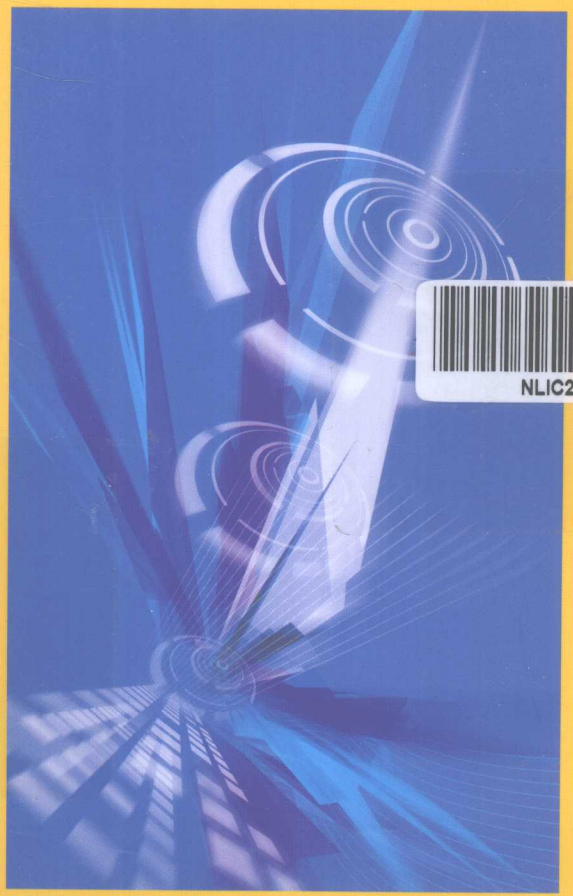
DSP 处理器算法概论

DSP CHULIQI SUANFA GAILUN

许邦建 唐涛 张坤赤 陈强 编著



国防工业出版社
National Defense Industry Press

DSP 处理器算法概论

许邦建　唐　涛　张坤赤　陈　强　编著

国防工业出版社
·北京·

内容简介

本书全面探讨了应用于DSP处理器的算法知识，包括快速算法、精确实数计算算法、算术算法、语音信号处理算法、视频信号处理算法、通信信号处理算法等。本书内容全面、系统性强、概念清晰、叙述深入浅出，特别适合于DSP应用科技工作者和微电子等相关专业老师和学生参考。

图书在版编目(CIP)数据

DSP处理器算法概论／许邦建，唐涛，张坤赤编著.
—北京：国防工业出版社，2012.2
ISBN 978-7-118-07799-5

Ⅰ.①D... Ⅱ.①许... ②唐... ③张... Ⅲ.①数字信号－信号处理－微处理器－概论 Ⅳ.①TN911.72 ②TP332.3

中国版本图书馆CIP数据核字(2012)第014546号

※

国防工业出版社出版发行
（北京市海淀区紫竹院南路23号　邮政编码100048）
北京奥鑫印刷厂印刷
新华书店经售
*
开本 787×1092　1/16　**印张** 18¼　**字数** 415千字
2012年2月第1版第1次印刷　**印数** 1—3500册　**定价** 38.00元

（本书如有印装错误，我社负责调换）

国防书店：(010)88540777　　发行邮购：(010)88540776
发行传真：(010)88540755　　发行业务：(010)88540717

前　言

数字信号处理理论和实现技术是现代电子科学技术的重要基础。DSP 处理器大量用于数字信号处理系统的研究和开发。另外，随着国内微电子设计行业的发展，DSP 处理器本身体系结构的研究和设计也在蓬勃发展。为了更好地使用 DSP，为了更深入地研究 DSP 处理器体系结构，都需要对 DSP 处理器的典型算法有比较全面深入的认识。

国内目前已经出版了很多深受读者喜爱的数字信号处理理论和算法方面的书籍。与国内外已出版的同类书籍比较，本书试图从一些新的角度来阐述 DSP 的问题。这些新视角包括：

（1）理论的简化。数字信号处理的理论，涉及概率论、矩阵、复变函数等数学领域。对于工程技术人员来说，希望看到的不是高深的数学推导，而是简练的算法原理分析和描述。因此，本书对相关的数学理论部分进行了大幅度的精简。

（2）比较全面的算法介绍。目前，国内外出版的数字信号处理书籍的内容范围普遍比较窄，一般只包含了理论基础、DFT 理论和算法、数字滤波器设计这几个方面。但实际上，数字信号处理是一门实践性学科，读者迫切希望了解常见的视频、通信等方面的常见算法。特别是对于计算机、自动控制等非电子专业的读者，他们更希望这类书能比较全面、浅显易懂地描述各种典型应用算法。因此，本书在内容方面除了增加国内外同类书籍都缺乏的精确实数计算算法、算术算法（从事 VLSI 数字信号处理系统设计人员特别需要补充此类算法知识）之外，又着重增加了快速信号处理算法、音频信号处理算法、视频信号处理算法、数字通信典型算法这几类算法知识。

