第一部分

DSP原理

第 1 章 绪论

习近平总书记在党的二十大报告中指出：“教育、科技、人才是全面建设社会主义现代化国家的基础性、战略性支撑。必须坚持科技是第一生产力、人才是第一资源、创新是第一动力，深入实施科教兴国战略、人才强国战略、创新驱动发展战略，开辟发展新领域新赛道，不断塑造发展新动能新优势。”

数字信号处理(Digital Signal Processing，简称 DSP)技术是现代科技发展的重要组成部分，它的应用范围广泛，涵盖了通信、音视频处理、医疗、环境监测、图像处理等各个领域。随着科技的不断进步，DSP 技术也在不断地发展和升级，从最初的数字滤波器到现在的高级算法和芯片设计，业已成为现代科技的核心。

DSP 技术的发展对于现代科技的发展有着重要的推动作用。它不仅推动了通信、音视频处理、医疗、环境监测、图像处理等各个领域的技术创新和应用，还促进了全球产业链的升级和转型，推动了国际贸易和经济的发展。此外，DSP 技术在国防和军事领域的应用也对国家的安全产生着重要影响。

为了更好推动我国 DSP 技术的发展，需要采取一系列措施。首先，需要加强 DSP 技术的基础研究和应用研究，提高自主创新能力。其次，需要加强 DSP 技术的人才培养和引进，培养和吸引一批高水平的专业技术人才。此外，还要加强 DSP 技术的国际合作和交流，推动 DSP 技术的全球化发展。

1.1 引言

数字信号处理是一种对数字信号进行处理的技术。它是利用计算机或专用处理设备，以数字的形式对信号进行分析、采集、合成、变换、滤波、估算、压缩、识别等加工处理，以便提取有用的信息并进行有效的传输与应用。

DSP 的发展可以追溯到 20 世纪 50 年代。然而，由于硬件技术的限制，数字信号处理理论的应用受到了很大的限制，直到 20 世纪 80 年代，第一片数字信号处理器的诞生才使得数字信号处理理论得到了广泛的应用。数字信号处理器的出现及发展极大地推动了数字信号处理的理论和应用领域的发展。

DSP 技术的发展历史可以分为以下几个阶段：

1. 早期阶段(1950s–1970s)

DSP 技术最初应用于军事领域，主要用于雷达信号处理和通信。在这个阶段，DSP 技术主要依赖于模拟电路和模拟计算机。

2. 中期阶段(1970s–1980s)

随着计算机技术的发展，DSP 技术开始向数字化方向发展。数字信号处理器的出现使得数字信号处理技术得到了快速发展。数字信号处理器是一种专门用于数字信号处理的微处理器，它的出现使得数字信号处理速度得到了大幅提高。

3. 现代阶段(1990s 至今)

随着计算机和通信技术的快速发展，以及大规模集成芯片的日益更新，DSP 技术在通信、音频、视频、医学图像处理、雷达、控制系统等领域中得到广泛应用，已经深刻地改变了我们的生活。

1.2 DSP 概述

DSP 可以代表数字信号处理技术(Digital Signal Processing)，也可以代表数字信号处理器(Digital Signal Processor)。前者是理论和计算方法上的技术，后者是指实现这些技术的通用或专用可编程微处理器芯片。数字信号处理是一种利用计算机或专用处理设备，对数字信号进行分析、采集、合成、变换、滤波、估算、压缩、识别等加工处理，以便提取有用的信息并进行有效的传输与应用。

DSP 技术主要包括以下几个方面：

1. 数字信号滤波

数字信号滤波是指对数字信号进行滤波去除不需要的信号成分的过程。数字信号滤波可以分为数字滤波器和模拟滤波器两种类型。数字滤波器是通过数字信号处理器实现的，而模拟滤波器是通过模拟电路实现的。

数字滤波器主要分为有限冲激响应(Finite Impulse Response，简称 FIR)滤波器和无限冲激响应(Infinite Impulse Rresponse，简称 IIR)滤波器两种类型。FIR 滤波器的特点是稳定性好、易于设计、相应速度较快，但需要更多的计算资源。IIR 滤波器的特点是计算资源少、相应速度快，但是不稳定性较差、易于产生振荡。

2. 数字信号变换

数字信号变换是指将一个数字信号转换为另一个数字信号的过程。常见的数字信号变换包括傅里叶变换、离散余弦变换、小波变换等。数字信号变换可以用于信号压缩、特征提取、频谱分析等领域。

傅里叶变换是将一个信号分解成一些基本频率的和的过程。离散余弦变换是一种将信号分解成基本余弦函数的和的过程。小波变换是一种将信号分解成一些小波函数和的过程。小波变换的优点是可以同时提供时域和频域信息，因此在图像处理领域得到广泛

应用。

3. 数字信号编码和解码

数字信号编码是将数字信号转换为特定编码形式的过程。常见的数字信号编码包括脉冲编码调制(Pulse Code Modulation，简称 PCM)、压缩编码、差分编码等。数字信号解码是将数字信号编码转换为原始数字信号的过程。

PCM 是将数字信号转换为脉冲序列的过程。差分编码是将数字信号转换为两个相邻样本之间的差值的过程。压缩编码是将数字信号进行压缩，以减少存储空间和传输带宽。

4. 数字信号处理算法

数字信号处理算法是指用于数字信号处理的数学算法。常见的数字信号处理算法包括滤波算法、变换算法、压缩算法、特征提取算法等。

滤波算法主要用于对数字信号进行滤波处理，来改善信号的质量、消除信号中的噪声，以及抽取有用信息。变换算法则是将数字信号进行变换的算法，以实现数字信号在时域和变换域之间的转换。压缩算法是将数字信号进行压缩处理，一般分为压缩和还原(解压缩)两个步骤。特征提取算法是从数字信号中提取有用信息和主要特征的算法。

DSP 技术在各个领域中都有广泛的应用。在通信领域，DSP 技术主要应用于调制解调、信道编码、信号检测等方面，以提高通信质量和可靠性。在音频、视频领域，DSP 用于音频、视频压缩、降噪等方面，以提高音视频的质量和传输效率。在医学图像处理领域，DSP 技术常用于医学图像分析、诊断等方面，以提高医学诊断的准确性和效率。在控制系统中，DSP 被广泛应用于控制算法设计、信号处理等方面，以提高控制系统的性能和稳定性。由此可见，数字信号处理技术在各个领域中都具有重要的应用价值。随着技术的不断发展，数字信号处理技术将继续发挥越来越重要的作用。

数字信号处理包括算法的研究和数字信号处理的实现两方面的内容。

1.2.1　算法的研究

数字信号处理算法的研究是 DSP 技术的重要组成部分。数字信号处理算法是指对数字信号进行处理的数学方法和技术，主要包括数字信号的采样、量化、变换、滤波、编码、解码、压缩、识别等方面。这些算法的研究是 DSP 技术能够实现高效处理的关键。数字信号处理算法的研究需要掌握数学、信号处理、计算机科学等多个学科的知识，需要深入理解信号处理的基本原理和方法，以及掌握各种数字信号处理算法的设计和实现技巧。

近年来，数字信号处理的理论和方法得到了迅速的发展，涉及到语音和图像的压缩编码、识别和鉴别，信号的调制和解调、加密和解密，信道的辨识和均衡，智能天线，频谱分析等多个领域。这些算法成为了研究的热点，并取得了长足的进步，为各种实时处理的应用提供了算法基础。这些算法的发展为数字信号处理技术的应用提供了更加可靠和高效的解决方案，推动了数字信号处理技术的不断发展和创新。

1.2.2 数字信号处理的实现

数字信号处理的实现是DSP技术的另一个重要组成部分。数字信号处理的实现是指利用数字信号处理器或其他专用的数字信号处理设备，将数字信号处理算法转化为可执行的程序或指令，以实现数字信号的加工处理。数字信号处理器是一种专门用于数字信号处理的微处理器，它的出现极大地推动了数字信号处理的理论和应用领域的发展。数字信号处理器的核心是一组高速的算术逻辑单元和存储器，它可以高效地执行各种数字信号处理算法，实现数字信号的高速处理和实时处理。

数字信号处理的实现需要掌握数字信号处理器的架构和指令集，以及各种数字信号处理算法的实现技巧。数字信号处理器的架构和指令集的设计和实现是DSP技术的关键，它决定了数字信号处理器的性能和功能。数字信号处理器的指令集包括各种数字信号处理算法的指令，如加、减、乘、除、滤波、变换、编码、解码、压缩、识别等指令，以及各种控制指令和数据传输指令。数字信号处理器的指令集的设计和实现需要考虑到数字信号处理算法的特点和需求，以及数字信号处理器的资源和限制。

数字信号处理的实现方法主要有以下几种：

（1）通用计算机的CPU：优点是软件丰富、编程实现较容易，但不易实现实时信号处理和嵌入式应用。计算机的CPU由于成本高、功耗大，因此只有在系统对体积、功耗和实时性要求不高的情况下采用，且只适合于固定场合使用。

（2）单片机（MCU）：优点是价格低廉、编程简单，但性能差、速度慢。单片机一般适用于数字处理要求较简单、数据量小的小型控制系统。

（3）专用集成电路（ASIC）：将特定的信号处理算法由一个集成电路来实现，处理速度快、系统规模化成本低，但功能有限、系统灵活性差、开发成本高，目前只适合处理任务不很复杂而要求大批量生产的应用情况。例如FFT专用集成电路芯片。

（4）可重构系统（RCPS）：采用FPGA等现场可编程门阵列可以根据需要组建不同的系统逻辑，但成本高只适合实现小规模简单系统，不便于大量应用。

（5）DSP处理器：DSP处理器是专门针对数字信号处理应用而设计的微处理器，在处理器结构和工作原理上采用乘法累加运算单元（MAC）、多总线和多流水线作业，具有高效、灵活、功耗低、便于嵌入式应用、性价比高等特点，成为当前数字信号处理系统设计中采用的主流处理器。

1.3 DSP芯片

DSP芯片是一种专门用于数字信号处理的微处理器，具有高效、灵活、功耗低、便于嵌入式应用等特点。DSP芯片的结构和工作原理采用乘法累加运算单元（MAC）、多总线和多流水线作业，使得信号处理高效和灵活。此外，DSP芯片还具有高速缓存、DMA控制器、中断控制器等特殊功能模块，能够更好地支持数字信号处理算法的实现和优化。

DSP 芯片的应用优势在于其高效、灵活、功耗低、便于嵌入式应用等特点，能够更好地支持数字信号处理算法的实现和优化，提高系统的性能和可靠性。同时，随着 DSP 芯片的不断发展和创新，其性能和应用范围也在不断拓展和提高，为 DSP 技术的发展和应用带来了更多的可能性和机遇。

1.3.1　DSP 芯片的发展概况

DSP 芯片的发展可以追溯到 20 世纪 70 年代，当时出现了第一款 DSP 芯片——TI 公司的 TMS32010。该芯片采用了 16 位固定点运算，主要应用于通信和音频领域。此后，DSP 芯片的发展主要经历了三个发展阶段：

第一阶段是 1980 年前后，是 DSP 的雏形阶段。1978 年，AMI 公司生产了第一片 DSP 芯片 S2811；1979 年，美国 Intel 公司发布了商用可编程 DSP 器件 Intel2920。这些芯片由于内部没有单周期的硬件乘法器，运算速度和数据处理能力受到了限制，应用领域仅局限于军事或航空航天部门。

这个时期的代表性芯片主要有：Intel2920(Intel)，PD7720(NEC)，TMS32010(TI)、DSP16(AT&T)，S2811(AMI)，ADSp—21(AD)等。

第二阶段是 1990 年前后，是 DSP 的成熟阶段。这个时期的 DSP 器件在硬件结构上更适合数字信号处理的要求，能进行硬件乘法、硬件 FFT 变换和单指令滤波处理，其单指令周期为 80～100ns。如 TI 公司研发的 TMS320C20 芯片，它是该公司的第二代 DSP 器件，采用了 CMOS 制造工艺，其存储容量和运算速度成倍提高，为语音处理、图像硬件处理技术的发展奠定了基础。20 世纪 80 年代后期，以 TI 公司的 TMS320C30 为代表的第三代 DSP 芯片问世，伴随着运算速度的进一步提高，其应用范围逐步扩大到通信、计算机领域。

这个时期的代表性芯片主要有：TI 公司的 TMS320C20、30、40、50 系列，Motorola 公司的 DSP5600、9600 系列，AT&T 公司的 DSP32 等。

第三阶段是 2000 年以后，是 DSP 的完善阶段。这一时期各 DSP 制造商不仅提高了信号处理能力，而且使系统开发更加方便、程序编辑调试更加灵活、功耗进一步降低、成本不断下降。尤其是各种通用外设集成到片上，大大地提高了数字信号处理能力。此时的 DSP 芯片运算速度可达到单指令周期 10ns 左右，可在 Windows 环境下直接用 C 语言编程，使用方便灵活。DSP 芯片不仅在通信、计算机领域得到了广泛的应用，而且逐渐渗透到人们日常消费领域。

当前，DSP 芯片的发展速度非常迅猛。在硬件方面，DSP 芯片主要向多处理器的并行处理结构、便于外部数据交换的串行总线传输、大容量片上 RAM 和 ROM、程序加密、增加 I/O 驱动能力、外围电路内装化、低功耗等方面发展。这些硬件方面的进步，为 DSP 芯片的应用提供了更加广泛的可能性。在软件方面，综合开发平台的完善，使得 DSP 的应用开发更加灵活方便。现在，DSP 芯片的软件开发已经可以直接使用 C 语言，而且还可以通过各种开发工具进行模拟和调试。这些软件方面的进步，为 DSP 芯片的应用开发提供了更加便利的条件。

1.3.2 DSP 芯片的特点

数字信号处理与普通的科学计算与分析不同，它强调运算的实时性。为了满足数字信号处理实时性的特点，DSP 芯片在处理器的结构、指令系统、指令流程上作了很大的改进，其主要特点如下：

1. 采用哈佛结构

DSP 芯片普遍采用数据总线和程序总线分离的哈佛结构或改进的哈佛结构，比传统处理器的冯・诺伊曼结构有更快的指令执行速度。

（1）冯・诺伊曼(Von Neumann)结构

传统处理器采用冯・诺伊曼结构，即程序指令和数据共用一个存储空间，使用单一的地址和数据总线，取指令和取操作数都是通过一条总线分时进行。由于这种结构无法同时进行取指令和取操作数，因此在进行高速运算时会出现数据传输通道的瓶颈现象，导致工作速度较慢。

