

一种流水线 CPU 设计中的 Hazard 消除法

刘秋菊^{1,2}, 李 飞², 刘书伦²

(1 重庆大学 自动化学院, 重庆 400044; 2 济源职业技术学院 信息工程系, 河南 济源 459000)

摘 要: 采用基本五步流水线 CPU 设计, 而 Hazard 是流水线 CPU 设计中必须要处理的问题. 介绍了三种 Hazard 类型, 并提出了解决这三种 Hazard 问题的方法. 测试结果表明, 该方案符合设计要求.

关键词: 五步流水线; CPU 设计; 指令; Hazard

中图分类号: TP332

文献标识码: A

文章编号: 1000-7180(2012)08-0033-04

Method of Hazard Elimination Based on Design of Pipeline CPU

LIU Qiu-ju^{1,2}, LI Fei², LIU Shu-lun²

(1 College of Automation, Chongqing University, Chongqing 400044, China;

2 Department of Information Engineering, Jiyuan Vocational and Technical College, Jiyuan 459000, China)

Abstract: We consider the design of 5 stage pipeline CPU, and it is must to deal with Hazard. In this paper three kinds of Hazard are given, and method is proposed to solve the question. The test results show that the scheme meets the design requirements.

Key words: 5 stage pipeline; design of CPU; instruction; Hazard

1 引言

CPU 作为微处理器的核心部件, 其性能直接影响整个系统的性能. 现代微处理器广泛采用流水线设计, 所以 CPU 流水线是决定其性能的关键^[1-2]. 本文介绍了基本五步流水线 CPU 设计, 从 MIPS 指令集中选取 15 条指令作为本 CPU 的基本指令, 其他指令在译码阶段均解释为未定义指令^[3], 流水线五个阶段的主要功能为: IF 为取指阶段, 主要为计算下一条指令的地址 PC, 并从指令存储器读取指令; ID 为译码阶段, 对指令进行译码, 从寄存器堆中取出源操作数; EX 为执行阶段, 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 为内存访问阶段, 从数据存储器读写数据; WB 为寄存器写回阶段, 将数据写回到寄存器堆^[4]. Hazard 是设计流水线 CPU 必须要处理的问题, Hazard 包括 Structural Hazard、Data Hazard 及 Control Hazard. 解决 Hazard 方法的好坏将直接影

响流水线 CPU 的性能, 因此本文将详细分析 CPU 设计中解决三个 Hazard 问题的方法.

2 Structural Hazard

Structural Hazard 即结构危害, 指的是多条指令需要在同一时刻使用相同的功能单元, 而当前 CPU 中的功能单元数量不够, 不能同时满足它们的需求. 在设计 CPU 中, 有两种潜在的 Structural Hazard:

(1) 对寄存器资源的同时访问, 如图 1 所示. 在 WB 阶段, R 类型指令需要将运算结果写回寄存器, 而与此同时, 其后面第三条指令正处在 ID 阶段, 可能需要读取寄存器的内容以提供给之后的 EX 阶段运算, 这就造成了在同一时刻有两条指令访问寄存器资源的问题. 解决办法很简单, 实际上写数据的同时可以读数据 (写和读为同一地址的属于 Data Hazard 将在后面分析), 只要实现单独的写地址线、写数据线和读地址线、读数据线即可.

收稿日期: 2011-12-26; 修回日期: 2012-02-17

基金项目: 国家自然科学基金项目 (11001075); 河南省科技厅科技攻关项目 (092102210327)

Rtype Instr	Hazard							
	IF	ID	EX	MEM	WB(REG)			
Instruction1		IF	ID	EX	MEM	WB		
Instruction2			IF	ID	EX	MEM	WB	
Instruction3				IF(MEM)	ID(REG)	EX	MEM	WB

图1 同时访问寄存器造成 Structural Hazard

(2)对内存资源的同时访问,如图2所示.在MEM阶段,Load或Store类型指令需要访问内存单元,而与此同时其后面第三条指令正处在IF阶段,需要从内存中读取指令,这就造成了在同一时刻有两条指令访问内存资源的问题.首先想到的办法是采用哈佛结构Cache,建立独立的指令Cache和数据Cache.但是如果在此时指令Cache和数据Cache同时miss怎么办?目前的内存一般都是采用冯诺依曼的指令数据结构,解决办法可能只有让后面的指令stall了,但是问题在于只有当MEM进行过程中才能发现Cache的miss,此时后面指令的IF也已经开始访问内存,如何做到只让MEM阶段发出地址上总线而屏蔽掉IF阶段发出的地址?由于内存访问涉及到总线等操作,步骤复杂,在本次设计的重点是CPU内部结构,因此并没有考虑此问题的解决,而采取较简单的形式,即内存也是采取哈佛结构,指令内存与数据内存分离.

