P6-Verilog 流水线 CPU 设计文档

一、CPU 设计方案综述

（一）总体设计概述

本 CPU 为 Verilog 实现的 MIPS 流水线处理器，支持的指令集包含{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}。为了实现这些功能，CPU 的顶层模块为 mips，包含了各个流水级和流水寄存器以及暂停控制器和转发控制器。

（二）F_Stage

包含了 F 级中所应有的所有模块，有 PC、Adder、MUX_NPC、IM。

其中 F_Stage 对外接口为

表 1 F_Stage 端口说明

序号	信号	方向	描述
1	clk	I	时钟信号
2	reset	I	同步复位信号，将 PC 的值设置为 0x00003000 1：复位 0：无效
3	NPCOp_D [1:0]	I	进入 PC 的 NPC 选择 00：计算顺序地址（PC+4） 01：计算 beq 地址 10：计算 jal 地址 11：计算 jr 地址

4	NPC_D [31:0]	I	从 D 级中的 NPC 模块计算得出的地址
5	PCEn	I	PC 的写使能： 1：写入 NPC 0：不写入 NPC
6	PC_F_D	O	从 F 级传到 D 级的 PC 值
7	PC4_F_D	O	从 F 级传到 D 级的 PC+4 的值
8	Instr_F_D	O	从 F 级传到 D 级的指令

1. PC

（1）端口说明

表 2 PC 端口说明

序号	信号	方向	描述
1	clk	I	时钟信号
2	reset	I	同步复位信号，将 PC 的值设置为 0x00003000 1：复位 0：无效
3	En	I	PC 的写使能： 1：写入 NPC 0：不写入 NPC
4	NPC [31:0]	I	NPC 的值
5	PC [31:0]	I	输出当前的 PC 值

（2）功能定义

表 3 PC 功能定义

序号	功能	描述
1	复位	Reset 有效时，将 PC 设置为 0x00003000
2	写入	reset 无效且 PCEn 有效时写入 NPC
5	输出 PC	输出 PC 值

2. Adder

（1）端口说明

表 4 Adder 端口说明

序号	信号	方向	描述
1	PC [31:0]	I	PC 的值
2	PC4 [31:0]	O	输出 PC+4

（2）功能定义

表 5 Adder 功能定义

序号	功能	描述
1	输出 PC+4	输出 PC+4 的值

3. IM

（1）端口说明

表 6 IM 端口说明

序号	信号	方向	描述
1	address	I	以 PC 作为地址
2	instr	O	输出该地址中的指令

（三）D_Stage

包含了 D 级中所应有的所有模块，有 Controller_D、Controller_W、NPC、CMP、EXT、RF、MUXA3_D、MUXWD_D。

其中 D_Stage 对外接口为

表 7 D_Stage 端口说明

序号	信号	方向	描述
1	clk	I	时钟信号
2	reset	I	同步复位信号，将 PC 的值设置为 0x00003000 1：复位

			0：无效
3	Instr_D [31:0]	I	从 F/D 级流水线寄存器传过来的指令值
4	PC_D [31:0]	I	从 F/D 级流水线寄存器传过来的 PC 值
5	PC4_D [31:0]	I	从 F/D 级流水线寄存器传过来的 PC+4 值
6	Forward_A_D	I	转发至 D 级的 A 口的值
7	Forward_B_D	I	转发至 D 级的 B 口的值
8	PC_W	I	从 M/W 级流水线寄存器传过来的 PC 值
9	A3_W	I	从 M/W 级流水线寄存器传过来的 A3 值
10	WD_W	I	从 M/W 级流水线寄存器传过来的 WD 值
11	Instr_W	I	从 M/W 级流水线寄存器传过来的指令值
12	RD1_D_E	O	RF 读出的第一个数据
13	RD2_D_E	O	RF 读出的第二个数据
14	imm32_D_E	O	传至 D/E 流水线寄存器的扩展后的立即数
15	A3_D_E	O	传至 D/E 流水线寄存器的 A3
16	WD_D_E	O	传至 D/E 流水线寄存器的 WD
17	NPCOp_D	O	在 D 级生成的 NPCOP
18	NPC_D	O	在 D 级生成的 NPC
19	TuseA	O	生成的 TuseA
20	TuseB	O	生成的 TuseB
21	Tnew_D_E	O	生成的 Tnew，将传至 D/E 流水线寄存器

1. NPC

（1）端口说明

表 3 GRF 端口说明

序号	信号	方向	描述
1	RA[31:0]	I	时钟信号
2	imm26 [25:0]	I	异步复位信号，将 32 个寄存器中的值全部置为 0 1：复位

			0：无效
3	PC [31:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个，将其中存储的数据读出到 RD1
4	NPCOp [1:0]	I	NPC 选择 00：计算顺序地址（PC+4） 01：计算 beq 地址 10：计算 jal 地址 11：计算 jr 地址
5	Equal	I	CMP 得到是否相等
6	NPC [31:0]	O	下一个 PC 的值

（2）功能定义

表 4 NPC 功能定义

序号	功能	描述
1	复位	reset 有效时，将 32 个寄存器中的值全部置为 0
2	计算 NPC	00：计算顺序地址（PC+4） 01：计算 beq 地址 10：计算 jal 地址 11：计算 jr 地址

2. RF

（1）端口说明

表 3 GRF 端口说明

序号	信号	方向	描述
1	clk	I	时钟信号
2	reset	I	同步复位信号，将 32 个寄存器中的值全部置为 0 1：复位 0：无效
3	A1 [4:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个，将其中

			存储的数据读出到 RD1
4	A2 [4:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个，将其中存储的数据读出到 RD2
5	A3 [4:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个作为写入的目标寄存器
6	WD [31:0]	I	32 位写入数据
7	WE	I	写使能信号 1：可向 GRF 中写入数据 0：不能向 GRF 中写入数据
8	RD1 [31:0]	O	输出 A1 指定的寄存器的 32 位数据
9	RD2 [31:0]	O	输出 A2 指定的寄存器的 32 位数据
10	PC [31:0]	I	当前 PC 的值

（2）功能定义

表 4 GRF 功能定义

序号	功能	描述
1	复位	reset 有效时，将 32 个寄存器中的值全部置为 0
2	读数据	读出 A1，A2 地址对应寄存器中所存储数据到 RD1，RD2
3	写数据	若 WE 有效且时钟上升沿来临时，将 WD 写入 A3 所对应的寄存器中

