思考题解答

1. 冲突处理

本设计采用了由控制信号控制的冲突控制单元来解决流水线冲突问题。

1. 思路概述

流水线冲突解决的问题是由流水线后级产生了写入寄存器的值，但还没有写入，但值有。转发电路和暂停机制。需求的值值可能是Rs和Rt，而写入的值只有一个。

转发机制的核心是由后级流水线转发到前级别。本设计采取新值产生后（大部分在EX阶段后）立即持续转发的策略。由于流水线IF级不需要对寄存器的操作，故这里我们可能进行的转发路径共有六条；再仔细分析，1. 对于ID 🡪 EX，在ID级需要使用寄存器值的只有Branch和Jump指令，由于延迟槽的存在，两条B, J指令并不能连续出现，而在EX级可能产生新值的指令只有Link型指令，故不需要构建EX 🡪 ID的转发；2. 对于WB 🡪 MEM，可以通过GRF片内转发实现。最后的分级结果如下表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 路径 | Rreg | Wreg | WregData | 结论 |
| EX 🡪 ID | Rs, Rt | Rt/Rd/31 | Rs,Rt | 不需要 |
| MEM 🡪 ID | Rs, Rt | Rt/Rd/31 | ALUOut/PC+8/Rs | 需要 |
| WB 🡪 ID | Rs, Rt | Rt/Rd/31 | WregData\_EX/MemReadData | GRF片内转发 |
| MEM 🡪 EX | Rs, Rt | Rt/Rd/31 | ALUOut/PC+8/Rs | 需要 |
| WB 🡪 EX | Rs, Rt | Rt/Rd/31 | WregData\_EX/MemReadData | 需要 |
| WB 🡪 MEM | Rt | Rt/Rd/31 | WregData\_EX/MemReadData | 需要 |

上表中，WReg和WRegData均有多个可能值；而本设计中，流水线每级中均会明确存在唯一的WReg和WRegData信号，便于冲突控制单元的实现。

处理转发的代码如下：

// -> EX\_Rt

assign Forward\_EX\_Rs = (Rs\_EX != 5'h00 && SigRegWrite\_MEM && WReg\_MEM == Rs\_EX) ? 2'b10 :

((Rs\_EX != 5'h00 && SigRegWrite\_WB && WReg\_WB == Rs\_EX) ? 2'b01 : 2'b00);

assign Forward\_EX\_Rt = (Rt\_EX != 5'h00 && SigRegWrite\_MEM && WReg\_MEM == Rt\_EX) ? 2'b10 :

((Rt\_EX != 5'h00 && SigRegWrite\_WB && WReg\_WB == Rt\_EX) ? 2'b01 : 2'b00);

// MEM -> ID

assign Forward\_ID\_Rs = (Rs\_ID != 5'h00 && SigRegWrite\_MEM && WReg\_MEM == Rs\_ID) ? 1'b1 : 1'b0;

assign Forward\_ID\_Rt = (Rt\_ID != 5'h00 && SigRegWrite\_MEM && WReg\_MEM == Rt\_ID) ? 1'b1 : 1'b0;

// WB -> MEM Only Rt

assign Forward\_MEM\_Rt = (Rt\_MEM != 5'h00 && SigRegWrite\_WB && WReg\_WB == Rt\_MEM) ? 1'b1 : 1'b0;

当以上的转发路径中，若在需求寄存器的值的时候新值还没有产生，便无法直接用转发解决冲突，需要通过暂停处理。相隔一级流水线的路径一定会满足转发条件，故可能产生暂停的路径一共三条：EX🡪ID, MEM🡪ID, MEM🡪EX。分别对应：Branch|Jump 🡨 AnyInstruction, Branch 🡨 Load, AnyInstruction 🡨 Load。本设计中在控制器生成是否需要使用在某级使用某寄存器的信号（如需要在ID级使用Rs寄存器的信号为BSigUseRs\_ID），在冲突控制单元中在指令进入ID级的时候判断是否暂停。

处理暂停的代码如下：

assign \_Block\_ID\_Rs = BSigUseRs\_ID && (

(SigMemtoReg\_MEM && WReg\_MEM != 5'h00 && Rs\_ID == WReg\_MEM )

|| (SigRegWrite\_EX && WReg\_EX != 5'h00 && Rs\_ID == WReg\_EX ));

assign \_Block\_ID\_Rt = BSigUseRt\_ID && (

(SigMemtoReg\_MEM && WReg\_MEM != 5'h00 && Rt\_ID == WReg\_MEM )

|| (SigRegWrite\_EX && WReg\_EX != 5'h00 && Rt\_ID == WReg\_EX ));

assign \_Block\_EX\_Rs =

BSigUseRs\_EX && SigMemtoReg\_EX && WReg\_EX!=5'h00 && Rs\_ID==WReg\_EX;

assign \_Block\_EX\_Rt =

BSigUseRt\_EX && SigMemtoReg\_EX && WReg\_EX != 5'h00 && Rt\_ID == WReg\_EX;

assign HSigBlock = \_Block\_ID\_Rs | \_Block\_ID\_Rt | \_Block\_EX\_Rs | \_Block\_EX\_Rt;