

**2021~2022学年 第二学期**

**《系 统 硬 件 综 合 设 计》**

设 计 报 告

班 级 计科2-4班 学 号 2020218037

姓 名 曲艺 成 绩 \_\_\_\_\_\_\_\_\_\_\_\_\_

2022年 12 月28

**目录**

[1 课程设计概述 3](#_Toc123838124)

[1.1 设计目的 3](#_Toc123838125)

[1.2 设计要求 3](#_Toc123838126)

[1.3 设计工具 3](#_Toc123838127)

[2 总体方案设计 3](#_Toc123838128)

[2.1 MIPS体系结构 3](#_Toc123838129)

[2.1.1 MIPS寄存器 4](#_Toc123838130)

[2.1.2 MIPS指令集 5](#_Toc123838131)

[2.1.3 MIPS五级流水线 6](#_Toc123838132)

[2.2 设计环境 6](#_Toc123838133)

[2.1.1 软件设计环境 6](#_Toc123838134)

[2.1.2 硬件设计环境 7](#_Toc123838135)

[2.3 设计思路 7](#_Toc123838136)

[2.3.1 五级流水线设计 7](#_Toc123838137)

[2.3.2 数据前推设计 8](#_Toc123838138)

[2.3.3 硬件延时模块设计 9](#_Toc123838139)

[3 设计实现 9](#_Toc123838140)

[3.1 五级流水线各模块设计 9](#_Toc123838141)

[3.1.1 IF段 9](#_Toc123838142)

[3.1.2 ID段 10](#_Toc123838143)

[3.1.3 EX段 11](#_Toc123838144)

[3.1.4 MEM段 13](#_Toc123838145)

[3.1.5 WB段 13](#_Toc123838146)

[3.2 数据相关解决方法 14](#_Toc123838147)

[4 实验设计与验证 15](#_Toc123838148)

[4.1 LED流水灯实验的设计 15](#_Toc123838149)

[4.2 CPU支持指令的仿真验证 17](#_Toc123838150)

[4.2.1 “五级流水线”&“逻辑运算”仿真验证 17](#_Toc123838151)

[4.2.2 数据相关仿真验证 18](#_Toc123838152)

[4.3 遇到的问题与解决方案 18](#_Toc123838153)

[5 总结 19](#_Toc123838154)

[6 参考文献 20](#_Toc123838155)

# 1 课程设计概述

## 1.1 设计目的

基于先修课程，根据系统设计思想，使用硬件描述语言设计实现一款基于MIPS32，ARM，RISC-V或者自定义指令集的微处理器（CPU）。

## 1.2 设计要求

完成单周期CPU设计，或多周期CPU设计，或5级流水线CPU设计（递进式、难度依次提升。所有学生必须至少完成单周期CPU的设计工作），并将设计的CPU下载至FPGA开发板（ego-1）上运行。以此贯穿数字逻辑、计算机组成原理、计算机体系结构课程，实现从逻辑门至完整CPU处理器的设计[1]。

