CPU设计文档

17373123 丁一芙

设计说明

- 处理器应支持MIPS-lite2指令集。MIPS-lite2={ addu, subu, ori, lw, sw, beq, lui, j, jal, jr, nop }

- 处理器为流水线设计。

1　构建数据通路

1.1　流水线寄存器统一命名

流水线寄存器命名表

名字	宽度	描述
instr	32	指令码
PC4	32	下一条指令地址
PC8	32	下下条指令地址
rs	5	指令的rs区段
rt	5	指令的rt区段
rd	5	指令的rd区段
shamt	5	指令的shamt区段
imm16	16	16位立即数
imm26	26	26位立即数
RA1	5	第一个源寄存器编号,对应指令的rs区段
RA2	5	第二个源寄存器编号,对应指令的rt区段
WAG	5	目的寄存器编号,对应指令的rt或rd区段.
WDG	32	GRF的存入数据
RD1	32	第一个源寄存器的值
RD2	32	第二个源寄存器的值
EXT	32	符号扩展结果
D1	32	CMP的第一个源操作数
D2	32	CMP的第二个源操作数
SA	32	ALU的第一个输入数
SB	32	ALU的第二个输入数
ALUO	32	ALU计算结果(ALU output)
RD	32	DM的输出值
WAD	5	DM的读写地址
WDD	32	DM的存入数据

1.2　建模指令的RTL

lw指令

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←pc+4	
decode/read register	RD1@E←GRF[rs@D]	EXTop:signext
	EXT@E←signext(imm16@D)	
	WAG@E←rt@D	
	RA1@E←rs@D	
execute	RA@M←alu(RD1@E,EXT@E)	ALUop:addu
	WAG@M←WAG@E	
memory	RD@W←dm(RA@W)	
	WAG@W←WAG@M	
write back	GRF[WAG@W]←RD@W	RegWrite:yes

sw指令

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←pc+4	
decode/read register	RD1@E←GRF[rs@D]	EXTop:signext
	RD2@E←GRF[rt@D]	
	RA1@E←rs@D	
	RA2@E←rt@D	
	EXT@E←signext(imm16@D)	
execute	ALUO@M←alu(RD1@E,EXT@E)	alu:addu
	RD2@M←RD2@E	
memory	DM[ALUO@M]←RD2@M	MemWrite:yes
write back	none	

addu指令

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←pc+4	
decode/read register	RD1@E←GRF[rs@D]	EXTop:signext
	RD2@E←GRF[rt@D]	
	RA1@E←rs@D	
	RA2@E←rt@D	
	WAG@E←rd@D	
execute	ALUO@M←alu(RD1@E,RD2@E)	alu:addu
	WAG@M←WAG@E	
memory	none	
write back	WAG@W←WAG@M	RegWrite:yes
	WDG@W←ALUO@M	

subu指令

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←pc+4	
decode/read register	RD1@E←GRF[rs@D]	EXTop:signext
	RD2@E←GRF[rt@D]	
	RA1@E←rs@D	
	RA2@E←rt@D	
	WAG@E←rd@D	
execute	ALUO@M←alu(RD1@E,RD2@E)	alu:subu
	WAG@M←WAG@E	
memory	none	
write back	WAG@W←WAG@M	RegWrite:yes
	WDG@W←ALUO@M	

ori指令

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←pc+4	
decode/read register	RD1@E←GRF[rs@D]	EXTop:zeroext
	EXT@E←zeroext(imm16@D)	
	RA1@E←rs@D	
	WAG@E←rt@D	
execute	ALUO@M←alu(RD1@E,RD2@E)	alu:or
	WAG@M←WAG@E	
memory	none	
write back	WAG@W←WAG@M	RegWrite:yes
	WDG@W←ALUO@M	

lui指令

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←pc+4	
decode/read register	WAG@E←rt@D	EXTop:lui
	EXT@E←luiext(imm16@D)	
execute	ALUO@M←alu(EXT@E)	alu:output directly
	WAG@M←WAG@E	
memory	none	
write back	WAG@W←WAG@M	RegWrite:yes
	WDG@W←ALUO@M	

beq指令

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←pc+4/npc	
decode/read register	RA1@E←rs@D	EXTop:signext
	RA2@E←rt@D	
	RD1@E←GRF[rs@D]	
	RD2@E←GRF[rt@D]	
	EXT@E←signext(imm16@D)	
execute	CMPO@E←cmp(RD1@E,RD2@E)	cmp:subu
memory	none	
write back	none	

jal指令

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←npc	
decode/read register	WAG@E←31	
	WDG@E←PC4	
execute	WAG@M←WAG@E	
	WDG@M←WDG@E	
memory	WAG@W←WAG@M	
	WDG@W←WDG@M	
write back	GRF[WAG@W]←WDG@W	

j

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←npc	
decode/read register	none	
execute	none	
memory	none	
write back	none	

jr指令

step	RTL	control signal
instruction fetch	instr←IM[pc]	pc.en:yes
	pc←npc	
decode/read register	RA1@E←rs@D	
	RD1@E←GRF[rs@D]	
execute	none	
memory	none	
write back	none	

