流水线 CPU
设计文档（P6）

贾博驿

初稿：2023 年 11 月 19 日

编辑：2024 年 8 月 28 日

设计实现指令

共 50 条
实际指令 47 条

R 型指令：

6	5	5	5	5	6
opcode	rs	rt	rd	shamt	funct

I 型指令：

6	5	5	16
opcode	rs (base)	rt	imm16 (immediate, offset)

J 型指令：

6	26
opcode	instr_index

注：以下的 RTL 除分支与跳转类指令均省略 PC←PC+4。

算数与逻辑类

1. 算数类

a. R 型

名称	funct	RTL
add	100000	GPR[rd] ← GPR[rs] + GPR[rt]
addu	100001	GPR[rd] ← GPR[rs] + GPR[rt]
sub	100010	GPR[rd] ← GPR[rs] - GPR[rt]
subu	100011	GPR[rd] ← GPR[rs] - GPR[rt]

注：实现的 add 实际上为 addu、sub 实际上为 subu。二者均按照无符号指令处理。

b. I 型

名称	opcode	RTL
addi	001000	GPR[rt] ← GPR[rs] + sign_extend(immediate)
addiu	001001	GPR[rt] ← GPR[rs] + sign_extend(immediate)

注：实现的 addi 实际上为 addiu。按照无符号指令处理。

2. 逻辑类

a. R 型

名称	funct	RTL
and	100100	GPR[rd] ← GPR[rs] AND(bitwise) GPR[rt]
or	100101	GPR[rd] ← GPR[rs] OR(bitwise) GPR[rt]
xor	100110	GPR[rd] ← GPR[rs] XOR(bitwise) GPR[rt]
nor	100111	GPR[rd] ← GPR[rs] NOR(bitwise) GPR[rt]
sll	000000	GPR[rd] ← GPR[rt]$_{(31-shamt)..0}$ || 0^{shamt}
srl	000010	GPR[rd] ← 0^{shamt} || GPR[rt]$_{31..shamt}$
sra	000011	GPR[rd] ← (GPR[rt]$_{31}$)^{shamt} || GPR[rt]$_{31..shamt}$
sllv	000100	s ← GPR[rs]$_{4..0}$ GPR[rd] ← GPR[rt]$_{(31-s)..0}$ || 0^{s}
srlv	000110	s ← GPR[rs]$_{4..0}$ GPR[rd] ← 0^{s} || GPR[rt]$_{31..s}$
srav	000111	s ← GPR[rs]$_{4..0}$ GPR[rd] ← (GPR[rt]$_{31}$)^{s} || GPR[rt]$_{31..s}$

注：根据寄存器值移位指令只依照寄存器值的低五位移位，不受高位值影响。

b. I 型

名称	opcode	RTL
andi	001100	GPR[rt] ← GPR[rs] AND(bitwise) zero_extend(immediate)
ori	001101	GPR[rt] ← GPR[rs] OR(bitwise) zero_extend(immediate)
xori	001110	GPR[rt] ← GPR[rs] XOR(bitwise) zero_extend(immediate)

3. 乘除法类（R 型）

名称	funct	RTL
mult	011000	prod ← GPR[rs]$_{31..0}$ × GPR[rt]$_{31..0}$ LO ← prod$_{31..0}$ HI ← prod$_{63..32}$
multu	011001	prod ← (0 || GPR[rs]$_{31..0}$) × (0 || GPR[rt]$_{31..0}$) LO ← prod$_{31..0}$ HI ← prod$_{63..32}$
div	011010	q ← GPR[rs]$_{31..0}$ ÷ GPR[rt]$_{31..0}$ r ← GPR[rs]$_{31..0}$ % GPR[rt]$_{31..0}$ LO ← q HI ← r
divu	011011	q ← (0 || GPR[rs]$_{31..0}$) ÷ (0 || GPR[rt]$_{31..0}$) r ← (0 || GPR[rs]$_{31..0}$) % (0 || GPR[rt]$_{31..0}$) LO ← q$_{31..0}$ HI ← r$_{31..0}$
mfhi	010000	GPR[rd] ← HI
mthi	010001	HI ← GPR[rs]
mflo	010010	GPR[rd] ← LO
mtlo	010011	LO ← GPR[rs]

