计算机组成实验指导书-LAB5

1. OVERVIEW

1.1 实验名称

简单的类 MIPS 单周期处理器实现 – 整体调试

1.2 实验目的

完成单周期的类 MIPS 处理器

设计支持 16 条 MIPS 指令（add, sub, and, or, addi, andi, ori, slt,lw, sw, beq, j, jal, jr, sll, srl,）的单周期 CPU

1.3 实验内容

1．Instruction memory 的实现

2．单周期 CPU 的实现与调试

3．功能仿真

4．上板验证

1.4 实验报告与验收办法

需提交电子版报告，仿真、上板检查验收

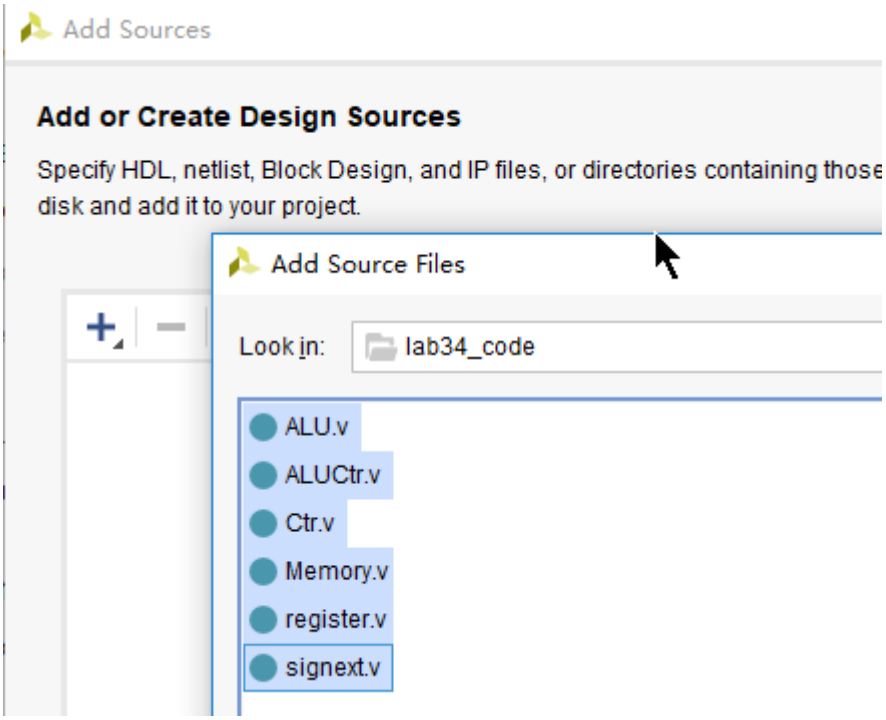
2. 新建工程，导入文件

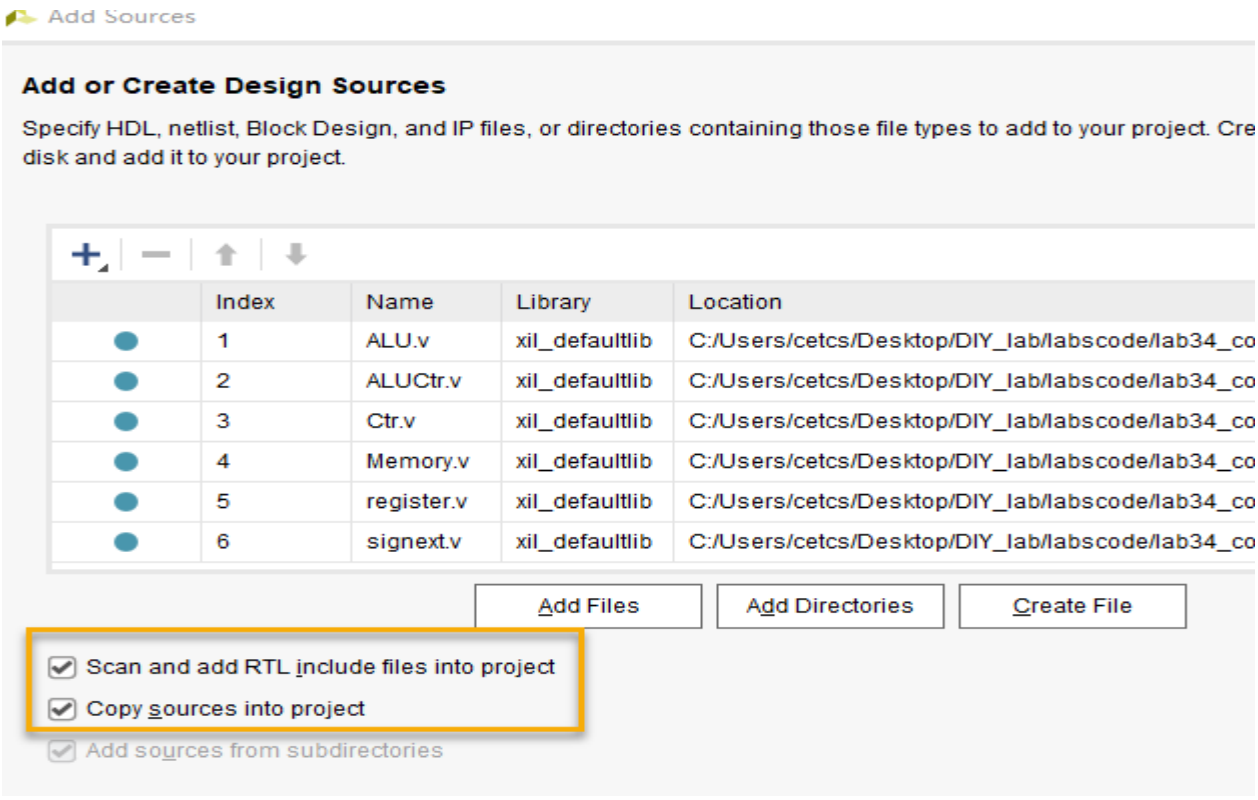
2.1 实验描述

2.1.1 新建工程

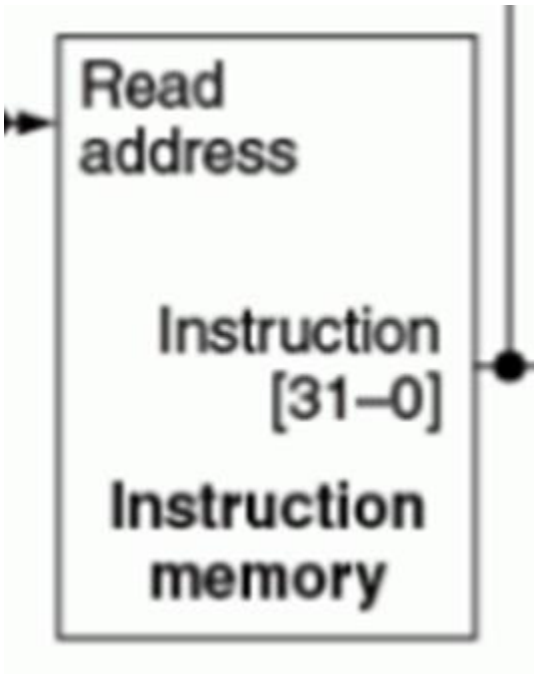
1. 启动 Vivado
2. 新建工程 lab5

3.. 将此前两次实验中的模块添加到 lab5 工程目录下