1.3 各级组合逻辑和流水线寄存器数据通路

各级组合逻辑和流水线寄存器数据通路表

寄存器所在位置		lw	sw	addu	subu	ori	lui	beq	jal	j	jr	MUX	0	1	2	3
PC		PC4	PC4	PC4	PC4	PC4	PC4	PC4|NPC	NPC	NPC	GRF.RD1	MPC	PC4	NPC(b类)	GRF.RD1	NPC(j类)
IM		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC				
D级	instr	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM				
	PC4	PC4	PC4	PC4	PC4	PC4	PC4	PC4	PC4	PC4	PC4	PC4				
	PC8	PC8	PC8	PC8	PC8	PC8	PC8	PC8	PC8	PC8	PC8	PC8				
GRF	RA1	rs@D	rs@D	rs@D	rs@D	rs@D		rs@D			rs@D	rs@D				
	RA2		rt@D	rt@D	rt@D		rt@D	rt@D				rt@D				
EXT		signext(@imm16@D)	signext(@imm16@D)	none	none	zeroext(@imm16@D)	lui(@imm16)	signext(imm16@D)	none	none	none	MEXTD	zeroext	signext	luiext	
CMP	D1							GRF[rs@D]				GRF[rs@D]				
	D2							GRF[rt@D]				GRF[rt@D]				
E级	RA1	rs@D	rs@D	rs@D	rs@D	rs@D						rs@D				
	RA2		rt@D	rt@D	rt@D							rt@D				
	WAG	rt@D		rd@D	rd@D	rt@D	rt@D		31			MWAGE	rt@D	rd@D	31	
	RD1	GRF[rs@D]	GRF[rs@D]	GRF[rs@D]	GRF[rs@D]	GRF[rs@D]						GRF[rs@D]				
	RD2		GRF[rt@D]	GRF[rt@D]	GRF[rt@D]							GRF[rt@D]				
	EXT	EXT	EXT			EXT	EXT					EXT				
	PC4								PC4@D			PC4@D				
	PC8								PC8@D			PC8@D				
ALU	SA	RD1@E	RD1@E	RD1@E	RD1@E	RD1@E	anything					RD1@E				
	SB	EXT@E	EXT@E	RD2@E	RD2@E	EXT@E	EXT@E					MALUB	RD2@E	EXT@E		
M级	RA2		RA2@E									RA2@E				
	RD2		RD2@E									RD2@E				
	ALUO	ALU	ALU	ALU	ALU	ALU	ALU					ALUO				
	WAG	WAG@E	none	WAG@E	WAG@E	WAG@E	WAG@E		WAG@E			WAG@E				
	PC4								PC4@E			PC4@E				
	PC8								PC8@E			PC8@E				
DM	ADDR	ALUO@M	ALUO@M	ALUO@M	ALUO@M	ALUO@M	ALUO@M					ALUO@M				
	WDD	none	RD2@M	none	none	none	none					RD2@M				
W级	WAG	WAG@M	none	WAG@M	WAG@M	WAG@M	WAG@M		WAG@M			WAG@M				
	ALUO		none	ALUO@M	ALUO@M	ALUO@M	ALUO@M					ALUO@M				
	RD	DM[ALUO@M]	none	none	none	none	none					DM[ALUO@M]				
	PC4								PC4@M			PC4@M				
	PC8								PC8@M			PC8@M				
GRF	WAG	WAG@W	none	WAG@W	WAG@W	WAG@W	WAG@W		WAG@W			WAG@W				
	WDG	RD@W	none	ALUO@W	ALUO@W	ALUO@W	ALUO@W		PC8@W			MMTR	ALUO@W	RD@W	PC8@W	

说明 根据每条指令的数据通路,综合生成无转发数据通路表.合并同类项,对于输入源在两个以上者,部署功能mux(紫色单元格标出),对于来源可能是后继流水线寄存器的转发的,部署转发mux(橙色单元格标出),形成完整的数据通路.

功能MUX与选择信号对应表

功能MUX名	选择信号	00	01	10	11
MPC	npc_sel	pc←pc+4	pc←pc+signext(imm16)<<2	pc←GRF.RD1	pc←pc[31:28]+imm26<<2
MEXTD	EXTop	zero_ext/unsigned_ext	sign_ext	lui	
MWAGE	RegDst_D	WA←rt@D	WA←rd@D	WA←31	
MALUB	ALUSrc_E	SourceB←MFALUBEO	SourceB←EXT@E		
MMTR	MemtoReg_W	WDG←ALUO@W	WDG←RD_DM@W	WDG←PC8@W	

备注: 因为延迟槽的缘故,一定会执行跳转指令的后一句,因此跳转指令的算法改为pc+imm<<2.

LdStType信号释义表

指令类型	000	001	010	011	100	101	110	111
load	lw	lbu	lb	lhu	lh	none	none	none
store	sw	sb	none	sh	none	none	none	none

1.4 构造功能MUX控制信号表达式

```
npc_sel=(lw+sw+addu+subu+ori+lui+beq&!zero)?2'b00:
    (beq&zero+jal+j)?2'b01:
    (jr)?2'b10:
    2'b11;
EXTop=(ori)?2'b00:
    (lw+sw+beq)?2'b01:
    (lui)?2'b10:
    2'b11;
```

```
RegDst=(lw+ori+lui)?2'b00:
    (addu+subu)?2'b01:
    (jal)?2'b10:
    2'b11;
ALUSrc=(addu+subu)?2'b00:
    (lw+sw+ori+lui)?2'b01:
    2'b11;
MemtoReg=(addu+subu+ori+lui)?2'b00:
    (lw)?2'b01:
    (jal)?2'b11;
```

1.5 构造转发MUX

转发MUX供给需求表

所在级	转发MUX名	功能	供给者	需求者
D	MFRSD	beq指令比较电路前移至D级,GRF[rs]需要被从E,M,W转发	D级:直接读出GRF[rs] E级:E级存储的RD1 M级:ALUO@M W级:WDG@W	CMP.D1 RD1@E
	MFRTD	beq指令比较电路前移至D级,GRF[rt]需要被从E,M,W转发	D级:直接读出GRF[rt] E级:E级存储的RD2 M级:ALUO@M W级:WDG@W	CMP.D2 RD2@E
E	MFALUAE	ALU的A输入前的转发MUX,需要被从M,W级转发	E级:RD1@E M级:ALUO@M W级:WDG@W	ALU.SA@E
	MFALUBE	ALU的B输入MUX前的转发MUX,及M级存储的RD2.需要被从M,W级转发	E级:RD2@E M级:ALUO@M W级:WDG@W	MALUB.0@E RD2@M
M	MFWDD	DM的输入数据端WDD,接收来自W级的转发	M级:RD2@M W级:WDG@W	WDD@M

