北航2021计组P7微系统设计方案综述

黄雨石20376156

一、设计方案

(一) 总体设计概述

本微系统为`verilog`实现的 MIPS 微系统，包括流水线 CPU、Bridge、计时器，CPU 支持的指令集包括`{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, mfc0, mtc0, eret}`。为了实现这些功能，`CPU`主要包含了`PC、CMP、GRF、ALU、DP、EXT、 CTRL、BE、CP0`以及每级模块之间的流水寄存器模块、确定转发和暂停机制的处理冒险的模块单元等

(二)CPU模块定义

- **F级**

  1.F_PC

端口名	方向	描述
npc[31:0]	I	D级npc当前值
pc[31:0]	O	F级pc当前值
pwe	I	使能信号
reset	I	复位信号
clk	I	时钟信号
req	I	中断异常信号
eret	I	回滚信号
epc[31:0]	I	EPC
ex_adel	O	取指异常

- **D级**

  1.D_REG

端口名	方向	描述
reset	I	复位信号
clk	I	时钟信号
we	I	使能信号
instr_in[31:0]	I	F级指令
instr_out[31:0]	O	F级流水过来的指令
pc_in[31:0]	I	F级pc值
pc_out[31:0]	O	F级流水过来的pc值
req	I	中断异常信号
ex_code_in[4:0]	I	F级异常编码
ex_code_out[4:0]	O	流入D级的F级异常编码
bd_in	I	F级指令是否是延迟槽指令
bd_out	O	流入D级的F级指令是否是延迟槽指令

2.D_CMP

端口名	方向	描述
rs_d[31:0]	I	D级rs寄存器数据
rt_d[31:0]	I	D级rt寄存器数据
type[2:0]	I	选择信号
b_j	O	是否跳转

- type:
  - 3'd0: 跳转指令为`beq`
  - 3'd1:跳转指令为`bne`
  - 3'd2:跳转指令为`blez`
  - 3'd3:跳转指令为`bgez`
  - 3'd4:跳转指令为`bltz`
  - 3'd5:跳转指令为`bgtz`

3.D_EXT

端口名	方向	描述
imm16[15:0]	I	D级16位立即数
ext_sel	O	选择信号
ext_out[31:0]	I	扩展后结果

- ext_sel:

    - 1'b0:0拓展
    - 1'b1:符号拓展

4.D_GRF

端口名	方向	描述
rs[4:0]	I	rs寄存器地址
rt[4:0]	I	rt寄存器地址
gwa[4:0]	I	写入寄存器的地址
gwd[31:0]	I	写入寄存器的值
rs_d	O	rs寄存器的值
rt_d	O	rt寄存器的值
gwe	I	寄存器写使能
reset	I	复位信号
clk	I	时钟信号
W_pc	I	W级pc值

5.D_NPC

端口名	方向	描述
D_pc[31:0]	I	D级pc值
F_pc[31:0]	I	F级pc值
imm26[25:0]	I	26位立即数
imm16[15:0]	I	16位立即数
rs_d[31:0]	I	rs寄存器的值
npc_sel[2:0]	I	选择信号
b_j	I	分支指令是否跳转
npc[31:0]	O	F级pc下一周期的值
req	O	中断异常信号
eret	I	回滚信号
epc[31:0]	I	EPC

    - npc_sel:
        - 3'd0: `F_pc + 4`
        - 3'd1 `·j_i` 类指令 `{D_pc[31:28], imm26, 2'b0}`
        - 3'd2 `j_r` 类指令: `rs_d`
        - 3'd3: `branch` 类指令 `D_pc + 4 + {{14{imm16[15]}}, imm16, 2'b0}`
- **E级**

1.E_REG

端口名	方向	描述
reset	I	复位信号
clk	I	时钟信号
we	I	使能信号
instr_in[31:0]	I	D级指令
instr_out[31:0]	O	D级流水过来的指令
pc_in[31:0]	I	D级pc值
pc_out[31:0]	O	D级流水过来的pc值
ext_in[31:0]	I	D级立即数拓展结果
ext_out[31:0]	O	D级流水过来的立即数拓展结果
rs_d_in[31:0]	I	D级rs寄存器值
rs_d_out[31:0]	O	D级流水过来的rs寄存器值
rt_d_in[31:0]	I	D级rt寄存器值
rt_d_out[31:0]	O	D级流水过来的rt寄存器值
req	I	中断异常信号
ex_code_in	I	D级异常编码
ex_code_out	O	流入E级的F级异常编码
bd_in	I	D级指令是否是延迟槽指令
bd_out	O	流入E级的D级指令是否是延迟槽指令
stall	I	阻塞信号