寄存器操作类

a. R 型

名称	funct	RTL
slt	101010	GPR[rd] ← 0$^{GPRLEN-1}$ || GPR[rs] < GPR[rt]
sltu	101011	GPR[rd] ← 0$^{GPRLEN-1}$ || (0 || GPR[rs]) < (0 || GPR[rt])

注：slt 类指令成立时置 1，**不成立时置 0**。

b. I 型

名称	opcode	RTL
slti	001010	GPR[rt] ← 0$^{GPRLEN-1}$ || GPR[rs] < sign_extend(immediate)
sltiu	001011	GPR[rt] ← 0$^{GPRLEN-1}$ || (0 || GPR[rs]) < (0 || **sign_extend**(immediate))
lui	001111	GPR[rt] ← immediate || 0^{16}

注：sltiu 对立即数的处理是先**有符号扩展**再进行**无符号比较**。

分支与跳转类

1. 分支类（I 型）

名称	opcode	rt	RTL
(all)			**I+1:** if condition then PC ← PC + sign_extend(offset || 0^2) endif
bltz	000001	00000	**I:** condition ← GPR[rs] < 0^{32}
bgez	000001	00001	**I:** condition ← GPR[rs] ≥ 0^{32}

beq	000100	rt	**I:** condition ← (GPR[rs] = GPR[rt])
bne	000101	rt	**I:** condition ← (GPR[rs] ≠ GPR[rt])
blez	000110	00000	**I:** condition ← GPR[rs] ≤ 0_{32}
bgtz	000111	00000	**I:** condition ← GPR[rs] > 0_{32}

注：流水线 CPU 分支指令不需要 PC + 4 后再加上 offset，直接在延迟槽的地址上加。

2. 跳转类

a. R 型

名称	funct	RTL
jr	001000	**I:** **I+1:** PC ← GPR[rs]
jalr	001001	**I:** GPR[rd] ← PC + 8 **I+1:** PC ← GPR[rs]

b. J 型

名称	opcode	RTL
j	000010	**I:** **I+1:** PC ← $PC_{31..28}$ || instr_index || 0_2
jal	000011	**I:** GPR[31] ← PC + 8 **I+1:** PC ← $PC_{31..28}$ || instr_index || 0_2

注：流水线 CPU 跳转指令由于延迟槽存在应该加上 PC + 8，而非单周期的 PC + 4。

内存操作类（I 型）

名称	opcode	RTL
(all)		addr ← sign_extend(offset) + GPR[base]
lb	100000	byte ← $addr_{1..0}$ GPR[rt] ← sign_extend(memory[addr]$_{7+8*byte..8*byte}$)
lh	100001	half ← $addr_1$ GPR[rt] ← sign_extend(memory[addr]$_{15+8*half..8*half}$)
lw	100011	GPR[rt] ← memory[addr]
lbu	100100	byte ← $addr_{1..0}$ GPR[rt] ← zero_extend(memory[addr]$_{7+8*byte..8*byte}$)
lhu	100101	half ← $addr_1$ GPR[rt] ← zero_extend(memory[addr]$_{15+8*half..8*half}$)
sb	101000	memory[addr]$_{7..0}$ ← GPR[rt]$_{7..0}$
sh	101001	memory[addr]$_{15..0}$ ← GPR[rt]$_{15..0}$
sw	101011	memory[addr]$_{31..0}$ ← GPR[rt]

