Verilog 流水线 CPU 设计文档

一.数据通路设计（分布式）

1.F 级流水线

(1).PC(当前地址)

序号	功能名称	输入/输出	位宽	功能描述
1	NPC	Input	[31:0]	要存到寄存器里的下一个 PC 的值
2	CLK	Input	1	CLK 时钟信号
3	Reset	Input	1	Reset 信号为 1 时，PC 的值置 0
4	Stall	Input	1	暂停信号
5	IMaddr	Output	[11:2]	IM 指令的地址
6	PC	Output	[31:0]	当前 PC 的值

(2).IM（指令存储器）

序号	功能名称	输入/输出	位宽	功能描述
1	IM_in	Input	[9:0]	取出 ROM 中地址为 IM_in 的指令
2	IM_out	Output	[31:0]	输出 32 位 mips 指令值

2.D 级流水线

(1).F_D（D 级流水寄存器）

序号	功能名称	输入/输出	位宽	功能描述
1	IMcode_F	Input	[31:0]	F 级的 32 位指令码
2	CLK	Input	1	CLK 时钟信号
3	Reset	Input	1	Reset 信号为 1 时，寄存器
4	Stall	Input	1	暂停信号
5	PC_F	Input	[31:0]	F 级的 PC 值
6	IMcode_D	Output	[31:0]	D 级的 32 位指令码
7	PC_D	Output	[31:0]	D 级的 PC 值

(2).GRF(同用寄存器堆)

序号	功能名称	输入/输出	位宽	功能描述
1	A1	Input	[25:21]	读出编号为 A1 的寄存器的值到输出端口 RD1
2	A2	Input	[20:16]	读出编号为 A2 的寄存器的值到输出端口 RD2
3	A3	Input	[4:0]	写入寄存器编号
4	WD_RF	Input	[31:0]	写入寄存器的值
5	PC	Input	[31:0]	当前（D 级）PC 值
6	CLK	Input	1	时钟信号

7	Reset	Input	1	Reset 信号为 1 时，寄存器堆中寄存器的值置 0
8	RegWrite	Input	1	为 1 时可以把值写入寄存器
9	RD1	Output	[31:0]	输出端口 RD1
10	RD2	Output	[31:0]	输出端口 RD2

（3）.CMP（比较器）

序号	功能名称	输入/输出	位宽	功能描述
1	RF_RD1	Input	[31:0]	转发后的 GRF[rs]的值
2	RF_RD2	Input	[31:0]	转发后的 GRF[rt]的值
3	IF_Equal	Output	1	为 1 时二者相当，否则为 0

（4）.EXT（拓展器）

序号	功能名称	输入/输出	位宽	功能描述
1	EXT_IN	Input	[15:0]	输入的 16 位立即数
2	OP	Input	[1:0]	控制如何拓展立即数
3	EXT_OUT	Output	[31:0]	输出拓展结果

（5）.NPC（PC 计算单元）

序号	功能名称	输入/输出	位宽	功能描述
1	PC_D	Input	[31:0]	D 级 PC 的地址
2	PC_F	Input	[31:0]	F 级 PC 的地址
3	j_if	Input	1	J 或 jal 指令执行时为 1(D 级)
4	Jr_if	Input	1	jr 指令执行时为 1(D 级)
5	Bne_if	Input	1	Bne 指令执行时为 1(D 级)
6	Beq_if	Input	1	Beq 指令执行时为 1(D 级)
7	IF_Equal	Input	1	CMP 比较的结果，1 时相等，0 时不等
8	rs	Input	[4:0]	Mips 指令的 25-21 位(D 级)
9	rt	Input	[4:0]	Mips 指令的 20-16 位(D 级)
10	imm	Input	[15:0]	Mips 指令的 15-0 位(D 级)
11	RD1	Input	[31:0]	转发后的 GRF[rs]的值
12	EXT_OUT	Input	[31:0]	扩展后的立即数
13	NPC	Output	[31:0]	下一个 PC 的值

