

一种朴素的造机方法

以RISC-V的流水线架构为例

挑战

- **流水线设计目标：**

- 正确：任意该指令集构建的程序都能正确运行
- 性能：尽可能使流水线充满

- **挑战：**

- 如何开始构建？面对复杂的概念难以入手。
- 代码量大，debug成本高。
- 多人合作，沟通成本高。

- **方法：需要一种定义清晰，产出可控，从设计到实现全程可追溯的造机方法。**

我们有什么？

- 指令级
 - 指令语义
 - 指令执行过程中对子模块（ALU）的操作
- 数据通路设计图
 - 流水层级
 - 子模块的流水段划分，如：ALU在EXE级
- 流水线的基本原理
 - 流水线时空图
 - 流水线冲突控制方法
 - 转发
 - 暂停

.....

简化问题

- 只关注两个层面：
 - 数据层（机制）
 - 控制层（策略）

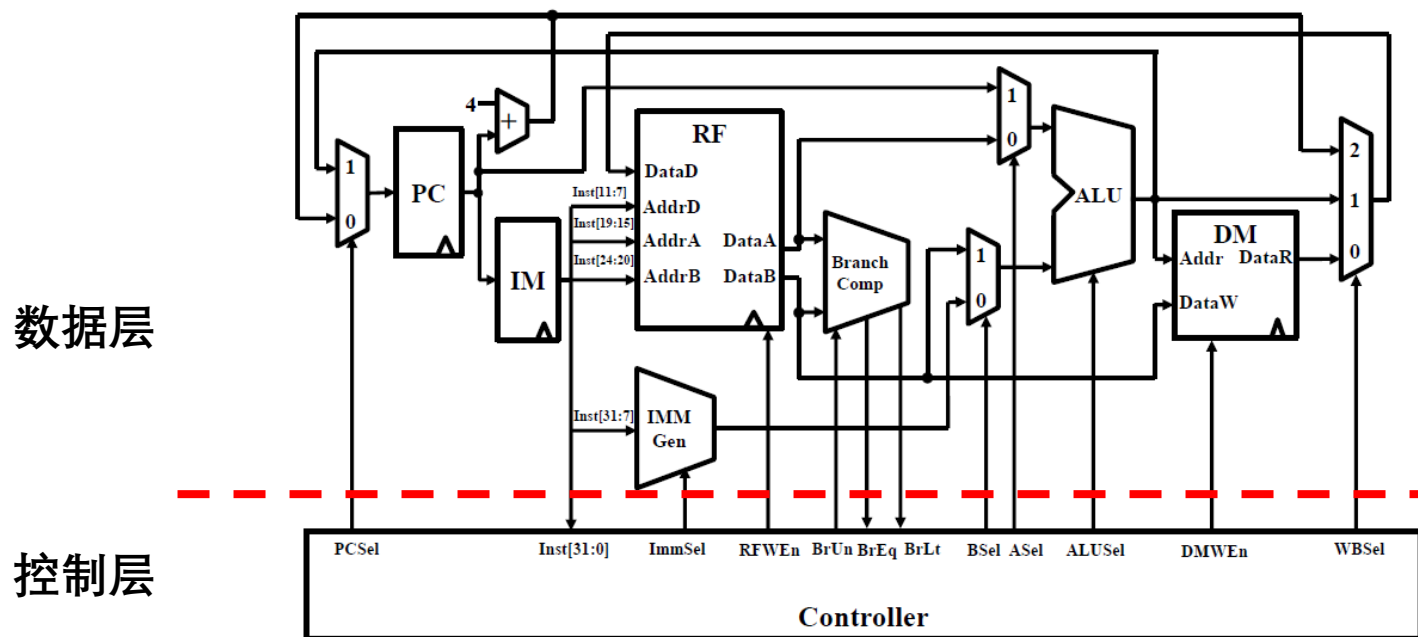
指导思想：机制和策略分离！

流水线方法概述

- S1: 数据通路及控制
- S2: 暂停策略及控制
- S3: 转发策略及控制

数据层

- 数据层：面向指令的**数据流**
 - 仅包含：功能部件、功能部件间的数据连接关系、数据信息的流水寄存器
 - 数据层负责机制问题，并为策略提供控制接口
 - **机制**：数据层提供各种各样的具体功能，如ALU的加减
 - **策略**：如何组合功能实现指令需求（控制信号）
- 思考：
 - 如何使得数据流可以呈现不同的状态？多路选择器
 - 什么是控制接口？多路选择器的控制型号



图片为单周期设计图
流水线更为复杂，但原理相同

从命名开始

名字	位宽	描述	命名原因
A1	5	第1个源寄存器地址	在RF中寻找寄存器的Address
A2	5	第2个源寄存器地址	
A3	5	目的寄存器地址	
V1	32	第1个源寄存器的值	RF中寄存器的Value
V2	32	第2个源寄存器的值	
E32	32	扩展后的立即数	Extend
AO	32	ALU的输出值	ALU Out
DO	32	DM的输出值	DM Out
PC4	32	PC+4	

- “望文生义”
 - 让自己明白
 - 让深夜debug的自己明白
 - 让队友明白
(不分先后)

规划基础数据通路

- 目的
 - 大致清楚CPU中有哪些**部件**
 - 大致清楚不同**流水阶段主要职能**，即清楚主要部件所在的流水阶段
 - 明确各部件的接口，即**输入和输出**
 - 大致清楚各级**流水寄存器**需要保存哪些值（在后续设计中可以查缺补漏）
 - **为形式化的表达准备基础**
- 方法
