083）

摘 要：流水线是制造高性能CPU的关键技术，目前许多学者研究在FPGA上实现具有流水线结构MIPS CPU，但是在解决流水线冲突上只是通过简单的停顿流水线实现。描述一种较为通用的具有五级流水线的MIPS CPU结构以及其中可能发生的流水线冲突，在此基础上详细介绍解决流水线冲突的技术——数据旁路以及动态分支预测在MIPS CPU中的设计和实现，最后通过一段指令序列进行仿真验证，解决流水线冲突的技术减少指令执行所需要的时钟周期数。

关键词：MIPS CPU；流水线；数据冲突；数据旁路；分支预测
中图分类号：TP332 **文献标识码**：B **文章编号**：1004-373X(2008)04-021-03

Design and Implementation of the Techniques of Solving Pipeline Hazards in CPU

ZHANG Dafa，CAO Qinghua，FU Cuijiao
（School of Computer Science and Engineering，Beihang University，Beijing，100083，China）

Abstract：Pipeline is the key implementation technique used to make fast CPU. Many developers design MIPS CPU with pipelined structure on FPGA，but they solve the problem of pipeline hazards by simply stalling pipeline. In this paper，a typical MIPS CPU with five-stage pipeline and pipeline hazards is discussed，then the methods of bypassing and dynamic branch prediction designed in the MIPS CPU，finally the CPU is simulated through a series of instructions，the methods of solving pipeline hazards reduce clock periods of executing instructions.

Keywords：MIPS CPU；pipeline；data hazards；bypassing；branch prediction

1 引 言

流水线是指在程序执行时多条指令重叠进行操作的一种技术。指令流水执行是将指令执行分成几个子过程，每个子过程对应一个工位，称为流水级或流水节拍，这个工位在计算机里就是可以重叠工作的功能部件，称为流水部件。这些不同的功能部件同时处理不同指令的不同子过程。流水级一个连着一个形成一个流水线，一条指令流过所有的流水级，就完成了他的任务[1]。他通过提高各流水部件的利用率提高指令的平均执行速度。在CPU流水线中，有一些称为冲突的情形，他使得指令流中下一条指令无法在设计的时钟周期内执行，这些冲突将会降低流水线可能获得的理想性能。笔者通过在已有的CPU基础上设计和实现解决流水线冲突的2大技术，并进行仿真验证，对通用的CPU流水线的设计具有一定的参考价值。

2 基于MIPS指令集的CPU流水线结构

MIPS指令集是一种精简指令集(RISC)，他的指令格式非常规整，所有的指令均为32位，而且指令操作码在固定的位置上[2]。这种特点易于将指令代码进行拆分，使之易于进行流水线CPU的设计。

在基本的MIPS处理器中有5个流水级，其中各流水级定义与主要功能为：

IF为计算下一条指令的地址PC，并从存储器读取这条指令；ID为对指令进行译码，从寄存器堆中取出源操作数；EX为当指令是运算类指令时执行运算，当指令是转移类指令时进行有效地址计算；MEM为从存储器存取数据；WB为将数据写回到寄存器堆。

按照这一流水线结构，本文在设计和实现中选用一种较为通用的MIPS CPU，他通过VHDL语言实现，各模块之间的关系简图如图1所示。

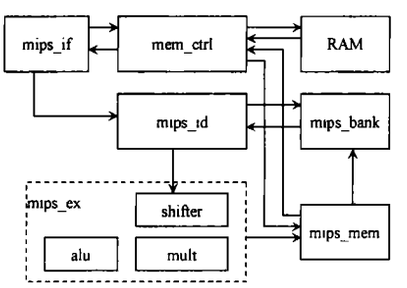

图1 MIPS CPU各模块关系简图

MIPS CPU各模块功能和流水段对应如表1所示。

在通常情况下，指令执行各阶段与时钟周期关系如表2所示，执行1条指令需要5个时钟周期，指令的平均执

收稿日期：2007-10-10

行时间为1个时钟周期。

3 CPU流水线中的冲突

在CPU流水线中存在3类冲突:资源冲突、数据冲突和控制冲突:

表1 MIPS CPU各模块功能和流水段对应关系表

模块	功　能	对应流水段
mips_if	计算下一条指令的地址	IF阶段
mips_id	对指令进行译码	ID阶段
mips_ex	执行运算和有效地址计算,包含了"alu"、"shifter"、"mult"三个模块	EX阶段
mips_mem	传递地址和数据到"mem_ctrl",用于读写内存操作,传递数据到"mips_bank",用于写回寄存器堆操作	MEM阶段
mips_bank	用于作寄存器堆,以及读写寄存器堆	ID和WB阶段
mem_ctrl	与内存连接的数据通道	IF和MEM阶段
ram	用于作内存	IF和MEM阶段