（3）写作特点。本书的写作遵循浅显易懂的大原则。对于各种理论和算法的介绍简单扼要，不纠缠于深奥的理论分析和推导。

本书在学术上具有理论和实际紧密联系、算法内容全面的特点。这些算法知识不仅有利于 DSP 处理器体系结构的深入研究，也有利于 DSP 系统的开发设计。目前，国内广泛地进行着 DSP 处理器设计和开发，这种情况迫切要求我国拥有自己的从事数字信号处理系统开发以及 DSP 处理器集成电路设计的专业队伍。本书的写作目的，就是希望能通过本书的出版，帮助国内在此方面提高理论和技术水平。

本书的写作和出版受到国家自然科学基金的支持（项目批准号：60903045）。

目　录

第 1 章　DSP 处理器中的算法表示及 VLSI 结构 …… 1
1.1　数字信号处理算法的表示及优化 …… 1
1.1.1　数字信号处理算法的图形化表示问题 …… 1
1.1.2　基于数据流图的数字信号处理算法优化 …… 3
1.2　VLSI 流水处理结构 …… 5
1.3　VLSI 并行处理结构 …… 7
1.3.1　脉动阵列结构 …… 7
1.3.2　波前阵列结构 …… 12
1.4　经典数字滤波运算的 VLSI 实现结构 …… 12
1.4.1　FIR 滤波算法的电路实现 …… 13
1.4.2　FIR 滤波器电路实现 …… 13
1.4.3　FIR 系统的一般实现结构 …… 14
1.4.4　IIR 系统的一般实现结构 …… 18
1.4.5　数字滤波运算的格型实现结构 …… 20
参考文献 …… 23
第 2 章　DSP 处理器中的算术算法 …… 24
2.1　经典的数值系统 …… 24
2.1.1　二进制数字系统 …… 24
2.1.2　m 数值的机器表示 …… 24
2.1.3　负数的表示 …… 24
2.2　非传统的固定基数值系统 …… 26
2.2.1　负基数系统 …… 26
2.2.2　符号位数值系统 …… 26
2.2.3　二进制的 SD 数字 …… 27
2.2.4　分布式运算 …… 30
2.3　快速加法算法 …… 30
2.3.1　基本的行波进位加法器 …… 30
2.3.2　基本的分组超前进位加法器 …… 31
2.3.3　一般化的超前进位加法器 …… 32
2.3.4　并行前缀加法器 …… 33
2.3.5　进位选择加法器 …… 35
2.3.6　Ling 加法器 …… 36

2.3.7 进位保留加法器与累加树 …… 37
2.4 乘法和除法的基本顺序算法 …… 40
2.4.1 顺序乘法算法 …… 40
2.4.2 顺序除法算法 …… 42
2.4.3 不恢复除法算法 …… 43
2.4.4 基本的开方算法 …… 44
2.5 高速乘法算法 …… 46
2.5.1 减少部分积数目的加速算法 …… 46
2.5.2 阵列乘法结构 …… 48
2.5.3 树型乘法结构 …… 49
2.6 快速除法 …… 54
2.6.1 SRT 除法 …… 54
2.6.2 高基数除法 …… 55
2.6.3 以乘代除算法 …… 55
2.7 有理算术算法 …… 56
2.7.1 有理算术算法概述 …… 56
2.7.2 MFT 变换 …… 58
2.7.3 SD - MFT …… 58
2.8 初等函数计算算法 …… 58
2.8.1 指数函数 …… 59
2.8.2 对数函数 …… 60
2.8.3 三角函数 …… 60
2.8.4 反三角函数 …… 62
2.8.5 双曲线函数 …… 62
2.9 浮点算术 …… 63
2.9.1 浮点算术原理 …… 63
2.9.2 浮点操作 …… 64
2.9.3 IEEE 浮点标准 …… 65
2.9.4 舍入机制 …… 66
2.10 算术算法中的有限字长问题 …… 67
2.11 精确实数计算算术算法 …… 69
参考文献 …… 70
第 3 章 DSP 处理器中的快速算法 …… 73
3.1 卷积运算的快速算法 …… 73
3.1.1 卷积的基本原理与定义 …… 73
3.1.2 卷积的时域快速算法 …… 73
3.2 DFT 运算 …… 76
3.2.1 DFT 的概念及定义 …… 76
3.2.2 DFT 的性质 …… 78

