



高等学校电子与通信类专业“十二五”规划教材

# DSP 技术及应用

江金龙 查代奉 谭小容 武良丹 编



西安电子科技大学出版社  
<http://www.xduph.com>

## 内 容 简 介

本书以 C5000 系列 DSP 中的 TMS320VC5416 芯片为主线,介绍 DSP 的 CPU 结构、片内外设、寻址方式、指令流水线、汇编指令、汇编程序设计、C 语言编程与混合编程和硬件系统设计等内容。

全书共分为 8 章。第 1 章介绍 DSP 系统设计的基础知识;第 2~3 章介绍 DSP 芯片的硬件结构和片内外设,内容从简单到复杂,逐步深入;第 4~6 章介绍寻址方式、指令流水线、汇编指令和汇编程序设计,并采用 C 语言与汇编语言相结合的方式讲授,便于读者深入理解和应用;第 7 章介绍 C 语言编程与混合编程,实现 C 语言与汇编语言的优势互补;第 8 章 DSP 硬件系统设计介绍键盘输入、液晶显示输出、A/D 与 D/A 等模块的设计及编程,帮助读者实现从理论到实践的转变。通过学习,读者可独立设计一个基本的 DSP 系统。

本书可作为工科类高等学校通信工程、电子信息工程、生物医学工程和自动化等专业的本科教材,也可作为相关技术人员和科研人员的参考书。

### 图书在版编目(CIP)数据

**DSP 技术及应用**/江金龙等编. —西安:西安电子科技大学出版社,2012.6

高等学校电子与通信类专业“十二五”规划教材

ISBN 978 - 7 - 5606 - 2771 - 7

I. ① D… II. ① 江… III. ① 数字信号处理—高等学校—教材

IV. ① TN911.72

### 中国版本图书馆 CIP 数据核字(2012)第 046056 号

策 划 毛红兵

责任编辑 刘玉芳 毛红兵

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)88242885 88201467 邮 编 710071

网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com

经 销 新华书店

印刷单位 陕西华沐印刷科技有限责任公司

版 次 2012 年 6 月第 1 版 2012 年 6 月第 1 次印刷

开 本 787 毫米×1092 毫米 1/16 印张 13

字 数 301 千字

印 数 1~3000 册

定 价 22.00 元

ISBN 978 - 7 - 5606 - 2771 - 7/TN · 0648

**XDUP 3063001 - 1**

\*\*\* 如有印装问题可调换 \*\*\*

# 前 言

21 世纪是一个典型的数字时代，数字技术已经深入到百姓生活中，其动力之一就是微电子技术和数字信号处理技术(Digital Signal Processing, DSP)的飞速发展。DSP 技术在数字通信、语音处理、图像处理、声呐、雷达、生物医学工程、电子信息、电力系统、自动化仪表等方面有着广泛的应用，也最具发展的潜力。

DSP 技术是集理论与实践、软件与硬件以及多种软件综合应用为一体的新技术，仅仅依靠课程讲授是难以掌握的，必须在实战中演练并加以运用才能掌握。本书结合作者自身的开发及教学经验，从工程实际出发，对 DSP 的基础理论作了必要的补充，对 DSP 的硬件结构和片内外设作了详细的介绍。本书采用 C 语言和汇编语言相结合的方式讲授软件开发的相关内容，实现两者的优势互补。在书的最后还介绍了 DSP 系统的硬件设计，帮助读者实现从理论到实践的转变。

由于 DSP 技术的综合性强，读者在初学时会感到比较困难，这是正常的。一个比较合理的学习方法是，先熟悉 DSP 的硬件结构，再根据硬件结构理解 DSP 汇编代码。开始时不必死记每条指令，只需大概了解有哪几类指令即可，然后通过后面章节的学习，逐步了解、掌握指令的使用。此时，花一些必要的时间记忆这些指令，乃至抄写、演练这些指令的使用可达到事半功倍的效果。一般主程序和对运算时间没有要求的程序用 C 语言编写，接口部分和对运算时间要求严格的程序用汇编语言编写，这种方式可明显降低程序的编写难度，提高编写效率。

DSP 软件开发离不开开发工具，熟悉 Code Composer Studio (CCS)集成开发环境是进行 DSP 软件开发的基础。在学习 CCS 中的一些选项设置时，开始只需掌握常用的选项设置，然后再逐步深入。熟悉 CCS 开发环境，开发出高效的 DSP 软件代码是需要经过一定时间的学习和实践锻炼的。

