摘　要

本文主要着眼于目标码率在 9.6-112kbps，典型码率为 44kbps 的基于 C6711 低码率监控视频编码的实现和优化。

首先，通过对目前的低码率视频编码技术的分析，结合 C6711 硬件特点，我们选择了 H.263 视频编码标准作为监控视频编码框架，并且通过对 H.263 标准的七个高级编码选项的测试和分析，选取了其中无限制运动矢量、高级预测和去块滤波等三个编码选项作为监控视频编码的主要方案。分析了低码率监控视频的特点后，我们对编码算法进行了修改，提出了基于钻石搜索的背景块运动搜索算法和全零块预判算法，提高了编码程序的效率。

随后，针对 C6711 的硬件特点，我们对程序的结构、具体代码以及总线读写进行了多方面的优化。在 DSP 嵌入式环境中，提高了视频编码的速度，使优化后的编码程序不但能满足目前的监控系统要求，也为以后的系统升级打下了良好的基础。除了对监控系统的核心 C6711DSP 的程序移植、优化工作以外，我们还分析了系统中的采集模块和串口输出模块对 DSP 的中断通讯，通过对 DSP 中断向量表和中断响应程序的编写，完成了 DSP 系统的集成。分析 C6711 的 ROM 引导过程，编写了 FLASH BOOT 程序，最终使整个系统在上电后能够独立启动，正常、稳定运行。

最后，为了在客户端程序中更好的提升图像分辨率，对超分辨率算法做了相应的研究。我们借鉴了基于 FFT 超分辨率算法中的频域处理方法，采用了更适应人眼视觉特性的小波变换作为空域到频域的方法，结合双线性插值算法对高频区进行插值，最后再利用逆小波变换得到重建图像。这个算法具有良好的时间效率，并且能够较好的恢复图像的边缘信息。

关键词：　C6711 DSP, 监控视频编码，H.263 标准, 超分辨率，小波变换

Abstract

This thesis proposed the realization and optimization of low bit-rate monitor video encoding application in C6711 DSP. The output bit-rate is between 9.6kpbs and 112kbps, typically 44kbps.

Firstly, based on the C6711 DSP hardware characteristics, H.263 was chosen as the core video compression framework. After seven advanced options of H.263 was tested, we analyzed the test results and used three options which are unlimited motion vector, advanced motion prediction and deblock filter as the scheme of the low bit-rate monitor video encoding. Consoidering the low bit-rate video encoding characteristics, background block search algorithm, which is based on Diamond search algorithm was used to improve the motion estimation speed. And detection of All-Zero block algorithm was chosen for the reason of static background sence in minotor video encoding application.

Secondly, We optimized the encoding application in C6711 DSP embedded platform. The application optimization in DSP embedded system was mainly based on the three sides: application architecture, function assembly codes and bus operations. After the optimization, the application not only meet with the requirements of the monitor system, but also make a good preparation for the system update in the future. Besides C6711 DSP, the monitor system included video capture module, UART output module and FLASH module. We set interrupt table vectors and finished the corresponding interrupt funcionts to integrate video capture module and UART output module. Flash boot program was used to boot DSP system from Flash ROM.

Lastly, we realized the client applicaiont and the super-resolution algorithm. We proposed a method combining wavelet transform and bilinear interpolation in the super-resolution algorithm. This method did the resolution improvement in the wavelet frequency domain and obtained good time efficiency and visual effect.

Key words: C6711 DSP, minotor vedio encoding, H.263, super-resolution, wavelet transform

第一章　概　述

1.1　视频监控技术简介

随着科学技术的发展和信息时代的到来，具有智能化、网络化、数字化特征的各种高科技新技术不断涌现。在社会生产和人们日常生活中的方方面面，这些技术得到了日益广泛的应用，降低了社会生产成本，提高了生产效率，节省了大量的人力物力，也给人们的生活带来了更多的便捷、舒适和享受。

视频监控技术正是其中之一。它综合利用了现代视频图像处理、光电传感、计算机网络、自动控制和人工智能等高新技术，实现了现场语音视频信息实时再现、数据存储、自动检测报警、自动远程控制等功能，以其直观、方便、信息内容丰富的特点，日益受到人们的青睐，被广泛应用于安全防范、无人职守、信息获取和指挥调度场合，如：银行柜台监控、交通违章和流量监控、边防监控、智能小区安全监控等等。

传统的视频监控技术的发展大致经历了三个阶段[1]。在九十年代以前，主要是以模拟设备为主的闭路电视监控系统，称为第一代模拟监控系统。主要特点是：使用黑白模拟摄象机，采用模拟方式传输图像信号，抗干扰能力低，图像质量差，系统功能单一。传输距离一般不能太远，主要应用于小范围内的监控，如大楼监控等，监控图像一般只能在控制中心查看。

九十年代初，随着计算机微处理器技术和彩色视频技术的发展与普及，视频监控技术有了第一次质的飞跃，原来的黑白图像变成了富有生机的彩色图像。人们利用计算机的高速数据处理能力进行视频的采集和处理，利用显示器的高分辨率实现图像的多画面显示，从而大大提高了图像质量。但由于网络技术和视频压缩技术的滞后，无法组建大型监控系统，监控信息局限于本地。这种基于 PC 机的多媒体主控台系统称为第二代数字化本地视频监控系统。

九十年代末，随着网络带宽、计算机处理能力和存储容量的快速提高，以及各种实用视频压缩处理技术的出现，视频监控步入了全数字化的网络时代，称为第三代远程数字视频监控系统。第三代视频监控系统以网络为依托，以数字视频的压缩、传输、存储和播放为核心，以智能实用的图像理解和分析为特色，引发了视频监控行业的技术革命。新的监控技术完全打破了传统的结构，依靠功能日益强大的计算机，不仅可以处理文本、数据、图形等，还可以处理视频、声音等信息，成为真正的多媒体监控终端。再加上网络和通信技术的发展，多

媒体信息的交互和共享趋向更广阔的空间。从局域网络到广域网络，从一个城市到另一个城市，从一个国家到另一个国家，都能完成在现场所能完成的一切任务。数字化、网络化的第三代视频监控技术，与传统的模拟监控技术相比较，还具有：便于模块化，通用性、可扩展性强；便于智能化、支持远程控制，监控效率更高；信号抗干扰强，便于对信号进行存取、查找、再次处理；易于安装管理维护，等优点。

第三代网络视频监控技术，融合了新兴的网络技术、多媒体技术、视频技术，是技术发展和社会进步的一次巨大飞跃，具有深远的现实意义。例如：交通监控系统不仅能实时收集交通流量参数，对违章车辆的拍照记录加强了交通监管力度，由此产生的警示作用有利于司机的行为自律，保障交通安全，倡导遵章守纪的良好社会风尚。对便捷的远程网络访问能力支持，使得视频监控技术可以进入普通百姓家庭，应用于幼儿看护，智能家居等场合，改变人们传统的生活方式。视频监控技术还可以应用于企业管理和生产经营管理，提高生产效率。第三代网络视频监控技术具有广阔的发展前景和巨大的商机，加之其强大实用的功能，可拓展的技术空间，良好的社会价值，因此受到了学术界、产业界和相关使用部门的高度重视，是当前信息产业发展的热点之一。

1.2 TIC6000 系列 DSP 的简介

1997 年，美国 TI 公司发布了新一代 DSPs 芯片 TMS320C6000，包括了定点系列和浮点系列[2]。其中定点系列是 TMS320C62xx，浮点系列是 TMS320C67xx，二者能够相互兼容。其中最早推出的 C6201 的运算速度已经达到了 1600MIPS，成为了当时业界首次突破 1000MIPS 的 DSP 产品，在数字信号处理器处理能力上创造了新的里程碑。在 2000 年 3 月，TI 发布了新的 C64xx 的内核，主频可以达到 1.1GHz，处理速度接近了 9000MIPS，使 64 系列的总体性能比 62 系列提高了 10-15 倍，成为了当今业界最强大的 DSP 产品之一，被广泛的应用在音视频多媒体领域。

TIC6000 系列 DSP 最主要的特点是在体系结构上采用了 VelociTI 的甚长指令字（VLIW, very long instruction word）结构。在 TI 的 VLIW 体系结构中由一个超长的机器指令字来驱动内部的 8 个功能单元，每个指令字包含了 8 个字段（指令），字段之间相互独立，各自控制一个功能单元，因此可以在单周期内发射出多条指令，实现很高的指令级并行效率，最多可以在单周期内同时执行 8 条指令。编译器在对汇编程序进行编译的过程中，决定代码中哪些指令合成一个甚长机器指令，在一个周期内并行执行。这种指令上的并行安排是静态的，也就是说所有的指令都是在编译期间决定的，一旦决定以后，无论 DSP 任何时候运行，它

都保持不变。TI 这样的 VLIW 指令体系结构也可以看作一种依赖于编译器的超标量（super scalar）实现方案，而且比起一般的超标量结构更易于实现。同时 C6000 的 VLIW 采用了类 RISC 指令集，多数指令拥有的相同流水级数，便于程序进行流水的优化。

TI 考虑到 DSP 面向的都是数据密集型的应用，频繁的数据访问和存储会大大的影响系统整体性能的发挥。在总线结构上，为了更有效的处理 DSP 上数据密集性算法，C6000 摒弃了传统计算机所采用的冯•诺依曼总线结构，采用了程序总线和数据总线分离的修正哈佛总线结构，提供了一套 256 位的程序总线，两套 32 位的数据总线以及一套 32 位的 DMA 专用总线，大大提高了总线上的数据吞吐量，使得在单周期内能够同时完成对程序代码以及两个操作数据的读取，缓解了数据瓶颈对系统性能的限制。同时，随着频繁的数据访问，数据地址计算的时间也会线性的增长，如果不在地址计算上做特殊的考虑，有时计算地址的时间比实际的算术操作的时间还长，因此 DSP 通常都提供了支持地址计算的算术单元——地址产生器。地址产生器与 ALU 并行工作，因此地址的计算不再额外占用 CPU 的时间。C6000 有 2 个地址产生器，可以满足单周期同时读取两个操作数据的目的，并且支持循环寻址模式。存储器的访问速度 DSP 的处理性能也有很大的影响。C6000DSP 内部集成有 1-8Mbits 的程序 RAM 和数据 RAM，对于有些片种，这些存储器还可以配制为程序 Cache 或者数据 Cache 使用。

除了以上所提及的特点以外，流水技术是 C6000DSP 具有高性能的另外一个主要原因。流水技术可以使得 2 个或者更多不同的操作重叠执行。在 C6000DSP 中所有的指令都按照取指（Fetch）、译码(Decode)、和执行(Execute)3 级流水运行，每一级又包含了不同的节拍(Phase)，每一个节拍可以被称为一级流水。流水处理使得若干条指令的不同执行阶段可以并行执行，理想情况下，一条 k 段流水能够在 k+(n-1)个周期内处理 n 条指令，而无流水的处理器处理 n 条指令则需要 nk 个周期，因此 C6000 的流水机制能够大大的提高程序的执行速度。

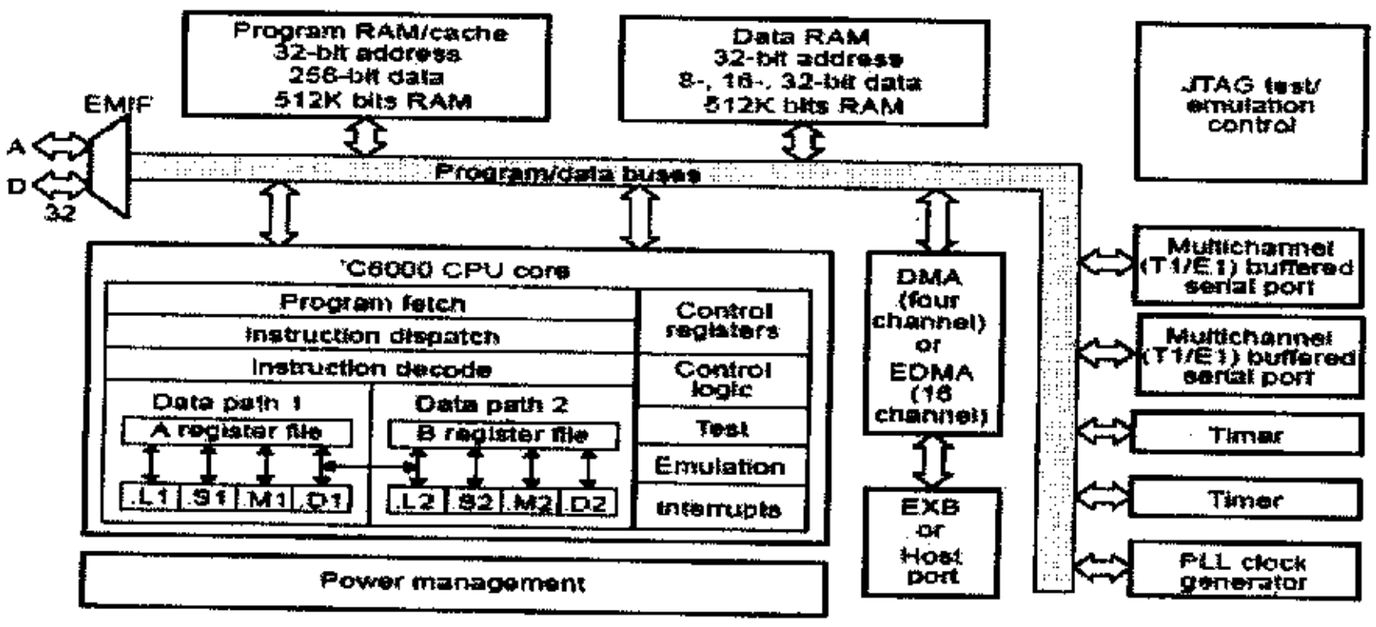


图 1.1： C6000DSP 结构图

1.3 低码率监控系统的结构

无线网络技术的发展，使得人们不再受电缆的束缚局限于有限的活动区域。2.5代无线通信网络GPRS和CDMA20001X都已经可以提供理论达到112kbps左右的带宽，而即将投入使用的第3代无线网络更可以把带宽提高到300kbps以上。无线网络也由传统的语音信号传输，发展为可以支持图像和视频的宽带传送。这些网络技术的发展为视频监控技术开拓了更为广阔的应用领域，也带来了新的挑战，无线监控系统就是其中之一。

图1.2就是针对无线信道设计的低码率监控系统的原理图，其核心部件是TI C6711DSP，用于对采集的原始图像数据进行压缩编码，压缩后的含有视频信息的码流发送至UART异步串口芯片，通过串口连接modem采用PSTN信道进行传输，或者是通过无线信道进行传输。由于整个监控系统的输出码率受到异步串口的限制，在9.6kbps-112kbps之间。PSTN网络理论带宽是56kbps左右，但是考虑到实际信道的状况以及用于报文头的开销，能够用于视频码流的典型带宽在44kbps左右，这也是目前2.5G无线技术数据业务的应用带宽，属于低码率视频监控系统。要实现这样的低码率监控系统除了硬件设计以外，更重要的是如何针对低码率监控视频的特点选择合适的编解码方案，并在以C6711 DSP为核心的硬件系统上实现和优化。

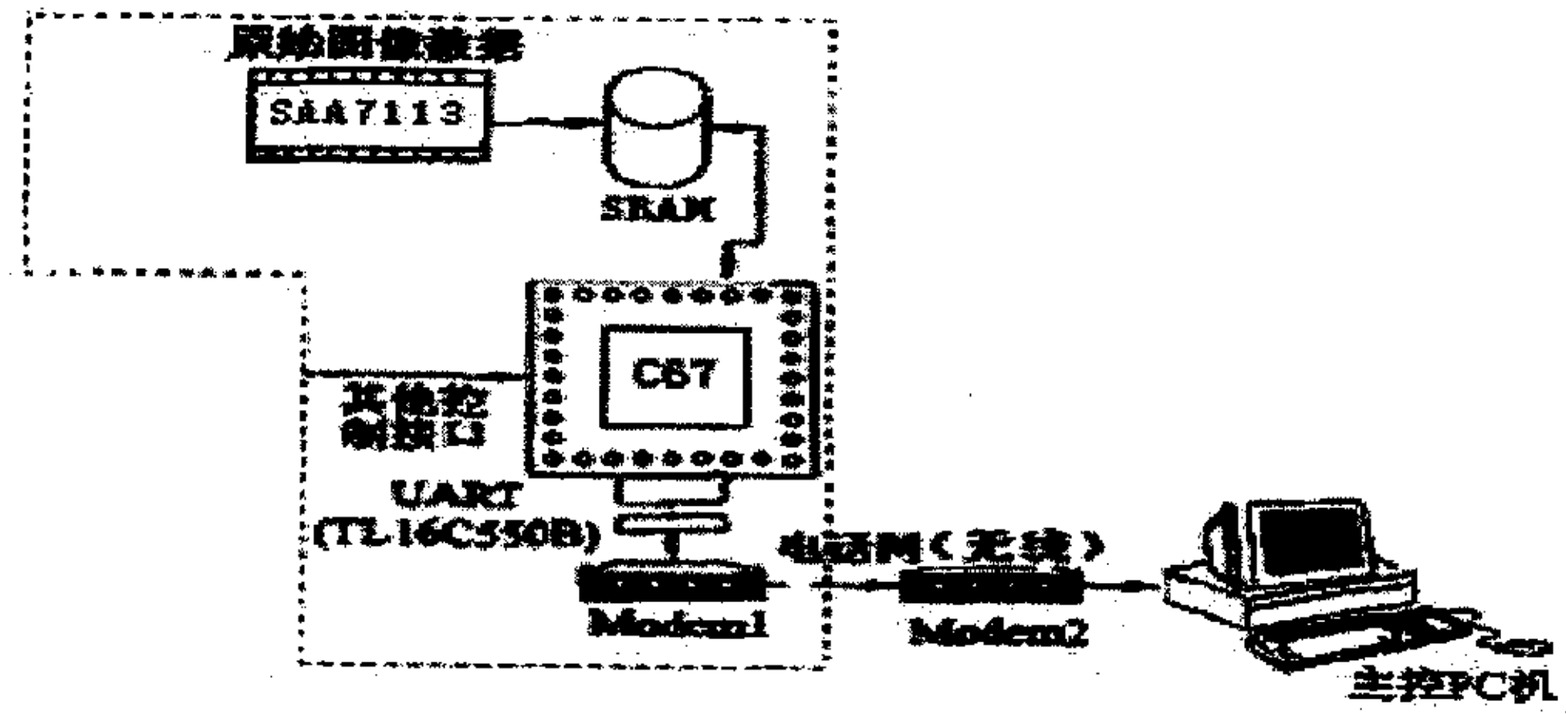


图1.2: 监控系统原理图

1.4 本文的工作

本文的工作着重于对9.6kpbs-112bps串口信道，特别是在典型的44kbps低码率信道中，监控视频编码解决方案的研究和基于C6711 DSP系统的优化。通过选取合适的视频编码手段，配合客户端的超分辨率算法，以及基于C6711 DSP的视频编码优化，在监控系统的串口低码率的范围内获取到较好的主客观质量。下面简单介绍本文的主要工作和创新点。

第二章中对现有的低码率视频压缩编码技术和国际标准进行了回顾和比较，并结合实际监控系统的软硬件条件，选择了 H.263 国际视频压缩标准作为监控视频编码算法的框架。

第三章中测试和分析 H.263 压缩算法主要高级压缩选项的性能，根据这些测试和分析结果，结合低码率监控视频和 C6711DSP 系统的特点，选取合适的编码方案。然后对编码程序中运动搜索算法和全零块预判算法进行了改进和优化。

第四章是本文的主要部分，其中主要针对 TIC6711 的特点，从编码程序的结构、函数代码和总线操作三个方面，进行优化和改进。根据整个监控系统的结构，集成视频采集模块和串口输出模块，并对 DSP 系统的 FLASH 引导做了相应的研究。

在第五章中我们给出了客户端软件的实现，并且提出了一种基于小波变换的图像超分辨率算法。

最后，在第六章中，对基于 TI6711DSP 的低码率监控视频编码的研究工作进行了总结，为将来的发展提出了展望和规划。

本文的主要贡献和创新之处有：

1．通过对 H.263 最主要的七个高级编码选项的性能测试，根据测试结果和 DSP 硬件特点，确定了低码率监控视频的编码方案。针对监控视频码率低、背景固定等特点，改进了 H.263 的运动搜索算法，在编码程序中采用了全零块预判算法，提高编码效率。

2．针对 TIC6711DSP 的特点对视频编码程序的结构、算法和总线操作进行了优化。其中包括根据 DSP 硬件资源限制，对编码程序结构进行调整，使调整后的程序对系统资源的需求更小，流程更规整；针对 C6711DSP 的 VLIW 指令体系，对程序局部函数代码进行汇编流水优化，提高程序的整体效率；针对 DSP 片内外总线结构的特点，对编码过程中的内存调用进行 EDMA 操作优化以及缓存结构的优化。经过优化后的编码程序在程序结构上更为紧凑、运行效率更高、数据流动更为顺畅。

3．针对监控系统硬件设计的结构，完成系采集和传输模块与 DSP 视频编码模块之间的集成，同时通过对 FLASH BOOT 系统引导的研究，保证了监控系统能够正常的脱机独立工作。

4．提出了基于小波变换的图像超分辨率算法，利用小波变化多分辨率的特点，结合双线性插值算法，使提升分辨率后的图像最大程度保留了原有的细节，达到较好的主客观效果。

下面，将分章节详细介绍这些具体的工作情况。

第二章 低码率视频压缩编码技术

视频编码压缩算法是监控系统的核心算法，选择一个合适的编码算法是能否成功实现整个系统的关键。由于整个监控系统的目标码率在 9.6kbps 到 112kbps 之间，典型码率是在 44kbps 左右，所以所选用的算法必须能够胜任这样的低码率视频压缩编码，并且在实现复杂度上能够符合 C6711DSP 的硬件能力。可喜的是，近年来围绕低码率视频压缩编码新技术的研究一直非常活跃，ITU 和 ISO 也相应陆续发布了针对甚低码率视频压缩编码的新标准，如 H.263、MPEG4、H.26L 以及最近提出的 H.264 等。本章，将对这些技术和标准作简单的介绍和对比，从中选择合适的编码技术，作为低码率监控系统的核心编码算法。

2.1 H.263 视频编码标准

H.263 是 ITU-T 于 1996 年制订的专门针对甚低码率视频通信应用的视频压缩标准。1996 年 3 月 H.263 推出了第一版，有 4 个高级模式[8]；1998 年 1 月推出了第二版，又称为 H.263+，修正了一个高级模式，增加了 10 个高级模式[9]；2000 年 11 月推出了第三版，称为 H.263++，再新增了 3 个高级模式[10]。H.263 版本的升级对旧版本保持兼容，但功能有了增强，提高了视频压缩效率，增强了传输鲁棒性，扩大了适用范围。

H.263 具有高压缩比、较强鲁棒性等特点，尤其适用于 PSTN 及无线或 Internet 网络环境下的视频传输，已被工业界广泛采用，如：可视电话标准 ITU-T H.324(PSTN)、H.320(ISDN)和 H.310(B-ISDN)等均采用了 H.263 为视频部分的编码标准。ISO/IEC MPEG-4 标准也大量借鉴了 H.263 的视频压缩方案。最初 H.263 确定目标比特率低于 64kbps，现在已无此限制，对 4CIF、16CIF 的支持使得 H.263 可以与高比特率视频编码 MPEG 系列标准相抗衡。

H.263 标准是在 H.261 标准的基础上发展起来的，两者的编码框架相似，不同的是在基本算法中，H.263 采用了更为精确的半象素运动估计取代 H.261 的整象素运动估计和环路滤波。H.263 采用的基本编码方式是帧内编码（INTRA）和基于运动估计和补偿的帧间编码（INTER）。

H.263 处理 4:1:1 的 YUV 空间视频信号，压缩处理的基本数据结构单元是 16×16 宏块（MacroBlock），一个宏块由 4 个 8×8 亮度块（Block）和 2 个色度块组成，相邻的若干宏块被定义为一个块组 GOB。运动估计在常规模式中是对 16×16 的宏块进行，得到的运动矢

量为半象素精度。协议本身没有规定具体运动估计算法。对经过预测的运动补偿帧间误差采用 8×8 的分块 DCT 变换去除空间冗余信息。DCT 系数经过量化、熵编码（Huffman）后形成码流。H.263 还引入了 MPEG 中 P、B 图像的思想，规定了 PB 帧、B 帧模式。

H.263 的高级模式主要有：

1．无限制的运动矢量：运动估计的匹配块可以部分位于图像区域之外，使得图像边界处的宏块仍然可以得到较好的预测。对摄像机运动和大图像格式的情况特别有用。

2．高级预测模式：运动估计是基于 8×8 块，而不是基于 16×16 的宏块，这样每个宏块可以具有四个运动矢量，运动估值更精确。解码器由预定义的加权表采用交迭块运动补偿技术（Overlapped Block Motion Compensation，OBMC）得到预测的象素值，OBMC 能消除块效应，改善解码图像质量。

3．PB 帧模式：PB 帧统一编码，PB 帧对应的宏块数据（共 12 个 Block）接在同一个宏块头后面。PB 帧中的 B 帧作双向预测。

4．基于语法的算术编码（SAC）：可以获得比 Huffman 编码更高的压缩比，但编解码器的复杂度会有所提高。

5．修正的非限制运动矢量模式：对运动矢量采用一种新的单精度的可逆变长编码（RVLC），支持任意大小的运动矢量编码。

6．高级帧内编码模式：由于帧内（INTRA）数据和帧间（INTER）预测误差数据具有不同的统计特性，对帧内数据采用单独的 VLC 编码表。同时，对帧内数据可以采用三种不同的预测方式：只对直流系数进行、对水平第一行系数进行、对垂直第一列系数进行。

7．去除块效应滤波模式：滤波可以很好地去除编码过程中引进的块效应。

8．片断结构（Slice）模式：形状位置可变的灵活片断结构（Slice）代替块组（GOB）。

9．追加增强信息模式：支持图像冻结、图像快照、视频分段等功能。

10．改进的 PB 帧模式：B 帧允许作双向、前向、后向预测。

11．参考图像选择模式：可灵活选择参考图像，克服帧间编码误差传播。

12．时间、空间、信噪比可伸缩模式：分级扩展，在同一码流中支持不同的时空分辨率，或不同信噪比的图像。

13．参考图像再抽样模式：对参考图像作变换后再用于预测。当参考图像帧与源图像格式不同时很有用。

14．简化的分辨率更新模式：编码器发送更新信息给较低分辨率图像帧来得到高分辨率图像。此模式对有复杂背景且运动剧烈的图像帧很有用。

15. 独立的分段解码模式：限制运动矢量估计于图像的某一分段内部。限制了错误传播。

16. 可选的帧间 VLC 模式：对某些大量化系数小零游程较多的帧间块使用帧内 VLC 编码表。

17. 修改的量化模式：对宏块、色度块、DCT 系数的量化作了更好的规定。

18. 数据分割模式：将图像帧中所有的宏块头信息、运动向量、DCT 系数集中在一起分别传输。这样便于分级保护。

H.263 的主要高级模式适合于不同的应用场合，能在不同程度上提高系统性能。

2.2 MPEG-4 视频编码标准

ISO MPEG 组织于 1999 年 1 月正式公布了 MPEG-4（ISO/IEC 14496）V1.0 版本，1999 年 12 月又公布了 MPEG-4 V2.0 版本。MPEG-4 制定的初衷是针对视频会议、视频电话的甚低码率编码。但在制定的过程中，MPEG 组织深切感受到，软硬件技术的发展和应用需求的变化，迫切要求将编码与基于内容的检索综合起来考虑。于是 MPEG 组织修改了计划，制定了现在意义上的基于内容（对象）的压缩编码标准：MPEG-4。