（6）.FMUX_rs_D（D 级选择转发信号 rs 多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	rd1	Input	[31:0]	D 级 GRF[rs]中的值
2	Wtod_rs_data	Input	[31:0]	W 级到 D 级 rs 转发的值
3	Mtod_rs_data	Input	[31:0]	M 级到 D 级 rs 转发的值
4	Etod_rs_data	Input	[31:0]	E 级到 D 级 rs 转发的值
5	F_rs_D	Input	[1:0]	D 级 rs 转发控制信号
6	Forwar_rd1_d	Output	[31:0]	D 级 rs 转发后的值

（7）.FMUX_rt_D（D 级选择转发信号 rt 多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	rd2	Input	[31:0]	D 级 GRF[rt]中的值
2	Wtod_rt_data	Input	[31:0]	W 级到 D 级 rt 转发的值
3	Mtod_rt_data	Input	[31:0]	M 级到 D 级 rt 转发的值
4	Etod_rt_data	Input	[31:0]	E 级到 D 级 rt 转发的值
5	F_rt_D	Input	[1:0]	D 级 rt 转发控制信号
6	Forwar_rd2_d	Output	[31:0]	D 级 rt 转发后的值

3. E 级流水线

(1). D_E（E 级流水寄存器）

序号	功能名称	输入/输出	位宽	功能描述
1	IMcode_D	Input	[31:0]	D 级的 32 位指令码
2	RF_RD1	Input	[31:0]	D 级传入的 GRF[rs]的值
3	RF_RD2	Input	[31:0}	D 级传入的 GRF[rt]的值
4	ext_imm	Input	[31:0]	D 级传入的 EXT 拓展后 imm 的值
5	CLK	Input	1	CLK 时钟信号
6	Reset	Input	1	Reset 信号为 1 时，寄存器
7	Stall	Input	1	暂停信号
8	PC_D	Input	[31:0]	D 级的 PC 值
9	RS_E	Output	[31:0]	向 E 级传入的 GRF[rs]的值
10	RT_E	Output	[31:0]	向 E 级传入的 GRF[rt]的值
11	EXT_E	Output	[31:0]	向 E 级传入的 EXT 拓展后 imm 的值
12	IMcode_E	Output	[31:0]	E 级的 32 位指令码
13	PC_E	Output	[31:0]	E 级的 PC 值

（2）. ALU（逻辑运算单元）

序号	功能名称	输入/输出	位宽	功能描述
1	ALUOp	input	[2:0]	SrcA 与 SrcB 的运算方式(见下表)
2	SrcA	input	[31:0]	SrcA 的值
3	SrcB	input	[31:0]	SrcB 的值
4	ALUOUT	output	[31:0]	SrcA 与 SrcB 经过 ALUop 的方式运算后的结果

ALUOp 含义

ALUop 编号	含义	位宽	功能描述
000	“+”	[3:0]	A + B
001	“-”	[3:0]	A - B
010	“||”	[3:0]	A || B
011		[3:0]	未定义
100		[3:0]	未定义
101		[3:0]	未定义
110		[3:0]	未定义

111		[3:0]	未定义

（3）.FMUX_rs_E（E 级选择转发信号 rs 多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	rf_rd1_e	Input	[31:0]	E 级流水寄存器中 GRF[rs]的值
2	Wtoe_rs_data	Input	[31:0]	W 级到 E 级 rs 转发的值
3	Mtoe_rs_data	Input	[31:0]	M 级到 E 级 rs 转发的值
4	F_rs_E	Input	[1:0]	E 级 rs 转发控制信号
5	Forwar_rd1_e	Output	[31:0]	E 级 rs 转发后的值

（4）.FMUX_rt_E（E 级选择转发信号 rt 多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	rf_rd2_e	Input	[31:0]	E 级流水寄存器中 GRF[rt]的值
2	wtoe_rt_data	Input	[31:0]	W 级到 E 级 rt 转发的值
3	mtoe_rt_data	Input	[31:0]	M 级到 E 级 rt 转发的值
4	F_rt_E	Input	[1:0]	E 级 rt 转发控制信号
5	forwar_rd2_e	Output	[31:0]	E 级 rt 转发后的值