转发控制信号生成模块端口定义

信号名	方向	描述
rs_D[4:0]	I	D级指令需求寄存器编号rs
rt_D[4:0]	I	D级指令需求寄存器编号rt
RA1_E[4:0]	I	E级指令需求寄存器编号rs
RA2_E[4:0]	I	E级指令需求寄存器编号rt
RA2_M[4:0]	I	M级指令需求寄存器编号rt(store)
WAG_E[4:0]	I	E级指令产生有效数据之后写回寄存器编号
WAG_M[4:0]	I	M级指令产生有效数据之后写回寄存器编号
WAG_W[4:0]	I	W级指令产生有效数据之后写回寄存器编号
Tnew_E[1:0]	I	E级指令产生有效数据剩余时间
Tnew_M[1:0]	I	M级指令产生有效数据剩余时间
RegWrite_E	I	E级指令是寄存器写指令
RegWrite_M	I	M级指令是寄存器写指令
RegWrite_W	I	W级指令是寄存器写指令
MFRSD_sel	O	D级rs寄存器的转发mux控制信号
MFRTD_sel	O	D级rt寄存器的转发mux控制信号
MFALUAE_sel	O	E级ALU/XALU的A输入源的转发mux控制信号
MFALUBE_sel	O	E级ALU/XALU的B输入源的转发mux控制信号
MFWDD_sel	O	M级DM的WD输入源的转发控制信号

1.6 需求者最晚时间模型

指令集的T_{use}表

instr	T_{use}^{rs}	T_{use}^{rt}
addu	1	1
subu	1	1
ori	1	none
lui	none	none
lw	1	none
sw	1	2
beq	0	0
jal	none	none
jr	0	none
j	none	none

1.7 供给者最早时间模型

指令集的T_{new}表

instr	功能部件	Tnew		
		E	M	W
addu	ALU	1	0	0
subu	ALU	1	0	0
ori	ALU	1	0	0
lui	EXT	0	0	0
lw	DM	2	1	0
beq	CMP	0	0	0
jal	PC	0	0	0
jr	PC	0	0	0
j	PC	0	0	0

1.8 构造暂停/转发策略矩阵

rs,rt策略矩阵

T_{use} \ T_{new}	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
2	F	F	F	F	F	F	F	F	F

1.9 建模Res流水寄存器(T_{new}计数器)

用变量表示T_{use}:

```
Tuse_RS0=beq+jr;
Tuse_RS1=addu+subu+ori+lw+sw;

Tuse_RT0=beq;
Tuse_RT1=addu+subu;
Tuse_RT2=sw;
```

建模T_{new}计数器:

在E级,根据指令(这是一条写寄存器的指令,而且结果有效),然后产生编码值(运算类指令T_{new}初值1,load类指令T_{new}初值2,函数调用类(jal)指令T_{new}初值0).在E/M分别设置1个2位的计数器(命名为Tnew_E,Tnew_M),逐级减1,直至0.

1.10 构造暂停条件表达式

说明 按需求者最晚时间模型T_{use}表定义每一个指令"即将使用寄存器数据的时间上限". 若当前D级指令在D级要使用GRF[rs]的数据,如beq,jr指令,则令Tuse_RS0=1,若在E级要使用GRF[rt]的数据,如计算类指令,则Tuse_RT1=1,以此类推.

满足以下条件则产生暂停信号:

1. 当前D级指令将要使用该数据的时间上限, 小于冲突指令产生有效数据的最短时间
2. 当前D级指令将要使用的数据的寄存器标号与冲突指令写寄存器的编号
3. 冲突指令是一条写寄存器指令

满足以上三条,使这种情况的stall信号置位,所有情况的stall信号用或逻辑组合起来,即为模块最终输出的Stall信号.

```
Tuse_RS0 = beq|jr;
Tuse_RS1 = addu|subu|ori|lw|sw|lui|jal|j;
Tuse_RT0 = beq;
Tuse_RT1 = addu|subu;
```

```
Tuse_RT2 = sw|lui|jal|j|ori|lw|jr;
Stall_RS0_E1 = Tuse_RS0 & (Tnew_E==2'b01) & (rs_D==WAG_E) & RegWrite_E;
Stall_RS0_E2 = Tuse_RS0 & (Tnew_E==2'b10) & (rs_D==WAG_E) & RegWrite_E;
Stall_RS0_M1 = Tuse_RS0 & (Tnew_M==2'b01) & (rs_D==WAG_M) & RegWrite_M;
Stall_RS1_E2 = Tuse_RS1 & (Tnew_E==2'b10) & (rs_D==WAG_E) & RegWrite_E;
Stall_RS = Stall_RS0_E1 | Stall_RS0_E2 | Stall_RS0_M1 | Stall_RS1_E2;
Stall_RT0_E1 = Tuse_RT0 & (Tnew_E==2'b01) & (rt_D==WAG_E) & RegWrite_E;
Stall_RT0_E2 = Tuse_RT0 & (Tnew_E==2'b10) & (rt_D==WAG_E) & RegWrite_E;
Stall_RT0_M1 = Tuse_RT0 & (Tnew_M==2'b01) & (rt_D==WAG_M) & RegWrite_M;
Stall_RT1_E2 = Tuse_RT1 & (Tnew_E==2'b10) & (rt_D==WAG_E) & RegWrite_E;
Stall_RT = Stall_RT0_E1 | Stall_RT0_E2 | Stall_RT0_M1 | Stall_RT1_E2;
Stall = Stall_RS | Stall_RT;
```

