CPU 设计文档

一、 数据通路设计

（1）pc（程序计数器）

模块端口说明如下：

表 1　pc 端口说明

序号	信号名	方向	描述
1	Clk	I	时钟信号
2	Reset	I	复位信号
3	En	I	使能信号
4	NPC[31:0]	I	下一个PC值
5	PC[31:0]	O	当前的PC值

模块功能定义如下：

表 2　pc 功能定义

序号	功能名称	功能描述
1	更新PC值	当时钟上升沿到来时，将NPC写入PC
2	输出	输出当前PC的值

（2）im（指令存储器）

模块端口说明如下：

表 3　im 端口说明

序号	端口名	方向	描述
1	Addr[11:2]	I	当前PC的[11:2]位
2	Instr[31:0]	O	指令存储器中以Addr为地址的指令

模块功能定义如下：

表 4　im 功能定义

序号	功能名称	功能描述
1	初始化	将code.txt中的内容读入指令存储器中
2	输出	输出指令存储器中Addr所对应地址的指令的值

（2）grf（通用寄存器组）

模块端口说明如下：

表 5　grf 端口说明

序号	信号名	方向	描述
1	RAddr1[4:0]	I	读寄存器地址1
2	RAddr2[4:0]	I	读寄存器地址2
3	WAddr[4:0]	I	写寄存器地址
4	WData[31:0]	I	写入寄存器的数据
5	RegWrite	I	寄存器写使能信号
6	Clk	I	时钟信号
7	RData1[31:0]	O	输出地址RAddr1的寄存器中的数据
8	RData2[31:0]	O	输出地址RAddr2的寄存器中的数据

模块功能定义如下：

表 6　grf 功能定义

序号	功能名称	功能描述
1	读寄存器	输出端口RData1和RData2分别输出以输入信号RAddr1和RAddr2为地址的寄存器中的数据
2	写寄存器	当始终上升沿到来时，若写使能信号为1，则将输入信号WData中的数据写入以输入信号WAddr为地址的寄存器中

（3）alu（算术逻辑单元）

模块端口说明如下：

表 7　alu 端口说明

序号	信号名	方向	描述
1	Data1[31:0]	I	参与ALU运算的第一个值
2	Data2[31:0]	I	参与ALU运算的第二个值
3	ALUOp[2:0]	I	ALU功能的选择信号 000 ： ALU进行加法运算 001 ： ALU进行减法运算 010 ： ALU进行或运算 011 ： ALU进行与运算
4	ALUResult[31:0]	O	ALU的计算结果

模块功能定义如下：

表 8　alu 功能定义

序号	功能名称	功能描述
1	加法运算	ALUResult = Data1 + Data2
2	减法运算	AUResult = Data1 - Data2
3	与运算	ALUResult = Data1 & Data2
4	或运算	ALUResult = Data1 | Data2

（3）dm（数据存储器）

模块端口说明如下：

表 9　dm 端口说明

序号	信号名	方向	描述
1	Addr[4:0]	I	数据存储器读写的地址
2	WData[31:0]	I	将要写进数据存储器的数据
3	MemWrite	I	数据存储器的写使能端
4	Clk	I	时钟信号
5	Reset	I	复位信号
6	RData[31:0]	O	输出从数据存储器中读取的值

模块功能定义如下：

表 10　dm 功能定义

序号	功能名称	功能描述
1	读数据存储器	输出端口Data输出数据存储器在地址为MemAddr处的数据
2	写数据存储器	当时钟上升沿到来时，若MemWrite为1，且Reset信号为0，则将输入信号MemData中的数据写入数据存储器在MemAddr所对应的地址中
3	复位	当时钟上升沿到来时，若Reset信号为1，将数据存储器的内容置为0

