文章编号:1007-1229(2010)05-0044-04

基于 DM642 的 H.264 视频编码优化

许春冬[a]，　刘亦晴[b]
(江西理工大学,a.信息工程学院;b.经济管理学院,江西 赣州 341000)

摘　要:分析了 T264 编解码的结构和复杂度. 结合 DM642 硬件特点对相关算法进行了选择、裁减和优化,给出了粗化搜索的算法. 在 DM642 EVM 上实现了硬件仿真,并重点进行了优化工作,包括使用编译器选项、C 代码改写、写线性汇编、数据搬移优化等有效的综合策略. 这些优化方法有效提高了存储器的访问效率和代码的并行性. 设计实现了 DM642 平台的 T264 算法深度移植.

关键词:视频编码; H.264; DM642; 优化方案
中图分类号:TN91　　**文献标识码**:A

Optimization of H.264 Video Encoder Based on DM642

XU Chun-dong[a], LIU Yi-qing[b]
(a.Faculty of Information Engineering; b.Faculty of Economics and Managerial Science,
Jiangxi University of Science and Technology,Ganzhou 341000,China)

Abstract: The thesis analyzes the codec structure and complexity of T264 and gives the coarse search algorithm combining with the hardware characteristics of DM642. In the DM642 EVM Emulation has been realized, and the focus is optimized,which includes the use of compiler options, C code rewriting, writing a linear compilation, data move optimization.The above optimizations help pwomote the effeciency and overall. optimization strategies of memory access and code parallelism. The design achieves the optimization and the depth migration of T264 algorithm on the DM642 platform.

Key words:video coding; H.264; DM642; optimization scheme

0　引　言

在现行视频编码标准中，如 MPEG-1(Moving Picture Experts Group)、MPEG-4 和 H.264 等标准[1]，其关键技术问题之一是整数离散余弦变换(Discrete Cosine Transform,DCT)和运动估计技术,它们是最新图像/视频编码标准中的核心部分,能为视频编解码带来高质量的图像.

研究从整数 DCT 变换算法和运动估计算法着手,以图形图像编码器的实时(Real-Time)性、视频流传输信息量对信道的适应性为研究目标. 以数字信号处理器(DigitalSignalProcessor,DSP)TMS320DM642 芯片来实现视频流的压缩[2],使论文中采用的算法和代码在视频图像实时性处理技术上得到较好的体现.

1　H.264/AVC 关键技术

H.264 是基于块的混合编码方法，相对于之前的视频编码标准采用了一些新的技术,同时编码性能也得到了进一步提高. H.264 的关键技术简介如下.

(1)运动补偿. 用已编码的前一帧(前向补偿)或后一帧(后向补偿)或前后帧来预测当前帧.

(2)整数变换. 采用与离散余弦变换相似特性

收稿日期:2010-03-31
基金项目:江西省教育厅资助项目(GJJ09230)
作者简介:许春冬(1976-),男,讲师.

的整数变换. 因为使用整数变换，所以反变换没有误差.

(3)熵编码. H.264 支持两种熵编码方法，最简单的熵编码方法是对所有的语法元素，除了量化系数外，使用单一无限可扩展的码字表.

(4)H.264 定义了一个自适应循环滤波器，滤波器的强度通过几个语法元素控制.

(5) 码率控制. H.264 编码器采用了基于 Lagrangian 优化算法的编码控制模型.

2 算法适应性裁减和移植

T264 是支持 H.264 的测试模型，编码器编码输出标准的 H.264 码流. 本研究采用 T264 测试模型.

H.264 标准定义了 4 个类型(profile)，即基本型(baseline profile)，主要型 (main profile)、扩展型(extended profile)和增强型(high profile)，每个类型支持一组特定的编码功能. 其中基本型支持帧内和帧间编码，同时熵编码采用 CAVLC(Contex-Adaptive Variable Length Coding). 实践中选择 H.264/AVC 的基本型.

JVT 相关文档显示运动估计在使用 1 个参考帧时约占全部运算量的 70 %，在使用 5 个参考帧时约占全部运算量的 90 %[3]. 可见参考帧的数量对运算复杂度影响很大，同时多参考帧适应高压缩的系统，对实时性要求高的应用可以选择参考帧数为 1. 同时，实践中选用 DS(Diamond Search)算法. 这样压缩性能可能有所降低，但计算量减少，且由于编码时去除双向预测，编码时延降低，图像帧速率有所提升. 所以，可据应用目的来裁减代码.

