P5 Verilog 流水线 CPU 设计文档

19373490-吴昊天

一、CPU 设计方案综述

（一）总体设计概述

本 CPU 为 Verilog 设计的流水线处理器，支持的指令集为{addu,subu,ori,sw,lw,beq,lui,j,jal,jr,nop}。整体设计分为 5 级，对应 F、D、E、M、W，并设计转发和暂停单元。顶层有效信号为时钟信号 clk 和复位信号 reset，复位为同步复位。

（二）关键模块定义

1、F 级

内部包括 PC（程序计数器）、IM（指令存储器）等相关逻辑。

（1）PC 用 32 位寄存器实现，具有同步复位功能，复位至 0x0000_3000.

表 1、PC 描述

功能描述	32 位可复位寄存器，reset 高电平有效。 同步复位至 0x0000_3000	
信号名	方向	描述
Clk	I	MIPS-C 处理器时钟
Reset	I	复位信号，同步复位
PCEn	I	PC 使能端
NPC[31:0]	I	32 位次地址输入
PC[32:0]	O	32 位输出，当前地址

（2）IM 用寄存器阵列实现，大小为 32bit×1024 字。初始读取指令时以系统任务$readmemh 读取 code.txt 的十六进制机器码。

表 2、IM 描述

功能描述	寄存器阵列，存储指令。以$readmemh 读取 code.txt 大小为 32×1024 字

信号名	方向	描述
addr[31:0]	I	取值地址，取 addr[11:2]
Instr[31:0]	O	32 为读出的指令

2、D 级

（1）IF/ID 寄存器，为连接 F 级和 D 级的寄存器。

信号名	方向	描述
Clk	I	时钟信号
Reset	I	复位信号
DregEn	I	D 级寄存器使能端
Instr_F[31:0]	I	F 级指令
PC_F[31:0]	I	F 级 PC
PC4F[31:0]	I	F 级 PC+4
Instr _D[31:0]	O	D 级指令
PC_D[31:0]	O	D 级 PC
PC4D[31:0]	O	D 级 PC+4

（2）GRF（通用寄存器组，也称为寄存器文件、寄存器堆）。具有写使能的寄存器阵列，大小为 32bit×32 字，具有同步复位功能。

表 4、GRF 描述

功能描述	寄存器组，reset 同步复位	
信号名	方向	描述
Clk	I	时钟信号
Reset	I	复位信号，同步复位
RFWr	I	使能信号，时钟上升沿时为 1 有效
A1[4:0]	I	第一个读出寄存器编号
A2[4:0]	I	第二个读出寄存器编号
A3[4:0]	I	回写寄存器编号
WData[31:0]	I	回写寄存器的数据值

WPC[31:0]	I	应题目输出要求，当前执行指令
RD1[31:0]	O	第一个读出的寄存器值
RD2[31:0]	O	第二个读出的寄存器值

（3）EXT 扩展

表 7、EXT 描述

信号名	方向	描述
Imm[15:0]	I	16 位立即数
EXTOp[1:0]	I	EXT 功能选择信号 00：符号扩展 01：无符号扩展 10：加载立即数至高位 11：保留
EXTout[31:0]	O	EXT 输出数据

（4）NPC 相当于一个组合逻辑，输入 PC 以及相关辅助信号，根据控制信号 NPCOp 选择计算方式，输出次地址 NPC。

表 3、NPC 描述

信号名	方向	描述
PC[31:0]	I	当前地址
Instr[31:0]	I	32 位指令
NPCOp[2:0]	I	NPC 地址选择： 000：PC+4（计算顺序地址） 001：计算 beq 地址 010：计算 j&jal 地址 011：计算 jr 地址
zero	I	rs 和 rt 相等判断 1：相等 0：不相等

RA[31:0]	I	32 位输入地址
NPC[31:0]	O	32 位次地址

3、E 级

（1）ID/IE 寄存器，为连接 D 级和 E 级的寄存器

信号名	方向	描述
Clk	I	时钟信号
Reset	I	复位信号
EClr	I	暂停清楚信号
Instr_D[31:0]	I	D 级流水指令
PC_D[31:0]	I	D 级流水 PC
A3D	I	D 级流水回写地址
RD1D[31:0]	I	D 级寄存器读出值 1
RD2D[31:0]	I	D 级寄存器读出值 2
EXToutD[31:0]	I	EXT 输出 32 位结果
Instr_E[31:0]	O	E 级指令
PC_E[31:0]	O	E 级 PC
A3E	O	E 级回写地址
RD1E[31:0]	O	E 级输入寄存器结果 1
RD2E[31:0]	O	E 级输入寄存器结果 2
EXToutE[31:0]	O	E 级输入 EXT 结果

