第 1 章　绪论

数字信号处理起源于 17 和 18 世纪数学的一个学科，今天它已广泛地深入到通信、遥感、地震测量、石油勘探、生物医学以及海洋学等各个科学和技术领域中，并成为一种重要的现代化工具。数字信号处理主要研究用数字或符号序列表示信号和处理这些序列。与模拟信号处理相比，数字信号处理具有灵活性强、精度高、处理成本低以及对环境没有特殊要求等优点。它不仅能实现模拟处理的大部分功能，而且还能完成模拟处理由于成本、可靠性等原因而无法具体实现的功能。同时随着信息技术、大规模集成电路和计算机的飞速发展，数字信号处理理论以及数字信号处理实现手段取得了长足的发展，从而使数字信号处理发展成为一门崭新而独立的学科。

1.1　数字信号处理技术的发展

信号处理有着悠久的历史，在各个不同的领域，如生物医学工程、仪器设备、海洋学、遥感、声学、声纳、雷达、地震学、语音通信、数据通信和核子科学等都充分显示了它的重要性。在某些应用中我们希望提取信号的某些特征，用以分析信号、识别信号或者压缩信号。如在语音识别应用中，我们需要提取语音信号的基音信息、声道参数等，以达到识别语音或识别讲话者的目的。而在语音压缩应用中，我们希望提取出语音信号的声道特征和激励特征，并利用合成分析等算法达到压缩信号的目的。另外，在某些应用中我们可能希望对信号进行增强处理，剔除或者减弱混在信号中的噪声或干扰。如在通信应用中，信号在信道传输过程中经常会受到信道噪声、多径衰弱等影响，为了提高接收质量，就需要使用信号处理技术来去掉或减弱这些干扰。

信号处理从分析的角度来看主要分为时域分析和频域分析两类。在时域中信号表示为幅度沿时间变化的关系，而在频域中信号表示为幅度沿频率变化的关系，图 1-1 给出了信号时频域表示之间的关系。图 1-1(c)给出了信号时域表示的一个例子，该信号由两个不同频率的谐波沿时间轴逐点相加得到，图 1-1(b)给出了这个信号的频域表示，图中的两个脉冲表示该信号包含两个频率成分，同时脉冲的幅值表征了谐波分量的能量大小。将两者综合起来，可得到图 1-1(a)所示的时间、频率和幅度关系示意图。时域和频域是对信号的两种表示方法，它们从不同角度表示了信号的重要特征，我们可以根据不同应用情况和不同需要来灵活选择，而

它们之间的关系由经典的傅里叶变换和傅里叶反变换联系起来。

信号处理最基本的手段是滤波和谱分析。滤波一般在时域进行，它的算法包括低通滤波、高通滤波、带通滤波和带阻滤波，其作用是去除信号中一些不需要的频率成分或者提取出信号中一些需要的频率成分，实现手段包括有限脉冲响应(FIR)滤波和无限脉冲响应(IIR)滤波。而谱分析一般在频域进行，以提取出信号的频率特征。

在早期，信号处理算法一般在模拟设备上实现。由于模拟实现的可靠性、灵活性差，而且对于某些复杂的算法实现起来困难，另外，在某些应用如地球物理数据处理中，往往需要把数据先记录在磁带上，然后在大型数字计算机上进行处理。另外在 20 世纪 50 年代还出现了用计算机作信号处理的另一种方法。由于计算机速度快又灵活，因而人们往往在用模拟器件实现信号处理系统之前，先在计算机上进行模拟，以确定方案的可行性。这些都是用数字方法进行信号处理的雏形。由于受当时信号处理算法和处理器能力的限制，这些早期的数字信号处理一般都是非实时的。

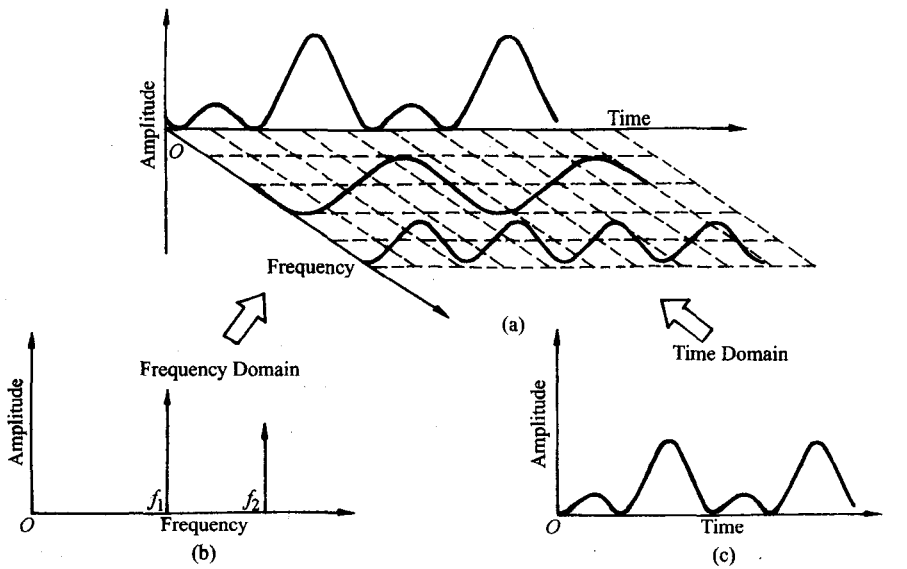


图 1-1　信号时频域关系图

现实世界中的信号一般都是模拟的，为了用数字方法对信号进行处理，首先要对信号进行数字化。奈奎斯特（Nyquist）采样理论给出了对模拟信号进行数字化，并用数字化后的信号无失真地恢复出原始模拟信号的条件，即采样频率必须大于等于模拟信号所包含的最高频率的两倍以上，正是奈奎斯特采样理论奠定了数字信号处理的基础。信号经过数字化后，信号的表现形式和一些特征发生了变