LW or SW Instruction	Hazard							
	IF	ID	EX	MEM	WB			
Instruction1		IF	ID	EX	MEM	WB		
Instruction2			IF	ID	EX	MEM	WB	
Instruction3				IF(MEM)	ID	EX	MEM	WB

图2 同时访问内存造成的 Structural Hazard

3 Data Hazard

Data Hazard即数据危害,指的是同处在流水线中的多条指令读写数据之间存在依赖关系而可能产生的数据不一致问题. Data Hazard有三种情况,如表1所示.可以看到WAR可能出现的问题是后面指令读前面指令的结果数据不一致, WAW可能出现的问题是后面指令修改的数据被前面指令的修改覆盖, RAW可能出现的问题是前面指令要读的数据先被后面的指令修改了.由以上分析可知, WAW和RAW问题均是由于后面的指令比前面的指令先执行而引发的,而本次设计的CPU只实现顺序执行,不存在乱序执行的情况,所以可能产生的Data Hazard只能是WAR问题.

下面来分析WAR问题产生的原因及其解决办法:

首先,指令写数据是在MEM阶段写内存或WB阶段回写寄存器,而读数据是在ID阶段读寄存

器或MEM阶段读内存. WAR问题可分为先写寄存器、后读寄存器和先写内存、后读内存.很显示,读写内存都是在同一阶段MEM,所以读到的数据都是已写过的;而读寄存器的ID阶段在写寄存器的WB之前,所以它是产生WAR的唯一情况.

表1 三种Data Hazard问题

Data Hazard	典型指令序列
WAR(写后读)	ADD R1, R0, R0 ADD R2, R1, R1
WAW(写后写)	ADD R1, R0, R0 ADD R1, R2, R2
RAW(读后写)	ADD R0, R1, R2 ADD R2, R1, R1

再来分析产生写后读寄存器造成Hazard的指令序列区间.如图3所示,可以看到,写R1指令之后的第四条指令执行到ID阶段需要读R1时,写R1指令已经执行完了,所以肯定不会产生WAR.因此可以得出结论,产生WAR的指令序列区间为本条写指令与其后的第三条指令之间.

	Hazard							
	IF	ID	EX	MEM	WB(R1)			
ADD R1, R0, R0		IF	ID(R1)	EX	MEM	WB		
ADD R2, R1, R0			IF	ID(R1)	EX	MEM	WB	
ADD R4, R1, R0				IF(MEM)	ID(R1)	EX	MEM	WB
ADD R5, R1, R0					IF	ID(R1)	EX	MEM

图3 写后读寄存器造成Hazard的指令序列区间

进一步分析可知,写指令的WB阶段与其后第三条指令的ID阶段处在同一时刻,之所以可能产生WAR问题是因为Register File要到下一个周期开始时产生上升沿才能锁存写数据,而Stage Register也是在每周周期开始时产生上升沿锁存上一阶段数据的.解决办法是让Register File在本周期内的下降沿就锁存写数据,这样当Stage Register锁存时就是前面已写的数据了.经过如此优化,产生WAR问题的指令序列区间就缩小为本条写指令与其后的第二条指令之间.

综合以上分析得出的最重要的结论是在检测是否存在Data Hazard时,只需比较相邻的三条指令即可.本CPU设计为在ID阶段检测,因此就只需要将当处在ID阶段的指令与流水中正处在EX和MEM阶段的两条指令相比较就可检测出是否存在Data Hazard.下面分析所有将产生Data Hazard的情况(假定当前处在ID阶段的指令为本条指令)^[6-9].