本书可作为工科类高等学校通信工程、电子信息工程、生物医学工程和自动化等相关专业的本科教材，也可作为相关技术人员和科研人员的参考书。

本书第 1、2 章和第 4 章由谭小容编写，第 5、6 章由武良丹编写，第 7 章由查代奉编写，第 3 章和第 8 章由江金龙编写。江金龙负责全书的统稿。

由于编者水平有限，疏漏之处在所难免，欢迎读者指正。

编 者

2012 年 1 月

# 目 录

<b>第 1 章 DSP 系统设计的基础知识</b> .....	1
1.1 数制和数制转换 .....	1
1.2 数据位数和符号 .....	2
1.3 补码 .....	2
1.4 补码加减运算 .....	3
1.5 符号扩展与补码乘法 .....	5
1.6 整数除法运算 .....	6
1.7 小数补码和运算 .....	6
习题 .....	7
<b>第 2 章 TMS320C54x 数字信号处理器硬件结构</b> .....	8
2.1 TMS320C54x 的内部结构及主要特性 .....	8
2.1.1 TMS320C54x 的内部结构 .....	8
2.1.2 TMS320C54x 的主要特性 .....	8
2.2 总线结构 .....	11
2.3 存储系统 .....	11
2.3.1 存储器空间 .....	12
2.3.2 片内 ROM 存储器 .....	13
2.3.3 数据存储器 .....	13
2.3.4 I/O 存储器空间 .....	16
2.4 中央处理单元(CPU) .....	16
2.4.1 CPU 状态和控制寄存器 .....	16
2.4.2 算术逻辑单元(ALU) .....	20
2.4.3 累加器 A 和 B .....	21
2.4.4 桶形移位器 .....	22
2.4.5 乘法器/加法器单元 .....	23
2.4.6 比较、选择和存储单元 .....	24
2.4.7 指数编码器 .....	25
2.5 TMS320VC5416 的引脚及说明 .....	25
习题 .....	31
<b>第 3 章 TMS320C54x DSP 片内外设</b> .....	32
3.1 时钟发生器 .....	32
3.1.1 硬件配置锁相环 .....	32
3.1.2 软件配置锁相环 .....	33
3.1.3 主时钟 CLKOUT 输出 .....	35
3.2 定时器 .....	35
3.2.1 定时器工作原理 .....	35

3.2.2	定时器的多时钟应用 .....	37
3.3	中断系统 .....	38
3.3.1	中断分类 .....	38
3.3.2	中断屏蔽寄存器(IMR)和中断标志寄存器(IFR) .....	39
3.3.3	中断请求及中断处理 .....	39
3.3.4	中断操作流程 .....	41
3.3.5	重新映射中断向量地址 .....	43
3.4	通用 I/O 接口 .....	43
3.4.1	XF 和 $\overline{\text{BIO}}$ .....	43
3.4.2	HPI 数据线用作通用 I/O 接口 .....	44
3.4.3	McBSP 用作通用 I/O 接口 .....	44
3.5	McBSP 用作 SPI 接口 .....	45
3.5.1	SPI 协议与 McBSP .....	45
3.5.2	McBSP 配置为 SPI 的方法 .....	46
3.5.3	McBSP 配置为 SPI 主机模式的操作 .....	47
3.5.4	McBSP 配置为 SPI 从机模式的操作 .....	48
3.5.5	McBSP 用作 SPI 接口的初始化 .....	48
3.6	多通道缓冲串口 McBSP .....	48
3.6.1	McBSP 的主要特性 .....	48
3.6.2	McBSP 的工作原理 .....	49
3.6.3	McBSP 的发送与接收 .....	59
3.7	主机接口 .....	61
3.7.1	概述 .....	61
3.7.2	HPI8 接口结构 .....	62
3.7.3	HPI8 引脚功能 .....	62
3.7.4	HPI8 有关寄存器 .....	63
3.7.5	非复用模式下的 HPI16 接口 .....	64
3.8	DMA 控制器 .....	65
3.8.1	DMA 寄存器 .....	65
3.8.2	DMA 各个通道寄存器 .....	68
3.9	外部总线操作 .....	73
3.9.1	外部总线的硬件组成 .....	73
3.9.2	外部总线时钟 .....	73
3.9.3	外部总线优先权及等待 .....	73
3.9.4	外部总线时序 .....	74
3.9.5	软件等待状态寄存器和软件等待控制寄存器 .....	76
3.9.6	可编程分区切换逻辑 .....	78
习题	.....	80
<b>第 4 章</b>	<b>寻址方式与流水线</b> .....	<b>81</b>
4.1	数据寻址方式 .....	81
4.1.1	立即数寻址 .....	81
4.1.2	绝对寻址 .....	82
4.1.3	累加器寻址 .....	83

