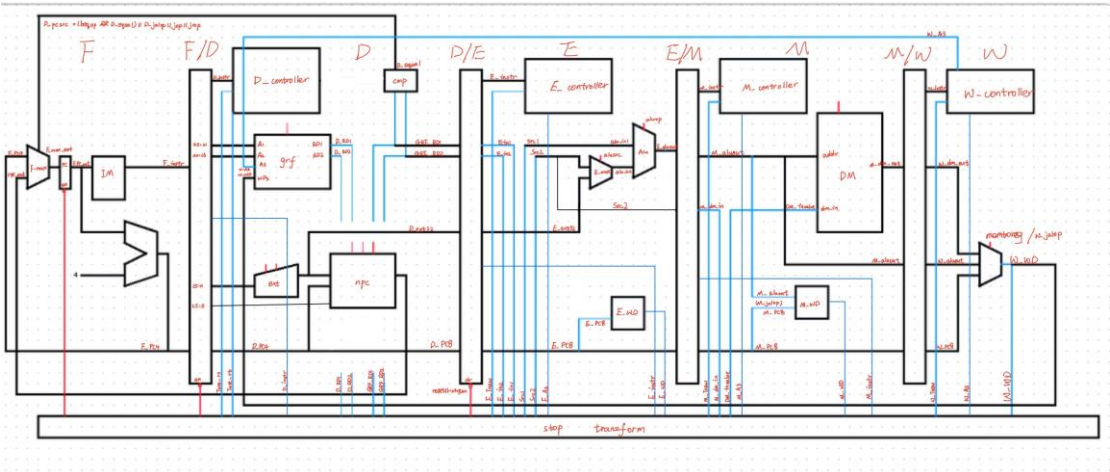
流水线 CPU 设计报告

一、数据通路设计

1、



二、模块规格

1. IFU

（1）端口说明

端口名	方向	描述
CLK	I	时钟信号
RESET	I	异步复位信号
PC_in	I	原 PC 信号
PC_out	O	PC+4 后的信号
INSTRUCTION[31:0]	O	输出 32 位指令

（2）功能定义

序号	功能名称	功能描述
1	复位	当 RESET 信号为 1 时，对保存 PC 地址的寄存器进行异步清零操作。
2	取指	以 PC 的值为地址，从 IM 中取出对应指令

2. GRF

（1）端口说明

端口名	方向	描述
CLK	I	时钟信号
RESET	I	异步复位信号
RegWrite	I	写入使能
Read_Reg1	I	读取 Rs
Read_Reg2	I	读取 Rt
Write_Reg	I	读取存储地址
Write_Data	I	读取存储数据
Read_Data	O	Rs 输出
Read_Data2	O	Rt 输出

（2）功能定义

序号	功能名称	功能描述
1	复位	当 RESET 信号为 1 时，对 GRF 中所有寄存器进行异步清零操作。
2	写入	Regwrite=1 时，将数据写入
3	读取	将 Rs，Rt 输出

3、ALU

（1）端口说明

端口名	方向	描述
Input1[31:0]	I	输入 1
Input2[31:0]	I	输入 2
ALUOp[31:0]	I	选择计算功能
Equ	O	输入是否相等
Ans[31:0]	O	输出

（2）功能定义

序号	功能名称	功能描述
1	逻辑运算	ALUOp=00，C=A&B ALUOp=01，C=A|B ALUOp=10，C=A+B ALUOp=11，C=A-B
2	相等判断	若 A=B，则 Equ=1 若 A≠B，则 Equ=0

4、EXT

（1）端口说明

端口名	方向	描述
Imm16[15:0]	I	16 位立即数输入
ExtOp	I	Ext 功能选择
LuiOp	I	Lui 功能选择
Ext32	O	32 位数输出

（2）功能定义

序号	功能名称	功能描述
1	符号拓展	ExOp=0，无符号拓展 ExOp=1，有符号拓展
2	Lui 指令功能	LuiOp=1，将立即数添加至 32 位数高位，低位补零

5、DM

（1）端口说明

端口名	方向	描述
CLK	I	时钟信号
RESET	I	异步复位信号

Addr[5:0]	I	地址读入
Input[31:0]	I	数据读入
MemWrite	I	写入使能
MemRead	I	读取使能
Data[31:0]	O	32 位数据输出

（2）功能定义

序号	功能名称	功能描述
1	写入	当写入使能=1 时，将 Input 写入 Addr 对应的寄存器
2	读取	当读取使能=1 时，将 Addr 对应的寄存器的值输出

6、Controller

（1）端口说明

端口名	方向	描述
OP	I	Operation
FUN	I	Function
Regdst	O	GRF 写入地址 MUX 控制 0：写入至 Rt 1：写入至 Rd
ALUsrc	O	ALU 输入 MUX 控制 0：input2 = Rt 1：input2 = ext32
MemtoReg	O	GRF 写入数据 MUX 控制 0：写入 DM 的输出 1：直接输出 ALU 结果
RegWrite	O	GRF 写入使能
MemWrite	O	DM 写入使能
MemRead	O	DM 输出使能