- 当本条指令与其前一条指令满足如下条件时

存在 Data Hazard:

(1)前一条指令需要回写寄存器(即不是 Store、Jump 和 Branch 指令)

(2)本条指令引用了前一条指令的目的寄存器
解决办法也需要分情况讨论:

(1)当前一条指令不是 Load 指令时,待写回结果实际上已在 EX 阶段产生,可以将 EX 阶段的执行结果直接 FORWARDING 到 EX 阶段之前的 Stage Register,如图 4 所示。

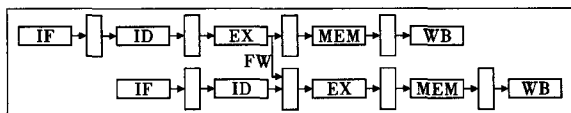


图 4 EX 结果数据 FORWARDING

(2)当前一条指令为 Load 指令且本条指令不为 Store 指令时,待写回结果要到 MEM 阶段才能产生,但此时前一条指令还处在 EX 阶段,所以本条指令必须 Stall 一个周期,即加一个 Bubble,这也意味着本条指令之后的指令也必须 Stall 跟着 Stall. 本条指令被 Stall 一个周期后,就可以使用 FORWARDING 技术来消除 Data Hazard 了,只需将 MEM 结果数据直接 FORWARDING 到 EX 阶段之前的 Stage Register,如图 5 所示。

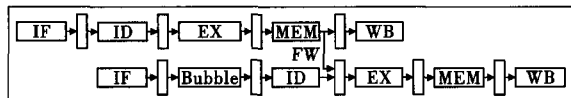


图 5 MEM 结果数据 FORWARDING 至 EX

(3)当前一条指令为 Load 指令且本条指令为 Store 指令时,这是个特殊情况,因为虽然说待写回结果要到 MEM 阶段才能产生,但是由于 Store 指令也是在 MEM 阶段才需要用到源寄存器,所以不需要 Stall,只需等到前一条 Load 指令到达 MEM 阶段后再将 MEM 阶段结果数据直接 FORWARDING 到 MEM 阶段之前的 Stage Register 即可,如图 6 所示。

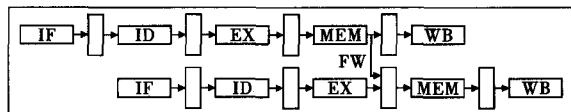


图 6 MEM 结果数据 FORWARDING 至 MEM

● 当本条指令与其前前条指令满足如下条件时存在 Data Hazard:

(1)前前条指令需要回写寄存器(即不是 Store、Jump 和 Branch 指令)

(2)本条指令引用了前前条指令的目的寄存器
解决办法同样需要分情况讨论:

(1)当前条指令是 Load 指令时,待回写结果是从内存读出来的数据,因此只需将内存读数据线 FORWARDING 到 EX 阶段之前的 Stage Register 即可,如图 7 所示。

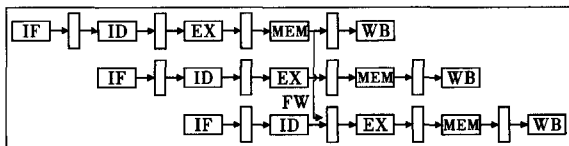


图 7 MEM 结果数据 FORWARDING 至 EX

(2)当前前条指令不是 Load 指令时,待回写结果实际上在前前条指令处于 EX 阶段时就产生了,所以只需直接将 MEM 阶段之前的 Stage Register 输出的 ALU 结果数据 FORWARDING 到 EX 阶段之前的 Stage Register 即可,如图 8 所示。

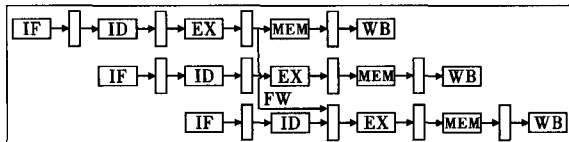


图 8 Reg_EX/MEM 锁存的结果数据
FORWARDING 至 EX

4 Control Hazard

Control Hazard 即控制危害,指的是由于跳转或分支指令而造成其后面已进入流水线的指令实际上无效的问题. 在本 CPU 中,无论是绝对跳转还是条件分支指令,对其是否跳转的判断和跳转目的地址的计算都是在 ID 阶段就完成,因此实际上只有一条指令可能产生 Control Hazard,如图 9 所示。

Hazard	Instruction 2 when untaken Branch target when taken					
	IF	ID(J&B)	EX	MEM	WB	WB
JUMP or BRANCH Instruction 1		IF	ID(R1)	EX	MEM	WB
			IF	ID	EX	MEM

