实验六 CPU综合设计

1. **实验目的**

1 掌握复杂系统设计方法。

2 深刻理解计算机系统硬件原理。

**二、实验内容**

1）设计一个基于MIPS指令集的CPU，支持以下指令：{add, sub, addi, lw, sw, beq, j, nop}；

2）CPU需要包含寄存器组、RAM模块、ALU模块、指令译码模块；

3）该CPU能运行基本的汇编指令；（D~C+）

以下为可选内容：

4）实现多周期CPU（B-~B+）；

5）实现以下高级功能之一（A-~A+）：

(1)实现5级流水线CPU；

(2)实现超标量；

(3)实现4路组相联缓存；

可基于RISC V 、ARM指令集实现。

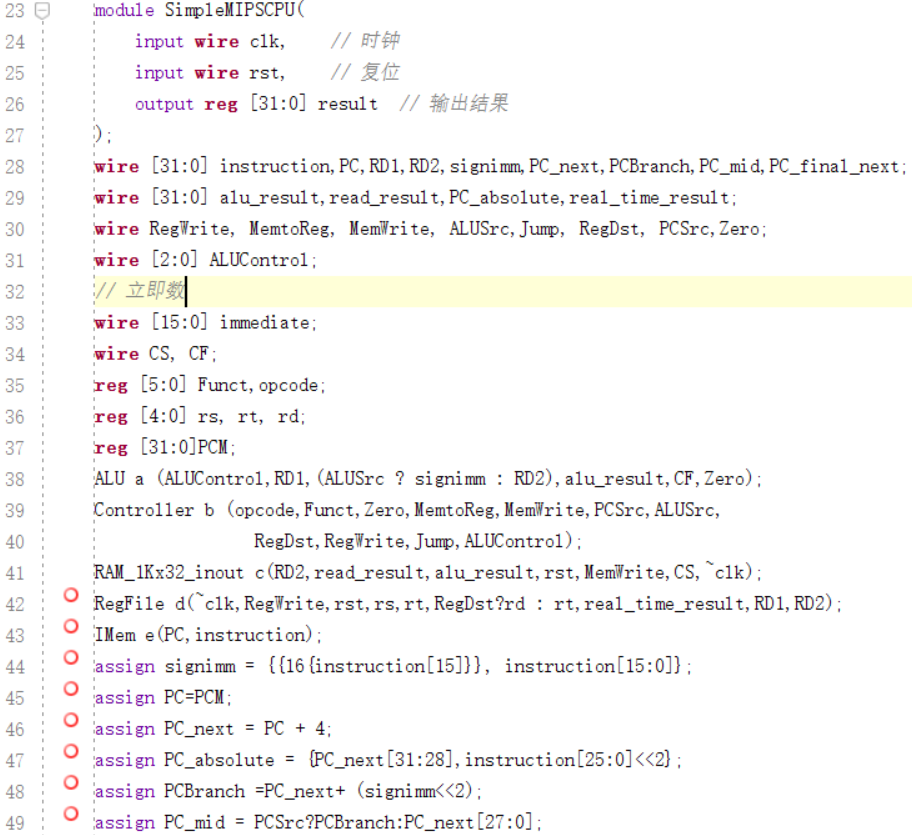
