CPU设计文档

一、CPU设计方案综述

（一）总体设计概述

本CPU为verilog实现的支持异常中断的流水线MIPS - CPU，支持的指令集为MIPS-C3={LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、 SUB、 SUBU、 MULT、 MULTU、 DIV、 DIVU、 SLL、 SRL、 SRA、 SLLV、 SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、 BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO}。为了实现这些功能，CPU主要包含了IM、GRF、IFU、ALU、Controller、DM、mulanddiv、IF/ID、ID/EX、EX/MEM、MEM/RB、CP0、TC1、TC2、Bridge模块。

（二）运算模块定义

1. GRF

GRF为寄存器堆，有32个寄存器，具有同时读取两个寄存器、写寄存器、复位功能。

表1 GRF端口及功能

序号	信号名	方向	功能名称	功能描述
1	A1[4:0]	I	地址输入信号1	选择一个寄存器将其值输出到RD1
2	A2[4:0]	I	地址输入信号2	选择一个寄存器将其值输出到RD2
3	A3[4:0]	I	地址输入信号3	选择一个寄存器作为WD信号的写入目标
4	WD[31:0]	I	数据写入信号	当WE有效时，内容写入A3指定的寄存器
5	CLK	I	时钟信号	时钟信号，上升沿更新寄存器的值
6	reset	I	复位信号	有效时将所有寄存器异步复位到0
7	WE	I	写使能信号	有效时将WD信号写入A3指定的寄存器
8	RD1	O	数据输出信号	输出A1指定的寄存器中的数据
9	RD2	O	数据输出信号	输出A2指定的寄存器中的数据

2. IFU

IFU为取指令单元，内部包含PC（程序计数器）与IM（指令存储器）。指令存储在IFU中的ROM（IM）中，起始地址为0x00000000，由PC决定读取的指令位置。

表2 IFU端口及功能

序号	信号名	方向	功能名称	功能描述
1	PC_brench[31:0]	I	分支跳转信号	为brench类分支判断指令跳转后的PC值
2	PC_jump[31:0]	I	直接跳转信号	为j型指令跳转后的PC值
3	PC_jr[31:0]	I	跳寄存器	jr指令后的PC值
4	ctrl_src	I	PC写入判断信号1	有效时，将PC_brench置为新的PC
5	ctrl_jr	I	PC写入判断信号2	有效时，将PC_jr置为新的PC
6	ctrl_j	I	PC写入判断信号3	有效时，将PC_jump置为新的PC
7	reset	I	复位信号	有效时，PC异步复位到0
8	CLK	I	时钟信号	时钟信号，上升沿更新PC的值
9	Instruction[31:0]	O	输出指令信号	当前PC指向的在IM中的指令
10	PCplus4[31:0]	O	下一个PC	下一条指令的PC值
11	PC[31:0]	O	当前PC	当前指令的PC值，便于j指令使用

3.ALU

ALU为算术逻辑单元，可以根据指令ALUctrl执行相应的计算功能。

表3 ALU端口及功能

序号	信号名	方向	功能名称	功能描述
1	A[31:0]	I	输入信号1	输入参与计算的第一个信号A
2	B[31:0]	I	输入信号2	输入参与计算的第二个信号B
3	ALUctrl[2;0]	I	计算信号	由这个信号决定如何进行计算
4	Y[31:0]	O	输出信号	计算结果
5	zero	O	是否为0判断信号	计算结果为0时置1

表4 ALUctrl对应ALU计算方式

序号	ALUctrl[2:0]	输出Y值	ALUzero
1	000	A AND B	Y==0
2	001	A OR B	Y==0
3	010	A+B	Y==0
4	011	B<<16	Y==0
5	100	A=B?1:0	Y==0
6	101	A OR ~B	A>0
7	110	A-B	Y==0
8	111	SLT(A<B?1:0)	Y==0

