**简单的类MIPS单周期处理器实现——整体调试**

**目的**

完成单周期类MIPS处理器。

**设计思路**

考虑到一些指令比较复杂，实验中选取了以下指令实现：add, sub, and, or, andi, ori, slt, lw, sw, beq, j, sll, srl。

结合既有设计和实际情况，设计以下指令表：

Add, sub, and, or, slt, lw, sw, beq, j与指导书的定义相同；

andi rs, rt, immediate 000011 rs rt immediate

ori rs, rt, immediate 000101 rs rt immediate

addi rs, rt, immediate 000001 rs rt immediate

sll rs, rt, rd 000000 rt rd rs shamt(00000) 000100

srl rs, rt, rd 000000 rt rd rs shamt(00000) 000101

基本思路为将此前各模块整合。但是进行了一些改进：

为实现立即数操作，考虑到含有立即数的指令低16位为立即数，不可能加入func码，故在指令操作码最高6位opcode中加入3种指令000001（加立即数）、000011（与立即数）、000101（或立即数），并使得ALUCtr考虑opcode；

为减少连线，将算术-逻辑单元(ALU)及其控制单元(ALUCtr)封装为一个新模块ALUComp；

为减少连线，将程序计数器(PC)、无条件跳转(beq)、条件跳转(j)、和指令存储器封装为一个新模块InstMemory；

为使模块更清晰，加入新模块32位MUX和5位MUX；

考虑到寄存器、内存仅在时钟变化时进行操作，存在时钟上升、下降沿不够的问题，采用双时钟，其中快时钟控制寄存器和数据内存，慢时钟控制指令内存，快、慢时钟频率之比为3:1。

**模块描述**

在主控制模块加入立即数操作指令解析，即对000001、000011、000101的解析，产生的控制信号与R-type指令基本相同，但需令RegDst=0（目标寄存器的编码在25至21位），ALUSrc=1（从符号位扩展获取操作数）。

在ALUComp的ALU模块中加入左移、右移运算，其中左移的指令码为0100，右移为0101。

在ALUComp的ALU控制模块中加入对指令操作码最高6位opcode的解析，在opcode为000001、000011和000101的情况下直接对ALU输出对应的指令。加入对func为1100（左移），1110（右移）的解析，当遇到R-type指令，且func为1100或1110时对ALU输出对应的指令。

**仿真描述**

预先在数据存储器的0号至7号地址分别写入1至8，共8个数据；在指令存储器内写入指令如下：

00001000000000000000000000000100 j, 4

00000000000000000000000000000000 nop

000000 00000000000000000000000000 nop

000000 00000000000000000000000000 nop

100011 00000 00001 0000000000000100 lw $1, 4($0)

100011 00000 00010 0000000000000000 lw $2, 0($0)

000000 00001 00010 00011 00000100000 add $3, $1, $2

000000 00001 00010 00100 00000100010 sub $4, $1, $2

000000 00001 00010 00101 00000100100 and $5, $1, $2

000000 00001 00010 00110 00000100101 or $6, $1, $2

000000 00001 00010 00111 00000101010 slt $7, $1, $2

000000 00001 00010 01111 00000101100 sll $15, $1, $2

000000 00001 00010 01000 00000101110 srl $8, $1, $2

000001 00001 10100 0000000000000001 addi $20, $1, 1

000011 01000 10011 0000000000001111 andi $19 $8, 15

000101 00001 10010 0000000000111101 ori $18, $1, 61

101011 00000 00000 0000000000000010 sw $0, 2($0)

