MPEG-4视频编码器在TMS320DM642上的实现与优化

曾 浩，高秀娟，刘 玲
（重庆大学通信工程学院，重庆 400030）

摘 要：本文结合C64系列DSP芯片的特点，讨论基于TMS320DM642 DSP的MPEG-4视频实时编码算法实现和优化方法。优化通过修改适于DSP的数据结构，有效地分配片上核心内存，合理应用EDMA、缓存Cache、线性汇编、软件流水、CCS优化工具等多种方法综合来完成。实验结果证明，这些方法提高了程序的并行性和存储器的访问效率，优化过的编码器可以实现实时编码。

关键词：图像信息处理；移植；优化；图像编码
中图分类号：TP391.41 **文献标识码**：A **文章编号**：1673－7180(2008)01－0032－5

Realization and optimization of MPEG-4 encoder on TMS320DM642

ZENG Hao，GAO Xiujuan，LIU Ling
（*College of Communication Engineering, Chongqing University, Chongqing* 400030）

Abstract: The realization and optimization of MPEG-4 encoding algorithm based on TMS320DM642 DSP are proposed. In order to realize the real-time system, many methods are used, e.g., modification of data structure, effective assignment of core memory on chip, skillful configuration of EDMA and Cache, linear assemble language, software pipeline, and optimizing tools of CCS. Experimental results show that the optimized encoder can operate well in real time.

Key words: image information process；transplant；optimization；image encode

MPEG-4是由ITU-T的视频编码专家组(VCEG)及ISO/IEC的移动图像专家组(MPEG)大力发展研究的新一代基于对象的视频压缩标准[1]。目前基于PC平台的MPEG-4视频编码器在互联网的多媒体业务方面已有较多的应用，但运用于数码摄像机、无线3G视频业务等嵌入式系统时，由于硬件资源的限制，需要进行各种优化才能满足实时性的要求[2-4]。传统优化往往基于某一个方面，而综合应用多种优化方法，可以取得更好效果。

1 DM642的主要结构特性

TMS320DM642具有强大的运算能力和高速的数据通道，其时钟频率达到了600 MHz，处理能力为每秒4800×10^6条指令，可以实现MPEG-4视频的实时编解码。其主要结构特点有：具有增强型的VelocTI结构，支持全面优化的超长指令字(VLIW)

基金项目：重庆大学自然科学基金（2007CD056）
作者简介：曾浩（1977— ），男，讲师，博士. E-mail：haoz@cqu.edu.cn

核心；先进的2级缓存结构，L1为16 K+16 KB，L2为256 KB，其中L2缓存可灵活分配为CACHE模式或RAM模式；增强的DMA控制器(EDMA)，具有64个独立的DMA通道；64位的外部存储器接口(EMIF)可实现与异步存储器和同步存储器无缝连接，外部存储器空间寻址范围达到1024 MB；10M/100 M以太网MAC，兼容IEEE802.3标准；2路视频输入，PAL/NSTC制式，1路视频输出，PAL/NSTC制式，VGA，S端子。

2 MPEG-4编码器的移植

2.1 算法简介

基于PC平台的MPEG-4标准软件主要有DIVX和XVID，本文采用了XVID1.1.0版本作为实现基础。MPEG-4通过二维时间和空间坐标的方式获得视频数据，在空间域，MPEG-4逐帧的消除帧内冗余。在时间域上，充分利用相邻帧间的相关性，在帧间进行操作。首先，在空间域压缩，是对每个宏块进行离散余弦变换、量化、熵编码。在时间域，也是最能体现Cache性能的部分，使用了运动估计与补偿，可以从参考帧重建当前帧。整个处理都是以16*16的宏块为单位，把宏块转换为4个8*8块进行具体操作的。

2.2 算法的移植

算法移植就是将在PC机上运行成功的程序移植到DSP平台上，并针对DSP特点及具体应用而进行改写以使其能达到要求。一般移植，主要需完成两个工作。

第一，消减冗余代码。

XVID1.1.0参考版本为了以后扩展需要，留了很多接口，也包含了很多冗余代码，如大量trace信息、assert信息和printf函数，这些信息在DSP终端设备上对实现MPEG-4算法没有任何必要。另外，原参考软件中通过读取文件来实现系统参数的配置非常耗时，不利于编码器的实时实现，所以预先把视频文件放入片外SDRAM中，编码时使用独立于CPU工作的EDMA进行数据搬移。还有，计算信噪比SNR等工作也没有必要由DSP来同步完成，可通过后继计算得到。而且SNR计算为浮点运算，在定点DSP上实现也很费时。程序中存在很多没必要的判断跳转，这些判断既影响了程序并行化的执行，又浪费了DSP有限的片内空间，在移植代码时削减了这些不必要的判断，大大减少了代码空间，提高了程序运行速度。