1.11 构造转发条件表达式

```
MFRSD_sel = ((rs_D==WAG_E) & |WAG_E & (Tnew_E==2'b00) & RegWrite_E)? 'E2D:
            ((rs_D==WAG_M) & |WAG_M & (Tnew_M==2'b00) & RegWrite_M)? 'M2D:
            ((rs_D==WAG_W) & |WAG_W                   & RegWrite_W)? 'W2D:
                                                                    2'b00;
MFRTD_sel = ((rt_D==WAG_E) & |WAG_E & (Tnew_E==2'b00) & RegWrite_E)? 'E2D:
            ((rt_D==WAG_M) & |WAG_M & (Tnew_M==2'b00) & RegWrite_M)? 'M2D:
            ((rt_D==WAG_W) & |WAG_W                   & RegWrite_W)? 'W2D:
                                                                    2'b00;
MFALUAE_sel = ((RA1_E==WAG_M) & |WAG_M & (Tnew_M==2'b00) & RegWrite_M)? 'M2E:
              ((RA1_E==WAG_W) & |WAG_W                   & RegWrite_W)? 'W2E:
                                                                     2'b00;
MFALUBE_sel = ((RA2_E==WAG_M) & |WAG_M & (Tnew_M==2'b00) & RegWrite_M)? 'M2E:
                ((RA2_E==WAG_W) & |WAG_W                   & RegWrite_W)? 'W2E:
                                                                       2'b00;
MFWDD_sel = ((RA2_M==WAG_W) & |WAG_W & RegWrite_W)? 'W2M:
                                                  2'b00;
```

2 模块规格

2.1 mips.v

说明 顶层文件

端口说明

文件	模块端口说明

mips.v	module mips(clk,reset);
	input clk; //clock
	input reset; //reset

功能定义

序号	功能名称	功能描述
1	复位	reset=1时, 将整个CPU复位

2.2 F级部件

说明 包括 PC(程序计数器)、IM(指令存储器)及相关逻辑。

2.2.1 pc.v

说明 PC(程序计数器)模块,在clk上升沿触发,更改指令地址.

端口说明

信号名	方向	描述
clk	I	处理器时钟
reset	I	复位信号
halt	I	冻结pc信号
npc[31:0]	I	下一个pc的值
pc[31:0]	O	F级的pc值, 输出给IM

功能定义

序号	功能名称	功能描述
1	存储npc	存储npc的值
2	输出pc	输出存储的pc值
3	复位	reset=1时, 将存储的pc复位, PC指向0x0000_3000,此处为第一条指令的地址。
4	冻结pc	当halt=1时,冻结pc的值为当前pc值

2.2.2 im.v

说明

- IM(指令存储器)模块,用存储器(IMmemory)实现容量为容量为4KB（32bit×1024字）.
- 起始地址：0x0000_3000。
- 因 IM 实际地址宽度仅为 10 位，故将PC输出的[11:2]位作为IM的输入。
- 起始时,用$readmemb指令从code.txt中读取指令的16进制代码,放入IMmemory中

端口说明

信号名	方向	描述
pcAddr[9:0]	I	输入F级pc的[11:2]位, 为指令存储器寻址的地址
instr[31:0]	O	输出32位的指令

功能定义

序号	功能名称	功能描述
1	读取并存储指令	用$readmemb指令从code.txt中读取指令的16进制代码 并存储在容量为4KB(32bitx1024字)的存储器IMmemory中
2	输出指令	根据pcAddr[9:0]的值, 从存储器中找到对应的指令并输出

2.3 D级部件

2.3.1 GRF

说明

- GRF(通用寄存器组)模块,用具有写使能的寄存器组实现，寄存器总数为 32 个,每个寄存器32bit。
- 0 号寄存器的值始终保持为 0。其他寄存器初始值均为 0。

端口说明

信号名	方向	描述
clk	I	处理器时钟
reset	I	复位信号
RA1[4:0]	I	读寄存器文件时的第一个寄存器地址
RA2[4:0]	I	读寄存器文件时的第二个寄存器地址
WA[4:0]	I	写寄存器文件时的寄存器地址
WD[31:0]	I	寄存器文件写入数据
WE	I	寄存器文件写使能
pc[31:0]	I	当前pc
RD1[31:0]	O	读寄存器文件时的第一个寄存器输出
RD2[31:0]	O	读寄存器文件时的第二个寄存器输出

功能定义

序号	功能名称	功能描述
1	读寄存器	RD1[31:0]输出RA1[4:0]所寻址的寄存器 RD2[31:0]输出RA2[4:0]所寻址的寄存器 用assign持续赋值实现 (地址来源于D级流水线寄存器)
2	写寄存器	当时钟上升沿到来时，并且WE有效时， WD被写入WA[4:0]所寻址的寄存器 (写功能的地址,数据,使能信号,pc值均来源于W级寄存器)
3	复位	reset=1时，32个寄存器单元清零(同步复位)

2.3.2 CMP模块

端口说明

信号名	方向	描述
op[5:0]	I	当前D级指令种类
rt[4:0]	I	当前D级指令rt域
D1[31:0]	I	GRF[rs]的值(转发后)
D2[31:0]	I	GRF[rt]的值(转发后)
branch	O	若要跳转,则置位

功能定义

序号	功能名称	功能描述
1	判断b类指令类型	有beq,bne,blez,bgtz,bltz,bgez这6种
2	判断是否跳转	根据不同指令跳转条件判断是否跳转

跳转指令控制信号真值表

signal name	pre_nPC_sel	branch	nPC_sel
b_type	01	0	00
b_type	01	1	01
jr	10	X	10
jal	11	X	11
j	11	X	11
jalr	10	X	10

2.3.3 ImmExtUnit

说明 16位立即数的扩展模块,具有无符号扩展、有符号扩展、加载到高位的功能,模块内部用函数实现。

端口说明

信号名	方向	描述
imm16[15:0]	I	输入D级指令中的16位立即数
EXTctrl[1:0]	I	选择某一扩展功能 00: 无符号扩展 01: 有符号扩展 10: 加载到高位
EXTresult[31:0]	O	输出扩展之后的结果

功能定义

序号	功能名称	功能描述
1	UnsignedExt	把16位输入进行无符号扩展至32位
2	SignedExt	把16位输入进行有符号扩展至32位
3	lui	把16位输入加载到32位输出的高位, 低16位零填充

2.3.4 npcctrl

说明 nPC_sel控制信号模块,输入为2位的pre_nPC_sel(仅为cpuctrl中判断指令的结果, 结合beq指令的zero比较结果,生成正确的EXT功能选择信号nPC_sel.

nPC_sel控制信号真值表

signal name	pre_nPC_sel	branch	nPC_sel
b_type	01	0	00
b_type	01	1	01
jr	10	X	10
jal	11	X	11
j	11	X	11
jalr	10	X	10

2.3.5 Stall

说明 暂停信号生成模块.

