

**信息科学与技术学院**

计算机组成原理实验报告

学 院： 信息科学与技术学院

班 级： 计科1808

组 员： 尚文达 郑博文

**2020年12月24日**

# 目录

[目录 2](#_Toc59713433)

[实验1 二选一开关设计实验 4](#_Toc59713434)

[一、实验目的 4](#_Toc59713435)

[二、实验原理 4](#_Toc59713436)

[三、实验原理图 4](#_Toc59713437)

[实验2 计数器+ALU 4](#_Toc59713438)

[一、实验目的 4](#_Toc59713439)

[二、实验原理 4](#_Toc59713440)

[三、实验步骤 5](#_Toc59713441)

[四、实验原理图 5](#_Toc59713442)

[实验3 ROM+ALU实验 6](#_Toc59713443)

[一、实验目的 6](#_Toc59713444)

[二、实验内容 6](#_Toc59713445)

[三、实验步骤 7](#_Toc59713446)

[(一)LPM\_ROM 7](#_Toc59713447)

[(二)ROM+ALU 7](#_Toc59713448)

[四、实验原理图 7](#_Toc59713449)

[实验4 RAM实验 9](#_Toc59713450)

[一、实验目的 9](#_Toc59713451)

[二、实验原理 9](#_Toc59713452)

[三、实验步骤 10](#_Toc59713453)

[四、实验原理图 10](#_Toc59713454)

[实验5 程序计数器PC与地址寄存器AR实验 11](#_Toc59713455)

[一、实验目的 11](#_Toc59713456)

[二、实验原理 11](#_Toc59713457)

[三、实验原理图 12](#_Toc59713458)

[实验6 微控制器组成实验 12](#_Toc59713459)

[一、实验目的 12](#_Toc59713460)

[二、实验原理 13](#_Toc59713461)

[三、实验原理图 14](#_Toc59713462)

[实验7 课设 8位基本结构模型计算机设计 15](#_Toc59713463)

[一、实验目的 15](#_Toc59713464)

[二、实验内容 15](#_Toc59713465)

[三、实验原理 15](#_Toc59713466)

[1.运算部件 16](#_Toc59713467)

[2.寄存器组 16](#_Toc59713468)

[3.指令寄存器 16](#_Toc59713469)

[4 .程序计数器 16](#_Toc59713470)

[5 .地址寄存器 17](#_Toc59713471)

[6.指令系统设计 17](#_Toc59713472)

[7.建立数据通路 18](#_Toc59713473)

[8 .控制执行单元 19](#_Toc59713474)

[四、CPU子部件 20](#_Toc59713475)

[运算器模块ALU\_MD 20](#_Toc59713476)

[数据传输模块REGS\_MD 21](#_Toc59713477)

[微指令控制模块uPC 23](#_Toc59713478)

[CPU顶层设计 24](#_Toc59713479)

[本学期课程实验总结： 30](#_Toc59713480)

[组内人员工作量划分： 30](#_Toc59713481)

# 实验1 二选一开关设计实验

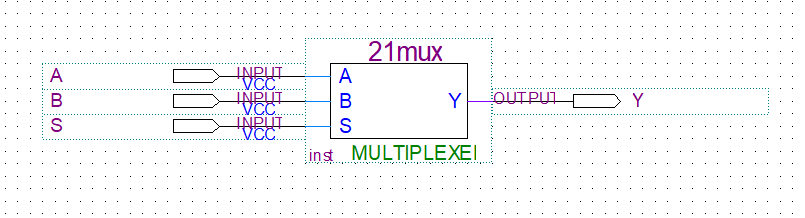
## 一、实验目的

* 学习了解verilog语言
* 学习了解设备操作
* 学习了解QuartusⅡ的使用
* 学习了解二选一开关设计的原理和实现
* 熟悉硬件连接操作

## 二、实验原理

用Quartus2设计二选一开关，并导入到实验箱中，通过开关产生两种不同蜂鸣声。

## 三、实验原理图



# 实验2 计数器+ALU

## 一、实验目的

* 了解运算器的数据传输通路；
* 验证运算器的组合功能；
* 掌握算术逻辑运算加、减工作原理；
* 验证实验台运算的 8 位加、减、与、直通功能；
* 按给定数据，完成几种指定的算术和逻辑运算
* 熟悉掌握代码及用代码实现元件操作
* 了解计数器原理

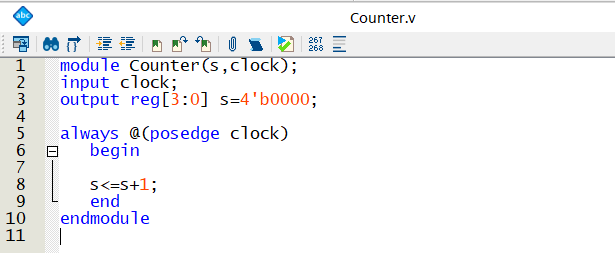
## 二、实验原理

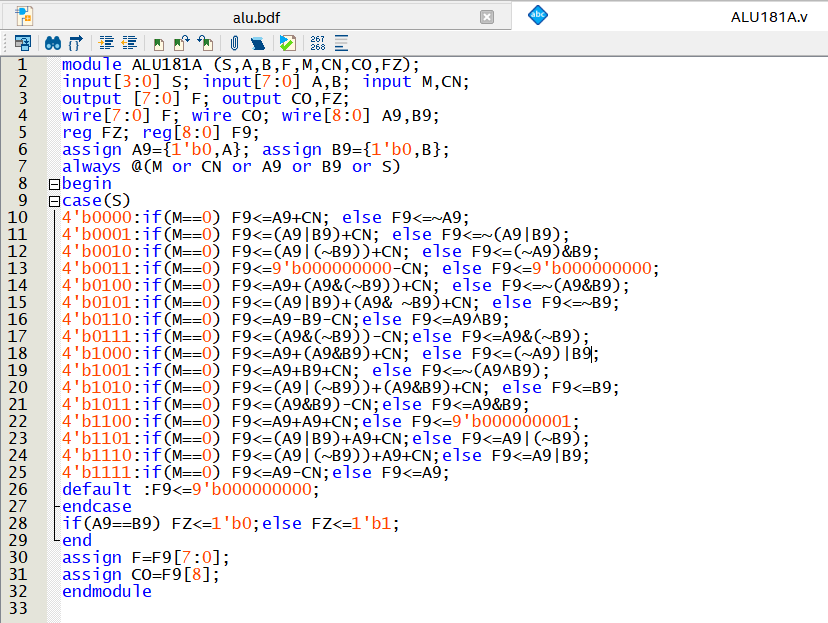
用 Quartus2 编写代码设计计数器的Verilog HDL ，进行波 形仿真，并导入到实验箱中， 通过按钮实现+1 操作 。其中运算器ALU181根74LS181的功能用 Verilog HDL 硬件描述语言编辑而成，构成 8 位字长的 ALU。参加运算的两个 8 位数据分别为 A[7..0]和 B[7..0]。 运算模式由 S[3..0]的16 种组合决定，而 S[3..0]的值由 4 位 2 进制计数器 LPM\_COUNTER 产生，计数时钟是 CLK；此外，设 M=0，选择算术运算，M=1 为逻辑运算，CN 为低位的进位；F[7..0]为输出结果；CO 为运算后的输出进位。两个 8 位数据 A 和 B 由总线 IN[7..0]分别通过两个电平锁存器 74373 锁入。

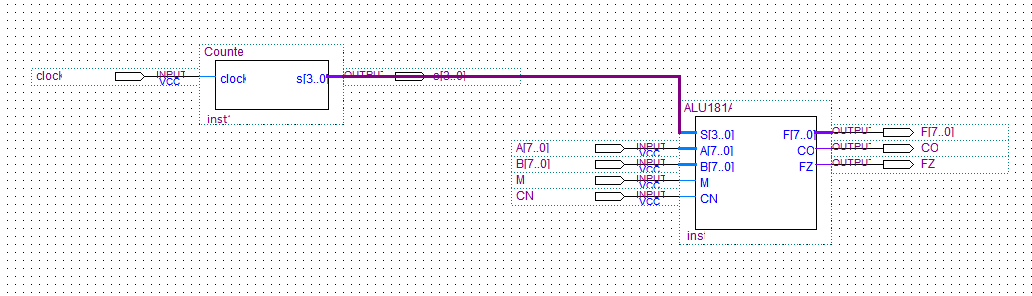
## 三、实验步骤

1. 设计ALU 元件。在 Quartus II 环境下，用文本输入编辑器 Text Editor 输入 ALU181.VHD 算术逻辑.单元文件，编译 Verilog HDL 文件，并将 ALU181.VHD 文件制作成一个可调用的原理图元件。
2. 以原理图方式建立顶层文件工程。选择图形方式。输入实验电路图，从 Quartus II 的基 本元件库中将各元件调入图形编辑窗口、连线，添加输入输出引脚。将所设计的原理图图形文件 ALU.bdf 保存到原先建立的文件夹中，将当前文件设置成工程文件，以后 的操作就都是对当前工程文件进行的。
3. 器件选择。选择 Cyclone 系列，在 Devices 中选择器件 EP3C/4C/5C/10CL 系列，根据芯片具体型号，对照康芯所提供的表格，查找引脚并锁定。编译，引脚锁定，再编译。引脚锁定后需要再次进行编译，才能将锁定信息确定下来，同时生成芯片编程/配置所需要的各种文件。
4. 引脚锁定：选择电路模式 0，IN 输入数据端选择键 1、2 分别锁PIO8~PIO18，A0\_B1 选通输入 A 或 B 选择键 3 锁定 PIO3，CN 进位位选择键 7 锁定 PIO6，M 选择键 8 锁定 PIO7，CLK 选择键 6 锁定 PIO5,F 计算结果锁定数码 5、6（PIO32~PIO39)；A[7..0]锁定数码 1、2（PIO16~PIO23)；B[7..0]锁定数码 3、4 （PIO24~PIO31)；CN4 计算后输出进位位锁定数码管 7 最低位 PIO40，数码管共接 4 位，空脚时默认高电平，为了观察直观，建议其它 3 个脚（PIO42~PIO44)置地；S[3..0]锁定数码管 8（PIO44~PIO47)。
5. 芯片编程 Programming（也可以直接选择光盘中的示例）。打开编程窗口。将配置文件 ALU.sof下载进 KX-CDS 系列现代计算机组成原理系统中的 FPGA 中。
6. 选择实验系统的电路模式是 NO.0，验证 ALU 的运算器的算术运算和逻辑运算功能。根据表 4-1，用按键 1、2 输入数据 A[7..0]和 B[7..0]，并设置 S[3..0]、M、CN，验证 ALU 运算器的算术运算和逻辑运算功能，记录实验数据。