（2）哈佛(Harvard)结构

哈佛结构采用双存储空间，即程序存储器和数据存储器分开，有各自独立的程序总线和数据总线，可独立编址和独立访问，可对程序和数据进行独立传输，使取指令操作、指令执行操作、数据吞吐并行完成，大大提高了数据处理能力和指令的执行速度，非常适合于实时的数字信号处理。

（3）改进型的哈佛结构

改进型的哈佛结构采用双存储空间和多条总线，即一条程序总线和多条数据总线。它允许在程序空间和数据空间之间相互传送数据，使这些数据可以由算术运算指令直接调用，增强了芯片的灵活性。此外，改进型的哈佛结构还提供了存储指令的高速缓冲器(cache)和相应的指令，当重复执行这些指令时，只需读入一次就可连续使用，不需要再次从程序存储器中读出，从而减少了指令执行所需要的时间。这些特点使得改进型的哈佛结构在数字信号处理领域得到广泛应用。

2. 采用多总线结构

DSP 芯片普遍采用多总线结构，该结构能够同时进行取指令和多个数据存取操作，并由辅助寄存器自动增减地址进行寻址，使 CPU 在一个机器周期内可多次对程序空间和数据空间进行访问，大大提高了 DSP 的运行速度。例如，TMS320C54x 系列内部有 P、C、D、E 等 4 组总线，每组总线中都有地址总线和数据总线，因此在一个机器周期内，可以完成：从程序存储器中取一条指令、从数据存储器中读两个操作数、向数据存储器写一个操作数等操作。

3. 采用流水线技术

DSP 芯片采用流水线技术，将指令执行过程分为多个阶段，每个阶段由不同的硬件电路完成，从而实现指令的并行处理。流水线技术可以大大提高 DSP 芯片的运算效率，使其

能够在一个时钟周期内执行多条指令。同时，流水线技术还可以减少指令执行的等待时间，提高 DSP 芯片的吞吐量。但是，流水线技术也会带来一些问题，如指令间的依赖关系、流水线的中断处理等，需要通过一些技术手段进行优化和解决。

4. 配有专用的硬件乘法-累加器

DSP 芯片配有专用的硬件乘法-累加器(MAC)，它是一种高速运算器件，能够在一个时钟周期内完成乘法和加法运算，并将结果存储在累加器中。与通用处理器相比，DSP 芯片的 MAC 具有更高的运算速度和更低的功耗，能够快速地处理数字信号处理中的复杂运算，如滤波、调制、解调等。同时，DSP 芯片的 MAC 还支持多种数据格式，如定点数、浮点数等，能够满足不同应用场景的需求。

5. 具有特殊的 DSP 指令

为了满足数字信号处理的需要，在 DSP 的指令系统中，设计了一些完成特殊功能的指令。这些指令是针对数字信号处理场景进行优化的，能够快速地执行各种常见的数字信号处理操作，如滤波、变换、乘法累加等。例如 FIRS 和 LMS 指令，就是专门用于完成系数对称的 FIR 滤波器和 LMS 算法。与通用处理器相比，DSP 指令具有更高的运算速度和更低的功耗，能够有效地提高 DSP 芯片的运算效率和处理能力。

6. 快速的指令周期

DSP 芯片具有快速的指令周期，这是由于 DSP 芯片内部采用了专门的硬件电路和优化的指令集，能够在一个时钟周期内完成多个指令的执行。与通用处理器相比，DSP 芯片的指令周期更短，能够以更高的速度进行运算和处理，同时还能够支持多种数据格式和复杂的运算操作。由于数字信号处理应用对运算速度和实时性要求较高，因此 DSP 芯片的快速指令周期使其成为数字信号处理领域的理想选择。此外，DSP 芯片还可以通过一些技术手段，如流水线技术和指令预取技术等，进一步提高指令执行的效率和速度。例如 TMS320C54x 芯片的运算速度为 100MIPS，即 100 百万条/秒。

7. 硬件配置强大

新一代的 DSP 芯片的硬件配置强大，其内部集成了多个专用硬件模块，如乘法累加器、DMA 控制器、中断控制器等，能够快速地执行各种数字信号处理操作，并实现高效的数据传输和处理。此外，DSP 芯片还具有多组数据总线和程序总线，支持多种数据格式和复杂的运算操作，能够满足数字信号处理领域对高性能、高效率、高精度的要求。与通用处理器相比，DSP 芯片的硬件配置更加专用化和优化，能够在数字信号处理领域发挥出更大的优势。

8. 支持多处理器结构

DSP 芯片支持多处理器结构，具有高度的可扩展性和灵活性，这种结构可以将多个 DSP 芯片通过高速接口连接起来，形成一个 DSP 系统，能够共同完成复杂的数字信号处理任务。多处理器结构可以大大提高 DSP 系统的运算速度和处理能力，使其能够应对更加复杂的数字信号处理场景。同时，多处理器结构还可以实现任务的并行处理，提高 DSP 系统的效率和实时性。在多处理器结构中，各个 DSP 芯片之间可以通过高速总线进行数据交换

和协同计算，从而实现更加高效的数据传输和处理。例如 TMS320C40 芯片提供了 6 个用于处理器间高速通信的 32 位专用通信接口，使处理器之间可直接对通，应用灵活、使用方便。

9. 省电管理和低功耗

DSP 芯片具有省电管理和低功耗的特性，这主要是由于其内部采用了一些节能技术和低功耗设计。例如，DSP 芯片可以通过动态电压调节技术和时钟门控技术等，实现对芯片电压和时钟频率的动态调整，从而降低功耗和热量的产生。此外，DSP 芯片还支持多种省电模式，如睡眠模式、待机模式等，能够在不影响系统性能的情况下降低功耗。在数字信号处理领域，由于 DSP 系统通常需要长时间运行，因此省电管理和低功耗设计对于延长系统寿命、降低系统成本和提高系统可靠性具有重要意义。

1.3.3 DSP 芯片的分类

DSP 芯片是一种专门用于数字信号处理的芯片，它可以高效地处理数字信号，广泛应用于音频、视频、通信、雷达等领域。根据不同的分类标准，DSP 芯片可以分为多种不同的类型。

1. 根据应用领域分类

根据应用领域的不同，DSP 芯片可以分为音频处理、视频处理、通信处理、雷达处理等多种类型。其中，音频处理 DSP 芯片主要用于音频信号的采集、处理和输出，如音频编解码器、音频滤波器等；视频处理 DSP 芯片主要用于视频信号的处理和编解码，如视频压缩解压缩芯片、视频滤波器等；通信处理 DSP 芯片主要用于无线通信、有线通信等领域，如调制解调器、信道编解码器等；雷达处理 DSP 芯片主要用于雷达信号的处理和分析，如雷达信号处理器、雷达控制器等。

2. 根据处理能力分类

根据处理能力的不同，DSP 芯片可以分为低端、中端和高端三种类型。低端 DSP 芯片的处理能力较弱，主要用于一些简单的数字信号处理应用，如音频信号的处理、数字滤波等；中端 DSP 芯片的处理能力较强，可以处理复杂的数字信号，如视频编解码、通信信号处理等；高端 DSP 芯片的处理能力非常强，可以处理高速、高精度的数字信号，如雷达信号处理、医学图像处理等。

3. 根据内核类型分类

根据内核类型的不同，DSP 芯片可以分为定点型、浮点型和混合型三种类型。定点型 DSP 芯片采用定点运算方式，具有较高的运算速度和较低的功耗，主要用于一些实时性要求较高的应用，如音频信号处理、通信信号处理等；浮点型 DSP 芯片采用浮点运算方式，具有较高的运算精度和较低的误差，主要用于一些需要高精度计算的应用，如科学计算、图像处理等；混合型 DSP 芯片则将定点型和浮点型的优点结合起来，具有较高的运算速度和较高的运算精度，主要用于一些复杂的数字信号处理应用，如雷达信号处理、医学图像处理等。

4. 根据外设接口分类

根据外设接口的不同，DSP 芯片可以分为串行接口、并行接口、USB 接口和以太网接口等多种不同类型。串行接口 DSP 芯片，主要用于串行通信接口，如 I^2C 总线接口、SPI 接口等；并行接口 DSP 芯片，主要用于并行通信接口，如 PCI 总线接口、ISA 接口等；USB 接口 DSP 芯片，主要用于 USB 通信接口，支持设备的即插即用和热插拔功能，是目前常用的通信接口；以太网接口 DSP 芯片，主要用于网络通信接口等。

1.3.4 TI 公司的 DSP 产品

TI 公司 DSP 产品主要分为 TMS320C2000 系列、TMS320C5000 系列、TMS320C6000 系列。

1. TMS320C2000 系列

TMS320C2000 系列 DSP 又称为数字信号控制器(Digital Signal Controller，DSC)，该系列芯片具有高性能、低功耗、易于开发等特点，被广泛应用于工业控制、汽车电子、电机控制、航空航天、家用电器和消费电子等领域。

TMS320C2000 系列芯片采用了 TI 自主研发的 C28x 内核，采用 32 位 RISC 架构，可以实现高速的浮点运算和定点运算。此外，该系列芯片还支持多种存储器类型，包括片内闪存、RAM、EEPROM 等，可以满足不同应用的存储需求。

TMS320C2000 系列芯片还具有丰富的外设资源，包括模数转换器(ADC)、数字模拟转换器(DAC)、通信接口、定时器等。其中，ADC 和 DAC 可以实现高速、高精度的模拟信号采集和输出，通信接口支持多种通信协议，如 SPI、I^2C、CAN 等，定时器可以实现精确的定时和计数功能。

除了硬件资源的丰富，TMS320C2000 系列芯片还具有易于开发的特点。TI 提供了完整的开发工具链，包括 CCS 集成开发环境、C 语言编译器、仿真器等，可以大大降低开发难度和时间。

该系列 DSP 目前主要由 TMS320C24x 和 TMS320C28x 组成，所有 TMS320C2000 平台控制器均实现了全面的软件兼容性。

2. TMS320C5000 系列

TMS320C5000 系列 DSP 是为实现低功耗、高性能而专门设计的定点 DSP 处理器。主要用于音频处理、语音识别、图像处理等领域。

TMS320C5000 系列芯片采用了 C55x 内核，该内核采用了 16 位定点运算方式，具有高效的运算能力和低功耗的特点。同时，该内核还支持多达 8 个并行运算功能单元，可实现高速的浮点运算和定点运算。此外，该系列芯片还拥有丰富的存储器资源，包括片内 RAM、闪存、ROM 等，可满足各种应用的存储需求。

TMS320C5000 系列芯片具备丰富的外设资源，如 ADC、DAC、通信接口和定时器等，可满足不同应用的需求。此外，该系列芯片易于开发，TI 提供了大量的应用代码和参考设计，可快速构建基于 TMS320C5000 系列芯片的应用系统。

该系列使用最广泛的是 TMS320C54x 和 TMS320C55x 两大类，这两类处理器软件完全兼容，均为 16 位定点数字处理器。

3. TMS320C6000 系列

TMS320C6000 系列芯片是 TI 公司推出的高性能 DSP。采用 TI 新的超长指令字结构 VLIW(Very Long Instruction Word)，使该系列 DSP 的性能达到很高的水平，并且支持定点浮点兼容。

该系列主要面向图像、视频、网络和无线宽带通信等需要大规模数据处理的应用领域，例如，视频会议系统、高清晰数字电视、无线局域网、安防视频监控等。

TMS320C6000 系列主要包括 TMS320C62x、TMS320C64x、TMS320C66x、TMS320C67x 等。TMS320C62x、TMS320C64x 为 16 位定点数字信号处理器，TMS320C67x 为 32 位浮点信号处理器，TMS320C66x 为融合了定点和浮点处理能力的多核处理器。

1.4 DSP 系统

1.4.1 DSP 系统的基本构成

DSP 系统是由 DSP 芯片和周边电路组成的。其基本构成包括输入信号处理、DSP 芯片、存储器和输出信号处理四个部分。

输入信号处理部分是将模拟信号转换为数字信号的过程。通常采用模数转换器(ADC)将模拟信号转换为数字信号，然后将数字信号输入到 DSP 芯片中进行处理。在输入信号处理部分中，还可以包括前置滤波器、放大器等电路，以保证输入信号的质量和稳定性。

DSP 芯片是整个系统的核心部件，它负责对输入信号进行数字信号处理。DSP 芯片通常由运算器、存储器、时钟和外设等组成。其中，运算器用来执行各种算法和运算，存储器用来存储程序和数据，时钟用来控制芯片的运行，外设用来与其他设备进行通信。

存储器是 DSP 系统中非常重要的一部分，它用来存储程序和数据。DSP 芯片通常具有片内存储器和片外存储器两种类型。片内存储器通常包括 RAM 和 ROM 两种，用来存储程序和数据。片外存储器通常包括闪存、SDRAM 等，用来扩展存储容量。

输出信号处理部分是将数字信号转换为模拟信号的过程。通常采用数字模拟转换器(DAC)将数字信号转换为模拟信号，然后经过输出滤波器等电路进行滤波和放大，最终输出到外部设备中。

除了基本构成外，DSP 系统还可以添加各种外设和接口，如通信接口、定时器、PWM 输出等，以满足不同应用的需求。同时，还可以通过软件编程来实现各种算法和功能，如滤波、变换、控制等。

图 1-1 示出了一个典型的 DSP 系统基本结构框图。一个典型的 DSP 系统应包括抗混叠滤波器、数据采集 A/D 转换器、数字信号处理器 DSP、D/A 转换器和低通滤波器等。

同一般的微处理器应用系统类似，DSP 应用系统除了 DSP 处理器之外，还必须有能够与其他系统和器件连接的接口。

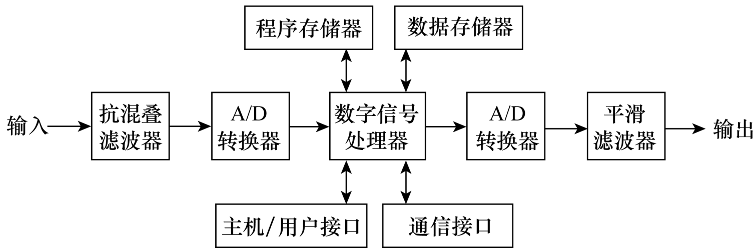

图 1-1　DSP 系统基本结构框图

上述的 DSP 系统模型是一个典型的模型，然而并非所有的 DSP 系统都必须具有模型中的所有部件，因为不同的应用场景需要不同的系统设计。下面是对 DSP 系统模型的一些补充说明：

