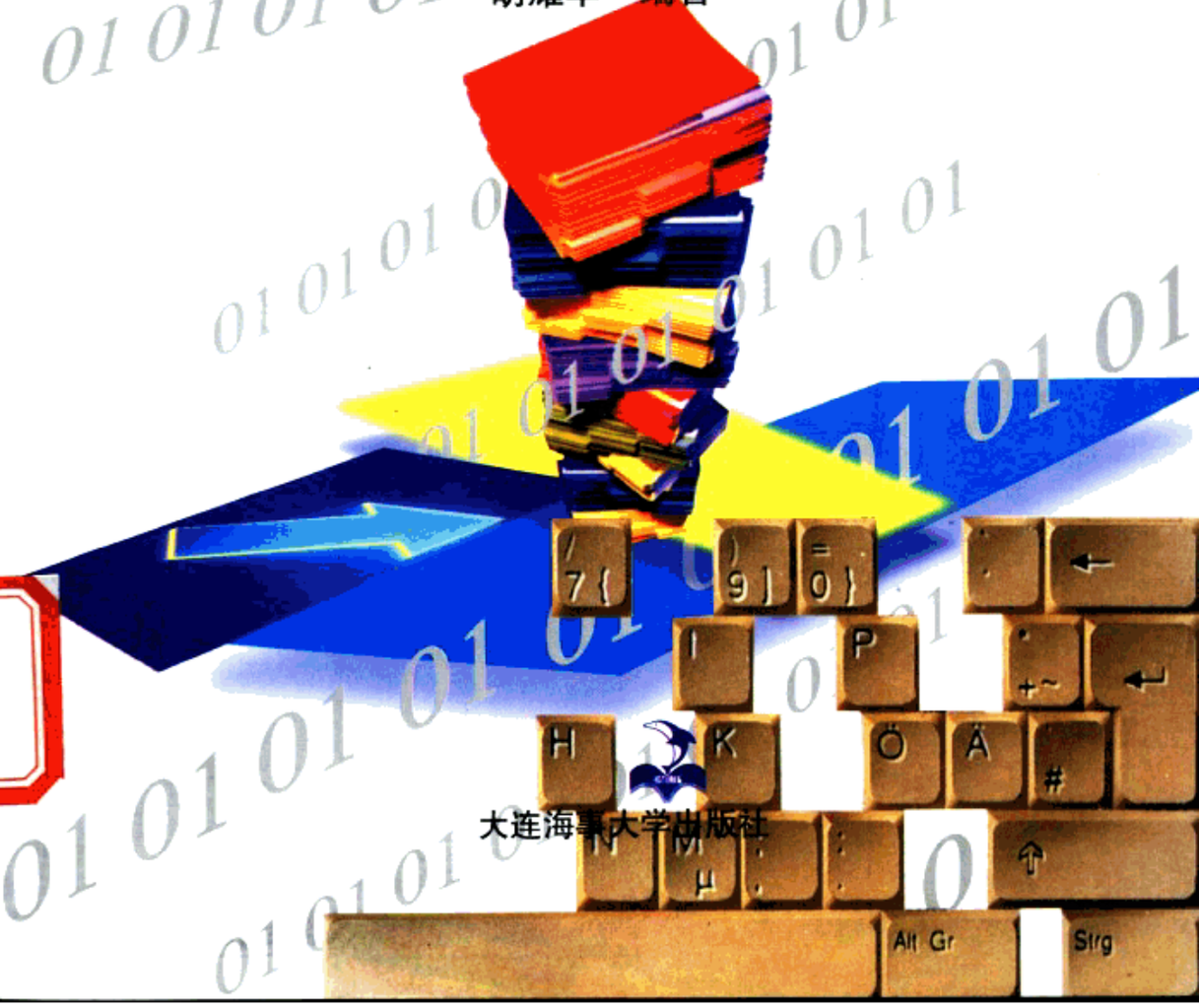
数字系统设计与在系统可编程技术

胡耀华　编著



大连海事大学出版社

本书由

大连海事大学学术著作出版基金资助出版

The published book is sponsored by

The Academic Works Publishing Foundation
of the Dalian Maritime University

前　言

随着电子设计自动化(EDA)技术的发展，可编程逻辑器件(PLD)在电子产品的设计中的应用日趋广泛。由于PLD具有灵活性高、开发周期短、前期开发费用较低等特点，深受电子工程师们的青睐，实现了用户"自制"大规模数字集成电路或数字系统的理想。

早期的PLD如PROM、PLA、PAL和GAL等器件的集成规模较小，一般相当于几十到几百个标准门单元。随着集成电路工艺的飞速发展，PLD的集成规模越来越大，达到上万个到几十万个门单元。这些大规模PLD器件主要可分为两类：现场可编程门阵列(FPGA)和复杂的可编程逻辑器件(CPLD)。前者的内部结构以基本门单元为基础构成门阵列，可编程的连线分布在门与门之间的布线区；后者的内部结构以可编程的逻辑宏单元为基础，其可编程连线集中在一个全局布线区。FPGA和CPLD的迅速崛起，使用户在实验室快速方便地开发专用集成电路(ASIC)成为可能，因而它正越来越受到业内人士的密切关注。