3.2.3 利用 DFT 实现线性卷积的频域计算 …… 78
3.3 快速傅里叶变换算法 …… 80
3.3.1 递归型 FFT 算法 …… 80
3.3.2 基于快速卷积算法的 FFT 算法 …… 89
3.3.3 实现的 DFT 和 FFT 的相关 MATLAB 语句 …… 91
3.3.4 FFT 算法的电路实现 …… 91
3.4 多速率信号处理算法 …… 91
3.4.1 概述 …… 91
3.4.2 相关基本理论 …… 92
3.4.3 多采样率系统的实现结构 …… 95
参考文献 …… 98
第 4 章 DSP 中的现代数字滤波算法 …… 100
4.1 概述 …… 100
4.2 自适应线性组合器 …… 100
4.3 LMS 自适应算法 …… 101
4.3.1 LMS 算法的性能函数 …… 101
4.3.2 LMS 算法中的梯度与最小均方误差 …… 102
4.3.3 LMS 算法中性能表面的搜索 …… 103
4.3.4 LMS 自适应算法 …… 104
4.4 RLS 自适应算法 …… 104
4.4.1 线性最小二乘滤波器的概念 …… 105
4.4.2 线性最小二乘的数据加窗问题 …… 105
4.4.3 线性最小二乘的正则方程解 …… 106
4.4.4 时间平均相关矩阵的性质 …… 107
4.4.5 线性最小二乘的完全矩阵求解 …… 107
4.4.6 线性最小二乘求解的奇异值分解算法 …… 108
4.4.7 RLS 滤波器的原始概念 …… 108
4.4.8 基本 RLS 算法 …… 109
4.4.9 基于 LDU 分解的改进 RLS 算法 …… 111
4.4.10 基于 QR 分解的改进 RLS 算法 …… 112
4.5 自适应信号处理的应用 …… 119
参考文献 …… 120
第 5 章 音频信号处理算法 …… 122
5.1 音频信号处理概论 …… 122
5.2 音频信号的量化算法 …… 122
5.2.1 均匀与非均匀量化 …… 122
5.2.2 自适应量化 …… 126
5.2.3 差分量化 …… 127
5.2.4 自适应差分量化 …… 127

5.2.5 音频信号的矢量量化算法 …… 128
5.3 基于合成原理的语音编码算法 …… 129
5.3.1 语音信号合成原理 …… 129
5.3.2 线性预测编码 …… 131
5.3.3 CELP 编码 …… 141
5.4 GSM 语音编解码算法 …… 142
5.4.1 GSM 标准简介 …… 142
5.4.2 GSM 中的语音编码算法简介 …… 143
5.4.3 GSM 全速率语音编码算法 …… 147
5.4.4 GSM 全速率语音解码算法 …… 155
5.5 音频信号的数据压缩算法 …… 157
参考文献 …… 158
第 6 章 视频信号处理算法 …… 159
6.1 视信号处理概论 …… 159
6.1.1 视频信号处理标准 …… 159
6.1.2 数字图像中的表示问题 …… 160
6.1.3 视频压缩基本理论 …… 161
6.2 静态图像的编解码原理 …… 162
6.2.1 DCT …… 162
6.2.2 量化 …… 163
6.2.3 熵编码 …… 164
6.3 视频编解码原理 …… 165
6.3.1 预测编码 …… 165
6.3.2 模型编码 …… 166
6.3.3 帧间预测 …… 166
6.4 视频运动估计算法 …… 167
6.4.1 运动估计算法原理 …… 167
6.4.2 运动估计中的搜索算法 …… 168
6.4.3 运动估计中的其他问题 …… 170
6.5 DCT 算法 …… 170
6.5.1 DCT 算法原理 …… 170
6.5.2 一维 DCT …… 171
6.5.3 二维 DCT …… 185
6.5.4 整数 DCT 算法 …… 186
参考文献 …… 204
第 7 章 数字通信中的有限域算法 …… 206
7.1 概述 …… 206
7.2 有限域的基本理论 …… 207
7.3 有限域乘法 …… 210

7.3.1 基本算法原理 …… 210
7.3.2 有限域乘法算法研究现状 …… 212
7.3.3 有限域串行乘法器的基本结构 …… 214
7.3.4 并行有限域乘法 …… 218
7.3.5 并行乘法器比较 …… 221
7.4 并行有限域求逆和除法算法 …… 222
7.4.1 算法研究现状 …… 222
7.4.2 基于费马定理的有限域求逆和除法算法 …… 222
参考文献 …… 227
第8章 数字通信中的BCH译码算法 …… 229
8.1 概述 …… 229
8.2 BCH信道编解码算法 …… 230
8.2.1 BCH编码算法原理 …… 230
8.2.2 BCH码的有限域定义 …… 231
8.2.3 BCH译码算法原理 …… 232
8.3 BCH编码算法的电路实现 …… 235
8.3.1 串行电路实现 …… 235
8.3.2 并行电路实现 …… 236
8.4 BCH译码算法的电路实现 …… 236
8.4.1 有限域乘法器 …… 236
8.4.2 串行译码器的设计 …… 237
8.4.3 并行译码器的设计 …… 240
参考文献 …… 240
第9章 数字通信中的Viterbi译码算法 …… 243
9.1 概述 …… 243
9.2 卷积编码原理 …… 243
9.2.1 卷积编码器结构 …… 243
9.2.2 卷积编码器的多项式描述 …… 244
9.2.3 卷积编码器的状态图描述 …… 245
9.2.4 卷积编码器的网格图描述 …… 245
9.3 卷积码的最大似然译码 …… 246
9.3.1 硬判决下的最大似然译码 …… 246
9.3.2 软判决下的最大似然译码 …… 247
9.4 Viterbi译码算法原理 …… 248
9.4.1 状态量度计算与路径选择 …… 248
9.4.2 路径回溯 …… 250
9.4.3 ACS计算 …… 250
9.4.4 滑窗算法 …… 252
9.5 定点Viterbi算法 …… 253