MPEG-4 标准为多媒体数据压缩提供了一个更为广阔的平台，它更多定义的是一种格式和框架，而不是具体的算法。人们可以在系统中随时加入新的有效算法模块，可以将各种各样的多媒体技术充分用于编码中，如压缩工具和算法、计算机视觉、计算机图形、图像分析合成、虚拟现实和语音合成等。MPEG-4 具有面向基于内容的交互性、高压缩率、灵活多样的存取模式等特点，在这里就不一一赘述，而只详细讨论 MPEG-4 的甚低码率视频编码特点。

MPEG-4 视频支持的码率和相应的功能如图 2.1 所示。可以看到，其功能集的底层核心是甚低码率视频压缩。MPEG-4 视频压缩算法借鉴了很多 H.263 中的思想和算法，但也有其独到的地方。

为支持面向对象的交互，MPEG-4 中引入了视频对象（Video Object）的概念。视频码流的语法分为视频会话（VS,Video Session）、视频对象（VO,Video Object）、视频对象平面层（VOL,Video Object Layer）、视频对象平面组（GVO）和视频对象平面（VOP,Video Object Plane）五层。一个视频序列由若干个 VS 构成。VO 是场景中的某个物体，由时间上连续的许多帧构成，是用户能够存取和操作（如剪切、粘贴等）的实体。若干个 VO 构成一个 VS。VOL 对应着一个 VO 的不同空间或时间分辨率，每个 VO 可以有多个 VOL。VOP 是 VO 或 VOL 在某一时刻的表象，即某一帧 VO。根据采用的编码方式，VOP 可以分为 I、P、D 和 S 四种类型，分别对应帧内编码、帧间预测、双向帧间预测、全景图（Sprite）编码方式。多个 VOP 构成

一个 GOV。GOV 是一个可选层，主要目的是为了提供随机访问、重同步、数据恢复等能力。

VOP 是 MPEG-4 中重要的数据结构，通过 VOP，高压缩比和基于内容的访问得以实现。VOP 的获取往往涉及图像分割、运动图像分析等技术，目前实时分割 VOP 还具有较大的难度。

VOP 由形状、纹理、运动三部分编码信息组成。其中纹理、运动估计、运动补偿在原理上同 H.263 是一致的，如半象素运动搜索、无限制的运动估计、高级预测模式、交迭块运动补偿 OBMC、DCT 变换及量化、Huffman 编码等。当 VOP 的形状取为传统的 MB 矩形时，编码算法和 H.263、MPEG-1、MPEG-2 是相近的。

形状编码是图像编码中第一次引入的技术。基于 16×16 宏块的形状编码算法可对任意形状的 VOP 进行编码。形状编码分为二进制和灰度 alpha 平面格式两种。二进制平面中的点只能取 0 和 255 两个值，而灰度 alpha 平面中的点可以在 0 到 255 之间取值。MPEG-4 允许采用基于语义的算术编码（CAE）或其他基于几何轮廓的编码技术对二进制形状块（BAB）编码。对灰度 alpha 值则视为二进制形状的纹理特征，附加在二进制形状编码之后。

MPEG-4 中还引入了全景 Sprite 图像的概念。Sprite 主要是针对背景图像提出的，为了有效编码背景视频对象，可以将其在一段时间的内容拼接成一副完整的背景图像，这样的图像就叫做 Sprite 图像。Sprite 图只需编码传输一次并存放在解码端，随后的图像只需要传输摄影机相对于背景的运动参数，就可以从 Sprite 上恢复所有的图像背景。Sprite 作为 MPEG-4 的重要概念之一，极大地提高了编码的效率。但 Sprite 编码的实现，必须满足两个前提条件：一个是前景与背景要能很好地分开，另一个是要做到无痕迹地从一段视频中拼接出 Sprite 图像。

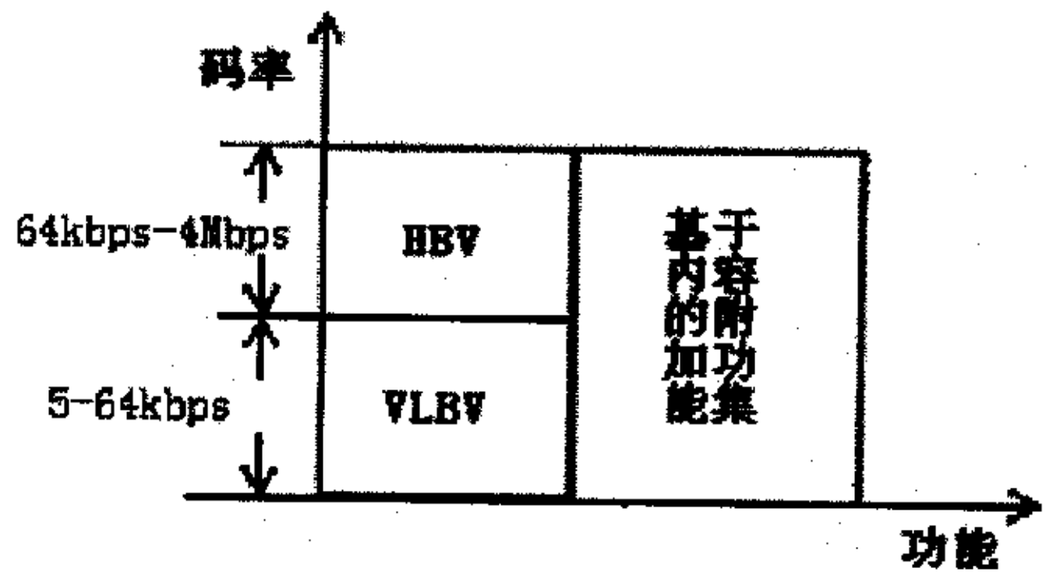


图 2.1：MPEG-4 视频支持的码率段和相应的功能组成

2.3 H.264 视频编码标准

H.264 标准的主导思想是与其他的视频编解码标准一致的——基于块的混合编码方法，

但是它同时运用了大量不同的技术，使得其视频编码性能优于现有的其他任何标准[14]。

H.264 与以往的编码方法不同的是如下几个方面：

1．H.264 采用了不同大小和形状的宏块分割与亚分割的方法。一个宏块的 16x16 亮度值可以按照 16x16、16x8、8x16 或 8x8 进行分割，而如果选择了 8x8 分割，那么还可以按照 8x8、8x4、4x8 或 4x4 进行亚分割，这些宏块分割与亚分割的模式可以组合出许多种宏块的分割方法。

2．H.264 可以达到四分之一像素的运动精度，这是通过利用整像素点的亮度值进行内差得到的。内差过程先是通过 6 抽头的滤波器来获得半像素精度，然后用线性滤波器来获得四分之一像素的精度。又由于 4:2:0 采样的关系，色度的运动精度就达到八分之一像素，这也是通过线性滤波器插值得到的。

3．H.264 还可以采用多参考图像（最多前向和后向各 5 帧）来进行运动预测。

4．H.264 也在 B 图像中利用后向运动预测，这和以前的标准是一致的，但不同的是 B 图像通过加权也能作为其他图像的参考图像。

5．H.264 根据相邻像素可能有相同的性质，利用了相邻像素的相关性，采用了新的 Intra 预测这种模式。通过当前像素块的左边和上边的像素（已编码重建）进行预测，只对实际值和预测值的差值进行编码，这样就能用较少的比特数来表达 Intra 编码的像素块信息

6．H.264 把运动估值和 Intra 预测的残差结果从时域变换到频域，使用了类似于 4x4 离散余弦变换 DCT（Discrete Cosine Transform）的整数变换。

7．H.264 使用了两种熵编码方法，即基于上下文的自适应变长编码 CAVLC（Context-based Adaptive Variable Length Coding）与一致变字长变码 UVLC（Universal Variable Length Coding）相结合的编码和基于上下文的自适应二进制算术编码 CABAC（Context-based Adaptive Binary Arithmetic Coding）。

以上所有的这些新技术使得 H.264 有着令人惊讶的压缩效率，但是同时也造成了编解码复杂度的大大增加，在 P4 2.0GHZ， 256MDDR 的 PC 上，H.264 压缩一帧 CIF 格式的图像就需要近一秒的时间，显然这样的压缩效率离实际应用的要求还有不小的差距。

2.4 小结：视频压缩标准的选择

在前面几节中，我们对当前最流行的三个低码率视频压缩编码标准做了简单的介绍。MPEG-4 基于对象的压缩算法引入了很多新的思想和模型，基于形状的编码、Sprite 编码、

网格对象编码、人脸对象编码等等，都有助于提高压缩性能。但我们也应该看到，上述的这些新颖的算法思想，或者针对特定的对象模型，或者在现有的图像理解、计算机视觉发展水平阶段，并不具备实时性或普遍适用性。要应用于监控系统视频编码，仍然还有待相关算法的研究和硬件计算速度的提高。

H.264 是性能最优异视频编码标准，它引入了 CABAC、帧内预测、多帧参考帧、多块分割等最新的视频编码技术，达到出色的视频压缩效果，但是同时我们也可以看到优异的性能是以高复杂的运算和高要求的资源所换取的，所以要在 TI C6711 DSP 上实现 H.264 编码标准，也不太现实。

H.263 是性能优异的，针对低码率视频压缩的国际标准。其多种高级编码模式，有助于用户根据不同的应用需求选取最优的编码方案。对比起其他两个标准，H.263 在压缩性能和计算复杂度上都有比较出色的表现，目前被广泛的应用在低码率嵌入式视频领域。

综合考虑算法的计算量、压缩性能、成熟度、资源开销等各方面的因素，考虑到 C6711 DSP 系统开发的需求，我们决定选用 H.263 作为低码率监控视频压缩编码的核心框架。在下一章中，我们将对 H.263 的各个高级编码模式的性能进行考察，从中选定适用于监控场景低码率视频编码和 6711DSP 硬件系统的编码方案，并且对相关的算法进行优化和改进。

第三章　编码方案的选择及其优化

H.263 的第二版本 H.263＋发布的时候，H.263 体系的高级编码模式已经多达 14 个了。这些高级编码模式，有的着眼于提高压缩效率，有的着眼于提高压缩后图像的质量，有的扩展了 H.263 算法的适用范围。不同的高级编码模式适用于不同的应用环境。图 3.1 就是 H.263 视频压缩编码的基本框架。本章中我们将根据该框架，详细分析 H.263 视频编码器的各个模块特点以及相关高级选项的性能，针对低码率监控视频的特点以及 C6711 DSP 的硬件资源状况，选取最合适的编码选项，并对一些相关的算法进行改进和优化[7, 8, 9, 10]。

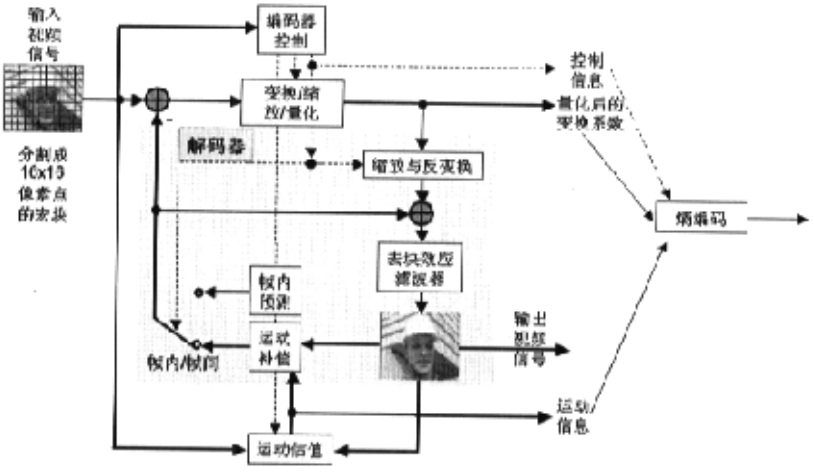


图 3.1 H.263 编码器框图

3.1　H.263 的编码方案选择

本节中将对提高压缩率或图像质量有宜的 H.263 的最主要七个高级编码模式做简单介绍，并实际考察其对编码性能的提升情况，结合监控场景低码率视频编码特点和 C6711DSP 的硬件要求，确定可以应用于实际 C6711 DSP 编码方案中的高级编码模式。

为保证测试的客观性和可重复性，在这一章中针对 H.263 高级模式的性能测试，除 PB 帧模式的测试以外，都是在 TMNENC 3.2 中进行的。使用的测试视频序列，是 foreman.yuv 标准测试视频，该测试序列前 170 帧，背景固定，摄像机参数变化不大，和监控场景视频有相近的特点。foreman.yuv 序列是 QCIF 格式，帧率为每秒 30 帧。

3.1.1 无限制的运动矢量模式

在普通情况下，运动估计的范围限于图像边界之内，运动矢量的大小也只能在[-16,15.5]内取值。而在无限制运动矢量模式下，运动估计的范围可以超出图像区域，使得图像边界处的宏块可以得到更好的运动预测。运动矢量指向的点如果位于图像边界之外，则用图像边界点的值代替。运动矢量的大小的范围也得到扩展。

在 H.263 第一版的码流语义环境下（PLUSPTYPE 不出现），运动矢量的值域从默认的[-16,15.5]增加到[-31.5,31.5]。此时运动矢量的编码表不变，只是运动矢量 MV_c 的取值范围变得和其预测值P_C相关：

$$-31.5 \le MV_C \le 0 \qquad if -31.5 \le P_C \le -16$$

$$-16 + P_C \le MV_C \le 15.5 + P_C \qquad if -15.5 \le P_C \le 16$$

$$0 \le MV_C \le 31.5 \qquad if 16.5 \le P_C \le 31.5$$

在 H.263 第二版（H.263+）的码流语义环境下（PLUSPTYPE 出现），运动矢量范围和预测值无关，而分为范围有限和范围无限制两种情况。在范围有限的情况下，运动矢量值域和图像格式有关。对 CIF 为[-32,31.5]，对 4CIF 为[-64,63.5]，对 16CIF 为[-128,127.5]，更大的图像为[-256,255.5]。对用户定义图像，水平和垂直的运动矢量值域可以不同。在范围无限制时，对运动矢量的值域没有限制。在这两种情况下，需要修改运动矢量的编码表，以支持大运动矢量。在半象素精度表示下的运动矢量“$x_n x_{n-1} \ldots x_1 x_0$”将会被编码为“$0x_n 1x_{n-1} 1 \ldots 1x_1 1x_0 1s0$”，其中 s 为符号位。

景物的运动速度越大，两帧间的时间间隔越大，运动估计得到的运动矢量就越大。无限制的运动矢量模式，有利于增加存在大运动矢量的视频场景的压缩比率。监控场景由于帧率低，具备这一特点，因此使用此高级编码选项有利于压缩性能的提高。特别是对于 PLUSPTYPE 不出现时的运动矢量的编码，并没有由于矢量值域的增加而耗费多余的比特。统计表明，大概率的运动矢量集中在小矢量值区域内。[-31.5,31.5]的矢量范围，对于图像尺寸不大于 CIF 的大多数视频场景已经足够。而当 PLUSPTYPE 出现（H.263+）时，不仅此模式下的运动矢量的编码要付出额外的比特代价，而且 PLUSPTYPE 也会占用去额外的一些数据比特。因此，对监控场景视频压缩，无限制运动矢量模式在无 PLUSPTYPE 时能取得最佳收益。

表 3.1 是对 QCIF 的 foreman.yuv 前 120 帧（帧率每秒 30 帧）的视频进行无限制运动矢量模式编码效果测试的数据，量化参数 QUANT 统一设定为 16，目标帧率为每秒 10 帧，编码生成 1INTRA＋39INTER，仅对参与编码的 39INTER 进行了统计：

	普通编码	无限制矢量 1	无限制矢量 2	无限制矢量 3
平均亮度 SNR	29.97	30.04	30.06	30.07
比特率（kbps）	32.46	30.31	30.20	30.85
提升压缩率	—	6.6%	7.0%	5.0%
编码时间（秒）	6.6	7.2	15.0	92.0

注：无限制矢量 1 为 263 无限制矢量模式；无限制矢量 2 为 263+范围受限的无限制矢量模式；
无限制矢量 3 为 263+范围无限的无限制矢量模式。编码程序为未经优化的 tmnenc3.2。

表 3.1：无限制运动矢量模式编码测试数据（不含 I 帧）

从表 3.1 中的测试数据可以看出，无限制运动矢量模式的使用，对上述测试序列压缩率的提升约为 6%，对图像质量略有提升。表 3.1 中的数据也印证了前面对三个无限制运动矢量模式性能的分析。使用 H.263 无限制运动矢量模式时，是在预测矢量的[-16,15.5]范围内作运动搜索，所以对编码速度的影响不大。而在 H.263+无限制运动矢量模式下，需要在完整的[-32,31.5]范围，甚至整幅图像的区域作运动搜索，因此使得编码速度大大下降，而对编码的性能却没有多大提升。同时，H.263 无限制运动适量模式一个宏块的运动搜索操作只需要 48*48 大小的内存用于存放搜索范围的数据， H.263+无限制模式则需要 64*64 大小，增加了对 C6711 DSP 存储空间的压力。总的来说，对于小图像，在没有好的运动搜索方法的支持时，H.263 无限制运动矢量模式已经足够，没有必要使用 H.263+的无限制运动矢量模式。

3.1.2 基于语法的算术编码（SAC）模式

在基于语法的算术编码（Syntax-based Arithmetic Coding, SAC）模式下，量化后的数据不经过 Huffman 的 VLC 编码，而是采用算术编码。算术编码所采用的先验累积概率表，随编码语义段的不同而不同。采用 SAC 编码，可以获得比 Huffman 编码更高的压缩比。表 3.2 是 SAC 模式编码测试数据，测试序列和测试条件同上节。从表中可看出，SAC 对上述测试序列的压缩率的提升也是在 6%左右。

由于熵编码特别是 SAC 编码过程是一个串行的操作，并不适合 C6711 的 VLIW 指令系统的优化，SAC 比起 Huffman 编码在 C6711 DSP 上的运行效率低很多，考虑到硬件系统的优化，在当前的系统中，暂时不采纳 SAC 模式。

目标帧率（Hz）	30	15	10	5
普通编码	64.11	44.87	36.30	25.18
SAC	60.29	42.26	34.16	23.74
性能提升	6.0%	5.8%	5.9%	5.7%

表 3.2：SAC 模式编码测试数据（平均码率，单位 kbps）

3.1.3 高级预测模式

在此模式下，运动矢量可以超出图像边界，如同在无限制运动矢量模式中一样，但不采用无限制运动矢量模式的大矢量范围。当和 PB 帧模式一起使用时，此模式只对 INTER 有效，B 帧不采用。

在此模式下，运动估计是基于 8×8 块，而不是基于 16×16 的宏块，这样每个宏块拥有四个运动矢量，运动估计更精确。同时，对亮度分量使用交迭块运动补偿（Overlapped Block Motion Compensation, OBMC），解码器由预定义的加权表采用交迭运动补偿技术得到预测的象素值，能够消除块效应，改善解码后的图像质量。

解码方在恢复图像象素值时，每个象素的亮度值是由三个预测亮度值加权求和，再除以 8 得到。象素色度值的恢复不进行交迭加权。这三个预测亮度值，分别由三个运动矢量进行运动预测得到：一个是本子块的运动矢量，另外两个是在水平和垂直方向上离此象素最近的邻近子块的运动矢量。OBMC 的作用其实相当于对各子块的边界进行中值平滑滤波，所以能够有效消除块效应。

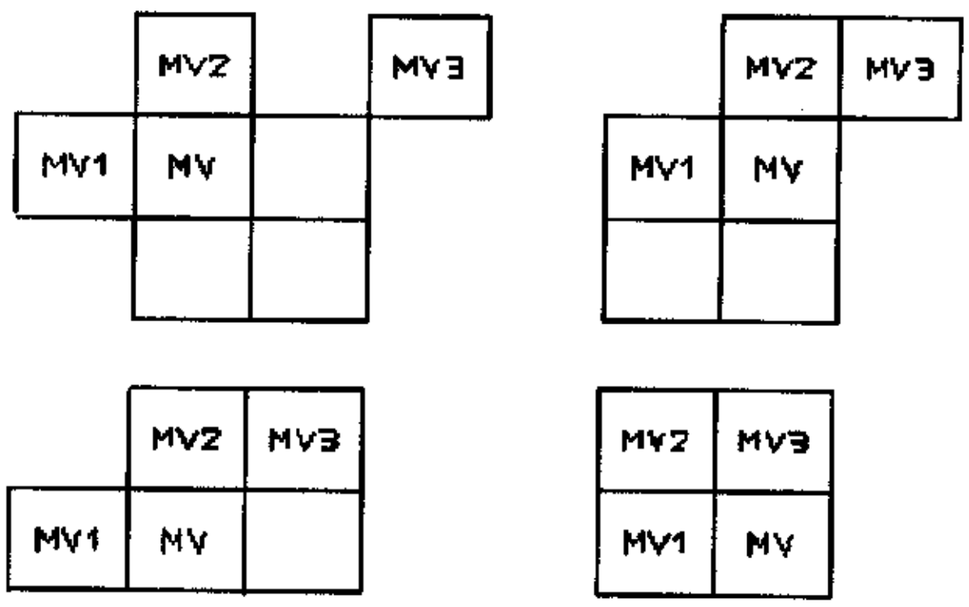


图 3.2：高级预测模式下的运动矢量预测

用于运动矢量预测的块位置在此模式下被重新定义，如图 3.2 所示。当只有一个运动矢量传输时，使用左上图的预测形式。色度块的运动矢量为这四个运动矢量和的八分之一，并精确到半象素点。

下面两个表格是对高级预测模式性能进行测试获得的数据，测试序列和测试条件同上节，统计不包含第一帧 INTRA。从表 3.3 中可以看出，高级预测模式有利于视频序列压缩率的提高，特别是在帧率较小，帧间差异较大的情况下。在高级预测模式下，每个宏块有四个运动矢量需要确定，但由于在 TMNENC 编码算法中，仍然是先做宏块运动预测，然后再在小范围内对四个块的运动矢量进行搜索，所以计算量和普通编码相当。

目标帧率（Hz）	30	15	10	5
普通编码	60.24	41.03	32.46	21.34
高级预测	57.34	38.50	30.67	19.78
性能提升	4.8%	6.2%	5.5%	7.3%

表 3.3：高级预测模式编码测试数据（平均码率，单位 kbps，不含 I 帧）

目标帧率（Hz）	30	15	10	5
普通编码	30.09	30.03	29.97	29.79
高级预测	30.33	30.36	30.35	30.20

表 3.4：高级预测模式编码 SNR 测试数据（亮度块）

3.1.4 PB 帧模式

PB 帧模式下时，每个宏块的 PB 帧数据是一个整体：P、B 帧对应的宏块数据接在同一个宏块头后面，一个宏块中有 12 个块，前 6 个为 P 块，后 6 个为 B 块。B 块中不含直流数据，只做双向预测。

B 帧（块）用前一帧和当前 INTER 数据预测。PB 帧模式下的 INTRA 块模式表示 P 块为 INTRA 编码，B 块仍然是 INTER 编码。此时当图像类型为 INTER 时，对 INTRA 块仍然有运动矢量，但只用于 B 块的运动补偿。

B 块的前、后向运动矢量的计算：设 MV 为 8×8 P 块的运动矢量，MV_D 为在 P 块运动矢量的基础上，再传送的一个用于 B 块运动矢量计算的差值（PB 块头部 MVDB 域未出现时 MV_D 为 0，出现时即为四个亮度 B 块的统一 MV_D 值）。H.263 中用 TR 值表示相邻帧之间的时域关系。设 TR_D、TR_B 分别为 P、B 帧相对于前一参考帧的 TR 增量，则前向和后向的 B 块运动矢量 MV_F 、MV_B 为：

$$MV_F = (TR_B \times MV) / TR_D + MV_D$$

$$MV_B = ((TR_B - TR_D) \times MV) / TR_D \quad \text{当 } MV_D = 0 \text{ 时}$$

$$MV_B = MV_F - MV \quad \text{当 } MV_D \neq 0 \text{ 时}$$

B 块的运动补偿：INTER 先解码为 P_{REC}，当 B 块的后向运动矢量 MV_B 所指的区域落在 P_{REC} 边界内时，使用双向运动补偿，预测值为前、后向运动补偿预测值的平均；对其他落在边界外的象素点，只使用前向预测。

B 块的量化阶梯由帧头内的 DBQUANT 域指明其与 P 块量化阶梯之间的关系。B 块的量化阶梯可以是 P 块量化阶梯的 5/4，6/4，7/4，8/4，并限幅于[1,31]。B 块的量化阶梯总不小于 P 块量化阶梯，因此解码后 B 帧的图像质量一般会比 INTER 差。测试结果数据如下表（测试序列同上节）：

目标帧率（Hz）		30	15	10	5
普通编码	Q=16	69.98	48.39	39.05	26.74
	Q=20	55.71	38.50	31.18	21.48
	平均	62.84	43.45	35.12	24.11
PB 帧		56.43	43.74	38.61	28.24
性能提升		10.2%	0.7%	-9.9%	-17.1%

表 3.5：PB 帧模式编码测试数据（平均码率，单位 kbps）

目标帧率（Hz）		30	15	10	5
普通编码	Q=16	29.78	29.81	29.78	29.73
	Q=20	28.66	28.65	28.66	28.58
	平均	29.22	29.23	29.22	29.16
PB 帧		29.77	29.55	29.38	29.09

表 3.6：PB 帧模式 SNR 测试数据（亮度块）

PB 帧中 B 帧的量化阶梯和 INTER 的不一样，测试中我们选定 INTER 的 QUANT 为 16，B 帧与 INTER 量化阶梯的关系为 5/4（其他可选关系还有 6/4，7/4，8/4），也就是说，B 帧的 QUANT 为 20。由于量化阶梯对编码后的码率大小和图像质量有比较大的影响，为更客观考察 PB 帧的压缩性能，我们分别考察了 QUANT 为 16 和 20 条件下的普通编码压缩性能，取两者的平均值与 PB 帧的编码结果进行对比。这里要特别指出的是，这种对普通编码性能上的折中只是一种近似。PB 帧与普通编码情况下的运动预测不一样，造成预测后进行编码的误差也会有较大差异；QUANT 值不同的两种普通编码压缩，由于 QUANT 的不同，后继 INTER 的参考帧会不一样，因此也会造成预测后的差值不一样。

纵观测试数据，PB 帧对编码效果的影响波动较大，在低帧率的情况下甚至出现了性能倒退。这是由 B 帧的编码方式所造成的。B 帧即便是能双向预测，预测值也是前后向预测的平均，对性能的提升有限。在低帧率的情况下，帧间的运动变化差异较大，PB 帧模式的使用，使 INTER 的间隔变为普通模式下的 2 倍，更加大了 INTER 间的运动差异。此时如果没有大运动矢量的支持，会出现 INTER 间运动匹配的恶化，进而影响 B 帧的运动补偿。因此，在低帧率的情况下，PB 帧反而出现了性能倒退。而监控场景视频在甚低码率下又必定是低码率。另外，PB 帧的编码，由于 B 帧双向预测的需要，对编码图像缓存的需求也是普通模式下的 2 倍。考虑到这些负面因素，我们决定，在监控场景甚低码率压缩方案中不考虑使用 PB 帧模式。