（4）EXT（数据扩展单元）

模块端口说明如下：

表 11　ext 端口说明

序号	信号名	方向	描述
1	In[15:0]	I	扩展单元的输入信号
2	ExtOp[1:0]	I	扩展方式的选择信号 00 ： 进行符号扩展 01 ： 进行零扩展 10 ： 进行低位零扩展
3	Out[31:0]	O	扩展单元的输出信号

模块功能定义如下：

表 12　ext 功能定义

序号	功能名称	功能描述
1	符号扩展	输出信号的低16位与输入信号相同， 高16位为输入信号的符号位
2	零扩展	输出信号的低16位与输入信号相同，高16位为0
3	低位零扩展	输出信号的高16位与输入信号相同，低16为为0

(5) NPC

模块端口定义如下：

表 13 NPC 端口说明

序号	端口名称	方向	说明
1	PCplus4[31:0]	I	当前的PC值加4
2	I_Addr[25:0]	I	当前32位指令的低26位
3	R_Addr[31:0]	I	保存在寄存器中的地址
4	NPCSel[1:0]	I	选择下一个pc的值 00 ： 选择PC+8作为下一个PC的值 01 ： 选择PC + sign_extend(Addr[15:0]||02)作为下一个PC的值 10 ：选择{PC[31:28],Addr}作为下一个PC的值 11： 选择R_Addr作为下一个PC的值
5	Equal	I	beq的分支条件是否成立
6	NPC	O	下一个PC的值（若发生分支或跳转）
7	PCplus8	O	当前的PC值加8

模块功能定义如下：

表 14 NPC 功能定义

序号	功能名称	功能描述
1	输出	根据选择信号输出下一个PC的值（若发生分支或跳转）

（6）CMP

模块端口说明如下：

序号	信号名	方向	描述
1	Data1[31:0]	I	比较的数据1
2	Data2[31:0]	I	比较的数据2
3	CMPOp[2:0]	I	比较方式的选择信号 000 ： 结果为1 001 ： 结果为Data1 == Data2 010 ： 结果为Data1 != Data2
4	CMPResult	O	输出比较结果

模块功能定义如下：

序号	功能名称	功能描述
1	输出	根据选择信号输出比较结果

二、 控制器设计

（1） 控制器

ctrl_D 端口说明如下：

序号	信号名	方向	描述
1	Instr[31:0]	I	F/D级流水线寄存器中的Instr值
2	EXTOp[1:0]	O	扩展单元的功能选择信号
3	NPCSel[1:0]	O	NPC的功能选择信号
4	isBranch	O	是否发生跳转或分支

ctrl_E 端口说明如下：

序号	信号名	方向	描述
1	Instr[31:0]	I	D/E级流水线寄存器的Instr值
2	ALUOp[2:0]	O	ALU功能选择信号
3	ALUSrc[1:0]	O	ALU的运算数据选择信号

ctrl_M 端口说明如下：

序号	信号名	方向	描述
1	Instr[31:0]	I	E/M级流水线寄存器的Instr值
2	MemWrite	O	数据存储器的写使能信号

ctrl_W 端口说明如下：

序号	信号名	方向	描述
1	Instr[31:0]	I	M/W级流水线寄存器中的Instr值
2	RegWrite	O	寄存器的写使能信号
3	RegDst[1:0]	O	寄存器写入地址选择信号
4	RegSrc[1:0]	O	寄存器写入数据选择信号

控制信号意义如下：

序号	控制信号	意义
1	NPCSel[1:0]	控制分支的信号，分支指令需要将该信号置为1
2	RegWrite	寄存器写使能信号，但需要些寄存器时将此信号置为1
3	RegDst[1:0]	选择寄存器的写入地址， 当此信号为00时，选择指令的rt字段（[20:16]）为寄存器的写入地址； 当此信号为01时，选择指令的rd字段（[15:11]）为寄存器的写入地址； 当此信号为10时，选择0x1f为寄存器的写入地址
4	RegSrc[1:0]	选择寄存器的写入数据， 当此信号为00时，选择ALU的计算结果作为寄存器堆的写入值； 当此信号为01时，选择从数据存储器中取出的信号作为寄存器堆的写入值； 当此信号为10时，选择PC+4作为寄存器堆的写入值
5	ExtOp[1:0]	Ext功能选择信号，根据指令需要进行的扩展类型来设置为相应的值
6	ALUOp[2:0]	ALU功能选择信号，根据指令需要执行的运算种类来设置相应的值
7	ALUSrc	当此信号为0时，选择指令的rt字段（[20:16]）为地址的寄存器中的数据作为ALU的第二个运算数； 当此信号为1时，选择经扩展后的立即数作为ALU的第二个运算数。
8	MemWrite	数据存储器写使能信号，当需要写数据存储器时将此信号置为1