9.5.1 分支量度的定点计算 …… 253
9.5.2 状态量度的定点计算 …… 254
9.5.3 Yamamoto 定点计算 …… 255
9.5.4 回溯 …… 255
9.5.5 定点递推计算的位宽 …… 257
9.5.6 状态量度原位存储算法 …… 259
参考文献 …… 262
第 10 章 数字通信中的 Turbo 译码算法 …… 264
10.1 概述 …… 264
10.2 Turbo 编码器的 PCCC 结构 …… 265
10.2.1 PCCC 编码器结构 …… 265
10.2.2 PCCC 译码结构 …… 267
10.3 Turbo 编码器的 SCCC 结构 …… 268
10.4 3G 通信中的 Turbo 编码器 …… 269
10.4.1 3G Turbo 编码器结构 …… 269
10.4.2 3G Turbo 编码中的交织器 …… 271
10.5 3G 通信中的 Turbo 译码算法 …… 272
10.5.1 RSC 单元译码的 MAP 算法 …… 272
10.5.2 RSC 单元译码的滑窗算法 …… 277
10.5.3 SISO 迭代译码算法 …… 278
10.5.4 Turbo 译码算法中的定点问题 …… 280
参考文献 …… 280

第 1 章　DSP 处理器中的算法表示及 VLSI 结构

1.1　数字信号处理算法的表示及优化

1.1.1　数字信号处理算法的图形化表示问题

下面以有限冲激响应(FIR)数字滤波器为例进行讨论,具体以 3 阶 FIR 滤波器

$$y(n) = a \cdot x(n) + b \cdot x(n-1) + c \cdot x(n-2)$$

为例来说明 FIR 算法的框图、信号流图(SFG)、数据流图(DFG)和依赖图(DG)4 种图形表示。这些图形表示方法也可用于除 FIR 之外的一般 DSP 算法。

上面的 3 阶 FIR 可以用图 1.1 来描述,这种结构称为 FIR 滤波器的直接结构。图中包含加法器和乘法器两种功能模块。单位延时单元一般表示为 D 或 z^{-1},在实际中用寄存器来实现。

一个系统可以用不同的框图表示,每种表示意味着相同功能的不同实现方法。上面的 3 阶 FIR 也可以用图 1.2 来表示。这种结构称为 FIR 滤波器的数据广播结构。

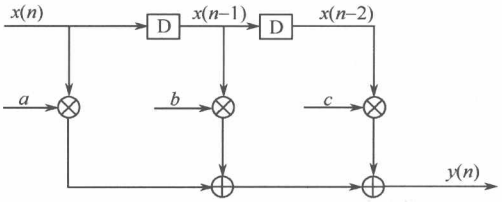


图 1.1　FIR 滤波器的直接结构

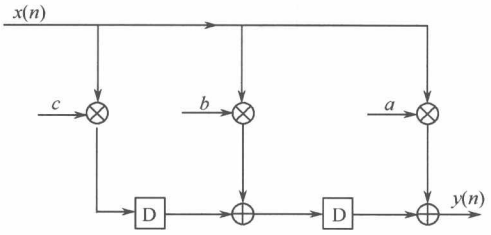


图 1.2　FIR 滤波器的数据广播结构

从以上可以看出,框图是描述系统确切功能的一种形象模型,能清楚地表达出每个信号和功能单元。同一个系统可以有不同的框图,它们具有不同的排列,从而有不同的实现方法。

此 3 阶 FIR 滤波器也可以用如图 1.3 所示的信号流图描述。信号流图是节点和有向边的集合。节点表示运算或任务。一条有向边用(j,k)表示,它对应的是从节点 j 处的信号到节点 k 处信号的线性变换。SFG 已广泛用来分析、表示以及评价线性数字网络,尤其是数字滤波器结构。

线性 SFG 在不改变系统功能的情况下,可以变换为不同形式。流图反转或转置是其中的一种。它可用于单输入、单输出系统,获得等价的转置结构。转置的方法:将所有边反向,然后交换输入和输出节点并保持边的增益和延时不变,则这样产生的 SFG 将保持同样的系统功能。