3.1.5 高级帧内编码模式

在此模式下，对 INTRA 块的低频部分数据也采用预测编码，提高对 INTRA 数据的压缩能力。同时，由于 INTRA 数据和 INTER 预测误差数据具有不同的统计特性，对 INTRA 数据采用单独的 VLC 编码表。

INTRA 块数据可以采用三种不同的预测方式：只对直流 DC 系数预测、垂直方向 DC 和 AC 系数预测、水平方向 DC 和 AC 系数预测。具体采用何种预测方式，由编码器决定。在宏块头应标明预测方式。

只对 DC 系数预测时，预测值从上方和左方最近相邻块的 DC 系数得到，除非这些块不在同一个图像片段（Slice 或 GOB）内，或不是 INTRA 块。垂直方向 DC 和 AC 预测，是针对 DC 和水平第一行 AC 系数进行，其预测值，从上方最近相邻块中的 DC 和水平第一行 AC 得到；水平方向 DC 和 AC 预测，是针对 DC 和垂直第一列 AC 系数进行，其预测值，从左方最近相邻块的 DC 和垂直第一列的 AC 得到。

预测后系数的扫描方式也有变化，对 DC 预测，仍是传统的 ZigZag，其他两种预测方式下，垂直预测进行水平扫描，水平预测进行垂直扫描。

表 3.7 和表 3.8，是对 foreman.yuv 的第一帧进行高级帧内编码性能测试得到的数据，编码程序为 TMNENC。从中我们可以看到，高级帧内编码模式能够提高图像压缩率和图像质量，特别是在大量化阶梯时，效果更明显。但是在监控视频中摄像头固定，背景变化不明显，鲜有全场景的变换，没有必要高频率的进行 INTRA 编码，所以 INTRA 占整体码流的比重很小，同时高级帧内预测需要存储一行宏块的 DCT 变换后系数，增大对 DSP 存储空间的压力。

量化参数 Q	8	12	16	20	24
普通模式	35.10	32.58	30.86	29.55	28.61
高级 INTRA	36.08	33.55	31.91	30.53	29.49

表 3.7：高级帧内编码模式 SNR 测试数据（亮度块）

量化参数 Q	8	12	16	20	24
普通模式	26968	19384	15360	12912	11400
高级 INTRA	26976	19120	14784	11696	9712
性能提升	0%	1.4%	3.8%	9.4%	14.8%

表 3.8：高级帧内编码模式测试数据（单位 bit）

3.1.6 去除块效应滤波模式

此模式对 INTER、INTRA、改进 PB 帧的 INTER、EINTER、EINTRA 的 8×8 亮度和色度块边缘，进行四象素滤波，消除块效应。由于滤波会影响帧间预测，故需要内嵌于运动补偿环节中。去块效应滤波和高级预测模式中的 OBMC 联合使用，可以取得更好的图像效果。

滤波前需要对[0,255]的象素值进行限幅，以确保滤波后的值在这个范围内。对图像边缘不进行滤波，在独立片断编码模式下，滤波不超过 Slice 边界或 GOB 边界。滤波后的值要用于预测。对改进的 PB 帧，B 帧的后向预测使用限幅后滤波前的 INTER 值，前向预测使用滤波后的值。

滤波使用垂直和水平方向边界上的 4 个像素，两两各属于不同的块。设为 A、B、C、D。其中 A、B 属于同一块 Block1，C、D 同块 Block2，Block2 位于 Block1 的右方或下方。滤波条件：相邻两块至少有一个为编码块。设滤波后的值为 A1、B1、C1、D1，则滤波算法如下：

B1 = clip(B + d1)　　C1 = clip(C - d1)

A1 = A - d2　　D1 = D + d2

d = (A-4B+4C-D) / 8

d1 = UpDownRamp(d, STRENGTH)

d2 = clipd1((A – D) / 4, d1/2)

注：UpDownRamp(x, STRENGTH) = SIGN(x) * (MAX(0, abs(x)- MAX(0,2*(abs(x) – STRENGTH)))); clip(x)为限幅函数，将 x 的值限于[0, 255]；clipd1(x, lim)将 x 的值限于 abs(lim)。

STRENGTH 和量化参数 QUANT 有关。如果 Block2 为编码块，QUANT 为 Block2 的量化参数，否则为 Block1 的量化参数。滤波先对水平边缘，再对垂直边缘进行。

表 3.9 和表 3.10，是对 foreman.yuv 的前 120 帧视频，在量化参数 QUANT 设定为 16 时，对不同目标帧率，进行去除块效益滤波测试得到的数据，编码程序为 TMNENC。可以看到，随着帧率的减小，去除快效应滤波对压缩性能的提高越明显。这是因为在低帧率的情况下，由于帧间差异增大，运动补偿的残差也会增大，量化引入的块效益就越明显。应用去除块效应滤波模式后，解压缩图像的视觉效果也有所改善。上述测试证实了去除快效益滤波的有效性，可以考虑在监控场景甚低码率压缩方案中应用。

目标帧率（Hz）	30	15	10	5
普通编码	64.11	44.87	36.30	25.18
去除块效应	63.70	43.56	35.24	24.00
性能提升	0.6%	2.9%	2.9%	4.7%

表 3.9：去除块效应滤波模式编码测试数据（平均码率，单位 kbps）

目标帧率（Hz）	30	15	10	5
普通编码	30.09	30.03	29.97	29.79
去除块效应	30.06	30.10	30.13	30.09

表 3.10：去除块效应滤波模式编码 SNR 测试数据（亮度块）

3.1.7 可选的帧间 VLC 模式

帧内（INTRA）VLC 编码表与帧间（INTER）VLC 编码表相比，对大系数，短零游程的数据块能够提供更好的压缩比。在可选的帧间 VLC 编码模式下，可以对这种系数变化明显（有较多大系数）的 INTER 块采用 INTRA VLC 进行编码。对使用 INTRA VLC 的 INTER 块，编码器不在码流中插入额外的标识信息，而是留待解码器判别。使用 INTRA VLC 编码 INTER 块时，解码器若仍使用原来的 INTER VLC 解码，则得到的系数将会多于每块 64 个。因此，解码器先用 INTER 解码，若得到多于 64 个的系数，则可判定为 INTRA 编码，换用 INTRA VLC 解码。

在此模式下，对指示四个亮度块中，编码块情况的 CBPY 的编码也进行了修改。当 $CBPC_5$ =1 且 $CBPC_6$=1 时（即两个色度块都含有非零交流系数），INTER 块的 CBPY 采用 INTRA CBPY 编码。

目标帧率（Hz）	30	15	10	5
普通编码	30.09	30.03	29.97	29.79
帧间 VLC	30.13	30.04	29.97	39.82

表 3.12：帧间 VLC 模式编码 SNR 测试数据（亮度块）

目标帧率（Hz）	30	15	10	5
普通编码	60.24	41.03	32.46	21.34
帧间 VLC	60.61	40.84	32.56	21.29
性能提升	-0.6%	0.5%	-0.3%	0.2%

表 3.11：帧间 VLC 模式编码测试数据（平均码率，单位 kbps，不含 I 帧）

上面两个表格，是对可选的帧间 VLC 进行性能测试得到的数据，测试条件同上节。可以看到，这一模式对编码性能的提高几乎没有什么作用。可以在监控场景甚低码率压缩中考虑不使用。

3.1.8 编码方案的确定

我们对 H.263 最主要的七个高级编码选项的编码性能进行了测试，主要考察其对压缩比、图像质量的提高以及对 DSP 系统的硬件要求。从中我们可以看到，对压缩性能提高效果最好的是：无限制运动向量、高级预测、高级帧内编码、去除块效应滤波。其中，高级帧内编码只对 INTRA 帧有效，去除块效应滤波在帧率小的时候效果明显。无限制运动向量推荐使用 H.263 第一版本中的模式。这几个高级模式联合使用，在图像质量不降低的情况下，将有可能减少码率 20%以上。但是高级帧内编码模式需要保存一行宏块的变换后的系数，用以进行帧内预测，这大大增加了编码时的内存开销。以 QCIF 格式为例，一行有 11 个宏块，每个宏块的变换后系数需要占用 512 个字节大小内存，那么如果存储一行宏块所需要的内存开销就达到了 5.5k 字节，对于片内只有 64k 字节大小 RAM 的 TIC6711DSP 而言，这样的内存开销是巨大的。同时由于监控视频背景固定，没有剧烈的场景变换，所以 INTRA 编码间隔在监控视频中通常很大，因此在实际 DSP 应用中不采用高级帧内预测。

去块滤波模式的作用不仅仅是体现在降低码率，提高 SNR 上，更重要的是在低码率视频压缩中，由于输出码率有限，帧率低，造成帧间差异变大，可能出现明显的块效应现象，会大大影响主观视觉质量，所以去块模式对于低码率视频压缩而言是至关重要。

通过测试还发现，在低帧率下，PB 帧模式并没有如预期的那样，对提高压缩性能起到积极的作用。使用 PB 帧模式的确可以提高压缩比，但其主要原因是因为 PB 帧中的 B 帧的量化参数比 INTER 大。将量化值分别设为 PB 帧中的 INTER、B 帧的量化值，对视频进行一般压缩编码，取其压缩性能（SNR 和压缩比）均值，与 PB 帧的压缩性能进行比较。可以看到，PB 帧对编码效果的影响波动较大，在低帧率的情况下甚至出现了性能倒退。这主要是因为，B 帧加大了 INTER 的间距和帧间差异，使 INTER 间运动匹配恶化，进而影响 B 帧的运动补偿所致。另外 PB 帧模式需要对 B 帧的图像先进行缓存，然后再编 INTER 后再进行编码，这样的编码模式不能满足监控视频对视频延迟的要求，同时对 B 帧编码图像的缓存增加了硬件系统的存储器要求。考虑到这些负面因素，在我们的 DSP 系统的视频编码方案中，将不使用 PB 帧模式。另外，经过测试，可选的帧间 VLC 模式，对压缩性能的提高几乎没有什么作用，在我们的 DSP 系统的视频编码方案中，也将不予采用。

经过上面的分析，最后在监控系统视频编码方案中采用了无限制运动向量、高级预测、去除块效应滤波这 3 个高级选项，用以提高视频压缩的效率。在选定了编码模式以后，我们在监控系统的典型 44kbps 左右的码率下，对三个高级模式联合编码的几个方案进行了测试，结果如表 3.12 所示. 我们可以看到在 5 个方案中方案 1 和 5 编码帧率都不到 5，当视频播放

帧率低于 5 的时候，会有明显的停顿现象，同时视频图像由于压缩所造成的图像缺陷，很容易被人眼所察觉。方案 4 虽然帧率高，视频流畅，但是由于量化系数过大，造成视频图像信息丢失严重，重建图像质量差（SRN 值只有 30），如再经过放大处理，图像质量更不能为人眼所接受。方案 2、3 在视频流畅度和重建图像质量上达到一定的平衡，帧率和 SNR 都还能为人眼所满足，所以我们选定方案 2、3 作为在典型码率下监控系统的视频编码方案。由于方案 2、3 都是对低分辨率的 QCIF 进行编码，为了使客户端能够看到高分辨率的图像，我们将在第五章对超分辨率算法做研究，以完善监控系统的编码方案。

方案	编码格式	量化系数	平均码率（kbps）	平均帧率（帧/秒）	SNR（db）
1	QCIF	8	45．71	5	33．91
2	QCIF	10	45．18	7．5	32．68
3	QCIF	12	48．18	10	31．65
4	QCIF	16	45．16	15	30．16
5	CIF	16	49．37	3．5	31．86

表 3.12：几个编码方案测试

3.2 编码算法的改进和优化

上一节分析了 H.263 编码器的高级选项，并且明确了监控视频的编码方案。这节将对 H.263 编码算法在原有的基础上进行改进和优化，优化的目的是针对低码率监控视频特点，减少编码复杂度，提高算法效率，增加编码速度，为下一步移植到 C6711 DSP 系统打好基础。H.263 是一个面向普通视频编码的标准，可是监控视频除了具有普通视频的特点以外，还有摄像头固定，背景块多，经过量化后全零块多等这些独有的特点。所以下面我们首先对运动搜索关键算法进行了改进，随后针对监控视频前后背景相对固定，帧间预测、量化后会出现一定数量的零宏块，提出了背景块预判的运动搜索算法和零块预测算法，减少计算复杂度，加快编码速度。

3.2.1 运动搜索算法的改进

视频信号序列中，相邻两帧图像之间具有很强的相关性。采用运动补偿技术进行帧间预测，消除帧间冗余，是视频压缩编码的常用技术之一。如基于 MC-DCT 技术的 MPEG、H.263 视频压缩，基于 MC 的三维小波压缩，等等。

运动估计是运动补偿技术的第一步，也是最重要的一步。运动估计的目的是得到帧间运

动物体的运动矢量。常用的运动估计算法有梯度法、像素递归法和块匹配法等[15]。块匹配算法由于运算量小，实时性好，而被众多国际编码标准所采纳。

在块匹配算法中，首先将图像划分为若干互不重叠的子块，并认为子块内所有象素的运动矢量都相同（不进行背景与运动物体的区分），在参考帧中寻找最相似的子块（匹配块），以此确定子块的运动矢量。寻找匹配块的过程就被称为运动搜索。为节省计算量，在参考帧中的运动搜索一般只在一定的范围内进行。

基于块匹配的算法中，最重要的有两个问题：一是子块匹配判决准则，二是运动搜索算法。这两者选择不同，常常构成不同的算法。

子块匹配判决准则有：归一化互相关函数 CCF、子块亮度均方差 MSE、子块亮度平均绝对误差 MAD 等。CCF 和 MSE 的计算量较大，在实际中一般不采用。而 MAD 计算方便，对匹配的精度也影响不大，故最常用。绝对误差之和 SAD 与 MAD 相比，只是少了一步求均值的除法操作，因此现在的运动估计算法中大多是采用 SAD 作为块匹配准则。

SAD 的定义是：

$$SAD(i,j)=\sum_{m=1}^{M}\sum_{n=1}^{N}\left|S_K(m,n)-S_{K-1}(m+i,n+j)\right|$$

式中，$S_K(m,n)$为第 K 帧位于(m, n)的象素值；i, j 分别为水平和垂直方向的偏移量，取值范围为 $-dm \le i, j \le dm$。(i, j) 即为待求的运动矢量，dm 为运动搜索范围。

运动搜索是一个计算量耗费较大的过程，因此，对运动搜索新算法的研究在学术界一直没有停止过。评价运动搜索算法性能的主要参数有：搜索计算量和搜索精确度。

常用的运动搜索算法主要有[17]：

1．FS（全搜索算法）：全搜索即在搜索范围内逐点搜索，每搜索一次计算一次 SAD，当 SAD 达到最小值时，求得最佳匹配宏块。全搜索算法的计算量最大，需要进行$(2dm+1)^2$次搜索。全搜索算法搜索到的结果是全局最优。

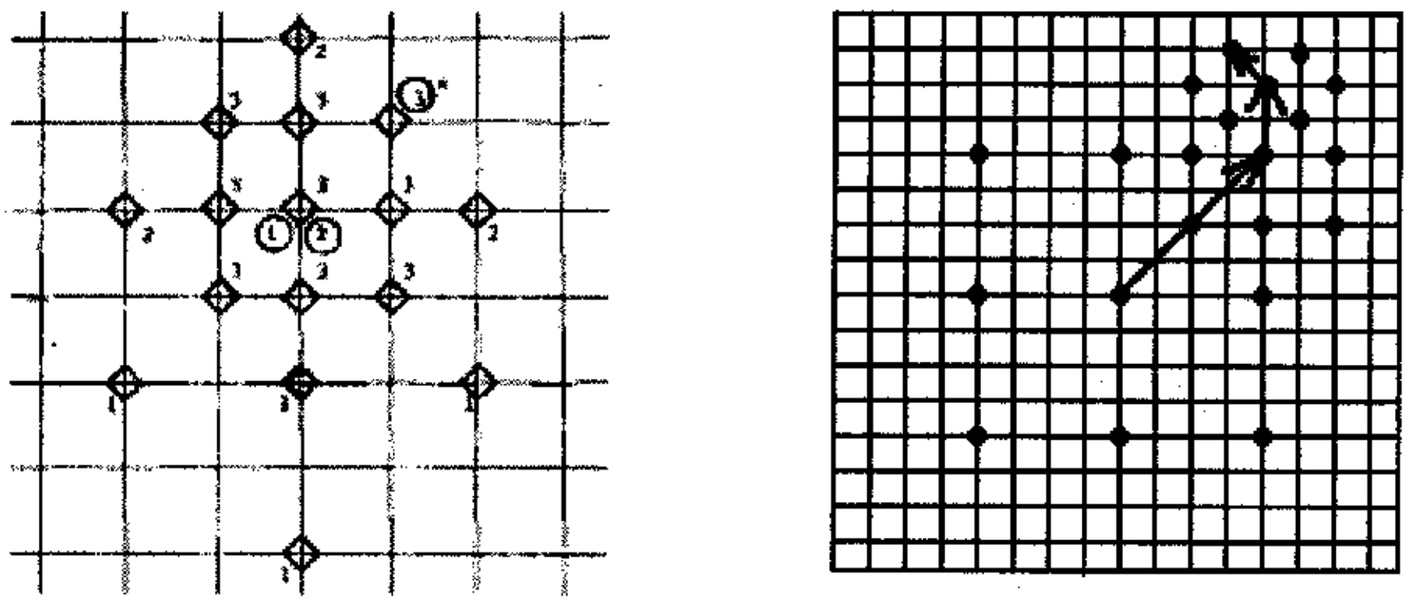
2．LOGS（二维对数搜索法）[18]：初始搜索半径为 $t=\log_2 dm$，计算水平、垂直方向上与中心相距 t 的四个点，和中心，共 5 个位置上的 SAD，取其中 SAD 最小的点做为下一步搜索的中心；当 SAD 最小的点是中心点时，改变搜索半径 n=n-1，继续进行 5 点 SAD 比较和搜索中心移动；当 n=1 且 SAD 最小的点是中心点时，结束搜索，中心点即为最佳匹配点。图 4.1(a)为 LOGS 搜索算法示意图。

3．CDS(共轭方向搜索法)[19]：先沿水平方向进行搜索，对中心点及其左右水平相邻点共三点进行 SAD 判决，更新下次搜索的中心点为 SAD 最小的点。如果到达搜索边界或者 SAD 最小点为本次搜索中心点，则换搜索为垂直方向。对中心点及其垂直相邻点进行 SAD

判决，搜索中心更新，直到垂直搜索结束，则为最佳匹配位置。

4．CS（交叉搜索法、正交搜索法）[20]：从搜索区域中心开始，取在水平方向上和中心点相聚 t=dm/2 的两点，与中心点比较 SAD，最小的为新中心；在新中心的垂直方向上，取距中心 t 的两点，与中心点比较 SAD，最小的为新中心。再次换为水平方向，但 t = t-1。重复进行搜索，直到找到最佳匹配点。

5．3SS（三步搜索法）：初始搜索步长取为搜索范围宽度的 1/4，即为 dm/2，计算中心和和外围 8 个方向共 9 个搜索点的 SAD 值，取其中 SAD 最小的点做为下一次搜索的中心。下一次的搜索步长取为上一次搜索步长值的一半。重复这一过程，直到搜索点位于搜索区域的边界，或搜索步长为 1，则找到最佳匹配位置。当 dm=8 时，3SS 算法的搜索步长分别为 4，2，1，这就是 3SS 算法名字的由来。显然，随着 dm 的增大，这种搜索算法的步骤可以不止三步。当搜索区域 dm=8 时，3SS 算法的初始搜索步长 4，对于小运动矢量来说太大，因此有了改进的三步搜索算法（N3SS）[21]、4 步搜索算法（4SS）[22]等。3SS 搜索算法的示意图见图 3.3。

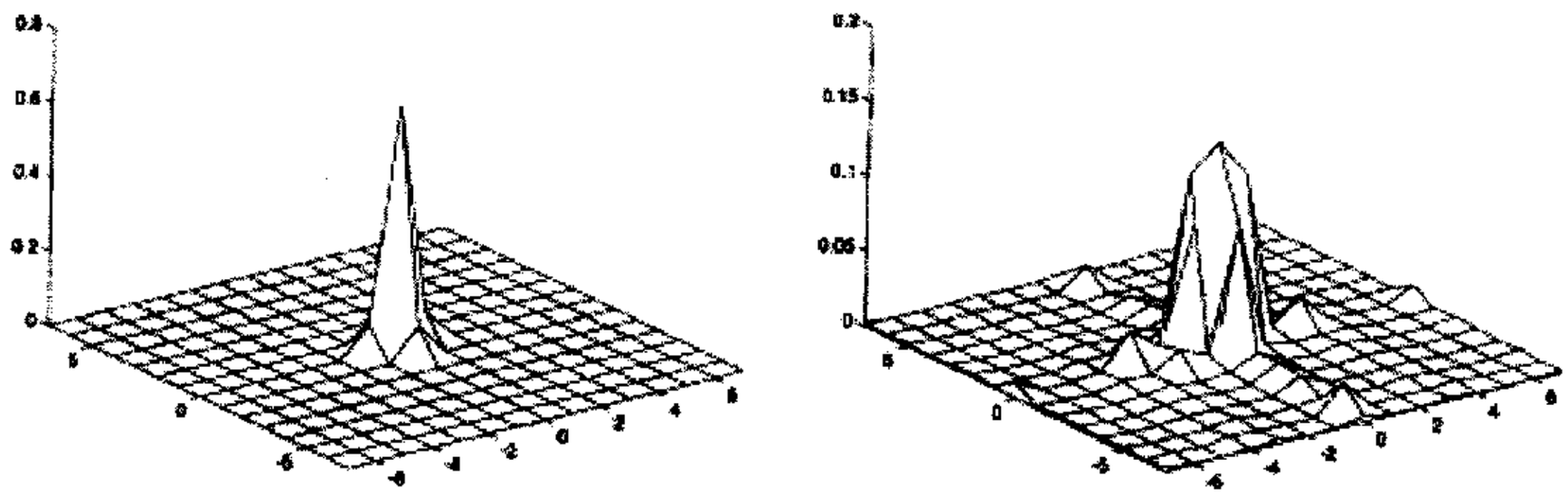


(a)LOGS 搜索算法　　(b)3SS 搜索算法

图 3.3：搜索算法示意图

原有的编码程序中就采用了 FS 和简化了的 LOGS（与中心进行匹配比较的，只有上下左右四个最近相邻点）两种搜索算法。

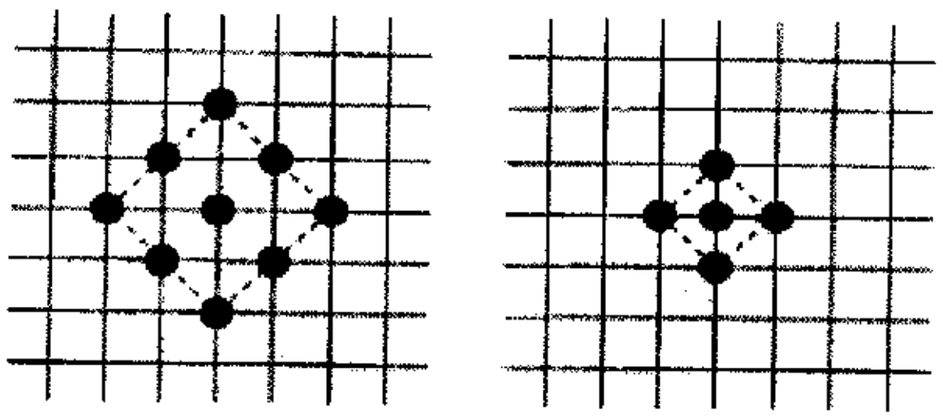
上述 2 到 4 算法都利用了如下假设：当偏离最小误差方向时，判决函数是单调上升的，因此沿使判决函数值减小的方向进行搜索，就能达到最佳匹配点。而实际上，在搜索区域内的判决函数曲面往往不是单调的，而是存在局部最小点。当每步搜索考察的点间距太小时（如在 3×3 领域内），往往容易仅仅达到局部最小点，使得运动补偿的性能下降；而搜索点间距过大，又不利于对小运动矢量的搜索。文献[23]中对视频运动矢量的大小进行了统计，结果表明，如图 3.4 大多数的运动矢量值都比较小，集中于搜索区域的中心。



(a)Salesman 序列 100 帧　　(b)Miss America 序列 100 帧

图 3.4 运动矢量概率分布图

文献[24]中所述的钻石搜索算法 DS 很好的解决了这两个问题。在 DS 中使用了大、小两个搜索模板，见图 3.5 所示。搜索时，先使用大钻石模板，当最优点位于大钻石模板中心时换用小钻石模板，小钻石模板比较得到的最优点为最佳匹配点。



(a)大钻石模板　　(b)小钻石模板

图 3.5 钻石搜索算法中使用的两种搜索模板

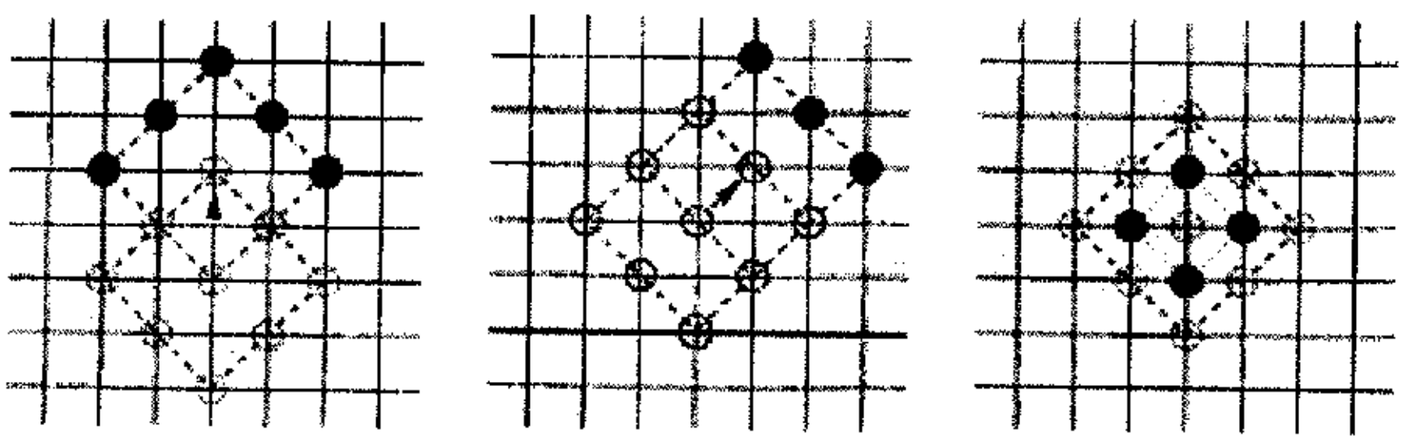


图 3.6 钻石搜索的三种搜索点位置更新方式

由于每次搜索位置改变时，都有部分判决函数计算点与上次重叠，所以，在新的搜索位置上，没有必要计算所有模板上位置的判决函数值，而只需要计算新加入点的判决函数值，如图 3.6 上实心黑点所示。

图 3.7 是一次完整的钻石搜索示例图，共计算了 24 个点的判决函数值。与图 3.3 的 3SS 搜索算法图相比，很明显的可以看出钻石搜索法具有搜索模板半径合适，抗局部最小干扰强的特点。大、小钻石模板的换用，又可以提高对小运动矢量的搜索速度。在实际测试中，钻

石搜索算法也表现出了比传统搜索算法更为优良的性能[25]。使用钻石搜索算法，平均搜索次数一般小于 20 次/宏块，同时还能基本保持和全搜索算法基本相同的压缩比和图像质量。

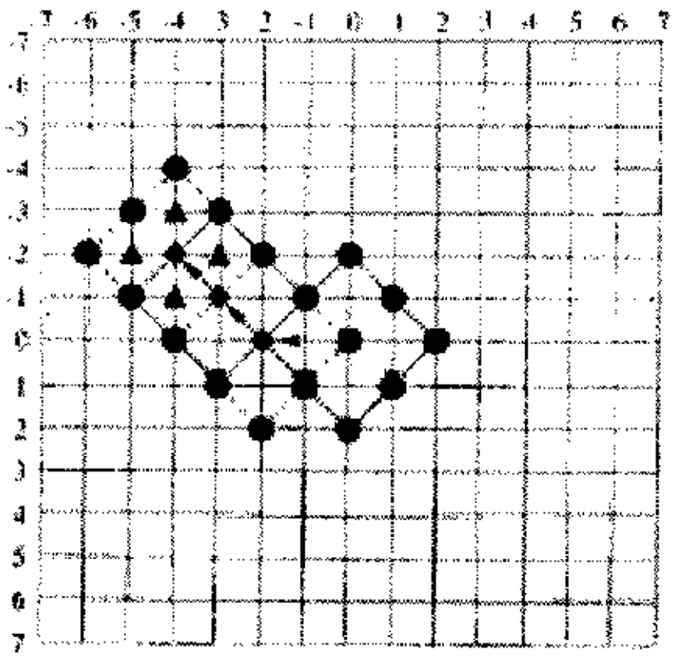


图 3.7 一次完整的钻石搜索示例

针对监控视频的特点，我们在钻石搜索算法的基础上稍做修改，进一步降低运动搜索的计算量。由于在监控系统的应用中摄像头基本是固定的，这个特点决定了监控视频对象有相对固定的背景，视频信息主要集中在前景的变化中。我们可以把待搜索块分为未被运动物体覆盖的背景块和被运动物体覆盖的前景块。由于前者的运动矢量是零矢量，所以对于背景块不必要进行运动搜索。如果我们能够在运动搜索前或者在搜索中预先判断出零矢量的背景块，就可以大大的减少运动搜索的计算量。为此，我们设计了一个简单而有效的背景块判断方法。首先，保存前一帧运动搜索的结果，把所有零矢量宏块作为当前帧背景块的候选块，对于这些块在进行钻石搜索之前，将零矢量处的 SAD 和一个阈值做比较，小于该值的则直接判断为背景块，不进行运动搜索；大于该值的块，则认为可能被前景所覆盖，进行运动搜索。该方法十分的简单，而且存储后选块的所需要的空间也十分的有限，对于 QCIF 图像而言，一共 99 个宏块，那么用于存储后选块的空间只需要 99 个字节，对于 C6711 DSP 而言也是一个很小的代价。对于高级运动搜索选项，如果一个宏块被判为背景块，那么它所属于的 4 个块也被认为是是背景块，这是因为如果 4 个块中有运动的前景块，那么必然影响到整个宏块的 SAD，也可以在对宏块的判决中反映出来，所以不必专门考虑对块的判断。下面给出我们最后采用的运动搜索算法的简单流程：

1、 初始化，所有的宏块为非背景块，memset(bkbuf,0)，设置阀值为 thrd=800.

