基于 Verilog 语言的 MIPS 微系统设计文档

17373051 郭骏

一、顶层设计

顶层设计采用之前 CPU 的设计思路，在顶层模块 mips.v 下配置两个文件 cpupath.v、bridge.v，将 CPU 和外设放在不同的.v 文件中。

cpupath.v 下配置功能原件模块，如 PC、ADD4、IM、RF、EXT、CMP、NPC、ALU、SFU、MDU、BE、DM、LET、CP0 以及一些多路选择器 MUX。这些模块在 DATAPATH 模块中进行连线综合，最终汇集到顶层模块 mips，实现 CPU 的总体功能。

二、数据通路设计

数据通路设计的思路来源于理论课流水线 CPU 构造方法的 PPT，由于数据通路表格实在过于庞大，故不在此处列出，附在同时上交的表格中。在这里，只写上每个模块的端口说明和功能定义。

表 1 数据通路模块端口说明

流水级	模块名	信号名称	方向	功能描述
F	PC	clk	I	时钟信号
		reset	I	同步复位信号
		stall	I	流水线暂停信号
		cpc[31:0]	I	后级流水线返回的下条指令地址
		npc[31:0]	O	即将进入 IM 的指令地址
	ADD4	PCout	I	来自 PC 模块的输入
		ADD4out	O	PC+4 的结果
	IM	addr[31:0]	I	来自 PC 模块的指令地址信号
		ir[31:0]	O	指令地址对应的指令数据
D&W	RF	clk	I	时钟信号
		reset	I	同步复位信号
		RegWrite	I	写使能信号 0：不能写入 1：可以写入
		a1[4:0]	I	寄存器组待读出地址 1
		a2[4:0]	I	寄存器组待读出地址 2
		a3[4:0]	I	寄存器组待写入地址
		wd[31:0]	I	寄存器组待写入数据
		rd1[31:0]	O	寄存器组读出数据 1
		rd2[31:0]	O	寄存器组读出数据 2
D	EXT	in[15:0]	I	待扩展 16 位数据
		ExtOp[1:0]	I	位扩展器功能选择信号 0：零扩展 1：符号扩展 2：高位扩展
		out[31:0]	O	扩展后的 32 位数据
	CMP	d1[31:0]	I	待比较数据 1
		d2[31:0]	I	待比较数据 2
		CMPOp[2:0]	I	比较器功能选择信号 0：d1==d2 1：d1!=d2 2：d1>=0 3：d1>0 4：d1<=0 5：d1<0
		CMPout	O	是否满足比较条件的输出信号 0：不满足 1：满足
	NPC	PC4[31:0]	I	PC+4 的地址
		i26[25:0]	I	跳转指令中的立即数（16 位和 26 位共用端口）
		NPCOp	I	NextPC 计算选择信号 0：branch 指令 1:j 或者 jal 指令
		CMPout	I	来自 CMP 的判断信号
		NPCout[31:0]	O	下一条地址的计算结果
E	ALU	A[31:0]	I	ALU 待计算数 1
		B[31:0]	I	ALU 待计算数 2
		ALUOp[3:0]	I	ALU 功能选择信号 0：符号加 1：无符号加 2：符号减 3：无符号减 4：按位与 5：按位或 6：按位异或 7：按位或非 8：符号小于置 1 9：无符号小于置 1
		result[31:0]	O	ALU 计算结果

	SFU	SD[31:0]	I	待移位数据
		NUM[4:0]	I	待移动的位数
		SFUOp[1:0]	O	移位器功能选择信号 0：逻辑左移 1：逻辑右移 2：算术右移
		SFUout[31:0]	O	移位之后的数据
	MDU	clk	I	时钟信号
		reset	I	同步复位信号
		start	I	乘除法运算开始信号
		A[31:0]	I	乘除模块待计算数 1
		B[31:0]	I	乘除模块待计算数 2
		WD[31:0]	I	乘除模块待写入数据
		MDUOp[2:0]	I	乘除模块功能选择信号 0：什么都不做 1：符号乘法 2：无符号乘法 3：符号除法 4：无符号除法 5：写 HI 寄存器 6：写 LO 寄存器
		HI[31:0]	O	乘除模块 HI 寄存器的值
		LO[31:0]	O	乘除模块 LO 寄存器的值
		Busy	O	乘除模块是否正在计算中的信号 0：空闲 1：正在计算
M	BE	A1_0[1:0]	I	数据存储器地址最后 2 位
		BEOp[1:0]	I	指令要求写入或输出方式的信号 0：按字 1：按半字 2：按字节
		BEout[3:0]	O	字节使能信号，即控制一个字内哪些字节被写入或输出的信号
	DM	clk	I	时钟信号
		reset	I	同步复位信号
		MemWrite	I	数据存储器写使能信号 0：不可写入 1：可写入
		addr[31:0]	I	数据存储器读/写地址
		BE[3:0]	I	字节使能信号，即控制一个字内哪些字节被写入的信号
		din[31:0]	I	数据存储器写入数据
		dout[31:0]	O	数据存储器读出数据
W	LET	A1_0[1:0]	I	数据存储器地址最后两位
		din[31:0]	I	数据存储器读出的数据
		LETOp[2:0]	I	位扩展方式选择信号 0：无扩展 1：无符号半字扩展 2：符号半字扩展 3：无符号字节扩展 4：符号字节扩展
		dout[31:0]	O	扩展后的数据