000010 00000000000000000000000100 j 0

若正常运行，则每个周期结束时结果如下（值为16进制表示）：

$1 5

$2 1

$3 6

$4 4

$5 1

$6 5

$7 0

$8 2

$15 a

$20 6

$19 2

$18 3d

Mem[2] 0

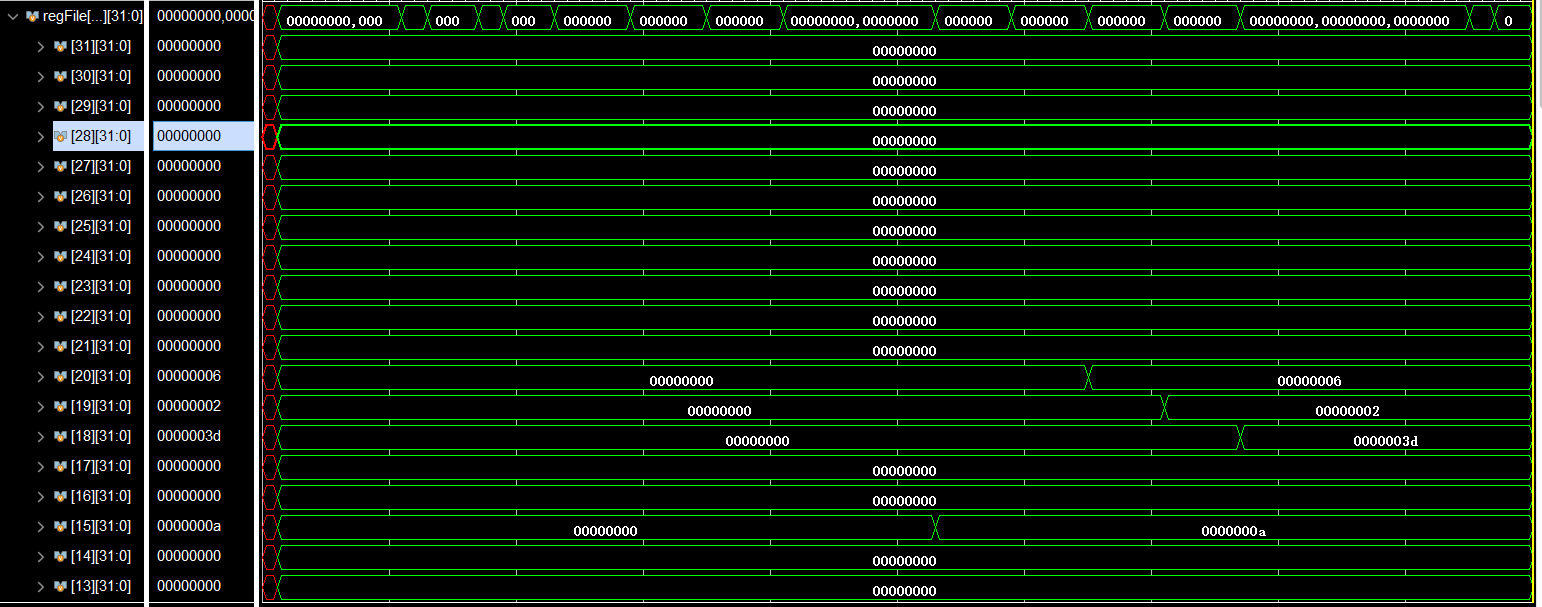
**上板验证描述**

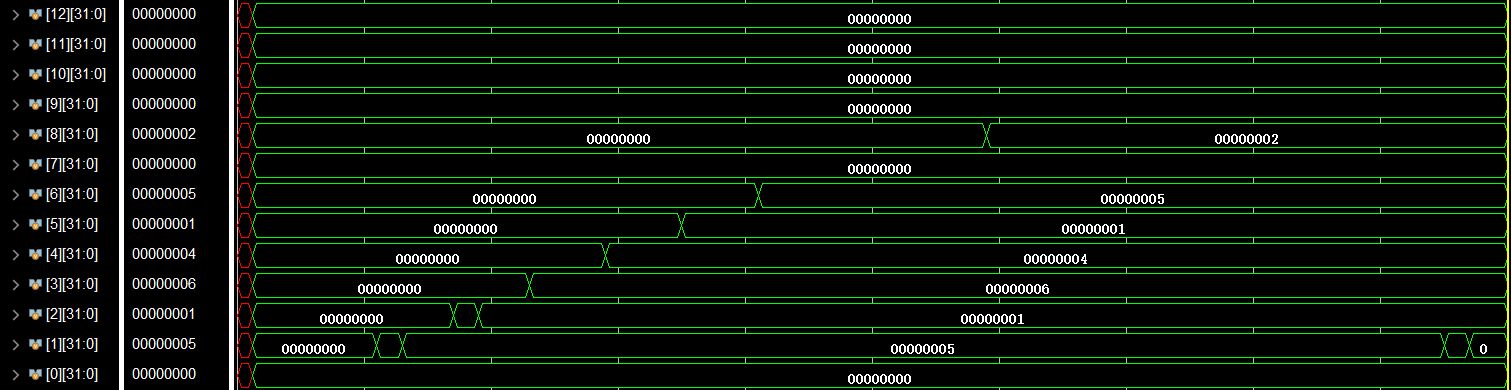
由于时间原因，未能完成上板验证。

**实验结果**

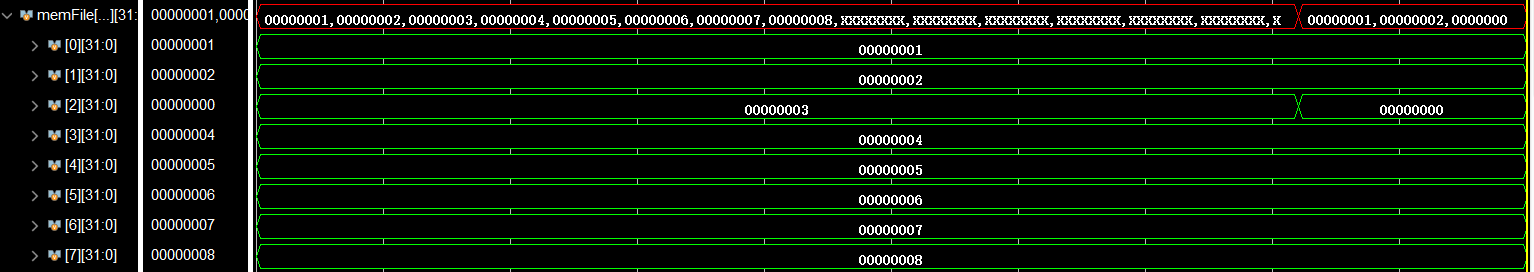
仿真波形如下所示（由于软件显示问题，显示的值可能并非完整值）。

寄存器波形：





数据存储器波形：



通过分析相应仿真波形，结果正确。

**总结**

本次实验难点在于top模块的接线命名问题和多个操作在同一周期内执行时的顺序问题。

对于top模块接线命名问题，可采取的方法是在设计图上标出对wire的命名。

模块顺序问题较为复杂。不同模块间不可避免地存在数据依赖。如果同一周期内指令运行顺序出错，例如在计算得到写入内存地址之前将内存写入，则会造成写入地址错误。因此需要通过合理设计每个模块的执行条件，控制确保模块运行顺序的正确。我采用的方法是用多个时钟执行操作，由慢时钟控制指令发布，快时钟的上升、下降沿用于读写数据，以实现对模块运行顺序的控制。