P5.Verilog 开发流水线 cpu

一、实验目的

使用 verilog 设计一个流水线 CPU。

二、设计要求

1.处理器使用 Verilog 设计。

2.处理器能支持 addu、subu、ori、lw、sw、beq、lui、nop、jal、jr、j、nop 指令。

3 处理器为流水线设计。

三、实验设计

一、模块设计

1.PC 模块

（1）基本描述

PC 主要功能是完成地址转移工作，当复位信号有效时，将地址变为首地址，否则维持原有地址不变，将地址传输给指令存储器（IM）。

（2）模块接口

信号名	方向	描述
inp	I	输入地址。
Clk	I	时钟信号。
Reset	I	复位信号。
outp[31:0]	O	当前的 32 位地址。
en	I	使能信号。

（3）功能定义

序号	功能名称	功能描述
1	复位	复位信号有效时，PC 被设置为 0x00003000。
2	转移地址	当复位信号为 0 时，PC 将输入的地址转移给 IM。
3	停止转移	当使能信号为 0 时，PC 值固定不变。

2.IM 模块

（1）基本描述

IM 的主要功能是根据当前的地址，将地址相对应的 32 位数据导出并进行后续操作。

（2）模块接口

信号名	方向	描述
add	I	输入当前的地址。
cod	O	输出地址所对应的 32 位数据。

（3）功能定义

序号	功能名称	功能描述
1	取指令	根据当前输入的地址，从存储器里取出对应的数据进行操作。

3.GRF 模块

（1）基本描述

GPR 主要功能是利用寄存器以实现对数据的取出和存入操作。通过一个 32 位 MIPS 指令对指令中的指定寄存器的值进行读或写操作。

（2）模块接口

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号。
Rw	I	寄存器写信号，当信号有效时，可以向寄存器里写入数据。
Rr1[25:21]	I	读入数据的寄存器地址 1。
Rr2[20:16]	I	读入数据的寄存器地址 2。
Wr[4:0]	I	写寄存器的地址。
Wd[31:0]	I	写寄存器的数据。
Rd1[31:0]	O	rr1 中数据的输出。
Rd2[31:0]	O	rr2 中数据的输出。

（3）功能定义

序号	功能名称	功能描述
1	取数	取出 RS 接口和 RT 接口对应的寄存器中存储的值。
2	写数	将指定的 32 位数据写入指定的寄存器中。

4.ALU 模块

（1）基本描述

ALU 的功能是对指定的两个 32 位数进行加、减、或运算以及对两个数进行大小比较。

（2）模块接口

信号名	方向	描述
I1	I	第一个 32 位数据的输入。
I2	I	第二个 32 位数据的输入。
Aluop[1:0]	I	alu 输出数据的控制信号。 00：输出加法结果。 01：输出减法结果。 10：输出或运算结果。
R1	O	运算输出结果。

（3）功能定义

序号	功能名称	功能描述
1	加法运算	取出 RS 接口和 RT 接口对应的寄存器中存储的值。
2	减法运算	将指定的 32 位数据写入指定的寄存器中。
3	或运算	对输入的两个数进行或运算。

5.DM 模块

（1）基本描述

DM 的功能是对数据存储器里指定的地址里的数据进行读或写操作。

（2）模块接口

信号名	方向	描述

Clk	I	时钟信号。
Reset	I	复位信号。
Addr[31:0]	I	写入或读出数据的地址。
Memw	I	写数据操作信号。
Inp[31:0]	I	写入的数据。
Outp[31:0]	O	读出的数据。

（3）功能定义

序号	功能名称	功能描述
1	写数据	将 inp 写入向 addr 指定的地址中。
2	读数据	将 addr 地址中的数据输出到 outp。

6. EXT 模块

（1）基本描述

EXT 的功能是对一个十六位二进制数进行扩展，并根据扩展信号判断进行符号扩展还是零扩展。

（2）模块接口

信号名	方向	描述
Extd[15:0]	I	需要扩展的 16 位数据。
Extop	I	控制输出的信号。
Exo[31:0]	O	扩展完毕的 32 位数据。

（3）功能定义

序号	功能名称	功能描述
1	扩展	根据扩展信号将 16 位数据变为 32 位数据。

7. npc 模块

（1）基本描述

npc 的功能是对下一次指令的地址进行计算。

（2）模块接口

信号名	方向	描述

dpc[31:0]	I	当前的地址。
Imm[25:0]	I	当前地址对应的数据的后 26 位数
Npc_sel	I	判断指令的类型。 01：beq， 10：jal， 11：jr， 00：dpc
Zero	I	判断两个操作数是否相等
Dra[31:0]	I	31 号寄存器的值
Npc[31:0]	O	下一条指令地址