4.DM

DM为数据存储器，使用RAM实现，作为内存的模拟。起始位置为0x00000000，具有异步复位功能，复位值为0x00000000。

表5 DM端口及功能

序号	信号名	方向	功能名称	功能描述
1	ALUResult_Address[31:0]	I	地址信号	读写内存的位置
2	WriteData[31:0]	I	写入信号	写入内存的值
3	CLK	I	时钟信号	时钟信号，下降沿更新RAM中的值
4	reset	I	复位信号	有效时，RAM异步复位为0x00000000
5	WE	I	写使能信号	有效时，将WriteDara写入ALUResult_Address指定的位置
6	MemOut	O	输出内存信号	WE信号无效时，输出ALUResult_Address指定的RAM位置的值

注：因为MIPS指令不允许同时读写内存，故在DM中将读使能信号置为WE的反，写使能不生效时允许读取。

5.IM

IM为指令存储器，读取IFU的PC，输出当前位置地址

表6 IM端口及功能

序号	信号名	方向	功能名称	功能描述
1	Address[9:0]	I	当前读取指令位置	由此地址决定读取指令
2	Instruction[31:0]	O	指令输出	输出当前指令

（三）各级流水段

1.IF流水段

IM模块和IFU模块内置在IF流水段中。

IF流水段将把PC输入IFU并实现指令的读出，随后将指令写入IF/ID流水线寄存器。同时将接收ID段传递回来的PC各类分支值以及控制信号计算出新的PC值。

流水输出信号：PC_IFout、Ins_IFout

2.ID流水段

流水输入信号：PC_IDin、Ins_IDin、RegWrite_WB、RegWriteData_WB、PC_WB

冒险处理输入信号：transDaddr,transEaddr,transWaddr,transDdata,transEdata,transWdata,transDTnew,transETnew,transWTnew

流水输出信号：单周期后续需要信号、Tnew_IDout、datatrans_IDout、ARegWrite_IDout、Ruse1_IDout、Ruse2_IDout

根据rd、rs、rt、imm以及控制模块读GRF并获得结果。

如果RegWrite为0，则将Tnew置0，ARegWrite_IDout置0，即视为写0寄存器。

WB阶段GRF写使能信号始终有效，因为如果不写GRF，在该指令的ID段时已经将写目标置0，可以减少流水信号数。

无论任何情况，PC+8将存入datatrans。

注：采用集中译码，控制信号在此处产生并流水。

3.EX流水段

流水输入信号：单周期所需信号、Ruse1_EXin、Ruse2_EXin、Tnew_EXin、datatrans_EXin、ARegWrite_EXin

冒险处理输入信号：transEaddr,transWaddr,transEdata,transWdata

流水输出信号：单周期所需信号、Ruse1_EXout、Ruse2_EXout、Tnew_EXout、datatrans_EXout、ARegWrite_EXout

实现ALU运算并存入EX/MEM寄存器。

当Tnew=1时，将ALUresult写入datatrans。

注：branch指令在ID得到判断，不需要ALU。MEM段仅可能第二行转发，不流水Ruse1。

4.MEM流水段

流水输入信号：单周期所需信号、Ruse2_MEMin、Tnew_MEMin、datatrans_MEMin、ARegWrite_MEMin

冒险处理输入信号：transWaddr,transWdata

流水输出信号：单周期所需信号、Tnew_MEMout、datatrans_MEMout、ARegWrite_MEMout

操作MEM，存入MEM/WB。

当Tnew=1时，将Memout写入datatrans。

5.WB流水段

根据ARegWrite写入GRF。

（四）冒险处理方法

每个模块之间都需要传递Tnew、ARegWrite(即A)、Ruse(仍可能需要的.)

