CPU 设计文档

一、 数据通路设计

1. 模块规格 datapath.v

1） 端口说明

表 1- 1 端口说明

信号	方向	描述
PauseF	I	
PauseD	I	
ClearE	I	
Forward_RS_D_src[2:0]	I	
Forward_RS_D_src[2:0]	I	
Forward_RS_D_src[1:0]	I	
Forward_RS_D_src[1:0]	I	
Forward_RS_D_src	I	
Clk	I	时钟信号
Reset	I	复位信号
IRD_out[31:0]	O	传入冲突控制的指令信号

2） 功能定义

表 1- 2 功能定义

功能	描述
grf 写入	若 grf 需要写入数据，则输出写入的位置及写入的值。
dm 写入	若 dm 需要写入数据，则输出写入的位置及写入的值。

2. 取指

1） 模块规格 F_unit.v

表 2- 1 端口说明

信号	方向	描述
NPC[31:0]	I	
PCsrc[1:0]	I	下一指令地址控制信号
Branch	I	是否跳转
Clk	I	时钟信号
Reset	I	复位信号

PauseF	I	暂停更新 PC
PauseD	I	暂停更新 D 级流水线寄存器
IRD[31:0]	O	当前指令
PC4D[31:0]	O	

2） 程序计数器 pc_F.v

表 2- 2 端口说明

信号	方向	描述
NPC[31:0]	I	下一指令地址/j 型和分支
Clk	I	时钟信号
Reset	I	复位信号
En	I	使能信号
add4	I	Pc+4 的值
j_r	I	跳转寄存器的指令地址
Branch	I	分支指令控制信号
PCsrc	I	下一指令控制
PC[31:0]	O	当前指令地址

表 2- 3 功能定义

功能	描述
存储指令地址	输出当前指令地址。时钟上升沿到来，且使能端有效时保存下一指令地址
复位	复位信号有效时，PC 同步复位于 0x0000_3000

3） 指令存储器 im.v

表 2- 4 端口说明

信号	方向	描述
addr[11:2]	I	指令地址
instr[31:0]	O	指令

表 2- 5 功能定义

功能	描述
存储指令	初始化时，加载指令码。容量为 32bit*1024
输出指令	输出地址信号对应的指令内容

3. 译码

1） 模块规格 D.v

表 3- 1 端口说明

信号	方向	描述
IRD[31:0]	I	当前指令

PC4D[31:0]	I	下一指令地址
PC4W	I	用于写寄存器时输出指令的地址
Clk	I	时钟信号
Reset	I	复位信号
ClearE	I	清空 E 流水级寄存器
Forward_RS_D_src[2:0]	I	转发 MUX 控制信号
Forward_RT_D_src[2:0]	I	转发 MUX 控制信号
PC4_forw_E[31:0]	I	
PC4_forw_M[31:0]	I	
AO[31:0]	I	
W_RF_WD_OUT[31:0]	I	
W_RF_A3_OUT[31:0]	I	
Branch	O	是否跳转
PCsrc[1:0]	O	
NPC[31:0]	O	跳转至地址
RS_D_F[31:0]	O	D 级读寄存器值-jr
IRE[31:0]	O	
PC4E[31:0]	O	
RSE[31:0]	O	
RTE[31:0]	O	
EXTE[31:0]	O	
We	I	

2） 寄存器文件 rf_D.v

表 3- 2 端口说明

信号	方向	描述
A1[4:0]	I	第一操作数地址
A2[4:0]	I	第二操作数地址
A3[4:0]	I	第三操作数地址
WD[31:0]	I	待写入数据
En	I	写使能信号
Clk	I	时钟信号
Reset	I	复位信号
RD1[31:0]	O	第一操作数
RD2[31:0]	O	第二操作数

表 3- 3 功能定义

功能	描述
复位	复位信号有效时，全部寄存器同步复位为 0

3） 扩展单元 ext_D.v

表 3- 4 端口说明

信号	方向	描述
EXTop[1:0]	I	扩展功能
In[15:0]	I	待扩展立即数
Out[31:0]	O	扩展结果

表 3- 5 功能定义

功能	ExtOp[1:0]	描述
{16'b0, Imm}	00	无符号扩展
{{16{Imm[15]}}, Imm}	01	有符号扩展
{Imm, 16'b0}	10	扩展至低位
32'b0	11	复位

