# P3-Logisim单周期CPU设计文档

## 一、设计说明

使用 Logisim 开发一个简单的 MIPS 单周期处理器，设计说明如下：

1. 处理器为32位处理器。
2. 处理器应支持的指令集为：{addu, subu, ori, lw, sw, beq, lui, nop}。
3. nop机器码为0x00000000。
4. addu,subu可以不支持溢出。
5. 处理器为单周期设计。

## 二、模块规格

### 1. IFU

#### (1) **端口说明**

表1 IFU端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号 | 方向 | 描述 |
| 1 | Clk | I | 时钟信号 |
| 2 | Reset | I | 异步复位信号，将PC的值设置为0x00000000  1：复位  0：无效 |
| 3 | Branch | I | 判断当前指令是否为beq指令  1：是  0：不是 |
| 4 | Zero | I | 判断PC是否满足beq指令跳转条件  1：满足  0：不满足 |
| 5 | Offset [15:0] | I | 16位立即数偏移量 |
| 6 | Instr [31:0] | O | 32位IM取出指令 |

#### (2)功能定义

表2 IFU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 复位 | Reset有效时，PC被设置为0x00000000 |
| 2 | 计算PC的下一个值 | 当Branch信号和Zero信号均有效时，否则 |
| 3 | 指令输出 | 根据PC值取出32位指令 |

### 2. GRF

#### (1) **端口说明**

表3 GRF端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号 | 方向 | 描述 |
| 1 | Clk | I | 时钟信号 |
| 2 | Reset | I | 异步复位信号，将32个寄存器中的值全部清零  1：复位  0：无效 |
| 3 | A1 [4:0] | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出到RD1 |
| 4 | A2 [4:0] | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出到RD2 |
| 5 | A3 [4:0] | I | 5位地址输入信号，指定32个寄存器中的一个作为写入的目标寄存器 |
| 6 | WD [31:0] | I | 32位写入数据 |
| 7 | WE | I | 写使能信号  1：可向GRF中写入数据  0：不能向GRF中写入数据 |
| 8 | RD1 [31:0] | O | 输出A1指定的寄存器的32位数据 |
| 9 | RD2 [31:0] | O | 输出A2指定的寄存器的32位数据 |

#### (2)功能定义

表4 GRF功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 复位 | Reset有效时，将32个寄存器中的值全部清零 |
| 2 | 读数据 | 读出A1, A2地址对应寄存器中所存储数据到RD1, RD2 |
| 3 | 写数据 | 当WE有效且时钟上升沿来临时，将WD写入A3所对应的寄存器中 |

### 3. ALU

#### (1) **端口说明**

表5 ALU端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号 | 方向 | 描述 |
| 1 | A [31:0] | I | 参与ALU计算的第一个32位数据 |
| 2 | B [31:0] | I | 参与ALU计算的第二个32位数据 |
| 3 | ALUOpe [2:0] | I | ALU功能的选择信号：  000：ALU进行与运算  001：ALU进行或运算  010：ALU进行加法运算  011：ALU进行减法运算 |
| 4 | C[31:0] | O | ALU的计算结果 |

#### (2)功能定义

表6 ALU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 与运算 |  |
| 2 | 或运算 |  |
| 3 | 加运算 |  |
| 4 | 减运算 |  |

### 4. DM

#### (1) **端口说明**

表7 DM端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号 | 方向 | 描述 |
| 1 | Clk | I | 时钟信号 |
| 2 | Reset | I | 异步复位信号，将DM中的值全部清零  1：复位  0：无效 |
| 3 | MemRead | I | 写使能信号  1：可向DM中写入数据  0：无效 |
| 4 | MemWrite | I | 读使能信号  1：可读取DM中数据  0：无效 |
| 5 | Address [4:0] | I | 5位地址输入信号，对DM指定地址进行读写操作 |
| 6 | WriteData [31:0] | I | 32位写入数据 |
| 7 | ReadData [31:0] | O | 32位输出数据 |