## 1.3 设计工具

语言：Verlog语言

仿真软件：Vivado

开发板：FPGA EG01开发板

# 2 总体方案设计

## 2.1 MIPS体系结构

MIPS是RISC处理器，是一种无内部互锁流水线的未处理器。

我们在设计的时候完成了五级流水线的设计，并且使用数据前推解决的数据相关问题。

基础五级流水线的设计，以下图为参考进行设计和实现：

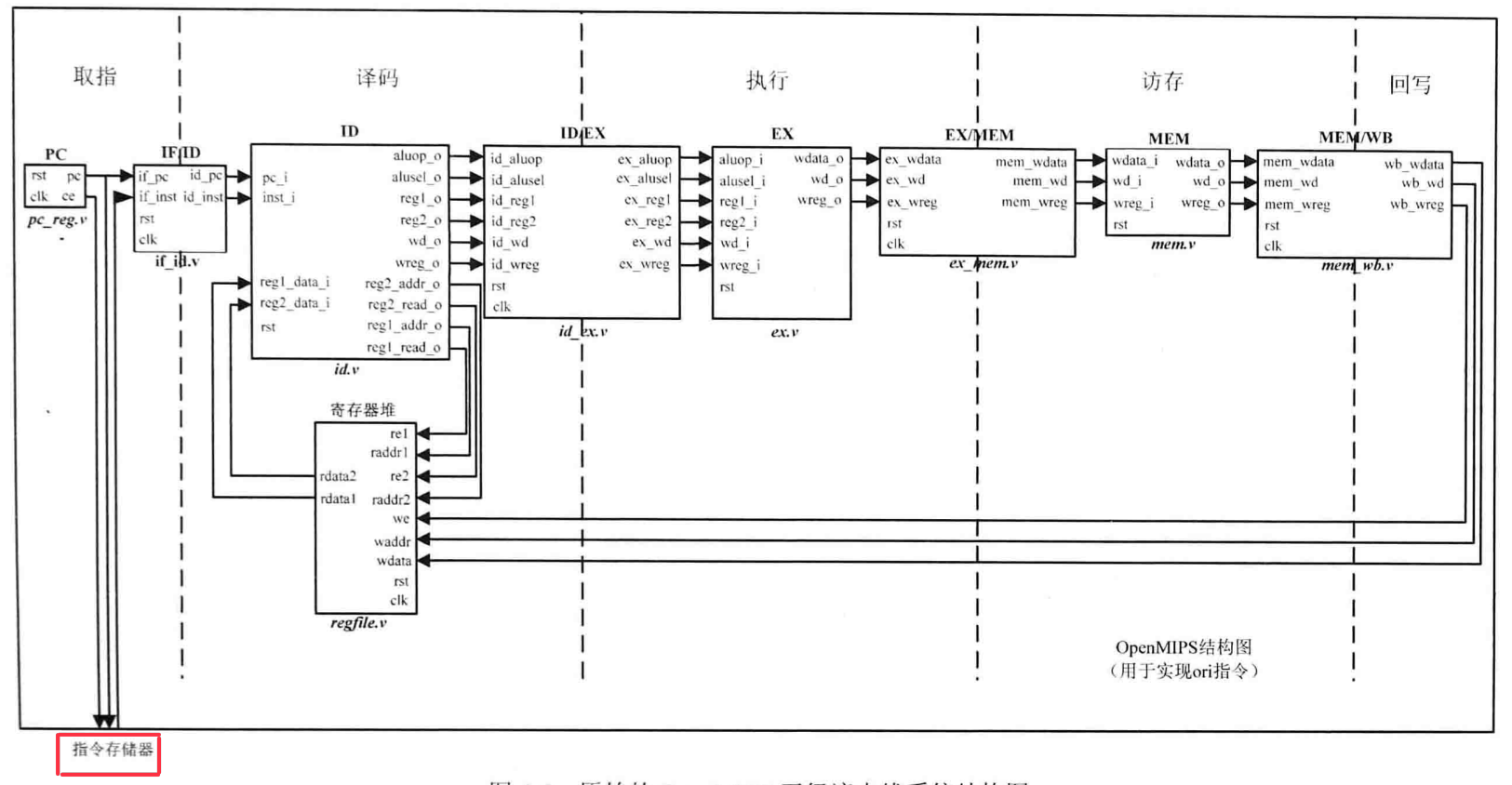


图2-1 五级流水线CPU整体设计

### 2.1.1 MIPS寄存器

实现的CPU内部有32个通用寄存器，其中0号寄存器的值，恒为零，其他31个寄存器可以改变。

另外为了支持乘法运算，加入了HILO寄存器模块。

|  |  |
| --- | --- |
| 寄存器名 | 作用 |
| 0 | 恒为零 |
| 1 | 编译暂存寄存器，为编译器保留 |
| 2-3 | 用来存放子程序的返回值（非浮点） |
| 4-7 | 用来传递子程序的前四个参数（非浮点） |
| 8-15 | 暂存寄存器，离开时子函数时不需要对其进行存储和恢复 |
| 16-23 | 暂存寄存器，离开时子函数时不需要对其进行存储和恢复 |
| 26，27 | 为中断/陷阱指令保留 |
| 28 | 全局指针;某些对时间敏感的系统可以使用该寄存器为static或extern变量提供快速的寻址方式 |
| 29 | 堆栈指针 |
| 30 | 子程序用它来作帧指针 |
| 31 | 子程序用它来作帧指针 |
| HI | 存乘法指令低位结果或除法指令的商； |
| LO | 保存乘法指令高位结果或除法指令的余数 |

表2-1 MIPS寄存器详情

### 2.1.2 MIPS指令集

MIPS指令集可以划分为R、I、J三类指令，指令各部分划分如下：

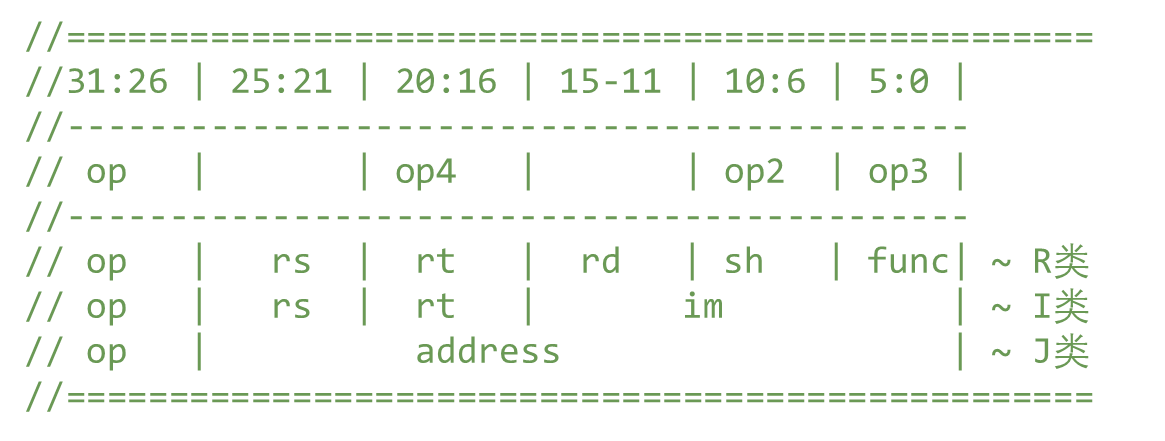


图2-2 MIPS指令集划分

本次设计预计实现五类指令，

①空指令

②逻辑运算

③移位操作

④移动操作

⑤简单算数运算

部分指令设计展示：

①指令：无符号加addu

指令格式：addu rd,rs,rt

指令功能：rd ← rs + rt

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000000 | rs | rt | rd | 00000 | 100001 |

②指令：无符号减subu

指令格式：subu rd,rs,rt

指令功能：rd ←rs - rt

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000000 | rs | rt | rd | 00000 | 100011 |

③指令：或立即数ori

指令格式：ori rt,rs,imm16

指令功能：rt ← rs | （zero-extend）imm16

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 001101 | rs | rt | immediate | | |

### 2.1.3 MIPS五级流水线

将各模块划分为IF、ID、EX、MEM、WB五个阶段：

|  |  |  |
| --- | --- | --- |
| 阶段 | 包含的模块 | 作用 |
| IF | PC | 计算下一题指令地址以及获得下一条指令 |
| ID | ID | 将指令划分，然后按照各个段的对应机器码来确定是哪条指令，根据对应的指令取得操作数以及其他控制信号 |
| Regfile | 寄存器组，提供操作数，以及数据的写入 |
| EX | EX | 根据指令进行计算或其他操作 |
| MEM | MEM | 访问数据存储器 |
| WB | Refile | 写入数据到数据存储器或通用寄存器 |