4） 指令自增单元 npc_D.v

表 3- 6 端口说明

信号	方向	描述
PC4[31:0]	I	
I26[25:0]	I	指令的低 26 位
NPCsrc	I	控制信号
NPC[31:0]	O	

表 3- 7 功能定义

功能	NPCOp	描述
Branch	0	NPC = PC+4+{{14{I26[15]}}, I26[15:0], 2'b00}
Jump	1	NPC = {PC[31:28], I26[25:0], 2'b00}

5） 比较单元 cmp_D.v

表 3- 8 端口说明

信号	方向	描述
RD1[31:0]	I	第一个操作数
RD2[31:0]	I	第二个操作数
OPCode[5:0]	I	指令类型
Branch	O	是否跳转

4. 执行

1） 模块规格 E_unit.v

表 4- 1 端口说明

信号	方向	描述
IRE[31:0]	I	

PC4E[31:0]	I	
RSE[31:0]	I	
RTE[31:0]	I	
EXTE[31:0]	I	
Clk	I	时钟信号
Reset	I	复位信号
IRM[31:0]	O	
PC4M[31:0]	O	
AOM[31:0]	O	
RTM[31:0]	O	
Forward_RS_E_src[2:0]	I	转发 MUX 控制信号
Forward_RT_E_src[2:0]	I	转发 MUX 控制信号
AO[31:0]	I	
PC4_forw_M[31:0]	I	
W_RF_WD_OUT[31:0]	I	

2） 计算单元 alu_E.v

表 4- 2 端口说明

信号	方向	描述
SrcA[31:0]	I	A
SrcB[31:0]	I	B
ALUOp[2:0]	I	运算类型
Shift[4:0]	I	移位
ALUresult[31:0]	O	

表 4- 3 功能定义

功能	ALUOp[2:0]	描述
Nop	000	0
Add	001	A+B
Sub	010	A-B
And	011	A&B
Or	100	A|B
Xor	101	A^B
Le	110	A<B
B	111	B

5. 访存

1） 模块规格 M.v

表 5- 1 端口说明

信号	方向	描述

IRM[31:0]	I	
PC4M[31:0]	I	
AOM[31:0]	I	
RTM[31:0]	I	
Clk	I	时钟信号
Reset	I	复位信号
IRW[31:0]	O	
PC4W[31:0]	O	
AOW[31:0]	O	
DRW[31:0]	O	
Forward_RT_M_src[2:0]	I	转发 MUX 控制信号
W_RF_WD_OUT[31:0]	I	

2） 数据寄存器 dm.v

表 5- 2 端口说明

信号	方向	描述
Addr[11:2]	I	数据地址
WD[31:0]	I	待写入数据
WE	I	写使能信号
Clk	I	时钟信号
Reset	I	复位信号
LStype	I	存取类型
Instr	I	存取指令地址
RD[31:0]	O	数据

表 5- 3 功能定义

功能	描述
复位	复位信号有效时，dm 同步复位于 0x0000_0000

6. 写回

1） 模块规格 W_unit.v

表 6- 1 端口说明

信号	方向	描述
IRW[31:0]	I	
PC4W[31:0]	I	
AOW[31:0]	I	
DRW[31:0]	I	
W_RF_A3_OUT[4:0]	O	写回寄存器编号
W_RF_WD_OUT[31:0]	O	写回数据
RegWrite	O	

二、 控制器设计

1. 控制单元 ctrl.v

表 1- 1 端口说明

信号	方向	描述
OPCode[31:26]	I	
FunctCode[5:0]	I	
NPCsrc	O	下一指令地址来源
PCsrc[1:0]	O	指令跳转方式
ExtOp@D[1:0]	O	立即数扩展方式
ALUSrc@E	O	ALU 第二操作数来源
ALUOp@E[2:0]	O	ALU 运算方式
MemWrite@M	O	数据存储器写使能信号
RegDst@W[1:0]	O	待回写寄存器地址来源
MemtoReg@W[1:0]	O	待回写数据来源
RegWrite@W	O	寄存器写使能信号
LStype	O	主存读写方式

