计算机组成原理实验报告（Verilog 流水线）

一， 设计通路设计

（1） PC

信号名	方向	描述
clk	I	处理器时钟信号
reset	I	复位信号
next_PC [31:0]	I	下一个 PC 的值
PC [31:0]	O	输出当前指令的地址

序号	功能名称	功能描述
1	取出指令地址	取出将要执行的指令地址

（2） PC_calculator

信号名	方向	描述
PC [31:0]	I	当前 PC 值
PCOp [1:0]	I	下条 PC 输出控制信号
equal	I	寄存器值是否相等
instr_index [25:0]	I	指令的后 26 位
rs [31:0]	I	寄存器储存的值
imm [31:0]	I	扩展后的立即数
PC_plus_four [31:0]	O	PC+4
next_PC [31:0]	O	下条 PC 的值

序号	功能名称	功能描述
1	计算下一条 PC	计算下一条指令的地址

（3） im

信号名	方向	描述
addr [9:0]	I	当下 PC 的值
Instr [31:0]	O	取出的 32 位指令

序号	功能名称	功能描述
1	取出指令	取出将要执行的指令

（4） grf

信号名	方向	描述
clk	I	处理器时钟信号
reset	I	清零信号
RA1[4:0]	I	读寄存器时第一个寄存器的编号（地址）
RA2[4:0]	I	读寄存器时第二个寄存器的编号（地址）
WD[31:0]	I	寄存器写入数据
WA[4:0]	I	写寄存器时的编号（地址）
RegWrite	I	寄存器写使能信号
WPC	I	当前 PC 的值
RD1[31:0]	O	读寄存器时第一个寄存器的输出数据
RD2[31:0]	O	读寄存器时第二个寄存器的输出数据

序号	功能名称	功能描述

1	读寄存器	RD1 输出 RA1 所寻址的寄存器中的数据 RD2 输出 RA2 所寻址的寄存器中的数据
2	写寄存器	当时钟上升沿到来且 RegWrite 信号有效时， WD 被写入 WA 所寻址的寄存器

（5） alu

信号名	方向	描述
A[31:0]	I	参与 ALU 计算的第一个值
B[31:0]	I	参与 ALU 计算的第二个值
ALUOp[2:0]	I	ALU 功能的选择信号： 000：ALU 进行加法运算 001：ALU 进行减法运算 010：ALU 进行或运算 011：ALU 进行与运算 100：ALU 进行逻辑移位 101：ALU 进行算术移位
C[31:0]	O	ALU 的计算结果
Equal	O	判断两数是否相等： 0：A!=B 1：A=B

序号	功能名称	功能描述
1	无符号加运算	Result = A + B
2	无符号减运算	Result = A - B
3	或运算	Result = A | B
4	与运算	Result = A & B
5	算术移位	Result=A>>B;
6	逻辑移位	Result=$signed(A)>>>B
7	判断是否相等	If(A == B)　Equal=1

（6） dm

信号名	方向	描述
clk	I	控制器时钟信号
reset	I	存储器复位
MemWrite	I	写使能信号
Addr [9:0]	I	读或写时对应存储器时地址
WD[31:0]	I	写存储器时写入数据
PC[31:0]	I	存储器写使能信号
RD[31:0]	O	读存储器时的输出

序号	功能名称	功能描述

1	读存储器	RD 输出存储器中地址 Addr 存储的数据
2	写存储器	当时钟上升沿到来并且 MemWrite 有效时，WD 被写入存储器中地址 Addr 的位置

（7） mux32_4

信号名	方向	描述
In0 [31:0]	I	第一个输入
In1 [31:0]	I	第二个输入
In2 [31:0]	I	第三个输入
In3 [31:0]	I	第四个输入
Select [1:0]	I	选择信号
Out[31:0]	O	输出结果

序号	功能名称	功能描述
1	选择输出	Select=0：输出 In0 Select=1：输出 In1 Select=2：输出 In2 Select=3：输出 In3

（8） mux 5_4

信号名	方向	描述
In0 [4:0]	I	第一个输入
In1 [4:0]	I	第二个输入
In2 [4:0]	I	第三个输入
In3 [4:0]	I	第四个输入
Select [1:0]	I	选择信号
Out[4:0]	O	输出结果

序号	功能名称	功能描述
1	选择输出	Select=0：输出 In0 Select=1：输出 In1 Select=2：输出 In2 Select=3：输出 In3

（9） mux 32_2

信号名	方向	描述
In0 [31:0]	I	第一个输入
In1 [31:0]	I	第二个输入
Select	I	选择信号
Out[31:0]	O	输出结果

序号	功能名称	功能描述
1	选择输出	Select=0：输出 In0 Select=1：输出 In1

（10） ext

信号名	方向	描述
imm[15:0]	I	需要被扩展的 16 位立即数
EOp[1:0]	I	扩展方式选择信号： 00：符号扩展到 32 位 01：高位 0 扩展到 32 位 10：将立即数加载到高位，低位补 0 11：符号扩展之后左移两位
ext_imm	O	相应扩展后的立即数