2、 计算当前宏块(i,j)中心位置（零矢量处）的 SAD_0，vector(i,j)={0,0}。

3、 if(SAD_0 <thrd)

{

set(bkbuf(i,j))

```
        go step 5
    }
    4、diamond search
    5、if(zerovector)
            set(bkbuf(i,j)
       else
           zero(bkbuf(i,j))
```

3.2.2 全零块预判

在 INTER、PB 帧等进行 INTER 编码的图像中，往往存在这样的 8×8 块，其运动补偿后的残差较小，经过 DCT 和量化操作后，为全零块。在低码率监控视频中，由于背景固定，背景块的残差很小，通常是全零块，另外，低码率视频的量化值比较大，也会造成很多块成为全零块。在 H.263 编码标准中，对全零块，无需进行后继的 VLC 编码，只要在宏块头信息的 CBPY(Coded Block Pattern for Luminance)、CBPC(Chrominance)、CBPB(B-blocks)域中进行标识即可。

对于 INTER 编码的图像帧，在运动搜索的过程中，需要进行匹配函数计算。这些匹配函数值，可以提供有关运动补偿残差的信息，在进行 DCT 变换前，对全零块进行提前判定，可以节省 DCT 变换和量化操作所需的计算量。文献[24-27]等，在这一方面进行了探讨。其全零块提前判定的算法，大致可以分为无错判和有错判两类。无错判方法，经过严格的不等式推导，得到全零块判定的充分条件，据此进行判决，不存在误判，但会有漏判。有错判方法，在不等式推导过程中，一般利用了 DCT 变换的某些大概率性质，减少漏判，同时也会出现误判，但误判对编码的性能影响不大。两类方法都可以根据量化阶梯的大小自适应改变判决阈值。对于不同的图像序列和不同的量化阶梯大小，使用这些全零块提前判定方法，可以减少 20%-70%不等，均值在 40%左右的 DCT 和量化操作时间[25、26]。

性能较好的一种无误判全零块提前判定方法及其推导过程为：

设运动补偿后的残差值为 $r(i,j)$， $i,j=0,\cdots 7$，

则，对于最常用的 SAD 运动搜索匹配函数， $\mathrm{SAD}=\sum_{i=0}^{7}\sum_{j=0}^{7}|r(i,j)|$， $i,j=0,\cdots 7$

又，8×8 二维 DCT 变换定义为：

$$F(u,v)=\frac{1}{4}C(u)C(v)\sum_{i=0}^{7}\sum_{j=0}^{7}r(i,j)\cos[\frac{(2i+1)u\pi}{16}]\cos[\frac{(2j+1)v\pi}{16}]$$

$$C(u)=\begin{cases}\dfrac{1}{\sqrt{2}} & u=0\\ 1 & \text{其他}\end{cases}\qquad u,v=0,\cdots 7\ ,$$

在 H.263 中，当量化参数为 Q 时，DCT 系数的量化公式为：

$$Cof\ \ (u,v)=signF\ \ (u.v)\frac{|F(u,v)|-Q/2}{2Q}$$

从上式可知，全零块判定的充要条件是：$|F(u,v)|-Q/2<2Q$，

即：$|F(u,v)|<\frac{5}{2}Q$

又，根据“和的绝对值小于绝对值的和”，以及余弦函数小于 1 的特性，可以推导得：

$$|F(u,v)|<\left|\frac{1}{4}C(u)C(v)\sum_{i=0}^{7}\sum_{j=0}^{7}r(i,j)\right|<\frac{1}{4}\left|\sum_{i=0}^{7}\sum_{j=0}^{7}r(i,j)\right|=\frac{1}{4}\mathrm{SAD}$$

故，当 $\frac{1}{4}\mathrm{SAD}<\frac{5}{2}Q$ 时，必有 $|F(u,v)|<\frac{5}{2}Q$ 可以无误差的断定该块为全零块。

表 4.3 是对全零块提前判定进行测试得到的数据表格。测试条件同上节，测试序列仍然是 300 帧的 foreman.yuv QCIF 图像。可以看到，量化值越大，所判决出的全零块平均比例就越高。在判决出的全零块比例较高时，全零块提前判定，能够节省编码时间。

3.2.3 小 结

运动搜索是监控视频编码的主要部分，约占整个编码时间的 40%左右，一个好的运动搜索算法能够高效的完成运动搜索，保证运动估计的结果正确性。在这节里，我们分析了包括全搜索、二维对数搜索法、三步法以及钻石搜索法在内的目前最常用的搜索算法的优劣，并且根据监控视频的特点，在钻石搜索的基础上，添加了对背景块预判算法，该算法通过上一帧的搜索结果得到当前帧的后选背景块，然后对后选背景块进行阀值比较来判断当前的背景块。由于监控视频中背景块较多，通过这样的搜索算法可以节省相当的运动搜索时间，也不会造成内存的浪费。除了运动搜索算法外，我们还针对监控背景的特点，采用零块预判算法，减少了计算量，这些算法的改进，都使得原有的 H.263 编码算法，更适应我们监控视频

编码特点，提高了视频编码的效率和效果。

第四章 基于 C6711 DSP 的监控编码系统

在上一章里，我们针对监控视频的特点对 H.263 视频编码程序进行了改进和优化。在这章中，我们把程序移植到 C6711 DSP 嵌入式系统中，并根据 C6711 DSP 的特点在 DSP 的平台上进行程序的优化和改进，以满足整个监控系统的视频编码需求。

根据第 3 章所确定的方案，DSP 在典型码率下只需要有每秒 10 帧 QCIF 的编码能力，但是考虑到监控系统将来信道带宽的改善和系统升级的需求，所以对于 DSP 的编码程序而言，我们将把优化的目标编码帧率定为 15 帧左右 CIF 每秒。对于最高主频为 150MHZ 的 C6711 而言，要达到这样的编码帧率还需要做大量的优化工作。这些工作不仅仅包括运算指令上的加速，更为重要的是对程序中许多不合理的结构和算法进行调整，以适应 C6711DSP 的硬件环境，比如在内存分配上，C6711 只有 64kB 的片内存储器，而原有的程序单单一个 QCIF 格式的半像素插值就会产生一个 4*176*144，大约 102kB 大小的内存区域。如果不做修改不仅仅浪费了大量的内存资源，而且不得不把运动搜索的数据也放在片外，增加了 DSP 数据总线的负担，降低了程序的效率。象这样不适合 DSP 嵌入式环境的代码在程序中还有许多，所以在本章中我们将从程序结构、内存安排、DSP 汇编指令、CACHE 缓冲等多个方面对视频编码程序进行优化和改进。我们首先分析 C6711 的硬件特点，包括中央处理器、存储结构、流水线系统等，然后根据这些特点，结合程序的具体情况，对编码程序进行结构、代码以及总线等三个方面的优化，最后介绍整个编码监控系统的集成。其中在编码程序优化中，我们暂时没有对高级预测部分进行优化。

4.1 C6711DSP 的硬件特点

C6711DSP是TIC6000系列中最早发布的，高性价比的浮点DSP芯片，最高时钟频率可以达到150MHZ。图4.1是C6711的整体结构图[28]，其中包括了使用VelociTI VLDW TIC62x的DSP内核，这个DSP内核拥有两路共32个32位的通用寄存器和8个高度独立功能单元，这8个独立的功能单元分为A、B两路，分别由共6个逻辑运算单元和2个16位的乘法单元构成，最高可以同时执行8条指令，理论计算能力达到1200MIPS。同时还包括了一套256位的程序总线，两套32位的数据总线以及一套32位的EDMA专用总线。我们还可以看到C6711并没有采用片内程序和数据存储器分开的结构，而采用了L1/L2二级片内缓冲的存储结构[29,30]，提供了64kB大小的程序、数据统一片内L2存储器和大小分别为4kB的直接映射一级L1P程序

缓存和双路联想的L1D一级数据缓存，以提高程序和数据的读写速度。它的32位EMIF接口支持最多512M的片外存储寻址，支持SDRAM、SBRAM以及各种异步储存器。所有这些都为C6711处理大量的数据提供了强而有力的保证，使它能够胜任我们监控系统的视频编码任务，下面的几节中，我们将对C6711的各个部分做详细的分析，同时针对C6711DSP的特点对编码程序进行优化和改进。

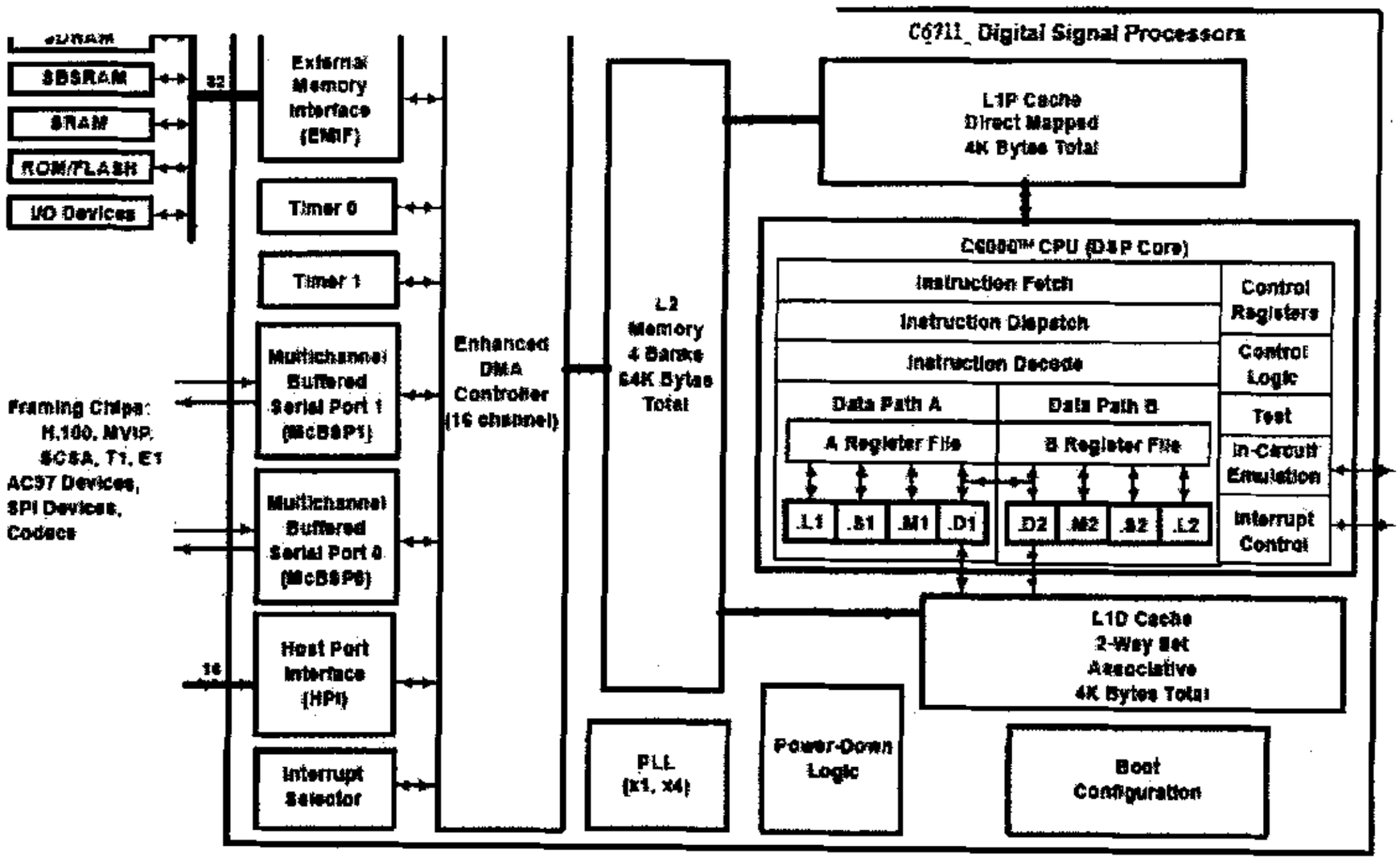


图 4.1 C6711 DSP 结构图

4.1.1 中央处理器（CPU）

在图 4.1 的最右边就是 C6711 的中央处理器，也就是 DSP 的内核。和其他的 C6000 系列 DSP 一样，C6711 采用 TI 最新的 VelociTI VLDW TIC62x 的 DSP 内核，主要包括以下三个部分[28]：

1、 程序读入以及译码、分配机构，包括程序取指单元、指令分配单元和指令译码单元。

2、 程序执行机构，包括了 2 个对称的数据通道(A 和 B)，2 个对称的通用寄存器组（每组 16 个），2 个对称的功能单元（每组 4 个），控制寄存器和一些控制逻辑以及中断逻辑等。

3、 芯片测试和仿真端口及其控制逻辑。

C6711 的中央处理器采用增强型的哈佛总线结构，其程序总线和数据总线分开，从图 4.1 中可以看到 C6711 中央处理器从 L1P 一级程序缓存中直接读入指令，从 L1D 一级数据缓存中读取数据。它的程序总线宽度为 256bit，程序取指单元每次从 L1P 中读取 32 个字节

大小的指令字，组成一个指令包，一个指令包中包含了 8 条 DSP 指令。随后的指令分配单元和指令译码单元都具备每个周期处理并传递 8 条 32 位 DSP 指令的功能。同在一个取指包内的 8 条 DSP 指令，可以组成一个或者多个执行包，译码单元把执行包按顺序送入执行机构，进行程序的执行。理想的状况下一个指令包只包含一个执行包，也就是说 8 条 DSP 指令在单周期内完全的并行执行。中央处理器和程序读入机构保证了 C6711 在理论上能够单周期执行 8 条 DSP 指令。

采用大而且统一的寄存器堆以及完全并行的功能单元，是 C6711 中央处理器执行结构的最大特点。执行机构由 2 个类似的可以进行数据处理的数据通路 A 和 B 构成，每个数据通路有 4 个独立的功能单元(.L 、.S、.M 和.D)和一组 16 个 32 位寄存器的通用寄存器组。8 个功能单元负责所有的 DSP 指令执行，并且完全独立，以保证在单周期内能够并行 8 条 DSP 指令。除了个别内存读取和跳转指令外，其他所有的 DSP 算术逻辑指令都采用通用寄存器作为操作数，而且使用 RISC 的指令结构，大多数 DSP 指令有相同的流水级数，保证程序能够高速的运行。其中. D 功能单元负责专门的数据寻址计算，保证寄存器和存储器之间数据交换的顺畅和高效。另外，A、B 通路之间还有 2 个交叉通路(1X 和 2X)，进行两个通路之间的数据交换。

除了通用寄存器外，中央处理器中还有许多控制寄存器，这些 32 位的寄存器不用做一般 DSP 指令的操作数，而是专门负责 C6711 的某些功能和状态的选择与配置，对于控制寄存器的操作要使用专门的 MVC 指令。常用的控制寄存器如表 4.1 所示。

分析 C6711 中央处理器的结构，强大的并行处理能力是它最大的特点，可见要在最高主频 150MHZ 的 C6711 上使编码程序达到理想的编码速度，如何优化程序，使其能够充分利用 C6711 中央处理器并行能力是至关重要的。尽管 TI 提供了一个强而有力的编译环境 CCS，通过 CCS 优化编译能够使程序代码达到相当程度的并行，但是仅仅依靠编译器的优化对我们的编码程序而言是远远不够的，在后面的内容中我们将分析程序代码并行优化的过程，使得程序能够充分利用中央处理器的资源，发挥出其强大的并行计算功能。

控制寄存器缩写	控制寄存器名称	描述
AMR	寻址模式寄存器	指定使用线性或者寻址模式
CSR	控制状态寄存器	包括全局中断使能位、高速缓冲存储器控制位和其他控制和状态位。
IFR	中断标志寄存器	显示中断状态。
ISR	中断设置寄存器	允许软件控制中断
IER	中断使能寄存器	允许使能或者禁止个别中断
ISTP	中断服务表指针	指向中断向量服务表的起始地址
IRP	中断返回指针	保存从可屏蔽中断返回的地址
NRP	不可屏蔽中断返回指针	保存从不可屏蔽中断返回的地址
ICR	中断清楚寄存器	允许软件清楚挂起的中断

表 4.1：C6711 常用控制寄存器

4.1.2 片内 2 级高速缓存结构

C6711 片内 RAM 采用 2 级高速缓存结构，程序和数据拥有各自独立的高速缓存。片内的第 1 级程序缓存称为 L1P，第 1 级数据缓存称为 L1D，程序和数据共享的第 2 级存储器称为 L2[30]。

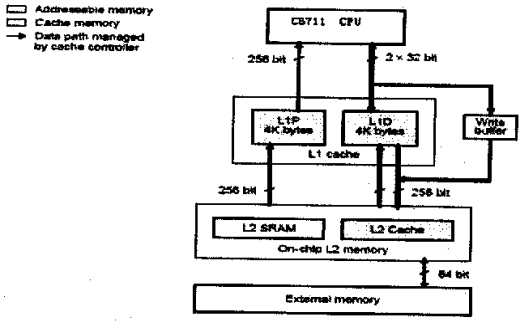


图 4.2：C6711 2 级缓存结构

在图 4.2 中可以看到 C6711 的两个 1 级缓存 L1P 和 L1D 都是 4K 字节大小，缓存中的数据读写是由 C6711 缓存控制器自动管理的，不需要程序员的干预，从 1 级缓存中读写数据都可以在单周期内完成，不会造成正在运行程序的停顿。C6711 的 L2RAM 是程序和数据共用的，大小 64KB，除了可以作为 RAM 存放程序和数据外，还可以划分出部分或者全部的 L2 作为 2 级缓存，L2 用做缓存的部分将不能再映射成地址用做程序或者数据的存储。

C6711 的 L1P 采用直接映射结构（direct map cache）,只能作为缓存，不能设置为映射

存储器。L1P 的行大小为 64B，即 2 个取指包的宽度，可以缓存 64 组。CPU 发出 32bit 的取指地址以确定 L1P 的映射位置。用户可以通过 CSR、L1P flush 基址寄存器、L1P 冲洗字寄存器和 cache 配制寄存器对 L1P 进行控制。不过通常情况下，我们不对 L1P 进行人为的干预，因为 L1P 是单口缓存，即 CPU 只从 L1P 读取指令，没有 CPU 向 LIP 写指令的情况发生。

L1D 采用的是双路联想结构(2-way set associative cache)，所谓的双路缓存就是由两个直接映射缓存组成的，每路直接映射缓存的大小为 2KB，这样 L1D 相当于由两个独立的缓存组成，一共 4KB 大小。L1D 的行大小为 32B，每路缓存可以缓存 64 组。L1D 这样的设计结构是专门针对 DSP 数据密集型运算和 C6711 中央处理器双路结构特点的。正如上节所分析的，C6711 的中央处理器可以在单周期内读取两个数据，那么如果只有一路缓存，两个数据容易映射到缓存的同一个位置，CPU 需要刷新缓存，造成数据读取的阻塞，而双路的设计可以使得这种情况的发生几率大大减小，有利于 C6711 程序数据的高速读取。L1D 缓存更新采用 LRU(least recently used)策略，即目前缓存中最少被读写的数据最先更新，这样的更新策略也是符合 DSP 的数据读写规律的，同时 L1D 和 L1P 不同，它是一个双口缓存，CPU 既可以读也可以写 L1D。由于在写 L1D 时，CPU 不会自动更新缓存中对应的存储器的数据，所以需要在程序中控制 L1D 的读写，特别是在配合 EDMA 进行数据读写时，以保证程序数据的正确。对 L1D 的控制可以通过对 CSR、L1D flush 基址寄存器、L1D 冲洗寄存器和 cache 配制寄存器的读写来完成。

C6711 的片内 RAM L2 是由 4 个 16KB 块所组成的 64KBRAM，可以同时存放程序和数据。每一个 16KB 的内存块都可以被单独配置成为地址映射存储器或者是缓存，这样的设计结构可以让用户根据不同的程序需求灵活的配置 L2。

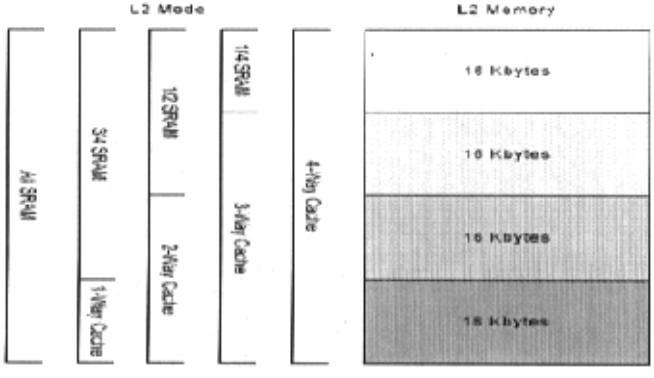


图 4.3：C6711 L2 配置模式

可以从图 4.3 中看到，每增加一个块的缓存，L2 的缓存就多一路，如果 64KB 的 L2 都被配置为缓存，那么就是一个 4 路的缓存。

C6711 的片内存储结构最大特点就是 2 级缓存结构，它的数据读写流程如图 4.4 所示。可以看到缓存的设计使得中央处理器能够在单周期内获取 1 级缓存中的数据和程序，如果没有命中 1 级缓存，则向 L2 发送请求，CPU 被阻塞 5 个周期，如果 L2 也没有命中，那么 CPU 将被继续阻塞，直到 L2 从片外存储空间取得相应的数据，送入 L1，再送入 CPU。对外部存储的访问往往会引起 10 个周期以上的 CPU 阻塞，所以当数据或者程序在外部存储空间时，容易造成总线阻塞，使得程序性能大幅度下降。针对 C6711 这个特点，如何安排程序和数据的存放以及对缓存的合理控制，是程序能否高速运行的关键之一。

另外需要注意的是，当使用 EDMA 进行数据传输时，对于 L2 内的数据，C6711 的缓存控制器会自动控制缓存中的数据回写和缓存更新，保证 RAM 中和缓存中的数据一致，但是对于外部存储器中的数据，需要程序员对缓存中数据的回写和缓存的更新做人为的控制，如果没有将缓存中的数据写回到 RAM 或者及时更新缓存，造成缓存内和映射储存器内的数据不一致，从而导致程序出错。