 - 理论上：参考理论课的方法推导出数据通路
 - 操作上：直接细化理论课数据通路
- 产物
 - 一份部件**接口定义明确**的流水线设计图

形式化表达数据通路的关键

- 思考

- 什么是数据通路的关键元素？
- 如何没有损失地表达数据通路？
- 如何表示数据通路能够方便代码产出？

- 感性的思考过程

- 数据通路 == “部件A的输出端1->部件B的输入端1”
- 若以“->”（**连接关系**）为主体形式化.接口1冗杂，没有什么启发
- 考虑以**部件为主体**形式化描述数据通路
- 部件的某一个接口可能有多个输入，考虑添加多路选择器给控制层

- 形式化表达

- 必要元素：部件，输入端，输出端，输出端->输入端
- 构建关系表

流水线数据通路描述表

- 纵向标签
 - 部件的输入端
 - 若输入端只有一个则省略（如：PC）
- 横向标签
 - 指令名
- 表格内容(对于表格中的每一列)
 - 某条指令在各流水段执行时
 - 各个部件的**输入端数据来源于哪一输出端**
- 思考
 - 这样的描述是否能完备地表示数据通路？
 - 这样的描述是否能方便地支持优化关键路径？

部件的输入端		某条指令
PC		
IM		
ADD4		
D	IR	
	PC4	
RF	A1	
	A2	
EXT		
NPC	PC4	
	IR	
CMP	D1	
	D2	
E	V1	
	V2	
	A1	
	A2	
	A3	
	EXT	
	PC4	
ALU	A	
	B	
M	V2	
	A2	
	A0	
	A3	
	PC4	
DM	A	
	WD	
W	A3	
	PC4	
	A0	
	DR	
RF	A3	
	WD	

根据RTL填写数据通路描述表

- 以LW指令为例填写数据通路描述表

Stage	RTL	Func Control Signal
IF	$IR@D \leftarrow IM[PC]$	
	$PC \leftarrow ADD4(PC)$	
ID	$V1@E \leftarrow RF[IR[rs1]@D]$	ImmSel:imm12
	$E32@E \leftarrow EXT(IR[31:20]@D)$	
	$A3@E \leftarrow IR[rd]@D$	
	$A2@E \leftarrow IR[rs1]@D$	
EXE	$AO@M \leftarrow ALU(V1@E, E32@E)$	ALUSel:add
	$A3@M \leftarrow A3@E$	
MEM	$DR@W \leftarrow DM[AO@M]$	
	$A3@W \leftarrow A3@M$	
WB	$RF[A3@W] \leftarrow DR@W$	

