实验报告

设计

概述

用verilog实现的五级流水线MIPS-CPU，支持add,sub,ori,lw,sw,beq,lui,j,jal,jr,nop指令，

实现包括了IFU，GRF，ALU，DM，EXT，Controller，流水线寄存器模块，冒险模块。

模块

IFU

包括两个子模块，PC，IM

功能：

同步复位，收到复位信号是，PC设置为0x00003000

取指，从IM中取出指令

计算下一条指令

F2D

功能：

F/D级流水线寄存器

GRF

功能：

复位，收到复位信号时寄存器设置为0x00000000

读寄存器，根据地址读出数据

写寄存器，根据地址写入寄存器

CMP

比较两个数据是否相等

EXT

进行符号拓展操作

D2E

D/E级流水线寄存器

ALU

功能：

提供加减与或，结果是否为零，大小比较功能

E2M

E/M级流水线寄存器

DM

功能：

复位，复位信号有效时将所有数据设置为0x0

读，根据寄存器地址读数据

写，根据地址写入数据

M2W

M/W级流水线寄存器

Controller

控制单元，产生控制信号

冒险单元

暂停控制表：

IF/ID当前指令			ID/EX当前指令			EX/MEM
指令类型	源寄存器	Tuse	Tnew			
			cal_r/rd	cal_i/rt	load/rt	load/rt
			1	1	2	1
beq	rs/rt	0	stall	stall	stall	stall
cal_r	rs/rt	1			stall	
cal_i	rs	1			stall	
load	rs	1			stall	
store	rs	1			stall	
store	rt	2				
jr	rs	0	stall	stall	stall	stall

转发控制表:

						E	M			W			
要转发的端	寄存器	指令	MUX	选择信号	默认输出	jal 31	cal_r rd	cal_i rt	jal 31	cal_r rd	cal_i rt	load rt	jal 31
D	rs	cal_r cal_i beq load store jr	forRsD	selRsD	RD1	pc_E8	aluRet_M	aluRet_M	pc_M8	WD	WD	WD	pc_W8
	rt	cal_r store beq	forRtD	selRtD	RD2	pc_E8	aluRet_M	aluRet_M	pc_M8	WD	WD	WD	pc_W8
E	rs	cal_r cal_i load store	forRsE	selRsE	grf_RD1		aluRet_M	aluRet_M	pc_M8	WD	WD	WD	pc_W8
	rt	cal_r store	forRtE	selRtE	grf_RD2		aluRet_M	aluRet_M	pc_M8	WD	WD	WD	pc_W8
M	rt	store	forRtM	selRtM	rt_M					WD	WD	WD	pc_W8

测试方案

使用code.txt进行测试，并对比MARS的数据

code:

```
341c0000
341d0000
34011010
3c028723
34037856
3c0485ff
34050001
3c06ffff
3407ffff
00220820
00234820
```

```
00224022
00e00022
13910003
00000000
10000015
00000000
10220013
00000000
3402000c
00000000
00000000
00000000
0c000c1b
ac410000
1000000b
00220820
00220820
00220820
00220820
ac5f0000
8c410000
00000000
00000000
00000000
00200008
ac5f0000
1000ffff
00000000
```

思考题

1.分支预测失败时会浪费WB之前的流水线级数，会损失若干个时钟周期。

2.在D级决定跳转时，下一条指令已经进入流水线，所以跳转指令的下一条必须执行。充分利用流水线性能

3.可能会出现修改过的寄存器数值在后续指令中使用未修改过的原始数据。

4.防止W级还未写入的数据在之后的指令中使用。

5.供给者通常是储存上一级数据的流水线寄存器，而需求者是下一级流水线的操作。

6.修改控制单元，并实现特定指令

7.根据每个指令的opcode，funcode和对应的控制信号关系列出真值表，在always@(*)中使用case语句进行匹配。

端口名字\输出信号	addu	subu	jr	lw	sw	beq	lui	ori	jal	j
op	000000	000000	000000	100011	101011	000100	001111	001101	000011	000010
func	100001	100011	001000	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx
regDst	01	01	01	00	xx	00	00	00	10	xx
aluSrc	0	0	0	1	1	0	x	1	x	x
regWrite	1	1	0	1	0	0	1	1	1	0
memRead	0	0	0	1	0	0	0	0	0	0
memWrite	0	0	0	0	1	0	0	0	0	0
memToReg[1:0]	00	00	11	01	xx	xx	10	00	xx	xx
extOp[1:0]	00	00	00	00	00	00	10	01	00	0
branch	0	0	0	0	0	1	0	0	0	0
aluCtrl[2:0]	010	011	xxx	010	010	011	xxx	001	xxx	xxx
jump	0	0	0	0	0	0	0	0	1	1
pcSrc	0	0	1	0	0	0	0	0	1	1