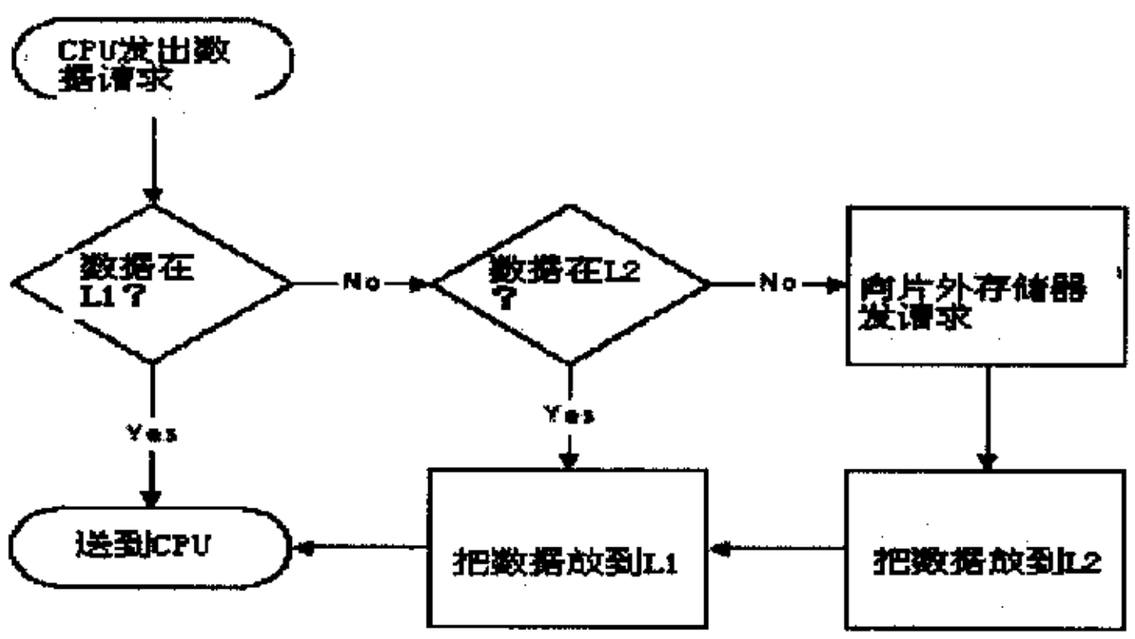


图 4.4：C6711 数据读取流程

4.1.3 流水线结构和指令系统

现在的微处理器通常都是用结构的高复杂性来换取速度的提高，其中最常用的方式就是流水线结构。所谓的流水线，就是微处理器把指令的处理分成独立的几个子操作，每个子操作都由不同的部件来完成。对于微处理器的每个部件来说，每隔 1 个时钟周期就可以进入一条新的指令，这样在同一时间内，就有多条指令交迭的在不同部件内同时被处理，这样的工作方式就被称为“流水线”(pipeline)[2]。在 4.1.1 节中提到 C6711 的 CPU 具有取指、分

配、译码和执行等 4 个单元部件，所以 C6711 的所有指令都可以按照这样的 4 级流水来运行。其中取指的单元又可以分为 4 个节拍，分别是程序地址产生（PG）、程序地址发送（PS）、程序访问等待（PW）和程序取指包接收（PR）；执行单元根据不同的指令有不同的节拍。图 4.5 是 C6711 的流水线流程图，说明了流水线连续运行的时序。其中连续的各个取指包都包含了 8 条并行指令，即每个取指包只有一个执行包。各个指令包以每个时钟 1 个节拍的方式通过流水线，可以看到，在周期 7，取指包 FP_n 的指令达到了 E1 节拍，取指包 FP_{n+1} 的指令正在译码节拍（DC），取指包 FP_{n+2} 处在分配节拍（DP），其他取指包也在相应的节拍中。流水线的结构使得 C6711 的指令能够交迭并行的运行，当流水被充满时，例如图 4.5 中的周期 11，DSP 达到最高的运行效率。

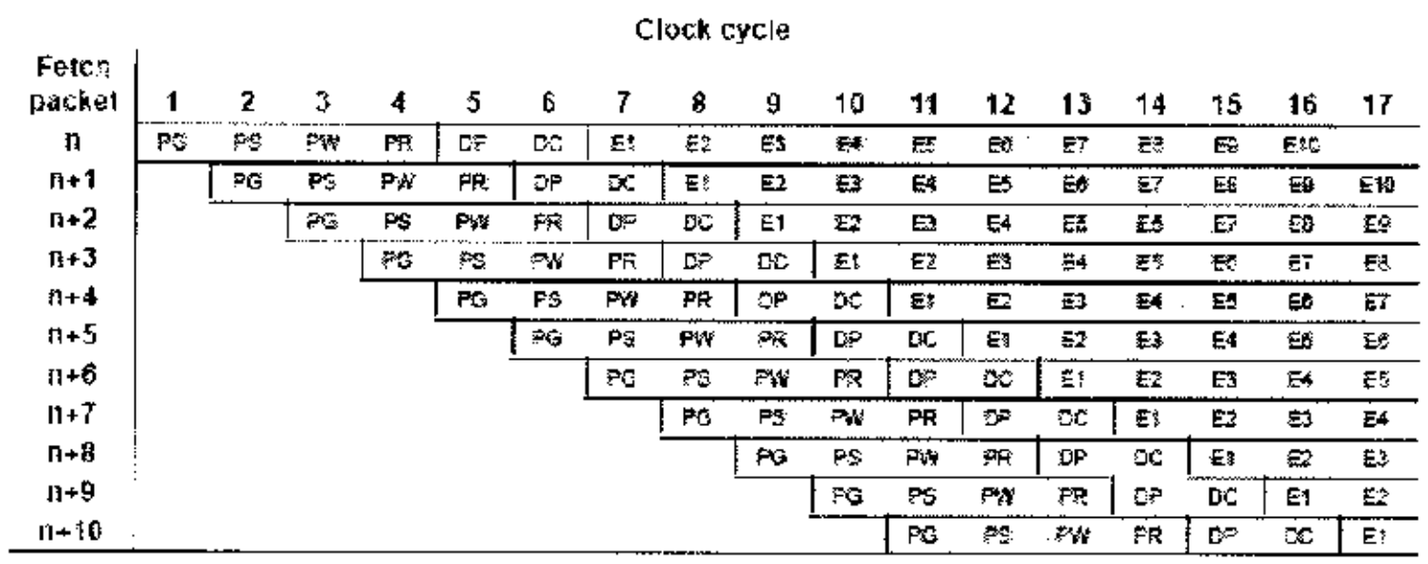


图 4.5：C6711 流水线流程图

C6711 采用 RISC 的指令系统，所有的指令都是 32 位的，除了乘法、内存读写和跳转等少数指令以外，其他的 DSP 指令都可以在单周期内完成。所有的指令都是由 CPU 的功能单元来完成的，不同的功能单元负责执行不同的指令。大多数指令都是以通用寄存器作为操作数，所有指令都可以条件执行，通用寄存器中的 A1、A2、B0、B1、B2 这 5 个寄存器可以被用作条件寄存器。同时 C6711 的指令系统采用了超长指令字（VLIW）的超标量技术，即每 8 条指令组成一个 256 位的指令字，如果 8 条指令完全并行，那么一个指令字就能在单周期内被执行。这样的指令系统完全是由 C6711 的 CPU 结构所决定的，正是因为 CPU 具有大量的整齐的通用寄存器和 8 个独立的功能单元，才能保证一个指令字中的 8 条并行指令在一个周期内能够被分配到各自的功能单元并且从寄存器中获得操作数。

图 4.5 是理想的流水线时序，可是实际中往往这样理想的流水线是难以达到的，流水线会因为各种原因被阻塞，其中最常发生的一种原因就是一个取指包中含有多个执行包。例如，

如果图 4.5 中的 FP_n 取指包中含有三个执行包，那么在第 8 周期流水线就会被阻塞，因为 FP_n 的第 2 个执行包需要在这个周期被执行，而 FP_{n+1} 只能停留在 DC 节拍，其他随后的指令包都会被阻塞，第 9 周期，FP_n 的第 3 个执行包被送入执行单元，流水线依然阻塞，直到第 10 周期 FP_{n+1} 的执行包被送入执行单元，流水线才恢复。所以对 C6711 程序而言，做到每个取指包都只包含一个执行包，即 8 条指令并行是十分重要的，但是理想的 8 条指令并行通常非常不容易做到，因为 CPU 中的 8 个功能单元具有不同的特点，而一些特定的指令也只能在特定的功能单元中运行，比如.M 单元只能执行乘法指令，而乘法指令也只能在.M 单元中被执行。当一段代码中没有乘法运算时，.M 单元就不会被用到，而这段代码就不可能达到 8 条指令并行的理想状态了，所以对于 C6711 而言，乘法越少并不是越好的，有时候还需要将一些没有乘法的算法进行修改，增加乘法操作，以尽可能的保证所有的功能单元能够被用及，使每个取指包只包含一个执行包，减少流水被阻塞的可能。除了取指包的原因以外，流水线还有可能被其他一些原因阻塞，其中最常见的是数据读写的时候，没有命中缓存，正如在上节所分析的，当缓存没有被命中的时候，CPU 阻塞，自然流水线也被阻塞，直到从外部的存储器读取数据到缓存后，流水线才会恢复。

流水线结构是 C6711 能够进行高速计算的关键原因，对所有程序的优化和改进的说到底就是为了保证流水线不被阻塞或者是尽量少的被阻塞。其中修改代码，充分发挥 C6711 指令系统并行的特点，是保证流水线顺畅进行的主要因素之一，也是程序优化最直接的目的之一。

4.2 编码程序的优化

在第三章中，我们已经确定了在典型码率下监控视频的编码方案，并且针对监控视频的特点对编码算法进行了优化。在这节中，我们根据上面所分析的 C6711 的硬件特点，对编码程序进行优化，优化的目的是使 DSP 上的编码程序能够达到每秒钟 15 帧左右的 CIF 的编码速度。所有程序的优化都在 TI 的 C6711 DSK 上完成，C6711 DSK 主频 150MHZ，片外 4MB SDRAM，视频序列采用 foreman。

4.2.1 编码程序结构的优化

原有的视频编码程序是基于 PC 开发的测试程序，程序的结构并不适合 DSP 嵌入式系

统，主要原因是 PC 上的程序往往只重视功能的实现，程序代码结构和数据的安排只是从实现的难易上考虑，而 DSP 嵌入式程序受到 DSP 硬件资源的限制，对数据组织和程序流程需要从硬件资源和代码运行效率上做详细的考虑。下面我们将分析编码程序的流程，优化程序的代码结构和数据结构，减少不必要代码的运行，合并某些函数，以充分利用 C6711 并行指令系统。根据程序数据的大小和生命周期，合理安排其在存储器中的位置，减少片内外存储器数据交换的频率，避免频繁地分配和释放内存。

视频编码程序可以分成 INTRA 编码和 INTER 编码，图 4.6 是原程序的 INTRA 编码流程。

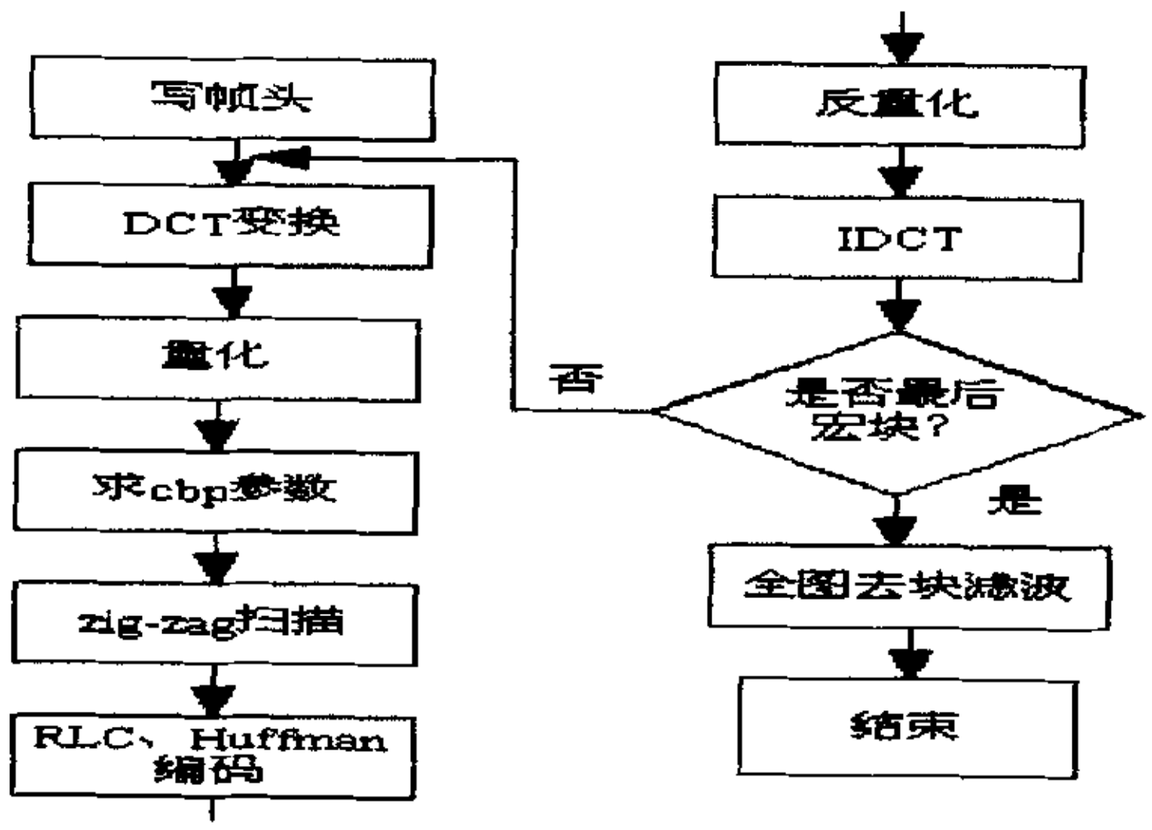


图 4.6：原程序 INTRA 编码流程

图中的每个部分都是单独的函数，可以看到其中有许多不适应 DSP 系统的结构。比如象 zig-zag、求 cbp 参数这样简单操作也单独作为一个函数，根据上面所分析的 C6711 中央处理器特点，就不能充分发挥出 C6711 的 8 个并行功能单元的作用；同时原有的程序还产生了许多不必要的储存空间，比如量化函数会生成新的系数用来存储量化后系数，显然这是对存储空间的浪费，因为量化前和量化后的系数完全可以放在同一个存储空间中。为此，我们将对原有的程序结构进行修改，调整程序运行流程，合并部分函数，保证程序结构能符合 C6711 的特点，尽量节省存储空间。INTRA 的整个编码过程基本可以分成三个部分：

1、DCT 变换、量化、求 cbp 参数。这个部分的输入是 384 字节大小的一个宏块，经过 DCT 和量化之后产生 384*2 字节大小的系数，然后根据量化后系数求 cbp 参数。其中求 cbp 参数是一个非常简单的函数，只要判断量化后系数是否全为零即可，如果只把它单独作为一个函数，那对于 C6711 的 8 个功能单元显然是大材小用了，所以我们将求 cbp 的过程并入

量化函数之中，即在量化数据输出之前进行非零判断，最后返回 cbp。通过这样的调整，可以少一次内存数据的读写，而做非零判断几乎不需要增加额外的指令，也不会影响量化函数的效率。再求出 cbp 后，判断 cbp 是否为零，对于 cbp 参数为零的块，INTRA 只需要编码它的 DC 系数即可，就不必再进行第 2 部分的操作了。

2、 zig-zag、VLC 编码和 Huffman 查表编码。在原程序中经过量化后的系数通过 zig-zag 扫描重新排列，产生新的系数 zcoeff（之前的系数需要进行参考帧重建），对重新排列后的数据进行 VLC 编码和 Huffman 编码。这个部分中，zig-zag 也是一个很简单的函数，要做的只是改变数据的存放顺序。后面的 VLC 和 Huffman 编码则是一个完全串行的过程，如果将它们放在一起，会大大的浪费 C6711 的资源。所以我们把 zig-zag 和 VLC 结合起来，在进行 zig-zag 扫描同时直接进行 run、level 的统计，这样做可以增加 DSP 功能单元的利用率，充分发挥出 C6711 的并行计算能力，也不需要把 zcoeff 系数进行输出。但是存储 run 和 level 需要额外的空间，在最差的情况下存储 level 需要 384*2 字节，存储 run 需要 384 字节，可是为了程序的安全性，我们不得不考虑这个最糟糕的情况，这样比起没有修改前多占用了 384 字节存储空间。不过一个好消息是 INTER 编码会需要更多的存储空间，我们可以在这些空间中挖掘出这 384 个字节，并不会造成存储空间的浪费。而对于统计出来的 run、level 则可以直接送入 Huffman 编码器，提高了编码的效率。

3、 反量化、反 DCT 变换、去块滤波，重建参考帧。由于量化的原因，造成块与块之间出现人为的边界，也就是经常所说的块效应。为了消除块效应，在上面的编码选项分析中，在参考帧重建部分添加了去块滤波器，对每个 8x8 块的边界进行滤波处理。可以从图 4.6 中看到原程序是在整个参考帧建立完毕后，再对全图进行去块滤波。由于程序是以宏块为单位进行编码的，每个宏块重建后的数据会存放到片外的存储器中，所以如果最后再对全图进行去块滤波，程序需要对片外数据进行大量的读写，增加了总线负担。为了减少对片外数据的读写，我们把去块滤波放在每个宏块编码之后。这时，亮度宏块中 4 个 8x8 块的 2 条内部边界数据还在片内的，上一宏块的右边界可以预先保存在片内，只有上边界 16*2 字节大小的数据是在片外了，可以减少对片外数据的操作，节省数据总线的操作时间。同时由于是编码的最后阶段，许多数据的生命周期也已经基本结束，片内可利用的空间很多，也不会增加片内存储器的负担。只需要在片内多留出 32 字节大小空间存储宏块右边界，用作下个宏块的去块滤波。

图 4.7 就是经过调整后的 INTRA 编码流程，比起图 4.6 序结构更紧凑，函数效率更高，符合 C6711DSP 的硬件环境。

INTER 编码采用了帧间预测技术，比起 INTRA 多了运动预测部分，运动预测部分是 INTER 编码程序的关键。一个有效的运动预测程序结构能够大大的提高 INTER 的编码速度以及编码的效果。图 4.8 是原 INTER 程序的结构。

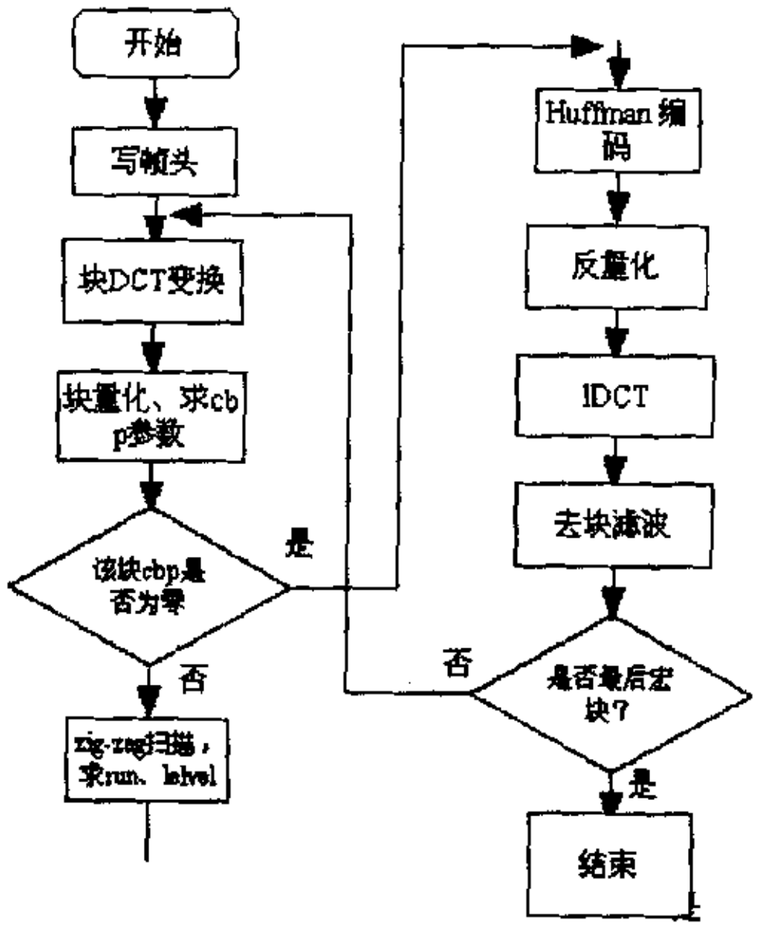


图 4.7：调整后 I 帧编码流程

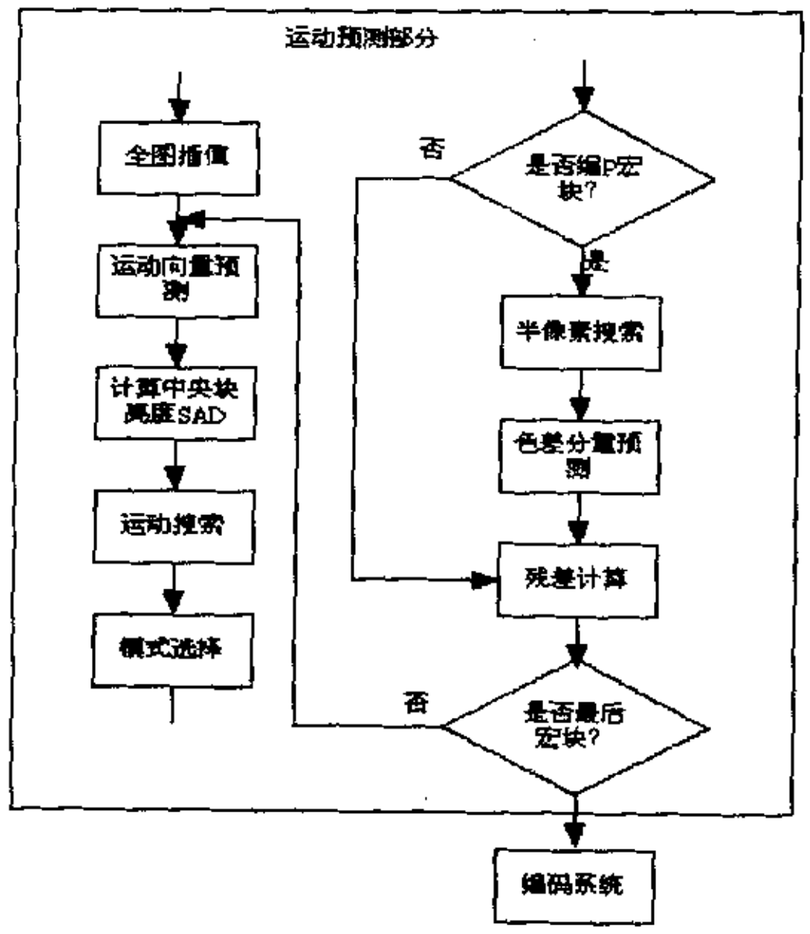


图 4.8：原程序 P 帧编码流程

如图 4.8 所示，原程序中 INTER 的运动预测部分与编码部分是完全分离的，先对所有宏块做运动预测，存储所有的残差和运动向量，再进行编码。另外，原程序先将参考帧进行全图的半像素插值，产生一幅半像素精度的全图，用于半像素搜索。为了存储这个半像素精度的图，程序需要一块原图 4 倍大小的存储空间。这都对 C6711 DSP 系统造成了很大的负担。所以对 INTER 程序结构的调整主要是为了获取更高效的内存组织空间。首先，我们把宏块运动预测和编码部分合在一起。一个宏块完成运动预测后，直接进行编码，如此只需要存储一个宏块的运动向量和残差信息就可以了。然后，运动搜索也不需要在全图上进行，我们的编码方案运动搜索的范围为 [-16,15.5]，一个宏块的整像素搜索只需要 48*48 字节大小的空间即可，而半像素搜索的空间也只占 34*34 字节大小。所以我们调整程序，不进行全图的半像素插值，直接先从片外参考帧载入 48*48 字节大小的搜索参考块，用作整像素搜索。同时因为在整像素搜索完毕后，这 48*48 字节空间的生命周期也就结束，就可以用作下面的半像素插值以及后面的编码系数比如 INTRA 中的 run 系数的存储。经过 INTER 编码运动预测部分程序的调整，程序只需要存储一个宏块信息的内存空间大小即可，避免了全图半像素插值而引起的大量存储空间的浪费。而对于预测后残差数据编码和参考帧重建部分，INTER 和 INTRA 流程基本相同，可以采用上述的 INTRA 编码结构，可以用图 4.7 的结构。

经过调整后的程序不但结构上更紧凑，而且数据安排上更经济，表 4.1 就是调整后程序数据的安排。可以看到基本所有的数据都放入了片内，整个程序运行期间片内只需要 3726 字节大小的空间。而片外的 SDRAM 用来存放原始图像和参考帧，片外和片内的数据交换可以通过 C6711 的 EDMA 通道完成，不需要占用 DSP 的计算周期。

数据结构	作用	大小（bytes）	位置
Raw_img	当前编码帧	152064	片外 SDRAM
Pre_rec_img	参考帧	152064	片外 SDRAM
Rec_img	当前重建帧	152064	片外 SDRAM
MB	原宏块、重建宏块	384	片内 RAM
coeff	DCT 变换后系数、量化后系数、反量化后系数	768	片内 RAM
level	块 level 系数	128	片内 RAM
Search_area	整像素搜索、半像素插值、半像素搜索、残差数据、参考宏块、块 run 系数	2304	片内 RAM
Back_bitbuf	标志是否背景宏块	99	片内 RAM
Left_buf	deblock 前一宏块的右边界	48	片内 RAM

表 4.1：编码程序数据内存安排

4.2.2 编码程序的代码优化

在 TI 的编译开发工具 CCS 中为程序开发人员提供了多种优化选项，通过这些优化选项的配合，能够对 C 代码进行极大的优化[31]。但是由于原有编码程序在各个函数代码内部结构和数据安排上都有许多不合理的地方，如果不对代码进行任何手工的优化，光靠编译器的优化，编码程序每秒只能编 1 帧左右的 CIF，显然这样的编码速度与我们的目标编码速度相差甚远。为了使得程序中的代码有最优的汇编结果，能够充分发挥出 C6711 强大的中央处理器功能，接下来我们对编码程序的主要函数进行汇编优化。

根据前面所分析 C6711 中央处理器的结构，我们知道如果一段代码经过编译后的汇编结果没有任何的并行，那么 C6711 就只是一个主频 150MHZ 的处理器。要实现 C6711 1200MIPS 强大计算功能，使代码达到最快的运行速度，就要充分发挥出 C6711 中央处理器并行处理的强大功能，使得代码编译后的汇编结果尽量达到理想的 8 条指令并行，所以修改现有的代码，提高代码的并行性，是优化程序代码的目的之一。同时，C6711 提供了一些数据包处理指令，比如 ADD2，就可以一个周期内完成 2 个 16 位的加法，代码优化的另一个目的就是合理安排数据计算结果，充分发挥这些特殊指令，提高代码效率。在上面所提及的

两个因素中，提高代码并行性是我们进行代码优化的主要目的。

为了使代码有理想的并行性，对多数代码的优化，特别是循环代码的优化，多采用软件流水技术。所谓的软件流水是一种用于安排循环内的指令运行方式的技术，它的目的是使得循环的多次迭代能够并行的执行。软件流水的实现是基于循环中的代码有多个独立步骤，和C6711DSP 硬件流水的原理一样，利用 C6711 中央处理器的 8 个独立功能单元和 32 个通用寄存器这些大规模的硬件资源，使得不同迭代的不同步骤在同一周期内并行的执行[32]。如图 4.9 所示，如果一次迭代中需要完成从 A 到 E 5 个独立的步骤，通过合理的安排指令和数据操作，理想情况下能够实现 E1、D2、C3、B4、A5 这样的 5 次不同迭代的不同步骤在同一个周期内执行。通过软件流水的优化，可以大大提供循环代码的效率，极大的实现指令的并行性。由于 PC 中开发的 c 代码不会考虑软件流水的情况，通常会有一些破坏软件流水优化的因素存在，主要有以下这些因素：