为什么要传递A1到E级流水为止？



部件的输入端		LW
PC		ADD4
IM		PC
ADD4		PC
D	IR	IM
	PC4	ADD4
RF	A1	$IR[rs1]@D$
	A2	
EXT		$IR[31:20]@D$
NPC	PC4	
	IR	
CMP	D1	
	D2	
E	V1	$RF.RD1$
	V2	
	A1	$IR[rs1]@D$
	A2	
	A3	$IR[rd]@D$
	EXT	EXT
ALU	PC4	
	A	$V1@E$
M	B	$EXT@E$
	V2	
	A2	
	A0	ALU
	A3	$A3@E$
DM	PC4	
	A	$AO@M$
W	WD	
	A3	$A3@M$
	PC4	
	A0	
RF	DR	DM
	A3	$A3@W$
	WD	$DR@W$

综合无转发数据通路

对于一个输入端有多个数据来源则在该输入端之前添加多路选择器（如：**MPC**）。

部件的输入端		LW	BEQ	ADD	JAL	MUX	0	1	2
PC		ADD4	NPC	ADD4	NPC	MPC	ADD4	NPC	
IM		PC	PC	PC	PC	PC			
ADD4		PC	PC	PC	PC	PC			
D	IR	IM	IM	IM	IM	IM			
	PC4	ADD4	ADD4	ADD4	ADD4	ADD4			
RF	A1	IR[rs1]@D	IR[rs1]@D	IR[rs1]@D		IR[rs1]@D			
	A2		IR[rs2]@D	IR[rs2]@D		IR[rs2]@D			
EXT		IR[31:20]@D	IR[immBEQ]@D		IR[immJAL]@D	MEXT	IR[31:20]@D	IR[immBEQ]@D	IR[immJAL]@D
NPC	PC		(PC4@D) - 4		(PC4@D) - 4	(PC4@D) - 4			
	IR		IR@D		IR@D	IR@D			
CMP	D1		RF.RD1			RF.RD1			
	D2		RF.RD2			RF.RD2			
E	V1	RF.RD1		RF.RD1		RF.RD1			
	V2			RF.RD2		RF.RD2			
	A1	IR[rs1]@D		IR[rs1]@D		IR[rs1]@D			
	A2			IR[rs2]@D		IR[rs2]@D			
	A3	IR[rd]@D		IR[rd]@D	IR[rd]@D	IR[rd]@D			
	EXT	EXT				EXT			
	PC4				PC4@D	PC4@D			
ALU	A	V1@E		V1@E		V1@E			
	B	EXT@E		V2@E		MALUB	EXT@E	V2@E	
M	V2								
	A2								
	A0	ALU		ALU		ALU			
	A3	A3@E		A3@E	A3@E	A3@E			
	PC4				PC4@E	PC4@E			
DM	A	AO@M				AO@M			
	WD								
W	A3	A3@M		A3@M	A3@M	A3@M			
	PC4				PC4@M	PC4@M			
	A0			AO@M		AO@M			
	DR	DM				DM			
RF	A3	A3@W		A3@W	A3@W	A3@W			
	WD	DR@W		AO@W	PC4@W	MRFWD	DR@W	AO@W	PC4@W

构造功能MUX控制信号表达式

部件的输入端	LW	BEQ	ADD	JAL	MUX	0	1	2
PC	ADD4	NPC	ADD4	NPC	MPC	ADD4	NPC	

$PC_{sel} = (LW+ADD) ? \text{`ADD4} :$
 $(BEQ+JAL) ? \text{`NPC} :$
.....