由Lattice公司开发的在系统可编程(ISP)逻辑器件具有容量大、速度快等特点，有些型号的器件内部还包含了存储器、寄存器和计数器等模块，可以满足复杂的逻辑功能设计的需要。尤其值得一提的是，由Lattice首创的"在系统可编程"设计方法，采用计算机辅助进行逻辑功能的设计和下载，其设计方法灵活多样；ISP器件采用了E^2CMOS工艺，融合了PLD的高性能和灵活性，其集成度大大提高；ISP器件的内部逻辑编程区与输入输出管脚之间没有一一对应关系，因此在修改系统的逻辑功能时，不用拔下芯片或拆下系统板，就可以随时通过软件在系统板上对器件的逻辑功能进行修改和重构，实现了"在系统上"编程的梦想。这使得用户在不改变电路系统的设计或印刷线路板硬件设置的情况下，在产品设计、制造过程的每个环节，甚至产品卖给最终用户后，都可利用软件随时对器件的逻辑功能进行修改或重构，这样进一步减少了设计流程，缩短了设计和调试周期，并打破了软件与硬件之间最后的屏障，实现了硬件逻辑设计的软件化，并使利用计算机辅助设计(CAD)的电子设计活动变成了实实在在的设计实体。随着各种EDA软件功能的不断完善，其开发过程变得更加形象直观，仿真方式也更便捷实时。由于设计中涉及的硬件因素相当少，设计的流程也更简洁，因此可以在很短的时间内完成较复杂的数字系统设计，为产品进入市场提供了宝贵的时间，掌握这种技术已成为从事电子技术的研究开发人员的必备技能。

本书是作者在原讲义的基础上，总结近几年来在教学、学习和开发应用ISP技术的经验和体会，吸收国内外相关文献的精华编著而成的。其目的是向读者介绍ISP技术的基本原理和编程方法，促进我国电子设计技术水平的提高。

全书共分五章，第一章介绍数字系统的基本概念和设计方法，包括算法状态机(ASM)和MDS图；第二章介绍了中小规模的可编程逻辑器件的结构，重点介绍了常用的PAL和GAL器件；第三章介绍了各种类型的在系统可编程器件的内部结构、工作原理和编程方式；第四章重点介绍在系统可编程器件的开发、设计和编程方法，包括编程设计语言ABEL和VHDL的语法结构、ISP Synario、ispExpert和菊花链下载软件的使用，并对初学者容易出错的问题予以重点说明；第五章给出一些具体的设计例子和实习课题。

本书的编写得到了大连海事大学学术专著出版基金的资助。在编书过程中，得到了信息学院朱义胜、王百锁、张淑芳、谢公福、夏志忠和出版社袁林新、林晓阳等教师的支持和帮助，责任编辑王铭霞对本书的修改提出了许多宝贵建议，在此一并表示感谢。

由于作者水平有限，错漏之处，恳请读者指正。

编　者

2001 年 3 月

目　录

第 1 章　数字系统设计

数字电路课程着重研究一般组合逻辑电路和时序逻辑电路的分析和设计方法，介绍编码器、译码器、选择器、分配器、比较器、全加器、计数器、移位寄存器和存储器等各种单一功能的集成器件，如何将各种逻辑器件组合起来协调工作是数字系统设计的主要研究内容。本章对数字系统的结构、设计步骤、设计方法等知识进行了介绍。

1.1 数字系统的基本知识

1.1.1 数字系统的结构

数字系统是具有一定逻辑功能的器件组合，组合后各器件之间相互协调工作，完成某个完整的功能，如数字密码锁、数字时钟、信号处理器、数字频率计、交通灯控制器、电梯控制器等。

数字系统通常包括输入输出接口、存储器和中央处理器(CPU)，如图 1.1 所示。输入输出接口用于与外界交换信息，存储器用于存放数据和各种控制信息。中央处理器是整个系统的核心，它包括数据处理器和控制器两部分，数据处理器由组合电路和寄存器组成，负责对数据的操作和检验，组合电路通过算术和逻辑运算实现对数据的加工和处理，寄存器用于暂存信息；控制器由组合电路和存储器构成，它是时序逻辑电路，规定数据处理器的具体操作步骤并使之协调工作，每个操作步骤对应一个存储器状态。

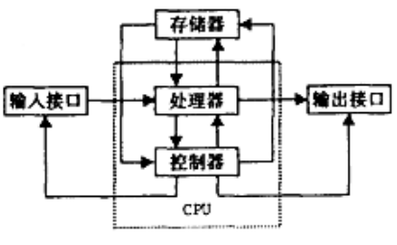


图 1.1　数字系统结构示意图

数字系统的逻辑功能通过一个有序的操作序列和检验序列完成，如图 1.2。在每个操作步骤内，控制器向数据处理器发出一组控制信号 T，数据处理器在完成控制信号规定的操作或计算后，将处理器产生的状态信号 S 反馈给控制器；在下一操作步骤中，控制器根据外部输入信号 Y 和处理器反馈的状态信息 S，产生一组新的控制信号 T 发送给处理器。经过多步操作(操作序列)后，完成某个完整的功能。