4. 按下图，新建指令模块源文件，可取名 InstMemory.v

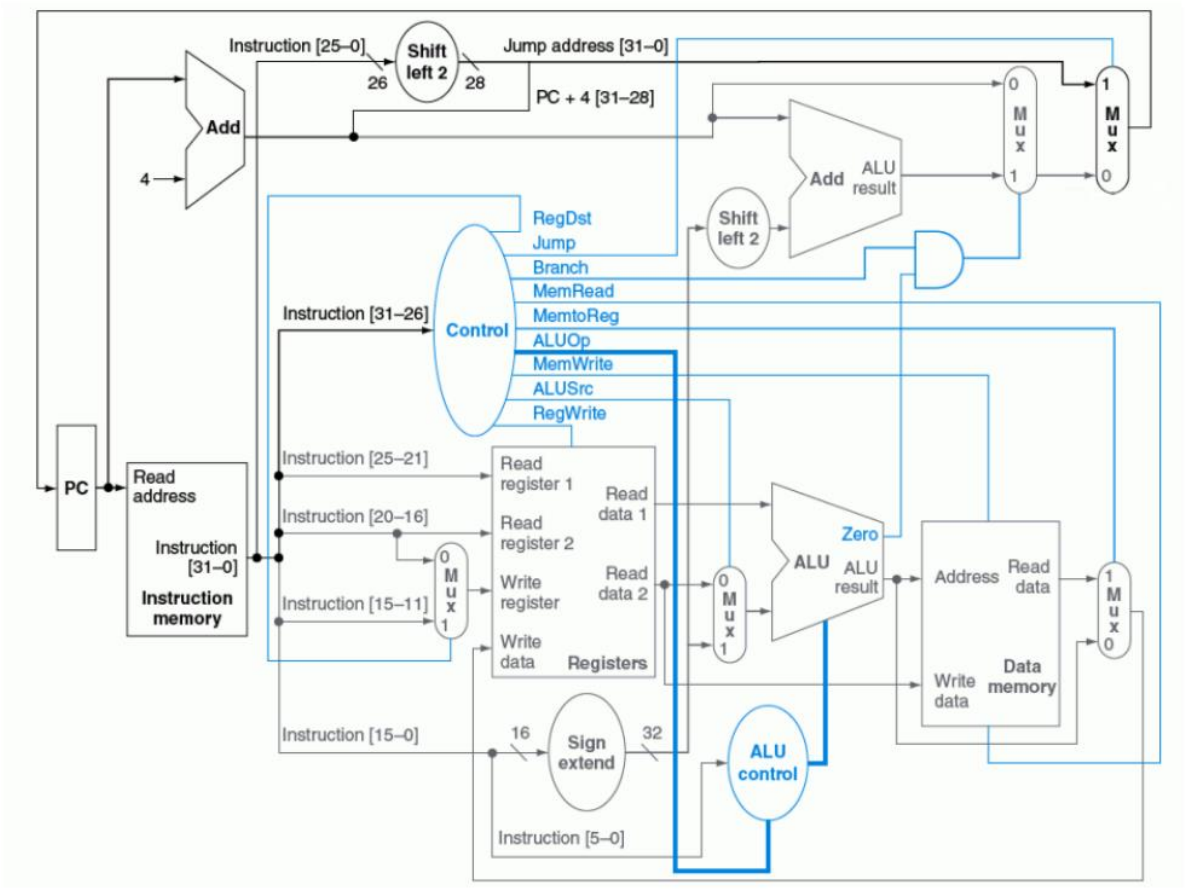


3. 顶层模块 TOP

3.1 实验描述

联合调试单周期类 MIPS 处理器

3.1.1 模块描述



MIPS 单周期处理器原理图

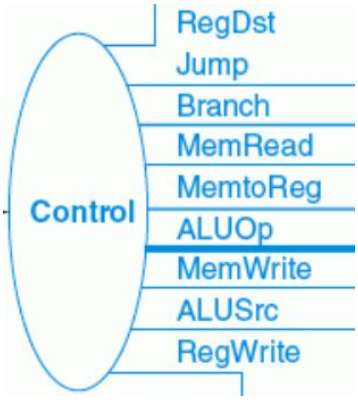
3.1.2 新建顶层模块文件，可命名 Top

3.1.3 定义信号线

为 top 模块内的每一根/组连接信号线命名，并在 top 模块中声明定义它们。

例如，主控制模块输出端口上的连线：

```
47      wire REG_DST,
48           JUMP,
49           BRANCH,
50           MEM_READ,
51           MEM_TO_REG,
52           MEM_WRITE;
53      wire[1:0] ALU_OP;
54      wire ALU_SRC,
55           REG_WRITE;
```



3.1.4 程序计数器 PC

程序计数器是这个简单 CPU 能够跑起来的关键。定义一个 32 位 reg 类型 PC，在时钟上升沿(下降沿已在 Lab4 被用作寄存器的写了)做 PC<=PC+4。

注：简单的讲，在组合逻辑中用阻塞赋值“=”，时序逻辑中用非阻塞赋值“<=”。两者综合出来的电路不一样，具体区别查阅参考书。时序逻辑和组合逻辑不要放在同一个 always 块中。

3.1.5 RESET

PC 置 0x00000000，各寄存器清零，这是 reset 要做的工作。同步或异步，边沿或电平，同学们可以自由实现。

寄存器清零，要适当修改上次实验的模块，并给模块添加 reset 信号。

注：添加 reset 要注意，写在原来“写”的 always 块中。假如新加一个 always 块，当两个“写”always 同时满足时，就将混乱不知赋什么值了。

3.1.6 模块实例化，连接模块

实例化前两次实验中编写的模块和新建的模块，实例化的过程中连接模块的端口。实例化有以下两种方法：

1. 严格按照模块定义的端口顺序来连接，不用表明原模块定义时规定的端口名：

   模块　模块名(连接端口 1 信号名，连接端口信号名 2…)

