P7 实验设计报告

18373085 张海渝

一. CP0 设计

信号设计：

序号	功能名称	方向	功能描述
1	A1[4:0]	I	写入或者写出的寄存器编号
2	Din[31:0]	I	写入 CP0 寄存器中的数值
3	ExcCode[5:0]	I	异常编号(下面会进行具体的说明)
4	HWInt[5:0]	I	外部的中断信号
5	BD	I	判断当前指令是否为延迟槽中的指令
6	WE	I	写使能
7	clk	I	时钟信号
8	reset	I	复位信号
9	EXLclr	I	将 SR 中的 EXL 为复位
10	IntReq	O	是否中断异常信号
11	EPCout[31:0]	O	CP0 中的 EPC 的值
12	Dout	O	对应于 A1 编号的寄存器

解释：

1. CP0 中共有 4 个寄存器，分别为 SR，Cause，EPC，PRID（只需考虑有效位，其余为置为 0）

   SR：SR[15:10]：为允许外部中断位，
   SR[1]：当进入中断或者异常的状态时，置为 1
   SR[0]:全局中断使能，是否允许中断

   Cause: Cause[6:2]为异常类型信号
   Cause[31]为 BD 位
   Cause[15:10]为中断信号位

   EPC: 中断异常时的指令 PC 值（在 BD 为 1 的时候为其前序指令）

   PRID：随意
2. 首先考虑处理中断异常的时候，可知

   IntReq=(|(HWInt[5:0]&SR[15:10])&SR[0]&~SR[1])|(ExcCode[5])

   当发生中断或者异常的时候需要
3. WE 是在 D 级中的控制器中产生信号传至 M 级（因为 CP0 在 M 级）。
4. 在中断异常信号（IntReq）置位的时候，需要将 EPC 传为当前 PC 值，并且需要考虑 BD 位，BD 为 1 需要将 EPC-4
5. 需要注意在中断异常的时候需要将进入 pc 值为 0x4180 的指令。

二. Bridge 设计

信号设计：

序号	功能名称	方向	功能描述

1	PrAddr[31:0]	I	写或读外部数据的地址
2	Dout0[31:0]	I	Timer0 中的数据值（对应于 PrAddr）
3	Dout1[31:0]	I	Timer1 的数据值（对应于 PrAddr）
4	PrWe	I	是否写 Timer 的信号
5	WE0	O	是否写 Timer0 的信号
6	WE1	O	是否写 Timer1 的信号
7	PrRD[31:0]	O	输出值 cpu 的值

解释：

1. 需要注意在将 PrRD 传回的时候需要注意不要将 DM 与这个值混淆，所以可以加上一个 HITDM 信号来区分
2. 对于 WE 来说，有几个外部设备就有几个 WE，因为这次只用两个 Timer 所以只需要两个 WE
3. 产生信号都是组合逻辑，见下图：

```
wire [31:0] addr;
assign addr={PrAddr[31:2],2'b0};
assign PrRD=   addr>=32'h7f00 && addr<=32'h7f0B ? Dout0 :
               addr>=32'h7f10 && addr<=32'h7f1B ? Dout1 : 0 ;
assign WE0=addr>=32'h7f00 && addr<=32'h7f0B&&PrWe?1:0;
assign WE1=addr>=32'h7f10 && addr<=32'h7f1B&&PrWe?1:0;
```

图一：Bridge 信号图

三． 测试软件

1. 测试 cpu：

1.R 型指令（addu 为例，乘除包含 mflo，mfhi，slt 等类型指令）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-M-RS	subu	E	rs	subu $1,$2,$3 addu $4,$1,$2	转发
R-M-RT	subu	E	rt	subu $1,$2,$3 addu $4,$2,$1	转发