3. EXT

（1）端口说明

表 9 EXT 端口说明

序号	信号	方向	描述
1	imm16 [15:0]	I	16 位待扩展数据
2	imm32 [31:0]	O	32 位扩展后的数据
3	EXTOp [1:0]	I	EXT 功能选择 00：进行无符号扩展 01：进行符号扩展

			10：左移 16 位

（2）功能定义

表 10 EXT 功能定义

序号	功能	描述
1	无符号扩展	将 16 位立即数 imm16 无符号扩展至 32 位输出至 imm32
2	符号扩展	将 16 位立即数 imm16 符号扩展至 32 位输出至 imm32
3	左移 16 位	将 16 位立即数 imm16 低位添加 16 个 0 后输出（imm32=imm16 ‖ 0^{16}）

4. CMP

（1）端口说明

表 9 EXT 端口说明

序号	信号	方向	描述
1	A [31:0]	I	16 位待扩展数据
2	B [31:0]	I	32 位扩展后的数据
3	Equal	O	输出 A 与 B 是否相等，相等为 1，不相等为 0

（2）功能定义

表 10 EXT 功能定义

序号	功能	描述
1	判断相等	输出 A 与 B 是否相等，相等为 1，不相等为 0

（四）E_Stage

包含了 E 级中所应有的所有模块，有 Controller_E、ALU、MUXA3_E、MUXWD_E。

其中 E_Stage 对外接口为

表 7 D_Stage 端口说明

序号	信号	方向	描述
1	Instr_E [31:0]	I	从 D/E 级流水线寄存器传过来的指令值
2	Forward_A_E	I	转发至 E 级的 A 口的值

3	Forward_B_E	I	转发至 E 级的 B 口的值
4	ALUOut_E_M	O	传至 E/M 流水线寄存器的 ALUOut
5	RD2_E_M	O	传至 E/M 流水线寄存器的 RD2
6	WD_E_M	O	传至 E/M 流水线寄存器的 WD
7	imm32_E	I	从 D/E 流水线寄存器传过来的扩展后的立即数
8	A3_E_M	O	传至 E/M 流水线寄存器的 A3
9	Tnew_ E	I	D/E 流水线寄存器传过来的 Tnew
10	Tnew_ E_M	O	传给 E_M 流水线寄存器的 Tnew
11	WD_E	I	从 D/E 流水线寄存器传过来的 WD

1. ALU

（1）端口说明

表 5 ALU 端口说明

序号	信号	方向	描述
1	A [31:0]	I	参与 ALU 计算的第一个 32 位数据
2	B [31:0]	I	参与 ALU 计算的第二个 32 位数据
3	ALUOp [1:0]	I	ALU 功能选择 00：ALU 进行加法运算 01：ALU 进行减法运算 10：ALU 进行或运算
4	C [31:0]	O	ALU 的计算结果
5	Zero	O	A 与 B 是否相等 0：两者不相等 1：两者相等

（2）功能定义

表 6 ALU 功能定义

序号	功能	描述
1	加法运算	C = A + B

2	减法运算	C = A − B
3	或运算	C = A | B
4	判断相等	Zero =（A==B）

（五）M_Stage

包含了 E 级中所应有的所有模块，有 Controller_M、DM、MUXA3_M、MUXWD_M。

其中 M_Stage 对外接口为

表 7 D_Stage 端口说明

序号	信号	方向	描述
1	Instr_M [31:0]	I	从 E/M 级流水线寄存器传过来的指令值
2	Forward_DMWD_M	I	转发至 M 级的 DMWD 口的值
3	ALUOut_M	I	从 E/M 级流水线寄存器传过来的 ALUOut
4	WD_M	I	从 E/M 级流水线寄存器传过来的 WD
5	WD_M_W	O	传至 M/W 流水线寄存器的 WD
6	A3_M_W	O	传至 M/W 流水线寄存器的 A3
7	PC_M	I	E/M 流水线寄存器传过来的 PC 值
8	clk	I	时钟信号
9	reset	I	异步复位信号，将 32 个寄存器中的值全部置为 0 1：复位 0：无效

1. DM

（1）端口说明

表 7 DM 端口说明

序号	信号	方向	描述
1	clk	I	时钟信号
2	reset	I	异步复位信号，将 DM 的值全部清零

			1：复位 0：无效
3	WE	I	写使能信号 1：可向 DM 中写入数据 0：无效
4	A [31:0]	I	32 位地址输入信号，对 DM 指定地址进行读写操作
5	WD [31:0]	I	32 位写入数据
6	RD [31:0]	O	32 位输出数据
7	SSel [1:0]	I	控制 store 时位宽的信号 00：原 32 位数据 01：8 位数据 10：16 位数据
8	LSel [1:0]	I	控制 load 时位宽的信号 00：原 32 位数据 01：8 位数据 10：16 位数据
9	PC [31:0]	I	此时的 PC 值

（2）功能定义

表 8 DM 功能定义

序号	功能	描述
1	复位	Reset 有效时，将 DM 的值全部清零
2	读操作	将 A 地址对应的数据输出至 RD
3	写操作	当时钟上升沿来临时，若写使能信号 WE 有效，将 WD 数据写入 A 地址对应位置