ExtOp	O	Ext 使能信号
LuiOp	O	Lui 使能信号
BeqOp	O	Beq 使能信号
ALUOp[1:0]	O	ALU 功能选择信号

（2）功能定义

序号	功能名称	功能描述
1	控制	根据 Op 和 Fun 输出相应指令与 MUX 的控制信号

7、Pipeline_Register

（1）端口说明

端口名	方向	描述
CLK	I	时钟信号
RESET	I	异步复位信号
En	I	写入使能
In	I	数据读入
Out	O	数据读出

三、控制器设计

1. 端口说明

端口名	方向	描述
OP	I	Operation
FUN	I	Function
Regdst	O	GRF 写入地址 MUX 控制 0：写入至 Rt 1：写入至 Rd
ALUsrc	O	ALU 输入 MUX 控制

		0：input2 = Rt 1：input2 = ext32
MemtoReg	O	GRF 写入数据 MUX 控制 0：写入 DM 的输出 1：直接输出 ALU 结果
RegWrite	O	GRF 写入使能
MemWrite	O	DM 写入使能
MemRead	O	DM 输出使能
ExtOp	O	Ext 使能信号
LuiOp	O	Lui 使能信号
BeqOp	O	Beq 使能信号
ALUOp[1:0]	O	ALU 功能选择信号

2. 指令真值表

Instr	addu	subu	ori	lw	sw	beq	lui
Op	000000	000000	001101	100011	101011	000100	001111
Func	100001	100011	NA	NA	NA	NA	NA
Regdst	1	1	0	0	X	X	0
ALUsrc	0	0	1	1	1	0	1
MemtoReg	1	1	1	0	X	X	1
RegWrite	1	1	1	1	0	0	1

MemWrite	0	0	0	0	1	0	0
MemRead	0	0	0	1	0	0	0
ExtOp	X	X	0	1	1	0	X
LuiOp	X	X	0	0	0	0	1
BeqOp	X	X	X	X	X	1	X
ALUOp	10	11	01	10	10	XX	10

四、转发模块设计

1.端口说明

端口名	方向
D_instr	I
E_instr	I
M_instr	I
E_Tnew	I
E_WD	I
M_WD	I
W_WD	I
E_A3	I
M_A3	I
W_A3	I
D_RD1	I
D_RD2	I
E_in1	I
E_in2	I
M_aluout	I

M_dm_in	I
GRF_RD1	O
GRF_RD2	O
Src1	O
Src2	O
DM_invalue	O

2. 算法逻辑

```
Assign GRF_RD1 =
(D_instr[`rs] == E_A3 && E_Tnew == 0 && D_instr[`rs] != 0) ? E_WD :
(D_instr[`rs] == M_A3 && M_Tnew == 0 && D_instr[`rs] != 0) ? M_WD :
(D_instr[`rs] == W_A3 && W_Tnew == 0 && D_instr[`rs] != 0) ? W_WD :
                                                          D_RD1;
```

如对于 GRF_RD1 的转发，通过判断 E\M\W 级指令对应的写入寄存器地址是否与 D 级需求的寄存器地址相同且不为 0，并判断 E\M\W 级的 Tnew 是否为 0，即结果是否已经传入流水线寄存器，来确定转发的数据与接口。

四、暂停模块设计

1. 端口说明

端口名	方向
Tuse_rs	I
Tuse_rt	I
E_Tnew	I
M_Tnew	I
W_Tnew	I
Stop_en	O

2. 算法逻辑

```
Assign stop_en = (Tuse_rs < E_Tnew) || (Tuse_rs < M_Tnew) || (Tuse_rs < W_Tnew) ||
                 (Tuse_rt < E_Tnew) || (Tuse_rt < M_Tnew) || (Tuse_rt < W_Tnew) ;
```

直接根据 E\M\W 级的 Tnew 与 D 级的 Tuse 进行大小比较，如果 Tuse<Tnew,

则立刻暂停。

五、Tuse、Tnew 表格

	Tuse/D级开始		注：没有使用的直接记为3，初始化同 / 一段一段数 add：D到E ->1
	rs/offset	rt	指令描述
addu	1	1	GPR[rd] = GPR[rs] + GPR[rt]
subu	1	1	GPR[rd] = GPR[rs] - GPR[rt]
ori	1	3	GPR[rt] = GPR[rs] OR zero_extend(immediate)
lw	1	3	GPR[rt] = memory[GPR[rs]+offset]
sw	1	2	memory[Addr] = GPR[rs] + sign_ext(offset)
lui	3	3	GPR[rt] = immediate 16 || {16{1`b0}}
j	3	3	PC = PC31..28 || instr_in || 00
jal	3	3	GPR[31] = PC + 4
jr	0	3	PC = GPR[rs]
beq	0	0	(GPR[rs] == GPR[rt])