表2-2 MIPS流水线五阶段详情

## 2.2 设计环境

### 2.1.1 软件设计环境

仿真平台：vivavo2018

编程平台：Vscode

编程语言：Verlog

### 2.1.2 硬件设计环境

开发板：FPGA EG01开发板

## 2.3 设计思路

### 2.3.1 五级流水线设计

（1）取指阶段IF

功能：从指令存储器读出指令，同时确定下一条指令地址。

PC模块：给出指令地址，其中实现指令指针寄存器PC，该寄存器的值就是指令地址，对应pc.v文件。

IF/ID模块：实现取指与译码阶段之间的寄存器，将取指阶段的结果（取得的指令、指令地址等信息）在下一个时钟传递到译码阶段。对应if\_id.v文件

（2）译码阶段ID

功能：对指令进行译码，从通用寄存器中读出要使用的寄存器的值，如果指令中含有立即数，那么还要讲立即数进行符号扩展或无符号扩展。如果是转移指令并且满足转移条件，那么给出转移目标，作为新的指令地址。

ID模块：将指令划分，然后按照各个段的对应机器码来确定是哪条指令，根据对应的指令取得操作数以及其他控制信号

Rgefile模块：包含32个通用寄存器，提供操作数，以及数据的写入

ID/EX模块：实现ID模块到EX模块的数据传递

（3）执行阶段EX

功能：按照译码阶段给出的操作数、运算类型，进行运算，给出运算结果

EX模块：根据ID段对指令的翻译结果，使用源操作数进行计算或其他操作

EX/MEM模块：实现EX模块到MEM模块的数据传递

（4）访存阶段MEM

功能：访问数据存储器，由于完成的指令中没有L/S指令，因此没有数据存储器不需要访存，因此此时的MEM只需要将得到的信号继续向后传递即可。

MEM/WB模块：实现MEM模块到WB段的数据传递

（5）写回阶段WB

功能：将结果保存到目标寄存器中

HILO模块：实现乘法和除法，HI寄存器存乘法指令低位结果或除法指令的商，LO寄存器保存乘法指令高位结果或除法指令的余数

（6）延迟模块delay\_clk

功能：使用硬件将时钟信号变慢，方便烧录的开发板上流水灯效果的显示。

连接：延迟模块与其他模块的连接如下图所示，右侧4个LED灯受到delay\_clk模块的控制。

### 2.3.2 数据前推设计

数据前推在ID段和EX段均有使用

（1）ID段数据前推

由于流水线的存在可能会产生3中数据相关的问题，分别是WAR、RAW、WAW，这些问题关键在于对源操作数的读和写上。

主要思想：

如果其他段产生了我需要的结果，那么我可以直接将这个结果拿来使用，并不需要等待其写入到寄存器之后，再从寄存器中取出来，这样可以在中途得到数据，同时我们还需要保证得到的数据是最新的，我们知道MEM段的结果最先出来，因此MEM段的结果最新，其次是EX段的结果，最后不需要使用数据前推的时候再从指定寄存器中取，方便快捷。

（2）EX段数据前推

主要用于处理HI、LO寄存器数据相关问题，与ID段的思想类似，同样是在读取的时候依次考虑MEM段和WB段是否要写，以及写的地址是否与我相同。

具体代码如下：

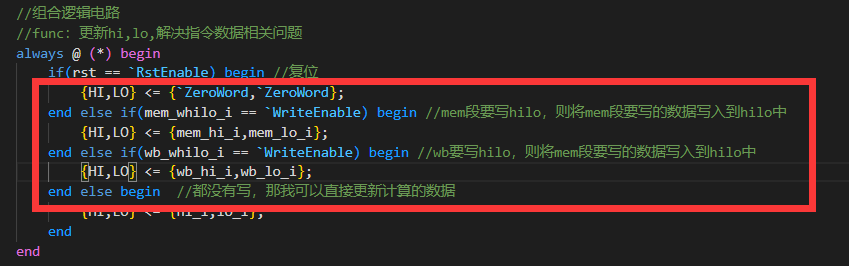


图2-3 EX段数据前推

### 2.3.3 硬件延时模块设计

该模块使用一个寄存器记录原时钟上升沿的次数，使用一个clk\_o的寄存器作为输出，clk\_o初始为0，当原来时钟上升沿到达次数达到10000000次的时候，将clk\_o反转，将clk\_o作为新的时钟信号，这样可以在硬件的层面将时钟频率降低，从而使流水灯的效果更加明显。

模块设计图如下：

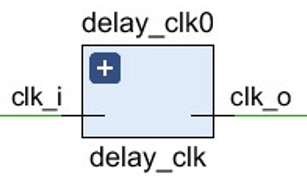


图2-4 硬件延时模块设计图

# 3 设计实现

## 3.1 五级流水线各模块设计

### 3.1.1 IF段

该段包括PC模块和IF\_ID模块，共两个模块。不包括指令寄存器instrom模块。

PC模块负责指令地址的计算，然后将pc信号传送给指令寄存器，指令寄存器instrom再将取得的指令信号传送给IF \_ID段，IF\_ID段再将指令传送给ID段。