2. 在连接时用“.”符号，表明原模块是定义时规定的端口名：

   模块　模块名(.端口 1 名(信号 1), .端口 2　名(信号 2))

大家采用第 2 种实例化方法。

以主控制模块为例，以下代码实例化一个 Ctr：mainCtr。并连接其端口。INST 是定义好的指令存储器输出的连接信号，其它信号线我们在 3.1.3 中已定义。

```
100     Ctr mainCtr(
101             .opcode(INST[31:26]),
102             .regDst(REG_DST),
103             .jump(JUMP),
104             .branch(BRANCH),
105             .memRead(MEM_READ),
106             .memToReg(MEM_TO_REG),
107             .aLUOp(ALU_OP),
108             .memWrite(MEM_WRITE),
109             .aLUSrc(ALU_SRC),
110             .regWrite(REG_WRITE));
```

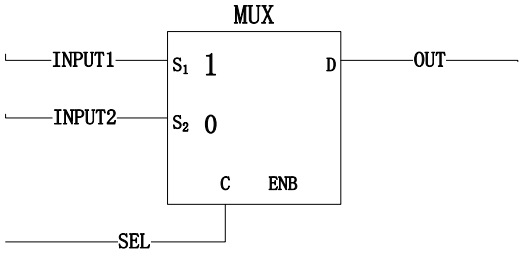
实例化 Ctr

3.1.7 连接其它信号线

1. MUX
   Mux 实现简单，一个三目运算符

   Assign　OUT = SEL ? INPUT1 : INPUT2;

   OUT, SEL, INPUT1, INPUT2 都是预先定义的信号。



2. 左移两位，用移位运算符：左移(“<<”)，右移(“>>”)
3. 加法器，直接用无符号加法运算。
   注：verilog 中寄存器类型被解释成无符号数，整数类型(integer)被解释成二进制补码形式的有符号数。因此要综合成无符号算术算符需要使用寄存器类型，而要得到有符号算术算符就需要使用整数。网线类型被解释成无符号数。

4. 与门，使用位运算符&(位与)。注意&和&&的区别。

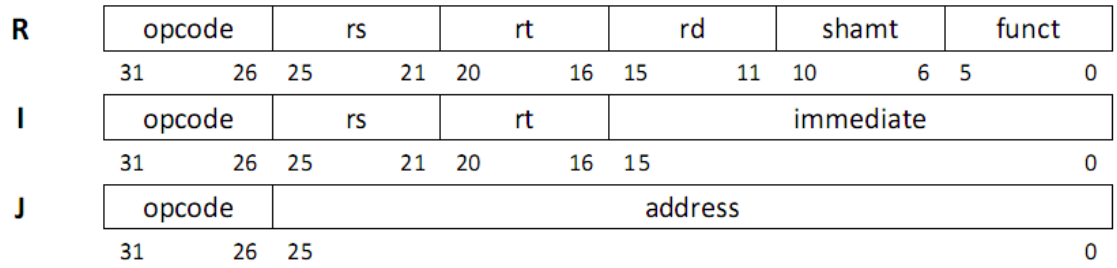
4. 仿真

1. 编写二进制测试程序

   请编写自己的测试汇编。下面提供一个简易汇编器供参考。

   一些相关的基本知识：

   指令格式：



Mips 基本指令格式

汇编格式：注意汇编中寄存器的顺序跟指令格式中的不一样

add $1,$2,$3 ： $1=$2 + $3

sub $1,$2,$3 ： $1=$2 - $3

and $1,$2,$3 ： $1=$2 & $3

or $1,$2,$3 ： $1=$2 | $3

slt $1,$2,$3 ： if($2<$3) $1=1 else $1=0

lw $1,10($2) ： $1=memory[$2+10]

sw $1,10($2) ： memory[$2+10]=$1

beq $1,$2,10 ： if($1==$2) goto PC+4+40

[10 是 PC+4 后的指令间隔数，故为 PC+4+40]

j 10000 ： goto 10000

2. 系统任务$readmemb 和$readmemh，放在 initial 初始化块中。

   Verilog 中这两个系统任务用来从文件中读取数据到存储器中，格式如下：

   $readmemh(“datafile”, memoryName); 相对路径

   $readmemh("Instruction",InstMemFile);