图 9 Control Hazard 问题

由于只有一条指令可能产生 Control Hazard,所以首先想到的解决办法是在检测出 Branch-Taken 后,将其后第一条指令标志为无效,即不改变运行环境. 这样一来,相当于为了解决这个 Hazard 而使流水线延迟了一个周期。

然而,在本设计中采用了延迟转移技术来消除这个周期的延迟. 延迟转移技术主要是由编译器将某一条指令放在 Delay Slot 中,Branch 指令后将紧

跟着执行 Delay Slot 中指令,无论 Branch 是 taken 还是 untaken,这条指令的执行不会造成整个运算的错误.本设计假定运行的代码是具备延迟转移技术的编译器产生的,所以实际上 Control Hazard 问题已经不会再出现.

5 Data Hazard 的检测方法

在 ID 阶段需要完成的任务比较重,其中一个重要的任务就是检测是否存在可能的 Data Hazard,并选择是否使用 Bubble 法和 Forwarding 法消除 Hazard.其中需要特别说明的是寄存器文件只实现了 8 个,而不是 MIPS 指令集的 32 个.最重要的部分是控制单元,它是用 Verilog 语言实现的,任务是解释指令、产生控制信号、检测与消除 Data Hazard.控制单元的输出脚都是控制信号,而输入脚中有六个是为了 Data Hazard 的检测与消除.为了讨论方便,将这些输出与输入通过列表整理,如表 2、表 3 所示.

在检测 Data Hazard 时,首先测试 EWREG 和 EM2REG 输入脚:如果 EWREG 为 1,那么本条指令的前一条指令是回写寄存器指令;如果 EM2REG 为 1,那么前一条指令是 Load 指令.再用本条指令的 RS 和 RT 字段(即本条指令可能引用的寄存器地址)与 EDESR 比较就可以知道是否存在数据相关性.同样通过测试 MWREG、MM2REG 及 MDESR 即可知道本条指令与前条指令是否存在数据相关性.如果存在数据相关性,则根据以上讨论的不同情况发出不同的信号控制 Hazard 的消除.NOSTALL 输出信号用于 Bubble 技术的控制,而 FWDA、FWDB 及 FWDC 输出信号用于 FORWARDING 技术的控制.

表 2 为控制 FORWARDING 设置的输入脚

输入脚	描述
EWREG	当前处刚结束了 ID 阶段的指令的回写寄存器控制信号,为 1 表示其要回写寄存器
EM2REG	当前处刚结束了 ID 阶段的指令的读数据 MEM 控制信号,为 1 表示要读 MEM
EDESR	当前处刚结束了 ID 阶段的指令的回写寄存器地址
MWREG	当前处刚结束了 EX 阶段的指令的回写寄存器控制信号,为 1 表示其要回写寄存器
MM2REG	当前处刚结束了 EX 阶段的指令的读数据 MEM 控制信号,为 1 表示要读 MEM
MDESR	当前处刚结束了 EX 阶段的指令的回写寄存器地址

表 3 控制单元输出控制信号

输出控制信号	作用阶段	描述
NOSTALL	ID	为 1 表示当前指令需要 stall
FWDA	ID	选择是否需要 FORWARDING 且需要哪个 FORWARDING 端
FWDB	ID	选择是否需要 FORWARDING 且需要哪个 FORWARDING 端
FWDC	ID	选择是否需要 FORWARDING 且需要哪个 FORWARDING 端
BEACH	ID	为 1 表示要分支
JUMP	ID	为 1 表示要绝对跳转
SHIFT	EX	ALU port1 选择信号,为 1 选择 sa,为 0 选择寄存器 rs
ALUIMM	EX	ALU port2 选择信号,为 1 选择立即数,为 0 选择寄存器 rt
SEXT	ID	选择立即数扩展方式,为 1 选符号扩展,为 0 选零扩展
ALUC	EX	ALU 操作命令
WMEM	MEM	写数据 MEM 控制信号,为 1 表示要写 MEM
WREG	WB	回写寄存器信号,为 1 表示要回写寄存器
M2REG	WB	读数据 MEM 控制信号,为 1 表示要读 MEM
REGRT	ID	回写寄存器选择信号,为 1 选择 rt 标识的寄存器为回写寄存器,为 0 则选择 rd

作为整个流水线的控制,功能模拟测试需要设计典型的代码序列来测试流水线的运行状况及 Hazard 问题的解决.测试结果与预期的波形图进行比较,结果是吻合的.