端口说明

信号名	方向	描述
op[5:0]	I	当前指令的op码
func[5:0]	I	当前指令的func码
notnop	I	当前指令若不是nop指令,则置位
Tnew_D[1:0]	I	D级指令产生有效结果的剩余周期数
Tnew_E[1:0]	I	E级指令产生有效结果的剩余周期数
Tnew_M[1:0]	I	M级指令产生有效结果的剩余周期数
rs_D[4:0]	I	D级指令的rs域
rt_D[4:0]	I	D级指令的rt域
WAG_E[4:0]	I	E级指令将要写入的寄存器地址
WAG_M[4:0]	I	M级指令将要写入的寄存器地址
RegWrite_E	I	E级指令是否对寄存器有写入
RegWrite_M	I	M级指令是否对寄存器有写入
Stall	O	输出暂停信号

2.4 E级部件

2.4.1 ALU

说明

- ALU算术逻辑单元模块,提供 32 位加、减、或、与运算,直接输出第二个运算数,及大小比较功能。
- 不支持溢出

端口说明

信号名	方向	描述
SourceA[31:0]	I	输入的第一个运算数,来源是转发mux(MFALUAE)
SourceB[31:0]	I	输入的第二个运算数,来源是功能mux(MALUB)
sa[4:0]	I	指令的shamt域,用于逻辑左移、逻辑右移 来源是E级寄存器存储的指令的shamt
ALUctrl[3:0]	I	ALU的功能选择信号 来源是E级寄存器存储的ALU功能选择信号 0000: 无符号加法运算 0001: 无符号减法运算 0010: 或运算 0011: 与运算 0100: 小于置位signed 0101: 小于置位unsigned 0110: 逻辑左移 0111: 逻辑右移 1010: 算术右移 1011: 可变算术右移 1100: 可变逻辑左移 1101: 可变逻辑右移
ALUresult[31:0]	O	输出运算结果

2.5 M级部件

2.5.1 DM

说明

- DM(数据存储器)模块,使用带有使能信号的存储器DMmemory实现，容量为4KB（32bit×1024字）
- 起始地址：0x0000_0000。
- 起始时整个存储器清零.

端口说明

信号名	方向	描述
clk	I	处理器时钟
reset	I	复位信号
LdStType[2:0]	I	load store type
Addr_full[31:0]	I	读写存储器文件时的地址 来源是M级寄存器存储的ALU运算结果
WD[31:0]	I	存储器文件的写入数据,来源是转发mux(MFWDD)
RE	I	存储器文件的读使能,来源是M级寄存器存储的控制信号
WE	I	存储器文件的写使能,来源是M级寄存器存储的控制信号
pc[31:0]	I	M级寄存器存储的指令的地址
RD[31:0]	O	读存储器文件时的存储单元输出

功能定义

序号	功能名称	功能描述
1	读存储器	RE有效时,RD[31:0]输出Addr所寻址的存储单元
2	写存储器	当时钟上升沿到来时, 并且WE有效时, WD[31:0]被写入Addr[9:0]所寻址的存储单元 当WE无效时, 忽略WD, 保持当前状态
3	复位	reset=1时， 存储器32个存储单元清零

2.5.2 DM内置BE扩展单元

端口说明

用于store类型指令,将指令要写入存储器的字节位转换为独热码

信号	方向	描述
which_byte[1:0]	I	ALU算出的地址的末两位,来源于M级寄存器
Type[2:0]	I	store指令的种类,LdStType_M 000:sw 001:sb 011:sh
BE[3:0]	O	译码后的写使能信号,传递给DM

功能定义

sw,sh,sb指令地址与BE信号的译码表

指令类型	输入端口Type	地址[1:0]	输出端口BE[3:0]	用途
sw	000	XX	1111	WD整个字写入DM对应字空间
sh	011	0X	0011	WD[15:0]写入DM对应低半字
		1X	1100	WD[15:0]写入DM对应高半字
sb	001	00	0001	WD[7:0]写入DM对应byte0
		01	0010	WD[7:0]写入DM对应byte1
		10	0100	WD[7:0]写入DM对应byte2
		11	1000	WD[7:0]写入DM对应byte3

2.5.3 DataEXT

说明: load类型指令的数据扩展模块,输入的datai[31:0]是从DM中获取的读出数据.根据op指令控制选择合适的方式扩展后输出.

端口说明

信号	方向	描述
LSB[1:0]	I	最低2位地址,来自W级寄存器
datai[31:0]	I	输入32位数据
op[2:0]	I	数据扩展控制码,是顶层接线LdStType_W 000: 无扩展(lw) 001: 无符号字节数据扩展(lbu) 010: 符号字节数据扩展(lb) 011: 无符号半字数据扩展(lhu) 100: 符号半字数据扩展(lh)
datao[31:0]	O	扩展后的32位数据输出

(备注:该模块在MEM/WB之后,而不是在DM之后)

2.6 流水线寄存器

2.6.1 IF/ID级流水线寄存器

说明 D级流水寄存器,位于IF(读取指令)和ID(指令译码)之间.

