DIGITAL SIGNAL PROCESSOR

DSP

Specially Designed
for Engineers and Technicians of Electronics

DSP Program Development

DSP程序开发

—— MATLAB调试及直接目标代码生成

李真芳　苏涛　黄小宇　编著

西安电子科技大学出版社
http://www.xduph.com

DSP 程序开发

——MATLAB 调试及直接目标代码生成

李真芳　苏　涛　黄小宇　编著

西安电子科技大学出版社

内 容 简 介

当前，DSP 芯片技术飞速发展，旧型号不断被淘汰，新产品功能越来越强大，而硬件结构和汇编指令也越来越复杂。面对这种形势，DSP 程序开发人员必须转变传统的编程思想，采用开发流程简化的系统级集成环境，以缩短软件开发周期，加快产品的上市时间。

本书针对程序开发人员和 DSP 初学者，介绍了当前最为流行的几种高性能通用 DSP，包括 TI 公司的 TMS320C5000/C6000 DSP 和 AD 公司的 SHARC DSP；详细介绍了当前最新的开发环境及最新的编程思路；介绍了在 MATLAB 及开发工具提供的系统级集成环境下，完成设计、仿真、目标代码产生、调试和运行的过程。

本书面向通信、雷达和电子工程领域的 DSP 算法研究和程序开发人员以及相关专业的研究生和高年级本科生，亦可作为 DSP 爱好者的自学教程。

图书在版编目（CIP）数据

DSP 程序开发：MATLAB 调试及直接目标代码生成 / 李真芳等编著.

—西安：西安电子科技大学出版社，2003.10

ISBN 7-5606-1298-9

Ⅰ.D… Ⅱ.李… Ⅲ.数字信号—信号处理—程序设计 Ⅳ.TN911.72

中国版本图书馆 CIP 数据核字（2003）第 083821 号

策　　划　陈宇光　戚文艳

责任编辑　阎　彬

出版发行　西安电子科技大学出版社（西安市太白南路 2 号）

电　　话　(029)8242885　8201467　　邮　编　710071

http: //www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印　　刷　陕西画报社印刷厂

版　　次　2003 年 10 月第 1 版　2003 年 10 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印张 24.625

字　　数　587 千字

印　　数　1～4000 册

定　　价　35.00 元

ISBN 7-5606-1298-9 / TN・0239

XDUP 1569001-1

前　言

如同当前PC机的更新速度一样，DSP技术也在飞速发展，旧型号不断被淘汰，新型号不断产生，且硬件结构和汇编指令越来越复杂。面对这种形势，人们开发DSP产品的开发思路也逐渐发生转变，很多设计者趋向于购买专业公司提供的现成的DSP板或处理机。这样做虽然费用较高，但换来了可靠性，更重要的是节省了硬件的开发周期。因此，现在整个新产品的开发越来越集中在软件开发上。

传统的开发工具利用DSP汇编语言进行低层次的设计，现在这已不适应IT市场的激烈竞争。为了在竞争中获胜，软件开发人员需要利用简单易学的高层次集成环境，来帮助他们摆脱底层设计的困扰，集中精力探索新的算法，获得技术上的突破。作者编写本书的主要目的之一，就是要把一些最新的编程方法、编程思路和调试方法介绍给读者，以帮助他们快速适应不断变化的技术潮流和市场节奏。

MATLAB作为目前最强大的数值计算和分析工具，已被算法研究人员、工程技术人员广泛应用。MATLAB用于DSP的算法模拟/仿真已大有所为，而人们期望着把DSP的所有开发工具都集成在MATLAB中，即在MATLAB统一环境下就可以完成DSP程序开发的整个过程。顺应此潮流，TI公司和SDL公司与MathWorks公司合作，分别针对TMS320C5000/C6000 DSP和SHARC DSP，推出了MATLAB－DSP系统级集成环境，即在MATLAB统一环境下完成概念设计、模拟/仿真、目标代码产生、运行和调试。利用系统级开发环境可以极大地节省耗费在编程和修正错误上的时间，把开发者解放出来，去紧跟前沿科技，探索新的思路，准时提交第一流的产品设计。

编写本书的另一个目的就是为DSP初学者提供一个完整、系统、详细的介绍，这些介绍包括DSP的内部功能结构、源代码编写和开发工具等。学习完本书后，读者基本上就能够进行DSP程序开发了。