化，简单地套用原来的模拟信号处理算法已难以满足需要，因而对新的数字信号处理算法的研究成为了热点。

在 20 世纪 60 和 70 年代，数字信号处理技术以理论研究为主，在这期间，各国研究人员尝试和建立了许多巧妙的信号处理算法，一些算法已不单是简单地利用数字计算机的灵活性，而是以前从未在模拟系统中实现过。其中 1965 年发现的计算傅里叶变换的快速算法-快速傅里叶变换（ FFT）是最重要也是最具代表性的算法之一。快速傅里叶变换在离散时域上直接计算离散时域信号或序列的傅里叶变换，把传统的计算傅里叶变换的计算量减少了几个数量级，从而为数字信号处理的实时实现提供了可能。同时快速傅里叶变换提供了一套在离散时域上精确成立的特性和数学关系，它已经不单纯是连续时域傅里叶变换的近似了。它的重要作用是促使人们利用时域离散数学方法建立许多数字信号处理的概念和算法，在离散时域和频域上形成一套严格的关系式，从而使数字信号处理作为一门新的学科出现在世人面前并为世人所接受。在理论研究获得进展的同时也出现了多本数字信号处理方面的著作，其中美国科学家 A.V.Oppenheim 和 R.W.Schafer 共同撰写的《Digital Signal Processing 》是数字信号处理历史上一本里程碑式的经典著作。

数字信号处理技术的发展速度惊人，在短短几十年间，以经典的傅里叶频谱分析为基础，相继出现了自适应信号处理、现代谱估计、时频分析、多维信号处理等很多分支，近几年更出现了小波分析，所有的这一切使数字信号处理技术研究的对象从早期的确定性信号拓展到平稳随机信号、非平稳随机信号、时变信号和非高斯信号等，几乎覆盖了现实生活的每一个角落。同时数字信号处理也因此成为一门具有完整体系的交叉学科。

数字信号处理理论的渐渐成熟，使之越来越多地被应用到各个领域。在使用的过程中，传统的处理器因其结构上的限制，愈来愈显得力不从心，从而在 20 世纪 80 年代出现了专门为数字信号处理设计的电子器件，其中以数字信号处理器（Digital Signal Processor，DSP ）最引人注目。随着微电子技术的突飞猛进，DSP 的更新步伐也越来越快，其使用已扩展到军事、工业、通信等诸多领域，成为现代化技术的重要组成部分。

1.2 数字信号处理系统简介

根据信号处理系统的构成、处理能力以及计算问题到硬件结构映射方法的不同，将现代信号处理系统分为三大类：

(1）指令集结构(ISA)系统。在由各种微处理器、DSP 处理器、单片机或其它专用指令集处理器等组成的信号处理系统中，都需要通过系统中的处理器所提供

的指令系统（或微代码）来描述各种算法，并在指令部件的控制下完成对各种可计算问题的求解。在这类系统中，由于 DSP 处理器采用改进的哈佛（Harvard）总线结构，内部配有专用的硬件乘法器、累加器，以流水线方式工作，具有良好的并行特性，并有专门设计的适于数字信号处理的指令系统，因而使DSP 处理器成为现代数字信号处理系统中最重要也是最常用的处理单元。

(2）硬连线结构系统。主要是指由专用集成电路（ASIC)构成的系统，其基本特征是功能固定、通常用于完成特定的算法，这种系统适合于实现功能固定和数据结构明确的计算问题。不足之处主要在于：设计周期长、成本高，且没有可编程性，可扩展性差。

(3）可重构系统。基本特征是系统中有一个或多个可重构器件（如 FPGA)，可重构处理器之间或可重构处理器与 ISA 结构处理器之间通过互连结构构成一个完整的计算系统

本书将要介绍的数字信号处理系统主要是以 DSP 处理器为核心的指令集结构系统。

1.2.1 基本的数字信号处理系统

一个基本的数字信号处理系统的框图如图 1-2所示。

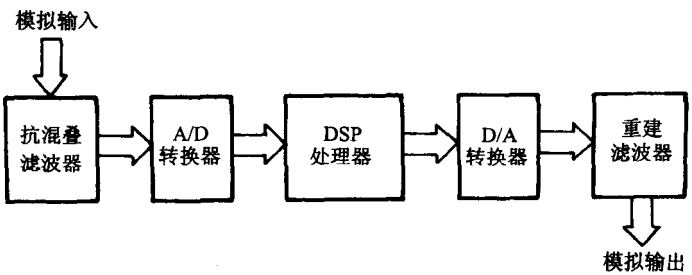


图 1-2 基本数字信号处理系统

图 1-2 中输入为模拟信号，它可以是电信号、声音信号、物理信号、化学信号等经过传感器和放大器等器件得到的一定幅值的电信号。在送给 DSP 处理器进行处理之前，这些模拟电信号需首先经过抗混叠滤波器和 A/D（模拟/数字）转换器。其中抗混叠滤波器一般是低通滤波器，也可以是带通滤波器，其作用是滤除模拟信号中不需要的频率分量，使得经过抗混叠滤波器后的信号变为带限信号，从而保证后面模数转换时满足奈奎斯特采样定理。 A/D 转换器的作用是对输入的带限信号进行离散化，包括时间离散化和幅值离散化，即采样、量化。

DSP 处理器的任务是将 A/D 转换器送来的数字信号按照一定的算法进行处理，如滤波、傅里叶变换、模型参数提取、频谱分析等等。根据处理任务的要求，

DSP 处理器可以由一个 DSP芯片及外部总线构成，也可以由多个 DSP 芯片构成。信号经 DSP处理器处理后，可以数字方式输出，如果期望输出是数字信号，则数字信号处理系统可以不包括图 1-2 中的最后两个模块，如果系统最后需要模拟输出，则 DSP 处理器将处理后的数字信号送给其后的 D/A（数字／模拟）转换器。经 D/A 转换后的信号含有许多高频成分，为此需在其后接一个重建滤波器以对信号进行平滑。