## 四、实验原理图







# 实验3 ROM+ALU实验

## 一、实验目的

* 掌握 FPGA 中 lpm\_ROM 的设置，作为只读存储器 ROM 的工作特性和配置方法；
* 用文本编辑器编辑 mif 文件配置 ROM，学习将程序代码以 mif 格式文件加载于 lpm\_ROM 中；
* 在初始化存储器编辑窗口编辑 mif 文件配置 ROM；
* 验证 FPGA 中 LPM\_ROM 的功能。

## 二、实验内容

通过将ROM中读取出的16位数据q[15..0]，分别给ALU的A（q[15..8]）和B（q[7..0]），进行相加运算，输出结果在实验箱上显示。

## 三、实验步骤

### (一)LPM\_ROM

1. 创建工程ROMMD
2. 创建mif文件，并编辑相应内容。

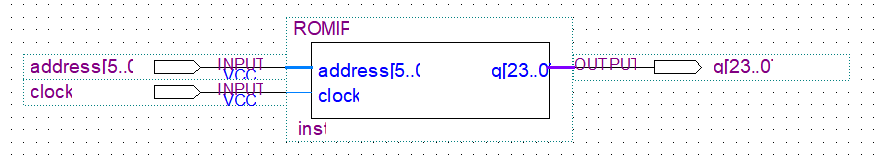
File->New->Memory Files->Memory Initialization File,设置参数：Number of words:64；Word size:24。编辑mif文件，设置相应内容。

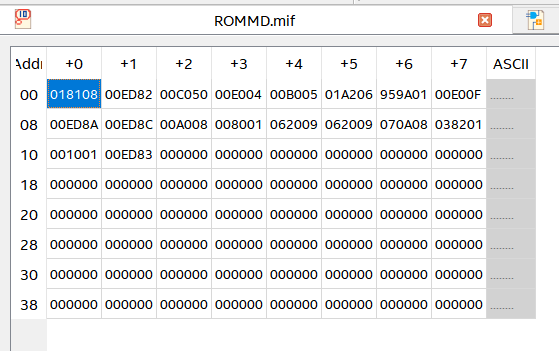
1. 创建原理图文件，另存为ROMMD.bdf
2. 引用ROM元件
3. 自动生成输入、输出端口。右键->Generate pins for symbol ports。
4. 编译。成功的话进行仿真，引脚绑定。
5. 仿真
6. 引脚绑定（模式0）

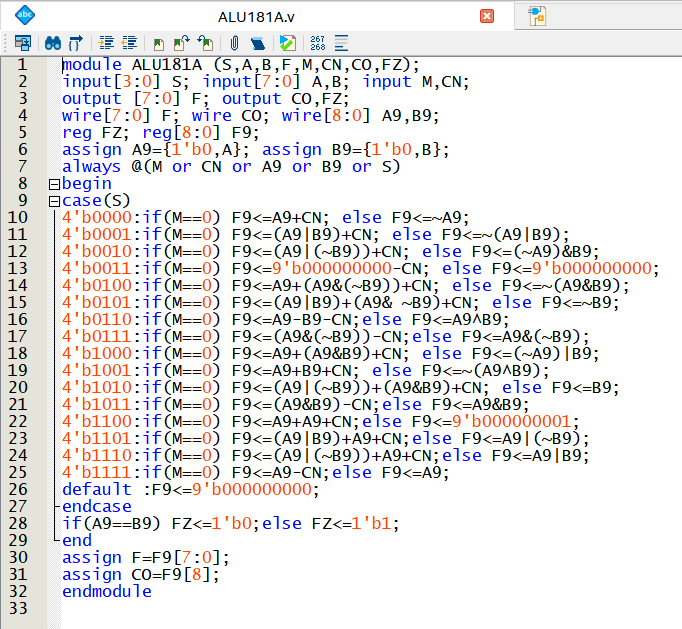
### (二)ROM+ALU

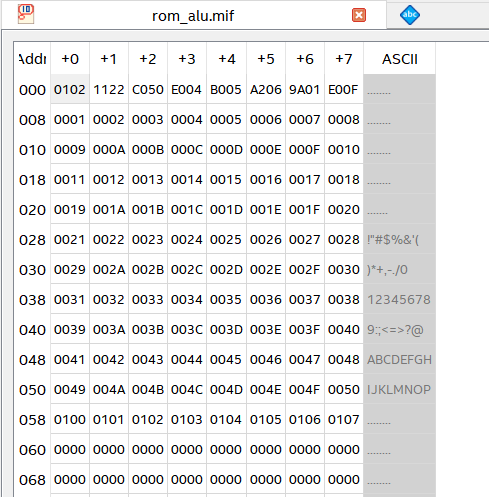
1. 创建工程rom\_alu
2. 创建mif文件，并编辑。
3. 创建原理图文件，另存为rom\_alu.bdf
4. 引用ROM元件
5. 参照第3周实验，使用.v文件生成ALU元件并调用。
6. 自动生成输入、输出端口。右键->Generate pins for symbol ports。并将ALU的A、B的Input端口删掉。
7. 进行相应连线。将ROM输出q[15..0]与ALU的输入A（q[15..8]）、B（q[7..0]）分别相连。
8. 编译。成功的话进行仿真，引脚绑定。
9. 引脚绑定提示：模式1

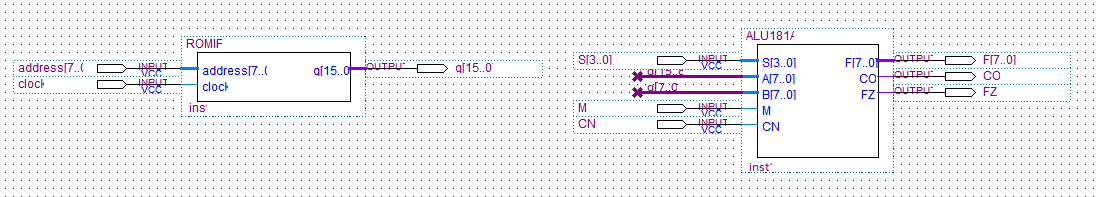
## 四、实验原理图











# 实验4 RAM实验

## 一、实验目的

* 了解 FPGA 中 RAMlpm\_ram\_dq 的功能；
* 掌握 lpm\_ram\_dq 的参数设置和使用方法；
* 掌握 lpm\_ram\_dq 作随机存储器RAM仿真测试方法，工作特性和读写方法。

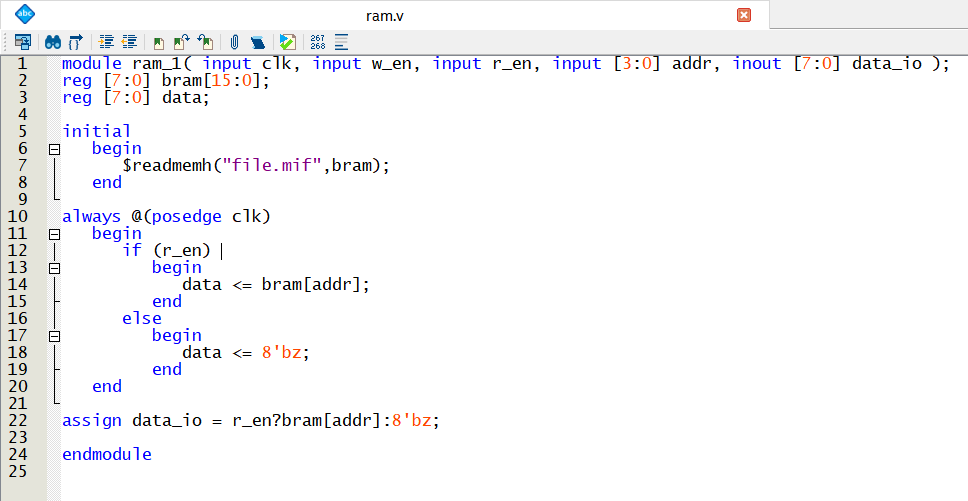
## 二、实验原理

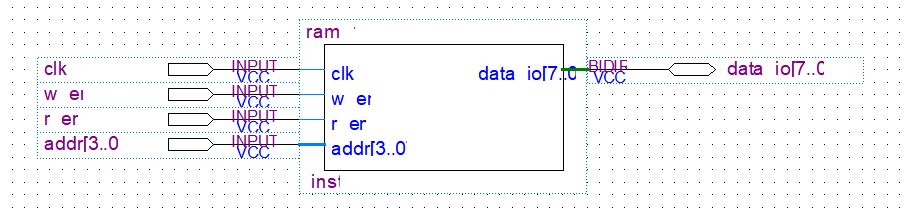
在 FPGA 中利用嵌入式阵列块 EAB 可以构成存储器，数据从 ram\_dp0 的左边 D[7..0]输入，从右边 Q[7..0]输出，R/W 为读/写控制信号端。当输入数据和地址准备好以后，在 inclock 是地址锁存时钟，当信号上升沿到来时，地址被锁存，数据写入存储单元。数据的读出控制是从 A[7..0]输 入存储单元地址，在 CLK 信号上升沿到来时，该单元数据从 Q[7..0]输出。R/W 是读/写控制端，低电平时 进行读操作，高电平时进行写操作；CLK 是读/写时钟脉冲信号；DATA[7..0]是 RAM\_dq0 的 8 位数据输入 端；A[7..0]是 RAM 的读出和写入地址；Q[7..0]是 RAM\_dq0 的 8 位数据输出端。