本书的内容安排如下：

第1章介绍当前DSP软件设计的过程和可选择的设计方法，介绍了MATLAB在DSP算法模拟、代码调试中的重要作用，也指出了仅用MATLAB去辅助DSP设计所存在的困难。

第2章介绍当前最为流行的几种高性能通用DSP芯片，包括TI公司的TMS320C5000/C6000 DSP和AD公司的SHARC DSP。主要针对程序开发人

员和初学者详细介绍这些DSP的片内CPU核、寄存器、存储器组织、中断、DMA数据传送、几个常用的片内外设以及这些DSP的源代码编写，包括汇编指令、C/C++环境以及C/C++程序和汇编程序接口时注意的问题等。

第3章和第4章介绍两大主流DSP开发工具，其中：第3章详细介绍TMS320C5000/C6000 DSP的开发工具CCS；第4章详细介绍SHARC DSP（ADSP2106x和ADSP2116x）的开发工具VisualDSP++。

第5章、第6章和第7章介绍DSP软件开发的MATLAB系统级集成环境，其中：第5章介绍MATLAB为TMS320系列DSP代码调试提供的支持；第6章介绍MATLAB为TMS320系列DSP的整个代码开发过程提供的支持，在此环境下可以直接从Simulink模型生成TMS320C6000 DSP的可执行代码；第7章介绍如何直接由Simulink模型生成SHARC DSP的可执行代码。

对于熟悉DSP程序开发的人员，可以根据需要，直接选读本书的第1章、第5章、第6章或第7章以及第3章和第4章中的MATLAB应用例子；对于DSP初学者，则需要先仔细阅读第2章、第3章或第4章，然后再阅读其它各章。

每章的最后都提供了一系列思考题，这些思考题可以帮助读者加深对本章内容的理解并开拓读者的思路。

本书第1章由苏涛编写；第2、3章由李真芳、苏涛合写；其余部分由李真芳、黄小宇合写。全书由李真芳统稿。

由于作者掌握的资料和水平有限，加之时间仓促，书中错误之处在所难免，敬请读者批评指正。

作　者

2003年7月

目　录

第 1 章　MATLAB 与 DSP 软件设计方法综述

1.1　DSP 程序开发过程的演变

1.1.1　DSP 技术综述

数字信号处理技术在最近 20 年获得了广泛的应用。数字信号处理理论和算法是这项技术的一个核心，数字信号处理器(DSP)是这项技术的另一个核心，其中可编程的 DSP 可以将性能很好的信号处理算法方便地应用到实时信号处理中。

MATLAB 是一个强大的分析、计算和可视化工具，特别适用于数字信号处理算法的分析和模拟，使用非常方便。但由于 MATLAB 程序的执行速度相对于实时信号处理来说，仍显得太慢，而 MATLAB 所依赖的平台是计算机等设备，这类设备的体积、功耗不适合于实时信号处理，设备的结构也无法满足实时信号处理所要求的高速数据输入/输出，因此 MATLAB 在数字信号处理技术中，适合于对算法的模拟及对实测数据的事后处理。不过，目前有一种新的技术，可以将 DSP 和 MATLAB 两者密切结合起来，充分利用两者的特长，有力地促进数字信号处理算法的实现。

我们将可编程的 DSP 称为通用 DSP，以区别于 ASIC、CPLD/FPGA 等用硬件进行数字信号处理的器件。本书所讨论的 DSP 都是通用 DSP。就软件可编程而言，DSP 与单片机、PC 机的 CPU 的编程设计方法有类似之处，但 DSP 比单片机的运算速度高得多，又比 CPU 的功耗、设计复杂度低得多。因此，DSP 是进行实时数字信号处理的最佳选择。

为了能低成本、低功耗地进行实时信号处理，各类 DSP 都具备如下特点：

- 采用了数据总线和程序总线分离的哈佛结构及改进的哈佛结构，比传统处理器的冯·诺依曼结构有更高的指令执行速度和数据输入/输出速度。
- 采用流水技术，即每条指令都由片内多个功能单元分别完成取指、译码、取数、执行等多个步骤，从而在不提高时钟频率的条件下减少每条指令的执行时间。
- 具有指令流控制，可以完成无附加开销的循环功能以及延迟跳转指令。
- 具有专门的指令集和较长的指令字，一个指令字同时控制片内多个功能单元的操作。
- 配有独立的算术逻辑单元、乘法器、移位器，可在单周期内完成多次乘、加、移位运算。
- 片内有大容量、多端口存储器，访问速度很快。
- 片内有多条总线可以同时进行取指令和多个数据存取操作。
- 用于寻址的地址寄存器能在其它操作进行的同时，以多种方式自动修改地址寄存器内容，以指向下一个要访问的数据地址。特殊寻址方式有循环寻址、位反序寻址。

