

版权声明

本实验教程的版权归西安唐都科教仪器开发有限责任公司所有，保留一切权利。非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本实验教程的部分或全部，并以任何形式传播。

西安唐都科教仪器开发有限责任公司，1999-2003(C)，All right reserved.

计算机组成原理与系统结构实验教程

©版权所有 非经许可 严禁复制

技术支持邮箱: product-support@tangdu.com

唐都公司网址: <http://www.tangdu.com/>

前言

本书是为唐都科教仪器公司开发生产的“计算机组成原理及系统结构教学实验系统”配套的实验教程。本书实验项目丰富，内容完备。

“计算机组成原理及系统结构教学实验系统”是一套完全开放性的实验装置，通过实验，可使学生对计算机系统的基本原理有一个清晰的概念和认识，掌握设计计算机系统的原理与方法，能更好的培养学生的创新意识和设计能力。同时，可对新型计算机的体系结构方面也有一个比较深入的认识和理解。

本书分为八章，其中：第一章为系统认识实验，让学生对计算机整机的工作过程有一个感性的认识并对本实验系统有一个了解；第二章到第五章为部件实验，研究组成计算机的每个部件的工作原理及设计方法；第六章、第七章为计算机整机实验，通过对几种不同复杂程度的模型计算机的设计，来研究计算机各部件是如何来配合工作的，并掌握设计一个计算机系统的方法，在此基础上扩展到对输入输出系统方面的设计研究；第八章为计算机体系结构方面的设计和研究，让学生学习先进的计算机体系结构方面的原理和设计方法。

各学校可以根据自己的教学计划和教学特点选取教学内容。例如，对于书中应用大规模可编程逻辑器件 CPLD 的章节，需要读者具有 CPLD 器件及其设计方法等方面的基础知识；对于没有学习过“计算机接口”课程内容的，也可以通过第七章的输入输出系统部分来学习。

由于编者水平有限，加上计算机技术飞速发展，新的理念和技术层出不穷，在教材中肯定存在不少的问题和错误，恳请广大读者批评指正。

编者

2003 年 7 月

目 录

前言

第一章 概 论	1
1.1 计算机系统的基本组成	1
1.2 计算机系统的层次结构	3
1.3 教学实验系统认识	4
1.4 系统认识实验	22

第二章 运算器	29
2.1 基本的二进制加法器	29
2.1.1 全加器	29
2.1.2 并行加法器	30
2.2 并行加法器设计实验	32
2.3 定点乘法运算	36
2.3.1 原码一位乘法	36
2.3.2 补码一位乘法	37
2.3.3 阵列乘法	37
2.4 阵列乘法器设计实验	39
2.5 多功能算术逻辑运算单元 (ALU)	40
2.5.1 一位 ALU 逻辑	41
2.5.2 多功能算术逻辑运算单元 ALU	41
2.6 算术逻辑运算实验	43
2.7 进位控制实验	47
2.8 移位运算实验	49

第三章 存储系统	51
3.1 存储系统概述	51

3.1.1	存储器的分类	51
3.1.2	存储器的分级结构	52
3.1.3	存储器的主要技术指标	53
3.2	半导体存储器	52
3.2.1	静态随机存储器	52
3.2.2	动态随机存储器	56
3.2.3	半导体只读存储器	56
3.3	存储器的扩展	57
3.4	双端口存储器	59
3.5	高速缓冲存储器 (Cache)	59
3.6	静态随机存储器实验	60
3.7	FIFO 先进先出存储器实验	64
第四章 控制器		
67		
4.1	控制器的基本功能和结构	67
4.1.1	控制器的基本功能	67
4.1.2	控制器的组成	67
4.1.3	控制器的结构	68
4.2	时序控制信号	68
4.2.1	时序部件的组成	68
4.2.2	时序控制方式	69
4.3	微程序控制器	69
4.3.1	微程序控制器的原理及结构	69
4.3.2	微指令的编码方式	71
4.3.3	微指令的格式分类	71
4.3.4	后续微地址的形成方法	72
4.4	微程序控制器实验	72
4.5	硬布线控制器	79
4.5.1	硬布线控制器的基本原理及结构	79
4.5.2	硬布线控制器设计步骤	80
4.6	硬布线控制器实验	81
第五章 系统总线		
83		
5.1	总线的概念及分类	83
5.1.1	总线的基本概念	83

5.1.2 总线的分类·····	83
5.1.3 总线的连接方式·····	84
5.2 总线的通信方式·····	85
5.3 总线仲裁·····	86
5.4 总线基本实验·····	88
5.5 总线控制实验·····	90
第六章 中央处理器 ·····	
93	
6.1 CPU 的基本组成结构·····	93
6.1.1 中央处理器的功能·····	93
6.1.2 中央处理器的组成·····	93
6.1.3 寄存器组织·····	94
6.2 指令周期·····	94
6.3 指令系统·····	95
6.4 寻址方式·····	97
6.4.1 指令寻址方式·····	97
6.4.2 操作数的寻址方式·····	97
6.5 基本模型机设计实验·····	98
第七章 模型计算机及其设计 ·····	
106	
7.1 一台模型计算机的总体设计·····	106
7.2 复杂模型机设计实验·····	107
7.3 用 CPLD 实现模型计算机的设计实验·····	116
7.4 输入输出系统·····	119
7.4.1 输入输出系统概述·····	119
7.4.2 输入输出的基本控制方式·····	120
7.4.3 程序中断方式·····	121
7.5 具有中断处理功能的模型机设计实验·····	124
7.6 扩展 8255 并行口实验·····	134
7.7 扩展 8253 定时器/计数器实验·····	138
第八章 计算机系统结构的设计及研究 ·····	
142	
8.1 精简指令系统计算机·····	142
8.1.1 精简指令系统思想的提出·····	142

8.1.2 RISC 结构采用的基本技术	143
8.2 基于 RISC 处理器构成模型计算机实验	144
8.3 重叠处理机	151
8.3.1 重叠的原理及基本思想	151
8.3.2 相关处理	152
8.4 基于重叠技术构成的模型计算机实验	153
8.5 流水线处理机	163
8.5.1 流水线的原理及基本思想	163
8.5.2 相关处理	164
8.6 基于流水技术构成模型计算机的实验	166
附录 实验用芯片介绍	173

第一章 概论

一个完整的计算机是由硬件系统和软件系统共同组成的。它们是一个有机的整体，必须协同工作才能发挥计算机的作用。硬件是计算机的物理实体，它由运算器、控制器、存储器、输入设备、输出设备五大部件组成，一般又将运算器和控制器合称中央处理器（CPU）。软件是支持计算机硬件系统工作的程序的统称，它分为系统软件和应用软件两大类。计算机工作是由软件和硬件相辅相成共同来完成的，有些功能既可以由硬件来完成也可以由软件来完成。数字计算机普遍采用冯·诺依曼体制。

本章首先简单介绍了计算机系统的基本组成及冯·诺依曼体制的基本思想，然后从一个更高的角度来介绍计算机系统的层次划分；为了便于后续实验的学习，本章也安排了“计算机组成原理与系统结构教学实验系统”装置的结构特点及其软件使用的介绍，并通过一个系统认识实验来使你对计算机系统的基本组成及其如何工作产生一个感性认识，同时，对实验系统也有一个初步的了解。

1.1 计算机系统的基本组成

一、数字计算机的组成

一台典型的数字计算机是由五大部分组成的，即运算器、存储器、控制器、输入设备、输出设备。其基本硬件结构图如图 1.1-1 所示。

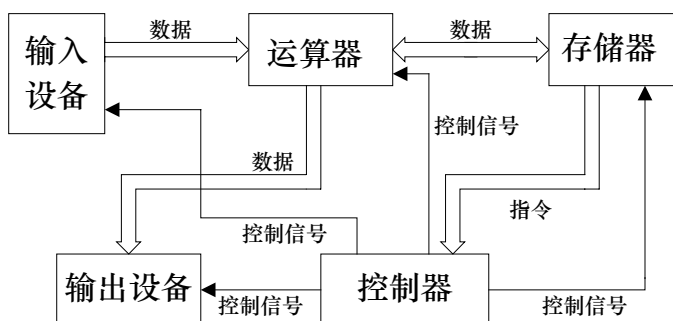


图 1.1-1 数字计算机的结构图

运算器：是用二进制进行算术和逻辑运算的部件。它由算术逻辑部件（ALU）和若干通用寄存器组成。它的主要功能是进行加、减、乘、除等算术运算和其他的逻辑运算。

存储器：是用来存放程序和数据的部件。存储器以单元为单位线性编址，按地址读写其单元。

输入/输出设备：计算机由输入设备接受外部信息，而通过输出设备来将信息送往外界。

控制器：控制器负责协调上述部件的操作，发出控制命令，是计算机的指挥中心。它从存储器中取出指令，进行分析，然后发出由该指令规定的一系列微操作命令，控制所有其他部件，来完成指令规定的功能。

通常，又把运算器和控制器合在一起称为**中央处理器**，即 **CPU**。

根据上图可看出，在计算机中，基本上有两股信息在流动：一种为数据，即各种原始数据、中间结果、程序等；而另一股即为控制命令，由控制器来控制装置的启动或停止、控制运算器按一定的步骤进行各种运算和处理、控制存储器进行读写、控制输出设备输出结果等。

所以，一个计算机的操作过程可以简单的归纳为以下几点：

- 1) 它通过输入设备接受信息，包括程序和数据，并将其传送到存储器中。
- 2) 经过控制器分析存放在存储器中的程序后将其中的数据信息读取到运算器进行处理。
- 3) 通过输出设备将处理的结果送到计算机的输出设备。
- 4) 在计算机内部所有的部件的活动都由控制器来指挥。

二．数字计算机是如何工作的

虽然计算机技术已经发展了几十年，计算机体系结构也发生了许多演变，但计算机还是普遍采用冯·诺依曼 (John Von Neumann) 原理为工作基础的。

冯·诺依曼体制中广泛采用的要点如下：

1) 采用二进制代码表示指令和数据

数据和指令在代码的形式上没有区别，都是以二进制形式数据 0 和 1 组成的，只是它们各自的含义不同。程序本身也可以作为被处理的对象，进行加工处理。

2) 采用存储程序方式

这是冯·诺依曼思想的核心内容。将事先编制好的程序（包括指令和数据代码）连续存放在存储器中，计算机在运行程序时就能够自动地、连续地从存储器中依次取出指令并且执行。这是计算机高速自动运行的基础。

冯·诺依曼的这种工作方式，可称为控制流（指令流）驱动方式。在这种方式下，按照指令的执行的序列，依次读取指令并根据所含有的控制信息，调用数据进行处理。因此在执行的过程中，始终以控制信息流为驱动工作因素，而数据信息流则是被动地被调用处理。

为了对指令流进行控制，通过设置一个程序计数器 PC (Program Counter)，用它来存放当前正在执行指令所在单元的地址。对于顺序执行的程序，每取出一条指令后 PC 的内容自动加 1，指向下一条指令的地址。对于转移程序，就将转移后的地址送入 PC 中，以便按新地址读取后续指令。所以，用 PC 就可以正确的指示并控制指令序列的执行顺序。

三．计算机系统的组成

作为一个完整的计算机系统，它应该由硬件系统和软件系统来共同组成。其组成关系如图 1.1-2 所示。

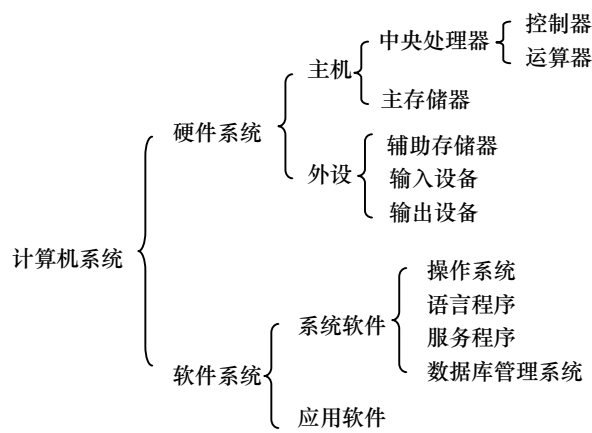


图 1.1-2 计算机系统组成关系图

硬件是计算机的物质基础，而软件是在这个基础上可以运行的各类程序和文件，它们实际上是由一些算法（说明如何完成某任务的指令序列就是算法的程序体现）以及他们在计算机中的表示所构成。

1. 2 计算机系统的层次结构

一. 计算机系统的多级层次结构

计算机系统以硬件为基础，通过配置软件扩充功能，形成一个可能是相当复杂的有机组合的系统。它通常由 4 个以上不同的级组成，每一级都能进行程序设计。如图 1.2-1 所示。

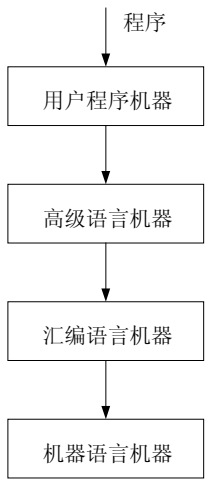


图 1.2-1 计算机系统的层次结构图

计算机最终所能执行的只能是二进制机器指令。机器语言机器是一个硬件级，从这一级可以看到一台实际的机器。它也是计算机软硬件的分界面，在它的下层是计算机的硬件，它的上层是各级的软件及汇编语言。

在机器语言级的上层是汇编语言级，汇编语言是和机器语言最接近的，它是用规定的一些助记符来编程，助记符和指令系统一一对应。由于计算机只能识别机器语言，所以运行时需要把汇编语言翻译成机器码程序，再在计算机上执行。汇编语言便于记忆，比直接用机器语言编程方便的多。

高级语言机器是用比汇编语言更高级的，与数学模型相近的算法语言甚至人类的自然语言编程。程序员在这级上可以不需要了解计算机的硬件、编译及操作系统等。

用户程序机是专用于一个具体的应用专门设计的应用软件，这里用户可以对计算机的硬件及软件不须了解而能方便的使用。

多数机器是将程序设计语言编写的程序直接翻译成机器语言进行执行，有些是先翻译成层次低一些的中间语言，然后再翻译成机器语言进行执行。

二. 软件与硬件的逻辑等价性

随着大规模集成电路技术的发展和软件硬化的趋势，要明确划分硬件与软件之间的界限已经显得比较困难了。有许多功能既可以由硬件来实现，也可以在硬件的支持下靠软件来实现，对用户来说在功能上是等价的。尤其现在 CPLD/FPGA 等技术的飞速发展，对于硬件的设计完全可以像设计软件一样方便，而且可以对所设计硬件功能随时修改，具有在线测试、仿真等功能，使这两者之间的区别更加模糊了。例如在本书后面的章节中要设计的一个乘法器实验，我们既可以用由基本单元提供的加法器和移位器功能部件的支持下，编写微程序，进行多次移位相加来实现乘法，也可以由 CPLD 直接设计一个具有乘法功能的运算部件。

一般来说，功能采用硬件实现的，速度上会比较快一些，但是需要增加成本，功能的变更周期也可能会长一些；采用软件实现的，虽然速度会慢一些，但不需要增加成本，功能也较容易改变。总之，选择恰当的软、硬件功能分配，这取决于所选定的设计目标、系统性能、价格、速度、可靠性、存储容量、变更周期等因素，并与当时的技术水平有关。

1. 3 教学实验系统认识

一. 系统功能及特点

计算机组成原理与系统结构教学实验系统是西安唐都科教仪器公司推出的一套高效的、开放性的教学实验系统，该系统可以通过对多种原理性计算机的设计、实现和调试来高效率地支持“计算机组成原理”和“计算机系统结构”等课程的开放式实验教学，为高校各个教学层次的计算机原理教学提供了完善的解决方案。

系统有如下功能特点：

1. 结构清晰的单元式实验电路，可构造出不同结构及复杂程度的原理性计算机

系统采用部件单元式结构，包括运算器及数据通路、存储器、控制器、信号及时序控制、

内总线、外总线、外围接口及输入输出设备、大规模可编程逻辑器件等计算机部件的单元电路, 用户可使用排线连接方式或计算机电子自动逻辑设计方式, 根据自己所设计的模型计算机结构方案, 来构造出不同结构及复杂程度的原理性计算机, 使学生能够对计算机组成结构有清楚的认识和理解。

2. 对实验设计具有完全的开放性, 增强学生综合设计能力

系统所具有的软硬件结构对用户的实验设计具有完全的开放性, 其数据线、地址线、控制线都由用户来操作连接, 系统中的运算器结构、控制器结构及微程序指令的格式及定义均可由用户根据教学需要来做灵活改变或重新设计。这对于用户自行设计各种结构及不同复杂程度的模型计算机提供了强大的软硬件操作平台, 从而避免了单纯验证性的实验模式, 极大提高了学生计算机系统的综合设计能力。

3. 通用逻辑器件和大规模可编程逻辑器件相结合, 可面向不同层次的学生

系统采用通用逻辑器件和大规模可编程逻辑器件并用的方式, 既能给熟练掌握复杂逻辑系统设计的学生提供高档的实验平台, 又能对不熟悉这些内容为学生提供易操作的实验平台。符合循序渐进、先基础后提高的教学原则。

4. 具有实时调试功能的图形方式操作界面, 也可用于多媒体辅助教学

系统具有与 PC 微机联机实时调试的功能, 提供了图形方式的调试界面, 在调试过程中可动态实时显示模型计算机各部件之间的数据传送以及各部件和总线上的所有信息。这种图形调试界面也可用于多媒体辅助教学, 从而获得极佳的教学效果。

5. 多种输入输出方式及逻辑信号测量功能, 实验操作及观察更容易

系统提供多种输入输出方式。通过 RS-232 串口与 PC 微机联机, 可在 PC 机上进行编程并向系统装载实验程序, 在图形界面下进行动态调试及运行。另外还具有两路逻辑信号测量平台, 可在 PC 机上看到信号测量波形; 如单独使用本系统, 则可通过开关及 LED 以二进制码形式进行编程、显示及调试运行。

6. 实验电路的实时在线检测功能, 便于检查接线错误

系统具有实验电路检测功能, 通过人机交互方式可实时在线检测各实验单元电路的好坏以及模型机实验线路连接是否正确。

二. 系统与 PC 联机说明

实验系统安装有一个标准的 DB 型 9 针 RS-232C 串口插座, 使用配套的串行通讯电缆分别插在实验系统及 PC 微机的串口, 即可实现系统与 PC 的联机操作。系统配套的集成操作软件具有专为联机操作而开发的图形方式操作界面, 其操作简便、直观且具有动态调试功能, 可以完全根据实验系统的数据通路图来实时、动态的显示用户设计的实验数据流的流向、数据值、控制线和各单元的内容。

本系统软件是通过 PC 机串行口向实验系统上的单片机控制单元发送指令, 由实验系统的单片机直接对程序存储器、微程序控制器进行读写, 控制单拍或单步微程序、单步机器指令和程序连续运行等操作, 实时监测各数据流和控制流的情况, 从而实现实时动态图形方式下的系统跟踪调式和运行。系统通讯电缆连接方式如图 1.3-1:

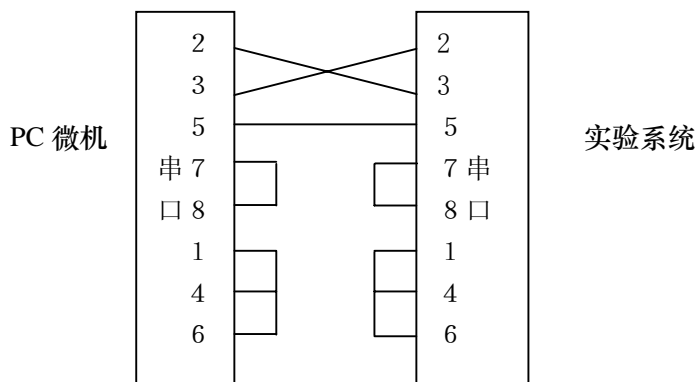


图 1.3-1 PC 机和实验系统用串行口连接方式

三. 软件的安装与卸载

软件运行环境

操作系统：中英文 Windows 95/98/ 2000/ NT/ME

最低配置：

CPU：奔腾 133Mhz

内存：16MB

显示卡：标准 VGA，256 色显示模式以上

硬盘：15MB 以上

驱动器：2X 倍速 CD-ROM 以上

其它设备：鼠标器

建议配置：

CPU：奔腾 166 或更高

内存：16MB 以上

显示卡：SVGA，16K 色以上显示模式，分辨率 800×600。

其它设备同“最低配置”

安装软件

安装操作如下：

可以通过“资源管理器”，找到光盘驱动器本软件安装目录下的 Setup.EXE，双击执行它，按屏幕提示进行安装操作。

“TDN-CM++1.03(W)”安装成功后，在“开始”的“程序”里将出现“CMPP”程序组，点击“CMPP”即可执行程序。

启动软件

软件的启动方式有三种：

用户可以在【开始】/【程序】菜单中单击“CMPP”的程序组启动。

用户也可以在【开始】/【程序】/【启动】菜单中启动“CMPP”。

用户在安装“TDN-CM++1.03(W)”以后桌面上自动出现“CMPP”快捷键，用户直接在桌面上双击快捷键就可以启动该程序组。

卸载软件

联机软件提供了自卸载功能，使您可以方便地删除“TDN-CM++1.03(W)”的所有文件、程序组或快捷方式。单击【开始】/【程序】打开“CMPP”的程序组，然后运行“卸载”项，就可执行卸载功能，按照屏幕提示操作即可以安全、快速地删除“TDN-CM++1.03(W)”。

四．功能介绍

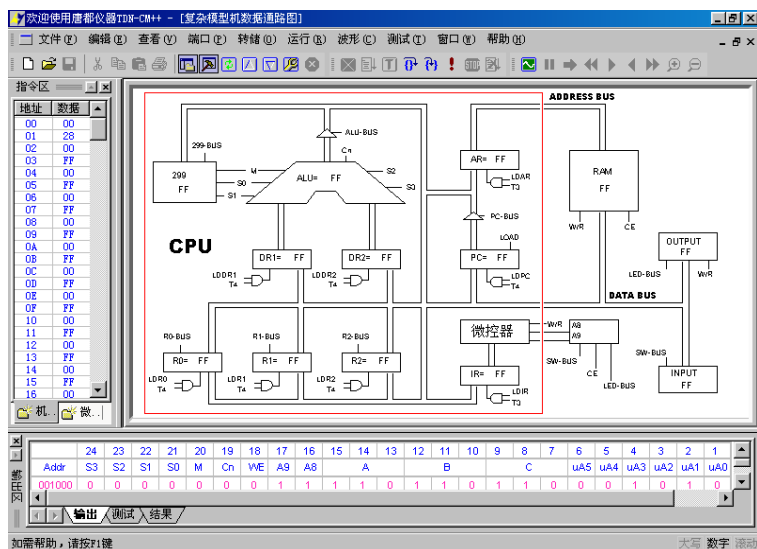
(一) 界面窗口介绍

主界面主要分为三部分：指令区、输出区和图形区，下面分别加以介绍。

指令区：

分为两部分，即机器指令区和微指令区，在指令区的下方有两个 Tab 按钮，您可以通过按钮在两者之间来回切换。

机器指令区：分为两列，第一列为下位机主存地址（00—FF，共 256 个单元），第二列为每一地址中所对应的数值。如果串口通讯正常且系统不忙（即串口没有被占用），您可以直接修改指定单元的内容，方法是用鼠标单击要修改单元的数据，此单元格会变成一个编辑框，等待您输入，该编辑框只接收两位合法的 16 进制数（请注意：非 16 进制数不认），如果输入正确，您可以按回车键确认，或用鼠标点击别的区域，这样就完成了修改工作。如果想要结束修改，您可以按下 ESC 键，编辑框就会自动消失，恢复显示原来的值。一旦编辑框出现，您可以通过上下键让编辑框上下移动，从而选中需要修改的地址单元。如果输入不正确，如输入少于 2 个字符，则不进行修改。



微指令区：分为两列，第一列为下位机微控器地址（00—3F，共 64 个单元），第二列为

每一地址中所对应的微指令，共 6 字节，对应微控器的微指令 24 位。如果串口通讯正常且系统不忙（即串口没有被占用），您可以直接修改指定单元的内容，方法是用鼠标单击要修改单元的数据，此单元格会变成一个编辑框，等待您输入，该编辑框只接收 6 位合法的 16 进制数（请注意：非 16 进制数不认），如果输入正确，您可以按回车键确认，或用鼠标点击别的区域，这样就完成了修改工作。如果想要结束修改，您可以按下 ESC 键，编辑框就会自动消失。一旦编辑框出现，您可以通过上下键让编辑框上下移动，从而选中需要修改的地址单元。如果输入不正确，如输入少于 6 个字符，则不进行修改。

输出区：

分为三页：输出页、测试页和结果页。

输出页：在打开复杂模型机数据通路图或重叠模型机数据通路图，并运行程序时用来显示下条将要执行的 24 位微码及其微地址，这是和下位机一起实时变动的。

测试页：在您进行复杂模型机系统测试时为您提供信息。显示当前下位机正在测试的单元及测试结果。

结果页：基本上是一个公共区域，用来显示一些提示信息或一些错误信息，如 RISC 模型机中，如果上位机检测到下位机运行有误，就会在这一区域加以显示，为您提供信息。保存或装载程序时也会在这一区域为您提供一些提示信息。

图形区：

这一区域是您操作的主要区域，您可以在此区域编辑相应的指令，可以显示各个模型机的数据通路图，可以打开示波器界面等。

（二）功能菜单介绍

1. 文件菜单项：

文件菜单提供了以下命令：

新建 建立一个新文档。

打开 打开一个现存文档。

关闭 关闭一个打开的文档。

保存 用同样的文件名保存一个打开的文档。

另存为 用指定的文件名保存一个打开的文档。

打印 打印一个文档。

打印预览 在屏幕上按被打印出的格式显示文档。

打印设置 选择一个打印机以及打印机连接。

退出 退出 CMPP。

①. 新建 (N)：

用此命令在 CMPP 中建立一个新文档。在文件新建对话框中选择您所要建立的新文件的类型。



②. 打开 (O)

用此命令在一个新的窗口中打开一个现存的文档。您可同时打开多个文档。您可用窗口菜单在多个打开的文档中切换。

③. 关闭 (C)

用此命令来关闭包含活动文档的所有窗口。CMPP 会建议您在关闭文档之前保存对您的文档所做的改动。如果您没有保存而关闭了一个文档，您将会失去自从您最后一次保存以来所做的所有改动。在关闭一无标题的文档之前，CMPP 会显示另存为对话框，建议您命名和保存文档。

④. 保存 (S)

用此命令将活动文档保存到它的当前的文件名和目录下。当您第一次保存文档时，CMPP 显示另存为对话框以便您命名您的文档。如果在保存之前，您想改变当前文档的文件名和目录，您可选用另存为命令。

⑤. 另存为 (A) ...

用此命令来保存并命名活动文档。CMPP 会显示另存为对话框以便您命名您的文档。

⑥. 打印 (P) ...

用此命令来打印一个文档。在此命令提供的打印对话框中，您可以指明要打印的页数范围、副本数、目标打印机，以及其它打印机设置选项。

⑦. 打印预览 (V)

用此命令按要打印的格式显示活动文档。当您选择此命令时，主窗口就会被一个打印预览窗口所取代。这个窗口可以按它们被打印时的格式显示一页或两页。打印预览工具栏提供选项使您可选择一次查看一页或两页，在文档中前后移动，放大和缩小页面，以及开始一个打印作业。

⑧. 打印设置 (R) ...

用此命令来选择一台打印机和一个打印机连接。在此命令提供的打印设置对话框中，您可以指定打印机及其连接。

⑨. 最近使用文件

您可以通过此列表，直接打开最近打开过的文件，共四个。

⑩. 退出 (X)

用此命令来结束您 CMPP 的运行阶段。您也可使用在应用程序控制菜单上的关闭命令。

2. 辑菜单项:

编辑菜单提供了以下命令:

撤销 撤销先前的编辑操作。

剪切 从文档中删除数据并将其移到剪贴板上。

复制 从文档中将数据复制到剪贴板上。

粘贴 从剪贴板上将数据粘贴到文档中。

	撤销 (U)	Ctrl+Z
	剪切 (T)	Ctrl+X
	复制 (C)	Ctrl+C
	粘贴 (V)	Ctrl+V

①. 撤销 (U)

如果可能的话，可用此命令来撤销上一步编辑操作。该命令名会根据您所执行的上一步操作而变化。如果您无法撤销上一步操作，菜单上的撤销命令会变成‘无法撤销’。

②. 剪切 (T)

用此命令将当前被选取的数据从文档中删除并放置于剪贴板上。如当前没有数据被选取时，此命令则不可用。

③. 复制 (C)

用此命令将被选取的数据复制到剪贴板上。如当前无数据被选取时，此命令则不可用。

④. 粘贴 (P)

用此命令将剪贴板上内容的一个副本插入到插入点处。如剪贴板是空的，此命令则不可用。

3. 查看菜单项:

查看菜单提供了以下命令:

工具栏 显示或隐藏工具栏。

状态栏 显示或隐藏状态栏。



①. 工具栏 (T)

a. 标准工具栏 (T)

用此命令可显示和隐藏标准工具栏。标准工具栏包括了 CMPP 中一些最普通命令的按钮，如文件打开。在工具栏被显示时，一个打勾记号出现在该菜单项目的旁边。

b. 指令区 (W)

用此命令可显示和隐藏指令区。

c. 输出区 (O)

用此命令可显示和隐藏输出区。

d. 自定义 (C)

见自定义项。

②. 状态栏 (S)

此命令可用来显示和隐藏状态栏。状态栏描述了被选取的菜单项目或被按下的工具栏按钮，以及键盘的锁定状态将要执行的操作。当状态栏被显示时，在菜单项目的旁边会出现一个打勾记号。

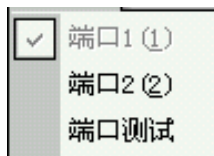
4. 端口菜单项：

端口菜单提供了以下命令：

端口 1 选择 1 号串口进行联机。

端口 2 选择 2 号串口进行联机。

端口测试 对当前选择的串口进行联机测试。



①. 端口 1 (1)

此命令用来选择串口 1 进行联机通讯，该命令会对串口 1 进行初始化操作，并进行联机测试，报告测试结果，如果联机成功，则会将指令区初始化。

②. 端口 2 (2)

此命令用来选择串口 2 进行联机通讯，该命令会对串口 2 进行初始化操作，并进行联机测试，报告测试结果，如果联机成功，则会将指令区初始化。

③. 端口测试

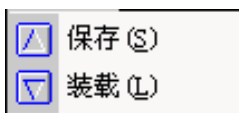
此命令用来对当前选择的串口进行联机通讯测试，并报告测试结果，只测一次，如果联机成功，则会将指令区初始化。

5. 转储菜单项：

转储菜单提供了以下命令：

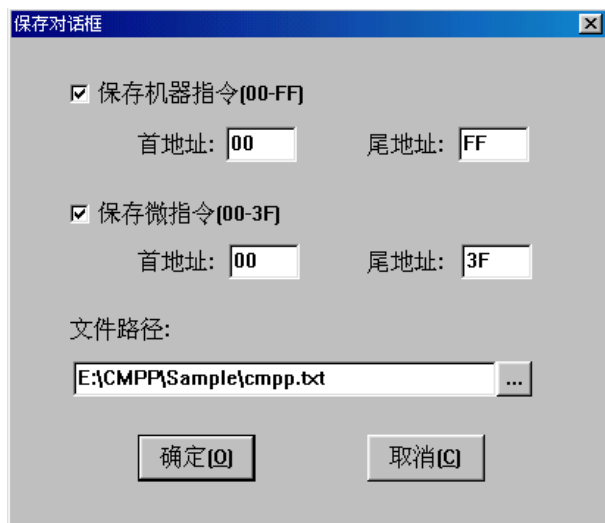
保存 将下位机中指令数据保存到上位机。

装载 将上位机中指令数据装载到下位机。



①. 保存 (S)

此命令将下位机中（主存，微控器）的数据保存到上位机中，您选择该命令会弹出一个保存对话框，如下图：



可以选择保存机器指令，此时首尾地址输入框将会变亮，否则首尾地址输入框将会变灰，在允许输入的情况下您可以指定需要保存的首尾地址，微指令也是如此。保存的数据以固定格式存入*.TXT 格式的文件中，文件的路径由您指定。机器指令格式为：\$P00FF，“\$”为标记号，“P”代表机器指令，“00”为机器指令的地址，“FF”为该地址中的数据。微指令格式为：\$M00AA77FF，“\$”为标记号，“M”代表微指令，“00”为机器指令的地址，“AA77FF”为该地址中的数据。

②. 装载 (L)

此命令将上位机指定文件中的数据装载到下位机中，您选择该命令会弹出一个打开文件对话框，如下图：



可以打开任意路径下的*.TXT 文件，如果是合法的指令文件，系统将把这些指令装载到下位机中，装载指令时，系统提供了一定的检错功能，如果指令文件中有错误的指令，将会导致系统退出装载，并提示错误的指令行。

6. 运行菜单项：

运行菜单提供了以下命令：

通路图	选择适当的数据通路图。
单节拍	单节拍调试。
单周期	单周期调试。
单步微指令	单步微指令调试。
单步机器指令	单步机器指令调试。
连续	连续运行。
停止	停止运行。
退出单节拍	退出单节拍运行模式。
流动速度	对数据通路图中数据的流动速度进行调节。



①. 通路图

此命令用于选择需要的数据通路图，您选择该命令会弹出一个数据通路图选择对话框，如下图：

系统为您提供了四个数据通路图，以满足不同的实验的需要，但是每次只允许打开一个数据通路图（因为数据通路图需要占用串口）。

②. 单节拍 (P)

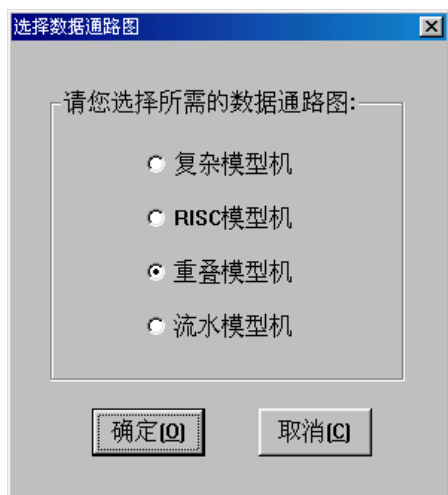
此命令用于向下位机发送单节拍命令，每发一次下位机将会完成一个节拍的工作。

③. 单周期 (T)

此命令用于向下位机发送单周期命令，每发一次下位机将会完成一个周期的工作。

④. 单步微指令 (C)

此命令用于向下位机发送单步微指令命令，每发一次下位机将运行完一条微指令。



⑤. 单步机器指令 (M)

此命令用于向下位机发送单步机器指令命令，每发一次下位机将会运行完一条机器指令。

⑥. 连续 (R)

此命令用于向下位机发送连续运行命令，一旦发下该命令，下位机将会进入连续运行状态，直到您发送停止命令。

⑦. 停止 (S)

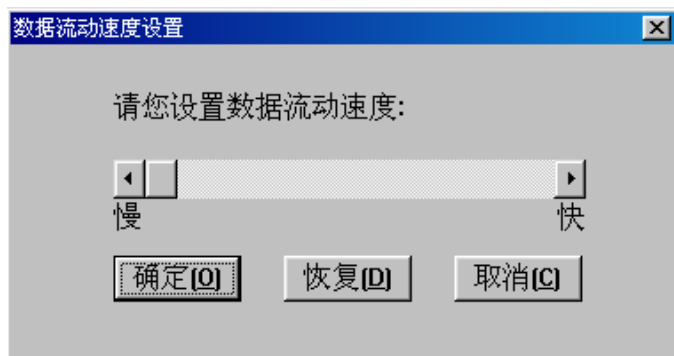
如果您已经发送了连续运行命令，使得下位机进入连续运行状态，那么此命令可以使得下位机停止运行，每次都需将当前指令周期运行完成后才能停止运行。

⑧. 退出单节拍 (E)

此命令项用于退出单节拍运行模式，如果程序运行在单节拍模式下，是不允许用户直接修改机器指令和微指令的，只有在其退出单节拍运行模式后才能直接修改。其实退出单节拍模式就是将本周期运行完毕，当用户选择单步机器指令或是连续运行时将会自动退出单节拍运行模式。

⑨. 流动速度 (L)

此命令用于指定数据通路图中数据的流动速度，您选择该命令会弹出一个流动速度设置对话框，如下图：



可以通过设置滑动块的位置来调节数据的流动速度，按下恢复按钮将会设置成默认值。

7. 波形菜单项：

波形菜单提供了以下命令：

启动	启动示波器。
放大	放大波形。
缩小	缩小波形。
暂停	暂停波形数据采集。
继续	继续采集波形数据。
左移	使游标左移一个单位。
右移	使游标右移一个单位。
快速左移	使游标左移五个单位。
快速右移	使游标右移五个单位。



①. 启动 (R)

用此命令用来启动 CM++双踪逻辑示波器，下位机已启动示波器，该命令项变灰。

②. 放大 (M)

用此命令用来放大波形的显示。

③. 缩小 (L)

用此命令用来缩小波形的显示。

④. 暂停 (S)

用此命令用来暂停波形数据的采集，稳定波形的显示，并显示游标。

⑤. 继续 (C)

用此命令用来继续采集波形数据，并使游标消失，实时显示波形。

⑥. 左移 (N)

在波形暂停状态下此命令用来左移游标，每发一次该命令，游标左移一个单位。

⑦. 右移 (M)

在波形暂停状态下此命令用来右移游标，每发一次该命令，游标右移一个单位。

⑧. 快速左移

在波形暂停状态下此命令用来快速左移游标，每发一次命令，游标左移五个单位。

⑨. 快速右移

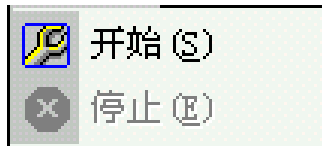
在波形暂停状态下此命令用来快速右移游标，每发一次命令，游标右移五个单位。

8. 测试菜单项：

测试菜单提供了以下命令。

开始 开始启动系统测试。

停止 停止系统测试。



①. 开始 (S)

如果您启动的是复杂模型机的数据通路图，此命令可以启动系统测试，系统测试将会逐单元地测试复杂模型机系统，并报告测试结果。

②. 停止 (E)

如果您已启动了系统测试功能，用此命令可以停止系统测试。

9. 窗口菜单项：

窗口菜单提供了以下命令。这些命令使您能在应用程序窗口中安排多个文档的多个视图：

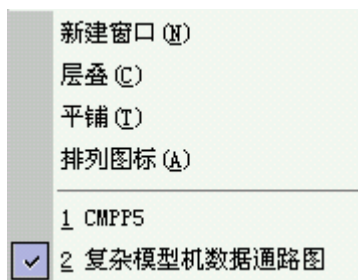
新建窗口 建立一个查看同样文档的新窗口。

层叠 按重叠方式安排窗口。

平铺 按互不重叠平铺方式安排窗口。

安排图标 安排已关闭窗口的图标。

转到指定的窗口。



①. 新建窗口 (N)

用此命令来打开一个具有与活动的窗口相同内容的新窗口。您可同时打开数个文档窗口以显示文档的不同部分或视图。如果您对一个窗口的内容做了改动，所有其它包含同一文档的窗口也会反映出这些改动。当您打开一个新的窗口，这个新窗口就成了活动的窗口并显示于所有其它打开窗口之上。

②. 层叠 (C)

用此命令按相互重叠形式来安排多个打开的窗口。

③. 平铺 (T)

用此命令按互不重叠形式来安排多个打开的窗口。

④. 排列图标 (A)

用此命令在主窗口的底部安排被最小化的窗口的图标。如果在主窗口的底部有一个打开的窗口，则有可能会看不见某些或全部图标，因为它们在这个文档窗口的下面。

⑤. 窗口选择

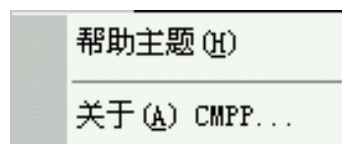
CMPP 在窗口菜单的底部显示出当前打开的文档窗口的清单。有一个打勾记号出现在活动的窗口的文档名前。从该清单中挑选一个文档可使其窗口成为活动窗口。

10. 帮助菜单项：

帮助菜单提供以下的命令，为您提供使用这个应用程序的帮助：

帮助主题 提供您可从其得到帮助的主题索引。

关于 显示这个应用程序的版本号。



①. 帮助主题 (H)

用此命令来显示帮助的开场屏幕。从此开场屏幕，您可跳到关于使用 CMPP 的一步指令以及各种不同类型参考资料。

②. 关于 (A) CMPP...

用此命令来显示您的 CMPP 版本的版权通告和版本号码。

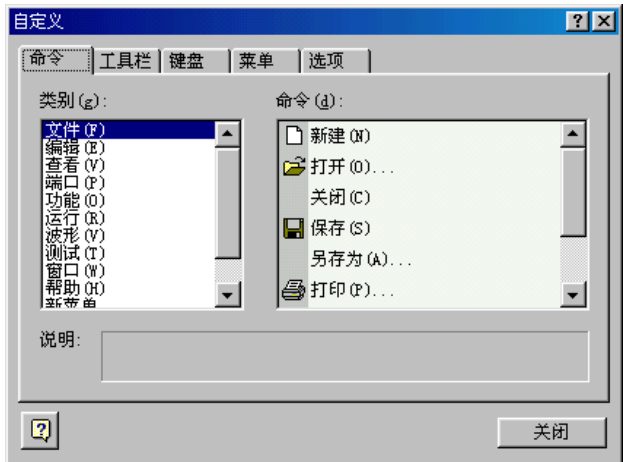
11. 关于自定义功能介绍

系统为您提供了方便的快捷键设置方式，您可以通过自定义的方法定做自己的操作界面。自定义设置是一个对话框，此对话框上有几个属性页，分别为：命令、工具栏、键盘、菜单、选项，下面分别加以介绍。

命令属性页：其页面如下：

该属性页主要三项：类别、命令、说明。

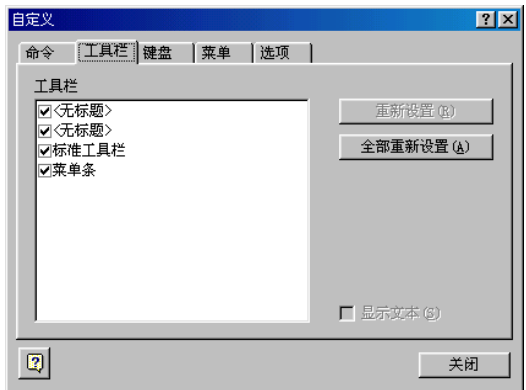
类别：列出了应用程序所有的菜单项。



命令：列出了选中的菜单项中所对应的菜单命令。如上图选中的是“文件”菜单项，则在“命令”栏中列出了“文件”项所对应的菜单命令。

说明：如果选中了一个菜单命令，则在说明栏中注明该菜单命令的作用。

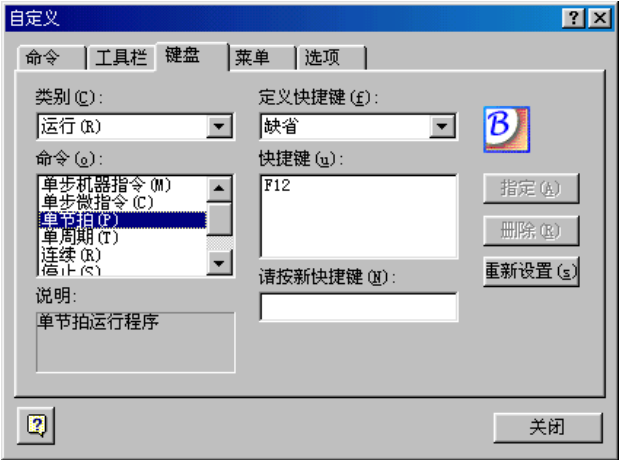
工具栏属性页：其页面如下：



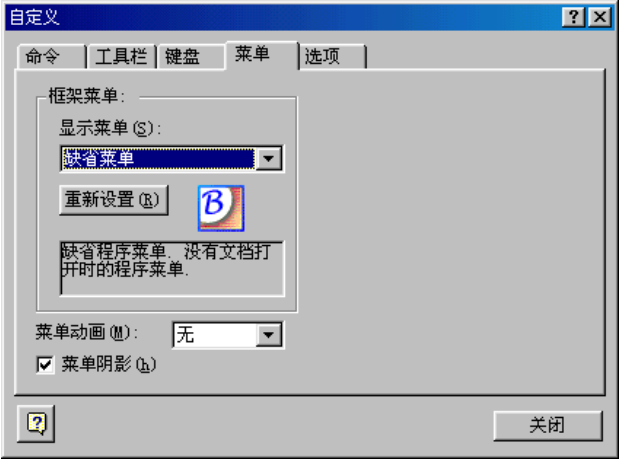
这一属性页用来设置工具栏，系统提供了三个工具栏，您可以在这里控制其显示或隐藏。并且，如果选中某一工具栏，可以勾选右下角的“显示文本 (S)”，使每一按钮下都显示该按钮的名称。

键盘属性页：其页面如下：

该属性页用来设置菜单命令的快捷键。方法是：在“类别”栏选择菜单项，然后在“命令”栏选择需设置快捷键的菜单命令，此时说明栏会列出该菜单命令的作用，如果该菜单命令已设置了快捷键，则在“快捷键”栏会显示已设置的快捷键。如要设置新的快捷键，则可在“请按新快捷键”栏中键入新的快捷键，然后按下指定按钮即可。如要删除已有的快捷键，只要在“快捷键”栏中选中需删除的快捷键，然后按下删除按钮即可。



菜单属性页：其页面如下：

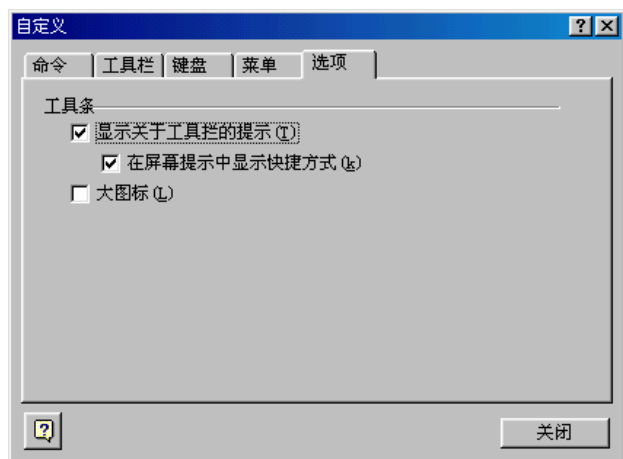


在该属性页中，您可以设置程序运行时主框架的菜单，但建议用默认值，也可以设置菜单展开时的动画，以及是否产生菜单阴影。

选项属性页：其页面如下：

该属性页用来设置工具栏的一些特性，如果勾上“显示关于工具栏的提示 (T)”，使鼠标在某一按钮上停留片刻，则会显示该按钮的作用提示条。如果勾上“在屏幕提示中显示快捷方式 (K)”，则在显示按钮作用提示条时还会显示该按钮的快捷键。

大图标：使工具栏按钮显示为大图标。



五. 实验程序清单

在安装程序后，系统会将实验指导书中的实验程序对应的存盘文件拷贝到安装目录下\CMPP\Sample 目录中，其对应文件分别为：

EX.TXT	计算机系统认识实验机器指令及微指令存盘文件
EX1.TXT	简单模型机实验机器指令及微指令存盘文件
EX2.TXT	带进位移位模型机实验机器指令及微指令存盘文件
EX3.TXT	复杂模型机实验机器指令及微指令存盘文件
8255.TXT	扩展 8255 实验机器指令及微指令存盘文件
8253.TXT	扩展 8253 实验机器指令及微指令存盘文件
8259.TXT	具有中断处理功能模型机实验机器指令及微指令存盘文件
RISC.TXT	基于 RISC 处理器构成的模型机机器指令存盘文件
CHD.TXT	基于重叠技术构成的模型机实验机器指令及微指令存盘文件
LSH.TXT	基于流水技术构成的模型机实验机器指令存盘文件

对于实验中的所有 CPLD 应用实验的源程序也都安装于\CMPP\CPLD 目录中，其对应文件目录分别为：

ALU_ABL	以 ABEL 语言描述的并行加法器实验
ALU_P	以原理图描述的并行加法器设计实验
MULTIPLY	乘法器实验
1032FIFO	FIFO 实验
COM_LOC	硬布线控制器实验
BUS_CTR	总线控制实验
MICROP	用 CPLD 实现模型计算机的设计实验
8259	8259 实验
RISC	RISC 实验

CHD1032	重叠实验
LSH1032	流水实验

六. 注意问题

在使用的过程中可能会碰到一些常见问题，现列出来须加以注意：

1. 启动应用程序时报告串口初始化失败。

这说明当前串口已经被别的应用程序占用，或该串口不存在，您可以关闭占用串口的程序或换一个串口试试，如果提示问题依旧，请重新启动计算机再运行程序。

2. 程序启动时报告串口通讯失败。

这说明当前程序已完成对串口的初始化工作，但是上位机和下位机的连接不正常，请确认下位机电源已经打开，串口线两端接触良好，上位机串口连接在软件设定的串口号上，一切没问题后再进行一次串口测试，可以得到测试报告。

3. 测试功能不能使用。

因为系统测试功能只针对复杂模型机数据通路图而言，所以只有在您打开复杂模型机数据通路图后才可以进行系统测试，系统测试应在下位机按实验要求将实验连线连好后才进行。

4. 示波器游标不能快速移动。

实际上您可以通过鼠标将游标拽动任何合法的位置，当您将鼠标光标移动到游标附近时光标会变成拽取状，此时按下鼠标左键就可以拖动游标了。

5. 输出区和指令区不能相互重叠。

在拖动输出时按下 Ctrl 键，就可以将两个区域重叠起来了。

6. 单节拍运行程序之后不能修改机器指令或微指令。

在单节拍运行模式下是不允许您修改机器指令或微指令的，只有在退出单节拍模式后才能修改，退出单节拍模式就是将当前指令周期执行完。

7. 在连续运行程序时按下停止时没有立即停止运行。

在复杂模型机通路图中连续运行是以单步机器为单位的，所以即使按下停止运行按钮，系统也不会立即停止，而是要将当前机器指令执行完。在其他的通路图中，连续运行是以指令周期为单位的，因按下停止运行按钮后，系统会将当前指令周期执行完后才停止。

8. 示波器测量波形时显示不稳定。

在这种情况下您可以通过放大或缩小波形来加以调节，当然最好的办法就是暂停波形显示，这样显示的波形就不会闪动。

9. 做基于流水技术构成模型计算机的实验时注意问题

在做基于流水技术构成模型计算机的实验时，由于本实验不需要微程序，当每次选择流水数据通路图时系统先将微程序的 00H、01H 两个单元的内容改为所需要的值。而实验中对于这两个单元中的微程序不能随意改变，软件需要使用。

1. 4 系统认识实验

一. 实验目的

1. 搭建并操作一个最基本的模型计算机。
2. 建立对计算机组成及其原理的基本认识。

二. 实验设备

1. TDN-CM+或 TDN-CM++教学实验系统一套。
2. PC 微机一台。

三. 实验原理

1. 一台简单模型计算机的结构

为了更好地理解计算机的各组成部件是如何相互配合进行工作的，我们将设计一个最基本的模型计算机。根据前面小节的知识，我们将算术逻辑运算器、控制器、寄存器、内部总线等部件搭接起来构成一个 CPU，然后再加上存储器、输入设备、输出设备即构成一台完整的模型计算机。其逻辑框图见图 1.4-1。

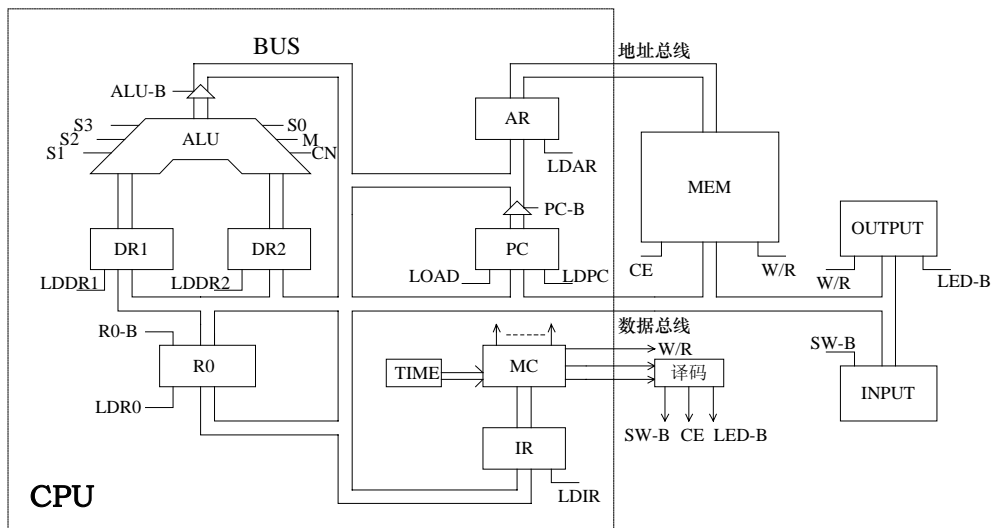


图 1.4-1 模型机逻辑框图

其中 ALU 为运算器、DR1、DR2 为工作暂存器、R0 为通用寄存器、AR 为地址寄存器、PC 为程序计数器、IR 为指令寄存器、TIME 为时序发生器、MEM 为程序存储器、INPUT 为输入设备、OUTPUT 为输出设备、MC 为微程序控制器。

2. 模型计算机的程序

本系统设计了四条指令，构成了此模型计算机的指令系统，即：

助记符	机器指令码	说 明
IN	0000 0000	INPUT \longrightarrow R0
ADD addr	0001 0000 $\times \times \times \times \times \times \times \times$	$R0+[addr] \longrightarrow R0$
OUT	0010 0000	$R0 \longrightarrow$ LED
JMP addr	0011 0000 $\times \times \times \times \times \times \times \times$	addr \longrightarrow PC

应用该指令系统可以编写一段反映计算机操作的指令序列，它们就构成了所谓的计算机程序，并将其以二进制存放在主存储器的连续的单元中。计算机通过连续运行该段程序，就可以解决各种复杂的计算或是控制问题。

3. 微程序 Microprogram

为实现以上计算机程序的操作，控制器对应于每一条机器指令都需要进行一系列的微操作来完成该机器指令的操作。一个微操作则对应一条微指令。如果控制器采用最普遍使用的微程序控制器，则一条机器指令的操作就需要一系列微指令来完成。它们构成计算机的微程序并且是以二进制数的形式存放在控制存储器的存储单元中。与以上机器指令对应的微操作内容如表 1.4-1 所示。

表 1.4-1 机器指令对应的微操作

机器指令助记符	微操作	说明
IN R0	① PC \rightarrow AR, PC+1 \rightarrow PC	预备取指
	② RAM \rightarrow BUS, BUS \rightarrow IR	取指
	③ INPUT \rightarrow R0	向 R0 中输入一个数
ADD X,R0	① PC \rightarrow AR, PC+1 \rightarrow PC	预备取指
	② RAM \rightarrow BUS, BUS \rightarrow IR	取指
	③ PC \rightarrow AR, PC+1 \rightarrow PC	预备取数据
	④ RAM \rightarrow BUS, BUS \rightarrow DR2	取数据送入 DR2
	⑤ R0 \rightarrow DR1	将 R0 中的数送入 DR1
	⑥ [DR1]+[DR2] \rightarrow R0	两数相加，结果送入 R0
OUT R0	① PC \rightarrow AR, PC+1 \rightarrow PC	预备取指
	② RAM \rightarrow BUS, BUS \rightarrow IR	取指
	③ R0 \rightarrow OUTPUT	将结果输出显示
JMP 00	① PC \rightarrow AR, PC+1 \rightarrow PC	预备取指
	② RAM \rightarrow BUS, BUS \rightarrow IR	取指
	③ PC \rightarrow AR, PC+1 \rightarrow PC	预备取数据
	④ RAM \rightarrow BUS, BUS \rightarrow PC	取数据送入 PC

四. 实验步骤

1. 构造一台模型计算机

首先，参照图 1.4-2，在教学实验系统中使用连接导线（排线）将模型计算机的各个部件连接在一起，构成一台完整的模型计算机。连接图中凡是标有小圆圈的连线都是需要连接导线的，而未标小圆圈的连线是系统已经连接好的。

连接完成后，请仔细检查，以保证连接的正确性。

2. 我们来编写一段简单程序操作的例子来说明计算机工作的过程。

这个程序要执行的功能是：

- 1) 由输入设备向 CPU 的通用寄存器 R0 中输入一个数。
- 2) 将输入的数值与程序中的一个立即数相加。
- 3) 将运算结果输出到输出设备上显示。
- 4) 跳转返回到执行第一条指令的状态和位置。

完成以上指令操作的程序内容如表 1.4-2 所示。

表 1.4-2 计算机操作程序

地址	指令码	指令助记符	说明
00	00000000	IN R0	INPUT→R0
01	00010000	ADD X,R0	R0+X→R0
02	X		X 为立即操作数，存放在 02 单元中
03	00100000	OUT R0	R0→OUTPUT
04	00110000	JMP 00	00→PC
05	00000000		

3. 模型机操作前的准备工作

使用通讯电缆将实验系统的串行接口与 PC 微机的串行接口相连接，并将实验系统的电源线接到电源插座中。然后启动 PC 微机，进入 Windows 系统，安装本设备提供的应用软件 CMPP。（安装方法及软件使用可见用户手册）。