(1) 输入信号处理部分：输入信号处理部分的主要作用是将模拟信号转换为数字信号，但是有些输入信号本身就是数字信号，如 CD(Compact Disk)，此时就不需要进行模数转换了。而在语音识别系统中，输出端并不是连续的波形，而是识别结果，如数字、文字等，因此输入信号处理部分可能还需要进行信号预处理、降噪等操作。

(2) DSP 芯片：DSP 芯片是整个系统的核心部件，它负责对输入信号进行数字信号处理。不同的 DSP 芯片具有不同的处理能力和资源，因此需要根据具体应用场景选择适合的 DSP 芯片。

(3) 存储器：存储器用来存储程序和数据，包括片内存储器和片外存储器两种类型。在一些小型的 DSP 系统中，可能只需要使用片内存储器，而在一些大型的 DSP 系统中，则需要使用片外存储器扩展存储容量。

(4) 输出信号处理部分：输出信号处理部分的主要作用是将数字信号转换为模拟信号输出，但是在一些应用场景中，输出信号可能是数字信号，如数字信号处理系统、通信系统等。

综上所述，在实际应用中，需要根据具体应用场景进行系统设计，可能需要增加或减少模型中的部件，以满足不同应用的需求。

1.4.2　DSP 系统的特点

与传统的模拟信号处理系统相比，DSP 系统具有以下特点：

1. 接口简单、容易实现

数字信号的电气特性非常简单，不同的 DSP 系统相互连接时，在硬件接口上容易实现。此外，在数据流接口上，各系统只需遵循特定的标准协议即可相互连接。这种简单方便的接口设计，为数字信号处理系统的应用和开发提供了便利。

2. 精度较高、稳定性好

与模拟信号处理不同的是，数字信号处理过程仅受到量化误差和有限字长的影响，不

会引入其它噪声，因此具有较高的信噪比。此外，模拟系统的性能会受到元器件参数性能影响比较大，而数字系统的性能基本不变，因此数字系统更便于测试、调试及批量生产。

3. 可编程性强、易实现复杂算法

在 DSP 系统中，DSP 芯片提供了一个高速计算平台，系统功能依赖于软件编程实现。当现代信号处理理论和计算数学与 DSP 系统相结合时，可以实现复杂的数字信号处理功能。这种编程方便的特点使得数字信号处理系统具有更高的灵活性和可扩展性，能够适应不同领域和应用的需求。

4. 集成方便、可靠性高

现代 DSP 芯片将 DSP 芯核及其外围电路综合集成在单一芯片上，这种集成结构使得数字信号处理系统的设计更加简化和便捷，同时具有高度的可靠性和稳定性，可以有效地提高数字信号处理系统的性能和效率。此外，这种集成结构也便于设计便携式高集成度的数字产品，如智能手机、平板电脑、数码相机等。这些数字产品不仅具有更高的性能和功能，而且还具有更小的尺寸和更轻便的重量，更加符合现代人们的生活需求。

1.4.2 DSP 系统的设计过程

DSP 系统的设计流程如图 1-2 所示。设计过程包括以下几个步骤：

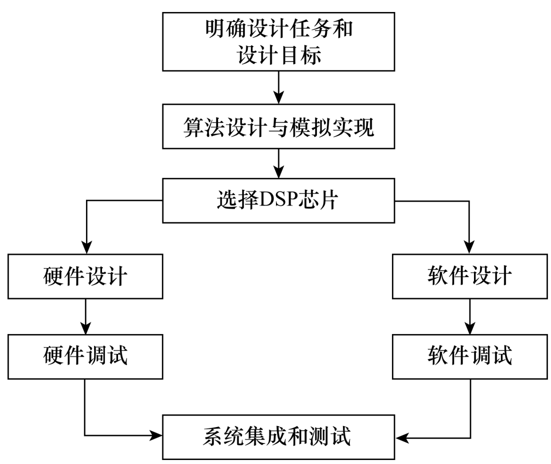

图 1-2 DSP 系统设计流程图

1. 明确设计任务和设计目标

在设计 DSP 系统之前，首先要明确系统的设计任务，确定系统的性能指标以及信号处理的要求，如运算速度、运算精度、存储器片内硬件资源等。这些要求通常可用数据流程图、数学运算序列、正式的符号或自然语言来描述。这一步骤是数字信号处理系统设计的基础，也是后续设计工作的指导依据。

2. 系统算法设计与模拟实现

DSP 系统设计的第二步是根据系统要求进行算法设计与模拟实现。为了实现预定的目

标任务，需要对输入信号进行适当的处理，选择合适的数字信号处理算法，并考虑算法的复杂度、实现难度、精度等因素。不同的处理方法会导致不同的系统性能，为了得到最佳的系统性能，必须在这一步确定最佳的处理方法。因此，这一步也被称为算法模拟阶段。例如，语音压缩编码算法需要在确定的压缩比条件下，获得最佳的合成语音。算法模拟所用的输入数据是实际信号经采集而获得的，通常以计算机文件的形式存储为数据文件。有些算法模拟时所用的输入数据并不一定要是实际采集的信号数据，只要能够验证算法的可行性，输入假设的数据也是可以的。

3. 系统整体架构设计

DSP 系统的整体架构设计包括硬件设计和软件设计两个方面。硬件设计的第一步是选择合适的 DSP 芯片，考虑系统运算量的大小、对运算精度的要求、系统成本限制以及体积、功耗等要求。然后需要设计 DSP 芯片的外围电路及其他电路。软件设计和编程主要根据系统要求和所选的 DSP 芯片编写相应的 DSP 汇编程序，如果系统运算量不大且有高级语言编译器支持，也可以使用高级语言(如 C 语言)编程。由于现有的高级语言编译器效率还比不上手工编写汇编语言的效率，因此在实际应用系统中常常采用高级语言和汇编语言的混合编程方法。具体而言，算法运算量大的地方采用手工编写汇编语言的方法，而运算量不大的地方则采用高级语言。采用这种方法，既可缩短软件开发周期，提高程序的可读性和可移植性，又能满足系统实时运算的要求。

4. 系统硬件和软件调试

完成 DSP 硬件和软件设计后，需要进行相应的硬件和软件调试。硬件调试通常使用硬件仿真器进行，如果没有相应的硬件仿真器，也可以使用一般的工具进行调试。软件调试方面，通常使用 DSP 开发工具，例如软件模拟器、DSP 开发系统或仿真器等。在调试 DSP 算法时，一般采用比较实时结果与模拟结果的方法，以确保实时程序和模拟程序的输入相同、输出一致。其他应用系统的软件调试则需要根据实际情况进行调试。

5. 系统集成和系统测试

系统的硬件和软件调试完成后，可以进行系统集成，将软件脱离开发系统直接在应用系统上运行。然而，DSP 系统的开发，尤其是软件开发是一个反复进行的过程。尽管通过算法模拟可以基本了解实时系统的性能，但是模拟环境无法完全与实时系统环境一致。同时，将模拟算法移植到实时系统时，必须考虑算法是否能够实时运行的问题。如果算法运算量过大，不能在硬件上实时运行，就必须重新修改或简化算法，以满足实时运行的要求。系统集成和系统测试是 DSP 系统设计的最后一步，也是最重要的一步，能够保证系统的整体性能和稳定性。

总之，DSP 系统的设计是一个复杂而又系统的过程，需要设计人员具备扎实的理论基础和丰富的实践经验，同时还需要不断地学习和更新技术，以适应不断变化的市场需求和技术发展。

1.4.3 DSP 系统的发展趋势

随着科技的不断发展，DSP 系统也在不断演进和发展。在未来的发展趋势中，DSP 系

统将会朝着以下几个方面发展。

1. 更加高效和智能化

随着人工智能和大数据技术的不断发展，DSP 系统也将会更加智能化，能够更好地处理和分析海量数据。例如，在人工智能领域，DSP 系统可以用于语音识别、图像识别、自然语言处理等方面，能够更好地满足人们对于智能化的需求。同时，DSP 系统也将会更加高效，能够更快地处理数据和算法，提高系统的响应速度和处理能力。例如，在无线通信领域，DSP 系统可以用于信号处理和调制解调等方面，能够更好地提高通信效率和质量。

2. 更加可靠和安全

在未来的发展中，DSP 系统的可靠性和安全性将会成为一个重要的发展方向。DSP 系统将会更加注重数据的安全性和隐私保护，以及系统的稳定性和可靠性，避免出现系统故障和数据泄露等问题。例如，在网络安全领域，DSP 系统可以用于数据加密和解密、数据压缩和解压等方面，能够更好地保护数据的安全性和隐私。

3. 更加多样化和灵活化

未来的 DSP 系统将会更加多样化和灵活化，能够适应不同的应用场景和需求。例如，在汽车领域，DSP 系统可以用于车载音响和语音识别等方面，能够更好地提高驾驶体验和安全性。同时，DSP 系统也将会更加注重人机交互和自适应性，能够更好地适应用户的需求和操作习惯。例如，在智能家居领域，DSP 系统可以用于智能家居控制和语音交互等方面，能够更好地提高家居的智能化和便捷性。

4. 更加节能和环保

在未来的发展中，DSP 系统将会更加注重能源的节约和环境的保护。DSP 系统将会采用更加节能和环保的设计方案，例如采用低功耗芯片和可再生能源等技术，以减少能源的消耗和对环境的影响。例如，在智能电网领域，DSP 系统可以用于电力负荷预测和调度等方面，能够更好地提高电网的能源利用效率和环保性。

综上所述，未来的 DSP 系统将会更加高效、智能、可靠、安全、多样化、灵活化、节能和环保。这些发展趋势将会推动 DSP 系统在各个领域的广泛应用，为人们的生产和生活带来更多的便利和效益。同时，这些发展趋势也将会带来新的技术挑战和机遇，需要我们不断地探索和创新，以推动 DSP 系统的发展和进步。

习题

1. 简述典型的 DSP 系统的基本构成。
2. 数字信号处理有哪些实现方法？
3. DSP 芯片有什么特点？
4. DSP 系统的基本构成是什么？
5. DSP 系统设计的一般步骤是什么？

第 2 章 TMS320C55x DSP 的硬件结构

DSP 芯片的硬件结构与通用微处理器相似，由 CPU、存储器、总线、外设、接口、时钟等部分组成。DSP 芯片有其鲜明的特点，是特别适合于进行数字信号处理运算的微处理器。根据数字信号处理的要求，DSP 芯片一般具有如下主要特点：

(1)在一个指令周期内可完成一次乘法和一次加法；

(2)程序和数据空间分开，可以同时访问指令和数据；

(3)片内具有快速 RAM，通常可通过独立的数据总线在两块中同时访问；

(4)具有低开销或无开销循环及跳转的硬件支持；

(5)快速的中断处理和硬件 I/O 支持；

(6)具有在单周期内操作的多个硬件地址产生器；

(7)可以并行执行多个操作；

(8)支持流水线操作，使取指、译码和执行等操作可以重叠执行。

(9)与通用微处理器相比，DSP 芯片的其他通用功能相对较弱。

2.1 TMS320C55x 系列的基本结构

TMS320C55x 是一种新一代数字信号处理器，是在 C54x 基础上发展起来的。它具有低功耗和高性能的特点，并且软件具有 C54 兼容模式，这大大缩短了从 C54x 转向 C55x 的时间。

C55x 采用了新的半导体工艺，其工作时钟比 C54x 系列处理器更快。CPU 内部增加了功能单元，从而增强了 DSP 的运算能力。相比之下，C55x 具有更高的性能和更低的功耗。这些特点使得 C55x 在无线通信、便携式个人数字系统以及高效率的多通道数字压缩语音电话系统中得到广泛应用。

与 C54x 相比，C55x 在硬件方面做了许多扩展，具体如表 2-1 所示。

表 2-1　C55x 与 C54x 的比较

内容	C54x	C55x
乘法累加器(MAC)	1	2
累加器(ACC)	2	4

续表

内容	C54x	C55x
读总线	2	3
写总线	1	2
地址总线	4	6
指令字长	16 位	8/16/24/32/40/48 位
数据字长	16 位	16 位
算术逻辑单元(ALU)	1（40 位）	1（16 位） 1（40 位）
辅助寄存器字长	2 字节(16 位)	3 字节(24 位)
辅助寄存器	8	8
存储空间	独立的程序/数据空间	统一的程序/数据空间
数据寄存器	0	4

C55x 的一系列特征使它具有处理效率高、低功耗和使用方便等优点，如表 2-2 所示。

表 2-2　C55x 的特征及优点

特征	优点
一个 32 位×16 指令缓冲队列	缓冲变长指令并完成有效的块重复操作
两个 17×17 位的乘法累加器	在一个单周期执行双乘法累加操作
一个 40 位算术逻辑单元(ALU)	实现高精度算术和逻辑操作
一个 40 位桶形移位寄存器	能够将一个 40 位的计算结果最高向左移 31 位或向右移 32 位
一个 16 位算术逻辑单元(ALU)	对主 ALU 并行完成简单的算术操作
4 个 40 位的累加器	保留计算结果，减少对存储单元的访问
12 条独立总线，其中包括： 3 条读数据总线 2 条写数据总线 5 条数据地址总线 1 条读程序总线 1 条程序地址总线	为各种计算单元并行地提供将要处理的指令和操作数——利用 C55x 的并行机制的优点
用户可配置 IDLE 域	改进了低功耗电源管理的灵活性