（2）ALU 计算单元

ALU（算术逻辑单元）。提供 32 位加、减、或运算及比较大小功能，控制信号为 ALUOp。

表 5、ALU 描述

信号名	方向	描述
A[31:0]	I	第一个操作数
B[31:0]	I	第二个操作数

ALUOp[2:0]	I	ALU 功能选择 000：A+B 001：A-B 010：A or B 011：SLT 其余保留
ALUout[31:0]	O	32 位计算结果

4、M 级

（1）IE/IM 寄存器，为连接 E 级和 M 级的寄存器

信号名	方向	描述
Clk	I	时钟信号
Reset	I	复位信号
Instr_E[31:0]	I	E 级流水指令
PC_E[31:0]	I	E 级流水 PC
A3E	I	E 级流水回写地址
RD2E[31:0]	I	E 级输入值 2
ALUoutE[31:0]	I	E 级 ALU 计算结果
Instr_M[31:0]	O	M 级指令
PC_M[31:0]	O	M 级 PC
A3M	O	M 级回写地址
ALUout_M[31:0]	O	M 级 ALU 计算结果 （DM 写入地址）
RD2M[31:0]	O	M 级输入值 2 （DM 写入数据）

（2）DM（数据存储器）由寄存器阵列表示。Reset 同步复位，写使能 DMWr 为 1 时可以写入。

表 6、DM 描述

功能描述	寄存器阵列，存储数据。Reset 同步复位 以$display 按要求输出，大小为 32×1024 字	
信号名	方向	描述
Clk	I	时钟信号
Reset	I	同步复位信号
addr[31:0]	I	访问地址，取 addr[11:2]
WPC[31:0]	I	按要求输出当前 PC 地址
DMWr	I	DM 写使能信号
MemWrite[31:0]	I	DM 写入数据
MemRead[31:0]	I	DM 读出数据

5、W 级

（1）IM/IW 级寄存器，为连接 M 级和 W 级的寄存器

信号名	方向	描述
Clk	I	时钟信号
Reset	I	复位信号
Instr_M[31:0]	I	M 级流水指令
PC_M[31:0]	I	M 级流水 PC
A3M	I	M 级流水回写地址
ALUout_M[31:0]	I	M 级流水 ALU 计算结果
DMreadM[31:0]	I	M 级 DM 读出数据
Instr_W[31:0]	O	W 级指令
PC_W[31:0]	O	W 级 PC
A3W	O	W 级回写地址
ALUout_W[31:0]	O	W 级 ALU 计算结果
DMreadW[31:0]	O	W 级 DM 读出数据

三、控制信号单元

本 CPU 中控制信号采用分布式译码，将执行指令流水，在需要时再分别译

码，译码逻辑和单周期译码逻辑完全相同。

共有 8 个控制信号如下表。

信号名	描述
NPCOp[2:0]	NPC 地址选择： 000：PC+4（计算顺序地址） 001：计算 beq 地址 010：计算 jal 地址 011：计算 jr 地址
M1Sel[1:0]	回写寄存器选择：（写哪里） 00：回写 rt 01：回写 rd 10：回写$ra 11：保留
M2Sel[1:0]	回写数据选择：（写什么） 00：ALU 计算所得数据 01：DM 读出数据 10：PC+4 11：保留
RFWr	寄存器写使能信号 1：上升沿写入数据 0：写入无效
EXTOp[1:0]	扩展单元功能选择： 00：符号扩展 01：无符号扩展 10：加载立即数至高位 11：保留
M3Sel	ALU 计算数选择： 0: 寄存器读出数据二 1：扩展单元所得立即数

ALUOp[2:0]	ALU 功能选择 000：A+B 001：A-B 010：A or B 011：SLT 其余保留
DMWr	DM 写使能信号： 1：上升沿数据写入 0：写入无效

指令\信号名	NPCOp	M1Sel	M2Sel	RFWr	EXTOp	M3Sel	ALUOp	DMWr
addu	000	01	00	1	00	0	000	0
subu	000	01	00	1	00	0	001	0
ori	000	00	00	1	01	1	010	0
lw	000	00	01	1	00	1	000	0
sw	000	00	00	0	00	1	000	1
beq	001	00	00	0	00	0	001	0
lui	000	00	00	1	10	1	000	0
jal	010	10	10	1	00	0	000	0
j	010	00	00	0	00	0	000	0
jr	011	00	00	0	00	0	000	0