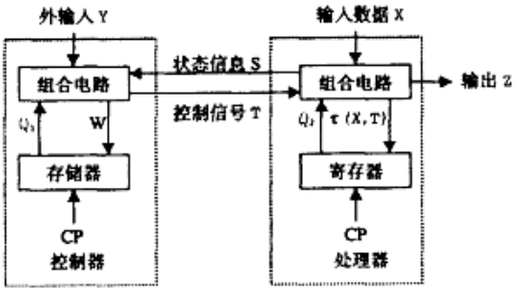


图 1.2　中央处理器结构图

1.1.2 数字系统各信号之间的关系

在了解了数字系统的概念和基本组成后，接下来具体介绍数字系统各模块间信号的流向和相互关系，读者可借此了解数字系统的工作过程。

1.1.2.1 处理器的寄存器次态

数据处理器的组合电路根据控制信号 T 对输入数据 X 进行处理后，产生信号 τ 和输出 Z，

信号 τ 作用到寄存器的功能控制端，使寄存器在时钟信号的作用下执行相应的操作，改变寄存器的状态。寄存器次态 Q_2^{n+1} 与输入数据 X、寄存器现态 Q_2^n 和控制信号 T 有关，其关系可表示为：

$$Q_2^{n+1} = F_1(Q_2^n, \tau) = F_1(Q_2^n, X, T)$$

1.1.2.2 数据处理器的输出

数据处理器的输出 Z 由组合电路产生，它由输入数据 X、控制信号 T 和寄存器状态决定，这可表述为：

$$Z = F_2(X, Q_2, T)$$

1.1.2.3 数据处理器的状态

控制器的操作序列一般是不固定的，控制器要根据处理器返回的状态变量 S 和外输入信号 Y 确定下一个操作步骤。处理器的状态 S 是通过对被处理信息的检验产生的，反映被处理的信息的状态，其关系式为：

$$S = F_3(X, Q_2)$$

在每个时钟周期内，数据处理器根据控制信号 T，对外输入信号 X 进行加工处理，执行相应的操作，产生新的寄存器状态和输出 Z，同时将执行后的状态 S 反馈给控制器，以确定下一时钟周期的控制信号。

1.1.2.4 控制器的状态

控制器给数据处理器提供操作序列，决定处理器的操作步骤，它是时序电路，其中存储器的状态同操作步骤相对应，它记忆控制器处于哪一个操作步骤，该发出什么样的控制信号，并根据控制器的现态 Q_1^n、处理器返回的状态 S 和外输入信号 Y，在时钟信号 CP 的作用下完成状态转换，其表达式为：

$$Q_1^{n+1} = F_4(W, Q_1^n) = F_4(Y, S, Q_1^n)$$

操作步骤是一个时间序列，因此通常采用状态转换表或状态转换图描述控制器的状态变化，根据状态转换图表进行控制器的设计。

1.1.2.5 控制信号

控制信号 T 是控制器的输出，它是控制器的现态 Q_1^n、外输入信号 Y 和数据处理器返回的状态信号 S 的逻辑函数：

$$T = F_5(Q_1^n, Y, S)$$

实现数字系统的逻辑功能一般需要经过多步操作，因此控制信号是一组信号，如 T_1，T_2，…，T_r，为了便于理解，在设计中通常采用助记符来表示控制信号，例如 CLR(清零)、ADD(寄存器信号相加)、INC(寄存器加 1)和 DEC(寄存器减 1)等。

1.1.3 数据处理器明细表

为方便设计，通常将处理器在各个控制信号下应执行的操作任务以表格的形式列写出来，形成数据处理器明细表，它包含操作表和状态变量表两个子表：操作表列出数据处理器在控制信号作用下应实现的操作和产生的输出，状态变量表定义数据处理器返回的状态变量。在建立明细表时，在一个时钟周期内可同时完成的操作应归结在一起作为一个操作步骤，受一个控制信号控制。图 1.3 描述了一个简单的数据处理器，其输入信号为 X，输出为 Z，系统包含 A 和

B 两个寄存器，控制信号序列为 T0，T1，T2，T3*（分别定义对应的助记符为 NOP，ADDA，ADDB，CLRAB），状态信号为 S，对应的数据处理器明细表如表 1.1，表中：

A←A+X 表示寄存器 A 的内容加 X 后送入 A 中；

B←B+X 表示寄存器 B 的内容加 X 后送入 B 中；

A←0 和 B←0 表示将 A 和 B 的内容清零；

S X>0 表示 X>0 时 S 有效，X<0 时 S 无效。

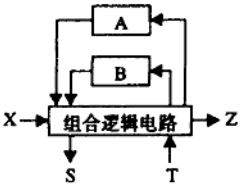


图 1.3 简易数据处理器

1.1.4 控制器状态转换表

控制器决定算法步骤，算法步骤与控制器中存储器的状态对应。在每个状态下，控制器根据收到的状态变量和外输入产生控制信号，在系统时钟到来时，控制器转换到下一个状态。控制器的状态转换如表 1.2 所示，表格中第一列表示现态，第一行表示状态变量，表格中的内容则分别为控制器的次态和控制信号的值。例如第 J 行、第 I 列的内容为 $P(S_I,Q_J)/M(S_I,Q_J)$，表示控制器的次态为 $P(S_I,Q_J)$，对应的控制信号为 $M(S_I,Q_J)$。