端口说明

信号名	方向	描述
clk	I	cpu时钟
reset	I	复位信号
halt	I	冻结信号
instr_F[31:0]	I	32位指令码
pc_F[31:0]	I	来源于IF段的当前pc
pc_D[31:0]	O	输出存储在D级的pc
op_D[5:0]	O	instr[31:26],op
rs_D[4:0]	O	instr[25:21],rs_D
rt_D[4:0]	O	instr[20:16],rt_D
rd_D[4:0]	O	instr[15:11],rd_D
shamt_D[4:0]	O	instr[10:6],shamt
func_D[5:0]	O	instr[5:0],func
immediate_D[15:0]	O	instr[15:0],imm16
instr_index_D[25:0]	O	instr[25:0],imm26
PC4_D[31:0]	O	PC4_D
PC8_D[31:0]	O	PC8_D
Tnew_D[1:0]	O	当前D级指令产生有效值的剩余周期数

功能定义

序号	功能名称	功能描述
1	流水线寄存器	存储D级指令相关信息
2	生成D级T_{new}	根据供给者时间模型表, 生成当前D级指令的产生有效结果的时间上限

2.6.2 ID/EX级流水线寄存器

说明 E级流水寄存器位于ID(指令译码)和EX(计算)之间.

端口说明

信号	方向	描述
clk	I	cpu时钟
reset	I	复位信号
clr	I	E级流水线寄存器清零信号
pc_D[31:0]	I	来源于D级流水寄存器存储的pc
rs_D[4:0]	I	来源于D级流水寄存器的rs
rt_D[4:0]	I	来源于D级流水寄存器的rt
shamt_D[4:0]	I	来源于D级流水寄存器的shamt
MWAGE[4:0]	I	来源于功能mux(MWAGE)的输出
EXT_D[31:0]	I	来源于EXT扩展单元的输出
MFRSD[31:0]	I	来源于D级转发mux(MFRSD)的输出
MFRTD[31:0]	I	来源于D级转发mux(MFRTD)的输出
PC4_D[31:0]	I	来源于D级存储的PC+4
PC8_D[31:0]	I	来源于D级存储的PC+8
RegDst_D[1:0]	I	来源于ControlUnit输出的控制信号
ALUSrc_D[1:0]	I	来源于ControlUnit输出的控制信号
MemtoReg_D[1:0]	I	来源于ControlUnit输出的控制信号
LdStType_D[2:0]	I	来源于ControlUnit输出的控制信号
RegWrite_D	I	来源于ControlUnit输出的控制信号
MemRead_D	I	来源于ControlUnit输出的控制信号
MemWrite_D	I	来源于ControlUnit输出的控制信号
ALUctrl_D[3:0]	I	来源于ControlUnit输出的控制信号
Tnew_D[1:0]	I	当前E级指令在D级时产生有效结果的时间上限
pc_E[31:0]	I	输出E级存储的pc
RA1_E[4:0]	O	输出E级存储的RA1
RA2_E[4:0]	O	输出E级存储的RA2
shamt_E[4:0]	O	输出E级存储的shamt域
WAG_E[4:0]	O	输出E级存储的WAG
EXT_E[31:0]	O	输出E级存储的EXT
RD1_E[31:0]	O	输出E级存储的RD1
RD2_E[31:0]	O	输出E级存储的RD2
PC4_E[31:0]	O	输出E级存储的PC+4
PC8_E[31:0]	O	输出E级存储的PC+8
RegDst_E[1:0]	O	输出E级存储的控制信号
ALUSrc_E[1:0]	O	输出E级存储的控制信号
MemtoReg_E[1:0]	O	输出E级存储的控制信号
LdStType_E[2:0]	O	输出E级存储的控制信号
RegWrite_E	O	输出E级存储的控制信号
MemRead_E	O	输出E级存储的控制信号
MemWrite_E	O	输出E级存储的控制信号
ALUctrl_E[3:0]	O	输出E级存储的控制信号
Tnew_E[1:0]	O	输出E级指令产生有效信号的时间上限

功能定义

序号	功能名称	功能描述
1	流水线寄存器	存储E级指令相关信息
2	保存Tnew	保存并传递D级寄存器生成的Tnew

2.6.3 EX/MEM级流水线寄存器

说明 M级流水寄存器位于EX(计算)和MEM(访存)之间.

端口说明

信号	方向	描述
clk	I	cpu时钟
reset	I	复位信号
pc_E[31:0]	I	来源于E级流水寄存器存储的pc
RA2_E[4:0]	I	来源于E级存储的RA2
WAG_E[4:0]	I	来源于E级存储的WAG
RD2_E[31:0]	I	来源于E级存储的RD2
ALUO_E[31:0]	I	来源于E级部件ALU的运算结果
PC4_E[31:0]	I	来源于E级存储的PC+4
PC8_E[31:0]	I	来源于E级存储的PC+8
MemtoReg_E[1:0]	I	来源于E级存储的控制信号
LdStType_E[2:0]	I	来源于E级存储的控制信号
RegWrite_E	I	来源于E级存储的控制信号
MemRead_E	I	来源于E级存储的控制信号
MemWrite_E	I	来源于E级存储的控制信号
Tnew_E[1:0]	I	来源于E级指令在E级时产生有效结果的时间上限
pc_M[31:0]	I	输出M级存储的pc
RA2_M[4:0]	O	输出M级存储的RA2
WAG_M[4:0]	O	输出M级存储的WAG
RD2_M[31:0]	O	输出M级存储的RD2
ALUO_M[31:0]	O	输出M级存储的ALU运算结果
PC4_M[31:0]	O	输出M级存储的PC+4
PC8_M[31:0]	O	输出M级存储的PC+8
MemtoReg_M[1:0]	O	输出M级存储的控制信号
LdStType_M[2:0]	O	输出M级存储的控制信号
RegWrite_M	O	输出M级存储的控制信号
MemRead_M	O	输出M级存储的控制信号
MemWrite_M	O	输出M级存储的控制信号
Tnew_M[1:0]	O	计算并输出M级指令产生有效结果的时间上限

功能定义

序号	功能名称	功能描述
1	流水线寄存器	存储M级指令相关信息
2	生成并传递Tnew	保存,递减并传递E级寄存器生成的Tnew 若Tnew_E已经到0,则不再递减

2.6.4 MEM/WB流水线寄存器

说明 W级流水寄存器位于MEM(访存)和WB(写回)之间.