序号	功能名称	功能描述
1	符号扩展	将 imm 进行符号扩展到 32 位
2	零扩展	将 imm 进行高位补 0 扩展到 32 位
3	加载到高位	将 imm 加载到高位，低位补 0
4	符号扩展之后左移 2 位	将 imm 进行符号扩展之后左移 2 位

（11） D 级流水寄存器

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
Stall	I	暂停控制信号
IR_F[31:0]	I	来自 F 级的指令

PC4_F[31:0]	I	来自 F 级的 PC+4
IR_D[31:0]	O	D 级指令
PC4_D[31:0]	O	D 级 PC +4

（12） E 级流水寄存器

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
Stall	I	暂停控制信号
IR_D[31:0]	I	来自 D 级的指令
PC4_D[31:0]	I	来自 D 级的 PC+4
RD1_D[31:0]	I	来自 D 级的寄存器第一个输出值
RD2_D[31:0]	I	来自 D 级的寄存器第二个输出值
Imm_D[31:0]	I	来自 D 级的扩展后的立即数
RegDst_D[1:0]	I	来自 D 级的寄存器 WA 选择控制信号
ALUOp_D[2:0]	I	来自 D 级的 alu 控制信号
MemWrite_D	I	来自 D 级的存储器写使能信号
RegWrite_D	I	来自 D 级的寄存器写使能信号
ALUSrc_D	I	来自 D 级的 alu B 端选择控制信号
MemtoReg_D[1:0]	I	来自 D 级的寄存器 WD 选择控制信号
IR_E[31:0]	O	来自 E 级的指令
PC4_E[31:0]	O	来自 E 级的 PC+4
RD1_E[31:0]	O	来自 E 级的寄存器第一个输出值
RD2_E[31:0]	O	来自 E 级的寄存器第二个输出值
Imm_E[31:0]	O	来自 E 级的扩展后的立即数
RegDst_E[1:0]	O	来自 E 级的寄存器 WA 选择控制信号
ALUOp_E[2:0]	O	来自 E 级的 alu 控制信号
MemWrite_E	O	来自 E 级的存储器写使能信号
RegWrite_E	O	来自 E 级的寄存器写使能信号
ALUSrc_E	O	来自 E 级的 alu B 端选择控制信号
MemtoReg_E[1:0]	O	来自 E 级的寄存器 WD 选择控制信号

(13) M 级流水寄存器

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
IR_E[31:0]	I	来自 E 级的指令
PC4_E[31:0]	I	来自 E 级的 PC+4
RT_E[31:0]	I	来自 E 级的寄存器第二个输出值或者转发值
ALUOut_E[31:0]	I	来自 E 级的 ALU 结果
Imm_E[31:0]	I	来自 E 级的扩展后的立即数
WA_E[4:0]	I	来自 E 级的寄存器写入地址

MemWrite_E	I	来自 E 级的存储器写使能信号
RegWrite_E	I	来自 E 级的寄存器写使能信号
MemtoReg_E[1:0]	I	来自 E 级的寄存器 WD 选择控制信号
IR_M[31:0]	O	来自 M 级的指令
PC4_M[31:0]	O	来自 M 级的 PC+4
RT_M[31:0]	O	来自 M 级的寄存器第二个输出值或者转发值
ALUOut_M[31:0]	O	来自 M 级的 ALU 结果
Imm_M[31:0]	O	来自 M 级的扩展后的立即数
WA_M[4:0]	O	来自 M 级的寄存器写入地址
MemWrite_M	O	来自 M 级的存储器写使能信号
RegWrite_M	O	来自 M 级的寄存器写使能信号
MemtoReg_M[1:0]	O	来自 M 级的寄存器 WD 选择控制信号

(14) W 级流水寄存器

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
IR_E[31:0]	I	来自 E 级的指令
PC4_E[31:0]	I	来自 E 级的 PC+4
RT_E[31:0]	I	来自 E 级的寄存器第二个输出值或者转发值
ALUOut_E[31:0]	I	来自 E 级的 ALU 结果
Imm_E[31:0]	I	来自 E 级的扩展后的立即数
WA_E[4:0]	I	来自 E 级的寄存器写入地址
MemWrite_E	I	来自 E 级的存储器写使能信号
RegWrite_E	I	来自 E 级的寄存器写使能信号
MemtoReg_E[1:0]	I	来自 E 级的寄存器 WD 选择控制信号
IR_M[31:0]	O	来自 M 级的指令
PC4_M[31:0]	O	来自 M 级的 PC+4
RT_M[31:0]	O	来自 M 级的寄存器第二个输出值或者转发值
ALUOut_M[31:0]	O	来自 M 级的 ALU 结果
Imm_M[31:0]	O	来自 M 级的扩展后的立即数
WA_M[4:0]	O	来自 M 级的寄存器写入地址
MemWrite_M	O	来自 M 级的存储器写使能信号
RegWrite_M	O	来自 M 级的寄存器写使能信号
MemtoReg_M[1:0]	O	来自 M 级的寄存器 WD 选择控制信号