表 1- 2 R 型指令功能定义

Funct	000000	001000	001001	100001	100011	101010
OpCode	000000	000000	000000	000000	000000	000000
	nop	jr	jalr	addu	subu	slt
NPCsrc@F	00	11	11	00	00	00
PCsrc@D[1:0]	x	x	x	x	x	x
ExtOp@D[1:0]	xx	xx	xx	xx	xx	xx
ALUSrc@E	x	0	0	0	0	0
ALUOp@E[2:0]	xxx	100	100	001	010	110
MemWrite@M	0	0	0	0	0	0
RegDst@W[1:0]	xx	xx	10	00	00	00
MemtoReg@W[1:0]	xx	xx	10	00	00	00
RegWrite@W	0	0	1	1	1	1

表 1- 3 I 型指令功能定义

OpCode	000100	001001	001101	001111	100011	101011
	beq	addiu	ori	lui	lw	sw
NPCsrc@F	10	00	00	00	00	00
PCsrc@D[1:0]	0	x	x	x	x	x
ExtOp@D[1:0]	xx	01	00	10	01	01
ALUSrc@E	0	1	1	1	1	1
ALUOp@E[2:0]	010	001	100	111	001	001
MemWrite@M	0	0	0	0	0	1

RegDst@W[1:0]	xx	01	01	01	01	xx
MemtoReg@W[1:0]	xx	00	00	00	01	xx
RegWrite@W	0	1	1	1	1	0

表 1- 4 J 型指令功能定义

OpCode	000010	000011
	j	jal
NPCsrc@F	01	01
PCsrc@D[1:0]	1	1
ExtOp@D[1:0]	xx	xx
ALUSrc@E	x	x
ALUOp@E[2:0]	xxx	xxx
MemWrite@M	0	0
RegDst@W[1:0]	xx	10
MemtoReg@W[1:0]	xx	10
RegWrite@W	0	1

2. 冲突单元 Hazard_unit.v

表 2- 1 端口说明

信号	方向	描述
IRD[31:0]	I	D 级流水线寄存器指令
Clk	I	时钟信号
Reset	I	复位信号
PauseF	O	
PauseD	O	
ClearE	O	
Forward_RS_D_src[2:0]	O	
Forward_RT_D_src[2:0]	O	
Forward_RS_E_src[2:0]	O	
Forward_RT_E_src[2:0]	O	
Forward_RT_M_src[2:0]	O	

3. AT 解码器

表 3- 1 端口说明

信号	方向	描述
Instr[31:0]	I	D 级流水线寄存器指令
RS[4:0]	O	第一个读寄存器编号，如不需要则为 0
RT[4:0]	O	第二个读寄存器编号，如不

		需要则为 0
WR[4:0]	O	写寄存器编号，如不需要则为 0
Instype[3:0]	O	指令类型

4. 不确定写寄存器控制单元

信号	方向	描述
DJ_E	I	E 级流水线寄存器(D 信号)
DJ_M	O	M 级流水线寄存器(D 信号)
DJ_W	O	W 级流水线寄存器(D 信号)
EJ_M		
EJ_W		
MJ_W		