表 2 数据通路模块功能定义

流水级	模块名	功能序号	功能名称	功能描述
F	PC	1	同步复位	当 clk 上升沿到来且 reset 信号有效时，PC 被设置为 0x00003000
		2	取指令地址	当 clk 信号上升沿到来时，输出下一条指令地址
	ADD4	1	计算	给定 PC，计算 PC+4 的值并输出
	IM	1	取指令	输入 PC 的地址，输出对应位置指令的代码
D&W	RF	1	同步复位	当 clk 上升沿到来且 reset 信号有效时，所有寄存器值清零
		2	读数据	读出 a1、a2 地址对应寄存器中的数据到 rd1、rd2
		3	写数据	当 clk 上升沿到来且 RegWrite 信号有效时，将 wd 写入 a3 对应的寄存器中
D	EXT	1	零扩展	在 16 位数据前添加 16 个 0 以将其扩展为 32 位数据
		2	符号扩展	在 16 位数据前添加 16 个 0 或 1（与原数据的第 16 位相同）以将其扩展为 32 位数据
		3	高位扩展	在 16 位数据末尾添加 16 个 0 以将其扩展为 32 位数据
	CMP	1	比较	判断 d1 和 d2 是否满足特定关系，或者 d1 和 0 之间是否满足特定关系
	NPC	1	计算下条指令	当需要跳转时，根据跳转指令类型计算下一条指令的地址
E	ALU	1	两数运算	根据 ALUOp 的要求，对 A 和 B 进行加减或者逻辑运算并将结果存到 result 输出
		2	两数比较	根据 ALUOp 的要求，比较 A 和 B，并将比较结果（0 或 1）存到 result 输出
	SFU	1	移位	根据 SFUOp 的要求，对 SD 进行逻辑左移、逻辑右移或者算术右移 NUM 位，并将结果输出
	MDU	1	同步复位	当 clk 上升沿到来且 reset 信号有效时，HI 和 LO 寄存器值清零并终止正在进行的运算
		2	乘法运算	运算 A×B（有或无符号），并将高 32 位存在 HI 寄存器，低 32 位存在 LO 寄存器
		3	除法运算	运算 A÷B（有或无符号），并将余数存在 HI 寄存器，商存在 LO 寄存器
		4	写 HI 或 LO	在 clk 上升沿到来且 MDUOp 允许时，将指定数据写入 HI 或者 LO 寄存器
		5	输出忙碌信号	当乘除模块正在进行乘除法运算时，输出 Busy 信号并暂停其他乘除指令
M	BE	1	译码	根据 A1_0 和 BEOp 的要求，判断应当写入哪些字节并输出
	DM	1	同步复位	当 clk 上升沿到来且 reset 信号有效时，将存储器所有数据置 0
		2	读数据	读出 addr 地址对应存储器数据到 dout
		3	写数据	当 clk 上升沿到来且 MemWrite 信号有效时，将 din 写入 addr 和 BE 对应的存储器中
W	LET	1	位扩展	按 LETOp 的要求，对读入数据进行字节或者半字的有无符号扩展