表1.1 数据处理器明细表

操作表		状态变量表	
控制信号	操作	状态变量	定义
NOP (T0)	不操作	S	X>0
ADDA (T1)	A←A+X	输出 Z=A	
ADDB (T2)	B←B+X		
CLRAB(T3)	A←0，B←0		

表1.2 控制器状态转换表

Q \ S	S_1	…	S_I	…	S_M
Q_1					
⋮					
Q_J			$P(S_I,Q_J)$ / $M(S_I,Q_J)$		
⋮					
Q_N					

在得到了处理器明细表和状态转换表后，就可很容易地描述数字系统的工作过程。

1.2 系统时钟

数字系统的逻辑功能是在控制信号作用下一步一步地实现的，这些控制信号的状态转换由系统时钟控制。实际设计的数字系统大多为同步系统，因此这里假设：

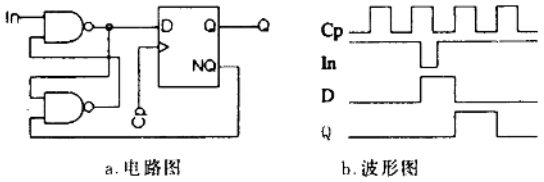
(1) 控制器的状态变化以及数据处理器中的寄存器传输操作都由系统时钟控制，即时钟脉冲同时到达所有存贮单元的时钟输入端。

(2) 系统中所有触发器是边沿触发的：时钟脉冲有效边沿（上升沿或下降沿）出现之后，寄存器内容更新。在同步系统中，时钟脉冲有效边沿出现后，数据处理器中寄存器的内容更新，它和输入信号一起形成状态变量 S；S 达到稳定后，控制器根据状态变量 S、控制器状态和外部输入 Y 形成控制信号 T；T 稳定后，才能建立稳定的寄存器功能选择信号 τ，并产生输出信号 Z，此后下一个时钟脉冲边沿才允许出现。因此数字系统的时钟选择存在一个最小时钟周期限制：从时钟脉冲有效边沿到达开始，到与系统中寄存器传输操作有关的信号均达到稳定值的

* 编程软件中的字体都采用正体、平排的格式，为保持统一性，本书仅对必要的信号采用上、下标等格式，请读者注意区分。

这段时间间隔。通常，时钟周期的范围从 20 ns 到几十微秒，在系统调试时可采用单步脉冲。

此外，由于外部输入的信号是随机的，与系统时钟无关，为了保证系统正常工作，应采用异步输入信号同步化电路将异步信号转换为同步信号，使所有输入信号都与时钟脉冲同步。图 1.4a 给出了一个异步信号同步化的电路，In 为开关或按钮产生的异步信号（已消抖动，低电平有效），Cp 为系统时钟，同步输出信号是 D 触发器的 Q 端（Q 高有效，也可以取 D 触发器的反相输出端 NQ 低有效输出）；图 1.4b 是该电路的波形图。



a. 电路图　　b. 波形图

图 1.4　异步信号同步化电路

1.3 数字系统的设计步骤

数字系统的设计是一个循序渐进的过程，根据设计者采用的设计方法的不同，实现同一逻辑功能的数字系统可能多种多样，如何评价并从中选择较好的系统往往是很难的，这在一定程度上取决于设计者的经验，这里仅讨论设计的基本步骤。

(1)分析设计任务

设计数字系统的第一步是对设计任务做透彻的了解，分析系统的设计目的、作用及性能要求，确定系统的输入输出信号和算法，在确定算法时还应兼顾逻辑器件型号的选择，并力求令用户操作使用方便。

(2)确定数据处理器的结构

确定算法的过程就是数据处理器结构的建立过程，一种算法对应一种结构，算法定了，结构也就确定了。由于所有的寄存器传输操作都在数据处理器中完成，所以确定数据处理器的结构包括确定处理器中的寄存器。

(3)数据处理器的详细说明

这一步确定处理器接收的控制信号，以及在每个控制信号作用下要完成的寄存器传输操作；确定在处理信息时要完成的信息检验；确定输出信息。根据确定的算法可以建立数据处理器的明细表，同时对处理器的结构进行修改。

在建立了处理器明细表后，可根据算法流程图得到控制器的状态转换表，这样就通过状态表定义了控制器，通过处理器明细表定义了处理器。至此，数字系统的设计基本完成。

(4)数字系统的实现

在完成了数字系统的系统级设计后，下一步工作是转入逻辑级设计，即利用寄存器和组合电路来实现控制器和处理器。

1.4 寄存器传输语言

数据处理器接收控制器发出的控制信号，对信息进行加工存贮，并检验信息间的逻辑关系，产生状态变量，这些对寄存器内存储信息进行加工和存贮的操作称为寄存器传输操作。寄存器传输语言（Register Transfer Language—RTL）是描述这些操作的一种方便的工具。