4.1.4	直接寻址 .....	83
4.1.5	间接寻址 .....	84
4.1.6	存储器映射寄存器寻址 .....	88
4.1.7	堆栈寻址 .....	88
4.2	程序寻址方式 .....	89
4.2.1	程序存储器地址生成器 .....	89
4.2.2	程序计数器(PC)和扩展程序计数器(XPC) .....	90
4.2.3	延时转移下的 PC 操作 .....	91
4.3	指令流水线 .....	92
4.4	汇编程序流程控制 .....	93
4.4.1	条件操作 .....	93
4.4.2	分支转移指令 .....	94
4.4.3	重复操作指令 .....	96
4.4.4	函数调用与返回 .....	97
4.4.5	中断 .....	99
	习题 .....	99
<b>第 5 章</b>	<b>TMS320C54x DSP 汇编指令</b> .....	<b>100</b>
5.1	汇编程序格式 .....	100
5.2	汇编表达式 .....	102
5.3	指令中的符号及缩写 .....	103
5.4	指令系统 .....	105
5.4.1	加载和存储指令 .....	105
5.4.2	程序控制指令 .....	108
5.4.3	算术运算指令 .....	110
5.4.4	逻辑运算指令 .....	116
5.4.5	并行操作指令 .....	118
5.5	可重复和不可重复执行的指令 .....	120
5.5.1	重复执行时变成单周期的多周期指令 .....	120
5.5.2	不可重复执行的指令 .....	120
5.6	汇编伪指令 .....	121
5.6.1	变量定义和常数初始化 .....	121
5.6.2	段定义相关伪指令 .....	123
5.6.3	引用其他文件和条件汇编 .....	123
5.6.4	宏定义和宏引用 .....	123
5.6.5	MEMORY 和 SECTIONS 伪指令 .....	124
	习题 .....	125
<b>第 6 章</b>	<b>汇编程序设计</b> .....	<b>126</b>
6.1	算法设计方法 .....	126
6.1.1	查表法 .....	126
6.1.2	数学变换法 .....	127
6.2	FIR 滤波器的设计 .....	128
6.2.1	直接型 FIR 滤波器实现 .....	129
6.2.2	系数对称 FIR 滤波器实现 .....	132

6.2.3	FIR 滤波系数的 MATLAB 辅助设计 .....	134
6.2.4	DSPLIB 库中的 FIR 滤波函数 .....	135
6.2.5	调用 DSPLIB 库文件的方法 .....	136
6.3	IIR 滤波器设计 .....	137
6.3.1	IIR 滤波器传递函数及实现结构 .....	137
6.3.2	IIR 滤波系数的 MATLAB 辅助设计 .....	138
6.3.3	DSPLIB 库的 IIR 滤波器程序 .....	140
6.4	快速傅立叶变换的 DSP 实现 .....	141
6.4.1	基 2 复数 FFT 算法的原理 .....	141
6.4.2	实序列 FFT 算法(RFFT)原理 .....	142
6.4.3	蝶形运算的实数实现 .....	144
6.4.4	16 点 RFFT 算法流程 .....	144
	习题 .....	145
<b>第 7 章 C 语言编程与混合编程 .....</b>		<b>146</b>
7.1	C 语言编程 .....	146
7.1.1	C54x 支持的基本数据类型 .....	146
7.1.2	常量与变量 .....	147
7.1.3	运算符与表达式 .....	147
7.1.4	函数及调用规则 .....	148
7.1.5	C 语言库函数 .....	148
7.1.6	DSPLIB 汇编库函数 .....	149
7.2	C 程序编译 .....	151
7.2.1	C 编译器生成的段 .....	151
7.2.2	C 编译器的寄存器规则 .....	152
7.2.3	C 程序的系统初始化 .....	153
7.3	C 语言与汇编语言混合编程 .....	153
7.3.1	混合编程方式 .....	153
7.3.2	C 程序访问汇编变量 .....	153
7.3.3	C 程序访问汇编函数 .....	155
	习题 .....	156
<b>第 8 章 DSP 硬件系统设计 .....</b>		<b>157</b>
8.1	DSP 应用领域 .....	157
8.2	DSP 系统设计流程 .....	157
8.3	DSP 硬件系统设计 .....	159
8.3.1	电源模块 .....	159
8.3.2	复位电路 .....	160
8.3.3	存储器 .....	160
8.3.4	A/D 和 D/A 模块 .....	161
8.3.5	键盘和液晶显示器 .....	162
8.3.6	仿真接口 .....	162
8.4	键盘输入接口设计 .....	163
8.4.1	行列式键盘工作原理 .....	163
8.4.2	独立式键盘工作原理 .....	164

8.4.3	标准 PS/2 键盘工作原理 .....	164
8.4.4	行列式键盘与 DSP 接口编程 .....	166
8.5	液晶显示器接口设计 .....	167
8.5.1	显示结构 .....	167
8.5.2	引脚说明 .....	168
8.5.3	串行传输时序图 .....	169
8.5.4	指令说明 .....	169
8.5.5	液晶显示器与 DSP 接口应用 .....	174
8.6	A/D 接口设计 .....	175
8.6.1	TLV1572 工作原理 .....	175
8.6.2	TLV1572 与 DSP 接口应用 .....	177
8.7	D/A 接口设计 .....	178
8.7.1	TLV5617 工作原理 .....	178
8.7.2	TLV5617 与 DSP 接口应用 .....	180
	习题 .....	181
<b>附录 1</b>	<b>线性相位 FIR 低通滤波器设计的 MATLAB 程序 .....</b>	<b>182</b>
<b>附录 2</b>	<b>浮点数转化为定点十六进制数的 MATLAB 程序 .....</b>	<b>184</b>
<b>附录 3</b>	<b>RFFT 汇编程序 .....</b>	<b>187</b>
<b>参考文献</b>	.....	<b>198</b>