**如发现代码为抄袭代码，成绩一律按不及格处理。**

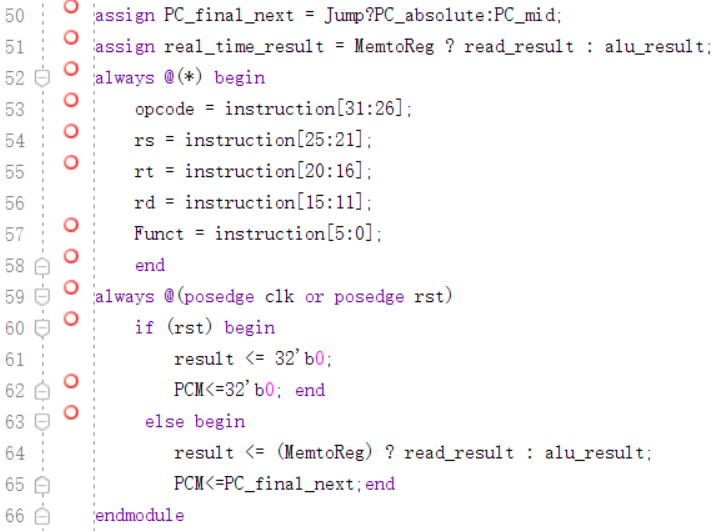
**三、实验要求**

编写相应测试程序，完成所有指令测试。

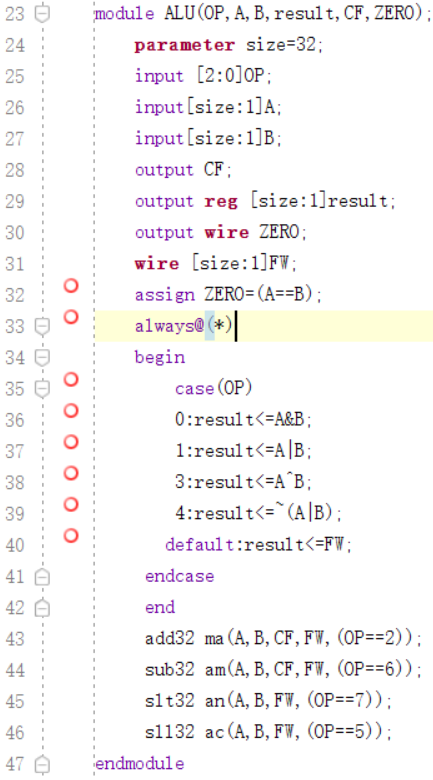
**四、实验代码及结果**

SimpleMIPSCPU模块

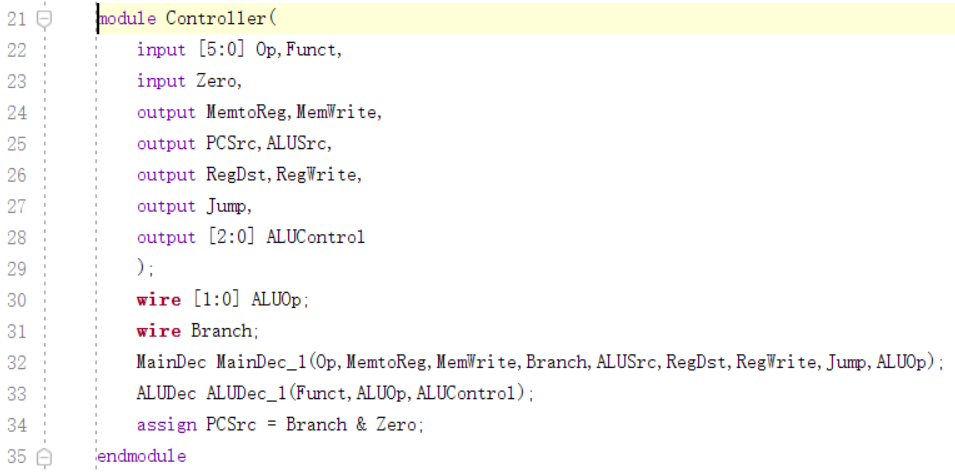




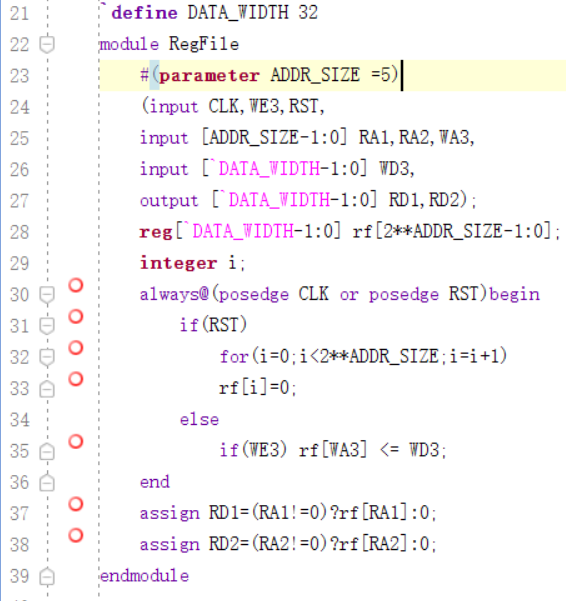
ALU模块：



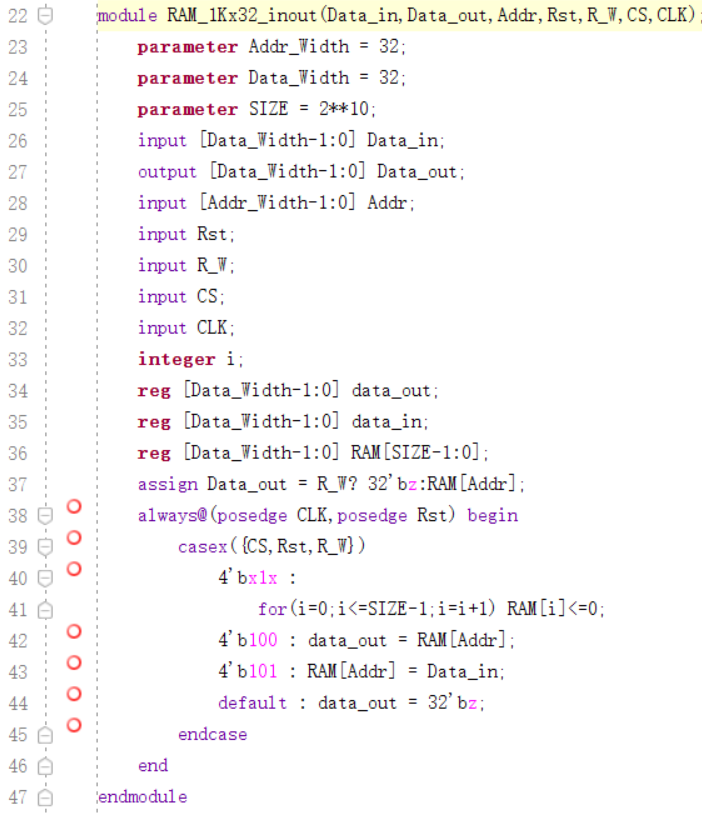
Controller模块：