（六）Controller

采用分布式译码，但是仅写一个控制器，在每一级实例化一次，对该级的指令进行译码。

1. 控制器设计

参考高小鹏老师的 PPT

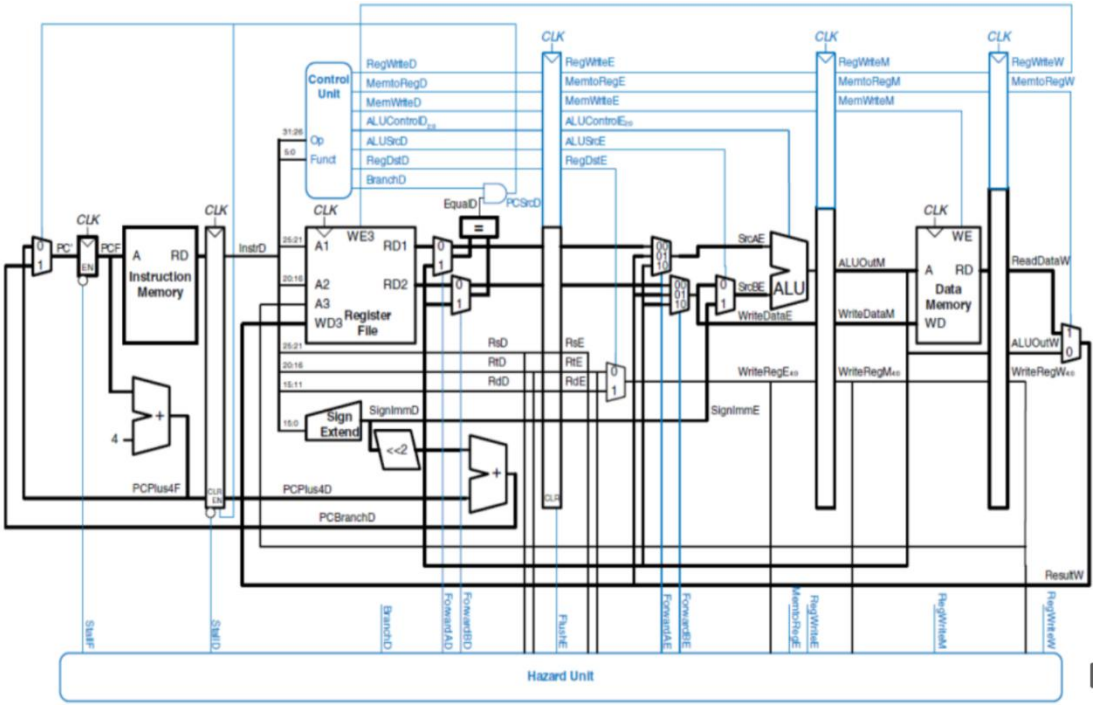

图 1 参考 CPU 电路图

（1）端口说明

表 11 Controller 端口说明

序号	信号	方向	描述
1	Instr [31:0]	I	指令
3	NPCOp [1:0]	O	00：计算顺序地址（PC+4） 01：计算 beq 地址 10：计算 jal 地址 11：计算 jr 地址
4	RFWr	O	0：GRF 写使能无效 1：GRF 写使能有效
5	EXTOp [1:0]	O	00：进行无符号扩展 01：进行符号扩展 10：左移 16 位
6	ALUOp [3:0]	O	0000：ALU 进行加法运算

			0001：ALU 进行减法运算 0010：ALU 进行或运算 0011：ALU 进行逻辑左移运算 0100：小于置 1（有符号） 0101：小于置 1（无符号） 0110：与 0111：或非 1000：异或 1001：算数右移 1010：逻辑右移
7	DMWr	O	0：DM 写使能无效 1：DM 写使能有效
8	WRSel [1:0]	O	00：将 rd 作为 GRF 的 A3 写入地址 01：将 rt 作为 GRF 的 A3 写入地址 10：将 31 作为 GRF 的 A3 写入地址
9	WDSel [1:0]	O	00：将 ALU 计算结果写入 GRF 01：将 DM 输出值写入 GRF 10：将 PC+4 写入 GRF 11：将 HI 或 LO 写入 GRF
	start	O	乘除法启动信号
	isMD	O	是否是涉及到乘除法部件的指令
	MDUOp [1:0]	O	乘除法选项： 00：mult 01：multu 10：div 11：divu
	HIWr	O	高位寄存器的写使能
	LOWr	O	低位寄存器的写使能
	HILOSel	O	选择输出 HI 还是 LO 寄存器的值

	CMPOp	O	B 类指令跳转判断： 000：beq 001：bne 010：blez 011：bgtz 100：bltz 101：bgez
	ASel	O	0：将 RD1 作为参与 ALU 计算的第一个数据 1：将 s 作为 ALU 计算的第一个数据
10	BSel	O	0：将 RD2 作为参与 ALU 计算的第二个数据 1：将 EXT 的输出值作为 ALU 计算的第二个数据
11	SSel	O	00：将 WD 写入 DM 01：将 WD 的 0~7 位写入 DM 10：将 WD 的 0~15 位写入 DM
12	LSel	O	000：将对应地址中数据输出 001：将对应地址中的 8 位数据符号扩展后输出 010：将对应地址中的 16 位数据符号扩展后输出 101：将对应地址中的 8 位数据无符号扩展后输出 110：将对应地址中的 16 位数据无符号扩展后输出

（2）Controller 模块真值表

见 excel

（七）转发器

把所有供给者的数据全收集至转发器中，再通过比较 A3 的值和 T 的值将数据转发给需求者。

端口：

序号	信号	方向	描述
1	A1_D [4:0]	I	第一个端口在 D 级的地址
2	A2_D [4:0]	I	第二个端口在 D 级的地址

3	RD1_D [31:0]	I	第一个端口在 D 级输出的原始数据
4	RD2_D [31:0]	I	第二个端口在 D 级输出的原始数据
5	A1_E [4:0]	I	第一个端口在 E 级的地址
6	A2_E [4:0]	I	第二个端口在 E 级的地址
7	RD1_E [31:0]	I	第一个端口在 E 级输出的原始数据
8	RD2_E [31:0]	I	第二个端口在 E 级输出的原始数据
9	A3_E [4:0]	I	写入地址在 E 级的地址
10	WD_E [31:0]	I	在 E 级得到的写入的数据
11	A3_M [4:0]	I	写入地址在 M 级的地址
12	WD_M [31:0]	I	在 M 级得到的写入的数据
13	A2_M [4:0]	I	第二个端口在 M 级的地址
14	RD2_M [31:0]	I	第二个端口在 M 级输出的原始数据
15	Forward_A_D [31:0]	O	转发至 D 级第一个端口的数据
16	Forward_B_D [31:0]	O	转发至 D 级第二个端口的数据
17	Forward_A_E [31:0]	O	转发至 E 级第一个端口的数据
18	Forward_B_E [31:0]	O	转发至 E 级第二个端口的数据
19	Forward_DMWD_M [31:0]	O	转发至 M 级 DMWD 端口的数据
20	Tnew_E	I	Tnew 在 E 级的值
21	Tnew_M	I	Tnew 在 M 级的值

（八）暂停控制器

通过分析 Tuse 是否小于 Tnew 和地址是否相等，得到暂停的信号

端口：

序号	信号	方向	描述
1	Instr_D [31:0]	I	D 级指令
2	A3_E [4:0]	I	写入地址在 E 级的地址
3	A3_M [4:0]	I	写入地址在 M 级的地址
4	Tnew_E	I	Tnew 在 E 级的值
5	Tnew_M		Tnew 在 M 级的值

6	TuseA	I	TuseA
7	TuseB	I	TuseB
8	PCEn	O	暂停 F 级
9	F_D_RegEn	O	暂停 D 级
10	Flush_E	O	清除 E 级

（八）乘除法器

端口：

序号	信号	方向	描述
1	clk	I	时钟信号
2	reset	I	复位信号
3	start	I	开始乘除运算信号
4	MDUOp [1:0]	I	乘除法选择信号
5	HIWr		HI 寄存器的写使能
6	LOWr	I	LO 寄存器的写使能
7	A	I	运算数 A
8	B	I	运算数 B
9	busy	O	正在运算信号
10	HI	O	HI 寄存器的值
11	LO	O	LO 寄存器的值