（2） 阻塞控制器

阻塞控制器端口说明如下：

序号	信号名	方向	描述
1	FD_Instr[31:0]	I	F/D级流水线寄存器中的Instr值
2	DE_Instr[31:0]	I	D/E级流水线寄存器中的Instr值
3	EM_Instr[31:0]	I	E/M级流水线寄存器中的Instr值
4	StallPC	O	是否阻塞PC
5	StallFD	O	是否阻塞F/D流水线寄存器
6	FlushDE	O	是否清空D/E流水线寄存器中的值

阻塞发生条件如下:

IF/ID当前指令			ID/EX(Tnew)				EX/MEM(Tnew)				MEM/	
指令类型	源寄存器	Tuse	cal_r 1/rd	cal_i 1/rt	load 2/rt	jal 0/31	cal_r 0/rd	cal_i 0/rt	load 1/rt	jal 0/31	cal_r 0/rd	cal_i 0/rt
branch	rs/rt	0	暂停	暂停	暂停				暂停			
jr	rs	0	暂停	暂停	暂停				暂停			
cal_r	rs/rt	1			暂停							
cal_i	rs	1			暂停							
load	rs	1			暂停							
store	rs	1			暂停							
	rt	2										

(3) 转发控制器

转发控制器端口定义如下：

序号	信号名	方向	描述
1	FD_Instr[31:0]	I	F/D级流水线寄存器中的Instr值
2	DE_Instr[31:0]	I	D/E级流水线寄存器中的Instr值
3	EM_Instr[31:0]	I	E/M级流水线寄存器中的Instr值
4	MW_Instr[31:0]	I	M/W级流水线寄存器中的Instr值
5	BypassDrs	O	转发信号1，控制D阶段相等比较的输入
6	BypassDrt	O	转发信号2，控制D阶段相等比较的输入
7	BypassErs	O	转发信号3，控制ALU的输入
8	BypassErt	O	转发信号4，控制ALU的输入
9	BypassMrt	O	转发信号5，控制DM的写入值

转发条件如下：

流水级	源寄存器	涉及指令	转发MUX	控制信号	输入0
IF/ID	**rs**	branch, jr	MUXB_D_rs	BypassDrs	D_GPR_RData1
	rt	branch	MUXB_D_rt	BypassDrt	D_GPR_RData2
ID/EX	**rs**	cal_r, cal_i, ld, st	MUXB_E_rs	BypassErs	DE_RData1
	rt	cal_r, st	MUXB_E_rt	BypassErt	DE_RData2
EX/MEM	**rt**	store	MUXB_M_rt	BypassMrt	EM_M_Wdata

ID/EX	EX/MEM(Tnew)		
jal 0/31	**cal_r 0/rd**	**cal_i 0/rt**	**jal 0/31**
DE_PCplus8	EM_ALUOut	EM_ALUOut	EM_PCplus8
DE_PCplus8	EM_ALUOut	EM_ALUOut	EM_PCplus8
	EM_ALUOut	EM_ALUOut	EM_PCplus8
	EM_ALUOut	EM_ALUOut	EM_PCplus8

MEM/WB(Tnew)			
cal_r 0/rd	**cal_i 0/rt**	**load 0/rt**	**jal 0/31**
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8