1.2.2 典型的数字信号处理系统

图 1-2 所示的数字信号处理系统仅包含了一些最基本的模块，在实际的信号处理中，往往还需要附加其他的功能模块，如存储器、人机界面、逻辑控制等，图 1-3给出了一个典型数字信号处理系统的框图。

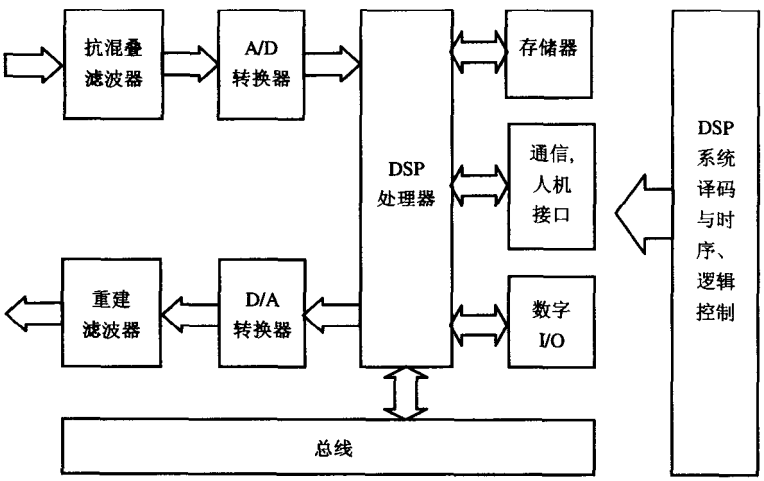


图 1-3 典型数字信号处理系统

图 1-3 中存储器可以是 RAM 或 ROM，用来存放 DSP 处理器所需的程序、表格或者临时数据等。通信包括串行通信和并行通信，人机接口可以有键盘、显示设备等，根据具体的应用情况，它们可以通过专用的驱动器与 DSP 处理器相连，也可以直接与 DSP处理器相连。数字 I/O 用来提供需要的数字信息。DSP系统译码与时序、逻辑控制用来保证数字信号处理系统根据用户的期望来工作，它包括地址译码、产生片选信号、读写逻辑控制等，它可以由逻辑门电路来实现，也可以选用可编程逻辑门阵列来实现，如 CPLD 等。在许多应用中，数字信号处理系统往往以计算机或工控机的插卡形式出现，因此需要有 DSP 处理器与总线（ PCI、ISA、VXI 等）之间的接口。

1.3 DSP的应用

自从美国德州仪器(TI)公司1982年推出通用可编程DSP芯片以来，DSP技术获得了突破性的发展，它的应用范围也变得越来越广泛。从最初DSP只用于实时大数据量处理，到现在应用于网络与互联网、高速调制解调器、无线通信、语音信号处理、多媒体产品、机顶盒、汽车、工业控制、军事、生物医学工程、机械、遥感遥测、航空航天自动化等， DSP技术的应用已覆盖全球各行各业。在表1-1中大致归纳出了一些DSP技术的应用。

表1-1 TMS320 DSP的典型应用

一般应用	自适应滤波器、卷积、相关、数字滤波、快速傅里叶变换、希尔伯特变换、信号产生
通信	高速调制解调器、编/解码器、自适应均衡器、传真、无线移动电话、数字留言机、语音信箱、回音抵消、电视会议、扩频通信、网络数据设备
多媒体信号处理	语言识别、语音合成、音频编码、语音增强、TTS、图像编码、三维图形处理、模式识别、图像增强、动画、电子地图、桌面出版系统
自动控制	磁盘/光盘伺服控制、激光打印机伺服控制、机器人控制、发动机控制、电机调速、无刷直流电机、卡尔曼滤波
仪器设备	谱分析、函数发生、数据采集、模态分析、石油/地质勘探、飞行器风洞试验
医学	助听器、X线CT、超声设备、心电图机、脑电图机、核磁共振监护仪器、假肢控制、助残装置、康复应用
军事	雷达与声纳信号处理、导航、制导、保密通信、全球定位、航空/航海应用
消费类电子	数字音频、数字电视、教育玩具、音乐合成器、数字应答/留言机
计算机	阵列处理器、图形加速器、工作站、神经网络、多媒体计算机

面对DSP的巨大的市场和广阔发展前景，世界上最大的几个半导体公司都在DSP上开展竞争。如TI、AD、Agere、Motorola、Siemens Semiconductor等公司都在全力开发和生产DSP器件。不同公司的DSP侧重于不同的应用，为开发者提供了一个较大的选择范围。

1.4 本章小结

本章首先简单地介绍了数字信号处理技术的发展过程；然后给出了基本数字信号处理系统和典型数字信号处理系统的结构框图，并对构成数字信号处理系统各个的模块作了简单介绍，其中的绝大部分都将在本书后续章节中作详细介绍；最后对DSP技术的应用作了大致的归纳。

第 2 章　数字信号处理器基础

数字信号处理器（DSP）可在数字信号处理领域获得巨大的应用，与其本身所具有的软硬件结构特征是密不可分的。在硬件上，DSP 采用多总线的哈佛结构或改进的哈佛结构，从而使 DSP 可以在同一时刻进行程序读、一个或多个数据读、数据写。另外，DSP 一般采用流水线工作方式，在一条指令执行的同时，进行后续指令的取操作数、程序译码、程序读取等功能，大大提高了 DSP 的处理速度。DSP 的另一个重要特点是配置有专用的硬件乘法器，它可以在一个时钟周期内计算相应长度两个数的乘积。在软件上，DSP 一般采用复杂指令集，即一条 DSP 指令往往同时完成多个任务，而且为了提高执行数字信号处理算法的效率，DSP 一般包含一些专用指令，如用于 FFT、滤波、多项式求和等计算的指令。