SFG 提供了线性网络的一种抽象流图表示方法,已经广泛应用于数字滤波器结构设

计中。通常,SFG 只能用于线性网络。

此 3 阶 FIR 也可以用如图 1.4 所示的数据流图描述。

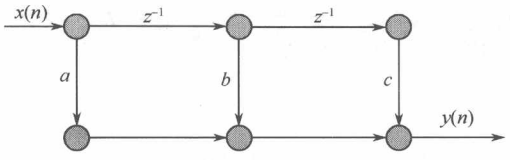


图 1.3　3 阶 FIR 滤波器的信号流图

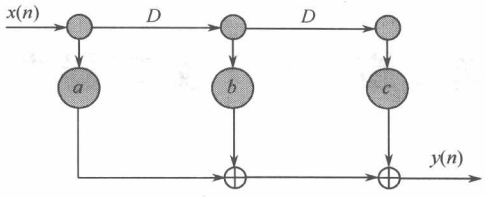


图 1.4　3 阶 FIR 的数据流图

在 DFG 表示方法中,节点表示运算,有向边表示数据通路,每条边有一个非负的延时数与之相关。与节点相关的一般还有它的执行时间。

数据流图捕捉 DSP 算法的数据驱动性质,一旦数据流图中的任意节点的所有输入数据准备好后就可以启动,进行运算。

此 3 阶 FIR 也可以用如图 1.5 所示的依赖图描述。

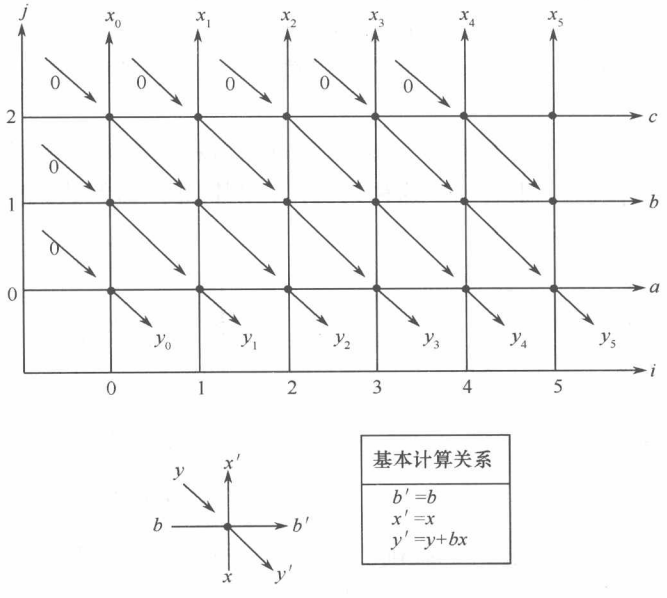


图 1.5　3 阶 FIR 滤波器的依赖图

依赖图是一种有向图,它表明算法中计算之间的依赖关系。DG 中的节点表示运算,边表示节点之间的优先级约束。在 DG 中,只要算法中调用一个新的运算,就会产生一个新的节点,从而基于单一计算的节点不会出现复用的情况。

DG 与 DFG 类似,主要区别:DFG 中节点只涵盖了对应算法在一次迭代中的计算,而这些计算每次迭代都是重复执行的;而 DG 包含算法中所有迭代的运算。DFG 中包含延时单元,而 DG 不包含延时单元。

DG 用于脉动阵列设计,在脉动阵列设计中,通过用不同的方式探究 DG 中的并行性,可以从单个 DG 中推导出各种不同的实现方法。

在依赖图中,最感兴趣的一般是规则依赖图,其具体定义是:在依赖图中,最感兴趣的

一般是“规则依赖图”，其具体定义是如果依赖图中，所有节点都包含同样方向和数量的边，这时就说该依赖图是规则依赖图。

图 1.5 所示的依赖图有三条基本边：$[0 \quad 1]^T$ 是上移的输入方向；$[1 \quad 0]^T$ 是右移的系数方向；$[1 \quad -1]^T$ 是结果移动的方向。在图 1.5 中，所有的节点都包含同样的三条边。因此，它是一个规则依赖图。

1.1.2 基于数据流图的数字信号处理算法优化

1. 数据流图的迭代边界

很多数字信号处理算法都包含反馈环路。对于可实现系统，迭代存在内在的最低边界，称为迭代周期边界（或迭代边界）。迭代边界是用数据流图表示算法时的一种特征值。同样算法可以有不同的 DFG 表示，而这些不同的表示方法就可能具有不同的迭代边界。

DFG 无延时单元的所有路径中，具有最长运算时间的路径称为关键路径。关键路径实际上对应于系统电路实现时具有最长的执行时间的组合逻辑路径。因此也可以说，关键路径的运算时间就是 DFG 中一次迭代所需的最短运算时间，它是电路的时钟周期下界。DFG 可分为非递归 DFG 和递归 DFG 两种，只有递归 DFG 才具有迭代边界。