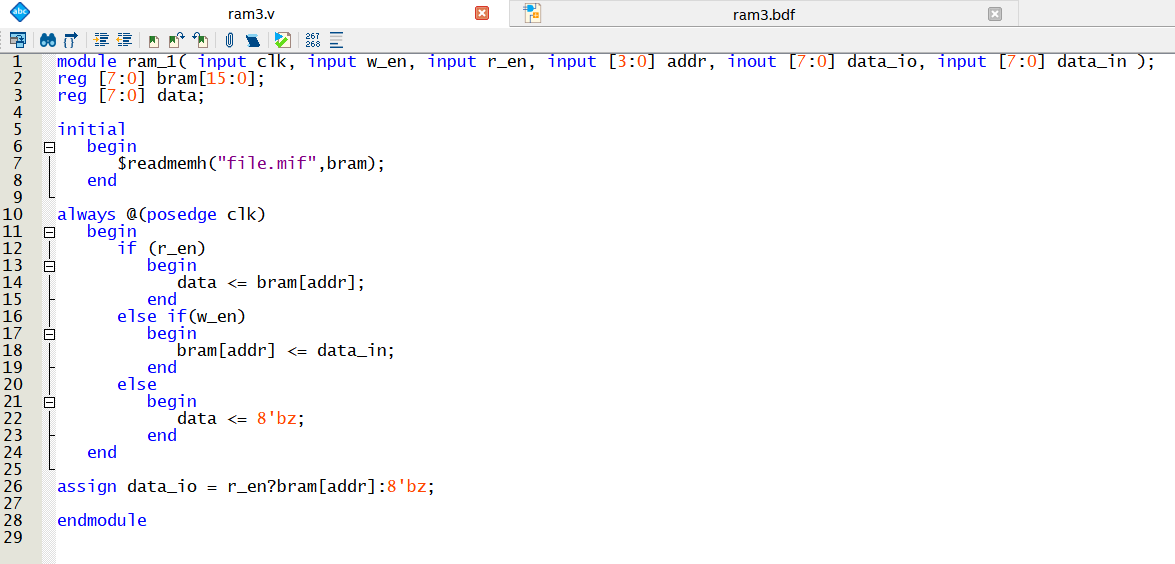
## 三、实验步骤

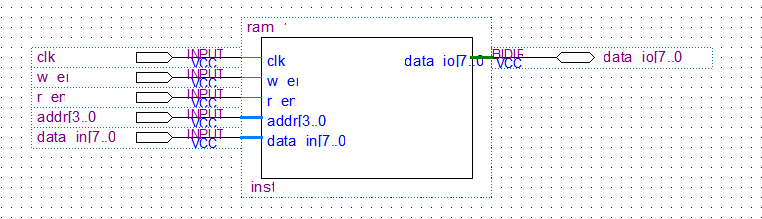
* 1. 编程实现ram，不能调用自带的IP核，也即编写Verilog HDL语言实现功能2和3(详见ram1代码)。
  2. 通过mif文件初始化ram数值
  3. 实现ram的读取数据的功能。
  4. 进行波形仿真。注意时钟周期的设置。
  5. 自行选择一个合适的模式，进行引脚绑定，下载到实验箱进行验证，设置一个地址，读取该地址的数据。
  6. 在上述基础上，增加写功能。

## 四、实验原理图









# 实验5 程序计数器PC与地址寄存器AR实验

## 一、实验目的

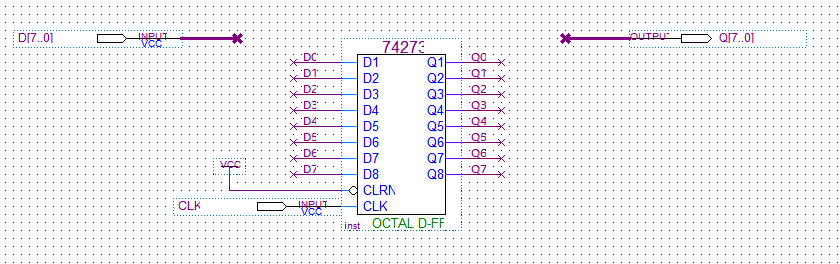
* 掌握地址单元的工作原理；
* 掌握 PC 的两种工作方式，加 1 计数和重装计数器初值的实现方法；
* 掌握地址寄存器 AR 从程序计数器 PC 获得数据和从内部总线 BUS 获得数据的实现方法。

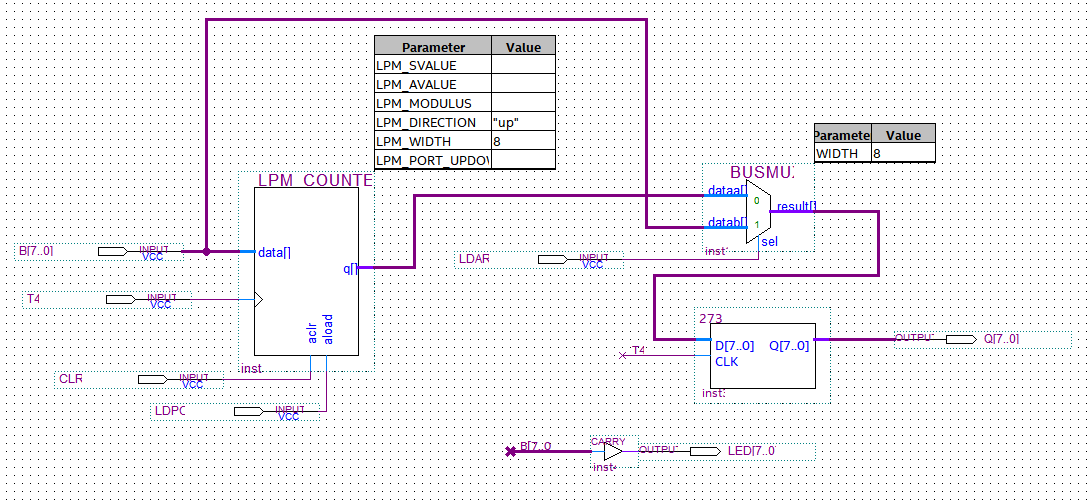
## 二、实验原理

采用总线多路开关联接方式 地址单元主要由三部分组成：程序计数器 PC、地址寄存器 AR 和多路开关 BUSMUX。程序计数器 PC用以指出下一条指令在主存中的存放地址，CPU 正是根据 PC 的内容去存取指令的。因程序中指令是顺序执行的，所以 PC 有自增功能。程序计数器提供下一条程序指令的地址，在 T4 时钟脉冲的作用下具有自动加 1 的功能；在 LDPC 信号的作用下可以预置计数器的初值（如子程序调用或中断相应等）。当 LDPC 为高电平时，计数器装入 data[ ]端输入的数据。aclr 是计数器的清 0 端，高电平有效（高电平清零）；aclr 为低电平时，允许计数器正常计数。地址寄存器 AR（DFF\_8 或 273）锁存访问内存 SRAM 的地址。地址寄存器 AR 的地址来自两个渠道，一是程序计数器 PC 的输出，通常是下一条指令的地址；二是来自于内部数据总线的数据，通常是被访问操作数的地址。为了实现对两路输入数据的切换，在 FPGA 的内部通过总线多路开关 BUSMUX 进行选择。LDAR与多路选择器的 sel 相连，当 LDAR 为低电平，选择程序计数器的输出；当 LDAR 为高电平时，选择内部数据总线的数据。

采用 PC、AR 通过三态门 lpm\_bustri 与 BUS 联接程序计数器 PC 与地址寄存器 AR 结合，产生对存储器 RAM 进行读写的地址。地址单元主要由三部分组成：程序计数器 PC、地址寄存器 AR 和三态门 lpm\_bustri。程序计数器 PC 用以指出下一条指令在主存中的存放地址，CPU 正是根据 PC 的内容去存取指令的。程序计数器提供下一条程序指令的地址，在时钟脉冲 PC\_CLK 的作用下具有自动加 1 的功能；在 LOAD\_PC 信号的作用下可以预置计数器的初值（如子程序调用或中断相应等）。当 LOAD\_PC 为高电平时，计数器装入 data[7..0]端输入的数据。RST是计数器的清 0 端，高电平有效（高电平清零）；RST 为低电平时，允许计数器正常计数。地址寄存器 AR采用锁存器 lpm\_latch 结构,锁存访问内存 SRAM 的地址。