本章主要介绍 DSP 的基础知识，包括 DSP 的硬件结构、软件特征以及 DSP 的发展历史，从而使读者对 DSP 能有一个基本认识。

2.1　数字信号处理器的硬件结构

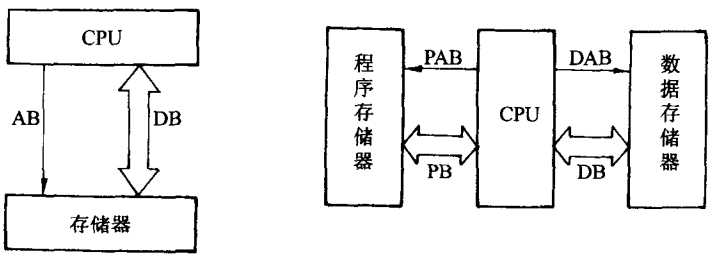
为了能高速完成数字信号处理算法，DSP 硬件系统有着自身特定的结构。作为 DSP 硬件开发人员自然要充分了解 DSP 的硬件结构，而作为 DSP 软件开发人员只有在了解 DSP 硬件结构的基础上，才能合理地设计程序流程、分配存储空间以及设置外围设备等，从而成功开发出高效的 DSP 软件。DSP 的硬件结构可以通过四个方面来了解：总线结构、存储器配置、CPU 结构和片上外设。

2.1.1　总线结构

在数字信号处理的运算中，常见的相关函数计算、卷积运算、信号滤波和各种变换算法大多可以归结为 $y=\sum a_i x_i$ 的乘加运算，因此 $y=x+a\times b$ 的形式出现最为频繁，所以 DSP 内部的结构设计都以优化上述乘加运算为主要目的，而 DSP 的总线结构是实现快速乘加运算的基石。

在大多数微处理器以及单片机中，一般采用传统的冯·诺依曼（Von Neumann）结构，图 2-1(a)给出了冯·诺依曼总线结构的示意图。如图所示，冯·诺依曼结构包含地址和数据两组总线，这两组总线是由程序空间和数据空间共享的，因而程

序取指和数据读写只能分时进行，从而在高速数据运算时容易造成瓶颈效应。与冯·诺依曼结构相区别的另一种总线结构为哈佛（Harvard）结构，如图 2-1(b)所示。在哈佛总线结构中，程序空间和数据空间相对独立，并且分别拥有各自的地址总线和数据总线，因而 CPU 可以在程序取指的同时，进行数据读写操作，从而加强了 CPU 并行工作的能力。因此哈佛结构是现代 DSP 总线结构的一种基本形式。



（a）冯·诺依曼结构；（b）哈佛结构

图 2-1　总线结构

从式 $y=\sum a_i x_i$ 中可以看出，每次乘加运算 CPU 至少需要两个数据 a_i 和 x_i，因此为了保证在程序取指的同时，能同时获得操作数 a_i 和 x_i，DSP 的结构中至少需要两套总线来同时访问数据空间。在当前的绝大多数 DSP 中，为保证在最短的时间内完成乘加运算，一般都拥有两套或两套以上的数据总线，这大大提高了 CPU 访问数据的能力，如 TI 公司的 TMS320C54x 系列 DSP 就拥有 3 套数据总线，CPU 可以在一个时钟周期内完成两次操作数读和一次数据写。

在图 2-1(b)所示的哈佛结构中，数据空间和程序空间相对独立，两者之间只能通过 CPU 进行通信，但是在一些应用中我们可能需要数据空间和程序空间有相互访问的能力，比如在 DSP 的一些循环指令运行时，程序总线处于空闲状态，为了提高 DSP 的访问能力，此时我们希望能用程序总线来访问数据空间，为此提出了一种改进的哈佛结构，如图 2-2 所示。在改进的 Harvard 结构中，数据总线和程序总线之间增加了一个交互的桥，因此 DSP 可以在不需要读取程序时，利用程序总线来访问数据以提高效率。

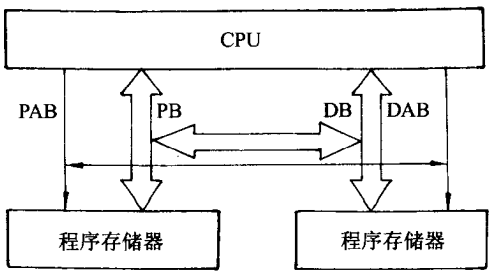


图 2-2　改进的哈佛结构

2.1.2　存储器配置

DSP 内部大都有自己的存储器,从使用功能上可以分为程序空间和数据空间,从访问形式上又可以分为只读存储器 ROM 和随机存取存储器 RAM。程序空间一般以页为单位，而数据空间一般以块为单位，块的大小各种 DSP 都不一样。ROM 部分主要包含系数表格、自举程序和生产厂商的测试代码等，还有一部分是没有内容的，留待为用户自定义的代码或数据表格作掩膜用。 RAM 部分用来存放运行时的程序和数据，一般有 SARAM 和 DARAM 两类,其中 DARAM 可以在同一时刻进行两次访问，往往用来存放需频繁访问的数据，而 SARAM 更多的是用来存放程序和表格。虽然 DSP 内部一般都配置有一定容量的 ROM,但是这些 ROM 只有在用户大规模需要同一芯片实现同一功能时才会要求生产厂商为自己的程序作掩膜使用。而在通常的 DSP 系统开发中，用户程序和数据表格一般存放在片外的各类 ROM 或主机中，在系统上电时再通过自举过程将用户程序和表格下载到片内的 RAM 中。为了增加系统的紧凑程度，在有些 DSP 内部配置有闪存（Flash ROM）,如 TI 公司的 TMS320F240、Agere 公司的 DSP1609 等,用户可以通过 JTAG 将程序烧录到 Flash ROM 中，从而可以省却用来存放程序和表格的片外 ROM。