在 RTL 中，寄存器是基本的逻辑单元，它是广义的，即在 RTL 中定义的寄存器，不仅包括普通的寄存器，还包括移位寄存器、计数器、存贮器及其它类型的寄存器；存储在寄存器中的二元信息包括二进制数、BCD 码、字符、控制信息等。例如，加法计数器被看作是具有增 1 功能的寄存器，存贮器是存储信息的寄存器堆，单个的触发器则是一位寄存器。

RTL 适用于描述功能部件级的数字系统的工作，它以与程序设计语言中使用的语句相类似的一组表达式，简明精确地描述寄存器存储信息的操作处理，在系统的设计要求与实际的硬件之间建立起一一对应关系。RTL 语句通常包含以下 4 个成分：

(1)系统中的寄存器及其功能；

(2)寄存器中存贮的二进制编码信息；

(3)存贮在寄存器中的信息要实现的操作；

(4)启动操作的控制函数。

数字系统中常用的寄存器传输操作可分为 4 类：

(1)寄存器间的信息传输：这种操作将信息从一个寄存器传送到另一个寄存器，而不改变源寄存器内的内容；

(2)算术运算：对存储在寄存器中的数字量进行加法、减法等算术运算；

(3)逻辑运算：对存储在寄存器中的信息进行与、或、非等逻辑运算；

(4)移位操作：对存储在寄存器中的二元信息进行移位操作。

1.4.1 寄存器间的信息传输

在 RTL 语言中，寄存器用大写字母表示，寄存器中的每一位触发器按顺序从右到左依次用 0 到 n－1 编号，每位的编号用方括号括起来，例如用 A[3]表示寄存器 A 的右起第四位。

寄存器之间的信息传输通过置换操作表示，以符号的形式指明从一个寄存器到另一个寄存器的信息传输，它不改变源寄存器的内容，而是在时钟脉冲有效时将源寄存器中各触发器的值复制到目标寄存器中对应的各个触发器中。例如语句 B←A 将源寄存器 A 的内容传输给箭头指向的目标寄存器 B，当 A 和 B 都为 3 位时，上述操作等价于：B[2]←A[2]、B[1]←A[1]、B[0]←A[0]。

通常寄存器传输操作是有条件发生的，这种条件由控制逻辑函数规定，例如语句 $X\overline{T}$：B←A表示当 X 为 1、T 为 0 时，寄存器 A 的内容传输到寄存器 B。

寄存器传输语句与实现操作的硬件电路之间存在一一对应关系。例如，为实现上面的条件传输语句，硬件电路应包含位数相同的源寄存器 A 和目标寄存器 B，A 的输出端与 B 的数据输入端相连接，寄存器 B 应有一个并行输入数据控制端，由控制电路产生的控制函数控制。

相互间不发生冲突的几个寄存器传输操作，可放到同一个时钟脉冲周期内并行执行，以节省工作时间。例如，语句 A←B 和 A←C 是不能并行执行的，因为寄存器 A 不可能同时接收 B 和 C 两个寄存器送来的信息；而语句 B←A 和 C←A 可以并行执行，因为寄存器传输操作不改变源寄存器 A 的内容，两条语句不发生冲突。

1.4.2 算术运算操作

算术运算操作包括加法、减法、取反、移位，这些操作可组合得到其他的算术操作，常用的算术操作见表 1.3。

1.4.3 逻辑运算操作

逻辑运算操作包括逻辑与、或、非，它是两个寄存器对应位之间的操作，为了与算术运算符

区别开，逻辑运算中利用“∧”表示逻辑与，用“∨”表示逻辑或。但在控制函数中，仍用“·”表示逻辑与，用“+”表示逻辑或，因为控制函数中不包含加、减运算，不会发生混淆。逻辑运算符及其说明见表 1.3。

表1.3 寄存器传输语言的常用符号

	符号表示法	说　明
算术操作	F←A+B	A与B之和传输给F
	F←A-B	A与B之差传输给F
	B←$\overline{B}$+1	求寄存器B存数的补码
	B←$\overline{B}$	求寄存器B存数的反码
	F←A+$\overline{B}$+1	A+(B的补码)传输给F
	A←A+1	A增加1
	A←A-1	A减1
逻辑操作	A←$\overline{A}$	逻辑非操作
	F←A∨B	A与B的内容按位相或后传输给F
	F←A∧B	A与B的内容按位相与后传输给F
	F←A⊕B	A与B的内容按位异或后传输给F
移位操作	X←SL(X,A)	寄存器X的内容逻辑左移(低位向高位移动)一位,A的值移入X的最低位
	X←SR(X,A)	X的内容逻辑右移(高位向低位移动)一位,A的值移入X的最高位
	X←SL(X)	X的内容逻辑左移一位,最低位补0
	X←SR(X)	X的内容逻辑右移一位,最高位补0