宏定义!

流水线方法概述

- S1: 数据通路及控制
- S2: 暂停策略及控制
- S3: 转发策略及控制

根据Tuse和Tnew构造策略矩阵

- Tuse(time-to-use): 指令进入D级后, 其后的某个功能部件再经过多少周期就必须使用寄存器值
 - 特点1: Tuse是静态值
 - 特点2: 同一条指令可以有多个不同的Tuse
- Tnew(time-to-new): 位于E级及其后各级的指令, 再经过多少周期能够产生要写入寄存器的结果
 - 特点1: 动态值, 随着指令的流动, 该值在不断减小, 直至0
 - 特点2: 一条指令可以有多个不同的Tnew
 - 例如, 计算类指令的Tnew为1或0
 - 1: 指令位于E级, ALU正在计算
 - 0: 指令位于M级或W级, 结果已经存储在相应级

根据Tuse和Tnew构造策略矩阵

$T_{new} \backslash T_{use}$		E			M			W		
		ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
		1	2	0	0	1	0	0	0	0
0		S	S	F	F	S	F	F	F	F
1		F	S	F	F	F	F	F	F	F

`Stall_RS0_E1` = `Tuse_RS0` & (`Res_E` == ``ALU`) & (``OPCODE` == `A3_E`)

`Stall_RS0_E2` = `Tuse_RS0` & (`Res_E` == ``DM`) & (``OPCODE` == `A3_E`)

.....

`Stall_RS0_M1` = `Tuse_RS0` & (`Res_M` == ``DM`) & (``OPCODE` == `A3_M`)

流水线方法概述

- S1: 数据通路及控制
- S2: 暂停策略及控制
- S3: 转发策略及控制

转发控制的基本分析方法

部件的输入端		LW	BEQ	ADD	JAL	MUX	0	1	2
PC		ADD4	NPC	ADD4	NPC	MPC	ADD4	NPC	MFPC
IM		PC	PC	PC	PC	PC			
ADD4		PC	PC	PC	PC	PC			
D	IR	IM	IM	IM	IM	IM			
	PC4	ADD4	ADD4	ADD4	ADD4	ADD4			
RF	A1	IR[rs1]@D	IR[rs1]@D	IR[rs1]@D		IR[rs1]@D			
	A2		IR[rs2]@D	IR[rs2]@D		IR[rs2]@D			
EXT		IR[31:20]@D	IR[immBEQ]@D		IR[immJAL]@D	MEXT	IR[31:20]@D	IR[immBEQ]@D	IR[immJAL]@D
NPC	PC		(PC4@D) - 4		(PC4@D) - 4	(PC4@D) - 4			
	IR		IR@D		IR@D	IR@D			
CMP	D1		RF.RD1			MFCMPRD1			
	D2		RF.RD2			MFCMPRD2			
E	V1	RF.RD1		RF.RD1		RF.RD1			
	V2			RF.RD2		RF.RD2			
	A1	IR[rs1]@D		IR[rs1]@D		IR[rs1]@D			
	A2			IR[rs2]@D		IR[rs2]@D			
	A3	IR[rd]@D		IR[rd]@D	IR[rd]@D	IR[rd]@D			
	EXT	EXT				EXT			
ALU	PC4				PC4@D	PC4@D			
	A	V1@E		V1@E		MFALUA			
M	B	EXT@E		V2@E		MALUB	EXT@E	MFALUB	
	V2					MFV2M			
DM	A2								
	AO	ALU		ALU		ALU			
	A3	A3@E		A3@E	A3@E	A3@E			
	PC4				PC4@E	PC4@E			
W	A	AO@M				AO@M			
	WD					MFDMD			
	A3	A3@M		A3@M	A3@M	A3@M			
	PC4				PC4@M	PC4@M			
RF	AO			AO@M		AO@M			
	DR	DM				DM			
	A3	A3@W		A3@W	A3@W	A3@W			
	WD	DR@W		AO@W	PC4@W	MRFWD	DR@W	AO@W	PC4@W