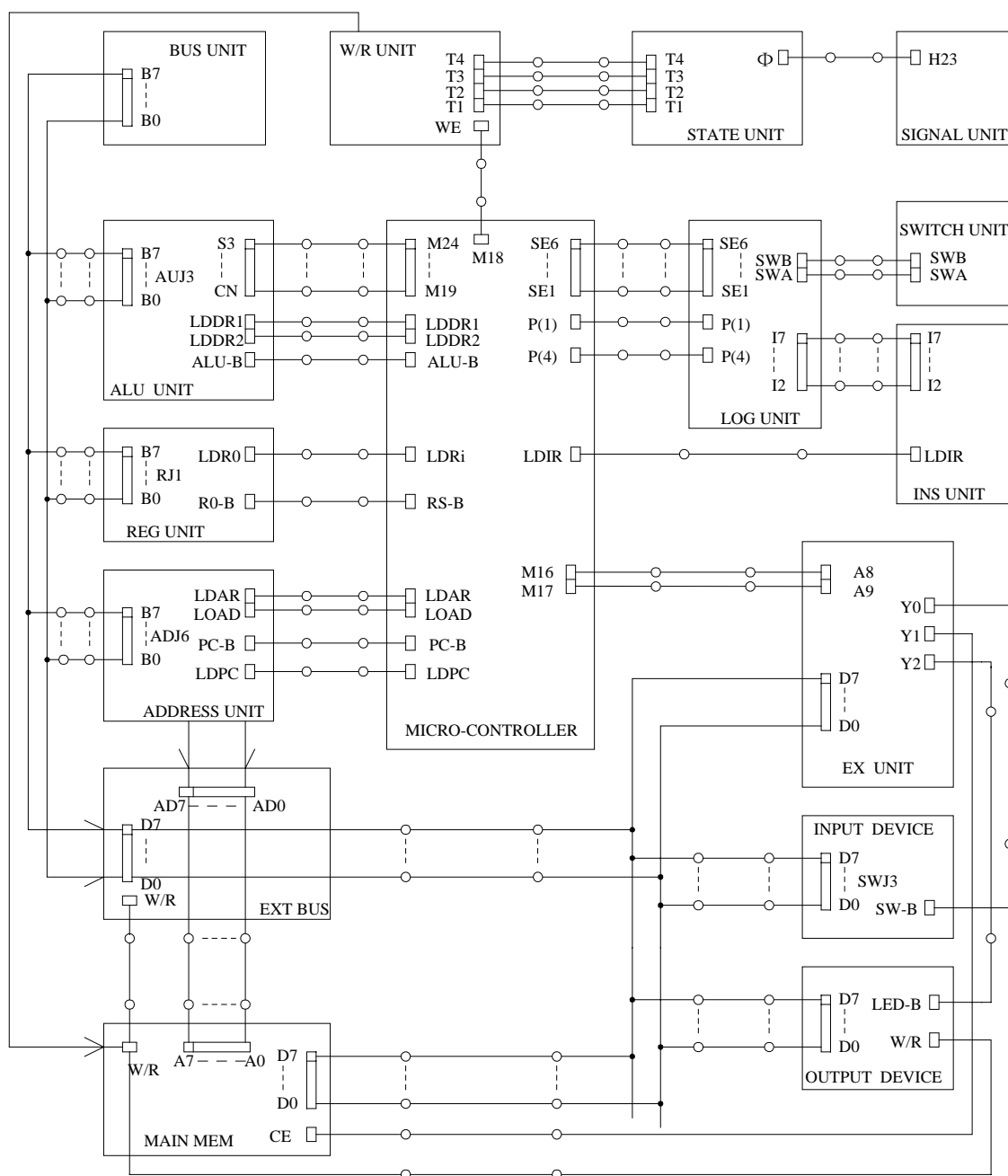


图 1.4-2 系统认识实验接线图

4. 模型计算机的运行操作

1) 打开实验系统的电源开关，点击图标 CMPP，运行软件。若联机正常后，将显示如图 1.4-3 所示界面。

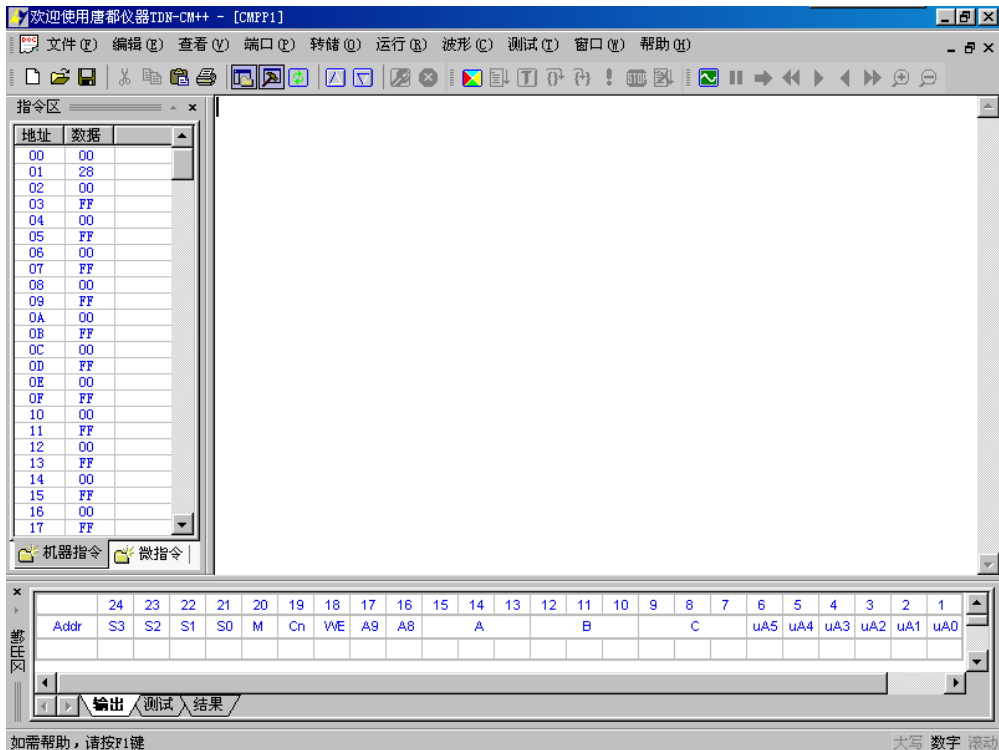


图 1.4-3 联机操作软件主界面

2) 未联机正常, 也可以进入软件界面, 但是所有的菜单里的功能全是灰色不可用 (除“文件”及“端口”菜单), 且指令区窗口中的数据也全以星号显示。本软件的默认串口为 1 号串口, 若通讯电缆连接到 2 号串口上, 可进入“【端口】”菜单, 选择 2 号串口, 然后进行“【端口】-【端口测试】”, 若还不正常, 请确保打开系统电源及检查通讯电缆的连接。具体排除故障见《使用手册》。

3) 进入“【转储】--【转载】”, 选择系统软件安装时在\CMPP\SAMPLE 目录下的一个例题 EX.TXT, 点击“打开”后即进行装载。此文件包含有上述设计的模型机要执行的机器指令程序及定义该机器指令系统的微程序。可从“【文件】—【打开】”来打开此文件, 可查看模型计算机操作的程序及其微程序。其内容为:

机器指令:

\$P0000

\$P0110

\$P0208

\$P0320

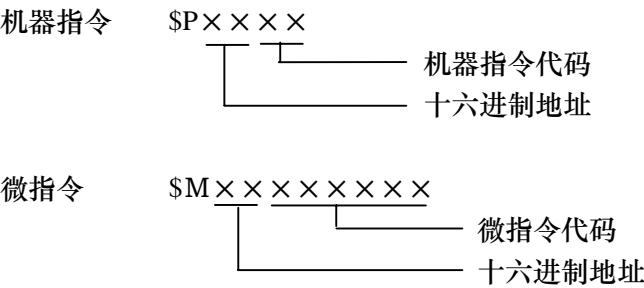
\$P0430

\$P0500

微指令:

\$M00018001
\$M0101ED82
\$M0200C048
\$M0300B004
\$M0401A205
\$M05959A01
\$M0600D181
\$M08001001
\$M0901ED83
\$M0A070201
\$M0B01ED86

机器指令及微指令的描述格式为：



4) 装载完成后，选择 “【运行】 - 【通路图】 - 【复杂模型机】” 可打开一个对应的数据通路图，如图 1.4-4 所示：

5) 在执行指令之前，要将实验系统右下角的 CLR 清零开关向上拨到 0 位再拨回 1 位，以将程序计数器和微地址寄存器清为零，使得程序可从零地址开始运行。

选择 “【运行】 — 【单步微指令】” 功能，每按动一次，系统运行一条微指令并在界面中显示动态数据流及微地址等的变化，仔细观察运行过程，则可了解并掌握计算机的工作过程。

6) 每按动一次 “【运行】 — 【单步机器指令】”，则单步执行一条机器指令。一条机器指令对应一段微程序，每执行一条微指令时，计算机同时显示数据流，执行完这条机器指令对应的所有微指令后则自动停止。此时可以继续单步执行下一条机器指令。

当模型计算机执行完一条指令后，PC 微机则根据指令的执行过程，在屏幕上显示出其数据流，图中各部件的有效控制信号则用高亮显示，并将下一条微指令代码显示在下方。这样就可以形象地看到一条指令的执行过程。

“【运行】 — 【单步微指令】” 的功能是单步执行一条微指令，同时显示其数据流。
“【运行】 — 【连续运行】”，则连续运行全部程序，同时连续显示整个数据流。当按动
“【运行】 — 【停止】” 时才会停止执行，但不是立即停止，只有当一条机器指令运行完后才会停止。

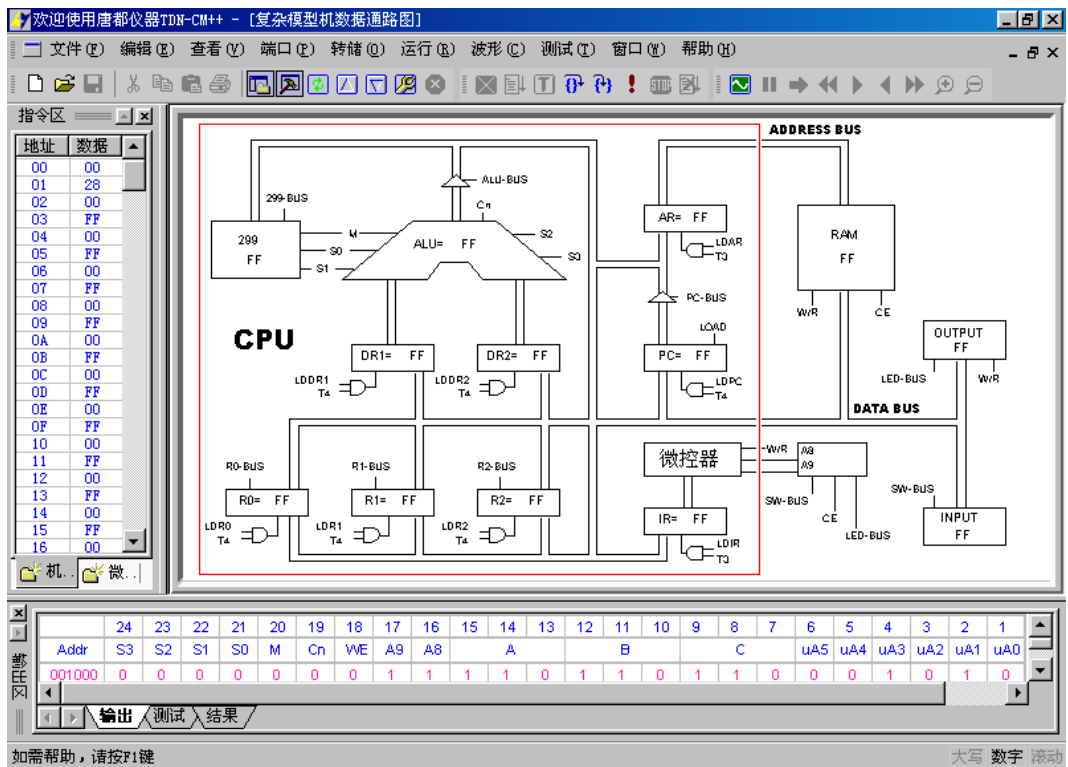


图 1.4-4

7) 单步执行机器指令，并对照表 1.4-2，观察对应一条机器指令的一系列微操作的运行过程。

思考问题

1) 单步执行微指令，观察应用软件的数据通路图中各部件的有效控制信号（高亮显示），思考这些控制信号的作用。并对照图 1-2，找到这些控制信号的来源，并思考它们是如何产生的，它们与微代码的关系。思考微程序控制器在整个模型计算机运行中的作用。

2) 单步执行指令 ADD X,R0，观察微操作 $[DR1]+[DR2] \rightarrow R0$ 执行时，运算器 ALU 的有效控制信号 S0-S3、M、CN，思考它们对运算器算术逻辑操作的作用。

第二章 运算器

运算器是计算机进行数据处理的核心部件。它主要由算术逻辑运算部件 ALU、累加器、暂存寄存器、通用寄存器堆、移位器、进位移位控制电路及其结果判断电路等组成。

运算方法的基本思想是：各种复杂的运算处理最终可分解为四则运算和基本的逻辑运算，而四则运算的核心是加法运算。通过补码运算可以化减为加，减法运算与移位运算配合可以实现乘除运算，阶码运算与尾数的运算组合可以实现浮点运算。

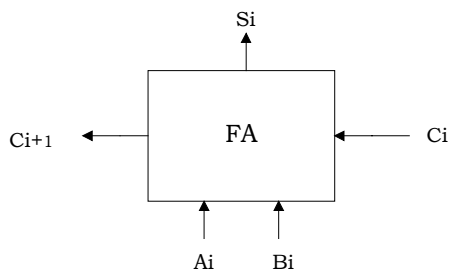
本章先讨论基本二进制加法器，用基本的门电路构成一位全加器，然后由多个全加器构成不同的并行加法器，紧接着安排一个并行加法器的设计实验；接下来讨论了乘法的原理及构成，安排了一个快速阵列乘法器的设计实验；讨论了如何构成一个多功能算术、逻辑运算部件及一般的运算器组织，安排了三个实验：算术逻辑运算实验、进位控制实验及移位运算实验。

2.1 基本的二进制加法器

二进制加法器是算术逻辑部件 ALU 的核心，将多个一位全加器连同进位信号传送逻辑，可构成一个 n 位并行加法器，再通过输入选择逻辑可扩展为具有多种运算、逻辑功能的 ALU。

2.1.1 全加器

现在普遍采用的一位全加器是由异或逻辑实现半加，再用两次半加实现一位全加。一位全加器 (FA) 的逻辑结构图见 2.1-1。一位二进制加法器单元有三个输入量：两个二进制数 A_i 、 B_i 和低位传来的进位信号 C_i 。两个输出量：本位和输出 S_i ，以及向高位的进位输出 C_{i+1} 。这种考虑了全部三个输入量的加法单元称为全加器。如果只考虑两个输入相加而不考虑进位量的，就称为半加器。表 2.1-1 列出了一位全加器进行加法运算的真值表。



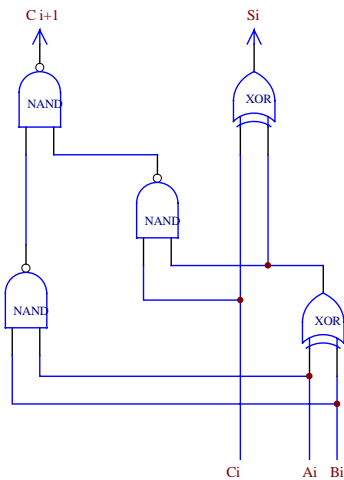


图 2.1-1 一位全加器（FA）逻辑电路图

表 2.1-1 一位全加器真值表

输入			输出	
Ai	Bi	Ci	Si	Ci+1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

根据上述真值表及逻辑结构图，可看出它们的逻辑方程：

$$Si = Ai \oplus Bi \oplus Ci$$

$$Ci+1 = AiBi + (Ai \oplus Bi) Ci$$

为方便讨论，将上述进位信号的基本逻辑写为通式：

$$Ci+1 = Gi + PiCi$$

式中 G_i 称为进位产生函数，其逻辑含义是：若本位的两个输入均为 1，必然产生进位，此分量与低位进位无关。 P_i 称为进位传递函数，其逻辑含义是：当 $P_i=1$ 时，若低位传来的进位必将通过本位传递向更高位。

2.1.2 并行加法器

并行加法器就是用 n 位全加器一步实现 n 位同时相加。但是这也存在一个问题：虽然操

作数的各位是同时提供的，但是进位的传递，即低位运算所产生的进位将会影响高位的运算结果，从而影响整个全加器结果的输出。所以，不同的操作数组合，进位的情况可能不同，加法器从获得稳定输入到产生稳定输出的运算时间也就可能不同。

1. 串行进位

串行进位是指将 n 个全加器按进位串起来，其中的进位是逐步形成的，每一级的进位都依赖于前一级的进位。其加法逻辑图见图 2.1-2。

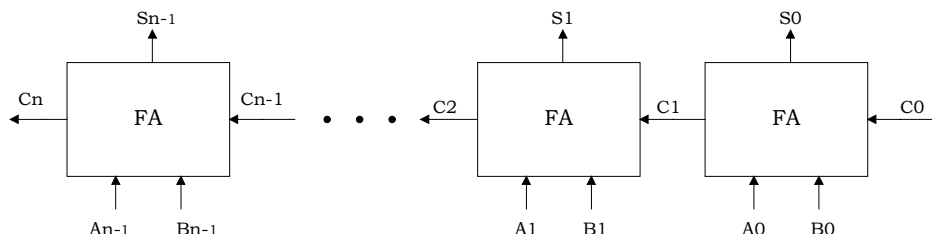


图 2.1-2 串行进位加法逻辑

其各进位信号的逻辑式如下：

$$C1 = G0 + P0C0 = A0B0 + (A0 \oplus B0) C0$$

$$C2 = G1 + P1C1 = A1B1 + (A1 \oplus B1) C1$$

...

$$Cn = Gn-1 + Pn-1Cn-1 = An-1Bn-1 + (An-1 \oplus Bn-1) Cn-1$$

采用串行进位的方式运算的时间较长，但是它用的元器件较少。

2. 并行进位

为了提高运算速度，就必须解决进位信号传递问题。可以让各级的进位同时形成，即并行地形成各级进位。逻辑关系如下：

$$C1 = G1 + P1C0$$

$$C2 = G2 + P2G1 + P2P1C0$$

$$C3 = G3 + P3G2 + P3P2G1 + P3P2P1C0$$

⋮

$$Cn = Gn + PnGn-1 + \dots + (Pn \dots P1) C0$$

可以看出，在上述的逻辑中，各位的进位信号是独自形成的，不依赖于前一级的进位。

从上面的并行进位逻辑可以看出，随着位数的增多，在高位的进位的逻辑输入量增多，这就会需要更多的器件。所以，对于较多位数的加法器，可以采用分级、分组的进位链结构。

3. 组内并行、组间串行进位

以 16 位加法器为例，将它们分为 4 组，每组 4 位，各组内采用并行进位而组间采用串行进位的方式。

4. 组内并行、组间并行进位

如还是 16 位加法器，每组 4 位，可将进位链分为两级。第一级小组内并行进位，第二级小组间并行进位。

2. 2 并行加法器设计实验

一. 实验目的

- 1. 掌握并行加法器的原理及其设计方法。
- 2. 熟悉 CPLD 应用设计及 EDA 软件的使用。

二. 实验设备

- 1. TDN-CM+或 TDN-CM++教学实验系统一套。
- 2. PC 微机一台。

三. 实验原理

本节实验使用大规模可编程逻辑器件 CPLD 来设计实现一个 4 位的并行进位加法器。传统的数字系统设计只能是通过设计电路板来实现系统功能，而采用可编程逻辑器件，则可以通过设计芯片来实现系统功能。从而有效地增强了设计的灵活性，提高了工作效率。并能够缩小系统体积，降低能耗，提高系统的性能和可靠性。

实验系统中采用的器件是 Lattice 公司的 ispLSI 1032 芯片，isp 是指芯片具有“在系统可编程功能”，这种功能可随时对系统进行逻辑重构和修改，而且只需要一条简单的编程电缆和一台 PC 计算机就可以完成器件的编程。

ispLSI1032 芯片的等效逻辑门为 6000 门，具有 128 个宏单元，192 个触发器和 64 个锁存器，其共有 84 个引脚，其中 64 个为 I/O 引脚。ispLSI1032 芯片的结构图如图 2.1-3 所示。

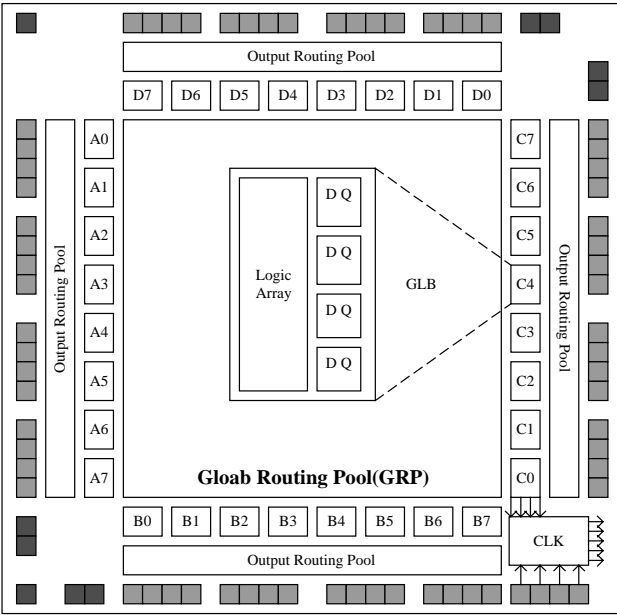


图 2. 2-1 1032 芯片结构图

对该器件的逻辑系统设计是通过使用**硬件描述语言**或**原理图输入**来实现的，硬件描述语言有 ABEL、VHDL 等多种语言，本节实验是使用原理图输入来进行编程的。

下面是一个用原理图输入设计一个四位并行加法器加法器的例子。该加法器采用并行进位，有两组四位加数 A3~A0、B3~B0 输入，四位本地和 F3~F0 输出，一个低位进位 C0 输入及一个本地进位 CY 输出。

系统采用 ispDesignEXPERT 软件来对可编程逻辑器件 ispLSI1032 进行编程设计实验。ispDesignEXPERT 可采用原理图或硬件描述语言或这两种方法的混合输入共三种方式来进行设计输入，并能对所设计的数字电子系统进行功能仿真和时序仿真。其编译器是此软件的核心，它能进行逻辑优化，并将逻辑映射到器件中去，自动完成布局与布线并生成编程所需要的熔丝图文件。该软件支持所有 Lattice 公司的 ispLSI 器件。

四．实验步骤

1. 安装 EDA 软件

打开计算机电源，进入 Windows 系统，安装上述 ispDesignEXPERT 软件。安装完成后，桌面和开始菜单中则建有 ispDesignEXPERT 软件图标。

2. 建立新项目

用鼠标双击该软件图标，则出现其操作界面 ispDesignEXPERT Project Navigator。

在界面左上角 File 菜单中点 New Project ...或点击左上角“新建”图标，则出现界面 Create New Project，在其中 Project 栏中输入 ALU1.syn，在 Project type 栏中选 Schematic/ABEL，并点保存，则在 Sources in Project 栏中建立了新的项目。

双击第一行 Untitled 对该项目命名，在名称栏中填入 ALU_EX 并点 OK。

双击第二行选择器件。根据实验系统中所使用的器件型号，例如对 ispLSI1032-70LJ 这一器件，在 Family 栏中选 ispLSI 1K Device，在 Device 栏中选 ispLSI1032，在 Speed Grade 栏中选 70，在 Package 栏中选 84PLCC，点 OK，再用 Yes 确定，则选定器件为 ispLSI1032-70LJ84。

3. 输入编辑原理图

单击界面左下角的按钮 New...，则出现界面 New Source: (Schematic/ABEL)，在其中选择 Schematic，并点击按钮 OK，则出现原理图编辑界面 Schematic edit，输入模块名称 ALU 并点 OK，则就可在原理图编辑界面中输入电路原理图了。输入逻辑图完成后，将其存盘并退出编辑界面。

输入设计加法器原理图如图 2.2-2。

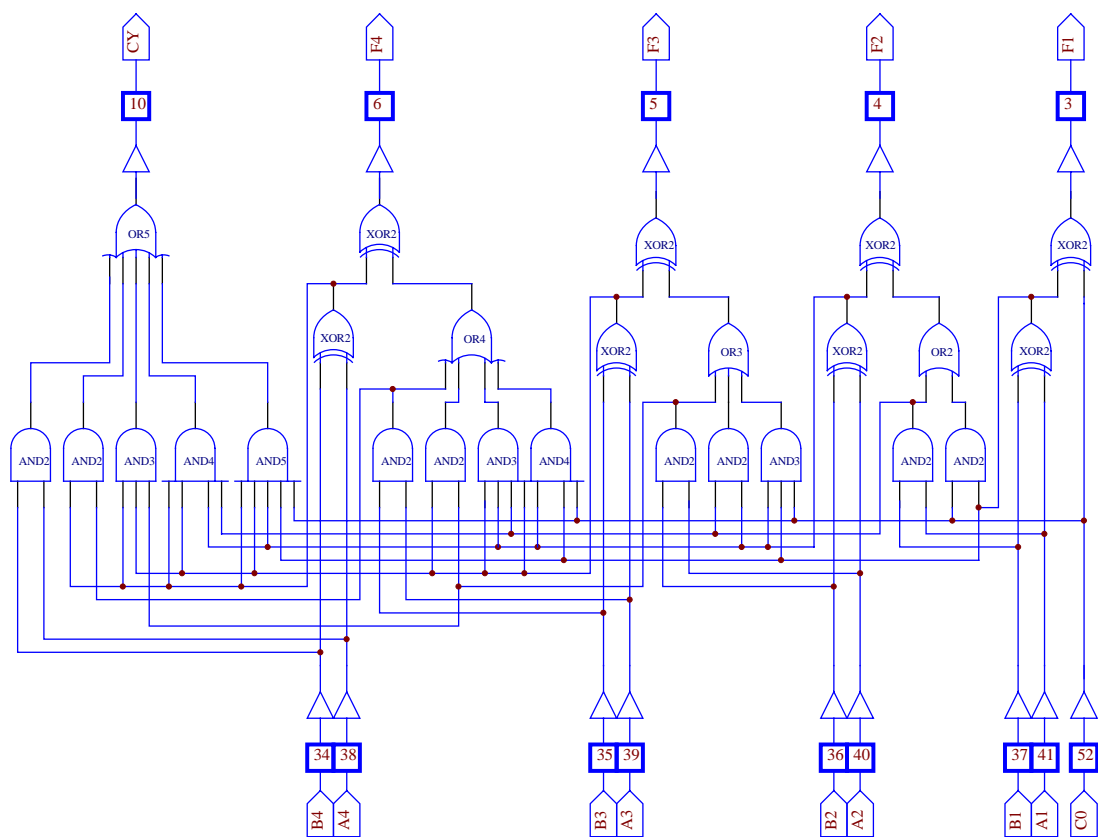


图 2.2-2 并行进位加法器逻辑原理图

4. 对源程序进行编译

在左方 Sources in Project 栏中选中第二行 ispLSI1032-70LJ84, 在右方 Processes for Current Source 栏中双击第七行 JEDEC File, 则开始编译。如果编译正确, 则生成可下载的文件 JEDEC File, 即使出现警告提示, 也表明成功生成了可下载文件。如果提示错误, 则需修改程序, 然后重新编译。

5. 连接下载电缆

在打开 PC 计算机和实验系统的电源之前, 将下载电缆的一端与 PC 计算机的打印机口相连接, 另一端与实验系统中的 ispLSI1032 器件编程接口相连。

6. 将 JEDEC 文件下载到 ispLSI1032

首先打开实验系统的电源。

在以上界面菜单 Tools 中点击 ispDCD, 则进入文件下载界面。

在下载界面中, 点击菜单 Configuration 中的 Scan Board 项或 SCAN 图标, 则出现扫描界面, 其下方的信息显示已检测到 ISP 芯片电路。然后点按钮 BROWSE, 在其中选择要下载的文件 ALU.JED。并在 Command 菜单中, 点 Run Operation in Sequential Mode 项或 Run Operation

图标, 则进入文件下载过程。在进行下载时, 实验系统下载电路的指示灯闪烁。下载完成后, 界面下方显示下载过程是否正确的有关信息。

7. 连接实验电路

按图 2.2-3 连接实验电路, 其中 ispLSI1032 的输入输出引脚已在程序中定义。

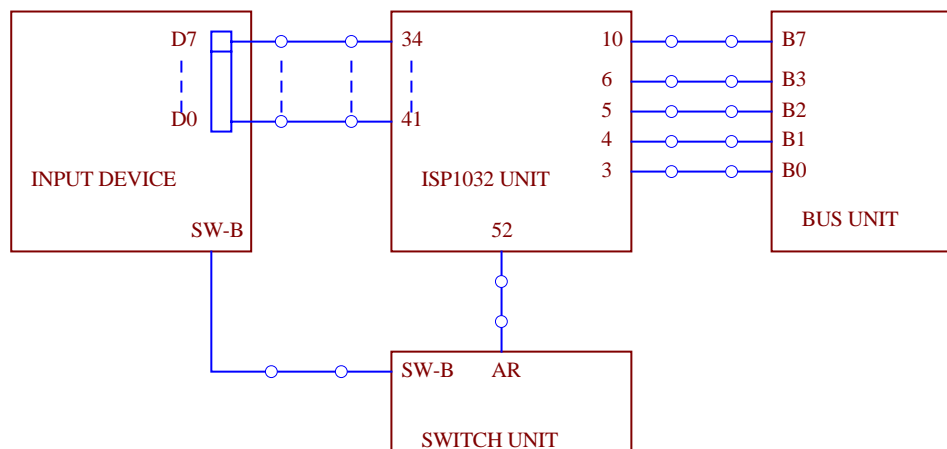


图 2.2-3 并行加法器实验接线图

8. 验证所设计器件的逻辑功能

本实验所设计的是一个 4 位并行进位加法器, 实验中用 INPUTDEVICE 单元的高 4 位为 B3~B0, 低 4 位为 A3~A0, 以总线单元的低 4 位 B3~B0 发光二极管来显示运算结果。B7 位来显示进位输出。而低位进位输入由一个开关 AR 来给出。使 SWITCH UNIT 单元中的开关 SW-B=0 (打开数据输出三态门), 拨动 INPUT DEVICE 单元的输入开关置 A 和 B 的值, 然后从总线单元的显示灯来观察运算结果。

9. 以上所设计的并行加法器在应用 ispDesignEXPERT 软件时是以原理图输入形式来编程的, 目的是为了让学生能更好的理解并行进位加法器的实现原理。为了学生学习以硬件描述语言来进行编程, 描写器件功能, 下面用 ABEL 语言编程来实现上述加法器, 步骤如下:

① 建立新工程

打开 ispDesignEXPERT 软件, 建立一个新的目录来创建一个新的工程文件。在界面左上角 File 菜单中点 New Project ... 或点击左上角“新建”图标, 则出现界面 Create New Project, 在其中 Project 栏中输入 ALU2.syn, 在 Project type 栏中选 Schematic/ABEL, 并点保存, 则在 Sources in Project 栏中建立了新的项目。器件型号还是选 ispLSI1032-70LJ84。

② 编辑源程序

单击界面左下角的按钮 New..., 则出现界面 New Source: (Schematic/ABEL), 在其中选择 ABEL-HDL Module, 点击按钮 OK, 则出现 new ABLE-HDL source 窗口, 输入模块名称和文件名并点 OK, 就可在出现的源程序编辑界面中输入源程序了。输入完成后, 将其存盘并

退出编辑界面。

上述并行加法器设计用 ABEL 语言来描述程序如下：

```
MODULE alu
TITLE '4 bit adder'

"Inputs
A4,A3,A2,A1          PIN 38,39,40,41;
B4,B3,B2,B1          PIN 34,35,36,37;
C0                    PIN 52;

"Outputs
F4,F3,F2,F1          PIN 6,5,4,3;
CY                    PIN 10;

"VAR
A=[0,A4,A3,A2,A1];
B=[0,B4,B3,B2,B1];
C=[0, 0, 0, 0,C0];
F=[CY,F4,F3,F2,F1];
"

EQUATIONS
"
F = A + B + C;
"

END
```

③ 对源程序进行编译

④ 将生成的 JED 文件下载至 1032 芯片中。

⑤ 实验连线及实验操作步骤同上。

2.3 定点乘法运算

乘法运算是计算机中一个重要的基本运算。定点数的乘法可分为原码和补码两种运算。

2.3.1 原码一位乘法

原码一位乘法的算法是：

① 两数的绝对值相乘，符号位按“同号为正，异号为负”处理。

② 部分积初始值为 0，以乘数的最低位为乘法的判别位，若为 1，则在部分积上累加被乘数，然后连同乘数一起右移一位；若为 0，则在部分积上累加 0，然后连同乘数一起右移一位。

③ 重复②步直到运算 n 次为止（ n 为乘数部分的长度）。

例如，现有 $x = 0.1101$ ， $y = 0.1011$ ，求 $z = x \cdot y$ 。

部分积	乘数	说明
00. 0000	.101 <u>1</u>	$z = 0$
+00. 1101		$= 1, z = z + x$
00. 1101		
00. 0110	1.10 <u>1</u>	右移
+00. 1101		$= 1, z = z + x$
01. 0011		
00. 1001	11.1 <u>0</u>	右移
+00. 0000		$= 0, z = z + 0$
00. 1001		
00. 0100	111. <u>1</u>	右移
+00. 1101		$= 1, z = z + x$
01. 0001		
00. 1000	1111.	右移 得 z

加符号位 $x \cdot y = 0.10001111$ 。

2.3.2 补码一位乘法

补码一位乘法又称加减交替乘法，也称之为比较法。设参加运算的操作数为：

被乘数 $x = x_0x_1x_2 \cdots x_n$

乘数 $y = y_0y_1y_2 \cdots y_n$

其中 x 、 y 均为补码表示， x_0 、 y_0 为各数的符号位。

补码一位乘法的算法是：

- ① 参加运算的两数均以补码表示，符号位参加运算且部分积与被乘数采用双符号位。
- ② 乘数末位增设一个附加位 y_{n+1} ，其初始值为 0。
- ③ 以 $y_n y_{n+1}$ 作为乘法判断位：
 - 若 $y_n y_{n+1} = 00$ ，则部分积连同乘数右移一位，初始部分积为 0。
 - 若 $y_n y_{n+1} = 01$ ，则在前次部分积上加 $[x]_{补}$ ，然后连同乘数右移一位。
 - 若 $y_n y_{n+1} = 10$ ，则在前次部分积上加 $[-x]_{补}$ ，然后连同乘数右移一位。
 - 若 $y_n y_{n+1} = 11$ ，则部分积连同乘数右移一位。
- ④ 重复第③步，共做 $n + 1$ 次，注意最后一次不移位，移位时须按照补码移位规则进行。

2.3.3 阵列乘法

硬件乘法器常规的设计是采用“串行移位”和“并行加法”相结合的方法，这种方发法并不需要很多的器件，然而“加法-移位”的方法毕竟太慢。随着大规模集成电路的发展，采用高速的阵列乘法器，无论从计算机的计算速度，还是从提高计算效率，都是十分必要的。阵列乘法器分带符号和不带符号的阵列乘法器，本节只讨论不带符号阵列乘法。

高速组合阵列乘法器是采用标准加法单元来构成乘法器的,即利用多个一位加法器实现乘法运算。

对于一个 4 位二进制数相乘，有如下算式：

					a3	a2	a1	a0
					b3	b2	b1	b0
<hr/>								
					a3b0	a2b0	a1b0	a0b0
				a3b1	a2b1	a1b1	a0b1	
		a3b2	a2b2	a1b2	a0b2			
+	a3b3	a2b3	a1b3	a0b3				
<hr/>								
	p7	p6	p5	p4	p3	p2	p1	p0

这个 4×4 阵列乘法器的原理图见图 2.3-1。

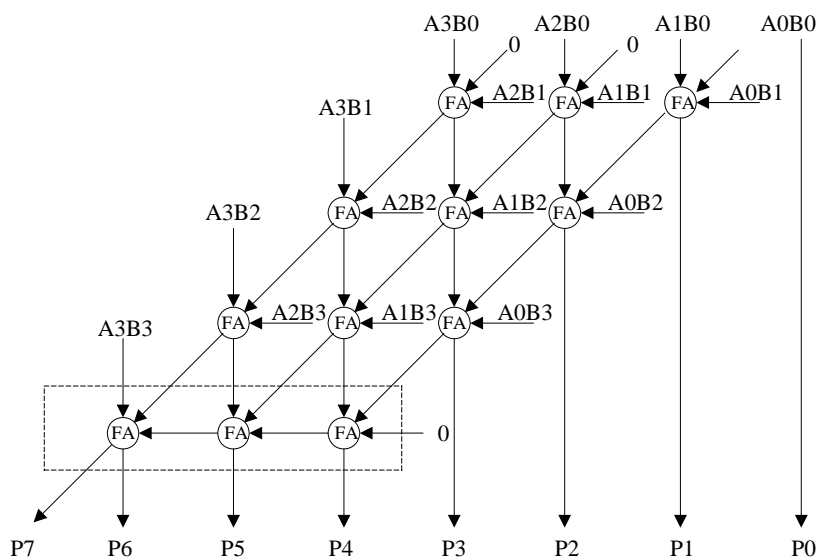


图 2.3-1 4×4 阵列乘法器原理图

其中的 FA 就是前边讲过的全加器。FA 的斜线方向为进位输出，竖线方向为和输出。图中阵列的最后一行构成了一个串行进位加法器。由于 FA 一级是无需考虑进位的，它的进位被暂时保留下来不往前传递，因此同一级中任意一位 FA 加法器的进位输出与和输出几乎是同时形成的，所以送往最后一行串行加法器的输入延迟仅与 FA 的级数（行数）有关，即与乘数位数有关。

2.4 阵列乘法器设计实验

一. 实验目的

1. 掌握乘法器的原理及其设计方法。
2. 熟练应用 CPLD 设计及 EDA 操作软件。

二. 实验设备

1. TDN-CM+或 TDN-CM++教学实验系统一套。
2. PC 微机一台。

三. 实验原理

本实验用 CPLD 来设计一个 4×4 位乘法器，相对于画电路图输入，用 ABEL 语言描述是比较方便的。其算式如下（其中括号中的数字表示在 ABEL 源程序描述中的功能块调用编号）：

$$\begin{array}{r}
 \begin{array}{cccc}
 & a3 & a2 & a1 & a0 \\
 \times & b3 & b2 & b1 & b0 \\
 \hline
 & & a3b0(10) & a2b0(6) & a1b0(3) & a0b0(1) \\
 & & a3b1(13) & a2b1(9) & a1b1(5) & a0b1(2) \\
 & & a3b2(15) & a2b2(12) & a1b2(8) & a0b2(4) \\
 + & a3b3(16) & a2b3(14) & a1b3(11) & a0b3(7) & \\
 \hline
 p7 & p6 & p5 & p4 & p3 & p2 & p1 & p0
 \end{array}
 \end{array}$$

四. 实验步骤

1. 根据上述乘法的逻辑原理用 ABEL 语言编写功能描述程序。其在 1032 芯片中对应的管脚如图 2.4-1：

P7	P6	P5	P4	P3	P2	P1	P0
45	46	47	48	49	50	51	52
CPLD1032							
34	35	36	37	38	39	40	41
A3	A2	A1	A0	B3	B2	B1	B0

图 2.4-1 实验中乘法器对应的 1032 的引脚

2. 编辑、编译和下载

使用 ispDesignEXPERT 软件编辑源程序并进行编译，然后打开实验系统电源，将生成的 JEDEC 文件下载到 ispLSI1032 中去。

3. 连接实验电路

按图 2.4-2 连接实验电路。

4. 给定操作数，观察乘法器输出

将 SWITCH UNIT 单元中的 SW-B、AR 开关置为低电平状态。在 INPUT DEVICE 单元中的 8 个开关的高 4 位为乘数 A，低四位为被乘数 B，而相乘的结果将在 OUTPUT DEVICE 单元中的数码管中以十六进制形式显示。给 A 和 B 置不同的数，观察相乘的结果。

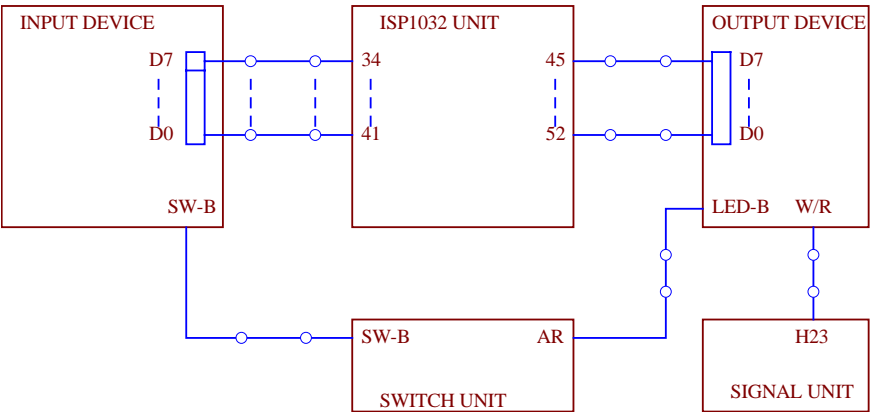


图 2.4 -2 阵列乘法器接线图

2. 5 多功能算术逻辑运算单元 (ALU)

计算机的一个最主要的功能就是处理各种算术和逻辑运算,这个功能要由 CPU 中的运算器这一部件来完成。运算器也称作作为算术逻辑部件 ALU,它可将若干位全加器、并行进位链、输入选择门三部分集成于一块芯片之上。

因为，任何复杂的运算都可以化为加减乘除算术运算和基本逻辑运算，而各种算术运算又可以变为加法运算，如运用补码就可以变减法为加法，用加减运算与移位运算相结合就能实现乘除法运算。所以要实现一个运算器，其基础是构造其中的加法器。

设计构造一个运算器的基本思路是：

1. 首先用基本门电路构成 1 位全加器。
2. 然后用进位传递逻辑将其构成 N 位并行加法器。
3. 再利用多路选择逻辑实现多种输入输出组合选择，使加法器扩展成为多功能的算术逻辑运算部件 ALU。

- 4. 另外利用多路选择逻辑还可实现移位功能。
- 5. 使用加法器与移位器的组合则可构成乘法器和除法器。
- 6. 使用两个（定点）运算器部件的组合则可构成一个浮点运算器。

2. 5. 1 一位 ALU 逻辑

图 2.5-1 所示的是用一个全加器构成的 1 位 ALU 单元。其中的控制信号 S3~S0 与输出 Xi 和 Yi 的关系如表 2.5-1 所示。另控制信号 M= 0，则接收低位来的进位信号 Ci-1 而执行算术运算；M= 1，则不接收进位信号 Ci-1 而执行逻辑运算。选择不同的控制信号，则可获得不同的输出 Fi。

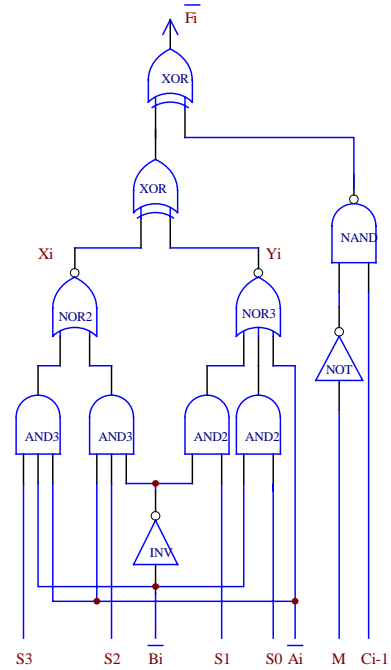


图 2. 5-1 1 位 ALU 单元

表 2. 5-1 1 位 ALU 单元逻辑关系表

S3	S2	Xi	S1	S0	Yi
0	0	1	0	0	Ai
0	1	$Ai + \overline{Bi}$	0	1	$AiBi$
1	0	$Ai + Bi$	1	0	$Ai\overline{Bi}$
1	1	Ai	1	1	0

2. 5. 2 多功能算术逻辑运算单元 ALU

集成逻辑芯片 74LS181 是一个 4 位 ALU 单元，其引脚如图 2.5-2 所示，逻辑构成如图 2.5-3 所示，逻辑功能如表 2.5-2 所示。

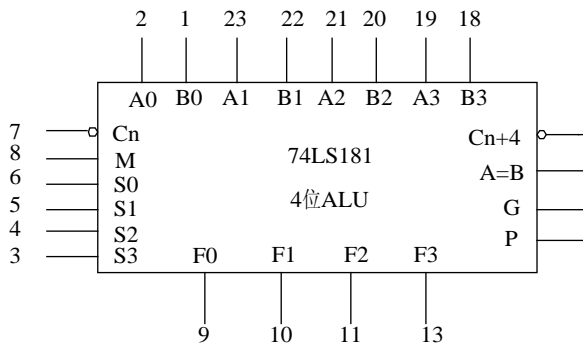


图 2.5-2 74LS181 引脚图

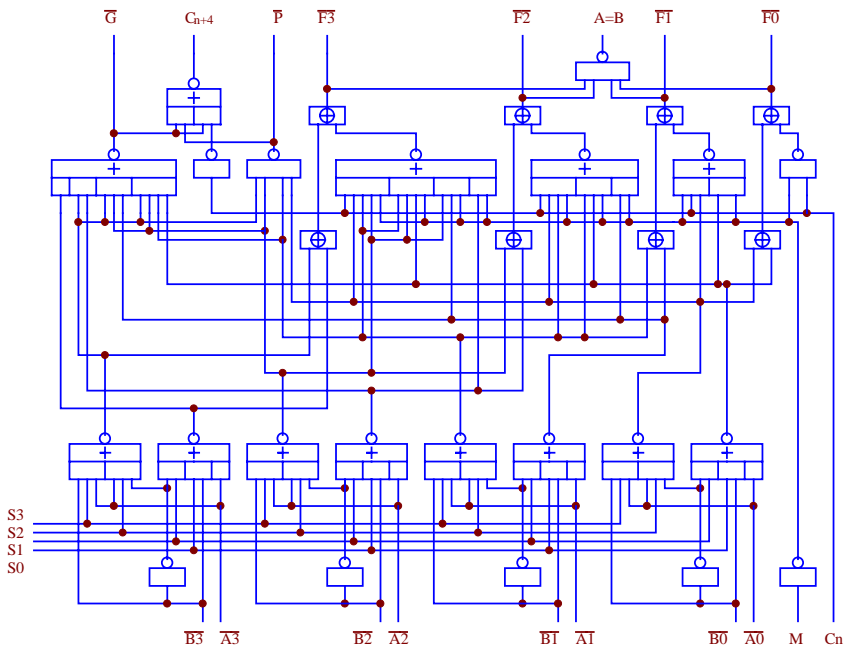


图 2.5-3 74LS181 逻辑图

74LS181 芯片的设计就是由表 2.5-1 中的 X_i 、 Y_i 与 S_0 - S_3 的逻辑关系，而导出可能实现的算术与逻辑运算各十六种运算功能。其中包含了一些基本运算功能，如：A 加 B、A 减 B、A 加 1、A 减 1、逻辑与 AB、逻辑或 A+B、求反 A 或 B、异或 A⊗B、传送 A、传送 B、输出 0、输出 1 等。

如果采取组间串行进位结构，将几片 74LS181 芯片各片的进位输出 C_{n+4} 与高位芯片的进位输入端 C_n 相连，就可构成一个多位字长的运算器 ALU 部件。

表 2.5-2 74LS181 的逻辑功能表

输入为 A 和 B，输出为 F，为正逻辑。

S3 S2 S1 S0	M=0(算术运算)		M=1(逻辑运算)
	Cn=1(无进位)	Cn=0(有进位)	
0 0 0 0	$F=A$	$F=A$ 加 1	$F=\overline{A}$
0 0 0 1	$F=A+B$	$F=(A+B)$ 加 1	$F=\overline{A+B}$
0 0 1 0	$F=A+\overline{B}$	$F=(A+\overline{B})$ 加 1	$F=\overline{A}B$
0 0 1 1	$F=0$ 减 1	$F=0$	$F=0$
0 1 0 0	$F=A$ 加 $\overline{A}B$	$F=A$ 加 $\overline{A}B$ 加 1	$F=\overline{A}B$
0 1 0 1	$F=\overline{A}B$ 加 $(A+B)$	$F=\overline{A}B$ 加 $(A+B)$ 加 1	$F=\overline{B}$
0 1 1 0	$F=A$ 减 B 减 1	$F=A$ 减 B	$F=A \oplus B$
0 1 1 1	$F=\overline{A}B$ 减 1	$F=\overline{A}B$	$F=\overline{A}B$
1 0 0 0	$F=A$ 加 AB	$F=A$ 加 AB 加 1	$F=\overline{A}+B$
1 0 0 1	$F=A$ 加 B	$F=A$ 加 B 加 1	$F=\overline{A} \oplus \overline{B}$
1 0 1 0	$F=AB$ 加 $(A+\overline{B})$	$F=AB$ 加 $(A+\overline{B})$ 加 1	$F=B$
1 0 1 1	$F=AB$ 减 1	$F=AB$	$F=AB$
1 1 0 0	$F=A$ 加 A	$F=A$ 加 A 加 1	$F=1$
1 1 0 1	$F=A$ 加 $(A+B)$	$F=A$ 加 $(A+B)$ 加 1	$F=A+\overline{B}$
1 1 1 0	$F=A$ 加 $(A+\overline{B})$	$F=A$ 加 $(A+\overline{B})$ 加 1	$F=A+B$
1 1 1 1	$F=A$ 减 1	$F=A$	$F=A$

2.6 算术逻辑运算实验

一. 实验目的

1. 了解运算器的组成结构。
2. 掌握运算器的工作原理。
3. 学习运算器的设计方法。
4. 掌握简单运算器的数据传送通路。
5. 验证运算功能发生器 74LS181 的组合功能。

二. 实验设备

TDN-CM+或 TDN-CM++教学实验系统一套。

三. 实验原理

实验中所用的运算器数据通路图如图 2.6-1。图中所示的是由两片 74LS181 芯片以并/串形式构成的 8 位字长的运算器。右方为低 4 位运算芯片，左方为高 4 位运算芯片。低位芯片

的进位输出端 C_{n+4} 与高位芯片的进位输入端 C_n 相连，使低 4 位运算产生的进位送进高 4 位运算中。低位芯片的进位输入端 C_n 可与外来进位相连，高位芯片的进位输出引至外部。两个芯片的控制端 $S0 \sim S3$ 和 M 各自相连，其控制电平按表 2.6-1。

为进行双操作数运算，运算器的两个数据输入端分别由两个数据暂存器 DR1、DR2（用锁存器 74LS273 实现）来锁存数据。要将内总线上的数据锁存到 DR1 或 DR2 中，则锁存器 74LS273 的控制端 LDDR1 或 LDDR2 须为高电平。当 T4 脉冲来到的时候，总线上的数据就被锁存进 DR1 或 DR2 中了。

为控制运算器向内总线上输出运算结果，在其输出端连接了一个三态门（用 74LS245 实现）。若要将运算结果输出到总线上，则要将三态门 74LS245 的控制端 ALU-B 置低电平。否则输出高阻态。

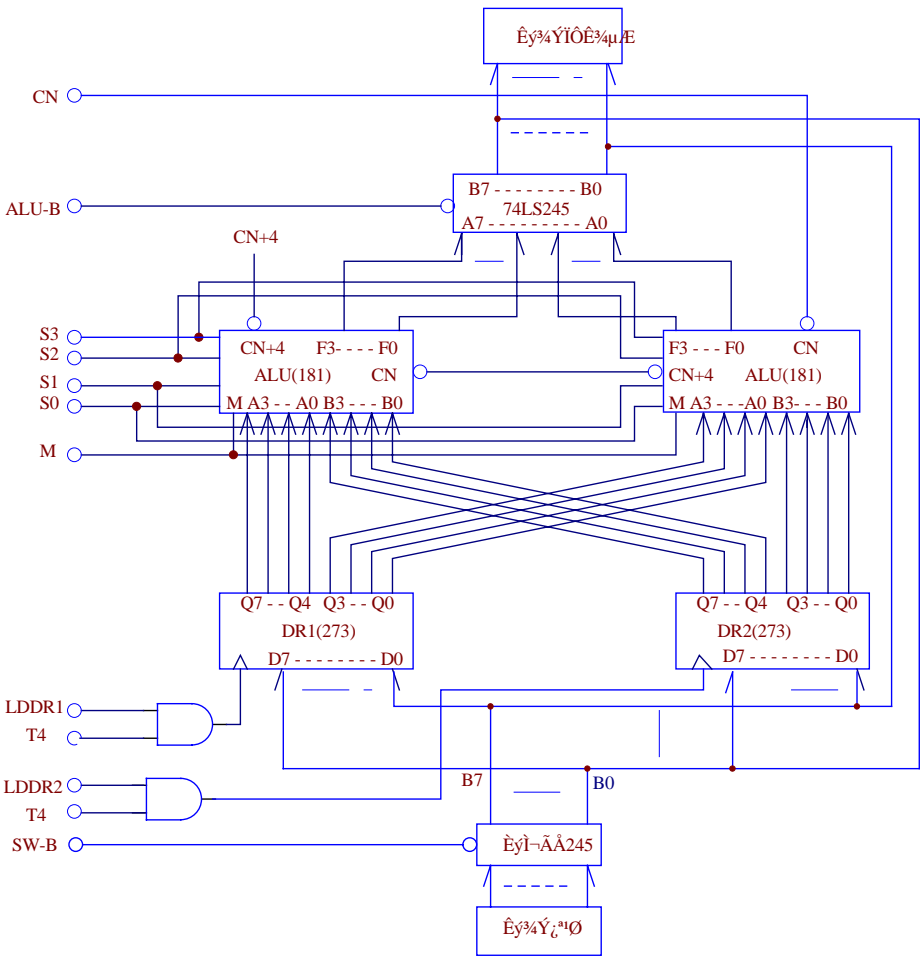


图 2.6-1 运算器通路图

数据输入单元(实验板上印有 INPUT DEVICE)用以给出参与运算的数据。其中，输入开关经过一个三态门（74LS245）和内总线相连，该三态门的控制信号为 SW-B，取低电平时，开关上的数据则通过三态门而送入内总线中。

总线数据显示灯（在 BUS UNIT 单元中）已与内总线相连，用来显示内总线上的数据。控制信号中除 T4 为脉冲信号，其它均为电平信号。

由于实验电路中的时序信号均已连至“W/R UNIT”单元中的相应时序信号引出端，因此，需要将“W/R UNIT”单元中的 T4 接至“STATE UNIT”单元中的微动开关 KK2 的输出端。在进行实验时，按动微动开关，即可获得实验所需的单脉冲。

S3、S2、S1、S0、Cn、M、LDDR1、LDDR2、ALU-B、SW-B 各电平控制信号则使用“SWITCH UNIT”单元中的二进制数据开关来模拟，其中 Cn、ALU-B、SW-B 为低电平有效，LDDR1、LDDR2 为高电平有效。

对于单总线数据通路，作实验时就要分时控制总线，即当向 DR1、DR2 工作暂存器打入数据时，数据开关三态门打开，这时应保证运算器输出三态门关闭；同样，当运算器输出结果至总线时也应保证数据输入三态门是在关闭状态。

四．实验步骤

1. 按图 2.6-2 连接实验电路并检查无误。图中将用户需要连接的信号线用小圆圈标明（其它实验相同，不再说明）。

2. 开电源开关。

3. 用输入开关向暂存器 DR1 置数。

①拨动输入开关形成二进制数 01100101（或其它数值）。（数据显示灯亮为 0，灭为 1）。

②使 SWITCH UNIT 单元中的开关 SW-B=0（打开数据输入三态门）、ALU-B=1（关闭 ALU 输出三态门）、LDDR1=1、LDDR2=0。

③按动微动开关 KK2，则将二进制数 01100101 置入 DR1 中。

4. 用输入开关向暂存器 DR2 置数。

①拨动输入开关形成二进制数 10100111（或其它数值）。

②SW-B=0、ALU-B=1 保持不变，改变 LDDR1、LDDR2，使 LDDR1=0、LDDR2=1。

③按动微动开关 KK2，则将二进制数 10100111 置入 DR2 中。

5. 检验 DR1 和 DR2 中存的数是否正确。

①关闭数据输入三态门（SW-B=1），打开 ALU 输出三态门（ALU-B=0），并使 LDDR1=0、LDDR2=0，关闭寄存器。

②置 S3、S2、S1、S0、M 为 11111，总线显示灯则显示 DR1 中的数。

③置 S3、S2、S1、S0、M 为 10101，总线显示灯则显示 DR2 中的数。

6. 改变运算器的功能设置，观察运算器的输出。

①SW-B=1、ALU-B=0 保持不变。

②按表 2-2 置 S3、S2、S1、S0、M、Cn 的数值，并观察总线显示灯显示的结果。

例如：置 S3、S2、S1、S0、M、Cn 为 100101，运算器作加法运算。

置 S3、S2、S1、S0、M、Cn 为 011000，运算器作减法运算。

7. 验证 74LS181 的算术运算和逻辑运算功能（采用正逻辑）

在给定 DR1=65、DR2=A7 的情况下，改变运算器的功能设置，观察运算器的输出，填入下表中，并和理论分析进行比较、验证。

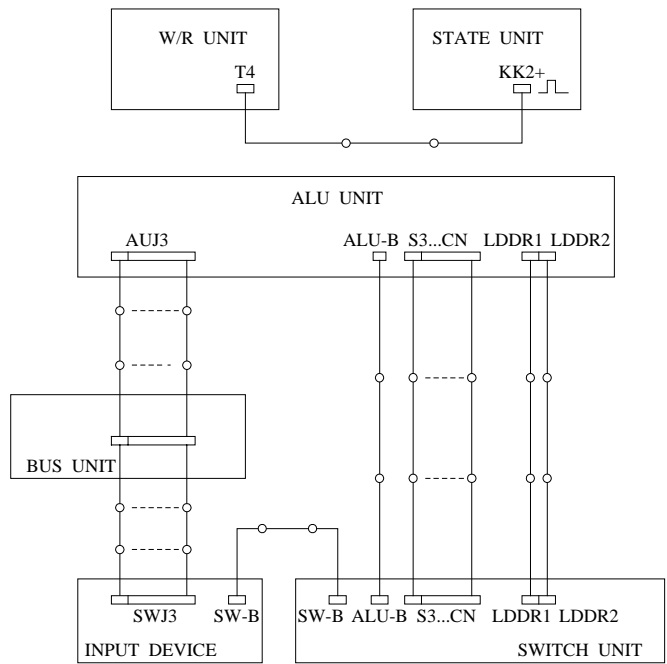


图 2.6-2 算术逻辑实验接线图

表 2.6-1

DR1	DR2	S3 S2 S1 S0	M=0 (算术运算)		M=1 (逻辑运算)
			Cn=1 无进位	Cn=0 有进位	
65	A7	0 0 0 0	F= (65)	F= (66)	F= (9A)
65	A7	0 0 0 1	F= (E7)	F= (E8)	F= (18)
65	A7	0 0 1 0	F= (7D)	F= (7E)	F= (82)
		0 0 1 1	F= ()	F= ()	F= ()
		0 1 0 0	F= ()	F= ()	F= ()
		0 1 0 1	F= ()	F= ()	F= ()
		0 1 1 0	F= ()	F= ()	F= ()
		0 1 1 1	F= ()	F= ()	F= ()
		1 0 0 0	F= ()	F= ()	F= ()
		1 0 0 1	F= ()	F= ()	F= ()
		1 0 1 0	F= ()	F= ()	F= ()
		1 0 1 1	F= ()	F= ()	F= ()
		1 1 0 0	F= ()	F= ()	F= ()
		1 1 0 1	F= ()	F= ()	F= ()
		1 1 1 0	F= ()	F= ()	F= ()
		1 1 1 1	F= ()	F= ()	F= ()