#### (2)功能定义

表8 DM功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 复位 | Reset有效时，将DM中的值全部清零 |
| 2 | 读操作 | 读出Address地址对应存储数据到ReadData |
| 3 | 写操作 | 当时钟上升沿来临时，将WriteData写入Address地址对应位置 |

### 5. EXT

#### (1) **端口说明**

表9 EXT端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号 | 方向 | 描述 |
| 1 | Imm16 [15:0] | I | 16位待扩展数据 |
| 2 | Imm32 [31:0] | O | 32位输出数据 |
| 3 | EXTop | I | 扩展操作信号  0：进行无符号扩展  1：进行符号扩展 |

#### (2)功能定义

表10 EXT功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 无符号扩展 | 将16位立即数Imm16无符号拓展至32位输出Imm32 |
| 2 | 符号扩展 | 将16位立即数Imm16符号拓展至32位输出Imm32 |

### 6. Controller

见“三、控制器设计”章节

## 三、控制器设计

在设计单周期CPU时我参考了牛建伟老师的作业图——

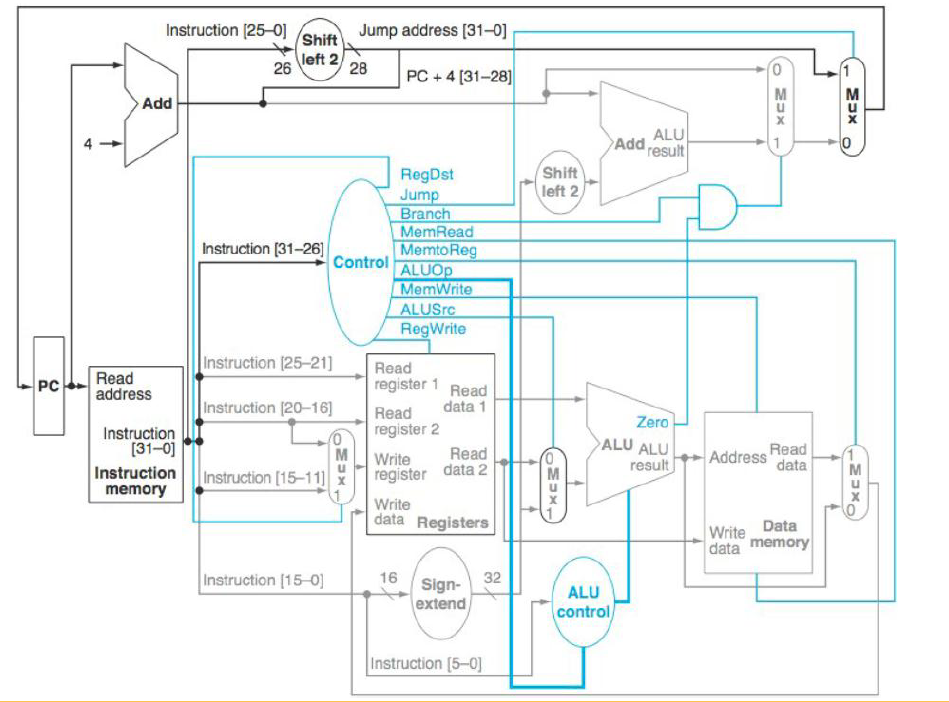


图1 参考CPU电路图

将控制器部分分为Control模块和ALU Control模块，具体说明如下：

### 1. Control模块**端口与功能说明**

表11 Control模块端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号 | 方向 | 描述 |
| 1 | Op [5:0] | I | 6位控制信号 |
| 2 | RegDst | O | 0：将Rt作为GRF的A3写入地址  1：将Rd作为GRF的A3写入地址 |
| 3 | ALUSrc | O | 0：将RD2作为参与ALU计算的第二个数据  1：将EXT输出作为参与ALU计算的第二个数据 |
| 4 | MemToReg [1:0] | O | 00：将ALU计算结果写入GRF  01：将DM输出值写入GRF  10：将填充到高位的 16位立即数写入GRF |
| 5 | RegWrite | O | 0：GRF写使能信号无效  1：GRF写使能信号有效 |
| 6 | MemRead | O | 0：DM读使能信号无效  1：DM读使能信号有效 |
| 7 | MemWrite | O | 0：DM写使能信号无效  1：DM写使能信号有效 |
| 8 | Branch | O | 0：当前指令不是beq指令  1：当前指令为beq指令 |
| 9 | ExtOp | O | 0：进行无符号扩展  1：进行符号扩展 |
| 10 | ALUOp [2:0] | O | 000：R型指令由功能码字段决定  001：lw/sw指令所用加法  010：beq指令所用减法  011：ori指令所用或运算 |

### 2. Control模块**真值表**

表12 Control模块**真值表**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Op字段 | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 |