• 具有软、硬件等待功能，能与各种类型、不同速度的存储器接口，片内集成的同步 DRAM(SDRAM)控制器支持对高速、大容量存储器的访问。

• 带有 DMA 控制器以及串行通信口等，配合片内多总线结构，使数据块传送速度大大提高。

• 配有中断处理器和定时控制器，可以很方便地构成一个小规模系统或单片系统，易于小型化设计。

• 功耗低，一般为 0.5～4 W，采用低功耗技术的 DSP 只有 0.1 W，待机功耗更低，可用电池供电，对嵌入式系统很适合。因为 DSP 需要的外围设备很少，所以一个紧凑的 DSP 系统的功耗也很低。

采用 DSP 是为了进行高速的实时信号处理，针对各种各样的实际应用，有多种类型、多种档次的 DSP。可以按 DSP 的数据格式，把通用 DSP 划分为定点 DSP 和浮点 DSP。衡量 DSP 性能的主要指标是它完成相应处理任务的速度以及处理精度(数据字长)，最常用的指标是每秒百万次指令执行个数(MIPS，即指令执行时钟)。对大多数定点 DSP 来说，单周期内可以完成一次乘法和一次加法；对浮点 DSP 来说，单周期内可以完成多次乘法和加法。因此，每秒百万次浮点运算(MFLOPS)也是衡量浮点 DSP 运算能力的重要指标。MFLOPS 指标是 MIPS 指标的若干倍。DSP 片内除了运算单元外还有许多其它功能部件，合理设计后，这些功能部件可以同时工作，与此对应的每秒百万次操作(MOPS)也是衡量 DSP 并行操作能力的又一指标，这一指标是 MIPS 指标的若干倍。这些指标都是在最优设计情况下得到的，并不能与 DSP 的实际处理速度等同，因此执行 FFT、FIR 滤波等算法的执行时间就成为一个比较客观的评价标准。在实际设计中，能否达到上述指标，还取决于应用要求、硬件配置、软件编程方法。只有精心设计，才能尽可能地充分利用 DSP 的各种资源，达到比较高的 DSP 运行效率，但这种设计方法往往与设计的通用化、可移植性相矛盾。例如，采用 C 语言设计 DSP 的难度低，设计、调试周期短，设计的 DSP 程序可以在重编译后，运行在各种 DSP 上；而根据特定的 DSP 资源配置，采用特定 DSP 汇编指令编写 DSP 代码的难度大、代码可移植性差，但运行速度比 C 语言程序高得多，常常达到 C 语言程序的 10 倍以上。

1.1.2 DSP 设计过程

利用 DSP 实现一个实时信号处理系统的一般步骤包括：

第一步：根据要求，确定信号处理方案和算法，需要对算法进行原理或功能级的模拟。其中，在满足处理性能的前提下，要对算法的可行性、系统的成本进行评估。

第二步：根据算法，选择合适的实现方法，具体内容就是选择一种合适的 DSP 以及外围器件。这时候的主要工作就是针对这种 DSP 的硬件配置、工作特点，进行算法模拟，考核所选算法在特定 DSP 上能否达到所要求的处理性能和处理速度。

第三步：一方面设计 DSP 硬件电路板；另一方面编写 DSP 程序代码，通常用 C 语言和 DSP 的汇编语言。

第四步：在 DSP 上调试所编写的程序，做到程序和硬件电路都能满足要求。

第五步：把调试成功的 DSP 代码固化到 DSP 目标板上。

然而，并不是所有的 DSP 系统设计都是这一种固定模式。例如，用户可以根据应用需要，购买和采用现成的 DSP 电路板，省去步骤 3 中的硬件设计，用户所作的工作纯粹是编写、调试 DSP 程序。本书的内容不涉及硬件设计，而是集中介绍在 DSP 上实现实时信号处理算法的整个软件设计过程。