三、控制器设计

控制器单元采用 assign 信号构造，具体真值表如下表所示。在处理时，所有 X 信号一律当做 0 信号处理。

表 3 控制器指令和控制信号对应真值表

流水级	指令分类	ld 类					str 类			j 类	jal 类
	控制信号	lb	lbu	lh	lhu	lw	sb	sh	sw	j	jal
	Funct[5:0]										
	Opcode[6:0]	100000	100100	100001	100101	100011	101000	101001	101011	000010	000011
D	ExtOp[1:0]	1	1	1	1	1	1	1	1	X	X
	CMPOp[2:0]	X	X	X	X	X	X	X	X	X	X
	NPCOp	X	X	X	X	X	X	X	X	1	1
	MPCsel[2:0]	0	0	0	0	0	0	0	0	1	1
E	ALUOp[3:0]	1	1	1	1	1	1	1	1	X	X
	SFUOp[1:0]	X	X	X	X	X	X	X	X	X	X
	MDUOp[2:0]	0	0	0	0	0	X	X	X	0	0
	MNUMsel	X	X	X	X	X	X	X	X	X	X
	MALUBsel	1	1	1	1	1	1	1	1	X	X
M	BEOp[1:0]	2	2	1	1	0	2	1	0	X	X
	MemWrite	0	0	0	0	0	1	1	1	0	0
	MDRWsel	0	0	0	0	0	0	0	0	X	X
W	LETOp[2:0]	4	3	2	1	0	X	X	X	X	X
	RegWrite	1	1	1	1	1	0	0	0	0	1
	MRFA3sel[1:0]	0	0	0	0	0	X	X	X	X	2
	MRFWDsel[2:0]	0	0	0	0	0	X	X	X	X	3
流水级	指令分类	calr 类									
	控制信号	add	addu	sub	subu	and	or	xor	nor	slt	sltu
	Funct[5:0]	100000	100001	100010	100011	100100	100101	100110	100111	101010	101011
	Opcode[6:0]	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
D	ExtOp[1:0]	X	X	X	X	X	X	X	X	X	X
	CMPOp[2:0]	X	X	X	X	X	X	X	X	X	X
	NPCOp	X	X	X	X	X	X	X	X	X	X
	MPCsel[2:0]	0	0	0	0	0	0	0	0	0	0
E	ALUOp[3:0]	0	1	2	3	4	5	6	7	8	9
	SFUOp[1:0]	X	X	X	X	X	X	X	X	X	X
	MDUOp[2:0]	0	0	0	0	0	0	0	0	0	0
	MNUMsel	X	X	X	X	X	X	X	X	X	X

	MALUBsel	0	0	0	0	0	0	0	0	0	0
M	BEOp[1:0]	X	X	X	X	X	X	X	X	X	X
	MemWrite	0	0	0	0	0	0	0	0	0	0
	MDRWsel	X	X	X	X	X	X	X	X	X	X
W	LETOp[2:0]	X	X	X	X	X	X	X	X	X	X
	RegWrite	1	1	1	1	1	1	1	1	1	1
	MRFA3sel[1:0]	1	1	1	1	1	1	1	1	1	1
	MRFWDsel[2:0]	1	1	1	1	1	1	1	1	1	1
流水级	指令分类	md 类				sft 类			sftv 类		
	控制信号	mult	multu	div	divu	sll	srl	sra	sllv	srlv	srav
	Funct[5:0]	011000	011001	011010	011011	000000	000010	000011	000100	000110	000111
	Opcode[6:0]	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
D	ExtOp[1:0]	X	X	X	X	X	X	X	X	X	X
	CMPOp[2:0]	X	X	X	X	X	X	X	X	X	X
	NPCOp	X	X	X	X	X	X	X	X	X	X
	MPCsel[2:0]	0	0	0	0	0	0	0	0	0	0
E	ALUOp[3:0]	X	X	X	X	X	X	X	X	X	X
	SFUOp[1:0]	X	X	X	X	0	1	2	0	1	2
	MDUOp[2:0]	1	2	3	4	X	X	X	X	X	X
	MNUMsel	X	X	X	X	1	1	1	0	0	0
	MALUBsel	X	X	X	X	X	X	X	X	X	X
M	BEOp[1:0]	X	X	X	X	X	X	X	X	X	X
	MemWrite	0	0	0	0	0	0	0	0	0	0
	MDRWsel	X	X	X	X	X	X	X	X	X	X
W	LETOp[2:0]	X	X	X	X	X	X	X	X	X	X
	RegWrite	0	0	0	0	1	1	1	1	1	1
	MRFA3sel[1:0]	X	X	X	X	1	1	1	1	1	1
	MRFWDsel[2:0]	X	X	X	X	2	2	2	2	2	2
流水级	指令分类	cali 类							lui 类	mfhi 类	mflo 类
	控制信号	addi	addiu	andi	ori	xori	slti	sltiu	lui	mfhi	mflo
	Funct[5:0]									010000	010010
	Opcode[6:0]	001000	001001	001100	001101	001110	001010	001011	001111	000000	000000
D	ExtOp[1:0]	1	1	0	0	0	1	1	2	X	X
	CMPOp[2:0]	X	X	X	X	X	X	X	X	X	X
	NPCOp	X	X	X	X	X	X	X	X	X	X
	MPCsel[2:0]	0	0	0	0	0	0	0	0	0	0
E	ALUOp[3:0]	0	1	4	5	6	8	9	1	X	X