二， 控制器设计

详细见 excel

三， 测试程序与结果

观察方法：
1，通过观察转发信号（五个 Forward）
2，grf 与 dm 的 display

暂停测试代码：

```
addu $1,$2,$3      #Stall
beq $1,$4,label1
nop
label1:
nop
nop
nop
addu $1,$2,$3      #Stall
beq $4,$1,label2
nop
label2:
nop
nop
nop
ori $1,$2,100      #Stall
beq $3,$1,label3
nop
label3:
nop
nop
nop
ori $1,$2,100      #Stall
beq $1,$3,label4
nop
label4:
nop
nop
nop
######################
lw $1,($0)      #Stall
```

```
beq $1,$2,label9
nop
label9:
nop
nop
nop
lw $1,($0)      #Stall
beq $2,$1,label10
nop
label10:
nop
nop
nop
lw $1,($0)      #Stall
addu $2,$1,$3
nop
nop
nop
lw $1,($0)      #Stall
subu $2,$3,$1
nop
nop
nop
lw $1,($0)      #Stall
lw $2,($1)
nop
nop
nop
lw $1,($0)      #Stall
sw $2,($1)
nop
nop
nop
lw $1,($0)      #Stall
sw $1,($2)
nop
nop
nop
lw $1,($0)      #Stall
nop
beq $1,$2,label11
nop
label11:
nop
```

```
nop
nop
lw $1,($0)      #Stall
nop
beq $2,$1,label12
nop
label12:
nop
nop
nop
```

RS_D 测试代码：

```
lui $1,100      #5
beq $1,$2,label1
nop
label1:
nop
nop
nop
jal label2      #3
label2:
beq $31,$1,label3
nop
label3:
nop
nop
nop
addu $1,$2,$3      #1
nop
beq $1,$2,label4
nop
label4:
nop
nop
nop
ori $1,$2,100      #1
nop
beq $1,$2,label5
nop
label5:
nop
nop
nop
lui $1,100      #4
```

```
nop
beq $1,$2,label6
nop
label6:
nop
nop
nop
jal label7      #2
label7:
nop
beq $31,$1,label8
nop
label8:
nop
nop
nop
lw $1,($0)      #1
nop
beq $1,$2,label9
nop
label9:
nop
nop
nop
```

RS_E 测试代码：

```
addu $1,$2,$3    #RS_E
subu $5,$1,$4
nop
nop
nop
addu $1,$2,$3    #RS_E
ori $4,$1,7
nop
nop
nop
addu $1,$2,$3    #RS_E
lw $4,($1)
nop
nop
nop
addu $1,$2,$3    #RS_E
sw $4,($1)
nop
```

```
nop
nop
ori $1,$2,100    #RS_E
subu $5,$1,$4
nop
nop
nop
ori $1,$2,100    #RS_E
ori $4,$1,7
nop
nop
nop
ori $1,$2,100    #RS_E
lw $4,($1)
nop
nop
nop
ori $1,$2,100    #RS_E
sw $4,($1)
nop
nop
nop
jal label          #RS_E
subu $5,$31,$4
nop
nop
nop
jal label          #RS_E
ori $4,$31,7
nop
nop
nop
jal label          #RS_E
lw $4,($31)
nop
nop
nop
jal label          #RS_E
sw $4,($31)
nop
nop
nop
lui $1,100         #RS_E
subu $5,$1,$4
```

```
nop
nop
nop
lui $1,100          #RS_E
ori $4,$1,7
nop
nop
nop
lui $1,100          #RS_E
lw $4,($1)
nop
nop
nop
lui $1,100          #RS_E
sw $4,($1)
nop
nop
nop
addu $1,$2,$3    #RS_E
nop
subu $5,$1,$4
nop
nop
nop
addu $1,$2,$3    #RS_E
nop
ori $4,$1,7
nop
nop
nop
addu $1,$2,$3    #RS_E
nop
lw $4,($1)
nop
nop
nop
addu $1,$2,$3    #RS_E
nop
sw $4,($1)
nop
nop
nop
ori $1,$2,100    #RS_E
nop
```

```
subu $5,$1,$4
nop
nop
nop
ori $1,$2,100    #RS_E
nop
ori $4,$1,7
nop
nop
nop
ori $1,$2,100    #RS_E
nop
lw $4,($1)
nop
nop
nop
ori $1,$2,100    #RS_E
nop
sw $4,($1)
nop
nop
nop
jal label          #RS_E
nop
subu $5,$31,$4
nop
nop
nop
jal label          #RS_E
nop
ori $4,$31,7
nop
nop
nop
jal label          #RS_E
nop
lw $4,($31)
nop
nop
nop
jal label          #RS_E
nop
sw $4,($31)
nop
```

```
nop
nop
lui $1,100          #RS_E
nop
subu $5,$1,$4
nop
nop
nop
lui $1,100          #RS_E
nop
ori $4,$1,7
nop
nop
nop
lui $1,100          #RS_E
nop
lw $4,($1)
nop
nop
nop
lui $1,100          #RS_E
nop
sw $4,($1)
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
label:
jr $31
```