1、多重循环的嵌套。由于软件流水的优化能针对一个循环，所以如果出现了多重循环的嵌套，只能优化最里面的循环，而对于外循环而言，则很难进行优化。为了解决这个问题，我们通常把具有多重循环的代码进行循环展开。循环展开后会增加代码的长度，但是有利于代码的优化。比如在编码程序的 SAD 计算函数中的这么一段代码：

```
for(i=0;i<16;i++)
    for(j=0;j<16;j++)
        sad+=abs(mb1[i][j]- mb2[i][j]);
```

很简单的一个函数，双重循环，循环体中的操作十分简单，主要是加减和绝对值。我们知道流水技术的本质就是利用 C6711 中央处理器丰富的硬件资源，使多个独立的操作在同一个周期内完成。所以如果只对内循环体进行优化，由于内循环中操作过于简单，不能充分利用中央处理器的 8 个计算单元，而且外循环因此不能进行优化，所以我们将内循环展开，修改代码如下：

```
for(i=0;i<16;i++)
    sad+= abs(mb1[i][0]- mb2[i][0])+ abs(mb1[i][1]- mb2[i][1])+ abs(mb1[i][2]-
mb2[i][2])+ .....+ abs(mb1[i][14]- mb2[i][14])+ abs(mb1[i][15]- mb2[i][15]);
```

虽然循环展开后增加了代码的大小，但是有利于进行流水优化，提高代码运行速度。

2、循环中包含了函数的调用。如果出现了函数的调用，那么循环中出现了跳转指令，不能进行软件流水。值得注意的是这里所说的函数，不光光指那些显式的用函数名进行调用的函数，还包括 C6711 中央处理器的功能单元没有提供的所有操作，比如除法、取模等计

算符号。由于没有功能单元支持这些操作，也需要通过调用函数的方法进行的。对于比较简单的函数，可以将函数展开，变成循环内的指令操作。如果是除法、取模这样的算术函数，则可以根据具体函数中的应用进行修改、优化，由功能单元所提供的指令进行近似操作。比如编码程序的量化函数中，量化操作的代码如下：

```
qceoff[i]=(ceoff[i]+quant/2)/quant
```

其中有两个除法运算，前一个是除以 2，后一个是除以量化系数。对于以 2 为指数的除法，我们可以用简单的算术右移操作来代替。而对于 quant 变量的除法不能简单用移位进行优化，不过因为 quant 是一个 5 位整数，我们可以利用整数除法计算精度有限的特点，做如下修改：

```
qcoeff[i]=((coeff[i]+quant>>1)*quant_table[quant])>>quant_shift
```

通过乘一个系数，然后再移位的方法实现对 quant 除法函数的修改，满足计算精度。通过以上的代码优化，除法函数可以用 C6711 功能单元的指令代替，为一下步的流水优化做好的准备。

3、循环内有复杂的条件判断代码。由于 C6711 中可用于条件判断的通用寄存器只有 5 个，如果有过于复杂的条件判断代码，使得这 5 个寄存器不够使用，那么软件流水也不能进行。一般来说我们不在循环内加过于复杂的条件判断，把条件判断尽量移到循环体的外部，有时候宁可增加代码的大小，也要让条件判断在循环的外部，保证流水优化可以顺利进行。比如编码程序中的 Huffman 编码函数的循环中的一个条件判断：

```
if(pic.adv_intra==1||mb_type==mb_inter)
    ……
else
  if(i==0)
        ……
  else
    ……
```

由于判断条件过于复杂，不能对循环展开流水，为此我们把条件判断放到循环体外，对不同的条件编写不同的循环代码，把不同条件的代码单独进行流水优化。

4、代码尺寸太大。由于硬件资源的限制，如果循环体内的代码太多或者太复杂，使得 C6711 的 32 个通用寄存器不够用，影响到流水的展开。遇到这样的情况，如果代码可以简化就简化，如果不能简化，尝试把代码分割成几个相对独立的小代码，然后分步完成小代码，

从而便于流水的优化。

5、 有条件终止循环或者提前退出循环。由于任何的流水优化都需要进行填充流水和排空流水这两个操作（见图 4.9），特别是排空流水，对循环结束后的环境有明确的要求，如果循环提前退出，那么就会造成流水排空出错，从而使得整个代码的运行结果出错。

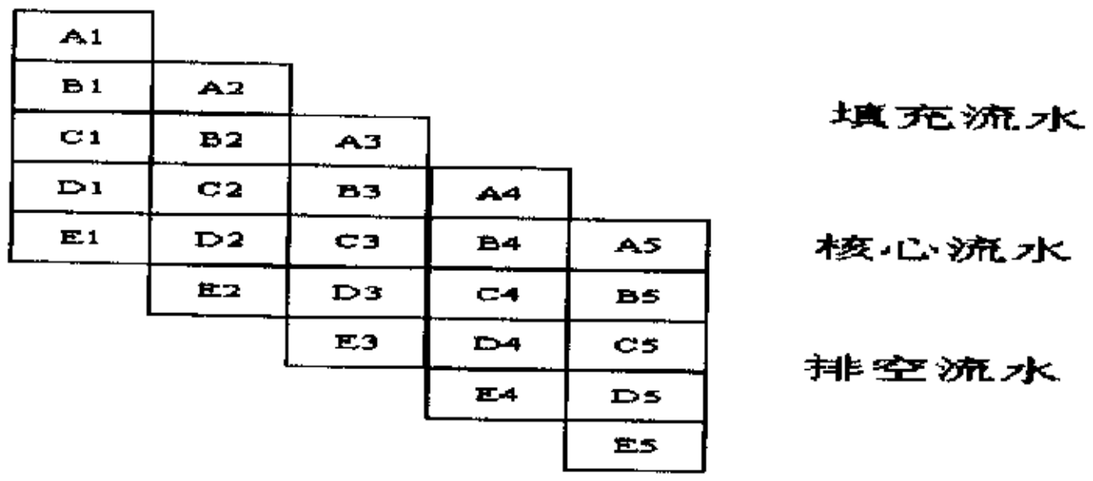


图 4.9：软件流水技术

在消除了代码中破坏流水优化的因素后，我们再来看一下如何对一段代码进行合理的流水优化。正确的分析代码结构，明确代码中的指令操作、数据存取和 C6711 中央处理器各个功能单元的关系，从而确定循环的最小迭代周期是进行流水优化的关键。下面以一个简单的点乘函数为例子，说明进行流水优化的过程和技巧。

```
int dotm(short * a, short *b)
{
   int sum1=0， sum2=0,sum;
   int I;
   for(i=0;i<100;i++)
    {
     sum1+=a[i]*b[i];
     sum2+=a[i+1]*b[i+1];
     }
     sum=sum1+sum2;
        return sum;
}
```

上面的点乘代码已经经过初步优化，在循环体内采用两个乘法的原因是利用 C6711 的 32 位通用寄存器和 MPYH 的 16 位乘法指令，在同一周期内处理两个 16 位的数据。我们首先编写对应的汇编伪代码，，以分析循环体内的操作需要用到哪些功能单元、哪些指令，以

及这些指令运行的周期状况。

```
        BEGIN
      initial
LOOP:
        LDW   .D1     *a_ptr++, a_var
        LDW   .D2     *b_ptr++, b_var
        MPY   .M1     a_var, b_var, m_var1
        MPYH  .M2     a_var, b_var, m_var2
        ADD   .L1     m_var1, sum1, sum1
        ADD   .L2     m_var2, sum2, sum2
        [cnt] .S1  SUB   cnt, 1, cnt
        [cnt] .S2  B   LOOP

        END
```

功能单元/运行周期	0、8…	1、9…	2、10…	3、11…	4、12…	5、13…	6、14…	7、15…
.D1	LDW							
.D2	LDW							
.M1						MPY		
.M2						MPYH		
.L1								ADD
.L2								ADD
.S1		SUB						
		B						

表 4.2：初步迭代间隔表

根据上面的伪汇编代码，可以初步绘制迭代间隔编排表 4.2。这个表给出了软件流水循环的执行情况和按周期跟踪所使用的资源，确保资源在任何给定的周期内不会使用两次。一个循环的迭代间隔就是这个循环代码两次迭代开始之间的周期，显而易见迭代间隔越小，执行一个循环所用的周期越小，代码效率越高。

下一步在表 4.2 的基础上确定最小的迭代间隔。循环中的最小迭代间隔是由使用最多的资源数和数据相关性所决定，例如如果在一个迭代中有 4 条指令都用到了.S1 单元，那么最

小迭代周期至少是 4 了，因为使用同一资源的 4 条指令不能并行执行。从表 4.2 可以看到这个点乘函数没有两条指令使用同一个资源，基于资源相关性的最小迭代间隔是 1，同时这个函数也没有数据相关性影响到最小迭代间隔，所以我们可以确定最小迭代间隔为 1，也就是表示每隔一个周期就可以开始一个新的迭代，在每个周期可编排 LDW 和 MPY 指令。表 4.3 是调整后的完全流水迭代间隔编排表。表中最后一列是包含全部循环指令的一个单周期循环，0-6 周期为建立循环过程，即填充流水。表中的*号表示了在每个周期正在执行的指令在循环迭代次数上的差，比如最后一列表示：

- ADD 指令正在执行 n 次迭代
- MPY 和 MPYH 指令正在执行 n+2 次迭代
- B 指令正在执行 n+5 次迭代
- SUB 指令正在执行 n+6 次迭代
- LDW 指令正在执行 n+7 次迭代

在这种情况下，这个循环的多次迭代在软件流水中并行执行，软件流水是从 n 到 n+7 次迭代并行执行的 8 个迭代深度。

功能单元/运行周期	0	1	2	3	4	5	6	7、8、9...
.D1	LDW	LDW *	LDW **	LDW ***	LDW ****	LDW *****	LDW ******	LDW *******
.D2	LDW	LDW *	LDW **	LDW ***	LDW ****	LDW *****	LDW ******	LDW *******
.M1						MPY	MPY *	MPY **
.M2						MPYH	MPYH *	MPYH **
.L1								ADD
.L2								ADD
.S1		SUB *	SUB *	SUB **	SUB ***	SUB ****	SUB *****	SUB ******
			B *	B **	B ***	B ****	B *****	B ******

表 4.3：完全流水迭代间隔编排表

点乘函数的流水是最理想的软件流水情况，在实际函数通常要比点乘函数复杂的多，多数没有这样理想的流水情况，所以在做优化的时候我们不能拘泥于上述软件流水的模式，要根据各个函数的具体情况，做出不同的优化处理。只要我们明确优化的最终目的是使得优化

后的汇编代码有最多的并行，充分发挥出 C6711 中央处理器的并行处理功能就可以了。

我们对 DSP 编码程序代码优化原则是：利用 C 代码逻辑表达性强的优点，整体框架采用 C 代码编写；对于独立的函数操作，C 代码编写效率不高，进行汇编优化。这样的优化方案使得程序结构清晰，模块明确，便于程序的维护和升级。表 4.4 是几个主要函数优化前后的效率情况，表中我们用中央处理器的运行周期来衡量代码效率。可以看到大多数代码经过汇编优化后都有 3-8 倍左右的效率提高。有些代码优化后的大小比优化前大，比如 SCAN、DEQUANT 等，这是因为在优化时对循环做了展开处理，而有些代码经过优化效率提升不明显，比如 SCAN、MB_edge_filter 等，SCAN 函数是因为 C 代码操作过于简单，所以我们把 run、level 统计也加入到 SCAN 的汇编代码中，增加了操作，加快了后面 HUFFMAN 编码的速度，其实比起 C 代码的效率提高了很多。MB_edge_filter 是去块滤波的边缘滤波函数，这个函数内除了有利于 DSP 优化的一个 4 抽头的 FIR 滤波器以外，还有许多比较判断操作，正如上面所分析的过多的条件判断限制了函数的优化潜力，所以经过汇编优化后的效率并没有提高很多，而代码大小反而增大了。总体来说，经过对编码程序各个函数的汇编优化，还是较大的提高了编码程序的整体效率。

函数名	C 代码效率 (cycles)	C 代码大小 （bytes）	汇编代码效率 (cycles)_	汇编代码大小 (bytes)
DCT	1677	3264	310	1216
QUANT	2481	2560	368	1088
SCAN	448	224	255	896
DEQUANT	317	384	240	704
IDCT	3492	2560	653	1088
8*8 inter	1017	2400	233	480
16*16 SAD	460	1120	121	736
MB_edge_filter	824	800	513	1328

表 4.4： 编码程序主要函数优化表

注：代码效率栏中的单位是该函数运行一次所需要的 DSP 周期数。C 代码效率是指用 O3 编译优化选项进行优化后的代码效率，汇编代码效率是指经过我们手工优化的代码效率。

4.2.3 编码程序的总线优化

在上面的两节中，我们对程序的结构和代码分别进行了优化，程序的速度获得了相当大的提高。但是我们发现程序速度的提升，特别是代码优化的速度提升并没有达到所预想的理论效果。经过多次的调试和分析，我们发现编码程序中有频繁的数据搬运和读写，是低速的总线读写大大影响了整个编码程序的整体效率。C6711 总线可以分为片外总线和片内总

线。对于片外总线操作而言，由于 C6711 片内有限存储空间和大量视频数据之间的冲突，我们把视频原始数据和参考数据放在片外的 SDRAM 中，C6711 在读写这些数据时，受到外部数据总线速率的限制，造成中央处理器经常处于等待数据的状态，浪费了大量的时间。对于片内的数据和程序总线操作，不合理的数据安排，造成频繁的 CACHE MISS，使得中央处理器读写数据和程序的效率不高。在这一节中，我们将根据编码程序的数据结构和部分代码结构，针对片外的数据总线和片内的数据、程序总线进行优化，提升 C6711 编码程序的整体效率。

如上面的图 4.4 所示，如果中央处理器直接从外部存储空间读写数据，由于外部的 SDRAM 和 SRAM 的时钟周期以及位宽效率都不如 C6711 中央处理器的效率，所以中央处理器需要等待慢速的外部总线处理完毕以后，才能继续进行下一个指令的处理。高速的 DSP 受制于低速的外部数据总线，影响了编码程序的效率。为了解决这个问题，我们需要避免中央处理器直接读写外部数据，采用 DMA 方式将外部数据先搬移到片内的 RAM 中，然后中央处理器再进行操作。这样的安排使得中央处理器不需要理会外部数据的读写，可以集中精力进行数据的处理。

C6711 的 DMA 方式是一种重要的数据访问方式，它可以在没有中央处理器参与的情况下，由 EDMA 控制器完成对 DSP 存储空间内的数据搬移，数据搬移的源地址和目的地址可以是片内和片外的任何可映射空间地址。C6711 的 EDMA 控制器有以下主要特点[33]：

1、 后台操作：EDMA 控制器可以独立于中央处理器工作。

2、 高吞吐率：可以按中央处理器时钟的速度进行数据吞吐。

3、 多通道：C6711 的 EDMA 控制器可以控制 16 个独立通道的数据传输，同时通道间的优先级可设。

4、 强大、灵活的数据传输设置：通过配置 EDMA 控制器的参数，可以实现不同数据结构的数据传输，比如一维到二维、多帧传输等等。

5、 事件同步和中断反馈：数据的传输可以由指令事件触发同步，传输完毕也可以想中央处理器发送中断。

用 EDMA 控制器基本可以实现编码程序中所有的数据搬运。比如宏块编码的开始，EDMA 控制器可以把存放在片外 SDRAM 中原始宏块数据以块为单位搬运到片内 RAM 中。片外的 8x8 块是以二维结构存放的，而在片内 RAM 中为了方便程序的操作，需要把块的二维数据变成一维连续的数据格式，这只需要对 EDMA 控制寄存器进行设置，调整数据搬运方式为从二维到一维即可，如图 4.10。EDMA 控制器就可以自动实现这样的数据搬运。除

了对于原始宏块数据的搬运，还需要将重建后的宏块数据从片内搬运到片外，以供 INTER 的运动搜索以及预测。表 4.5 就是程序中需要搬运的数据以及相关情况。

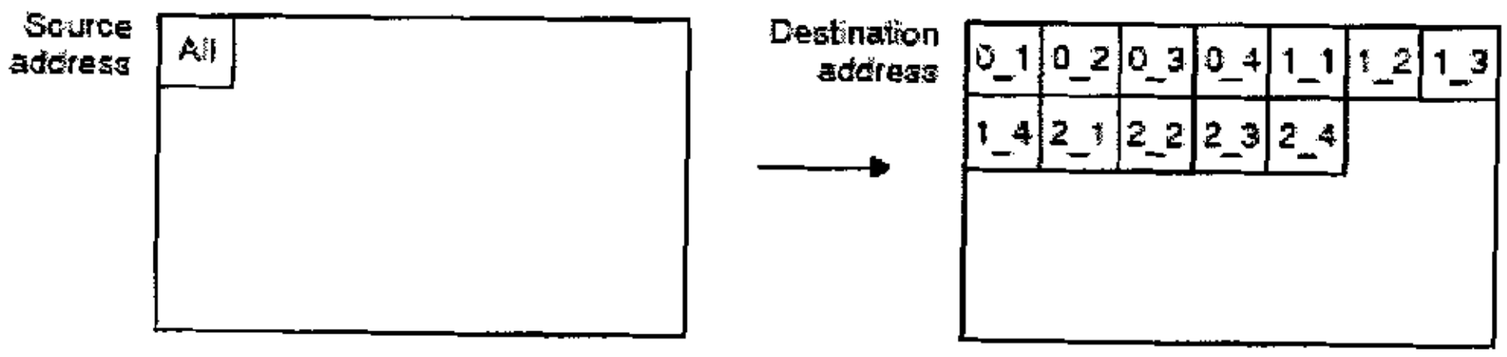


图 4.10 二维 to 一维的 EDMA 搬运

数据	大小（bytes）	源地址	目的地址	传输类型
原始块	64	片外 SDRAM	片内 RAM	二维 to 一维
重建参考块	64	片内 RAM	片外 SDRAM	一维 to 二维
亮度搜索范围	48*48=2304	片外 SDRAM	片内 RAM	二维 to 一维
预测参考块	64	片外 SDRAM	片内 RAM	二维 to 一维

表 4.5：EDMA 数据搬运

通过对 EDMA 控制器的使用，基本解决了片外总线的效率问题，而对于片内总线优化的关键在于对 C6711 缓存结构的理解，并针对其缓存特点调整程序以及数据的存储结构。根据 4.1.2 节的分析，C6711DSP 采用的是 2 级高速缓存结构。中央处理器只从一级缓存读写数据和加载指令，所以能否使得一级缓存总是被命中（CACHE HIT）是影响片内总线效率的主要因素。如果缓存命中，意味着在单周期内可以完成总线操作，不会影响中央处理器效率；如果没有被命中，造成 CACHE MISS，则中央处理器停止操作，以等待缓存更新的完成。所以如果能保证每次运行时程序的代码和数据都已经在一级缓存中，读写效率是最高的，反之，如果每次运行时都产生了 CACHE MISS，必然造成片内总线效率的低下，影响到整个程序的运行效率。

C6711 片内总线分为程序总线和数据总线，分别对应 C6711 片内的一级程序缓存 L1P 和一级数据缓存 L1D。两个缓存的大小都是 4k 字节。程序缓存采用直接映射结构（direct map cache），L1P 的行大小为 64B，即 2 个取指包的宽度，可以缓存 64 组。对于程序缓存的优化可以从以下两个方面着手[30]：

1、避免循环体内的代码大小超过 4K 字节，因为小于 4K 字节的代码在第一次循环中可

以全部被载入到 LIP 中，以后就可以一直被命中，提高了程序总线的效率。如果一个循环体内的代码大小大于 4K 字节，则可以将一个大循环体拆开成几个小循环体。如：

```
  for (i=0; i<N; i++)
{
  function_1(in[i], tmp);
  function_2(tmp, out[i]);
}
```

可以拆开成

```
  for (i=0; i<N; i++)
{
    function_1(in[i], tmp[i]);
}
  for (i=0; i<N; i++)
{
function_2(tmp[i], out[i]);
}
```

2、对于循环体内的多个函数代码，保证这些代码在内存中连续存放，或者是存放在对应不同缓存组的内存中，避免 L1P 冲突。L1P 是单路缓存，所以如果循环体中的不同函数代码对应的缓存组相同，如图 4.11 中 function_1 和 function_2，就会造成 L1P 冲突，引起缓存频繁更新。

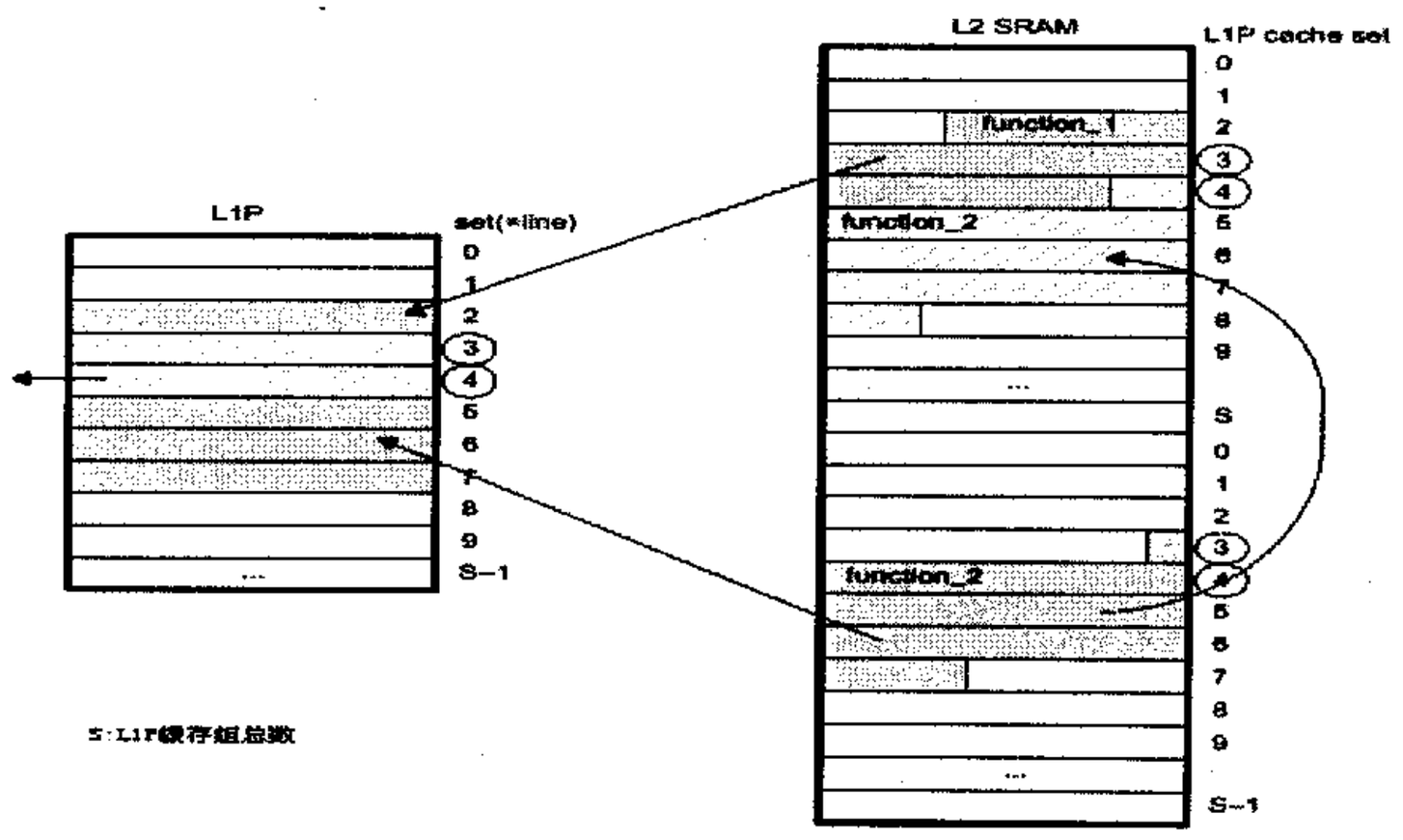


图 4.11: 两个函数的 L1P 冲突

在 DSP 编码程序中，主要循环是以块为单位的变换量化操作，循环体内包括 DCT、量化、ZIG-ZAG 扫描、反量化和 IDCT 这 5 个独立函数，并依次运行。这 5 个函数的代码大小在上面的表 4.4 中，为了保证循环中代码大小总和小于 4k 字节，我们拆分循环，后 4 个函数作为一个循环，而第一个 DCT 函数单独做一个循环。把 DCT 单独放在最前的循环，是为了利用 DCT 循环把下面所需要的数据载入到一级数据缓存中，使接下来的数据操作都可以命中数据缓存。

片内的数据总线对应着一级数据缓存 L1D，采用的是双路联想结构。所谓的双路缓存就是由两个直接映射缓存组成的，每路直接映射缓存的大小为 2KB，这样 L1D 相当于由两个独立的缓存组成，一共 4KB 大小。L1D 的行大小为 32B，每路缓存可以缓存 64 组[2]。L1D这样的设计结构是专门针对DSP数据密集型运算和C6711中央处理器双路结构特点的。下面结合程序的情况，分析针对 L1D 缓存做优化的几个主要措施。

1、尽量减少数据类型的大小，可以用 16 位的数据就不要用 32 位的数据，这样不但节省空间，而且提高 L1D 的效率。因为 L1D 的行大小是固定的 32 字节，如果采用 16 位的数据类型，就可以在一行内比 32 位的数据类型多放一倍的数据，从而减少了以后程序中 cache miss 的情况发生。在上面的程序结构优化时，我们已经将不合理的数据类型进行了调整。

2、尽量构成数据链，如图 4.12 所示，func1 的输出数据就是 func2 的输入数据，那么在 func2 运行时，它的输入数据就已经在 L1D 缓存中了，不需要重新更新 L1D。在编码程序中，数据的使用和程序的结构一样，主要分为三个部分：运动搜索（INTER）、变换量化和编码。运动搜索中数据调用过程主要是对输入数据的分析，输出只是少量的分析结果，所以不容易形成数据链。能够形成数据链的主要变换量化部分，DCT->量化->反量化->IDCT 构成了一条很好的数据链。另外，我们把在原来的结构中量化后的 scan 函数的位置稍做调整，放到 IDCT 后，huffman 编码前，这样 scan 输出的 run、level 数组和 huffman 编码也形成了数据链。