1.AT信号产生

AT信号都在ID译码的同时产生。

当某指令Tuse少于2个时，将该Tuse对应的Ruse置为0（即忽略），并且将Tuse置为7（最大值）。

Tuse1指上路使用，即RD1（jr、beq）、RD1_EXin（ALU_A)（R、ori、lw、sw）

Tuse2指下路使用，即RD2（beq）、RD2_EXout(ALU和imm选择一个)（R）、RD2_MEMin(写入内存的数据)（sw）

可以看出，Tuse1仅与rs段对应寄存器的读出信息相关；Tuse2仅与rt段对应寄存器的读出信息相关，故转发时判断可以仅连接这几个位置。

当某条指令不写寄存器时，即RegWrite为0，此时将Tnew置0，且A置0。

注：jal在ID级即可产生可靠的datatrans，故另设一信号负责进行转发，以此保证jal的全力转发。

表7 不同指令的Tnew和Tuse与对应A

指令	A	Tnew	Tuse个数	Tuse1	Tuse2
R	rd	2	2	1	1
ori	rt	2	1	1	7
lw	rt	3	1	1	7
sw	0	0	2	1	2
beq	0	0	2	0	0
lui	rt	2	0	7	7
j	0	0	0	7	7
jal	31	1	0	7	7
jr	0	0	1	0	7

2.转发方式

在流水线寄存器IDtoEX，EXtoMEM，MEMtoWB设置向前转发，即向前面的模块转发ARegWrite、transdata。

转发的接收端有ID级的RD1、RD2，EX的ALU_A、RD2_EXin、MEM的RD2_MEMin，在这些端口均进行比较，如果Tuse!=0且Tuse==ARegWrite，则进行转发，其中靠前的模块优先。

在转发时不需要检测Tnew以判断转发端是否准备好，因为目前没有进行阻塞，则Tnew小于等于Tuse，转发端一定会在转发数据被使用之前准备好，在每一处使用时均会再进行转发，故会将可能以前错误转发的数据覆盖。使用的数据一定是已经准备好的数据。

3.阻塞方式

阻塞的判断均在ID进行，Tuse不进行流水。

当发现Ruse!=0&&Ruse==ARegWrite&&Tuse<Tnew时或usehilo和hilobusy同时为1时stop置1标志进行阻塞。阻塞信号生效时，PC保持不变，IFtoID寄存器不更新，IDtoEX清零（即等价插入一条nop）。

（五）指令分类和控制信号

1、指令分类

类型	指令	共性
load	lb, lbu, lh, lhu, lw	读内存
save	sb, sh, sw	写内存
branch	beq, bne, blez, bgtz, bltz, bgez	分支跳转
R	add, addu, sub, subu, sllv, srlv, srav, and, or, nor, xor, slt, sltu, sll, srl, sra	写寄存器
i	addi, addiu, andi, ori, xori, lui, slti, sltiu	立即数
md	mult, multu, div, divu	乘除法
mt	mtlo, mthi	直接写hilo
mf	mflo, mfhi	读取hilo
jump	jalr, jal, jr, j	直接跳转

2、Branch指令和compareCtrl

指令	compareCtrl	跳转条件
beq	1	相等
bne	2	不等
blez	3	RD1小于等于0
bgtz	4	RD1大于0
bltz	5	RD1小于0
bgez	6	RD1大于等于0

3、R/i指令和ALU控制信号

指令	ALUCtrl	ALU行为	imm扩展
add/addi	0	A+B	sign
addu/addiu	1	A+B	sign
sub	2	A-B	
subu	3	A-B	
sllv	4	B<<A[4:0]	
srlv	5	B>>A[4:0]	
srav	6	B>>>A[4:0]	
and/andi	7	A&B	zero
or/ori	8	A|B	zero
xor/xori	9	A^B	zero
slt/slti	10	A<B?1:0（有符号）	sign
sltu/sltiu	11	A<B?1:0（无符号）	sign
sll	12	B<<S	
srl	13	B>>S	
sra	14	B>>>S	
lui	15	B<<16	
nor	24	~(A|B)	

4、存取指令和MemWrite信号

MemWrite = (sb)?1 :(sh)?2 :(sw)?3 :(lb)?4 :(lbu)?5 :(lh)?6 :(lhu)?7 :0;