PC模块设计图：

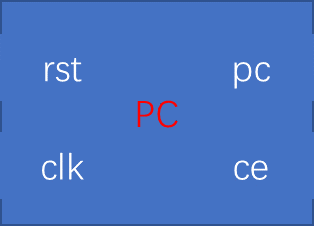


图3-1 PC模块设计图

IF/ID模块设计图：

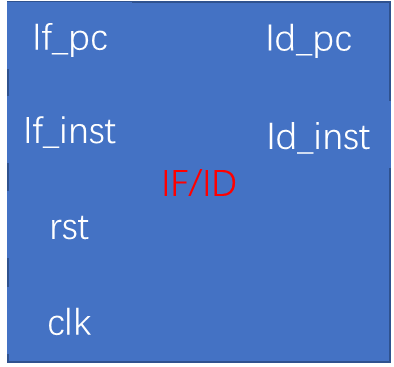


图3-2 IF/ID模块设计图

### 3.1.2 ID段

ID段主要模块是ID模块和寄存器模块regfile。

其中ID模块主要作用是分析指令，regfile模块的主要作用是写入数据和送出数据。

对于ID段中一条指令的处理的大致过程如下：

1、Id接收到指令开始译码：

①首先获取指令中4段位置的数据。

②根据op，op2，op3找到所属的类型

③根据指令的格式和用法，赋予写使能、

④运算类型（主/子）、读写使能、写的地址、

立即数的扩展

⑤根据使能获取“源操作数1、2”

2、Id段信号传到id\_ex段，ide\_ex段在时钟上升沿传到ex段

需要注意的是，ID模块在识别指令的时候，首先默认这条指令是R类指令，这样在，后续的处理过程中就可以减少许多的代码书写，在I类指令时在根据指令的格式和功能进行修改。

ID模块接口设计如下

|  |  |  |
| --- | --- | --- |
| 接口名 | 方向 | 作用 |
| rst | i | 复位信号 |
| pc\_i | i | 取得的指令 |
| inst\_i | i | 取得的指令 |
| ex\_wreg\_i | i | ex段是否要“写” |
| ex\_wdata\_i | i | ex段写“数据” |
| ex\_wd\_i, | i | ex段写“地址” |
| mem\_wreg\_i | i | mem段是否要“写” |
| mem\_wdata\_i | i | mem段写“数据” |
| mem\_wd\_i | i | mem段写“地址” |
| reg1\_data\_i | i | 端口1读取的“数据”寄存器 |
| reg2\_data\_i | i | 端口2读取的“数据”寄存器 |
| reg1\_read\_o | o | 端口1读取“使能” |
| reg2\_read\_o | o | 端口2读取“使能” |
| reg1\_addr\_o | o | 端口1读取“地址” |
| reg2\_addr\_o | o | 端口2读取“地址” |
| aluop\_o | o | 运算子类型 |
| alusel\_o | o | 运算类型 |
| reg1\_o | o | 源操作数1 |
| reg2\_o | o | 源操作数2 |
| wd\_o | o | 写地址 |
| wreg\_o | o | 是否有写的目的寄存器 |

表3-1 ID模块接口详情

### 3.1.3 EX段

EX段主要包括EX模块和EX/MEM模块，其中EX模块主要负责根据译码段指令翻译的结果来执行，包括以下几种运算：逻辑运算、移位运算、算数运算（简单算数运算、乘法运算）、移动运算。EX/MEM负责将EX段的信号传递到MEM段。

EX模块设计了多个组合逻辑电路，将不同的运算分开并行进行运算，设置不同类型运算结果的中间变量，最后再根据指令的操作大类选择一个作为最终结果，这样的好处在于方便缩短各类指令的操作时间，不足在于任意指令都需要进行多种运算，浪费资源，但是在CPU这种追求高性能的模块是值得的。

（1）逻辑运算：

根据操作的类型直接判断出来进行那种操作，然后直接进行逻辑运算即可。

可以执行的操作包括：or、and、nor、xor。

（2）移位运算

根据操作的类型直接判断进行哪种移位操作，并进行移位运算。

可以执行的操作包括：SLL（逻辑左移）、SRL（逻辑右移）、SRA（算数右移）。

注：算数左移和逻辑左移相同

（3）算数运算

准备工作：

首先计算源操作数2的补码，计算源“源操作数1”与“源操作数2的补码”的加和，并且进行语出判断。

①简单算数运算

根据操作类型返回对应计算的结果。

对于CLZ和CLO这两个计数指令，使用嵌套的三目运算符计算可以大幅减少设计出来的硬件数量。

②乘法操作

如果是乘法则获得乘法的结果，但是要注意：如果是异号相乘则需要返回结果的补码。

（4）移动运算

根据指令的类型将对应的移动结果放到移动操作的临时结果中去。

（5）汇总——挑选出真正的运算结果

这一步从上述并行计算的结果中挑选出真正的结果，将真正的结果放到wdata\_o中，传递给EX/MEM模块。

|  |  |  |
| --- | --- | --- |
| 接口名 | 方向 | 作用 |
| Rst | i | 复位信号 |
| aluop\_i | i | 运算子类 |
| alusel\_i | i | 运算主类 |
| reg1\_i | i | ex段接收到的源操作数1 |
| reg2\_i | i | ex段接收到的源操作数2 |
| wd\_i | i | ex段接收到的写地址 |
| wreg\_i | i | ex段接收到的写使能 |
| hi\_i | i | HI寄存器 |
| lo\_i | i | LO寄存器 |
| wb\_hi\_i | i | wb段写hi的值 |
| wb\_lo\_i | i | wb段写lo的值 |
| wb\_whilo\_i | i | wb段是否要写hilo |
| mem\_hi\_i | i | mem段写hi的值 |
| mem\_lo\_i | i | mem段写lo的值 |
| mem\_whilo\_i | i | mem段是否要写hilo |
| wd\_o | o | ex段“写地址” |
| wreg\_o | o | ex段“写使能” |
| wdata\_o | o | ex段“写数据” |
| hi\_o | o | ex段写hi的值 |
| lo\_o | o | ex段写lo的值 |
| whilo\_o | o | ex段是否要写hilo |