实际上，随着 DSP 处理性能的飞速提高，以及用户要求产品的研制周期越来越短，DSP 的设计内容越来越侧重于软件方面。一方面，强大的通用化硬件平台为实现实时信号处理的软件化提供了性能保障，使许多 DSP 设计人员摆脱了硬件设计、配置的困扰，同时也帮助许多纯算法研究人员能轻松地进入 DSP 设计领域。另一方面，DSP 的开发设计环境更加完善，即使要调试 DSP 程序代码，也可以脱离 DSP 硬件电路板。DSP 的调试手段有两种：一种是脱离 DSP 硬件电路板，利用 PC 机的资源模仿 DSP 及其外围电路的工作方式，营造出一个模拟环境，在此环境下调试 DSP 代码，我们称之为软件模拟器(Simulator)的方式；另一种是将 PC 机和 DSP 电路板(称目标板)用专用的 DSP 仿真器及电缆连接起来，从 PC 机上实时监视、控制 DSP 的运行，我们称这种手段为实时仿真器(Emulator)的方式。

模拟器(Simulator)一般用于 DSP 代码的前期调试，只是验证代码的功能和性能。它实用方便，无需添加设备，但模拟器的缺点是速度太慢。例如一段图像压缩代码在真正的 DSP 上运行的时间是 1 秒，而在 2.0 GHz 奔腾 IV 机型上的模拟器下运行时则需要 1 个半小时，其速度相差 5400 倍，因此，用模拟器验证运算复杂度高、运算量大的代码很不合适。此外，用模拟器很难验证 DSP 在实际运行过程中的输入/输出操作。

仿真器(Emulator)对实际的 DSP 硬件目标板进行监控，可以比较真实地得到 DSP 实际运行过程中的状态信息。

1.1.3　DSP 软件设计方法的变革

随着计算机技术、DSP 技术的发展，以 DSP 为核心进行信号处理所用到的软件实现方法经过了多次变革。表 1.1 按 DSP 软件设计方法的变革历程，列出了各种软件设计实现方法及其特点。

表 1.1　不同的软件设计方法

软件设计方法	第一、二步 模拟	第三、四步 编程、调试	开发 周期	难度	代码 效率	代码 长度	可移 植性
① 完全汇编	汇编	汇编	最长	最大	最高	最小	无
② C+汇编	C	汇编	长	最大	最高	最小	无
③ C+C/汇编	C	C/汇编混合	中	中	高	中	部分
④ C+C	C	C	短	小	低	大	完全
⑤ MATLAB+C/汇	MATLAB	C/汇编混合	短	中	高	中	部分
⑥ MATLAB+C	MATLAB	C	最短	小	低	大	完全
⑦ MATLAB 集成	MATLAB	MATLAB	最短	最小	最低	最大	完全

表 1.1 中，代码的效率指运行速度，而代码长度指的是代码所占用的存储器空间。代码的效率和代码长度在大多数情况下是一致的，即代码长度越短，代码的效率/执行速度就越高，但在某些常见情况下，两者是相矛盾的。例如，编写一段循环执行的代码，代码长度

很短，但由于每次循环执行时，DSP 都要判断循环计数器并使用跳转指令(打断了流水线)，执行速度并不高，而如果将这个循环体全部展开，循环体内的代码按循环次数重复写出，代码长度大大加长，但 DSP 不再判断循环计数器，也不使用跳转指令(不打断流水线)，执行速度会显著提高。类似地还有使用子程序还是使用宏的选择问题。对此，将 C 语言等高级语言编写的程序编译成 DSP 的汇编代码时，可以进行编译时的优化选择：使用代码长度的优化，还是代码速度的优化。

表 1.1 中，代码的可移植性指的是，同样的程序能否在不同的 DSP 型号，或同一 DSP 但不同硬件配置的电路板上运行。显然，用 DSP 汇编语言编写的程序可移植性差，而用 C 语言编写的程序可移植性好。

1. 汇编语言编程

20 世纪 80 年代，DSP 刚刚出现并应用于信号处理领域，DSP 的性能指标比较低，运算速度大约在 2 千万次每秒，设计人员为达到足够的运算速度，只能采用表 1.1 中的方法①和②，用 DSP 的汇编语言编写高效、专用的程序代码；只是在算法模拟阶段采用 C 语言。

2. C 语言编程

20 世纪 90 年代前半期，DSP 的运算速度接近 1 亿次每秒，设计人员开始采用 C 语言的编程方法，以求降低开发难度、缩短开发周期，但受限于 DSP 的速度、存储器容量、整个硬件系统的成本，只能部分地采用 C 语言设计程序，关键程序段仍然要结合 DSP 的硬件特点，编写 DSP 汇编程序。