（5）.ALU_SrcB_MUX（ALU 的 SrcB 多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	forward_rd2_e	Input	[31:0]	E 级 rt 转发后的值
2	ext_out_e	Input	[31:0]	E 级流水寄存器里的 EXT 扩展后的 imm 的值
3	alu_srcb_e	Input	[1:0]	E 级控制 ALU 的 SrcB 输入的控制信号
4	alu_inb	Output	[31:0]	E 级要写入 ALUSrcB 的值

4.M 级流水线

(1).E_M（M 级流水寄存器）

序号	功能名称	输入/输出	位宽	功能描述
1	IMcode_E	Input	[31:0]	E 级的 32 位指令码
2	AO_E	Input	[31:0]	E 级传入的 ALU_OUT 的值
3	RT_E	Input	[31:0}	E 级传入的 GRF[rt]的值
4	CLK	Input	1	CLK 时钟信号
5	Reset	Input	1	Reset 信号为 1 时，寄存器
6	PC_E	Input	[31:0]	E 级的 PC 值
7	PC_M	Output	[31:0]	M 级的 PC 值
8	RT_M	Output	[31:0]	向 M 级传入的 GRF[rt]的值
9	AO_M	Output	[31:0]	向 M 级传入的 ALU_OUT 的值
10	IMcode_M	Output	[31:0]	M 级的 32 位指令码

（2）.DM（数据存储器）

序号	功能名称	输入/输出	位宽	功能描述
1	AO_E	Input	[31:0]	E 级传入的 ALU_OUT
2	RT_E	Input	[31:0]	写入 RAM 的 32 位数据
3	PC_E	Input	[31:0]	E 级 PC 的值
4	CLK	Input	1	时钟信号
5	Reset	Input	1	当 Reset 为 1 时，将 RAM 清零
6	MemWrite	Input	1	当且仅当 MemWrite 为 1 时，RAM 允许写入
7	DM_OUT	Output	[31:0]	RAM 输出的数据

（3）.FMUX_rt_M（M 级选择转发信号 rt 多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	rd2_m	Input	[31:0]	M 级流水寄存器中 GRF[rt]的值
2	wtom_rt_data	Input	[31:0]	W 级到 M 级 rt 转发的值
3	F_rt_M	Input	[1:0]	M 级 rt 转发控制信号
4	forwar_rd1_m	Output	[31:0]	M 级 rt 转发后的值

（4）.MUX_AO_PC8_M（M 级向其他级转发的数据多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	ao_m	Input	[31:0]	E 级流水寄存器中 ALU_OUT 的值
2	pc8_m	Input	[31:0]	W 级 PC+8 的值
4	j_if_m	Input	[1:0]	M 级是否是 j 型指令
5	forwar_data_m	Output	[31:0]	M 级转发出去的数据

5.W 级流水线

（5）.M_W（W 级流水寄存器）

序号	功能名称	输入/输出	位宽	功能描述
1	IMcode_M	Input	[31:0]	M 级的 32 位指令码
2	AO_M	Input	[31:0]	M 级传入的 ALU_OUT 的值
3	DO_M	Input	[31:0}	M 级传入的 DM 的值
4	CLK	Input	1	CLK 时钟信号
5	Reset	Input	1	Reset 信号为 1 时，寄存器
6	PC_M	Input	[31:0]	M 级的 PC 值
7	PC_W	Output	[31:0]	W 级的 PC 值
8	DO_M	Output	[31:0]	向 W 级传入的 DM 中的值
9	AO_M	Output	[31:0]	向 W 级传入的 ALU_OUT 的值
10	IMcode_W	Output	[31:0]	W 级的 32 位指令码

（6）.MUX_AO_DM_PC8_W（W 级向其他级转发数据多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	dm_out_w	Input	[31:0]	W 级流水寄存器中 DM_OUT 的值

2	ao_w	Input	[31:0]	W 级流水寄存器中 ALU_OUT 的值
4	pc8_w	Input	[31:0]	W 级 PC+8 的值
5	MemtoReg_w	Input	[1:0]	W 级寄存器堆写入的值选择信号
6	forwar_data_w	Output	[31:0]	W 级向其他级的转发数据，也是 W 级写入 GRF 的数据

