

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位为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	CMPOpResult	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

(7) MDU（乘除法模块）

模块端口说明如下：

序号	信号名	方向	描述
1	Clk	I	时钟信号
2	Reset	I	复位信号
3	Data1[31:0]	I	比较的数据1
4	Data2[31:0]	I	比较的数据2
5	MDOp[2:0]	I	乘除运算的选择信号 001：进行有符号乘法 010：进行无符号乘法 011：进行有符号除法 100：进行无符号除法 101：写入HI寄存器 110：写入LO寄存器
6	Busy	O	Busy为1表示正在进行乘除运算
7	HI	O	输出HI寄存器的值
8	LO	O	输出LO寄存器的值

模块功能定义如下：

序号	功能名称	功能描述
1	有符号乘法	$\{HI, LO\} \leftarrow Data1 * Data2,$ Data1、Data2为有符号值
2	无符号乘法	$\{HI, LO\} \leftarrow Data1 * Data2,$ Data1、Data2为无符号值
3	有符号除法	$\{HI, LO\} \leftarrow \{Data1 \% Data2, Data1 / Data2\},$ Data1、Data2为有符号值
4	无符号除法	$\{HI, LO\} \leftarrow \{Data1 \% Data2, Data1 / Data2\},$ Data1、Data2为无符号值
5	写HI寄存器	$HI \leftarrow Data1$
6	写LO寄存器	$LO \leftarrow Data1$

(8) DEXT（符号扩展单元）

模块端口说明如下：

序号	信号名	方向	描述
1	DM_Rdata[31:0]	I	DM的输出数据
2	Addr2[1:0]	I	DM读取地址的低2位
3	DEXTOp[2:0]	I	DEXT的功能选择信号 000：不进行扩展 001：进行半字符符号扩展 010：进行半字零扩展 011：进行字节符号扩展 100：进行字节零扩展
4	Dout[31:0]	O	数据扩展的结果

模块功能定义如下：

序号	功能	描述
1	数据扩展	根据DEXTOp的值进行相应的数据扩展

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]	0	扩展单元的功能选择信号
3	NPCSel[1:0]	0	NPC的功能选择信号
4	isBranch	0	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]	0	ALU功能选择信号
3	ALUSrc[1:0]	0	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	0	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	0	寄存器的写使能信号
3	RegDst[1:0]	0	寄存器写入地址选择信号
4	RegSrc[1:0]	0	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)							E	EX/MEM (Tnew)					
指令 类型	源寄 存器	Tuse	jal 0/31	jalr 0/rd	cal_r 1/rd	cal_i 1/rt	mf 1/rd	load 2/rt	md	MD U.bu	jal 0/31	jalr 0/31	cal_r 0/rd	cal_i 0/rt	mf 0/rd	load 1/rt
branch	rs / rt	0			暂停	暂停	暂停	暂停								暂停
jr	rs	0			暂停	暂停	暂停	暂停								暂停
jalr	rs	0			暂停	暂停	暂停	暂停								暂停
cal_r	rs / rt	1						暂停								
cal_i	rs	1						暂停								
md	rs / rt	1						暂停	暂停	暂停						
mt	rs	1						暂停	暂停	暂停						
mf									暂停	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	0	转发信号1，控制D阶段相等比较的输入
6	BypassDrt	0	转发信号2，控制D阶段相等比较的输入
7	BypassErs	0	转发信号3，控制ALU的输入
8	BypassErt	0	转发信号4，控制ALU的输入
9	BypassMrt	0	转发信号5，控制DM的写入值

转发条件如下：

		ID/EX			
控制信号	输入0	jal 0/31	jalr 0/rd		
BypassDrs	D_GPR_RData	DE_PCplus8	DE_PCplus8		
BypassDrt	D_GPR_RData	DE_PCplus8	DE_PCplus8		
BypassErs	DE_RData1				
BypassErt	DE_RData2				
BypassMrt	EM_M_Wdata				
EX/MEM (Tnew)					
jal 0/31	jalr 0/rd	cal_r 0/rd	cal_i 0/rt	mf 0/rd	
EM_PCplus8	EM_PCplus8	EM_ALUOut	EM_ALUOut	EM_HILO	
EM_PCplus8	EM_PCplus8	EM_ALUOut	EM_ALUOut	EM_HILO	
EM_PCplus8	EM_PCplus8	EM_ALUOut	EM_ALUOut	EM_HILO	
EM_PCplus8	EM_PCplus8	EM_ALUOut	EM_ALUOut	EM_HILO	
MEM/WB (Tnew)					
jal 0/31	jalr 0/rd	cal_r 0/rd	cal_i 0/rt	mf 0/rd	load 0/rt
MW_PCplus8	MW_PCplus8	MW_ALUOut	MW_ALUOut	MW_HILO	W_Dout
MW_PCplus8	MW_PCplus8	MW_ALUOut	MW_ALUOut	MW_HILO	W_Dout
MW_PCplus8	MW_PCplus8	MW_ALUOut	MW_ALUOut	MW_HILO	W_Dout
MW_PCplus8	MW_PCplus8	MW_ALUOut	MW_ALUOut	MW_HILO	W_Dout
MW_PCplus8	MW_PCplus8	MW_ALUOut	MW_ALUOut	MW_HILO	W_Dout

思考题

- 1、乘除法运算的运算时间比较长，如果把乘除法部件整合进 ALU，会大大延长 CPU 的运行周期。

乘除法的结果为 64 位，因而无法通过一次写入操作将结果写入通用寄存器，需要用 HI、LO 寄存器来保存结果。采用 HI、LO 寄存器也可以使得进行乘除法的同时可以执行其他无关的指令。

- 2、乘除法的运算有多个周期的延迟，在延迟过程中可以使与乘除法无关的指令继续执行。
- 3、把数据扩展模块放在 MEM/WB 之后，可以减少 M 阶段的组合逻辑，从而减少时钟周期。
- 4、当需要以字节为单位进行操作时，按字节访问内存可以减少额外的操作，从而相对与按字访问更有优势。
- 5、大致为规划者型。

抽象和规范手段：对指令进行分类；规范化命名