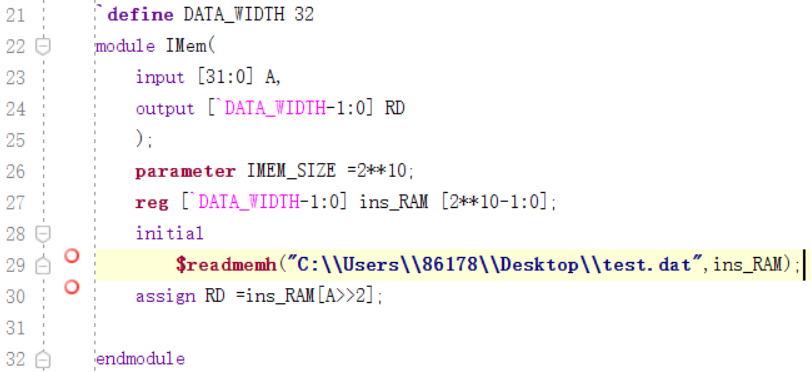
RegFile模块：



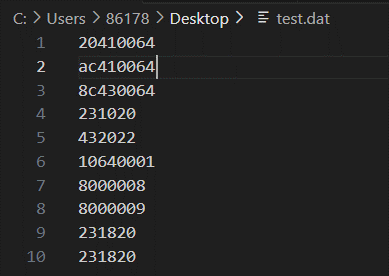
RAM\_1Kx32\_inout模块



IMem模块：



C:\\Users\\86178\\Desktop\\test.dat对应的内容为：



对应的指令为

00100000010000010000000001100110 //addi 1 2 100

10101100010000010000000001100110 //sw 1 100(2)

10001100010000110000000001100110 //lw 3 100(2)