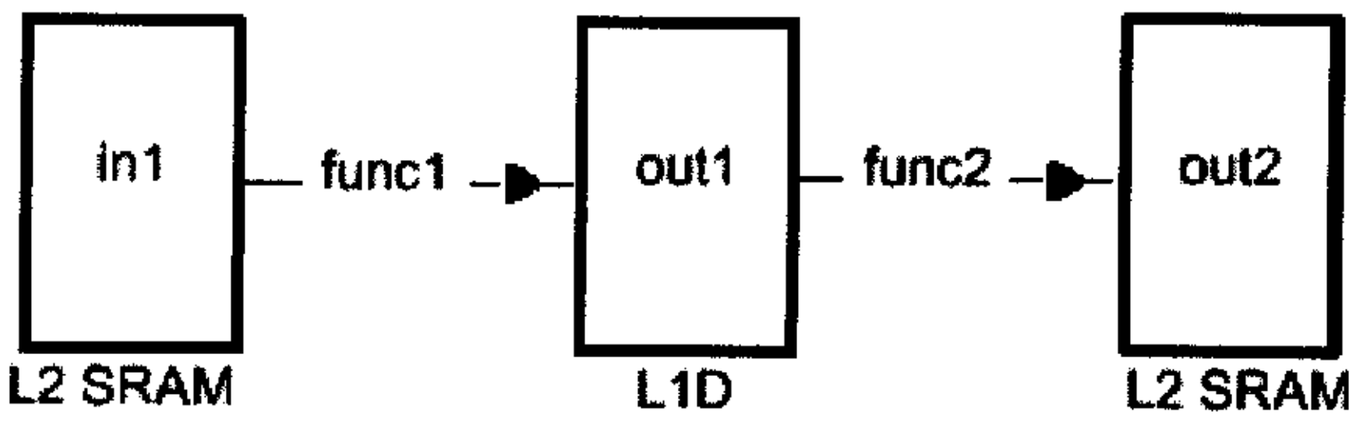


图 4.12： L1D 的数据链结构

3、合理安排数据的存放，避免在读写时发生不必要的数据冲突。具体代码中经常因为数据存放位置不妥，造成 L1D 里相同组的数据冲突。比如在 DCT 代码中，如果输入的数据和输出数据所存放的内存映射到 L1D 缓存的同路、同组，那么在对输出数据进行写操作时，会与在缓存中的输入数据形成冲突。所以在编写程序的 link cmd 文件时，需要对数据的安放做一定的考虑，连续使用的数据尽量安排在 4k 字节对齐的连续地址就可以避免这种情况的发生。

我们针对 C6711 片外和片内总线的不同特点，对片外的数据总线采用 EDMA 方式进行数据搬运，使得中央处理器不必为等待片外数据，耗费大量的时钟周期；同时对片内程序和数据总线，主要分析了 L1P 和 L1D 缓存的特点和程序中相关的情况，对程序代码结构和数据安排进行了调整。

经过对总线的优化，基本完成了对编码程序在 C6711 DSP 上的优化工作，比起未优化前的程序，优化后的时间性能得到了极大的提高，经过对 foreman CIF 格式的视频序列测试，量化系数固定为 10，可以达到 15 帧左右每秒的编码速度，不但可以满足目前编码系统的要求，同时也为以后系统的升级打下了良好的基础。

4.3 DSP 编码监控系统的集成

从第一章的图 1.2 监控系统原理图中，我们可以看到 C6711 DSP 编码系统除了 DSP 以外，还包括了前端的 7113 视频采集模块和后端 UART 异步串口输出模块，另外还有外围的存储器件以及用于系统启动 FLASH 模块，所以如何将 DSP 程序和前后端模块集成以及如何完成系统的 FLASH BOOT 引导，是整个编码监控系统能够独立正常运行的两个重要环节。在这一节中，我们将分别对这两个环节做简单的介绍和分析。

4.3.1 前后端模块的集成

对于 DSP 编码程序而言，其输入图像来自于前端 7113 视频采集模块，而编码后产生的码流通过后端的异步串口模块向外发送。

如图 4.14 所示，视频采集模块主要由 7113 采集芯片、CPLD 和两块 SRAM 构成。7113 采集芯片将摄像头中的模拟信号转换成数字信号，采集原始视频信号。它所采集的原始视频信息中还包含了一些诸如行、场消影信号等无用信号，同时视频格式也不符合 DSP 编码程序的要求，所以在 7113 和 SRAM 之间加入一块 CPLD，用于对 7113 采集的视频信息进行

逻辑处理，去除无用的信息并调整视频格式为编码程序所需要的 4：1：1 的 CIF 或者 QCIF 格式，然后将产生视频数据存储到两块 SRAM 的其中一块。采集模块中采用两块 SRAM，CPLD 和 DSP 同时操作不同的一块 SRAM，构成一个乒乓结构。当视频采集模块完成一帧图像的采集，存储到一块 SRAM 中后，CPLD 会向 DSP 发送一个中断，以通知 DSP 一帧图像已经就续，同时将该块 SRAM 对 DSP 选通。此时，DSP 对采集模块的中断信号进行处理：如果 DSP 已经完成了上一帧图像的编码，就响应中断，将当前 SRAM 中的该帧图像搬移到 SDRAM 中，然后再开始新的编码，这时视频采集模块将继续采集下一帧图像，存储到另一块 SRAM 中；如果 DSP 还没有完成上一帧图像的编码，这个中断被 DSP 所屏蔽，不做响应，直到完成当前图像的编码为止。视频采集模块不受 DSP 状态的影响，一直进行采集工作。

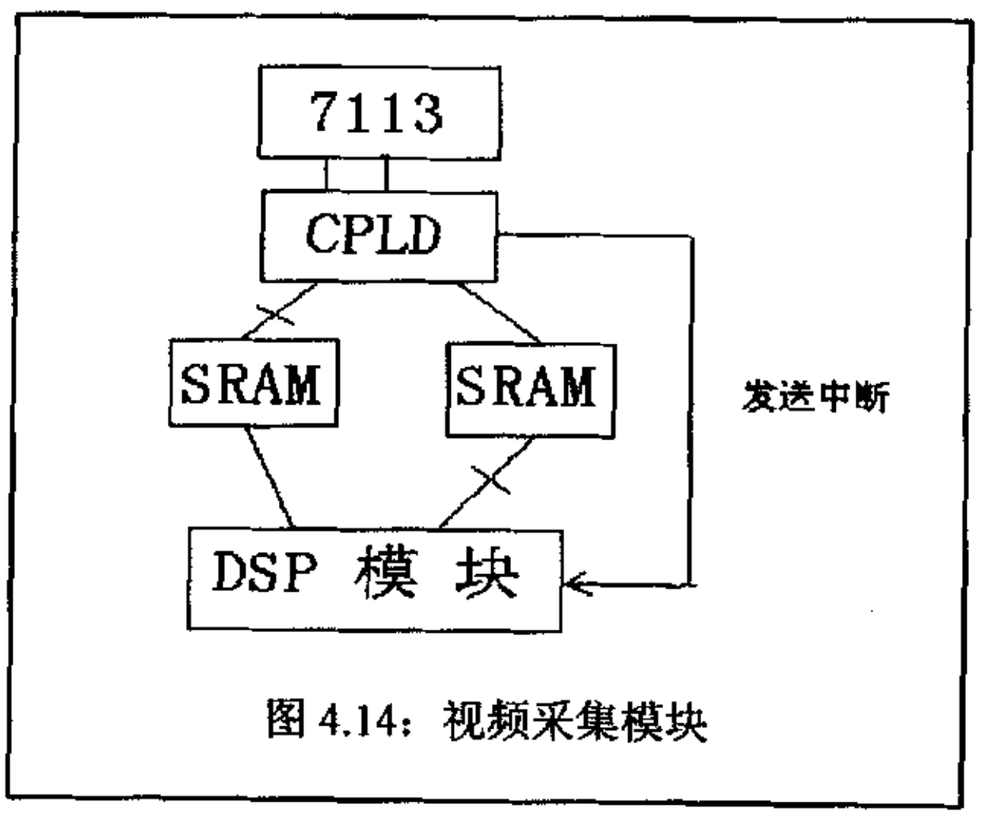


图 4.14：视频采集模块

后端的异步串口输出模块结构很简单，整个模块主要由 UART 串口芯片和一些外围电路组成，连接在 DSP 的 CE3 空间上，所以 UART 芯片的配置寄存器以及串口缓冲都被映射为 DSP 地址。在程序开始时，DSP 需要对串口模块进行初始化配置：通过向 UART 配置寄存器发送初始化配置命令，完成对串口模块的工作模式以及波特率的配置。DSP 向串口模块发送数据时，只需要向串口缓冲地址发送数据，串口模块在侦测到缓冲中有数据的时候，会根据配置的波特率，将缓冲中的数据向外发送。当在缓冲中数据全部被发送完毕后，模块向 DSP 发出一个中断，以通知 DSP 继续发送数据。需要注意的是，串口缓冲一次最多只能容纳 16 字节大小的数据，也就是说 DSP 最多一次只能向串口缓冲发送 16 字节大小的数据。当串口模块接收到数据，也会向 DSP 发出中断，DSP 程序响应该中断，接收缓冲读取数据。

根据串口模块的特点，编码程序一次最多只能发送 16 个字节大小的数据到串口模块，然后需要等待串口模块将这 16 个字节发送完毕后，才能继续 16 个字节的发送。如果编码程序采用边编码边发送码流的方法，因为编码过程中码流产生速率不平衡，会造成有时无码流

可发，有时又有过多的码流阻塞在输出缓冲中的情况。为了解决这个问题，我们把编码器的输出缓冲做成一个乒乓的形式，两块缓冲空间分别用来存储前后帧图像编码产生的码流。在当前帧编码时，发送的是前一帧的码流，而当前帧码流则存放到另一个缓冲中。在一般情况下，编码产生码流的速度通常比串口模块发送的速度快，所以在一帧图像编码完成后，我们需要判断前一帧的码流是否已经发送完成，只有完成后才能打开与前端视频采集模块的中断响应，继续下一帧的编码。

DSP 系统对于前后端模块的集成都是采用中断响应的模式，所以如何编写 DSP 程序的中断向量表、中断处理程序以及对各个中断的屏蔽、挂起状态的控制是整个集成工作的关键。C6711DSP 的中断向量表是放于 0 地址处的大小为 512 字节的程序，其中包括了对 reset 中断、不可屏蔽中断和 12 个可屏蔽中断在内的 14 个中断的服务路径。在我们的系统集成中只需要其中 3 个中断，加上 reset 和不可屏蔽中断，DSP 程序一共需要处理 5 个中断，下面说明这 5 个中断处理程序的内容。

1. reset 和不可屏蔽中断通常是由硬件原因所引起的，一旦这两个中断发送，DSP 程序就会停止任何当前操作，重新启动程序，以恢复初始化系统。

2. 前端视频采集模块的中断是用来通知 DSP 已经完成一帧新图像的采集。处理该中断时，首先屏蔽该中断（等到编码和发送完成后再由 DSP 编码程序打开这个中断），然后采用 EDMA 方式将 SRAM 中的图像搬运到 SDRAM 里，在搬运完毕后通知 DSP 编码程序图像已经准备就绪，开始编码操作。

3. 串口发送模块有两个中断，分别是发送中断和接收中断。发送中断表明串口缓冲已空。这个中断的处理首先判断是否还有码流需要发送，如果没有，则通知编码程序发送完毕；反之，则将下 16 个字节送到串口缓冲。串口接收中断通知 DSP 程序有数据到达串口，一般串口的接收都是用于客户端对 DSP 服务器端的参数调整，比如分辨率、码率、清晰度等等，所以在这个中断响应里首先将接收到的数据读入 DSP，然后进行参数分析，以便在下一帧编码的时候进行参数调整。

通过对各个中断响应程序的设置，基本保证了 DSP 与前后端模块的正常通信和工作。

4.3.2 FLASH BOOT

我们的 DSP 编码系统需要能够脱机独立运行，所以采用了 ROM 加载的引导方式。在 DSP 上电以后，位于外部 CE1 空间的 FLASH 中的代码首先通过 EDMA 方式被搬运入 DSP 的 0 地址处[2]。加载过程在 reset 信号撤消之后开始，此时 C6711 内部恢复复位状态，由 EDMA

控制器将 FLASH 的前 1K 字节的连续数据块传送到 C6711 的 0 地址处，当传送完毕以后，DSP 退出复位状态，开始执行 0 地址出的指令。在这个引导过程中，烧写在 FLASH 中的前 1K 字节代码就是系统 BOOT 程序。所谓 BOOT 程序就是烧写在 FLASH 中，专门用于加载其他 DSP 程序的引导程序，BOOT 程序在完成加载后，对 DSP 以后的程序运行没有任何影响。一般在 BOOT 程序中需要完成以下几个工作：

1、 完成 DSP 的初始化工作，包括对外围存储器的初始化设置，对中断控制器的初始化设置等。

2、 从 FLASH 中加载未完成的 BOOT 代码和数据。

3、 加载真正 DSP 程序代码到正确的地址空间中。

4、 加载中断向量表到 0 地址处。

5、 进入 DSP 程序的运行入口，开始运行编码程序。

在我们的 C6711 监控系统中有 SRAM 和 SDRAM 两种外围存储器件，分别位于 DSP 外部的 CE0 和 CE2 空间。为了使它们能够正常的工作，在 BOOT 程序的开始需要对 DSP 的 GBLCTL、SDTCL、CE0SEC、CE2SEC 等寄存器进行相应的计算和配置。在完成外围存储器件的配置后，如果 BOOT 程序的代码大于 1K 字节，还有代码在 FLASH 中未被载入，需要把剩下的 BOOT 代码载入到 DSP 中。接下来是对 FLASH 中的真正 DSP 程序代码的加载。烧写在 FLASH 中的是经过编译和连接生成的 2 进制可执行的 DSP 程序，由于编译、连接后的 DSP 程序的代码和数据结构分布在不同的空间中，有片内 RAM，也有片外的 SDRAM，所以在加载时，就需要分别把这些位于不同存储空间的 2 进制代码正确存放到各自的位置。为方便 BOOT 程序加载不同块的不同代码，我们在烧写 FLASH 程序中加入如图 4.15 的 BOOT 数据链表结构。链表的每个单元都是一块连续代码或者数据的加载信息，其中源地址表示存放在 FLASH 中的地址，目的地址是需要加载到 DSP 空间中的地址，大小表示这块代码或者数据的大小。BOOT 程序只需要沿着链表的各个单元，将每个单元的数据块加载到 DSP 中，即可完成 DSP 程序的加载工作。需要注意的是最后对中断向量表的加载，因为中断向量表在地址 0 处，上电后 DSP 自动加载的 BOOT 程序也是在地址 0 处，随意加载，就会造成 BOOT 程序被覆盖，而引起错误。所以我们在加载中断向量表之前，将 BOOT 程序跳转到地址 512 后，把 0-512 的地址空间腾出来，再进行中断向量表的加载。完成中断向量表的加载后，BOOT 程序就完成了它的使命，跳转到 DSP 程序的入口，开始执行编码程序。

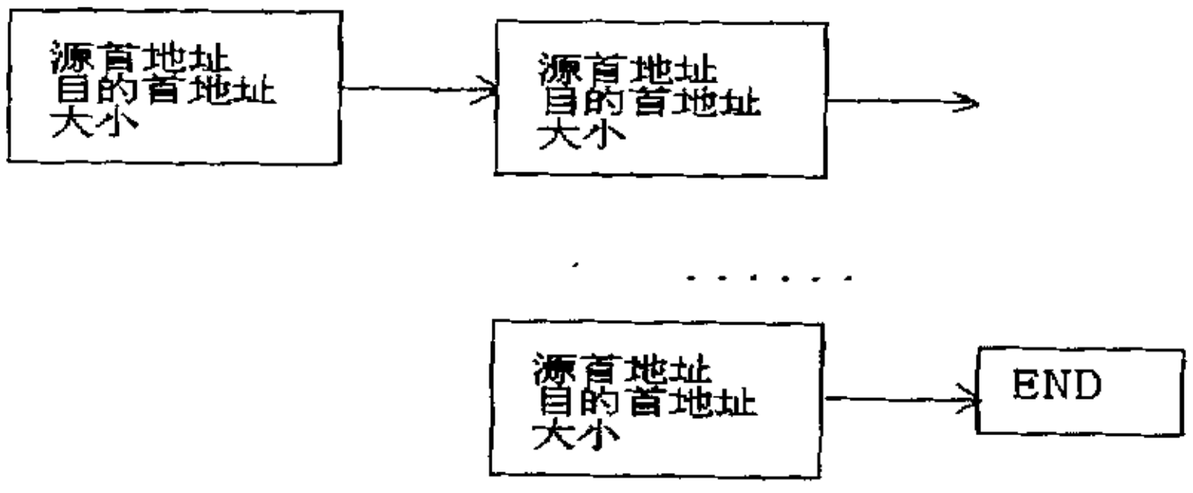


图 4.15：BOOT 链表结构

4.4 小结

DSP 上的软件系统和 PC 上的软件系统有很大的差别，这是因为 DSP 为了增强对数字信号的处理能力，相对于 PC 的通用 CPU 增加了许多不同的东西，包括多个并行处理的功能单元、VLIW 指令系统、双路数据通道、EDMA 数据访问方式以及特殊的 2 级缓存配置，这些不同都导致了原来在 PC 上的程序代码不能充分发挥 DSP 强大的功能。原始的 PC 编码程序在 DSP 上只能做到每秒 1 帧 QCIF 的编码速度。为此，在这章中我们针对 C6711 的这些特点对我们的视频编码程序进行了多方面的调整和优化。首先，调整编码程序的整体结构，使整个编码过程都以宏块为单位进行操作，改变一些函数的运行顺序；同时对某些函数进行合并或者分解。这样的调整是为了使程序中的各个函数更加紧凑，增加函数之间数据调用的相关性；以宏块为单位的结构消除了以图像为对象的操作所需要的大量储存空间，大大降低了整个程序对存储空间的要求，保证多数数据能够存放在 C671164KB 的片内 RAM 中，提高了整个程序对数据读写的速度；对于操作过于简单或者复杂的函数进行合并或者分解，使得函数的复杂度达到合适的程度，能够充分发挥 C6711 中央处理器 8 个并行功能单元的作用，有利于代码的优化。在对整个程序和数据结构进行调整后，我们针对各个函数进行具体的汇编优化，优化的目的是使得每个函数代码通过 VLIW 汇编指令，充分利用 C6711 中央处理器丰富的硬件资源，达到并行运算的目的。在代码优化过程中，我们主要采用了软件流水技术，分析函数中指令操作和数据读写通路，使得函数循环中各个操作阶段形成流水结构，提高了函数运行的速度。除了对程序代码本身的优化外，还针对 C6711DSP 总线读写的特点，对片外总线使用 EDMA 控制器搬运片外数据到片内再进行处理，避免 DSP 中央处理器浪费大量的时间在片外总线的读写上；对片内的程序和数据，结合 C6711 的 2 级缓存结构，分解大的循环，调整数据的安放顺序，尽量避免片内程序和数据一级缓存的 CACHE MISS。

通过对结构、代码和总线的调整和优化，监控视频编码效率得到极大的提高，在C6711DSK 上，优化后的程序每秒编码约 15 帧的 CIF，不但满足了我们系统的需求，也为以后的系统升级打好了良好的基础。除了 DSP 核心编码模块以外，C6711 监控编码系统里还有前端视频采集、后端串口发送和 FLASH 等其他模块，前后端模块利用中断与 DSP 的编码模块进行通信，而在 FLASH 中烧写 BOOT 程序使独立的 DSP 系统可以正常的启动和初始化，这些集成工作保证了单独的监控编码系统能够正常的运行。

第五章　客户系统和超分辨率算法研究

上面所分析的 C6711DSP 监控编码系统只是整个监控系统的服务器端，除了 DSP 系统以外，监控系统还需要有终端客户软件系统，用以显示监控的画面，并向服务器端发送指令，按照客户要求调整编码参数。这里的终端可以是固定的 PC，也可以是移动终端，比如 PDA 或者手机。在这章中，我们将介绍 PC 终端上的客户系统实现，并对超分辨率算法开展研究。

5.1 客户端系统

PC 上的客户端软件是基于 windows9x,2000 系统开发的，主要有以下三个核心部分组成：

1．收发部分。PC 客户程序通过串口拨号到 DSP 端，建立连接后通过串口接收压缩后的视频码流，并将读取的码流送入解码缓冲区中，保证解码器能从解码缓冲区中获得正确的码流。同时可以向 DSP 发送指令，调整编码参数。为了方便串口的读写操作，采用了异步的串口传输模式。

2．H.263 解码部分。这是客户端程序的核心部分。解码部分与接收部分共享一个解码缓冲区，从该缓冲区中读取待解码码流，进行 H.263 解码。随后将解码出来的图像放到图像存储区中，同时通知显示线程处理。

3．图像处理显示部分。该部分从图像存储区中读取原始的 Y，U，V 信号，并通过超分辨率算法提升图像分辨率，将图像格式转换成系统可显示的 R，G，B 格式，并在显示设备上显示。

这三个主要部分由两个独立的模块构成：接收解码模块和处理显示模块。图 5.1,5.2 分别是这两个模块的结构图。

两个模块分别在各自的线程中并行的处理，在程序中共享两块缓冲区：接收缓冲和图像存储缓冲区，线程之间对共享缓冲的读写需要进行同步，同步主要通过事件（EVENT）驱动完成。

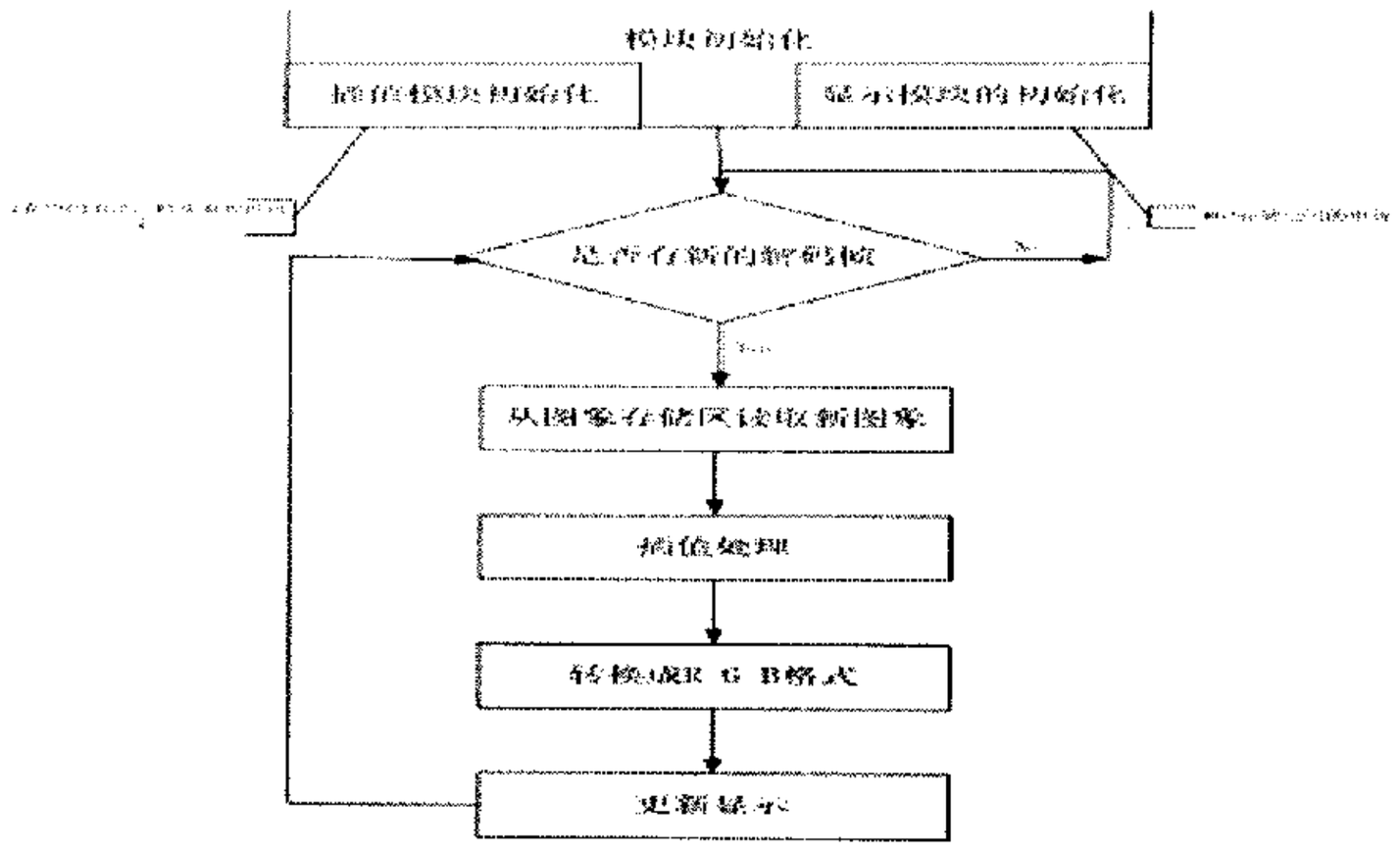


图 5.1:接收解码模块

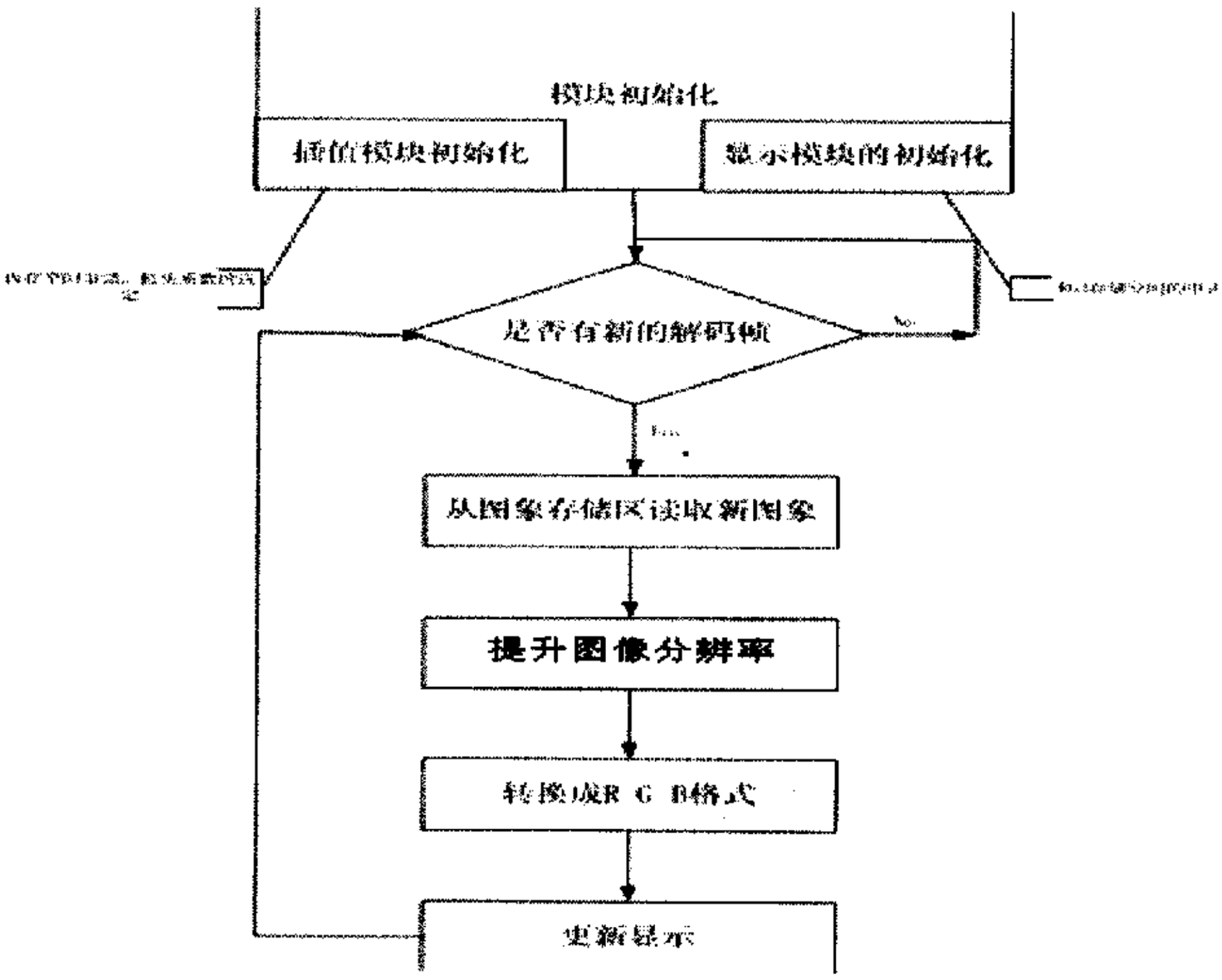


图 5.2: 处理显示模块

5.2 超分辨率算法的研究

根据第三章中的编码方案 2、3，典型码率下传输的是低分辨率的 QCIF 视频，所以我们需要在客户端提升重建图像的分辨率，以满足客户的要求，获取更好的主客观视频质量。提升图像的分辨率，从信号处理的角度上讲，就是对已经采样的数字化二维图像信号，进行