	SFUOp[1:0]	X	X	X	X	X	X	X	X	X	X
	MDUOp[2:0]	X	X	X	X	X	X	X	X	0	0
	MNUMsel	X	X	X	X	X	X	X	X	X	X
	MALUBsel	1	1	1	1	1	1	1	1	X	X
M	BEOp[1:0]	X	X	X	X	X	X	X	X	X	X
	MemWrite	0	0	0	0	0	0	0	0	0	0
	MDRWsel	X	X	X	X	X	X	X	X	X	X
W	LETOp[2:0]	X	X	X	X	X	X	X	X	X	X
	RegWrite	0	0	0	0	0	0	0	1	1	1
	MRFA3sel[1:0]	0	0	0	0	0	0	0	0	1	1
	MRFWDsel[2:0]	1	1	1	1	1	1	1	1	4	5
流水级	指令分类	mt 类		br 类		brz 类				jalr 类	jr 类
	控制信号	mthi	mtlo	beq	bne	bgez	bgtz	blez	bltz	jalr	jr
	Funct[5:0]	010001	010011			rt00001			rt00000	001001	001000
	Opcode[6:0]	000000	000000	000100	000101	000001	000111	000110	000001	000000	000000
D	ExtOp[1:0]	X	X	1	1	1	1	1	1	X	X
	CMPOp[2:0]	X	X	0	1	2	3	4	5	X	X
	NPCOp	X	X	0	0	0	0	0	0	X	X
	MPCsel[2:0]	0	0	1	1	1	1	1	1	2	2
E	ALUOp[3:0]	X	X	X	X	X	X	X	X	X	X
	SFUOp[1:0]	X	X	X	X	X	X	X	X	X	X
	MDUOp[2:0]	5	6	0	0	0	0	0	0	0	0
	MNUMsel	X	X	X	X	X	X	X	X	X	X
	MALUBsel	X	X	X	X	X	X	X	X	X	X
M	BEOp[1:0]	X	X	X	X	X	X	X	X	X	X
	MemWrite	0	0	0	0	0	0	0	0	0	0
	MDRWsel	X	X	X	X	X	X	X	X	X	X
W	LETOp[2:0]	X	X	X	X	X	X	X	X	X	X
	RegWrite	0	0	0	0	0	0	0	0	1	0
	MRFA3sel[1:0]	X	X	X	X	X	X	X	X	1	X
	MRFWDsel[2:0]	X	X	X	X	X	X	X	X	3	X

流水级	指令分类	eret	mfc0	mtc0
	控制信号	eret	mfc0	mtc0
	Funct[5:0]	011000	rs00000	rs00100
	Opcode[6:0]	010000	010000	010000
D	ExtOp[1:0]	X	X	X
	CMPOp[2:0]	X	X	X

	NPCOp	X	X	X
	MPCsel[2:0]	3	0	0
E	ALUOp[3:0]	X	X	X
	SFUOp[1:0]	X	X	X
	MDUOp[2:0]	X	X	X
	MNUMsel	X	X	X
	MALUBsel	X	X	X
M	BEOp[1:0]	X	X	X
	MemWrite	0	0	0
	CP0Write	0	0	1
	MDRWsel	X	1	1
W	LETOp[2:0]	X	X	X
	RegWrite	0	1	0
	MRFA3sel[1:0]	X	0	X
	MRFWDsel[2:0]	X	0	X

四、数据冒险处理

数据冒险处理采用暂停和转发两种机制，为了增加流水线的效率，能够进行转发的数据一律进行转发，在数据还没有生成的时候才考虑使用暂停。阻塞矩阵和转发控制器采用讨论区中所提供的方法，加上一些个人的理解。

首先需要为每个类型的指令确定他们的“寄存器写入地址”（A3）和寄存器写入数据来源（Res）。Res 为 1 代表来源于 LET，2 代表来源于 ALO，3 代表来源于 SFU，4 代表来源于 HI，5 代表来源于 LO，6 代表来源于 PC+8。A3 和 Res 表如下所示。

