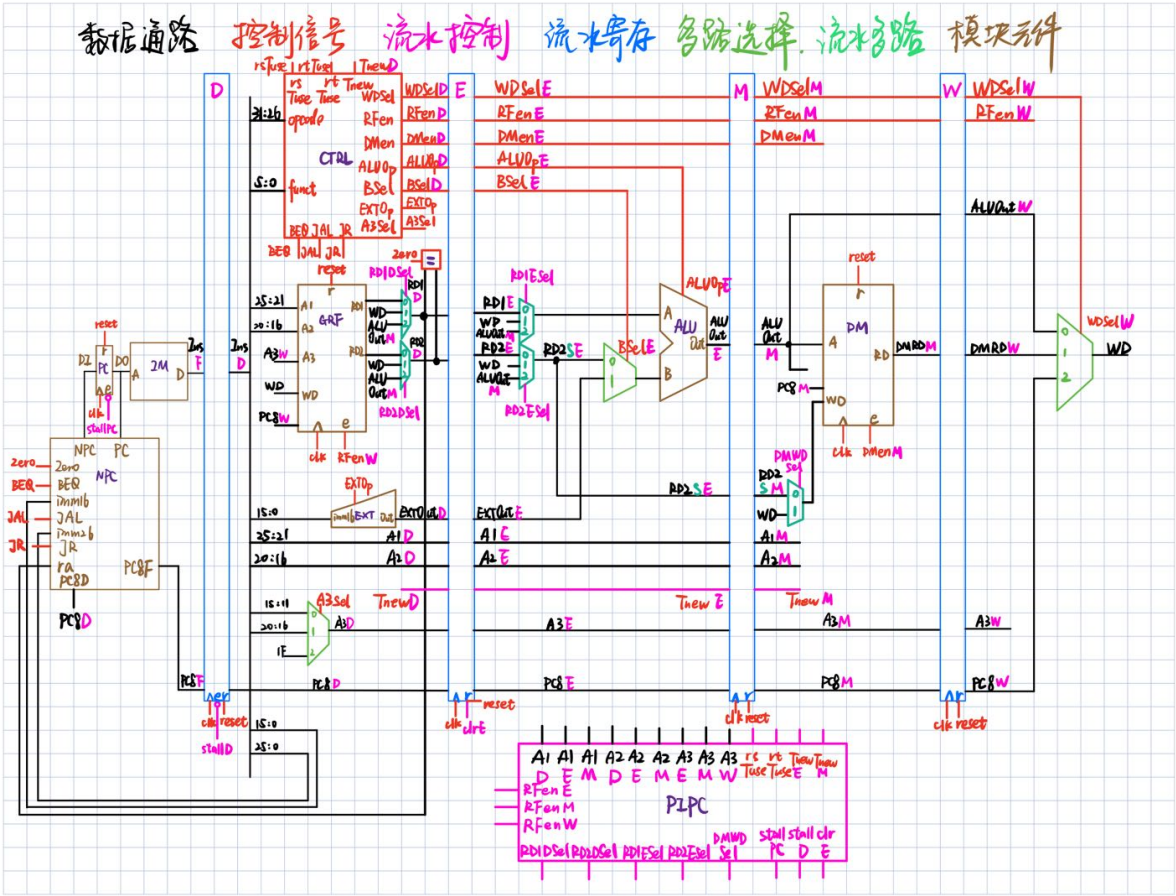
P5 流水线CPU设计

一、设计草稿

1.电路图与模块建模



2.主控制器指令真值表

指令	opcode	funct	BEQ	JAL	JR	A3Sel	EXTOp	BSel	ALUOp	DMen	RFen	WDSel	rsTuse	rtTuse	Tnew	指令
add	000000	100000	0	0	0	00	xx	0	000	0	1	00	1	1	2	add
sub	000000	100010	0	0	0	00	xx	0	001	0	1	00	1	1	2	sub
ori	001101	xxxxxx	0	0	0	01	00	1	010	0	1	00	1	x(5)	2	ori
lw	100011	xxxxxx	0	0	0	01	01	1	011	0	1	01	1	x(5)	3	lw
sw	101011	xxxxxx	0	0	0	xx	01	1	100	1	0	xx	1	2	0	sw
lui	001111	xxxxxx	0	0	0	01	10	1	101	0	1	00	1	x(5)	2	lui
nop	000000	000000	0	0	0	xx	xx	0	xxx	0	0	xx	x(5)	x(5)	0	nop
beq	000100	xxxxxx	1	0	0	xx	xx	0	xxx	0	0	xx	0	0	0	beq
jal	000011	xxxxxx	0	1	0	10	xx	x	xxx	0	1	10	x(5)	x(5)	3	jal
jr	000000	001000	0	0	1	xx	xx	x	xxx	0	0	xx	0	x(5)	0	jr

Tuse：数据用不用？用到哪里？

Tnew：数据产生不产生？还要多久产生？

Tuse<Tnew：暂停

Tuse>=Tnew：转发

二、测试程序

1.Weak

```
.text
ori $1,$0,1
ori $2,$0,4
sw $2,0($0)
lw $3,0($0)
add $4,$1,$3 #4:5
sub $1,$4,$3 #1:1
```

2.Strong

```
.text
ori $1,$0,1
ori $2,$0,4
ori $6,$0,6
beq $3,$6,C
sw $2,0($0)
lw $3,0($0)
add $4,$1,$3
sub $1,$4,$3
jal A
lw $5,0($0)
ori $ra,$0,0x300c
B:
add $3,$3,$1
A:
beq $5,$3,B
nop
sw $3,4($0)
jr $ra
nop
C:
```

三、思考题

1.我们使用提前分支判断的方法尽早产生结果来减少因不确定而带来的开销，但实际上这种方法并非总能提高效率，请从流水线冒险的角度思考其原因并给出一个指令序列的例子。

比较器前置可能会产生与比较的两个数有关的数据相关，产生数据冒险，需要通过暂停和转发来解决问题。若数据相关为计算指令且来自E级，或为读取指令且来自E或M级，则只能通过暂停解决。

示例代码：

```
.text
ori $1,$0,1
add $2,$1,$1
beq $2,$1,end
ori $3,$0,$1
end:
```

2.因为延迟槽的存在，对于 jal 等需要将指令地址写入寄存器的指令，要写回 PC + 8，请思考为什么这样设计?