（3）功能定义

序号	功能名称	功能描述
1	计算地址	根据当前指令和当前指令类型判断下一指令地址。

8. control（control0、control1、control2、control3）模块

（1）基本描述

control 模块的功能是对各部分的运算、存储、读取等进行控制。

（2）模块接口

信号名	方向	描述
Op[5:0]	I	六位 op 信号。
fun[5:0]	I	六位 func 信号。
Npcsel[1:0]	O	下一位地址输出控制。
Extop[1:0]	O	Ext 输出控制。
alusrc	O	Alu 计算数据控制。
Aluop[1:0]	O	Alu 输出控制。
memwrite	O	Dm 写控制。
memtoreg[1:0]	O	Grf 存储数据控制。
Regdst[1:0]	O	Grf 存储地址控制。
regwrite	O	Grf 写控制。

（红色：control0；绿色：control1；蓝色：control2；黄色：control3）

（3）功能定义

序号	功能名称	功能描述
1	总体控制	根据输入信号，判断需要控制的数据，从而得到我们想要的结果。

（4）控制器真值表

指令	addu	subu	ori	lw	sw	beq	lui	jal	j	jr
func	100001	100011	n/a							001000
op	000000	000000	001101	100011	101011	000100	001111	000011	000010	000000
Npcsel[1:0]	00	00	00	00	00	00	00	10	10	11
Extop[1:0]	11	11	00	10	10	11	01	11	11	11
alusrc	0	0	1	1	1	0	1	x	x	x
Aluop[1:0]	00	01	10	00	00	xx	00	xx	xx	00
memwrite	0	0	0	1	0	0	0	0	0	0
memtoreg[1:0]	00	00	00	01	00	00	00	10	00	00
Regdst[1:0]	01	01	00	00	00	00	00	10	00	00
regwrite	1	1	1	1	0	0	1	1	0	0

9. hazard 模块

（1）基本描述

由于一条指令处理多个周期，可能会造成不同数据间的冲突，hazard 的作用是处理这些冲突数据，使得流水线能够正常工作。

（2）模块接口

信号名	方向	描述
ird[31:0]	I	If/Id 寄存器中的指令
ire[31:0]	I	Id/Ex 寄存器中的指令。
irm[31:0]	I	Ex/mem 寄存器中的指令。
irw[31:0]	I	mem/wb 寄存器中的指令。
Inadd	I	寄存器写地址

Pcen	O	Pc 使能信号
irden	O	If/Id 寄存器使能信号
Ireclr	O	Id/Ex 寄存器复位信号
FORWARDRSD	O	Id 级 rs 选择信号
FORWARDRTD	O	Id 级 rt 选择信号
FORWARDRSE	O	Ex 级 rs 选择信号
FORWARDRTE	O	Ex 级 rt 选择信号
FORWARDRTM	O	Mem 级 rt 选择信号

（3）功能定义

序号	功能名称	功能描述
1	解决冲突	解决产生冲突的数据。

（4）暂停、转发机制构建

暂停机制构建表

if/id当前指令			tnew									备注指令	
			id/ex			ex/mem			mem/wb				
指令类型	源寄存器	tuse	cal-r/1/rd	cal-i/1rt	load/2/rt	cal-r/0/rd	cal-i/0/rt	load/1/rt	cal-r/0	cal-i/0	load/0		
beq	rs/rt	0	暂停	暂停	暂停			暂停					
cal-r	rs/rt	1			暂停							addu	subu
cal-i	rs	1			暂停							lui	ori
load	rs	1			暂停								
store	rs	1			暂停								
store	rt	2											
jr	rs	0	暂停	暂停	暂停			暂停					