注：

1.针对半字操作需要 addr 为 2 的非负整数倍，针对字节操作需要 addr 为 4 的非负整数倍，本次实现时实际只涉及针对字节操作，且保证地址是合法的。

2.针对字节和半字操作时只修改对应字节和半字的值，**同一字的其他部分不变**。

转发与暂停设计

指令分类

类型名	信号名	数量	指令
算数逻辑寄存器	ALREG	16	_arith（算数）：add, addu, sub, subu _logic（逻辑）：and, or, xor, nor _shift（移位）：sll, srl, sra, sllv, srlv, srav _comp（比较）：slt, sltu
算数逻辑立即数	ALIMM	7	_sign（有符号拓展）：addi, addiu, slti, sltiu _zero（无符号拓展）：andi, ori, xori
乘除法	MLUTDIV	4	mult, multu, div, divu
乘除寄存器写入	MLTO	2	mthi, mtlo
乘除寄存器读取	MLFROM	2	mfhi, mflo
高位加载	EXTLUI	1	lui
两数比较分支	BRANCHE	2	beq, bne
与零比较分支	BRANCHZ	4	bltz, bgez, blez, bgtz
仅读取跳转	JUMPR	1	jr
仅写入跳转	JUMPW	1	jal
读取写入跳转	JUMPRW	1	jalr
读取内存	LOAD	5	lb, lh, lw, lbu, lhu
写入内存	STORE	3	sb, sh, sw

注：不包括 j 指令，因为 j 指令不读取通用寄存器也不写入通用寄存器。

Tuse 与 Tnew

信号名	Tuse_rs	Tuse_rt	GRF_WD_W_Sel	Tnew_E	Tnew_M	Tnew_W
ALREG	1	1	00	1	0	0
ALIMM	1	(3)	00	1	0	0
MLUTDIV	1	1	(00)	(0)	(0)	(0)
MLTO	1	(3)	(00)	(0)	(0)	(0)
MLFROM	(3)	(3)	00	1	0	0
EXTLUI	(3)	(3)	10	0	0	0
BRANCHE	0	0	(00)	(0)	(0)	(0)
BRANCHZ	0	(3)	(00)	(0)	(0)	(0)
JUMPR	0	(3)	(00)	(0)	(0)	(0)
JUMPW	(3)	(3)	11	0	0	0
JUMPRW	0	(3)	11	0	0	0
LOAD	1	(3)	01	2	1	0
STORE	1	2	(00)	(0)	(0)	(0)

注：

1.(3)表示该类指令不会读取 rs 或 rt 域翻译为寄存器编号后对应的寄存器，除 GRF 内部转发外不产生任何转发和暂停。

2.(00)和(0)表示表示该类指令 GRFWE 为 0，虽然使用 ALU 数据但无意义。

转发与暂停

<table>
<tr><td colspan="10">SPL_rs != 5'd0 && GRFWE_X && SPL_rs == GRF_A3_X</td></tr>
<tr><td>流水段(X)</td><td colspan="4">E</td><td colspan="4">M (~ rs_E_premise)</td><td>W</td></tr>
<tr><td>GRF_WD_W_Sel_X(Tnew_X)</td><td>00(1)</td><td>01(2)</td><td>10(0)</td><td>11(0)</td><td>00(0)</td><td>01(1)</td><td>10(0)</td><td>11(0)</td><td>All</td></tr>
<tr><td>Tuse_rs == 0</td><td>S1</td><td>S2</td><td rowspan="2">F43</td><td rowspan="2">F44</td><td rowspan="2">F21</td><td>S3</td><td rowspan="2">F23</td><td rowspan="2">F24</td><td rowspan="2">F1</td></tr>
<tr><td>Tuse_rs == 1</td><td>F51</td><td>S4</td><td>F32</td></tr>
<tr><td colspan="10">SPL_rt != 5'd0 && GRFWE_X && SPL_rt == GRF_A3_X</td></tr>
<tr><td>流水段(X)</td><td colspan="4">E</td><td colspan="4">M (~ rs_E_premise)</td><td>W</td></tr>
<tr><td>GRF_WD_W_Sel_X(Tnew_X)</td><td>00(1)</td><td>01(2)</td><td>10(0)</td><td>11(0)</td><td>00(0)</td><td>01(1)</td><td>10(0)</td><td>11(0)</td><td>All</td></tr>
<tr><td>Tuse_rt == 0</td><td>S1</td><td>S2</td><td rowspan="3">F43</td><td rowspan="3">F44</td><td rowspan="3">F21</td><td>S3</td><td rowspan="3">F23</td><td rowspan="3">F24</td><td rowspan="3">F1</td></tr>
<tr><td>Tuse_rt == 1</td><td rowspan="2">F51</td><td>S4</td><td rowspan="2">F32</td></tr>
<tr><td>Tuse_rt == 2</td><td>F62</td></tr>
<tr><td colspan="10">ISMULTDIV & (MULT_Start | MULT_Busy)</td></tr>
<tr><td colspan="10">SMULTDIV</td></tr>
</table>