2.ALU

端口名	方向	描述
a[31:0]	I	数据a
b[31:0]	I	数据b
alu_out[31:0]	O	alu计算结果
alu_sel[4:0]	I	选择信号
alu_ari_of	I	add,addi,sub信号
alu_dm_of	I	访存类指令信号
ex_ov_ari	O	计算类指令溢出信号
ex_ov_dm	O	访存类指令地址计算溢出信号

- alu_sel:
  - 5'd0: `add:a+b`
  - 5'd1: `sub:a-b`
  - 5'd2: `or:a|b`
  - 5'd3: `and:a&b`
  - 5'd4: `xor:a^b`
  - 5'd5: `sll:a<<b`
  - 5'd6: `srl:a>>b`
  - 5'd7: `sra:$signed($signed(a)>>>b)`
  - 5'd8: `slt:($signed(a)<$signed(b))?32'b1:32'b0`
  - 5'd9: `sltu:a<b`
  - 5'd10: `lui:b<<16`

3.E_MDU

端口名	方向	描述
clk	I	时钟信号
reset	I	复位信号
d1[31:0]	I	数据1
d2[31:0]	I	数据2
md_sel[2:0]	I	选择信号
md_stall	O	阻塞信号
md_out[31:0]	O	输出结果
req	I	中断异常信号

- md_sel:
  - 4'd0: `mult`
  - 4'd1: `multu`
  - 4'd2 `div`
  - 4'd3: `divu`
  - 4'd4: `mfhi`
  - 4'd5 `mflo`
  - 4'd6 `mthi`
  - 4'd7 `mtlo`

- **M级**

  1.M_REG

端口名	方向	描述
reset	I	复位信号
clk	I	时钟信号
we	I	使能信号
instr_in[31:0]	I	E级指令
instr_out[31:0]	O	E级流水过来的指令
pc_in[31:0]	I	E级pc值
pc_out[31:0]	O	E级流水过来的pc值
alu_in[31:0]	I	E级alu的计算结果
alu_out[31:0]	O	E级流水过来的alu的计算结果
rt_d_in[31:0]	I	流水到E级的rt寄存器的值
rt_d_out[31:0]	O	流水到M级的rt寄存器的值
req	I	中断异常信号
ex_code_in[4:0]	I	E级异常编码
ex_code_out[4:0]	O	流入M级的E级异常编码
bd_in	I	E级指令是否是延迟槽指令
bd_out	O	流入M级的E级指令是否是延迟槽指令
ex_ov_dm_in	I	访存类指令地址计算溢出信号
ex_ov_dm_out	O	流入M级的访存类指令地址计算溢出信号

2.M_BE

端口名	方向	描述
dwa[31:0]	I	地址
dwd_temp[31:0]	I	写入数据（中间）
be_sel[2:0]	I	选择信号
byteen[3:0]	O	替换信号
dwd[31:0]	O	写入数据（最终）
ex_ov_dm	I	访存类指令地址计算溢出信号
ex_ades	O	储存类指令异常
store	I	储存类指令

- be_sel:

- 3'd0: `sw`
- 3'd1: `sh`
- 3'd2: `sb`

3.M_DP

端口名	方向	描述
dm_temp[31:0]	I	DM中间结果
dp_sel[2:0]	I	选择信号
dp_a[31:0]	I	地址
dm_out[31:0]	O	DM最终结果
load	I	读数类指令
ex_ov_dm	I	访存类指令地址计算溢出信号
ex_adel	O	读数类指令异常
dwa[31:0]	I	地址

- dm_sel:
  - 3'd0: `lw`
  - 3'd1: `lh`
  - 3'd2: `lb`
  - 3'd3: `lhu`
  - 3'd4: `lbu`

4.M_CP0

端口名	方向	描述
a1[4:0]	I	mfc0读数地址
a2[4:0]	I	mtc0写数地址
bd_in	I	异常指令是否是延迟槽指令
cwd[31:0]	I	写入寄存器的数
pc[31:2]	I	异常指令的pc值
ex_code_in[4:0]	I	异常编码
hwint[5:0]	I	外部中断信号
cwe	I	寄存器写使能
exl_clr	I	中断异常状态清空信号
clk	I	时钟信号
reset	I	复位信号
req	O	中断异常信号
epc_out[31:0]	O	epc寄存器的值
cp0_out[31:0]	O	cp0寄存器读取值
int_en	O	是否响应外部的interrupt信号