满足下面行为：

1、自 Start 信号有效后的第 1 个 clock 上升沿开始，乘除部件开始执行运算，同时 Busy 置位为 1。

2、在运算结果保存到 HI 和 LO 后，Busy 位清除为 0。

3、当 Busy 或 Start 信号为 1 时，mfhi、mflo、mthi、mtlo、mult、multu、div、divu 均被阻塞，即被阻塞在 IF/ID。

4、数据写入 HI 或 LO，均只需 1 个 cycle。

二、测试方案

（一）典型测试样例

1. ALU 功能测试

使用下面代码进行测试，该代码使用到了或运算、加法运算、减法运算、判零运算。

```
.text
    ori $1,$0,1
    l:
    addu $1,$1,$1
    subu $2,$1,$0
    sw $2,4($0)
    lw $3,4($0)
    beq $2,$3,l
```

图 3 ALU 测试代码

使用下面代码进行测试，该代码使用到了或运算、加法运算、减法运算、判零运算。

测试结果如图所示：

```
@00003000: $ 1 <= 00000001
@00003004: $ 1 <= 00000002
@00003008: $ 2 <= 00000002
@0000300c: *00000004 <= 00000002
@00003010: $ 3 <= 00000002
@00003004: $ 1 <= 00000004
@00003008: $ 2 <= 00000004
@0000300c: *00000004 <= 00000004
@00003010: $ 3 <= 00000004
@00003004: $ 1 <= 00000008
@00003008: $ 2 <= 00000008
@0000300c: *00000004 <= 00000008
@00003010: $ 3 <= 00000008
@00003004: $ 1 <= 00000010
@00003008: $ 2 <= 00000010
@0000300c: *00000004 <= 00000010
@00003010: $ 3 <= 00000010
@00003004: $ 1 <= 00000020
@00003008: $ 2 <= 00000020
@0000300c: *00000004 <= 00000020
@00003010: $ 3 <= 00000020
@00003004: $ 1 <= 00000040
@00003008: $ 2 <= 00000040
@0000300c: *00000004 <= 00000040
```

```
@00003000: $ 1 <= 00000001
@00003004: $ 1 <= 00000002
@00003008: $ 2 <= 00000002
@0000300c: *00000004 <= 00000002
@00003010: $ 3 <= 00000002
@00003004: $ 1 <= 00000004
@00003008: $ 2 <= 00000004
@0000300c: *00000004 <= 00000004
@00003010: $ 3 <= 00000004
@00003004: $ 1 <= 00000008
@00003008: $ 2 <= 00000008
@0000300c: *00000004 <= 00000008
@00003010: $ 3 <= 00000008
@00003004: $ 1 <= 00000010
@00003008: $ 2 <= 00000010
@0000300c: *00000004 <= 00000010
@00003010: $ 3 <= 00000010
@00003004: $ 1 <= 00000020
@00003008: $ 2 <= 00000020
@0000300c: *00000004 <= 00000020
@00003010: $ 3 <= 00000020
@00003004: $ 1 <= 00000040
@00003008: $ 2 <= 00000040
@0000300c: *00000004 <= 00000040
```

图 4 ALU 测试结果

2. ori 测试

利用测试样例，将 mars 写好的文件转为 txt 文件，观察 mars 运行结果和 ISE 输出，发现两者一致。

测试代码如下：

```
.text
    ori $28,$0,0
    ori $29,$0,0
    ori $0,$0,33016
    ori $14,$23,871
    ori $29,$6,42297
    ori $5,$3,50652
    ori $30,$5,4081
    ori $18,$3,44829
    ori $22,$10,26114
    ori $30,$15,29351
    ori $23,$1,65315
    ori $25,$29,25758
    ori $16,$20,0xbf9b
    ori $30,$31,0x4e0c
```

图 7 测试代码

测试结果如下：

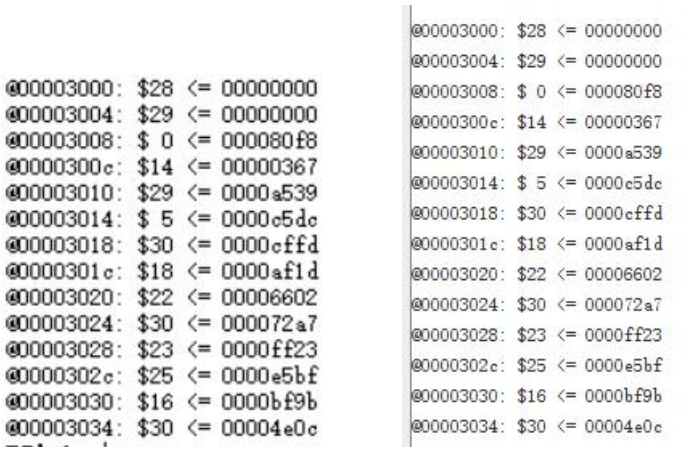

图 8 测试结果

3. beq 测试

```
.text
ori $s0,$0,1
ori $s1,$0,1
ori $s2,$0,2
ori $t0,$0,1
beq $s0,$s1,label1
ori $t1,$0,2
label1:
beq $s0,$s2,label2
subu $s2,$s2,$t0
beq $s0,$t0,label1
label2:
ori $t3,$0,4
```

图 9 beq 测试代码

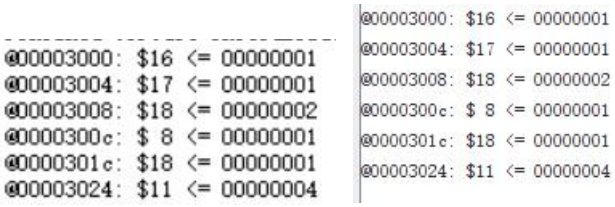

图 10 beq 测试结果

4. sblbshlh 测试

```
.text
    ori $t1,$0,1244
    addu $t2,$0,$t1
    ori $t3,$0,3
    sb $t3,3($0)
    lb $t4,3($0)
    ori $t5,$0,0x1234
    sh $t5,6($0)
    lh $t6,6($0)
    sb $t6,1($0)
    lb $t7,7($0)
```