2.7 进位控制实验

一. 实验目的

1. 了解带进位控制的运算器的组成结构。
2. 验证带进位控制的运算器的功能。

二. 实验设备

TDN-CM+或 TDN-CM++教学实验系统一套。

三. 实验原理

图 2.7-1 所示为进位锁存及其显示电路。运算器最高位进位输出 C_{n+4} 连接到一个锁存器（用 74LS74 实现）的输入端 D，锁存器控制端的控制信号 AR 必须置为低电平，当 T4 脉冲来到时，进位结果就被锁存到进位锁存器中了，发光二极管这时显示为“灭”。同时也将本次的进位输出结果带进了下次的运算中，作为下次运算的进位输入。

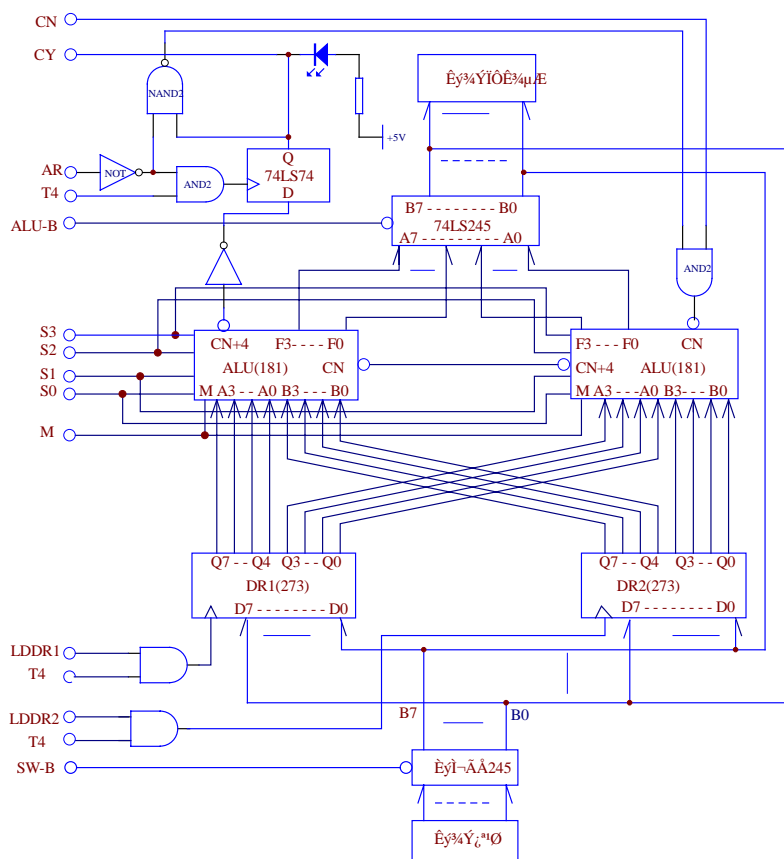


图 2.7-1 带进位运算器通路图

四. 实验步骤

1. 按图 2.7-3 连接实验电路并检查无误。
2. 打开电源开关。
3. 用输入开关向暂存器 DR1 和 DR2 置数,方法同前。
4. 关闭数据输入三态门(SW-B=1),打开 ALU 输出三态门(ALU-B=0),并使 LDDR1=0、LDDR2=0, 关闭寄存器打入控制门。

5. 对进位标志清零。

实验板上“SWITCH UNIT”单元中的 CLR 开关为标志 CY、ZI 的清零开关,它为零状态时是清零状态,所以将此开关做 1→0→1 操作,即可使标志位清零。

注意:进位标志指示灯 CY 亮时表示进位标志为“0”,无进位;标志指示灯 CY 灭时表示进位为“1”,有进位。

6. 验证带进位运算及进位锁存功能。

使 $C_n=1$, $AR=0$,进行带进位算术运算。

例如做加法运算,使 $ALU-B=0$, $S_3 S_2 S_1 S_0 M$ 状态为 1 0 0 1 0,此时数据总线上显示的数据为 DR1 加 DR2 加当前进位标志的和,但这时的进位状态位还没有打入进位锁存器中,它是要靠 T4 节拍来打入的。这个结果是否有进位产生,则要按动微动开关 KK2,若进位指示灯亮,则无进位,反之则有进位。因做加法运算时数据总线一直显示的数据为 $DR1+DR2+CY$,所以当有进位输入到进位锁存器后,总线显示的数据将为加上当前进位锁存器中锁存的进位位的结果。

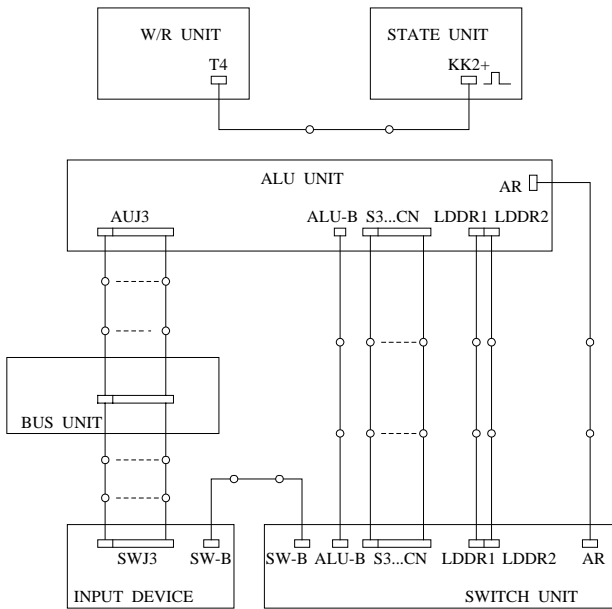


图 2.7-3 带进位运算实验接线图

2.8 移位运算实验

一. 实验目的

1. 了解移位寄存器 74LS299 的功能。
2. 验证移位控制电路的组合功能。

二. 实验设备

TDN-CM+或 TDN-CM++教学实验系统一台。

三. 实验原理

图 2.8-1 所示为移位器及其控制电路。其中使用了一片 74LS299 作为移位发生器，其八位输入／输出端可连接至内总线。74LS299 移位器的片选控制信号为 299-B，低电平有效。T4 为其控制脉冲信号，由“W/R UNIT”单元中的 T4 接至“STATE UNIT”单元中的单脉冲发生器 KK2 上而产生，S0、S1、M 作为移位控制信号，此移位控制逻辑功能如表 2.8-1 所示。

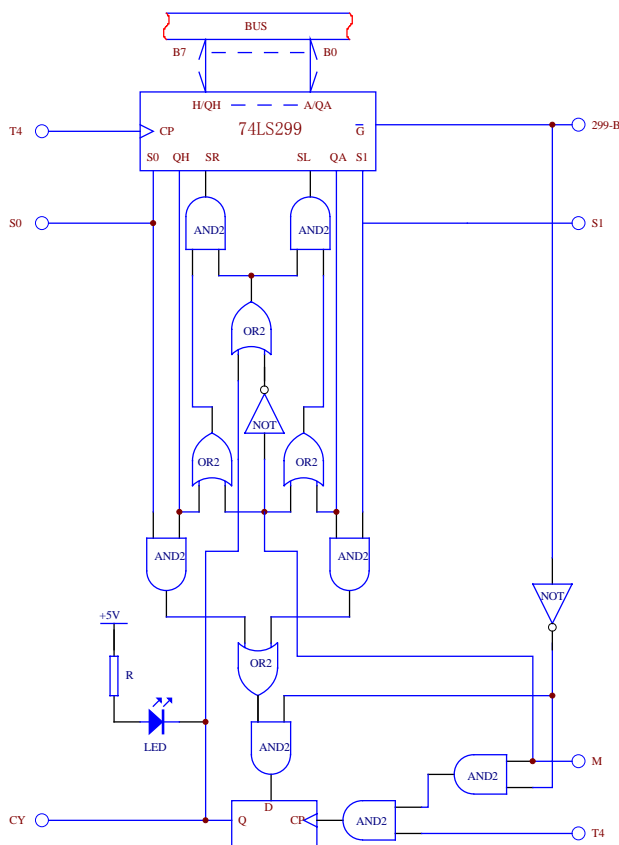


图 2.8-1 移位控制电路原理图

表 2.8-1 移位控制电路功能表

299-B	S1	S0	M	功 能
0	0	0	任意	保持
0	1	0	0	循环右移
0	1	0	1	带进位循环右移
0	0	1	0	循环左移
0	0	1	1	带进位循环左移
任意	1	1	任意	装数

四．实验步骤

- 1. 按图 2.8-2 连接实验电路并检查无误。
- 2. 打开电源开关。
- 3. 向移位寄存器置数。
 - ①拨动输入开关形成二进制数 01101011（或其它数值）。
 - ②使 SWITCH UNIT 单元中的开关 SW-B=0，打开数据输入三态门。
 - ③使 S0=1、S1=1，并按动微动开关 KK2，则将二进制数 01101011 置入了移位寄存器。
 - ④使 SW-B=1，关闭数据输入三态门。
- 4. 移位运算操作。

参照表 2.8-1 中的内容，先将 S1、S0 置为 0、0，检查移位寄存器单元装入的数是否正确，然后通过改变 S0、S1、M、299-B 的状态，并按动微动开关 KK2，观察移位结果。

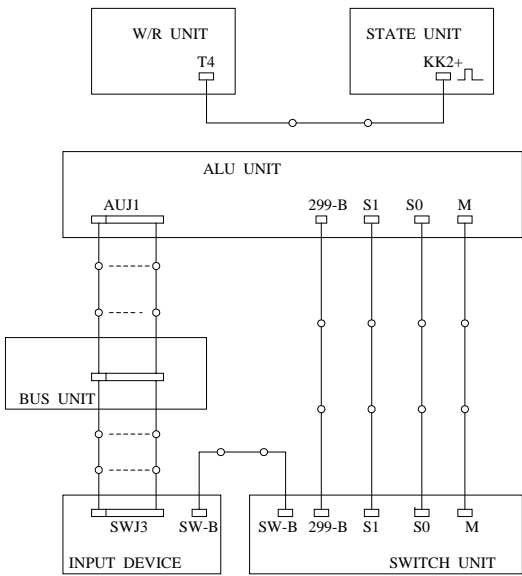


图 2.8-2 移位运算实验接线图

第三章 存储系统

存储器是计算机各种信息存储与交换的中心。在程序执行过程中,所要执行的指令是从存储器中获取,运算器所需要的操作数是通过程序中的访问存储器指令从存储器中得到,运算结果在程序执行完之前又必须全部写到存储器中,各种输入输出设备也直接与存储器交换数据。把程序和数据存储在存储器中,是冯·诺依曼型计算机的基本特征,也是计算机能够自动、连续快速工作的基础。

本章简单介绍了当前应用广泛的各种存储器的原理及使用特性,主要讨论了半导体静态随机存储器、存储器扩展等。安排了两个实验:静态随机存储器实验及 FIFO 存储器实验。

3.1 存储系统概述

3.1.1 存储器的分类

根据存储器的性能和使用方法不同,可以有多种分类方法。

1. 按存储器的位置分类

- (1) 内部存储器 又称主存,位于主机内部。其容量小,但速度快。
- (2) 外部存储器 又称辅存,位于主机的外部。其特点是容量大,但速度较低。

2. 按存取方式分类

- (1) 随机存储器 存储器中任何单元都可以被随机存取,且存取时间与存储单元的物理地址无关。
- (2) 顺序存储器 只能按某种顺序读写存储单元的存储器,存取时间和存储单元的物理地址有关。

3. 按存储介质分类

- (1) 半导体存储器 存储介质为 TTL 或 MOS 半导体器件。
- (2) 磁介质存储器 存储介质为磁性材料,如磁盘、磁带等。
- (3) 光介质存储器 存储介质为金属或磁性材料,但都是通过激光束来读写信息,如光盘等。

4. 按存储读写功能分类

- (1) 读写存储器 既能读出也能写入的存储器。
- (2) 只读存储器 只能读出,不能写入的存储器。

5. 按信息的可保存性分类

- (1) 易失性存储器 断电后信息即消失的存储器。
- (2) 永久性存储器 断电后仍能保存信息的存储器。

3. 1. 2 存储器的分级结构

在计算机系统中常采用三级存储器结构：

1. 高速缓冲存储器 它和主存相比，存取速度快，但容量小，成本较高。
2. 主存储器 是计算机系统的主要存储系统。可存放大量的指令和数据。对于不含高速缓冲存储器的计算机，主存也称为内存。
3. 辅助存储器 又称外部存储器。一般由磁介质存储器构成。它的特点是容量大，但速度慢。

3. 1. 3 存储器的主要技术指标

1. 存储容量 存储容量反映了存储空间的大小。

2. 速度

存储器的速度有三个指标来反映：

- (1) 存取时间：从启动访问存储器的操作到本次操作完成的时间。
- (2) 存储周期：连续启动两次独立的存储器操作之间所需的最小间隔的时间。
- (3) 传输率：又称带宽，单位时间读写二进制的位数。

3. 价格

3. 2 半导体存储器

半导体存储器分为双极型半导体存储器和 MOS 型半导体存储器两种。它们原理相似，只不过双极型存储器速度快，但工艺复杂且集成度低。MOS 半导体存储器又分为静态 MOS 存储器和动态 MOS 存储器。

3. 2. 1 静态随机存储器

一. 基本存储单元

一个基本存储位元能够存储一位二进制信息，能够读出和写入。它的基本结构是由几个 MOS 反向器和控制门管组成一个可控制的双稳态触发器。它的操作示意图如图 3.2-1。

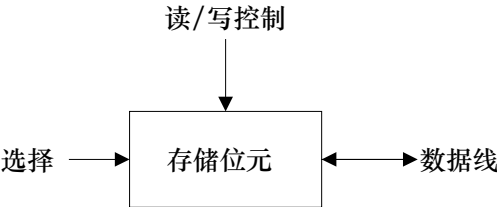


图 3.2-1 存储单元操作示意图

一般每个存储位元有 3 个能传输信号的接线端。选择接线端用于选择这个基本位元是否被选中；读/写控制接线端用于控制对这个基本位元是输入操作还是输出操作；数据接线端用于输入或输出一位二进制数。

二. 静态随机存储器的组成

如图 3.2-2 所示，一个静态存储器由存储体、地址译码器、读/写放大器、I/O 电路和控制逻辑等组成。

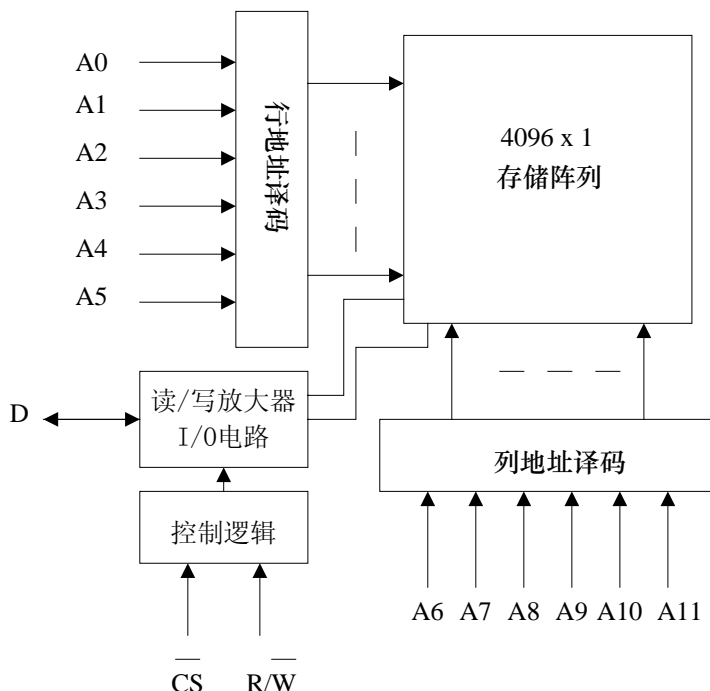


图 3.2-2 4096×1 存储单元结构图

1. 存储体

存储器是所有存储单元的集合。上图是一个 4096 单元的存储器，它用 64 行×64 列的形式构成一个矩阵。由于每个单元只有一位，所以它称为 4K×1 存储体。整个存储体只是一根数据线 D。

2. 地址译码器

地址译码器的功能是把二进制表示的地址转换成输出端的有效电平，来选中所要访问的存储单元。

地址译码器有两种方式：单译码方式和双译码方式。

(1) 单译码方式适用于小容量的存储器。它的地址译码只有一个，译码输出叫字选线，而字选线选择某个字（某存储单元）的所有位。

(2) 双译码可以减少选择线的数目，它分成 X 向和 Y 向两个译码器。若地址的输入量二进制数为 n ，它可分每个译码器有 $n/2$ 个输入端，可以输出 $2^{n/2}$ 个输出状态，那么两个译码器交叉译码结果，共可以输出 $2^{n/2} \times 2^{n/2} = 2^n$ 个输出状态，而译码线只有 $2 \times 2^{n/2}$ 根。如上图中的 4096×1 个存储单元，采用的是双译码结构，地址线共需 12 根，每个译码器分 6 根，X 向译码器输出 64 条选择线，Y 向译码器输出 64 条选择线，共 128 条。要是上图采用单译码方式，则需要 4096 条选择线。

3. 读/写放大器

对写入存储单元或从存储单元中读出的信号进行驱动放大。

4. I/O 电路

控制被选中单元读出或写入，并具有数据缓冲的作用。

5. 控制逻辑

选择该芯片是否被选中。

上面的图示是 4096×1 存储器，若要想得到一个 4096×4 的存储器，最简单的办法就是重叠上述 4 片 4096×1 存储器，将它们的行地址线和列地址线分别并联在一起，而数据线将得到 4 根，D3、D2、D1、D0。

三. 静态 MOS 存储器芯片实例

Intel 2114 是曾广泛使用的小容量存储器，它是 $1K \times 4$ 位静态随机存储器。图 3.2-3、图 3.2-4 分别给出了它的管脚和内部结构。

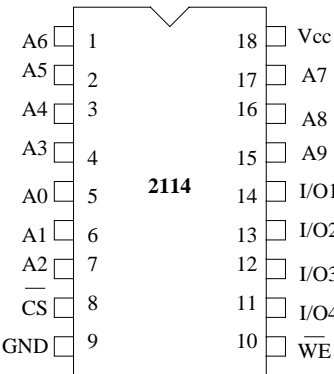


图 3.2-3 2114 管脚图

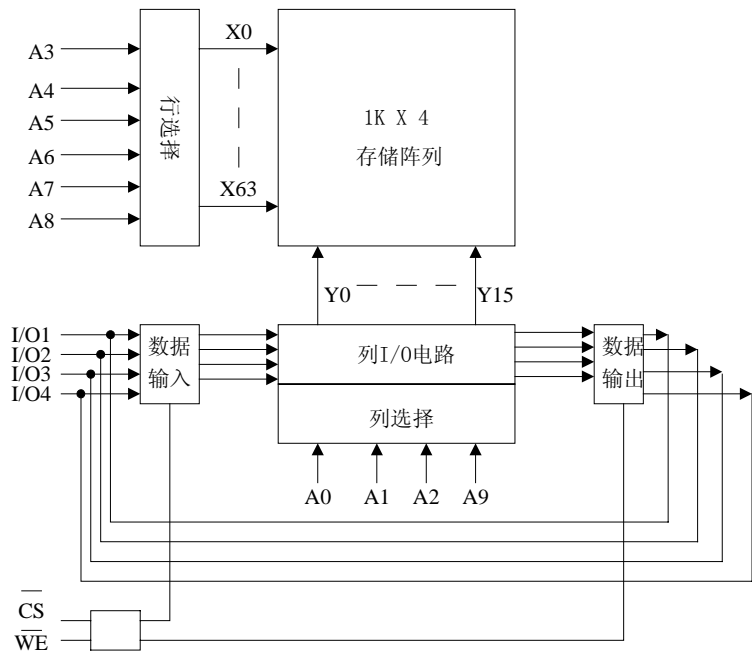


图 3.2-4 2114 逻辑结构图

2114 的容量为 1K，有 10 根地址线 A0~A9，字长 4 位。4 根双向数据线 I/O1~I/O4，WE 为读写控制线，0 为写入，1 为读出，CS 为片选信号线，低电平有效。

图 3.2-5 为 2114 读周期波形图。读周期从地址信息到达地址线开始，经过 t_A 时间，数据读出，这段时间称为读取时间。同时，在数据读出之前应当使片选 CS 有效并有足够的时间。从 CS 有效到数据在数据线上稳定所需要的时间为 t_{CO} ，称为数据有效时间。

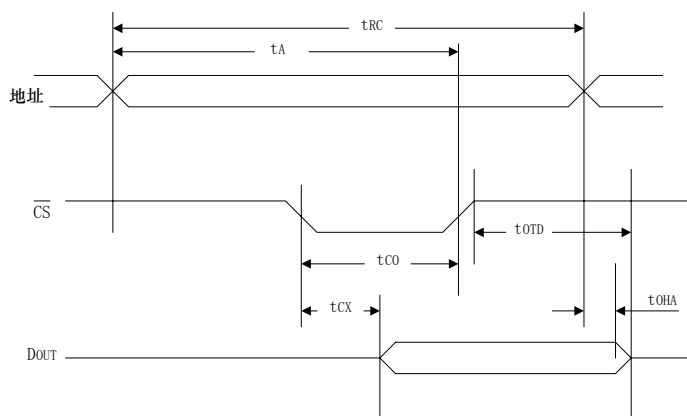


图 3.2-5 2114 读周期波形图

表 3.2-1 2114 读时间周期参数

符号	参数名称	Min (ns)	Max (ns)
t_{RC}	读周期时间	200	
t_A	读出时间		200
t_{CO}	片选到数据输出稳定		70
t_{CX}	片选到输出出现	20	
t_{OTD}	从断开片选到输出变为三态		60
t_{OHA}	地址改变后的维持时间	50	

由于 2114 没有地址锁存器，所以在整个读周期 t_{RC} 中，所加的地址线必须稳定。当读周期结束，地址码改变后，数据仍可以在数据线上保持 t_{OHA} 时间。表 3.2-1 列出了读周期的各项参数。

图 3.2-6 为 2114 写周期波形图。写周期也是从地址信息有效开始，在整个写周期 t_{WC} 期间，地址信息必须保持稳定。待写入的数据必须在 WE 变高之前送到数据线上，且要有 t_{DW} 的稳定时间。在 WE 写线无效之后，地址线还必须保持 t_{WR} 时间，以免误写入。表 3.2-2 为写周期的各项参数。

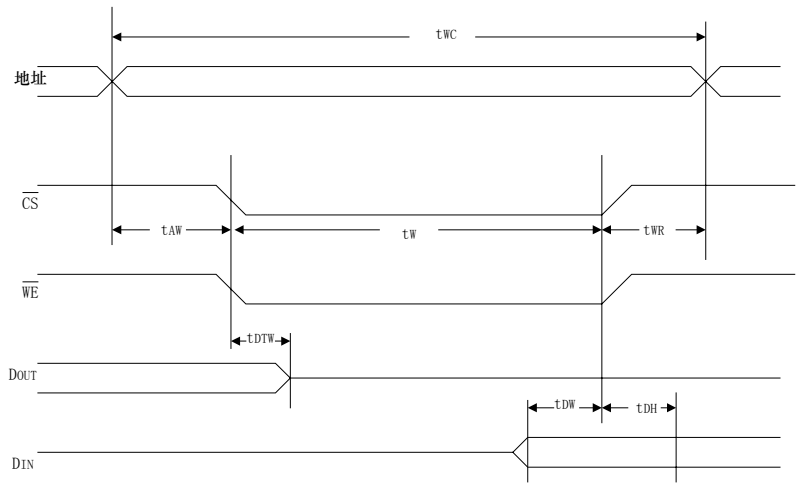


图 3.2-6 2114 写周期波形图

表 3.2-2 2114 写周期时间参数

符号	参数名称	Min (ns)	Min (ns)
tWC	写周期时间	200	60
tW	写数时间	120	
tWR	写恢复时间	0	
tDTW	从写信号有效到输出三态的时间		
tDW	数据有效时间	120	
tDH	写信号无效后数据保持时间	0	

3.2.2 动态随机存储器

动态 MOS 存储器 DRAM 的基本存储原理是：将存储信息以电荷的形式存于电容之上，这种电容可以是 MOS 管栅极电容，或是专用 MOS 电容。通常定义电容充电至高电平时为 1 状态，放电至低电平时为 0 状态。

采用存储电荷方式存储信息，不需要双稳态电路，因而可以简化结构。充电后 MOS 管断开，既可以使电容电荷泄放极少，而且大大降低了芯片的功耗。使得集成度大大提高。

但是电容总存在泄放，时间一长，所存的信息会丢失。所以，必须经过一定的时间就需要对所存的内容重新写一遍，也就是对存 1 的电容重新充电，称为刷新。动态 MOS 存储器采用“读出”方式进行刷新。因为在读出过程中恢复了存储单元的 MOS 栅极电容电荷，并保持原单元的内容，所以读出的过程就是再生的过程。

3.2.3 半导体只读存储器

半导体只读存储器 ROM 是在正常工作时只能读出，而不能写入。它的结构比 RAM 简单的多，从而集成度高，造价低，功耗也小。而且可靠性高，工作时里边的信息不会被改写，掉电后存储的信息也不会丢失。ROM 可以分为以下几类：

1. 掩模式只读存储器 (ROM)

这类存储器由制造厂家做成，用户不能加以修改。

2. 可编程的只读存储器 (PROM)

这类存储器用户可以根据自己的需要来确定 ROM 中的内容，但只能写一次。

3. 紫外线可擦除只读存储器 (EPROM)

这类存储器可以多次擦除编程。使用紫外线照射来擦除。

4. 电可擦除只读存储器 (E²PROM)

这种可编程 ROM 采用电可擦除，可多次编程改写。

3.3 存储器的扩展

存储器同 CPU 连接时，要完成地址线、数据线、和控制线的连接，还要涉及到芯片间的片选译码等。

存储器的扩展一般有三种方法：

1. 位扩展法

位扩展是由于存储器的位数不够，而将多片并联起来，即将所有片子的地址线、片选线及读/写线分别对应并联，而数据线分别引出即可构成多位存储器。下图是用 8 个 8K×1 位的 SRAM 构成 8K×8 位的存储器。如图 3.3-1 所示。

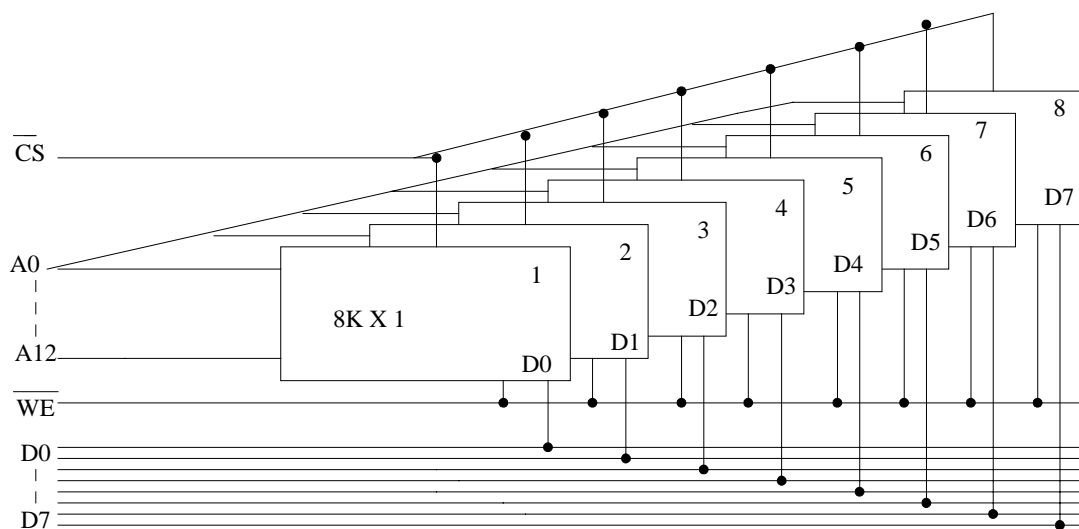


图 3.3-1 位扩展连接图

2. 字扩展法

字扩展是存储器的位数够了，但是总的容量不够，字扩展是仅在字向扩充。它的方法是将芯片的地址线、数据线、读/写线分别对应并联，而将片选信号用译码器将地址线的高位地址译码后分别连接。下图是用 4 个 16K×8 位的存储器通过字扩展构成 64K×8 位的存储器。如图 3.3-2 所示。

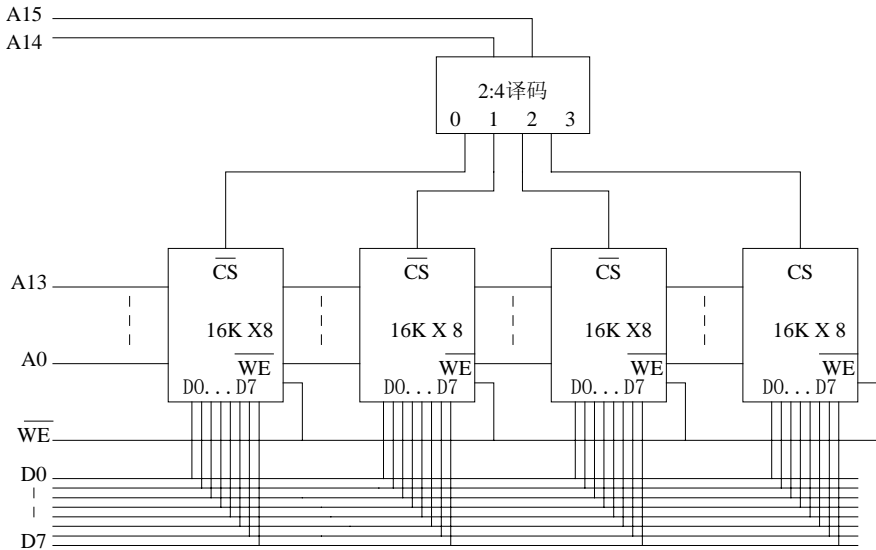


图 3.3-2 字扩展连接图

图中每片 16K×8 芯片的地址线为 A0~A13，数据线 D0~D7，读/写信号线 WE，将它们都对应并联，每个芯片的片选信号线连接至高位地址线 A14、A15 经过 2:4 译码的输出上。当 A15A14 = 00 时，选中 1# 芯片，为 01 时选中 2，10 时选中 3，11 时选中 4#，每片是 16K，就共有 64K 的存储空间。它们的地址分配空间见表 3.3-1。

表 3.3-1 各芯片地址空间分配表

地址 芯片号	A15 A14	A13A12...A1A0	地址空间 A15...A0
1	0 0	0 0 ... 0 0 1 1 ... 1 1	0000H~3FFFH
2	0 1	0 0 ... 0 0 1 1 ... 1 1	4000H~7FFFH
3	1 0	0 0 ... 0 0 1 1 ... 1 1	8000H~BFFFH
4	1 1	0 0 ... 0 0 1 1 ... 1 1	C000H~FFFFH

3. 字位同时扩展法

既有位扩展又有字扩展。扩展时先进行位扩展，扩展到要求的位数，再以上述扩展的结构为组件，进行字扩展到要求的存储单元数。

3. 4 双端口存储器

双端口存储器具有两组相互独立的读/写控制电路，两个端口可以并行地进行读/写操作，从而达到提高存储器的使用效率，提高速度。

双端口存储器的机构框图如图 3.4-1。它有两组数据总线、地址总线和控制总线，从而形成了两个端口。每个端口有自己的地址寄存器（MAR）、地址译码器、数据缓冲器（MDR）和读/写控制电路。两个端口彼此独立，可同时访问存储器。

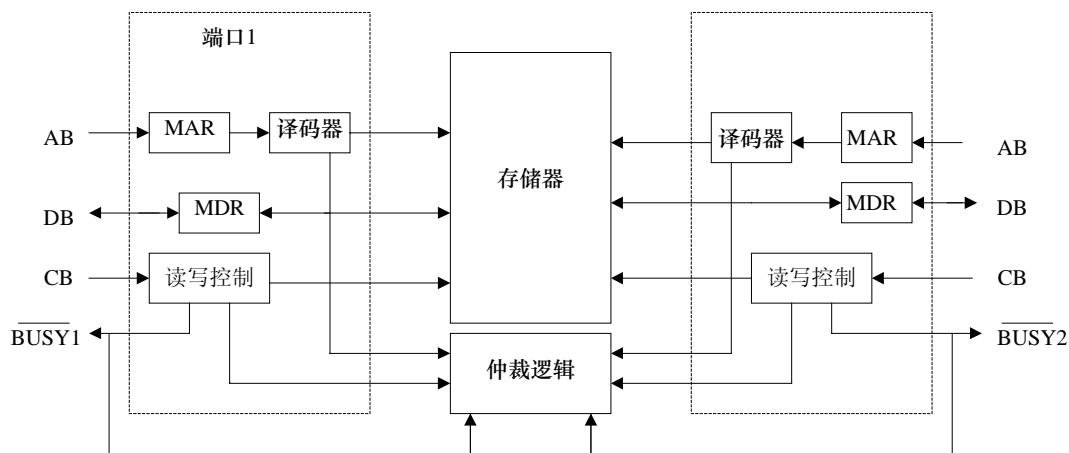


图 3.4-1 双端口存储器结构框图

当两个端口同时访问不同的地址时，可以无冲突的进行读写操作。但当两个端口同时访问相同的地址时，便发生冲突，此时将要根据 BUSY 标志来决定哪个口操作。由片上的判断逻辑决定哪个端口优先进行操作，而对另一个端口置 BUSY 标志，暂时关掉此端口，当优先端口完成操作后，使被延迟端口的 BUSY 标志复位，允该端口进行操作。

3. 5 高速缓冲存储器（Cache）

高速缓冲存储器（Cache）是插在 CPU 和主存之间的一个快速小容量的存储器，它主要是为解决 CPU 和主存之间的速度匹配问题而设置的。它的速度比主存快，具有和 CPU 相近的速度，有了 Cache，就能高速地向 CPU 提供指令和数据，从而加快了程序的执行速度。它们的关系图如图 3.5-1。

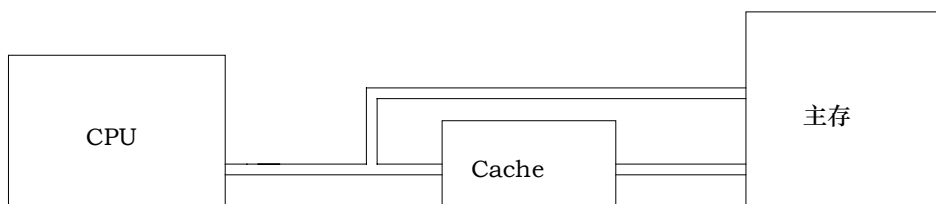


图 3.5-1 CPU 与 Cache 关系图

在计算机中增加 Cache 的目的，就是在性能上使主存储器的平均读出时间尽可能的接近与 Cache 的读出时间。要实现这个目的，就是要将 CPU 将要读取的主存单元内容先预先读到 Cache 中。由于程序编制的局部性，即程序总是按顺序编写和编址的，指令地址顺序存放在存储器中；一条指令执行后一般是执行紧接着的那条指令，只有遇到转移指令时才会跳到另一个区域，但跳转之后又会顺序执行；还有就是当执行一个循环程序时，会在一块小的区域内重复循环执行若干指令；对于存储的数据也是如此。按照一定的预读算法，可以实现很高的 Cache 命中率。

当 CPU 要从主存中读取一个字，先把这个字的地址传给 Cache，检查这个字是否在 Cache 中，如果在，就把这个字直接从 Cache 送到 CPU 中，如果不在，则把这个地址传给主存，从主存中读出这个字。同时，把这个字附近地址单元的内容取到 Cache 中，以满足下次访问的字能在 Cache 中。

为把信息读取到 Cache 中，必须应用某种函数把主存地址映象到 Cache 中定位，称为地址映象。这些函数通常称为映象函数。在信息按这种映象关系装入到 Cache 后，执行程序时，将主存地址变换成 Cache 地址，这个变换过程称为地址变换。

当要将新的主存中的一块数据装入到 Cache 替换掉原存储的一块数据，必须要有相应的替换算法。常用的替换策略有：

最近最少使用 (LRU) 策略，即替换掉那些在 Cache 中驻留时间最长且未被引用的块。

先进先出 (FIFO) 策略，即替换掉那些在 Cache 中停留时间最长的块。

最不常使用 (LFU) 策略，即替换掉引用次数最少的块。

3.6 静态随机存储器实验

一. 实验目的

掌握静态随机存储器 RAM 工作特性及数据的读写方法。

二. 实验设备

1. TDN-CM+或 TDN-CM++教学实验系统一台。
2. PC 微机（或示波器）一台。

三. 实验原理

实验所用的半导体静态存储器电路原理如图 3.6-1 所示,实验中的静态存储器由一片 6116 (2K×8) 构成,其数据线接至数据总线,地址线由地址锁存器 (74LS273) 给出。地址灯 AD0~AD7 与地址线相连,显示地址线内容。数据开关经一个三态门 (74LS245) 连至数据总线,分时给出地址和数据。

因地址寄存器为 8 位,所以接入 6116 的地址为 A7~A0,而高三位 A8~A10 接地,所以其实际容量为 256 字节。6116 有三个控制线:CE (片选线)、OE (读线)、WE (写线)。当片选有效 (CE=0) 时,OE=0 时进行读操作,WE=0 时进行写操作。本实验中将 OE 常接地,在此情况下,当 CE=0、WE=0 时进行读操作,CE=0、WE=1 时进行写操作,其写时间与 T3 脉冲宽度一致。

实验时将 T3 脉冲接至实验板上时序电路模块的 TS3 相应插孔中,其脉冲宽度可调,其它电平控制信号由“SWITCH UNIT”单元的二进制开关模拟,其中 SW-B 为低电平有效,LDAR 为高电平有效。

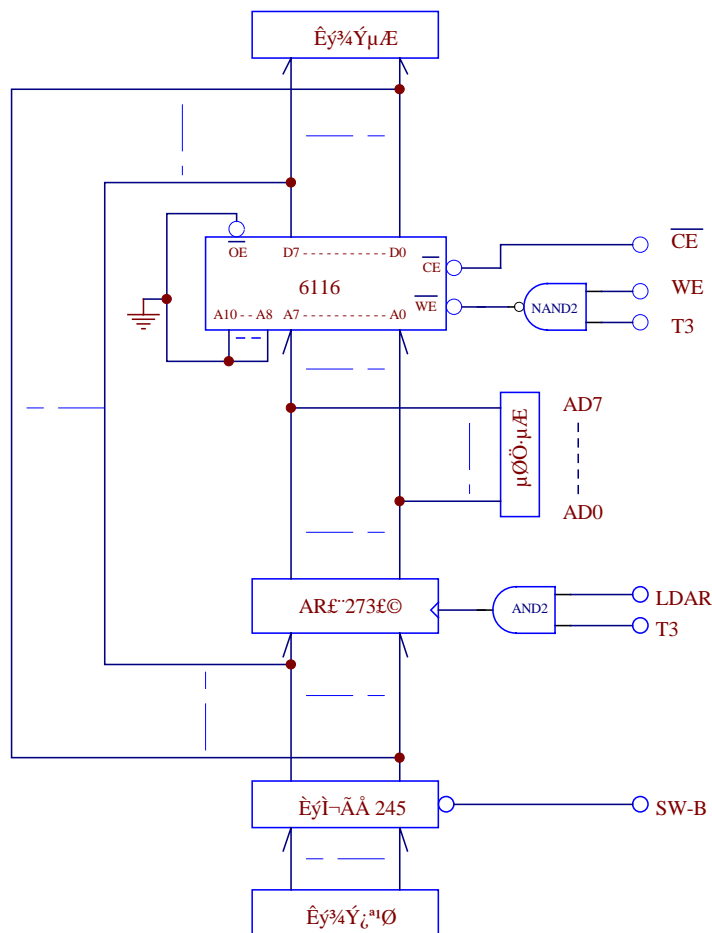


图 3.6-1 存储器实验原理图

四．实验步骤

(1) 形成时钟脉冲信号 T3。具体接线方法和操作步骤如下：

- ① 接通电源，用示波器接入方波信号源的输出插孔 H23，调节电位器 W1 及 W2，使 H23 端输出实验所期望的频率及占空比的方波。
 - ② 将时序电路模块 (STATE UNIT) 单元中的 ϕ 和信号源单元 (SIGNAL UNIT) 中的 H23 排针相连。
 - ③ 在时序电路模块中有两个二进制开关 “STOP” 和 “STEP”。将 “STOP” 开关置为 “RUN” 状态、“STEP” 开关置为 “EXEC” 状态时，按动微动开关 START，则 TS3 端即输出为连续的方波信号，此时调节电位器 W1，用示波器观察，使 T3 输出实验要求的脉冲信号。当 “STOP” 开关置为 “RUN” 状态、“STEP” 开关置为 “STEP” 状态时，每按动一次微动开关 START，则 T3 输出一个单脉冲，其脉冲宽度与连续方式相同。若用 PC 联机软件中的示波器功能也能看到波形，可以代替真实示波器。
- (2) 按图 3.6-2 连接实验线路，仔细查线无误后接通电源。

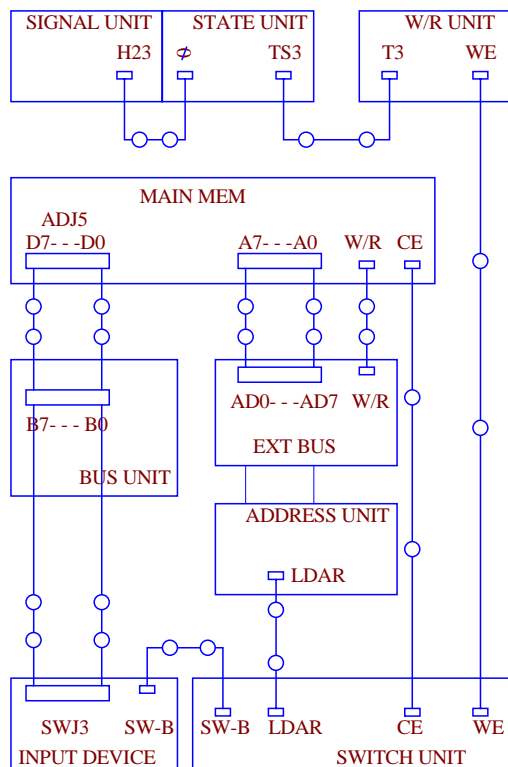


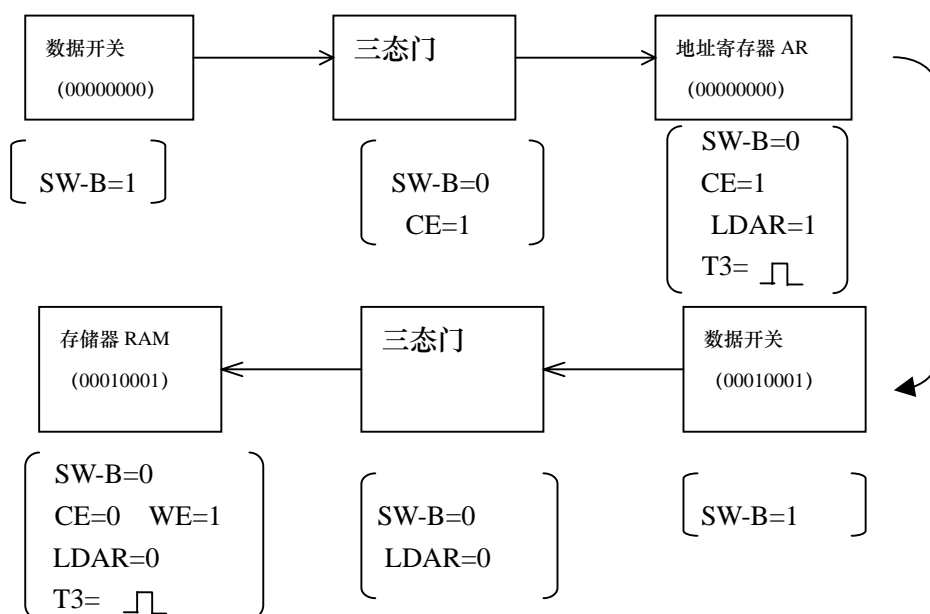
图 3.6-2 静态随机存储器实验接线图

(3) 写存储器

给存储器的 00、01、02、03、04 地址单元中分别写入数据 11、12、13、14、15。

由上面的存储器实验原理图看出，由于数据和地址全由一个数据开关来给出，这就要分时地给出。下面的写存储器要分两个步骤，第一步写地址，先关掉存储器的片选（CE=1），打开地址锁存器门控信号（LDAR=1），打开数据开关三态门（SW-B=0），由开关给出要写存储单元的地址，按动 START 产生 T3 脉冲将地址打入到地址锁存器，第二步写数据，关掉地址锁存器门控信号（LDAR=0），打开存储器片选，使处于写状态（CE=0，WE=1），由开关给出此单元要写入的数据，按动 START 产生 T3 脉冲将数据写入到当前的地址单元中。写其它单元依次循环上述步骤。

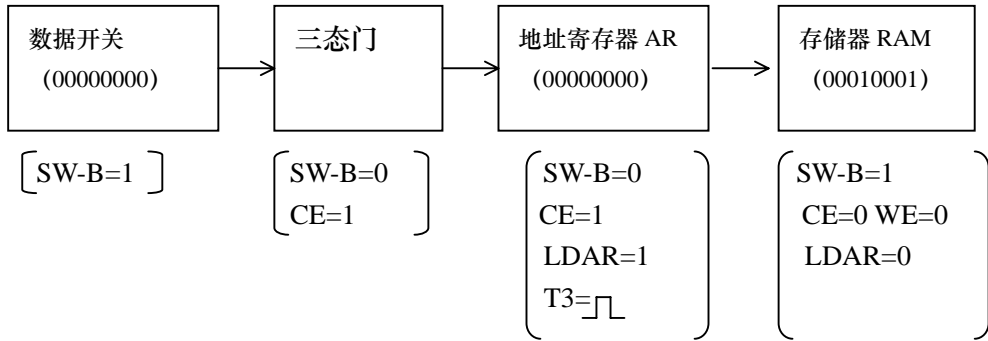
写存储器流程如下：（以向 00 号单元写入 11 为例）



(4) 读存储器

依次读出第 00、01、02、03、04 号单元中的内容，观察上述各单元中的内容是否与前面写入的一致。同写操作类似，读每个单元也需要两步，第一步写地址，先关掉存储器的片选（CE=1），打开地址锁存器门控信号（LDAR=1），打开数据开关三态门（SW-B=0），由开关给出要写存储单元的地址，按动 START 产生 T3 脉冲将地址打入到地址锁存器；第二步读存储器，关掉地址锁存器门控信号（LDAR=0），关掉数据开关三态门（SW-B=1），片选存储器，使它处于读状态（CE=0，WE=0），此时数据总线上显示的数据即为从存储器当前地址中读出的数据内容。读其它单元依次循环上述步骤。

读存储器操作流程如下：（以从 00 号单元读出 11 数据为例）



3. 7 FIFO 先进先出存储器实验

一. 实验目的

了解及掌握先进先出 (FIFO) 存储器的工作特性及其读写方法。

二. 实验设备

1. TDN-CM+或 TDN-CM++教学实验系统一台。
2. PC 微机一台。

三. 实验原理

本实验用 isp1032 芯片来实现一个简单的 8 位 \times 4 的 FIFO，其信号引脚如图 3.7-1：

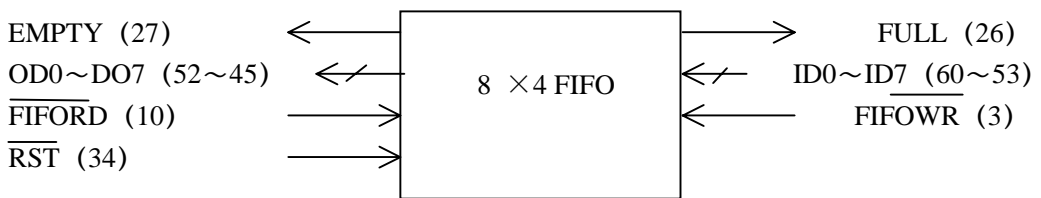


图 3.7-1 定义 FIFO 在 1032 中对应的管脚

其各信号的功能为：

\overline{EMPTY} ：FIFO 存储器为空标志，高电平有效。

$FULL$ ：FIFO 存储器满标志，高电平有效。

\overline{RST} ：清 FIFO 存储器为空。

\overline{FIFOWR} ：FIFO 存储器写入信号，低电平有效。

$\overline{\text{FIFORD}}$: FIFO 存储器读信号, 低电平有效。

ID0~ID7: FIFO 存储器输入数据线。

OD0~OD7: FIFO 存储器读出数据线。

各信号后的括号内的数字为本设计在 CPLD 中定义的相应的管脚号。

此 8×4 FIFO 的内部逻辑图如下图 3.7-2:

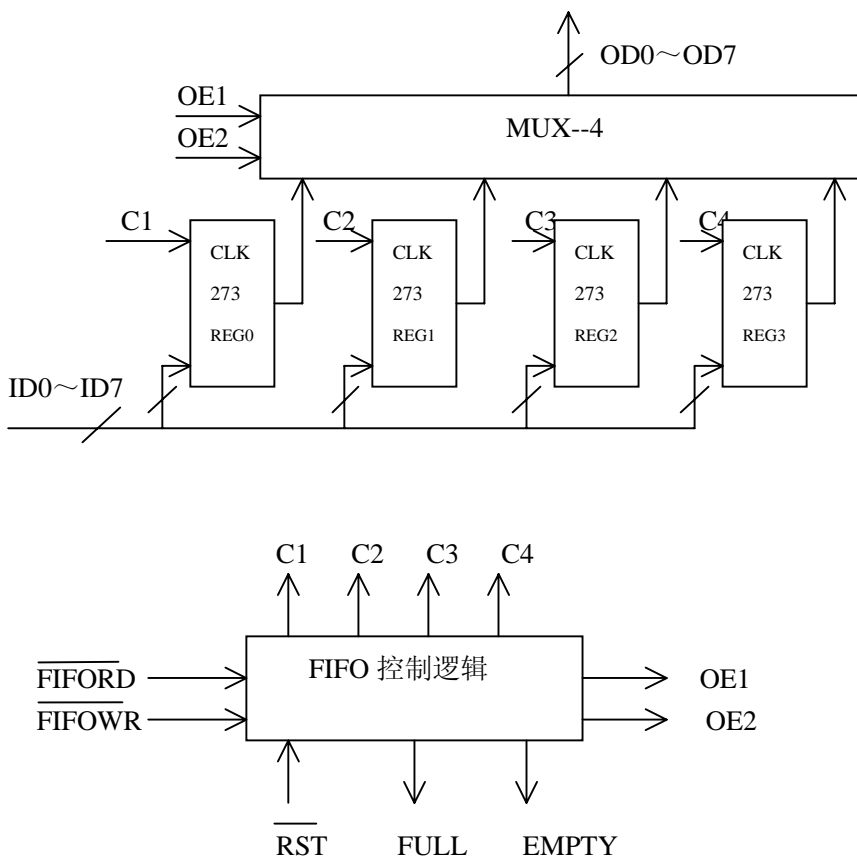


图 3.7-2 FIFO 内部逻辑图

四. 实验步骤

(1) 编写 cpld 芯片设计程序

按照上述功能要求及管脚说明, 进行 CPLD 芯片设计。

(2) 编译所设计的程序, 并将生成的 JEDEC 文件下载至 ISPLSI1032 中。

(3) 按图 3.7-3 实验连线图接线。

(4) 实验操作步骤

接线图中 OO1、OO2、OOE1、OOE2、OOEE1、OOEE2 是六个观察记数的指示灯, 其

中 OO1、OO2 是写信号记数，OOE1、OOE2 是读信号记数，OOEE1、OOEE2 是 FIFO 中的数据个数。FULL 及 EMPTYy 是满和空标志灯。

实验时，将 SWITCH UNIT 单元中的 SW-B 开关置为“0”，然后拨动系统右下脚的 CLR 清零开关使读、写信号记数清零。给 INPUT DEVICE 单元中置一个数，按动 START，此时将该数写入到 FIFO 中，依次写四次后，FULL 满标志置位。此时再也写不进去；然后连续按动 KK2-读信号，将顺序读出所存的四个数，数据总线显示灯及 OUTPUT UNIT 单元中的数码管显示所读出的数据。四个数全部读出后，EMPTYy 空标志置位。检查执行结果是否与理论值一致。

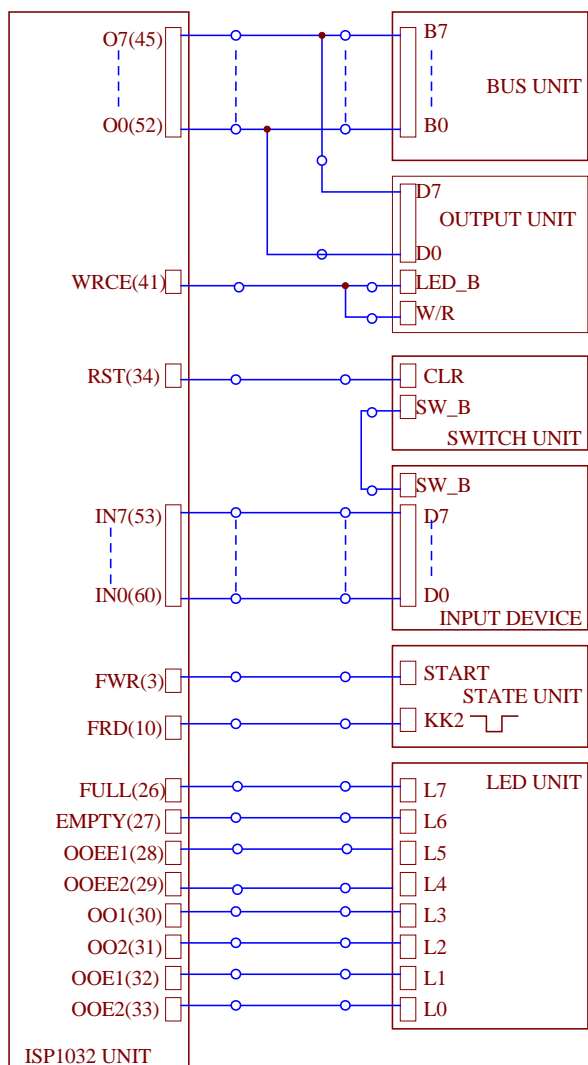


图 3.7-3 FIFO 实验接线图

第四章 控制器

控制器是计算机的核心部件，计算机的所有硬件都是在控制器的控制下，完成程序规定的操作。控制器的基本功能就是把机器指令转换为按照一定时序控制机器各部件的工作信号，使各部件产生一系列动作，完成指令所规定的任务。

控制器的实现有两大类：硬布线控制和微程序控制。

本章讨论了控制器的概念、基本功能和结构，然后从两个方面分别讨论了微程序控制器和硬布线控制器各自的结构特点。安排了两个实验：微程序控制器实验和硬布线控制器实验。

4.1 控制器的基本功能和结构

4.1.1 控制器的基本功能

计算机是根据人们编写的放在主存中的程序来完成一系列任务的。但如何来完成每一条指令的操作呢，这就需要控制器来“翻译”，即把指令转化为按照一定的时序控制机器各部件的工作信号，使各部件产生一系列动作，完成指令所规定的任务。

计算机程序是由若干条指令组成的，而每一条机器指令又执行若干条微命令，每条微命令又由若干条微操作组成，每个微操作控制一个现实微命令的硬件逻辑电路。

计算机对信息处理的过程就是不断地取指令、分析指令和执行指令。所以控制器具有如下基本功能：

1. 取指令

从主存中取出当前程序计数器（PC）所指当前指令地址中的指令。

2. 分析指令

对取出的指令进行分析，指出它要求什么操作，产生相应的操作控制信号。

3. 执行指令

根据分析指令的结果，得到“操作命令”和“操作数地址”，产生一系列对各部件进行控制的信号序列，从而完成指令的执行。

4. 对异常情况及中断请求处理。

对发生异常情况或设备发出中断请求，则暂停当前的程序并转到相应的中断服务程序，处理完后返回原程序继续执行。

4.1.2 控制器的组成

控制器由如下基本部分组成：

1. 程序计数器（PC）：用来存放当前或下条将要执行的指令的地址。

2. 指令寄存器（IR）：用来存放当前正在执行的指令。

3. 指令译码器 (ID): 它对指令寄存器中的指令进行译码, 向控制器提供特定的操作控制信号。

4. 时序发生器: 用于产生机器所需的各种时序信号, 以控制有关部件在不同的时间完成不同的操作。

5. 微操作控制信号形成部件: 根据不同的时序关系、操作码等有关状态条件给出所需要的操作控制信号序列。

6. 中断机构: 对异常情况及其某些请求的处理。

7. 总线控制逻辑: 对总线上各部件使用总线的仲裁控制机构。

4.1.3 控制器的结构

控制器的结构可分为硬布线控制和微程序控制。

1. 硬布线控制器: 又称组合逻辑控制器。用组合逻辑电路, 将时序信号、指令译码信息、条件信号等输入逻辑信号转换成一组各部件需要的控制信号。

2. 微程序控制器: 它是将指令的微操作信号编成微指令, 存放在控制存储器中。当机器运行时, 一条一条地读出这些微指令, 从而产生各部件所需要的控制信号。

4.2 时序控制信号

4.2.1 时序部件的组成

计算机的工作是按照时序分步地执行。这就需要能产生周期节拍、脉冲等时序信号的部件, 称为时序发生器。如图 4.2-1 所示。

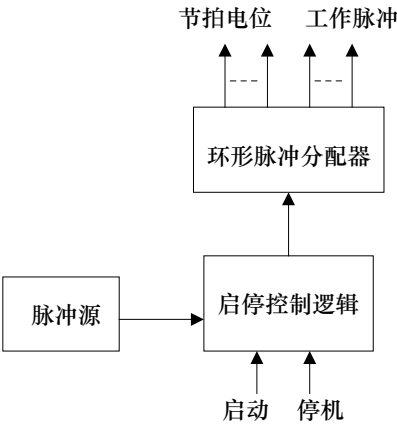


图 4.2-1 时序发生器

它包括:

1. 脉冲源 又称主振荡器, 为计算机提供基准时钟信号。

2. 脉冲分配器 对主频脉冲进行分频, 产生节拍电位和脉冲信号。时钟脉冲经过脉冲发生器产生时标脉冲、节拍电位及周期状态电位。一个周期状态电位包含多个节拍电位, 而

一个节拍单位又包含多个时标脉冲。

3. 启停控制电路 用来控制主脉冲的启动和停止。

4. 2. 2 时序控制方式

由于不同的机器指令对应的操作控制步序列的长短不一样，序列中各操作的执行时间也不相同，所以就需要有不同的时序控制方式来形成不同的操作控制步序列。

时序控制方式一般有三种：同步控制方式、异步控制方式及联合控制方式。

1. 同步控制方式

所谓同步控制方式，就是在指令执行时所需要的机器周期及时钟周期是固定不变的。根据不同的情况，同步控制方式可选如下方案：

(1) 采用完全相同的机器周期执行各种不同的指令。这种方式实现简单，但是对于需要周期少的指令将造成时间浪费。

(2) 采用延长机器周期的方法解决执行时间不统一的问题。对于大多数指令安排在固定的周期内完成，对于少数需要周期多的指令可采用延长机器周期的办法解决。

(3) 中央控制和局部控制相结合。中央控制就是对大多数指令采用固定的周期执行。而局部控制就是对于少数复杂的需要机器周期多的指令可采用另外的时序进行。

2. 异步控制方式

这种方式执行指令没有固定的机器周期数，前一个操作完成后产生一个完成信号，去继续启动下一个操作。这种方式可以没有统一的时钟，由各部件自己控制。最大的优点是没有浪费时间，但是设计复杂，硬件花费较多。

3. 联合控制方式

这种方式是同步方式和异步方式的结合。即对于大多数操作序列可以统一的，安排在具有固定周期和时钟同步的时序信号控制下执行，而对于那些难于统一的就采用异步控制，不需要时钟信号同步，按实际占用时间操作。通过“结束”——“起始”方式和公共的同步控制部分衔接起来。

4. 3 微程序控制器

4. 3. 1 微程序控制器的原理及结构

1. 微程序控制器的原理

微程序控制器的基本思想可以概括如下：

① 将控制器所需要的微命令，以微代码的形式编成微指令，存入一个控制存储器中，这个控制存储器由只读存储器 ROM 构成。在计算机运行时，从控存中取出微指令，用其所包含的微命令来控制有关部件的操作。

② 将每种机器指令分解为若干条微操作序列，用若干条微指令来解释一条机器指令。再根据整个指令系统的需要，编制出一套完整的微程序，预先存入控存中。

2. 有关术语与概念

这里涉及两个层次，一层是机器指令级——机器指令，它是程序员可以操作的，放在主存中；另一层是微程序级——微指令，它对程序员是透明的，不能操作，而底层硬件设计者可以操作的，它存放在控制存储器中。

微命令：又称微信号，是直接作用于控制电路的控制命令，是控制信号序列的最小单位。控制部件通过控制总线向执行部件发出各种微命令。

微操作：由微命令控制实现最基本的操作。

微周期：从控存中读取一条微指令并执行完相应的所有微操作所用的时间。通常为一个时钟周期。

微指令：每个微周期所需的微命令组成一条微指令，它以二进制形式编码存放在控存中，一个控存单元存放一条微指令。

微程序：由若干条微指令组成的有序序列。每条机器指令都对应着一段微程序。

3. 微程序控制器的结构

微程序控制器的组成结构图如图 4.3-1。

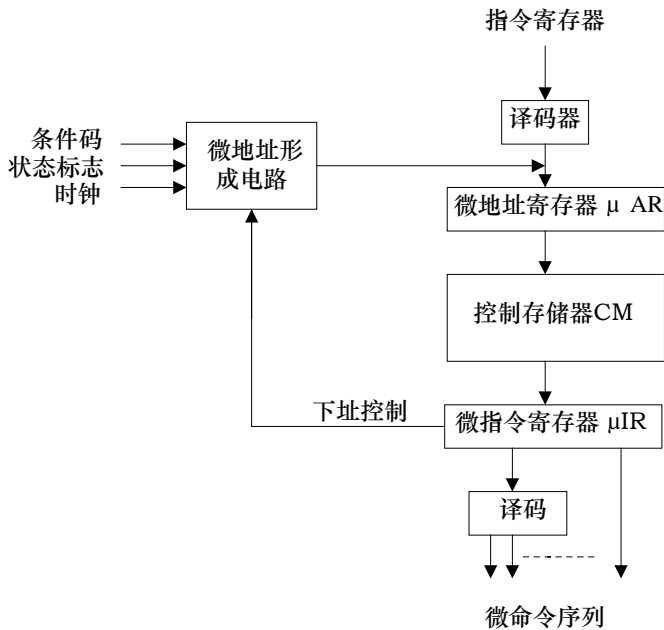


图 4.3-1 微程序控制器组成结构图

①控制存储器（CM）：用来存放微程序的存储器，它是只读存储器。在它的每个单元存放一条微指令的代码。它的字长就是微指令字的长度。

②微指令寄存器（μIR）：用于存放由控存中读出的一条微指令。它分为两大部分：一部分提供微命令的控制字段，其代码直接或经过译码后产生微命令；另一部分为顺序控制字段，它指明后续微地址的形成方式，以控制微程序连续运行。

③微地址形成电路：根据微指令寄存器中的顺序控制字段、机器指令及有关状态等确定后续微地址形成的电路。

④微地址寄存器（μAR）：用于保存下一条微指令的地址。读取 CM 中的微指令时是按

微地址寄存器中指向的单元读取的，当微指令读出或执行完后，微地址形成电路将后续微地址打入到微地址寄存器中，为下次读取微指令作好准备。

4. 机器指令的执行过程

①在微指令序列中，有一些指令是进行公共操作的，如取指等。在一个机器周期的开始，先执行取指操作，从主存中取出机器指令，送入指令寄存器，修改程序计数器 PC 的值。

②根据指令译码器对指令译码，取出操作码通过微地址形成电路产生对应的微程序入口地址，并送入微地址寄存器。

③从被选中的单元中取出相应的一条微指令送入微指令寄存器。

④微指令寄存器中的微指令操作控制字段经过译码或直接输出产生一组微命令，送往有关的执行部件，在时序的控制下完成其规定的微操作。

⑤微指令寄存器中的顺序控制字段及有关状态通过微地址形成电路产生后续微地址，并打入到微地址寄存器中，继续读取下一条微指令。

⑥执行完一条机器指令对应的一段微程序后，又返回到公共微指令取指，开始下一个机器指令的执行。

4.3.2 微指令的编码方式

微指令的编码，就是微指令中的操作控制字段采用什么样的表示方法，通常有三种方法：

1. 直接表示法

微指令操作控制字段中的每一位就是一个微命令，直接对应一种微操作，送往相关部件的控制点。这种方式简单直观，编码方便，但是效率低，微指令字长会变的很长。

2. 分段译码表示法

将微指令控制字段分为若干的小组，每组再经过分别译码就可以得到更多的微命令。基本的分组原则是将互斥的微操作（即不能在同一时间或同一个机器周期并行执行的微操作）分在一个组中，将相容的微操作分在不同的组中。

这种方式结构比较清晰，易于编制微程序，也较容易修改扩充。

3. 混合表示法

将直接表示法和分段译码表示法混合使用。这种方法使微指令编码更为灵活多样。

4.3.3 微指令的格式分类

微指令格式和在一条微指令中能同时执行的微操作的种类和数量有关。大体分为两类：

1. 垂直型微指令

每条微指令只要求能控制执行一个微命令，完成一个基本操作。这种指令微指令较短，控存位码利用率较高，微程序容易编制，但并行度低，执行效率低，编写的微程序较长，完成一条机器指令需要更多条微指令。

2. 水平型微指令

在一条微指令中可以并行地执行多个基本操作。这种方式指令的优缺点正好和垂直型相反，执行效率高，灵活性强，微程序条数少，但微指令长，复杂程度高。

从上面的说明可看出，微指令格式和 CPU 数据通路图的结构有关。如果 CPU 只有简单的单数据通路，一般就只能采用垂直型微指令。若数据通路有多总线结构，才可以使用水平型微指令。

4.3.4 后续微地址的形成方法

微指令要按顺序一条一条的执行下去，就要能确定下一条微指令的地址。通常有两种方法：

1. 计数器方式（增量方式）

这种方式采用一个微地址计数器（ μ PC）来产生下一条微指令的微地址。当顺序执行程序时，后续微指令地址由现行的微指令地址加上一个增量来产生；当遇到转移时，根据测试条件将要转移的地址打入到 μ PC 中。

2. 断定方式

这是一种直接给定与测试断定相结合的方式。后续微地址直接包含在当前微指令的代码中。将指令中的微地址分成两部分：直接给定的高位部分，根据断定条件形成的低位部分。在微指令的顺序控制段中设置或注明断定条件，即微地址低位段的形成条件，则根据测试条件就可以形成不同的低位地址，产生不同的微程序分支。

4.4 微程序控制器实验

一. 实验目的

1. 掌握时序发生器的组成原理。
2. 掌握微程序控制器的组成原理。
3. 掌握微程序的编制、写入，观察微程序的运行。

二. 实验设备

TDN-CM+或 TDN-CM++教学实验系统一台。

三. 实验原理

微程序控制器的基本任务是完成当前指令的翻译和执行，即将当前指令的功能转换成可以控制硬件逻辑部件工作的微命令序列，完成数据传送和各种处理操作。它的执行方法就是将控制各部件动作的微命令的集合进行编码，即将微命令的集合仿照机器指令一样，用数字代码的形式表示，这种表示称为微指令。这样就可以用一个微指令序列表示一条机器指令，这种微指令序列成为微程序。微程序存储在一种专用的存储器中，称为控制存储器。

实验所用的时序控制电路框图如图 4.4-1 所示，可产生 4 个等间隔的时序信号 TS1~TS4，其中 Φ 为时钟信号，由实验台左上方的方波信号源提供，可产生频率及脉宽可调的方波信号。学生可根据实验自行选择方波信号的频率及脉宽。图中 STEP（单步）是来自实验板上方中

部的一个二进制开关 STEP 的模拟信号。START 键是来自实验板上左部的一个微动开关 START 的按键信号。当 STEP 开关为 0 时 (EXEC), 一旦按下 START 启动键, 时序信号 TS1~TS4 将周而复始地发送出去。当 STEP 为 1 (STEP) 时, 一旦按下 SATRT 启动键, 机器便处于单步运行状态, 即此时只发送一个 CPU 周期的时序信号就停机。利用单步方式, 每次只读一条微指令, 可以观察微指令的代码与当前微指令的执行结果。另外, 当机器连续运行时, 如果 STEP 开关置 “1” (STEP), 也会使机器停机, 或使 CLR 开关执行 1→0→1 操作也可以使时序清零。时序状态图见图 4.4-7。

由于时序电路的内部线路已经连好, 所以只需将时序电路与方波信号源连接, 即将时序电路的时钟脉冲输入端 Φ 接至方波信号发生器输出端 H23 上, 按动启动键 START 后, 就可产生时序信号 TS1~TS4。时序电路的 CLR 已接至实验板右下方的 CLR 模拟开关上。

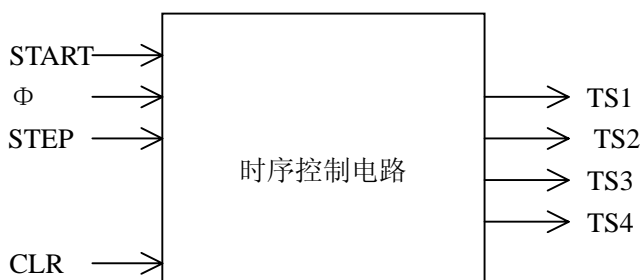


图 4.4-1 时序控制电路框图

微程序控制电路与微指令格式

①微程序控制电路

微程序控制器的组成见图 4.4-2, 其中控制存储器采用 3 片 2816 的 E²PROM, 具有掉电保护功能, 微命令寄存器 18 位, 用两片 8D 触发器 (273) 和一片 4D (175) 触发器组成。微地址寄存器 6 位, 用三片正沿触发的双 D 触发器 (74) 组成, 它们带有清 “0” 端和预置端。在不判别测试的情况下, T2 时刻打入微地址寄存器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置端将某一触发器置为 “1” 状态, 完成地址修改。

在该实验电路中设有一个编程开关 (位于实验板右上方), 它具有三种状态: PROM (编程)、READ (校验)、RUN (运行)。当处于 “编程状态” 时, 学生可根据微地址和微指令格式将微指令二进制代码写入到控制存储器 2816 中。当处于 “校验状态” 时, 可以对写入控制存储器中的二进制代码进行验证, 从而可以判断写入的二进制代码是否正确。当处于 “运行状态” 时, 只要给出微程序的入口微地址, 则可根据微程序流程图自动执行微程序。图中微地址寄存器输出端增加了一组三态门, 目的是隔离触发器的输出, 增加抗干扰能力, 并用来驱动微地址显示灯。

②微指令格式

微指令字长共 24 位, 其控制位顺序如表 4.4-1:

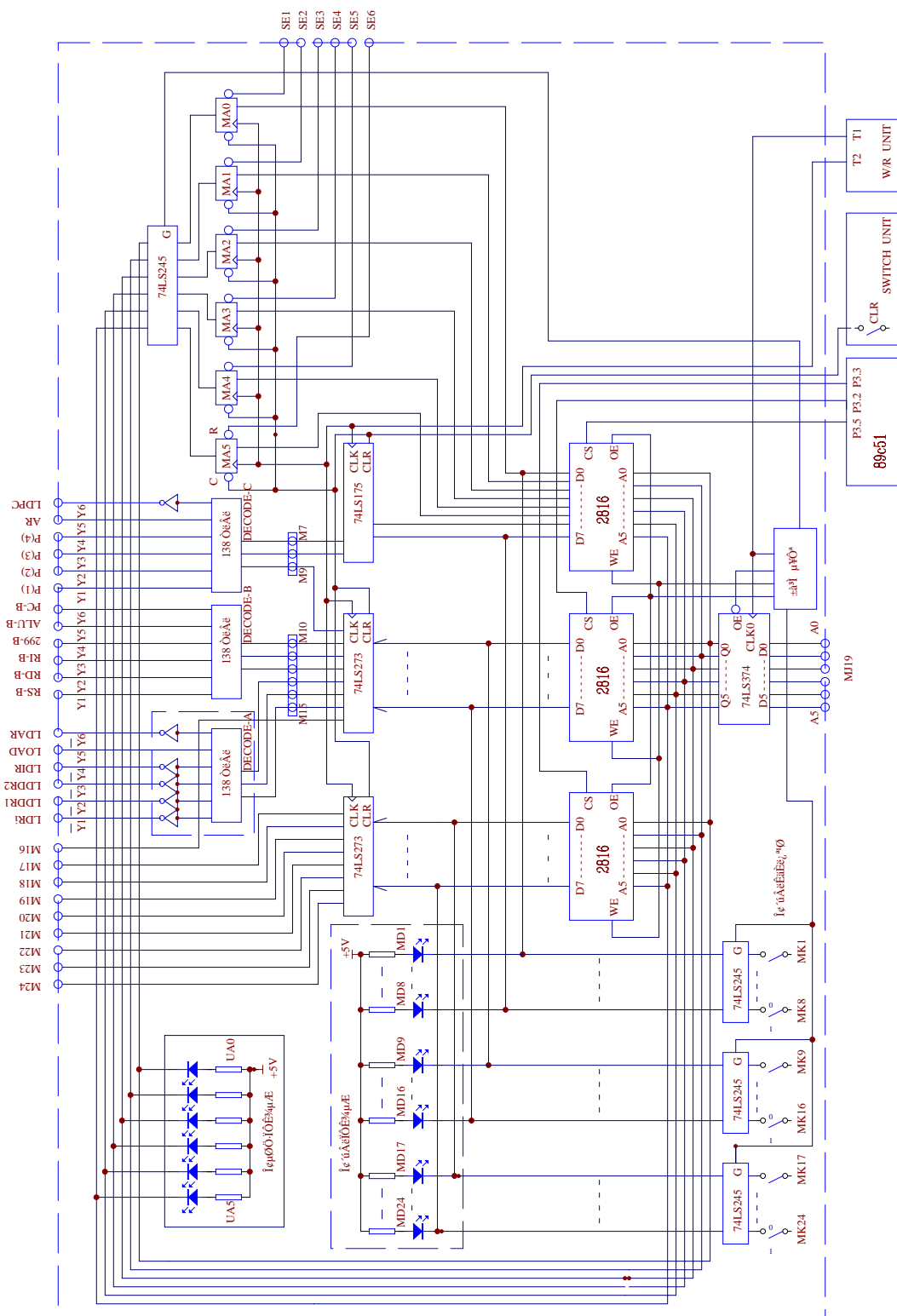


图 4.4-2 微控器实验原理图

表 4.4-1

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
S3	S2	S1	S0	M	Cn	WE	A9	A8	A	B	C	uA5	uA4	uA3	uA2	uA1	uA0						

A字段

15	14	13	$\bar{N}_i\bar{O}_n$
0	0	0	
0	0	1	LDRi
0	1	0	LDDR1
0	1	1	LDDR2
1	0	0	LDIR
1	0	1	LOAD
1	1	0	LDAR

B字段

12	11	10	$\bar{N}_i\bar{O}_n$
0	0	0	
0	0	1	RS-B
0	1	0	RD-B
0	1	1	RI-B
1	0	0	299-B
1	0	1	ALU-B
1	1	0	PC-B

C字段

9	8	7	$\bar{N}_i\bar{O}_n$
0	0	0	
0	0	1	P \bar{E} 1 \bar{E} ⊙
0	1	0	P \bar{E} 2 \bar{E} ⊙
0	1	1	P \bar{E} 3 \bar{E} ⊙
1	0	0	P \bar{E} 4 \bar{E} ⊙
1	0	1	AR
1	1	0	LDPC

其中 UA5~UA0 为 6 位的后续微地址, A、B、C 为三个译码字段, 分别由三个控制位译码出多位。C 字段中的 P (1) ~P (4) 是四个测试字位。其功能是根据机器指令及相应微代码进行译码, 使微程序转入相应的微地址入口, 从而实现微程序的顺序、分支、循环运行, 其原理如图 4.4-3 所示, 图中 I7~I2 为指令寄存器的第 7~2 位输出, SE5~SE1 为微控制器单元微地址锁存器的强置端输出。AR 为算术运算是否影响进位及判零标志控制位, 其为零有效。B 字段中的 RS-B、R0-B、RI-B 分别为源寄存器选通信号、目的寄存器选通信号及变址寄存器选通信号, 其功能是根据机器指令来进行三个工作寄存器 R0、R1 及 R2 的选通译码, 其原理如图 4.4-4, 图中 I0~I4 为指令寄存器的第 0~4 位, LDRi 为打入工作寄存器信号的译码器使能控制位。

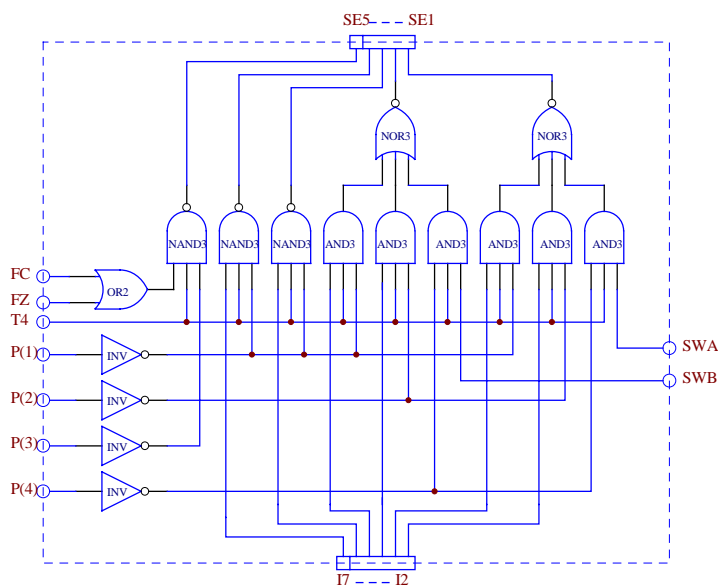


图 4.4-3 指令译码

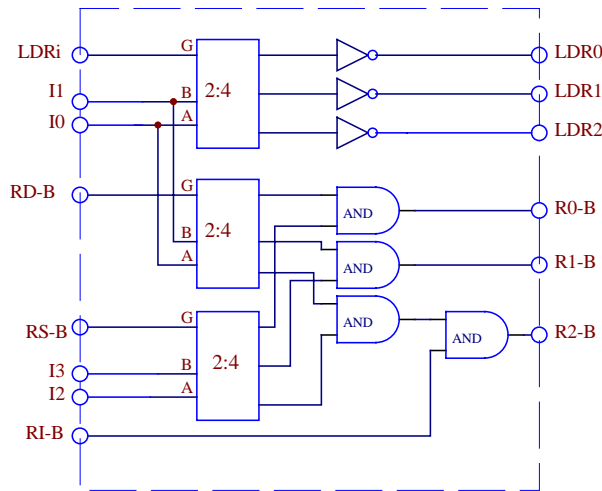


图 4.4-4 寄存器译码

四. 实验步骤

(1) 图 4.4-5 为所设计的几条机器指令对应的参考微程序流程图，将全部微程序按微指令格式变成二进制代码，可得到表 4.4-2 的二进制代码表。

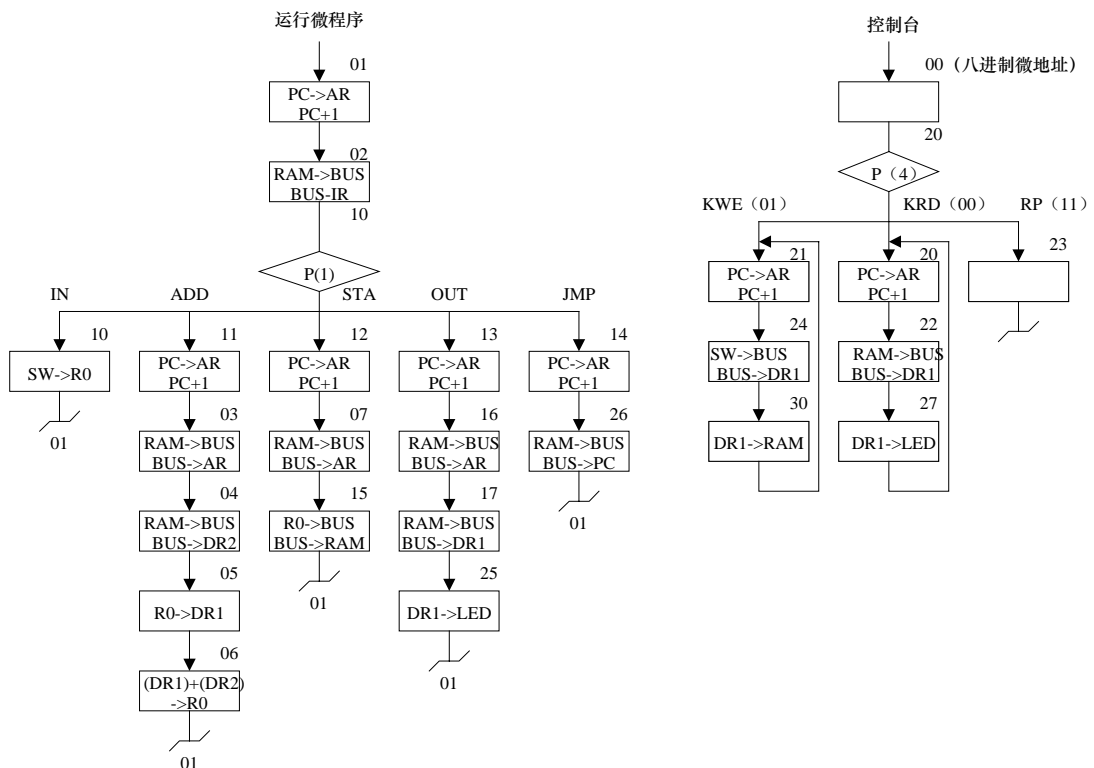


图 4.4-5 微程序流程图

上图中一个矩形方框表示一条微指令，方框中的内容为该指令执行的微操作，右上角的数字是该条指令的微地址，为表示方便，所有微地址是用8进制表示。向下的箭头指出了下一条要执行的指令。P（1）、P（4）为测试字，根据条件使微程序产生分支。

表 4.4-2 二进制代码表

微地址	S3 S2 S1 S0 M CN WE A9 A8	A	B	C	$\mu A5 \cdots \mu A0$
0 0	0 0 0 0 0 0 0 1 1	000	000	100	010000
0 1	0 0 0 0 0 0 0 1 1	110	110	110	000010
0 2	0 0 0 0 0 0 0 0 1	100	000	001	001000
0 3	0 0 0 0 0 0 0 0 1	110	000	000	000100
0 4	0 0 0 0 0 0 0 0 1	011	000	000	000101
0 5	0 0 0 0 0 0 0 1 1	010	001	000	000110
0 6	1 0 0 1 0 1 0 1 1	001	101	000	000001
0 7	0 0 0 0 0 0 0 0 1	110	000	000	001101
1 0	0 0 0 0 0 0 0 0 0	001	000	000	000001
1 1	0 0 0 0 0 0 0 1 1	110	110	110	000011
1 2	0 0 0 0 0 0 0 1 1	110	110	110	000111
1 3	0 0 0 0 0 0 0 1 1	110	110	110	001110
1 4	0 0 0 0 0 0 0 1 1	110	110	110	010110
1 5	0 0 0 0 0 0 1 0 1	000	001	000	000001
1 6	0 0 0 0 0 0 0 0 1	110	000	000	001111
1 7	0 0 0 0 0 0 0 0 1	010	000	000	010101
2 0	0 0 0 0 0 0 0 1 1	110	110	110	010010
2 1	0 0 0 0 0 0 0 1 1	110	110	110	010100
2 2	0 0 0 0 0 0 0 0 1	010	000	000	010111
2 3	0 0 0 0 0 0 0 1 1	000	000	000	000001
2 4	0 0 0 0 0 0 0 0 0	010	000	000	011000
2 5	0 0 0 0 0 1 1 1 0	000	101	000	000001
2 6	0 0 0 0 0 0 0 0 1	101	000	110	000001
2 7	0 0 0 0 0 1 1 1 0	000	101	000	010000
3 0	0 0 0 0 0 1 1 0 1	000	101	000	010001

(2) 按图 4.4-6 连接实验线路，仔细查线无误后接通电源。

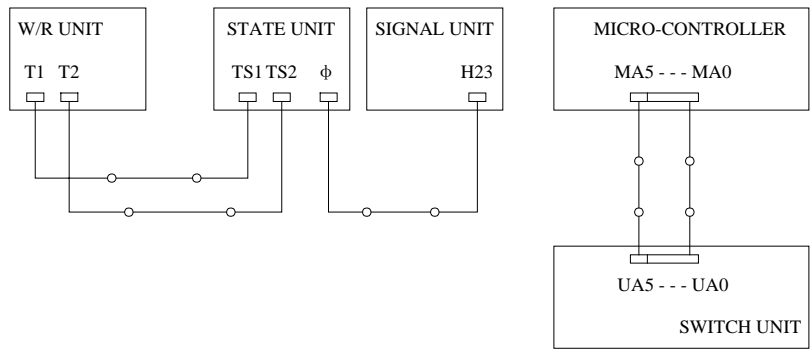


图 4.4-6 实验接线图

(3) 观测时序信号

用双踪示波器（或用联机软件的 PC 示波器功能）观察方波信号源的输出端 H23，调节电位器 W1，使输出波形的频率最慢。将时序电路中的“STOP”开关置为“RUN”，“STEP”开关置为“EXEC”。按动 START 按键，测量 TS1、TS2、TS3、TS4 各点的波形，比较它们的相互关系，画出其波形，并标注测量所得的脉冲宽度，见图 4.4-7。

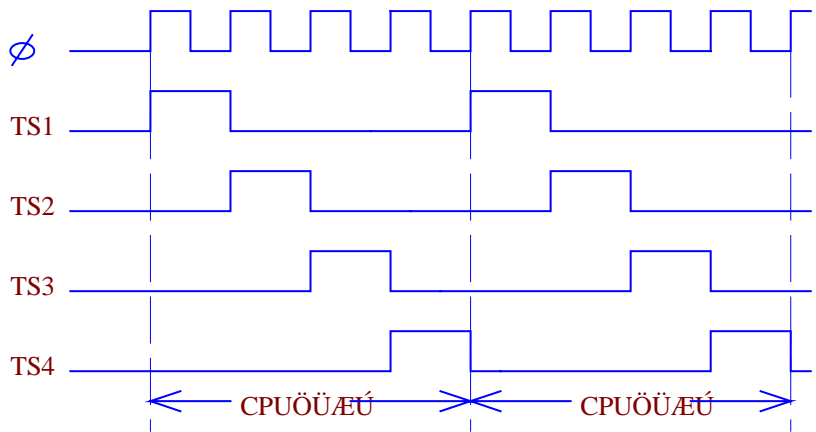


图 4.4-7

(4) 观察微程序控制器的工作原理

① 编程

- A. 将编程开关置为 PROM（编程）状态。
- B. 将实验板上“STATE UNIT”中的“STEP”置为“STEP”，“STOP”置为“RUN”状态。
- C. 用二进制模拟开关置微地址 MA5 — MA0。
- D. 在 MK24—MK1 开关上置微代码，24 位开关对应 24 位显示灯，开关量为“0”时灯亮，开关量为“1”时灯灭。

E. 启动时序电路（按动启动按钮“START”），即将微代码写入到 E²PROM 2816 的相应地址对应的单元中。

F. 重复 C—E 步骤，将表 4-1-2 的微代码写入 2816。

② 校验

A. 将编程开关设置为 READ（校验）状态。

B. 将实验板的“STEP”开关置为“STEP”状态。“STOP”开关置为“RUN”状态。

C. 用二进制开关置好微地址 MA5—MA0。

D. 按动“START”键，启动时序电路，读出微代码。观察显示灯 MD24—MD1 的状态（灯亮为“0”，灭为“1”），检查读出的微代码是否与写入的相同。如果不同，则将开关置于 PROM 编程状态，重新执行①即可。

③ 单步运行

A. 将编程开关置于“RUN（运行）”状态。

B. 实验板的“STEP”及“STOP”开关保持原状。

C. 操作 CLR 开关（拨动开关在实验板右下角）使 CLR 信号 1—>0—>1，微地址寄存器 MA5—MA0 清零，从而明确本机的运行入口微地址为 000000（二进制）。

D. 按动“START”键，启动时序电路，则每按动一次启动键，读出一条微指令后停机，此时实验台上的微地址显示灯和微命令显示灯将显示所读出的一条指令。

注意：在当前条件下，可将“MICRO-CONTROLLER”单元的 SE6—SE1 接至“SWITCH UNIT”中的 S3—Cn 对应二进制开关上，可通过强置端 SE1—SE6 人为设置分支地址。首先将 SE1—SE6 对应二进制开关置为“1”，当需要人为设置分支地址时，将某个或几个二进制开关置“0”，相应的微地址位即被强置为“1”，从而改变下一条微指令的地址。（二进制开关置为“0”，相应的微地址位将被强置为“1”）

④ 连续运行

A. 将编程开关置为“RUN（运行）”状态。

B. 将实验板的单步开关“STEP”置为“EXEC”状态。

C. 使 CLR 从 1—>0—>1，此时微地址寄存器清“0”，从而给出取指微指令的入口地址为 000000（二进制）。

D. 启动时序电路，则可连续读出微指令。

4.5 硬布线控制器

4.5.1 硬布线控制器的基本原理及结构

硬布线控制器本质上就是一个组合电路，它将输入逻辑信号转换成一组输出逻辑信号，即控制信号。它是根据指令系统的操作时间表用组合逻辑线路形成的微命令序列。

硬布线控制器的输入信号有：指令译码器的输出、时序信号和运算结果标志状态信号等。输出的就是所有各部件需要的各种微操作信号。硬布线控制器结构如图 4.5-1 所示。

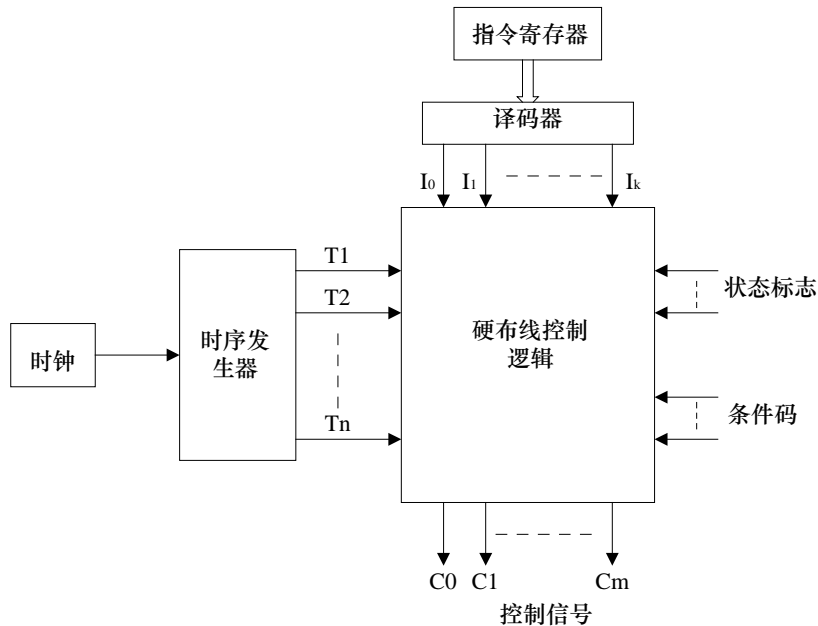


图 4.5-1 硬布线控制器

硬布线控制器的设计思想是：在硬布线控制器中，操作控制器发出的各种控制信号是时间因素和空间因素的函数。各个操作定时的控制构成了操作控制信号的时间特征，而各种不同部件的操作所需要的不同操作信号则构成了操作控制信号的空间特征。硬布线控制器就是把时间信号和操作信号组合，产生具有定时特点的控制信号。

4.5.2 硬布线控制器设计步骤

硬布线控制器的设计步骤：

1. 根据给定数据通路和指令功能排列出每条指令的操作控制步骤。
2. 确定机器的状态周期、节拍与工作脉冲。
3. 根据指令功能和 CPU 的结构，绘制每条机器指令的操作流程。
4. 根据操作流程及时序关系安排每条微指令的时间表。
5. 根据机器指令的流程图，找出产生同一个微操作信号的所有条件，结合时序关系，列出每个操作控制信号的表达式。
6. 根据各控制信号的表达式，在门阵列器件中编程，生成硬布线控制逻辑。

可以看出，因为硬布线控制器的操作控制信号是由硬件逻辑直接产生的，不需要象微程序控制器一样需要读取控存，所以它的执行速度较快。但硬布线控制逻辑的核心部分比较烦琐、设计效率较低，检查调试也比较困难。还有就是这种方式不易修改或扩展。因为在每个逻辑表达式中包含着好多的条件，一旦改变一些逻辑变量，其他表达式有可能都会发生改变。

4.6 硬布线控制器实验

一. 实验目的

1. 掌握硬布线控制器的组成原理、设计方法。
2. 了解硬布线控制器和微程序控制器的各自优缺点。

二. 实验设备

1. TDN-CM+或 TDN-CM++教学实验系统一台。
2. PC 微机一台。

三. 实验原理

本实验设计一个简单的硬布线控制器，用开关置不同的指令，触发时序，就可以实现不同的指令操作。实验所设计的三条指令如下表 4.6-1。

表 4.6-1

指令码	操作	说明
0 0	INPUT DEVICE → DR1	将数据开关中的数打入到工作暂存器 DR1 中，并由 OUTPUT DEVICE 中的数码管来显示。
0 1	INPUT DEVICE → DR2	将数据开关中的数打入到工作暂存器 DR2 中，并由 OUTPUT DEVICE 中的数码管来显示。
1 0	DR1 + DR2 → OUTPUT DEVICE	将运算器中 DR1 加 DR2 的运算结果打入输出单元 OUTPUT DEVICE 中数码管来显示。

硬布线控制器设计

用 CPLD 设计此控制逻辑的功能描述，各控制信号在 ispLSI1032 中对应的管脚见图 4.6-1。

现在有三个部件需要控制：ALU、INPUT DEVICE、OUTPUT DEVICE。它们的控制信号由指令码 I1、I0 和时序信号 T4、T3、T2、T1 组合产生。

LDDR1	LDDR2	WR	ALU_B	SW_B	LED_B
11	12	18	26	33	34
ISPLSI1032					
51	52	10	9	4	3
83	82	81	80	79	78
I1	I0	T1	T2	T3	T4
S3	S2	S1	S0	M	CN

图 4.6-1 硬布线控制逻辑对应 1032 中的管脚

四. 实验步骤

- 1) 编译上述设计文件，并将生成的 JEDEC 文件下载至 CPLD 芯片 1032 中。
- 2) 按图 4.4-2 接线。
- 3) 将 SIGNAL UNIT 单元中的 STEP 开关置为 STEP 状态。
- 4) 将 SWICHTH UNIT 单元中的对应的指令输入开关 ALU_B,299_B 置为 0、0，给 INPUT DEVICE 单元置第一个数。按动 SIGNAL UNIT 单元中的 START 按键，触发时序，完成 INPUT DEVICE -> DR1 的操作并在数码管显示结果。
- 5) 将 SWICHTH UNIT 单元中的对应的指令输入开关 ALU_B,299_B 置为 0、1，给 INPUT DEVICE 单元置第二个数。按动 SIGNAL UNIT 单元中的 START 按键，触发时序，完成 INPUT DEVICE -> DR2 的操作并在数码管显示结果。
- 6) 将 SWICHTH UNIT 单元中的对应的指令输入开关 ALU_B,299_B 置为 1、0，按动 SIGNAL UNIT 单元中的 START 按键，触发时序，完成 ALU -> OUTPUT DEVICE 的操作。数码管上将显示两数相加的结果。

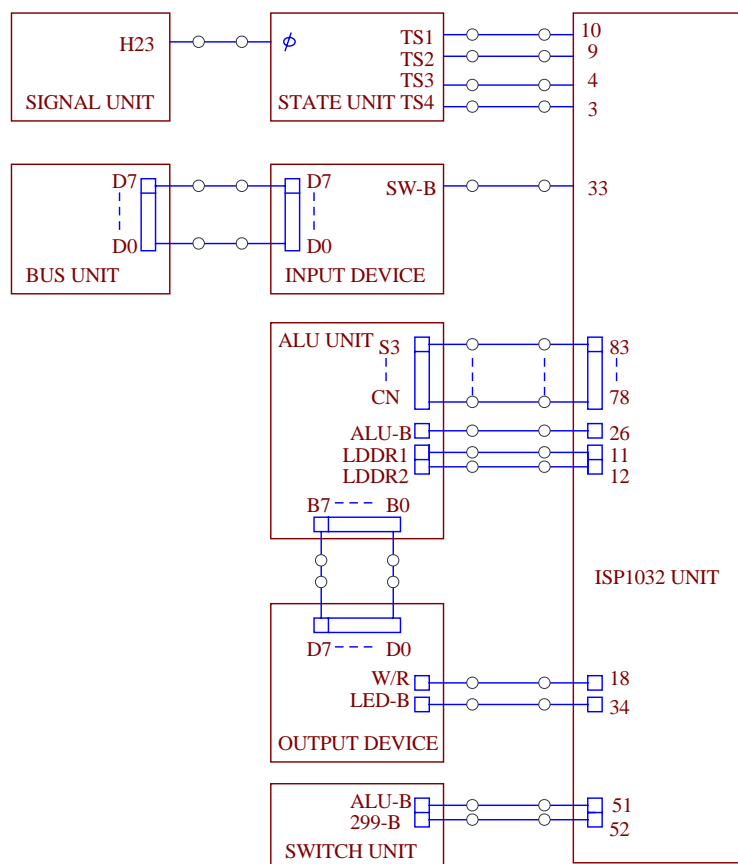


图 4.6-2 硬布线控制实验接线图

第五章 系统总线

总线是计算机中连接各个功能部件的纽带，是计算机各部件之间进行信息传输的公共通路。总线结构是决定计算机性能、功能、可扩展性和标准化程度的重要因素。

本章主要介绍了总线的分类、连接方式、通信方式和总线仲裁机构。安排了两个实验：总线基本实验和总线控制实验。

5.1 总线的概念及分类

5.1.1 总线的基本概念

总线是一组能为多个部件共享的公共信息传送线路，可以分时地发送和接收各部件的信息。计算机是由许多不同的系统部件组成的，这些部件之间的相互通信，就可以采用总线结构来实现。所以说总线不只是一组简单的信号传输线，它还是一组协议。分时与共享是总线的两大特征。所谓共享，在总线上可以挂接多个部件，它们都可以使用这一信息通路来和其他部件传送信息。所谓分时，同一总线在同一时刻，只能有一个部件占领总线发送信息，其他部件要发送信息得在该部件发送完释放总线后才能申请使用。但是同一时刻可以有多个部件接收信息。总线协议一般包括：信号线定义、数据格式、时序关系、信号电平、控制逻辑等，它确定了一个系统使用总线的方法。

5.1.2 总线的分类

从不同的角度对总线可以有不同的分类：

一．按总线在计算机系统中所处的地位：

1. CPU 内部总线。这是一组数据线，用来连接 CPU 内部的各个寄存器与算术逻辑部件。
2. 部件内总线。除了 CPU 内部总线外，在计算机各个部件、外设接口和外设控制器等能功能模块中也都采用总线传送信息。这一级是芯片间的总线。
3. 系统总线。用来连接计算机系统内部各大部件，如 CPU、主存、输入输出设备等。系统总线包含地址线、数据线及控制线。
4. 外总线。这是计算机系统之间或其他外部设备的连接总线。

二．按数据的传送格式：

1. 并行总线。有多根数据线，可并行地传送多个二进制位，一般为一个字节或多个字节，其位数称为该总线的数据通路的宽度。而单位时间的传送的数据量被定义为总线的数据传输率。

2. 串行总线。只有一根数据线，只能串行地逐位传送数据。

系统总线一般采用的是并行总线，其距离短，传送速度快。外总线较多的采用串行总线，

以节约成本，实现远距离传输，但是速度低。

三．按时序控制方式：

1．同步总线。同步总线进行数据传送时，有着严格的时钟周期定时，一般设置同步定时信号，如时钟同步、读/写信号等。

2．异步总线。在数据传输时，没有固定的时钟周期定时，而采用应答方式工作，操作时间根据需要可长可短。

同步总线控制较简单，应用于各部件间数据传送时间差异较小的场合，但时间的利用率可能不高。异步总线应用于各部件间数据传送时间差比较大的系统，可以根据需要调整时间的长短，时间利用率较高，但相应的控制较复杂。

四．按总线的传送方向：

1．单向总线。单向总线使挂在总线上的一些部件将信息有选择地传向另一些部件。

2．双向总线。双向总线可以使挂在总线上的任何设备都可以从总线上有选择地接收其他设备的信息，或通过总线有选择地将信息发送给其他设备。一般的方法是：设备的输出端以三态门和总线相连，当不使用总线时就让三态门呈现高阻抗。总线将信息发往各接收部件的输入端，用定时打入脉冲将信息打入指定的部件。

5.1.3 总线的连接方式

总线的排列布置方式与其他部件的连接方式对计算机系统的性能来说，将起着十分重要的作用。单机系统中采用的总线结构有三种基本类型：

一．单总线结构。

使用一条单一的总线来连接 CPU、内存及 I/O 设备叫单总线结构，如图 5.1-1 所示。

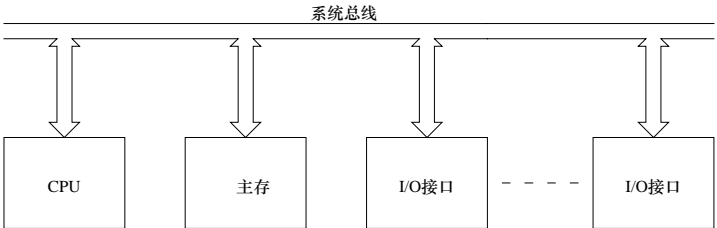


图 5.1-1 单总线结构

这种总线方式结构简单，但是所有的部件都挂在同一个总线上，因总线只能分时地工作，所以对信息的传送率受到限制。

二．双总线结构。

双总线结构如图 5.1-2 所示。这种结构的总线保持了单总线系统简单、易于扩充的优点，同时又在 CPU 和内存之间专门设置了一组高速的存储总线，使 CPU 可通过专用总线与存储器交换信息，并减轻了系统总线的负担，同时内存仍可通过系统总线与外设之间实现 DMA 操作，而不必经过 CPU。

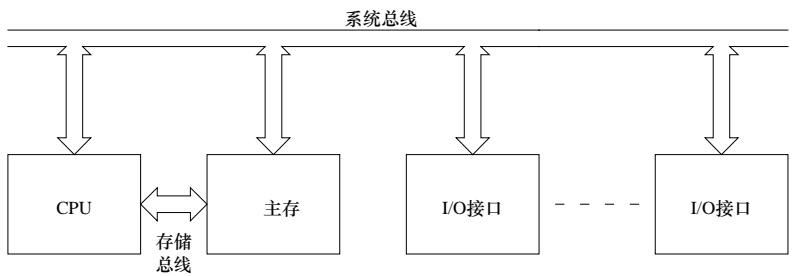


图 5.1-2 双总线结构

三. 三总线结构。

三总线结构图如图 5.1-3 所示。它是在双总线结构的基础上增加 I/O 总线形成的。I/O 总线采用一个 I/O 处理器来统一管理多个 I/O 接口及它们与系统总线的数据传送。这又使整个系统的效率大大提高。

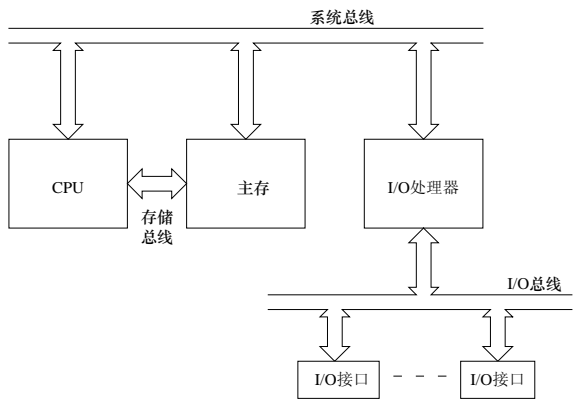


图 5.1-3 三总线结构

5. 2 总线的通信方式

总线的通信方式分为同步方式和异步方式两种。

1. 同步通讯

以同步通信方式进行通信时，总线上的部件都使用一个公共的时钟信号进行同步。这个公共时钟可以由 CPU 总线控制部件发送到每一个部件，可以部件有自己的时钟发生器，但是必须都要与总线控制器发出的时钟信号同步。

这种控制方式一般由时钟周期为划分时间段的基准。因为总线周期较时钟周期长，所以可以让一个总线周期占用多个 CPU 周期。同步通信适用于总线长度较短，各部件存取时间比较接近的情况。它的同步时钟也由存取速度最慢的部件来决定，如果存取时间相差很多，会大大降低总线传送效率的。

2. 异步通讯

异步通信方式的主要特征是没有统一的时钟同步信号，部件之间采用应答方式来实现总

线传送操作的，所需的时间根据需要而定。一般异步应答方式分为不互锁、半互锁及全互锁三类，如图 5.2-1 所示。

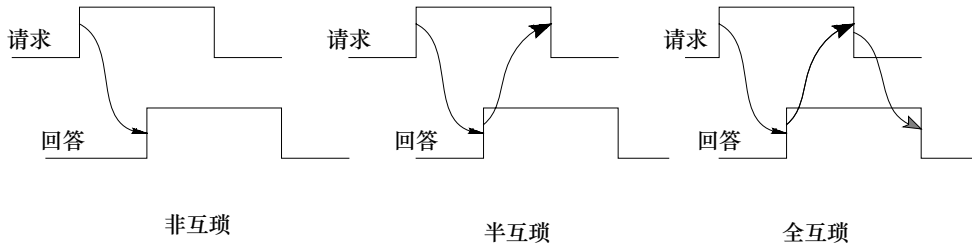


图 5.2-1 异步通讯的三种方式

不互锁方式是设备 1 发出请求信号后，经过一定时间关闭请求信号，而设备 2 在收到请求信号后发出回答信号，经过一定的时间后自动关闭回答信号。这种方式两设备发出的请求及回答信号不存在互锁关系，它们的结束都是由设备自身定时的。

半互锁方式是设备 1 在收到设备 2 的回答信号后才结束请求信号，但是设备 2 的回答信号是定时结束，由设备 2 本身定时决定。

全互锁方式是设备 1 收到设备 2 的回答信号后结束请求信号，而设备 2 在获知请求信号撤消后便撤消其回答信号。

采用互锁方式时间安排紧凑，但实现比较复杂。采用非互锁方式较易实现，可靠性高些，但由于其信号的维持时间必须满足最长操作的需要，所以对其他操作就会浪费时间，降低总线的操作速度。

5.3 总线仲裁

总线是多个部件共享的，为了有效地进行部件间的通信，必须要一个总线控制机构对总线进行合理的分配和管理，否则会造成争用局面毁坏系统。总线仲裁机构就是完成这个任务的部件。每个部件要使用总线，就得先向总线仲裁机构发出请求，而在同一时刻，也可能有多个部件发出请求，总线仲裁机构就会根据一定的原则、优先顺序来同意哪个部件可以使用总线。

总线仲裁机构可以分为集中式仲裁控制和分散式仲裁控制。总线控制逻辑集中于一个设备时称为集中式仲裁；而总线控制逻辑分散于各个部件中时称为分散式仲裁。集中式仲裁有 3 种方式：串行链式查询方式、计数器定时查询方式及独立请求方式。

1. 串行链式查询方式

串行链式查询方式如图 5.3-1 所示。这种方式所用的设备使用同一根“总线请求”线，当一个或多个设备发出“总线请求”信号时，该请求线向总线仲裁控制器发出信号，总线控制器只有在“总线忙”信号无效时进行响应，发出“总线授予”信号，顺序经过各个部件，若该部件没有发出请求，则继续顺序查询下一个设备，若该设备有请求，则停止继续查询，并撤消该部件总线请求信号，发出“总线忙”信号，接着该部件可以控制总线进行数据传送。

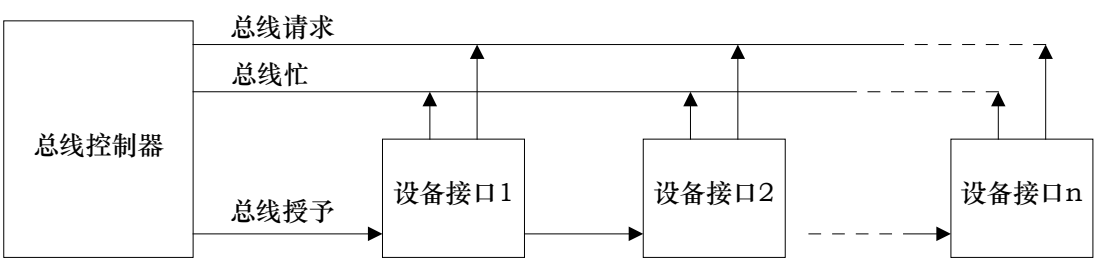


图 5.3-1 串行链式查询方式

可以看出，离总线控制器越近优先级就越高。这种方式结构简单，只用很少几根线就能实现总线请求判断控制，并易于扩充，但如果优先级高的设备出现故障，则影响后面设备的工作。

2. 计数器定时查询方式

计数器定时查询方式如图 5.3-2 所示。这种方式每个部件都有一个地址编号，从 0~n-1 连续编排。当总线控制器收到“总线请求”信号且“总线忙”信号无效时，使计数器开始计数，计数值通过地址线发向各设备接口，当地址线上的计数值与发出请求的设备的地址相同时，撤消该部件总线请求信号，发出“总线忙”信号，接着该部件可以控制总线进行数据传送。

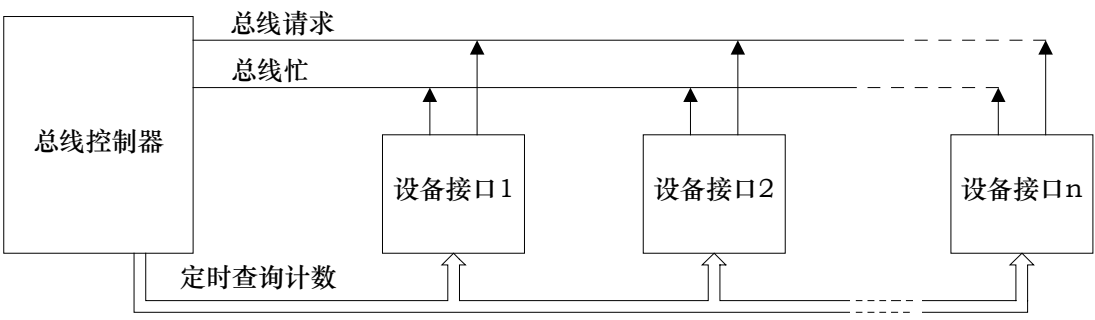


图 5.3-2 计数器定时查询方式

每次计数器可以从 0 开始计数，也可以从上次终止点继续计数，也可以由程序设定初值，可以由程序灵活的设置各设备的优先权。

3. 独立请求方式

独立请求方式如图 5.3-3。这种方式每个部件都有独立的“总线请求”线和“总线授予”线。当设备发出总线请求时，由总线控制器按照一定的优先顺序决定响应哪个设备的请求，发出“总线授予”信号。

这种方式响应时间最快，但是连线较多。

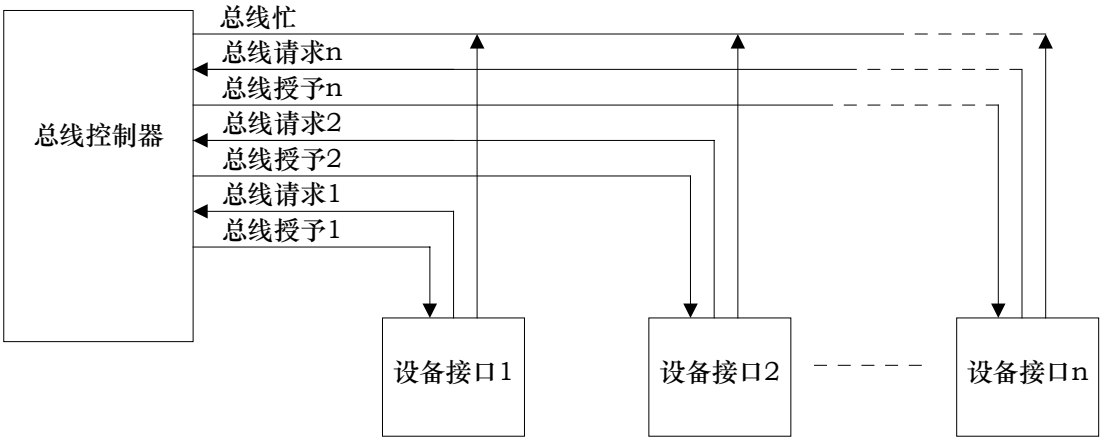


图 5.3-3 独立请求方式

5.4 总线基本实验

一. 实验目的

1. 理解总线的概念及其特性。
2. 掌握总线传输控制特性。

二. 实验设备

TDN-CM+或 TDN-CM++教学实验系统一台。

三. 实验原理

实验所用总线传输实验框图如图 5.4-1 所示，它将几种不同的设备挂至总线上，有存储器、输入设备、输出设备、寄存器。这些设备都需要有三态输出控制，按照传输要求恰当有序的控制它们，就可实验总线信息传输。

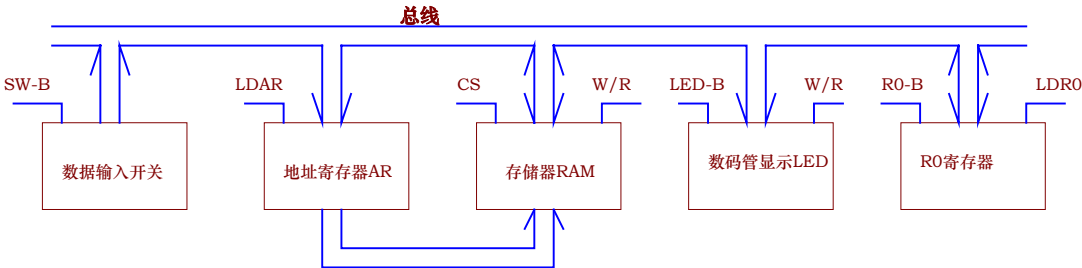


图 5.4-1 总线传输实验框图

实验要求

根据挂在总线上的几个基本部件，设计一个简单的流程：

- ① 输入设备将一个数打入 R0 寄存器。
- ② 输入设备将另一个数打入地址寄存器。
- ③ 将 R0 寄存器中的数写入到当前地址的存储器中。
- ④ 将当前地址的存储器中的数用 LED 数码管显示。