	0	1	2	3
MFALUBE	V2@E	DR@W	AO@W	AO@M

- 0. 有几个转发点？
- 1. 转发优先级？
- 2. 0号寄存器的转发？

```
`define M2E_ALU 3
`define W2E_ALU 2
`define W2E_DM 1
.....
FALUBE = ((A2_E==A3_M) & (Res_M=='ALU) ? `M2E_ALU :
          ((A2_E==A3_W) & (Res_W=='ALU) ? `W2E_ALU :
          ((A2_E==A3_W) & (Res_W=='DM) ? `W2E_DM :
          0
```

综合控制层

- 控制层：面向指令的控制流
- 控制器架构
 - 主控制器
 - 运行时所需参数编码器
 - 冲突检测与控制器：将D级指令信息与E、M、W级指令信息进行比较
 - 转发控制器：控制各转发多路选择器
 - 暂停控制器：控制流水线是否暂停
- 思考
 - 为什么需要分为3个控制器？如何划分更“好”？“好”的标准是什么？
 - 为什么在D级进行比较？
 - 暂停流水时如何确定寄存器应保持原值或写入新值？

其他注意事项

模块实例化注意事项

- 模块间的信号应显式定义后再使用

```
module pipeline( clk, rst ) ;
```

```
    wire [31:0] instr ;
```

```
    wire [4:0] A3 ;
```

```
    wire RFWr, DMWr ;
```

```
    wire PCEn, DEn, Eclr ;
```


```
    .....
```

```
    datapath U_datapath( ..., instr, A3, RFWr, ... ) ;
```

```
    controller U_controller( ..., instr, A3, RFWr, ... ) ;
```

```
endmodule
```

注意信号名在各模块中的**正确顺序**
或者**按名传输**



MUX设计

- 可以在一个设计文件中定义多种MUX

```
// multiplexor.v
// 2-1 module
MUX2to1( in0, in1, sel, out ) ;
...
endmodule
// 3-1 module
MUX3to1( in0, in1, in2, sel, out ) ;
...
endmodule
// 4-1 module
MUX4to1( in0, in1, in2, in3, sel, out ) ;
...
endmodule
```

- 为了灵活支持多种位宽的MUX, 应使用parameter定义位宽

```
module MUX3to1( in0, in1, in2, sel, out ) ;

    parameter WIDTH_DATA = 32 ;

    input [WIDTH_DATA:1] in0;
    ...
    output [WIDTH_DATA:1] out;
    assign out = (sel==2'b10) ? in2 :
(sel==2'b01) ? in1 : in0 ;

endmodule
```

- 实例化时, 用defparam设置实际参数

```
MUX3to1 MRFA3(ir[15:11], ir[20:16], 5'd31,
A3Sel, A3) ;
defparam MRFA3.WIDTH_DATA = 5 ;
```

流水线寄存器实例化

- 流水线寄存器有多种实现方式，其中比较直观的是采用**实例化方式**。

```
module datapath( ..... ) ;
...
wire [31:0] ..., AO_M, AO_W, DR_W ;

// E级功能部件
ALU U_ALU(..., AO, ...) ;

// M级流水寄存器
R_Pipe PIPE_AO_M(AO, AO_M, clk) ;
defparam PIPE_AO_M.WIDTH_DATA = 32 ;

// M级功能部件
DM U_DM(..., AO_M, DMO, ... ) ;

// W级流水寄存器
R_Pipe PIPE_AO_W(DMO, DR_W, clk) ;
defparam PIPE_AO_W.WIDTH_DATA = 32 ;
```

实例化方式实现流水线寄存器，可以使得代码结构布局尽可能与设计布局保持一致。

Thanks