图 11 测试代码

```
@00003000: $ 9 <= 000004dc
@00003004: $10 <= 000004dc
@00003008: $11 <= 00000003
@0000300c: *00000003 <= 03000000
@00003010: $12 <= 00000003
@00003014: $13 <= 00001234
@00003018: *00000006 <= 12340000
@0000301c: $14 <= 00001234
@00003020: *00000001 <= 03003400
@00003024: $15 <= 00000012
```

```
@00003000: $ 9 <= 000004dc
@00003004: $10 <= 000004dc
@00003008: $11 <= 00000003
@0000300c: *00000003 <= 03
@00003010: $12 <= 00000003
@00003014: $13 <= 00001234
@00003018: *00000006 <= 1234
@0000301c: $14 <= 00001234
@00003020: *00000001 <= 34
@00003024: $15 <= 00000012
```

图 12 测试结果

（二）自动测试工具

1. 测试样例生成器

使用 c 语言编写，如下：

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
int flag=0;
int th=1;
int Tuse[5][50] =
{{20,21,22,23,24,25,28,30},{31,32,33,34,8,9,0,2,7,12,13,15,14,16,17,18,19,47,48,1,4
```

```
9,4,3,8,5,11,6,38,41,42,44,43,39,40},{20,21},{0,2,0,49,8,9,4,3,5,6,35,38,41,44,36,39,
37,40,31,32,33,34},{17,18,19}};
    int Tnew[6][50] =
{{12,13,14,15,16},{0,1,2,3,4,5,6,7,8,9,10,11,27,35,36,37,38,39,40,41,42,43,44,45,46,
49},{17,18,19,20,21,22,23,24,25,26,28,29,30,31,32,33,34,47,48},{12,13,14,15,16},{
0,1,2,3,4,5,6,7,8,9,10,11,27,35,36,37,38,39,40,41,42,43,44,45,46,49,17,18,19,20,21,2
2,23,24,25,26,28,29,30,31,32,33,34,47,48},{12,13,14,15,16,0,1,2,3,4,5,6,7,8,9,10,11,
27,35,36,37,38,39,40,41,42,43,44,45,46,49,17,18,19,20,21,22,23,24,25,26,28,29,30,3
1,32,33,34,47,48}};
    int maxTuse[5]={8,34,2,22,3};
    int maxTnew[6]={5,26,19,5,45,50};

    FILE* testi;
    void sel_instr(int index,int i);
    void addu(int i);
    void add(int i);
    void sub(int i);
    void subu(int i);
    void and_(int i);
    void or_(int i);
    void xor_(int i);
    void nor(int i);

    void ori(int i);
    void addi(int i);
    void addiu(int i);
    void andi(int i);
    void xori(int i);
```

```
void lw(int i);
void lb(int i);
void lh(int i);
void lbu(int i);
void lhu(int i);
void sw(int i);
void sb(int i);
void sh(int i);

void beq(int i);
void bne(int i);
void blez(int i);
void bgtz(int i);
void bltz(int i);
void bgez(int i);

void jal(int i);
void lui(int i);
void jr(int i);
void j(int i);
void jalr(int i);

void mult(int i);
void multu(int i);
void div(int i);
void divu(int i);

void sll(int i);
void srl(int i);
```

```
void sra(int i);

void sllv(int i);
void srlv(int i);
void srav(int i);

void slt(int i);
void slti(int i);
void sltiu(int i);
void sltu(int i);

void mfhi(int i);
void mflo(int i);
void mthi(int i);
void mtlo(int i);



int    randrf(){
    int i=rand()%32;
    while (i == 0 || i == 28||i == 29){
        i = rand()%32;
    }
    return i;
}

void rand_instr(int i){
    int index = rand()%50;
    while (index == 20||index == 21||index == 22||index == 23 || index ==
24||index == 25||index == 26||index == 27||index == 28||index == 29||index == 30){
        index = rand()%50;
```

```
    }
    sel_instr(index,i);
    return;
}
void sel_instr(int index,int i){

    if(index == 0) addu(i);
    else if (index == 1) add(i);
    else if (index == 2) subu(i);
    else if (index == 49) sub(i);
    else if (index == 3) and_(i);
    else if (index == 4) or_(i);
    else if (index == 5) xor_(i);
    else if (index == 6) nor(i);

    else if (index == 7) ori(i);
    else if (index == 8) addi(i);
    else if (index == 9) addiu(i);
    else if (index == 10) andi(i);
    else if (index == 11) xori(i);
    else if (index == 27) lui(i);

    else if (index == 12) lw(i);
    else if (index == 13) lb(i);
    else if (index == 14) lh(i);
    else if (index == 15) lbu(i);
    else if (index == 16) lhu(i);
    else if (index == 17) sw(i);
    else if (index == 18) sb(i);
    else if (index == 19) sh(i);
```

```
else if (index == 20) beq(i);
else if (index == 21) bne(i);
else if (index == 22) blez(i);
else if (index == 23) bgtz(i);
else if (index == 24) bltz(i);
else if (index == 25) bgez(i);

else if (index == 26) jal(i);
else if (index == 28) jr(i);
else if (index == 29) j(i);
else if (index == 30) jalr(i);

else if (index == 31) mult(i);
else if (index == 32) multu(i);
else if (index == 33) div(i);
else if (index == 34) divu(i);

else if (index == 35) sll(i);
else if (index == 36) srl(i);
else if (index == 37) sra(i);

else if (index == 38) sllv(i);
else if (index == 39) srlv(i);
else if (index == 40) srav(i);

else if (index == 41) slt(i);
else if (index == 42) slti(i);
else if (index == 43) sltiu(i);
else if (index == 44) sltu(i);
```

```
    else if (index == 45) mfhi(i);
    else if (index == 46) mflo(i);
    else if (index == 47) mthi(i);
    else if (index == 48) mtlo(i);
}

void addu(int i){
    fprintf(testi,"addu $%d,$%d,$%d\n",i,i,i);
}
void subu(int i){
    fprintf(testi,"subu $%d,$%d,$%d\n",i,i,i);
}
void sub(int i){
    fprintf(testi,"sub $%d,$%d,$%d\n",i,i,i);
}
void add(int i){
    fprintf(testi,"addu $%d,$%d,$%d\n",i,i,i);
}
void and_(int i){
    fprintf(testi,"and $%d,$%d,$%d\n",i,i,i);
}
void or_(int i){
    fprintf(testi,"or $%d,$%d,$%d\n",i,i,i);
}
void xor_(int i){
    fprintf(testi,"xor $%d,$%d,$%d\n",i,i,i);
}
void nor(int i){
    fprintf(testi,"nor $%d,$%d,$%d\n",i,i,i);
```