上采样，增加采样率，输出图像的采样率大于输入图像，使得重建图像的细节更容易能够被识别，也被称为图像的超分辨率处理。

5.2.1 常用的超分辨率算法

超分辨率图像处理的概念和方法最早由 Hrris 和 Goodman 于 60 年代提出。随后有许多人对其进行了研究并相继提出了各种复原方法，目前超分辨率技术主要分成两类方法：频域方法和空域方法[34]。频域方法实际上是在频域内解决图像内插问题，其观察模型是基于傅里叶变换的移位特性[35]。频域方法有以下优点：理论简单；运算复杂度低；很容易实现并行处理；具有直观的去变形超分辨率机制。但这类方法还存在以下缺点：只能局限于全局平移运动和线性空间不变降质模型；包含空域先验知识的能力有限，只能对在原有带宽内进行重建。所以频域方法一般适用于对提升分辨率后质量要求不苛刻，计算条件有限的场合中。在空域类方法中，其线性空域观测模型涉及全局和局部运动、光学模糊、帧内运动模糊、空间可变点扩散函数、非理想采样、压缩路像以及其他一些内容。空域方法的优点是不局限于线性模型，具有很强的包含空域先验约束的能力，例如马尔科夫随机场和凸集等先验约束等，这样在超分辨率复原过程中可以产生带宽外推。目前所研究的空域方法主要包括非均匀空域样本内插法[36]、迭代反投影方法[37]、集合理论复原方法(凸集投影 POCS)[38]、统计复原方法(最大后验概率估计器 MAP 和最大似然估计器 ML)[39]等。下面对对这些算法的思想做一些简单的介绍。

非均匀空间样本内插方法是先对低分辨率视频序列进行运动补偿，再采用内插方法产生单幅高密度合成图像. 以这个合成图像为初始值，采用 Landweber 迭代法来重建超分辨率图像。非均匀空间样本内插方法的缺点是过于简单化，无法重建比单幅低分辨率图像更多的频域内容，其降质模型有限，也没有使用先验约束。迭代反投影方法（IBP）是首先用输出图像的一个初始估计作为当前结果，并把这个当前结果投影到低分辨率观测图像上以获得低分辨率模拟图像. 低分辨率模拟图像与实际观测图像的差值称为模拟误差，根据模拟误差不断更新当前估计。迭代反投影方法通过观测方程使超分辨率复原与观测数据匹配，但这种方法的超分辨率重建结果不惟一，而且把先验约束引入到这种方法中也不是一件容易的事情。凸集投影方法(POCS)是目前一类解决超分辨率图像复原问题的流行算法。超分辨率图像解空间与一组凸形约束集合相交叉，而这组凸形约束集合代表了期望的超分辨率图像的一些特性，如正定、能量有界、数据可靠、平滑等，这样通过这些约束集合就可以得到简化的解空间。POCS 是指一种迭代过程，在给定超分辨率图像空间中任意一个点的前提下，可以定位

一个能满足所有凸形约束集合条件的收敛解。在统计复原方法中，超分辨率复原问题可以解释为一个统计估计问题。最大后验概率(MAP)的含义就是在已知低分辨率视频序列的前提下，使出现高分辨率图像的后验概率达到最大。根据贝叶斯原理，高分辨率图像的后验概率等价于以下两项之积：①已知理想高分辨率图像的前提下，低分辨率视频序列出现的条件概率；②理想高分辨率图像的先验概率。条件概率项通常采用高斯模型，先验概率项在不同的算法中采用不同的模型。最大后验概率估计方法的收敛稳定性取决于先验概率项。先验概率模型应该具有下面3个特点：①是一个局部平滑函数；②具有边缘保持能力；③是一个凸函数。最大似然复原方法可以认为是最大后验概率复原方法在等概率先验模型下的特例。

上面所有的这些空域超分辨率算法，尽管在重建图像质量上有相对出色的效果，但是缺点也十分明显：运算复杂度极高，收敛慢。就算对100*100的小图像进行处理，提升到200*200，也需要1秒以上的时间[40]，所以上面这些空域方法通常使用的场合需要有良好的计算条件，无实时性要求，同时对重建图像的质量要求很高，比如气象部门的卫星图像分析等。

目前在实时应用中最常用的方法还是插值算法，插值就是在存在的数据点之间划分新的间隔，计算新样本的过程．典型的插值过程涉及到估算一个拟合函数的函数值问题。因为多项式插值具有计算简单的优点，所以拟合函数在存在的数据点之间通常通过低阶多项式来拟合数据．例如，经常使用的有线性插值、双线性插值和三次样条插值。不恰当的拟合函数会使得插值后图像高频分量的丢失，从而造成重建图像边缘细节的模糊和损失。

5.2.2 基于小波变换的超分辨率算法

空域算法的时间效率太低，而单纯的插值算法又不能很好的还原图像的高频细节，文献[41]中提出了一种基于FFT的超分辨率算法，较好的解决了这两个问题。图5.3是该算法的思想，首先用二维FFT把原图像变换到频域，然后在频域中进行频率的填零扩展，扩展到原图像大小的二倍，然后再进行反FFT变换，形成分辨率提升后的重建图像。该算法在频域进行操作，比起普通的插值算法能保留原始图像的高频分量，较好的保持了图像边缘细节。在时间效率上，尽管比起空域统计算法有很大的提高，但是仍然有较高的计算复杂度。对于一副图像的二维FFT变换的时间效率与主要与FFT计算复杂度相关，我们知道一维FFT的复杂度是$O(N*\log_2 N)$，其中N满足$N=\min\{2^k \mid 2^K > L\}$，K是整数，L是信号实际长度。二维FFT的复杂度就是$O(N^2\log_2 N)$，对于二维图像信号来说，N满足$N=\min\{2^k \mid 2^k > X, 2^k > Y\}$，K是整数，X、Y分别是图像的长宽。以176*144的QCIF

格式视频图像为例，N=256。为了减少计算法复杂度，我们把图像进行分割，分割成 32*32 大小块，对每个块分别进行 FFT 添零扩展的超分辨率处理。由于对每个块来说 N=32，由于 FFT 的复杂度与 N 成指数关系，所以块处理后的复杂度大大小于对全图直接进行 FFT 处理的复杂度。但是通过划块的方法有其不合理的地方，因为我们不能保证图像中相邻的块与块之间没有相关性，如果块和块之间存在着一定的相关性，那么以块为单位进行超分辨重建后的图像必然丢失了这些块之间相关的信息，造成的直接后果就是在图像中的块与块之间出现边界效应，人的视觉系统对这种有规律的边界是十分敏感的，所以以块为单位的 FFT 方法虽然较大提高算法的时间效率，但是重建后的图像会出现块边界效应，大大影响了重建图像的主观质量。

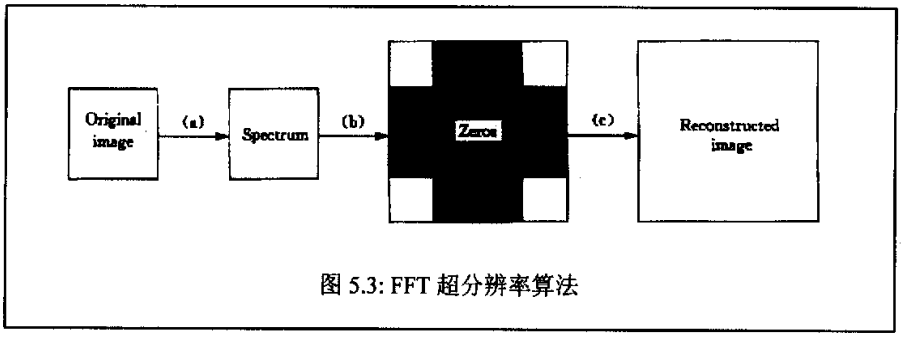


图 5.3: FFT 超分辨率算法

尽管 FFT 变换的复杂度影响到了这种算法的时间效率，但是它那种在频域中进行超分辨率处理的思想还是值得我们借鉴的。FFT 变换复杂度高，而且变换后的信息也不符合人眼的视觉机制，所以并不是最合适的空域-频域变换方式。小波理论是最近几年来由 Meyer Y，Mallat S 和 Daubechies I 等人的奠基工作迅速发展起来的一门新兴学科。小波变换是一种新的可达到时(空)域或频率域局部化的时(空)—频域分析方法，具有多分辨率分析功能和逐渐逐步细分等性质，是信号处理的一种强有力的新手段，同时，小波变换的多尺度分解特性更加符合人类的视觉机制。

设一幅图像 f(x,y)经过一次小波分解后，如图 5.4 所示被分成四个部分.MA1 为图像低频信息，MH1 为水平方向上的高频细节信息，MV1 为垂直方向上的高频细节信息，MD1 为对角线方向上的高频细节信息，而 MA1 如图 5.5 可以继续进行二次小波分解为 MA2、MH2、MV2、MD2。可以说小波的分解过程就是不断的“剥落”过程，随着逼近的越来越粗，丢掉的信息越来越多，而被抛弃的信息可用小波的线性组合来表示。而小波的重建过程就是将丢掉的细节加起来作为原始信号的近似表示，只要足够多的相同步骤这种近似就可以达到非常的精确。文献[42]将小波变换这种多尺度和多分辨率特性利用在超分辨率算法中。

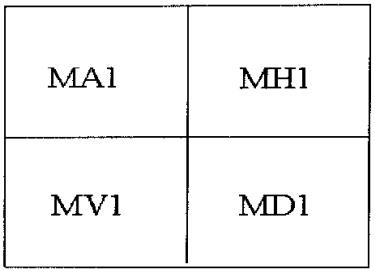


图 5.4：小波分解

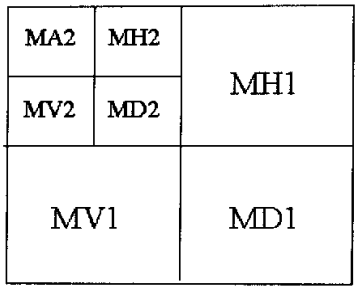


图 5.5：二次小波分解

基于小波变化的超分辨率思想就是要以低分辨率的子图通过某种方法近似为高分辨率的子图，然后通过小波逆变换得到较之原图像更高分辨率的图像。具体算法描述如图 5.6 分为三个阶段：

a、 原图像 f(x,y)做小波分解，形成 MA、MH、MV、MD 四个细节子图。

b、 将高频的 MH，MV，MD 子图利用采用双线性插值算法（记做算子 L）分别进行插值，得到高分辨率子图 MH^{*}=L(MH)，MV*=L(MV)和 MD*=L(MD)，把原图 f(x,y)作为高分辨率的 MA*。

c、 对 MA*，MH*，MV*，MD*四个子图形成的高分辨率子图做小波逆变换，变换后的结果即为所需要的高分辨率重建图像。

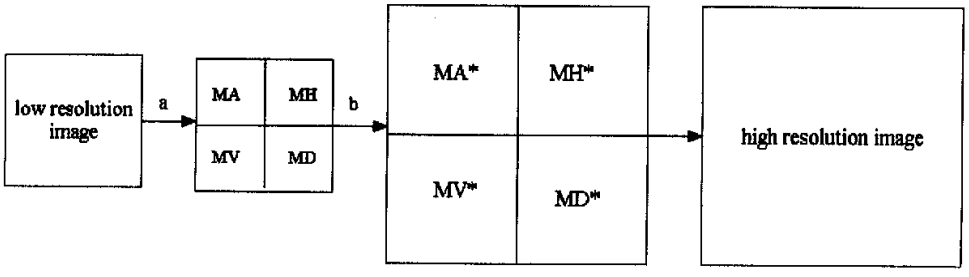


图 5.6: 基于小波的超分辨率过程

图 5.7 是该算法对 foreman 的第一帧进行分辨率提升的结果，左边是用插值算法得到的重建图像，右边是基于小波超分辨率的结果，用的是 JPEG2000 中的 5-3 整数小波算子。可以看到左边图像的背景边缘有锯齿状模糊，而右边图像无论是衣领、面部还是背景都得到很好的恢复，可以说明基于小波的超分辨率算法能够很好的对图像进行分辨率提升，获取较好的主客观质量，在监控系统的客户系统中也能够得到很好的应用。

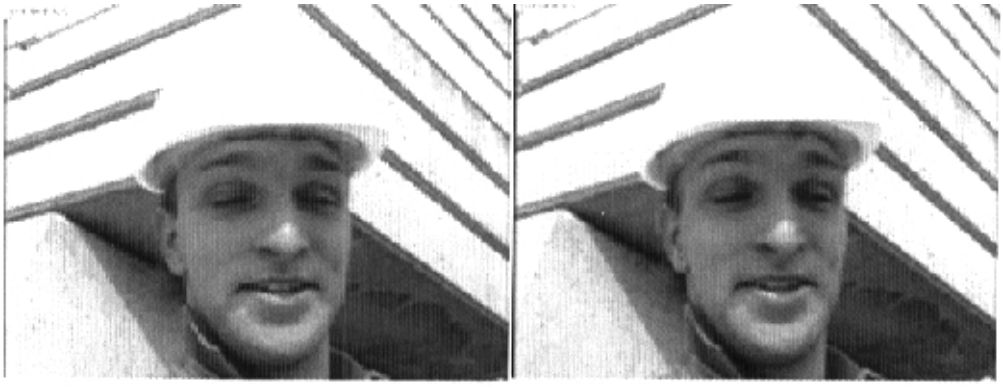


图 5.7: 重建图像

第六章　总结与展望

本文的工作主要集中于对码流输出范围在 9.6 到 112kbps 之间，典型码率为 44kbps 的 C6711 DSP 低码率监控视频的实现和优化。我们从现有的低码率视频压缩编码技术中，选择了 H.263 国际视频压缩标准作为编码算法的框架。

在实际应用中，我们感受到 H.263 是性能优秀的视频压缩算法，但是对于典型码率为 44kbps 低码率监控视频压缩仍然需要根据具体需求，探索合适的压缩编码方案。为此，我们实际测试和分析了 H.263 压缩算法中七个主要高级压缩选项的性能。根据这些测试和分析结果，结合低码率监控视频和 C6711 DSP 硬件的特点，为视频压缩方案选取了无限制运动向量、高级预测、去除块效应滤波等三个合适的高级压缩选项，确定了编码方案。然后，我们开展了对 H.263 主要编码算法的改进工作。通过查阅相关文献和进行实际的性能测试，我们在改进的 H.263 编码程序中，采用了性能更佳的钻石搜索算法和 INTER 块全零块提前判定算法。同时，我们根据监控视频背景固定的特点，在钻石搜索算法的基础上，进一步改进了运动搜索算法，提高了搜索的效率。

考虑到低码率下的重建视频的主观质量，我们在编码端对低分辨率的视频进行编码，然后在客户端进行分辨率提升，以充分利用信道，保证重建视频的流畅。为此，我们研究和探索了超分辨率算法。相关文献中的众多空域超分辨率算法能够得到高质量的重建图像，但是时间效率过低，不符合实时的要求，而单纯的插值算法又很难找到合适的拟合函数，重建图像容易丢失边缘细节。我们借鉴分块 FFT 算法中的频域处理方法，引入小波作为空域到频域的变换方法，利用小波变换多尺度分解的特性，结合双线性插值算法，具有良好的时间效率，能够较好的恢复图像边缘信息。

在把编码程序移植到 C6711 DSP 后，我们看到原有程序完全没有发挥 DSP 强大的数据处理功能。为此，我们针对 DSP 的硬件特点，对程序的结构、具体代码以及总线读写进行了多方面的优化，使优化后的编码程序不但满足目前的监控系统要求，也为以后的系统升级打下了良好的基础。

仅仅完成 DSP 的编码程序还不能使监控编码服务器正常工作。为此，我们分析了前端视频采集模块和后端串口输出模块与 DSP 之间的中断通讯状况，构造了 DSP 中断向量表和响应程序，确保了模块之间的同步和信息交换。同时，我们还研究独立 DSP 系统的 FLASH BOOT 过程，编写了 BOOT 程序，到此为止，独立的基于 C6711 低码率视频监控系统得以

完成。

回顾整个 DSP 低码率视频监控系统的研究，我们感觉到还存在着一些不足，也颇有一些其感想：

1. 编码端还需要一个针对低码率编码特点的码率控制算法。在视频编码中，一个优秀的码率控制算法不但能够保证码流的平稳输出，而且能够很好的提高重建视频的主客观质量。特别在我们的低码率视频编码中，由于信道带宽的不足，编码时往往会丢失更多的视频信息，这个时候更需要一个好的码率控制算法，来平衡码流的输出，保留重要的视频信息，提高重建视频的质量。

2. H.263 是一个优秀的视频编码压缩算法，我们只选用了其中三个选项作为低码率监控视频编码方案，主要是考虑到 TI C6711 DSP 的资源限制。随着 TI 公司不断推出新型的 C6000DSP，包括 64x 系列的高端 DSP，可用的硬件资源越来越充分，不再是限制视频压缩应用的瓶颈。所以在以后的系统升级中，可以考虑将 H.263 的其他高级选项，比如 SAC、INTRA 预测等模式加入到现有的视频编码 DSP 程序中。

3. 视频信息处理技术已经成为当前的信息技术的热点，象 H.264, AVS 这样新的视频标准不断涌现，特别是我国第一个拥有自主知识产权的 AVS 标准将眼光投向了高清晰度视频，但是我们觉得无线技术和视频技术的结合是以后视频的发展趋势，无线带宽和无线终端的条件限制导致高分辨率的视频很难应用在无线技术中，所以我们也不能忽视对诸如 CIF 这种低分辨率视频编码的研究。

通过参与 C6711 低码率视频监控系统的研究，使我对视频压缩领域和 DSP 处理器的新技术有了一个较为全面的了解，对 H.263 视频压缩标准也有了深入的认识，同时对脱离 PC 的 DSP 嵌入式编程有了全新的理解。这些都将使我终身受益，也为我今后的发展打下了良好的基础。

科技在进步，技术在发展，让我们共同努力，进步吧！

参考文献

[1] 数字视频监控系统，白木，周洁。有线电视技术，2002 年 18 期，2002 年 9 月：39—43，67。

[2] TMS320C6000 系列 DSPs 原理与应用，李方慧， 王飞， 何佩琨。电子工业出版社

[3] 甚低码率视频编码，苏洁，曹忠什，冯玉才。计算机研究与发展，第 36 卷第 4 期，1998 年 4 月：375—379。

[4] 甚低码率视频压缩编码算法的标准化，王建颖，徐立中。电力系统通信，1998 年第 5 期：23—26。

[5] 多媒体数字压缩原理与标准，[美]Jerry D. Gibson，Toby Berger 等。电子工业出版社。

[6] 结合运动补偿的三维小波图像编码，张文忠，沈兰荪。电子测量与仪器学报，第 11 卷第 4 期，1997 年 12 月：1-4。

[7] 基于监控场景的甚低码率视频压缩，刘 艳。浙江大学硕士学位论文，2003 年 2 月。

[8] ITU-U DRAFT H.263, VIDEO CODING FOR LOW BITRATE COMMUNICATION, 2, May, 1996.

[9] ITU-U Draft Text of Recommendation H.263 Version 2 ("H.263+") for Decision, 9, Feb, 1998.

[10] ITU-U DRAFT Recommendation H.263 Version 3, 11, 2000.

[11] 基于对象的多媒体数据压缩编码国际标准—MPEG4 及其校验模型，钟玉琢，王琪等。科学出版社。

[12] ISO/IEC JTC1/SC29/WG11 Document N2501, Information Technology - Generic Coding of Audio-Visual Objects, Part 1: Systems. ISO.IEC:14496-1. Final Draft International Standard. Nov, 1998.

[13] ISO/IEC JTC1/SC29/WG11 Document N2502, Information Technology - Generic Coding of Audio-Visual Objects, Part 2: Visual. ISO.IEC:14496-2. Final Draft International Standard. Nov, 1998.

[14] H.264 标准的特点和改进方法，楼剑，陆亮，虞露，董洁。电视技术，2003 年第 6 期：13—15。

[15] Motion Estimation Techniques for Digital TV: A Review and A new Contribution,

Dufaux, F.Moscheni. Proc. IEEE, vol.83, no.6, Jun. 1995: 858-876.

[16] 视频编码中的块运动估计算法，骆立俊，邹采荣。广播与电视技术，1997 年第 10 期：39-42。

[17] 多媒体通信技术基础，蔡安妮，孙景鳌。电子工业出版社。

[18] Displacement measurement and its application in interframe image coding, J. Jain, A. Jain. IEEE Trans. Commun., vol. COMM-29, Dec, 1981: 1799-1808.

[19] Predictive coding based on efficient motion estimation, R. Srinivasan, K. R. Rao. IEEE Trans. Commun., vol. COMM-33, Aug 1985: 888-896.

[20] the cross-search algorithm for motion estimation, M,Ghanbari. IEEE Trans. Commun., vol38, July 1990: 950-953.

[21] A new three-step search algorithm for block motion estimation, Reoxiang Li, Bing Zeng, M. L. Liou. IEEE Trans. Circuit Syst. Video Technol., Vol. 4, Aug, 1994: 438-442.

[22] A novel four-step search algorithm for fast block motion estimation, L. M. Po, W..C. Ma, IEEE Trans. Circuit Syst. Video Technol., Vol. 6, June 1996: 313-317.

[23] A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation, Shan Zhu, Kai-Kuang Ma. IEEE Trans. on Image Processing, Vol. 9, NO. 2, Feb 2000:287-290.

[24] Intelligent pre-quantization in motion compenstated video coding. ITU-T Study Group 16. Q15-D-35. Tampere, Finland, ITU-T 1998,04.

[25] H.263 中预先判别全零系数的新方法，周璇，谭径微，余松煜。上海交通大学学报，第 32 卷第 9 期。1998 年 9 月：107-109。

[26] Efficient method for early detection of all-zero DCT coefficients, Shi Jun, Songyu Yu. IEEE Electronics Letters, vol. 37, Issue3. Feb. 2001: 160-161.

[27] H.263 中基于全零块发现的运动搜索方法，兰军强，何芸，曹志刚。中国图像图形学报，第 5 卷第 11 期。2000 年 11 月：925-927。

[28] TMSC320C6711, C6711B FLOAT-POINT Digital Signal Processors, Texas Instrument incorporated, 2001

[29] TMS320C6211 Cache Analysis, Texas Instrument incorporated, 1998 .

[30] TMS320C6000 DSP Cache User’ s Guide, Texas Instrument incorporated, 2003

[31] TMS320C6000 Programmer’ s Guide, Texas Instrument incorporated, 2001

[32] TMS320C6000 Optimizing C/C++ Compiler User' s Guide, Texas Instrument incorporated, 2001

[33] TMS320C621x/TMS320C671x EDMA Queue Management Guidelines, Texas Instrument incorporated, 2001

[34] 超分辨率复员技术的发展，张新明，沈兰荪。 2002年5月：33-35

[35] Recursive reconstruction of high resolution image from nosiy undersampled multiframes, Kim S, Bose N, Valenzuela H. IEEE Trans ASSP, 1990, 38(6), :1013-1027

[36] Resolution enhancement of color video sequences, Shah N R, Zakbor A. IEEE Trans IP. 1999, 8 (6):879-885

[37] Improving resolution by image registration, Irani M, Peleg S. CV. GIP: Graph Models Iamge Process, 1991, 53(3):231-239

[38] Super resolution video reconstruction with arbitrary sampling lattices and nonzero aperture time, Patti A J, Sezan M, Tekalp A M. IEEE Trans IP, 1997, 6(8):1064-1076

[39] Extraction of high-resolution frames from video sequences[J], Scholtz R R, Stevenson R L. IEEE Trans IP, 1996, 5(6):996-1011.

[40] Practical Super-Resolution from Dynamic Video Sequences, Zhongding Jiang, Tien-Tsin Wong and Hujun Bao. *Proceedings of IEEE Computer Vision and Pattern Recognition 2003 (CVPR 2003)*, Vol.2: 549-554.

[41]使用分块FFT的单帧图像上行采样算法，郭晓新，李文辉， 庞云阶。吉林大学学报，Vol.4 ,No.2。2002年4月：148-152.

[42]基于小波变换和插值和超分辨率图像处理算法，陶洪久，柳建，田金文。武汉理工大学学报，Vol.24, No.8.2002.8:63-66.

致　谢

衷心感谢导师王洪玉副教授，在我攻读硕士学位期间，所给予我的关心和指导。王老师具有渊博的专业知识和丰富的实践经验，在王老师带领的科研工作中，王老师严谨细致、精益求精的科研作风，和勤奋进取、勇于探索的科研精神，为我们起到了很好的表率作用。当我们在科研工作中遇到问题和困难的时候，王老师总是和我们一起仔细思索，热烈探讨，引导我们将这些问题和困难各个击破。正是在王老师的关怀和指导下，我锻炼和积累了一定的科研能力。所有的这些，都将使我终生受益。本论文的完成，也和王老师的悉心指导分不开。

衷心感谢我的启蒙导师李宏东副教授。在李老师的支持和安排下，我得以就进入实验室，耳濡目染，参与科研项目的锻炼。李老师对于当时知识有限、经验匮乏的我，总是有问必答，耐心指点。还特意为我选择难度深度合适的科研工作，让我的科研能力能够得到由浅入深的逐步锻炼。

在浙江大学求学的这近七年的时间里，王洪玉老师和李宏东老师，是对我的学业帮助最大的两个人。他们给我的教导，和他们对待科学研究的热忱与精神，都将影响我的一生。在研究生学业即将完成之际，我谨在这里，向他们表示最深的感谢！

感谢项目组里的洪宇光、楼旭旦、乔耿佳、李洁冰、章勇等各位同学，对我在多方面的帮助和交流，给了我很多有益的启发。难忘和他们愉快的交流、探讨与合作，感谢他们对我的关怀与帮助。

感谢我的父母在学习和生活上给予我多方面的关心和帮助，他们的勉励和支持，永远是我前进的动力。

最后，感谢所有未及提及，曾经帮助过我的师长、同学和好友！

李 甬

2004 年 2 月

求是园

作者在攻读硕士期间发表的文章

- 《MPEG-21 的数字项技术及其应用》， 李甬， 楼剑 已被《电视技术》录用。
- 《一种用于极低码率视频编码的码率控制算法》， 李甬，王洪玉，李宏东，李洁冰 已被《浙大学报工学版》录用。