# 第 1 章

## DSP 系统设计的基础知识

DSP 可看做是一款高性能的单片机，数值运算是它的主要任务。因此，本章介绍二进制补码及其运算等基础知识。

计算机(也包括单片机、DSP)的基本任务是先从某个“地方”取数(读操作数)，然后按某种规律进行运算，再把结果存放到某个“地方”(写操作数)。这里的“地方”是指存储器、I/O 端口等。因此，地址和数据是计算机中两个重要的基本概念。由于计算机只能存储“0”和“1”两种状态，因此计算机中的地址和数据都是用二进制数表示的。

同计算机原理一样，DSP 的地址和数据也用二进制数表示，但二进制数在阅读和书写上都很不方便，而十六进制数和二进制数有一个十分简单的转换关系。因此，在书写时，一般采用十六进制数。如：十进制数 66 可写成 8 位二进制数或 2 位十六进制数： $66 = 0100\ 0010B = 0x42 = 42H$ 。其中二进制数用后缀 B 表示，十六进制数用前缀 0x 或后缀 H (不分大小写)表示。书写时十六进制数只能以符号 0~9 开始，当十六进制数的第一个符号为 A~F 时，在其符号前加 0，如 255 用 2 位十六进制数表示为 0x0FF 或 0FFH。

### 1.1 数制和数制转换

在日常生活中经常使用的是十进制数，由 0~9 共 10 个符号组成，按照“逢十进一”的规则运算。计算机中使用的数是二进制数，它只有 0 和 1 两个符号，按照“逢二进一”的规则运算。与二进制数密切相关的是十六进制数，由 0~9 和 A~F 共 16 个符号组成，按照“逢十六进一”的规则运算。二进制数、十六进制数与十进制数的转换关系如下：

$$[b_{n-1}b_{n-2}\cdots b_0.b_{-1}\cdots b_{-m}]_B = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \cdots + b_0 \times 2^0 + b_{-1} \times 2^{-1} + \cdots + b_{-m} \times 2^{-m}$$

$$[b_{n-1}b_{n-2}\cdots b_0.b_{-1}\cdots b_{-m}]_H = b_{n-1} \times 16^{n-1} + b_{n-2} \times 16^{n-2} + \cdots + b_0 \times 16^0 + b_{-1} \times 16^{-1} + \cdots + b_{-m} \times 16^{-m}$$

由于  $2^4 = 16$ ，因此 1 位十六进制数可用 4 位二进制数表示，它们之间的关系如表 1.1 所示。

利用 Windows 操作系统“附件”中的计算器可以方便地进行数据转换。方法是在计算器的“查看”菜单中选择“科学型”选项，先选择输入数据类型，再输入数据，然后选择输出数据类型，即可得到转换后的数据。计算器的“编辑”菜单中有“复制”和“粘贴”选项。



表 1.1 十进制与二进制、十六进制的关系

十进制	十六进制	二进制	十进制	十六进制	二进制
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

## 1.2 数据位数和符号

计算机(包括单片机、DSP)除了受“0”和“1”两种状态的限制外,还受到数据长度的限制。计算机的数据总线一般为 8 位、16 位、32 位或 64 位,C54x DSP 的数据总线为 16 位,可表示的数有 16 位二进制数(称为 1 个字)和 32 位二进制数(称为 2 个字或双字)。当用 C 语言编写的程序在 DSP 系统中运行时,在 C 语言中用 1 个字节(8 位二进制数)表示的数在 DSP 系统中均扩展为 1 个字。

### 1. 无符号整数

在 C54x 中,当一个 16 位二进制数(1 个字)定义为无符号数时,它表示的值的范围为  $0 \sim 65\,535$ 。类似地,一个 32 位二进制数(1 个双字)可表示  $0 \sim 2^{32} - 1$  之间的无符号数,它们的数据精度(能表示的最小的数)都是 1。一般情况下,用  $n$  位二进制数表示无符号数的范围是  $0 \sim 2^n - 1$ 。由于数据位数的限制,在进行数据运算时要注意数据表示的范围,超出数据范围会发生数据溢出错误。

### 2. 有符号整数

除了要注意数据表示的范围外,还要关注数据的符号。数据有正、负两种,可用“0”和“1”两种状态表示,“0”表示正数,“1”表示负数。一般把符号位放在最高位,其余位表示数据的绝对值。在计算机中,以这种格式表示的二进制数称为原码。一个  $n$  位原码表示的有符号整数的范围为  $-(2^{n-1} - 1) \sim 2^{n-1} - 1$ ,其中 0 有“+0”和“-0”两种表示方式。

## 1.3 补 码

计算机的加法器和乘法器一般都是按无符号整数设计的,因此用这种硬件结构实现有符号整数运算时,转换关系复杂,运行速度慢,很不方便。为此,计算机用“补码”来表示有符号数。整数  $X$  用  $n$  位二进制补码表示的规则定义如下:

(1) 若  $X \geq 0$ ,它的原码就是它的补码。



(2) 若  $X < 0$ , 它的补码为: 其原码符号位不变, 数据位取反加 1。

同样, 二进制补码也有数据范围问题, 一个  $n$  位二进制补码表示的范围为  $-2^{n-1} \sim 2^{n-1}-1$ , 超出数据范围会发生数据溢出错误, 这里 0 只有“+0”一种表示方式。

无符号整数、有符号整数(原码和补码)与十六进制数的对应关系如表 1.2 所示。

表 1.2 无符号整数、有符号整数(原码和补码)与十六进制数的对应关系

4 位十六进制数	无符号整数	有符号整数(原码)	有符号整数(补码)
0x7FFF	32 767	32 767	32 767
0x7FFE	32 766	32 766	32 766
...	...	...	...
0x0001	1	1	1
0x0000	0	0	0
0xFFFF	65 535	-32 767	-1
0xFFFE	65 534	-32 766	-2
...	...	...	...
0x8001	32 769	-1	-32 767
0x8000	32 768	-0	-32 768

利用 Windows 操作系统“附件”中的计算器也可以求出用补码表示的数。当补码表示正数时, 方法与无符号数的转换方法相同; 当补码表示负数时, 先将其作为无符号数转换为十进制数, 然后减去  $2^n$ , 其中  $n$  为数据位数。例如计算 16 位二进制补码数 0x0FFFF, 先将其作为无符号整数转换成十进制数 65 535, 再减去  $2^{16} = 65 536$ , 得 -1, 这就是它对应的十进制数的值。

## 1.4 补码加减运算

用补码表示有符号数后, 就可以用同无符号数一样的硬件运算结构实现有符号数运算。如果有符号数位数不够, 则在其高位填充符号位, 即正数前补 0, 负数前补 1, 计算结果仍为补码表示的数。只要计算结果不发生溢出, 其答案就是正确的。

例 1.1 设数据位数  $n=8$ , 试用二进制补码求  $X+Y=100+77=?$

解:  $[X]_{\text{补}} = 0110\ 0100\text{B}$ ,  $[Y]_{\text{补}} = 0100\ 1101\text{B}$

$$\begin{array}{r} 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0 \\ +\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1 = 0\text{x}B1 \end{array} \quad (1.1)$$

这个运算的特点是符号位没有进位, 但数据的最高位有进位。显然,  $100+77=177$ , 其结果超出了 8 位二进制补码数表示的范围 ( $-128 \sim 127$ ), 称之为溢出(OV), 这时其运算结果 0xB1 作补码 -79 来解释是错误的。

例 1.2 设数据位数  $n=8$ , 试用二进制补码求  $X+Y=(-100)+(-77)=?$

解:  $[X]_{\text{补}} = 1001\ 1100\text{B}$ ,  $[Y]_{\text{补}} = 1011\ 0011\text{B}$

$$\begin{array}{r}
 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0 \\
 +\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\
 \hline
 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1 = 0x4F (\text{取 8 位})
 \end{array}
 \tag{1.2}$$

这个运算的特点是符号位有进位，但数据的最高位没有进位。显然， $(-100)+(-77)=-177$ ，其结果也超出了 8 位二进制补码数表示的范围 $(-128\sim127)$ ，产生了溢出，这时其运算结果  $0x4F=79$ (补码)也是错误的。

**例 1.3** 设数据位数  $n=8$ ，试用二进制补码求  $X+Y=50+77=?$

解：

$$\begin{array}{r}
 [X]_{\text{补}} = 0011\ 0010\text{B}, [Y]_{\text{补}} = 0100\ 1101\text{B} \\
 \begin{array}{r}
 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \\
 +\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \\
 \hline
 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 = 0x7F
 \end{array}
 \end{array}
 \tag{1.3}$$

这个运算的特点是符号位和数据的最高位都没有进位。自然， $50+77=127$ ，其结果没有超出 8 位二进制补码数表示的范围 $(-128\sim127)$ ，未产生溢出，其运算结果  $0x7F=127$ (补码)是正确的。

**例 1.4** 设数据位数  $n=8$ ，试用二进制补码求  $X+Y=(-50)+(-77)=?$

解：

$$\begin{array}{r}
 [X]_{\text{补}} = 1100\ 1110\text{B}, [Y]_{\text{补}} = 1011\ 0011\text{B} \\
 \begin{array}{r}
 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0 \\
 +\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\
 \hline
 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 = 0x81 (\text{取 8 位})
 \end{array}
 \end{array}
 \tag{1.4}$$

这个运算的特点是符号位和数据的最高位都有进位。自然， $(-50)+(-77)=-127$ ，其结果没有超出 8 位二进制补码数表示的范围 $(-128\sim127)$ ，未产生溢出，其运算结果  $0x81=-127$ (补码)是正确的。