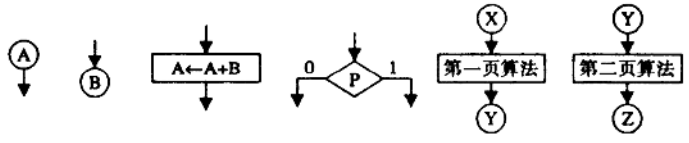
1.4.4 移位操作

移位操作包括左移和右移操作，如表 1.3 所示。数据可串行移入或移出寄存器，每次移动一位，移位后寄存器内容被更新，因此移位操作也常被当作是一种寄存器传输操作。右移操作完成后，寄存器中每位的权值为原来的一半，因此右移操作还经常用于除 2 运算；而左移操作则经常用于乘 2 运算。

1.5 一般流程图

数字系统逻辑功能的实现要通过一个有序的操作序列和检验序列来完成，或者说通过一个算法来完成，算法通常用流程图来表示。流程图使用 4 种符号：入口点、出口点、传输框和判断框，如图 1.5 所示。

入口点(如图 1.5a)指明算法的起点或继续点，出口点则指明算法的结束点(如图 1.5b)。当算法较长，一页写不完而另起一页时，就需要一个继续点，前一页的结束点为出口点，后一页的起始点为入口点，通常出口点和入口点是成对出现的，如图 1.5e 所示。通过第一页的出口点 Y 和第二页的入口点 Y，算法从第一页转向第二页。



a.入口点 b.出口点　c. 传输框　　d. 判断框　　e.入口点和出口点

图 1.5　流程图符号

传输框(如图 1.5c)表明在算法的特定点要完成的寄存器传输操作，这些操作通常采用寄存器传输语言(RTL)来表示。

判断框(如图 1.5d)中的 P 是被检验的状态变量，当条件满足时选择一个引出分支，条件不满足时引出另一个分支。

流程图的画法是根据设计的数字系统的要求，按照系统操作的先后顺序排列的，与系统的状态没有明显的对应关系。下面通过例子来说明流程图的应用。

例 1.1:绝对值求和计算。

设计一个数字系统,求 $Z=|X_1|+|X_2|+|X_3|$ 的值。

输入 X:输入端提供数列 X_1, X_2, X_3。

输出 Z:当标志 C 为 1 时,输出 Z 有效。

本例要实现加法运算,因此数据处理器应有一个加法器实现加法操作、一个寄存器 A 暂存累加器的值、一个寄存器 B 暂存输入数据 $X_i(i=1,2,3)$、一个触发器 C 作为运算结束的标志位,系统的结构如图 1.6 所示。

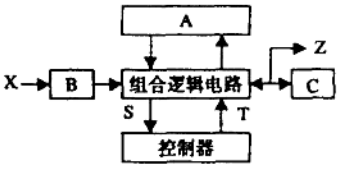


图 1.6 系统结构

表1.4 数据处理器明细表

操作表		状态变量表	
控制信号	操 作	状态变量	定义
CLR	A←0, C←0	S=0	X≥0
ADDB	A←A+B	C为1时, Z=A	
SUBB	A←A-B		
ADDF	A←A+B, C←1		
SUBF	A←A-B, C←1		

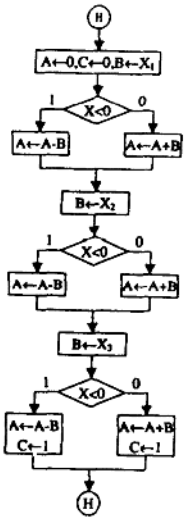


图 1.7 流程图

此外,在数据处理过程中涉及到有符号数的绝对值运算,因此要对输入数据 X_i 的符号进行判断,当 X_i 为正时做加法运算,当 X_i 为负时做减法运算,经过三次运算后将标志位 C 置 1,系统的流程如图 1.7 所示。

根据算法的流程图确定数据处理器应完成的操作,并将可同时完成的操作归在一起作为一个操作步骤,受一个控制信号控制,由此得出的数据处理器明细表如表 1.4,其中状态 S 表示符号位。

例 1.2:求函数 $Z=4X_1+2X_2+X_3$ 的值。

输入 X:输入端提供数列 X_1, X_2, X_3。

输出 Z:计算结束后,输出 Z 的值。

本例中函数的运算涉及乘法及求和运算,为实现乘法运算,我们将函数重新写为 $Z=2(2X_1+X_2)+X_3$,这样可采用寄存器左移实现乘 2 操作,实现该函数的算法流程如图 1.8 所示。为实现流程中的运算,系统应有一个寄存器 A 暂存计算结果,另有一个组合电路在控制信号的作用下实现加法运算,系统的结构框图如图 1.9 所示,对应的处理器操作如表 1.5。

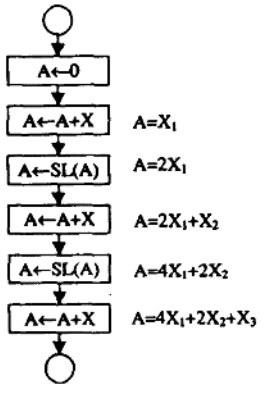


图 1.8 流程图

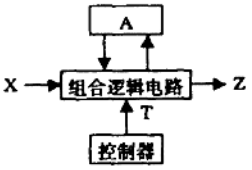


图 1.9 系统结构

表1.5 数据处理器操作表

控制信号	操 作
CLR	A←0
ADD	A←A+X
SLA	A←SL(A)