DSP 片内的 RAM 不仅用来存放数据，也往往用来存放程序，因此片内 RAM 的大小会直接影响 DSP 的性能。RAM 的工作方式有两种，一种是当作高速缓存（Cache）使用，它只是当前使用到的程序或数据在片上的一个镜像，其内容随程序进行而动态变化，用户一般无法预测；另一种是作为固定地址映射的存储器使用，其内容由用户指定。前一种方法主要是针对程序或数据量大的应用而设计的，后一种方法针对消耗存储资源相对小的应用。一般在中低档 DSP 中,片内 RAM 只能作为固定地址映射的存储器使用，如 TI 公司的 TMS320C2xx、TMS320C5x、TMS320C54x、Agere 公司的 DSP16h、DSP16k、ADI 公司的 ADSP-21xx 等，只有一些高档 DSP 的片内 RAM 才支持这两种方式，如 TI 公司的 TMS320C6x 系列

DSP。当内部 RAM 作为固定地址映射的存储器，DSP 指令所消耗的时间可以根据指令的运行过程准确估计；而当作为高速缓存使用时，如果某次操作的数据在缓存中则情况与前者相同，反之则需要从外部读取且时间无法精确推算，因此指令运行的时间就无法预测了。在片内存储器资源不够的情况下， DSP 允许在外部扩展存储器，但是片外存储器的使用会明显影响 DSP 的性能，这包括两方面的原因。一方面，与不断加快的 DSP 内部时钟相比，外部存储器的速度相对较慢，因而在访问外部存储器时，DSP 一般需要插入额外的等待时钟。以一个 100MHz 运行的 DSP 为例，若外部存储器的访问速度为 10ns，为保证正确存取，DSP 需插入一个等待时钟。另一方面，虽然 DSP 内部往往配有多套总线，但由于受功耗和管脚的限制，其外部总线一般只有一套，因而程序取指和数据读写只能分时进行，从而难以发挥 DSP 内部哈佛结构所固有的高效率。因此在条件允许的情况下，为保证 DSP 运行效率，通常都将要访问的数据和程序放置在 DSP 内部,这就对 DSP 软件开发人员提出了巨大的挑战。

2.1.3 CPU 结构

每一个 DSP 的 CPU 是计算的真正执行者，它结构上的优劣很大程度上决定了 DSP 的性能。DSP 的 CPU 一般包括以下几个单元：算术逻辑单元（ALU)、累加器、桶形移位器、乘法器、加法器、比较选择存储单元（CSSU)、数据地址产生单元和程序地址产生单元等。ALU 实现二进制补码运算和布尔(Boolean)运算。累加器的运用极其灵活，它可以用来存储 ALU 和乘加模块的输出，也可以作为 ALU 的输入，有些 DSP 的累加器还可以作为乘加模块的输入。桶形移位器的输入一般与累加器或数据空间相连，输出与 ALU 或数据空间相连，它用来实现输入数据的移位，包括算术左移、算术右移、逻辑左移和逻辑右移等。包含硬件乘法器是 DSP 的一个重要特点，乘法器一般与加法器一起构成乘加单元（MAC)，它可以在单指令周期内完成两个一定长度数据的补码乘和一次加法。 MAC 和 ALU 可以并行工作，因而从硬件上保证 DSP 可以高效地实现卷积、求相关和滤波等各种数字信号处理算法。CSSU 完成数据的比较及测试功能。数据地址产生单元和程序地址产生单元为 DSP 提供多种灵活的寻址方式，包括一些特殊的寻址，如循环寻址、位翻转寻址等。另外，为优化循环结构， CPU 中通常包含特殊的硬件模块，为 DSP 提供零开销的循环操作。除此之外，CPU 中还提供一些专门的算术操作，如饱和、归一化处理等，大大增强了 DSP 的运算能力。

为减少指令独立运行的时间，DSP 的 CPU 往往采用流水线技术。所谓流水线是指将指令分成几个不同的阶段，然后 CPU 在不同时刻完成同一指令的不同阶段。流水线可以有两种方式：一种是 DSP 先对工作主频进行倍频，然后在每个倍频周期内完成指令的一个阶段，而整条指令一般在一个主频周期内完成，实际上

这并不是一种真正意义上的流水线工作方式；另一种是保持主频不变，将一条指令的不同阶段分配在连续的几个指令周期内完成，而在一个指令周期内， CPU 执行不同指令的不同阶段，从而使每条指令真正独立运行的时间减少到最低。图 2-3 给出了后一种流水线技术的一个示例。图中假设一条指令分为取指、译码、取数、执行四个阶段，即流水线为四级，由图可知 CPU 在执行第 N 条指令的同时，执行第 N+1、N+2、N+3 条指令的取数、译码和取指。两种流水线工作方式各有优缺点，前一种方式需要对主频进行倍频，因而限制了主频速度的提高，但是由于每条指令都是在一个主频周期内完成，因此不会产生流水线冲突问题；而在后一种方式中，即使主频保持不变，只要加深流水线级数，也能不断地提高 DSP 的性能，但是流水线级数加深必然会加剧流水线冲突，因此 CPU 的流水线级数应该合理选择，以获得最佳的性能。