（7）.A3_MUX（W 级写入 GRF 写入地址多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	rt_w	Input	[31:0]	W 级的 rt
2	rd_w	Input	[31:0]	W 级的 rd
4	A3_jal	Input	[31:0]	31 号寄存器
5	RegDst_w	Input	[1:0]	W 级寄存器堆写入的地址的选择信号
6	a3	Output	[31:0]	W 级写入 GRF 的地址

二.控制模块设计（分布式）

1.Splitter（D,E,M,W 各级指令的指令分离器）

序号	功能名称	输入/输出	位宽	功能描述
1	A	Input	[31:0]	32 位 mips 指令
2	op	Output	[31:26]	Op
3	rs	Output	[25:21]	Rs
4	rt	Output	[20:16]	Rt
5	rd	Output	[15:11]	Rd
6	immediate	Output	[15:0]	立即数
7	funct	output	[5:0]	Funct

2.Control(D,E,M,W 各级指令的信号产生器)

序号	功能名称	输入/输出	位宽	功能描述
1	op	input	[5:0]	OP 信号
2	funct	input	[5:0]	FUNCT 信号
3	Jr_if	Output	1	Jr 指令执行时为 1
4	Bne_if	Output	1	Bne 指令执行时为 1
5	J_if	Output	1	J 或 jal 指令执行时为 1
6	RegDst	Output	[1:0]	选择 GRF 的写入地址
7	ALUSrc	Output	1	选择传入 ALU 作为 ScrB 的数据
8	MemtoReg	Output	[1:0]	选择 GRF 的写入数据
9	RegWrite	Output	1	GRF 的写入使能控制指令
10	MemWrite	Output	1	DM 的写入使能控制指令
11	Beq_if	Output	1	Beq 指令执行时为 1
12	ALUOp	Output	[2:0]	对 ALU 的计算类型的选择信号

13	EXTOp	Output	[1:0]	对立即数拓展类型的选择信号

Control 控制信号生成真值表

op	000000	000000	001101	100011	101011	000100	001111	000000	000011	000000
funct	100001	100011	NULL					000000	NULL	001000
指令名称	addu	subu	ori	lw	sw	beq	lui	j	Jal	jr
RegDst	01	01	00	00	00	00	00	00	10	00
RegWrite	1	1	1	1	0	0	1	0	1	0
ALUsrc	0	0	1	1	1	0	1	0	0	X
ALUOp	000"+"	001"-"	010"||"	000"+"	000"+"	001"-"	000"+"	000“+”	000“+”	000“+”
Beq_if	0	0	0	0	0	1	0	0	0	0
MemWrite	0	0	0	1	0	0	0	0	0	0
MemToReg	01	01	01	00	00	00	01	00	10	00
EXTOp	00	00	00	01	01	01	10	00	00	00
J_if	0	0	0	0	0	0	0	0	1	1
Jr_if	0	0	0	0	0	0	0	0	0	1

3. Make_TuseTnew（D,E,M,W 各级 TuseTnew 产生器）

序号	功能名称	输入/输出	位宽	功能描述
1	IMcode	Input	[31:0]	32 位 mips 指令
2	Tuse_rs	Output	[1:0]	产生 D 级的 Tuse_rs
3	Tuse_rt	Output	[1:0]	产生 D 级的 Tuse_rt
4	Tnew_E	Output	[1:0]	产生 E 级的 Tnew_E
5	Tnew_M	Output	[1:0]	产生 M 级的 Tnew_M
6	Tnew_W	Output	[1:0]	产生 W 级的 Tnew_W

4. A3_MUX（D,E,M,W 各级 A3 多选器）

序号	功能名称	输入/输出	位宽	功能描述
1	rt	Input	[4:0]	对应级的 rt
2	rd	Input	[4:0]	对应级的 rt
3	A3_jal	Input	[4:0]	31 号寄存器
4	RegDst	Input	[1:0]	对应级的向 GRF 写入地址的选择信号
5	a3	Output	[4:0]	对应级写入 GRF 的地址