R-W-RS	subu	E	rs	subu $1,$2,$3
				nop
				addu $4,$1,$2
					转发

R-W-RT	subu	E	rt	subu $1,$2,$3
				nop
				addu $4,$2,$1
					转发

I-M-RS	ori	E	rs	ori $1,$0,0x1111 addu $3,$1,$2	转发
I-M-RT	ori	E	rt	ori $1,$0,0x1111 addu $3,$2,$1	转发
I-W-RS	ori	E	rs	ori $1,$0,0x1111 nop	转发

				addu $3,$1,$2	
I-W-RT	ori	E	rt	ori $1,$0,0x1111 nop addu $3,$2,$1	转发

LD-W-RS	lw	E	rs	lw $1,0($2)
				nop
				addu $3,$1,$2
					转发

LD-W-RT	lw	E	rt	lw $1,0($2)
				nop
				addu $3,$2,$1
					转发

LD-M-RS	lw	E	rs	lw $1,0($2) addu $3,$1,$2	暂停
LD-M-RT	lw	E	rt	lw $1,0($2) addu $3,$1,$2	暂停
LD-E-RS	lw	D	rs	lw $1,0($2) addu $3,$1,$2	暂停
LD-E-RT	lw	D	rt	lw $1,0($2) addu $3,$1,$2	暂停
JAL-M-RS	jal	D	rs	jal dfs nop dfs: addu $1,$31,$0	转发
JAL-M-RT	jal	D	rt	jal dfs nop dfs: addu $1,$0,$31	转发
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop addu $1,$31,$0	转发
JAL-W-RT	jal	D	rt	jal dfs nop dfs: nop addu $1,$0,$31	转发

2.I 型指令（ori 为例，其中包含了 sll 等指令的测试）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-M-RS	subu	E	rs	subu $1,$2,$3 ori $4,$1,0x1111	转发
R-W-RS	subu	E	rs	subu $1,$2,$3 nop ori $4,$1,0x1111	转发

I-M-RS	ori	E	rs	ori $1,$0,0x1110 ori $2,$1,0x1111	转发
I-W-RS	ori	E	rs	ori $1,$0,0x1110 nop ori $3,$1,0x1111	转发
LD-W-RS	lw	E	rs	lw $1,0($2) nop ori $3,$1,0x0	转发
LD-M-RS	lw	D	rs	lw $1,0($2) nop ori $3,$1,$2	暂停
LD-E-RS	lw	D	rs	lw $1,0($2) ori $3,$1,$2	暂停
JAL-M-RS	jal	D	rs	jal dfs nop dfs: ori $1,$31,0x0	转发
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop ori $1,$31,0x0	转发

3.LD 型指令（lw 为例）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-M-RS	subu	E	rs	subu $1,$2,$3 lw $4,0($1)	转发

R-W-RS	subu	E	rs	subu $1,$2,$3
				nop
				lw $4,0($1)
					转发

I-M-RS	ori	E	rs	ori $1,$0,0x1110 lw $4,0($1)	转发
I-W-RS	ori	E	rs	ori $1,$0,0x1110 nop lw $4,0($1)	转发

LD-W-RS	lw	E	rs	lw $1,0($2)
				nop
				lw $4,0($1)
					转发

LD-M-RS	lw	D	rs	lw $1,0($2)
				nop
				lw $4,0($1)
					转发

LD-E-RS	lw	D	rs	lw $1,0($2) lw $4,0($1)	暂停

JAL-M-RS	jal	D	rs	jal dfs nop dfs: lw $4,0($31)	转发
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop lw $4,0($31)	转发

4.Store 型指令（sw 为例）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-M-RS	subu	E	rs	subu $1,$2,$3 sw $4,0($1)	转发
R-M-RT	subu	E	rt	subu $1,$2,$3 sw $1,0($0)	转发

R-W-RS	subu	E	rs	subu $1,$2,$3
				nop
				sw $4,0($1)
					转发

R-W-RS	subu	E	rt	subu $1,$2,$3
				nop
				sw $1,0($0)
					转发

I-M-RS	ori	E	rs	ori $1,$0,0x1110 sw $4,0($1)	转发
I-W-RS	ori	E	rs	ori $1,$0,0x1110 nop sw $4,0($1)	转发
I-M-RT	ori	E	rt	ori $1,$0,0x1110 sw $1,0($0)	转发
I-W-RT	ori	E	rt	ori $1,$0,0x1110 nop sw $1,0($0)	转发