三、 测试程序

```
.text
# init
addiu $t0, $0, 10
addiu $t1, $0, 11

# cal_r
addu $s0, $t0, $t1
addu $s1, $s0, $t0 # r_e_rs
addu $s2, $s0, $s1 # r_e_rt, r_m_rs
addu $s3, $s0, $s1 # r_m_rt, r_w_rs
addu $s4, $s3, $s1 # r_w_rt
addiu $s0, $t0, 11
subu $s1, $s0, $t1 # i_e_rs
subu $s2, $s0, $t1 # i_m_rs
subu $s3, $s0, $t1 # i_w_rs
addiu $s0, $t1, 20
subu $s1, $t1, $s0 # i_e_rt
subu $s2, $t1, $s0 # i_m_rt
subu $s3, $t1, $s0 # i_w_rt
sw $s0, 0x00000000
sw $s1, 0x00000004
sw $s2, 0x00000008
sw $s3, 0x0000000c
```

```
lw $s4, 0x00000000
slt $s5, $s4, $t0 # load_e_rs
slt $t6, $s4, $t0 # load_m_rs
slt $t7, $s4, $t0 # loadw_rs
lw $s4, 0x00000004
slt $s5, $t0, $s4 # load_e_rt
slt $s6, $t0, $s4 # load_m_rt
slt $s7, $t0, $s4 # load_w_rt
jal mark1
    addu $s0, $t0, $ra # jal_e_rt
    addu $t2, $s1, $ra
mark1:
addu $s1, $t0, $ra # jal_m_rt
addu $s2, $t0, $ra # jal_w_rt
jal mark2
    addu $s0, $t1, $ra # jal_e_rs
    addu $t3, $s1, $ra
mark2:
addu $s1, $ra, $t1 # jal_m_rs
addu $s2, $ra, $t1 # jal_w_rs
jal mark3
    addu $s1, $t0, $ra # jalr_m_rt
    addu $s2, $t0, $ra # jalr_w_rt
    j mark5
    nop
mark3:
jalr $ra
addu $s0, $t0, $ra # jalr_e_rt
mark5:
jal mark4
    addu $s4, $t0, $ra # jalr_m_rs
    addu $s5, $t0, $ra # jalr_w_rs
mark4:
jalr $ra
addu $s3, $t0, $ra # jalr_e_rs


ori $s0, $0, 0
ori $s1, $0, 0
ori $s2, $0, 0
ori $s3, $0, 0
ori $s4, $0, 0
ori $s5, $0, 0
ori $s6, $0, 0
```

```
ori $s7, $0, 0
ori $t2, $0, 0
# cal_i
addu $s0, $t0, $t1
addiu $s1, $s0, 1 # r_e
addiu $s2, $s0, 2 # r_m
addiu $s3, $s0, 3 # r_w
lui $s3, 5
addiu $s4, $s3, 4 # i_e
addiu $s5, $s3, 5 # i_m
addiu $s6, $s3, 6 # i_w
lw $s0, 0x00000000
ori $s1, $s0, 0x00000020 # load_e
ori $s2, $s0, 0x00000020 # load_m
ori $s3, $s0, 0x00000020 # load_w
jal mark6
    sw $ra, 0x00000010 # jal_e
    addiu $t2, $ra, 100
mark6:
sw $ra, 0x00000004 # jal_m
sw $ra, 0x00000008 # jal_w
jal mark7
nop
    addiu $s2, $s0, 2 # jalr_m
    addiu $s3, $s0, 3 # jalr_w
    j mark8
    nop
mark7:
jalr $s0, $ra
addiu $s1, $s0, 1
mark8:


ori $s0, $0, 0
ori $s1, $0, 0
ori $s2, $0, 0
ori $s3, $0, 0
ori $s4, $0, 0
ori $s5, $0, 0
ori $s6, $0, 0
ori $s7, $0, 0
ori $t2, $0, 0
# b
addu $s0, $t0, $t1
```

```
addu $s1, $t0, $s0
beq $s0, $s1, mark9 # r_m_rs, r_e_rt
    nop
    ori $t3, $0, 1
mark9:
ori $s0, $0, 0
ori $s1, $0, 0
addu $s0, $t0, $t1
nop
addu $s1, $t0, $s0
beq $s1, $s0, mark10 # r_e_rs, r_w_rt
    nop
    ori $t3, $0, 1
mark10:
ori $s0, $0, 0
ori $s1, $0, 1
addu $s0, $t0, $t1
addu $s1, $t0, $t1
nop
beq $s0, $s1, mark11 # r_w_rs, r_m_rt
    nop
    ori $t3, $0, 1
mark11:
ori $s0, $0, 0
ori $s1, $0, 0
ori $s0, $0, 1
ori $s1, $0, 2
beq $s0, $s1, mark12 # i_m_rs, i_e_rt
    nop
    ori $t3, $0, 1

9@00003000: $ 8 <= 0000000a
11@00003004: $ 9 <= 0000000b
13@00003008: $16 <= 00000015
15@0000300c: $17 <= 0000001f
17@00003010: $18 <= 00000034
19@00003014: $19 <= 00000034
21@00003018: $20 <= 00000053
23@0000301c: $16 <= 00000015
25@00003020: $17 <= 0000000a
27@00003024: $18 <= 0000000a
29@00003028: $19 <= 0000000a
31@0000302c: $16 <= 0000001f
33@00003030: $17 <= fffffffec
```

```
35@00003034: $18 <= fffffec
37@00003038: $19 <= fffffec
37@0000303c: *00000000 <= 0000001f
39@00003040: *00000004 <= fffffec
41@00003044: *00000008 <= fffffec
43@00003048: *0000000c <= fffffec
47@0000304c: $20 <= 0000001f
51@00003050: $21 <= 00000000
53@00003054: $14 <= 00000000
55@00003058: $15 <= 00000000
57@0000305c: $20 <= fffffec
61@00003060: $21 <= 00000001
63@00003064: $22 <= 00000001
65@00003068: $23 <= 00000001
67@0000306c: $31 <= 00003074
69@00003070: $16 <= 0000307e
71@00003078: $17 <= 0000307e
73@0000307c: $18 <= 0000307e
75@00003080: $31 <= 00003088
77@00003084: $16 <= 00003093
79@0000308c: $17 <= 00003093
81@00003090: $18 <= 00003093
83@00003094: $31 <= 0000309c
85@00003098: $17 <= 000030a6
87@000030a8: $31 <= 000030b0
89@000030ac: $16 <= 000030ba
91@0000309c: $18 <= 000030ba
97@000030b0: $31 <= 000030b8
99@000030b4: $20 <= 000030c2
101@000030bc: $31 <= 000030c4
103@000030c0: $19 <= 000030ce
105@000030b8: $21 <= 000030ce
107@000030bc: $31 <= 000030c4
109@000030c0: $19 <= 000030ce
111@000030c4: $16 <= 00000000
113@000030c8: $17 <= 00000000
115@000030cc: $18 <= 00000000
117@000030d0: $19 <= 00000000
119@000030d4: $20 <= 00000000
121@000030d8: $21 <= 00000000
123@000030dc: $22 <= 00000000
125@000030e0: $23 <= 00000000
127@000030e4: $10 <= 00000000
129@000030e8: $16 <= 00000015
```

```
131@000030ec: $17 <= 00000016
133@000030f0: $18 <= 00000017
135@000030f4: $19 <= 00000018
137@000030f8: $19 <= 00050000
139@000030fc: $20 <= 00050004
141@00003100: $21 <= 00050005
143@00003104: $22 <= 00050006
145@00003108: $16 <= 0000001f
149@0000310c: $17 <= 0000003f
151@00003110: $18 <= 0000003f
153@00003114: $19 <= 0000003f
155@00003118: $31 <= 00003120
155@0000311c: *00000010 <= 00003120
157@00003124: *00000004 <= 00003120
159@00003128: *00000008 <= 00003120
163@0000312c: $31 <= 00003134
167@00003144: $31 <= 0000314c
169@00003148: $17 <= 0000314d
171@00003134: $18 <= 0000314e
173@00003138: $19 <= 0000314f
179@0000314c: $16 <= 00000000
181@00003150: $17 <= 00000000
183@00003154: $18 <= 00000000
185@00003158: $19 <= 00000000
187@0000315c: $20 <= 00000000
189@00003160: $21 <= 00000000
191@00003164: $22 <= 00000000
193@00003168: $23 <= 00000000
195@0000316c: $10 <= 00000000
```

