Verilog流水线CPU设计文档

一． 数据通路　　设计者：忽骁

1. IFU

1）接口定义

文件名	模块接口定义
IFU.v	module IFU(input clk, input reset, input [31:0] npc, input [2:0] pcOp, input en, output reg [31:0] instr, output reg [31:0] pcPlus4, output reg [31:0] pc);

2）说明

时钟上升沿的时候更新 PC 和 instr， 若 pcOp==3’b000，则 pc<=pc+4，否则 pc<=npc(来自于 Npc 模块)

2. GRF

1） 接口定义

文件名	模块接口定义
GRF.v	module GRF(input [31:0] pc, input [4:0] a1, input [4:0] a2, input clk, input reset, input w, input [4:0] a3, input [31:0] wd, output [31:0] out1, output [31:0] out2);

2） 功能

信号	功能
pc	输入当前 pc
a1, a2	两个读端口，分别输出 out1, out2
a3	写端口，写入数据来自 wd
w	写使能端
clk	时钟控制
reset	同步复位

3. Sub

1) 接口定义

文件名	模块接口定义
Sub.v	module Sub(input [31:0] a, input [31:0] b, output [31:0] cmpOut);

2) 功能

信号	功能
a[31:0]	输入第一个数
b[31:0]	输入第二个数
cmpOut[31:0]	输出 a-b，用于 Npc 模块判断是否跳转

4. Ext

1）接口

文件名	模块接口定义
Ext.v	module Ext(input [width-1:0] in, input op, output [31:0] out); parameter width = 16;

2）功能

信号	功能
in	数据输入端， 经过扩展后由 out 输出
op	op==0，进行 0 扩展 op==1，进行 1 扩展

5. Npc

1. 接口

文件名	模块接口定义
Npc.v	module Npc(input [31:0] instr, input [2:0] pcOp, input [31:0] pcPlus4, input [31:0] cmpOut, input [31:0] ra, output reg [31:0] npc);

2. 功能

根据 pcOp 和 cmpOut 决定 npc

信号	功能
pcOp	3'b000: npc <= pcPlus4; 3'b001: npc <=(cmpOut==0)? pcPlus4 + {{14{instr[15]}},instr[15:0], 2'b00}: pcPlus4+4; 3'b010: npc <= {pcPlus4[31:28], instr[25:0], 2'b00}; 3'b011: npc <= ra; 3'b100: npc <=(cmpOut!=0)? pcPlus4 + {{14{instr[15]}},instr[15:0], 2'b00}: pcPlus4+4; default: npc <= pcPlus4;

6. ALU

1) 接口定义

文件名	模块接口定义
ALU.v	module ALU(input [31:0] a, input [31:0] b, input [3:0] aluOp, output reg [31:0] aluOut, output reg [31:0] hi_out, output reg [31:0] lo_out);

2) 功能

信号	功能
aluOp	0000 + 0001 - 0010 & 0011 | 0100 div 0101 divu 0110 mult 0111 multu 1000 << 1001 slt

7. DM

1) 接口

文件名	模块接口定义
DM.v	module DM(input [31:0] a, input [31:0] pc, input [31:0] wd, input clk,w,reset,

	output [31:0] out);

2) 功能

信号	功能
a	地址， 读出数据通过 out 输出，写入数据来自 wd
pc	输入当前 pc
clk	时钟信号
w	写使能端
reset	同步复位信号

8. Classifier

1) 接口

文件名	模块接口定义
HC.v	module Classifier(input [5:0] op, f, output calR,call,st,ld,br,jal,jr);

2) 功能： 根据输入的 op 和 f，判断该指令属于哪一类。

9. Bypass

1）接口

文件名	模块接口定义
HC.v	module Bypass(input [31:0] instrD,instrE,instrM,instrW, output [2:0] fRsD,fRtD,fRsE,fRtE,fRtM);

2）功能

根据输入的四个阶段的指令，输出旁路转发的控制信号

10. Stall

1) 接口

文件名	模块接口定义
HC.v	module Stall(input [31:0] instrD, input [31:0] instrE, input [31:0] instrM, input [31:0] instrW, output stall);

2）功能

根据输入的四个阶段的指令，判断是否要暂停

11. Controller

1) 接口定义

文件名	模块接口定义
Controller.v	module ControllerD(input [31:0] instr, output [2:0] pcOp, output exOp); module ControllerE(input [31:0] instr, output [1:0] aluASrc, output [1:0] aluBSrc, output [3:0] aluOp); module ControllerM(input [31:0] instr, output memWrite,); module ControllerW(input [31:0] instr, output regWrite, output [2:0] mem2Reg, output [1:0] regDest);

2）功能

信号	功能
pcOp	控制 pc 的跳转
exOp	控制扩展器的扩展方式
aluASrc	选择 ALU A 口的输入
aluBSrc	选择 ALU B 口的输入
aluOp	ALU 的运算方式
memWrite	DM 的写信号
mem2Reg	控制寄存器堆的输入数据来源
regDest	控制寄存器堆写入地址