第二，重新编写原程序的头文件。

由于DM642 DSP的CCS代码开发环境具有比VC更为严格的编译环境，所以如果将在PC机上运行的程序简单照搬到DSP上是无法编译的，如在VC下的int64在DSP中应使用long代替，因C64最多支持40位的long型不支持64位。此外CCS中不需要像VC那样添加头文件，系统会自动添加。

3 MPEG-4编码器的优化

把MPEG-4编码器移植到DSP端，从而实现实时编码的要求，优化工作是十分繁重的。一般的优化顺序是先做DSP端的程序结构优化，再做MPEG-4算法级优化，然后是应用层优化，最后做代码级优化，该过程如图1所示。算法级优化中快速算法的实现会降低图像质量，而底层的优化工作不会对图像质量产生影响。故本文主要阐述底层优化方法。

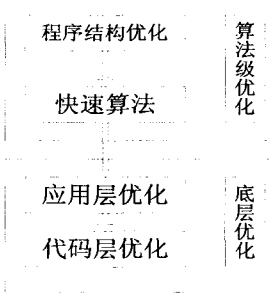

图1 编码器优化流程

Fig. 1 Optimization steps of encoder

3.1 应用层优化

根据实验经验提出了一些基于DM642硬件体系结构特征的一些优化措施。

3.1.1 编译器优化

在DSP开发环境CCS下设置编译选项，利用编译器对程序进行自动优化。

-o3选项表示可得到最高程度的优化。编译器将执行各种优化循环的方法，比如软件流水、循环展开和SIMD（单指令多数据流）等技术。

-pm选项表示程序级优化，编译器在编译时可以从整个程序的角度来优化程序。

-mt选项向编译器说明代码中没有使用混迭技术。对于汇编优化器，明确告知程序中不存在存储器相关性，即允许编译器在无存储器相关性假设下优化。

在使用-o3 选项进行优化编译时，尽量联合使用-pm 选项。实际测试表明，-o3 选项的打开对程序的优化相当显著。如果程序满足无存储器相关性的假设，打开-pm 和-mt 选项可以编码速度提高 3～4 倍。

3.1.2 Cache 优化策略

在优化过程中，我们需要合理配置 L2 的 Cache 和片内 SRAM 的大小。配置的总原则是将尽量多的关键数据分配在片内，Cache 越大越好[3]。对于不同的应用需要用不同的配置。最优配置需要在开发中根据经验和实际的测试结果进行选择，同时需要合理组织缓冲区数据，减少内存访问冲突。实际测试表明把 L2 的 128 KB 配置成 Cache，剩余的 128 KB 作为 ISRAM 使用。L2 缓存设置时使用 CSL 中的 CACHE 动态配置或使用 BIOS 静态配置。动态配置如下：

```
CSL_init( );
CACHE_clean(CACHE_L2ALL, 0, 0);
CACHE_setL2Mode(CACHE_128KCACHE);
CACHE_enableCaching(CACHE_EMIFA_CE00);
```

3.1.3 使用 DMA 技术

利用 DMA 完成在片外 SDRAM、片内 ISRAM 和 Cache 之间转移数据的任务和数据重排等功能，提高数据传输的效率和内部存储空间的利用效率。DMA 不需要 CPU 的介入，这样可以节省 CPU 取数据所花费的时间。我们使用了 TI 的 CSL 支持功能，它有专门的 DMA 模块，便于对 DMA 的各个存储器控制，还有一个 DAT 模块，使用 DMA 进行存储器间的数据传输。使用 DAT_copy( )和 DAT_fill( )就象常用的内存操作 memcpy( )、memset( )一样，只需要在 API 接口指出源地址、目的地址和长度，或其他的维数属性等即可，而不需要再去管理具体的寄存器[4]。下面的代码就是从 SDRAM 中视频序列（格式 352×288 的 CIF）中，利用 EDMA 搬移一帧数据的例子。

```
DAT_open(DAT_CHAANY,DAT_PRI_HIGH,DAT_OPEN_2D);
id=DAT_copy(rawbuf1, in_buffer, 152064);
DAT_wait(id);
```

3.2 代码级优化

代码级优化分为 C 语言级和汇编语言级，这里主要讨论汇编语言级的程序优化方案。

通过对 MPEG-4 算法各部分耗时情况使用 CCS 特有的 profile clock 工具进行测试，测试结果如图 2 所示。从中可看出运动估计补偿，量化和 DCT 模块耗时较多，因此对其进行重点优化。