四. 实验步骤

- (1) 按照图 5.4-2 实验接线图进行连线。

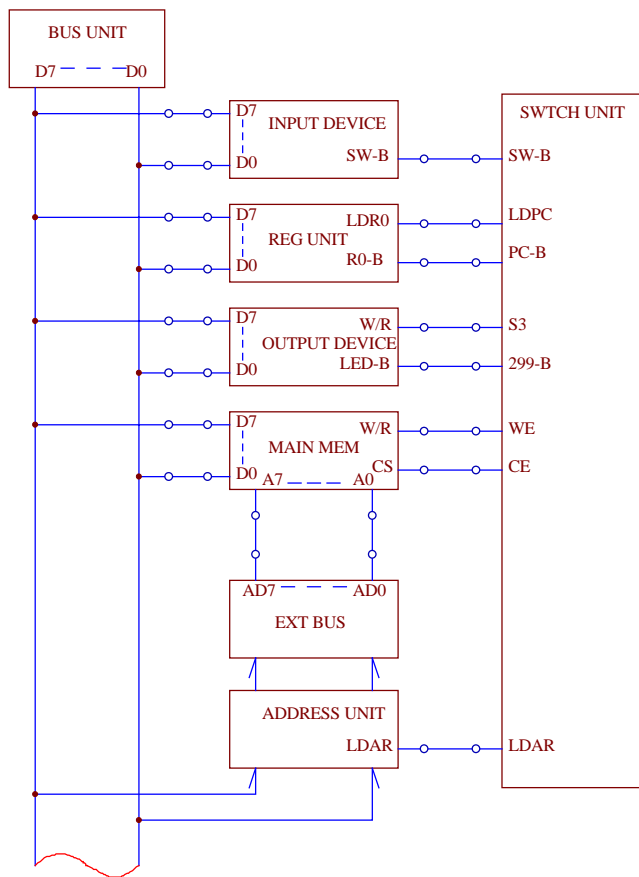
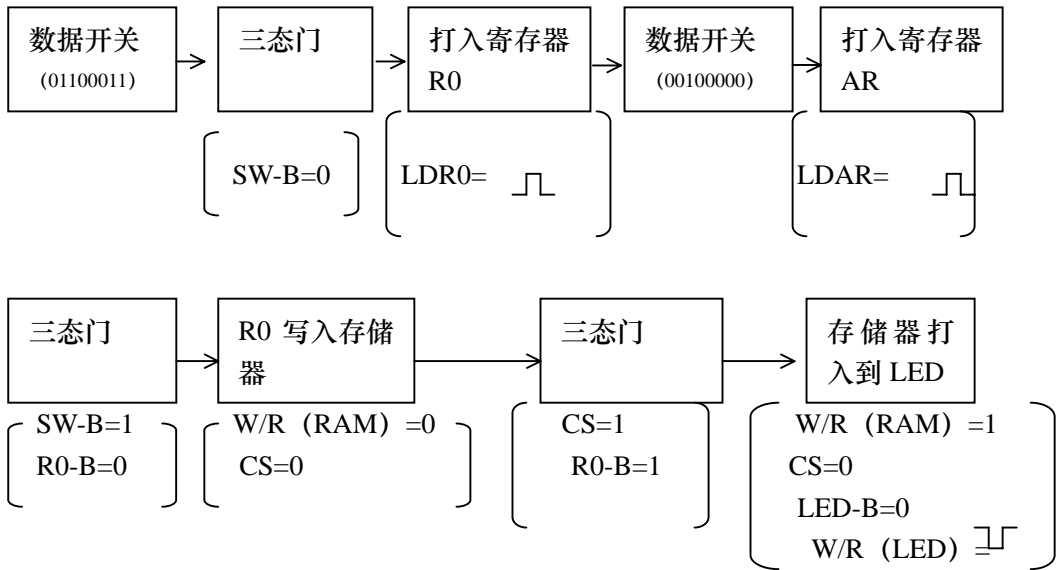


图 5.4-2 实验接线图

(2) 具体操作步骤图示如下:

首先应关闭所有三态门 (SW-B=1, CS=1, R0-B=1, LED-B=1), 并将关联的信号置为: LDAR=0, LDR0=0, W/R (RAM) =1, W/R (LED) =1。然后参照如下操作流程, 先给数据开关置数, 打开数据输出三态门, 拨动 LDR0 控制信号做 0→1→0 动作, 使产生一个上升沿将数据打入到 R0 中; 然后继续给数据开关置数, 拨动 LDAR 控制信号做 0→1→0 动作, 使产生一个上升沿将数据打入到 AR 中; 关闭数据开关三态门, 打开 R0 寄存器输出控制, 使存储器处于写状态 (W/R=0、CS=0) 将 R0 中的数写到存储器中; 关闭存储器片选, 关闭 R0 寄存器输出, 使存储器处于读状态 (W/R=1、CS=0), 打开 LED 片选, 拨动 LED 的 W/R 控制信号做 1→0→1 动作, 使产生一个上升沿将数据打入到 LED 中。



5.5 总线控制实验

一. 实验目的

掌握总线仲裁的方式及其方法。

二. 实验设备

1. TDN-CM+或 TDN-CM++教学实验系统一台。
2. PC 微机一台。

三. 实验内容

上节总线基本实验中, 关于总线的仲裁问题是由人为控制的, 本实验将设计一个控制逻辑, 来实现总线仲裁功能。实验将图 5.4-1 中控制输出部件的使能输入端接入控制逻辑, 然

后由控制逻辑输出至各对应的模块。其中的输出设备有 INPUT、RAM 及 R0，这里设其优先级依次降低，即 INPUT DEVICE 设备的优先级最高，当它输出有效时，即使给其他输出设备输入有效的输出信号也不能将数据输出至总线。其他设备依次类推。这样可以避免几个设备若同时输出数据至总线时的冲突，造成器件损坏。

实验规定总线控制逻辑在 CPLD1032 中定义的管脚如图 5.5-1。

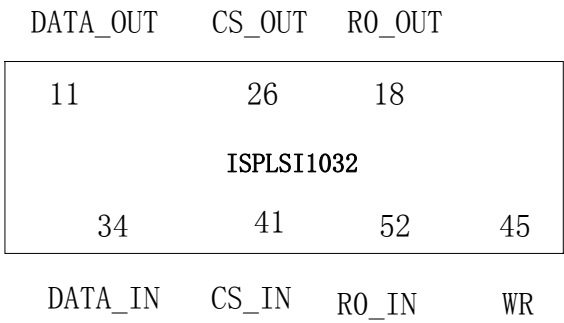


图 5.5-1

四．实验步骤

- 1. 用 ABEL 语言设计上述控制逻辑。
- 2. 在 ispDesignEXPERT 环境下编辑并编译上述所设计的源程序，并将生成的 JED 文件下载至 CPLD 中。
- 3. 按图 5.5-2 连接实验接线。
- 4. 具体实验操作步骤同上小节。分析两个实验在总线控制上的不同。

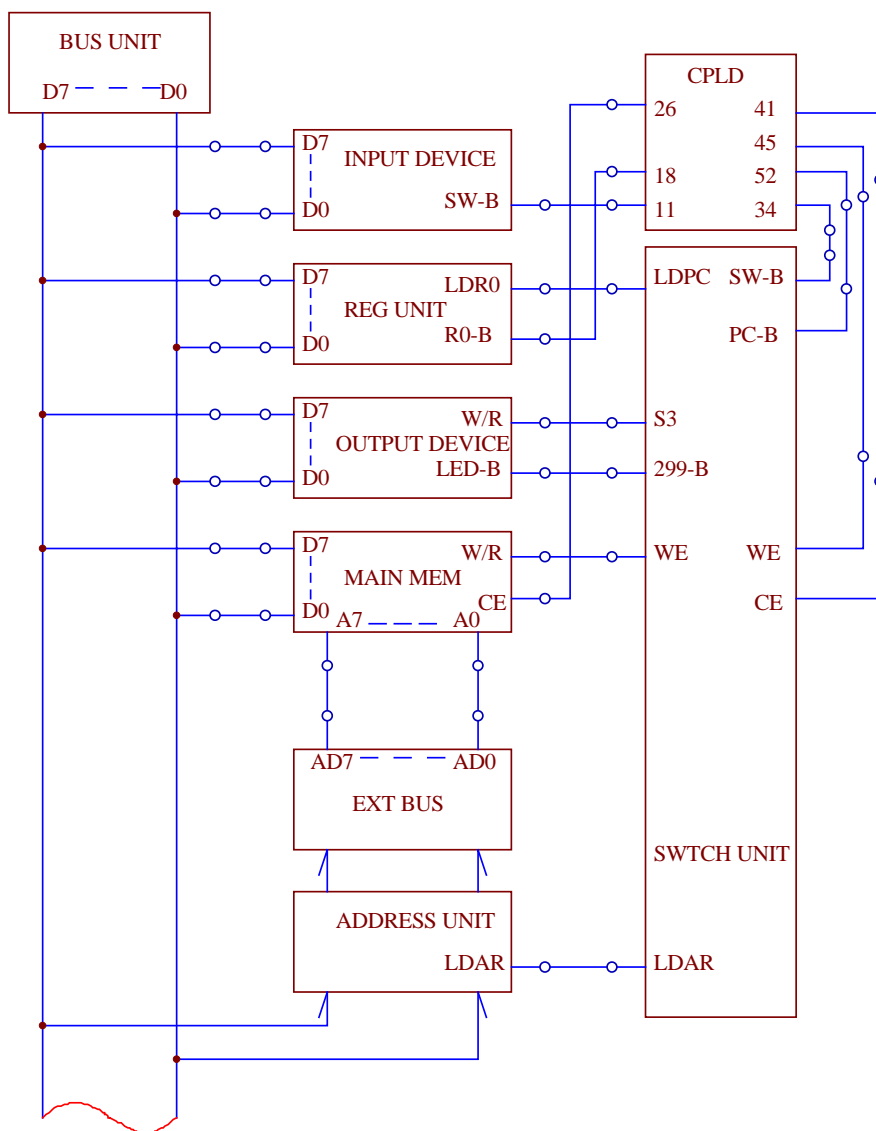


图 5.5-2 总线控制实验接线图

第六章 中央处理器

中央处理器（CPU）是计算机系统的核心组成部件，它通常包括控制器和运算器两大部分。CPU 硬件组织所能完成的基本功能是读取并执行指令。

前面我们已经对运算器和控制器分别进行了讨论，本章将中央处理器作为一个整体来讨论。介绍了中央处理器的组织、计算机指令系统、指令周期及寻址方式。通过一个基本模型计算机实验来介绍 CPU 一般的设计方法。

6.1 CPU 的基本组成结构

6.1.1 中央处理器的功能

计算机必须要有一个控制并执行指令的部件，该部件不仅要与计算机的其他功能部件进行信息交换，还有控制它们的操作，所以这个部件被称为中央处理器（CPU）。CPU 硬件组织所完成的基本功能是：读取并执行指令。对于整个计算机系统，它有如下几方面功能：

1. **指令控制** 控制指令按一定顺序执行。
2. **操作控制** 操作其他功能部件按指令要求进行动作。
3. **时间控制** 整个计算机系统程序的执行及各种操作实施都在严格的时间控制下有条不紊地自动工作。
4. **数据加工** 对数据进行各种运算。

6.1.2 中央处理器的组成

根据计算机对 CPU 的要求，它必须做的事情有：

取指令：读取某一主存单元的内容，并将其装入 CPU 的某个寄存器中。

解释指令：对指令进行译码，以确定所需要的动作。

取数据：一条指令的执行可能要求存储器或 I/O 模块读取数据。

处理数据：一条指令的执行可能要求对数据完成某些算术或逻辑运算。

写数据：执行的结果可能要求写数据到存储器或 I/O 模块。

显然，要完成这些事情，CPU 需要由运算器（ALU）、控制器及寄存器组成。如图 6.1-1 所示为一个简化的 CPU 的视图。

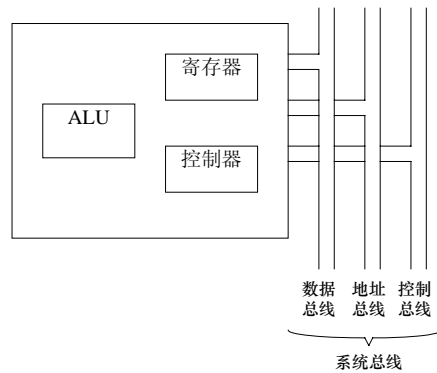


图 6.1-1 CPU 视图

ALU 完成数据的实际计算或处理。控制器控制数据和指令移入移出 CPU 并控制 ALU 的操作。寄存器为 CPU 暂存数据和指令。

6.1.3 寄存器组织

在 CPU 中有两类功能寄存器：用户可见寄存器和控制状态寄存器。

用户可见寄存器：指用户可以通过使用机器指令显式或隐式可访问的寄存器。这些允许机器语言或汇编语言的编程人员通过优化寄存器的使用而减少对主存的访问。它们可以是：

通用寄存器：可被程序员指派各种用途。

数据寄存器：仅可以用于保持数据而不能用于操作数地址的计算。

地址寄存器：用于保存地址，它可以有自身的某些通用性，也可以是某种具体的寻址方式，如段指针、变址寄存器或堆栈指针等。

条件代码寄存器：用于保持条件代码，CPU 硬件设置这些条件作为操作的结果。通常机器指令允许这些位以隐式读出，但不能被程序员修改。

控制和状态寄存器：用于被控制器用来控制 CPU 的操作并被特权的操作系统程序用来控制程序的执行。大多数是用户不可见的。它们有：

程序计数器（PC）：保存将要执行指令的地址。

指令寄存器（IR）：保存当前执行的指令。

存储地址寄存器（MAR）：保存 CPU 所访问存储器单元的地址。

存储缓冲寄存器（MBR）：保存从内存中读出的数据或是将要写入内存的数据。

程序状态字（PSW）：保存运算或测试结果的各种条件码内容及中断和系统工作状态等信息。

6.2 指令周期

CPU 从主存中取出一条指令到执行完这条指令的所有的操作所需要的时间通常叫做一个指令周期。由于指令从简单到复杂不尽相同，所以各种指令的指令周期也是不同的。见图 6.2-1 所示指令周期图。

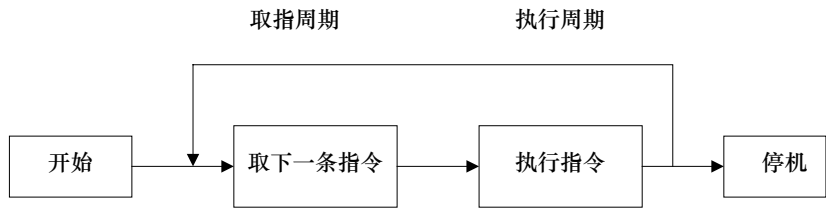


图 6.2-1 指令周期图

一个指令周期通常又由若干个 CPU 周期组成，CPU 周期也称为机器周期。而一个 CPU 周期又由若干个时钟周期组成，时钟周期是计算机中的最小处理单位。CPU 周期有定长不定长两种，即固定或是可变时钟周期数。图 6.2-2 是采用定长 CPU 周期的指令周期示意图。

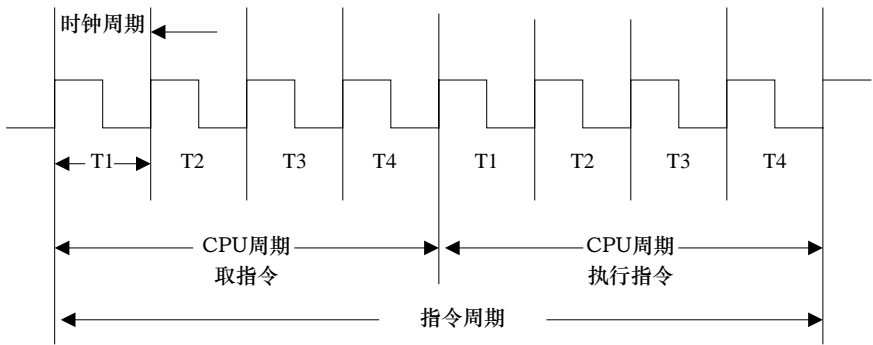


图 6.2-2 定长 CPU 周期的指令周期图

一个指令至少应包括一个取指和执行两个 CPU 周期，对于复杂的指令可能就需要更多的 CPU 周期。

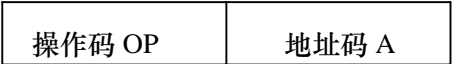
6.3 指令系统

计算机的工作，基本上体现为执行指令。一台计算机所能执行的全部指令，称为该计算机的指令系统。计算机的性能与它所设置的指令系统有很大的关系，它不仅与计算机的硬件结构密切相关，而且直接关系到用户的使用和编译程序的编制及运行效率。一个完善的指令系统应满足如下四个方面的要求：

- 1. 完备性 要求指令系统丰富、功能齐全完备、使用方便。
- 2. 有效性 利用该指令系统所编写的程序能高效率的运行，主要表现在程序占用存储空间小、执行速度快。
- 3. 规整性 包括指令系统的对称性、均齐性、指令格式和数据格式的一致性。
- 4. 兼容性 包括对不同机型的基本指令兼容和对同一系列机型的向上兼容性。

指令格式

指令由二进制代码表示的，为能够表示不同的要素，可以分成不同的字段。一般指令应表现两个方面的信息：一是指明操作的性质，即要求 CPU 做什么操作，这段代码称做操作码。二是给出操作有关的信息，指出该操作需要的数据的地址的编码，这段代码称做地址码。它可以表示操作数相关的地址，也可以表示操作数本身，也可以表示操作结果存放的地址等。所以一条指令通常可以表示为：



一. 操作码

一般来说，如果操作码采用固定长度 n 位的话，那么就可以表示 2^n 条指令。一条指令的操作码和地址码的二进制位数之和称为指令字。指令字越长，所能表示的指令就多，操作信息和地址信息也就越丰富。功能越强。但位数多会使从存储器读取指令时间增长，而且占用存储空间也越大。

如果计算机系统中所有指令字长是相等的，称为定长的指令格式。如果长度是变化的，称为变长的指令格式。定长的指令格式指令字结构简单，读取指令方便，但会造成存储空间浪费。

指令的操作码也有两种编码格式：一种是等长操作码，另一种是变长操作码。等长操作码可以简化硬件设计，减少指令译码时间。

二. 地址码

根据一条指令有几个操作数地址，可将该指令称为几地址指令。

三地址指令

格式：

OP	A1	A2	A3
----	----	----	----

其中，A1 为被操作数地址，A2 为操作数地址，A3 为存放操作结果的地址。指令功能是将 A1 和 A2 指定的操作数进行 OP 给定的某种操作运算，将结果存入 A3 指定的地址中。

二地址指令

格式：

OP	A1	A2
----	----	----

其中，A1 和 A2 指出两个参与运算的数的地址。指令功能是将 A1 和 A2 指定的操作数进行 OP 给定的某种操作运算，将结果存入 A2 指定的地址中。

一地址指令

格式：

OP	A
----	---

指令功能有两种情况：一是将 A 所指的内容进行 OP 给定操作运算，并将结果存入 A 中。另一种情况是将 A 所指的内容与累加器的内容进行 OP 给定的操作运算，并将结果存入累加器中。

零地址指令

格式：

OP

这种指令本身不需要任何操作码或所需的操作码是系统默认的。

6.4 寻址方式

根据指令字如何在主存中寻找指令和操作数或操作数地址，称为寻址方式。寻址方式分为指令寻址方式和操作数寻址方式。

6.4.1 指令寻址方式

计算机取指令的基本方式有两种，即顺序寻址方式和跳跃寻址方式

1. 顺序寻址

顺序寻址方式就是指令是顺序执行的，指令的地址由程序计数器 PC 给出。当每取出一条指令后，PC 会自动加 1，指向下一条将要执行的指令的地址。

2. 跳跃寻址

跳跃寻址就是指令不是顺序执行的，下一条指令的地址是根据当前指令执行结果或其他判断条件来确定的。这类指令根据判断修改 PC 的值，它或是在当前 PC 的基础上加减一个位移量，或是直接替换 PC 的值。

6.4.2 操作数的寻址方式

操作数的寻址就是寻找形成操作数在主存中的地址的方法。

设指令格式如下：

OP	寻址特征	形式地址
----	------	------

形式地址 (D)：指令地址字段中给出的地址。

有效地址 (EA)：形式地址经过一定计算而得到的操作数的实际地址。

常用寻址方式如下：

1. 隐含寻址：指令中不指出操作数的地址，而是隐含在累加器或堆栈等，由它们给出操作数。

2. 立即寻址：指令的地址字段指出的不是操作数的地址，而是操作数本身。即数据 data = D。

3. 直接寻址：操作数的地址直接在指令中给出，即操作数的有效地址为 $EA = D$ 。

4. 间接寻址：指令的形成地址 D 在主存相应单元中的内容是操作数的地址，即操作数的有效地址为 $EA = (D)$ 。

5. 寄存器寻址：指令中给出的是寄存器号 R，操作数就是寄存器中的内容，即 data = (R)。

6. 寄存器间接寻址：指令中给出的是寄存器号 R，而操作数的地址就是寄存器中的内容，即 $EA = (R)$ 。

7. 相对寻址：操作数地址为程序计数器 PC 中的内容与指令中给出的地址偏移量 D 之和，即 $EA = (PC) + D$ 。位移量 D 通常以补码形式给出，可正可负。

8. 基址寻址：操作数地址为基址寄存器中的内容与指令中给出的地址偏移量 D 之和，即 $EA = (R)_{\text{基址}} + D$ 。

9. 变址寻址：操作数地址为变址寄存器中的内容与指令中给出的地址偏移量 D 之和，即 $EA = (R)_{\text{变址}} + D$ 。

6.5 基本模型机设计实验

一. 实验目的

1. 在掌握部件单元电路实验的基础上，进一步将其组成系统构造一台基本模型计算机。
2. 为其定义五条机器指令，并编写相应的微程序，具体上机调试掌握整机概念

二. 实验设备

1. TDN-CM+或 TDN-CM++教学实验系统一台。
2. PC 微机一台。

三. 实验原理

部件实验过程中，各部件单元的控制信号是人为模拟产生的，而本次实验将在微程序控制下自动产生各部件单元控制信号，实现特定指令的功能。这里，计算机数据通路的控制将由微程序控制器来完成，CPU 从内存中取出一条机器指令到指令执行结束的一个指令周期全部由微指令组成的序列来完成，即一条机器指令对应一段微程序。

本实验采用五条机器指令：IN（输入）、ADD（二进制加法）、STA（存数）、OUT（输出）、JMP（无条件转移），其指令格式如下（前 4 位为操作码）：

助记符	机器指令码	说 明
IN	0000 0000	“INPUT DEVICE” 中的开关状态 \rightarrow R0
ADD addr	0001 0000 $\times \times \times \times \times \times \times \times$	$R0 + [addr] \rightarrow R0$
STA addr	0010 0000 $\times \times \times \times \times \times \times \times$	$R0 \rightarrow [addr]$
OUT addr	0011 0000 $\times \times \times \times \times \times \times \times$	$[addr] \rightarrow \text{LED}$
JMP addr	0100 0000 $\times \times \times \times \times \times \times \times$	$addr \rightarrow PC$

其中 IN 为单字长（8 位），其余为双字长指令， $\times \times \times \times \times \times \times \times$ 为 addr 对应的二进制地址码。

为了向 RAM 中装入程序和数据，检查写入是否正确，并能启动程序执行，还必须设计三个控制台操作微程序。

存储器读操作（KRD）：拨动总清开关 CLR 后，控制台开关 SWB、SWA 为“0 0”时，按 START 微动开关，可对 RAM 连续手动读操作。

存储器写操作（KWE）：拨动总清开关 CLR 后，控制台开关 SWB、SWA 置为“0 1”时，按 START 微动开关可对 RAM 进行连续手动写入。

启动程序：拨动总清开关 CLR 后，控制台开关 SWB、SWA 置为“1 1”时，按 START 微动开关，即可转入到第 01 号“取址”微指令，启动程序运行。

上述三条控制台指令用两个开关 SWB、SWA 的状态来设置，其定义如下：

SWB	SWA	控制台指令
0	0	读内存 (KRD)
0	1	写内存 (KWE)
1	1	启动程序 (RP)

根据以上要求设计数据通路框图,如图 6.5-1。微代码定义如表 6.5-1 所示。

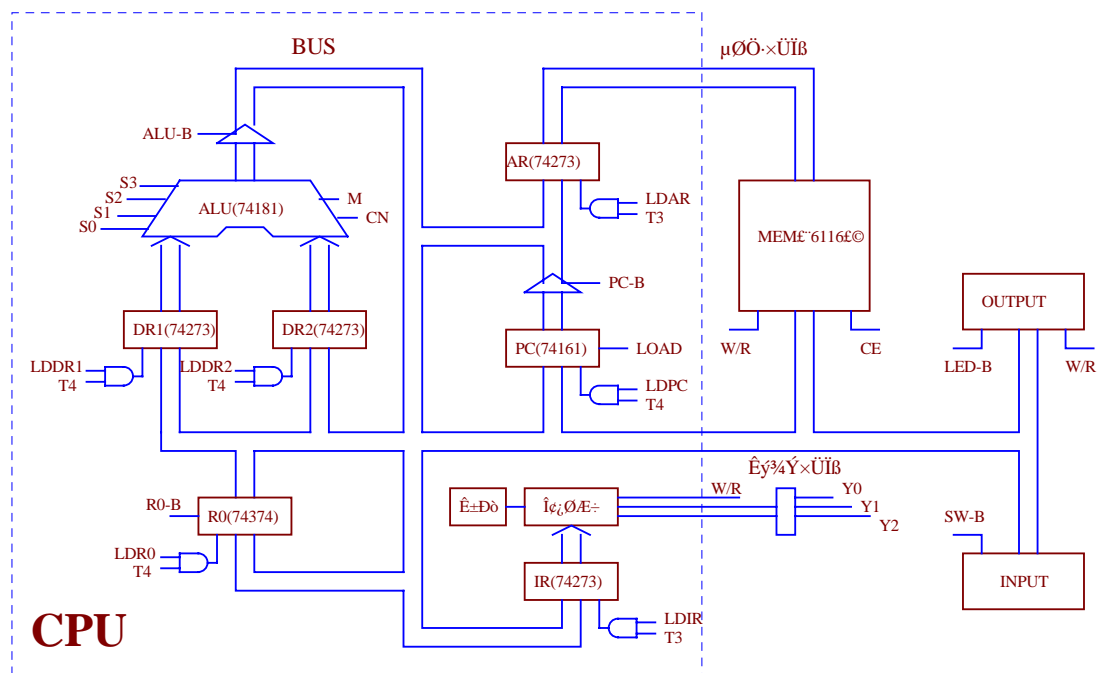


图 6.5-1 数据通路框图

系统涉及到的微程序流程见图 6.5-2, 这里“取指”是公用微指令, 为了能确定不同机器指令有各自不同的微程序转向, 我们在这里以指令寄存器的前 4 位 (IR7—IR4) 作为测试条件, 引入了 P (1) 指令测试字段, 如此, 对于 5 条机器指令, 就可以有 5 路 P (1) 测试分支, 对于每一指令分别予以微程序解释。

控制台操作为 P (4) 测试, 它以控制台开关 SWB、SWA 作为测试条件, 出现了 3 路分支, 占用 3 个固定微地址单元。当分支微地址单元固定后, 剩下的其它地方就可以一条微指令占用控存一个微地址单元随意填写。注意: 微程序流程图上的单元地址为 8 进制。

当全部微程序设计完毕后,应将每条微指令代码化,表 6.5-2 即为将图 6.5-2 的微程序流程图按微指令格式转化而成的“二进制微代码表”。

表 6.5-1

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
S3	S2	S1	S0	M	Cn	WE	A9	A8	A			B		C			uA5	uA4	uA3	uA2	uA1	uA0	

A 字段						B 字段						C 字段					
15	14	13	$\bar{N}_i \bar{O}_n$			12	11	10	$\bar{N}_i \bar{O}_n$			9	8	7	$\bar{N}_i \bar{O}_n$		
0	0	0				0	0	0				0	0	0			
0	0	1	LDRi			0	0	1	RS-B			0	0	1	P \bar{E} 1 \bar{E} 0		
0	1	0	LDDR1			0	1	0				0	1	0			
0	1	1	LDDR2			0	1	1				0	1	1			
1	0	0	LDIR			1	0	0				1	0	0	P \bar{E} 4 \bar{E} 0		
1	0	1	LOAD			1	0	1	ALU-B			1	0	1			
1	1	0	LDAR			1	1	0	PC-B			1	1	0	LDPC		

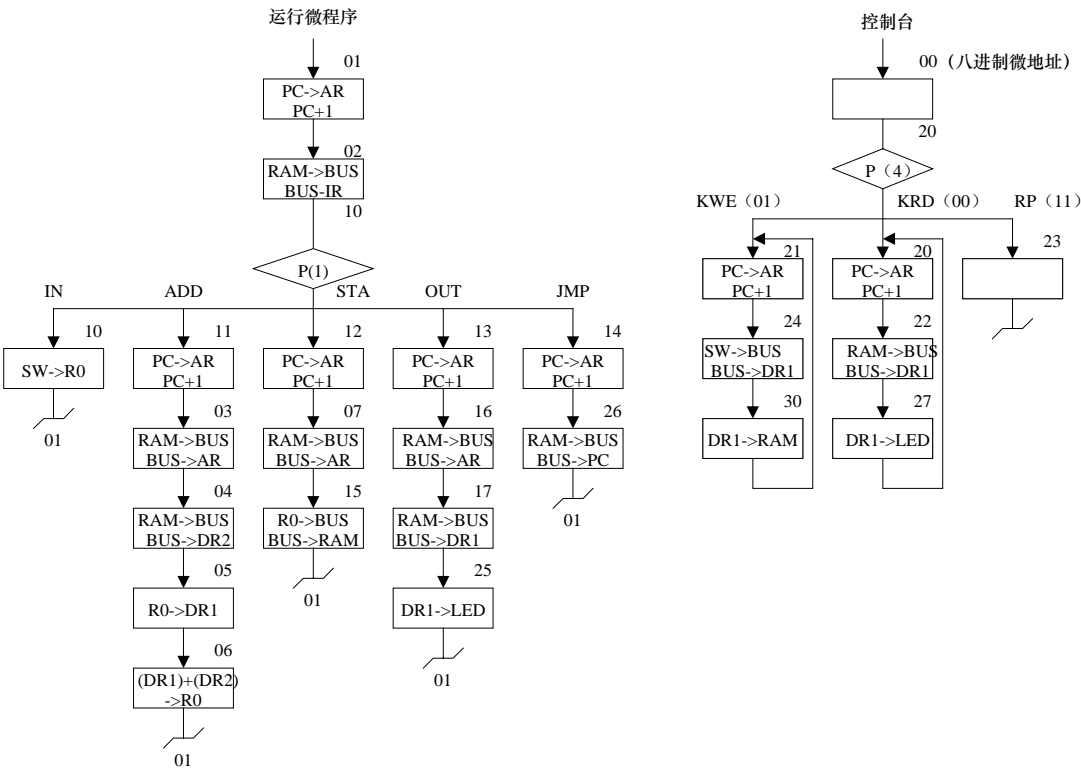


图 6.5-2 微程序流程图

表 6.5-2 二进制代码表

微地址	S3 S2 S1 S0 M CN WE A9 A8	A	B	C	μ A5... μ A0
0 0	0 0 0 0 0 0 0 1 1	000	000	100	010000
0 1	0 0 0 0 0 0 0 1 1	110	110	110	000010
0 2	0 0 0 0 0 0 0 0 1	100	000	001	001000
0 3	0 0 0 0 0 0 0 0 1	110	000	000	000100
0 4	0 0 0 0 0 0 0 0 1	011	000	000	000101
0 5	0 0 0 0 0 0 0 1 1	010	001	000	000110
0 6	1 0 0 1 0 1 0 1 1	001	101	000	000001
0 7	0 0 0 0 0 0 0 0 1	110	000	000	001101
1 0	0 0 0 0 0 0 0 0 0	001	000	000	000001
1 1	0 0 0 0 0 0 0 1 1	110	110	110	000011
1 2	0 0 0 0 0 0 0 1 1	110	110	110	000111
1 3	0 0 0 0 0 0 0 1 1	110	110	110	001110
1 4	0 0 0 0 0 0 0 1 1	110	110	110	010110
1 5	0 0 0 0 0 0 1 0 1	000	001	000	000001
1 6	0 0 0 0 0 0 0 0 1	110	000	000	001111
1 7	0 0 0 0 0 0 0 0 1	010	000	000	010101
2 0	0 0 0 0 0 0 0 1 1	110	110	110	010010
2 1	0 0 0 0 0 0 0 1 1	110	110	110	010100
2 2	0 0 0 0 0 0 0 0 1	010	000	000	010111
2 3	0 0 0 0 0 0 0 1 1	000	000	000	000001
2 4	0 0 0 0 0 0 0 0 0	010	000	000	011000
2 5	0 0 0 0 0 1 1 1 0	000	101	000	000001
2 6	0 0 0 0 0 0 0 0 1	101	000	110	000001
2 7	0 0 0 0 0 1 1 1 0	000	101	000	010000
3 0	0 0 0 0 0 1 1 0 1	000	101	000	010001

下面介绍指令寄存器 (IR): 指令寄存器用来保存当前正在执行的一条指令。当执行一条指令时, 先把它从内存取到指令寄存器中, 然后再对其进行译码、执行。指令划分为操作码和地址码字段, 由二进制数构成, 为了执行任何给定的指令, 必须对操作码进行测试 [P (1)], 通过节拍脉冲 T4 的控制以便识别所要求的操作。“指令译码器”(实验板上标有“INS DECODE”的芯片) 根据指令中的操作码译码强置微控器单元的微地址, 使下一条微指令指向相应的微程序首地址。

本系统有两种外部 I/O 设备, 一种是二进制代码开关, 它作为输入设备 (INPUT DEVICE); 另一种是数码块, 它作为输出设备 (OUTPUT DEVICE)。例如: 输入时, 二进制开关数据直接经过三态门送到总线上, 只要开关状态不变, 输入的信息也不变。输出时,

将输出数据送到数据总线上，当写信号（W/R）有效时，将数据打入输出锁存器，驱动数码块显示。

本实验设计机器指令程序如下：

地 址（二进制）	内 容（二进制）	助记符	说 明
0000 0000	0000 0000	IN R0	“INPUT DEVICE” → R0
0000 0001	0001 0000	ADD [0AH],R0	R0+[0AH] → R0
0000 0010	0000 1010		
0000 0011	0010 0000	STA R0,[0BH]	R0 → [0BH]
0000 0100	0000 1011		
0000 0101	0011 0000	OUT [0BH]	[0BH] → LED
0000 0110	0000 1011		
0000 0111	0100 0000	JMP 00H	00H → PC
0000 1000	0000 0000		
0000 1001			
0000 1010	0000 0001		自定
0000 1011			求和结果

四．实验步骤

(1) 按图 6.5-3 连接实验线路。

(2) 写程序

方法一：手动写入

- ① 先将机器指令对应的微代码正确地写入 2816 中，由于在实验三微程序控制实验中已将微代码写入 E²PROM 芯片中，对照表 6-2 校验正确后就可使用。
- ② 使用控制台 KWE 和 KRD 微程序进行机器指令程序的装入和检查。
 - A. 使编程开关处于“RUN”，STEP 为“STEP”状态，STOP 为“RUN”状态。
 - B. 拨动总清开关 CLR（1→0→1），微地址寄存器清零，程序计数器清零。然后使控制台 SWB、SWA 开关置为“0 1”，按动一次启动开关 START，微地址显示灯显示“010001”，再按动一次 START，微地址灯显示“010100”，此时数据开关的内容置为要写入的机器指令，按动两次 START 键后，即完成该条指令的写入。若仔细阅读 KWE 的流程，就不难发现，机器指令的首地址总清后为零，以后每个循环 PC 会自动加 1，所以，每次按动 START，只有在微地址灯显示“010100”时，才设置内容，直到所有机器指令写完。
 - C. 写完程序后须进行校验。拨动总清开关 CLR（1→0→1）后，微地址清零。PC 程序计数器清零，然后使控制台开关 SWB、SWA 为“0 0”，按动启动 START，微地址灯将显示“010000”，再按 START，微地址灯显示为“010010”，第三次按 START，微地址灯显示为“010111”，再按 START 后，此时输出单元的数码管显示为该首地址中的内容。不断按动 START，以后每个循环 PC 会自动加 1，可检查后续单元内容。每次在微地址灯显示为“010000”时，是将当前地址中的机器指令写入到输出

设备中显示。

方法二：联机读／写程序

按照规定格式，将机器指令及表 6.5-2 微指令二进制表编辑成十六进制的如下格式文件。微指令格式中的微指令代码为将表 6.5-2 中的 24 位微代码按从左到右分成 3 个 8 位，将此三个 8 位二进制代码化为相应的十六进制数即可。

程 序		机器指令格式说明：
\$P0000		\$P××××
\$P0110		└── 机器指令代码
\$P020A		└── 十六进制地址
\$P0320		
\$P040B		
\$P0530		
\$P060B		
\$P0740		
\$P0800		微指令格式说明：
\$P0A01		\$M××××××××
微程序		└── 微指令代码
		└── 十六进制地址
\$M00018110	\$M0D028201	
\$M0101ED82	\$M0E00E00F	
\$M0200C048	\$M0F00A015	
\$M0300E004	\$M1001ED92	
\$M0400B005	\$M1101ED94	
\$M0501A206	\$M1200A017	
\$M06959A01	\$M13018001	
\$M0700E00D	\$M14002018	
\$M08001001	\$M15070A01	
\$M0901ED83	\$M1600D181	
\$M0A01ED87	\$M17070A10	
\$M0B01ED8E	\$M18068A11	
\$M0C01ED96		

用联机软件的“【转储】—【装载】”功能将该格式文件装载入实验系统即可。

(3) 运行程序

方法一：本机运行

① 单步运行程序

- A. 使编程开关处于“RUN”状态，STEP 为“STEP”状态，STOP 为“RUN”状态。
- B. 拨动总清开关 CLR (1→0→1)，微地址清零，程序计数器清零。程序首址为 00H。
- C. 单步运行一条微指令，每按动一次 START 键，即单步运行一条微指令。对照微程序流程图，观察微地址显示灯是否和流程一致。

D. 当运行结束后,可检查存数单元(0BH)中的结果是否和理论值一致。

② 连续运行程序

A. “STATE UNIT”中的STEP开关置为“EXEC”状态。STOP开关置为“RUN”状态。

B. 拨动CLR开关,清微地址及程序计数器,然后按动START,系统连续运行程序,稍后将STOP拨至“STOP”时,系统停机。

C. 停机后,可检查存数单元(0BH)结果是否正确。

方法二:联机运行

联机运行程序时,进入软件界面,装载机器指令及微指令后,选择“【运行】—【通路图】—【复杂模型机】”功能菜单打开相应动态数据通路图,按相应功能键即可联机运行、监控、调试程序。(软件使用说明请看《用户手册》)

总清开关CLR清零(1→0→1)后,将使程序首址及微程序地址为00H,程序可从头开始运行。

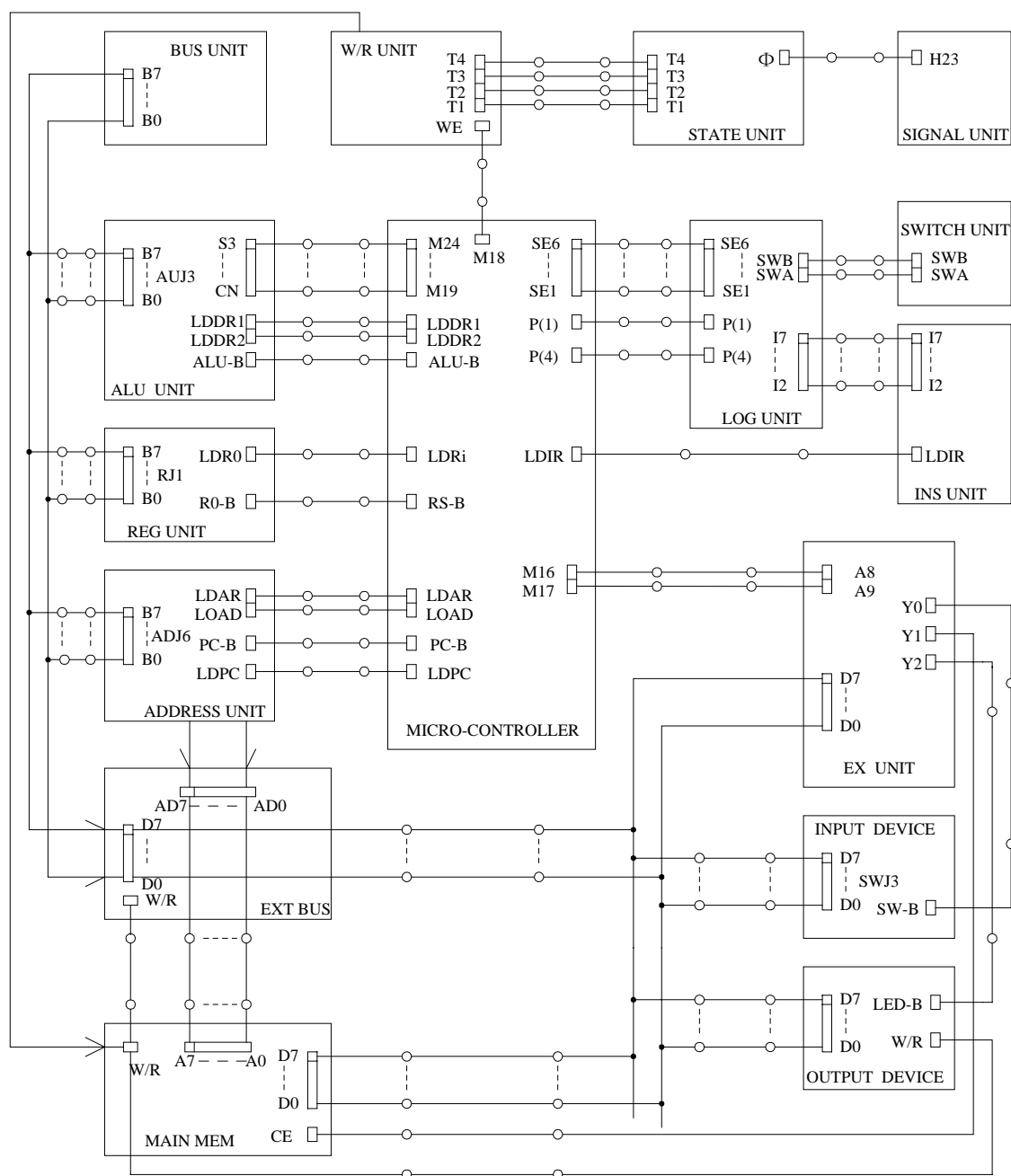


图 6.5-3 实验接线图

第七章 模型计算机及其设计

在前面的章节中，我们重点讨论计算机中每个部件的组成及特性。本章中，我们将重点讨论如何完整设计一台模型计算机，进一步建立整机的概念。我们先通过一个复杂模型机的设计实验来进行实际的计算机设计和实现，然后安排了用 CPLD 来实现一个 CPU 中的大部分功能的设计实验，接着讨论了输入输出系统的概念、分类、接口、寻址、基本控制方式等，并据此安排了三个实验：具有中断处理功能的模型机的设计实验、扩展 8255 并行口实验及扩展 8253 定时/记数器实验。

7.1 一台模型计算机的总体设计

设计一台完整的计算机，大致需按如下的顺序来考虑：

1. 确定设计目标

确定所设计计算机的功能和用途。

2. 确定指令系统

确定数据的表示格式、位数、指令的编码、类型、需要设计哪些指令及使用的寻址方式。

3. 总体结构与数据通路

总体结构设计包含确定各部件设置以及它们之间的数据通路结构。在此基础上，就可以拟出各种信息传送路径，以及实现这些传送所需要的微命令。

对于部件设置，比如要确定运算器部件采用什么结构，控制器采用微程序控制还是硬布线控制等。

综合考虑计算机的速率、性能价格比、可靠性等要求，设计合理的数据通路结构，采用何种方案的内总线及外总线。数据通路不同，执行指令所需要的操作就不同，计算机的结构也就不一样。

4. 设计指令执行流程

数据通路确定后，就可以设计指令系统中每条指令的执行流程。

根据指令的复杂程度，每条指令所需要的机器周期数。对于微程序控制的计算机，根据总线结构，需考虑哪些微操作可以安排在同一个微指令中，哪些微操作不能安排在同一条微指令中。

5. 确定微程序地址

根据后续微地址的形成方法，确定每条微程序地址及分支转移地址。

6. 根据微指令格式，将微程序流程中的所有微指令代码化，转化成相应的二进制代码，写入到控制存储器中的相应单元中。

7. 组装、调试。

在总调试前，先按功能模块进行组装和分调，因为只有各功能模块工作正常后，才能保证整机的运行正确。

当所有功能模块都调试正常后，进入总调试。连接所有模块，用单步微指令方式执行机器指令的微程序流程图，当全部微程序流程图检查完后，若运行结果正确，则在内存中装入一段机器指令，进行其他的运行方式等功能调试及执行指令的正确性验证。

7.2 复杂模型机设计实验

一. 实验目的

综合运用所学计算机原理知识，设计并实现较为完整的计算机。

二. 实验设备

- 1. TDN-CM+或 TDN-CM++教学实验系统一台。
- 2. PC 微机一台。

三. 数据格式及指令系统

1. 数据格式

模型机规定采用定点补码表示法表示数据，且字长为 8 位，其格式如下：

7	6 5 4 3 2 1 0
符号	尾 数

其中第 7 位为符号位，数值表示范围是： $-2^7 \leq X \leq 2^7 - 1$ 。

2. 指令格式

模型机设计四大类指令共十六条，其中包括算术逻辑指令、I/O 指令、访问及转移指令和停机指令。

(1) 算术逻辑指令

设计 9 条算术逻辑指令并用单字节表示，寻址方式采用寄存器直接寻址，其格式如下：

7 6 5 4	3 2	1 0
OP-CODE	rs	rd

其中，OP-CODE 为操作码，rs 为源寄存器，rd 为目的寄存器，并规定：

Rs 或 rd	选定的寄存器
00	R0
01	R1
10	R2

9 条算术逻辑指令的名称、功能和具体格式见表 7.2-1。

(2) 访问指令及转移指令

模型机设计 2 条访内指令，即存数 (STA)、取数 (LDA)，2 条转移指令，即无条件转移 (JMP)、结果为零或有进位转移指令 (BZC)，指令格式为：

7 6	5 4	3 2	1 0
00	M	OP -CODE	rd
D			

其中，OP-CODE 为操作码，rd 为目的寄存器地址 (LDA、STA 指令使用)。D 为位移量 (正负均可)，M 为寻址模式，其定义如下：

寻址模式 M	有效地址 E	说 明
00	$E = D$	直接寻址
01	$E = (D)$	间接寻址
10	$E = (RI) + D$	RI 变址寻址
11	$E = (PC) + D$	相对寻址

本模型机规定变址寄存器 RI 指定为寄存器 R2。

(3) I/O 指令

输入 (IN) 和输出 (OUT) 指令采用单字节指令，其格式如下：

7 6 5 4	3 2	1 0
OP-CODE	addr	rd

其中，addr=01 时，选中 “INPUT DEVICE” 中的开关组作为输入设备，addr=10 时，选中 “OUTPUT DEVICE” 中的数码块作为输出设备。

(4) 停机指令

指令格式如下：

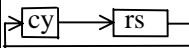
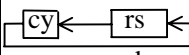
7 6 5 4	3 2	1 0
OP-CODE	00	00

HALT 指令，用于实现停机操作。

3. 指令系统

本模型机共有 16 条基本指令，其中算术逻辑指令 7 条，访问内存指令和程序控制指令 4 条，输入输出指令 2 条，其它指令 1 条。表 7.2-1 列出了各条指令的格式、汇编符号、指令功能。

表 7.2-1

助记符号	指令格式	功能								
CLR rd	<table><tr><td>0111</td><td>00</td><td>rd</td></tr></table>	0111	00	rd	$0 \rightarrow \text{rd}$					
0111	00	rd								
MOV rs, rd	<table><tr><td>1000</td><td>rs</td><td>rd</td></tr></table>	1000	rs	rd	$\text{rs} \rightarrow \text{rd}$					
1000	rs	rd								
ADC rs, rd	<table><tr><td>1001</td><td>rs</td><td>rd</td></tr></table>	1001	rs	rd	$\text{rs} + \text{rd} + \text{cy} \rightarrow \text{rd}$					
1001	rs	rd								
SBC rs, rd	<table><tr><td>1010</td><td>rs</td><td>rd</td></tr></table>	1010	rs	rd	$\text{rs} - \text{rd} - \text{cy} \rightarrow \text{rd}$					
1010	rs	rd								
INC rd	<table><tr><td>1011</td><td></td><td>rs</td></tr></table>	1011		rs	$\text{rd} + 1 \rightarrow \text{rd}$					
1011		rs								
AND rs, rd	<table><tr><td>1100</td><td></td><td>rs</td></tr></table>	1100		rs	$\text{rs} \wedge \text{rd} \rightarrow \text{rd}$					
1100		rs								
COM rd	<table><tr><td>1101</td><td></td><td>rs</td></tr></table>	1101		rs	$\overline{\text{rd}} \rightarrow \text{rd}$					
1101		rs								
RRC rs, rd	<table><tr><td>1110</td><td>rs</td><td>rd</td></tr></table>	1110	rs	rd	 $\text{rs} \rightarrow \text{rd}$					
1110	rs	rd								
RLC rs, rd	<table><tr><td>1111</td><td>rs</td><td>rd</td></tr></table>	1111	rs	rd	 $\text{rs} \rightarrow \text{rd}$					
1111	rs	rd								
LDA M, D, rd	<table><tr><td>00</td><td>M</td><td>00</td><td>rd</td></tr><tr><td colspan="4">D</td></tr></table>	00	M	00	rd	D				$E \rightarrow \text{rs}$
00	M	00	rd							
D										
STA M, D, rd	<table><tr><td>00</td><td>M</td><td>01</td><td>rd</td></tr><tr><td colspan="4">D</td></tr></table>	00	M	01	rd	D				$\text{rd} \rightarrow E$
00	M	01	rd							
D										
JMP M, D	<table><tr><td>00</td><td>M</td><td>10</td><td>00</td></tr><tr><td colspan="4">D</td></tr></table>	00	M	10	00	D				$E \rightarrow \text{PC}$
00	M	10	00							
D										
BZC M, D	<table><tr><td>00</td><td>M</td><td>11</td><td>00</td></tr><tr><td colspan="4">D</td></tr></table>	00	M	11	00	D				当 $\text{CY}=1$ 或 $\text{Z}=1$, $E \rightarrow \text{PC}$
00	M	11	00							
D										
IN addr, rd	<table><tr><td>0100</td><td>01</td><td>rd</td></tr></table>	0100	01	rd	$\text{addr} \rightarrow \text{rd}$					
0100	01	rd								
OUT addr, rd	<table><tr><td>0101</td><td>10</td><td>rd</td></tr></table>	0101	10	rd	$\text{rd} \rightarrow \text{addr}$					
0101	10	rd								
HALT	<table><tr><td>0110</td><td>00</td><td>00</td></tr></table>	0110	00	00	停机					
0110	00	00								

四．总体设计

本模型机的数据通路框图如图 7.2-1。根据机器指令系统要求，设计微程序流程图及确定微地址，如图 7.2-2。

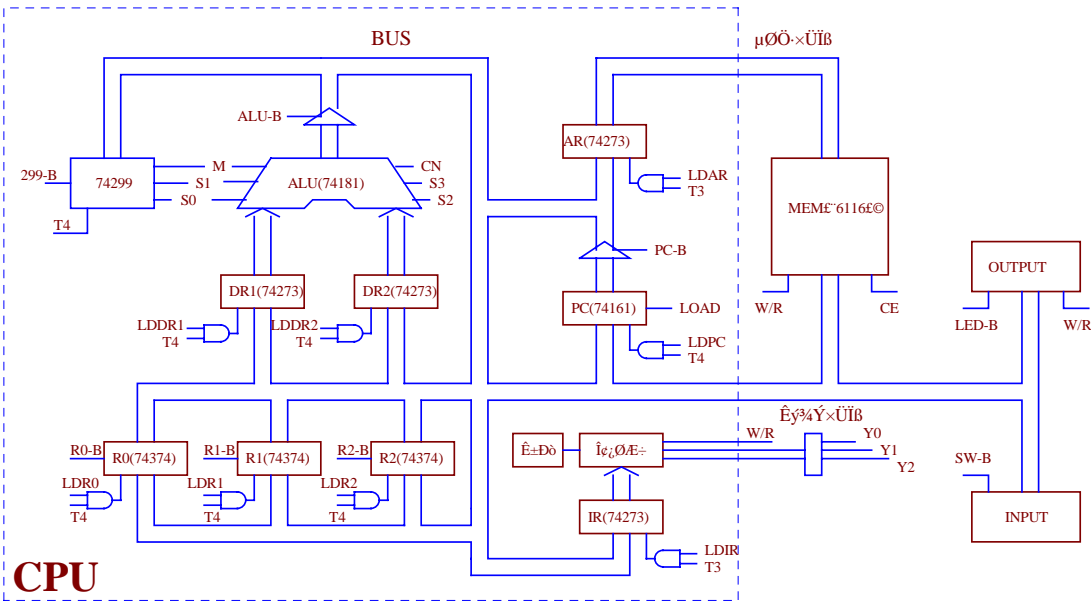


图 7.2-1 数据通路框图

按照系统建议的微指令格式，参照微指令流程图，将每条微指令代码化，译成二进制代码表，并将二进制代码表转换为联机操作时的十六进制格式文件，见表 7.2-2。

表 7.2-2

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
S3	S2	S1	S0	M	Cn	WE	A9	A8	A	B	C	uA5	uA4	uA3	uA2	uA1	uA0						

A 字段

B 字段

C 字段

15	14	13	$\tilde{N}_i \hat{O}_n$
0	0	0	
0	0	1	LDRi
0	1	0	LDDR1
0	1	1	LDDR2
1	0	0	LDIR
1	0	1	LOAD
1	1	0	LDAR

12	11	10	$\tilde{N}_i \hat{O}_n$
0	0	0	
0	0	1	RS-B
0	1	0	RD-B
0	1	1	RI-B
1	0	0	299-B
1	0	1	ALU-B
1	1	0	PC-B

9	8	7	$\tilde{N}_i \hat{O}_n$
0	0	0	
0	0	1	Pf'1f©
0	1	0	Pf'2f©
0	1	1	Pf'3f©
1	0	0	Pf'4f©
1	0	1	AR
1	1	0	LDPC

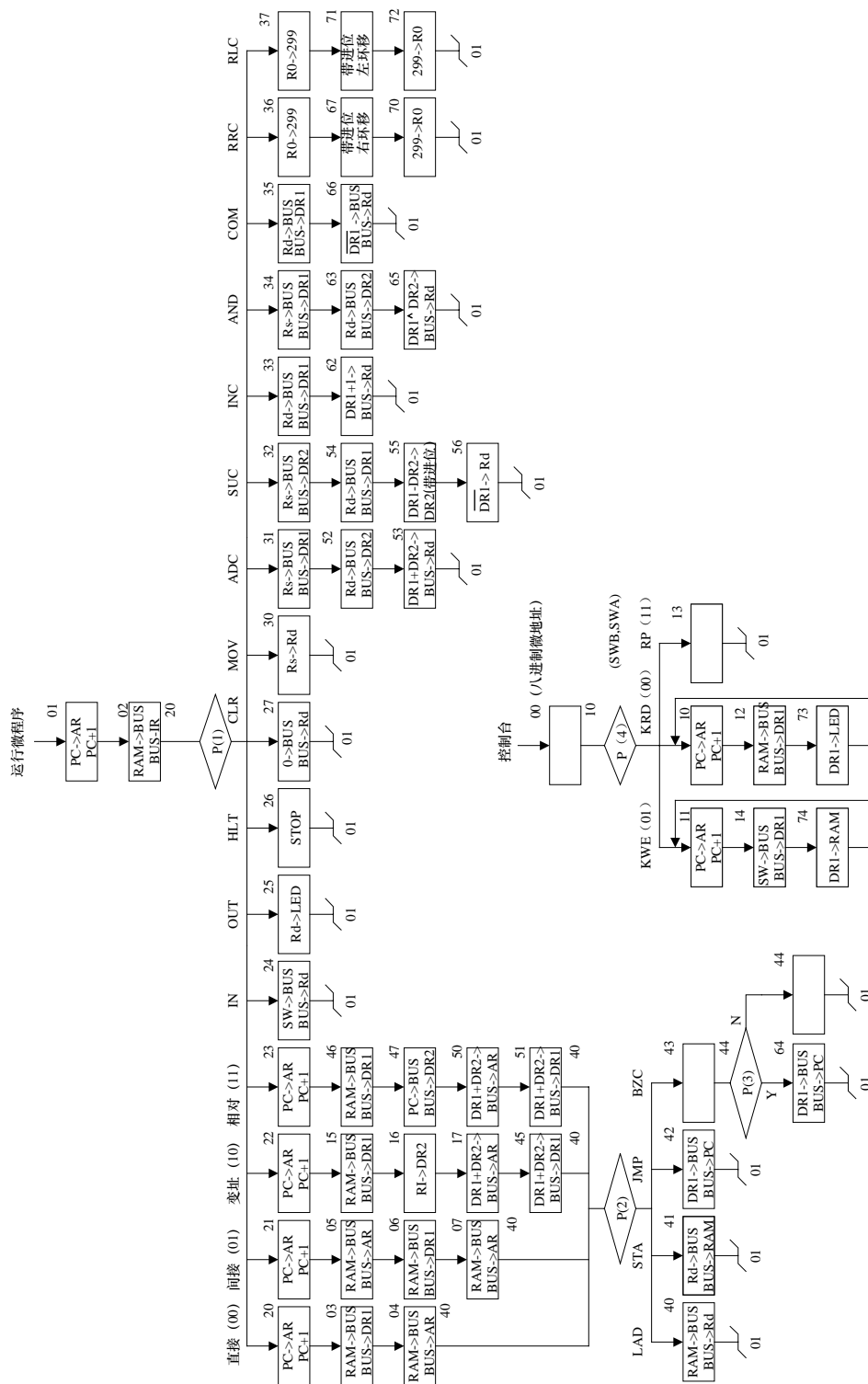


图 7.2-2 微程序流程图

实验程序如下：

程 序	助记符	微程序
\$P0044	IN 01, R0	\$M0D00A00E
\$P0146	IN 01,R2	\$M0E01B60F
\$P0298	ADC R2, R0	\$M0F95EA25
\$P0381	MOV R0, R1	\$M1001ED83
\$P04F5	RLC R1, R1	\$M1101ED85
\$P050C	BZC 00, 00	\$M1201ED8D
\$P0600		\$M1301EDA6
微程序		\$M14001001
\$M00018108		\$M15030401
\$M0101ED82		\$M16018016
\$M0200C050		\$M173D9A01
\$M0300A004		\$M18019201
\$M0400E0A0		\$M1901A22A
\$M0500E006		\$M1A01B22C
\$M0600A007		\$M1B01A232
\$M0700E0A0		\$M1C01A233
\$M0801ED8A		\$M1D01A236
\$M0901ED8C		\$M1E318237
\$M0A00A03B		\$M1F318239
\$M0B018001		\$M20009001
\$M0C00203C		\$M21028401
\$M2205DB81		\$M300D8171
\$M230180E4		\$M31959B41
\$M24018001		\$M32019A01
\$M2595AAA0		\$M3301B435
\$M2600A027		\$M3405DB81
\$M2701BC28		\$M35B99B41
\$M2895EA29		\$M360D9A01
\$M2995AAA0		\$M37298838
\$M2A01B42B		\$M38019801
\$M2B959B41		\$M3919883A
\$M2C01A42D		\$M3A019801
\$M2D65AB6E		\$M3B070A08
\$M2E0D9A01		\$M3C068A09
\$M2F01AA30		