- **W级**

  1.W_REG

端口名	方向	描述
reset	I	复位信号
clk	I	时钟信号
we	I	使能信号
alu_in[31:0]	I	流水到M级的alu计算结果
alu_out[31:0]	O	流水到W级的alu计算结果
pc_in[31:0]	I	M级pc值
pc_out[31:0]	O	W级pc值
dm_in[31:0]	I	M级dm输出结果
dm_out[31:0]	O	流水到W级dm输出结果
instr_in[31:0]	I	M级指令
instr_out[31:0]	O	W级指令
md_in[31:0]	I	E级乘除槽计算结果
md_out[31:0]	O	流入M级乘除槽计算结果
req	I	中断异常信号
cp0_in	I	cp0输出结果
cp0_out	O	流·入W级cp0输出结果

- **控制**

  1.CTRL

端口名	方向	描述
instr[31:0]	I	指令
rs[4:0]	O	rs寄存器地址
rt[4:0]	O	rt寄存器地址
rd[4:0]	O	rd寄存器地址
imm16[15:0]	O	16位立即数
imm26[25:0]	O	26位立即数
shamt[4:0]	O	移位数
load	O	读取dm类指令
store	O	储存到dm类指令
count_i	O	有立即数的计算类指令
count_r	O	无立即数的计算类指令
branch	O	分支类指令
shifts	O	用shamt的移位指令
shiftv	O	用寄存器值进行移位的指令
j_r	O	用寄存器值进行跳转类指令
j_i	O	用立即数进行跳转类指令
j_l	O	跳转并链接类指令
md	O	乘除指令
mf	O	读lo, hi 寄存器
mt	O	写lo,hi寄存器
type[2:0]	O	CMP选择信号
ext_sel	O	EXT选择信号
gwe	O	GRF写使能信号
gwa_res[4:0]	O	GRF写入地址选择信号
gwd_sel[2:0]	O	GRF写入地址值选择信号
npc_sel[2:0]	O	NPC选择信号
alu_sel[4:0]	O	ALU选择信号
alu_a_sel[1:0]	O	ALU数据a选择·信号
alu_b_sel[2:0]	O	ALU数据b选择信号
be_sel[2:0]	O	BE选择信号

端口名	方向	描述
dp_sel[2:0]	O	DP选择信号
alu_ari_of	O	add,addi,sub信号
alu_dm_of	O	访存类指令信号
ex_ri	O	无法识别指令信号
cwe	O	cp0写使能
mfc0	O	mfc0信号
mtc0	O	mtc0信号
eret	O	eret信号

    - gwa_res:
        - `count_r:rd`
        - `count_i | load:rt`
        - `jal:5'd31`
    - gwd_sel:
        - 3'd0: `alu`
        - 3'd1: `dm`
        - 3'd2: `pc+8`
    - alu_a_sel:
        - 2'b0: `rs_d`
        - 2'b1: `rt_d`
    - alu_b_sel:
        - 3'd0: `rt_d`
        - 3'd1: `ext_out`
        - 3'd2: `rs_d[4:0]`
        - 3'd3: `shamt`

- **冒险**

  1.SU

端口名	方向	描述
D_instr[31:0]	I	D级指令
E_instr[31:0]	I	E级指令
M_instr[31:0]	I	M级指令
md_stall	I	乘除槽繁忙信号
stall	O	阻塞信号

(三)BRIDGE模块定义

- **桥**

  1.BRIDGE

端口名	方向	描述
pr_a[31:0]	I	写入外设地址
pr_we	I	外设写使能
interrupt	I	外部interrupt信号
irq1	I	tc1中断信号
irq2	I	tc2中断信号
tc1_out[31:0]	O	tc1读取值
tc2_out[31:0]	O	tc2读取值
tc1_we	O	tc1写使能
tc2_we	O	tc2写使能
pr_rd[31:0]	O	外设读取值
hwint[5:0]	O	外部中断信号

二、重要机制实现方法

- 将异常信号流水到`M`级，同时结合外部中断信号，判断此时是否要进行中断，如果进行中断就清空所有流水寄存器，但同时注意每个流水线寄存器流水的`pc`值附为`0x00004180`，并将`nPC`改为`0x00004180`。这样做`W`级流水线寄存器的指令如果需要写入`GRF`，可以恰好写入而不用再特殊处理。
- 如果`D`级检测到到`eret`回滚信号，那么`F`级`pc`改为`epc`，`npc`改为`epc+4`。
- 如果处于阻塞状态，那么插入的`nop`要接着流水`pc`、`bd`。
- 注意如果`eret`在`D`级，而`E`级或`M`级是`mtc0`且恰好是写`epc`寄存器，则进行阻塞，直到`epc`已经被写入为止
- 在本设计中桥仅连接`cpu`与`tc1`，`tc2`，`dm`则不需要通过桥与`cpu`交互，采用直接交互的方法。