端口说明

信号	方向	描述
clk	I	cpu时钟
reset	I	复位信号
pc_M[31:0]	I	来源于M级流水寄存器存储的pc
RD_M[31:0]	I	来源于M级部件DM的输出
ALUO_M[31:0]	I	来源于M级存储的ALU运算结果
WAG_M[4:0]	I	来源于M级存储的WAG
PC4_M[31:0]	I	来源于M级存储的PC+4
PC8_M[31:0]	I	来源于M级存储的PC+8
MemtoReg_M[1:0]	I	来源于M级存储的控制信号
LdStType_M[2:0]	O	来源于M级存储的控制信号
RegWrite_M	I	来源于M级存储的控制信号
pc_W[31:0]	I	输出W级流水寄存器存储的pc
RD_W[31:0]	O	输出W级存储的DM的输出
ALUO_W[31:0]	O	输出W级存储的ALU运算结果
WAG_W[4:0]	O	输出W级存储的WAG
PC4_W[31:0]	O	输出W级存储的PC+4
PC8_W[31:0]	O	输出W级存储的PC+8
MemtoReg_W[1:0]	O	输出W级存储的控制信号
LdStType_W[2:0]	O	输出W级存储的控制信号
RegWrite_W	O	输出W级存储的控制信号

功能定义

序号	功能名称	功能描述
1	流水线寄存器	存储W级指令相关信息

3 控制单元

3.1 ControlUnit控制单元

端口定义

信号	方向	描述
instr_D[31:0]	I	输入D级指令
RegDst_D[1:0]	O	MWAGE功能mux的选择信号
ALUSrc_D[1:0]	O	MALUB功能mux的选择信号
MemtoReg_D[1:0]	O	MMTR功能mux的选择信号
LdStType[2:0]	O	load/store类型指令的种类信号
RegWrite_D	O	寄存器写使能
MemRead_D	O	内存读使能
MemWrite_D	O	内存写使能
pre_nPC_sel_D[1:0]	O	MPC功能mux选择信号的预选择信号
ALUctrl_D[3:0]	O	4位ALU控制信号
EXTctrl[1:0]	O	2位扩展单元控制信号

功能定义

序号	功能名称	功能描述
1	判断指令	根据D级指令码判断指令
2	输出控制信号	根据指令类型输出对应控制信号

指令与控制信号表

Instr	RegDst	ALUSrc	MemtoReg	LdStType	RegWrite	MemRead	MemWrite	pre_nPC_sel	ALUctrl	EXTctrl	Notes
addu	01	00	00	000	1	0	0	00	0000	XX	
subu	01	00	00	000	1	0	0	00	0001	XX	
ori	00	01	00	000	1	0	0	00	0010	00	
lui	00	01	00	000	1	0	0	00	0010	10	
lw	00	01	01	000	1	1	0	00	0000	01	
sw	XX	01	XX	000	0	0	1	00	0000	01	
beq	XX	XX	XX	000	0	0	0	01	XXXX	01	
jal	10	XX	10	000	1	0	0	11	XXXX	XX	
j	XX	XX	XX	000	0	0	0	11	XXXX	XX	
jr	XX	XX	XX	000	0	0	0	10	XXXX	XX	
lb	00	01	01	010	1	1	0	00	0000	01	
lbu	00	01	01	001	1	1	0	00	0000	01	
lh	00	01	01	100	1	1	0	00	0000	01	
lhu	00	01	01	011	1	1	0	00	0000	01	
sb	XX	01	XX	001	0	0	1	00	0000	01	
sh	XX	01	XX	011	0	0	1	00	0000	01	
sll	01	00	00	000	1	0	0	00	0110	XX	
srl	01	00	00	000	1	0	0	00	0111	XX	
sra	01	00	00	000	1	0	0	00	1010	XX	
sllv	01	00	00	000	1	0	0	00	1100	XX	
srlv	01	00	00	000	1	0	0	00	1101	XX	
srav	01	00	00	000	1	0	0	00	1011	XX	
slt	01	00	00	000	1	0	0	00	0100	XX	
slti	00	01	00	000	1	0	0	00	0100	01	
bne	XX	XX	XX	000	0	0	0	01	XXXX	01	
blez	XX	XX	XX	000	0	0	0	01	XXXX	01	
bgtz	XX	XX	XX	000	0	0	0	01	XXXX	01	
bltz	XX	XX	XX	000	0	0	0	01	XXXX	01	
bgez	XX	XX	XX	000	0	0	0	01	XXXX	01	
jalr	01	XX	10	000	1	0	0	10	XXXX	XX	jalr:RegDst=01:rd,RegDst

4 测试程序

测试程序1

说明 无跳转指令，无暂停的情况,测试ori、addu、subu、lui、lw、sw指令和转发机制是否正常。

```
ori $5 $0 0x1234
ori $6 $0 0x2345
addu $7 $5 $6
ori $8 $0 0x3579
subu $9 $8 $5
sw $7 0($0)
lw $10 0($0)
sw $8 4($0)
sw $9 8($0)
lw $11 4($0)
lw $12 8($0)
addu $13 $10 $11
lui $14 0xffff
```

期望输出:

@00003000: $ 5 <= 00001234

@00003004: $ 6 <= 00002345

@0000300c: $ 7 <= 00003579

@00003008: $ 8 <= 00003579

@00003010: $ 9 <= 00002345

@00003014: ⋆00000000 <= 00003579

@00003018: $10 <= 00003579

@0000301c: ⋆ 00000004 <= 00003579

@00003020: ⋆ 00000008 <= 00002345

@00003024: $11 <= 00003579

@00003028: $12 <= 00002345

@0000302c: $13 <= 00006af2

@00003030: $14 <= ffff0000

测试程序2

说明 测试beq的暂停机制是否正常

```
label1:
  ori $5 $0 0x1234
  addu $6 $6 $5
  beq $5 $6 label1
  ori $7 $5 0x2345
  ori $8 $7 0x3456
  beq $7 $8 label1
  lui $8 0x6789
```