目前，最快的 DSP 运算速度已经超过 10 亿次每秒，外围器件，特别是高速、大容量的 SDRAM 型存储器的性能也越来越高。随着 DSP 速度不断提高和硬件成本不断降低，以及用户要求的产品开发周期的不断缩短，C 语言设计的优越性越来越明显，它不仅降低了开发难度，缩短了开发周期，而且有很好的通用性和可移植性，即使淘汰或更换了 DSP 型号，大部分源程序仍然可以移植到新的 DSP 电路板上；同时，DSP、存储器的性能/价格比的大幅提高也有利于克服 C 语言设计的种种局限性，例如代码效率低、代码占用的存储容量太大等缺陷已不再是设计者考虑的主要问题，设计者考虑的主要问题是如何缩短开发周期，尽快推出产品。

3. MATLAB 辅助设计方法

20 世纪 90 年代后期，MATLAB 作为一种有效的信号处理工具出现后，逐渐渗透到 DSP 的设计当中。MATLAB 是一个强大的分析、计算和可视化工具，使用非常方便。用 C 语言要比用汇编语言编程方便得多，而用 MATLAB 又比用 C 语言编程方便得多。

在设计一个实时 DSP 系统前，常常用 MATLAB 对算法在 DSP 上运行的性能进行模拟，以验证算法本身的正确性。这种模拟分为精确的模拟和功能的模拟，这两种模拟都不简单，前者更复杂。在本章后半部分将对此进行介绍。

在 DSP 系统的程序调试过程中，利用 MATLAB 还可以产生模拟数据,供调试 DSP 时使用，并且将 DSP 的处理结果和 MATLAB 的处理结果进行比较和验证。

截止目前，我们在将一个新的数字信号处理算法应用于实际前，最方便的方法就是先用 MATLAB 进行模拟验证，当模拟结果满意时再把算法修改成 C 或 DSP 汇编语言，在目标 DSP 上实现。目标 DSP 可以是实际的一个 DSP 硬件电路板，也可以是 PC 机上的 DSP

软件模拟器。在 DSP 软件设计的整个过程中，这是最花费时间的部分，编程者要花费大量的时间编写程序，并在目标 DSP 上调试程序。当 MATLAB 模拟结果令人满意后，我们常常把 MATLAB 的结果作为标准值，与用 C 或 DSP 汇编语言编写的 DSP 代码执行结果进行比较。这两者之间通常会有差别，出现差别的主要原因在三个方面：① 代码编写有误；② 实时处理时，DSP 外围硬件接口的问题；③ 算法用 DSP 实现时，数据要量化，存在量化误差。考虑到 MATLAB 与 DSP 上的 C 或汇编在运算精度、动态范围上的不同，即使 DSP 上的 C 或汇编程序编写无误，也有必要在目标 DSP 上重新验证算法的性能。在大多数情况下，MATLAB 的模拟结果和 DSP 的运行结果在性能上的差别是很小的，可以忽略，但有时必须留心这一差别。

编写 DSP 的 C、汇编语言程序与编写 MATLAB 程序当然不是一个概念，前者复杂得多，但借助于 MATLAB，可以降低这一复杂度。我们在设计 DSP 软件时的主要工作，就是保证运行在 DSP 上的 C、汇编程序编写正确、无误，一种简便方法就是通过 DSP 的开发工具把目标 DSP 程序运行的中间结果保存到 PC 机的硬盘上，然后再调入到 MATLAB 工作空间中，与 MATLAB 模拟算法的中间结果进行比较，以发现 DSP 程序编写的错误以及由精度问题导致的结果偏差；或者反过来，用 MATLAB 产生模拟数据文件，以供测试 DSP 程序用。此模拟数据文件可以直接包括到 DSP 的程序代码中，也可以通过 DSP 的开发工具把模拟数据文件调入目标 DSP 中，由 DSP 处理，观察或保存其结果，并与 MATLAB 对同一数据的处理结果进行比较。当信号处理比较复杂时，需要分步验证各个中间结果和最终结果。如果是因为精度问题导致的结果偏差太大，需要用 MATLAB 对算法进行修正，再在 DSP 上用 C、汇编语言编写修正的算法程序。