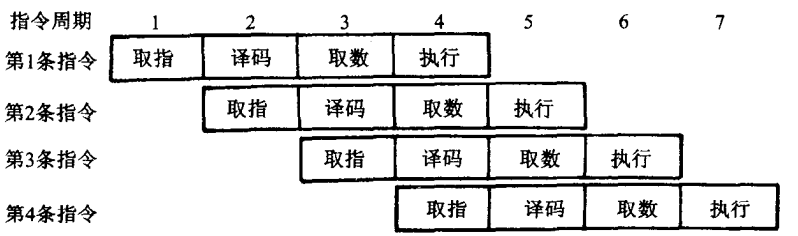


图 2-3　流水线示意图

2.1.4　片上外设

由于 DSP 主要从事快速的数学计算，所以其接收数据和发送数据的速度也相对较快，其外设也以接口快速设备为主要目的。通常，片上外设包括主机接口（HPI）、同步串行口、直接存储器访问（DMA）、外部存储器接口、中断逻辑、硬件定时器、时钟发生器、锁相环等。各种 DSP 的片上外设往往各不相同，其接口和特性也不尽相同，本书的第 5 章将以 TI 公司的 TMS320C54x 系列 DSP 为例对各类外设作详细介绍。

总而言之，DSP 的硬件结构完全是为保证 DSP 进行高速运算而设计的，其基本特点如下：整体采用哈佛结构或改进的哈佛结构；一般 RAM 作为固定地址映射的存储器使用，因此指令运行时间可以预测；CPU 的运算单元能够完成如 MAC 等复杂计算；片上外设主要接口快速设备，通常包含同步串行口。

2.2 数字信号处理器的软件特征

DSP 指令是硬件结构的客观体现，并且是 DSP 软件的基石，熟悉、掌握和灵活运用指令是每个 DSP 软件工程师的必备技能。DSP 种类繁多，相应的指令也各不相同，然而就指令集的特性而言，在当前各种处理器中主要包括复杂指令集（CISC）和精简指令集（RISC）两类。下面先简单介绍 CISC 和 RISC，然后分析 DSP 的指令特性及其软件特征。

2.2.1 复杂指令集（CISC）

CISC 是复杂指令集计算机的简称，它是芯片设计的一种思想，目的是使编程容易并有效利用存储空间。 CISC 的每一条指令可以在处理器内部执行一系列操作，从而可以减少为实现给定任务所需要的指令数，同时程序员只需学习小而灵活的指令集。由于最早的机器以汇编语言进行编程，并且存储器速度慢而昂贵，因而 CISC 思想不仅在大型计算机设计而且在许多通用微处理器设计中被普遍遵循 。

为增强性能，在 CISC 设计中主要是采用了三个重要技术：使用微指令、建立丰富的指令集、建立高级指令集。

1) 使用微指令

在最早的处理器设计中，采用专用的硬件逻辑来译码和执行处理器指令集中的每条指令，这种方法对于仅包含少量寄存器的简单设计工作良好，但是当控制路径逻辑难以实现时，将使得复杂结构难以建造。为此，设计者们在处理器内部引入了一些简单逻辑来控制处理器不同单元间的数据通道，并用简化的微指令集来控制这些数据通道逻辑，这就是所谓的微程序实现。在微程序系统中，主处理器内部配有存储器（一般是 ROM），这些内部存储器包含一组与每条机器指令相对应的微指令，到机器指令到达中央处理器时，处理器就执行相应的一系列微指令。由于从内部 ROM 取指的速度比从主存储器要快上 10 倍，因而设计者们开始将尽可能多的指令做成微指令形式，而且有些处理器还可以根据用户要求为某类应用特制微指令。使用微指令不仅能大大加快处理器的速度，而且使得修改微程序设计以处理全新的指令集变得非常快速。

2) 建立丰富的指令集

使用微程序设计的一个理由是设计者可以让每条指令完成更多的功能，这不仅可以减少实现某一任务所需的指令数以有效利用相对慢速的主存储器，而且使

汇编程序的编制变得更加容易。同时设计者们还在不断增强面向汇编语言程序员的指令，包括字符串操作指令、特殊循环结构以及特殊寻址方式等。

3) 建立高级指令集

为了建立编程友好的指令集，设计者们构造了直接从高级语言映射过来的指令集，这不仅可以简化编译器编写者的任务，同时使每行源代码编译出较少的指令。

正是这三种技术的合理应用，使得 CISC 思想成为到 20 世纪 80 年代末期为止所有计算机设计的驱动力，并且到如今依旧是处理器设计领域的一种主流。

从总体上说，CISC 的指令集设计方便了汇编语言程序员，但硬件设计比较复杂。CISC 指令集的特点包括：采用两操作数形式，即指令一般包含一个源操作数和目的操作数；包含寄存器到寄存器、寄存器到存储器和存储器到寄存器命令；多种寻址方式；变长度指令；多周期指令。相应地为配合指令集的这些特点，CISC 一般具有：比较复杂的译码逻辑、较少的通用目的寄存器、一些专用寄存器等。

CISC 的设计要求每条指令在下一条指令开始之前执行完毕，许多处理器将每条指令分成取指、译码、执行、写存储器四个阶段，各个阶段依次进行。在理想情况下 CISC 所能达到的最快速度是每条指令完整执行只需一个周期，但是 CISC 并不禁止多周期指令的存在，因为 CISC 的设计思想是让处理器在每个时钟周期内执行复杂的功能以最小化总周期数。

综上所述，CISC 具有如下几个优点：

- 微程序编写如汇编语言一样容易实现，并且比硬件实现控制单元便宜得多；
- 新指令用微指令编码容易使得 CISC 可以方便地实现向下兼容；
- 随着每条指令的功能增强，实现一个给定任务所需的指令数变得更少；
- 由于微程序指令集可以编写得与高级语言结构相匹配，因此可以简化编译器。