00000000001000110001000000100000 //add 2 1 3

00000000010000110010000000100010 //sub 4 2 3

00010000011001000000000000000001 //beq 3 4 1

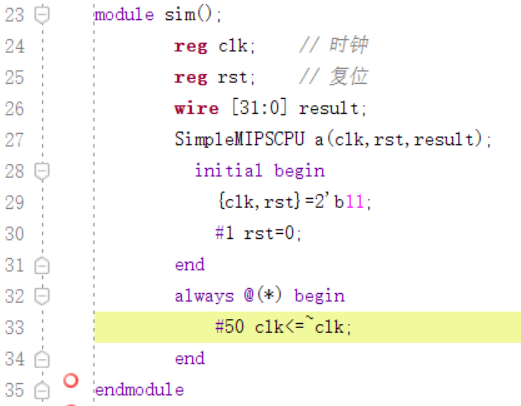
00001000000000000000000000001000 // j 8

00001000000000000000000000001001 // j 9

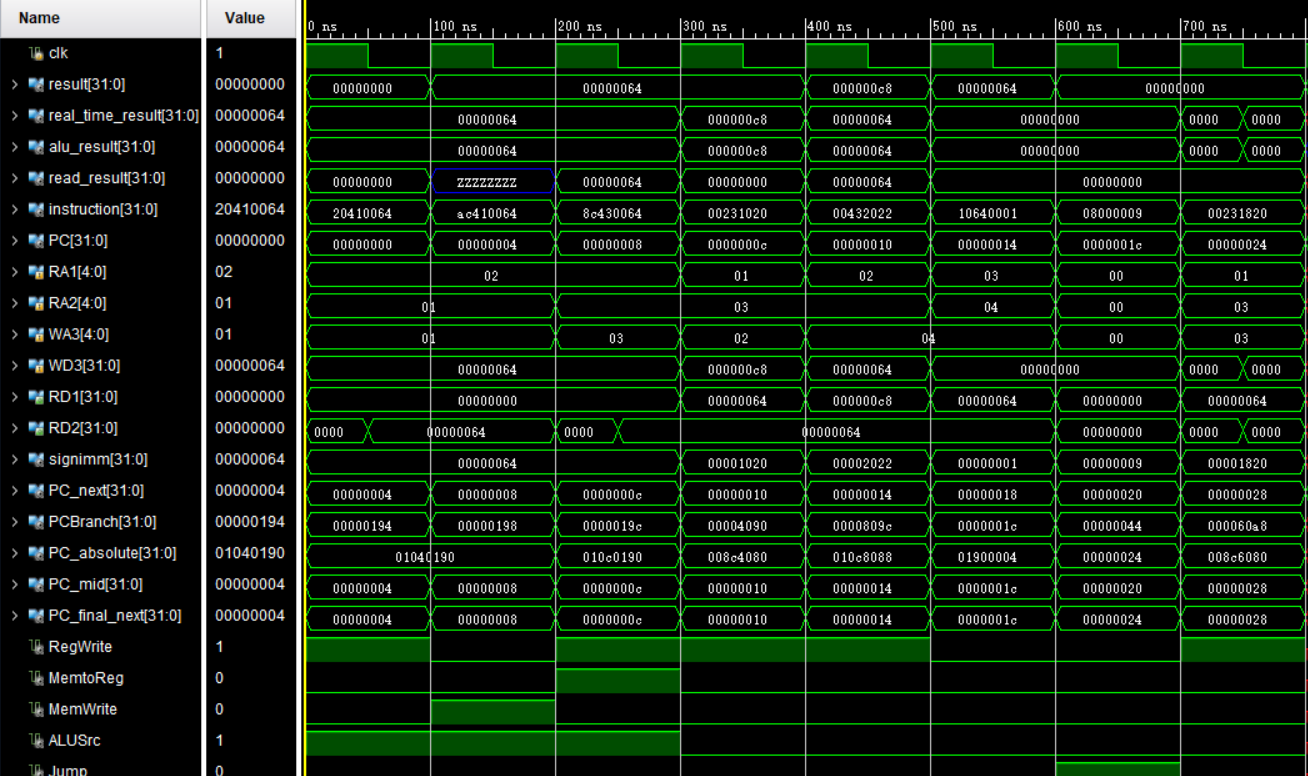
00000000001000110001000000100000 //add 3 1 3