如此过程反反复复，在 DSP 的开发工具、MATLAB 工作空间之间来回多次切换，仍然显得繁琐、不便。对于熟悉并依赖于 MATLAB 的 DSP 开发人员来说，他们特别期望有一种新的工具能够把 MATLAB 和 DSP 开发工具集成在一起，在 MATLAB 下就能完成 DSP 软件开发的全部过程，而对于熟悉 MATLAB 并开始进行 DSP 设计的初学者，或者利用 MATLAB 专门研究算法并关心其可实现性的算法研究/分析人员来说，更希望有这样的一种工具。

4．MATLAB - DSP 集成设计环境(系统级集成环境)

MATLAB 使用方便的一个原因是它是一种解释型的语言。但解释型语言的一个缺点是执行速度很慢，另一个缺点是必须在 MATLAB 环境下才能运行，而安装 MATLAB 环境需要几百兆字节以上的硬盘空间和相当大的计算机内存。只有将其编译成可执行的应用程序，才能提高执行速度，并独立于 MATLAB 环境运行，这样生成的代码长度和需要的内存空间都小得多。一般来说，MATLAB 程序总是先被翻译成 C/C++，然后被诸如 MSC++等开发工具编译成可执行文件。

既然 DSP 可以用 C 语言设计，而 MATLAB 又可翻译成 C 语言，MATLAB - DSP 设计人员、算法研究人员的这一梦想——把 MATLAB 和 DSP 开发工具集成在一起，就顺理成章地变为现实。

在较新版本的 MATLAB(6.0 以上)环境下提供了这一工具，能将 MATLAB 程序转换成 DSP 代码。其方法是：MATLAB 程序先被转换为 C 程序，再针对特定的 DSP 型号、DSP 目标板，编译(转换)成 DSP 汇编指令，最后生成 DSP 的可执行代码。但研究设计人员可以

不去关心 MATLAB 程序如何转换为 C 程序，C 程序如何转换为特定的 DSP 汇编指令等这些转换步骤是如何具体实现的，因为这是由 MATLAB 自动完成的。特别是对于专门研究算法的人员，他们无需熟悉，甚至无需了解具体的 DSP 硬件结构、功能、指令、DSP 的存储器配置，只要在 MATLAB 环境下，就可检验算法在一种或几种 DSP 上的实际运行效果。这样，就把在 MATLAB 下模拟 DSP 实现某个算法的繁琐过程，以及用 C 和汇编语言编写、调试 DSP 代码的复杂性掩盖起来了。用户只要会使用 MATLAB，即可在 DSP 上测试算法。

当然，通过这种方法得到的 DSP 代码，效率会低得多，这里的效率主要指程序的代码长度、运行速度。因为我们知道，由 MATLAB 得到的 C 程序比直接用 C 语言编写的程序效率低，用 C 程序编译后得到的汇编代码比直接用汇编语言编写的手工汇编代码效率又要低很多，代码也很长。如果不根据 DSP 结构和 DSP 目标板存储器配置对程序代码进行优化的话，生成的 DSP 代码肯定是低效率的。这种代码很有可能只是可运行、可模拟/仿真的，只具有分析意义。除非 DSP 的速度相对于算法所要求的高得多，例如采用目前最快的每秒运算 15 亿次的 DSP，即使只有 5%～10%的低效率，也能满足需要，否则代码必须优化，才具备实用性。

要对代码进行优化，除了前文讲述的软件层次的编译优化方法外，设计人员还必须熟悉 DSP 内部的结构特点，并花费较多的时间优化代码和硬件资源间的配置，同时还要考虑到 MATLAB 程序所生成的 DSP 代码长度可能远远超出了 DSP 目标板的总存储器容量。这些工作都是比较繁琐的，对集中精力研究算法的人员，确实不是一件容易的工作。因此，要想在 MATLAB 环境下高质量地完成 DSP 软件开发的全部过程，仍有许多工作要等待相关的 DSP 厂商、MATLAB 系列的软件开发商去完善。