转发信号表

SPL_rs != 5'd0 && GRFWE_X && SPL_rs == GRF_A3_X			
类型	描述	FMUX_V1_D_Sel	FMUX_V1_E_Sel
S	暂停		
F51	V1_E 转发 OP_M		11
F43	V1_D 转发 ext32_E	110	
F44	V1_D 转发 pc8_E	111	
F32	V1_E 转发 DM_Q_W		10
F21	V1_D 转发 OP_M	011	
F23	V1_D 转发 ext32_M	100	
F24	V1_D 转发 pc8_M	101	
F1	GRF 内部转发		
F0	不转发	000	00

SPL_rt != 5'd0 && GRFWE_X && SPL_rt == GRF_A3_X				
类型	描述	FMUX_V2_D_Sel	FMUX_V2_E_Sel	FMUX_OP_M_Sel
S	暂停			
F62	DM_D_M 转发 DM_Q_W			1
F51	V2_E 转发 OP_M		11	
F43	V2_D 转发 ext32_E	110		
F44	V2_D 转发 pc8_E	111		
F32	V2_E 转发 DM_Q_W		10	
F21	V2_D 转发 OP_M	011		
F23	V2_D 转发 ext32_M	100		
F24	V2_D 转发 pc8_M	101		
F1	GRF 内部转发			
0	不转发	000	00	0

暂停变化表

I	I+1	I+2
S1	F21	
S2	S3	F1

I	I+1	I+2
S3	F1	
S4	F32	

注：

1.优先级 **S > F4、F5、F6 > F2、F3。**

2.F2 和 F3 之间、F4、F5 和 F6 之间不会发生冲突，因为转发的是同一流水级的不同数据。

3.当 MULT_Busy 信号或 MULT_Start_E 信号为 1 时，MLUTDIV、MLTO、MLFROM 指令均被阻塞在 D 流水级，即 SMULTDIV 暂停。

4.未标注的信号取值与不转发相同。

模块设计

IFU

1. 描述

取指令模块。内有 PC 子模块、NPC 子模块。接收分支和跳转指令的控制信号和数据并实现跳转。输出指令地址，通过与外部 ROM 通信获取指令。

2. 接口

端口名	方向	描述
RESET	I	同步复位信号
clk	I	时钟信号
STALL_EN_N	I	暂停使能信号
NPCSel[1:0]	I	NPC 子模块控制信号
BranchComp	I	NPC 子模块分支指令控制信号
offset[15:0]	I	分支指令跳转偏移
instr_index[25:0]	I	j、jal 指令跳转地址
instr_register[31:0]	I	jr、jalr 指令跳转寄存器
InstrAddr[31:0]	O	即将进入流水线的指令地址，接入顶层 i_inst_addr