指令	MemWrite	操作
sb	1	写RD2_Memin[7:0]
sh	2	写RD2_Memin[15:0]
sw	3	写RD2_Memin
lb	4	读相应的8位并符号扩展
lbu	5	读相应的8位并0扩展
lh	6	读相应的16位并符号扩展
lhu	7	读相应的16位并0扩展
lw	0	读全部
mfc0	8	读c0
mtc0	9	写c0

5、乘除槽相关指令和ALUctrl

指令	ALUctrl	操作
mult	16	有符号乘
multu	17	无符号乘
div	18	有符号除
divu	19	无符号除
mtlo	20	直接写lo
mthi	21	直接写hi
mflo	22	读lo
mfhi	23	读hi

注意：24在nor，第一个未使用编号是25

p7设计文档

一、微系统设计

1. CP0设计

CP0处理器放在流水线的M级内部，负责接收外部中断和内部异常并进行处理。

表1 CP0端口定义

序号	端口名	方向	功能
1	A1[4:0]	I	读寄存器地址
2	A2[4:0]	I	写寄存器地址
3	DIn[31:0]	I	写寄存器内容
4	PC[31:0]	I	当前PC（宏观PC）
5	ExcCode[6:2]	I	中断/异常类型
6	HWInt[5:0]	I	六个设备中断情况
7	WE	I	CP0写使能
8	EXLClr	I	清空SR的EXL位
9	clk	I	时钟
10	reset	I	复位
11	Req	O	出现中断/异常
12	EPCout[31:0]	O	EPC信号
13	Dout[31:0]	O	读寄存器内容
14	delay	I	是否为延迟槽指令

CP0中的HWInt会在每个周期写入Cause[15:11]（无论中断与否）

EPC寄存器会在Req生效时根据delay判断写入PC或PC-4

eret会流水激活EXLClr清空EXL位

在中断异常发生时将byteen信号改为1111并将写地址改为0x7f20来响应中断（此时也不会错误写入信息因为为非法地址）

2. CP0相关指令设计

1、mtc0

mtc0为写相应CP0寄存器，由于不存在对CP0的寄存器直接计算的指令，故不需要向后转发，只需要接受前方对rt寄存器值的转发，Tuse和save指令一致。由于采用集中式译码，借用以前用来转发jal指令的PC+8的datatrans通路来携带rd的值（写入目标）。并复用MemWrite=9

2、mfc0

mfc0不需要接受转发而需要向后转发，由于CP0位置在M级，故Tnew和load类型指令一致。复用MemWrite=8。

3、eret

eret信号在ID级产生效果，非阻塞状态下清空延迟槽并将EPC输入PC选择器，同时信号向后流水在M级清空EXL。复用MemWrite=11。

3. Bridge与IO设计

由于发现counter的读写只使用访存地址信号的后3位，故直接将访存地址接在DM、TC1、TC2上，仅使用Bridge生成写使能信号。写使能信号成立条件是byteen信号不全为0且写地址在范围内。

读出时将DM、TC1、TC2信号都接入Bridge并通过地址进行选择，如果均不在地址范围内则返回0。

表2 Bridge端口

序号	端口名	方向	功能
1	PrAddr[31:0]	I	CPU发出访存地址
2	PrRD[31:0]	O	CPU返回读内容
3	PrWD[31:0]	I	CPU发出写内容
4	byteen[3:0]	I	CPU发出写使能信号
5	Addr[31:0]	O	向DM和TC发出访存地址
6	byteen_DM[3:0]	O	DM写使能信号
7	WD[31:0]	O	向DM和TC发出写内容
8	databack_DM[31:0]	I	DM返回Bridge信号
9	WE_C1	O	TC1写使能
10	data_C1[31:0]	I	读TC1返回数据
11	WE_C2	O	TC2写使能
12	data_C2[31:0]	I	读TC2返回数据
13	IRQ_C1	I	TC1返回中断信号
14	IRQ_C2	I	TC2返回中断信号
15	HWInt[5:0]	O	总中断信号（直接输入CP0）
16	interrupt	I	外部中断信号