五. 实验步骤

1. 按图 7.2-3 连接实验线路, 仔细查线无误后接通电源。

2. 写微程序

与 PC 联机, 将实验微程序装入实验装置中或脱机时手动将本实验微程序写入实验装置中, 手动写入的具体方法如下:

① 编程

A. 将编程开关置为 PROM (编程) 状态。

B. 将实验板上 “STATE UNIT “中的 “STEP” 置为 “STEP”, “STOP” 置为 “RUN” 状态。

C. 用二进制模拟开关置微地址 MA5~MA0 (如图 4-1-6 须将此六线排针接开关单元)。

D. 在 MK24~MK1 开关上置微代码, 24 位开关对应 24 位显示灯, 开关量为 “0” 时灯亮, 开关量为 “1” 时灯灭。

E. 启动时序电路 (按动启动按钮 “START”), 即将微代码写入到 E²PROM 2816 的相应地址对应的单元中。

F. 重复 C~E 步骤, 将本实验给出的十六进制格式文件转换的二进制代码写入 2816。

② 校验

A. 将编程开关设置为 READ (校验) 状态。

B. 将实验板的 “STEP” 开关置为 “STEP” 状态。“STOP” 开关置为 “RUN” 状态。

C. 用二进制开关置好微地址 MA5~MA0。

D. 按动 “START” 键, 启动时序电路, 读出微代码。观察显示灯 MD24~MD1 的状态 (灯亮为 “0”, 灭为 “1”), 检查读出的微代码是否与写入的相同。如果不同, 则将开关置于 PROM 编程状态, 重新执行①即可。

2. 写程序

方法一: 手动写入

使用控制台 KWE 和 KRD 微程序进行机器指令程序的装入和检查。

A. 使编程开关处于 “RUN”, STEP 为 “STEP” 状态, STOP 为 “RUN” 状态。

B. 拨动总清开关 CLR (1→0→1), 微地址寄存器清零, 程序计数器清零, 然后控制台 SWB、SWA 开关置为 “0 1”, 按动一次启动开关 START, 微地址显示灯显示 “001001”, 再按动一次 START, 微地址灯显示 “001100”, 此时数据开关的内容置为要写入的机器指令, 按动两次 START 键后, 即完成该条指令的写入。若仔细阅读 KWE 的流程, 就不难发现, 机器指令的首地址总清后为 0 0H, 以后每个循环 PC 自动加 1, 所以, 每次按动 START, 只有在微地址灯显示 “001100” 时, 才设置内容, 直到所有机器指令写完。

C. 写完程序后须进行校验。拨动总清开关 CLR (1→0→1) 后, 微地址清零。PC 程序计数器清零, 然后使控制台开关 SWB、SWA 为 “0 0”, 按动启动 START, 微地址灯将显示 “001000”, 再按 START, 微地址灯显示为 “001010”, 第三次按 START, 微地址灯显示为 “111011”, 再按 START 后, 此时输出单元的数码管显示为该首地址中的内容。不断按动 START, 以后每个循环 PC 会自动加 1, 可检查后续单元内

容。每次在微地址灯显示为“001000”时，是将当前地址中的机器指令写入到输出设备中显示。

方法二：联机读／写程序

用联机软件的“【转储】—【装载】”功能将该实验对应的文件载入实验系统即可。

3. 运行程序

方法一：本机运行

① 单步运行程序

- A. 使编程开关处于“RUN”状态，STEP 为“STEP”状态，STOP 为“RUN”状态。
- B. 拨动总清开关 CLR (1→0→1)，微地址清零，程序计数器清零，程序首址为 00H。
- C. 单步运行一条微指令，每按动一次 START 键，即单步运行一条微指令。对照微程序流程图，观察微地址显示灯是否和流程一致。

② 连续运行程序

- A. 使“STATE UNIT”中的 STEP 开关置为“ECEX”状态。STOP 开关置为“RUN”状态。
- B. 拨动 CLR 开关，清微地址及程序计数器，然后按动 START，系统连续运行程序，稍后将 STOP 拨至“STOP”时，系统停机。

方法二：联机运行

联机运行程序时，进入软件界面，装载机器指令及微指令后，选择“【运行】—【通路图】—【复杂模型机】”功能菜单打开相应动态数据通路图，按相应功能键即可联机运行、监控、调试程序。(软件使用说明请看《用户手册》)

总清开关 CLR 清零 (1→0→1) 后，使程序首址及微程序地址为 00H，程序可从头开始运行。

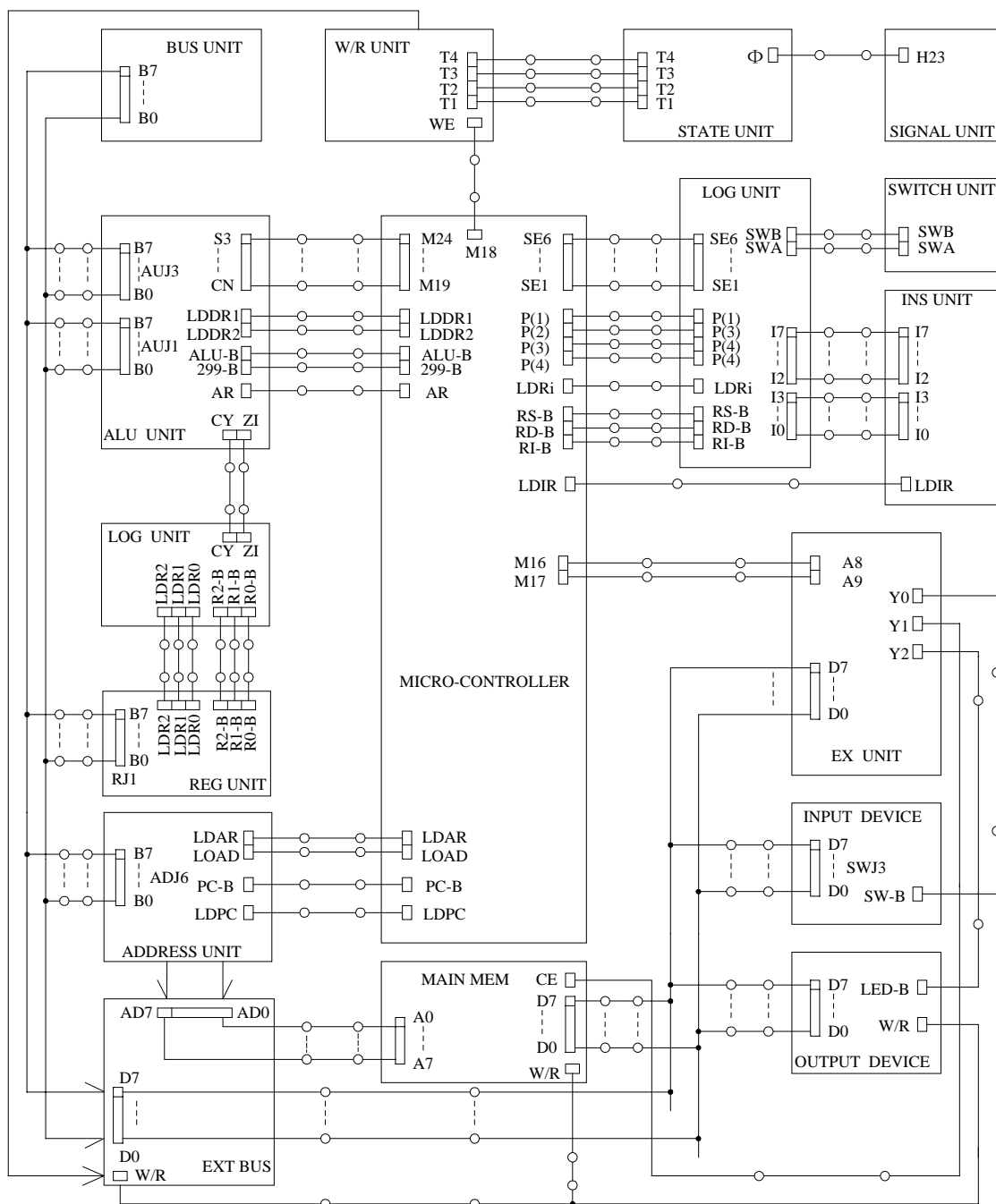


图 7.2-3 实验接线图

7.3 用 CPLD 实现模型计算机的设计实验

一. 实验目的

应用大规模可编程逻辑芯片设计简单的 CPU。

二. 实验设备

1. TDN-CM+或 TDN-CM++教学实验系统一台。
2. PC 微机一台。

三. 实验原理

本实验用 CPLD 芯片编程,来实现第一节基本模型机的设计实验中的 CPU 的功能。为方便地址显示灯观测,地址寄存器仍用实验装置上的电路单元,微程序控制器也由实验板上的单元电路提供,CPU 的其余各模块全部写入 CPLD 中。这样,可由 CPLD+时序电路+微控器+主存+输入输出设备构成一个完整的模型计算机。

实验程序设计:

1. 顶层电路图模块设计,见图 7.3-1:
2. 设计各子模块描述程序。

四. 实验步骤

1. 编辑编译所设计的工程文件,将生成的 JEDEC 文件下载至实验板上的 ispLSI1032 芯片中。
2. 连接实验接线图,见图 7.3-2。
3. 参照第一节基本模型机的设计实验的内容及步骤进行本实验。由于 CPLD 中的各模块的门控信号没有引出至实验板上的控制单元,所以不能用 PC 机的图形界面调试,只能在本机进行手动本机调试运行。

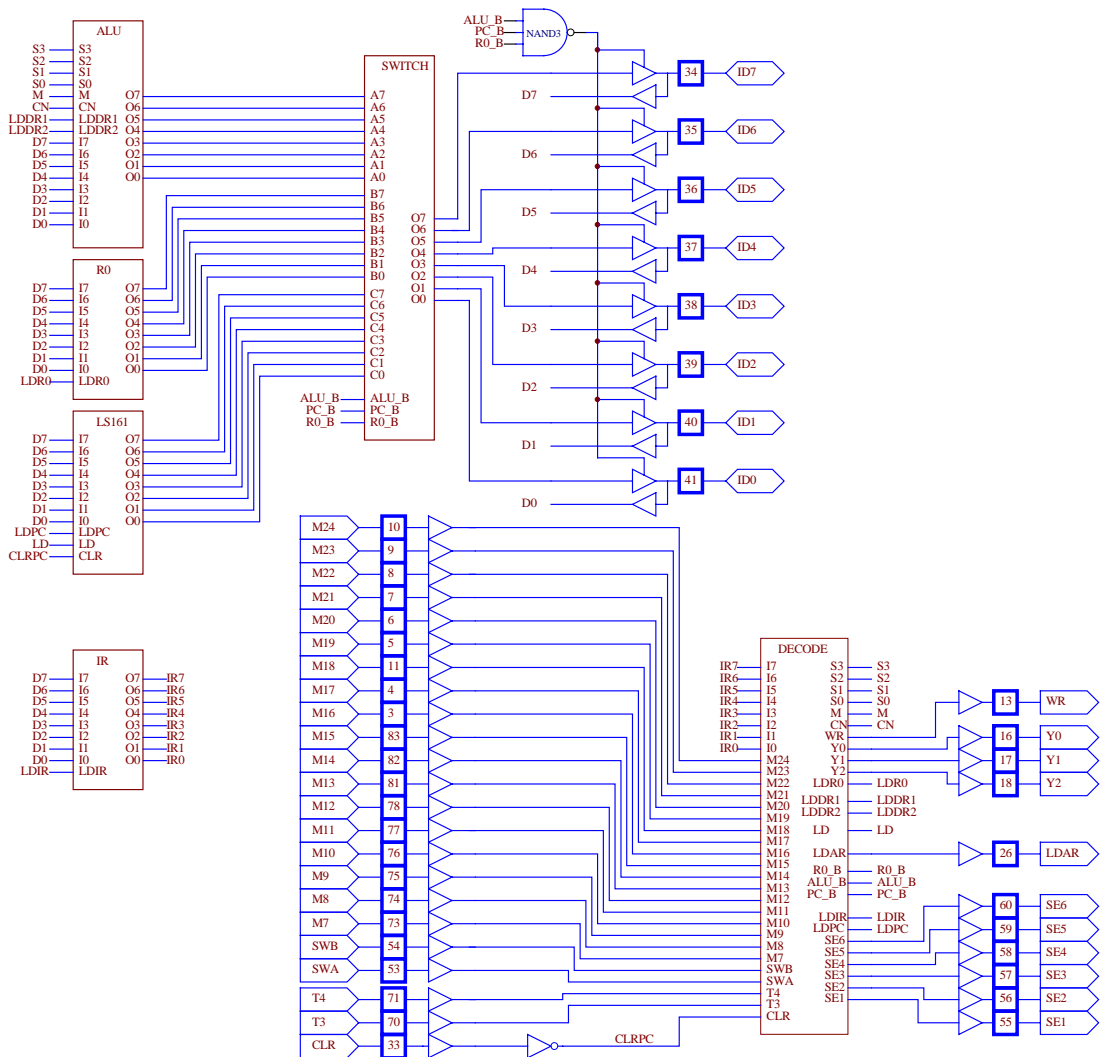


图 7.3-1 顶层电路图

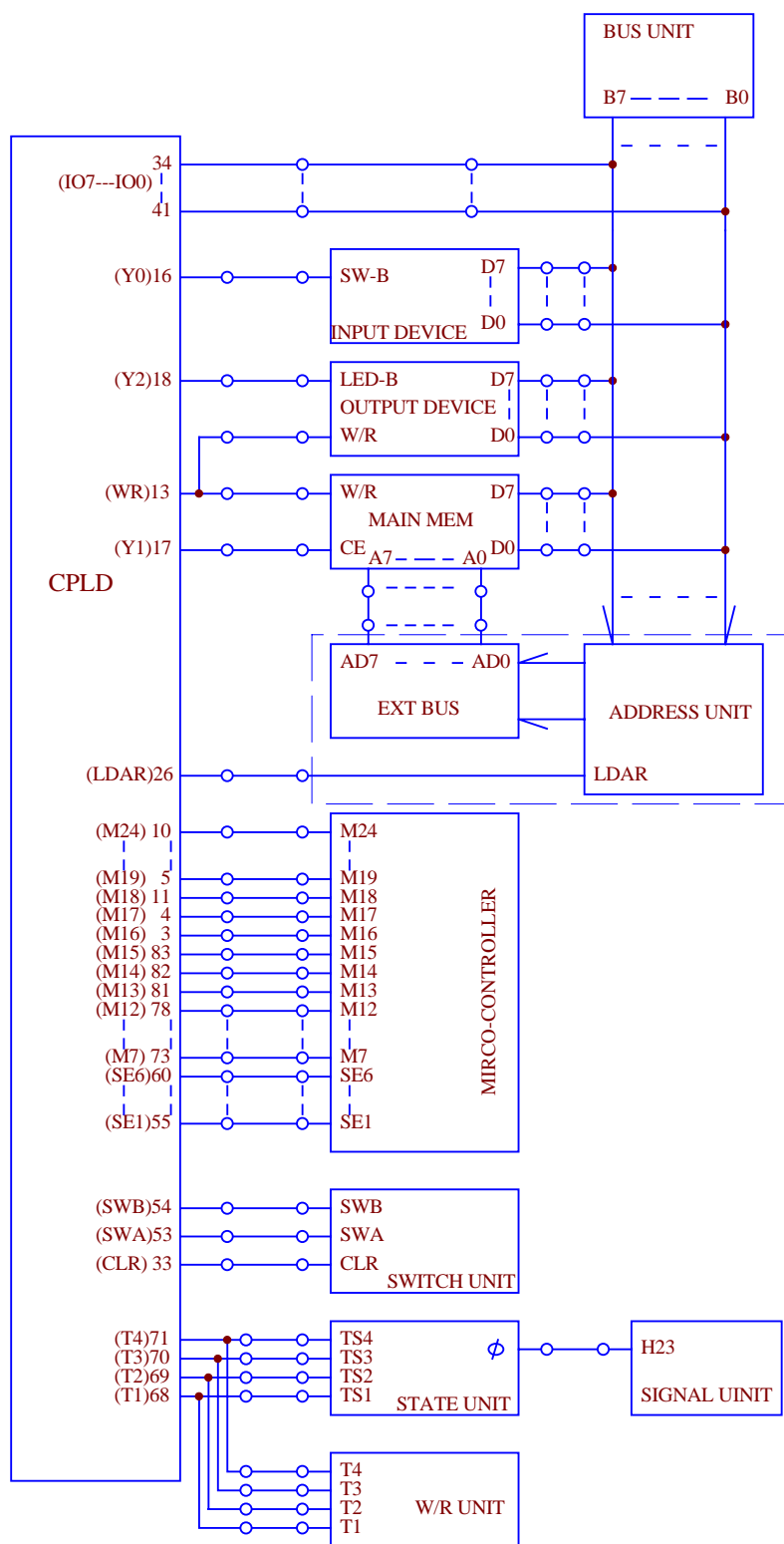


图 7.3-2 接线图

7.4 输入输出系统

7.4.1 输入输出系统概述

一. 输入输出系统基本概念

计算机的输入输出系统也称为 I/O 系统, 包括外围设备、设备控制器、I/O 接口以及一些专门为输入输出操作而设计的软件和硬件。

外围设备是计算机系统与其他机器或人进行信息交换的设备。从功能上可将外围设备分为四类:

- (1) 进行输入输出操作的设备。如键盘、鼠标、显示器、打印机等。
- (2) 辅助存储设备。如硬盘、软盘、光盘等。
- (3) 数据通信设备。主要用于计算机网络, 如调制解调器、网卡、传真机等。
- (4) 过程控制设备。如各种传感器、A/D 转换器、D/A 转换器等。

二. 接口的功能及分类

计算机和各种 I/O 设备是相对独立的, 它们之间要直接进行信息交换存在很多问题: 两者工作速度差别很大; 两者之间是异步的工作状态, 各自有自己独立的时钟及时序; 两者之间的数据格式不一样等。所以要实现它们之间的连接, 必须经过一个“转换”机构, 设置一套逻辑控制部件, 这样的转换机构称为“I/O 接口”, 简称“接口”。

为了实现设备间的通信, 不仅需要由硬件逻辑构成的接口部件, 还需要相应的软件。接口可分为软件接口和硬件接口。

软件接口: 软件模块之间的交接部分, 模块之间的传递参数规则或约定。

硬件接口: 硬件设备之间连接逻辑及信号传递协议。

软硬接口: 硬件与软件之间相互作用, 软件对硬件电路进行控制, 或者硬件要传递信息给软件, 相互之间共同遵守的协议。

1. I/O 接口的基本功能

(1) 设备寻址

由于外围设备可能有多个, 所以接口逻辑要根据总线上的寻址信息, 找到某一台设备或接口中的某个有关的寄存器。

(2) 数据传送与缓冲

在接口中设置一个或多个数据缓冲寄存器, 协调主机与外围设备的工作速度, 实现速度匹配。

(3) 数据格式变换、电平变换等预处理

接口和系统总线之间, 一般采取并行传送, 若接口与外围设备之间采用串行传送, 就需要进行串并格式转换。

若设备与系统总线使用的电源不同, 还需要进行电平转换。

(4) 控制逻辑

由接口来解释主机通过总线传送命令信息, 并变成相应的操作命令发送给设备。接口形成设备以及接口本身的有关状态信息, 通过总线回送给主机。

2. I/O 接口的分类

(1) 按数据传送格式分

并行接口：接口与系统总线及外围设备之间全部采用并行方式传送数据信息。

串行接口：接口与外围设备采用串行方式传送数据信息。通常，接口需要经过串并转换再和系统总线进行并行通信。

(2) 按时序控制方式分

同步接口：与同步总线连接的接口，接口与系统总线间的信息传送由统一的时序信号控制。接口与外设间则允许有独立的时序控制操作。

异步接口：与异步总线相连的接口，接口与系统总线间的信息传送采用异步应答的控制方式。

三. 外围设备的寻址

用程序实现输入/输出传送的机器，必须对连接到计算机系统上的多个外围设备进行编址，以便选择某一台设备参与某个指定的输入/输出操作。外围设备有两种编址方法：统一编址和单独编址。

统一编址又称存储器映射的寻址方法，就是将接口中的寄存器当作主存单元一样，和主存单元一起统一编址，从总的地址空间中分出一部分地址码，来为接口寄存器编址。这样就可以用访问内存的指令去访问 I/O 设备的某个寄存器，而不需要设置专门的 I/O 指令。这种方式适用于单总线结构，外围设备、主存、CPU 共用一条总线，既作为存储总线，又作为输入输出总线。这种方式占用了存储器的寻址空间，使存储器的寻址空间减少。

单独编址又称端口寻址方式，内存地址和 I/O 设备地址是分开的。设有专门的输入输出指令可以指示出设备码以及要完成的操作等，内存和 I/O 设备使用不同的读/写线控制。

7. 4. 2 输入输出的基本控制方式

在计算机系统中，输入输出控制方式主要有四种：程序查询方式、程序中断方式、直接内存访问方式及通道方式。

1. 程序查询方式

这种方式下外围设备的工作全部由 CPU 通过 I/O 指令进行直接控制。这种方式硬件结构比较简单，CPU 的操作和外围设备的操作能够同步。CPU 一直处于主动地位，不断查询各设备的状态标志，一旦设备准备就绪，就立即执行一条数据传送指令，完成 I/O 接口的数据缓冲寄存器与 CPU 寄存器之间的数据传送。

但外围设备工作速度很慢，CPU 在程序查询时会浪费很多时间而不能做其他事情。

2. 程序中断方式

在这种方式下 CPU 不去查询设备的状态，而是做自己的事情，当外围设备准备就绪后，它会主动向 CPU 发出“中断请求”信号。主机接到中断申请后就暂停它现行的程序，而转到中断处理程序，进行数据传送处理。处理完后，CPU 又返回到它原来的任务，从暂停处继续执行。虽然 CPU 在响应中断请求，进行中断处理要有一些辅助的操作，消耗一些时间，但是和等待外围设备相比要短得多，所以大大的提高了 CPU 的工作效率。

3. 直接内存访问方式 (DMA)

由于每次中断都需要一些辅助的操作，这对于高速外设成批数据传送，将很不适合。一

方面频繁的中断会消耗太多的额外的时间开销,另一方面会有两次数据传送的时间会比响应一次中断的时间还短,这样会造成数据丢失。所以 DMA 方式就是在主存与外围设备之间建立一条数据通路,由 DMA 控制器取得总线的控制权,来控制数据的交换。DMA 控制器从 CPU 完全接管对总线的控制,数据不经过 CPU 而直接在主存和外围设备间进行,以高速传送数据。只是在数据块传送的开始和结束时需要 CPU 的介入。这种方式是一种完全由硬件执行的数据传送,它不执行程序,不能处理较复杂的事件。因此不能完全取代中断方式,在以 DMA 方式传送完一批数据后,往往中断方式通知 CPU 进行结束处理。DMA 方式适合于内存和高速外围设备间大批数据交换的场合。

4. 通道方式

通道方式也是一种以主存为中心的系统。通道是计算机系统中一个独立的部件,它是一个具有特殊功能的处理器,它可实现对外围设备的统一管理和外围设备与内存之间的数据传送,可看作被多台设备所共享的 DMA 设施。通道和 DMA 的主要区别是:通道有自己的指令,即“通道命令”。它具有相对独立于 CPU 之外,执行通道程序的处理机功能。它通过执行通道程序实现外围设备直接和主存交换数据。它比 DMA 控制器有更强的逻辑功能,可以实现更为复杂的数据传送。

7.4.3 程序中断方式

一. 中断的基本概念

1. 定义

一般来说,在计算机的运行过程中,如果发生某种随机事态,CPU 将暂停执行现行的程序,转去另一个更为紧迫的服务性程序,并在执行结束后自动返回原来的程序,这一过程称为“中断”。

2. 中断的类型

(1) 根据中断的原因,可将中断分为两大类型:内中断和外中断。

内中断

这类中断来自主机内部,是由于主机内部发生了故障,而中止正常程序的运行,转去执行相应的中断处理程序。如掉电中断、CPU 故障中断、软中断等。

内中断主要有强迫中断和自愿中断两种。

强迫中断是由于故障、外部请求等所引起的强迫性中断,非程序本身安排的,这种请求的提出和相应的处理都是随机的。

自愿中断又称程序自中断,即软中断。这是程序有意安排的,以中断方式引出服务程序,实现某种功能。

外中断

这类中断来自主机的外部,也就是外围设备的中断。如打印机中断、键盘中断等。

(2) 按照中断的处理方式可将中断分为程序性中断和简单中断。

程序性中断就是一般所指的中断,CPU 响应中断前需要保护现场,而后转入执行中断服务程序。在执行完后返回主程序前必须恢复现场。

简单中断就是 DMA 方式,主机在响应中断申请后不需要执行服务程序,只须暂时让出

几个存取周期，让外围设备和主存直接交换数据即可。

(3) 根据中断是由软件引起还是硬件引起的可分为硬件中断和软中断。

硬件中断是由某个硬件中断请求信号引发的中断，软中断是由执行软中断指令所引起的中断。对他们的处理是相同的，其区别仅在于：硬件中断通过中断请求信号形成向量地址，而软中断由指令提供中断号 n ，再转换为向量地址。

二. 中断系统的功能

中断系统应具有如下功能：

1. 实现中断及返回

当某一中断源发出中断请求时，CPU 能暂停当前的任务，而转去执行中断服务程序，执行完后返回断点处继续执行主程序。

2. 能实现优先权排队

通常，在系统中有多个中断源，会出现两个或多个中断源同时提出中断请求的情况，这就必须给每个中断源确定一个中断级别——优先权。当多个中断源同时发出中断请求时，CPU 能找到优先权级别最高的中断源，响应它的请求。在级别高的中断源处理完以后，再响应级别较低的中断源。

3. 高级中断源能中断低级中断源的中断处理

当 CPU 响应某一中断源请求，在进行中断处理时，若有优先权级别更高的中断源发出中断请求，则 CPU 能中断正在执行的中断服务程序，保留断点和现场，响应更高级中断。高级中断处理完后，再继续执行被中断的中断服务程序。而当发出中断请求的中断源与现在正在处理的中断源是同级或更低，CPU 则不响应，直到中断服务程序执行完后才去处理新的中断请求。

三. 中断处理过程

计算机的中断处理过程，主要包括响应中断时执行中断隐指令、执行中断服务程序及执行返回指令。对于多级中断和单级中断有一些差别，见表 7.4-1 所示。

(1) 计算机响应中断过程

CPU 在执行每条指令的最后一个机器周期检测是否有中断源中断请求，若有且可以响应，就向外部发出中断响应信号 $INTA$ 和其他信号。当现行的机器指令执行完后，就转入中断周期，执行以下几项：

① 关中断

为了保证本次中断处理过程不受干扰，CPU 进入中断周期后便关掉中断。

② 保存断点

将程序计数器 PC 的内容压入堆栈中保存。此时 PC 中的内容为恢复原程序后的后继指令地址，称为断点。有些高档计算机也将程序状态字 PSW 及某些寄存器也压栈保存。

③ 取得中断服务程序入口地址

通过总线送入向量地址，CPU 据此在中断周期中访问中断向量表，从中读取服务程序入口地址，然后送给程序计数器 PC，就转入到中断服务程序了。

以上的操作是在中断周期中直接依靠硬件来实现，并非执行程序指令。所以称为中断隐指令操作。

(2) 执行中断服务程序

表 7.4-1

中断处理阶段	多级中断	单级中断
执行中断 隐指令	关中断 保存断点及 PSW 取得服务程序入口地址	关中断 保存断点及 PSW 取得服务程序入口地址
执行中断 服务程序	保护现场 送新屏蔽字 开中断	保护现场
	服务处理（可中断）	服务处理（不可中断）
	关中断	
	恢复现场及屏蔽字	恢复现场
	开中断 中断返回	开中断 中断返回
执行返回指令	从堆栈弹出断点地址及 PSW 返回	从堆栈弹出断点地址及 PSW 返回

① 保护现场

将程序状态字 PSW 及在处理程序中用到的寄存器进行压栈保护，以免造成改变，在返回断点时发生错误。

② 多级中断与单级中断

对于接下来的处理多级中断和单级中断会有些不同。多级中断主要是允许中断嵌套，开中断。

1) 设置新的屏蔽字

屏蔽掉与本次请求同一优先级别以及更低级别的其他请求。

2) 开中断

开放中断可以使更高级别的中断源申请得以成功。

3) 服务处理

执行中断后要执行的中断处理程序。

4) 关中断

服务处理程序执行完成后需要恢复现场，这期间不能被干扰，所以先关掉中断。

以上几步是多级中断需处理的操作，对于单级中断，不需要开中断及设置新屏蔽字，只有在中断服务程序处理完恢复现场后才开中断。

③ 恢复现场

将在堆栈中保存的程序状态字 PSW 及寄存器按压栈相反的顺序弹出恢复原来的内容。对于多级中断还需要恢复原来的中断屏蔽字。

④开中断

⑤中断返回

执行一条中断返回指令，将堆栈中保存的断点地址弹出赋给程序计数器 PC，机器就转

移回原程序断点继续运行。

四．向量中断

中断服务程序的入口地址获得一般采用向量中断方式。

中断向量 存放中断服务程序入口地址信息的单元称为中断向量。一个计算机系统有若干中断源，每个中断源有自己相应的服务程序，每个服务程序模块有自己的入口地址。每个入口地址是一个标量（或视为布尔量），而这些入口地址被组织为标量的一维的有序集合，因而称为中断向量。

中断向量表 所有中断服务程序入口地址组成一个一维的表格，存放在一段连续的存储区中。

向量地址 存储中断向量单元的地址。

向量中断是指这样一种中断响应方式：将各个中断的服务程序的入口地址组织成中断向量表；响应中断时，由硬件直接产生对应的中断源的向量地址；据此访问中断向量表，从中读取服务程序入口地址，由此转向服务程序。这些工作一般在中断周期中由硬件直接实现。

7. 5 具有中断处理功能的模型机设计实验

一．实验目的

1. 在构成一台完整模型机的基础上，控制真实的外围接口芯片，进行基本的接口实验。
2. 本实验外扩一片 8259 接口芯片，完成中断处理功能实验。
3. 掌握中断原理及其响应流程。
4. 掌握 8259 中断控制器原理及其应用编程。

二．实验设备

1. TDN-CM+或 TDN-CM++教学实验系统一台。
2. 接口实验板。
3. PC 微机一台。

三．实验原理

1. 8259 芯片引脚分配图如图 7.5-1 所示

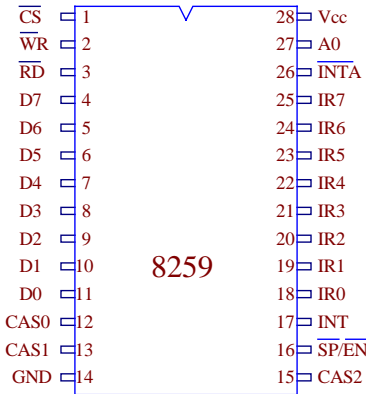


图 7.5-1 8259 芯片引脚说明

- D7~D0 为双向三态数据线
- \overline{CS} 片选信号线
- A0 用来选择芯片内部不同的寄存器，通常接至地址总线的 A0。
- \overline{RD} 读信号线，低电平有效，其有效时控制信息从 8259 读至 CPU。
- \overline{WR} 写信号线，低电平有效，其有效时控制信息从 CPU 写入至 8259。
- $\overline{SP/EN}$ 从程序/允许缓冲
- \overline{INTA} 中断响应输入
- INT 中断输出
- IR0~IR7 8 条外界中断请求输入线。
- CAS2~CAS0 级连信号线。

\overline{CS} 、A0、 \overline{RD} 、 \overline{WR} 、D4、D3 位的电平与 8259 操作关系如表 7.5-1 所示：

表 7.5-1 8259A 的读/写操作

A0	D4	D3	\overline{RD}	\overline{WR}	\overline{CS}	操 作
0			0	1	0	输入操作（读）
1			0	1	0	IRR, ISR 或中断级别 \rightarrow 数据总线 IMR \rightarrow 数据总线
0	0	0	1	0	0	输出操作（写）
0	0	1	1	0	0	数据总线 \rightarrow OCW2
0	1	X	1	0	0	数据总线 \rightarrow OCW3
1	X	X	1	0	0	数据总线 \rightarrow OCW1
						数据总线 \rightarrow OCW1,ICW2,ICW3,ICW4
X	X	X	1	1	0	断开功能
X	X	X	X	X	1	数据总线 \rightarrow 三态（无操作） 数据总线 \rightarrow 三态（无操作）

2. 指令系统

本模型机共设计 9 条基本指令及 3 个控制台操作指令。表 7.5-2 列出了基本指令的格式、助记符及其功能。

表 7.5-2

助记符号	指令格式			功能
IN rd	0000	00	rd	INPUTDEVICE \rightarrow rd
OUT rs	0001	rs	00	rs \rightarrow OUTPUTDEVICE
MOV rs, rd	0010	rs	rd	rs \rightarrow rd
JMP D	0011	00	00	D \rightarrow PC
	D			
INC rs rd	0100	rs	rd	rs+1 \rightarrow rd
STA rs, D	0101	rs	00	rs \rightarrow D
	D			
LAD D, rd	0110	00	rd	D \rightarrow rd
	D			
PIN P, rd	0111	00	rd	P \rightarrow rd
	P			
POUT rs, P	1000	rs	00	Rs \rightarrow P
	P			
IRET	1001	00	00	中断返回

其中，D 为立即数，P 为外设的端口地址；rs 为源寄存器，rd 为目的寄存器，并规定：

rs 或 rd	选定的寄存器
00	R0
01	R1
10	R2

三条控制台指令用两个开关 SWB、SWA 的状态来设置，其定义如下：

SWB	SWA	控制台指令
0	0	读内存（KRD）
0	1	写内存（KWE）
1	1	启动程序（RP）

3. 根据指令系统要求, 设计微程序流程及确定微地址。如图 7.5-2。

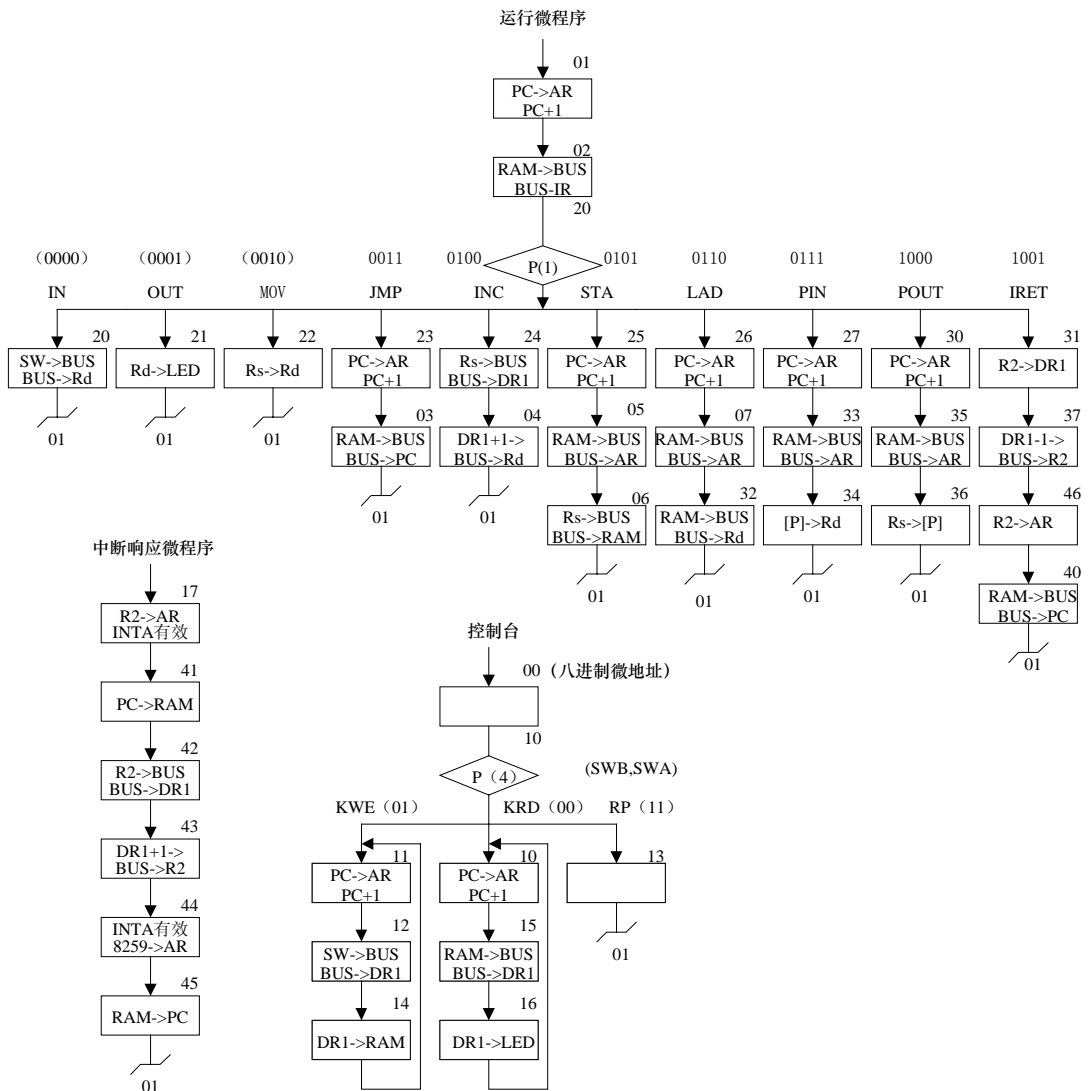


图 7.5-2 中断实验微程序流程图

4. 本实验由于需要断点保护, 须设置一个堆栈, 特设由 R2 寄存器专门做堆栈指针。

5. 由于此中断实验 CPU 还须有一个 INTA 信号、ICF 指令执行完成标志, 可以由微代码中的 M23、M24 位来定义。将 ALU 的控制信号 S3、S2、S1、S0 简化为只有 S1、S0 控制, 而 ALU 单元的 S3、S2、S1、S0 均由 CPLD 输出。LDDR2 控制信号也简化掉。这样, 微指令格式可设计为如表 7.5-3 所示。其中 M24 为 ICF 标志, M23 为 INTA 信号, 其为高有效, A 字段原来的 LDDR2 由 LDR2 代替, R2-B 代替 B 字段原 RI-B。

表 7.5-3

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
ICF	INTA	S1	S0	M	Cn	WE	A9	A8	A		B		C		uA5	uA4	uA3	uA2	uA1	uA0			

A 字段

15	14	13	$\tilde{N}_i\hat{O}_n$
0	0	0	
0	0	1	LDRi
0	1	0	LDDR1
0	1	1	LDDR2
1	0	0	LDIR
1	0	1	LOAD
1	1	0	LDAR

B 字段

12	11	10	$\tilde{N}_i\hat{O}_n$
0	0	0	
0	0	1	RS-B
0	1	0	RD-B
0	1	1	RI-B
1	0	0	299-B
1	0	1	ALU-B
1	1	0	PC-B

C 字段

9	8	7	$\tilde{N}_i\hat{O}_n$
0	0	0	
0	0	1	PE"1f©
0	1	0	PE"2f©
0	1	1	PE"3f©
1	0	0	PE"4f©
1	0	1	AR
1	1	0	LDPC

6. 要增加中断响应，所以指令译码电路须重新设计。它们全部用一片 CPLD 芯片描述。相应的原理图见 7.5-3、7.5-4：

The diagram shows a CPLD logic circuit for instruction decoding. It has several inputs: ICF, INT, T4, P(1), P(4), SWB, and SWA. The circuit is enclosed in a dashed box. At the top, there is a 4-bit bus labeled SE4---SE1. Inside the box, there are several logic gates: AND, OR, NAND, and NOR. The inputs are connected to these gates in a complex manner to generate control signals. At the bottom, there is a 4-bit bus labeled 17---14. The circuit is designed to decode instructions and generate control signals for various ALU and register operations.

图 7.5-3 指令译码

. 128 .

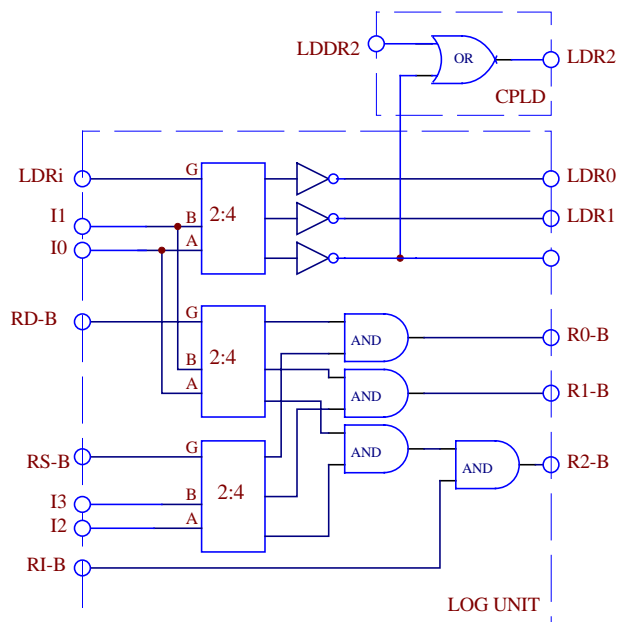


图 7.5-4 寄存器译码

其中,图 7.5-3 为指令译码电路,全由 CPLD 来描述,而 7.5-4 为本实验中的寄存器译码,它是在 LOG UNIT 单元的译码电路的基础上,增加一个或门,即 LOG UNIT 单元输出的 LDR2 和微代码的 LDR2(原微代码 A 字段的 LDDR2 位)相或后的输出 LDR2,这个或门是由 CPLD 描述的。

四. 实验步骤

1. 用 ABEL 语言编写上述设计的中断响应及指令译码电路逻辑,其在 CPLD 芯片 1032 中的对应管脚同图 7.5-5 对应。

I7	11		
I4	14	75	SE6
P1	79	70	SE1
P4	76	26	S3
SWB	51	29	S0
SWA	52	30	M
LLDR2	41	31	CN
LDDR2	45	34	LDR2
ICF	10	18	OINTA
IINTA	9		
M22	8		
M19	5		
T4	3		
INT	33		
		ISPLSI1032	

图 7.5-5 实验对应应在 1032 中设计的管脚

2. 按图 7.5-6、7.5-7 连接实验接线。

其中，接至 SWITCH UNIT 单元 SW-B 开关为中断请求信号，它处于高电平时为中断请求有效。

3. 根据微指令格式，参照微程序流程图，将每条微指令代码化，并编写一段机器指令，程序如下：

机器指令	助记符	说明
\$P0060	LAD 13, R0	将 13 装入 R0 中
\$P0113		
\$P0280	POUT R0, [00]	RO 中的内容写入端口 00 中
\$P0300		即初始化命令字 ICW1
\$P0460	LAD 30, R0	将 30 装入 R0 中
\$P0530		
\$P0680	POUT R0, [01]	R0 中的内容写入端口 01 中
\$P0701		即初始化命令字 ICW2，中断向量为 30-37
\$P0860	LAD 03, R0	
\$P0903		
\$P0A80	POUT R0, [01]	初始化命令字 ICW4
\$P0B01		
\$P0C60	LAD 00, R0	
\$P0D00		
\$P0E80	POUT R0, [01]	写工作方式字 OCW1
\$P0F01		
\$P1062	LAD A0, R2	初始化堆栈指针 R2 指向地址 A0
\$P11A0		
\$P1200	IN R0	从数据输入开关读一个数到 R0
\$P1321	MOV R0, R1	
\$P1430	JMP 13	程序转移至 13
\$P1513		
\$P1640	INC R0	R0+1
\$P1710	OUT R0	中断处理入口，将 R0 由数码管显示
\$P1890	IRET	中断返回
\$P3016		IRQ0 的中断矢量
微指令		
\$M00010108		\$M08016D8D
\$M01016D82		\$M09016D8A
\$M0200C050		\$M0A00200C
\$M0380D181		\$M0B010001

\$M04811A01	\$M0C068A09
\$M0500E006	\$M0D00A00E
\$M06828201	\$M0E070A08
\$M0700E01A	\$M0F416621
\$M10801001	\$M1C819001
\$M11830401	\$M1D00E01E
\$M12811201	\$M1E838201
\$M13016D83	\$M1F353A26
\$M14012204	\$M2080D181
\$M15016D86	\$M21028C22
\$M16016D9A	\$M22012623
\$M17016D9B	\$M23013A24
\$M18016D9D	\$M2441E025
\$M1901261F	\$M2500D181
\$M1A809001	\$M26016620
\$M1B00E01C	

4. 装入机器指令及微程序，运行程序，检验所设计的中断控制及指令系统。

可联机或在本机状态下运行调试。在本地运行时，将 STEP 开关置为 EXEC，按下启动键 START 连续运行机器指令程序。此时，可以通过拨动 SW-B 开关为“0→1”操作，使系统产生中断请求。每当进入中断程序，R0 寄存器进行加一，并由数码管显示后返回。

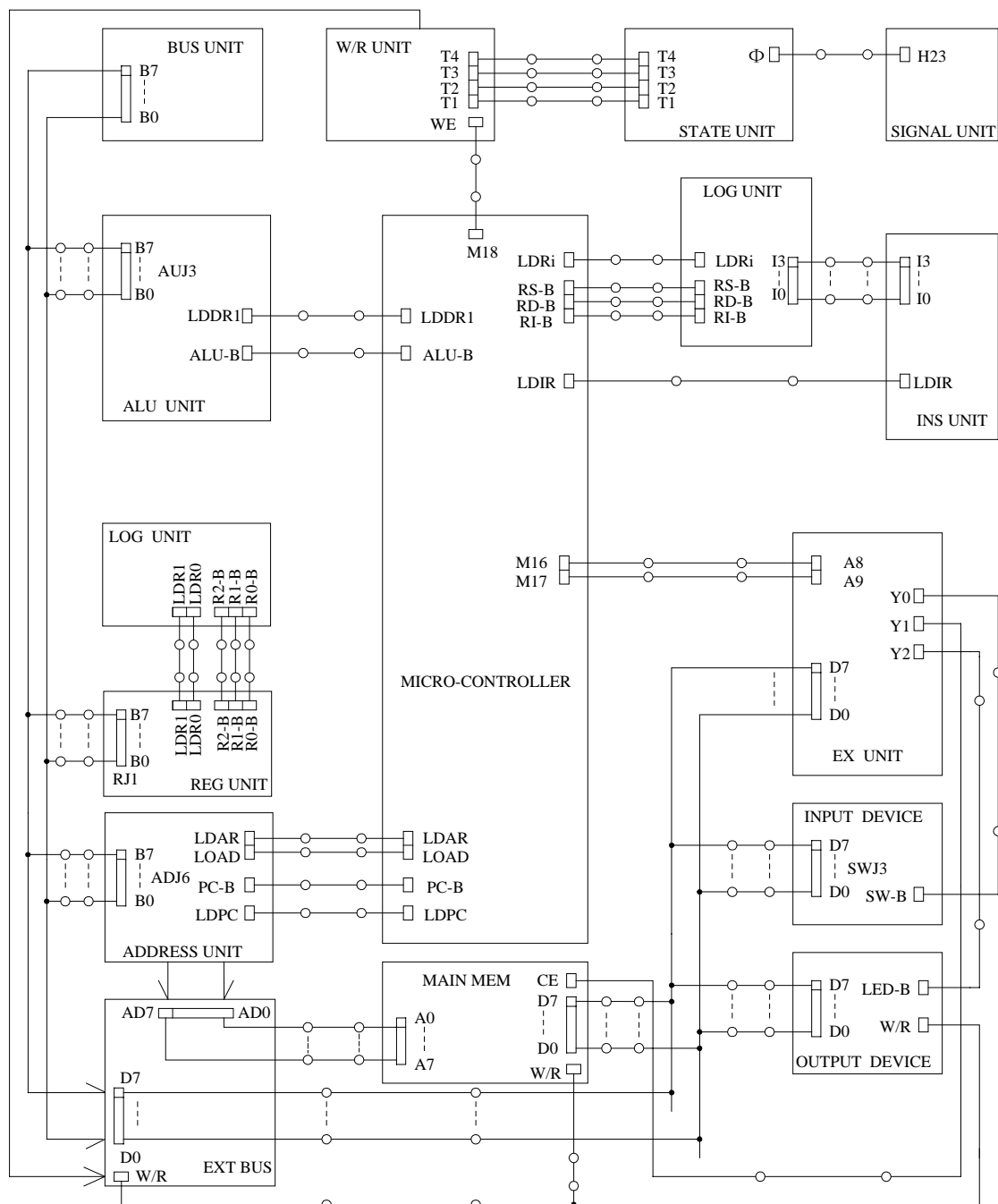


图 7.5-6 中断实验接线图 (1)

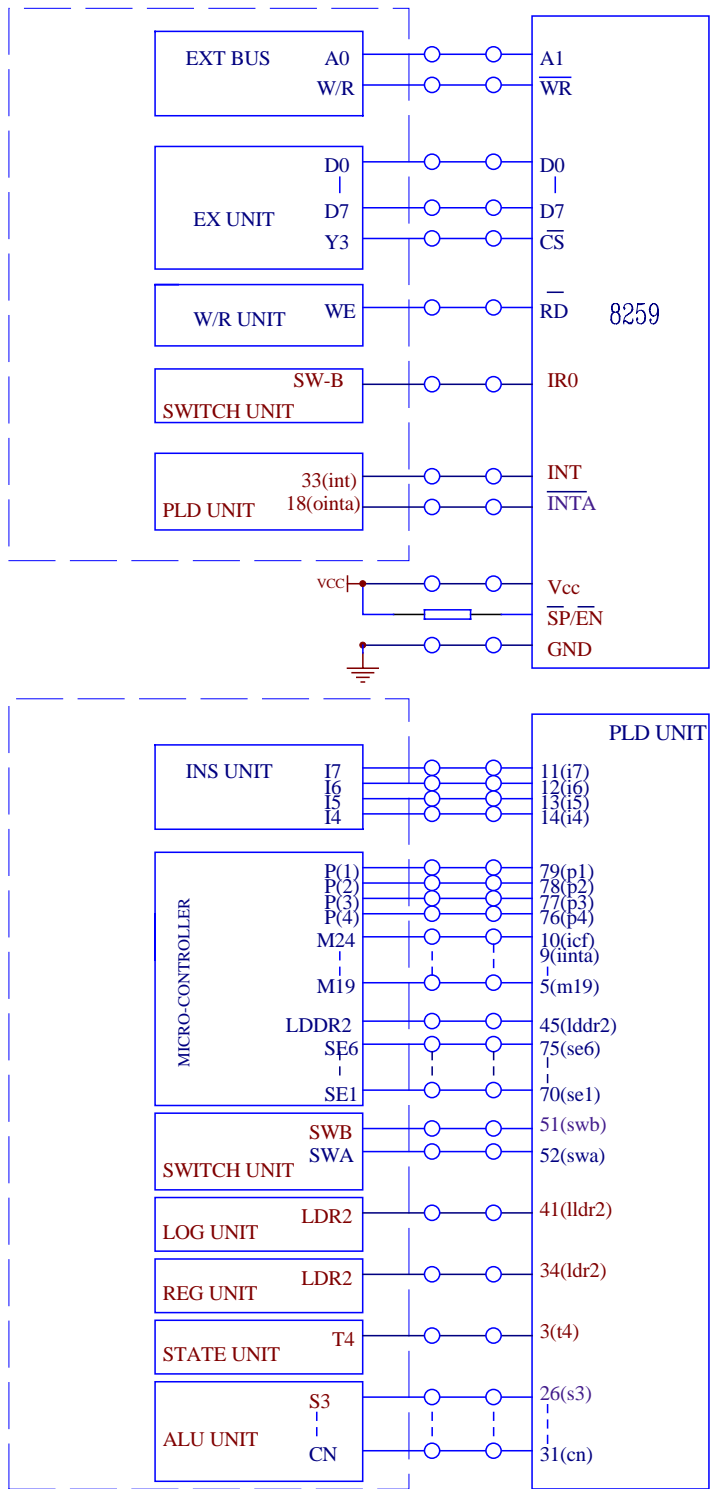


图 7.5-7 中断实验接线图 (2)

7. 6 扩展 8255 并行口实验

一. 实验目的

- 1. 在构成一台完整模型机的基础上, 控制真实的外围接口芯片, 进行基本的接口实验。
- 2. 本实验外扩一片 8255 接口芯片, 完成基本并行口实验。

二. 实验设备

- 1. TDN-CM+或 TDN-CM++教学实验系统一台。
- 2. 接口实验板。

三. 实验原理

- 1. 8255 芯片引脚特性及外部连接
- (1) 8255 的引脚分配图如图 7.6-1 所示

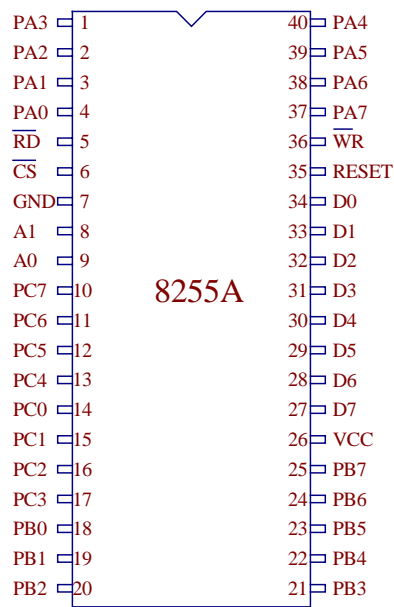


图 7.6-1 8255 芯片引脚说明

- (2) 芯片引脚说明
 - D7~D0 为 8 位数据线
 - \overline{CS} 为片选信号,低电平有效,由它启动 CPU 与 8255 间的通信
 - A0、A1 用来选择三个输入输出端口和控制寄存器
 - \overline{RD} 为读信号,低电平有效。它控制 8255 送出数据或状态信息至 CPU。

- \overline{WR} 为写信号，低电平有效。它控制把 CPU 输出的数据或命令信号写到 8255。
 - RESET 为复位信号，高电平有效，它清除控制寄存器和置所有端口到输入方式。
- \overline{CS} 、A0、A1、 \overline{RD} 、 \overline{WR} 五个引脚的电平与 8255 操作关系如表 7.6-1 所示：

表 7.6-1

A1	A0	\overline{RD}	\overline{WR}	\overline{CS}	操作
0	0	0	1	0	输入操作（读） 数据总线 \Longrightarrow 通道 A
0	1	0	1	0	数据总线 \Longrightarrow 通道 B
1	0	0	1	0	数据总线 \Longrightarrow 通道 C
0	0	1	0	0	输出操作（写） 数据总线 \Longrightarrow 通道 A
0	1	1	0	0	数据总线 \Longrightarrow 通道 B
1	0	1	0	0	数据总线 \Longrightarrow 通道 C
1	1	1	0	0	数据总线 \Longrightarrow 控制字寄存器
X	X	X	X	1	断开功能 数据总线 \Longrightarrow 三态
1	1	0	1	0	非法条件
X	X	1	1	0	数据总线 \Longrightarrow 三态

2. 为本实验设计两条指令

(1) 端口读指令

助记符

PIN P, Ri

指令格式

1110		Ri
P		

其中第一字节前四位为操作码，P 为端口地址，其功能是将端口地址为 P 的端口内容写入至寄存器 Ri 中。

(2) 端口写指令

助记符

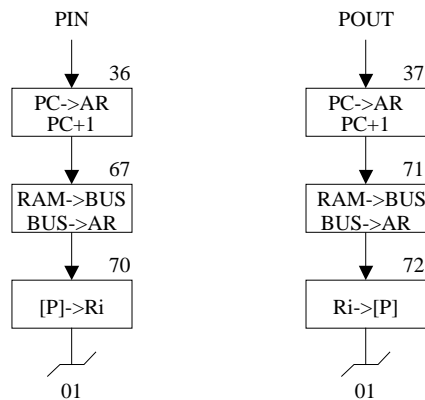
POUT Ri, P

指令格式

1111	Ri	
P		

其功能是将 Ri 寄存器中的内容写至以 P 为端口地址的端口中。

3. 两条指令的微程序流程可设计为：



根据以上的微程序流程图编写微程序，见表 7.6-2。

表 7.6-2

微地址	S3 S2 S1 S0 M CN WE A9	A8	A	B	C	μA5-μA0	十六进制代码
36	0 0 0 00 0 0 0 1	1	1 1 0 1 1 0 1 1 0			110111	01EDB7
37	0 0 0 00 0 0 0 1	1	1 1 0 1 1 0 1 1 0			111001	01EDB9
67	0 0 0 00 0 0 0 0	1	1 1 0 0 0 0 0 0 0			111000	00E038
70	0 0 0 00 0 0 0 1	0	0 0 1 0 0 0 0 0 0			000001	011001
71	0 0 0 00 0 0 0 0	1	1 1 0 0 0 0 0 0 0			111010	00E03A
72	0 0 0 00 0 1 1 1	0	0 0 0 0 0 1 0 0 0			000001	030201