移植后基于 DM642 的 T264 编码器软件功能模块如图 1 所示.

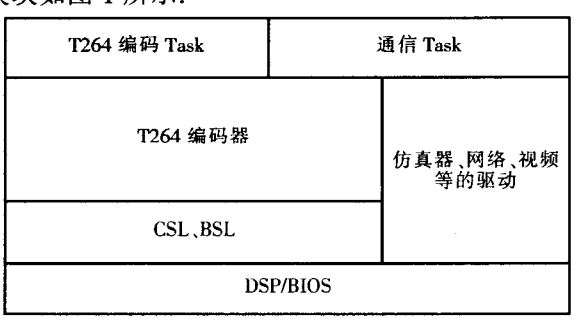

图 1 DM642-T264 编码器软件功能模块

编码器软件包括：DSP 基本输入输出系统(DSP/BIOS)、板级支持库 (BSL)、芯片支持库(CSL)、T264 编码器核心软件、视频采集驱动、网络接口驱动、仿真器驱动、通信 Task、T264 编码 Task 等部分[4].

3 DM642 中数据流程

DM642 EVM 视频图像处理平台实现的 T264 视频编解码算法，具体数据流见图 2 所示.数据流程描述如下：

采集一帧图像到输入缓存，获得的图像数据进行格式转换后得到 YUV4:2:0 格式；将数据进一步送到 T264 编码模块中，任务完成后得到编码码流；将编码码流送到 T264 的解码模块，任务完成后得到解码输出的一帧图像，图像经过重抽样得到 YUV4:2:2 格式；最后把图像传至显示设备输出.

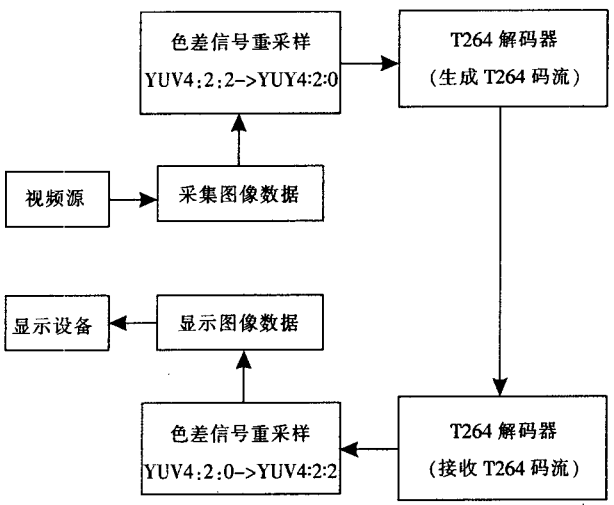

图 2 T264 编解码的数据流图

4 T264 在 DM642 上的优化

4.1 使用编译器选项

CCS 是一个完整的 DSP 集成开发环境[5]，它提供了很多编译选项，这些选项控制着编译器的操作.

-o3 选项：能进行文件级优化，选用这种级别可以得到最高程度的优化，编译器将执行各种优化循环的方法.

-pm 选项：能进行程序级优化，编译器在编译时能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.

根据 CCS 提供的以上系列编译优化参数(-o3, -pm,-mt 等)，按照代码性能要求进行组合和优选.在使用-o3 选项进行优化编译时，尽可能同时使用-pm 选项. 这可以通过 CCS2.21 的 PBC 选项来完成. 同时在代码链接过程中，对代码段链接顺序进行一定的安排，可以减少程序执行时代码调用带来的缓存缺失，提高程序的执行效率[6].

测试表明,-o3 选项的启用对代码有显著优化效果.如果程序满足无存储器相关性的假设,打开-pm 和-mt 选项可以提高编码一帧的时间 4 ms 左右.