3. 控制信号

BranchComp：接 COMP 的 BranchComp。

NPCSel：

指令（信号）名	取值	描述
其他	00	PC←PC+4
BRANCHE、BRANCHZ	01	使用 offset，结合 BranchComp 判断
j、jal	10	使用 instr_index
jr、jalr	11	使用 instr_register

IFU_PC

1. 描述

程序计数器子模块。在复位信号有效时复位为 32'h00003000。

2. 接口

端口名	方向	描述
RESET	I	同步复位信号
clk	I	时钟信号
STALL_EN_N	I	暂停使能信号
D[31:0]	O	下一指令地址
Q[31:0]	O	即将进入流水线的指令地址

IFU_NPC

1. 描述

下一条指令地址计算子模块。输入 IFU 接收的分支跳转指令控制信号和数据，计算出下一条指令的地址。

2. 接口

端口名	方向	描述

PC[31:0]	1	即将进入流水线的指令地址
NPCSel[1:0]	I	同 IFU
BranchComp	I	同 IFU
offset[15:0]	I	同 IFU
instr_index[25:0]	I	同 IFU
instr_register[31:0]	I	同 IFU
NPC[31:0]	O	下一指令地址

SPL

1. 描述

指令分线器模块。按照三种指令的格式将一条指令分为不同的部分，供其他模块使用。

2. 接口

端口名	方向	描述
Instr[31:0]	1	当前指令
opcode[5:0]	O	指令的操作数
rs[4:0]	O	R、I 型指令的 rs
rt[4:0]	O	R、I 型指令的 rt
rd[4:0]	O	R 型指令的 rd
shamt[4:0]	O	R 型指令的 shamt
funct[5:0]	O	R 型指令的 funct
imm16[15:0]	O	I 型指令的立即数
instr_index[25:0]	O	J 型指令的跳转地址

GRF
(GRF_R, GRF_W)

1. 描述

寄存器堆模块，共有 31 个寄存器（0 号寄存器直接接地）。通过 A1、A2 输入地址，可以取出指定寄存器的完成内部转发的数据。当 WE 使能时，通过 A3 输入地址，D 输入数据，可以修改指定寄存器存储的数据并进行内部转发。在流水线设计时分为两个部分体现。

2. 接口

端口名	方向	描述
RESET	I	同步复位信号
clk	I	时钟信号
WE	I	写入使能信号，同时输出到顶层 w_grf_we
A1[4:0]	I	读取的第一个寄存器的编号
A2[4:0]	I	读取的第二个寄存器的编号
A3[4:0]	I	写入的寄存器的编号，同时输出到顶层 w_grf_addr
D[31:0]	I	写入寄存器的数据，同时输出到顶层 w_grf_wdata
V1[31:0]	O	读取的第一个寄存器的完成内部转发的值
V2[31:0]	O	读取的第二个寄存器的完成内部转发的值

3. 控制信号

GRFWE：

指令信号名	取值	描述

其他	0	不涉及寄存器写入的指令
ALREG	1	
ALIMM、EXTLUI	1	
MLFROM	1	
LOAD	1	
JUMPW、JUMPRW	1	

4. 关联 MUX

MUX_GRF_A3_D，控制信号 CTRL_D_GRF_A3_D_Sel。

指令信号名	取值	描述
其他	00	接入 SPL_rd
ALIMM、EXTLUI	10	接入 SPL_rt
LOAD	10	接入 SPL_rt
JUMPW	11	固定 32’d31

MUX_GRF_WD_W，控制信号 GRF_WD_W_Sel_W。

指令信号名	取值	描述
其他	00	接入 OP_W
LOAD	01	接入 DM_Q_W
EXTLUI	10	接入 ext32_W
JUMPW、JUMPRW	11	接入 pc8_W

COMP

1. 描述

比较模块，完成两数相等、不等，一数有符号小于、小于等于、大于、大于等于零的比较运算。比较结果用于判断分支指令是否执行。

2. 接口

端口名	方向	描述
CompSel[2:0]	I	比较方式控制信号
A[31:0]	I	第一个操作数
B[31:0]	I	第二个操作数
BranchComp	O	比较结果