期望输出:

@00003000: $ 5 <= 00001234

@00003004: $ 6 <= 00001234

@0000300c: $ 7 <= 00003375

@00003000: $ 5 <= 00001234

@00003004: $ 6 <= 00002468

@0000300c: $ 7 <= 00003375

@00003010: $ 8 <= 00003777

@00003018: $ 8 <= 67890000

测试程序3

说明 测试jal、jr和j的跳转机制和延迟槽是否正常

```
.text
  ori $5 $0 0x1234
  ori $6 $0 0x2345
  subu $7 $6 $5
  jal label1
  ori $8 $0 0x6789
  addu $6 $8 $7
  j label2
  lui $16 0x1616
label1:
  beq $8 $6 label3
  lui $9 0x5678
  addu $10 $9 $8
label2:
  subu $11 $10 $9
  jr $ra
  addu $12 $11 $10
label3:
  ori $14 $0 0x1414
  ori $15 $0 0x1515
```

期望输出: @00003000: $ 5 <= 00001234

@00003004: $ 6 <= 00002345

@00003008: $ 7 <= 00001111

@0000300c: $31 <= 00003014

@00003010: $ 8 <= 00006789

@00003024: $ 9 <= 56780000

@00003028: $10 <= 56786789

@0000302c: $11 <= 00006789

@00003034: $12 <= 5678cf12

@00003014: $ 6 <= 0000789a

@0000301c: $16 <= 16160000

@0000302c: $11 <= 00006789

@00003034: $12 <= 5678cf12

@00003014: $ 6 <= 0000789a

@0000301c: $16 <= 16160000

@0000302c: $11 <= 00006789

@00003034: $12 <= 5678cf12

@00003014: $ 6 <= 0000789a

@0000301c: $16 <= 16160000

@0000302c: $11 <= 00006789

@00003034: $12 <= 5678cf12

@00003014: $ 6 <= 0000789a

@0000301c: $16 <= 16160000

@0000302c: $11 <= 00006789

@00003034: $12 <= 5678cf12

...

以此循环

测试程序4

说明 检测sw和lw指令的暂停和转发是否正常

```
ori $5 $0 0x0004
ori $6 $0 0x0008
addu $7 $5 $6
sw $7 0($5)
subu $8 $7 $6
sw  $8 4($8)
lw $9 4($8)
ori $14 $0 0x2222
sw $9 12($0)
lw $10 0($5)
addu $11 $9 $10
```

期望输出:

225@00003000: $ 5 <= 00000004

275@00003004: $ 6 <= 00000008

325@00003008: $ 7 <= 0000000c

325@0000300c: ⋆ 00000004 <= 0000000c

425@00003010: $ 8 <= 00000004

425@00003014: ⋆ 00000008 <= 00000004

525@00003018: $ 9 <= 00000004

575@0000301c: $14 <= 00002222

575@00003020: ⋆ 0000000c <= 00000004

675@00003024: $10 <= 0000000c

775@00003028: $11 <= 00000010

5 思考题

在本实验中你遇到了哪些不同指令组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？请有条理的罗列出来。(非常重要)

测试类型	前序指令	冲突位置	冲突寄存器	解决办法	测试序列
$R-M-RS$	subu	MEM	rs	转发GRF[rs]从M级到E级 转发GRF[rt]从W级到E级	ori \$5 \$0 0x1234 ori \$6 \$0 0x2345 subu \$7 \$6 \$5 addu \$8 \$7 \$6
$R-W-RS$	subu	MEM	rs	转发GRF[rs]从W级到E级	subu \$7 \$6 \$5 lui \$9 0x1234 addu \$8 \$7 \$6
$R-M-RT$	subu	MEM	rs,rt	转发GRF[rs]从W到D级 转发GRF[rt]从M到D级	ori \$6 \$0 0x2345 subu \$7 \$6 \$5 lui \$8 0x3456 addu \$9 \$6 \$7
$I-M-RS$	ori	MEM	rs	转发GRF[rs]从M级到E级	ori \$6 \$0 0x0004 sw \$5 0(\$6)
$I-M-RS$	ori	MEM	rs	转发GRF[rs]从M级到D级	ori \$5 \$0 0x1234 ori \$6 \$0 0x2345 sw \$5 0(\$0)
$I-M-RS$ $I-W-RS$	ori	MEM	rs,rt	转发GRF[rt]从M级到D级(第一句ori给addu) 转发GRF[rs]从M级到E级(第一句ori给第二句ori) 转发GRF[rs]从W到E级(第一句ori给addu) 转发GRF[rt]从M到E级(第二句ori给addu)	ori \$6 \$0 0x1234 ori \$5 \$6 0x2345 addu \$7 \$5 \$6
$I-D-RS/$ $I-D-RT$	ori	CMP	rs,rt	暂停一个周期,并且 转发GRF[rs]从W级到D级 转发GRF[rt]从M级到D级	ori \$5 \$0 0x1234 ori \$6 \$0 0x1234 beq \$5 \$6 label
$LD-D-RS$	load	CMP	rs	暂停两个时钟周期,并且 转发GRF[rs]从W级到D级	lw \$7 0(\$0) beq \$7 \$5 label
$LD-D-RS$	load	GRF	rs	暂停两个时钟周期,并且 转发GRF[rs]从W级到D级	lw \$9 0(\$5) jr \$9
$LD-D-RS$	load	CMP	rs	暂停一个时钟周期,并且 转发GRF[rs]从W级到D级	lw \$7 0(\$0) lui \$4 0x3456 beq \$7 \$5 label1
$LD-M-RS$	load	MEM	rs	暂停一个周期,并且 转发GRF[rs]从W级到E级	lw \$7 0(\$0) addu \$8 \$7 \$5
$LD-W-RS$	load	MEM	rs	转发GRF[rs]从W级到E级	lw \$7 0(\$0) lui \$8 0x3456 addu \$8 \$7 \$5