```
}

void ori(int i){
    int k = rand()%10000;
    fprintf(testi,"ori $%d,$%d,%d\n",i,i,k);
}
void addi(int i){
    int k = rand()%10000;
    fprintf(testi,"addi $%d,$%d,%d\n",i,i,k);
}
void addiu(int i){
    int k = rand()%10000;
    fprintf(testi,"addiu $%d,$%d,%d\n",i,i,k);
}
void andi(int i){
    int k = rand()%10000;
    fprintf(testi,"andi $%d,$%d,%d\n",i,i,k);
}
void xori(int i){
    int k = rand()%10000;
    fprintf(testi,"xori $%d,$%d,%d\n",i,i,k);
}
void lui(int i){
    int j = rand()%10000;
    fprintf(testi,"lui $%d,%d\n",i,j);
}
```

```
void lw(int i){
    int j = (rand()%4)*4;
    fprintf(testi,"lw $%d,%d($0)\n",i,j,i);
}
void sw(int i){
    int j = (rand()%4)*4;
    fprintf(testi,"sw $%d,%d($0)\n",i,j,i);
}
void lb(int i){
    int j = (rand()%4);
    fprintf(testi,"lb $%d,%d($0)\n",i,j,i);
}
void lh(int i){
    int j = (rand()%4)*2;
    fprintf(testi,"lh $%d,%d($0)\n",i,j,i);
}
void lbu(int i){
    int j = (rand()%4);
    fprintf(testi,"lbu $%d,%d($0)\n",i,j,i);
}
void lhu(int i){
    int j = (rand()%4)*2;
    fprintf(testi,"lhu $%d,%d($0)\n",i,j,i);
}
void sh(int i){
    int j = (rand()%4)*2;
    fprintf(testi,"sh $%d,%d($0)\n",i,j,i);
}
```

```
void sb(int i){
    int j = (rand()%4);
    fprintf(testi,"sb $%d,%d($0)\n",i,j,i);
}


void jal(int i){
    fprintf(testi,"jal lable%d\n",th);
    rand_instr(i);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}
void j(int i){
    fprintf(testi,"jal lable%d\n",th);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}
void jr(int i){
    fprintf(testi,"la $%d,lable%d\n",i,th);
    fprintf(testi,"jr $%d\n",i);
    rand_instr(i);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}
```

```
void jalr(int i){
    int j = rand()%32;
    while (j == i){
        j = rand()%32;
    }
    fprintf(testi,"la $%d,lable%d\n",i,th);
    fprintf(testi,"jalr $%d,$%d\n",j,i);
    rand_instr(i);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}

void beq(int i){
    int j = rand()%32;
    while (j==1||j==28||j==29){
        j = rand()%32;
    }
    fprintf(testi,"beq $%d,$%d,lable%d\n",i,j,th);
    rand_instr(i);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}
void bne(int i){
```

```
    int j = rand()%32;
    while (j==1||j==28||j==29){
        j = rand()%32;
    }
    fprintf(testi,"bne $%d,$%d,lable%d\n",i,j,th);
    rand_instr(i);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}
void blez(int i){
    fprintf(testi,"blez $%d,lable%d\n",i,th);
    rand_instr(i);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}
void bgtz(int i){

    fprintf(testi,"bgtz $%d,lable%d\n",i,th);
    rand_instr(i);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}
void bltz(int i){
```

```
    fprintf(testi,"bltz $%d,lable%d\n",i,th);
    rand_instr(i);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}
void bgez(int i){

    fprintf(testi,"bgez $%d,lable%d\n",i,th);
    rand_instr(i);
    rand_instr(i);
    rand_instr(i);
    fprintf(testi,"lable%d:\n",th);
    th++;
}


void mult(int i){
    fprintf(testi,"mult $%d,$%d\n",i,i);
}
void multu(int i){
    fprintf(testi,"multu $%d,$%d\n",i,i);
}
void div(int i){
    int j = rand()%10000;
    while (j==0) j = rand()%10000;
    int k = randrf();
    fprintf(testi,"addi $%d,$0,%d\n",k,j);
    fprintf(testi,"div $%d,$%d\n",i,k);
```

```
}
void divu(int i){
    int j = rand()%10000;
    while (j==0) j = rand()%10000;
    int k = randrf();
    fprintf(testi,"addi $%d,$0,%d\n",k,j);
    fprintf(testi,"divu $%d,$%d\n",i,k);
}


void sll(int i){
    int j = rand()%32;
    fprintf(testi,"sll $%d,$%d,%d\n",i,i,j);
}
void srl(int i){
    int j = rand()%32;
    fprintf(testi,"srl $%d,$%d,%d\n",i,i,j);
}
void sra(int i){
    int j = rand()%32;
    fprintf(testi,"sra $%d,$%d,%d\n",i,i,j);
}


void sllv(int i){
    fprintf(testi,"sllv $%d,$%d,$%d\n",i,i,i);
}
void srlv(int i){
    fprintf(testi,"srlv $%d,$%d,$%d\n",i,i,i);
}
```

```
void srav(int i){
    fprintf(testi,"srav $%d,$%d,$%d\n",i,i,i);
}

void slt(int i){
    int j = randrf();
    fprintf(testi,"slt $%d,$%d,$%d\n",i,i,j);

}
void slti(int i){
    int j = rand()%10000;
    fprintf(testi,"slti $%d,$%d,%d\n",i,i,j);

}
void sltiu(int i){
    int j = rand()%10000;
    fprintf(testi,"sltiu $%d,$%d,%d\n",i,i,j);

}
void sltu(int i){
    int j = randrf();
    fprintf(testi,"sltu $%d,$%d,$%d\n",i,i,j);

}


void mfhi(int i){
    fprintf(testi,"mfhi $%d\n",i);
}
void mflo(int i){
```

```
    fprintf(testi,"mflo $%d\n",i);
}
void mthi(int i){
    fprintf(testi,"mthi $%d\n",i);
}
void mtlo(int i){
    fprintf(testi,"mtlo $%d\n",i);
}

int lines = 100;
int main() {
    int i=1,rf,instr1,instr2,t;
    srand((unsigned)time(NULL));
    char s[100] = "";
    int tuse,tnew;
    for(tuse=0;tuse<5;tuse++){
        for(tnew = 0;tnew<6;tnew++){

            rf = rand()%32;
            while (rf==28||rf==29||rf==0) rf = rand()%32;
            sprintf(s,"test%d.asm",i);
            testi=fopen(s,"w");
                for(t=0;t<lines;t++){
                    instr1=Tnew[tnew][rand()%maxTnew[tnew]];
                    sel_instr(instr1,rf);
                    if(tnew<3){