三、测试方案

- 取址异常

```
.text

li $28, 0
li $29, 0

# jr PC mod 4 not 0
la $1, label1
la $2, label1
```

```
addiu $1, $1, 1
jr $1
nop
label1:

# jr PC < 0x3000
li $1, 0x2996
la $2, label2
jr $1
nop
label2:

# jr PC > 0x6ffc
li $1, 0x6fff
la $2, label3
jr $1
nop
label3:

end:j end

.ktext 0x4180
mfc0 $12, $12
mfc0 $13, $13
mfc0 $14, $14
mtc0 $2, $14#set epc = label
eret
ori $1, $0, 0
```

- 其它异常

```
.text

li $28, 0
li $29, 0
#l
#对齐
lw $1, 1($0)
lh $1, 1($0)
lhu $1, 1($0)
#ct1 ,ct2
lh $1, 0x7f00($0)
lhu $1, 0x7f04($0)
lb $1, 0x7f08($0)
lbu $1, 0x7f10($0)
lb $1, 0x7f14($0)
lb $1, 0x7f18($0)
#越界
lw $1, 0x3000($0)
lh $1, 0x4000($0)
lhu $1, 0x6000($0)
lb $1, 0x7f0c($0)
lbu $1, 0x7f1c($0)
#计算加法溢出
li $2, 0x7fffffff
lw $1, 1($2)
lh $1, 1($2)
```

```
lhu $1, 1($2)
lb $1, 1($2)
lbu $1, 1($2)

#s
#对齐
sw $1, 1($0)
sh $1, 1($0)
#ct1,ct2
sh $1, 0x7f00($0)
sb $1, 0x7f04($0)
sh $1, 0x7f08($0)
sb $1, 0x7f10($0)
sh $1, 0x7f14($0)
sb $1, 0x7f18($0)
#计算加法溢出
li $2, 0x7fffffff
sw $1, 1($2)
sh $1, 1($2)
sb $1, 1($2)
#ct1,ct2
sw $1, 0x7f08($0)
sh $1, 0x7f08($0)
sb $1, 0x7f08($0)
sw $1, 0x7f18($0)
sh $1, 0x7f18($0)
sb $1, 0x7f18($0)
#越界
sh $1, 0x4000($0)
sh $1, 0x6000($0)
sb $1, 0x7f0c($0)
sb $1, 0x7f1c($0)
#ri
msub $1, $2
#ov
li $1, 0x7fffffff
add $1, $1, $1
addi $1, $1, 1
li $1, 0x80000000
add $1, $1, $1
addi $1, $1, -1
sub $1, $1, $2
sub $1, $2, $1

end:j end


.ktext 0x4180
mfc0 $12, $12
mfc0 $13, $13
mfc0 $14, $14
addi $14, $14, 4# epc+4
mtc0 $14, $14
eret
ori $1, $0, 0
```

- 计时器功能测试

```
.text

li $1, 500
li $2, 9

sw $1, 0x7f04($0)
sw $2, 0x7f00($0)
li $1, 1000
sw $1, 0x7f14($0)
sw $2, 0x7f10($0)

lw $1, 0x7f00($0)
lw $1, 0x7f04($0)
lw $1, 0x7f10($0)
lw $1, 0x7f14($0)
```