1.1.4 MATLAB - DSP 集成设计环境下的工具包

目前，MathWorks 公司和 TI 公司联合开发的工具包——MATLAB Link for CCS Development Tools，已经能把MATLAB 和TI 的DSP集成开发环境CCS(Code Composer Stadiu)及目标 DSP 连接起来。MATLAB Link for CCS Development Tools 作为 MATLAB 的一个新工具箱被集成在 MATLAB 6.5(Release13)以及更新的版本中。利用此工具可以像操作 MATLAB 变量一样来操作 TI DSP 的存储器或寄存器，即整个目标 DSP 对于 MATLAB 像透明的一样，开发人员在 MATLAB 环境下，就可以完成对 CCS 的操作，对 DSP 目标程序中的函数的操作，可读写 DSP 中某一段存储器或寄存器，利用 RTDX 进行实时数据交换等，所有这一切操作只利用 MATLAB 命令和对象来实现，简单、方便、快捷。MATLAB Link for CCS Development Tools 可以支持 CCS 能够识别的任何目标板，包括 TI 公司的 DSK、EVM 板和用户自己开发的目标 DSP(C2000™, C5000™, C6000™)板。此工具只用于 DSP 程序的调试过程，如果把此工具与 MathWorks 公司和 TI 公司联合开发的另一工具包——Embedded Target for the TI TMS320C6000™ DSP Platform 配合使用，则可直接由 MATLAB 的 Simulink 模型生成 TIC6000DSP 的可执行代码，即在集成的、统一的 MATLAB 环境下可完成 DSP 开发的整个过程。

针对 ADI 公司的 SHARC 浮点型 DSP，也有类似的 MATLAB 工具包，具备类似的功能。

此外，因为定点 DSP 的定点型运算和普通的 MATLAB 模拟中用到的双精度浮点型运

算有很大差别，用 MATLAB 模拟定点 DSP 的运算难度很大，鉴于这一点，MATLAB 还提供了一个针对定点 DSP 的工具包——Fixed-point Blockset，它大大简化了对定点 DSP 的模拟。

本书主要介绍两类与 MATLAB 相关的 DSP 软件设计方法。一类是 MATLAB 辅助下的 DSP 软件设计方法，如表 1.1 中的方法⑤、⑥，把普通的 MATLAB 工具和 DSP 语言设计结合起来的组合方法，即先借助 MATLAB 工具进行算法模拟，再编写 DSP 程序，在 DSP 程序调试阶段用 MATLAB 辅助调试和验证。本章的后续内容将概要介绍这一方法，在第 3、4、5 章中也分别对这一方法进行了介绍。另一类是 MATLAB 环境下的 MATLAB - DSP 软件集成设计方法，即表 1.1 中的方法⑦——完全的 MATLAB 设计方法。本书将在第 6、7 章再对这一方法进行详细介绍。这两种方法各有优缺点，各有适用范围。

1.2　MATLAB 辅助下的 DSP 软件设计

MATLAB 作为一种有效的信号处理工具，已渗透到 DSP 的设计当中。我们在将一个新的数字信号处理算法应用于实际前，将先用 MATLAB 进行模拟验证，当模拟结果满意时再把算法修改成 C 或 DSP 汇编语言在目标 DSP 上实现，并验证。其具体步骤是：

(1) 用 MATLAB 模拟验证算法；

(2) 根据 MATLAB 程序，编写用于 DSP 的 C 或 DSP 汇编语言程序，生成可执行代码；

(3) 在 DSP 开发系统的模拟/仿真工具中，调试并验证代码的正确性、精度和实时性。

对于一个经过 MATLAB 模拟的算法，用 C/汇编语言编写和调试 DSP 程序时有诸多要注意的方面。一个最基本点就是 MATLAB 的数据格式与 DSP 的数据格式有明显差别，特别是与定点 DSP 的差别很大。

MATLAB 本身面对的是科学计算和分析，其数据格式默认为双精度浮点，其精度比 DSP 高得多，动态范围比 DSP 大得多。为了使经过 MATLAB 模拟的算法能够适用于 DSP，在 MATLAB 模拟这一算法时，必须注意使 MATLAB 尽量真实的模拟 DSP 的实际运算过程，这样就必须对普通的 MATLAB 程序进行改进。以下就浮点 DSP 和定点 DSP 设计分别讨论。

浮点数据的动态范围比定点数据大得多。IEEE754 标准的 32 位浮点数据表示范围为 -1.7014×10^{38}～1.7014×10^{38}，最小数据单位(精度)为 $1.175\,49\times10^{-38}$，动态范围为 1536 dB。在编程时几乎可以不考虑浮点数据的溢出，其处理精度也比定点方式高得多，因此编写浮点 DSP 处理程序(无论是用 DSP 汇编指令还是用高级语言)比编写定点 DSP 处理程序要简单方便得多。例如用浮点 DSP 进行浮点数据的 FFT 变换，程序简洁、精度高，而定点 DSP 的 FFT 程序要复杂得多。此外，浮点 DSP 同时可以进行 32 位或 24 位的定点数据运算，这也比 16 位的定点 DSP 强；在进行函数运算时，浮点 DSP 的效率也高得多，例如以迭代方式进行除法运算或求平方根时，浮点 DSP 的指令数就少得多。应注意的是大多数浮点 DSP 具有 40 位的扩展精度寄存器，其表示的 40 位浮点数比 IEEE 标准的 32 位浮点数增加了 8 位尾数，因而这些运算寄存器的运算精度也较高。