补码的减法方式同上，只要计算结果没有溢出，其答案就是正确的。

C54x DSP 芯片的 CPU(中央处理单元)有一个 16 位的状态寄存器 ST0，用来记录 CPU 的 ALU(算术逻辑单元)的计算结果标志。其中有 3 个比特位用来记录计算结果是否有溢出和进位。当 ALU 中的累加器 A 的计算有溢出时，比特位 OVA 自动置 1，当 ALU 中的累加器 B 的计算有溢出时，比特位 OVB 自动置 1；当加法有进位时，比特位 C 自动置 1，当减法有借位时，比特位 C 自动置 0。ST0 的结构如图 1.1 所示。ST0 其他比特位的含义见第 2 章。

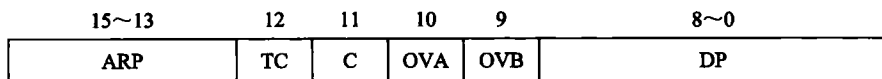


图 1.1 C54x DSP 的状态寄存器 ST0 的结构

ALU 在进行数值运算时，如果数据发生溢出，会产生错误结果。例如 16 位二进制有符号数(简称 16 bit，或 16b，以下同) $0x7FFF\ FFFF=2\ 147\ 483\ 647$ ，当它再加 1 时，按照 ALU 的运算规则，应等于  $0x8000\ 0000$ ，但有符号数  $0x8000\ 0000$  作补码解释为  $-2\ 147\ 483\ 648$ ，则会带来非常大的运算错误。为了减少此类错误，需要对溢出作近似处理。

C54x DSP 芯片的 CPU 还有一个 16 位的状态寄存器 ST1，其结构如图 1.2 所示。其中

有一个溢出模式位 OVM(Overflow Mode), 用来设置 ALU 计算的结果是否作溢出处理。

当 OVM=0 时, 不考虑溢出(默认方式); 当 OVM=1 时, 如果 ALU 发生溢出, 按溢出处理。即正数溢出时, 结果置成正的最大值, 如对 0x7FFF FFFF+1 作溢出处理, 仍等于 0x7FFF FFFF, 负数溢出时, 结果置成负的最小值(0x8000 0000), 这就避免了这种根本性错误。数据发生溢出时是否需要处理, 由用户的要求决定。ST1 的其他比特位的含义见第 2 章。

15	14	13	12	11	10	9	8	7	6	5	4~0
BRAF	CPL	XF	HM	INTM	0	OVM	SXM	C16	FRCT	CMPT	ASM

图 1.2 C54x DSP 的状态寄存器 ST1 的结构

## 1.5 符号扩展与补码乘法

在计算机的计算中, 有时为了提高计算精度, 常常需要用多字来表示一个数, 这就存在数据扩展的问题。在进行补码乘法时, 为了得到正确的结果, 也需要进行数据扩展。对于无符号整数来说, 这种扩展比较容易, 在前面补 0 即可。但对于用补码表示的有符号整数而言, 其扩展方法是在原数据前补符号位, 即在正整数前面全部补 0, 在负整数前面全部补 1(负数的符号位)。采用这种扩展方法, 只要结果不溢出, 其运算结果就是正确的。可见, 对于用补码表示的有符号整数来说, 其数据扩展就是符号扩展。在本书的其他部分, 若无特别声明, 有符号数均用补码表示。

**例 1.5** 设数据位数  $n=8$ , 现采用 16 位二进制补码求  $X+Y=50+(-77)=?$

解:

$$\begin{array}{r}
 [X]_{\text{补}} = 0011\ 0010\text{B}, [Y]_{\text{补}} = 1011\ 0011\text{B} \\
 \begin{array}{r}
 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \\
 +\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \\
 \hline
 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1 = 0\text{xFFE5} = -27
 \end{array}
 \end{array} \tag{1.5}$$

式中  $X=50$  是正数, 在其前面全部补 0,  $Y=-77$  是负数, 在其前面全部补 1, 最终结果是正确的(没有超过 16 位有符号整数表示的范围)。用二进制书写不方便, 用十六进制数书写式(1.5)为

$$X + Y = 0\text{x}0032 + 0\text{x}\text{FFB3} = 0\text{x}\text{FFE5} = -27 \tag{1.6}$$

**例 1.6** 设数据位数  $n=8$ , 现采用 16 位二进制补码求  $X \times Y = 50 \times (-77) = ?$

解:  $[X]_{\text{补}} = 0011\ 0010\text{B}$ ,  $[Y]_{\text{补}} = 1011\ 0011\text{B}$ , 为了书写方便, 将  $Y$  写在式(1.7)的第 1 行。

$$\begin{array}{r}
 \begin{array}{r}
 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \\
 \times\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \\
 \hline
 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\
 \hline
 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \\
 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\
 \hline
 10\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0 = 0\text{x}\text{F0F6} (\text{取 16 位})
 \end{array}
 \end{array} \tag{1.7}$$