表3-2 EX模块接口详情

### 3.1.4 MEM段

该段主要的模块是MEM模块，该模块主要是用来访问数据寄存器的，但是我们设计的指令中没有需要访存的，因此此时的MEM段只起到一个传送信号的作用。

|  |  |  |
| --- | --- | --- |
| 接口名 | 方向 | 作用 |
| rst | i | 复位信号 |
| wd\_i | i | 写地址 |
| wreg\_i | i | 写使能 |
| wdata\_i | i | 写数据 |
| hi\_i | i | hi数据 |
| lo\_i | i | lo数据 |
| whilo\_i | i | hilo写使能 |
| wd\_o | o | 写地址 |
| wreg\_o | o | 写使能 |
| wdata\_o | o | 写数据 |
| hi\_o | o | hi数据 |
| lo\_o | o | lo数据 |
| whilo\_o | o | hilo写使能 |

表3-3 MEM模块接口详情

### 3.1.5 WB段

该段主要的模块是WB模块，该模块主要用来回写数据寄存器和寄存器的，由于我们没有设计数据寄存器，因此我们只需要将数据会写到寄存器中即可，这段功能使用寄存器模块regfile实现的。

|  |  |  |
| --- | --- | --- |
| 接口名 | 方向 | 作用 |
| clk | i | 时钟信号 |
| rst | i | 复位信号 |
| we | i | 写使能 |
| waddr | i | 写地址 |
| wdata | i | 写数据 |
| re1 | i | 读使能1 |
| raddr1 | i | 读地址1 |
| rdata1 | i | 读数据1 |
| re2 | i | 读使能2 |
| raddr2 | i | 读地址2 |
| rdata2 | i | 读数据2 |
| led0\_o | o | led0 |
| led1\_o | o | led1 |
| led2\_o | o | led2 |
| led3\_o | o | led3 |

表3-4 WB模块接口详情

## 3.2 数据相关解决方法

使用数据前推解决数据相关问题，数据前推在ID段和EX段均有使用。

（1）ID段数据前推实现

ID模块在取源操作数的时候首先需要考虑MEM段是否在写，写的地址是不是和我要读的地址相同，相同那刚好直接得到数据，否则就对EX段进行考察，EX段也不是，那就从寄存器中去取数据，如果不需要读取该操作数，那么这个操作数就是ID模块分析出来的立即数IMM。