思考题

测试类型			cal_r	cal_i	b	load	store	jr	jalr
前序类型	冲突阶段	冲突寄存器	测试序列	测试序列	测试序列	测试序列	测试序列	测试序列	测试序列
cal_r	E	rs							
cal_r	E	rt							
cal_r	M	rs							
cal_r	M	rt							
cal_r	W	rs							
cal_r	W	rt							
cal_i	E	rs							
cal_i	E	rt							
cal_i	M	rs							

cal_i	M	rt							
cal_i	W	rs							
cal_i	W	rt							
load	E	rs							
load	E	rt							
load	M	rs							
load	M	rt							
load	W	rs							
load	W	rt							
jal	E	rs							
jal	E	rt							
jal	M	rs							
jal	M	rt							
jal	W	rs							
jal	W	rt							
jalr	E	rs							
jalr	E	rt							
jalr	M	rs							
jalr	M	rt							
jalr	W	rs							

除了部分不需要读第二个寄存器的指令以外，上表中每格基本都代表一种冲突。比如第一行第一格表示 cal_r 指令在 D 级时，若 E 级有 cal_r 指令，且后者 rd 值与前者 rs 值相等则会引发冲突。流水线多条指令冲突转发优先级的问题还没有测试全面。

D			E					M
type	register	Tuse	cal_i	cal_r	load	jalr	jal	load
			1/rt	1/rd	2/rt	0/rd	0/$31	1/rt
cal_i	rs	1			stall			
cal_r	rs/rt	1			stall			
beq	rs/rt	0	stall	stall	stall			stall
load	rs	1			stall			
store	rs	1			stall			
store	rt	2						
jr	rs	0	stall	stall	stall			stall
jalr	rs	0	stall	stall	stall			stall