   $readmemh("D:/Archlab/Lab05/Lab005.srcs/Src/Data",MemFile,10'h0);绝对路径

这里给出 mem_data 和 mem_inst 两个样例文件，分别用来装入 data memory 和 instruction memory。

mem_data

```
00000000
00000001
00000002
00000003
00000004
00000005
00000006
00000007
00000008
```

… …

mem_inst

```
00001000000000000000000000000100        // j a
00000000000000000000000000000000        // nop
00000000000000000000000000000000        // nop
00000000000000000000000000000000        // nop
10001100000000010000000100010100        // lw $1, 276($0)
10001100000000010000000100001000        // lw $2, 264($0)
00000000001000100001100000100000        // add $3, $1, $2
00000000001000100010000000100010        // sub $4, $1, $2
```

… …

3. 整个处理器设计基本完成后，编辑激励文件，进行仿真测试：

   1) 初始化 data memory、instruction memory 和 register 三大存储模块，这里以初始化 instruction memory 为例说明，其它类推。如下图，Verilog 中调用了系统任务 $readmemh 将 Instruction 文件中的指令数据读到 InstMemFile 数组中。

```
29    end
```

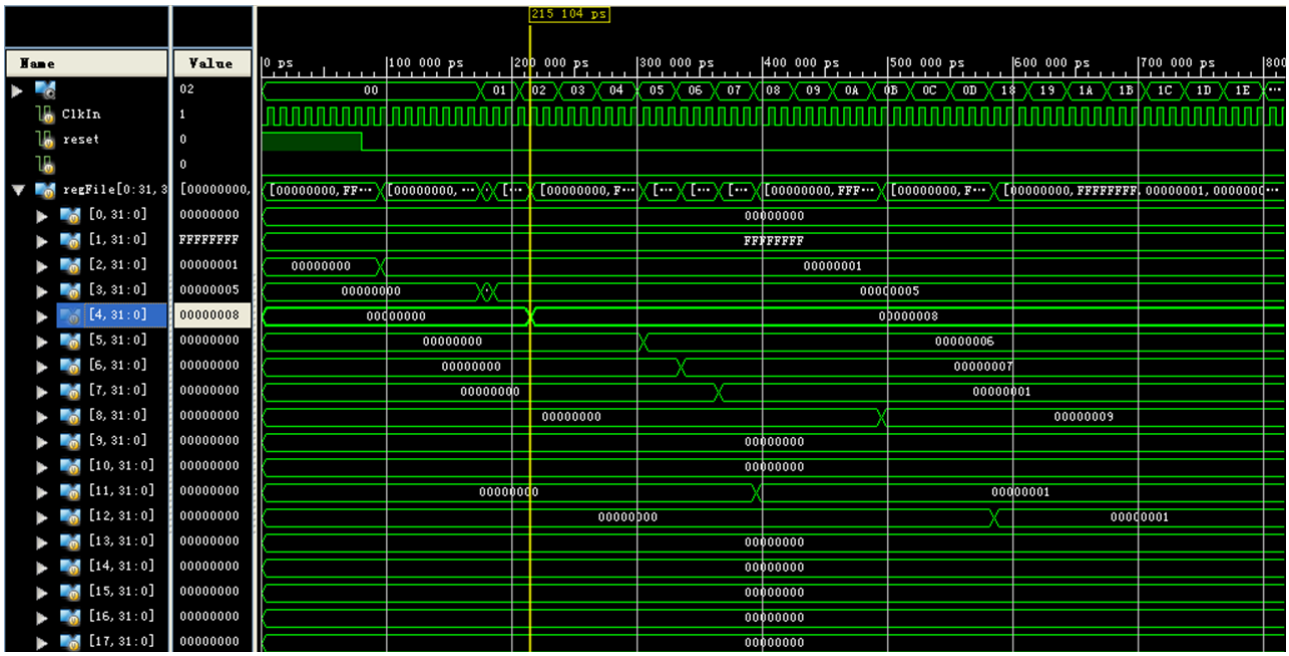
   2) 按需要添加时钟激励和其它输入信号的初始化 等。

end

3) 添加 register 模块中的 regfile 寄存器数组以及其它需要的变量或信号到仿真波形窗口,观察各个相关数值的变化情况

4) 在 Tcl Console 窗口中输入 restart 和 run 2000ns 命令，重新进行仿真。观察仿真波形与你提供的 MIPS 指令结果进行对比。

4. 下面给出一个仿真样例：



5. 上板调试

注：

1. 利用 switch、led 甚至是 pushbutton 来观察运行结果是否预期
2. 可参考 lab2 相关外设上板验证的方法

6. 实验报告