转发机制构建表

流水级	源寄存器	涉及指令	转发mux	控制信号	输入0	tnew				
						ex/mem		mem/wb		
						cal-r/0/rd	cal-i/0/rt	cal-r/0/rd	cal-i/0/rt	load/0
ird	rs	beq,jr	MFRSD	FORWARDRSD	RF.RD1	AO	AO	M4	M4	M4
	rt	beq	MFRTD	FORWARDRTD	RF.RD2	AO	AO	M4	M4	M4
ire	rs	cal-r,cal-i,ld,st	MFRSE	FORWARDRSE	RS@E	AO	AO	M4	M4	M4
	rt	cal-r，st	MFRTE	FORWARDRTE	RT@E	AO	AO	M4	M4	M4
irm	rt	st	MFRTM	FORWARDRTM	RT@M			M4	M4	M4

二、模块间连接

```
module mips(clk,reset);
    input clk//时钟信号;
    input reset//复位信号;
    wire clk,reset;
    wire [31:0] p1,p2,p3,p4,p5,p6;
    wire d1;
    wire [31:0] r1,r2,r3,r4;
    wire [31:0] re1,re2;
    wire [31:0] se1,se2,se3,se4;
    wire [31:0] t1,t2,t3,t4,t5;
    wire [31:0] ts1;
    wire [4:0] a1;
    wire [31:0] a2;
    wire [31:0] u1,u2,u3,u4,u5;
    wire [31:0] s1,s2,s3,s4;
    wire [31:0] m1,m2,m3,m4;
    wire zero;
    wire en,en2,clr;
    wire [1:0] h1,h2,h3,h4;
    wire h5;
    wire [31:0] b1,b2,b3,b4,b5;
    wire [1:0]regdst,npc_sel,extop,aluop,memtoreg;
    wire alusrc,regwrite,memwrite;
    assign zero = (b1==b2)?1:0;
    PC
QPC(.clk(clk),.reset(reset),.inputadd(p6),.outputadd(p1),.en(en));
    IM QIM(.imadd(p1[11:2]),.cod(p3));
    add4 qadd4(.inputadd(p1),.outputadd(p4));
    add4 qqadd4(.inputadd(p4),.outputadd(p5));
    reg1
qreg1(.clk(clk),.reset(reset),.pcd(p1),.pce(m1),.ird(p3),.pc4d(p4),.ire(r1),.pc4e(r2),.pc8d(p5),.pc8e(r3),.en(en2));
    GRF
QGRF(.inputadd1(r1[25:21]),.address(m4),.inputadd2(r1[20:16]),.outputdata1(re1),.outputdata2(re2),.regwrite(regwrite),.clk(clk),.reset(reset),.writeadd(a1),.writedata(a2));
    EXT QEXT(.inputdata(r1[15:0]),.extop(extop),.outputdata(r4));
```

```
    NPC
QNPC(.pc(r2),.npc_sel(npc_sel),.ra(b1),.npc(p2),.index(r1[25:0]),.zero(zero));
    reg2
qreg2(.clk(clk),.reset(reset||clr),.pce(m1),.pcm(m2),.ire(r1),.pc4e(r2),.rse(re1),.rte(re2),.exte(r4),.irm(s1),.pc4m(s2),.rsm(se1),.rtm(se2),.extm(s4),.pc8e(r3),.pc8m(s3));
    M1 QM1(.A(b4),.B(s4),.sel(alusrc),.C(se3));
    ALU QALU(.indata1(b3),.indata2(se3),.aluop(aluop),.outdata(se4));
    reg3
qreg3(.clk(clk),.reset(reset),.pcm(m2),.pcw(m3),.irm(s1),.pc4m(s2),.aom(se4),.rtm(b4),.irw(t3),.pc4w(t4),.aow(t1),.rtw(t2),.pc8m(s3),.pc8w(t5));
    DM
QDM(.address(t1),.add(m3),.inputdata(b5),.clk(clk),.reset(reset),.memwrite(memwrite),.outputdata(ts1));
    reg4
qreg4(.clk(clk),.reset(reset),.pcw(m3),.pcd(m4),.irw(t3),.ird(u3),.pc4w(t4),.pc4d(u4),.aow(t1),.aod(u1),.drw(ts1),.drd(u2),.pc8w(t5),.pc8d(u5));
    M2 QM2(.A(u1),.B(u2),.C(u5),.sel(memtoreg),.D(a2));
    M3
QM3(.A(u3[20:16]),.B(u3[15:11]),.C(5'b11111),.sel(regdst),.D(a1));
    hazard
qhazard(.ird(r1),.ire(s1),.irm(t3),.irw(u3),.pcen(en),.irden(en2),.ireclr(clr),.FORWARDRSD(h1),.FORWARDRTD(h2),.FORWARDRSE(h3),.FORWARDRTE(h4),.FORWARDRTM(h5),.inadd(a1));
    M2 MRSD(.A(re1),.B(t1),.C(a2),.sel(h1),.D(b1));
    M2  MRTD(.A(re2),.B(t1),.C(a2),.sel(h2),.D(b2));
    M2 MRSE(.A(se1),.B(t1),.C(a2),.sel(h3),.D(b3));
    M2 MRTE(.A(se2),.B(t1),.C(a2),.sel(h4),.D(b4));
    M1 MRTM(.A(t2),.B(a2),.sel(h5),.C(b5));
    M1 pcM1(.A(p4),.B(p2),.sel(d1),.C(p6));
    control0
qcontrol0(.op(r1[31:26]),.fun(r1[5:0]),.npc_sel(npc_sel),.extop(extop),.mu(d1));
    control1
qcontrol1(.op(s1[31:26]),.fun(s1[5:0]),.alusrc(alusrc),.aluop(aluop));
```

```
    control2
qcontrol2(.op(t3[31:26]),.fun(t3[5:0]),.memwrite(memwrite));
    control3
qcontrol3(.op(u3[31:26]),.fun(u3[5:0]),.regwrite(regwrite),.memtoreg(
memtoreg),.regdst(regdst));
endmodule
```