LD-W-RS	lw	E	rs	lw $1,0($2)
				nop
				sw $4,0($1)
					转发

LD-W-RT	lw	E	rt	lw $1,0($2)
				nop
				sw $1,0($0)
					转发

LD-M-RS	lw	D	rs	lw $1,0($2)
				nop
				sw $4,0($1)
					转发

LD-M-RT	lw	D	rt	lw $1,0($2)
				nop
				sw $1,0($0)
					转发

LD-E-RS	lw	D	rs	lw $1,0($2) sw $4,0($1)	暂停
LD-E-RT	lw	D	rt	lw $1,0($2) sw $1,0($0)	转发
JAL-M-RS	jal	D	rs	jal dfs nop dfs: sw $4,0($31)	转发
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop sw $4,0($31)	转发

5.B 型指令（beq 为例）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-E-RS	subu	D	rs	subu $1,$2,$3 beq $1,$2,loop	暂停
R-E-RT	subu	D	rt	subu $1,$2,$3 beq $2,$1,loop	暂停
R-M-RS	subu	D	rs	subu $1,$2,$3 nop beq $1,$2,loop	转发
R-M-RT	subu	D	rt	subu $1,$2,$3 nop beq $2,$1,loop	转发
I-E-RS	ori	D	rs	ori $1,$2,0x1111 beq $1,$2,loop	暂停
I-E-RT	ori	D	rt	ori $1,$2,0x1111 beq $2,$1,loop	暂停
I-M-RS	ori	D	rs	ori $1,$2,0x1111 nop beq $1,$2,loop	转发
I-M-RT	ori	D	rt	ori $1,$2,0x1111 nop beq $2,$1,loop	转发
LD-E-RS	lw	D	rs	lw $1,0($2) beq $1,$2,loop	暂停
LD-E-RT	lw	D	rt	lw $1,0($2) beq $2,$1,loop	暂停
LD-M-RS	lw	D	rs	lw $1,0($2) nop beq $1,$2,loop	暂停

LD-M-RT	lw	D	rt	lw $1,0($2) nop beq $2,$1,loop	暂停
LD-W-RS	lw	D	rs	lw $1,0($2) nop nop beq $1,$2,loop	转发
LD-W-RT	lw	D	rt	lw $1,0($2) nop nop beq $2,$1,loop	转发
JAL-M-RS	jal	D	rs	jal dfs nop dfs: beq $4,$31,loop	暂停
JAL-M-RT	jal	D	rt	jal dfs nop dfs: beq $31,$4,loop	暂停
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop beq $4,$31,loop	转发
JAL-W-RT	jal	D	rt	jal dfs nop dfs: nop beq $31,$4,loop	转发

6.jr 指令：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-E-RS	subu	D	rs	subu $1,$2,$3 jr $1	暂停
R-M-RS	subu	D	rs	subu $1,$2,$3 nop jr $1	转发
I-E-RS	ori	D	rs	ori $1,$2,0x1111 jr $1	暂停
I-M-RS	ori	D	rs	ori $1,$2,0x1111 nop jr $1	转发
LD-E-RS	lw	D	rs	lw $1,0($2) jr $1	暂停

LD-M-RS	lw	D	rs	lw $1,0($2) nop jr $1	暂停
LD-W-RS	lw	D	rs	lw $1,0($2) nop nop jr $1	转发
JAL-M-RS	jal	D	rs	jal dfs nop dfs: jr $31	暂停
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop jr $31	转发

7.乘除指令（以 mult 为例）

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-E-RS	mfhi	D	HI	mfhi $1 mult $2,$3	暂停

2. 测试异常：

1. AdEL

```
.text
ori  $2,$0,0x1c01 #允许所有的中断
mtc0  $2,$12
jal  loop
nop
nop  #  jr  $ra

loop:
addi  $ra,$ra,-1 #或者改为  ori  $ra,$0,0x4fff
jr  $ra
nop

.ktext  0x00004180
ori  $26,0x3010
mtc0  $26,$14
eret
ori  $1,$0,0x1111
```