2.1.1　TMS320C55x 的 CPU 体系结构

TMS320C55x CPU 的体系结构如图 2-1 所示，它包括 4 个单元：指令缓冲单元(I)、程序流程单元(P)、地址流程单元(A)和数据计算单元(D)。

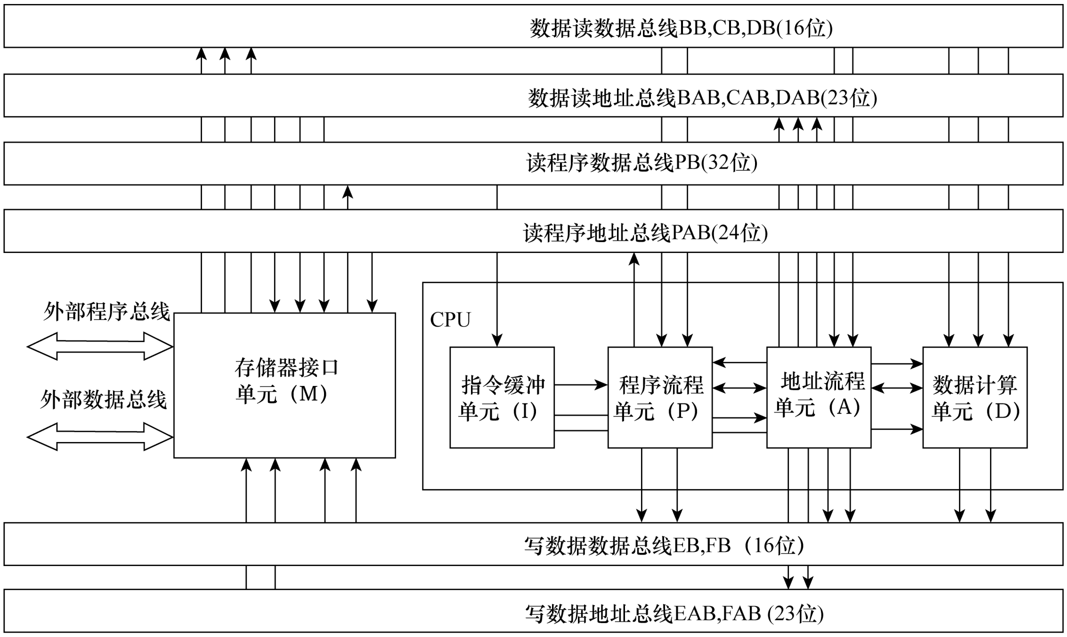

图 2-1　TMS320C55x 的 CPU 体系结构

该 CPU 使用 1 条 32 位程序数据总线(PB)、5 条 16 位数据总线(BB、CB、DB、EB、FB)以及 1 条 24 位程序地址总线(PAB)和 5 条 23 位数据地址总线(BAB、CAB、DAB、EAB、FAB)进行支持。这些总线分别与 CPU 内各单元相连。

总线通过存储器接口单元(M)与外部程序总线和外部数据总线相连，实现 CPU 对外部存储器的访问。这种并行的多总线结构，使 CPU 能在一个 CPU 周期内完成 1 次 32 位程序代码读、3 次 16 位数据读和 2 次 16 位数据写。

读程序地址总线(PAB)上传送 24 位的程序代码地址，由读程序数据总线(PB)将 32 位的程序代码送入指令缓冲单元 I 进行译码。

3 条读数据地址总线(BAB、CAB、DAB)与 3 条读数据数据总线(BB、CB、DB)配合使用，即 BAB 对应 BB、CAB 对应 CB 和 DAB 对应 DB。地址总线指定数据空间或 I/O 空间地址，通过数据总线将 16 位数据传送到 CPU 的各个功能单元。其中，BB 只与 D 单元相连，用于实现从存储器到 D 单元乘法累加器(MAC)的数据传送。特殊的指令也可以同时使用 BB、DB 和 CB 来读取三个操作数。

2 条写数据地址总线(EAB、FAB)与 2 条写数据数据总线(EB、FB)配合使用，即 EAB 对应 EB、FAB 对应 FB。地址总线指定数据空间或 I/O 空间地址，通过数据总线，将数据从 CPU 的功能单元传送到数据空间或 I/O 空间。所有数据空间地址由 A 单元产生。EB 和 FB 从 P 单元、A 单元和 D 单元接收数据，对于同时向存储器写两个 16 位数据的指令要使用 EB 和 FB，而对于完成单写操作的指令只使用 EB。地址总线和数据总线的功能如表 2-3所示。

表 2-3　地址总线和数据总线的功能

总线	宽度/位	功能
PAB	24	读程序的地址总线，每次从程序空间读时，传输 24 位地址
PB	32	读程序的数据总线，从程序存储器传送 4 字节的程序代码给 CPU
CAB、DAB	每组 24	这两组读数据的地址总线，都传输 24 位地址。DAB 在数据空间或 I/O 空间每读一次时传送一个地址，CAB 在两次读操作里送第二个地址
CB、DB	每组 16	这两组读数据的地址总线，都传输 16 位的数值给 CPU。DB 从数据空间或 I/O 空间读数据，CB 在读长类型数据或读两次数据时送第二个值
BAB	24	读数据的地址总线，读系数时传输 24 位地址。许多间接寻址模式来读系数的指令，都要使用 BAB 总线来查询系数值
BB	16	读数据的数据总线，从内存传送一个 16 位数据值到 CPU。BB 不和外存连接。BB 传送的数据，由 BAB 完成寻址某些专门的指令，在一个周期里用间接寻址方式，使用 BB、CB 和 DB 来提供 3 个 16 位的操作数。经由 BB 获取的操作数，必须存放在一组存储器里，区别于 CB 和 DB 可以访问的存储器组
EAB、FAB	每组 24	写数据的地址总线，每组传输 24 位地址。EAB 在向数据空间或 I/O 空间写时传送地址。FAB 在双数据写时，传送第二个地址
EB、FB	每组 16	写数据的数据总线，每组从 CPU 读 16 位数据。EB 把数据送到数据空间或 I/O 空间。FB 在写长类型数据或双数据写时传送第二个值

2.1.2　指令缓冲单元(I)

C55x 的指令缓冲单元由指令缓冲队列 IBQ(Instruction Buffer Queue)和指令译码器组成，如图 2-2 所示。在每个 CPU 周期内，指令缓冲单元将从读程序数据总线接收的 4 字节程序代码放入指令缓冲队列。指令译码器从队列中取出 6 字节程序代码，并根据指令的长度对 8 位、16 位、24 位、32 位和 48 位的变长指令进行译码。然后，指令译码器将译码数据送入程序流程单元、地址流程单元和数据计算单元去执行。

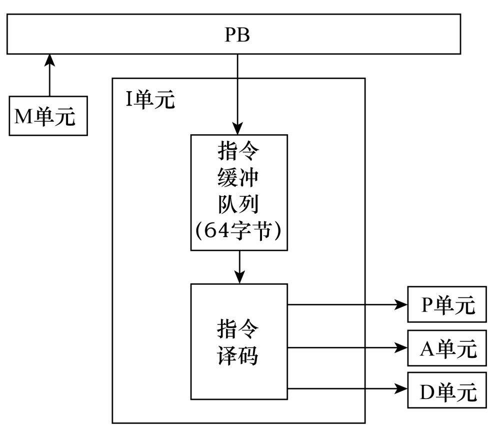

图 2-2　C55x 的指令缓冲单元

2.1.3 程序流程单元(P)

程序流程单元由程序地址产生电路和寄存器组构成，如图 2-3 所示。程序流程单元的主要任务是产生所有程序空间的地址，并控制指令的读取顺序。

程序地址产生逻辑电路的任务是生成 24 位读取程序空间的地址。通常情况下，它会生成连续地址。但如果指令要求读取非连续地址的程序代码，程序地址产生逻辑电路就能接收来自指令缓冲单元的立即数和来自数据计算单元的寄存器值，并将生成的地址传送到程序地址总线(PAB)。

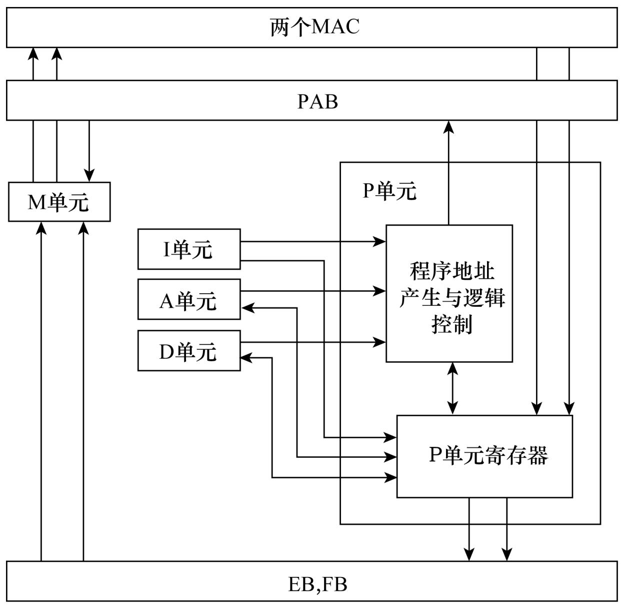

图 2-3 C55x 的程序流程单元

程序流程(P)单元中使用的寄存器分为五种类型：

(1)程序流寄存器：包括程序计数器(PC)、返回地址寄存器(RETA)和控制流程关系寄存器(CFCT)。

(2)块重复寄存器：包括块重复寄存器 0 和 1(BRC0，BRC1)、BRC1 的保存寄存器(BRS1)、块重复起始地址寄存器 0 和 1(RSA0，RSA1)以及块重复结束地址寄存器 0 和 1(REA0，REA1)。

(3)单重复寄存器：包括单重复计数器(RPTC)和计算单重复寄存器(CSR)。

(4)中断寄存器：包括中断标志寄存器 0 和 1(IFR0，IFR1)、中断使能寄存器 0 和 1(IER0，IER1)以及调试中断使能寄存器 0 和 1(DBIER0，DBIER1)。

(5)状态寄存器：包括状态寄存器 0、1、2 和 3(ST0-55、ST1-55、ST2-55 和 ST3-55)。

2.1.4 地址流程单元(A)

地址流程单元由数据地址产生电路、算术逻辑电路和寄存器组构成，它们的作用

如下：

数据地址产生电路(DAGEN)可以接收来自指令缓冲单元(I 单元)的立即数和来自地址流程单元(A 单元)的寄存器，用于产生读取数据空间的地址。如果指令采用间接寻址模式，程序流程单元(P 单元)会向数据地址产生电路说明采用的寻址模式。

A 单元包括一个 16 位的算术逻辑电路(ALU)，它可以接收来自指令缓冲单元的立即数，并与存储器、I/O 空间、A 单元寄存器、数据流程单元(D 单元)寄存器和程序流程单元寄存器进行双向通信。ALU 可以完成算术运算、逻辑运算、位操作、移位、测试等操作。

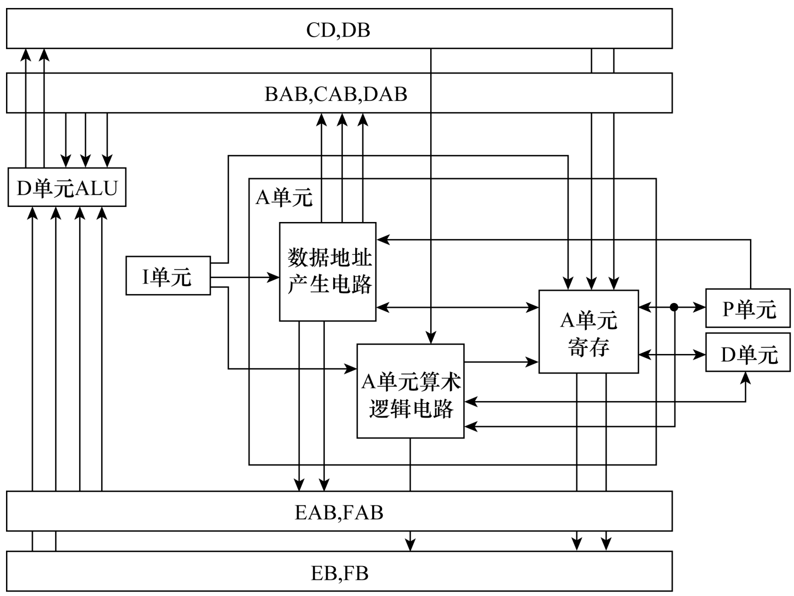

图 2-4　C55x 的地址流程单元

地址流程(A)单元包括以下几种类型的寄存器：

(1)数据页寄存器：包括数据页寄存器(DPH，DP)和接口数据页寄存器(PDP)。

(2)指针：包括系数数据指针寄存器(CDPH，CDP)、栈指针寄存器(SPH，SP，SSP)和 8 个辅助寄存器(XAR0～XAR7)。

(3)循环缓冲寄存器：包括循环缓冲大小寄存器(BK03，BK47，BKC)和循环缓冲起始地址寄存器(BSA01，BSA23，BSA45，BSA67，BSAC)。

(4)临时寄存器：包括临时寄存器(T0～T3)。

2.1.5　数据计算单元(D)

数据计算单元由移位器、算术逻辑电路、乘法累加器和寄存器组构成，如图 2-5 所示。D 单元包含了 CPU 的主要运算部件。

D 单元移位器能够接收来自 I 单元的立即数，能够与存储器、I/O 空间、A 单元寄存器、D 单元寄存器和 P 单元寄存器进行双向通信，此外，还可以向 D 单元的 ALU 和 A 单

元的 ALU 提供移位后的数据。

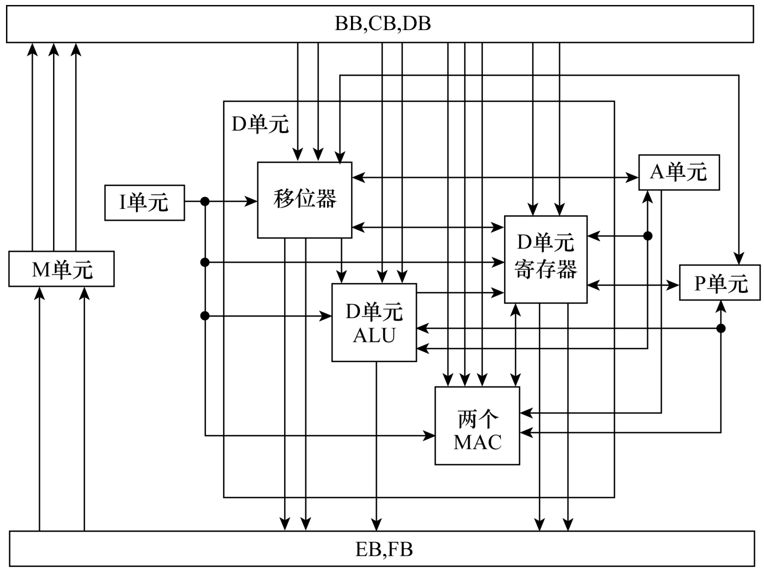

图 2-5　C55x 的数据计算单元

移位器可完成以下操作：

(1)对 40 位的累加器可完成向左最多 31 位和向右最多 32 位的移位操作，移位数可从临时寄存器(T0～T3)读取或由指令中的常数提供；

(2)对于 16 位寄存器、存储器或 I/O 空间数据可完成左移 31 位或右移 32 位的移位操作；

(3)对于 16 位立即数可完成向左最多 15 位的移位操作。

算术计算(D)单元的 40 位算术逻辑电路可完成以下操作：

(1)完成加、减、比较、布尔逻辑运算和绝对值运算等操作；

(2)能够在执行一个双 16 位算术指令时同时完成两个算术操作；

(3)能够对 D 单元的寄存器进行设置、清除等位操作。

两个 MAC 支持乘加和乘减操作，在一个周期内，每个 MAC 可同时完成 17 * 17 位的乘法和一个 40 位的加法或减法。

D 单元寄存器包括 4 个 40 位累加器 AC0-AC3 和两个过渡寄存器 TRN0、TRN1。

2.1.6　指令流水线

C55x CPU 采用指令流水线工作方式，C55x 的指令流水线包括两个阶段：

第一阶段是取流水线，即从内存中取出 32 位的指令包，放入指令缓冲队(IBQ)中，然后为流水线的第二阶段提供 48 位的指令包，如图 2-6 所示。

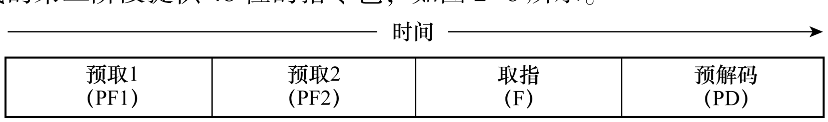

图 2-6　流水线的第一阶段(取流水线)