假设 DFG 中有 L 个环路，则第 l 个环路的环路边界是指 t_l/w_l，其中 t_l 为环路所有节点的运行时间总和，w_l 为环路中延迟单元（同步电路中的寄存器）数目。关键环路是指具有最大环路边界的环路，而关键环路的环路边界就是迭代边界。

计算迭代边界，可采用直观观察法，也可采用一些自动化计算方法。有关自动化计算方法，读者可参考相关文献。

2. 基于重定时技术的数据流图优化

重定时是一种可用于 DFG 优化的技术思路，其主要思想是，在不影响电路输入/输出特性的基础上，通过改变电路中延迟单元的位置来进行优化。重定时技术可以缩短 DFG 系统的时钟周期，以及减少电路中寄存器的数目。实际上，在集成电路中，人们还常用重定时技术来降低电路的功耗和逻辑综合的规模。

重定时概念图如 1.6 所示，图中每个节点都标明运算所需花费的单位时间数。

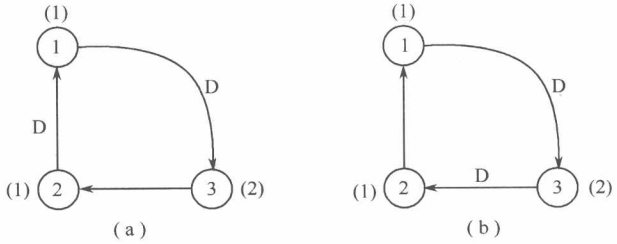


图 1.6 重定时概念图

（a）原始 DFG；（b）重定时后的 DFG。

可以证明：重定时不改变 DFG 中的迭代边界，以及环路中延迟单元的数目。

在重定时技术中，有两种特殊情况，即割集重定时和流水线。

为了说明概念，引入如下两个定义：

（1）割集：割集是一个图的边集合，如果从图中移去这些边，则图就变为不相连接。

割集重定时可将一个 DFG 转换为一个时间局部化的等效形式。

(2) 前馈割集:如果数据在割集的所有边上都沿着前进的方向移动,则这个割集就称为前馈割集。

割集重定时只影响割集中边的权重。若两个不相连的子图分别为 P_1 和 P_2,则割集重定时的具体做法是:从 P_1 到 P_2 的每条割集中的边都增加 k 个延迟单元;从 P_2 到 P_1 的每条割集中的边都减少 k 个延迟单元。

流水线是割集重定时的一种特殊情况。具体来说,流水线是在前馈割集处加入延迟单元。

限于篇幅,本书对重定时技术只作概念上的介绍。实际上,对于重定时,人们还发展了很多理论和算法,包括专门用于缩短 DFG 系统时钟周期的重定时算法和专门用于减少电路中的寄存器数目的重定时算法等。有兴趣的读者可以参考相关文献。

3. 基于展开技术的数据流图优化

展开技术在编译理论中应用最为常见。实际上,它也用于数据流图优化。具体来说,以展开因子 J 展开一个 DSP 程序,就会产生一个以原程序连续迭代 J 次的新程序。

下面以实例说明展开技术的概念。现在考虑如下数字信号处理算法:

$$y(n) = a_0 y(n-3) + x(n)$$

以因子 2 展开,得

$$\begin{cases} y(2k) = a_0 y(2k-3) + x(2k) \\ y(2k+1) = a_0 y(2k-2) + x(2k+1) \end{cases}$$

展开过程如图 1.7 所示。

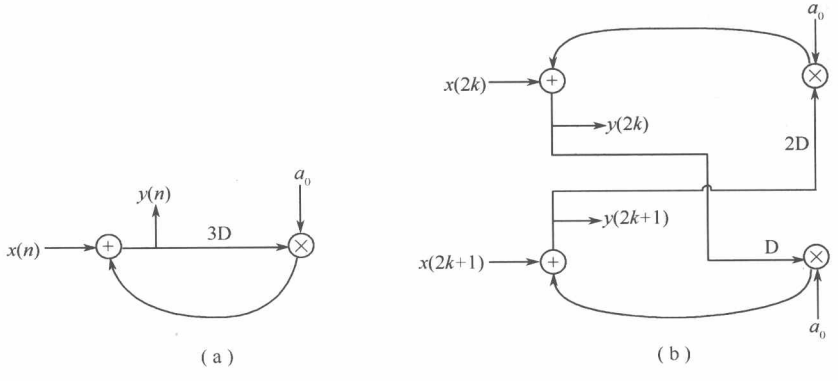


图 1.7 DFG 的展开

(a)原始 DFG;(b)展开后的 DFG。

以上是因子为2 的手工展开过程。实际上,也可以利用一些自动化算法来展开 DFG,读者可参考相关文献。

展开技术的典型应用是通过展开程序发现算法中的并行性,以便在一个更小的迭代周期中完成程序,从而增加该结构的吞吐率。总之,它可以揭示隐藏在 DSP 程序中尚未开发的并行性。