四、Mips 程序测试

```
lui $s0,0x3246
ori $s0,$s0,0xff3a
lui $s1,0x1889
ori $s1,$s1,0xde51
addu $s2,$s1,$s0
sw $s2,0($t0)
sw $s2,0($t0)
sw $s2,0($t0)
sw $s2,0($t0)
sw $s2,0($t0)
lui $s0,0x2345
ori $s0,$s0,0x1111
lui $s1,0x0321
ori $s2,$s1,0x6666
lui $s3,0x2344
ori $s3,$s3,0x6dea
lui $s4,0xaed1
subu $s1,$s2,$s3
addu $s4,$s1,$s2
subu $s7,$s2,$s3
addu $s3,$s1,$s3
addu $s5,$s2,$s1
beq $s5,$s7,loop
subu $s1,$s2,$s3
addu $s7,$s1,$s3
addu $s5,$s2,$s1
beq $s5,$s7,loop
subu $s1,$s2,$s3
lui $t1,0x8888
```

```
addu $s6,$s1,$s2
subu $s1,$s2,$s3
lw $s1,0($t7)
beq $s1,$s3,loop2
lui $t1,0xac05
lui $t2,0x5388
ori $t1,$t1,0x2333
beq $t1,$t2,loop2
addu $s7,$s2,$s1
ori $s1,$s2,0x3245
addu $t0,$s1,$s2
ori $s3,$t0,0x0003
beq $s4,$s3,loop2
addu $t2,$s1,$s3
lw $s1,0($t7)
addu $t5,$s1,$s3
beq $s1,$s3,loop2
ori $t3,$s2,0x3946
addu $t0,$s1,$t3
addu $t2,$s4,$t3
lui $t1,0x8888
j loop
sw $s0,0($t7)
sw $s1,4($t7)
sw $s2,8($t7)
loop2:
sw $s3,12($t7)
lw $s1,0($t7)
addu $t6,$s1,$t0
ori $s2,$s1,0x2453
lw $s2,0($t7)
ori $s2,$zero,0x0004
addu $t7,$t8,$s2
sw $t6,0($t7)
jr $ra
loop:
sw $k0,100($a0)
lw $s4,4($k0)
```

```
jal loo
addu $s6,$s5,$s4
jal end
loo:
lw $s2,0($t7)
addu $s2,$s2,$k0
ori $s4,$s1,0x0004
subu $t9,$s4,$s2
beq $s2,$t9,loop
beq $zero,$zero,loop2
nop
end:
sw $s4,8($t2)
lw $s4,4($t2)
addu $s4,$s4,$s1
subu $s4,$s4,$s3
```