但定点 DSP 的优势是结构简单，因而在速度、成本、功耗上均强于浮点 DSP。不过，本书不详细讨论究竟应采用定点 DSP 还是浮点 DSP 来实现一个算法的问题。

无论是浮点 DSP 还是定点 DSP 都可进行更高精度的数学运算，例如 16 位定点 DSP 可模仿进行 32 位/64 位定点运算和浮点运算，而 32 位浮点 DSP 可进行 64 位双精度浮点运算。但这两种做法的代价是每次运算都要用许多条指令，使程序的执行效率和可读性变差。

在设计 DSP 程序前和 DSP 程序调试过程中，经常要借助于 C、MATLAB 等工具模拟和验证 DSP 的处理效果和处理过程。对浮点 DSP 来说，这些步骤要简单得多。在 PC 机上，若利用 C 语言的 32 位单精度浮点格式来模拟验证，PC 机和 DSP 两者的执行结果只会有细微的差别，因为 DSP 有 40 位的扩展精度寄存器。在 PC 机上，若用 C 或 MATLAB 的双精度浮点格式来模拟验证时，会得到比 DSP 更精确的处理结果，但两者的差别仍很小，相对误差一般不超过 10^{-6}。

用高级语言模拟和验证定点 DSP 的处理过程就很复杂，一般不能直接使用 MATLAB。用 C 语言编程时必须做到数据格式、中间结果、溢出控制、移位等与 DSP 的操作相一致，这就过于费时费力。一种粗略的替代方法是用 MATLAB 或 C 只作原理功能的模拟和验证，这就可以采用浮点处理方式，然后再编写实际的 DSP C 语言程序或汇编代码。若 DSP 程序正确，两种结果应该是接近的。若由 DSP 程序得不到原理模拟结果，则应分析多种原因，除了 DSP 程序编程错误外，其它原因还有精度不够、中间过程发生溢出或饱和等。

1.2.1 MATLAB 模拟浮点 DSP

IEEE754 标准的单精度格式中，数据的 32 位从高到低是这样规定的：1 个符号位，8 个指数位，23 个尾数位；双精度(64 位浮点)格式是这样规定的：1 个符号位，11 个指数位，52 个尾数位。

双精度格式比单精度格式所表示的数值范围和精度都高很多，这对复杂的科学计算是很有意义的。而对实时信号处理来说，在多数情况下，不需要有像科学计算那样高的精度，甚至可以讲，双精度格式和单精度格式进行处理所得到的结果几乎无差别。这在用 MATLAB 模拟的许多 DSP 设计中已经被验证。因此，对于浮点 DSP 来说，多数情况下，DSP 软件设计步骤的第一步和第二步都比较简单，因为 DSP 的 32 位/40 位浮点数据格式的处理精度仅稍差于 MATLAB 的双精度(64 位浮点)格式。例如求解 10 阶非奇异方程，两者的精度仅相差约 10^{-6}，可以忽略。但在少数情况下，MATLAB 的模拟效果并不能直接由 DSP 实现。下面是两个例子。

例如求解一个高阶奇异方程，因为 MATLAB 的精度高、动态范围大，可以得到正确解，而 DSP 则不能得到正确解。对此，有两种解决方法：一种是用 DSP 进行双精度数据格式的运算，但 DSP 的处理速度会大大降低；第二种是修改算法，增加运算冗余量，例如采用对角加噪等技术，虽然能得到正确解，但处理误差会有所增加。

再例如设计带通 IIR 滤波器时，若用 MATLAB 设计出满足要求的一组滤波器系数，是双精度格式，其极点在单位圆内，但很接近单位圆。放到 DSP 程序中后，DSP 将系数截取成单精度的 32 位浮点数，其极点位置移到了单位圆外，造成这个 IIR 滤波器不稳定。输入有限的一段数据(例如冲激函数)后，稳定的滤波器的输出幅度会越来越小，不稳定的滤波器的输出幅度会越来越大。如有 MATLAB 语句：