四、转发和暂停单元

转发和暂停分别用 ForwardUnit 和 StallUnit 来处理，分布式译码在每一级译出当前级需求者的读取地址以及供给者的回写地址，还有 Tuse 和 Tnew 等信息，输入到两个单元中处理是否转发及暂停。

	Tuse_D	
指令	rs	rt

addu	1	1
subu	1	1
ori	1	3
lw	1	3
sw	1	2
beq	0	0
lui	3	3
jal	3	3
j	3	3
jr	0	3

指令	Tnew_E
addu	1
subu	1
ori	1
lw	2
sw	0
beq	0
lui	1
jal	1
j	0
jr	0

指令	Tnew_M
addu	0
subu	0
ori	0
lw	1
sw	0

beq	0
lui	0
jal	0
j	0
jr	0

四、测试样例阐述

说明文档附件含 test1-6，其中 test1 测试 cal_r 和 cal_i 之间的冲突，测试结果并比对 mars 结果。

test2 和 test3 测试 beq 和 cal 类冲突，看是否执行了延迟槽中的指令。

test4 和 test5 测试 sw 和 lw 以及 cal、beq 之间的冲突，主要结合 mars 结果中的内存数据来比对是否正确。

Test6 测试 jal 和 jr 是否正确。

五、思考题

1、流水线冒险

（1）在采用本节所述的控制冒险处理方式下，PC 的值应当如何被更新？请从数据通路和控制信号两方面进行说明。

本 CPU 中，PC 输入端设置一个 MUX，一个输入为 addr 每次加 4，另一个输入为 D 级 NPC 计算出的次地址。选择信号在 D 级译码得出，如果是跳转类指令则 PC 输入 NPC 计算出地址，否则迭代加 4。

（2）对于 jal 等需要将指令地址写入寄存器的指令，为什么需要回写 PC+8？

为了支持延迟槽，即无论是否跳转都通过编译调度一条必定执行的指令跟在跳转指令后面，并执行这条指令，如果 jal 还按原来设计存入 PC+4，则无法正确跳转到延迟槽后面的那条指令。

2、数据冒险分析

为什么所有的供给者都是存储了上一级传来的各种数据的流水级寄存器，而不是由 ALU 或者 DM 等部件来提供数据？

因为同一使用寄存器作为转发数据来源可匹配每一周期的时序性，否则假设

ALU 的两个输入端需要后面转发而来的数据，而又直接采用 ALU 计算得出结果，在这个组合逻辑中相当于陷入一定时间的循环而出错。

3、AT 法处理流水线数据冒险

（1）“转发（旁路）机制的构造”中的 Thinking 1-4。

Thinking1：如果不采用已经转发过的数据，而采用上一级中的原始数据，那么计算结果依旧是错误的。

比如 `addu $t0,$t1,$t2`

```
subu $t3,$t4,$t0
ori $t5,$t3,0xffff
```

这里 `t3` 计算结果需要转发 `t0` 获得，`t5` 的结果在此基础上需要转发 `t3` 获得，如果没有使用转发过的数据就会出错。

Thinking2：我们为什么要对 GPR 采用内部转发机制？如果不采用内部转发机制，我们要怎样才能解决这种情况下的转发需求呢？

使用内部转发可以省去 W 级向前级的转发，否则对于 D 级和 E 级的转发需要在转发单元中添加来自 W 级的数据。

Thinking3：为什么 0 号寄存器需要特殊处理？因为 0 号寄存器的值始终为零，而转发中如果前序存在向 0 号寄存器中写入非零值的指令，转发时不加以特殊判断，就会“矫枉过正”将非零值转发过来发生错误。

Thinking4：什么是“最新产生的数据”？即本条指令前序最新的指令产生的数据，从转发位点向后的寄存器，优先级依次降低。

（2）在 AT 方法讨论转发条件的时候，只提到了“供给者需求者的 A 相同，且不为 0”，但在 CPU 写入 GRF 的时候，是有一个 we 信号来控制是否要写入的。为何在 AT 方法中不需要特判 we 呢？为了用且仅用 A 和 T 完成转发，在翻译出 A 的时候，要结合 we 做什么操作呢？

对于不写入 GRF 的指令，即不会产生供给数据的指令，生成 A 信号时，对于 we 为零的指令，将它在本级的 Dst 地址置为 0，这样就可以避免出现转发错误。

4、在线测试相关说明

首先进行功能测试，每两条指令之间插入 4 个 nop，检验基本部件和无转发数据通路有无问题。然后将指令分为 cal_r，cal_i，load，store，b，j 这几种，两两构

造测试样例进行冲突测试。

再根据 Tuse 和 Tnew 的表格，反向构造样例。