表 5 指令的 A3 和 Res 表

	ld 类	str 类	calr 类	md 类	sft 类	sftv 类	cali 类	lui 类	mfhi 类
A3[4:0]	IR\W[rt]	0	IR\W[rd]	0	IR\W[rd]	IR\W[rd]	IR\W[rt]	IR\W[rt]	IR\W[rd]
Res[2:0]	1	X	2	X	3	3	2	2	4
	mflo 类	mt 类	br 类	brz 类	j 类	jal 类	jalr 类	jr 类	
A3[4:0]	IR\W[rd]	0	0	0	0	31	IR\W[rd]	0	
Res[2:0]	5	X	X	X	X	6	6	X	

表 5 阻塞矩阵

	ID/EX						EX/MEM						MEM/WB					
Res	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
Tuse	2	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
0	S	S	S	S	S	F	S	F	F	F	F	F	F	F	F	F	F	F
1	S	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

表 6 转发用多路选择器信号表

转发 MUX	0	1	2	3	4	5	6	7
MFRSD	RF.RD1	PC4\E	PC4\M	AO\M	SFO\M	HI\M	LO\M	MRFWD
MFRTD	RF.RD2	PC4\E	PC4\M	AO\M	SFO\M	HI\M	LO\M	MRFWD
MFRSE	V1\E	PC4\M	AO\M	SFO\M	HI\M	LO\M	MRFWD	
MFRTE	V2\E	PC4\M	AO\M	SFO\M	HI\M	LO\M	MRFWD	
MFRTM	V2\M	MRFWD						

具体的转发设计由于表格过长，无法在此列举，请参考附件中的 Excel 表格。

五、协处理器设计

表 5 CP0 端口定义

模块名	信号名称	方向	功能描述
CP0	clk	I	时钟信号
	reset	I	同步复位信号
	addr[4:0]	I	读写 CP0 的地址信号
	din[31:0]	I	写入 CP0 的数据
	PC[31:0]	I	当前指令的 PC 值
	ExcCode[6:2]	I	异常原因
	HWInt[7:2]	I	中断控制信号
	WE	I	写 CP0 使能 0：不可写入 1：可写入
	EXLSet	I	EXL 位置 1 信号
	EXLClr	I	EXL 位置 0 信号
	BD	I	BD 位判断信号
	IntReq	O	中断请求信号
	EPC[31:0]	O	受害指令的 PC 值，与 CP0 的 EPC 寄存器相连
	dout	O	从 CP0 中读取的值

六、测试程序

50 条指令带来的测试工程量是巨大的，难以编写一个遍历所有指令和冲突情况的程序，只能尽可能的模拟所有情况。测试程序、期望全部附在附件中。

思考题

1.我们计组课程一本参考书目标题中有“硬件/软件接口”接口字样，那么到底什么是“硬件/软件接口”？

接口，指的是不同部件之间的交接处。硬件接口指的是计算机不同功能层之间的连接规则，比如时钟信号和 CPU 中各部件之间的连接。而软件接口指的是对不同硬件的使用方式的分配，比如汇编代码和流水线 CPU 之间的沟通作用。

2.在我们设计的流水线中，DM 处于 CPU 内部，请你考虑现代计算机中它的位置应该在何处。

现代计算机中，DM 的位置应当位于 CPU 外部，之间可以由高速缓存处理器连接以加快读

取速度，也使得主存可以与外部存储并列。

3.BE 部件对所有的外设都是必要的吗？

并不是，对于某些不需要按字节写入或者读取的外设来说，BE 部件没有必要，如本次设计的 COCO 定时器。

4.请开发一个主程序以及定时器的 exception handler。整个系统完成如下功能：

1)定时器在主程序中被初始化为模式 0；

2)定时器倒计数至 0 产生中断；

3)handler 设置使能 Enable 为 1 从而再次启动定时器的计数器。2 及 3 被无限重复。

4)主程序在初始化时将定时器初始化为模式 0，设定初值寄存器的初值为某个值，如 100 或 1000。

```
ori $t1,$0,9
ori $t2,0x7f00
ori $t3,100
sw $t1,0($t2)
sw $t3,4($t2)
target:
j target
nop

.ktext 0x4180
sw $t1,0($t2)
eret
```

5.请查阅相关资料，说明鼠标和键盘的输入信号是如何被 CPU 知晓的？

当键盘、鼠标有信息时，会产生一个中断信号，根据不同的端口信号，对 CPU 的寄存器进行不同的写入处理，从而使得 CPU 知晓键盘和鼠标的信息。