Tnew/一级一级数	从X级之后的流水线写入，则X=0,倒推				无用=0
	部件		D	E	M	W
addu	ALU		2	1	0	0
subu	ALU		2	1	0	0
ori	ALU		2	1	0	0
lw	DM		3	2	1	0
sw	DM		0	0	0	0
lui	D		1	0	0	0
j	PC		0	0	0	0
jal	PC		1	0	0	0
jr	PC		0	0	0	0
beq	PC		0	0	0	0

六、暂停转发测试程序

1.R-R

```
subu $1,$2,$3
addu $4,$1,$2
```

```
subu $1,$2,$3
addu $4,$2,$1
```

```
ori $1,5
addu $4,$2,$1
```

```
ori $2,5
addu $4,$2,$1
```

```
jal loop
ori $1,$0,1
loop:
ori $2,$0,10


ori $1,$0,122
jal loop1
addu $2,$31,$1
loop1:
ori $3,$0,239
jal loop2
addu $4,$3,$31
loop2:
jal loop3
ori $5,$0,23
loop3:
addu $6,$31,$5
jal loop4
ori $7,$0,55
loop4:
addu $8,$7,$31
```

2.R-LW,SW

```
ori $3,$0,4
ori $4,$0,8
sw $3,0($4)
lw $1,0($4)
subu $4,$2,$1


ori $8,$0,0x021a
ori $2,$0,0x1234
sw $8,0($0)
```

```
lw $1,0($0)
addu $3,$1,$2
ori $5,$0,0x9287
lw $4,0($0)
addu $6,$5,$4
lw $7,0($0)
addu $9,$7,$8
ori $11,$0,0x091a
```

3.R-Beq-Jal-Jr-LW-SW

```
beq  $a0, $t0, out0
beq  $a0, $t1, out1
subu $sp, $sp, $t2
sw   $a0, 0($sp)
sw   $ra, 4($sp)
subu $a0, $a0, $t1
jal  fiber
lw   $a0, 0($sp)
lw   $ra, 4($sp)
addu $sp, $sp, $t2
subu $sp, $sp, $t2
sw   $a0, 0($sp)
sw   $ra, 4($sp)
subu $a0, $a0, $t3
jal  fiber
lw   $a0, 0($sp)
lw   $ra, 4($sp)
addu $sp, $sp, $t2
jr   $ra
out1:
```

```
addu $s0, $s0, $t1
jr   $ra
out0:
jr   $ra
end:
addu $8, $s0, $0
```

4.Jr-jal,Jr-R

```
jal loop1
ori $2,$0,1
ori $3,$0,2
j exit1
ori $4,$0,3
loop1:
jr $31
ori $5,$0,2
exit1:
jal loop2
ori $6,$0,6
ori $7,$0,7
j exit2
ori $8,$0,8
loop2:
ori $9,$0,9
jr $31
ori $10,$0,10
exit2:
jal loop3
ori $11,$0,11
ori $12,$0,12
```

```
j exit3
ori $13,$0,13
loop3:
ori $14,$0,14
ori $15,$0,15
jr $31
ori $16,$0,16
exit3:
```

七、思考题

答: P5 中的指令类型主要分为 R 型, SW,LW 型, J,JAL,JR 型, BEQ 型, 每种类型自身之间或与其他类型结合, 都会产生冲突。对于暂停的处理方法在于每级在 controller 解析指令的 T_new, 与 D 级对应指令的 T_use 进行比较, 如果 T_new>T_use, 则暂停。

而转发则通过判断 E\M\W 级指令对应的写入寄存器地址是否与需求位置的寄存器地址相同且不为 0, 并判断 E\M\W 级的 Tnew 是否为 0, 即结果是否已经传入流水线寄存器, 来确定转发的数据与接口。

- 为构造覆盖所有测试情况的测试集, 这里将所有指令组合分为以下情况:

  R: addu subu

  R-R(RS\RT) -> MEM WB

  LW-R(RS\RT) -> MEM WB

  J-R(RS\RT) -> MEM WB

  I: ori

  I-R(RS\RT) -> MEM WB

  LW-I(RS\RT)-> MEM WB

  I-I(RS\RT) -> MEM WB

  J-I(RS\RT) -> MEM WB

J: j -> 不用测试

LW:

R-LW(RS) -> MEM WB

I-LW(RS) -> MEM WB

J-LW(RS) -> MEM WB

SW:

R-SW(RS) -> MEM WB

I-SW(RS) -> MEM WB

J-SW(RS) -> MEM WB

BEQ:

R-BEQ(RS\RT) -> MEM WB

I-BEQ(RS\RT) -> MEM WB

LW-BEQ(RS\RT) -> MEM WB

JR:

R-JR(RS) -> MEM WB

I-JR(RS) -> MEM WB

J-JR(RS) -> MEM WB