二． 测试

```
lui $1, 0xffff
ori $1, 0x1234
addu $1, $1, $1
ori $2, 0x12
subu $1, $1, $2
sw $1, 0($0)
lw $2, 0($0)
addu $2, $2, $2
beq $2, $1, loop
nop
subu $5, $2, $1
loop:
jal tag_1
nop
tag_1:
nop
jr $ra
nop
```

```
$ 1 <= ffff0000
$ 1 <= ffff1234
$ 1 <= fffe2468
$ 2 <= 00000012
$ 1 <= fffe2456
*00000000 <= fffe2456
$ 2 <= fffe2456
$ 2 <= fffc48ac


$ 5 <= fffe2456

$31 <= 00003034
```

```
ori $8,$0,0xffff
ori $0,$8,0xffff
ori $9,$0,1
addu $8,$9,$8
addu $0,$8,$9
subu $8,$8,$9
sw $8,0($0)
sw $8,4($0)
sw $9,0($0)
lw $8,0($0)
lw $10,4($0)
beq $0,$0,L1
nop
L1:
beq $0,$8,L2
ori $11,$0,1
L2:
j L3
nop
lui $11,1
L3:
jal L4
addu $9,$0,$ra
L4:
addu $8,$0,$ra
jr $8
nop
```

```
$ 8 <= 0000ffff
$ 0 <= 0000ffff
$ 9 <= 00000001
$ 8 <= 00010000
$ 0 <= 00010001
$ 8 <= 0000ffff
*00000000 <=
0000ffff
*00000004 <=
0000ffff
*00000000 <=
00000001
$ 8 <= 00000001
$10 <= 0000ffff
$11 <= 00000001
$31 <= 00003050
$ 9 <= 00003050
$ 8 <= 00003050
$ 8 <= 00003050
$ 8 <= 00003050
$ 8 <= 00003050
$ 8 <= 00003050
$ 8 <= 00003050
$ 8 <= 00003050
```

三． 思考题

1） 转发类

编号	冲突类型	测试序列
1	R-M-RS	addu $1, $2, $3//转发到 ID nop beq $1, $2, start nop addu $1, $2, $3//转发到 EX subu $3, $1, $2
2	R-M-RT	addu $2, $2, $3//forward to ID nop beq $1, $2, start nop addu $1, $2, $3//forward to EX subu $3, $2, $1
3	R-W-RS	subu $1, $2, $3//forward to EX nop subu $3, $1, $2
4	R-W-RT	subu $1, $2, $3//forward to EX nop addu $3, $2, $1
5	I-M-RS	ori $2, $2, 100 nop beq $2, $3, start nop ori $1, 100 subu $3, $2, $1

6	I-M-RT	addi $2, $2, 100
		nop
		beq $3, $2, start
		nop
		
		addi $2, 100
		addu $3, $1, $2

7	I-W-RS	ori $2, $2, 100 nop addu $3, $2, $1
8	I-W-RT	lui $2, 100 nop

		addu $3, $1, $2

9	LD-W-RS	lw $2, 0($0)//forward to EX
		nop
		addu $3, $2, $0

10	LD-W-RT	lw $2, 0($0)//forward to EX
		nop
		subu $3, $0, $2
		
		lw $2, 0($0)//forward to MEM
		sw $2, 4($0)

2) 暂停类

编号	冲突类型	测试序列
1	LD-EX-RS	lw $2, 0($0) addu $3, $2, $3
2	LD-EX-RT	lw $2, 0($0) subu $4, $3, $2
3	LD-MEM-RS	lw $2, 0($0) nop jr $2 nop
4	LD-MEM-RT	lw $2, 0($0) nop beq $2, $4, start nop
5	R-EX-RS	addu $4, $2, $0 jr $4 nop
6	R-EX-RT	addu $4, $2, $0 beq $5,$4, start nop
7	I-EX-RS	addi $4, 0x3000 jr $4 nop
8	I-EX-RT	ori $4, 0xffff beq $5,$4, start nop

IF/ID 当前指令			ID/EX Tnew					EX/MEM Tnew					MEM/WB Tnew					
type	source	Tuse	calR 1/rd	calI 1/rt	ld 2/rt	jal/bgezal 0/$31	jalr 0/rd	calR 0/rd	calI 0/rt	ld 1/rt	jal/bgezal 0/$31	jalr 0/rd	calR 0/rd	calI 0/rt	ld 0/rt	jal/bgezal 0/$31	jalr	0/rd
calR	rs/rt	1			stop													
calI	rs	1			stop													
br	rs/rt	0	stop	stop	stop					stop								
ld	rs	1			stop													
st	rs	1			stop													
	rt	2																
jr/jalr	rs	0	stop	stop	stop					stop								