2. AdEL,AdEs（相当于 lw，sw 的区别）

```
.text
ori  $2,$0,0x1c01 #允许所有的中断
mtc0  $2,$12
ori  $3,0x7f00
ori  $4,0x0003
lw  $3,0($4)#  可以将 lw 换为 lh，lhu
                #或者将此指令换为 lh(lhu,lb,lbu)  $4,0($3)
                #或者改为 sw  $4,8($3)
nop
nop
nop
.ktext  0x00004180
ori  $3,$0,0x0004
ori  $4,$0,0x0004
eret
ori  $1,$0,0x1111
```

3. RI

```
.text
ori  $2,$0,0x1c01 #允许所有的中断
mtc0  $2,$12
ori  $3,0x7f00
ori  $4,0x0003
msub  $3,$4
nop
nop
nop
.ktext  0x00004180
ori  $3,$0,0x0004
ori  $4,$0,0x0004
eret
ori  $1,$0,0x1111
```

4. Ov

```
.text
ori  $2,$0,0x1c01 #允许所有的中断
mtc0  $2,$12
lui  $3,0x7fff
lui  $4,0x7fff
add  $5,$3,$4

nop
nop
nop

.ktext  0x00004180
```

```
ori  $3,$0,0x0004
ori  $4,$0,0x0004
eret
ori  $1,$0,0x1111
```

3.测试中断

1.测试 Timer(以 Timer0 为例)

```
.text
ori  $4,$0,0x0401
mtc0  $4,$12
ori  $1,$0,0x7f00
ori  $2,$0,0x0009
ori  $3,$0,100
sw  $2,0($1)  #timer0 中的 ctrl 寄存器
sw  $3,4($1)  #timer0 中的 present 寄存器

loop:
j  loop
nop

.ktext  0x00004180
ori  $26,$0,0x7f00
lw  $27,0($26)
ori  $27,$27,1
sw  $27,0($26)
lw  $27,4($26)
ori  $27,$0,1000  #改变 present 的值
sw  $27,4($26)
eret
nop
```

3. 中断异常同时发生

```
.text
ori  $2,$0,0x1c01  #允许所有的中断
mtc0  $2,$12
lui  $4,0x7fff
lui  $5,0x7fff
ori  $1,$0,0x7f00
ori  $2,$0,0x0009
ori  $3,$0,10
sw  $2,0($1)  #timer0 中的 ctrl 寄存器
sw  $3,4($1)  #timer0 中的 present 寄存器
nop
nop
nop
```

```
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        add  $6,$4,$5


        .ktext  0x00004180
        ori  $26,$0,0x7f00
        lw  $27,0($26)
        ori  $27,$27,1
        sw  $27,0($26)
        eret
nop
```

四. 思考题

1. 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”?
   答：硬件/软件接口即将软件和硬件行为都通过一套指令系统结构进行操作，而不需要关注于软件和硬件具体的行为。
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。
   答：应该在 CPU 的外部，与 cache 等一系列存储器通过交互而获得信息，这样能够加快访问速度以及加快访问周期。
3. BE 部件对所有的外设都是必要的吗?
   答：不是，只有需要读取字节的时候才需要，如果只是针对一个字的读取，只需要首地址，而不需要 BE。
4. 请阅读官方提供的定时器源代码，阐述两种中断模式的异同，并分别针对每一种模式绘制状态转移图（见计时器文档）

5. 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：

   i. 定时器在主程序中被初始化为模式 0；

   ii. 定时器倒计数至 0 产生中断；

   iii. 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。

iv. 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。（注意，主程序可能需要涉及对 CP0.SR 的编程，推荐阅读过后文后再进行。）

```
.text
ori  $4,$0,0x0401
mtc0  $4,$12
ori  $1,$0,0x7f00
ori  $2,$0,0x0009
ori  $3,$0,100
sw  $2,0($1)  #timer0 中的 ctrl 寄存器
sw  $3,4($1)  #timer0 中的 present 寄存器

loop:
j  loop
nop

.ktext  0x00004180
ori  $26,$0,0x7f00
lw  $27,0($26)
ori  $27,$27,1
sw  $27,0($26)
lw  $27,4($26)
ori  $27,$0,1000  #改变 present 的值
sw  $27,4($26)
eret
nop
```

6. 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?
答：通过中断获取的，即当输入一个数的时候，然后会产生一个中断信号，然后通过软件行为，将其读入某寄存器，或者 DM 中。