## 三、实验原理图





# 实验6 微控制器组成实验

## 一、实验目的

1．掌握微程序控制器的工作原理和构成原理；

2．掌握微程序的编写、输入，观察微程序的运行；

3.学习绿皮手册 P73-77 微程序控制器部分实验内容；

完成 P77-80 实验 4-8 微控制器实验 1：微控制器组成实验（实验任务1和任务2），并验证下列：

当无有效外部输入时，即 P1-P4 全部高电平，FC/FZ ，SWA 、 SWB 均为低电平，I 与 SE 信号的关系；

P1 低电平有效时，I 与 SE 的关系；

P2 低电平有效时，I 与 SE 的关系；

P3 低电平有效时，FC 、FZ 、I 与 SE 的关系；

P4 低电平有效时，SWA 有效，对 SE 的影响；

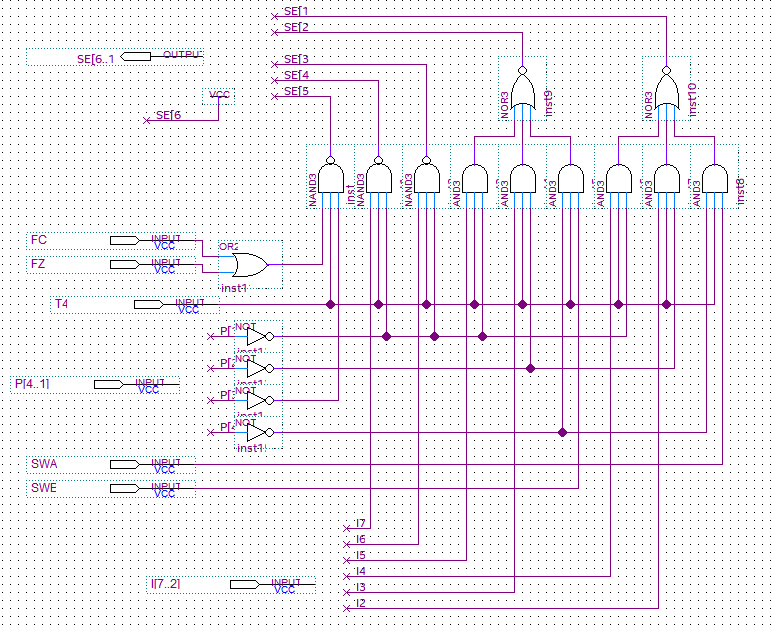
P4 低电平有效时，SWB 有效，对 SE 的影响；

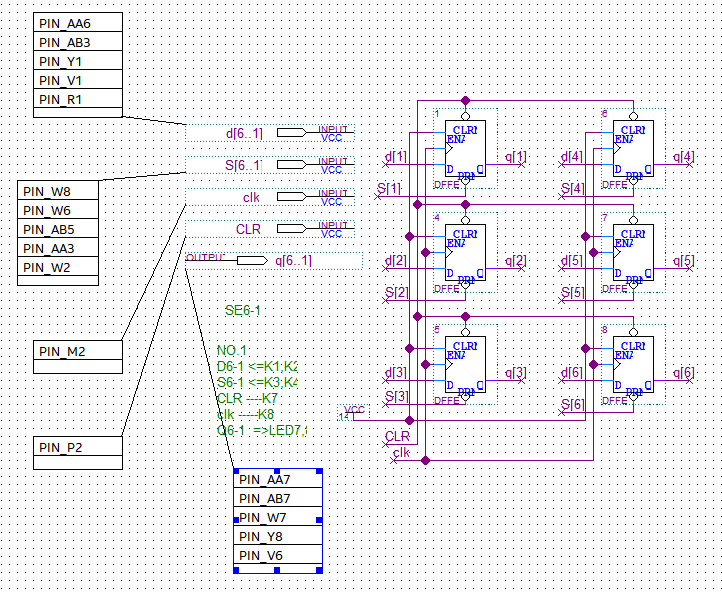
完成后将各种情况下的输入、输出信号做好记录，寻找规律。

## 二、实验原理

微程序控制电路是 CPU 控制器的核心电路，控制产生指令执行时各部件协调工作所需的所有控制信 号，以及下一条指令的地址。微程序控制器的组成，主要由三个部分组成，分别是微指令控-77-制电路、微地址寄存器和微指令存储器 lpm\_rom。其中微指令控制电路用组合电路对指令中的 I[7..2]、操 作台控制信号 SWA 和 SWB 的状态、状态寄存器的输出状态 FC、FZ，产生微地址变化的控制信号，实现对微地址控制；微地址寄存器控制电路的基本输入信号是微指令存储器的下地址字段 M[6..1]，同时还受微指令控制电路的输出信号 SE[6..1]和复位信号 RST 的控制，输出下一个微指令的地址；控制存储器由 FPGA中的 LPM\_ROM 构成，输出 24 位控制信号。在 24 位控制信号中，微命令信号为 18 位，微地址信号 6 位。在 T3 时刻将打入微地址寄存器 uA 的内容，即为下一条微指令地址。当 T4 时刻进行测试判别时，转移逻辑满足条件后输出的负脉冲，通过强制端将某一触发器置为“1”状态，完成地址修改。微程序控制器中的微控制代码可以通过对 FPGA 中 LPM\_ROM 的配置进行输入，通过编辑 LPM\_ROM.mif 文件来修改微控制代码。

## 三、实验原理图





# 实验7 课设 8位基本结构模型计算机设计

## 一、实验目的

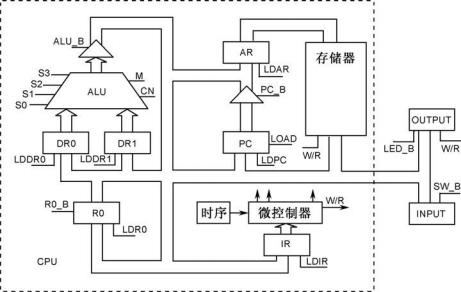
* 深入理解基本模型计算机的功能和组成知识；
* 深入了解计算机各类典型指令的执行流程；
* 学习微程序控制器的设计和相关技术，掌握LPM\_ROM的配置方法。
* 在掌握部件单元电路实验的基础上，进一步将单元电路组成系统，构造一台基本模型计算机；
* 定义五条机器指令，并编写相应的微程序，上机调试，掌握计算机整机概念。掌握微程序的设计方法，学会编写二进制微指令代码表；
* 通过完整的计算机的设计，全面了解并掌握微程序控制方式计算机的设计方法。

## 二、实验内容

1. 对各功能模块进行仿真，给出仿 真波形图，并通过详细分析这些波形，说明各模块的功能和特点，及在整机系统 中的作用。
2. 设计微程序。设计对应 5 条指令的微程序流程图，及对应的微 代码表。
3. 参考仿真波形图，对模型机整机在执行程序的完整过程进行 仿真。并将仿真波形图的所有细节，给出详细的报告。
4. 利用其中介绍的所有硬件测试工具完成对模型机执行应用程序 的测试，并将测试结果与仿真波形图仔细对照，给出报告。各测试工具的设置情 况及加入主系统的电路模块也完全按照此节的介绍。
5. 编辑通信模块，将模型机运行时的信息传输至外部液晶显示器显示出来， 并 能通过 STEP 键，逐段了解 CPU 的运行情况，包括微指令的每一步运行情况。

## 三、实验原理

在计算机中运算器和控制器是计算机系统的核心组成部件，它们组成了中央 处理器 CPU ，将 CPU 集成在一块芯片上，称为微处理器。CPU 通过内部总线， 建立芯片内各部件之间的信息传送通路。下图是一个简易的 CPU 结构图，图中虚线框内部分包括运算器、控制器、 程序存储器、数据存储器和微程序存储器等，实测时，它们都可以在单片 FPGA 中实现。虚线框外部分主要是输入/输出装置，包括键盘、数码管、LCD 显示器 等，用于向 CPU 输入数据，或 CPU 向外输出数据，以及观察 CPU 内部工作情 况及运算结果。该 CPU 主要由算术逻辑单元 ALU，数据暂存寄存器 DR1 、DR2 ， 数据寄存器 R0～R2， 程序计数器 PC， 地址寄存器 AR， 程序/数据存储器MEMORAY，指令寄存器 IR，微控制器 uC，输入单元 INPUT 和输出单元 OUTPUT所组成。



### 1.运算部件

运算部件的任务是对操作数进行加工处理。运算部件主要由三部分组成 ：

(1)输入逻辑。操作数可以来自各种寄存器，也可以来自 CPU 内部的数据线。 每次运算 多只能对两个数据进行操作， 所以运算部件设置了两个输入缓冲寄存 器（DR0 和 DR1），分别选择两个操作数参加运算。

(2)算术/逻辑运算部件 ALU 。ALU 是运算部件的核心， 完成具体的运算操 作。它的主要部件就是一个加法器，负责对两个操作数进行求和运算。两个数进 行算术加时有时能产生进位，所以加法器除了具有求和逻辑以外，还提供进位信 号传递的逻辑，称为进位链。

(3)输出逻辑。运算结果可以直接送往接收部件，也可以经左移或右移后再送 往接收部件，所以输出逻辑往往具有移位功能。常用移位寄存器，通过移位传送 实现左移、右移，并通过三态门，由控制信号 ALU\_B 控制送往内部数据总线。

### 2.寄存器组

计算机工作时，CPU 需要处理大量的控制信息和数据信息。例如对指令信 息进行译码，以便产生相应控制命令对操作数进行算术或逻辑运算加工，并且根 据运算结果决定后续操作等。因此，在 CPU 中需要设置若干寄存器，暂时存放 这些信息。在模型 CPU 中，寄存器组由 R0 、R1 、R2 所组成。

### 3.指令寄存器

指令寄存器（IR）用来存放当前正在执行的指令，它的输出包括操作码信息、 地址信息等，是产生微命令的主要逻辑依据。

### 4 .程序计数器

程序计数器（PC）也称指令指针，用来指示指令在存储器中的存放位置。当 程序顺序执行时， 每次从主存取出一条指令，PC 内容就增量计数， 指向下一条 指令的地址。增量值取决于现行指令所占的存储单元数。如果现行指令只占一个 存储单元，则 PC 内容加 1；若现行指令占了两个存储单元，那么 PC 内容就要 加 2。当程序需要转移时，将转移地址送入 PC，使 PC 指向新的指令地址。因 此，当现行指令执行完时，PC 中存放的总是后续指令的地址；将该地址送往主 存的地址寄存器 AR， 便可从存储器读取下一条指令。

### 5 .地址寄存器

CPU 访问存储器， 首先要找到需要访问的存储单元， 因此设置地址寄存器 (AR)来存放被访单元的地址。当需要读取指令时，CPU 先将 PC 的内容送入 AR ， 再由 AR 将指令地址送往存储器。当需要读取或存放数据时，也要先将该数据 的有效地址送入 AR， 再对存储器进行读写操作。

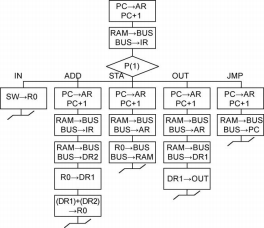
### 6.指令系统设计

计算机的性能与它所设置的指令系统有很大的关系。指令系统反映了计算机 的主要属性，而指令系统的设置又与机器的硬件结构密切相关。指令是计算机执 行某种操作的命令，而指令系统是一台计算机中所有机器指令的集合。通常性能 较好的计算机都设置有功能齐全、通用性强、指令丰富的指令系统，而指令功能 的实现需要复杂的硬件结构来支持。因此在设计 CPU 时，首先要明确机器硬件 应具有哪些功能，然后根据这些功能来设置相应指令，包括确定所采用的指令格 式、所选择的寻址方式和所需要的指令类型。计算机是通过执行指令来处理各种 数据的。为了指出数据的来源，操作结果的去向及所执行的操作，一条指令必须 包含下列信息：

* 操作码。它具体说明了操作的性质及功能。一台计算机可能有几十条至几百条 指令，每一条指令都有一个相应的操作码，计算机通过识别该操作码来完成不同 的操作。
* 操作数的地址。CPU 通过该地址就可以取得所需的操作数。
* 操作结果的存储地址。把对操作数的处理所产生的结果保存在该地址中，以便 再次使用。
* 下一条指令的地址。执行程序时，大多数指令按顺序依次从主存中取出执行， 只有在遇到转移指令时，程序的执行顺序才会改变。为了压缩指令的长度，可以 用一个程序计数器（PC）存放指令地址。每执行一条指令，PC 的指令地址就自 动加 1（设该指令只有一个主存单元），指出将要执行的下一条指令的地址。当 遇到执行转移指令时，则用转移地址修改 PC 的内容。由于使用了 PC，指令中 就不必明显地给出下一条将要执行指令的地址。

从上述分析可知，一条指令实际上包括两种信息，即操作码和地址码。操作 码(Operation Code ，OP)用来表示该指令所要完成的操作（如加、减、乘、除、数 据传送等），其长度取决于指令系统中的指令条数。地址码用来描述该指令的操 作对象，或者直接给出操作数，或者指出操作数的存储器地址或寄存器地址（即 寄存器名）。

一条指令往往是由若干条微指令组成的序列来实现的，而一条微指令往往包 含若干条微命令，每条微命令都对应着一个微操作。下图是一个具有五条指令 IN 、 ADD 、STA 、OUT 和 JMP 的微程序流程图。其中方框代表基本的微操作，菱形 框为分支判断框。每条指令都是由不同微操作来完成的，微操作的数量各不相同， 因此每条指令所需的执行时间也是不同的 。