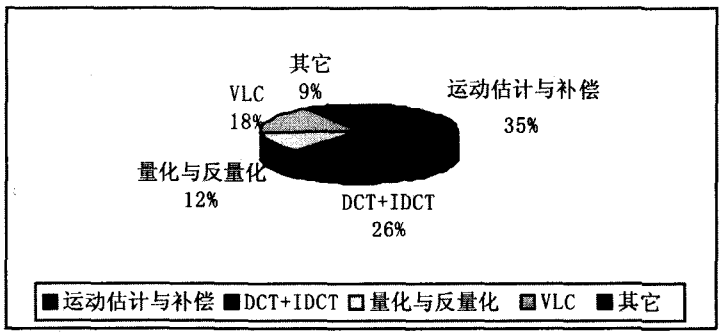

图 2 耗时测试结果图

Fig. 2 Trial result of time consume

3.2.1 使用图像库

TI 提供了基于 C64x 的图像库 IMGLIB，库中都是图像处理常用的函数，而且他们都是 C 可调用的，汇编优化好的，执行效率非常高，因此能用库函数的最好都用库函数。但是具体形式和我们的程序有些出入，需要改动。为了避免替换漏项，我们直接在 fdct.c 文件中进行改写，把里面的内容去掉，直接写入库函数 IMG_fdct_8x8((short *const)dct_data,1)。

另外，XVID 代码中函数使用了函数指针，也可利用其进行替换，这给我们的修改也带来了极大的方便。实验结果表明替换后函数执行比替换前的效率提高了 20 倍。

3.2.2 改写成线性汇编或汇编

线性汇编语言是为了简化 C6000 汇编语言程序

的开发而设计的。与标准汇编语言相比，采用线性汇编语言进行编程不需要考虑并行指令的安排、指令延迟和寄存器的使用[5]。上述工作由汇编优化器自动完成，产生的代码效率可以达到人工编写代码效率的95%～100%。如果编写线性汇编仍不能满足性能要求，可再运用纯汇编编写相关的代码。这时需要充分考虑C6000 DSP结构，尽可能并行其中的非相关语句，从而进一步减少代码的执行时间和提高程序的性能。我们把那些耗时却没法用库函数代替的函数改用这种方式进行处理。我们以SAD16为例，看看改写前后的效果。下面是改写为线性汇编的部分代码。

```
Loop:
LDDW      *cur++[cur_stride],   c_d2:c_d1
LDDW      *ref++[ref_stride],   r_d2:r_d1
SUBABS4   c_d1,   r_d1,   tmp1
SUBABS4   c_d2,   r_d2,   tmp2
DOTPU4    tmp1,   four_1,   sad1
DOTPU4    tmp2,   four_1,   sad2
ADD       sad1,   sad,   sad
ADD       sad2,   sad,   sad
[cntr]  SUB   cntr,   1,   cntr
[cntr]  B     Loop
```

从以上线性汇编画出相关图可知，迭代间隔至少是2个时钟周期，就可以用4个LDDW指令，每个周期使用2个LDDW指令，这样循环次数为16次。再考虑到交叉通路的影响，由于一个时钟周期内A侧和B侧的功能单元只能交叉1次，所以当使用4个LDDW指令时，循环迭代间隔2已经不够用了，此时循环迭代间隔应为3。通过手动汇编可得模迭代间隔表表1，汇编程序可参考此表进行编写。

表 1 模迭代间隔表

Table 1 Distance of module iterative

	0	3	6	9	12
.D1	LDDW(c1)	LDDW(c1)	LDDW(c1)	LDDW(c1)	LDDW(c1)
.D2	LDDW(r1)	LDDW(r1)	LDDW(r1)	LDDW(r1)	LDDW(r1)
.M1			DOTPU4(sad_1)	DOTPU4(sad_1)	DOTPU4(sad_1)
.M2			DOTPU4(sad_2)	DOTPU4(sad_2)	DOTPU4(sad_2)
.S2			B	B	B
.L1X			SUBABS4(d_d3)	SUBABS4(d_d3)	SUBABS4(d_d3)
.L2X			SUBABS4(d_d4)	SUBABS4(d_d4)	SUBABS4(d_d4)

	1	4	7	10	13
.D1	LDDW(c2)	LDDW(c2)	LDDW(c2)	LDDW(c2)	LDDW(c2)
.D2	LDDW(r2)	LDDW(r2)	LDDW(r2)	LDDW(r2)	LDDW(r2)
.M1			DOTPU4(sad_3)	DOTPU4(sad_3)	DOTPU4(sad_3)
.M2			DOTPU4(sad_4)	DOTPU4(sad_4)	DOTPU4(sad_4)
.S1X				ADD(sad_2)	ADD(sad_2)
.S2X					ADD(sad_3)