其中 PF1 表示向存储器提供的程序地址，PF2 表示等待存储器的响应，F 表示从存储器取一个指令包并放入指令缓冲队列中，PD 表示对指令缓冲队列中的指令预解码(确定指令的起始和结束位置，确定并行指令)。

第二阶段是指执行流水线，这部分的功能是对指令进行解码，完成数据的存取和计算，如图 2-7 所示。

时间 →

解码(D)	地址(AD)	存取1(AC1)	存取2(AC2)	读(R)	执行(X)	写(W)	写+(W+)

图 2-7　流水线的第二阶段(执行流水线)

流水线第二分段包括以下各阶段的执行情况：

(1) D 阶段：从指令缓冲队列中读取 6 个字节的指令，对一个指令对或一个单指令进行解码。将指令分配给对应的 CPU 功能单元，并读取与数据地址产生相关的位 ST1_55(CPL)、ST2_55(ARnLC)、ST2_55(ARMS)、ST2_55(CDPLC)。

(2) AD 阶段：读取或修改与数据地址产生相关的寄存器，例如 * ARx+(T0)中的 ARx 和 T0，BK03(如果 AR2LC=1)，SP(pushes 和 pops 过程中)，SSP(在 32 位栈模式中与对 SP 的操作一致)。在 A 单元的 ALU 中完成操作，例如使用 AADD 指令进行算术运算，用 SWAP 指令交换 A 单元中的寄存器，向 A 单元的寄存器写入常量(BKxx，BSAxx，BRCx，CSR 等)，在条件分支指令中如果 ARx 不等于 0，则 ARx-1。

(3) AC1 阶段：在存储器读操作中，将相应的 CPU 地址总线上传送地址。

(4) AC2 阶段：允许存储器对请求的响应是一个周期。

(5) R 阶段：从存储器和通过映射方式寻址的寄存器中读取数据；在 R 节拍执行 D 单元的预取 A 单元寄存器指令时，读取 A 单元的寄存器；在 R 节拍判断条件指令的条件。

(6) X 阶段：读取或修改不通过映射方式寻址的寄存器；读取或修改寄存器中的单个位；设置条件，如果指令不是向存储器中写，则判断 XCCPART 的条件；判断 RPTCC 指令的条件。

(7) W 阶段：向存储器映射方式寻址的寄存器或 I/O 空间写入数据；向存储器写入数据，从 CPU 来看，写操作在该节拍完成。

(8) W+阶段：向存储器写入数据，从存储器来看，写操作在该节拍完成。

2.2　TMS320VC5509A 的主要特性

VC5509A 是 C55x 系列的一种处理器，它内置了一个 C55x 内核和 128K×16 位的片上 RAM 存储器。此外，VC5509A 还具有最大 8M×16 位的外部寻址空间，并集成了 USB 总线、McBSP 和 I^2C 等外部接口。

2.2.1　VC5509A 的硬件组成

1. CPU

(1) 两个乘法累加单元(MAC)：每个乘法累加单元可在一个周期内处理 17 位×17 位

的乘法运算。

（2）40 位的算术逻辑单元(ALU)和一个 16 位的算术逻辑单元：算术逻辑单元具有并行处理能力，采用并行处理可以降低系统处理时钟从而降低功耗。这些资源由 CPU 中的地址单元(AU)和数据单元(DU)进行管理。

（3）采用先进的多总线结构：通过三条内部数据/操作数来读总线和两条内部数据/操作数来写总线，从而完成对指令和数据的访问。

2. 存储器

VC5509A 拥有 128K×16 位的片上 RAM，其中包括 64KB 的双端口 RAM(DARAM)和 192KB 的单端口 RAM(SARAM)。同时，它还具有最大 8M×16 位的外部寻址空间(同步 DRAM)。

VC5509A 的外部存储器接口(EMIF)与通用输入/输出(GPIO)共用引脚。当不使用外部存储器接口时，这些引脚可以被用作通用输入/输出接口。EMIF 接口可以无缝连接异步静态 RAM(SRAM)、异步 EPROM 和同步 DRAM(SDRAM)。

3. 片上外设

VC5509A 拥有两个 20 位的定时器和一个看门狗定时器。它还配备了 6 通道直接存储器存取控制器(DMA)，DMA 控制器可以在不需要 CPU 干预的情况下提供 6 路独立的通道用于数据传输，并且可达每周期两个 16 位数据的吞吐量。

外部存储器接口(EMIF)可以无缝连接异步存储器如 EPROM、SRAM 以及高密度存储器如同步 DRAM。

此外，VC5509A 还提供三个串口，支持最多三个多通道缓冲串口(McBSP)或最多两个多媒体/安全数字卡接口。三个全双工多通道缓冲串口(McBSP)提供了与各种工业级串行设备的无缝接口，其多通道通信最多可以实现 128 个独立通道。

增强型主机接口(EHPI)是一个 16 位的并行接口，主机能够提供 HPI 接口访问 5509A 上的 32KB 片上存储器。

此外，VC5509A 还配备了可编程锁相环(DPLL)时钟发生器、USB 全速(12Mbps)从端口、I^2C 主从接口和一个实时时钟。

2.2.2　VC5509A 的引脚

TMS320VC5509APGE 采用塑料四方扁平封装形式(LQFP)，共有 144 个引脚。这些引脚可以按其功能分为并行总线引脚、中断和复位引脚、位输入/输出信号引脚、I^2C 引脚、多通道缓冲串口信号引脚、USB 引脚、A/D 引脚、测试引脚和电源引脚等几部分。

（1）并行总线引脚：共 14 个引脚，可完成 HPI 地址总线、EMIF 地址总线或通用输入/输出功能。

（2）中断和复位引脚：INT[4：0]为低电平有效的外部中断输入引脚。

（3）位输入/输出信号引脚：GPIO[7：6，4：0]共 7 个输入/输出线，可单独配置成输入或输出引脚。

（4）时钟信号引脚：CLKOUT 为 DSP 时钟输出信号引脚；X2/CLKIN 为晶振输入引脚，X1 为内部振荡器输出引脚。

（5）I^2C 引脚：SDA 和 SCL 为 I^2C(双向)数据线和时钟引脚。

（6）McBSP 信号引脚：可与其他器件相连。

（7）USB 引脚：DP 和 DN 为差分(正/负)接收/发送引脚，PU 为上拉引脚。

（8）A/D 引脚：VC5509A 提供一个 10 位 A/D 转换器，AIN0 和 AIN1 为模拟输入通道 0 和 1。

（9）测试引脚：符合 IEEE1149. 1 标准的测试/仿真接口，包括 TCK、TDI、TDO 和 TMS 引脚。

（10）电源引脚：CVDD、DVDD、USBVDD、RCVDD 为数字电源，AVDD 为模拟电源，VSS 为数字地，AVSS 为模拟地，ADVDD 和 USBPLLVDD 为 10 位 A/D 数字部分和 USB 的 PLL 提供专用电源，ADVSS 和 USBPLLVSS 为 10 位 A/D 数字部分和 USB 的 PLL 接地。

2. 3　TMS320C55x 存储空间结构

TMS320C55x 存储空间结构是指该芯片内部的存储器组织方式，包括程序存储器、数据存储器、寄存器文件等。

TMS320C55x 的程序存储器包括两个部分：片内程序存储器和片外程序存储器。片内程序存储器包括指令存储器和常量存储器，指令存储器用于存储程序代码，常量存储器用于存储常量数据。指令存储器和常量存储器都采用哈佛存储结构，可以同时读取指令和数据，提高了程序的执行效率。片外程序存储器包括闪存、RAM 等，可以扩展程序存储器的容量。

TMS320C55x 的数据存储器包括片内数据存储器和片外数据存储器。片内数据存储器包括数据 RAM 和寄存器文件，数据 RAM 用于存储程序运行时的变量和中间结果，寄存器文件包括累加器、数据寄存器、指针寄存器等，用于存储运算过程中的数据。片外数据存储器包括闪存、RAM 等，可以扩展数据存储器的容量。

TMS320C55x 的寄存器文件是其存储空间结构中的一个重要组成部分，包括累加器、数据寄存器、指针寄存器、状态寄存器等。累加器是 TMS320C55x 中最重要的寄存器之一，可以进行加、减、乘、移位等运算，是程序运行过程中的重要变量。数据寄存器用于存储数据，指针寄存器用于指向数据存储器中的地址，状态寄存器用于存储程序的状态信息。

总的来说，TMS320C55x 存储空间结构是一个复杂的组合体系，包括程序存储器、数据存储器、寄存器文件等多个部分，这些部分相互协作，完成数字信号处理的各种任务。了解 TMS320C55x 存储空间结构的组成和特点，对于程序员来说是非常重要的，可以帮助他们更好地理解和掌握该芯片的操作方式，从而提高程序的执行效率和性能。

表 2-4　C55x 片内存储器配置

外设或存储器	5501	5502	5503	5506	5507	5509	5510
ROM（KB）	32	32	64	64	64	64	32
RAM（KB）	32	64	64	128	128	256	320

2.3.1　C55x DSP 的存储器映射

C55x DSP 的存储器映射是指该芯片内部的存储器分配方式。C55x DSP 的存储器映射包括程序空间、数据空间、I/O 空间以及堆栈操作等。这些存储器空间相互协作，完成数字信号处理的各种任务。C55x DSP 的寻址空间为 16MB，当 CPU 从程序空间读取程序代码时，使用 24 位地址，当访问数据空间时，使用 23 位的地址。但是在访问数据空间时，将 23 位地址左移一位，并将地址总线上的最低有效位(LSB)置 0，使得在对数据空间或程序空间寻址时，地址总线都传送 24 位地址。

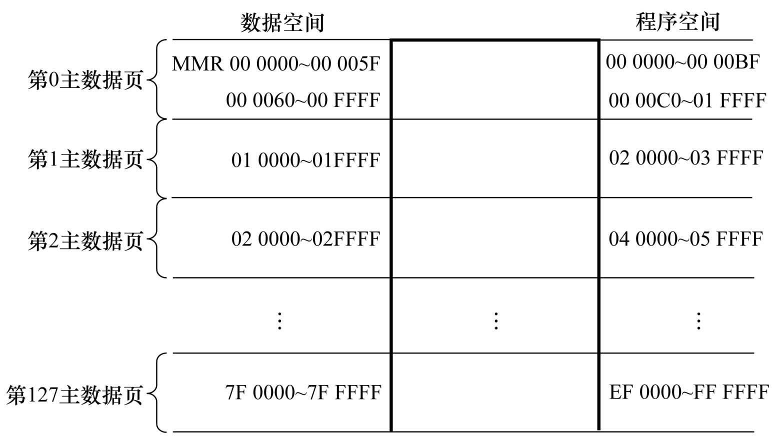

图 2-8　存储器映射

数据空间被分成 128 个主数据页(第 0 页到第 127 页)，每个主数据页的大小为 64K 字，指令通过 7 位的主数据页值和 16 位的偏移值共同来确定数据空间的任何一个地址。

在第 0 主数据页中，前 96 个地址(00 0000h～00 005Fh)为存储映射寄存器(MMR)保留，相对应在程序空间有 192 个地址(00 0000h～00 00BFh)，这段存储区为系统保留区，用户不能使用该区。

2.3.2　C55x DSP 的程序空间

C55x DSP 的程序空间包括程序存储器和常量存储器。程序存储器用于存储程序代码，常量存储器用于存储常量数据。C55x DSP 的程序存储器采用哈佛存储结构，即指令和数据存储在不同的存储器中，可以同时读取指令和数据，提高了程序的执行效率。程序空间的大小可以通过片外存储器进行扩展，满足不同应用的需求。当 CPU 读取指令时，程序

空间才被访问。CPU 采用字节寻址来读取变长的指令，指令的读取要和 32 位的偶地址对齐(地址的低两位为零)。

1. 字节寻址(24 位)

当 CPU 从程序空间读取指令时，采用字节寻址，即按字节分配地址，且地址为 24 位。一个行宽为 32 位存储器的地址分配由图 2-9 说明，每个字节分配一个地址，例如字节 0 的地址是 00 0100h，字节 2 的地址是 00 0102h。

字节地址 100~103h	字节0 00 0100h	字节1 00 0101h	字节2 00 0102h	字节3 00 0103h

图 2-9　存储器的地址分配

2. 程序空间的指令结构

C55x DSP 支持 8 位、16 位、24 位、32 位和 48 位长度的指令。表 2-5 和表 2-6 说明了指令在程序空间如何存放。在 32 位宽的存储器中存放了 5 条指令，每一条指令的地址是指操作码最高有效字节的地址，阴影部分表示没有代码。

表 2-5　指令长度及地址分配

指令	长度(位)	地址
A	24	000101h
B	16	00 0104h
C	32	00 0106h
D	8	00010Ah
E	24	00010Bh

表 2-6　存储器中的指令

字节地址	字节 0	字节 1	字节 2	字节 3
00 0100h～00 0103h		A（23～16）	A（15～8）	A（7～0）
00 0104h～000107h	B（15～8）	B（7～0）	C（31～24）	C（23～16）
00 0108h～00 010Bh	C（15～8）	C（7～0）	D（7～0）	E（23～16）
00 010Ch～00010Fh	E（15～8）	E（7～0）		