3. 控制信号

CompSel：

指令名	取值	描述
bltz(00000)	000	有符号小于 0
bgez(00001)	001	有符号大于等于 0
beq(000100)	100	两数相等
bne(000101)	101	两数不等
blez(000110)	110	有符号小于等于 0
bgtz(000111)	111	有符号大于 0

EXT

1. 描述

位扩展模块，将 16 位立即数无符号扩展、有符号扩展、右补 16 位 0 为 32 位。

2. 接口

端口名	方向	描述
EXTSel[1:0]	I	扩展方式控制信号
imm16[15:0]	I	16 位立即数
ext32[31:0]	O	32 位扩展结果

3. 控制信号

EXTSel：

指令名	取值	描述
其他	00	有符号拓展
andi、ori、xori	01	无符号拓展
lui	10	右补 16 位 0

ALU

1. 描述

算数逻辑模块，完成算数运算（加、减）、逻辑运算（与、或、异或、或非）、移位运算（左移、逻辑右移、算数右移）、比较运算（两数有符号小于，无符号小于）。计算结果用于保存到寄存器。

2. 接口

端口名	方向	描述
OPSel[1:0]	I	OP 输出控制信号
FuncSel[1:0]	I	计算方式控制信号
A[31:0]	I	第一个操作数
B[31:0]	I	第二个操作数
shamt[4:0]	I	移位立即数
OP[31:0]	O	计算结果

3. 控制信号

OPSel：

指令（信号）名	取值	描述
add、addu、sub、subu、addi、addiu	00	算数运算
LOAD、STORE	00	算数运算
and、or、xor、nor、andi、ori、xori	01	逻辑运算
sll、srl、sra、sllv、srlv、srav	10	移位运算
slt、sltu、slti、sltiu	11	比较运算

FuncSel：接 CTRL，根据 OPSel 选择。

OPSel == 00 时。

指令名	取值	描述
add、addu、addi、addiu	00	加法
LOAD、STORE	00	加法
sub、subu	01	减法

OPSel == 01 时，根据指令是否为 I 型选择。R 型接 funct[1:0]、I 型接 opcode[1:0]。

指令名	取值	描述
and(100100)、andi(001100)	00	与
or(100101)、ori(001101)	01	或

xor(100110)、xori(001110)	10	异或
nor(100111)	11	或非

OPSel == 10 时，接 funct[1:0]。

指令名	取值	描述
sll(000000)、sllv(000100)	00	逻辑左移
srl(000010)、srlv(000110)	10	逻辑右移
sra(000011)、srav(000111)	11	算数右移

OPSel == 11 时。

指令名	取值	描述
slt、slti	00	有符号两数小于
sltu、sltiu	01	无符号两数小于

4. 关联 MUX

MUX_ALU_B_E，控制信号 ALU_B_E_Sel_E。

指令信号名	取值	描述
其他	0	接入 FMUX_V2_E
ALIMM、LOAD、STORE	1	接入 EXT

MULT

1. 描述

乘除法模块，执行乘法的时间为 5 个时钟周期，执行除法的时间为 10 个时钟周期（包含写入内部的 HI、LO 寄存器）。自 1 个时钟周期 Start 信号有效后的第 1 个 clock 上升沿开始乘除法运算，同时 Busy 置位为 1。在运算结果保存到 HI 寄存器和 LO 寄存器后，Busy 清零。写入 HI、LO 均只需 1 个时钟周期，类似 GRF。读取 HI、LO 直接读取，类似 ALU。

2. 接口

端口名	方向	描述
RESET	I	同步复位信号
clk	I	时钟信号
ISMULTDIV	I	是否为乘除指令信号
MULTSel[1:0]	I	乘除模块控制信号
A[31:0]	I	第一个操作数
B[31:0]	I	第二个操作数
Start	O	乘除计算启动信号，输出 ISMULTDIV & MULTSel[2]
Busy	O	乘除计算进行信号
HILO[31:0]	O	读取的 HI、LO 的结果