机器码：3c103246 3610ff3a 3c111889 3631de51 02309021 ad120000 ad120000 ad120000 ad120000 ad120000 3c102345 36101111 3c110321 36326666 3c132344 36736dea 3c14aed1 02538823 0232a021 0253b823 02339821 0251a821 12b70028 02538823 0233b821 0251a821 12b70024 02538823 3c098888 0232b021 02538823 8df10000 12330015 3c09ac05 3c0a5388 35292333 112a0011 0251b821 36513245 02324021 35130003 1293000c 02335021 8df10000 02336821 12330008 364b3946 022b4021 028b5021 3c098888 08000c35 adf00000 adf10004 adf20008 adf3000c 8df10000 02287021 36322453 8df20000 34120004 03127821 adee0000 03e00008 ac9a0064 8f540004 0c000c3a 02b4b021 0c000c41 8df20000 025a9021 36340004 0292c823 1259fff6 1000ffec 03325823 00000000 ad540008 8d540004 0291a021 0293a023

预期结果：

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x4ad0dd8b
$t7	15	0x00000004
$s0	16	0x23451111
$s1	17	0x4ad0dd8b
$s2	18	0x4ad0dd8b
$s3	19	0x03216666
$s4	20	0x928054b0
$s5	21	0x03216666
$s6	22	0x03216666
$s7	23	0x03216666
$t8	24	0x00000000
$t9	25	0xb52f2279
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00002ffc
$fp	30	0x00000000
$ra	31	0x00003114

Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x4ad0dd8b	0x4ad0dd8b	0x00000004	0x03216666
0x00000000	0x00000000	0x00000000	0x00000000

实际结果：

00000000	00000000
00000000	00000000
03216666	00000004
4AD0DD8B	4AD0DD8B

00003114	00000000
00000000	00000000
00000000	00000000
B52F2279	00000000
03216666	03216666
03216666	928054B0
03216666	4AD0DD8B
4AD0DD8B	23451111
00000004	4AD0DD8B
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000

五、思考题

1. 在本实验中你遇到了哪些不同指令组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？请有条理的罗列出来。

暂停解决：

Beq 与 R 类型指令在 Ex 上产生的冲突：addu $s5,$s2,$s1　beq $s5,$s7,loop

Beq 与 I 类型指令在 Ex 上产生的冲突：ori $s3,$s2,0x0003 beq $s4,$s3,loop2

Beq 与 lw 类型指令在 Ex 上产生的冲突：lw $s1,0($t7)

beq $s1,$s3,loop2

Beq 与 lw 类型指令在 mem 上产生的冲突：lw $s1,0($t7)

addu $t5,$s1,$s3

beq $s1,$s3,loop2

R 指令与 lw 类型指令在 Ex 上产生的冲突：subu $s1,$s2,$s3

lw $s1,0($t7)

I 指令与 lw 类型指令在 Ex 上产生的冲突：ori $s2,$s1,0x2453

lw $s2,0($t7)

sw 指令与 lw 指令在 Ex 上产生的冲突：sw $k0,100($a0)

lw $s4,4($k0)

转发解决：

Beq 与 R 类型指令在 mem 上产生的冲突 rs：subu $s1,$s2,$s3

lw $s1,0($t7)

beq $s1,$s3,loop2

Beq 与 I 类型指令在 mem 上产生的冲突 rs：lui $t1,0x5388

ori $t1,$t1,0x2333

beq $t1,$t2,loop

Beq 与 R 类型指令在 wb 上产生的冲突 rt：subu $s7,$s2,$s3

addu $s3,$s1,$s3

addu $s5,$s2,$s1

beq $s5,$s7,loop

Beq 与 I 类型指令在 wb 上产生的冲突 rt：lui $t1,0xac05

lui $t2,0x5388

ori $t1,$t1,0x2333

beq $t1,$t2,loop2

Beq 与 lw 指令在 wb 上产生的冲突 rs：lw $s2,0($t7)

ori $s4,$s1,0x0004

subu $t9,$s4,$s2

beq $s2,$t9,loop

R 类型指令与 R 类型指令在 mem 上的冲突 rs：subu $s1,$s2,$s3

addu $s7,$s1,$s3

R 类型指令与 i 类型指令在 mem 上的冲突 rt：addu $t0,$s1,$s2

ori $s3,$t0,0x0003

R 与 R 在 wb 上的冲突 rt：subu $s1,$s2,$s3

addu $s7,$s1,$s3

addu $s5,$s2,$s1

R 与 I 在 wb 上的冲突 rt：addu $s2,$s2,$k0

ori $s4,$s1,0x0004

subu $t9,$s4,$s2

sw 指令与 R 类型指令在 wb 上产生的冲突：addu $t7,$t8,$s2

sw $t6,0($t7)

addu $t6,$s1,$t0

ori $s2,$s1,0x2453