	2	5	8	11	14
.D1				ADD(sad_1)	ADD(sad_1)
.D2		SUB(cntr)	SUB(cntr)	SUB(cntr)	SUB(cntr)
.L1X		SUBABS4(d_d1)	SUBABS4(d_d1)	SUBABS4(d_d1)	SUBABS4(d_d1)
.L2X		SUBABS4(d_d2)	SUBABS4(d_d2)	SUBABS4(d_d2)	SUBABS4(d_d2)
.S2				ADD(sad_4)	ADD(sad_4)

表 2 是各种方式下的耗时对比表。由此我们可以看到执行时间的改善效果。上面的时间是查看 profile 中的 cycle cpu 得到的，在不包括程序调度，数据等待等。可以使用 cycle total 来查看综合效果。有时 cycle cpu 是几十，但是 cycle total 可能是几百甚至上千。这种情况主要是 CPU 要到片外存储器读取数据或代码，因此我们把能放到调用最频繁的程序和常用的数据放到片上。具体实现分 C 环境和汇编环境。在汇编或线性汇编中定义数据段 ".usect section name，size in bytes"，程序段 ".sect section name，size in bytes"；在 C 程序中定义数据段 "# pragma DATA_SECTION(symbol, section name);"，程序段"# pragma CODE_SECTION(symbol, section name);"。这些定义完成后，需要在连接命令文件(.cmd)中，用 SECTIONS 伪指令指定他们在存储器上的存放位置。

表 2 函数周期数对比表

Table 2 Comparison of function periods

程序	周期数	
	不使用-O3	使用-O3
原 C 代码	5486	360
线性汇编代码	896	98
汇编代码	60	60

4 实验结果

我们采用的是目标帧率 256 kBs/s，两个相邻的关键帧之间参数设置为 10 帧，输入序列为 CIF 格式（352×288），标准序列大小为 300 帧作为视频源，实验结果如表 3。从该表可以看出，该编码器在保持了很高的图像质量和较低的码率的同时，基本实现了 CIF 格式图像的实时性编码。

表 3 三种标准测试序列测试性能

Table 3 Trail serials performances of three criterion

序列名称	平均 PSNR /dB	总编码长 /B	平均帧长度 /B	每帧平均耗时 / ms	帧率 /帧•秒$^{-1}$
Container	26.20	418391	1399	36.37	26.95
Hall	25.78	428017	1426	390.30	25.48
Coastguard	17.34	457749	1525	54.78	24.16
压缩比	Container：27.26		Hall：26.65	Coastguard：24.91	

5 结束语

本文介绍的基于 TMS320DM642 的图像处理系统，充分利用了 DSP 适用于快速实现数字信号处理算法的优点，结合各种优化方法，最终满足要求。这对各种嵌入式图像系统的实现具有借鉴意义。

[参考文献] (References)

[1] 钟玉琢. 基于对象的多媒体数字压缩编码国际标准 MPEG-4 及其校样模型[M]. 北京：科学出版社, 2000: 2~30.
ZHONG Yuzhuo. The multimedia digital compression coding international standard MPEG-4 based on object and its correction model [M]. Beigjing: Science Book Publishing Company, 2000: 2~30. (in Chinese)

[2] 张石，张明亮，鲍喜荣，等. MPEG-4 视频解码模块的设计与优化[J]. 计算机工程. 2007, 33(10)：193~195.
ZHANG Shi, ZHANG Mingliang, BAO Xirong, et al. Design and optimization of MPEG-4 video decoder [J]. Computer Engineering, 2007, 33(10): 193~195. (in Chinese)

[3] GUO H, ShENG T, SUN W. Cache optimization for an embedded MPEG-4 video decoder [A]. The 8th International Conference on Signal Procession [C], Nanning, China, 2006, 1: 16~20.

[4] Denolf K, Vleeschouwer C, Turney R. Memory centric design of an MPEG-4 video encoder [J]. IEEE Trans. on Circuits and Systems for Video Technology, 2005, 15(5): 609~619.

[5] 费伟，朱向军，裘塞海. 基于 C64x DSP 的 MPEG-4 视频编码器算法设计及优化[J]. 信号处理, 2007, 23(2): 256~261.
FEI Wei, ZHU Xiangjun, QIU Saihai. Algorithms design and optimization of MPEG-4 video encoder using C64x DSP [J]. Signal Processing, 2007, 23(2): 256~261. (in Chinese)