3. 程序空间的边界对齐

在程序空间存放指令时不需要边界对齐，当读取指令时要和 32 位的偶地址对齐。也就是说，在读取一条指令时，CPU 要从最低两位是 0 的地址读取 32 位的代码，这样的地址其最低位应是 0h，4h，8h 和 Ch。

CPU 不连续执行时，也会遇到写入到程序计数器 PC 中的地址值和程序空间的读取地址不一致的情况，例如，执行一个子程序 B：

CALL　B

假设子程序的第一条指令 C 的地址是 00 0106h，PC 的值是 00 0106h，但是读程序地址总线(PAB)上的值是 32 位边界地址 00 0104h，CPU 在 00 0104h 地址开始读取 4 字节的代码，而第一个被执行的指令是 C。

2.3.3　C55x DSP 的数据空间

C55x DSP 的数据空间包括数据存储器和寄存器文件。数据存储器包括数据 RAM 和数据 ROM，数据 RAM 用于存储程序运行时的变量和中间结果，数据 ROM 用于存储只读数据。寄存器文件包括累加器、数据寄存器、指针寄存器等，用于存储运算过程中的数据。数据空间的大小可以通过片外存储器进行扩展，满足不同应用的需求。C55x DSP 采用字寻址来读/写数据空间的 8 位、16 位或 32 位数据。

1. 字寻址(23 位)

当 CPU 访问数据空间时，采用字寻址，即为每个 16 位的字分配一个 23 位宽的地址，下面说明了一行 32 位宽的存储器的地址分配，字 0 的地址为 00 0100h，字 1 的地址为 00 0101h。

由于地址总线是 24 位宽，所以，当 CPU 读/写数据空间时，23 位的地址左移一位，最低位补 0。例如，一条指令在 23 位地址 00 0102h 上读一个字，读数据地址总线上传送的值是 00 0204h，如下所示。

字地址：000 0000 0000 0001 0000 0010

读数据地址总线：0000 0000 0000 0010 0000 0100

2. 数据类型

C55x DSP 指令处理的数据类型有 8 位、16 位和 32 位。

数据空间是采用字寻址，但 C55x 有专门的指令可以选择字的高字节或低字节，进行 8 位数据的处理。字节装载指令将从数据空间读取的字节进行 0 扩展或符号扩展，然后装入寄存器中；字节存储指令可将寄存器中的低 8 位数据存储到数据空间指定的地方。

表 2-7　字节装载和字节存储指令

指令	存取的字节	操作
MOV high_byte(Smem), dst	Smem(15～8)	字节装载
MOV low_byte(Smem), dst	Smem(7～0)	
MOV high_byte(Smem)<<#SHIFTW, ACx	Smem(15～8)	
MOV low _byte(Smem)<<#SHIFTW, ACx	Smem(7～0)	
MOV src, high_byte(Smem)	Smem(15～8)	字节存储
MOV src, low_ byte(Smem)	Smem(7～0)	

当 CPU 存取长字时，存取地址是指 32 位数据的高 16 位(MSW)的地址，而低 16 位(LSW)的地址取决于 MSW 的地址。具体说明如下所示。

如果 MSW 的地址是偶地址，则 LSW 的地址加 1。

如果 MSW 的地址是奇地址，则 LSW 的地址减 1。

对于已确定地址的 MSW(LSW)，将其地址的最低有效位取反，可得到 LSW(MSW)的地址。

3. 数据空间的数据结构

下面通过实例来说明数据在数据空间是如何存储的。有 7 种变长的数据存储在 32 位宽的存储器中。

根据表 2-8 和表 2-9 可以看出，为了存取一个长字必须参考它的 MSW，C 的存取地址是 00 0102h，D 的存取地址是 00 0105h；字地址也可以存取字节，如在地址 00 0107h 上，同时存放了数据 F(高字节)和数据 G(低字节)。利用表 2-7 中的专用指令可以进行字节的存取。

表 2-8　数据长度及地址分配

数据	数据类型	地址
A	字节	000100h(低字节)
B	字	000101h
C	长字	000102h
D	长字	000105h
E	字	000106h
F	字节	00 0107h(高字节)
G	字节	00 0107h(低字节)

表 2-9　存储器中的指令

<table>
<tr><th>字地址</th><th>字 0</th><th colspan="2">字 1</th></tr>
<tr><td>000100h～000101h</td><td>A</td><td colspan="2">B</td></tr>
<tr><td>000102h～000103h</td><td>C (31～16)</td><td colspan="2">C (15～0)</td></tr>
<tr><td>000104h～000105h</td><td>D (15～0)</td><td colspan="2">D (31～16)</td></tr>
<tr><td>000106h～000107h</td><td>E</td><td>F</td><td>G</td></tr>
</table>

2.3.4　C55x DSP 的 I/O 空间

C55x DSP 的 I/O 空间用于与外部设备进行数据交互。I/O 空间包括片内 I/O 空间和片外 I/O 空间。片内 I/O 空间包括控制寄存器和状态寄存器等，用于控制和监测芯片的状态。片外 I/O 空间包括 GPIO、UART、SPI、I^2C 等接口，可以与外部设备进行数据交互。I/O 空间的大小可以通过片外存储器进行扩展，满足不同应用的需求。C55x DSP 的 I/O 空间与数据/程序空间是分开的，采用 16 位宽的字寻址，即为每个字分配一个 16 位地址，

其寻址范围为 64K 字，如表 2-10 所示：

表 2-10　C55x DSP 的 I/O 空间

地址	I/O 空间
0000h～FFFFh	64K 字

当 CPU 访问 I/O 空间时，用 DAB 读数据，用 EAB 写数据。由于 DAB 和 EAB 都是 24 位的，所以在 16 位地址前补 0 构成 24 位地址。例如，一个指令在地址 0102h 处读取一个字，DAB 上传送的地址是 00 0102h。

2.3.5　C55x DSP 的堆栈操作

C55x DSP 的堆栈是指在程序执行过程中，用于存储局部变量、函数调用参数、返回地址等信息的一段内存空间。C55x DSP 的堆栈空间可以通过软件进行配置和管理，可以设置堆栈大小、堆栈指针等参数。在 C 语言中，可以使用标准库函数来操作堆栈，如 push、pop 等函数。在使用 C55x DSP 进行开发时，需要注意堆栈的大小和使用情况，以避免堆栈溢出等问题。同时，C55x DSP 还支持硬件堆栈保护机制，可以在发生堆栈溢出时自动进行异常处理，提高系统的可靠性。

C55x DSP 的堆栈配置可以通过软件进行设置，常见的堆栈配置包括以下三种：

（1）单一堆栈配置：在单一堆栈配置中，硬件堆栈和软件堆栈共享同一段内存空间。这种配置简单易用，但是需要特别注意堆栈空间的大小和使用情况，以避免堆栈溢出。

（2）双堆栈配置：在双堆栈配置中，硬件堆栈和软件堆栈分别使用不同的内存空间。这种配置可以提高系统的可靠性和安全性，但是需要特别注意堆栈指针的设置和切换。

（3）动态堆栈配置：在动态堆栈配置中，堆栈大小和位置可以在运行时进行动态调整。这种配置可以提高系统的灵活性和效率，但是需要特别注意堆栈空间的管理和分配。

C55x 提供 3 种可能的堆栈配置，其中一种使用快返回过程，另外两种使用慢返回过程，如表 2-11 所示。选择堆栈配置方式可通过设置 32 位复位向量的第 29、28 位实现，低 24 位为复位中断服务子程序的起始地址。

表 2-11　C55x DSP 堆栈操作

堆栈配置	描述	复位向量值(二进制)
双 16 位的快返回堆栈	数据堆栈和系统堆栈是独立的。当访问数据堆栈时，SP 被修改，SSP 不变，寄存器 REA 和 CFCT 用来实现快速返回	XX00 XXXX：（24 位 ISR 地址）
双 16 位的慢返回堆栈	数据堆栈和系统堆栈是独立的。当访问数据堆栈时，SP 被修改，SSP 不变，不使用寄存器 REA 和 CFCT	XX01 XXXX：（24 位 ISR 地址）
32 位的慢返回堆栈	数据堆栈和系统堆栈作为单一 32 位堆栈。当访问数据堆栈时，SP 和 SSP 同时被修改，寄存器 REA 和 CFCT 不使用。注意：如果通过 SP 的映射位置修改 SP，SSP 不会改变，这时必须独立修改 SSP 使两个指针对齐	XX10 XXXX：（24 位 ISR 地址）

C55x 的快返回和慢返回过程有所不同，其区别在于 CPU 如何保存和恢复两个内部存储器的值。在快返回过程中，返回地址保存在寄存器 RETA 中，循环现场保存在寄存器 CFCT 中。可使用专门的 32 位装入和存储指令同时读/写 RETA 和 CFCT，速度较快。而在慢返回过程中，返回地址和循环现场保存在堆栈中（即存储器中）。当 CPU 从子程序返回时，需要从堆栈中恢复这些数据，速度取决于访问存储器的速度。在程序计数器（PC）中，存储 I 单元中 1～6 个字节代码的 24 位地址。循环现场寄存器存储激活循环记录。当 CPU 执行中断或调用子程序时，会保存当前的循环现场，然后清零该寄存器，为新的子程序创建循环现场。当 CPU 从子程序返回时，会在该寄存器中恢复原有的循环现场。

2.4　TMS320C55xCPU 寄存器

C55x DSP 寄存器总表如表 2-12 所示。

表 2-12　C55x DSP 寄存器总表

缩写	名称	大小/位
AC0～AC3	累加器 0～3	40
AR0～AR7	辅助寄存器 0～7	16
BK03、BK47、BKC	循环缓冲区大小寄存器	16
BRC0、BRC1	块循环计数器 0 和 1	16
BRS1	BRS1 保存寄存器	16
BSA01、BSA23、BSA45、BSA67、BSAC	循环缓冲区起始地址寄存器	16
CDP	系数数据指针（XCDP 的低位部分）	16
CDPH	XCDP 的高位部分	7
CFCT	控制流关系寄存器	8
CSR	计算单循环寄存器	16
DBIER0、DBIER1	调试中断使能寄存器 0 和 1	16
DP	数据页寄存器（XDP 的低位部分）	16
DPH	XDP 的高位部分	7
IER0、IER1	中断使能寄存器 0 和 1	16
IFR0、IFR1	中断标志寄存器 0 和 1	16
IVPD、IVPH	中断向量指针	16
PC	程序计数器	24
PDP	外设数据页寄存器	9
REA0、REA1	块循环结束地址寄存器 0 和 1	24

续表

缩写	名称	大小/位
RETA	返回地址寄存器	24
RPTC	单循环计数器	16
RSA0、RSA1	块循环起始地址寄存器 0 和 1	24
SP	数据堆栈指针	16
SPH	XSP 和 XSSP 的高位	7
SSP	系统堆栈指针	16
ST0_55～ST3_55	状态寄存器 0～3	16
T0～T3	暂时寄存器	16
TRN0～TRN1	变换寄存器 0 和 1	16
XAR0～XAR7	扩展辅助寄存器 0～7	23
XCDP	扩展系数数据指针	23
XDP	扩展数据页寄存器	23
XSP	扩展数据堆栈指针	23
XSSP	扩展系统堆栈指针	23

2.4.1　累加器和 T 寄存器

C55x 的 CPU 包括 4 个 40 位的累加器：AC0～AC3，如表 2-13 所示。4 个累加器是等价的，任何一条使用一个累加器的指令，都可以通过编程来使用。最高 8 位为 8 个保护位。

表 2-13　C55x 的累加器

累加器	39～32	31～16	15～0
AC0	AC0G	AC0H	AC0L
AC1	AC1G	AC1H	AC1L
AC2	AC2G	AC2H	AC2L
AC3	AC3G	AC3H	AC3L

C55x DSP 的 CPU 包含 4 个 16 位通用 T 寄存器，分别为 T0～T3。这些寄存器用于多种运算，如乘法、乘加、乘减等操作中的一个乘数；D 单元中的加法、减法和装入运算中的移位数；以及在 D 单元 ALU 中进行双 16 位运算时，存放 Viterbi 蝶形的变换尺度。此外，当使用交换指令交换辅助寄存器(AR0～AR7)和 T 寄存器的内容时，T 寄存器还可以用于跟踪多个指针值。

2.4.2 用作数据地址空间和 I/O 空间的寄存器

用作数据地址空间和 I/O 空间的寄存器如表 2-14 所示。

表 2-14 用作数据地址空间和 I/O 空间的寄存器

寄存器	功能
XAR0～XAR7 和 AR0～AR7	指向数据空间中的一个数据，用间接寻址模式访问
SCDP 和 CDP	指向数据空间中的一个数据，用间接寻址模式访问
BSA01、BSA23、BSA45、BSA67、BSAC	指定一个循环缓冲区起始地址，加给一个指针
BK03、BK47、BKC	指定循环缓冲区大小
XDP 和 DP	指定用 DP 直接寻址方式访问的起始地址
PDP	确定访问 I/O 空间的外设数据页
XSP 和 SP	指向数据堆栈的一个数据
XSSP 和 SSP	指向系统堆栈的一个数据

1. 辅助寄存器(XAR0～XAR7/AR0～AR7)

辅助寄存器如表 2-15 所示，每个辅助寄存器的高 7 位用于指定要访问数据空间的数据页，低 16 位(如 AR0)的用途是：7 位数据页内的 16 位偏移量。

表 2-15 辅助寄存器的访问属性

寄存器	名称	可访问性
XARn	扩展辅助寄存器 n	只能用专用指令访问，XARn 没有映射到寄存器
ARn	辅助寄存器 n	可用专用指令访问，也可以作为存储器映射寄存器
ARnH	扩展辅助寄存器 n 的高字段	不能单独访问，必须通过访问 XARn 来访问 ARnH

2. 系数数据指针(XCDP/CDP)

CPU 可以连接这个寄存器形成一个扩展系数数据指针(XCDP)，高 7 位(CDPH)用于指定要访问数据空间的数据页，低字(CDP)用来作为 16 位偏移量与 7 位数据页形成一个 23 位地址，如表 2-16 所示。

表 2-16 XCDP 的访问属性

寄存器	名称	可访问性
XCDP	扩展的系数指针	只能用专用指令访问，不是映射到存储器的寄存器
CDP	系数指针	可用专用指令访问，也可作存储器映射寄存器访问
CDPH	XCDP 的高段部分	可用专用指令访问，也可作存储器映射寄存器访问