展开技术还可应用于字级或位级设计并行架构。具体来说,字级展开技术,可将字串行结构变为字并行结构;位级展开技术,可将位串行结构变为位并行结构。

可以证明：

（1）展开算法保持 DFG 中延迟单元的数目不变。

（2）展开一个迭代边界为 T_∞ 的 DFG，会得到一个迭代边界为 $J \cdot T_\infty$ 的 J 阶展开 DFG。

实际上，与展开技术对应的还有折叠技术。折叠技术为 DSP 架构提供了一种以时间换取面积的方法。一般而言，折叠技术可以将硬件单元的数目减少为原来的$\frac{1}{M}$，它也付出了代价，即计算时间增加到原来的 M 倍。限于篇幅，本书不对折叠技术作进一步介绍，有兴趣的读者可参考相关文献。

1.2 VLSI 流水处理结构

流水线是多条指令同时执行的一种实现技术。目前，流水线已经成为高速微处理器（包括现代通用 DSP 处理器）中采用的关键技术。另外，随着集成电路技术的发展，在全定制 VLSI 数字信号处理芯片中，流水技术也是提高其数字信号处理性能的关键技术之一。

流水线技术是在传统的冯·诺依曼结构计算机中实现时间重叠的技术，其并行性指完成一条指令的各个子部件在时间上可以重叠工作（属于时间并行），与空间上的阵列并行处理结构完全不同。

流水线的分类较复杂，传统上可以分为：位级流水线、字级流水线，静态流水线、动态流水线，单功能流水线、多功能流水线和线性流水线、非线性流水线。常见的分类方法是线性流水线、非线性流水线。本章仅讨论常用的线性流水线。非线性流水线的有关原理，可参考有关计算机原理的书籍。

线性流水线是将各个子流水段串联起来，无反馈通道。它对流入输入端的数据流进行流水化处理后流到输出端，一般还可将其工作方式分为异步流水线和同步流水线两种。

（1）异步流水线方式。在一个由 K 段构成的流水线中，其相邻的数据流由一个信号握手协议来控制。当第 i 段欲传送数据时，它就要给第 $(i+1)$ 段一个应答信号。在异步流水线中，不同的流水段可以有不同的延迟率，其吞吐率是可以变化的。随着集成电路技术的发展，人们对系统速度和功耗的要求越来越高。在此情况下，异步流水方式在一些中小规模的数字系统中得到了越来越多的应用。但就目前而言，还非常缺乏异步流水方式系统的设计工具和设计理论，因此它目前还不是流水线设计技术的主流。

（2）同步流水线方式。在此系统中，每个流水段由寄存器隔离，并寄存各段的数据。当时钟脉冲到达时，所有寄存器同时把数据传送到下一个流水段中。因此，每个时钟周期可以启动一个相继任务进入流水线，一旦流水线装满后，每个时钟周期可以从流水线送出一个运算结果。由于每一个流水段都有同样的处理速度，所以为各子流水段适当划分输入数据流成为同步流水线的设计关键。同步流水设计方式是目前数字系统设计的主流技术，不仅这种方式设计历史很久，而且可以利用很多现成的设计工具和经过考验的电路。

指令流水线类似于工厂中装配线。在工厂装配线上，吞吐量定义为单位时间内的产

品产量。与之相似，指令流水线的吞吐量取决于指令流出流水线的速度。由于人们希望指令在流水线中每一段的处理过程都在空间上紧密相连，在时间上同时工作，因此所有的流水节拍必须同时工作。指令沿流水线移动一次的时间间隔就是一个机器周期。因为所有指令流水段同时工作，所以机器时间取决于最慢的指令流水段。在计算机中，这个机器周期通常是一个时钟周期。

流水线设计者的任务是平衡各个流水节拍的时间长度，但是通常的流水线节拍之间不会得到这么好的平衡，而且流水线需要一些附加的时间开销。

通过上面分析可得出结论：流水线能够减少指令的平均执行时间。从工程上来讲，可以通过采用流水线来减少时钟周期的长度（或者减少每条指令的平均时钟周期数）。从理论上来讲，更多的流水线段数能更大地提高系统的性能。然而，在实际工程中，增加更多流水段所带来的增益，必须考虑到成本的增加，段间延时的增多，以及遇到转移指令时流水线可能需要清空的这个事实。通常，在一般的微处理器中，流水线的段数不超过10段。

与通用微处理器中情况类似，也可以在VLSI数字信号处理系统中采用流水线技术，同样可以提高这些系统的性能。在并行处理系统中，多个输出可以在一个时钟周期内并行计算，这样，有效采样速度提高到与并行级数相当的倍数。