表2 指令流水状态表

指令序列	流水线时钟数								
	1	2	3	4	5	6	7	8	9
指令i	IF	ID	EX	MEM	WB				
指令i+1		IF	ID	EX	MEM	WB			
指令i+2			IF	ID	EX	MEM	WB		
指令i+3				IF	ID	EX	MEM	WB	
指令i+4					IF	ID	EX	MEM	WB

资源冲突　当处理器进行流水处理的时候,指令的重叠执行要求功能部件能够流水,而且资源重复设置,以便流水线中的指令能自由组合。如果因为资源冲突而无法使用某种指令的组合,那么这台机器就被称为有资源冲突的。资源冲突的解决一般需要重复设置资源,本文不讨论这一问题。

数据冲突　当指令之间存在相关性,而且他们的执行时间相近时,可能会在流水线中重叠操作,或者指令的重新排序改变了有相关的操作数的访问顺序,这时数据冲突就会发生。在五级流水线的MIPS CPU中,可能存在写读冲突,比如有以下几条顺序执行的指令(指令序列1):

```
ORI    $2 $0 0x0000
ORI    $3 $0 0x000A
ADDI   $2 $2 0x0001
BNE    $2 $3 0xFFFF
```

第1条ORI指令在WB阶段才能将数据写回到寄存器,而ADDI指令在ID阶段就要读取该值,通过表2可以很明显看出ADDI指令读取值的周期早于第1条ORI指令写入值,读出来的将是旧值,要得到真实的结果需要停顿2个周期的流水线。

控制冲突　流水线中的分支指令或其他改写PC的指令造成的冲突。指令序列1的BNE指令为分支指令,该序列也存在控制冲突。控制冲突造成的损失比数据冲突更大。控制冲突决定了跟分支指令有关的指令的执行顺序,即非分支指令只能在该执行的时候才能执行。

4 数据旁路技术的设计与实现

数据旁路技术是解决数据冲突问题简单有效的方法,他的基本思想是把结果直接送到需要他的功能部件,即在同一周期内,将1个结果从一个部件的输出直接送到另一个部件的输入。

基于这一技术,设计解决数据冲突问题的流程:

(1) 判断在ID阶段的指令的2个操作数是否使用寄存器,如果不使用,则不存在数据冲突,如果使用,则转入(2);

(2) 判断EX阶段或MEM阶段正在执行的指令会写回值的寄存器是否为ID阶段使用的寄存器,如果否或者EX阶段或MEM阶段正在执行的指令没有写回值的寄存器,则不存在数据冲突,从寄存器堆中取出源操作数;如果是,则记录该阶段,转入(3);

(3) 判断所记录的阶段中的指令在该阶段是否已经产生结果,如果是,将这个结果赋给ID阶段指令的源操作数;如果否,发生数据冲突,需要停顿流水线。

在数据旁路技术的实现上,首先是增强CPU的译码功能,在译码时增加一些与数据冲突相关的功能码,流水等级功能码指示每条指令产生结果的流水阶段,根据对数据冲突问题的分析,该功能码的取值可以代表在DI阶段、EX阶段或者MEM阶段得到结果,如JR指令在DI阶段得到结果,ADD指令在EX阶段得到结果,LB指令在MEM阶段得到结果,源操作数是否使用寄存器功能码指示了指令的2个源操作是否使用到寄存器;另外,数据旁路技术还用到指示是否写回寄存器堆功能码以及目的数寄存器地址功能码。当增加1个时钟周期时,这些功能码随着指令在各流水线寄存器中传递。其次,增加1个新的模块命名为"bus_ctrl"来实现数据旁路技术,解决数据冲突,同时他也是"mips_id","mips_ex","mips_mem","mips_bank"之间相互沟通的桥梁。

5 分支预测技术的设计和实现

分支预测技术是为解决控制冲突问题而产生的。在程序中,分支转移很多,而且也很复杂,但是还是可以从中找到规律,发现一些有针对性的预测策略,从而提高系统的性能。比较常见的一类分支转移程序是循环程序,这里发现:在多重循环里,分支转移总是有很强的偏向性,也就

是说分支总是经过多次同向的执行，才会偶尔转向另外一个方向[3]。在MIPS CPU中利用分支预测表的动态预测技术实现分支预测功能。

构建分支预测表 分支预测表由4个字段构成，分别为预测记录使能信号、指令上一次执行时是否分支信号、指令地址以及预测分支地址。采用指令地址字段作为索引。分支预测表的大小由参数来指定。

添加预测记录 如果在ID阶段某指令被译出为分支跳转指令，且该指令地址在分支预测表中不存在时则要添加到预测表中。预测记录的替换策略采用先进先出的方法，即用一指针指示最早存入的1条预测记录或者空记录，新的预测记录存入时替换该记录，同时指针加1，当指针超出记录表大小时，将指针指向第一条记录。这类似于循环队列机制。