思考题

1、

指令类型	前序指令	前前序指令	冲突寄存器	解决方法	测试序列
cal_r	cal_r / cal_i		rs / rt	转发	subu $s3, $s1, $s0 addu $s4, $s3, $s0

		cal_r / cal_i	rs / rt	转发	subu $s7, $s0, $s6
					addu $t0, $s6, $s7
					addu $t1, $t0, $s7

	load		rs / rt	阻塞	lw $t0, 0($0) addu $t1, $t0, $s0

		load	rs / rt	转发	lw $t0, 0($0)
					addu $t1, $t0, $s0
					addu $t2, $t0, $t1

	jal		31	转发	jal next1 addu $s2, $s0, $ra next1:

		jal	31	转发	jal next3
					addu $s6, $s5, $ra
					subu $s6, $s5, $ra

cal_i	cal_r / cal_i		rs	转发	ori $s1, $s0, 0x8954 ori $s2, $s1, 0x6543
		cal_r / cal_i	rs	转发	ori $s6, $s4, 0x8592 ori $s6, $s6, 0x3478 ori $s7, $s6, 0x8594
	load		rs	阻塞	lw $t1, 8($0) ori $t2, $t1, 0x4352
		load	rs	转发	lw $t1, 8($0) ori $t2, $t1, 0x4352 ori $t3, $t1, 0x5435
	jal		31	转发	jal next1 ori $s1, $ra, 0x8594 next1:
		jal	31	转发	jal next3 ori $ra, $ra, 0x5438 ori $s6, $ra, 0x8954

load	cal_r / cal_i		rs	转发	addu $t2, $t0, $t1 lw $t2, 4($0)

		cal_r / cal_i	rs	转发	subu $t3, $t2, $t1
					subu $t4, $t3, $t2
					lw $t3, 8($0)

	load		rs	阻塞	lw $t0, 0($0) lw $t1, 0($t0)

		load	rs	转发	lw $t2, 4($t0)
					lw $t2, 0($t2)
					lw $t3, 4($t2)

	jal		31	转发	jal next2 lw $t1, 0($ra)

		jal	31	转发	jal next2
					lw $t1, 0($ra)
					next2:
					lw $t2, 0($ra)

store	cal_r / cal_i		rs / rt	转发	addu $s3, $s0, $s1 sw $s3, 0($0)

		cal_r / cal_i	rs / rt	转发	addu $s3, $s0, $s1
					sw $s3, 0($0)
					sw $s3, 4($0)

	load		rs	阻塞	lw $t1, 0($0) sw $t0, 0($t1)
	load		rt	转发	lw $t0, 0($0) sw $t0, 16($0)

		load	rs / rt	转发	lw $t0, 0($0)
					sw $t0, 16($0)
					sw $t0, 20($0)

	jal		31	转发	jal next1 sw $s0, 0($ra)

		jal	31	转发	jal next2
					next2:
					sw $s0, 0($ra)
					sw $s0, 8($ra)

branch	cal_r / cal_i		rs / rt	阻塞	addu $s2, $s0, $s1 beq $s2, $0, next1
		cal_r / cal_i	rs / rt	转发	subu $s3, $s1, $s1 nop beq $s3, $0, next4
	load		rs / rt	阻塞	lw $s2, 0($0) beq $s2, $0, next1
		load	rs / rt	阻塞	lw $s3, 0($0) nop beq $s3, $0, next2
	jal		31	转发	
		jal	31	转发	jal next1 nop next1: beq $ra, $0, next2
jr	cal_r / cal_i		rs	阻塞	addu $t0, $s0, $0 jr $t0
		cal_r / cal_i	rs	转发	addu $t1, $s1, $0 nop jr $t1
	load		rs	阻塞	lw $s2, 0($0) jr $s2
		load	rs	阻塞	lw $s3, 4($0) nop jr $s3
	jal		31	转发	
		jal	31	转发	jal next1 nop next1: jr $s3