                    }
                    else if(tnew<5){
                        rand_instr(randrf());
```

```
            }
            else{
                rand_instr(randrf());
                rand_instr(randrf());
            }
            instr2=Tuse[tuse][rand()%maxTuse[tuse]];
            sel_instr(instr2,rf);
        }
        printf("point%d\n",i);
fclose(testi);
i++;

rf = rand()%32;
while (rf==28||rf==29||rf==0) rf = rand()%32;
sprintf(s,"test%d.asm",i);
testi=fopen(s,"w");
        for(t=0;t<lines;t++){
            instr1=Tnew[tnew][rand()%maxTnew[tnew]];
            sel_instr(instr1,rf);
            if(tnew<3){

            }
            else if(tnew<5){
                rand_instr(randrf());
            }
            else{
                rand_instr(randrf());
                rand_instr(randrf());
            }
            instr2=Tuse[tuse][rand()%maxTuse[tuse]];
```

```
            sel_instr(instr2,rf);
        }
        printf("point%d\n",i);
fclose(testi);
i++;




rf = rand()%32;
while (rf==28||rf==29||rf==0) rf = rand()%32;
sprintf(s,"test%d.asm",i);
testi=fopen(s,"w");
        for(t=0;t<lines;t++){
            instr1=Tnew[tnew][rand()%maxTnew[tnew]];
            sel_instr(instr1,rf);
            if(tnew<3){

            }
            else if(tnew<5){
                rand_instr(randrf());
            }
            else{
                rand_instr(randrf());
                rand_instr(randrf());
            }
            instr2=Tuse[tuse][rand()%maxTuse[tuse]];
            sel_instr(instr2,rf);
        }
        printf("point%d\n",i);
```

```
fclose(testi);
i++;




rf = rand()%32;
while (rf==28||rf==29||rf==0) rf = rand()%32;
sprintf(s,"test%d.asm",i);
testi=fopen(s,"w");
      for(t=0;t<lines;t++){
            instr1=Tnew[tnew][rand()%maxTnew[tnew]];
            sel_instr(instr1,rf);
            if(tnew<3){

            }
            else if(tnew<5){
                  rand_instr(randrf());
            }
            else{
                  rand_instr(randrf());
                  rand_instr(randrf());
            }
            instr2=Tuse[tuse][rand()%maxTuse[tuse]];
            sel_instr(instr2,rf);
      }
      printf("point%d\n",i);
fclose(testi);
i++;
```

```
rf = rand()%32;
while (rf==28||rf==29||rf==0) rf = rand()%32;
sprintf(s,"test%d.asm",i);
testi=fopen(s,"w");
    for(t=0;t<lines;t++){
        instr1=Tnew[tnew][rand()%maxTnew[tnew]];
        sel_instr(instr1,rf);
        if(tnew<3){

        }
        else if(tnew<5){
            rand_instr(randrf());
        }
        else{
            rand_instr(randrf());
            rand_instr(randrf());
        }
        instr2=Tuse[tuse][rand()%maxTuse[tuse]];
        sel_instr(instr2,rf);
    }
    printf("point%d\n",i);
fclose(testi);
i++;
```

```
rf = rand()%32;
while (rf==28||rf==29||rf==0) rf = rand()%32;
sprintf(s,"test%d.asm",i);
testi=fopen(s,"w");
    for(t=0;t<lines;t++){
        instr1=Tnew[tnew][rand()%maxTnew[tnew]];
        sel_instr(instr1,rf);
        if(tnew<3){

        }
        else if(tnew<5){
            rand_instr(randrf());
        }
        else{
            rand_instr(randrf());
            rand_instr(randrf());
        }
        instr2=Tuse[tuse][rand()%maxTuse[tuse]];
        sel_instr(instr2,rf);
    }
    printf("point%d\n",i);
fclose(testi);
i++;




rf = rand()%32;
while (rf==28||rf==29||rf==0) rf = rand()%32;
sprintf(s,"test%d.asm",i);
```

```
testi=fopen(s,"w");
      for(t=0;t<lines;t++){
            instr1=Tnew[tnew][rand()%maxTnew[tnew]];
            sel_instr(instr1,rf);
            if(tnew<3){

            }
            else if(tnew<5){
                  rand_instr(randrf());
            }
            else{
                  rand_instr(randrf());
                  rand_instr(randrf());
            }
            instr2=Tuse[tuse][rand()%maxTuse[tuse]];
            sel_instr(instr2,rf);
      }
      printf("point%d\n",i);
fclose(testi);
i++;




rf = rand()%32;
while (rf==28||rf==29||rf==0) rf = rand()%32;
sprintf(s,"test%d.asm",i);
testi=fopen(s,"w");
      for(t=0;t<lines;t++){
            instr1=Tnew[tnew][rand()%maxTnew[tnew]];
```

```
            sel_instr(instr1,rf);
            if(tnew<3){

            }
            else if(tnew<5){
                rand_instr(randrf());
            }
            else{
                rand_instr(randrf());
                rand_instr(randrf());
            }
            instr2=Tuse[tuse][rand()%maxTuse[tuse]];
            sel_instr(instr2,rf);
        }
        printf("point%d\n",i);
fclose(testi);
i++;




rf = rand()%32;
while (rf==28||rf==29||rf==0) rf = rand()%32;
sprintf(s,"test%d.asm",i);
testi=fopen(s,"w");
        for(t=0;t<lines;t++){
            instr1=Tnew[tnew][rand()%maxTnew[tnew]];
            sel_instr(instr1,rf);
            if(tnew<3){
```

```
            }
            else if(tnew<5){
                rand_instr(randrf());
            }
            else{
                rand_instr(randrf());
                rand_instr(randrf());
            }
            instr2=Tuse[tuse][rand()%maxTuse[tuse]];
            sel_instr(instr2,rf);
        }
        printf("point%d\n",i);
fclose(testi);
i++;




rf = rand()%32;
while (rf==28||rf==29||rf==0) rf = rand()%32;
sprintf(s,"test%d.asm",i);
testi=fopen(s,"w");
        for(t=0;t<lines;t++){
            instr1=Tnew[tnew][rand()%maxTnew[tnew]];
            sel_instr(instr1,rf);
            if(tnew<3){