5. Hazard（冲突单元）

序号	功能名称	输入/输出	位宽	功能描述
1	Tuse_rs	Input	[1:0]	此时刻的 Tuse_rs
2	Tuse_rt	Input	[1:0]	此时刻的 Tuse_rt
3	Tnew_E	Input	[1:0]	此时刻的 Tnew_E
4	Tnew_M	Input	[1:0]	此时刻的 Tnew_M
5	Tnew_W	Input	[1:0]	此时刻的 Tnew_W
6	A1_D	Input	[4:0]	D 级用的 rs
7	A2_D	Input	[4:0]	D 级用的 rt
8	A1_E	Input	[4:0]	E 级用的 rs
9	A2_E	Input	[4:0]	E 级用的 rt
10	A2_M	Input	[4:0]	M 级用的 rt
11	RegWrite_E	Input	1	E 级 GRF 写入使能
12	RegWrite_M	Input	1	M 级 GRF 写入使能
13	RegWrite_W	Input	1	W 级 GRF 写入使能
14	A3_E	Input	[4:0]	E 级 GRF 写入地址
15	A3_M	Input	[4:0]	M 级 GRF 写入地址
16	A3_W	Input	[4:0]	W 级 GRF 写入地址
17	Stall	Output	1	暂停信号
18	F_rs_D	Output	[1:0]	D 级 rs 转发信号
19	F_rt_D	Output	[1:0]	D 级 rt 转发信号
20	F_rs_E	Output	[1:0]	E 级 rs 转发信号
21	F_rt_E	Output	[1:0]	E 级 rt 转发信号
22	F_rt_M	Output	[1:0]	M 级 rt 转发信号

三. 测试代码

```
ori $29,$0,0
ori $28,$0,0
ori $9,$0,20
lui $1,10
sw $1,0($0)
ori $3,$3,17
sw $3,4($0)
ori $4,$4,16
sw $4,8($0)
subu $5,$3,$4
sw $5,12($0)
lw $6,12($0)
lw $7,8($0)
addu $8,$6,$7
sw $8,16($0)
subu $8,$8,$6
one:
nop
beq $8,$3,yep
addu $8,$8,$6
nop
jal one
yep:
addu $8,$8,$6
beq $8,$9,end
jr $ra
end:
nop
```

```
341d0000
341c0000
34090014
3c01000a
ac010000
34630011
ac030004
34840010
ac040008
00642823
ac05000c
8c06000c
8c070008
00c74021
ac080010
01064023
00000000
11030003
01064021
00000000
0c000c10
01064021
11090001
03e00008
00000000
```

测试结果

预期结果

```
$29 <= 00000000
$28 <= 00000000
$ 9 <= 00000014
$ 1 <= 000a0000
*00000000 <= 000a0000
$ 3 <= 00000011
*00000004 <= 00000011
$ 4 <= 00000010
*00000008 <= 00000010
$ 5 <= 00000001
*0000000c <= 00000001
$ 6 <= 00000001
$ 7 <= 00000010
$ 8 <= 00000011
*00000010 <= 00000011
$ 8 <= 00000010
$ 8 <= 00000011
$31 <= 00003058
$ 8 <= 00000012
$ 8 <= 00000013
$31 <= 00003058
$ 8 <= 00000014
$ 8 <= 00000015
$31 <= 00003058
$ 8 <= 00000016
$ 8 <= 00000017
$31 <= 00003058
$ 8 <= 00000018
$ 8 <= 00000019
$31 <= 00003058
$ 8 <= 0000001a
$ 8 <= 0000001b
$31 <= 00003058
$ 8 <= 0000001c
$ 8 <= 0000001d
$31 <= 00003058
$ 8 <= 0000001e
$ 8 <= 0000001f
$31 <= 00003058
$ 8 <= 00000020
```