4.2 循环展开

在 T264 参考编解码器程序中, 循环结构占据很多部分,对于循环结构,采取的优化方法是:把内循环展开,尽可能避免采用循环的嵌套,减少循环次数,使程序只有一个外循环,有些不能展开的循环则合理安排循环内外层,可以有效提高流水效率(软件流水是一个功能强大的编译器优化技术[6]).这样可增加可能并行的指令数,从而改进软件流水编排,改善代码性能.T264_sad_u_c 程序循环展开后的代码如下:

```
T264_sad_u_c (U8* src, INT32 src_stride, U8* data, INT32 width, INT32 height, INT32 dst_stride)
{
register INT32 i, j;
register U32 sad;
sad = 0;
for(i = 0 ; i < height ; i ++)
{
sad+=_subabs4( *((U32*)(data+0)),
*((U32*)(src+0)) );
sad+=_subabs4( *((U32*)(data+4)),
*((U32*)(src+4)) );
sad+=_subabs4( *((U32*)(data+8)),
*((U32*)(src+8)) );
sad+=_subabs4( *((U32*)(data+12)),
*((U32*)(src+12)) );
src  += src_stride;
data += dst_stride;
}
return sad;
}
```

对循环展开时, 同时按照上述的线性编码指令_subabs4 进行改写,可以看出来,内层 for 循环语句加常量和加变量不一样, 常量在编辑时已算出,变量需在运行时计算, 由于编辑计算不需要时钟,即不占用运行时间,可以明显提高程序运行效率.

经过测试, 程序相对于只采用线性汇编而没有经过循环展开测得的运行效率提高了 3.4 倍. 而相对于未经任何优化的 C 程序,其效率提高了 35.1 倍.

循环拆解,将内部 for 循环打开,排流水线,提高并行性, 进一步提高了代码执行效率. 论文中还对 DCT 等模块进行了大量的循环拆解.

4.3 写线性汇编

一般在编译程序时选用的优化级别为 o3,但是,DSP 的 C/C++编译器对于实时性要求较高的应用,或者对于较为复杂的代码仍然很难达到优化的要求. 为此需要采用汇编语言编写代码中的低效率段,尤其是经常调用的函数或者较大计数的循环这样的关键段[6]. 但编写汇编语言较复杂,使用线性汇编语言较方便. 线性汇编是针对 C6000 的结构特点优化设计的介于 C 和汇编语言之间的一种编程语言.分析视频编解码器 T264 运算复杂度,得出耗时量大的模块主要有变换和量化、半像素插值及运动估计等. 优化时可以选择这些模块来进行线性汇编改写, 每个相应函数的实现速度要提高不同的倍数,节省更多的时钟周期,使用线性汇编语言改写关键函数不但可以提高编码效率,而且对视频图像的 PSNR 值不产生影响.

计算 SAD 是 T264 编码过程中的重要环节,帧间预测时用到的运动搜索和帧内模式选择时都要频繁的调用 SAD 模块.

线性汇编改写前的程序:

```
T264_sad_u_c (U8* src, INT32 src_stride, U8* data, INT32 width, INT32 height, INT32 dst_stride)
{
 register INT32 i, j;
 register U32 sad;
 sad = 0;
 for(i = 0 ; i < height ; i ++)
{
 for(j = 0 ; j < width ; j +=4)
{
 INT32 tmp;
 tmp = data[j] - src[j];
 sad += ABS(tmp);
 tmp = data[j+1] - src[j+1];
 sad += ABS(tmp);
 tmp = data[j+2] - src[j+2];
 sad += ABS(tmp);
 tmp = data[j+3] - src[j+3];
 sad += ABS(tmp);
}
 src  += src_stride;
 data += dst_stride;
}
 return sad;
}
```

采用线性汇编语言改写后的程序:

```
T264_sad_u_c (U8* src, INT32 src_stride, U8* data, INT32 width, INT32 height, INT32 dst_stride)
{
 register INT32 i, j;
```

```
    register U32 sad;
    sad = 0;
    for(i = 0 ; i < height ; i ++)
  {
    for(j = 0 ; j < width ; j +=4)
  {
    sad+=_subabs4( *((U32*)(data+j)),*((U32*)(src+j))
);
  }
    src  += src_stride;
    data += dst_stride;
  }
    return sad;
  }
```

对照原来 C 代码可以看出，sad+=_subabs4()函数代替了其中 for 循环的 9 条语句. 设宽和高均为 16，对 T264_sad_u_c()函数进行测试，经过线性汇编优化后，代码运行效率提高到原来的 10.4 倍. 可见线性汇编代码经汇编优化器优化后，函数运行速度提升明显.