| 指令 | addu(R) | subu(R) | ori | lw | sw | beq | lui |
| RegDst | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| ALUSrc | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| MemToReg [1:0] | 00 | 00 | 00 | 01 | 00 | 00 | 10 |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| MemRead | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Branch | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ExtOp | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| ALUOp [2:0] | 000 | 000 | 111 | 001 | 001 | 010 | 011 |

### 3. ALU Control模块**真值表**

表13 ALU Control模块**真值表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令 | Func字段 | ALUOp | ALU运算类型 | ALUOpe |
| addu | 100001 | 000 | 加 | 010 |
| subu | 100011 | 000 | 减 | 011 |
| ori | xxxxxx | 011 | 或 | 001 |
| lw | xxxxxx | 001 | 加 | 010 |
| sw | xxxxxx | 001 | 加 | 010 |
| beq | xxxxxx | 010 | 减 | 011 |
| lui | xxxxxx | xxx | 无 | 无 |
| nop | xxxxxx | xxx | 无 | 无 |

## 四、测试程序

### 1. 测试程序代码

1. # 测试程序代码：
2. .text
3. # ori指令测试
4. ori     $s0,$zero,15
5. ori     $s1,$zero,31
7. # addu指令测试
8. addu    $s2,$s0,$s1
9. addu    $s3,$s0,$s2
11. # subu指令测试
12. subu    $s4,$s3,$s2
13. subu    $s5,$s2,$s1
15. # lui指令测试
16. lui     $s7,2333
18. # sw指令测试
19. sw  $s0,0($zero)
20. sw  $s1,4($zero)
22. # lw指令测试
23. lw  $t0,0($zero)
24. lw  $t1,4($zero)
26. # beq指令测试
27. beq     $s0,$s1,equal1
28. ori $t2,$zero,1
30. equal1:
31. beq     $s1,$s1,equal2
32. ori $t3,$zero,2
33. equal2:
34. ori $t4,$zero,666

### 2. 测试程序结果

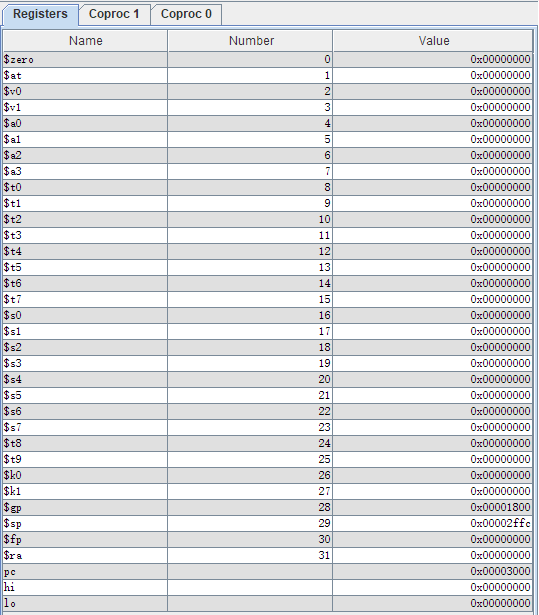


图2 测试程序结果(GRF)