微程序流程图

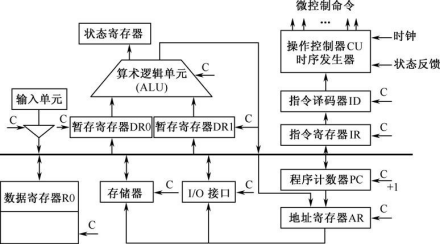
设模型机指令系统中包含有五条基本指令， 分为算术运算指令、存/取指令和控制转移指令等三种类型。其功能如表下所示。五条机器指令分别是：IN （输入）、 ADD （二进制加法）、STA （存数）、OUT （输出）、JMP （无条件转移），指令格 式如上表所示（高 4 位二进制数为操作码）。

|  |  |  |  |
| --- | --- | --- | --- |
| 助记符 | 机器指令码 | addr 地址码 | 功能说明 |
| IN | 00H | IM 36 | “INPUT”中的数据→R0 |
| ADD addr | 10H | IM 37  XX H | R0+[addr] →R0 |
| STA addr | 20H | IM 38  XX H | R0 → [addr] |
| OUT addr | 30H | IM 39  XX H | [addr] → BUS |
| JMP addr | 40H | IM 40  XX H | addr →PC |

其中，opcode 为操作码，rs 为源寄存器，rd 为目的寄存器，并规定了寄存器操数的格式。其中 IN 为单字长（8 位二进制），其余为双字长指令，XXH 为 addr对应的十六进制地址码。

### 7.建立数据通路

下面是 CPU 的数据通路框图，主要由运算器运算器、控制器、存储器和输入/输出装置组成，它们通过内部的数据总线相互连接起来。图中，ALU 为运算 器；DR0 和 DR1 为其输入端的两个暂存寄存器；R0 是数据寄存器，用来保存数 据和运算结果；PC 为程序计数器；IR 和 ID 分别为指令寄存器和指令译码器； AR 和 MOMERY 分别为存储地址寄存器和存储数据寄存器；INPUT 为输入装 置；OUTPUT 为输出装置。各基本单元模块的输出端通过三态门控制与内部数 据总线相连接；微控制器按照时序发生器的节拍，对指令进行译码后产生同步的 控制信号。各部件旁边的 C 是由微指令译码器输出的控制信号，用于控制数据 的输入和输出。

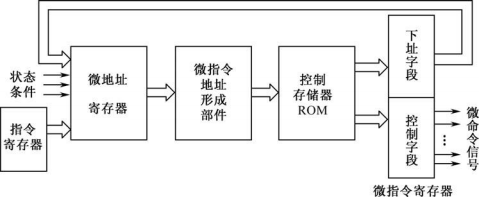


在图中与数据总线相连的信号有些是单向的，有些则是双向的。如数据寄存 器、存储器、I/O 接口的数据信号是双向的，而指令信号、地址信号、控制信号 是单向的。

### 8 .控制执行单元

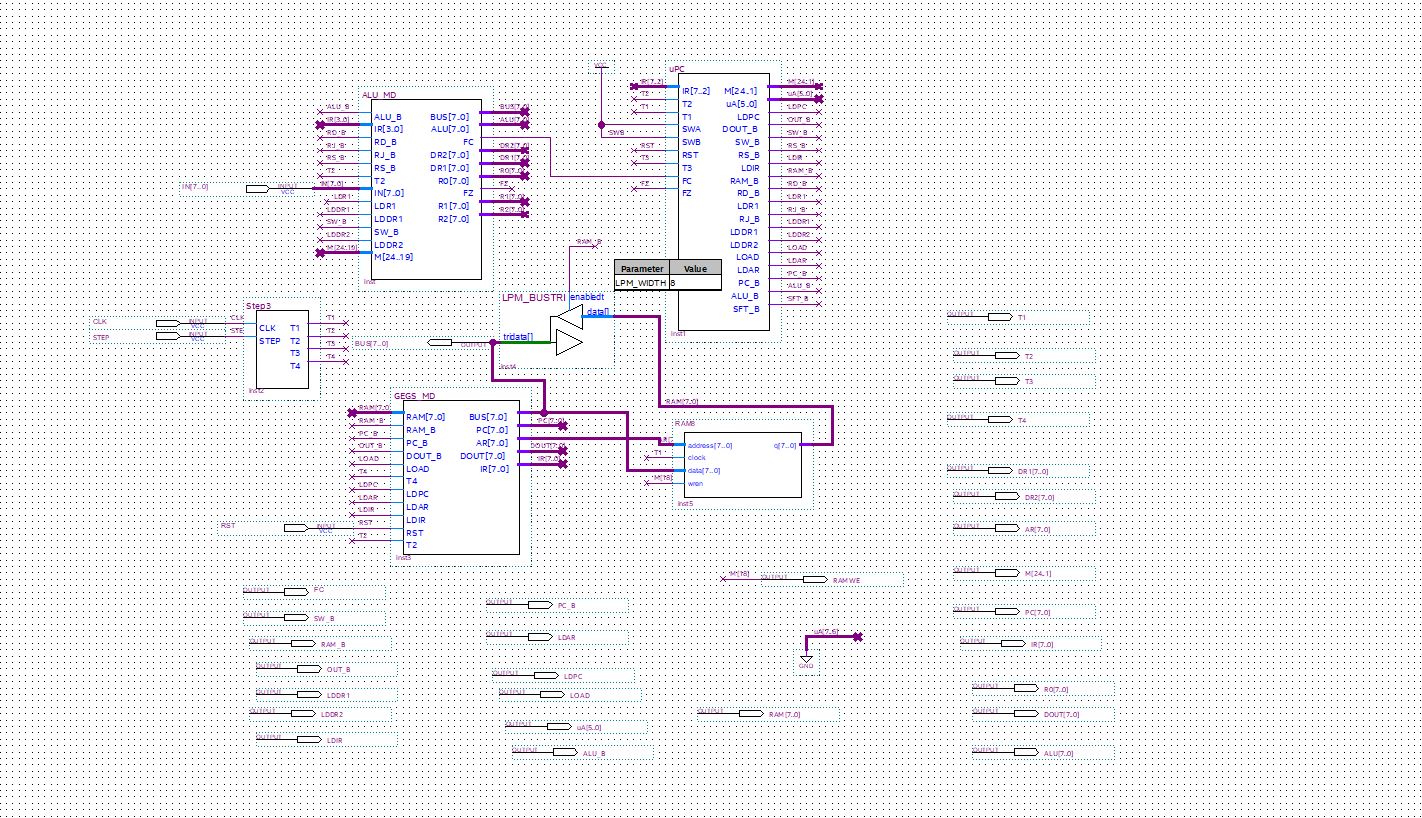
如下图所示，微程序控制器主要由控制存储器 ROM、微指令寄存器 uIR 和 微指令地址形成部件 uAR 三部分组成。

控制存储器用于存放指令系统所对应的全部微程序，可以用FPGA 中的嵌入 式 RAM 块构成的只读存储器 LPM\_ROM 构成，其字长由控制命令的多少、微 指令的编码格式及下地址字段的宽度而定。微指令寄存器 uIR 用来存放从控制 存储器读出的一条微指令的信息，由下地址字段和控制字段构成。下地址字段指 出将要执行的下一条微指令的地址，控制字段则保存一条微指令中的操作控制命 令。微指令地址形成部件又称微指令地址发生器，用来形成将要执行的下一条微 指令的地址，简称微地址。一般情况下，下一条微指令的地址由上一条微指令的 下地址字段直接决定。但当微程序出现分支时，将由状态条件的反馈信息去形成 转移地址。当取指令公共操作完成后，可以用操作码去产生执行阶段的微指令入 口地址。