            }
            else if(tnew<5){
```

```
                rand_instr(randrf());
            }
            else{
                rand_instr(randrf());
                rand_instr(randrf());
            }
            instr2=Tuse[tuse][rand()%maxTuse[tuse]];
            sel_instr(instr2,rf);
        }
        printf("point%d\n",i);
fclose(testi);
i++;




rf = 31;
sprintf(s,"test%d.asm",i);
testi=fopen(s,"w");
        for(t=0;t<lines;t++){
            instr1=Tnew[tnew][rand()%maxTnew[tnew]];
            sel_instr(instr1,rf);
            if(tnew<3){

            }
            else if(tnew<5){
                rand_instr(randrf());
            }
            else{
                rand_instr(randrf());
```

```
                    rand_instr(randrf());
                }
                instr2=Tuse[tuse][rand()%maxTuse[tuse]];
                sel_instr(instr2,rf);
            }
            printf("point%d\n",i);
        fclose(testi);
        i++;
    }
}

    return 0;
}
```

之后利用 ch 大佬的测评机进行测试。

三、思考题

（一）为什么需要有单独的乘除法部件而不是整合进 ALU？为何需要有独立的 HI、LO 寄存器？

因为乘除法运算所需时间远远大于之前 ALU 中的运算，如果将乘除法部件整合进 ALU，在运行乘除法运算时需要大量时间，会导致 CPU 的运行频率大幅下降。

因为 32 位数与 32 位数相乘会有 64 位，因此需要两个寄存器用来分别保存计算结果的高 32 位和低 32 位，同时做除法运算时，也需要一个寄存器去保存商，另一个寄存器去保存余数，因此在乘除法部件中内置了 HI 和 LO 寄存器，这也使得乘除法部件可以独立于 RF 运行，乘除法部件之外的部件可以继续运行其他指令（无冲突情况下），在需要访问乘除法计算结果时再从 HI 和 LO 中取出结果，提高了 CPU 的效率。

（二）参照你对延迟槽的理解，试解释“乘除槽”。

与延迟槽类似，当碰上一个乘除法指令时，会带来 5 或 10 个周期的延迟，使用乘除槽可以把后面不需要使用乘除法部件的指令调整到乘除槽，从而提高 CPU 的效率。

（三）举例说明并分析何时按字节访问内存相对于按字访问内存性能上更有优势。（Hint： 考虑 C 语言中字符串的情况）

当对字符或字符串操作时，因为一个字符仅占一个字节，如果按字访问将把这个字符所在的字读取出来，造成了较大浪费。

（四）在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？

遇到了很多冲突，比如 ALU 计算型指令接 beq 和 jr 出现的冲突。使用 AT 法使其暂停下来即可。

先列出 T 的集合如下：

Tuse_rs=0 ： {beq | bgtz | blez | bne | bltz | bgez| jr | jalr}

Tuse_rs=1 ：{mult|multu|div|divu|addi | addiu | addu | subu | ori | lw | lb | lbu | lh | lhu | sw |sh|sb| mthi | mtlo | add | sub | or_ | and_ | andi | xor_ | xori | nor_ | sllv | slt | slti | sltu | sltiu | srlv | srav}

Tuse_rt=0 ： {beq | bne}

Tuse_rt=1 ：{ addu | subu | add | sub |addi|addiu| or_ | and_ | xor_ | nor_ | sll | sllv | slt | sltu | srl | srlv | sra | srav |mult|multu|div|divu}

Tuse_rt=2 ： { sw | sh | sb}

Tnew_E = 2 、Tnew_M = 1 ： { lw | lb | lh | lbu | lhu}

Tnew_E = 1 、Tnew_M = 0 ：{ addu | subu | ori | lui | addi | add | sub | or_ | and_ | andi | xor_ | xori | nor_ | addiu | sll | sllv | slt | slti | sltu | sltiu | srl | srlv | sra | srav | mfhi | mflo}

再列出策略矩阵：

	Tnew_E=2	Tnew_E=1	Tnew_E=0	Tnew_M=1	Tnew_M=0	Tnew_W=0
Tuse_rs=0	S	S	F	S	F	F
Tuse_rs=1	S	F	F	F	F	F
Tuse_rt=0	S	S	F	S	F	F
Tuse_rt=1	S	F	F	F	F	F
Tuse_rt=2	F	F	F	F	F	F

即把 stall 和 forward 分成了 30 种情况，在生成代码时，根据 Tnew 的级数，在 Tnew 生成代码与 Tuse 生成代码之间插入相应数量的随机指令。如 Tnew 为 E 级，那么 Tnew 与 Tuse 产生的指令之间没有其他指令。

Eg：Tnew 是 E 级

load 类 $1,$x($y)

store 类 $1,$x($y)

这样就生成了可以检测 W 级向 M 级是否有转发通路的代码

当 Tnew 是 M 级，那么插入一条随机指令；如果 Tnew 是 W 级，那么插入 2 条随机指令。

对 30 种情况的 Stall 和 Forward 分别随机生成 11 个点，为保证足够冲突，每个点中都仅对一个寄存器操作，为保证考虑到 31 号寄存器的特殊情况，最后一个点固定为对 31 号寄存器生成。因此每次生成 30*11 个点。每个点中都对该种情况的 Tnew 和 Tuse 随机生成 180 次指令，可以达到较强的测试效果.

```
for(t=0;t<150;t++){
                instr1=Tnew[tnew][rand()%maxTnew[tnew]];
```

```
        sel_instr(instr1,rf);
        if(tnew<3){

        }
        else if(tnew<5){
            rand_instr(randrf());
        }
        else{
            rand_instr(randrf());
            rand_instr(randrf());
        }
        instr2=Tuse[tuse][rand()%maxTuse[tuse]];
        sel_instr(instr2,rf);
    }
```

（五）为了对抗复杂性你采取了哪些抽象和规范手段？这些手段在译码和处理数据冲突的时候有什么样的特点与帮助？

将功能相似的指令放在一起，如我把指令分为了以下几组：

{addu,add,sub,subu,and,or,xor,nor}

{ori,addi,addiu,andi,xori,lui}

{lw,lh,lhu,lb,lbu,sw,sh,sb}

{beq,bne,blez,bgtz,bltz,bgez}

{jal,jr,j,jalr}

{mult,mulu,div,divu}

{sll,srl,sra}{sllv,srlv,srav}

{slt,slti,sltiu,sltu}

{mfhi,mflo,mthi,mtlo}

这样在添加指令时，一组一组地添加，他们一般只有一个控制信号是不同的，加起来非常方便。

在处理冲突时，每一组的指令的 T 基本是一样的，因此添加 Tuse 与 Tnew 时也十分方便。