当然 CISC 也具有如下几个缺点：

- 为保证指令的向下兼容，随着处理器的不断升级，指令集和芯片硬件变得越来越复杂；
- 不同指令需要不同的执行时间，有可能会降低系统的整体性能；
- 许多特殊指令不被经常使用，通常在一个程序中只有指令集中将近 20% 的指令被使用；
- CISC 通常将条件码作为指令的副产物，这不仅使条件码的设置耗费时间，同时要求程序员在随后的指令改变条件码之前必须检查这些条件码。

2.2.2 精简指令集(RISC)

20 世纪 70 年代中期半导体技术的发展开始减小主存储器和处理器芯片之间的速度差别。随着存储器速度的增加，高级语言替代了汇编语言，同时随着处理器的发展 CISC 指令集和实现这些指令的硬件结构越来越复杂，另外认识到一系列简单指令可以产生与一系列复杂指令相同的结果，而简单指令可以用更简单和快速的硬件设计实现，在处理器设计领域出现了另一种设计思想，即精简指令集计算机 RISC。目前，普遍认为 RISC 应该是计算机设计的基本原则，它的出现标志着计算机体系结构发展上的一个重要里程碑。

RISC 的特点如下：

- 简单指令集。在 RISC 中，指令集仅包含简单、基本的指令，而复杂的指令可由它们的组合得到：
- 相同指令长度，从而使取指可以一次操作完成；
- 单周期指令，便于流水操作，使处理器可以同时处理几条指令；
- 大量的寄存器。

RISC 作为处理器的一种设计思想，还在不断地发展和丰富，到目前为止对 RISC 设计思想的说明主要有两种，分别来自于卡内基梅隆（Carngie Mellon）大学和 IEEE 的迈克尔·斯莱特(Michael Slater)，归纳如下：

- 指令集中的大多数指令只需执行简单和基本的功能，其执行过程在一个机器周期内完成。
- 由于存储器访问指令执行时间长，应尽量减少这类指令。一般采用加载／存储指令结构，即只保留加载指令和存储指令。面向运算部件的操作数都经过加载指令和存储指令，从存储器取出后预先放在寄存器内，以加快执行速度。
- 芯片逻辑不采用或少采用微程序技术，而采用硬布线逻辑，以减少指令解释的开销。
- 减少指令数和寻址方式，使控制部件简化，加快执行速度。
- 采用对称的和统一的或三地址指令格式，内部包含较多的寄存器，使优化编译器便于生成优化代码。
- 引入延迟转移和加载延迟，以适合流水线有效执行。
- 为解决存储器与处理器之间的速度问题，经常采用高速缓存（Cache）技术。

与 CISC 相比，RISC 具有如下优点：

- 速度。在相同的半导体技术和时钟速率下，采用简化指令集可使流水工作的、超标量体系结构设计的 RISC 性能达到 CISC 的 2～4 倍。
- 更简单的硬件。由于 RISC 处理器的指令集简单，它使用的芯片面积要小得多，从而一些额外的功能模块如存储器管理单元等可以放在同一块芯片上。较小

的芯片使半导体生产厂商可以在单一硅片上放置更多的部件，从而大大地降低每一块芯片的成本。

- 更短的指令周期。由于 RISC 处理器比相应的 CISC 处理器简单，因此它可以设计得更加快。

虽然 RISC 具有诸多优点，但是当从 CISC 设计策略向 RISC 设计策略过渡时，软件工程师需要警惕代码从 CISC 处理器转向 RISC 处理器时出现的几个关键问题：

- 代码质量。RISC 处理器的性能很大程度上依赖于它所执行的代码，由于 RISC 采用流水操作，当指令需要用到上一条指令的结果而上一条指令还未执行完时，就会出现指令延迟现象，为此需要对指令进行重新排序，这增加了编程人员的困难。
- 调试困难。指令的重新排序会使程序的可读性降低，同时增加了调试困难。
- 代码扩展。由于 RISC 采用同一格式且相同长度的指令，其效率较低，因此当从 CISC 处理器转向 RISC 时，程序代码的长度有可能增加很大。

2.2.3 DSP 指令及编程

DSP 内部采用哈佛结构，包含有多套总线，这使得 DSP 有着区别于 CISC 和 RISC 的硬件结构，其相应的指令系统也有着自己的特点，它综合了 CISC 和 RISC 的优点。大多数 DSP 指令是复合指令，即一条指令可以同时完成几个操作，最典型的是乘加运算，它在一个机器周期内可以同时完成两个操作数取、一次乘、一次加、寄存器写以及移位等功能，同时 DSP 一般具有多种灵活的寻址方式，指令长度和指令执行时间也可以不一样，在这些方面 DSP 指令集与 CISC 有很多相似之处。另一方面 DSP 又具有 RISC 的优良性能，如 DSP 一般都采用流水操作，大部分指令都能在单周期内执行完毕。TI 公司的 TMS320C6x 系列 DSP 更是一种仿 RISC 处理器，它采用超长指令字 VLIW（Very Long InstructionWord）架构，每条指令只完成简单的任务，通过简单指令的并行运行来提高性能，而不管是单条简单指令，还是由简单指令并行而构成的复杂指令，经编译后所有指令具有相同的长度。另外与 CISC 和 RISC 不同的是，大部分 DSP 的片内存储器都是固定地址映射的存储器（TMS320C6x 系列 DSP 除外，TMS320C6x 的片内存储器可以由用户配置成固定地址映射存储器，也可以配置成 Cache），因而指令的操作时间可以严格预测。

与 CISC 处理器相似，DSP 所采用的复合指令形式便于软件工程人员编制出高效率的汇编程序。作为汇编语言主体的指令集大多采用助记符指令形式或算术指令形式，因而其程序的可读性较差。另外为了防止和减少流水线冲突，与 RISC