RT_D 测试代码：

```
lui $1,100
beq $2,$1,label1
nop
label1:
nop
nop
nop
jal label2      #3
```

```
label2:
beq $1,$31,label3
nop
label3:
nop
nop
nop
addu $1,$2,$3      #1
nop
beq $2,$1,label4
nop
label4:
nop
nop
nop
ori $1,$2,100      #1
nop
beq $2,$1,label5
nop
label5:
nop
nop
nop
lui $1,100      #4
nop
beq $2,$1,label6
nop
label6:
nop
nop
nop
jal label7      #2
label7:
nop
beq $1,$31,label8
nop
label8:
nop
nop
nop
lw $1,($0)      #1
nop
beq $2,$1,label9
nop
```

```
label9:
nop
nop
nop
```

RT_E 测试代码：

```
addu $1,$2,$3    #RT_E
subu $5,$4,$1
nop
nop
nop
ori $1,$2,100    #RT_E
subu $5,$4,$1
nop
nop
nop
jal label          #RT_E
subu $5,$4,$31
nop
nop
nop
lui $1,100         #RT_E
subu $5,$4,$1
nop
nop
nop
addu $1,$2,$3    #RT_E
nop
subu $5,$4,$1
nop
nop
nop
ori $1,$2,100    #RT_E
nop
subu $5,$4,$1
nop
nop
nop
jal label          #RT_E
nop
subu $5,$4,$31
nop
nop
nop
```

```
lui $1,100          #RT_E
nop
subu $5,$4,$1
nop
nop
nop
addu $1,$2,$3    #RT_E
nop
sw $1,($0)
nop
nop
nop
ori $1,$2,100    #RT_E
nop
sw $1,($0)
nop
nop
nop
jal label           #RT_E
nop
sw $1,($0)
nop
nop
nop
lui $1,100          #RT_E
nop
sw $1,($0)
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
label:
jr $31
```

此外，我们还设计了专门用于检测 jal,jr 等跳转指令的检验代码：

```
jal label1    #Stall
nop

jal label2    #Stall
```

```
nop

jal label3    #Stall
nop

jal label4    #Stall
nop

beq $5,$6,end

addu $5,$6,$7
addu $6,$7,$8
label4:
sw $31,($0)
lw $31,($0)
nop
jr $31
nop

addu $5,$6,$7
addu $6,$7,$8
label3:
sw $31,($0)
lw $31,($0)
jr $31
nop

addu $5,$6,$7
addu $6,$7,$8
label2:
ori $31,$31,0
jr $31
nop

addu $5,$6,$7
addu $6,$7,$8
label1:
addu $31,$31,$0
jr $31
nop

end:
addu $1,$0,$0
nop
```

```
nop
nop
nop
nop
nop
```

最后附上综合测试代码：

```
lui $1,0xffff
beq $1,$0,label1
nop
jal label2
nop
addu $2,$1,$3
beq $1,$2,label1
nop
label2:
jr $31
nop
label1:
subu $1,$1,$2
lui $1,0xff00
lui $2,0x00ff
addu $3,$1,$2
sw $3,($0)
lw $3,($0)
lw $4,($0)
beq $3,$4,label3
nop
addu $5,$6,$7
addu $6,$5,$7
label3:
addu $7,$5,$6
nop
nop
nop
nop
nop
nop
```

测试结果：

```
45@00003000: $ 1 <= ffff0000
75@0000300c: $31 <= 00003014
115@00003014: $ 2 <= ffff0000
155@00003028: $ 1 <= 00000000
```

```
165@0000302c: $ 1 <= ff000000
175@00003030: $ 2 <= 00ff0000
185@00003034: $ 3 <= ffff0000
185@00003038: *00000000 <= ffff0000
205@0000303c: $ 3 <= ffff0000
215@00003040: $ 4 <= ffff0000
265@00003054: $ 7 <= 00000000
```

四， 思考题

excel 附上冒险分析已经详尽列举所以情况。