具体代码如下：

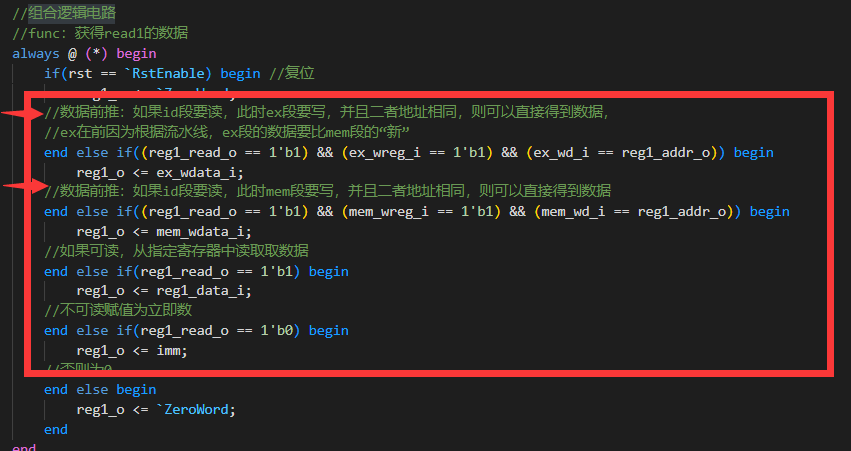


图3-1 ID段数据相关解决方案

（2）EX段数据前推实现

主要用于处理HI、LO寄存器数据相关问题，与ID段的思想类似，同样是在读取的时候依次考虑MEM段和WB段是否要写，以及写的地址是否与我相同。

具体代码如下：

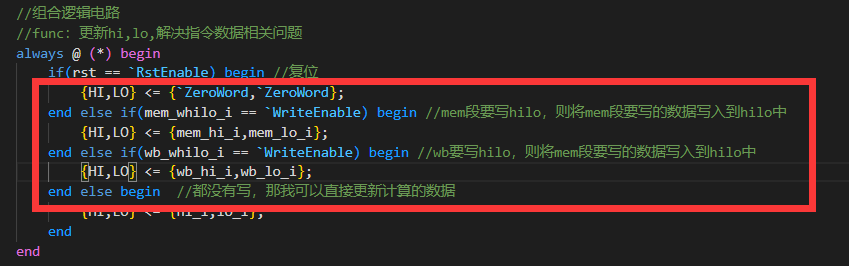


图3-2 EX段数据相关解决方案

# 4 实验设计与验证

## 4.1 LED流水灯实验的设计

这里使用硬件模块delay\_clk进行延时，右侧为4个LED灯，LED灯显示寄存器r1、r2、r3、r4的最低位，最低位是0，则led灭，反之则亮。

关键代码体现在这里：

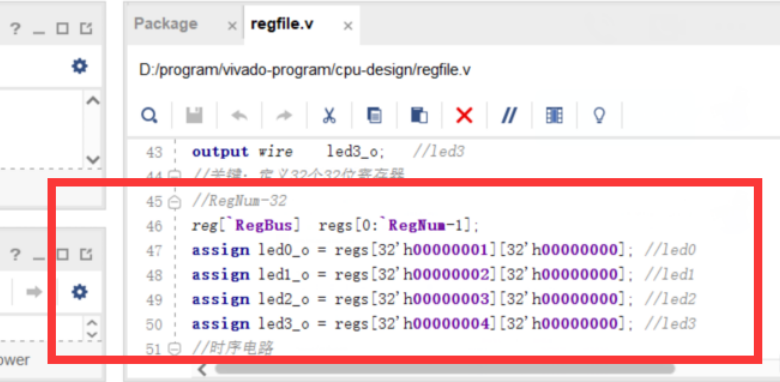


图4-1 LED与寄存器链接关系

这里再openmips模块添加了4个接口用来给LED灯进行连接，下面是综合主要模块之间的关系图：

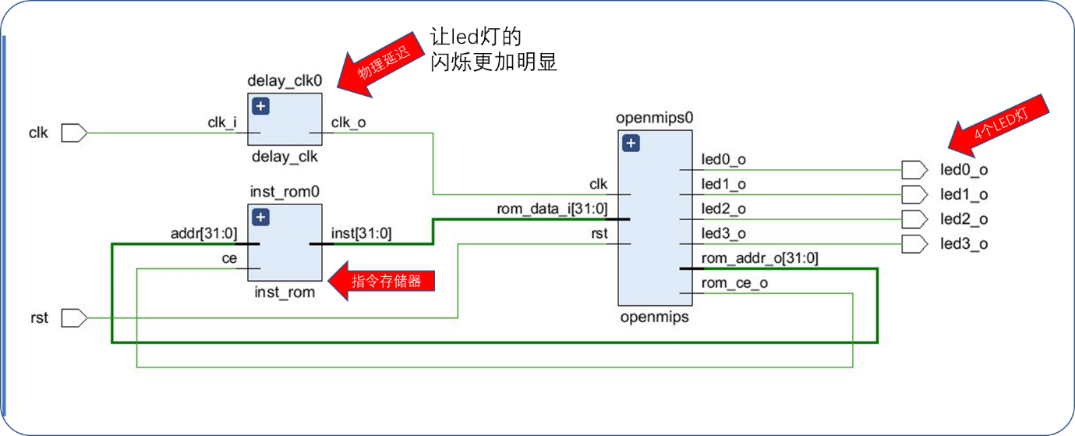


图4-2 添加了LED的顶层设计模块

设计好之后需要将LED灯的引脚与ledx\_o进行绑定，绑定关系如下：

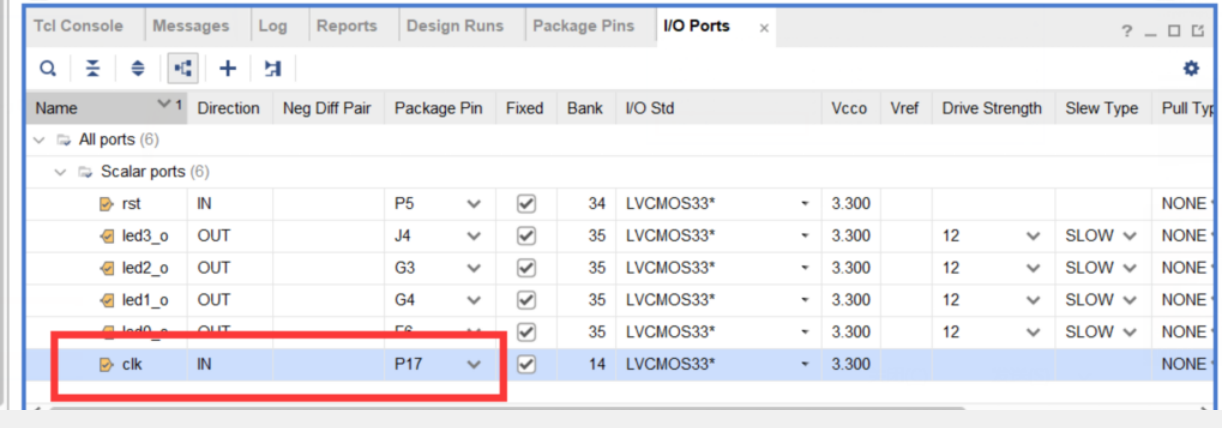


图4-3 引脚绑定关系

验证代码如下：

##一趟#################################

ori $1,$0,0x0001 # $1 = 0x0001 #led0亮

ori $1,$0,0x0000 # $1 = 0x0000 #led0灭

ori $2,$0,0x0001 # $2 = 0x0001 #led1亮

ori $2,$0,0x0000 # $2 = 0x0000 #led1灭

ori $3,$0,0x0001 # $3 = 0x0001 #led2亮

ori $3,$0,0x0000 # $3 = 0x0000 #led2灭

ori $4,$0,0x0001 # $4 = 0x0001 #led3亮

ori $4,$0,0x0000 # $4 = 0x0000 #led3灭

ori $3,$0,0x0001 # $3 = 0x0001 #led2亮

ori $3,$0,0x0000 # $3 = 0x0000 #led2灭

ori $2,$0,0x0001 # $2 = 0x0001 #led1亮

ori $2,$0,0x0000 # $2 = 0x0000 #led1灭

##一趟#################################

代码解释：给r1、r2、r3、r4依次赋值，改变寄存器的最低位，然后cpu可以通过信号线将最低位的值传递给led灯，从而引发led灯的亮灭。