3. 控制信号

ISMULTDIV：指令类型名为 MULTDIV、MLTO、MLFROM 时为 1。

MULTSel：接{funct[3], funct[1:0]}。

指令名	取值	描述
mult(011000)	100	准备执行有符号乘法操作
multu(011001)	101	准备执行无符号乘法操作
div(011010)	110	准备执行有符号除法操作
divu(011011)	111	准备执行无符号除法操作
mfhi(010000)	000	HILO 输出 HI

mflo(010010)	010	HILO 输出 LO
mthi(010001)	001	准备将 A 写入 HI
mtlo(010011)	011	准备将 A 写入 LO

4. 关联 MUX

MUX_OP_E，控制信号 OP_E_Sel_E。

指令信号名	取值	描述
其他	0	接入 ALU_OP
MLFROM	1	接入 MULT_HILO

DM

1. 描述

内存处理模块。将控制信号和数据处理为与外部 RAM 通信的合法信号。

2. 接口

端口名	方向	描述
WE	I	写入使能信号
DMSel[2:0]	I	读取写入模式控制信号
A[31:0]	I	读取或写入的地址，同时输出到顶层 m_data_addr
D[31:0]	I	原始的写入数据
rdata[31:0]	I	原始的读取数据，接入顶层 m_data_rdata
wdata[31:0]	O	处理后的读取数据，接入顶层 m_data_wdata
byteen[3:0]	O	字节使能信号，接入顶层 m_data_byteen
Q[31:0]	O	处理后的读取数据

注：字节使能信号表示 D 的哪些字节要被写入，若要将低位的数据写入高位需要进行移位。

3. 控制信号

DMWE：

指令信号名	取值	描述
其他	0	
STORE	1	

DMSel：接 opcode[2:0]。

指令名	取值	描述
lb(100000)、sb(101000)	000	针对 byte 操作，**读取时**有符号拓展
lh(100001)、sh(101001)	001	针对 halfword 操作，**读取时**有符号拓展
lw(100011)、sw(101011)	011	针对 word 操作
lbu(100100)	100	针对 byte 操作，读取时无符号拓展
lhu(100101)	101	针对 halfword 操作，读取时无符号拓展

4. 输出控制信号

byteen：

指令名	DMWE	DMSel	A[1:0]	byteen						
其他	0		00	0000	01	0000	10	0000	11	0000
sb	1	000		0001		0010		0100		1000
sh	1	001		0011		/		1100		/
sw	1	011		1111		/		/		/

FR_D, FR_E, FR_M, FR_W

1. 描述

D、E、M、W 段流水线寄存器。

2. 接口

通用：

端口名	方向	描述
RESET	I	同步复位信号
clk	I	时钟信号

以下端口均有 I 和 O 两个方向，I 方向信号名以 D_开头，O 方向信号名以 Q_开头。

FR_D：

端口名	接入
STALL_EN_N	CTRL_S_FR_D_EN
Instr[31:0]	顶层 i_inst_rdata
InstrAddr[31:0]	IFU_InstrAddr