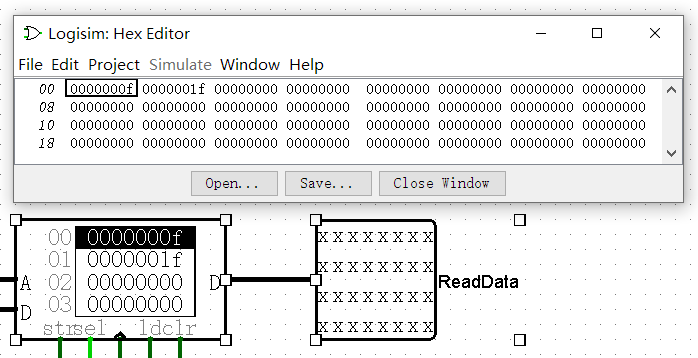


图3 测试程序结果(DM)

## 五、思考题

### 1. 模块规格（L0.T2）

#### (1) 若PC（程序计数器）位数为30位，试分析其与32位PC的优劣。

优势：30位PC所用硬件资源更少

劣势：30位PC进行jr等指令时需要对PC进行扩展，增加了CPU设计的复杂度。

#### (2) 现在我们的模块中 IM使用ROM， DM使用RAM， GRF使用寄存器，这种做法合理吗？ 请给出分析，若有改进意见也请一并给出。

合理，分析如下：

1. ROM为只读存储器，而IM指令寄存器在CPU运行过程中只进行读取操作，故IM使用ROM合理。
2. DM需要进行读写操作，对速度要求不高。而RAM和寄存器都能实现读写操作，但DM对内存空间需求较大，若采用寄存器成本较高，因此选择读写速度较慢但存储空间大的RAM合理。
3. GRF需要进行读写操作，使用频率高，对速度要求高。使用寄存器搭建GRF合理。

### 2. 控制器设计（L0.T3）

#### (1)结合上文给出的样例真值表，给出RegDst， ALUSrc， MemtoReg，RegWrite, nPC\_Sel, ExtOp与op和func有关的布尔表达式（表达式中只能使用“与、或、非”3 种基本逻辑运算。）

#### (2)充分利用真值表中的 X 可以将以上控制信号化简为最简单的表达式， 请给出化简后的形式。

#### (3)事实上，实现nop空指令，我们并不需要将它加入控制信号真值表，为什么？请给出你的理由。

nop的指令为0x00000000，仅进行PC+4操作，不对电路其它部分产生影响。

### 3. 测试CPU（L0.T4）

#### (1) 前文提到，“可能需要手工修改指令码中的数据偏移”，但实际上只需再增加一个 DM片选信号,就可以解决这个问题。请阅读相关资料并设计一个 DM 改造方案使得无需手工修改数据偏移。

假设DM有256MB容量，并且映射在0x3000\_0000-0x3FFF\_FFFF区间，那么只需要把高4位地址与0x3进行比较，结果就是DM的片选信号。[[1]](#footnote-1)

#### (2) 除了编写程序进行测试外，还有一种验证CPU设计正确性的办法——形式验证。 形式验证的含义是根据某个或某些形式规范或属性，使用数学的方法证明其正确性或非正确性。请搜索“形式验证（Formal Verification)"了解相关内容后，简要阐述相比与测试，形式验证的优劣。

形式验证的优点如下：

(1)形式验证是对指定描述的所有可能的情况进行验证，覆盖率达到了100%。

(2)形式验证技术是借用数学上的方法将待验证电路和功能描述或参考设计直接进行比较，不需要开发测试激励。

(3)形式验证的验证时间短，可以很快发现和改正电路设计中的错误，可以缩短设计周期。

形式验证的缺点如下：

形式验证只能检验电路设计的正确性，却无法检验其它方面如电路能耗等的优劣。[[2]](#footnote-2)

1. 参考百度文库《Project3\_Logisim完成单周期处理器开发》一文 [↑](#footnote-ref-1)
2. 参考百度百科“形式验证”词条 [↑](#footnote-ref-2)