流水线技术可以缩短VLSI数字信号处理系统的关键路径，这样，VLSI数字信号处理系统可以提高时钟速度或采样速度。

下面以3阶FIR数字滤波器为例说明。考虑这样的FIR滤波器：

$$y(n) = ax(n) + bx(n-1) + cx(n-2)$$

3阶FIR滤波器的实现框图如图1.8所示。

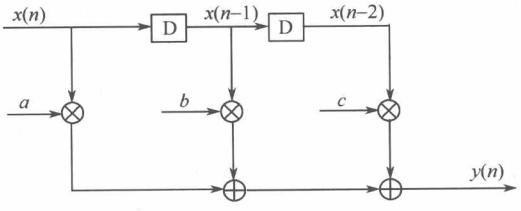


图1.8　3阶FIR滤波器的实现框图

其关键路径由一个乘法和两个加法时间来限定。即如果T_M是乘法所用时间，T_A是加法操作需要的时间，则信号的采样周期为

$$T_{sample} \geqslant T_M + 2T_A$$

因而，采样频率为

$$f_{sample} \leqslant \frac{1}{T_M + 2T_A}$$

上面的3阶FIR滤波器，其流水线实现是通过在图1.9中虚线位置引入两个附加寄存器而得到的。其中，垂直虚线代表一个前馈割集。

现在关键路径由$(T_M + 2T_A)$减小为$(T_M + T_A)$。在这种结构下，当左边的加法器启动当前迭代计算的同时，右边的加法器正在完成前次迭代结果的计算。

必须注意的是，虽然流水线减少了关键路径，但付出了增加迟滞的代价。迟滞实质上

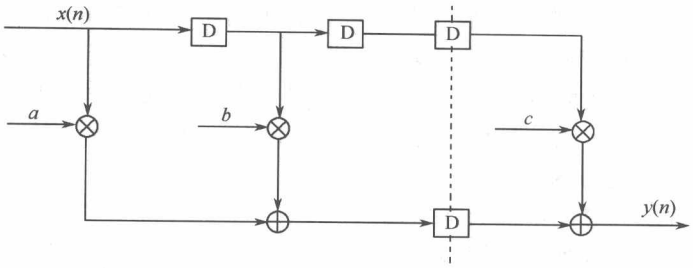


图 1.9　3 阶 FIR 滤波器的流水结构

是流水线系统第一个输出数据的时间与原来时序系统第一个输出数据时间相比的滞后。

可见，流水线的两个主要缺点是：增加了寄存器的数目和系统的迟滞。

1.3　VLSI 并行处理结构

并行处理涉及体系结构、算法映射、程序设计等多项技术，在 VLSI 数字信号处理中一直是研究的热点。

并行计算可分为单指令流多数据流（SIMD）、多指令流多数据流（MIMD）、VLSI 阵列处理三种结构。

SIMD 阵列计算机是用一系列运算器来实现并行计算的。指令由主控计算机发出，所有的运算器执行同样的指令，只不过每个运算器都对应不同的操作数。这种结构尤其适合于多媒体数据的并行处理。SIMD 结构已逐步深入到现代的微处理器和通用数字信号处理器体系结构中，例如 Intel x86 CPU 中的 MMX 指令集及其处理单元。

MIMD 阵列计算机由若干处理器构成，每个处理器都有自己的控制单元、程序和数据。这种结构的主要特点是整个处理任务可以在各个处理器之间进行分配，以增加处理的并行度。显然，MIMD 结构比 SIMD 结构具有更大的系统设计的灵活性。但这种结构也有很大的缺点：在多个处理器同时访问公共资源时，会遇到通信“瓶颈”问题。另外，多个处理器之间还有同步的难题。因此，MIMD 结构的应用远没有 SIMD 结构广泛。

与上述两种结构相比，VLSI 阵列计算结构在 VLSI 的并行信号处理系统中具有更为广泛的应用。这种结构的发展，得益于微电子设计与制造技术的飞速发展。现在可将越来越多的处理单元及其控制和通信单元都放入一个芯片中，因此可以在一个芯片内部充分发掘算法的并行性，并将这些并行性体现为芯片内部的结构。

VLSI 阵列处理器主要可分为脉动阵列、波前阵列两种。一个 VLSI 阵列算法是用多个互连处理单元以有限步骤解决问题的一套法则。并行性对于能否用 VLSI 阵列处理获得高吞吐率是至关重要的。在将算法映射到并行处理器时，将会有几个主要问题：怎样根据算法来设计阵列处理器；怎样用阵列处理器充分利用算法内在的并行性？

本节重点研究 VLSI 阵列计算结构。

1.3.1　脉动阵列结构

在脉动（systolic）阵列结构中，系统有规律地输入、输出数据，类似于人体的心脏活