1.6 数据处理器的实现

由于数字系统设计的复杂多样，处理器的设计方法和实现手段也多种多样，本节通过数字乘法器的设计来演示处理器的设计过程。

例 1.3：数字乘法器的设计。

(1)设计任务

求两个 4 位二进制数之积。

输入：乘数存于寄存器 Q 中，被乘数存于寄存器 M 中，启动信号为 MF。

输出：8 位二进制乘积。

(2)设计任务分析

乘法运算通常被分解为多步加法运算，4 位二进制乘法的笔算过程如表 1.6 所示。根据表 1.6 可总结出两个 4 位二进制数相乘的部分规律：

①两个 4 位二进制数相乘，乘积为 8 位数；

②如果乘数的第 i 位是 0，则第 i 个部分积为 0；如果第 i 位是 1，则第 i 个部分积为左移 i－1 位后的被乘数；

③乘积等于各个部分积之和。

在笔算中，我们首先求出所有部分积，最后相加在一起，但用数字电路实现时，全加器只能实现两个二进制数的累加，用它在最后一步对多个部分积累加是无法实现的，因此需要修改乘法过程，将多个数一次相加改为多次累加求和，每次累加得出的部分和暂存于累加寄存器中，其运算过程见表 1.7。

表1.6 乘法的笔算过程

				1	0	1	1	被乘数
			×	1	0	0	1	乘数
				1	0	1	1	第1个部分积
			0	0	0	0		第2个部分积
		0	0	0	0			第3个部分积
	1	0	1	1				第4个部分积
0	1	1	0	0	0	1	1	相加得乘积

表1.7 乘法的多步累加求和过程(部分积左移)

				1	0	1	1	被乘数
			×	1	0	0	1	乘数
0	0	0	0	0	0	0	0	累加器初值
				1	0	1	1	第1个部分积
0	0	0	0	1	0	1	1	第1个部分和
			0	0	0	0		第2个部分积
0	0	0	0	1	0	1	1	第2个部分和
		0	0	0	0			第3个部分积
0	0	0	0	1	0	1	1	第3个部分和
	1	0	1	1				第4个部分积
0	1	1	0	0	0	1	1	乘积=第4个部分和

根据表 1.7 的乘法过程可得 4 位二进制数乘法器的算法流程图和系统结构图分别如图 1.10 和图 1.11 所示。

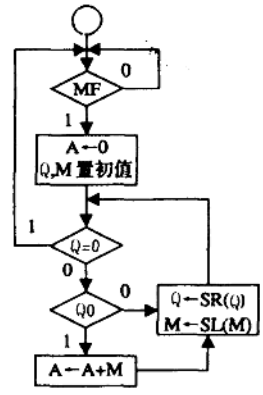


图 1.10 算法流程图

在结构图中，寄存器 Q 存放乘数，寄存器 M 的第 0 到 3 位存放被乘数，累加寄存器 A 初始清 0，在外部启动命令信号 MF 出现后开始计算。在每个部分积累加的循环过程中，首先判断 Q0，若 Q0 为 0，则只对 M 左移，否则将 M 的值累加到 A 之后再对 M 左移；接着对 Q 的内容右移，最左面的一位移入 0。经过 4 次循环后，寄存器 Q 的内容变为 0，乘法过程结束，乘积存放在累加寄存器 A 中。

这种部分积左移的算法是根据表 1.7 直接导出的，它的缺点是寄存器使用效率太低。例如被乘数寄存器 M 虽然存放 4 位信息，却具有 8 位长度。

乘法器的另一种方案是部分和右移，算法流程和系统结构分别如图 1.12 和图 1.13 所示。在这种算法中，累加寄存器 A 有 5 位(A4～A0)，最高位 A4 用来存放求和产生的进位，被乘数寄存器 M 和乘数寄存器 Q 也是 4 位。若将累加寄存器 A 和乘数寄存器 Q 看作合成寄存器 AQ，则新的部分和形成后，AQ 右移一位。乘数寄存器 Q 具有双重功能：它在操作开始时存放 4 位乘数，在乘法结束时存放乘积的 4 个最低有效位(乘积的 4 个高有效位存放在寄存器 A 中)。上述算法的运算过程留给读者自行分析，这种方案提高了寄存器的利用率，但延长了运算时间，每次求积必须完成 4 次右移操作。相对而言，第一种方案在乘数寄存器的内容为 0 时计算就结束了。

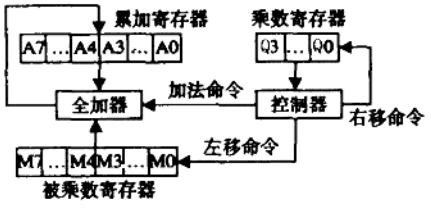


图 1.11 系统结构图

(3)数据处理器明细表

下面采用第二种方案来实现数字乘法器。根据算法流程图 1.12，可得出数据处理器应完成的寄存器传输操作，将可同时完成的操作归在一起作为一个操作步骤，并用适当的助记符号表示控制信号，便得到数据处理器的操作表；处理器的输出和状态变量定义为状态变量表，如表 1.8 所示。