分支预测 判断IF阶段取出的指令地址是否与预测表中某一记录的指令地址相同，如果相同，则该指令是一分支跳转指令，将该记录的预测分支地址传给取指构件，即下一条指令地址为该预测分支地址。

分支预测的验证 当指令执行到EX阶段时，已经计算出指令的实际地址，将此地址与预测的分支地址做比较，如果相同，则预测正确，指令继续执行；如果不同，则预测错误，这时需要把实际的地址传给取指部件，将已经执行的错误指令清除；同时修改分支预测表中与该预测相关的记录。

表3列出当有1条分支指令第1次以及第2次执行时IF，ID，EX阶段在各时钟周期与该指令相关的操作。在MIPS CPU中，用一独立的功能部件“mips_predict”实现分支预测功能，他与“mips_if”，“mips_id”，“mips_ex”有着紧密的联系，需要将指令地址随着指令在各流水线寄存器中传递，并传递给“mips_predict”进行分支预测。

表3 分支指令执行情况表

新的分支指令	IF阶段操作	ID阶段操作	EX阶段操作
时钟周期 i	取该分支指令		
时钟周期 $i+1$	判断指令地址是否与预测表中某一记录的指令地址相同，不相同，顺序取指	译码得到该指令是分支指令，	
时钟周期 $i+2$	继续顺序取指； 将该指令地址添加到预测表中，预测地址为顺序的下一条指令地址，预测表指针加1。		计算指令的实际地址，
时钟周期 $i+3$	继续顺序取指； 将指令的实际地址与预测的分支地址作比较，不同，预测错误，修改预测表中的相关记录，把实际的分支地址传给取指部件。		
时钟周期 $i+4$	取出分支后的指令	清除当前指令	清除当前指令
再次执行到该分支指令			
时钟周期 j	取该分支指令		
时钟周期 $j+1$	判断指令地址是否与预测表中某一记录的指令地址相同，相同，将预测分支地址传给取指部件进行取指	译码得到该指令是分支指令，	
时钟周期 $j+2$	顺序取值		计算指令的实际地址，
时钟周期 $j+3$	继续顺序取指； 将指令的实际地址与预测的分支地址作比较，相同，预测正确，		
时钟周期 $j+4$	继续顺序取指		

6 解决流水线冲突技术的仿真验证

设计实现后的MIPS CPU增加“bus_ctrl”和“mips_predict”两个模块。作为对整个流水线的控制，必须在流水通道内运行各种指令流，分析流水线的控制信号能否对这些指令流在流水线上的执行进行精确控制。对整个处理器进行虚拟仿真和验证，在仿真平台和验证板上运行各种可能的指令流，分析流水线控制信号的变化，确保这个处理器的中心控制模块对指令流水线的正确控制[4]。输入前文的指令序列1，通过Modelsim平台对MIPS CPU进行仿真验证，波形图如图2所示。

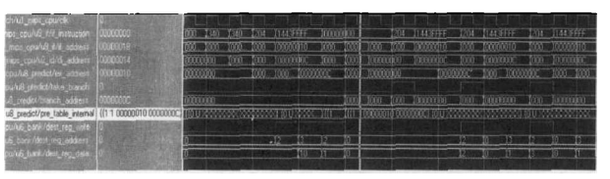

图2 MIPS CPU仿真验证波形图

ADDI指令在执行时没有流水线延迟，而且目标寄存器的值正确，说明第2条ORI指令在EX阶段结束后把值赋给正在ID阶段执行译码的ADDI指令，由图2可以看到预测表添加的过程，只有第1次执行分支指令时需要较长的时间，以后当遇到该分支指令时，总会在下一周期取出预测地址，直到该地址错误为止。

流水线技术提高了时钟频率，而解决流水线冲突的技术减少了指令执行所需要的时钟周期数，两者结合可以大大提高CPU的执行效率。

参 考 文 献

[1] John L Hennessy，David A Patterson. 计算机系统结构：量化研究方法[M]. 北京：电子工业出版社，2004.

[2] 朱子玉，李亚民. CPU芯片逻辑设计技术[M]. 北京：清华大学出版社，2004.

[3] 冯雷. 一种适用于MIPS指令系统的分支预测方法[D]. 北京：中国科学院研究生院，2002.

[4] 李明刚. 64位MIPS指令处理器的流水线设计[J]. 现代电子技术，2005，28(3)：98-100.

[5] 黄敏敏，林媛，徐中佑. 一种采用3级指令流水线的51内核设计[J]. 现代电子技术，2005，28(20)：83-85.