使用该代码的仿真结果如下：

图中可以看出来led\_0、led\_1、led\_2、led\_3只能依次保持一个周期的高电平，与流水灯功能对应。

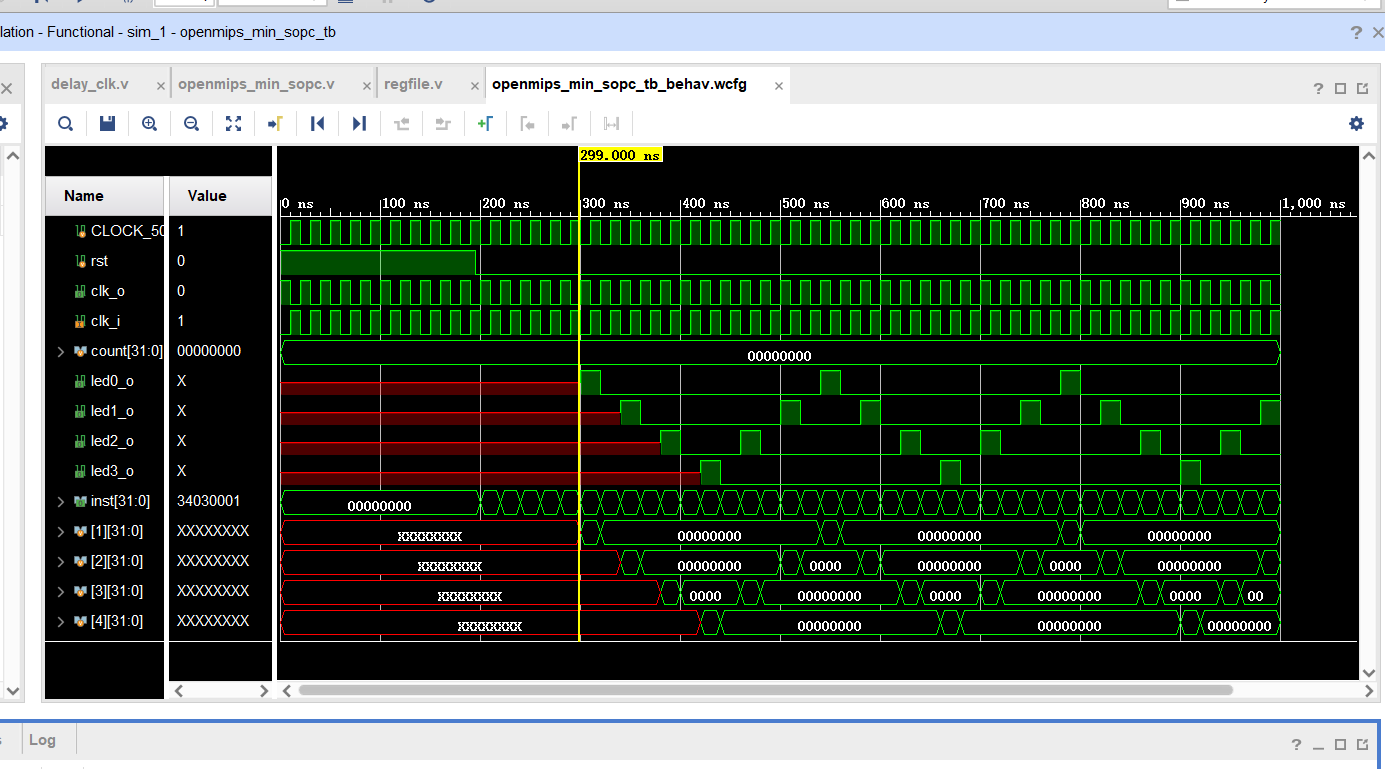


图4-4 流水灯仿真验证

## 4.2 CPU支持指令的仿真验证

### 4.2.1 “五级流水线”&“逻辑运算”仿真验证

（1）验证代码：

注释为执行后的结果

ori $7,$0,0x7 #$7=7

ori $5,$0,0x5 #$5=5

andi $1,$1,0x0000 #$1=0

ori $1,$1,0x000 #$1=1

andi $3,$1,0x0001 #$3=1

（2）仿真验证

能够实现五级流水线，并且逻辑运算结果正确。

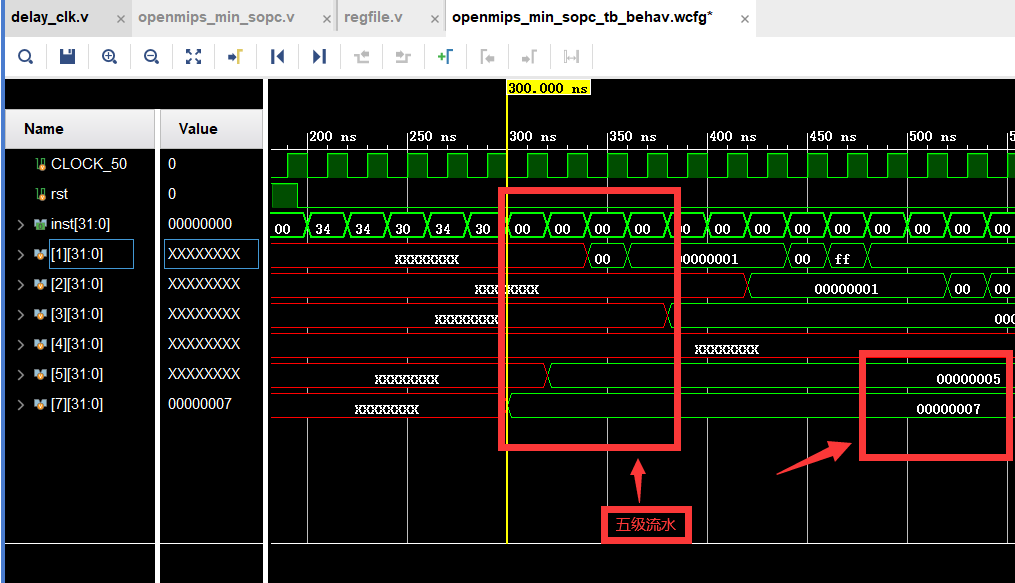


图4-5 流水线仿真验证

### 4.2.2 数据相关仿真验证

（1）验证代码

ori $1,$0,0x1100 # $1 = $0 | 0x1100 = 0x1100

ori $1,$1,0x0020 # $1 = $1 | 0x0020 = 0x1120

ori $1,$1,0x4400 # $1 = $1 | 0x4400 = 0x5520

ori $1,$1,0x0044 # $1 = $1 | 0x0044 = 0x5564

（2）仿真验证

观察仿真波形，可以发现代码执行的结果与正确结果相同，能够解决数据相关。

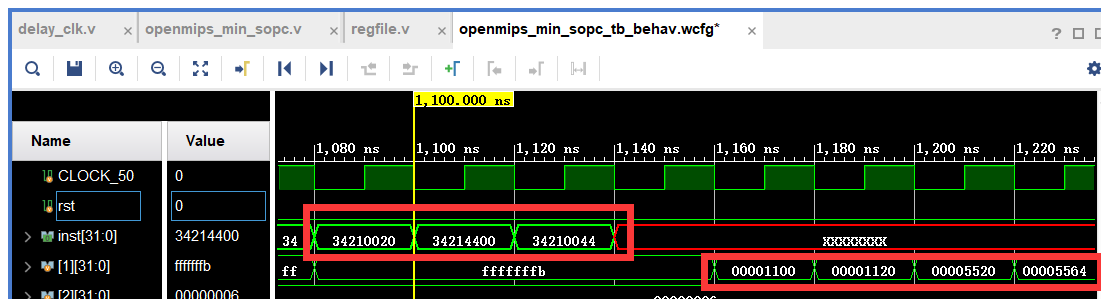


图4-6 数据相关解决仿真验证

## 4.3 遇到的问题与解决方案

（1）设计时端口连接错误

解决方案：

在后续方案的设计时，为模块的端口加上in和out的后缀，这样大幅减少了后续编写时出现的错误。

（2）模块编写名称错误

在模块编写的时候，端口和模块定义好了之后，在后面编写的时候名称错误，有时编译器认为这是个新的模块，不会报错，但是在仿真的时候发生错误，这种错误比较难找，需要一个一个的排查。

（3）LED流水灯绑定时出错

在绑定的时候将开关的引脚误认为时LED的引脚绑定到了ledx\_x引脚上了，排查了很长时间才找到灯的引脚标号在灯的上方。



图4-7 引脚对应图

（4）led灯实验的时候灯常亮

这是由于时钟频率过快导致的，灯闪烁的过快，人眼看起来是常量的。

解决方法：使用时钟延迟，这里使用硬件模块delay\_clk进行延迟，将时钟频率放慢了10000000倍，闪烁清晰可见。

（5）led灯实验的时候表现为“一个灭其余三个亮”

原因：测试代码中高低电平写错了

解决方案：将测试代码的最低为改正即可。