仿真结果

```
  90@00003000: $29 <= 00000000
 110@00003004: $28 <= 00000000
 130@00003008: $ 9 <= 00000014
 150@0000300c: $ 1 <= 000a0000
 150@00003010: *00000000 <= 000a0000
 190@00003014: $ 3 <= 00000011
 190@00003018: *00000004 <= 00000011
 230@0000301c: $ 4 <= 00000010
 230@00003020: *00000008 <= 00000010
 270@00003024: $ 5 <= 00000001
 270@00003028: *0000000c <= 00000001
 310@0000302c: $ 6 <= 00000001
 330@00003030: $ 7 <= 00000010
 370@00003034: $ 8 <= 00000011
 370@00003038: *00000010 <= 00000011
 410@0000303c: $ 8 <= 00000010
 470@00003048: $ 8 <= 00000011
 510@00003050: $31 <= 00003058
 530@00003054: $ 8 <= 00000012
 590@00003048: $ 8 <= 00000013
 630@00003050: $31 <= 00003058
 650@00003054: $ 8 <= 00000014
 710@00003048: $ 8 <= 00000015
 750@00003050: $31 <= 00003058
 770@00003054: $ 8 <= 00000016
 830@00003048: $ 8 <= 00000017
 870@00003050: $31 <= 00003058
 890@00003054: $ 8 <= 00000018
 950@00003048: $ 8 <= 00000019
 990@00003050: $31 <= 00003058
1010@00003054: $ 8 <= 0000001a
1070@00003048: $ 8 <= 0000001b
1110@00003050: $31 <= 00003058
1130@00003054: $ 8 <= 0000001c
1190@00003048: $ 8 <= 0000001d
1230@00003050: $31 <= 00003058
1250@00003054: $ 8 <= 0000001e
1310@00003048: $ 8 <= 0000001f
1350@00003050: $31 <= 00003058
1370@00003054: $ 8 <= 00000020
```

四.思考题

1、Cal_i(ori)和 Cal_r(addu,subu)的冲突

测试代码	预期结果	实际结果

ori $t1,34
ori $t1,53
ori $t2,21
ori $t1,13
addu $t1,$t2,$t1
subu $t2,$t1,$t2
addu $t2,$t1,$t2
addu $t1,$t2,$t2
ori $t1,15
addu $t2,$t2,$t2
ori $t1,15
ori $t2,15
	$ 9 <= 00000022
	$ 9 <= 00000037
	$10 <= 00000015
	$ 9 <= 0000003f
	$ 9 <= 00000054
	$10 <= 0000003f
	$10 <= 00000093
	$ 9 <= 00000126
	$ 9 <= 0000012f
	$10 <= 00000126
	$ 9 <= 0000012f
	$10 <= 0000012f
		90@00003000: $ 9 <= 00000022
		110@00003004: $ 9 <= 00000037
		130@00003008: $10 <= 00000015
		150@0000300c: $ 9 <= 0000003f
		170@00003010: $ 9 <= 00000054
		190@00003014: $10 <= 0000003f
		210@00003018: $10 <= 00000093
		230@0000301c: $ 9 <= 00000126
		250@00003020: $ 9 <= 0000012f
		270@00003024: $10 <= 00000126
		290@00003028: $ 9 <= 0000012f
		310@0000302c: $10 <= 0000012f