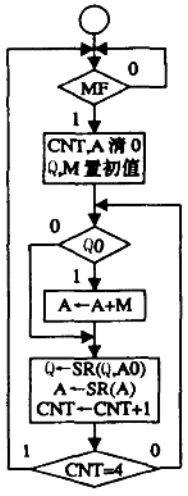


图 1.12 算法流程图

(4)乘法数据处理器的实现

根据乘法器的处理器明细表，数据处理器应包含累加寄存器 A、乘数寄存器 Q、被乘数寄存器 M 和计数器 CNT，与 A、Q 和 M 相关的操作有并入(用于清 0)、移位和保持，这些操作可用 4 位双向移位寄存器 74LS194 实现，分别记实现 A、Q 和 M 的 74LS194 为 A_194、Q_194 和 M_194。

图 1.14 和表 1.9 分别给出了 74LS194 的逻辑符号和功能表，从功能表可见，194 的功能取决于控制端 S0 和 S1。根据表 1.8，寄存器 A 的操作包括清 0、移位和保持。清零可用并行置数 0 实现；全加器完成加法操作后的和(F = A + M)也可连接到A_194 器件的数据输入端，赋值给 A；注意 A 寄存器为 5 位，而 194 只有 4 位输出，因此由加法器产生的进位 C4(即累加

表1.8 乘法器处理器明细表

操作表		状态信号表	
控制信号	操作	状态信号	定义
NOP	不操作	S1	MF
START	A←0, CNT←0, Q和M置初值	S2	Q0
ADD	A←A+M	S3	CNT=4
SHIFT	Q←SR(Q,A0), A←SR(A), CNT←CNT+1	输出Z=AQ	

器A的A4)在ADD信号有效时存入一个D触发器中，在右移信号有效时，D触发器的输出QD通过194的右移数据输入端DR移到累加器A的A3(即A_194的Q3)中。根据功能表1.9，可得到实现累加器A的194的各信号引脚的真值表如表1.10所示，由真值表可得控制端的逻辑表达式为：

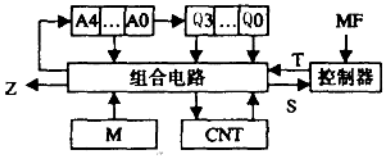


图1.13 系统结构图

$S0 = START + ADD + SHIFT$

$S1 = START + ADD$

$D_i = F_i \overline{START}(i = 3 \sim 0)$

DR = QD(触发器的输出)

D = C4·ADD(D为触发器的激励)

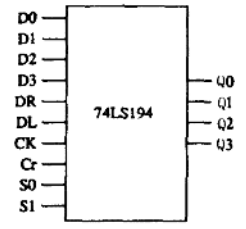


图1.14 74LS194的逻辑图

乘数寄存器Q执行的操作包括置数、移位和保持。在计算开始时，需要将乘数置入寄存器Q中，这可将乘数连接到194的数据输入端D3～D0，利用74LS194的置数功能实现；当控制信号SHIFT有效时执行右移操作(右移入累加器A的A0，即A_194的Q0)。根据上述分析，可得出实现Q的194器件的控制端的逻辑表达式为：

表1.9 74LS194功能表

Cr	S1	S0	CK	$Q3^{n+1}$	$Q2^{n+1}$	$Q1^{n+1}$	$Q0^{n+1}$	说明
0	×	×	×	0	0	0	0	异步清零
1	0	0	×	$Q3^n$	$Q2^n$	$Q1^n$	$Q0^n$	保持
1	0	1	↑	DR	$Q3^n$	$Q2^n$	$Q1^n$	右移
1	1	0	↑	$Q2^n$	$Q1^n$	$Q0^n$	DL	左移
1	1	1	↑	D3	D2	D1	D0	置数

S0 = SHIFT + START

S1 = START

DR = A0(即A_194的Q0)

D3～D0 = 乘数

表1.10 累加器A(74LS194)的控制端真值表

控制信号	S1	S0	D3~D0	DR	说明
START	1	1	0000		置0
ADD	1	1	F3~F0		F=A+M的值
SHIFT	0	1	×	QD	右移

被乘数寄存器M只在计算开始时有存入被乘数的操作，它也可用74LS194的置数功能实现，即控制端S0 = S1 = START，数据输入端接被乘数。

计数器CNT具有增1和同步清0的功能，这里选择4位二进制同步加法计数器74LS163来实现，163器件的功能表如表1.11。根据处理器明细表，当START有效时，Cr取0对CNT清零；当SHIFT有效时，S0 = S1 = 1进行加法计数。因此，$Cr = \overline{START}$，S0 = S1 = SHIFT。

表1.11 74LS163功能表

Cr	LD	S1·S0	CLK	说明
0	×	×	↑	清零
1	0	×	↑	置数
1	1	0	×	保持
1	1	1	↑	计数

当计数器为4(即163的输出Q2 = 1)时，状态S3 =