编译优化，将一条指令与jal等指令无关的指令移动到其下一条位置，减少暂停，提高了效率。这样做让jal等的下一条指令在跳转前执行过了，因此返回时应该返回的是jal等的下两条指令，故PC+8。

3.我们要求大家所有转发数据都来源于流水寄存器而不能是功能部件（如 DM 、 ALU ），请思考为什么?

为了保证每一级流水线内部的延时不影响最终结果，保证数据的稳定性。

4.我们为什么要使用 GPR 内部转发？该如何实现?

让W级产生的数据不用多花一个周期存到GPR中而直接给D级，减少暂停，提高效率。

实现方法：

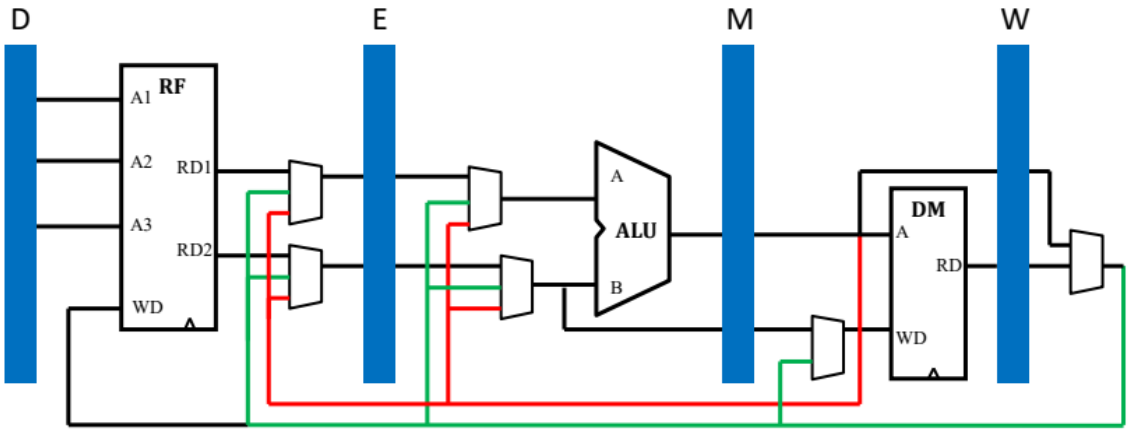
```
assign WD = (WDSelW == 2'b10) ? PC8W :
            (WDSelW == 2'b01) ? DMRDW :
            ALUOutW;
//内部转发信号WD，在W级产生
assign RD1D = (RD1DSel == 2'b10) ? ALUOutM :
            (RD1DSel == 2'b01) ? WD :
            grfRD1;
assign RD2D = (RD2DSel == 2'b10) ? ALUOutM :
            (RD2DSel == 2'b01) ? WD :
            grfRD2;
//产生最终的RD1D、RD2D，通过在D级增加MUX
```

5.我们转发时数据的需求者和供给者可能来源于哪些位置？共有哪些转发数据通路?

转发数据的需求者与供给者：

	M级ALU计算结果	W级回写结果
D级被转发	RS寄存器值 RT寄存器值	RS寄存器值 RT寄存器值
E级被转发	ALU的A ALU的B RT寄存器值	ALU的A ALU的B RT寄存器值
M级被转发		DM的WD

转发数据通路：



6.在课上测试时，我们需要你现场实现新的指令，对于这些新的指令，你可能需要在原有的数据通路上做哪些扩展或修改？提示：你可以对指令进行分类，思考每一类指令可能修改或扩展哪些位置。

a.考虑不同类别指令涉及的不同模块，指令分类有：

（1）算数运算：add、sub

涉及IFUF、GRFD、ALU、GRFW

（2）立即数运算：ori、lui

涉及IFUF、GRFD、EXT、ALU、GRFW

（3）内存访问：lw

涉及IFUF、GRFD、EXT、ALU、DM、GRFW

（4）内存写入：sw

涉及IFUF、GRFD、EXT、ALU、DM

（5）分支、返回：beq、jr

涉及IFUF、GRFD、IFUD

（6）跳转：jal

涉及IFUF、GRFD、IFUD、GRFW

b.针对每条指令涉及的特定模块，考虑信息有：

（1）模块内部实现

（2）模块接口需求

（3）模块接口来源（是否需要流水控制）

（4）控制信号设置（主控制+冒险控制、基础译码+新增扩展）

（5）流水传递寄存

7.简要描述你的译码器架构，并思考该架构的优势以及不足。

采用集中式译码，控制信号驱动型译码方式，把D级后面要用的控制信号进行流水。

优势：减少了后续流水级的逻辑复杂度，译码时代码量少

不足：增加了流水级之间需要传递的信号数量，译码时不容易定位错误

附：P3电路图