p8设计文档

一、外设各模块实现方法

1、数码管驱动模块

驱动的原理为：根据选择信号，选取寄存器中不同的字节，根据其中的数值译码对应为驱动数码管的8位编码

```
`define dt0 8'b10000001
`define dt1 8'b11001111
`define dt2 8'b10010010
```

```
`define dt3 8'b10000110
`define dt4 8'b11001100
`define dt5 8'b10100100
`define dt6 8'b10100000
`define dt7 8'b10001111
`define dt8 8'b10000000
`define dt9 8'b10000100
`define dtA 8'b10001000
`define dtB 8'b11100000
`define dtC 8'b10110001
`define dtD 8'b11000010
`define dtE 8'b10110000
`define dtF 8'b10111000
```

写寄存器按照和DM相同的字节使能方式。

字偏移量	字节偏移量	描述	R/W	初始值
0x0	0x0	对应数码管组 0 的右 1 和右 2 数码管显示的一字节的 16 进制表示	R/W	0
	0x1	对应数码管组 0 的右 3 和右 4 数码管显示的一字节的 16 进制表示	R/W	0
	0x2	对应数码管组 1 的右 1 和右 2 数码管显示的一字节的 16 进制表示	R/W	0
	0x3	对应数码管组 1 的右 3 和右 4 数码管显示的一字节的 16 进制表示	R/W	0
0x1	0x4	对应符号位数码管，其低 4 位为符号位数码管显示的 16 进制数值，高 4 位未定义	R/W	0
	0x5	未定义	-	-
	0x6	未定义	-	-
	0x7	未定义	-	-

2、通用IO

所有的通用IO集成在一个电路模块内。

在读写时均会先将读出/写入数据取反再进行传递，故在软件使用时可认为1为开关开、LED亮。

偏移(按字)	设备种类	描述	方向	R/W
0x0	64 位拨码开关	第 0-3 组，组数和位数由低到高对应	I	R
0x1	64 位拨码开关	第 4-7 组，组数和位数由低到高对应	I	R
0x2	8 位按键开关	低 8 位表示 8 个按键开关的电平值，高 24 位未定义	I	R
0x3	-	未定义	-	
0x4	32 位 LED	32 个 LED 的电平值	O	R/W

3、UART控制器

更改部分：新增interrupt输出端口，直接接在原bridge的interrupt端口，在内部interrupt和rs电平相同。

更改read_over = STB_I & (~WE_I) & (ADD_I==`OFF_UART_DATA)，即不写并且地址为该板块的时候允许读

更改head_uart中的时钟频率与CPU时钟频率相等

偏移(以字为单位)	寄存器名称	寄存器描述	R/W	复位值
0x0	DATA	数据寄存器 仅低 8 位(一个字节)可用。 写入表示发送 UART 数据 读取表示接收 UART 数据	R/W	0
0x1	Undefined	未定义	-	0
0x2	Undefined	未定义	-	0
0x3	Undefined	未定义	-	0
0x4	LSR	状态寄存器 第 0 位表示接收状态， 第 5 位表示发送状态， 其他位未定义。	R	详见附件
0x5	DIVR	接收除数因子寄存器	R/W	不确定
0x6	DIVT	发送除数因子寄存器	R/W	不确定

二、测试代码（部分）

```
li $t1, 0x7f40
```

```
li $8, 0x12345678
sw $8, 0($t1)
loop:
li $t0, 0x7f58
li $t2, 0x7f60
li $t7, 0x7f40
lw $t1, 0($t0)
sw $t1, 0($t2)
lw $t3, -4($t0)
lw $t4, -8($t0)
sw $t4, 0($t7)
j loop
end
```