由于 Cal_i 和 Cal_r 型指令的 Tuse 都是 1，且 E 级 Tnew 也是 1，故所有冲突都可以通过转发解决。

2、Save(sw)与 Load(lw)与 Cal_i(ori)的冲突

测试代码	预期结果	实际结果

ori $t1,48
sw $t1,4($t0)
ori $t1,36
sw $t1,4($t1)
sw $t1,-4($t1)
lw $t2,4($t1)
sw $t2,8($t0)
sw $t2,12($t0)
lw $t2,4($t0)
ori $t2,12
lw $t2,4($t1)
nop
ori $t2,12
	$ 9 <= 00000030
	*00000004 <= 00000030
	$ 9 <= 00000034
	*00000038 <= 00000034
	*00000030 <= 00000034
	$10 <= 00000034
	*00000008 <= 00000034
	*0000000c <= 00000034
	$10 <= 00000030
	$10 <= 0000003c
	$10 <= 00000034
	$10 <= 0000003c
		90@00003000: $ 9 <= 00000030
		90@00003004: *00000004 <= 00000030
		130@00003008: $ 9 <= 00000034
		130@0000300c: *00000038 <= 00000034
		150@00003010: *00000030 <= 00000034
		190@00003014: $10 <= 00000034
		190@00003018: *00000008 <= 00000034
		210@0000301c: *0000000c <= 00000034
		250@00003020: $10 <= 00000030
		290@00003024: $10 <= 0000003c
		310@00003028: $10 <= 00000034
		350@00003030: $10 <= 0000003c

Load 类在 E 级 Tnew 是 2，当 D 级有 Tuse 为 1 的指令，若操作寄存器一样，则只能暂停，其他时候都可以转发。

3、Save(sw)与 Load(lw)与 Cal_r(addu)的冲突

测试代码	预期结果	实际结果

ori $t1,48
ori $t2,36
addu $t1,$t1,$t2
sw $t1,4($t0)
addu $t1,$t1,$t1
 	
sw $t1,4($t1)
sw $t1,-4($t1)
lw $t2,4($t1)
sw $t2,8($t0)
sw $t2,12($t0)
lw $t2,4($t0)
addu $t2,$t2,$t2
lw $t2,4($t1)
nop
addu $t2,$t2,$t2
	$ 9 <= 00000030
	$10 <= 00000024
	$ 9 <= 00000054
	*00000004 <= 00000054
	$ 9 <= 000000a8
	*000000ac <= 000000a8
	*000000a4 <= 000000a8
	$10 <= 000000a8
	*00000008 <= 000000a8
	*0000000c <= 000000a8
	$10 <= 00000054
	$10 <= 000000a8
	$10 <= 000000a8
	$10 <= 00000150
		90@00003000: $ 9 <= 00000030
		110@00003004: $10 <= 00000024
		130@00003008: $ 9 <= 00000054
		130@0000300c: *00000004 <= 00000054
		170@00003010: $ 9 <= 000000a8
		170@00003014: *000000ac <= 000000a8
		190@00003018: *000000a4 <= 000000a8
		230@0000301c: $10 <= 000000a8
		230@00003020: *00000008 <= 000000a8
		250@00003024: *0000000c <= 000000a8
		290@00003028: $10 <= 00000054
		330@0000302c: $10 <= 000000a8
		350@00003030: $10 <= 000000a8
		390@00003038: $10 <= 00000150

Load 类在 E 级 Tnew 是 2，当 D 级有 Tuse 为 1 的指令，若操作寄存器一样，则只能暂停，其他时候都可以转发。

4、Cal_r(addu)与 beq，Cal_i(ori)与 beq 的冲突

测试代码	预期结果	实际结果
s0: ori $t1,1 ori $t2,2 addu $t1,$t1,$t2 beq $t1,$t2,s0 nop ori $t3,4 ori $t4,4 s1: beq $t3,$t4,s1 nop	$ 9 <= 00000001 $10 <= 00000002 $ 9 <= 00000003 $11 <= 00000004 $12 <= 00000004	90@00003000: $ 9 <= 00000001 110@00003004: $10 <= 00000002 130@00003008: $ 9 <= 00000003 210@00003014: $11 <= 00000004 230@00003018: $12 <= 00000004

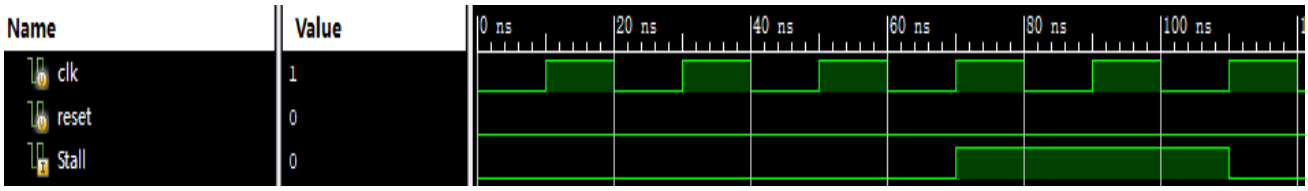
beq 的 Tuse 为 0，Cal_r 和 Cal_i 类在 E 级 Tnew 都是 1，在对寄存器操作相同时只能暂停。其他时候都可以转发。