式中, 在 X 前面全部补 0, 在 Y 前面全部补 1, 其结果是正确的(没有超过 16 位有符号整数表示的范围)。用十六进制数书写式(1.7)为

$$X \times Y = 0x0032 \times 0xFFB3 = 0xF0F6 = -3850 \quad (1.8)$$

C54x DSP 芯片的 CPU 的状态寄存器 ST1 有一个比特位 SXM 用来设置计算的输入数据是否作符号扩展。当 SXM=1 时(默认设置), 计算的输入数据作符号扩展, DSP 作有符号数运算; 反之, 输入计算的数据不作符号扩展, DSP 作无符号数运算。

用汇编语言设置 SXM 的指令为

RSBX SXM ; SXM=0, CPU 作无符号数运算

SSBX SXM ; SXM=1, CPU 作有符号数运算

C54x DSP 芯片有专用的硬件乘法单元和硬件加法单元, 可在一个时钟周期内, 同时完成一次乘法和一次加法运算。

## 1.6 整数除法运算

C54x DSP 没有除法指令, 若除数是常数, 则转化为乘法计算(乘以除数的倒数); 若除数是 2 的整数次方, 可用算术移位快速实现计算; 若除数是一个变量, 则比较复杂, 需要编程间接实现计算。

**例 1.7** 计算  $Y \div 2 = -0.0625$ , 用 C54x DSP 汇编指令如下:

SSBX OVM ; 溢出处理

SSBX SXM ; 符号扩展

SSBX FRCT ; 小数运算

SFTA A, -1 ; 设 Y 已载入到累加器 A 中, 左移一位

; 算术移位, 符号位不变, 若超过 32 位二进制数的表示范围, 则产生溢出

## 1.7 小数补码和运算

有符号小数的补码可由有符号整数的补码直接扩展而来。设 n 位二进制小数由 m 位整数(包括符号位)和 k 位小数组成( $n = m + k$ ), 这种表示方式称为  $Q_k$  方式。当  $k=0$  时, 则退化为完全整数( $Q_0$ ); 若  $m=1$ , 则为纯小数(若  $n=16$ , 为  $Q_{15}$ )。Q 表示法及其数值范围、数据精度如表 1.3 所示。

**表 1.3 Q 表示法及其数值范围、数据精度**

Q 表示	数据范围	数据精度
$Q_0$	$-32\ 768 \leq X \leq 32\ 767$	$2^0 = 1$
$Q_1$	$-16\ 384 \leq X \leq 16\ 383.5$	$2^{-1} = 0.5$
...	...	...
$Q_{14}$	$-2 \leq X \leq 1.999\ 939\ 0$	$2^{-14} = 0.000\ 061\ 0$
$Q_{15}$	$-1 \leq X \leq 0.999\ 969\ 5$	$2^{-15} = 0.000\ 030\ 5$



一个纯小数  $X$  转换成  $Q_n$  格式的数(也称定点数,即小数点位置固定的数)的公式为式(1.9)。

$$X_n = (\text{int})X \times 2^n \quad (1.9)$$

式中  $n$  为二进制表示的小数点后面的位数,  $\text{int}$  表示取整数部分。公式的含义是小数  $X$  先乘以  $2^n$  变成整数,然后按整数的方式转换成补码形式的二进制数(或十六进制数),但有  $n$  位小数,即  $Q_n$  格式的数。(这里要求  $X$  在  $Q_n$  能表示的数据范围内。)

**例 1.8** 用  $Q_{15}$  格式表示  $X=0.125$  和  $Y=-0.125$ 。

解:  $X_{15} = (\text{int})0.125 \times 2^{15} = 4096 = 0001\ 0000\ 0000\ 0000\text{B} = 0\text{x}1000$

$Y_{15} = (\text{int})-0.125 \times 2^{15} = -4096 = 1111\ 0000\ 0000\ 0000\text{B} = 0\text{x}\text{F}000$

### 习 题

1. 查阅相关资料,试分析单片机、DSP 和 SOPC 等系统的特点,领会它们在实际应用上有什么不同。
2. 查阅相关资料,试列举 DSP 在本专业上的应用实例,画出它们的系统框图。

## 第 2 章

# TMS320C54x 数字信号处理器硬件结构

TMS320C54x(以下简称 C54x)系列 DSP 是 TI 公司的 16 位定点数字信号处理器。TMS320VC5416(以下简称 VC5416 或 5416)就属于该系列产品。TMS320C54x 的体系结构采用改进的哈佛结构,程序与数据分开存放,内部有 8 条、每条 16 位的并行总线。片内集成了片内存储器、片内外设以及专用的硬件逻辑,并备有功能强大的指令系统,使得该芯片具有很高的处理速度和广泛的应用适应性,已广泛地应用于诸如移动通信、数字无线电、计算机网络以及各种专用的实时嵌入系统和仪器仪表中。下面对该系列芯片的结构体系和性能作详细介绍。

## 2.1 TMS320C54x 的内部结构及主要特性

### 2.1.1 TMS320C54x 的内部结构

TMS320C54x 系列 DSP 芯片产品虽然很多,但其体系结构基本上是相同的,特别是核心 CPU 部分,各个型号间的差别主要是片内存储器和片内外设的配置。图 2.1 给出了 TMS320C54x 的典型内部结构框图。