3. 循环缓冲区首地址寄存器(BAS01，BAS23，BAS45，BAS67，BASC)

可以自由地定义循环的首地址。每个循环缓冲区首地址寄存器与一个或两个特殊的指针相关联，如表 2-17 所示，当 ST2_55 寄存器配置了这些指针时，缓冲区首地址需要加上指针值。

表 2-17　循环缓冲区首地址寄存器的关联指针

寄存器	指针	提供主数据页的寄存器
BSA01	AR0 或 AR1	AR0H
BSA23	AR2 或 AR3	AR2H
BSA45	AR4 或 AR5	AR4H
BSA67	AR6 或 AR7	AR6H
BSAC	CDP	CDPH

4. 循环缓冲区大小寄存器(BK03、BK47、BKC)

3 个 16 位的循环缓冲区大小寄存器指定循环缓冲区大小(最大为 65535)。每个循环缓冲区大小寄存器与 1 个或 4 个特殊的指针相关联，如表 2-18 所示。

表 2-18　循环缓冲区大小寄存器的关联指针

寄存器	指针	提供主数据页的寄存器
BK03	AR0、AR1、AR2 或 AR3	AR0H 为 AR0 或 AR1，AR2H 为 AR2 或 AR3
BK47	AR4、AR5、AR6 或 AR7	AR4H 为 AR4 或 AR5，AR6H 为 AR6 或 AR7
BKC	CDP	CDPH

5. 数据页寄存器(XDP/DP)

在基于 DP 的直接寻址方式中，XDP 指定 23 位地址，其中 DPH 指定要访问数据空间的 7 位数据页，低字(DP)用来代表一个 16 位偏移地址。

6. 外设数据页寄存器(PDP)

对于 PDP 直接寻址方式，9 位的外设数据页寄存器(PDP)，在 64K 字 I/O 空间里，选择一个 128 字的数据页，PDP 在一个 16 位的寄存器里占有 9 位，CPU 忽略 15～9 位。

7. 堆栈指针(XSP/SP、XSSP/SSP)

堆栈指针及堆栈指针的访问属性如表 2-19 和表 2-20 所示。

表 2-19　XSP 和 XSSP

XSP、XSSP	22～16	15～0
XSP	SPH	SP
XSSP	SPH	SSP

表 2-20　堆栈指针的访问属性

寄存器	名称	访问属性
XSP	扩展数据堆栈指针	只能用专用指令访问，不是映射到存储器的寄存器
SP	数据堆栈指针	可用专用指令访问，是存储器映射寄存器
XSSP	扩展系统堆栈指针	只能用专用指令访问，不是映射到存储器的寄存器
SSP	系统堆栈指针	可用专用指令访问，是存储器映射寄存器
SPH	XSP 和 XSSP 的高段部分	可用专用指令访问，是存储器映射寄存器

2.4.3　程序流寄存器

程序流寄存器(PC、RETA、CFCT)及描述如表 2-21 所示。

表 2-21　程序流寄存器

寄存器	描述
PC	24 位的程序计数器。存放 I 单元里解码的 1～6 字节代码的地址。当 CPU 执行中断或调用子程序时，当前的 PC 值(返回地址)存起来，然后把新的地址装入 PC。当 CPU 从中断服务程序或子程序返回时，返回地址重新装入 PC。
RETA	返回地址寄存器。如果所选择的堆栈配置使用快速返回，则在执行子程序时，RETA 就作为返回地址的暂存器。RETA 和 CFCT 一起，高效执行多层嵌套的子程序。可用专门的 32 位装入和存储指令，成功的读写 REIA 和 CFCT。
CFCT	控制流关系寄存器。CPU 保存有激活的循环记录(循环的前后关系)。如果选择的堆栈配置使用快速返回，则在执行子程序时，CFCT 就作为 8 位循环关系的暂存器。RETA 和 CFCT 一起，高效执行多层嵌套的子程序。可用专门的 32 位装入和存储指令，成功的读写 RETA 和 CFCT。

说明：在 C55x DSP 中，CPU 内部按照一定规则存放循环的前后关系，即子程序中循环的状态(激活和未激活)。这些循环关系位会被存储在 CFCT 寄存器中，当 CPU 执行中断或调用子程序时，循环关系位就会被保存到 CFCT 寄存器中；而当 CPU 从中断或调用子程序返回时，循环关系位就会从 CFCT 寄存器中恢复。这样可以确保程序在执行子程序时能够正确地保存和恢复循环状态，从而保证程序的正确性和稳定性。CFCT 各位的含义如表 2-22 所示。

表 2-22　CFCT 各位的含义

位	描述
7	该位表示一个单循环是否激活：0——未激活，1——激活
6	该位表示一个条件单循环是否激活：0——未激活，1——激活
5～4	保留

续表

位	描述		
3～0	这 4 个位表示可能的两层块循环(外层 0 和内层 1)的状态。根据用户选择的块循环指令的类型，一个已被激活的循环，可以使内部的，也可以是外部的。		
	块循环代码	0 层循环	1 层循环
	0	未激活	未激活
	2	激活，外部	未激活
	3	激活，内部	未激活
	7	激活，外部	激活，外部
	8	激活，外部	激活，内部
	9	激活，内部	激活，内部
	其他：保留	—	—

2.4.4　中断管理寄存器

中断管理寄存器及功能如表 2-23 所示。

表 2-23　中断管理寄存器

寄存器	功能
IVPD	指向 DSP 中断向量
IVPH	指向主机中断向量
IFR0、IFR1	可屏蔽中断标志
IER0、IER1	使能或禁止可屏蔽中断
DBIER0、DBIER1	配置选择可屏蔽中断为时间重要中断

1. 中断向量指针(IVPD、IVPH)

2 个 16 位的中断向量指针指向程序空间的中断向量表，中断向量地址如表 2-24 所示。如果 IVPD 和 IVPH 的值相同，所有中断向量可能占有相同的 256B 大小的程序空间。DSP 硬件复位时 IVPD 和 IVPH 都被装入到 FFFFH 处，IVPD 和 IVPH 不受软复位的影响。

在修改 IVPs 之前应确保：

(1) INTM = 1，即所有可屏蔽中断不能响应。

(2) 每个硬件不可屏蔽中断对于原来的 IVPD 和修改后的 IVPD 都有一个中断向量和中断服务程序。

表 2-24　中断向量地址

向量	中断	向量地址		
		位 23～8	位 7～3	位 2～0
IV0	复位	IVPD	00000	000
IV1	不可屏蔽硬件中断	IVPD	00001	000
IV2～IV15	可屏蔽中断	IVPD	00010～01111	000
IV16～IV23	可屏蔽中断	IVPH	10000～10111	000
IV24	总线错误中断(可屏蔽)BERRINT	IVPD	11000	000
IV25	数据记录中断(可屏蔽)DLOGINT	IVPD	11001	000
IV26	实时操作系统中断(可屏蔽)RTOSINT	IVPD	11010	000
IV27～IV31	通用软件中断 INT27～INT31	IVPD	11011～11111	000

2. 中断标志寄存器(IFR0、IFR1)

中断标志寄存器如表 2-25 和表 2-26 所示。16 位的中断标志寄存器 IFR0 和 IFR1 包括所有可屏蔽中断的标志位，当一个可屏蔽中断向 CPU 提出申请时，IFR 中相应的标志位置 1，等待 CPU 应答中断。可以通过读 IFR 标志已发送申请的中断，或写 1 到 IFR 相应的位撤销中断申请，即写入 1 清相应位为 0。中断被响应后将相应位清零，器件复位将所有位清零。

表 2-25　中断标志寄存器 IFR1

位	名称	描述	访问性	复位值	解释
15～11	保留	-	-	-	保留
10	RTOSINTF	实时操作系统中断的标志位	读/写	0	0：RTOSINT 非未决 1：RTOSINT 采决
9	DLOGINTF	数据记录中断的标志位	读/写	0	0：DLOGINT 非未决 1：DLOGINT 未决
8	BERRINTF	总线错误中断的标志位	读/写	0	0：BERRINT 非未决 1：BERRINT 未决
7～0	IF23～IF16	中断标志位	读/写	0	0：与中断向量 x 关联的中断非未决 1：与中断向量 x 关联的中断未决

表 2-26　中断标志寄存器 IFR0

位	名称	描述	访问性	复位值	解释
15～2	IF15～IF2	中断标志位	读/写	0	0：与中断向量 x 关联的中断非未决 1：与中断向量 x 关联的中断未决
1～0	保留	-	-	-	保留

3. 中断使能寄存器(IER0、IER1)

中断使能寄存器(IER0、IER1)如表 2-27 和表 2-28 所示。

表 2-27　中断使能寄存器 IER0

位	名称	描述	访问性	复位值	解释
15～2	IE15～IE2	中断使能位	读/写	不受复位影响	0：禁止与中断向量 x 关联的中断 1：使能与中断向量 x 关联的中断

表 2-28　中断使能寄存器 IER1

位	名称	描述	访问性	复位值	解释
10	RTOSINTE	实时操作系统中断的使能位	读/写	不受复位影响	0：禁止 RTOSINT 1：使能 RTOSINT
9	DLOGINTE	数据记录中断的使能位	读/写	不受复位影响	0：禁止 DLOGINT 1：使能 DLOGINT
8	BERRINTE	总线错误中断的使能位	读/写	不受复位影响	0：禁止 BERRINT 1：使能 BERRINT
7～0	IE23～ IE16	中断使能位 23～16	读/写	不受复位影响	0：禁止与中断向量 x 关联的中断 1：使能与中断向量 x 关联的中断

4. 调试中断使能寄存器(DBIER0、DBIER1)

仅当 CPU 工作在实时仿真模式调试暂停时，这 2 个寄存器才会使用。如果在 CPU 工作在实时方式下，其将被忽略。

2.4.5　循环控制寄存器

1. 单指令循环控制寄存器(RPTC、CSR)

可以重复执行一个单周期指令或并行执行 2 个单周期指令，重复次数 N 被装在 RPTC 中，指令将被重复执行 N+1 次。在一些无条件单指令循环操作中，可以使用 CSR(计算的单循环寄存器)设置重复次数。

2. 块循环寄存器(BRC0、BRC1、BRS1、RSA0、RSA1、REA0、REA1)

块循环指令可以实现 2 级嵌套，一个块循环(1 级)嵌套在另一个块循环(0 级)内部。

表 2-29　块循环寄存器

0 层寄存器(外层)		1 层寄存器(内层)	
寄存器	描述	寄存器	描述
BRC0	16 位块循环计数器 0。存放一块循环代码第一次执行后重复的次数	BRC1	16 位块循环计数器 1。存放一块循环代码第一次执行后重复的次数

续表

0 层寄存器(外层)		1 层寄存器(内层)	
寄存器	描述	寄存器	描述
RSA0	24 位块循环起始地址寄存器 0。存放一块循环代码的第一条指令的地址	RSA1	24 位块循环起始地址寄存器 1。存放一块循环代码的第一条指令的地址。
REA0	24 位块循环结束地址寄存器 0。存放一块循环代码的最后一条指令的地址	REA1	24 位块循环结束地址寄存器 1。存放一块循环代码最后一条指令的地址。
		BRS1	BRC1 保存寄存器。只要 BRC1 装入数值，BRS1 不会在 1 层循环过程中修改。每触发一次 1 层循环，BRC1 都要重新将 BRS1 初始化。该特性使得在 0 层循环之外便可初始化 BRC1，减少了每次循环的时间。

2.4.6 变换寄存器

在 C55x DSP 中，变换寄存器(TRN0，TRN1)用于比较-选择-极值指令。这些指令会比较两个累加器的高位和低位，然后选择其中的最大值或最小值，并将结果更新到 TRN0 和 TRN1 寄存器中。具体来说，当比较累加器的高位时，会更新 TRN0 寄存器；当比较累加器的低位时，会更新 TRN1 寄存器。当比较完两个累加器的全部 40 位后，会执行选择一个 40 位极值的指令，以更新被选中的变换寄存器。

2.4.7 状态寄存器

C55x DSP 包含 4 个 16 位的状态寄存器，分别为 ST0_55、ST1_55、ST2_55 和 ST3_55。这些寄存器包含控制位和标志位，它们的控制位会影响 C55x DSP 的工作，而状态位则反映了 C55x DSP 的当前工作状态或运行结果。其中，ST0_55(以及 ST1_55 和 ST3_55)具有两个访问地址。所有的位都可以通过第一个地址进行访问，但在另一个地址(保护地址)中，加黑部分不能被修改。保护地址是为了支持将 C54x 的代码写入 ST0、ST1 和 PMST 寄存器。

1. ST0_55 状态寄存器

ST0_55 状态寄存器如表 2-30 所示，R/W 表示可进行读写访问，R/W-X 的 X 表示 DSP 复位的值。灰色的位表示向该位写操作无效、在读操作时该位总是为 0。

表 2-30　ST0_55 状态寄存器

ST0_55	15	14	13	12	11	10	9	8～0
	ACOV2	ACOV3	TC1	TC2	CARRY	ACOV0	ACOV1	DP
	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0	R/W-000000000

C55x DSP 的状态寄存器包含多个标志位和控制位，其中包括累加器溢出标志（ACOV0、ACOV1、ACOV2、ACOV3）、进位位（CARRY）、DP 位域和测试/控制位（TC1、TC2）。

累加器溢出标志用于指示累加器 AC0、AC1、AC2 或 AC3 是否发生数据溢出。当发生数据溢出时，相应的 ACOV0、ACOV1、ACOV2 或 ACOV3 会被置 1，直到发生复位、CPU 执行条件跳转、调用、返回或执行一条测试 ACOVx 状态的指令，或被指令清零。溢出检测方式受 M40 位的影响，当 M40 = 0 时，溢出检测在第 31 位，与 C54x 兼容；当 M40 = 1 时，溢出检测在第 39 位。

进位位用于指示 D 单元 ALU 在做加法或减法运算时是否产生进位或借位。当产生进位时，CARRY 会被置位；如果不产生进位，则将 CARRY 清 0。CARRY 位可以被逻辑移位指令修改，对带符号移位指令和循环移位指令，可以选择 CARRY 位是否需要修改。MIN、MAX、ABS、NEG 在目的寄存器是累加器时，修改 CARRY 位，以指示计算结果。进位/借位的检测取决于 M40 位，当 M40 = 0 时，由第 31 位检测进位/借位，当 M40 = 1 时，由第 39 位检测进位/借位。