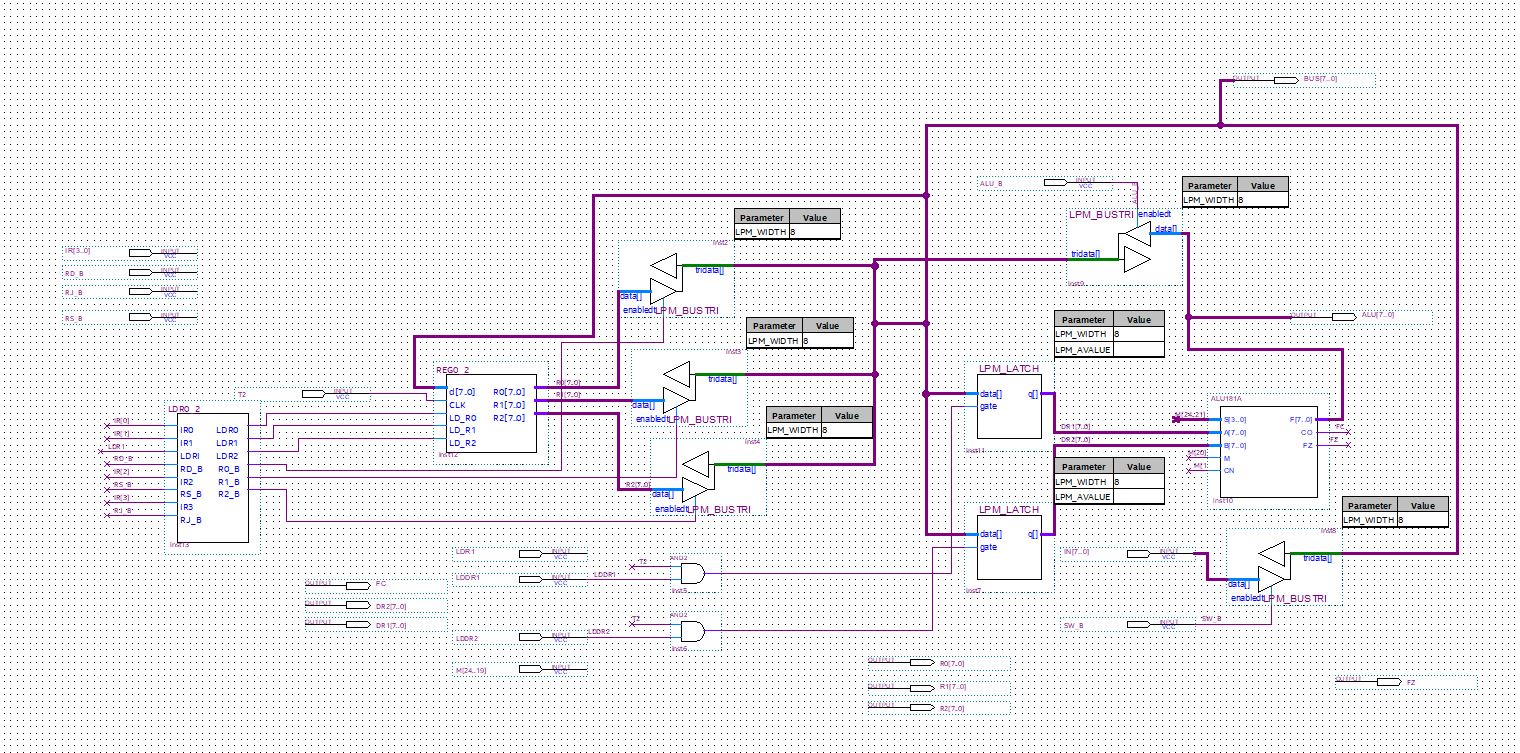
微程序控制的基本原理图

## 四、CPU子部件

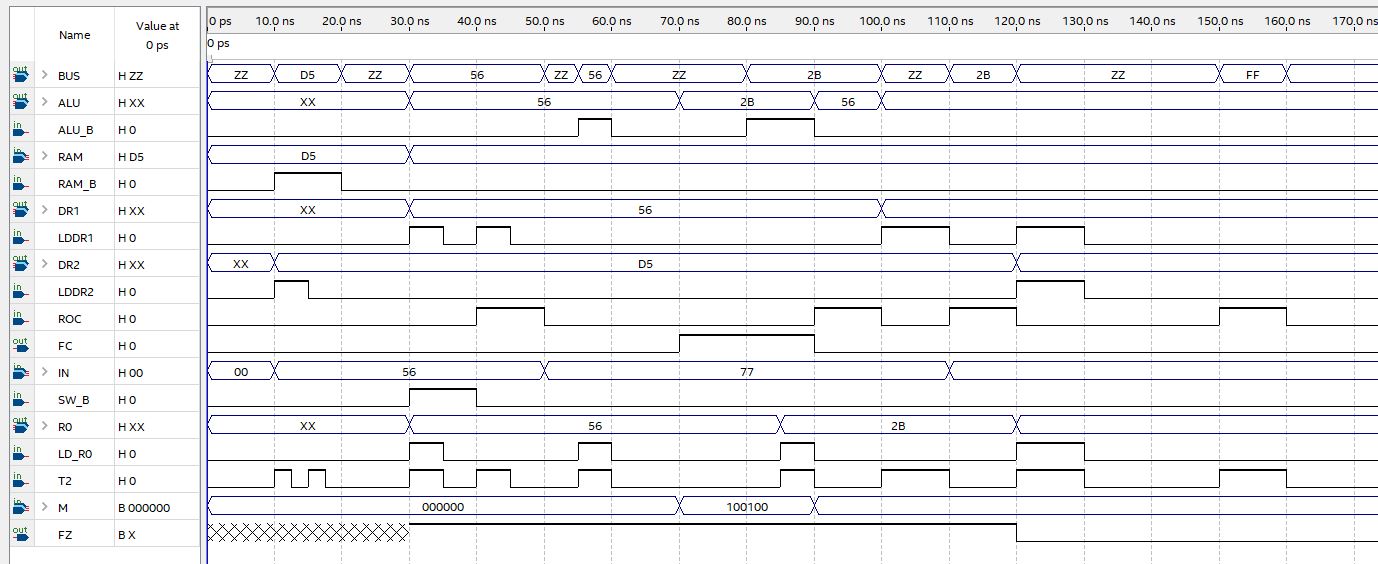


### 运算器模块ALU\_MD

此ALU的两个8位数据输入端的数据暂存器DR1和DR2，是用LPM模块数据锁存器LPM\_LATCH构成，分别为ALU提供两个操作数。数据寄存器由REG0\_2模块内R0、R1、R2组成，它们主要用于存放源操作数Rs、目的操作数Rd、运算结果和输入/输出数据。数据寄存器的输入端与内部总线BUS[7..0]相连，数据输出端通过三态门与数据总线相连，寄存器的读/写控制信号来自于微程序控制模块uPC。



#### 仿真分析：



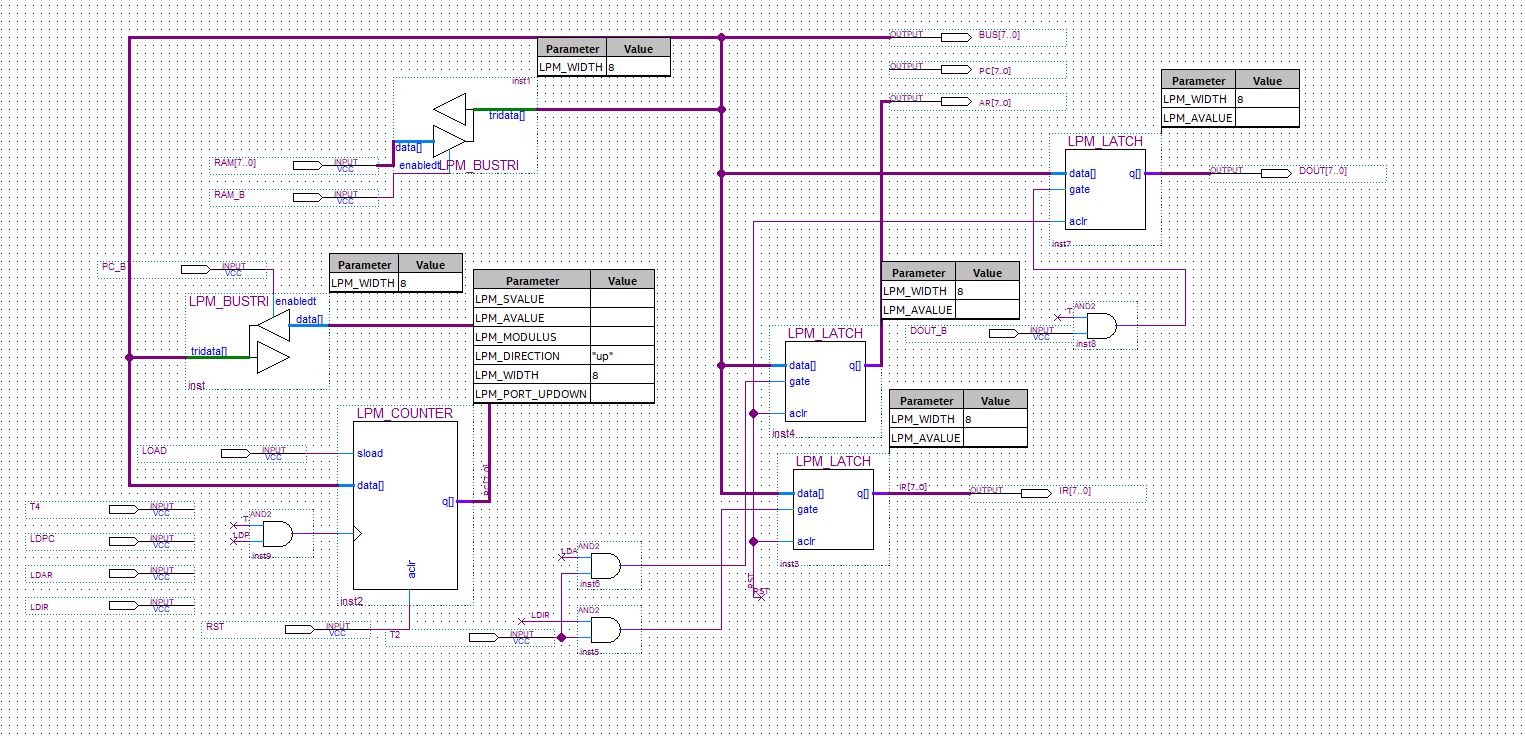
首先控制信号RAM\_B高电平，BUS读入RAM中的值D5H，随后控制信号LDDR1与脉冲T2高电平，DR2读入BUS值（即刚刚读入的RAM中的值）D5H。控制信号SW\_B高电平，BUS读入IN中的值56H。控制信号LDDR1与T2脉冲高电平，DR1读入BUS中的值56H。控制信号LD\_R0与T2高电平，控制信号ALU\_B高电平，R0、ALU读入BUS值56H。T2、R0\_C、LDDR1高电平，BUS读入R0的值56H，DR1读入BUS的值56H。ALU\_B、LD\_R0、T2高电平R0读入ALU的值56H。当M值读入100100时ALU181A实行加法，ALU值改变为2BH，进位信号FC高电平，值零信号FZ高电平（负逻辑）。ALU\_B高电平BUS读入ALU的值2BH，LD\_R0、T2高电平R0读入BUS值2BH。

ALU\_MD规律如下表格。

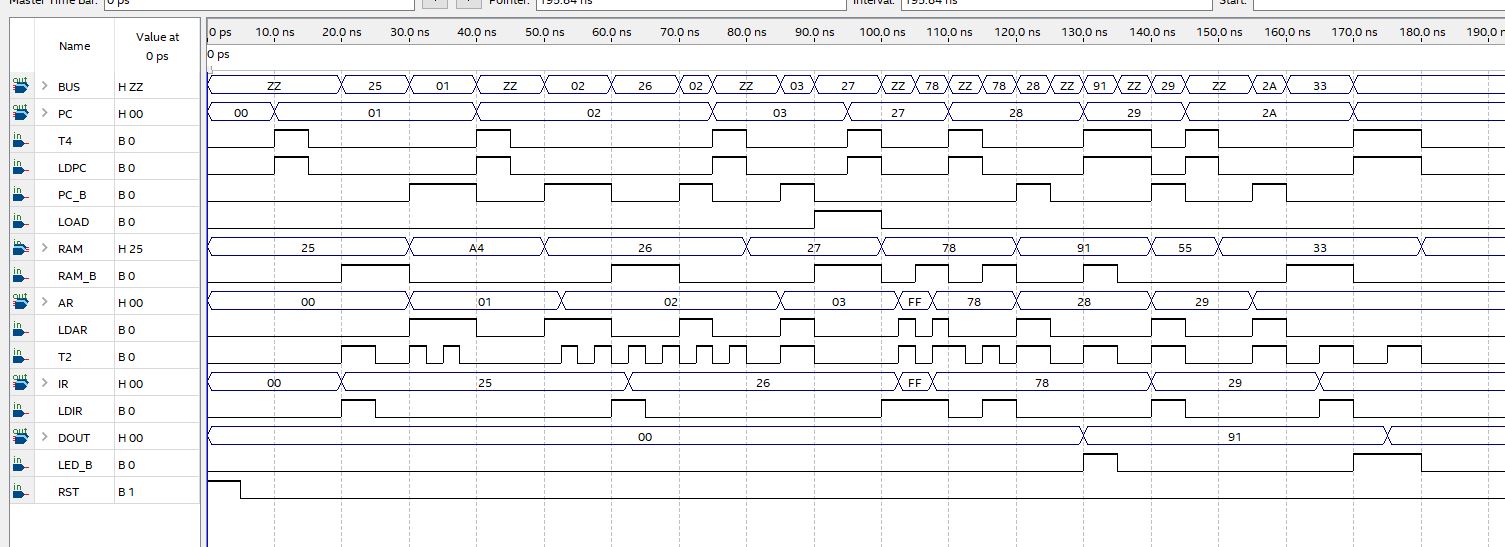


### 数据传输模块REGS\_MD

REG\_MD模块电路中的地址寄存器AR向储存器提供地址信号，AR调用LPM\_LATCH实现，模块中的其他寄存器，即指令寄存器IR和输出锁存器也都采用相同形式的锁存器担任；程序计数器PC是用LPM的可预置计数器担任的。其中异步时序复位端RST，高电平时将PC清零，低电平时允许计数器工作；CLK为计数时钟，由来自uPC模块的LDPC和T4通过一个与门共同控制；数据预置同步加载控制端为LOAD，当LOAD端为低电平时，计数器正常计数；当LOAD端为高电平时，且时钟有效时向计数器加载数据。数据输入端d[7..0]直接与内部总线BUS[7..0]连接，数据输出端q[7..0]通过三态门与内部总线相连。