C54x 的硬件结构基本上可分为三大块,如下所述。

(1) CPU 部分包括算术逻辑单元、累加器、乘法器/加法器、桶形移位寄存器、指数编码器、比较选择存储单元及各种专门用途的寄存器、地址生成器、内部总线等。

(2) 存储器部分包括片内程序 ROM、片内单访问数据 RAM(SARAM)、片内双访问数据 RAM(DARAM)及外接存储器接口等。

(3) 片内外设部分包括定时器、各种类型的串口、主机接口、片内的锁相环(PLL)时钟发生器以及各种控制电路等。

此外,芯片中还包含仿真功能及 IEEE 1149.1 标准接口,用于芯片开发应用时进行仿真。

### 2.1.2 TMS320C54x 的主要特性

#### 1. CPU 部分

(1) 先进的多总线结构,具有 1 条程序总线、3 条数据总线和 4 条地址总线。



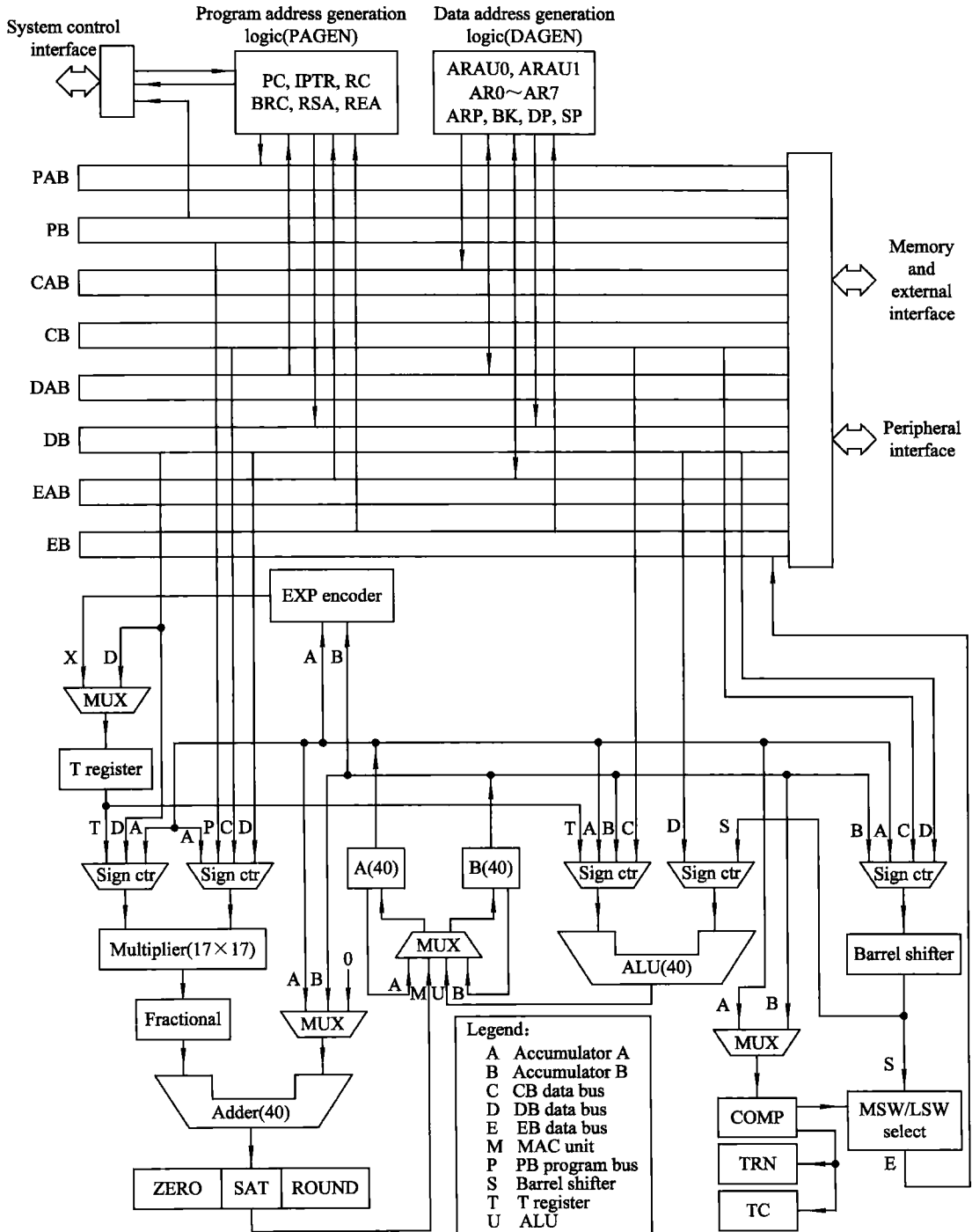


图 2.1 TMS320C54x 的典型内部结构框图