四．实验步骤

1．如图 7.6-2 连接实验接线。

2．在复杂模型机所设计的微程序基础上将表 7.6-2 的微程序写入到微程序控制存储器中(表 7.6-2 中的微地址为八进制表示，写入时应将其转换成十六进制。如 36 应转换成 1E)。

微程序如下所示：

\$M00018108	\$M0C00203C	\$M18019201
\$M0101ED82	\$M0D00A00E	\$M1901A22A
\$M0200C050	\$M0E01B60F	\$M1A01B22C
\$M0300A004	\$M0F95EA25	\$M1B01A232
\$M0400E0A0	\$M1001ED83	\$M1C01A233
\$M0500E006	\$M1101ED85	\$M1D01A236
\$M0600A007	\$M1201ED8D	\$M1E01EDB7
\$M0700E0A0	\$M1301EDA6	\$M1F01EDB9
\$M0801ED8A	\$M14001001	\$M20009001
\$M0901ED8C	\$M15030401	\$M21028401
\$M0A00A03B	\$M16018016	\$M2205DB81
\$M0B018001	\$M173D9A01	\$M230180E4

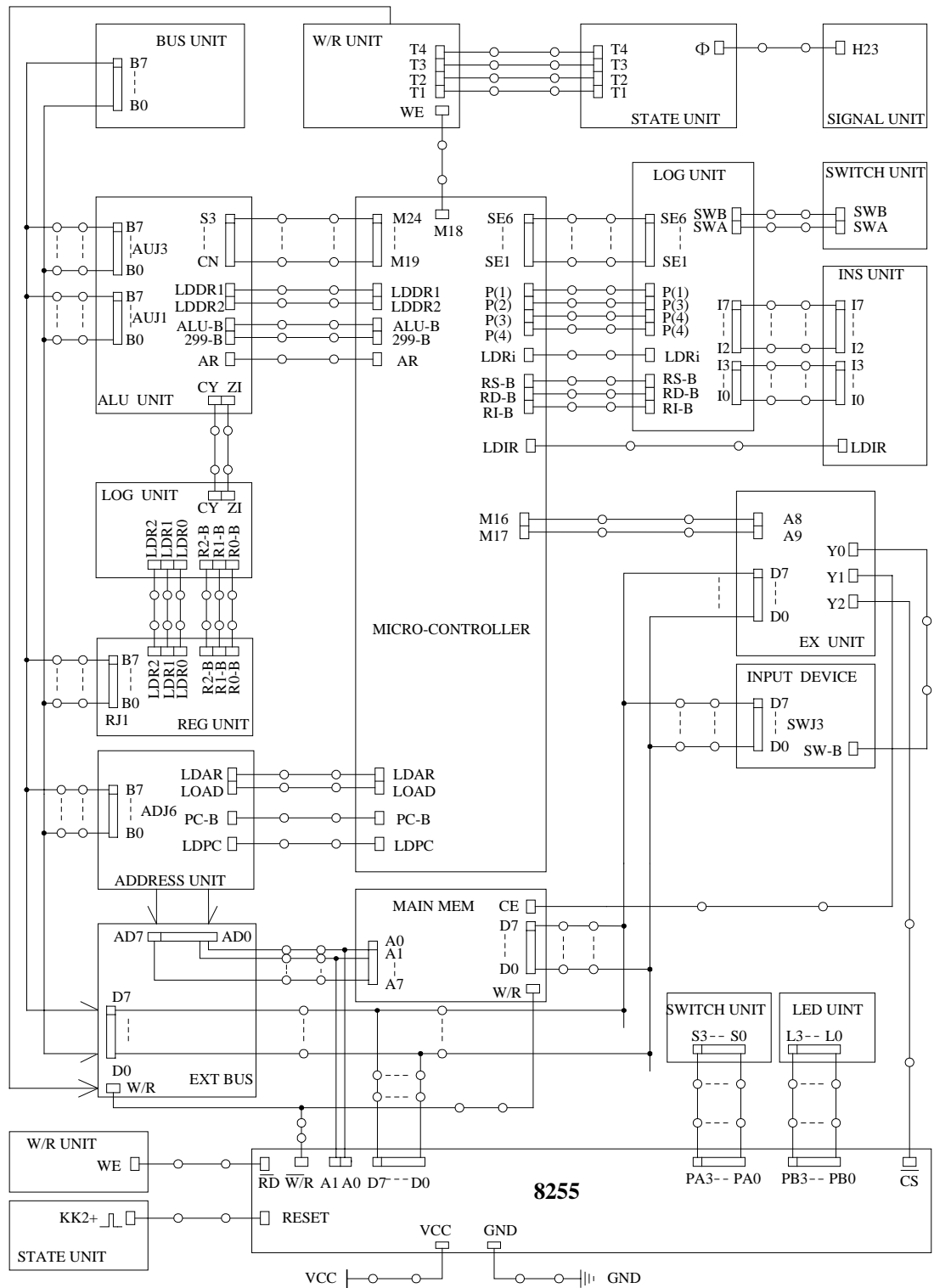


图 7.6-2 8255 实验接线图

\$M24018001	\$M2D65AB6E	\$M360D9A01
\$M2595AAA0	\$M2E0D9A01	\$M3700E038
\$M2600A027	\$M2F01AA30	\$M38011001
\$M2701BC28	\$M300D8171	\$M3900E03A
\$M2895EA29	\$M31959B41	\$M3A030201
\$M2995AAA0	\$M32019A01	\$M3B070A08
\$M2A01B42B	\$M3301B435	\$M3C068A09
\$M2B959B41	\$M3405DB81	
\$M2C01A42D	\$M35B99B41	

3. 根据上述两条端口读写的指令格式，对 8255 芯片编程，编写机器指令程序如下：

地址（二进制）	内容（二进制）	助记符	说明
00000000	01000100	IN R0	“INPUT DEVICE” → R0 输入开关置数 10010000
00000001	11110000	POUT R0, [03]	R0 → 以 03H 为地址的端口
00000010	00000011		
00000011	11100000	PIN [00], R0	将端口 A 的内容读至 R0
00000100	00000000		
00000101	11110000	POUT R0, [01]	将 R0 的内容写至端口 B
00000110	00000001		
00000111	00001000	JMP 03	
00001000	00000011		

其中，第一条指令从输入设备置数 10010000 为将 8255 置成方式 0，A 口输入，B 口输出的控制字。

4. 将机器指令程序写入主存，接线图中以“SWITCH UNIT”单元中的二进制开关作为 8255 的输入口（取四位），“LED UNIT”单元的显示灯作为 8255 的输出口（四位）。给输入开关置任意一数，运行程序，检查输出口显示灯指示是否一致。

实验中接至 KK2 的 RESET 为 8255 的复位信号，高电平有效。它清除控制寄存器和置所有端口到输入方式。运行前应先按动 KK2 使 8255 复位。

7.7 扩展 8253 定时器/计数器实验

一. 实验目的

1. 在构成一台完整模型机的基础上，控制真实的外围接口芯片，进行基本的接口实验。
2. 本实验外扩一片 8253 接口芯片，完成定时器和计数器电路控制实验。

二. 实验设备

- 1. TDN-CM+或 TDN-CM++教学实验系统一台。
- 2. 接口实验板。
- 3. PC 微机（或示波器）一台。

三. 实验原理

1. 8253 芯片引脚特性及外部连接

(1) 8253 的引脚分配图如图 7.7-1 所示

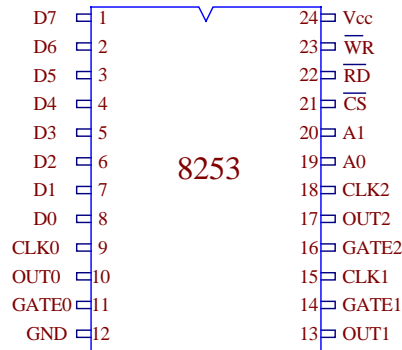


图 7.7-1 8253 芯片引脚说明

(2) 芯片引脚说明

- D7~D0 为数据线
 - \overline{CS} 为片选信号，低电平有效
 - A0、A1 用来选择三个计数器及控制寄存器
 - \overline{RD} 为读信号，低电平有效。它控制 8253 送出数据或状态信息至 CPU。
 - \overline{WR} 为写信号，低电平有效。它控制把 CPU 输出的数据或命令信号写到 8253。
 - CLK_n 、 $GATE_n$ 、 OUT_n 分别为三个计数器的时钟、门控信号及输出端。
- \overline{CS} 、A0、A1、 \overline{RD} 、 \overline{WR} 五个引脚的电平与 8253 操作关系如表 7.7-1 所示：

表 7.7-1

\overline{CS}	\overline{RD}	\overline{WR}	A1	A0	寄存器选择和操作
0	1	0	0	0	写入寄存器#0
0	1	0	0	1	写入寄存器#1
0	1	0	1	0	写入寄存器#2
0	1	0	1	1	写入控制寄存器
0	0	1	0	0	读计数器#0
0	0	1	0	1	读计数器#1
0	0	1	1	0	读计数器#2
0	0	1	1	1	无操作（3 态）
1	X	X	X	X	禁止（3 态）
0	0	1	X	X	无操作（3 态）

2. 本实验所用的指令系统及微程序同 7.6 节（扩展 8255 并行口实验）相同。

四. 实验步骤

1. 如图 7.7-2 搭接实验电路。
2. 本实验只用了计数器#0 通道，将它设置成方波速率发生器。CLK0 接至信号源 H23 上；GATE0=1，计数允许。OUT0 即为方波输出端。
3. 编写一段机器指令如下：

地址（二进制）	内容（二进制）	助记符	说明
00000000	00000000	LAD [0A],R0	0A 单元中内容读至 R0
00000001	00001010		
00000010	11110000	POUT R0, [03]	R0 的内容写至端口 03
00000011	00000011		
00000100	01000100	IN R0	置方波计数值
00000101	11110000	POUT R0, [00]	R0 的内容写至通道#0
00000110	00000000		
00000111	01100000	HALT	停机
00001010	00010110		

其中，0A 单元存放的数 16H 为 8253 的控制字，它的功能为：选择计数器 0，只读/写最低的有效字节，选择方式 3，采用二进制。INPUT DEVICE 单元的开关置的数 N 为计数值，即输出是 N 个 CLK 脉冲的方波。

4. 装载上述机器指令程序及微程序。

5. 连续运行上述程序，用示波器或 CM++ 联机操作软件的示波器功能测 8253 的 OUT0 端波形，INPUT DEVICE 单元的开关置不同的计数值，再重新连续运行机器指令后，观察 OUT0 端输出不同方波的频率。

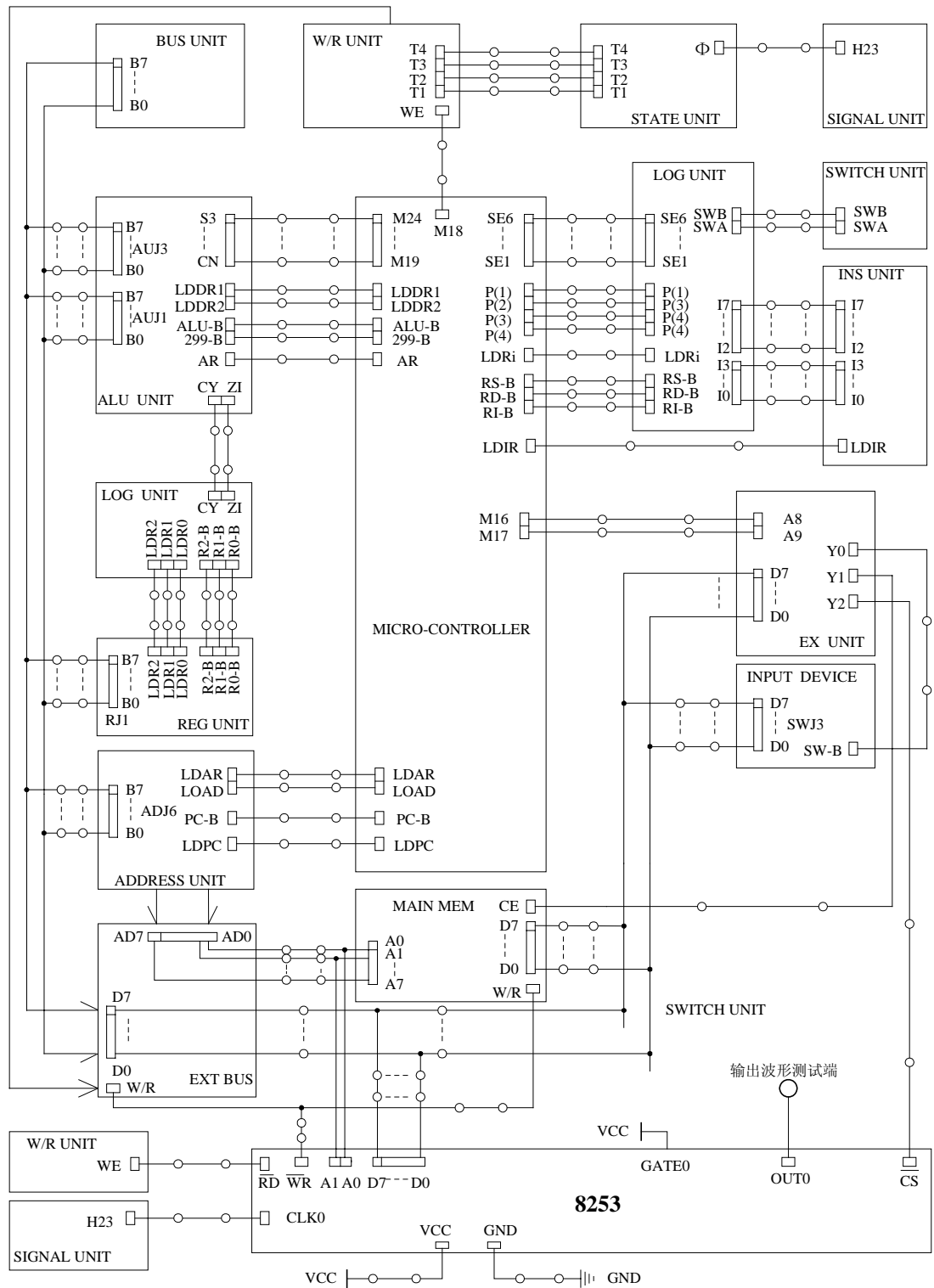


图 7.7-2 8253 接线图

第八章 计算机系统结构的设计及研究

计算机系统结构设计是整个计算机系统设计的一个重要内容，它研究的是软、硬件的功能分配以及如何最佳、最合理地实现分配给硬件的功能。

设计处理机的基本任务之一是要缩短解释指令的时间，即提高处理机指令执行的速度。通常提高指令执行速度的途径有三种：提高处理机的工作频率；采用更好的算法和设计更好的功能部件，如采用 RISC（精简指令系统计算机）技术减少执行指令的平均周期数等；多条指令并行执行。其中多条指令并行执行一般有两种基本方法：一种是采用流水线技术，称为流水线处理机，另一种是在处理机中设置多个独立可并行执行的功能部件，称为超标量处理机。还可以把超流水线技术和超标量技术结合起来称为超标量超流水处理机。

本章分为三个部分，先讨论了 RISC 处理器的基本思想及其设计方法，接下来对于指令并行执行的研究通过采用重叠和流水两种控制方式分别进行讨论。每一个部分后都安排有一个实验进行设计实现。

8.1 精简指令系统计算机

8.1.1 精简指令系统思想的提出

随着计算机技术要求的不断发展，为增强计算机系统的功能，简化编译器的工作量，更好地改善计算机的性能，减少系统的辅助开销，提高计算机的运行速度和效率，计算机结构设计者一直在致力研究为系统结构提供更好的硬件支持。过去的主要做法是：设计包含大量指令的指令系统和各种各样的寻址方式，期望达到使编译器设计者的任务变的容易；提高执行效率，因为复杂操作序列能以微代码实现；提供更复杂、更精致的高级语言的支持。但这样做就会使指令系统变的越来越庞大，这就是所谓的 CISC（复杂指令系统）结构。

经过人们的研究，发现 CISC 的结构和思路存在如下一些问题：

(1) 大量的统计数字表明，大约有 80% 的指令只有在 20% 的处理机运行时间内才被用到。所以对操作繁杂的指令，不仅增加机器设计人员的负担，也降低了系统的性能价格比。

(2) VLSI 技术的飞速发展，VLSI 工艺要求规整性，而 CISC 处理机中，为了实现大量的复杂指令，控制逻辑极不规整，给 VLSI 工艺造成很大的困难。

(3) 由于许多指令的操作繁杂，使得执行速度很低，甚至比用几条简单的指令来组合实现还要慢。而且由于庞大的指令系统，使得难以优化编译生成真正高效率的机器语言程序，也使编译程序本身太长、太复杂。

针对 CISC 结构存在的这些问题，人们提出了 RISC（精简指令系统）的思想。设计了 RISC 机器应当遵循的一般原则，包括：

(1) 确定指令系统时，选取使用频率最高的一些简单指令，以及很有用但不复杂的指令。

(2) 指令长度固定, 指令格式限制在 1~2 种之内。大大减少指令系统的寻址方式, 一般不超过 2 种。

(3) 大部分指令在一个机器周期内完成。

(4) 只有取 (LOAD)、存 (STORE) 指令可以访问存储器, 其他指令的操作一律在寄存器间进行。大大增加寄存器的数量。

(5) 以硬布线控制为主, 很少或不用微程序控制。

(6) 特别重视编译优化工作, 支持高级语言的实现。

8.1.2 RISC 结构采用的基本技术

目前在 RISC 处理机中主要采用如下几种技术:

(1) 延时转移技术

在 RISC 处理机中, 指令一般采用流水线方式工作。取指令和执行指令并行进行。如果取指令和执行指令各需要一个周期, 那么, 在正常情况下, 每个周期就能执行完一条指令。然而, 在遇到转移指令时, 流水就有可能断流。由于转移的目的地址要在指令执行完后才能产生, 这时下一条指令已经取出来了, 因此, 必须把取出来的指令作废, 并按照转移地址重新取出正确指令。为解决上述问题, 可以使编译器自动调整指令序列, 在转移指令后插入一条有效的指令, 而转移指令好象被延迟执行了, 这种技术称为延迟转移技术。

然而必须注意, 调整指令序列时一定不能改变原程序的数据相关关系, 如果找不到合适的指令调整程序中的指令序列, 编译程序可以在转移指令后插入一条空操作指令。

(2) 在处理器中设置数量较大的寄存器组, 并采用重叠寄存器窗口技术

由于在 RISC 程序中有很多的 CALL 和 RETURN 指令。在执行 CALL 指令时, 必须保存现场, 另外, 还要把执行子程序的参数从主程序中传送出去。在执行 RETURN 指令时, 要把保存的结果传送回主程序。为了尽量减少访问存储器, 在 RISC 处理器中采用重叠寄存器窗口技术。

(3) 硬布线实现为主微程序固件实现为辅

主要采用硬布线逻辑来实现指令系统, 对于那些必须的少量的复杂指令, 可以采用微程序实现。微程序便于实现复杂指令, 便于修改指令系统, 增加了机器的灵活性和适应性, 但执行速度低。

(4) 强调优化编译系统设计

编译器必须努力优化寄存器的分配和使用, 提高寄存器的使用效率, 减少访问存储器的次数。为了使 RISC 处理机中的流水线高效率的工作, 尽量不断流, 编译器还必须分析程序的数据流和控制流, 当发现有可能断流时, 要调整指令序列。对有些可以通过变量重新命名来消除数据相关, 要尽量消除。这样, 可以提高流水线的执行效率, 缩短程序的执行时间。

8.2 基于 RISC 处理器构成模型计算机实验

一. 实验目的

1. 了解精简指令系统计算机（RISC）和复杂指令系统计算机（CISC）的体系结构特点和区别。
2. 掌握 RISC 处理器的一般设计原则和指令系统特征。

二. 实验设备

1. TDN-CM++教学实验系统一台。
2. PC 微机一台。

三. 实验原理

（一）本实验中 RISC 处理器指令系统的定义

1. 选用使用频度比较高的五条基本指令：
MOV、ADD、STORE、LOAD、JMP
2. 寻址方式采用寄存器寻址及直接寻址两种方式。
3. 指令格式采用单字长及双字长两种格式：



其中 Rs、Rd 为不同状态，则选中不同寄存器：

Rs 或 Rd	寄存器
0 0	R0
0 1	R1
1 0	R2
1 1	Ac

MOV、ADD、JMP 三条指令为单周期执行完成。

STORE、LOAD 两条指令为两周期执行完成，A 为存或取数的直接地址。第一机器周期完成取操作码、作标记；第二机器周期完成取直接地址并完成取数或存数。

(二) 本实验 RISC 处理器的设计与实现

1. 本处理器的时钟及节拍电位如图 8.2-1 所示
2. 本模型机采用的数据通路图如图 8.2-2 所示:
3. 本通路中除 PLD 单元由 CPLD 来设计, 其他单元全是由板上的单元电路来实现。
4. 指令周期流程图如图 8.2-3 所示:

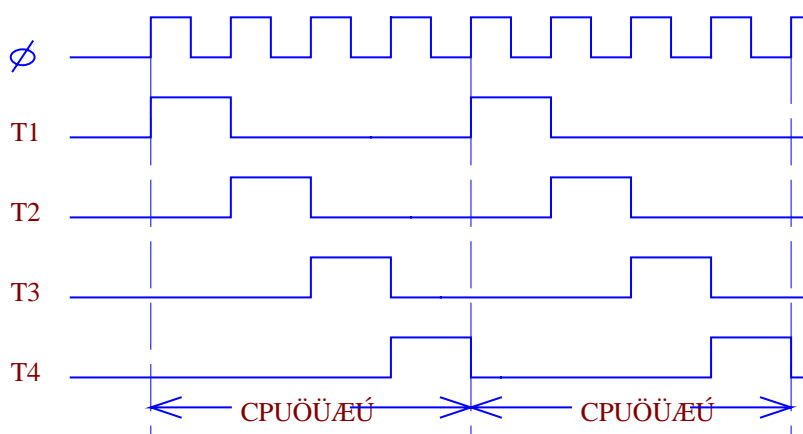


图 8.2-1 时序电路图

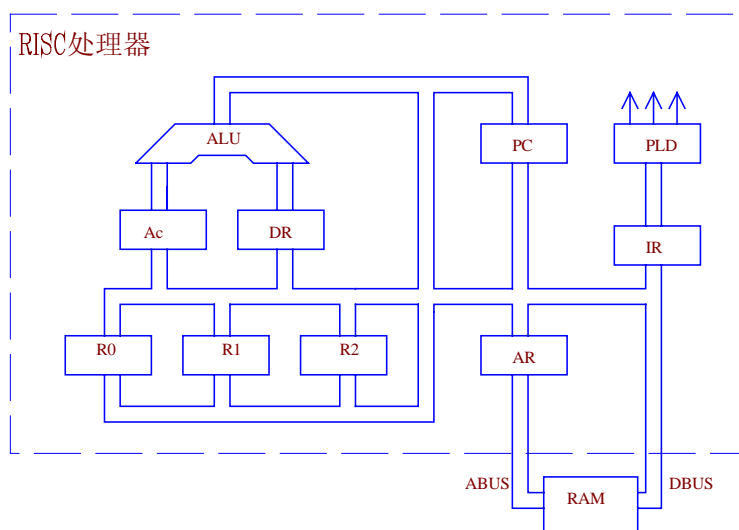


图 8.2-2 数据通路图

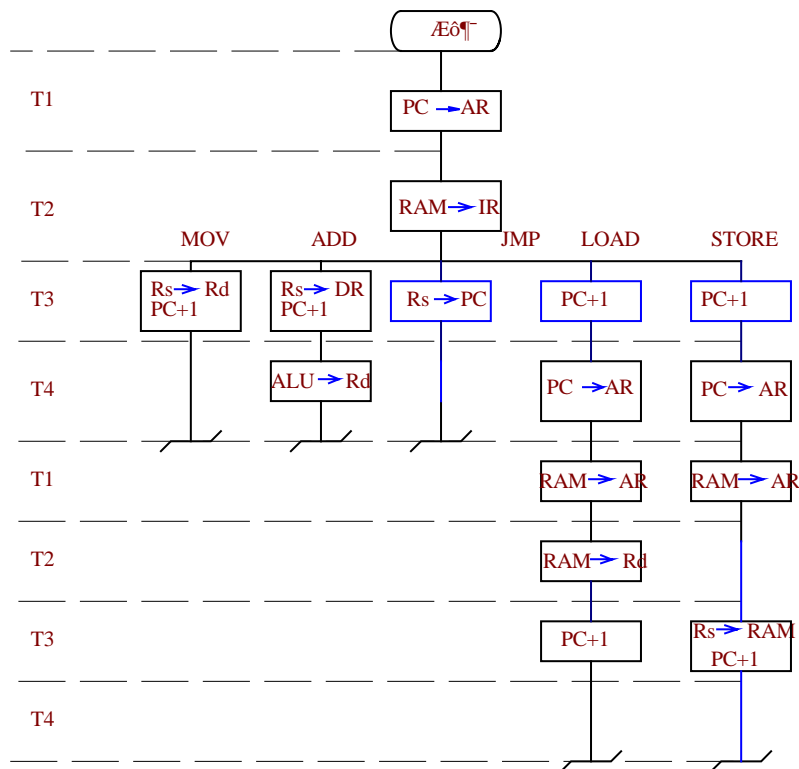


图 8.2-3 指令周期流程图

5. 本实验指令系统如下:

MOV	0 0 0 0	Rs	Rd
-----	---------	----	----

ADD	0 0 0 1	Rs	Rd
-----	---------	----	----

JMP	0 0 1 0	Rs	
-----	---------	----	--

LOAD	0 0 1 1		Rd
A			

STORE	0 1 0 0	Rs	
A			

6. 本实验为方便用实验装置上的显示灯来观察结果, 将 RISC 处理器中的地址寄存器等都是用板上的单元电路来构成, 只将上述数据框图中的 PLD 模块由 CPLD 来实现。只要将时序信号 TS1-TS4 引入 CPLD 既可。将存储器 RAM、CPLD 构成的处理器的外总线都挂至总线单元, 既构成一基本完整的精简指令系统 (RISC) 计算机。

(三) CPLD 芯片设计程序

1. 顶层模块电路图 (top.sch): 见图 8.2-4。
2. 用 ABEL 语言设计 PLD 子模块的功能描述程序。

四. 实验步骤

1. 编译所设计的程序, 将生成的 JED 文件下载至 CPLD 芯片 ispLSI1032 中。
2. 按实验接线图 8.2-5 连接线路。
3. 编写一段机器指令

地址 (H)	内容 (H)	助记符	说明
00	30	LOAD [40], R0	[40]—>R0
01	40		
02	03	MOV R0, Ac	R0—>Ac
03	10	ADD Ac, R0	R0+Ac—>R0
04	40	STORE R0, [0A]	R0—>[0A]
05	0A		
06	30	LOAD [41], R0	[41]—>R0
07	41		
08	20	JMP R0	R0—>PC
40	34		
41	00		

4. 联上 PC 机, 运行 TDN-CM++ 操作软件, 将上述程序写入相应的地址单元中或用 “【转储】—【装载】” 功能将该实验对应的文件载入实验系统。

5. 将时序单元的 STOP 开关拨至 RUN 状态, STEP 开关拨至 STEP 状态, 通过按 START 微动开关, 来单步机器指令, 可通过地址显示灯检查指令执行的地址, 运行完后通过 PC 机联机软件 检查 RAM 中需计算的结果。

实验中若将时序单元 (STATE UNIT) 中的 Φ 接至信号源单元 (SIGNAL UNIT) 的 H23 上则可用来单步执行机器指令操作; 若将 Φ 接至 KK2+ 上则可单步执行每个节拍来调试实验程序, 其方法是将 STEP 开关拨至 EXEC 状态后, 按一下 START 启动键, 然后每按动一次 KK2 键产生一个节拍。

CPLD 接至 SWITCH UNIT 单元中的“CLR”开关是清零开关，它执行标志位、时序及程序计数器清零。它是“0”状态时为清零。

6. 联机运行程序时，进入软件界面，装载机器指令后，选择“【运行】－【通路图】－【RISC 模型机】”功能菜单打开相应动态数据通路图，按相应功能键即可联机运行、监控、调试程序。

五．性能评价

将此 RISC 处理器和前面的由微程序控制模型机实验相比较，明显看出以下优点：

1. 由于指令条数相对较少，寻址方式简单，指令格式规整，控制器的译码和执行硬件相对简单，适合超大规模集成电路实现。

2. 机器执行的速度和效率大大提高。如上面的那段机器指令在本处理器中执行完需 9 个机器周期，而在复杂模型机实验中，需 34 个机器周期才能完成。

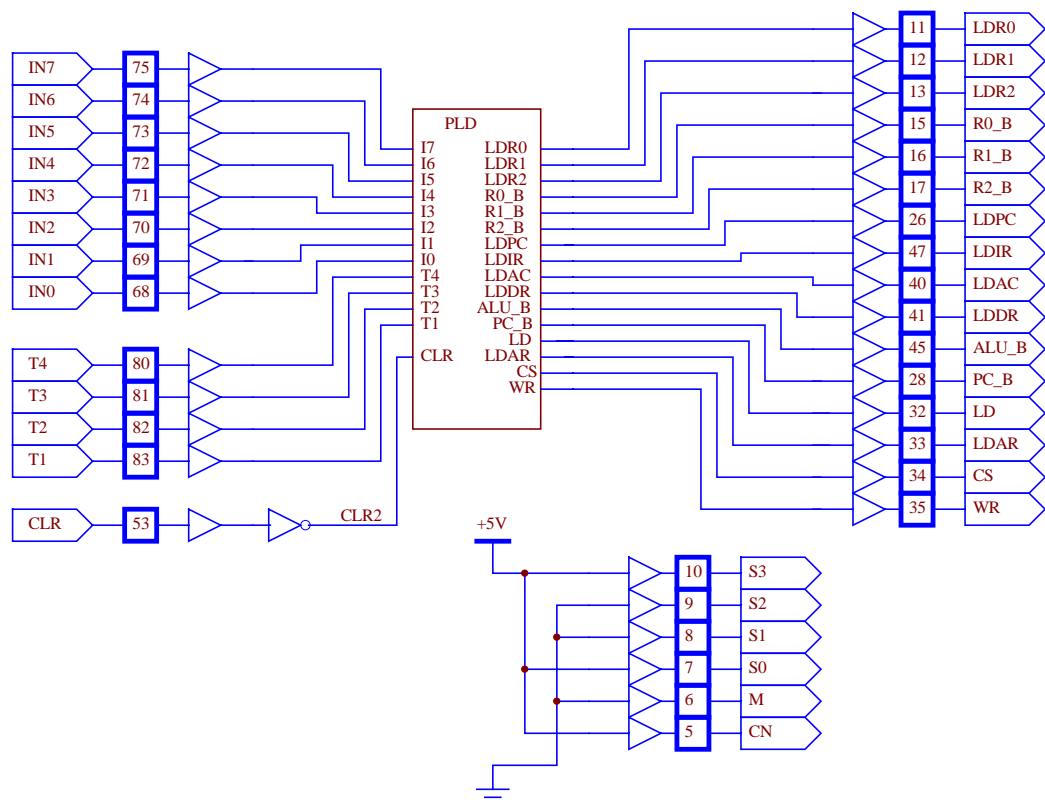


图 8.2-4 CPLD 顶层模块电路图

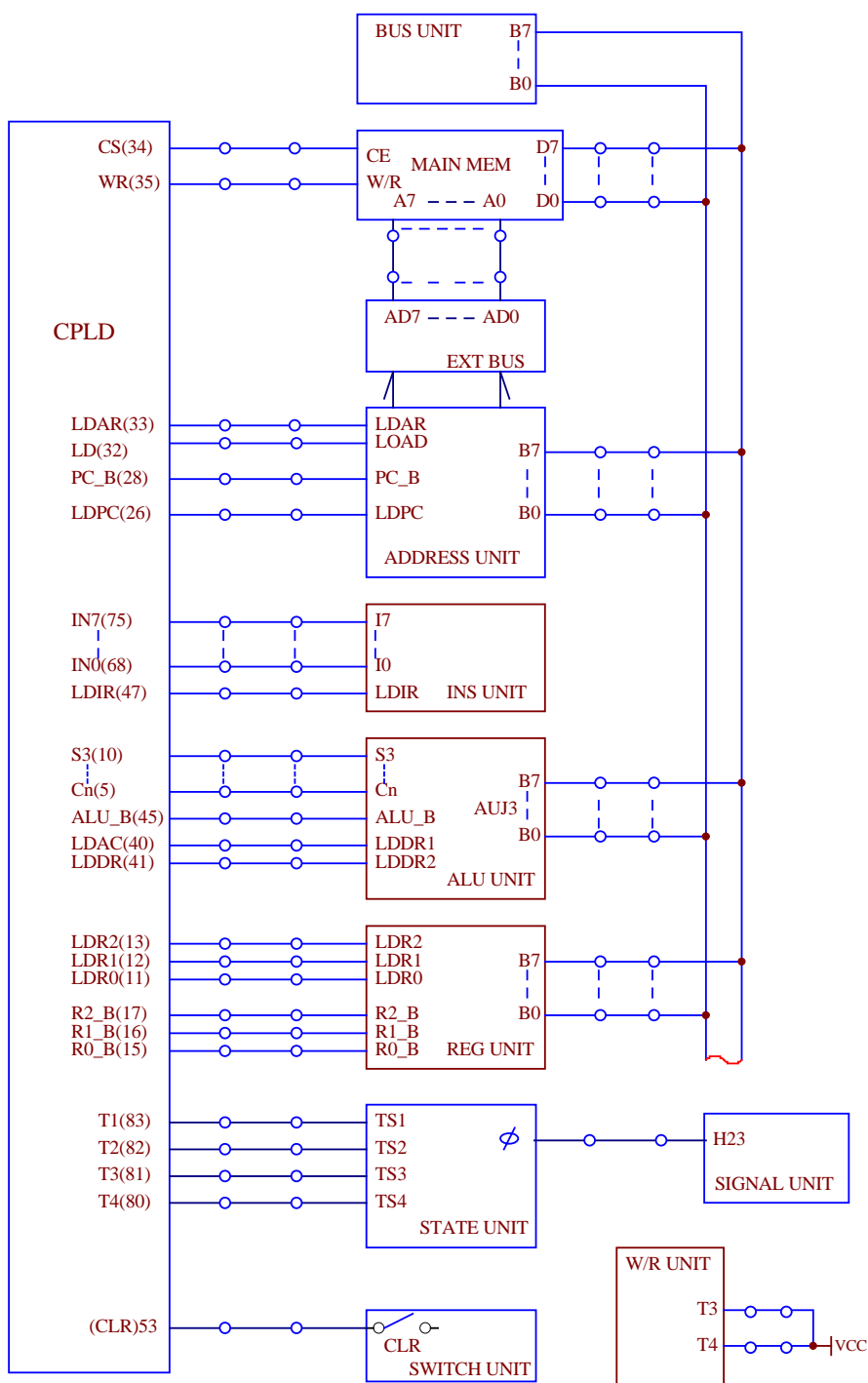


图 8.2-5 实验接线图

8.3 重叠处理机

8.3.1 重叠的原理及基本思想

一条指令的执行过程可以分为多个阶段，一般可以把它们归并为取指令、分析和执行两个阶段。其中，“取指令”就是按指令计数器的内容访问主存储器，取出一条指令送到指令寄存器。“分析和执行”是指对指令的操作码进行译码，按照给定的寻址方式和地址字段中的内容形成操作数的地址，并用这个地址读取操作数，操作数可能在主存储器中，也可能在寄存器中。然后再根据操作码的要求，完成指令规定的功能，在此期间，要把运算结果写到寄存器或主存储器中。

当多条指令要在处理机中执行时，一般有两种执行方式。

1. 顺序执行方式。指各条指令之间顺序串行地执行，执行完一条指令后才取出下一条指令来执行，而且每条指令内部的各个微操作也是顺序串行地执行。

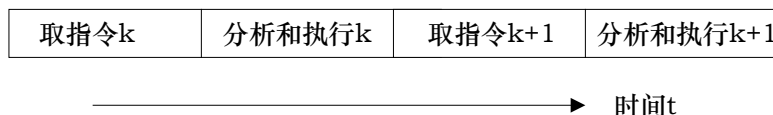


图 8.3-1 指令的顺序执行方式

采用顺序控制方式的优点是控制简单，节省设备。但主要缺点有两个，一个是执行指令速度慢，只有当上一条指令完全执行完后才能开始下一条指令的执行。另一个是功能部件的利用率低。例如在取指和取操作数期间，主存储器是忙碌的，但是运算器却是空闲着；而在对操作数执行运算期间，运算器是忙碌的，主存却空闲着。

2. 重叠执行方式。指在解释第k条指令的操作完成之前，就可开始解释第k+1条指令。如图8.3-2所示。

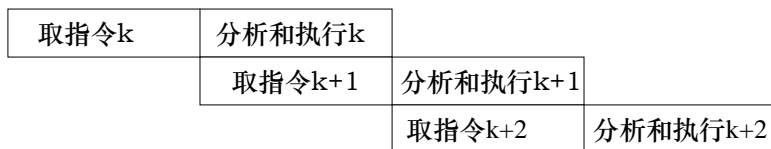


图 8.3-2 重叠执行方式

当然，要使指令能正确地重叠执行，必须解决如下两个问题：

第一，为使取指、分析和执行同时进行，就需要有独立的取指部件、分析和执行部件。

第二，解决访问主存储器冲突问题。在一条指令的执行当中，取指要访问主存，分析中取操作数也可能要访问主存，执行中有可能结果要存入存储器也要访问主存。在一般的机器中，指令和数据是混合放在一个主存储器中的，而且，在一个存储周期中只能访问一个主存单元。所以，需要在硬件上采取和常规存储体系结构不同的办法，一般有三种解决方法。一种是将指令和操作数分别放在两个独立编址且可同时访问的存储器中。这就解决了取指令和

操作数的冲突。但是这种方法主存总线控制十分复杂。另一种是采用多体交叉存储器结构，如果处理机同时执行的取指令和读操作数所访问的不是同一个存储体，则可以实现指令重叠执行，但当两者刚好在同一个存储体中时，就不能重叠执行。第三种是解决存储器冲突的根本方法，即采用先行控制技术，增设指令缓冲寄存器和预处理技术。使用指令缓冲寄存器可以在主存有空时就可以将下一条或几条指令取出来放在指缓中，预取指令的多少取决于指缓的容量。这样取指令操作就不必去访问主存，而是直接从指缓中读取，取指的时间就非常的短了。

8.3.2 相关处理

所谓相关是指在一段程序的相近指令之间有某种关系，这种关系可能影响指令的重叠执行。通常，把相关分为两大类，一类是数据相关，另一类是控制相关。对于重叠处理机，当采用独立的指令预取部件和独立的指令分析和执行部件来实现的话，处理机中的相关问题最主要的也就是控制相关了。

控制相关是指因为程序的执行方向可能改变而引起的相关。可能改变程序执行方向的指令通常有无条件转移、一般条件转移、子程序调用、中断等。下面分别介绍几种转移指令的处理方法。

1. 无条件转移指令

无条件转移指令一般能够在指令分析器中执行完成，形成转移地址送到先行程序计数器 PC1 和 PC 中，指令缓冲栈按照 PC1 的指示重新开始向存储控制器申请取指令。当要转移到的指令不在先行指令缓冲栈中，则要将先行指令缓冲栈中所有预取的指令作废，重新取指令。当要转移的指令在先行指令缓冲栈中时，只要把这条指令之前的预取的指令作废，就可以不用停顿的连续工作。

2. 一般条件转移

对于一般的条件转移指令，如果转移不成功，只要等待一个周期，就可以继续“分析和执行”，在先行指令缓冲栈中预取的指令也仍然有效。如果转移成功，且转移的距离比较近，指令已被取到先行指令缓冲栈中。这时，只要作废本指令之前的所有指令，接着进行分析就可以了。如果转移距离比较远，指令不在先行缓冲栈中，则要将指令缓冲栈的指令全部作废，相当于串行的开始取指令，分析指令。

8.4 基于重叠技术构成的模型计算机实验

一. 实验目的

1. 在基本模型机实验的基础上, 进一步将其构成一台具有重叠功能的模型机。
2. 以原基本模型机的五条机器指令为例, 并编写相应的微程序, 具体上机调试掌握重叠概念。

二. 实验设备

1. TDN-CM++教学实验系统一台。
2. PC 微机一台。

三. 实验原理

1. 实验原理

基本模型机实验过程中, 我们已经了解了在微程序控制下自动产生各部件单元控制信号, 实现特定指令的功能。而在本次试验中, 引入“指令预取”部件 BIU, 使指令预取与指令执行的工作重叠进行。这里, 计算机“执行部件”数据通路的控制仍由微程序控制器来完成, 而“指令预取”部件的数据通路由一片 CPLD 来模拟。“指令预取”部件的内部采用三字节的先进先出栈 (FIFO), 在程序运行过程中, 预取部件将指令从存储器中取到 FIFO 里, 满为止。执行部件从 FIFO 中取得指令, 在预取与执行的过程中互不影响。当执行部件遇到访内的指令时, 先给 BIU 发一个请求信号, BIU 收到该请求信号后, 停止给 FIFO 写数, 这时总线空闲, 执行部件就可以对外部设备进行读写操作。

在如下的数据通路图中, 主要的控制部件是微控器、BIU 控制器及 FIFO 控制器。微控器除了要产生执行部件所需的控制信号外, 还要产生控制指令预取的信号, 如: REQ、WRRD、ALOAD、FRD、LDAR。BIU 控制器根据微控器过来的 LDAR、ALOAD、REQ 及 FIFO 控制器的 FULL 满标志, 产生 FWR、LDPC、LOAD、ACK 信号控制指令预取。FIFO 控制器根据微控器发出的 ALOAD、FRD 和 BIU 控制器的 FWR、ACK 信号对 FIFO 先进先出栈进行读写控制, 本实验中采用 3 字节的 FIFO, 写满之后, FIFO 控制器发 FULL 满信号告诉 BIU 控制器。

FIFO 单元左边有一个三态缓冲器, 由 ACK 和 FRD 信号控制, 当 ACK=1 无访内请求时, 来一个 FRD 信号, 就将一条指令从 FIFO 中读走。FIFO 单元上面是一个双向三态缓冲器, 两端分别连着内数据总线和外数据总线, 由 ACK 和 WRRD 控制, BIU 控制器在收到访内请求 REQ 信号后, 在总线空闲时, 将 ACK 信号置 0, 允许访内, 这时双向三态缓冲器有效, 根据 WRRD 信号对外设进行读写操作。

在数据通路上面有一个多路开关, 它用来控制 AR 地址寄存器中的地址是来自 PC 计数器还是内部总线, 由 LDAR 信号控制。在正常情况下 AR 内容来自 PC, 在遇到间接寻址时, 间接地址放在 FIFO 中, 多路开关就为其提供了通路。

下面以实验程序中的第一条指令结合微程序流程图进行说明, 第一条为 IN 单字节输入指令。控制台操作的前两个机器周期为空操作, 用来向 FIFO 中预取两条指令, 然后转向微

程序入口，以机器周期(T)为序，T1 时，从 FIFO 中将预取的指令码打入 IR 寄存器进行译码，同时将第二条指令的操作数打入 FIFO 中。T2 时转入 IN 指令的执行阶段，IN 是一条访内指令，要用到总线，这时要发请求使用总线的 REQ 信号，等待应答，同时 FIFO 预取第三条指令。T3 时刻 ACK 应答信号已有效，总线开放，从输入单元将一数据通过总线打入 R0 寄存器。此时 FIFO 不能预取指令。以后的微指令不再赘述。请读者自己分析。

本实验采用的机器指令以实验五（基本模型机设计与实现）为例。

2. 数据通路框图如图 8.4-1。微代码定义见表 8.4-1 所示。

表 8.4-1

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
S3	S2	S1	S0	M	Cn	WE	A9	A8	A	B	C	uA5	uA4	uA3	uA2	uA1	uA0						

A 字段

15	14	13	$\tilde{N}_i\hat{O}_n$
0	0	0	
0	0	1	LDRi
0	1	0	LDDR1
0	1	1	LDDR2
1	0	0	LDIR
1	0	1	ALOAD
1	1	0	LDAR

B 字段

12	11	10	$\tilde{N}_i\hat{O}_n$
0	0	0	
0	0	1	RS-B
0	1	0	
0	1	1	
1	0	0	FIFORD
1	0	1	ALU-B
1	1	0	

C 字段

9	8	7	$\tilde{N}_i\hat{O}_n$
0	0	0	
0	0	1	PE'1f©
0	1	0	
0	1	1	
1	0	0	PE'4f©
1	0	1	REQ
1	1	0	

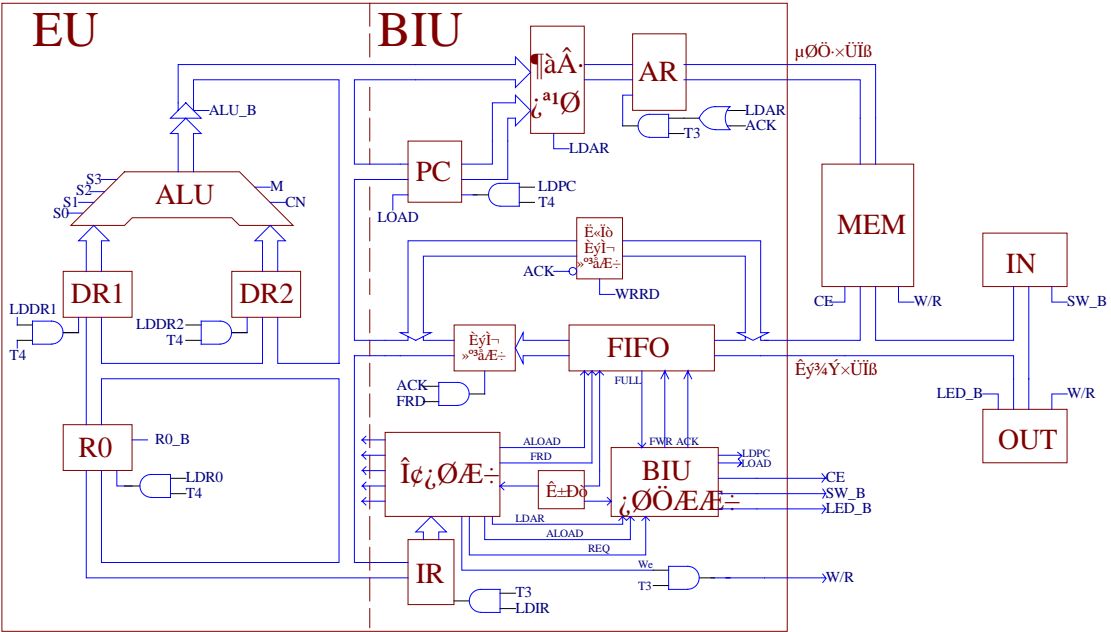


图 8.4-1 数据通路图

3. 处理器的时钟及节拍电位仍由时序电路产生。如图 8.4-2 所示:

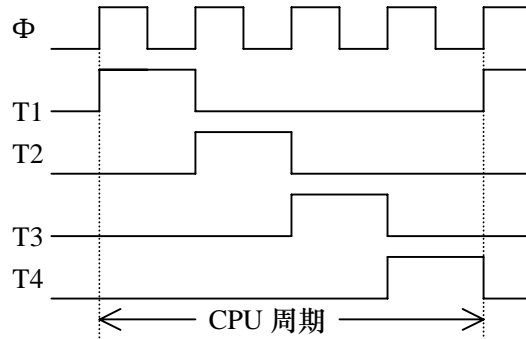


图 8.4-2 时序电路图

4. 本实验系统涉及到的微程序流程见图 8.4-3。

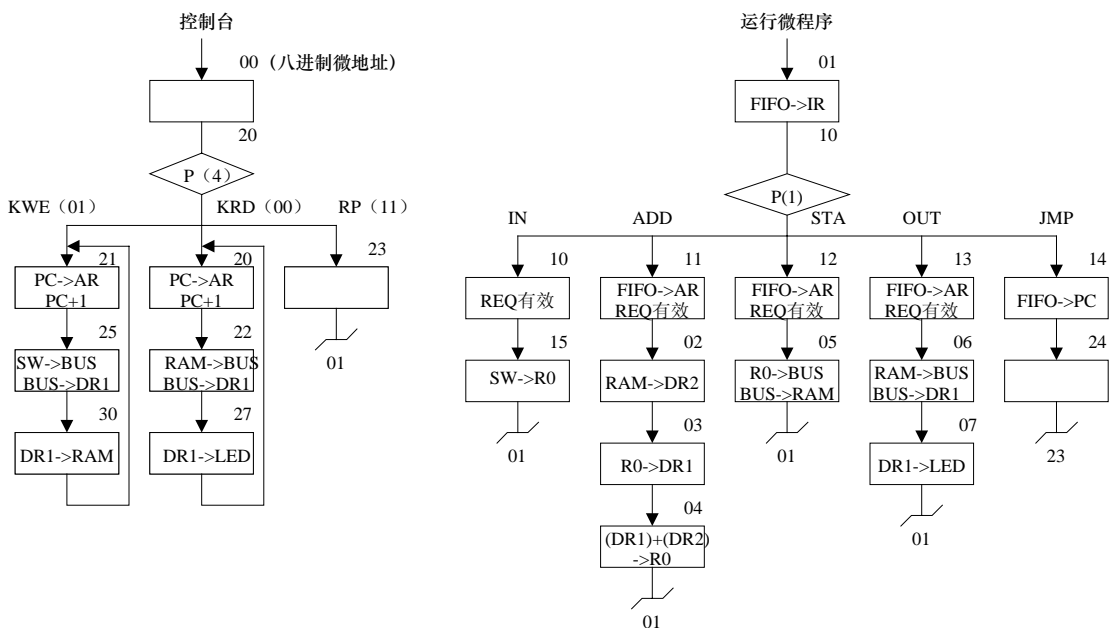


图 8.4-3 微程序流程图

当全部微程序设计完成后, 应将每条微指令代码化, 表 8.4-2 即为将图 8.4-3 的微程序流程图按微指令格式转化而成的“二进制微代码表”。

表 8.4-2 二进制代码表

微地址	S3 S2 S1 S0 M Cn We A9 A8	A	B	C	Ua5 Ua4 Ua3 Ua2 Ua1 Ua0
00	000000 0 1 1	000	000	100	0 1 0 0 0 0
01	000000 0 1 1	100	100	001	0 0 1 0 0 0
02	000000 0 0 1	011	000	000	0 0 0 0 1 1
03	000000 0 1 1	010	001	000	0 0 0 1 0 0
04	100101 0 1 1	001	101	000	0 0 0 0 0 1
05	000000 1 0 1	000	001	000	0 0 0 0 0 1
06	000000 0 0 1	010	000	101	0 0 0 1 1 1
07	000001 1 1 0	000	101	000	0 0 0 0 0 1
10	000000 0 1 1	000	000	101	0 0 1 1 0 1
11	000000 0 1 1	110	100	101	0 0 0 0 1 0
12	000000 0 1 1	110	100	101	0 0 0 1 0 1
13	000000 0 1 1	110	100	101	0 0 0 1 1 0
14	000000 0 1 1	101	100	000	0 1 0 1 0 0
15	000000 0 0 0	001	000	000	0 0 0 0 0 1
20	000000 0 1 1	110	110	110	0 1 0 0 1 0
21	000000 0 1 1	110	110	110	0 1 0 1 0 1
22	000000 0 0 1	010	000	000	0 1 0 1 1 1
23	000000 0 1 1	000	000	000	0 0 0 0 0 1
24	000000 0 1 1	000	000	000	0 1 0 0 1 1
25	000000 0 0 0	010	000	000	0 1 1 0 0 0
27	000001 1 1 0	000	101	000	0 1 0 0 0 0
30	000001 1 0 1	000	101	000	0 1 0 0 0 1

5. 本试验的程序及微程序如下:

程 序	助记符	说明
\$P0000	IN R0	“INPUT DEVICE” → R0
\$P0110	ADD [0A], R0	R0 + [0A] → R0
\$P020A		
\$P0320	STA R0, [0B]	R0 → [0B]
\$P040B		
\$P0530	OUT [0B]	[0B] → LED
\$P060B		
\$P0740	JMP 00	00 → PC

\$P0800

\$P0A01

微程序:

\$M00018110

\$M0A01E945

\$M18068A11

\$M0101C848

\$M0B01E946

\$M17070A10

\$M0200B003

\$M0C01D814

\$M0301A204

\$M0D001001

\$M04959A01

\$M1001ED92

\$M05028201

\$M1101ED95

\$M0604A147

\$M1200A017

\$M07070A01

\$M13018001

\$M0801814D

\$M14018013

\$M0901E942

\$M15002018

四. CPLD 芯片设计程序

1. 在如图 8.4-1 的数据通路图中的如下部分须要在 CPLD 中设计, 见图 8.4-4。

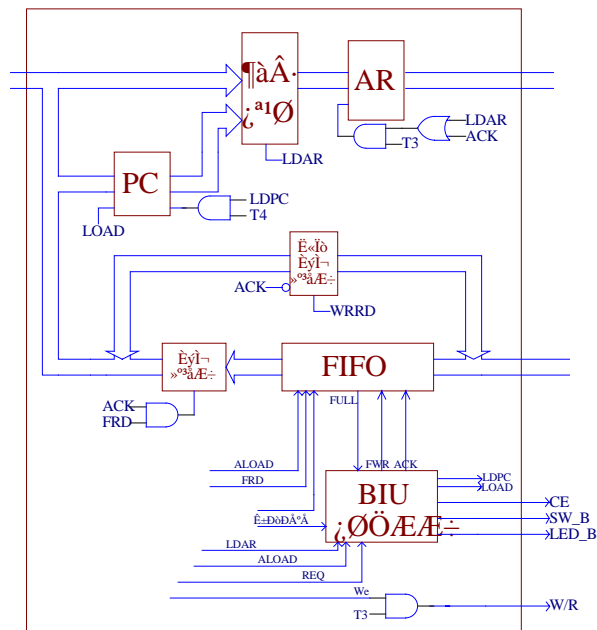


图 8.4-4

2. 在 CPLD 中的顶层模块电路图见图 8.4-5。

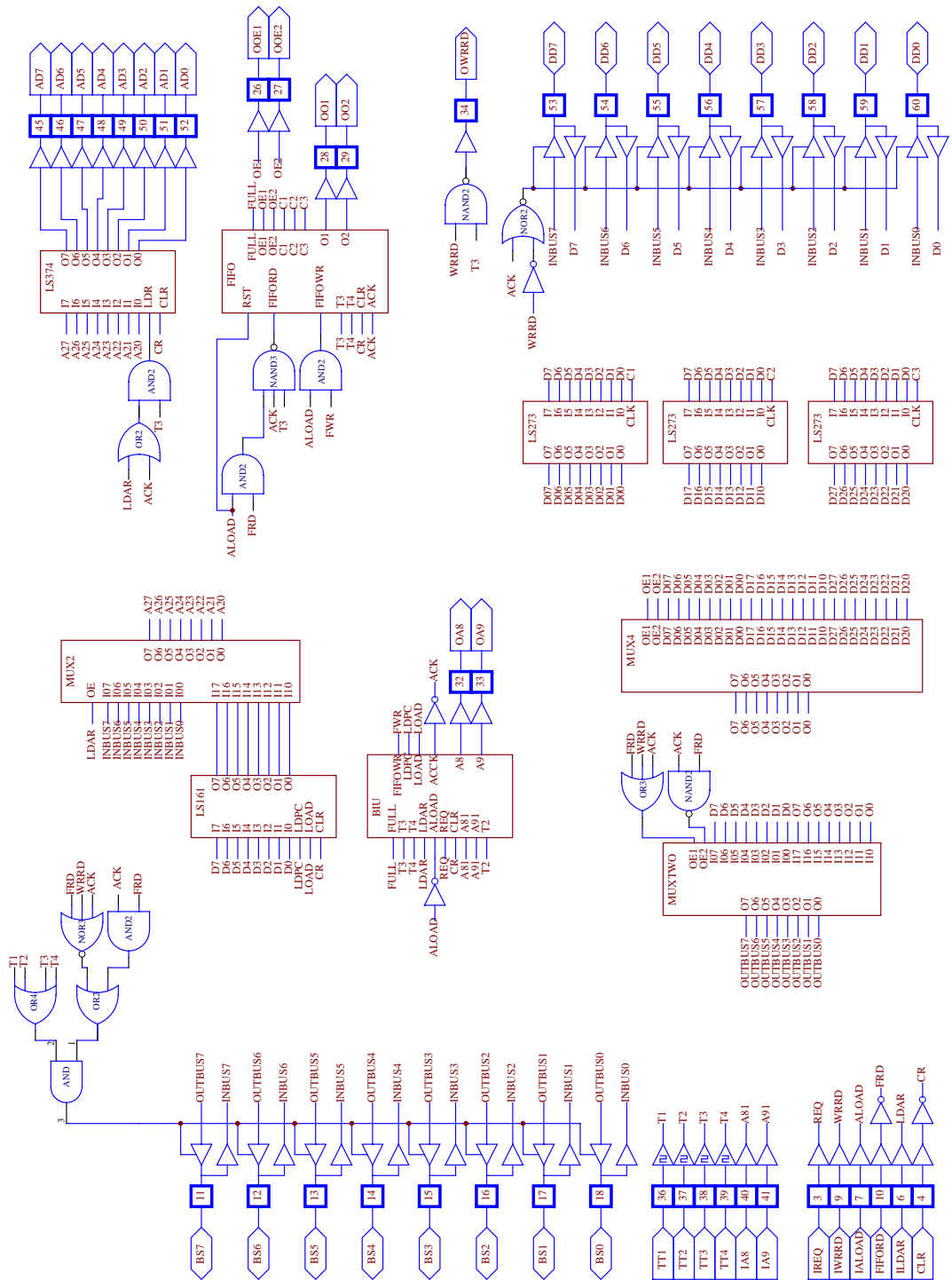


图 8.4-5 重叠实验 CPLD 部分顶层模块电路图

3. 设计各子模块功能描述程序。

五. 实验步骤

(1) 编译上述文件, 并将生成的 JEDEC 文件下载至 Isplsi 1032 芯片中。

(2) 按图 8-2-6 连接实验线路。

(3) 联机写程序

将前面的机器指令及微代码写入到机器的主存及控存中。

(4) 运行程序

1. 脱机运行

实验中若将时序单元 (STATE UNIT) 中的 Φ 接到信号源单元 (SIGNAL UNIT) 的 H23 上, STOP 开关打在 RUN 档, STEP 开关打在 STEP 档, 则可按动 START 按钮来单步执行微指令, 若 STEP 开关打在 EXEC 档, 则可按 START 按钮启动程序连续执行。若将 Φ 接至 KK2 上, STOP 开关打在 RUN 档, STEP 开关打在 EXEC 档, 则可单节拍来调试实验程序。CPLD 接至 SWITCH UNIT 单元中的“CLR”开关是程序计数器及标志清零开关, 同时“CLR”开关也对时序清零, “0”状态时为清零。