00000000001000110001000000100000 //add 3 1 3

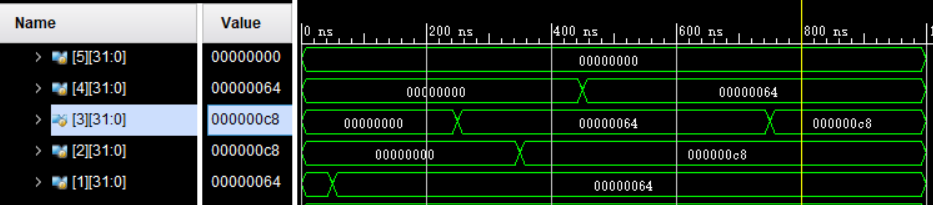
仿真文件：



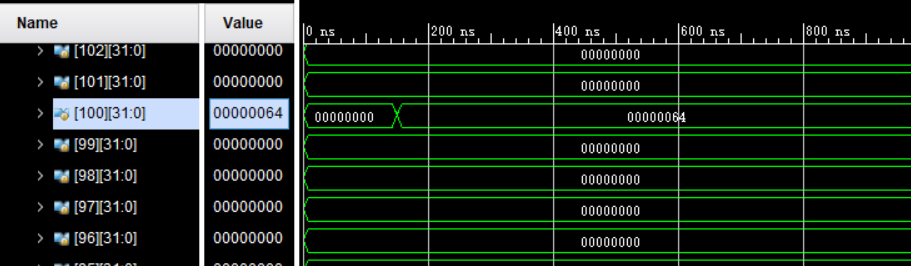
仿真图形：



其中寄存器的值：



存储器的值：



**分析：**

SimpleMIPSCPU 模块：

输入：

clk：时钟信号。

rst：复位信号。

输出：

result：32 位输出结果。

内部信号：

instruction：指令。

PC：程序计数器。

RD1、RD2：寄存器读取数据。

signimm：符号扩展的立即数。

PC\_next、PCBranch、PC\_mid、PC\_final\_next：程序计数器的各个状态。

alu\_result：ALU 计算结果。

read\_result：从 RAM 读取的数据。

PC\_absolute：绝对地址。

real\_time\_result：最终结果。

RegWrite、MemtoReg、MemWrite、ALUSrc、Jump、RegDst、PCSrc、Zero:各种控制信号。

ALUControl：ALU 控制信号。

CS、CF：进位标志和零标志。

Funct、opcode：指令中的功能码和操作码。

rs、rt、rd：指令中的寄存器地址。

PCM：用于存储程序计数器的寄存器。

实例化：

ALU 模块 (a)：用于执行 ALU 运算。

Controller 模块 (b)：用于生成控制信号。

RAM\_1Kx32\_inout 模块 (c)：用于存储数据。

RegFile 模块 (d)：寄存器文件。

IMem 模块 (e)：指令存储器。

逻辑：

分配 ALU 运算的输入和输出。

分配程序计数器的不同状态。

分配结果的选择：存储器到寄存器写入 (MemtoReg) 或 ALU 计算结果。

在时钟上升沿或复位上升沿触发的时候，更新程序计数器、指令解析等。

ALU 模块：

输入：

OP：ALU 操作码。

A、B：输入数据。

输出：

result：ALU 计算结果。

CF：进位标志。

ZERO：零标志。

逻辑：

根据 OP 选择不同的 ALU 操作。

内部实例化了其他模块 (add32、sub32、slt32、sll32)。

IMem 模块：

输入：

A：地址。

输出：

RD：读取的指令。

逻辑：

从存储器中读取指令。

RegFile 模块：

输入：

CLK：时钟。

WE3：写使能信号。

RST：复位信号。

RA1、RA2、WA3：读写寄存器地址。

WD3：写入的数据。

输出：

RD1、RD2：读取的数据。

逻辑：

根据时钟和写使能信号执行读写操作。

Controller 模块：

输入：

Op、Funct、Zero：操作码、功能码、零标志。

输出：

MemtoReg,MemWrite,PCSrc,ALUSrc,RegDst,RegWrite,Jump,ALUContro:各种控制信号。

逻辑：

实例化 MainDec 和 ALUDec 模块。分配 PCSrc 信号。

RAM\_1Kx32\_inout 模块：

输入：

Data\_in：写入的数据。

Addr：地址。

Rst：复位信号。

R\_W：读写使能信号。

CS：存储器清零信号。

CLK：时钟。