三、软件及其说明

1、简易计算器

微动开关3-0组：第一计算数B

微动开关7-4组：第一计算数A

通用按键开关：从右往左：addu, subu, or, and, xor, nor, sllv, srlv

```
li $t1, 0x7f50
li $t4, 0x7f40
li $t5, 0x7f58
li $t2, 0
li $t3, 0
li $t6, 0
li $t7, 0
loop:
lw $t2, 0($t1) #2 contain now lower
lw $t3, 4($t1) #3 contain now upper
lw $t6, 0($t5)
beq $t6, $0, end_if
nop
if_change:
    bne $t6, 1, case0e
    nop
        addu $t8, $t2, $t3
        j end_case
        nop
    case0e:
    bne $t6, 2, case1e
    nop
        subu $t8, $t3, $t2
        j end_case
        nop
    case1e:
    bne $t6, 4, case2e
    nop
        or $t8, $t3, $t2
        j end_case
        nop
    case2e:
    bne $t6, 8, case3e
```

```
    nop
        and $t8, $t3, $t2
        j end_case
        nop
    case3e:
    bne $t6, 16, case4e
    nop
        xor $t8, $t3, $t2
        j end_case
        nop
    case4e:
    bne $t6, 32, case5e
    nop
        nor $t8, $t3, $t2
        j end_case
        nop
    case5e:
    bne $t6, 64, case6e
    nop
        sllv $t8, $t3, $t2
        j end_case
        nop
    case6e:
    bne $t6, 128, case7e
    nop
        srlv $t8, $t3, $t2
        j end_case
        nop
    case7e:
    end_case:
    sw $t8, 0($t4)
end_if:
j loop
nop
```

2、计时器

用第 3~0 组拨码开关输入一个初始值。当系统复位后，程序读取拨码开关的值并从这个值开始向下计数（每秒减1）至 0。当程序运行时，若拨码开关的值发生变化，则重新从拨码开关读取初值，并重新计数。

注：当程序倒计时到0时会向串口传入字符 E，此时必须先按下reset才能再进行计数

```
.text
li $t0, 0xff11
mtc0 $t0, $12 #preset cause
li $s0, 0x7f50 #button
li $s1, 0x7f00 #counter
li $s2, 0x7f40 #tube
li $t0, 0
li $s3, 0

again:
lw $t1, 0($s0) #t1 is button numer now
beq $t1, $s3, continue #if not restart
nop
```

```
#begin or changed
move $s3, $t1  #s3 contains button number
move $t0, $s3 #preset reg counter
li $t7, 0
sw $t7, 0($s1)  #stop timmer
li $t2, 50000000
sw $t2, 4($s1) #preset
li $t7, 9
sw $t7, 0($s1)  #start timmer

continue:
sw $t0, 0($s2)
j again
nop


.ktext 4180
beq $t0, $0, end
nop
subu $t0, $t0, 1
li $t7, 0
sw $t7, 0($s1)  #stop timmer
li $t2, 50000000
sw $t2, 4($s1) #preset
li $t7, 9
sw $t7, 0($s1)  #start timmer
eret
nop
end:
    li $t9, 0x7f20
    li $t8, 69
    sw $t8, 0($t9)
loop:
    j loop
    nop
```

3、UART回显

设置波特率为9600，从UART输入的数据会按顺序原样输出回来

```
.text
li $s0, 0x7f20 #UART
li $t0, 0xff11
mtc0 $t0, $12 #preset SR
li $s1, 0 #head
li $s2, 0 #tail
loop:
    lw $t0, 16($s0)
    srl $t0, $t0, 5
    andi $t0, $t0, 1
    bne $t0, 1, loop #can't send
    nop
    beq $s1, $s2, empty #empty
    nop
    lw $t0, 0($s1)
    addiu $s1, $s1, 4
    bne $s1, 0x3000, inrange
```

```
    nop
        li $s1, 0
    inrange:
    sw $t0, 0($s0)
    j loop
    nop

empty:
    j loop
    nop

.ktext 4180
lw $t3, 0($s0)
sw $t3, 0($s2)
addiu $s2, $s2, 4
bne $s2, 0x3000, inrange2
nop
    li $s2, 0
inrange2:
beq $s1, $s2, over
nop
eret
nop
over:
    li $t7, 0xffffffff  # LED on.
    li $s4, 0x7f60
    sw $t7, 0($s4)
    beq $0, $0, over
    nop
```