# 5 总结

本次课程设计让我首次将模拟电路、计算机组成原理、计算机体系结构这三门课程综合的应用起来。开始本次课程设计之前我以为设计一个CPU是难如登天的事情，曾一度认为这个是无法完成的任务，但是当我真正的去做的时候，去分析每一个模块的时候，我惊奇的发现“我也可以”，就这样一步一步的完成每一个模块的设计，再将其拼装起来，组成一个整体，我也可以完成一个CPU了，并且这个cpu也是可以运行指令的，虽然这些指令并不复杂，但是这种“创造生命”的成果让我非常的骄傲和自豪。

通过本次课程设计，也加深了我对CPU硬件结构、工作方式的理解，更加清楚CPU处理一条指令需要经过哪些过程，曾经只是知道这个模块是做什么的，但是并不清楚是怎么做的，本次CPU设计之后我可以在脑海中构建出来一条指令执行时各部件的工作情况，这也相当于了解了计算机的“心”，对于我们后期的学习和发展有着很重要的作用。

最后我和队友合作完成LED流水灯的硬件设计，这部分我们在CPU的基础上添加了引脚，并为其更好的显示配置了延迟的时钟周期，并且学习了针脚的绑定，以及程序烧录到开发板中，这个过程中我们学到了很多书面上学不到的知识，例如怎么排错，怎么找引脚的标号，开发板的芯片上有哪些信息等等，这对我们后期开发硬件非常有帮助。

另外个人认为可以使用《自己动手写CPU》这本书作为指导手册，然后让学生在此基础上设计其他指令集的CPU，引领同学着手设计，不然在开始的时候真的是手足无措，但是这也同样锻炼了我们的动手能力，总之通过本次课程设计，我收获了很多，成长了很多。

# 6 参考文献

1. 唐朔飞. 计算机组成原理（第2版）[M]. 北京：高等教育出版社，2008.作者. 书名. 出版地: 出版社，出版年.
2. 雷思磊. 自己动手写CPU[M]. 北京：电子工业出版社，2014.
3. 计算机组成与设计：硬件、软件接口（原书第4版）(jb51.net) [M].北京：机械工业出版社，2011.

|  |
| --- |
| 一、原创性声明 |
| 本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。  特此声明！  **作者签字:** |
| |  |  |  |  | | --- | --- | --- | --- | |  | 评价内容 | 权重 | 得分 | | **验收** | RISC-V指令集，完成了23条指令，6类，自己设计的顶层结构图，多周期5级流水线，可以冲突检测，前推，Cache不停顿MEM停顿，实现了斐波拉契数列，测试数据自己写的，可下载，数码管和LED都可显示。 | 0.6 | 优 | | **设计报告** | 报告格式比较规范，语言流畅，条理清楚，实验原理和实验步骤描述正确、详实，实验结果正确，有实验结果的仿真图和下载后运行的照片，实验结果的分析、对比比较充分，有实验体会。 | 0.4 | 优 | | 合计 | 优 | | | | 指导教师（签章）： 2022 年 11 月 10 日 | | | | |