输出：

Data\_out：读取的数据。

逻辑：

根据读写使能信号执行读写操作。

总体：

这个 MIPS 处理器的 Verilog 描述是一个完整的单周期处理器。它包括指令存储器、寄存器文件、ALU 模块等组成部分，实现了 MIPS 指令集的基本功能。在时钟上升沿或复位上升沿时，根据控制信号执行相应的操作。整体上，该设计是一个基本的 MIPS 处理器实现。

由仿真图象可得CPU分别执行了0，1，2，3，4，5，7，9号指令，初始值均为0。

0号指令具体为addi 1 2 100，我们发现地址为1的寄存器单元在50ns时被更改为100；

1号指令为sw 1 100(2)，将1号寄存器的值存储在地址为100的存储器单元中，我们发现地址为100的存储器单元在150ns的时候被更改为100；

2号指令为lw 3 100(2)，将地址为100的存储器单元中的值加载到3号寄存器中，可以发现3号寄存器在250ns的时候被更改为100。

3号指令为add 2 1 3，将1号寄存器的值和3号寄存器的值相加，结果赋给2号寄存器，可以发现2号寄存器在350ns的时候被更改为200

4号指令为sub 4 2 3，将2号寄存器的值和3号寄存器的值相减，结果赋给4号寄存器，可以发现4号寄存器在450ns的时候被更改为100.

5号指令为beq 3 4 1，如果3号寄存器的值和4号寄存器的值相等，那么程序计数器的值要加上1，如果不相等，那么程序计数器的值不变。可以看到3号寄存器的值为100，4号寄存器的值也为100，所以3号寄存器的值和4号寄存器的值相等，此时程序计数器指向6号指令，在此基础上加1，于是程序计数器指向了7号指令。

7号指令为j 9，具体含义为执行次指令时，程序计数器无条件跳转到9号指令。

9号指令为add 3 1 3，具体含义为将1号寄存器的值和3号寄存器的值相加，结果赋给3号寄存器，可以发现3号寄存器的值在750ns的时候被更改为200.(如果之前5号指令和7号指令之中有一个出现问题，那么CPU会先执行8号指令（add 3 1 3）再执行9号指令，这样会导致在800ns的时候3号寄存器的值为300而不是200)

其余变量的值也完全符合预期结果。

这个 MIPS 处理器的 Verilog 描述是一个完整的单周期处理器。它包括指令存储器、寄存器文件、ALU 模块等组成部分，实现了 MIPS 指令集的基本功能。在时钟上升沿或复位上升沿时，根据控制信号执行相应的操作。整体上，该设计是一个基本的 MIPS 处理器实现。

**五、调试和心得体会**

我学会了通过合理的硬件抽象层次，将整个系统划分为模块，每个模块负责一个特定的功能。这样做有助于提高代码的可维护性和可读性，给信号、模块和操作起有意义的名字，并添加清晰的注释，以便团队成员或将来的维护者能够理解代码的目的和逻辑。我意识到了硬件设计与软件开发有很大的不同，需要考虑时序等方面的问题，理解和遵守硬件约束对于确保设计的正确性至关重要。将设计分解成合适的抽象层次，确保在高层次上有清晰的结构，同时低层次上有足够的细节。在设计的早期阶段使用仿真工具验证代码的正确性。这可以帮助设计者更早发现潜在的问题，并减少在实际硬件上进行调试的时间。变量和模块的命名应当清晰、具有描述性，而且注释应该解释代码的目的和关键步骤。这有助于他人理解代码，也为日后的维护提供支持。我学会了代码复用，在 ALU 模块中使用了其他模块，如 add32、sub32 等。这种代码复用有助于提高代码的可维护性和可扩展性。