FR_E：

端口名	接入
STALL_RESET	CTRL_S_FR_E_RESET
DMWE	CTRL_D_DMWE
GRFWE	CTRL_D_GRFWE
DMSel[2:0]	CTRL_D_DMSel
FuncSel[1:0]	CTRL_D_FuncSel
OPSel[1:0]	CTRL_D_OPSel
GRF_WD_W_Sel[1:0]	CTRL_D_GRF_WD_W_Sel
ALU_B_E_Sel	CTRL_D_ALU_B_E_Sel
OP_E_Sel	CTRL_D_OP_E_Sel
MULTSel[2:0]	CTRL_D_MULTSel
ISMULTDIV	CTRL_D_ISMUTLDIV
V1[31:0]	FMUX_V1_D
V2[31:0]	FMUX_V2_D
shamt[4:0]	SPL_shamt
GRF_A3[4:0]	MUX_GRF_A3_D
ext32[31:0]	EXT_ext32
pc8[31:0]	pc8_D
FMUX_V1_E_Sel[1:0]	CTRL_F_ FMUX_V1_E_Sel
FMUX_V2_E_Sel[1:0]	CTRL_F_ FMUX_V2_E_Sel
FMUX_DM_D_M_Sel	CTRL_F_ FMUX_DM_D_M_Sel

FR_M：

端口名	接入
DMWE	DMWE_E
GRFWE	GRFWE_E
DMSel[2:0]	DMSel_E
GRF_WD_W_Sel[1:0]	GRF_WD_W_Sel_E
V2[31:0]	FMUX_V2_E

OP[31:0]	MUX_OP_E
GRF_A3[4:0]	GRF_A3_E
ext32[31:0]	ext32_E
pc8[31:0]	pc8_E
FMUX_DM_D_M_Sel	FMUX_DM_D_M_Sel_E

FR_W：

端口名	接入
GRFWE	GRFWE_M
GRF_WD_W_Sel[1:0]	GRF_WD_W_Sel_M
OP[31:0]	OP_M
DM_Q[31:0]	DM_Q
GRF_A3[4:0]	GRF_A3_ M
ext32[31:0]	ext32_ M
pc8[31:0]	pc8_ M

CTRL_Decoder

1. 描述

   译码器模块。通过 opcode、funct 和 rt 判断指令的类型，输出各个模块的控制信号。

2. 接口

端口名	方向	端口名	方向
opcode[5:0]	I	DMSel[2:0]	O
funct[5:0]	I	DMWE	O
rt[4:0]	I	GRFWE	O
CompSel[2:0]	O	GRF_A3_D_Sel[1:0]	O
EXTSel[1:0]	O	GRF_WD_W_Sel[1:0]	O
NPCSel[1:0]	O	ALU_B_E_Sel	O
OPSel[1:0]	O	OP_E_Sel	O
FuncSel[1:0]	O	Tuse_rs[1:0]	O
MULTSel[2:0]	O	Tuse_rt[1:0]	O
ISMULTDIV	O		

CTRL_Stall

1. 描述

   暂停控制器模块。判断是否需要暂停，若需要暂停则产生暂停信号。

2. 接口

端口名	方向	端口名	方向
Tuse_rs[1:0]	I	GRF_A3_E[4:0]	I
Tuse_rt[1:0]	I	GRF_A3_M[4:0]	I
SPL_rs[4:0]	I	ISMULTDIV	I
SPL_rt[4:0]	I	MULT_Start	I
GRFWE_E	I	MULT_Busy	I
GRFWE_M	I	IFU_EN_N	O
GRF_WD_W_Sel_E[1:0]	I	FR_D_EN_N	O

GRF_WD_W_Sel_M[1:0]	I	FR_E_RESET	O

CTRL_Forward

1. 描述

   转发控制器模块。判断是否需要转发，若需要转发转发什么数据。

2. 接口

端口名	方向	端口名	方向
Tuse_rs[1:0]	I	GRF_A3_E[4:0]	I
Tuse_rt[1:0]	I	GRF_A3_M[4:0]	I
SPL_rs[4:0]	I	FMUX_V1_D_Sel[2:0]	O
SPL_rt[4:0]	I	FMUX_V2_D_Sel[2:0]	O
GRFWE_E	I	FMUX_V1_E_Sel[1:0]	O
GRFWE_M	I	FMUX_V2_E_Sel[1:0]	O
GRF_WD_W_Sel_E[1:0]	I	FMUX_DM_D_M_Sel	O
GRF_WD_W_Sel_M[1:0]	I		