- 延迟槽异常

```
.text

li $28, 0
li $29, 0

li $1, 1
bne $0, $0, end
lw $1, 1($0)
li $1, 1
bne $0, $0, end
lh $1, 1($0)
li $1, 1
bne $0, $0, end
lhu $1, 1($0)

li $1, 1
bne $0, $0, end
lh $1, 0x7f00($0)
li $1, 1
bne $0, $0, end
lhu $1, 0x7f04($0)
li $1, 1
bne $0, $0, end
lb $1, 0x7f08($0)
li $1, 1
bne $0, $0, end
lbu $1, 0x7f10($0)
li $1, 1
bne $0, $0, end
lb $1, 0x7f14($0)
li $1, 1
bne $0, $0, end
lb $1, 0x7f18($0)

li $2, 0x7fffffff
li $1, 1
bne $0, $0, end
lw $1, 1($2)
```

```
li $1, 1
bne $0, $0, end
lh $1, 1($2)
li $1, 1
bne $0, $0, end
lhu $1, 1($2)
li $1, 1
bne $0, $0, end
lb $1, 1($2)
li $1, 1
bne $0, $0, end
lbu $1, 1($2)

li $1, 1
bne $0, $0, end
lw $1, 0x3000($0)
li $1, 1
bne $0, $0, end
lh $1, 0x4000($0)
li $1, 1
bne $0, $0, end
lhu $1, 0x6000($0)
li $1, 1
bne $0, $0, end
lb $1, 0x7f0c($0)
li $1, 1
bne $0, $0, end
lbu $1, 0x7f1c($0)

li $1, 1
bne $0, $0, end
sw $1, 1($0)
li $1, 1
bne $0, $0, end
sh $1, 1($0)

li $1, 1
bne $0, $0, end
sh $1, 0x7f00($0)
li $1, 1
bne $0, $0, end
sb $1, 0x7f04($0)
li $1, 1
bne $0, $0, end
sh $1, 0x7f08($0)
li $1, 1
bne $0, $0, end
sb $1, 0x7f10($0)
li $1, 1
bne $0, $0, end
sh $1, 0x7f14($0)
li $1, 1
bne $0, $0, end
sb $1, 0x7f18($0)

li $2, 0x7fffffff
li $1, 1
bne $0, $0, end
```

```
sw $1, 1($2)
li $1, 1
bne $0, $0, end
sh $1, 1($2)
li $1, 1
bne $0, $0, end
sb $1, 1($2)

li $1, 1
bne $0, $0, end
sw $1, 0x7f08($0)
li $1, 1
bne $0, $0, end
sh $1, 0x7f08($0)
li $1, 1
bne $0, $0, end
sb $1, 0x7f08($0)
li $1, 1
bne $0, $0, end
sw $1, 0x7f18($0)
li $1, 1
bne $0, $0, end
sh $1, 0x7f18($0)
li $1, 1
bne $0, $0, end
sb $1, 0x7f18($0)

li $1, 1
bne $0, $0, end
sw $1, 0x3000($0)
li $1, 1
bne $0, $0, end
sh $1, 0x4000($0)
li $1, 1
bne $0, $0, end
sh $1, 0x6000($0)
li $1, 1
bne $0, $0, end
sb $1, 0x7f0c($0)
li $1, 1
bne $0, $0, end
sb $1, 0x7f1c($0)

li $1, 1
bne $0, $0, end
msub $1, $2



li $1, 0x7fffffff
li $11, 1
bne $0, $0, end
add $1, $1, $1
li $11, 1
bne $0, $0, end
addi $1, $1, 1
li $1, 0x80000000
li $11, 1
```

```
bne $0, $0, end
add $1, $1, $1
li $11, 1
bne $0, $0, end
addi $1, $1, -1
li $11, 1
bne $0, $0, end
sub $1, $1, $2
li $11, 1
bne $0, $0, end
sub $1, $2, $1

end:j end


.ktext 0x4180
mfc0 $12, $12
mfc0 $13, $13
mfc0 $14, $14
addi $14, $14, 8#pc+8
mtc0 $14, $14
eret
ori $1, $0, 0
```

- p6指令测试

  用`p6`自动对拍程序与`mars`结果进行对比，用的讨论区的`100`组强度为`100%`转发，`100%`阻塞的数据

四、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”？（Tips：什么是接口？和我们到现在为止所学的有什么联系？）**
   - “硬件/软件接口”是指令（机器码）。硬件实现了一些功能，并按照规约可以被相应的指令所操控。软件通过规约使用相应的指令操控硬件完成相应的功能，从而达到软件所期望的效果。指令在这个过程中实现了硬件软件的对接，因此是“硬件/软件接口”。
2. **BE 部件对所有的外设都是必要的吗?**
   - 不是，只有对**字节/半字**有存取需求的才有必要。
3. **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。**
   - 见计时器说明文档。
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。（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
        li $12, 0x0401
        mtc0 $12, $12
        # sr[0] = 1(global int enable), sr[1] = 0(exc state), sr[15:10] = 1(tc1 enable)
        ```
        ```

```
li $1, 100
li $2, 9
sw $1, 0x7f04($0)
# present = 100(initial)
sw $2, 0x7f00($0)
# ctrl[0] = 1(count enable), ctrl[3] = 1(int enable), ctrl[1:0] =
0(mode0)
for:j for
nop

.ktext 0x4180
li $1, 100
li $2, 9
sw $1, 0x7f04($0)
sw $2, 0x7f00($0)
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?**
    - 鼠标和键盘产生中断信号，进入中断处理区的对应位置，将输入信号从鼠标和键盘中读入寄存器。