#### 仿真分析：



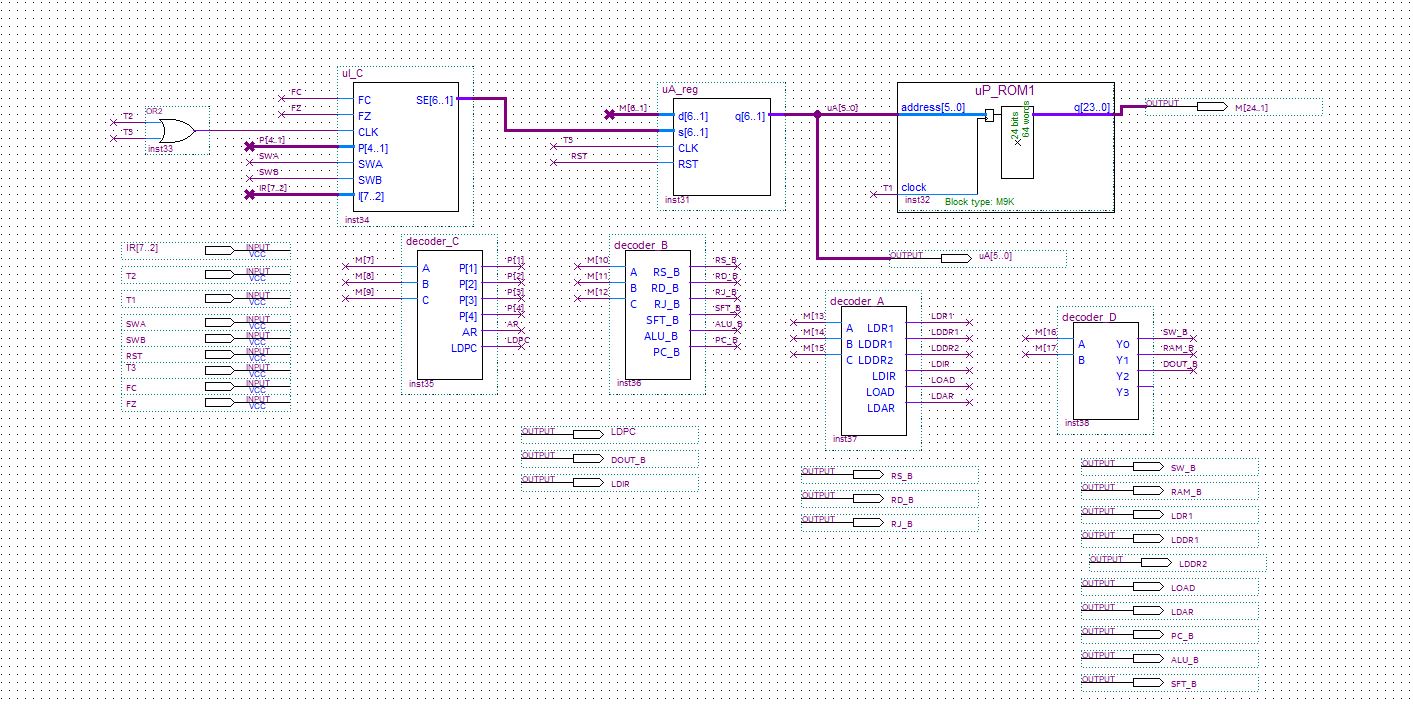
RST高电平，PC值被置零。T4、LDPC跳转，PC值加1。RAM\_B、LDIR、T2高电平，BUS读入RAM值25H，IR读入BUS值25H。PC\_B高电平，BUS读入PC值01H。T2、LDAR高电平，AR读入BUS值01H。随后PC++。PC\_B高电平，BUS读入PC值02H，LDAR、T2跳转，AR读入BUS值02H。RAM\_B、LDAIR、T2先后跳转，RAM值放入BUS再放入IR。PC\_B、LDAR、T2跳转，PC值传入AR。随后PC++。LOAD高电平、RAM\_B跳转、T2、LDPC跳转，RAM值被锁入PC中。随后的操作规律同理。

REGS\_MD规律如下。



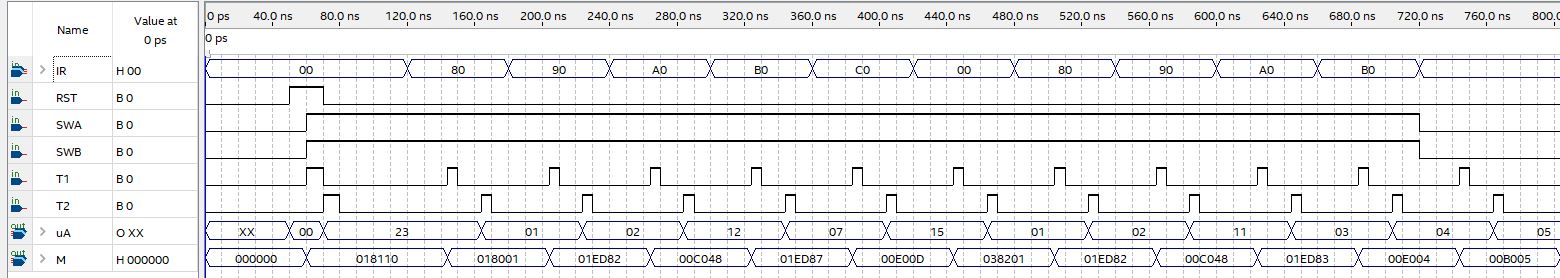
### 微指令控制模块uPC

微程序控制模块uPC是模型CPU中的重要部件。其中包括微地址控制模块ul\_C，即分支转移控制电路、微指令ABC字段译码电路模块decoder\_A\_B\_C、微操作译码器decoder\_D、微地址寄存器uA\_reg等。微程序控制器与外界的联系信号主要有24位微指令M[24..1]、指令寄存器输出信号IR[7..0]、操作台功能选择信号SWA和SWB、分支转移及进位标志信号FC和FZ，以及大量用于微控制的控制信号。译码器decoder\_A、decoder\_B、decoder\_C的作用是对24位微指令中的A、B、C字段进行指令译码。A字段译码后输出的信号主要用于控制向寄存器或锁存器输入数据；B字段译码后输出的信号主要用于控制运算器、寄存器或锁存器，并通过三态门向内部数据总线输出数据；C字段译码后输出的信号主要用于指令分支判断。decoder\_D译码器是对24位微指令中的第16位和第17位进行指令译码，产生对输入装置SW、存储器RAM、输出装置LED的输出允许控制信号。



#### 波形仿真：





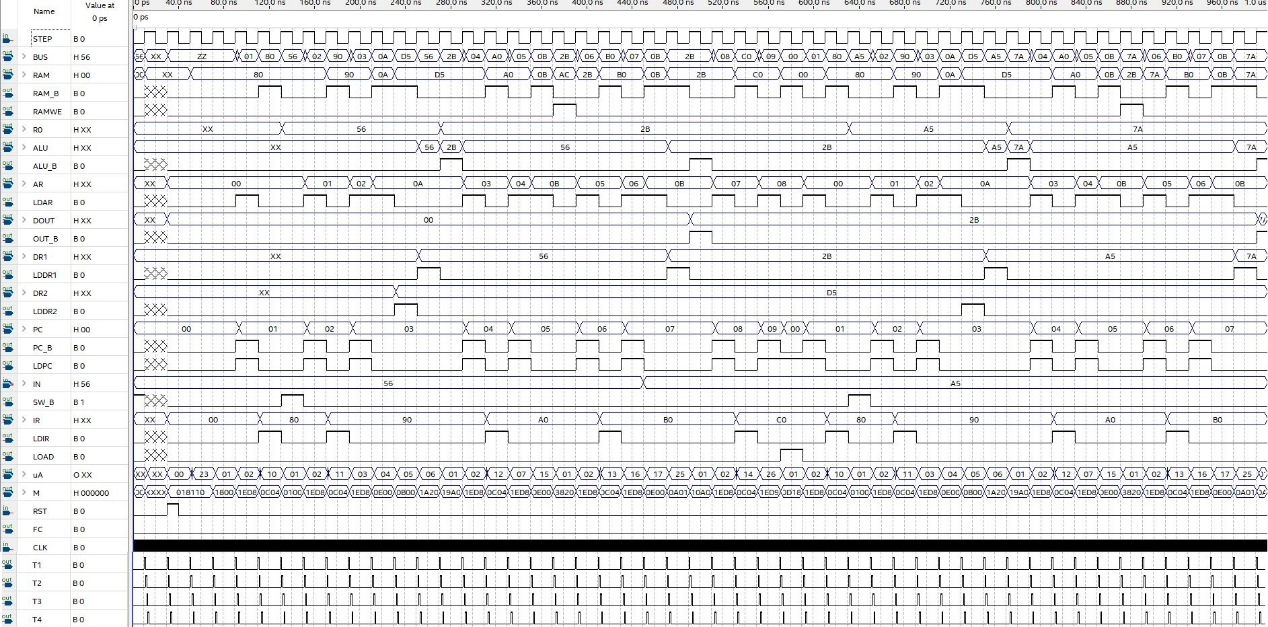
首先RST跳转，uA的值置零。T1跳转，M根据uA的值在uP\_ROM中找到并变为新的微地址018110B。T2跳转，根据最后两位10H得出测试前的uA值010000B。M[9..7]=100B通过Decoder\_C得出P[4..0]=0111B,得出S[5..0]=111100B，SE[i]=0,uA[i]则取1。修改后的uA=010011B（23O）。T1跳转，M根据新的uA值再次改变，循环往复。

uPC的规律如下。



### CPU顶层设计

#### 波形分析：





T1控制 微程序只读寄存器uP\_ROM、随机读取寄存器RAM8。

T2与LDAR、LDIR、DOUT\_B控制 数据流动（ALU181A的操作数，数据通路的门）

T3控制 微指令控制器ul\_C1（or T2）、微地址寄存器uA\_reg。

T4与LDPC控制 程序计数器PC。

RST高电平后，uPC、REG\_MD先后置零。

* T1时刻：M由uA的值00O读入uP\_ROM值018110H，随后M解码；信号更新；
* T2时刻：无；
* T3时刻：RST任为高电平，强制uA输出00O；
* T4时刻：无；
* T1时刻：uA为00O，M的值保持不变；信号更新（保持不变）；
* T2时刻：无；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址23O；
* T4时刻：无；
* T1时刻：M由uA的值23O读入uP\_ROM值018001H，随后M解码；信号更新；
* T2时刻：无；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址01O；
* T4时刻：无；
* T1时刻：M由uA的值01O读入uP\_ROM值01ED82H，随后M解码；