DP 位域占据 ST0_55 的第 8～0 位，用于提供与 C54x 兼容的数据页指针。C55x 有一个独立的数据页指针 DP，DP(15～7)的任何变化都会反映在 ST0_55 的 DP 位域上。基于 DP 的直接寻址方式，C55x 使用完整的数据页指针 DP(15～0)，因此不需要使用 ST0_55 的 DP 位域。如果想装入 ST0_55，但不想改变 DP 位域的值，可以使用 OR 或 AND 指令。

测试/控制位用于保存一些特殊指令的测试结果，所有能影响一个测试/控制位的指令，都可以选择影响 TC1 还是 TC2。TCx 或关于 TCx 的布尔表达式，都可以在任何条件指令里用作触发器。

2. ST1_55 状态寄存器

ST1_55 状态寄存器如表 2-31 所示，R/W 表示可进行读写访问，R/W-X 的 X 表示 DSP 复位的值。灰色的位表示向该位写操作无效、在读操作时该位总是为 0。

表 2-31　ST1_55 状态寄存器

<table>
<tr><td rowspan="6">ST1_55</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td>BRAF</td><td>CPL</td><td>XF</td><td>HM</td><td>INTM</td><td>M40</td><td>SATD</td><td>SXMD</td></tr>
<tr><td>R/W-0</td><td>R/W-0</td><td>R/W-1</td><td>R/W-0</td><td>R/W-l</td><td>R/W-0</td><td>R/W-0</td><td>R/W-1</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td colspan="5">4～0</td></tr>
<tr><td>C16</td><td>FRCT</td><td>C54CM</td><td colspan="5">ASM</td></tr>
<tr><td>R/W-0</td><td>R/W-0</td><td>R/W-l</td><td colspan="5">R/W-00000</td></tr>
</table>

以下是针对 C55x 处理器的一些配置位的说明：

（1）ASM 位

如果 C54CM = 0，C55x 将忽略 ASM。如果 C54CM = 1，C55x 将以兼容方式运行 C54x 代码，并且 ASM 用于给出某些 C54x 移位指令的移位值，移位范围为 -16 到 15。C55x 移

位指令可以在暂存寄存器(T0～T3)中指定累加器的移位值，也可以在指令中使用常数指定移位值。

（2）BRAF 位

如果 C54CM=0，C55x 将不使用 BRAF。如果 C54CM=1，C55x 将以兼容方式运行 C54x 代码，并且 BRAF 用于指定或控制块循环操作的状态。在由调用、中断或返回引起的代码切换过程中，都需要保存和恢复 BRAF 的值。当执行远程跳转(FB)或远程调用(FCALL)指令时，BRAF 会自动清零。

（3）C16 位

如果 C54CM=0，C55x 将忽略 C16，指令本身将决定使用单 32 位操作还是双 16 位操作。如果 C54CM=1，C55x 将以兼容方式运行 C54x 代码，并且 C16 会影响某些指令的执行。当 C16=0 时，关闭双 16 位模式，D 单元 ALU 执行一条指令是以单 32 位操作(双精度运算)形式。当 C16=1 时，打开双 16 位模式，D 单元 ALU 执行一条指令是以两个并行的 16 位操作(双 16 位运算)形式。

（4）C54CM 位

如果 C54CM=0，C55x CPU 不支持 C54x 代码。如果 C54CM=1，C55x CPU 支持 C54x 编写的代码。在使用 C54x 代码时必须置位该模式，以便使用所有 C55x CPU 的资源。在移植代码时，可以利用 C55x 增加的特性优化代码。

（5）CPL 位

如果 CPL=0，CPL 将选择 DP 直接寻址模式。如果 CPL=1，CPL 将选择 SP 直接寻址模式。

（6）FRCT 位

如果 FRCT=0，C55x 将打开小数模式。乘法运算的结果将左移一位进行小数点调整。如果 FRCT=1，C55x 将关闭小数模式。乘法运算的结果将不移位。

（7）HM 位

当 DSP 收到 HOLD 信号时，会将外部接口总线置于高阻态。根据 HM 的值，DSP 也可以停止内部程序执行。如果 HM=0，C55x 将继续执行内部程序存储器的指令。如果 HM=1，C55x 将停止执行内部程序存储器的指令。

（8）INTM 位

如果 INTM=0，C55x 将使能所有可屏蔽中断。如果 INTM=1，C55x 将禁止所有可屏蔽中断。使用 INTM 位需要注意：INTM 位能够全局使能或禁止可屏蔽中断；CPU 响应中断请求时，自动保存 INTM 位；CPU 把 ST1_55 保存到数据堆栈时，INTM 位也被保存起来；执行中断服务子程序(ISR)之前，CPU 自动置位 INTM 位，禁止所有的可屏蔽中断；ISR 可以通过清零 INTM 位，来重新开放可屏蔽中断；中断返回指令，从数据堆栈恢复 INTM 位的值。

（9）M40 位

当 M40=0 时，D 单元的计算模式选择 32 位模式，其中第 31 位是符号位。计算过程

中的进位取决于第 31 位，由第 31 位判断是否溢出。在饱和过程中，饱和值为 00 7FFF FFFFh(正溢出)或 FF 8000 0000h(负溢出)。累加器和 0 的比较使用第 31～0 位。整个 32 位可以进行移位和循环操作。累加器左移或循环移位时，从第 31 位移出。累加器右移或循环移位时，移入的位插入到第 31 位上。当 M40 = 1 时，D 单元的计算模式选择 40 位带符号移位模式。

(10) SATD 位

当 SATD = 0 时，关闭 D 单元的饱和模式，不执行饱和模式。当SATD = 1时，打开 D 单元的饱和模式。

(11) SXMD 位

当 SXMD = 0 时，关闭 D 单元的符号扩展模式。在 ALU 运算时，高位补 0 扩展至 40 位。在累加器带符号右移时，移位值补零扩展。当 SXMD = 1 时，打开符号扩展模式。在 ALU 运算时，需要符号扩展至 40 位。在累加器带符号移位时，其值带符号扩展。例外情况：限定符 uns( )表明它是无符号的；无符号运算(布尔逻辑运算、循环移位和逻辑移位运算)，补零扩展；MAC 里的运算，带符号扩展至 17 位。

(12) XF 位

XF 是通用的输出位，能用软件处理且可输出至 DSP 引脚。可以使用以下指令清零和置位 XF：BCLRXF(清零 XF)；BSETXF(置位 XF)。

3. ST2_55 状态寄存器

ST2_55 状态寄存器如表 2-32 所示，R/W 表示可进行读写访问，R/W-X 的 X 表示 DSP 复位的值。

表 2-32　ST2_55 状态寄存器

ST2_55	15	14	13	12	11	10	9	8
	ARMS	保留	保留	DBGM	EALLOW	RDM	保留	CDPLC
	R/W-0	-	-	R/W-1	R/W-0	R/W-0	-	R/W-0
	7	6	5	4	3	2	1	0
	AR7PLC	AR6PLC	AR5PLC	AR4PLC	AR3PLC	AR2PLC	AR1PLC	AR0PLC
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

(1) AR0LC～AR7LC 位域

ARnLC(n = 0～7)位决定 ARn 用作线性寻址还是循环寻址。当 ARnLC = 0 时，使用线性寻址。当 ARnLC = 1 时，使用循环寻址。可以使用状态位清零/置位指令来清零/置位 ARnLC：BCLRARnLC(清零 ARnLC)；BSETARnLC(置位 ARnLC)。

(2) ARMS 位

ARMS 位决定 CPU 在进行辅助寄存器(AR)间接寻址时，采用 DSP 模式操作数还是控制模式操作数。当 ARMS = 0 时，使用 DSP 模式操作数，能够有效执行 DSP 专用程序。当

ARMS＝1 时，使用控制模式操作数，能够为控制系统的应用优化代码大小。

（3）CDPLC 位

CDPLC 位决定系数数据指针（CDP）使用线性寻址（CDPLC＝0）还是循环寻址（CDPLC＝1）。可以使用以下指令清零和置位 CDPLC：BCLRCDPLC（清零 CDPLC）；BSETCDPLC（置位 CDPLC）。

（4）DBGM 位

DBGM 位用于调试程序中有严格时间要求的部分。当 DBGM＝0 时，启用该位。当 DBGM＝1 时，禁用该位。仿真器无法访问存储器和寄存器。软件断点仍然可以使 CPU 暂停，但不会影响硬件断点或暂停请求。

（5）EALLOW 位

EALLOW 使能（EALLOW＝0）或禁止（EALLOW＝1）对非 CPU 仿真寄存器的写访问。

（5）RDM 位

RDM 位影响对操作数取整 rnd（）的结果。当 RDM＝0 时，CPU 给 40 位操作数加上 8000h，然后清零第 15～0 位，产生一个 24 位或 16 位的取整结果。当 RDM＝1 时，取整至最接近的整数。取整结果取决于 40 位操作数的第 15～0 位。

4. ST3_55 状态寄存器

ST3_55 状态寄存器如表 2-33 所示，R/W 表示可进行读写访问，R/W-X 的 X 表示 DSP 复位的值，如果 X 是 pin，X 是 pin 上复位的电平。灰色的位表示向该位写操作无效、在读操作时该位总是为 0。

表 2-33　ST2_55 状态寄存器

<table>
<tr><td rowspan="6">ST3_55</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11～8</td><td>7</td></tr>
<tr><td>CAFRZ</td><td>CAEN</td><td>CACLK</td><td>HINT</td><td>保留</td><td>CBERR</td></tr>
<tr><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-l</td><td>write 1100b</td><td>R/W-0</td></tr>
<tr><td>6</td><td>5</td><td>4～3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>MPNMC</td><td>SATA</td><td>保留</td><td>CLKOFF</td><td>SMUL</td><td>SST</td></tr>
<tr><td>R/W-pin</td><td>R/W-0</td><td></td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td></tr>
</table>

（1）CACLR 位

CACLR 用于检查程序 cache 是否已经清零。当 CACLR＝0 时，cache 已经完成清零。清零过程完成时，cache 硬件会清零 CACLR 位。当 CACLR＝1 时，所有的 cache 块都无效。清零 cache 所需的时间周期数取决于存储器的结构。当 cache 清零后，指令缓冲器单元里的预取指令队列的内容会自动清零。

（2）CAEN 位

CAEN 用于使能或禁止程序 cache。当 CAEN＝0 时，cache 控制器不接收任何程序要求，所有的程序要求都由片内存储器或片外存储器（根据解码的地址而定）来处理。当

CAEN=1 时，依据解码的地址，可以从 cache、片内存储器或片外存储器提取程序代码。清零 CAEN 位禁止 cache 时，I 单元的指令缓冲队列的内容会自动清零。

(3) CAFRZ 位

CAFRZ 用于锁定程序 cache。当 CAFRZ=0 时，cache 工作在默认操作模式。当 CAFRZ=1 时，cache 被冻结(其内容被锁定)。没有访问该 cache 时，它的内容不会更改，但被访问时仍然可用。cache 内容一直保持不变，直到 CAFRZ 位清零。

(4) CBERR 位

CBERR 用于检测内部总线错误。当检测到内部总线错误时，CBERR 位会被置位。该错误会使 CPU 在中断标志寄存器 1(IFR1)里置位总线错误中断标志 BERRINTF。对 CBERR 位写 1 无效。该位只在发生内部总线错误时才为 1。总线错误的中断服务子程序，在返回控制中断程序的代码之前，必须清零 CBERR。

(5) CLKOFF 位

CLKOFF 用于禁止 CLKOUT 引脚的输出，且保持高电平。可以使用以下指令清零和置位 CLKOFF：BCLRCLKOFF(清零 CLKOFF)、BSETCLKOFF(置位 CLKOFF)。

(6) HINT 位 HINT 通过主机接口，向主机处理器发送一个中断请求

先清零 HINT 位，然后再置位，产生一个低电平有效的中断脉冲。可以使用以下指令清零和置位 HINT：BCLRHINT(清零 HINT)；BSETHINT(置位 HINT)。

(7) MPNMC 位

MPNMC 用于使能或禁止片上 ROM。当 MPNMC=0 时，微计算机模式下，片上 ROM 被使能，可以在程序空间寻址。当 MPNMC=1 时，微处理器模式下，片上 ROM 被禁止，不映射在程序空间里。MPNMC 的改变，反映复位过程中 MP/MC 引脚的逻辑电平(高电平为 1，低电平为 0)。仅在复位时才对 MP/MC 引脚采样。软件中断指令不影响 MPNMC。

(8) SATA 位

SATA 用于决定 A 单元 ALU 的溢出结果是否饱和。当 SATA=0 时，关闭饱和功能。当 SATA=1 时，如果 A 单元的 ALU 里的计算产生溢出，则结果饱和至 7FFFh(正向饱和)或 8000h(负向饱和)。

(9) SMUL 位

SMUL 用于打开或关闭乘法的饱和模式。当 SMUL=0 时，关闭饱和模式。当 SMUL=1 时，在 FRCT=1 且 SATD=1 的情况下，8000h 与 8000h 相乘的结果饱和至 7FFFFFFFh(不受 M40 位的影响)。这样，两个负数的乘积就是一个正数。对于乘加/减指令，在乘法之后、加法/减法之前，执行饱和运算。

(10) SST 位

SST 用于打开或关闭饱和-存储模式。在 C54x 兼容模式下，SST 将影响一些累加器-存储指令的执行。当 SST=1 时，在存储之前，40 位的累加器值要饱和为一个 32 位的值。如果累加器值要移位，则 CPU 执行移位后饱和。C54CM=0 时，CPU 忽略 SST，仅用指令判断是否产生饱和。

习题

1. TMS320C55x DSP 有哪些特征和优点？

2. TMS320C55x DSP 的内部结构有哪几部分组成？

3. 简述指令缓冲单元、程序流程单元、地址流程单元、数据计算单元的组成和功能。

4. TMS320VC5509A DSP 有哪些片上外设？

5. TMS320C55x DSP 的寻址空间是多少？当 CPU 访问程序空间和数据空间时，使用何种寻址方式、地址是多少位的？

6. TMS320C55x DSP 支持哪两种堆栈？CPU 是如何访问这两种堆栈的？

7. TMS320C55x DSP 有哪几个状态寄存器？累加器溢出标志是如何工作的？

8. 如何检测和处理 TMS320C55x DSP 的累加器溢出？

9. TMS320C55x DSP 的指令集包括哪些类型的指令？

10. TMS320C55x DSP 的中断控制器如何工作？