4.4 数据搬移优化

由于 DM642 片上存储空间的限制，只能将参考图像和重建图像等数据放置片外 SDRAM，但也带来访问外部存储器带来的巨大开销. 而 DM642 具有的 EDMA 和 QDMA 只需花费数个时钟周期进行参数初始化后，就可在 CPU 后台进行高速的数据搬移操作，提高了程序执行效率[7]. 因此，T264 编码中有大量的数据搬移操作，可交由 EDMA 来完成，这样可以大大节省系统进行视频序列处理的时间. EDMA 主要将片内 ISRAM、片外 SDRAM 和 Cache 之间的数据进行搬移，还可执行数据重排等操作.

(1)对于简单的数据搬移任务，可采用 CSL 库提供的 DAT 模块进行. DAT 模块依靠 DMA/EDMA 进行数据传递来实现功能. 对于 DM642，DAT 模块使用的是 QDMA 进行传输. 如紧耦合的循环代码中的数据搬移任务常采用 QDMA 进行传输. 使用 DAT 模块首先需要用 DAT open 语句打开该模块. DAT_copy 则实现了从源地址拷贝一块线性数据到目的地址的操作.

(2)对复杂的数据搬移，可采用多通道的 EDMA 来执行[8].同时指出，因为 SDRAM 中的待搬移数据在 L2Cache 中存在副本，因此在进行数据搬移前，需对 L2Cache 和 SDRAM 中的待搬移数据进行一致性操作(Coherence Operations)，否则将得不到正确的结果.

特别指出，有些操作并不依靠 DMA，如在 DMA 非常忙碌的时候，而此时待传输的数据量又很小，那么可直接用 CPU 来访问，或者将两者配合来使用.

因此，为便于 DMA 操作，降低 CPU 负荷，在实践中对读写帧存、插值函数的数据结构进行了相应调整.

5　实验结果及分析

经过上述优化后，对 QCIF 格式的 Foreman、Silent、Container、News、Suzie 测试序列进行实验. 序列均采用 IPPP…的编码模式，长度为 20 帧，QP=20，搜索范围(-16,16)，参考帧数为 1. 实验结果见表 1.

表 1　测试图像序列的性能参数

测试序列	QP	压缩比	PSNR /dB		
			Y	*U*	*V*
Foreman	22	55.6	39.09	41.40	43.64
Silent	22	108.1	45.86	48.65	49.11
Container	26	15.1	37.17	40.79	40.41
News	26	17.6	37.50	39.32	39.85
Suzie	32	5.8	33.15	41.76	41.05

实验测试表明：压缩 QCIF 格式视频序列的速度随图像复杂度的不同从移植时的不到 5 帧/s 上升到 35~60 帧/s 的实时压缩编码，优化效果明显，达到了实时编码的效果，且图像质量下降不明显，说明以上编码的优化方法是行之有效的. 同时，随着 QP 的增加，图像主观质量下降，QP=32 时，主观质量较差. 因此，可根据项目需要适当的设定 QP 等参数. 同时，YUV 视频文件经 T264 编解码输出后，在信噪比和压缩比方面具有良好的性能.

参考文献:

[1] 毕厚杰.新一代视频压缩编码标准-H.264/AVC[M].北京：人民邮电出版社，2005.

[2] Wiegand T, Zhang X, Girod B. Long-term Memory Motion-compensated Prediction for Video Coding [J]. IEEE Trans Circuit Syst Video Technol, 1999,9(3):70-84.

[3] 周 巍.基于 H.264 的快速运动估计算法[J].计算机工程与设计，2007,28(2):379-385.

[4] 吴君钦.H.264 嵌入式实时视频编码器设计与实现[J].江西理工大学学报，2009,30(5):55-57.

[5] Liao Ji-rong. Maximize the Throughput of Resource-constrained Software Pipeline by Unrolling [J].Pure and Applied Mathematics, 2004,(23):23-27.

[6] 李　慧.H.264 运动估计和运动补偿算法的优化与实现[D]. 北京：北京邮电大学，2007.

[7] 张　琛，戚文芽.基于 TMS320DM642 的 MPEG4 算法优化[J].电子工程师，2005,31(2):39-41.

[8] 汪安民.TMS320C6000DSP 实用技术与开发案例[M].北京：人民邮电出版社，2008.