① 单步运行程序

A. 使编程开关处于“RUN”状态, STEP 为“STEP”状态, STOP 为“RUN”状态。

B. 拨动总清开关 CLR (1 \rightarrow 0 \rightarrow 1), 微地址清零, 程序计数器清零, 读写计数器清零。程序首地址为 00H。

C. 单步运行一条微指令, 每按动一次 START 键, 即单步运行一条微指令。对照流程图, 观察微地址显示灯是否和流程一致。

D. 连续按动 START 键直至微地址灯为 000111 (07) 状态时, 再按动一次微动开关后, 观察数码块上显示的结果是否和理论值一致 (即开关所置得数和存储器的 0A 单元中的数相加)。

② 连续运行程序

A. “STATE UNIT”中的 STEP 开关置为“EXEC”状态。STOP 开关置为“RUN”状态。

B. 拨动 CLR 开关, 清微地址, 程序计数器及读写计数器, 然后按动 START, 系统连续运行程序。

C. 不断改变“INPUT DEVICE”中的数值, 检查数码块显示结果是否正确。

③ 单拍调试程序

A. “STATE UNIT”中的 STEP 开关置为“EXEC”状态。STOP 开关置为“RUN”状态。

B. 拨动 CLR 开关, 对计数器清零。

C. 然后按动一下 START 按钮将时序准备就绪, 以后每按动一下 KK2 按钮就会发一个 T 脉冲, 在实验系统中一个机器周期分为四拍, 就是说按动四下 KK2 发四个脉冲表示完成一个机器周期。这样可以单拍观察程序的执行情况。

调试情况如下:

第 1 周期: T1: 空操作; T2: 置下一条微指令码; T3: 第一条指令的指令码地址打

入 AR 地址寄存器；T4：测试位改变下一条微指令的地址，将第一条指令的指令码打入 FIFO 中。

第 2 周期：T1：写计数值加 1；T2：置下一条微指令码；T3：第二条指令的指令码地址打入 AR 地址寄存器；T4：将第二条指令的指令码打入 FIFO 中。

第 3 周期：T1：写计数值加 1；T2：置下一条微指令码并且第一条指令的指令码输出到总线，总线显示灯显示数据；T3：总线上的指令码打入 IR 指令寄存器；T4：读计数值加 1，总线上显示 FIFO 中第二条指令的指令码，并将第二条指令的操作数打入 FIFO 中。

第 4 周期：T1：写计数值加 1；T2：置下一条微指令码并发 REQ 总线请求信号；T3：第三条指令的指令码地址打入 AR 寄存器；T4：将第三条指令的指令码打入 FIFO 中。

第 5 周期：T1：写计数值加 1，这时三字节的 FIFO 已写满数据；T2：置下一条微指令码，发 ACK 应答信号，允许使用总线，输入设备置入的数值输出到总线，总线指示灯将显示这一数值；T3：空操作；T4：将总线上的数据打入 R0 寄存器。

第 6 周期：T1：空操作；T2：置下一条微指令码并且第二条指令的指令码输出到总线，总线显示灯显示数据；T3：总线上的指令码打入 IR 指令寄存器；T4：读计数值加 1，总线上显示 FIFO 中第二条指令的操作数，并将第三条指令的操作数打入 FIFO 中。

第 7 周期：T1：写计数值加 1；T2：置下一条微指令码并发 REQ 总线请求信号；T3：将第二条指令的操作数打入 AR 地址寄存器；T4：读计数值加 1，总线上显示 FIFO 中第三条指令的指令码。

第 8 周期：T1：空操作；T2：置下一条微指令码并且在总线上显示存储器中的间接操作数，同时发 ACK 响应信号，允许使用总线；T3：空操作；T4：将总线上的操作数打入 DR2 暂存器暂存。

第 9 周期：T1：空操作；T2：置下一条微指令码并且在总线上显示 R0 中的数据；T3：空操作；T4：将总线上的操作数打入 DR1 暂存器暂存并且将第四条指令的指令码打入 FIFO 中。

第 10 周期：T1：写计数值加 1；T2：置下一条微指令码并且将 DR1 加 DR2 的结果显示在总线上；T3：空操作；T4：将总线上的数据打入 R0 寄存器中。

后面的 10 个机器周期由学生自己分析，并思考以下问题：第 7、8、10 三个周期里为什么没有向 FIFO 预取数据？

2. 联机运行

联机运行程序时，进入软件界面，装载机器指令及微指令后，选择“【运行】—【通路图】—【重叠模型机】”功能菜单打开相应动态数据通路图，按相应功能键即可联机运行、监控、调试程序。此时只用将时序单元（STATE UNIT）中的Φ接到信号源单元（SIGNAL UNIT）的 H23 上，即可在软件中做单节拍的功能调试操作。

总清开关 CLR 清零（1→0→1）后，使程序首址及微程序地址为 00H，程序可从头开始

运行。

六. 性能评测

将此具有重叠功能的模型机实验与前面的基本模型机实验相比较,可明显看出以下优点:

1. 机器执行的速度和效率明显提高。如上面的那段机器指令在本处理器中执行完需要 20 个机器周期,而在基本模型机的实验中,需要 27 个机器周期。
2. 与简单模型机的微指令流程相比,执行部件取指令码时,可以从 FIFO 中得到,不用从存储器中取指,间接寻址时,操作数地址也可以直接从 FIFO 中取得,这样可节省机器的运行周期。
3. 本实验采用重叠方案实现模型机,重叠方案强调了“指令预取”和“指令执行”部件在“时空”上的并行性。具体来说,空间并行性反应在两个工作部件上,而时间并行性的体现是在指令执行过程中,如果指令没有访内的请求,那么“指令预取”部件就将后续的指令码及操作数提前取到 FIFO 中。在重叠方案中,冲突问题是不可避免的,在本实验中主要是解决总线的使用冲突。在这里是通过发送请求 REQ 和应答 ACK 信号来解决冲突问题,指令在经过译码后,已经知道是什么操作,是否要求占用总线访内,如要访内,可发总线请求 REQ 信号给 BIU 控制单元,BIU 控制单元在总线空闲时,发 ACK 应答信号,这时“指令预取”部件不再预取指令,让出总线,这样就避免了冲突的问题。在遇到转移指令时,BIU 单元发 LOAD 装载信号给 PC 计数器,将后续转移地址打入 PC 中并清指令预取队列 FIFO,完成转移。

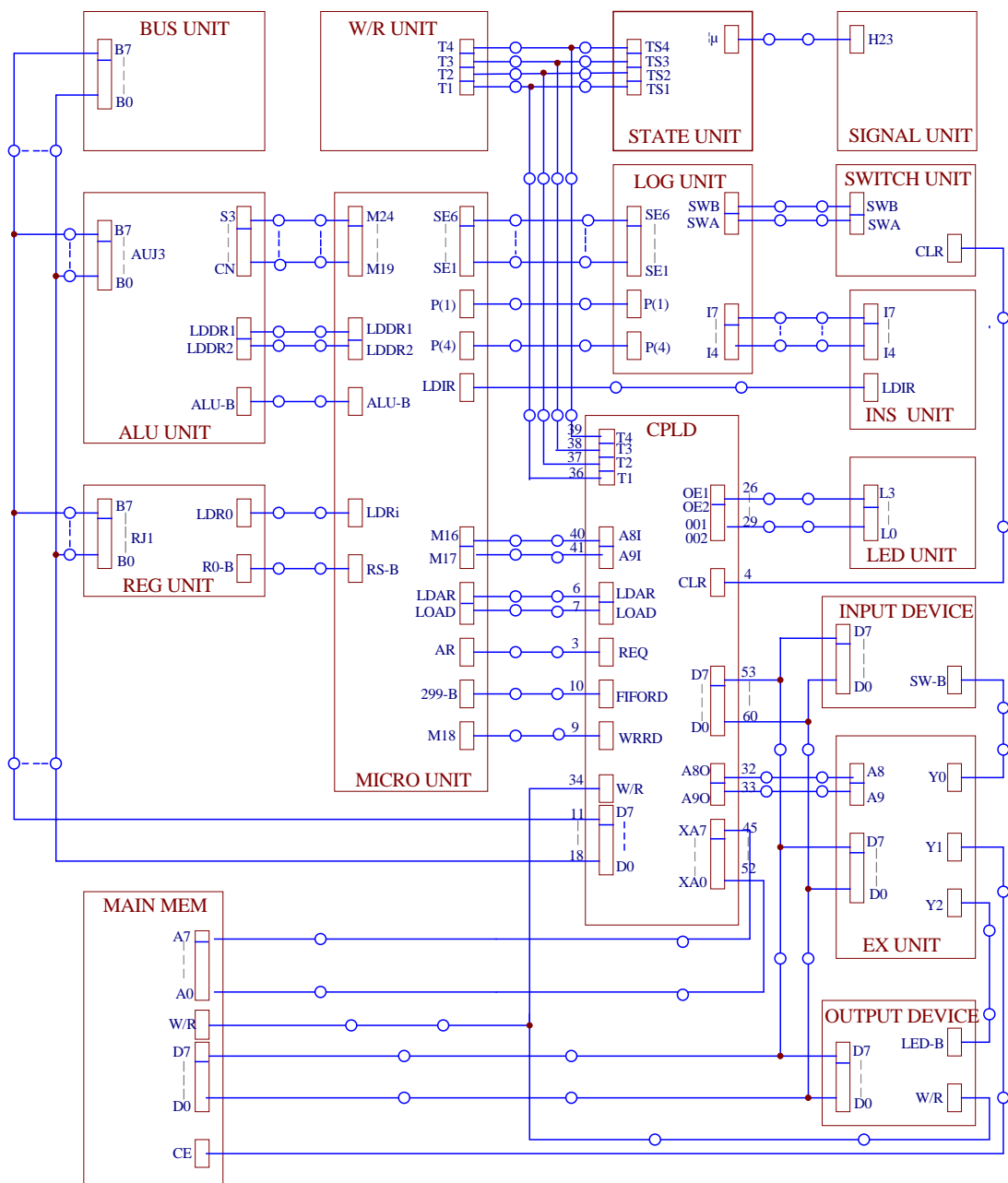


图 8.4-6 重叠实验接线图

8.5 流水线处理机

8.5.1 流水线的原理及基本思想

一. 流水的基本概念

流水方式是把一个复杂的过程分解为若干个子过程，每个子过程可以与其他子过程同时进行。由于这种工作方式与工厂中的生产流水线十分相似，因此，把它称为流水线工作方式。

流水可以看作是重叠的引申，一次重叠是一种简单的指令流水线。一次重叠是把一条指令分解为“分析”和“执行”两个子过程，这两个子过程分别在执行分析部件和指令执行部件中完成。如图 8.5-1 所示。由于在指令分析部件和指令执行部件的输出端各有一个锁存器，可以分别保存指令分析和指令执行的结果，因此，指令分析和指令执行部件可以完全独立并行地工作，而不必等一条指令的“分析”、“执行”子过程都完成之后才送入下一条指令。分析部件在完成一条指令“分析”并将结果送入指令执行部件的同时，就可以开始分析下一条指令。

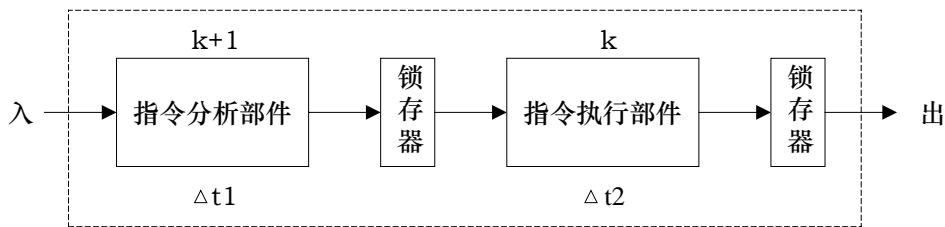


图 8.5-1 简单的流水

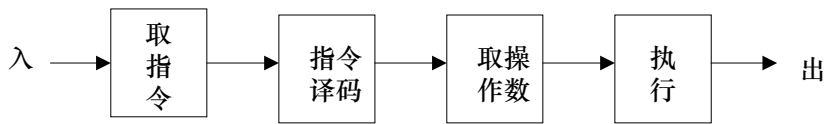
上图中如果指令分析部件分析一条指令所用的时间 Δt_1 与指令执行部件执行一条指令所用的时间 Δt_2 相等，即 $\Delta t_1 = \Delta t_2 = \Delta t$ ，就一条指令的解释来看还是需要 $2\Delta t$ ，但是从机器的输出来看，每过 Δt 就有一条指令执行完成。因此，机器执行指令的速度提高了一倍。

如果把“分析”子过程再细分成“取指令”、“指令译码”和“取操作数”3 个子过程，并加快“执行”子过程，使 4 个子过程都能独立地工作，且经过的时间都是 Δt 。如图 8.5-2 (a) 所示，则可以描述出流水的时空图如图 8.5-2 (a)。

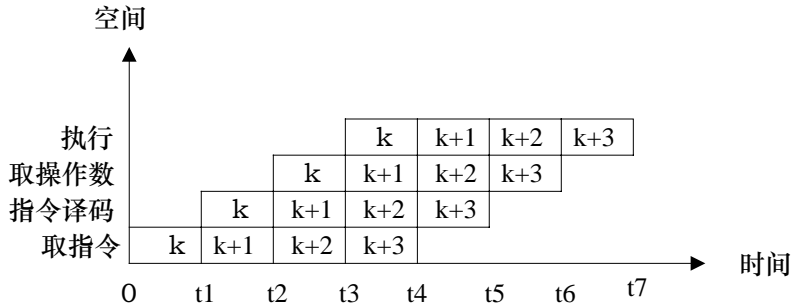
在时空图中，横坐标表示时间，也就是输入到流水线中的各个任务在流水线中所经过的时间。纵坐标表示空间，即流水线的各个子过程。在时空图中，流水线的的一个子过程通常成为“功能段”。

从时空图中，可以很清楚的看出各个任务在流水线的各段中的流动的过程。从横坐标方向看，流水线中的各个功能部件逐个连续地完成自己的任务；从纵坐标看，在同一时间段内有多多个功能段在同时工作。

在上面的流水线中，对于“取指令”、“指令译码”、“取操作数”、“执行”每个子过程都需要 Δt 时间完成，这样，虽然完成一条指令所需的时间还是一个 T ，但是每隔一个 Δt ($T/4$) 时间就会一条指令结果输出，这样的执行效率比顺序方式提高了 3 倍。



(a) 指令流水线



(b) 流水处理的时空图

图 8.5-2 流水处理

二. 流水线的特点

采用流水线方式的处理机与传统的顺序执行方式相比，具有如下特点：

1. 流水线中处理的必须是连续的任务，只有连续不断地提供任务才能发挥流水线的效率。流水线从开始启动到流出第一个结果需要一个“装入时间”，在这段时期内并没有流出任何结果，所以，对第一条指令来说，和顺序执行没有区别。
2. 在流水线每个功能部件的后面都要有一个缓冲寄存器，用于保存本段的执行结果，以保证各部件之间速度匹配及各部件独立并行的运行。
3. 流水线是把一个大的功能部件分解为多个独立的功能部件，并依靠多个功能部件并行工作来缩短程序执行时间。流水线中各段的执行时间应尽量相等，否则将引起“堵塞”、“断流”等。执行时间最长的一段将成为整个流水线的“瓶颈”，在流水线中应尽量解决“瓶颈”。

8.5.2 相关处理

由于流水是同时解释多条指令，肯定会出现更多的相关。

对于转移指令，它和其后的指令之间存在关联，使之不能同时解释，造成对流水线执行方向的改变和效率的下降，被称为全局相关。而指令相关、主存操作数相关、通用寄存器相关及变址相关等只是影响相关附近的几条指令，至多影响流水线的某些段的推后，所以被称为局部相关。

由于流水技术是重叠技术的发展，所以对于局部相关流水通常也是采用两种方法，一种是推后分析法，在遇到数据相关时，推后本条指令的分析，直至所需要的数据写入到相关的存储单元中。另一种方法是设置专用通路，即不必等所需要的数据写入到相关的存储单元中，而是经专门设置的数据通路读取所需要的数据。但由于流水是同时解释更多条指令，所以其

相关专用通路也比重叠的复杂。

对于全局相关，主要是进入流水线的转移指令和它后面的指令之间的相关，通常采用如下几种技术处理：

1. 猜测法

猜测法分软件猜测法和硬件猜测法。由于当转移不成功时，条件转移指令对流水的吞吐率和效率影响比较小。所以，对于软件猜测法，是编译器对源程序进行编译时，要尽量降低转移成功出现的概率。而硬件猜测法，则是当遇到转移指令时，就把转移分支的一个目标地址内容送入指令预取缓冲器，同时保留另一个分支目标地址内容到另一寄存器，以备猜错时恢复原来的程序执行方向。这样，当猜测正确时，转移指令对指令预取的影响很小，指令预取缓冲器仍然有效，可以继续执行；如果猜测不成功，则需要恢复保存下来的另一个分支的内容，清除指令缓冲器中的内容，重新开始取指令。

对于一般的不好猜测其转移成功与否的转移指令，比较有效的方法是采用两个指令预取缓冲栈。当分析到是条件转移指令时，按照转移成功的方向预取指令到一个指令预取缓冲栈中。而原来的仍然按照转移不成功的方向继续预取指令。如果转移不成功，则继续分析原来的指令预取缓冲栈中的指令；如果转移成功，则分析新增设的指令预取缓冲栈中的指令。

2. 加快和提前形成条件码

一方面是加快单条指令内部条件码的形成，这特别适合于转移条件码是由上一条运算型指令产生的情形。（对于有些运算型指令，条件码不一定非要等到执行完毕才得到，完全可以通过参与运算的数来得到，这样可设置专门部件来加速它们的条件码的形成）。另一方面是在一段程序内提前形成条件码，这特别适合于循环型程序在判断循环是否结束时的转移情况。这主要通过编译器将条件码的形成提前到与其不相关的其他指令之前甚至循环体的开始时。

3. 加快短循环程序的处理

短循环程序应是循环体的长度小于等于指令预取缓冲栈的深度，循环次数的控制采用记数转移指令实现，控制循环的条件转移指令一般是向后转移指令。短循环采取的各种措施的基本出发点，一是如何使整个循环程序都放入指令预取缓冲器中，以减少执行循环程序时的访问主存的次数。二是考虑对循环程序出口分支的处理。应尽量使指令缓冲器中的这种循环程序能首尾连接连续流动，连续解释指令。

8. 6 基于流水技术构成模型计算机的实验

一. 实验目的

在掌握 RISC 处理器构成的模型机实验基础上，进一步将其构成一台具有流水功能的模型机。

二. 实验设备

- 1. TDN-CM++教学实验系统一台。
- 2. PC 微机一台。

三. 实验原理

- 1. 本实验中 RISC 处理器指令系统的定义
 - A. 选用使用频度比较高的五条基本指令：
MOV、ADD、STORE、LOAD、JMP
 - B. 寻址方式采用寄存器寻址及直接寻址两种方式。
 - C. 指令格式采用单字长及双字长两种格式：

7	4	3	2	1	0
操作码		Rs		Rd	

7	4	3	2	1	0
操作码		R s		R d	
A					

其中 Rs、Rd 为不同状态，则选中不同寄存器：

Rs 或 Rd		寄存器
0	0	R0
0	1	R1
1	0	R2

Rd		寄存器
0	0	DR1
1	1	DR2

MOV、ADD 两条指令为单周期执行完成。STORE、LOAD、JMP 三条指令为两周期执行完成。在 STORE、LOAD 两条指令里，A 为存或取数的直接地址；在 JMP 指令里，A 为转移地址的立即数。

- 2. 基于 RISC 处理器的流水方案设计原理：

A. 本模型机采用的数据通路图如图 8.6-1 所示:

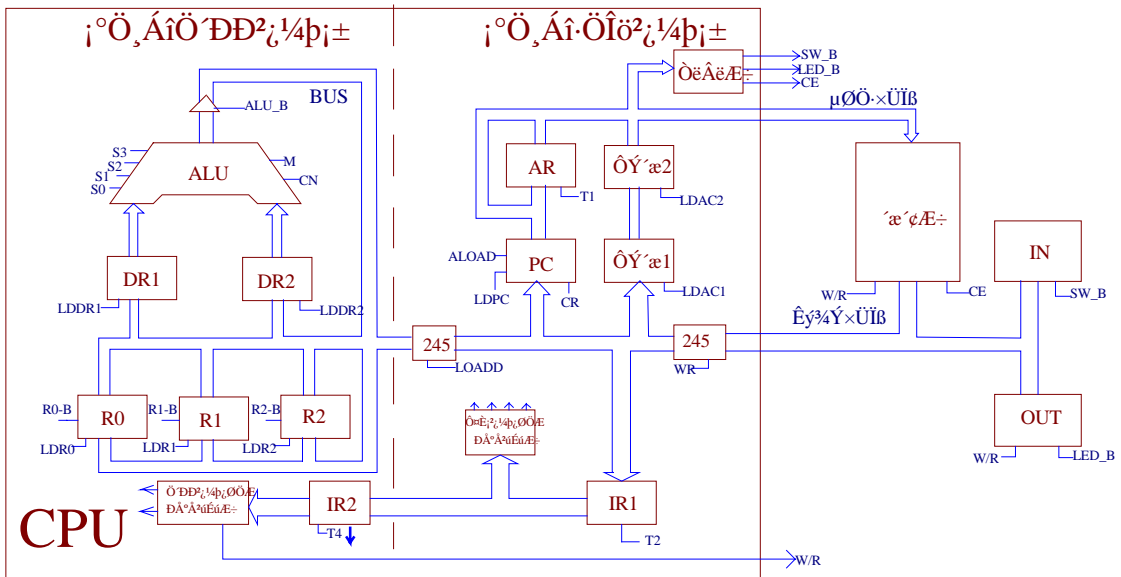


图 8.6-1 流水模型机数据通路图

B. 流水模型机工作原理示意图如图 8.6-2:

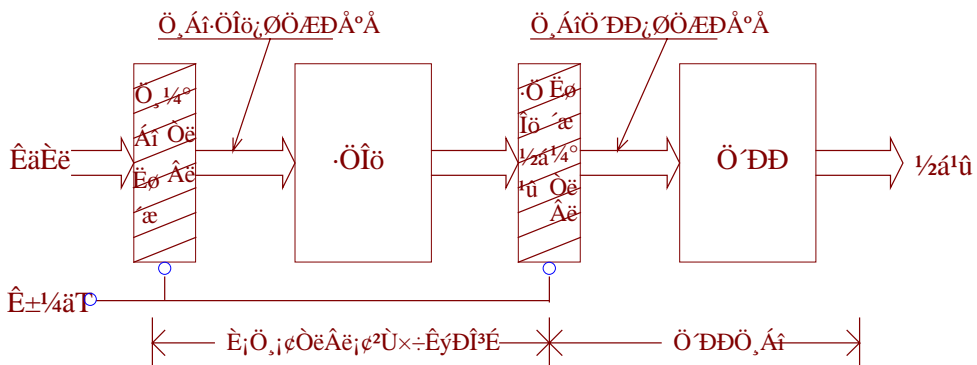


图 8.6-2 流水模型机工作原理示意图

本实验的流水模型机采用两级流水,将系统分为“指令分析部件”和“指令执行部件”,各部件的执行周期均为一个机器周期。如图 14-2 所示:“指令分析部件”主要是取指、译码、操作数形成,IR1 将指令码锁存,译码产生出分析部件所需的控制信号,形成操作数,在机器周期结束时,也就是 T4 的下沿将指令码递推到 IR2 锁存,完成指令的分析。“指令执行部

件”主要负责执行指令，在 IR2 锁存指令码后，就会译码出执行部件需要的控制信号，完成指令的执行。与此同时分析部件完成了下一条指令的分析。以上的过程反应出了流水技术在“时空”上的并行性。除第一个机器周期外，其它周期两个部件都是同时工作的，每一个周期都会有一个结果输出。

“指令分析部件”的设计主要采用了 PC 专用通路和两级暂存技术，PC 专用通路是为访存指令预取操作数地址而用，暂存器是用来暂存操作数地址，设计两级暂存可以避免连续两条访存指令带来的冲突。如果是一级暂存，在分析第一条访存指令时，在 T3 时刻将操作数地址存入暂存。在下一个机器周期里执行该访存指令，同时分析第二条访存指令，第一条访存指令的操作数地址要在 T4 时刻才用到，但是 T3 时刻已经被分析的第二条访存指令的操作数地址复盖，这样就引起了冲突。两级暂存可解决这问题。“指令执行部件”采用实验线路板上的“ALU UNIT”和“REG UNIT”两个单元。

下面介绍一下流水方案的逻辑实现。将一个机器周期分成四个节拍，分别为 T1、T2、T3、T4。首先在 T1 时刻的上沿，程序计数器 PC 将操作码地址打入地址寄存器 AR ($PC \rightarrow AR$)；然后在 T2 时刻的上沿， $PC+1$ 并且将指令的操作码打入指令寄存器；如果是单字节指令，如 MOV、ADD 指令，到此已经完成了指令的预取及分析，如果是双字节指令，如 STORE、LOAD 指令(JMP 指令例外)，在 T3 时刻的上沿选中 PC 专用通路，将操作数地址打入暂存 1 中保存，JMP 指令则将转移地址直接打入 PC 中；在 T4 时刻的上沿， $PC+1$ (JMP 指令则不加 1) 并且将暂存 1 的数据打入暂存 2 中保存；在 T4 的下沿将控制信号锁存。这时双字节指令的预取及分析也完成。

在下一个机器周期的 T4 时刻完成指令的执行。“指令分析部件”同时预取分析下一条指令。

C. 本实验的指令系统如下：

MOV	0000	Rs	Rd
ADD	0001		Rd
JMP	0010		
	A		
LOAD	0011		Rd
	A		
STORE	0100	Rs	
	A		

D. 本实验的程序如下：

地址 (H)	内容 (H)	助记符	说明
--------	--------	-----	----

00	30	LOAD [80], R0	[80H] -> R0
01	80		
02	00	MOV R0, DR1	R0 -> DR1
03	03	MOV R0, DR2	R0 -> DR2
04	10	ADD DR1, DR2, R0	DR1+DR2 -> R0
05	40	STORE R0, [82]	R0 -> [82H]
06	82		
07	20	JMP 00	00H -> PC
08	00		

3. 本实验除“指令执行部件”为板上的“ALU UNIT”和“REG UNIT”电路构成外，其余全部由CM++板上的一片CPLD芯片设计，输入设备、输出设备、RAM及时序仍由板上输入单元、输出显示单元、存储器单元及时序单元电路给出。在本实验的设计中，00H~7FH为存储器地址，80H为输入单元端口地址，82H为输出单元端口地址。

四. CPLD 芯片设计程序

1. 在图 8.6-1 中须用 CPLD 描述的部分见图 8.6-3。

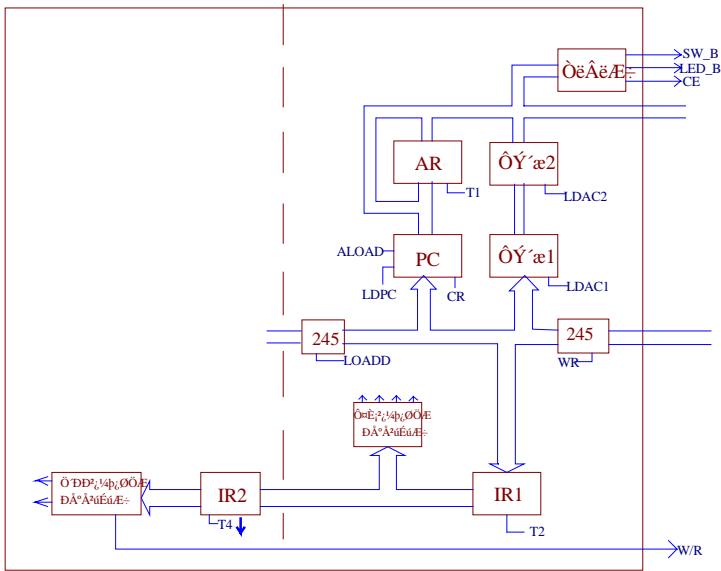


图 8.6-3

2. 顶层模块电路图见图 8.6-4。
3. 设计各子模块功能描述程序。

五. 实验步骤

1. 编译上述所设计的程序，将生成的 JEDEC 文件下载至 1032 芯片中。
2. 按图 8.6-5 连接实验线路，仔细查线无误后接通电源。
3. 向存储器中写入机器程序。
4. 在输入单元 (INPUT DEVICE) 上置一数据，然后拨动 CLR 总清开关 (1 → 0 → 1) 使 PC 清零。
5. 单步或连续运行程序，可以看见输出单元显示输入单元数据乘 2 的结果。
6. 联机运行程序时，进入软件界面，装载机器指令后，选择 “【运行】－【通路图】－【流水模型机】” 功能菜单打开相应动态数据通路图，按相应功能键即可联机运行、监控、调试程序。在本数据通路图中，上位机软件直接可做单节拍操作，以及单周期运行指令、单步机器指令、连续运行等调试操作。
总清开关 CLR 清零 (1→0→1)，使程序首址置为 00H，程序可从头开始运行。

六. 性能评测

1. 本实验在精简指令处理器的基础上以流水方案实现模型机功能，除第一个机器周期预取指令外，其它每个机器周期都有结果输出，与前面的基于 RISC 处理器构成的模型机相比大大提高了执行效率，前面基于 RISC 处理器的实验没有指令预取部件和指令执行部件的概念，在遇到访内指令时它需要两个机器周期才能完成。
2. 本实验流水方案清晰，易于理解。由于该实验是流水的原理性实验，故指令系统也比较简单。

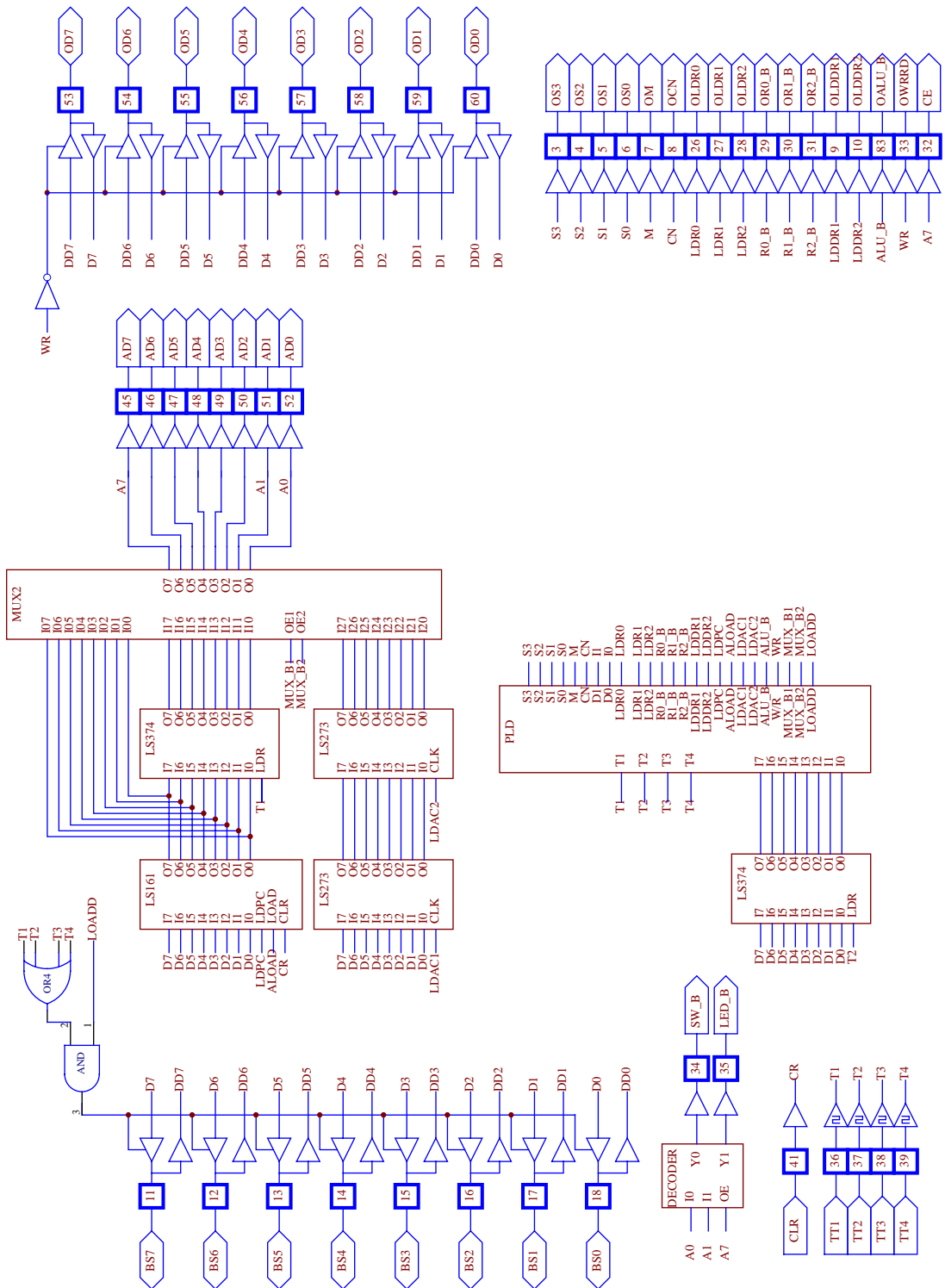


图 8.6-4 在 CPLD 中设计的顶层模块电路图

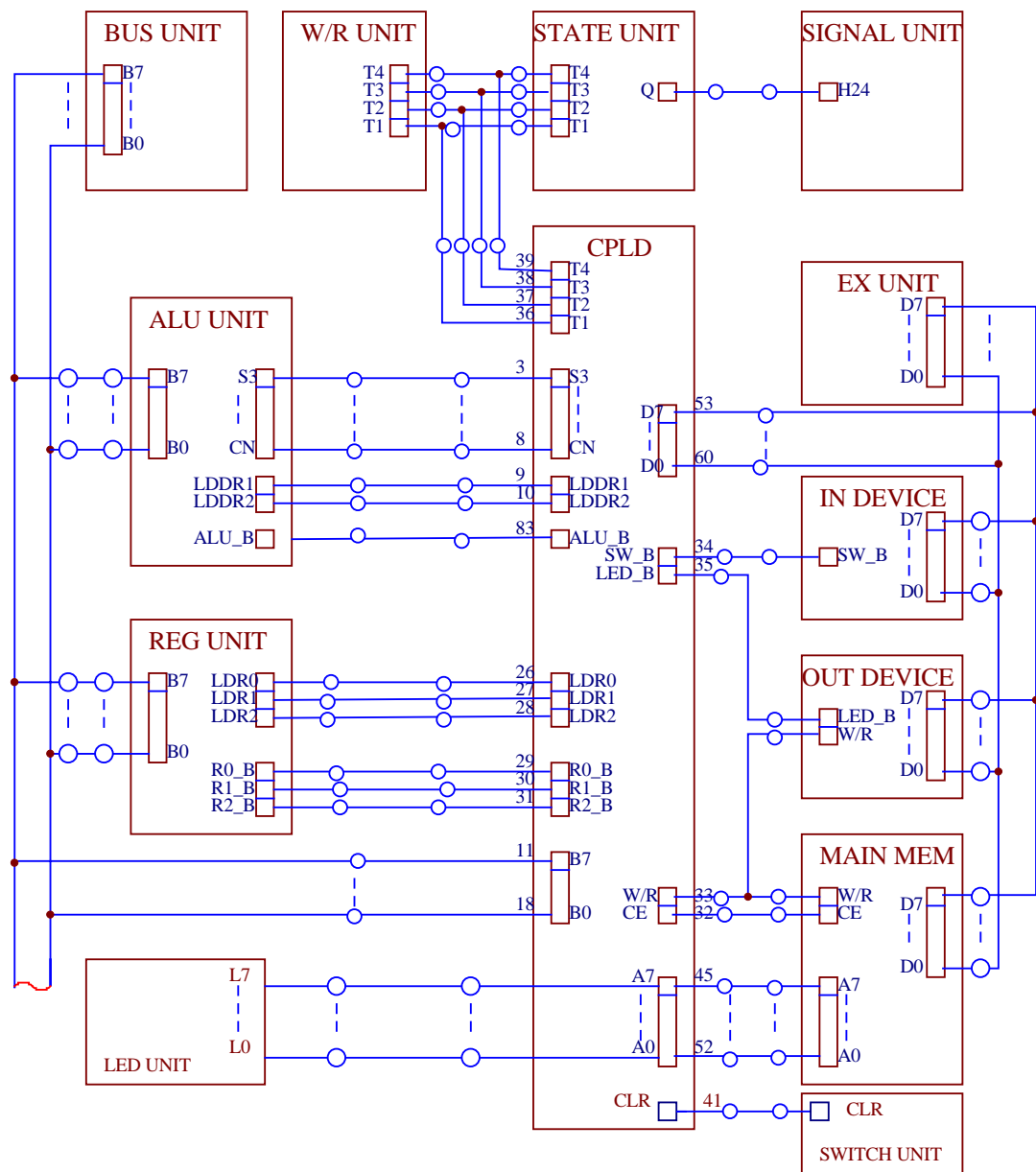
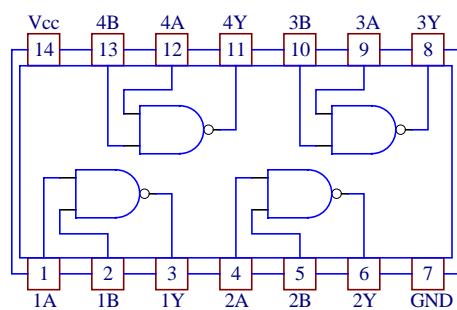


图 8.6-5 流水实验接线图

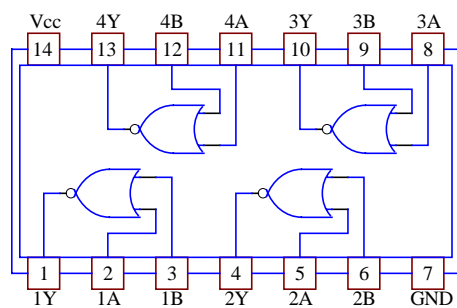
附录 实验用芯片介绍

本附录介绍一些实验电路中用到的中大规模数字功能器件，以供教学实验参考。

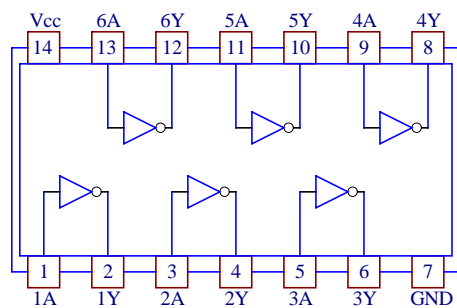
1 . 74LS00



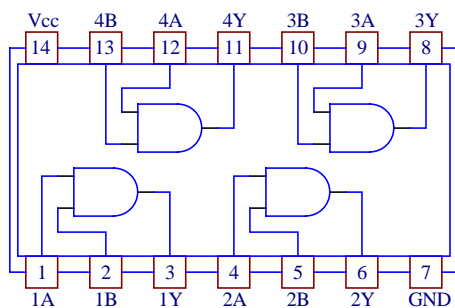
2 . 74LS02



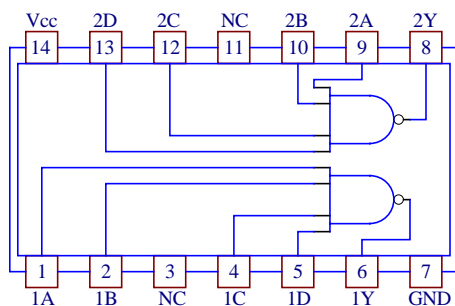
3 . 74LS04



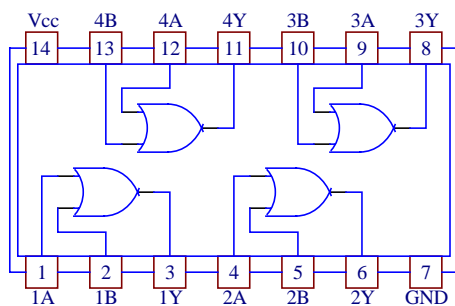
4 . 74LS08



5 . 74LS20

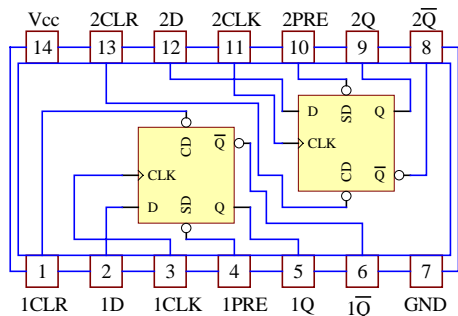


6 . 74LS32



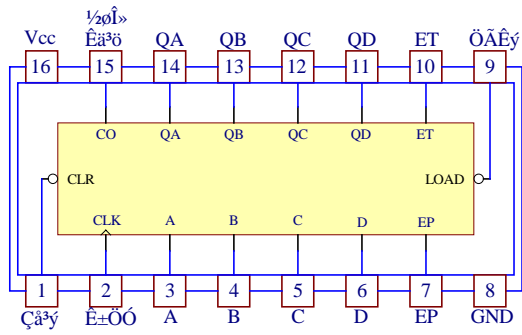
7 . 74LS74

74LS74 功能表



PRC	CLR	CLK	D	Q	\overline{Q}
0	1	×	×	1	0
1	0	×	×	0	1
0	1	×	×	1*	1*
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	×	Q0	$\overline{Q0}$

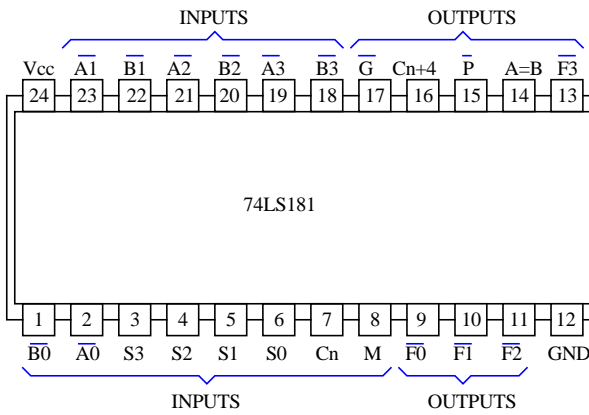
8 . 74LS161



74LS161 工作状态

输 入					输 出					工 作
清除	置数	时钟	使能		QA	QB	QC	QD	进位输出	
			EP	ET						
H	H	↑	H	H	—	—	—	—	—	计数
H	L		X	X	A	B	C	D	—	数据预置
↓ L	X	X	X	X	L	L	L	L	—	清除
H	X	X	X	H	H	H	H	H		—

9 . 74LS181



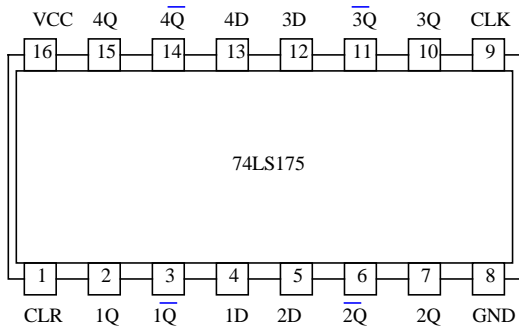
74LS181 功能表

输入为 A 和 B，输出为 F，为正逻辑。

S3 S2 S1 S0	M=0(算术运算)		M=1(逻辑运算)
	Cn=1(无进位)	Cn=0(有进位)	
0 0 0 0	$F=A$	$F=A$ 加 1	$F=\overline{A}$
0 0 0 1	$F=A+B$	$F=(A+B)$ 加 1	$F=\overline{A+B}$
0 0 1 0	$F=A+\overline{B}$	$F=(A+\overline{B})$ 加 1	$F=\overline{A}B$
0 0 1 1	$F=0$ 减 1	$F=0$	$F=0$
0 1 0 0	$F=A$ 加 \overline{AB}	$F=A$ 加 \overline{AB} 加 1	$F=\overline{AB}$
0 1 0 1	$F=\overline{AB}$ 加 $(A+B)$	$F=\overline{AB}$ 加 $(A+B)$ 加 1	$F=\overline{B}$
0 1 1 0	$F=A$ 减 B 减 1	$F=A$ 减 B	$F=A \oplus B$
0 1 1 1	$F=\overline{AB}$ 减 1	$F=\overline{AB}$	$F=\overline{AB}$
1 0 0 0	$F=A$ 加 AB	$F=A$ 加 AB 加 1	$F=\overline{A}+B$
1 0 0 1	$F=A$ 加 B	$F=A$ 加 B 加 1	$F=\overline{A} \oplus B$
1 0 1 0	$F=AB$ 加 $(A+\overline{B})$	$F=AB$ 加 $(A+\overline{B})$ 加 1	$F=B$
1 0 1 1	$F=AB$ 减 1	$F=AB$	$F=AB$
1 1 0 0	$F=A$ 加 A	$F=A$ 加 A 加 1	$F=1$
1 1 0 1	$F=A$ 加 $(A+B)$	$F=A$ 加 $(A+B)$ 加 1	$F=A+\overline{B}$
1 1 1 0	$F=A$ 加 $(A+\overline{B})$	$F=A$ 加 $(A+\overline{B})$ 加 1	$F=A+B$
1 1 1 1	$F=A$ 减 1	$F=A$	$F=A$

10. 74LS175

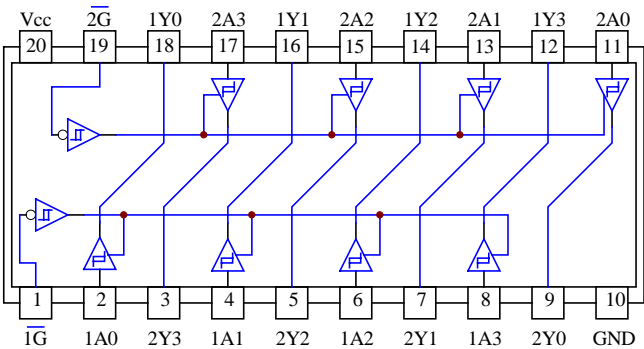
74LS175 功能表



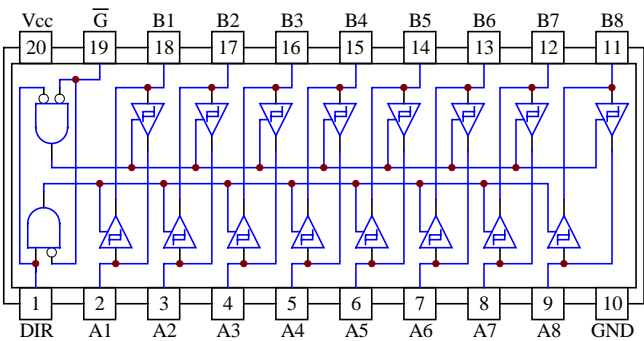
CLR	CLK	D	Q	\overline{Q}
0	×	×	0	1
1	↑	1	1	0
1	↑	0	0	1
1	0	×	Q0	$\overline{Q0}$

Q0= 在时钟脉冲的上升沿之前 Q 的输出

11. 74LS244



12. 74LS245

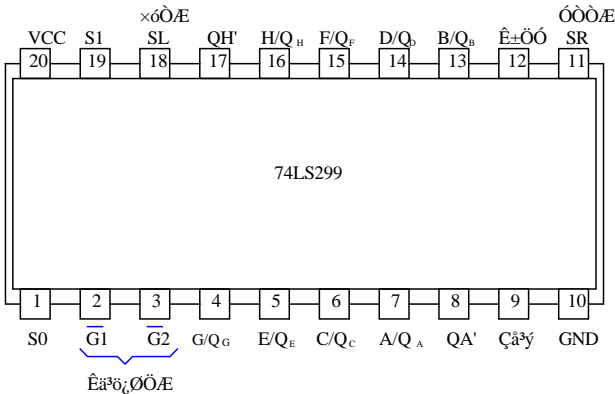


74LS245 功能表

使能	方向控制	操 作
G	DIR	
L	L	B 数据至 A 总线
L	H	A 数据至 B 总线
H	X	隔 开

注：H=高电平，L= 低电平，X=不定

13. 74LS299



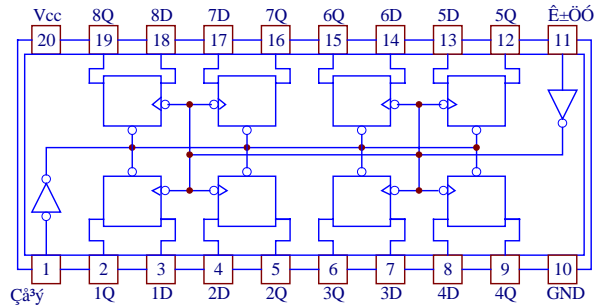
74LS299 功能表

模式	输入							输入/输出								输出 Q _{A'} Q _{H'}		
	清除	功能选择		输出控制		时钟	串入		A/Q _A B/Q _B C/Q _C D/Q _D E/Q _E F/Q _F G/Q _G H/Q _H									
		S1	S0	$\overline{G1}$	$\overline{G2}$		SL	SR										
清除	L	X	L	L	L	X	X	X	L	L	L	L	L	L	L	L	L	L
	L	L	L	L	L	X	X	X	L	L	L	L	L	L	L	L	L	L
保持	H	L	L	L	L	X	X	X	Q _{A0} A _{B0} Q _{C0} Q _{D0} Q _{E0} Q _{F0} Q _{G0} Q _{H0}									Q _{A0} Q _{H0}
	H	X	X	L	L	L	X	X	Q _{A0} A _{B0} Q _{C0} Q _{D0} Q _{E0} Q _{F0} Q _{G0} Q _{H0}									Q _{A0} Q _{H0}
右移	H	L	H	L	L	↑	X	H	H	Q _{An} A _{Bn} Q _{Cn} Q _{Dn} Q _{En} Q _{Fn} Q _{Gn}							H	Q _{Gn}
	H	L	H	L	L	↑	X	L	L	Q _{An} A _{Bn} Q _{Cn} Q _{Dn} Q _{En} Q _{Fn} Q _{Gn}							L	Q _{Gn}
左移	H	H	L	L	L	↑	H	X	A _{bn} Q _{Cn} Q _{Dn} Q _{En} Q _{Fn} Q _{Gn} Q _{Hn} H								Q _{Bn} H	
	H	H	L	L	L	↑	L	X	A _{Bn} Q _{Cn} Q _{Dn} Q _{En} Q _{Fn} Q _{Gn} Q _{Hn} L								Q _{Bn} L	
置数	H	H	H	X	X	↑	X	X	a	b	c	d	e	f	g	h	a	h

注：当输出控制 G1 或 G2 任意一个或两个为高时，八个输入/输出端禁止，为高阻态；
但寄存器的时序工作和清除功能不受影响。

14. 74LS273

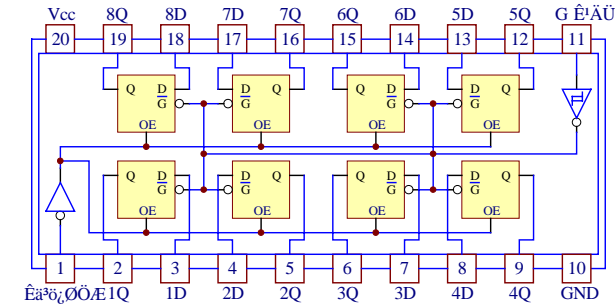
74LS273 功能表



输入			输出 Q
清除	时钟	D	
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q ₀

15. 74LS373

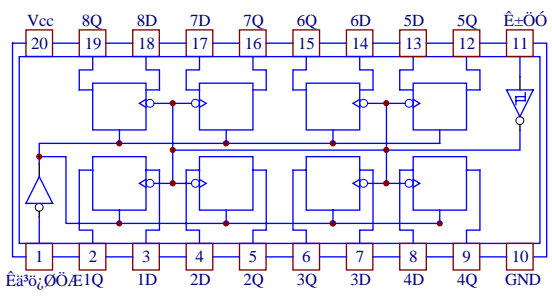
74LS373 功能表



输出 控制	输入		输出
	G	D	
L	H	H	H
L	H	L	L
L	L	X	Q ₀
H	X	X	Z

16. 74LS374

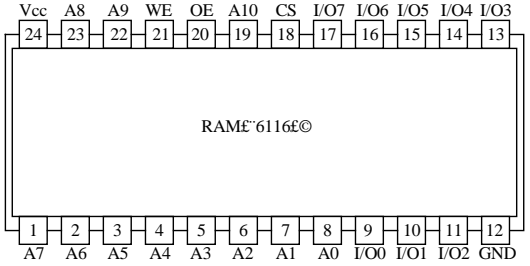
74LS374 功能表



输出 控制	G	D	输出
L	↑	H	H
L	↑	L	L
L	L	X	Q ₀
H	X	X	Z

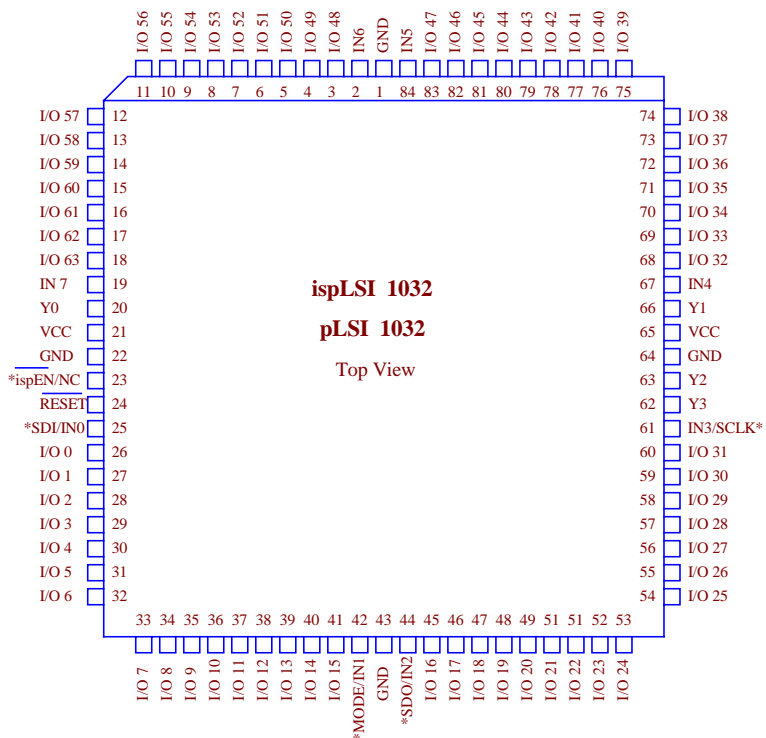
17. 6116

6116 功能表



CS	WE	OE	功能
1	×	×	不选择
0	1	0	读
0	0	1	写
0	0	0	写

18. LATTICE 1032 84-PIN PLCC 封装



参考文献

- [1] 俸远祯, 阎慧娟, 罗克露。计算机组成原理 (修订本), 电子工业出版社, 1996
- [2] 白中英。计算机组成原理, 科学出版社, 1994
- [3] 孟传良, 张庸一。计算机组成原理», 重庆大学出版社, 2002
- [4] 张代远。计算机组成原理, 北京邮电大学出版社, 2002
- [5] 朱怡健, 朱敏, 王健。计算机组成原理, 东南大学出版社, 1994
- [6] 李文兵。计算机组成原理。清华大学出版社, 2002.6
- [7] 蒋本珊。计算机组织与结构。清华大学出版社, 2002.3
- [8] 张新荣, 刘锋, 杨洁, 张钢。计算机组成原理教程。希望电子出版社, 2002.7
- [9] 李恒甫。计算机系统结构。重庆大学出版社, 2001.11
- [10] 徐福培, 袁春风, 鲍培明等。计算机组成与结构。电子工业出版社, 2001.9
- [11] 幸云辉, 杨旭东。计算机组成原理实用教程。清华大学出版社, 2001.9
- [12] 王爱英。计算机组成与结构。清华大学出版社, 2001.2
- [13] 胡越明。计算机组成和系统结构。上海科学技术文献出版社, 1999.3
- [14] 石教英。计算机体系结构。浙江大学出版社, 1998.10
- [15] 王闵。计算机组成原理。电子工业出版社, 2001.1
- [16] 唐朔飞。计算机组成原理。高等教育出版社, 2000.7
- [17] 顾一禾, 朱近, 路一新。计算机组成原理自学考试指导, 清华大学出版社, 2002
- [18] 李学干, 苏东庄。计算机系统结构 (第二版), 西安电子科技大学出版社, 1991
- [19] 郑纬民, 汤志忠, 计算机系统结构 (第二版), 清华大学出版社, 1998
- [20] Williamm Stallings 著、张困藏译。计算机组织与结构—性能设计 (第五版), 电子工业出版社, 2001
- [21] 刘笃仁、杨万海。在系统可编程技术及其器件原理与应用, 西安电子科技大学出版社, 1999
- [22] 杨晖, 张凤言。大规模可编程逻辑器件与数字系统设计。北京航空航天大学出版社, 1998.7