 ****

**产**

**品**

**说**

**明**

**多时钟实验报告**

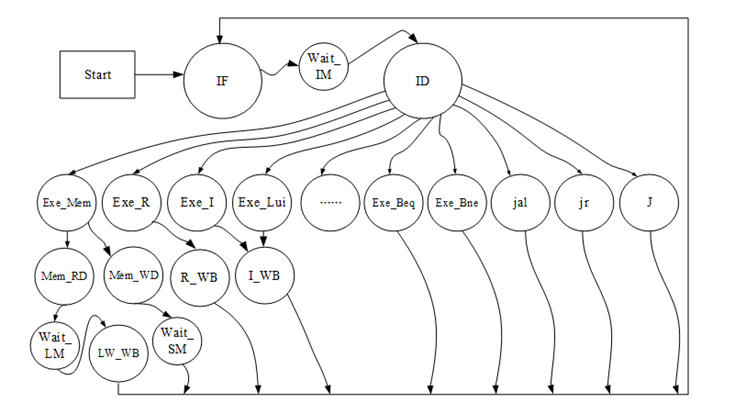
1. **原理阐述**

多时钟CPU是建立在单时钟CPU基础上，可以说是在单时钟的基础上建立的。对于单时钟CPU处理器，执行每条指令时所用的周期大于所需时间。执行指令时，也会使用到无关的数据通路部分，这就类似木桶原理，其所需时间取决于执行时间最长的一条指令（一个周期只执行一条指令），浪费了大量时间。

而多时钟周期把每条指令执行拆分成不同的阶段，使得一个周期的时间大大减少，每个周期只执行一个操作，即各阶段内最多完成一次访存或一次寄存器读写或一次 ALU操作。对于不同的指令，所需的操作不同，因此指令的周期数不同，比如ADD需要4个周期，SW和LW需要5个周期，最终可使CPU时钟的周期也可大大缩短，而不是取决于最长的指令执行。而与单时钟周期不同，一个指令需要多个时钟周期完成，因此各元件处理后的结果需要储存到寄存器内，以便下一个状态使用。

设计到状态，我们采用状态机处理。同时由于状态较多且复杂，我们用微处理器设计思路，将控制信号串联起来并储存在元件中，当输入指令，根据指令的opcode和func字段寻找对应的控制信号群。

总的来说，多时钟极大的提升了计算机的效率。



**二、实现模块**

本次实验共涉及三种类型的 MIPS 指令，分别为 R 型、I 型和 J 型，共三种类型，一共22条指令。

**R型指令**

|  |  |  |
| --- | --- | --- |
| 指令 | 指令格式 | 执行 |
| ADD | **Add rd, rs, rt** | **rd=rs+rt** |
| SUB | **Sub rd,rs,rt** | **Rd=rs-rt** |
| SLT | **Slt rd,rs,rt** | **rd=(rs<rt)?1:0** |
| SLTu | **Sltu rd,rs,rt** | **rd=((long)rs<(long)rt)?1:0** |
| AND | **And rd,rs,rt** | **rd=rs&rt** |
| OR | **Or rd,rs,rt** | **rd=rs|rt** |
| XOR | **Xor rd,rs,rt** | **rd=rs^rt** |
| NOR | **Nor rd,rs,rt** | **rd=!(rs|rt)** |
| SRL | **Srl rd,rs,sa** | **Rd=rs>>sa** |
| JR | **Jr rs** | **PC=rs** |

**I型指令**

|  |  |  |
| --- | --- | --- |
| 指令 | 指令格式 | 执行 |
| ADDi | **Addi rt, rs, dat** | **rt=rs+dat** |
| SLTi | **Slti rt,rs,dat** | **rt=(rs<dat)?1:0** |
| ANDi | **Andi rt,rs,dat** | **rt=rs&dat** |
| ORi | **Ori rt,rs,dat** | **rt=rs|dat** |
| XORi | **Xori rt,rs,dat** | **rt=rs^dat** |
| NORi | **Nori rt,rs,dat** | **rt=!(rs|dat)** |
| LW | **Lw rt, dat(rs)** | **按地址取值** |
| SW | **Sw rt,dat(rs)** | **按地址存值** |
| LUi | **Lui rt,dat** | **Rt=dat<<16** |
| BEQ | **Beq rs,rt,ofs** | **If(rs==rt) PC+=ofs** |
| BNE | **Bne rs,rt,ofs** | **If(rs!=rt) PC+=ofs** |

**J型**

|  |  |  |
| --- | --- | --- |
| 指令 | 指令格式 | 执行 |
| J | **J adr** | **---** |
| JAL | **Jal adr** | **---** |

**主要模块说明**

|  |  |
| --- | --- |
| 模块名称 | 作用 |
| Control | **状态机+产生控制信号** |
| ALU | **运算器（ADD,SLT等）** |
| REG32，Regs | **寄存器（A,B,IR等）** |
| Extension | **位拓展** |
| Seg7Device | **显示模块** |
| RAM | **存储器** |

**三、使用方法**

用户可以通过汇编器将汇编码转成机器码，转换成coe文件利用ISE转换成ip核的ram，就可以进行运行。（也可以直接使用程序自带的测试代码进行测试，测试代码如下，这段代码将会从1加至20）

and $t0,$t0,$zero

and $t1,$t1,$zero

and $t2,$t2,$zero

addi $t2,$t2,21

slt $t3,$t0,$t2

beq $t3,$zero,3

add $t1,$t1,$t0

addi $t0,$t0,1

j 5

将生成的bit文件加载进入，进行硬件测试。

本实验显示模块分为两块，一是上面4位数显管，另外是下面的8位数显管。主要使用到的开关：

**SW[15:13]:**

3’b000 :显示32’hFFFFFFFF

3’b001 :显示指令

3’b110: 显示当前PC

3’b010: 显示ram中读出的数据（指令/数值）

3’b100: 下一步的PC

3’b101: 下一步状态

3’b111: 切换显存模式（可以查看内存内容）

**SW[7]:**

可以切换为单步调试