LDPC、PC\_B、LDAR高电平；

* T2时刻：LDAR==1，AR读入PC值00H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址02O；
* T4时刻：LDPC==1，PC值01H；
* T1时刻：M由uA的值02O读入uP\_ROM值00C048H，随后M解码；

LDIR、RAM\_B高电平；

RAM\_B==1，BUS根据AR读入RAM值80H；

* T2时刻：LDIR==1，IR读入BUS值80H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址10O；
* T4时刻：无；
* T1时刻：M由uA的值10O读入uP\_ROM值001001H，随后M解码；

SW\_B，LDRi高电平；

SW\_B==1，BUS读入IN值56H；

* T2时刻：IR[1..0]==0与LDRi ==1 => LDR0==1；

R0读入BUS值56H；

* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址01O；
* T4时刻：无；
* T1时刻：M由uA的值01O读入uP\_ROM值01ED82H，随后M解码；

LDPC、PC\_B、LDAR高电平；

* T2时刻：LDAR==1，AR读入PC值01H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址02O；
* T4时刻：LDPC==1，PC值02H；
* T1时刻：M由uA的值02O读入uP\_ROM值00C048H，随后M解码；

LDIR、RAM\_B高电平；

RAM\_B==1，BUS根据AR读入RAM值90H；

* T2时刻：LDIR==1，IR读入BUS值90H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址11O；
* T4时刻：无；
* T1时刻：M由uA的值02O读入uP\_ROM值00C048H，随后M解码；

LDIR、RAM\_B高电平；

RAM\_B==1，BUS根据AR读入RAM值90H；

* T2时刻：LDIR==1，IR读入BUS值90H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址11O；
* T4时刻：无；
* T1时刻：M由uA的值11O读入uP\_ROM值01ED83H，随后M解码；

LDPC、PC\_B、LDAR高电平；

PC\_B==1，BUS读入PC值02H；

* T2时刻：LDAR==1，AR读入BUS值02H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址03O；
* T4时刻：LDPC==1,PC值03H；
* T1时刻：M由uA的值03O读入uP\_ROM值00E004H，随后M解码；

RAM\_B、LDAR高电平；

RAM\_B==1，BUS根据AR值读入RAM值0AH；

* T2时刻：LDAR==1，AR读入BUS值0AH；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址04O；
* T4时刻：无；
* T1时刻：M由uA的值04O读入uP\_ROM值00B005H，随后M解码；

RAM\_B、LDDDR2高电平；

RAM\_B==1，BUS根据AR值读入RAM值D5H；

* T2时刻：LDDR2==1，DR2读入BUS值D5H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址05O；
* T4时刻：无；
* T1时刻：M由uA的值05O读入uP\_ROM值01A206H，随后M解码；

LDDR1、RS\_B高电平；

M[24..19] => ALU去DR1值56H；

* T2时刻：IR[3..2]==0&&RS\_B==1&& RD\_B==0 => R0\_B==1

BUS读入R0的值56H；

LDDR1==1，DR1读入BUS值56H；

* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址06O；
* T4时刻：无；
* T1时刻：M由uA的值06O读入uP\_ROM值919A01H，随后M解码；

M[24..19] => ALU=DR1+DR2；

ALU\_B、LDRi高电平；

ALU\_B==1，BUS读入ALU值2BH；

LDRi==1与IR[1..0]==0 => LDR0==1；

* T2时刻：LDR0==1， R0读入BUS值2BH；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址01O；
* T4时刻：无；
* T1时刻：M由uA的值01O读入uP\_ROM值01ED82H，随后M解码；

LDPC、PC\_B、LDAR高电平；

ALU取R0的值56；

* T2时刻：LDAR==1，AR读入PC值03H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址02O；
* T4时刻：LDPC==1，PC值04H；
* T1时刻：M由uA的值02O读入uP\_ROM值00C048H，随后M解码；

LDIR、RAM\_B高电平；

RAM\_B==1，BUS根据AR读入RAM值A0H；

* T2时刻：LDIR==1，IR读入BUS值A0H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址12O；
* T4时刻：无；
* T1时刻：M由uA的值12O读入uP\_ROM值01ED87H，随后M解码；

LDPC、PC\_B、LDAR高电平；

PC\_B==1 => BUS读入PC值04H；

* T2时刻：LDAR==1，AR读入BUS值04H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址07O；
* T4时刻：无；
* T1时刻：M由uA的值07O读入uP\_ROM值00E00DH，随后M解码；

RAM\_B、LDAR高电平；

RAM\_B==1 => BUS读入RAM值0BH；

* T2时刻：LDAR==1，AR读入BUS值0BH；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址15O；
* T4时刻：无；
* T1时刻：M由uA的值15O读入uP\_ROM值038201H，随后M解码；

WE高电平；

* T2时刻：LDAR==1，AR读入BUS值0BH；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址01O；
* T4时刻：无；
* T1时刻：RAM读入值2BH；

M由uA的值01O读入uP\_ROM值01ED82H，随后M解码；

LDPC、PC\_B、LDAR高电平；

PC\_B==1，BUS读入PC值05H；

* T2时刻：LDAR==1，AR读入BUS值05H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址02O；
* T4时刻：LDPC==1，PC值06H；
* T1时刻：M由uA的值02O读入uP\_ROM值00C048H，随后M解码；

LDIR、RAM\_B高电平；

RAM\_B==1，BUS读入RAM值B0H

* T2时刻：LDIR==1，IR读入BUS值B0H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址13O；
* T4时刻：无；
* T1时刻：M由uA的值13O读入uP\_ROM值01ED8EH，随后M解码；

LDPC、PC\_B、LDAR高电平；

PC\_B==1，BUS读入PC值06H；

* T2时刻：LDAR==1，AR读入BUS值06H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址16O；
* T4时刻：LDPC==1，PC值07H；
* T1时刻：M由uA的值16O读入uP\_ROM值00E00FH，随后M解码；

RAM\_B高电平；

RAM\_B==1，BUS读入RAM值0BH；

* T2时刻：无；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址17O；
* T4时刻：无；
* T1时刻：M由uA的值17O读入uP\_ROM值00A015H，随后M解码；

LDDR1高电平；

BUS读入R0值2BH；

* T2时刻：LDDR1==1，DR1读入BUS值2BH；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址25O；
* T4时刻：无；
* T1时刻：M由uA的值25O读入uP\_ROM值010A01H，随后M解码；

OUT\_B、ALU\_B高电平；

ALU\_B==1，BUS读入ALU值2BH；

* T2时刻：OUT\_B==1，DOUT读入BUS值2BH；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址01O；
* T4时刻：无；
* T1时刻：M由uA的值01O读入uP\_ROM值01ED82H，随后M解码；

LDPC、PC\_B、LDAR高电平；

PC\_B==1，BUS读入PC值07H；

* T2时刻：LDAR==1，AR读入BUS值07H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址02O；
* T4时刻：LDPC==1，PC值08H；
* T1时刻：M由uA的值02O读入uP\_ROM值00C048H，随后M解码；

LDIR、RAM\_B高电平；

RAM\_B==1，BUS读入RAM值C0H；

* T2时刻：LDIR==1，IR读入BUS值C0H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址14O；
* T4时刻：无；
* T1时刻：M由uA的值14O读入uP\_ROM值01ED96H，随后M解码；

LDPC、PC\_B、LDAR高电平；

PC\_B==1，BUS读入PC值08H；

* T2时刻：LDAR==1，AR读入BUS值08H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址26O；
* T4时刻：LDPC==1，PC值09H；
* T1时刻：M由uA的值26O读入uP\_ROM值00D181H，随后M解码；

LOAD、RAM\_B高电平；

RAM\_B==1，BUS读入RAM值00H；

* T2时刻：无；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码得计算新uA地址01O；
* T4时刻：LOAD==1，PC读入BUS值00H；
* T1时刻：M由uA的值01O读入uP\_ROM值01ED82H，随后M解码；

LDPC、PC\_B、LDAR高电平；

PC\_B==1，BUS读入PC值00H；

* T2时刻：LDAR==1，AR读入BUS值00H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址02O；
* T4时刻：LDPC==1，PC值01H；
* T1时刻：M由uA的值02O读入uP\_ROM值00C048H，随后M解码；

LDIR、RAM\_B高电平；

RAM\_B==1，BUS读入RAM值80H；

* T2时刻：LDIR==1，IR读入BUS值80H；
* T3时刻：M[6..1]、SWA、SWB、M[9..7]的解码计算得新uA地址10O；
* T4时刻：无；

………

# 本学期课程实验总结：

本学期由简入繁地进行了一系列计算机组成原理实验，从一开始需要按照课件跟随老师一步步进行实验，到现在可以熟练掌握Quartus软件的使用，初步理解了基础的VHDL语言。通过实验我们更加深入了解了计算机组成原理的内容，对于很多相对抽象的概念原理，有了一个更加具体的认知。

相较于对实验本身知识的理解，我们收获更多的是在实验过程中所出现的情感体悟。做实验最重要的是严谨细心，有的时候可能只是一个引脚命名的错误，就需要排查好几个小时。另外保持一个平稳的心态也是十分关键的，实验不可能一次成功，我们会面临很多问题，会一次又一次的出错，我们需要在不断的出错、改错中始终保持着一个良好的心态，才能够更加有效率的完成实验。

分工合作最重要的就是合作，两个虽然做着不同的工作，但是都要对对方的工作内容有着深入的了解，在实验过程中互相帮助，理性讨论才能够完成实验。

# 组内人员工作量划分：

尚文达：画原理图，波形仿真，通过仿真结果分析CPU与其各部分之间的相互作用及原理。

郑博文：撰写实验报告，通过书籍原理分析CPU与其各个部分的作用。

过程中两个人会对有疑问的地方多次探讨，共同分析。