5、Load(lw)与 beq 的冲突

测试代码	预期结果	实际结果

s0:
ori $t1,100
sw $t1,0($t0)
lw $t3,0($t0)
beq $t2,$t3,s0
nop
lw $t4,0($t0)
lw $t5,0($t0)
s1:
beq $t4,$t5,s1
nop
	$ 9 <= 00000064
	*00000000 <= 00000064
	$11 <= 00000064
	$12 <= 00000064
	$13 <= 00000064
		90@00003000: $ 9 <= 00000064
		90@00003004: *00000000 <= 00000064
		130@00003008: $11 <= 00000064
		230@00003014: $12 <= 00000064
		250@00003018: $13 <= 00000064

由于 beq 的 Tuse 为 0，Load(lw)在 E 级的 Tnew 是 2，所以只能暂停且要暂停两周期。但是当 Load(lw)在 W 级时 Tnew 是 0，可以转发。



6、Jal、Jr 与 Cal_r(addu)与 Cal_i(ori)的冲突

测试代码	预期结果	实际结果
ori $t1,4 jal s1 addu $31,$31,$t1 ori $ra,0 ori $ra,0 ori $ra,12312 s1: jr $ra nop	$31 <= 0000300c $31 <= 00003010 $31 <= 00003010 $31 <= 00003018	110@00003004: $31 <= 0000300c 130@00003008: $31 <= 00003010 210@00003010: $31 <= 00003010 230@00003014: $31 <= 00003018

当 Jr 在 D 级，

Tuse 为 0，Cal_r(addu)在 E 级 Tnew 为 1，只能暂停，Cal_i(ori)同理。其他情况都可转发。

7、Jal、Jr 与 Sw 与 Lw 的冲突

测试代码	预期结果	实际结果

s0:
ori $t1,4
jal s1
sw $ra,-4($t1)
ori $ra,0
ori $ra,12312
s1:
ori $ra,0
lw $ra,-4($t1)
jr $ra
nop
	$ 9 <= 00000004
	$31 <= 0000300c
	*00000000 <= 0000300c
	$31 <= 0000300c
	$31 <= 0000300c
	$31 <= 0000300c
	$31 <= 0000301c
	$31 <= 0000301c
	$31 <= 0000300c
	$31 <= 0000300c
	$31 <= 0000301c
	$31 <= 0000301c
		90@00003000: $ 9 <= 00000004
		110@00003004: $31 <= 0000300c
		110@00003008: *00000000 <= 0000300c
		150@00003014: $31 <= 0000300c
		170@00003018: $31 <= 0000300c
		270@0000300c: $31 <= 0000300c
		290@00003010: $31 <= 0000301c
		310@00003014: $31 <= 0000301c
		330@00003018: $31 <= 0000300c
		430@0000300c: $31 <= 0000300c
		450@00003010: $31 <= 0000301c
		470@00003014: $31 <= 0000301c

由与 jr 在 D 级时 Tues 为 0，lw 的 Tnew_E 是 2，Tnew_M 是 1，若操作相同寄存器，只能暂停，且 lw 在 E 级时要暂停两周期。

如下附上 TnewTuse 和暂停转发表格：

Tuse				功能部件	Tnew		
指令	rs	rt			E	M	W
addu	1	1		ALU	1	0	0
subu	1	1		ALU	1	0	0
ori	1			ALU	1	0	0
lw	1			DM	2	1	0
sw	1	2					
beq	0	0					
lui				ALU	1	0	0
j							
jal				PC	0	0	0
jr	0						
	{1,0}	{1,2,0}			{1,2,0}	{1,0}	{0}

RS	E			M			W		
Tuse/Tnew	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

RT	E			M			W		
Tuse/Tnew	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
2	F	F	F	F	F	F	F	F	F