6 结束语

本文对在流水线 CPU 设计中,产生的 Hazard 进行详细介绍,在分析了产生的原因后,提出解决方案:Bubble 技术和 FORWARDING 技术,然后通过指令序列进行了测试,测试结果符合设计要求.CPU 流水线设计还有许多地方需要改进,今后要继续对此进行研究.

参考文献:

[1] 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,2005.

基准电压源,要放在芯片的中间,而一些对温度和噪声要求不高的模块,例如数字逻辑,可以放在靠近大功率管的地方,另外要做好大功率管低压和控制电路的隔离工作^[5-6].

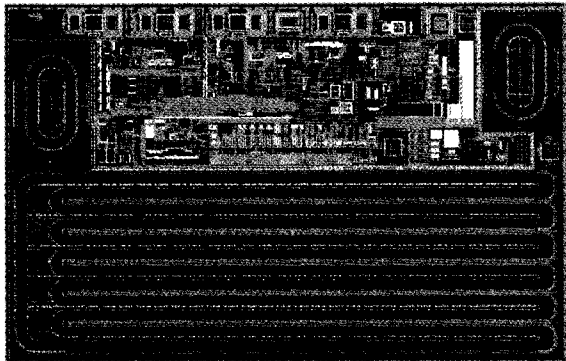


图 10 整个芯片的版图

7 结束语

本文从分析整个 AC/DC 电源管理系统各部分的功耗出发,在采用多模式控制策略的前提下,分别设计了一种低功耗启动电路、低功耗采样电路和低功耗驱动电路。仿真结果表明,将电路放在整个系统中可以如预期一样正常工作,大大降低了整个系统的功耗,并且可以完全集成在同一芯片内。

参考文献:

- [1] 王予,张培,胡海如. 开关电源原理及发展方向[J]. 中国科技信号, 2007(14):85-86.

- [2] Wang J, Gong W C, He L N. Design and implementation of high-efficiency and low-power DC-DC converter with PWM/PFM modes[C] // 7th International Conference on ASIC. China: Guilin, 2007:596.
- [3] James Lei, San Jose, Calif. High Voltage start-up circuit and method therefor[P]. USA, 5815383, 1998.
- [4] Cheung Fai Lee, Philip K T Mok. A monolithic current-mode CMOS DC-DC converter with on-chip current-sensing technique[J]. IEEE Journal of Solid-State Circuits, 2004(39):265-268.
- [5] 陈兴弼. 功率 MOSFET 与高压集成电路[M]. 南京: 东南大学出版社, 1989.
- [6] Alan Hastings. The Art of Analog Layout Second Edition [M]. 2nd ed. Beijing: Publishing House of Electronics Industry, 2006.

作者简介:



姜伟男, (1987—), 硕士研究生. 研究方向为电源管理集成电路设计.

赵野男, (1977—), 副研究员. 研究方向为模拟集成电路和电源管理 SoC 设计.

黑勇男, (1974—), 研究员. 研究方向为可重构数字信号处理器、低功耗数字基带芯片.

(上接第 36 页)

- [2] John L Computer architecture: a quantitative approach, 3E[M]. Boston: Ap Professional, 2006.
- [3] 薛勃, 周玉洁. MIPS32 指令集兼容的 CPU 模拟器设计[J]. 计算机工程, 2009, 35(1):263-265.
- [4] 东野长磊, 戚梅. 一种带 Cache 的嵌入式 CPU 的设计与实现[J]. 微型机与应用, 2010(14):17-19.
- [5] 易海峰. 一种兼容处理器指令译码单元的优化设计[J]. 微电子学与计算机, 2004, 21(12):119-122.
- [6] 赖兆磐, 潘明, 张辉. 基于 FPGA 流水线 CPU 控制器的

设计与实现[J]. 微计算机信息, 2008(7-2):233-235.

- [7] 曹学飞, 张盛兵, 张骏. 32 位 CISC 微处理器流水线的设计[J]. 微电子学与计算机, 2007, 24(4):186-189.

作者简介:

刘秋菊女, (1970—), 硕士, 副教授. 研究方向为计算机应用.

刘书伦男, (1978—), 讲师. 研究方向为计算机体系结构.