相似，软件人员往往需要对指令进行重新排序，这进一步降低了 DSP 程序的可读性。与高级语言依附于操作系统不同， DSP 指令集与 DSP 的硬件结构密切相关，因而不同型号 DSP 的指令集一般互相不兼容，所以汇编语言编制的程序可移植性也较差。为弥补这个缺陷，随着 DSP 速度的不断加快以及存储空间的不断增大，越来越多的 DSP 支持高级语言编程，大大增强了 DSP 程序的可读性和可移植性，为程序的维护和修改提供了方便。采用高级语言编制 DSP 程序，将使指令的调度由编译程序负责而不是 CPU 负责，因而冲突和数据依赖性也将由编译程序处理，为此要获得能充分利用 DSP 资源的高效率代码，在很大程度上受到编译器性能的影响，面向 DSP 的高级语言编译器将是今后 DSP 领域一个热门的研究方向。

由于 DSP 一般承担相对简单的任务，所以其软件通常不依赖操作系统，是一个独立的程序，因而软件人员必须自己设计软件结构、维护堆栈、保护中断或调用的现场、设置中断优先级等。目前在 DSP 软件设计中依旧延续着任务驱动的思路，主流程是一个消息处理的循环，不停地检测外部信号和内部事件，根据检测结果确定所需要完成的任务。为充分利用 DSP 资源，软件编程中还有一项重要的任务就是优化程序，合理地选择指令、选择寻址方式、设计数据存放位置等都能够使程序运行时间减少。

如前所述，在 DSP 编程中，为了获得良好的效率大多数程序都采用汇编语言编程，而程序的接口一般也由编程人员自行设计，因而随着个人习惯不同在软件接口规范上会相差较大。即使是编程时都遵循高级语言如 C 语言的调用规范，但其在接口参数等方面也各不相同，因而造成很多资源间无法协调，特别是在集成来自多个公司的软件时，该问题显得尤为突出。针对这种情况， DSP 开发人员正在将一些高级语言编程规范和软件工程概念移植到 DSP 软件设计中,使之更加系统化。其中比较典型的是 TI 公司制定的一套软件接口规范—XpressDSP Algorithm Interface Standard ，简称 XDAIS，为软件资源的融合构筑了一个通用的平台，从而规范了 DSP 编程，为软件资源的再利用，以及 DSP 应用的快速开发都创造了良好的条件。

2.3 数字信号处理器的发展

2.3.1 DSP 的历史

DSP 的发展历史大致可以分成四个时期：萌芽期、成长期、成熟期、突破期。

萌芽期指 1982 年以前的一段时期，在这段时期里为解决冯·诺依曼结构在进行数字信号处理时总线和存储器之间的瓶颈效应，许多公司投入大量人力和物力开展了很多探索工作，如表 2-1 所示。但这些产品的运算速度都太慢，而且开发

工具严重不足，无法进行大规模的开发工作，不是真正意义上的 DSP。

表 2-1　关于 DSP 的探索工作

时间	公　　司	产品
1978	American Microsystems, Inc　(到 1983 年左右才正式发布)	S2811
1979	Intel Corp　(有片上的 A/D 和 D/A，但没有乘法器)	2920
1979	AT&T　(仅供内部研发)	DSP-1
1980	Ricoh / AMI　(重新设计 S2811)	S28211
1980	NEC	uPD7720

第一片 DSP 是 1982 年 TI 公司出品的 TMS32010，它是一个 16 位的定点 DSP，采用了哈佛结构，有一个乘加器和一个累加器。 TMS32010 完成一次乘加操作需要 390ns，即在一秒钟的时间内可以完成 250 万次左右的乘加运算。或许正是因为生产出了第一个 DSP，TI 公司在此后的三十几年中，一直是 DSP 界的领军人物。

成长期指 1982～1987 年，这段时间内各公司相继研制出了自己的 DSP，并不断地改进，具备了一个雏形。 1985 年，TI 推出了 TMS32020，它具备单指令循环的硬件支持，寻址空间达到 64K 字，有专门的地址寄存器，一次乘加运算只需耗时 200ns。1987 年，Motorola 公司推出了 DSP56001，采用 24 位的数据和指令，有专门的地址寄存器，可以循环寻址，累加器有保护位，一次乘加运算只需耗时 75ns 此外，在成长期中还有一些代表产品，如 AT&T 的 DSP16A、AD 的 ADSP-2100 TI 的 TMS320C50。

成熟期指 1987～1997 年，这段时间里各公司不断借鉴相互的优点，并完善自身的设计，推出了特点分明的产品，如 TI 的 TMS320C54 系列（简写为 C54x 或 C5400）、AD 的 ADSP2100 系列、Lucent（前身为 AT&T）的 DSP1600 系列和 Motorola 的 DSP56000 系列。它们在供电上都支持 3.3V，片上的存储器也较大，都有 JTAG 模块支持用户在线调试。另外，TI 等公司还专门提供 DSP 的内核，为一些用集成电路（ASIC）的开发提供了空间。在这段时间，首次出现了多处理器的 DSP，如 TI 的 TMS320C80 和 Motorola 的 MC68356 等，虽然它们的推出在商业上并不算成功，但却指明了一个有潜力的发展方向。

突破期指 1997 年以后到现在，这段时间里 DSP 的发展非常迅速，各公司相继建立了自己从定点到浮点，从低端到高端，从通用到专用的完整的产品系列，并且在 DSP 设计上有了大的飞跃，推出了一些性能突出的产品。很多公司相继采用先进技术研制了计算性能很高的 DSP，如 AD 的 Blackfin 系列、TI 的 TMS320C6000 系列和 Agere（前身为 Lucent 微电子）的 StarPro 等，每秒钟可以完成 1G 条以上的指令，计算速度惊人。TI 公司还研制出功耗最小的 DSP