H.264编码器在TMS320DM642上的优化

彭超男[1],王京玲[2],张勤[2]
(1.北京新岸线软件科技有限公司,北京 100084;2.中国传媒大学信息工程学院,北京 100024)

摘 要:H.264是ISO/IEC和ITU-T共同制定的新一代视频编码国际标准,在数字视频通信领域有着广泛的应用前景。本文详细介绍了在TMS320DM642芯片上实现H.264实时编码的优化思想,采取的优化技术和具体优化过程,并给出优化前后编码器性能对比的实验结果。

关键词:视频编码;H.264;DSP;优化
中图分类号:TN391.41 **文献标识码**:A **文章编号**:1673-4793(2007)01-0044-06

The Optimization of The H.264 Encoder Based on TMS320DM642

PENG Chao-nan[1], WANG Jing-ling[2], ZHANG qin[2]
(1. Beijing Nufront Software Tech. Co. Ltd, Beijing100084, China; 2. Information Engineering School, Communication University of China, Beijing 100024, China)

Abstract: H.264 is the newest video compression standard jointly published by ISO/IEC and ITU-T, it will be applied widely in digital video communication field. This paper introduces the optimization ideas and describes some optimization techniques to realize real-time encoder on TMS320DM642 chip. Some corresponding experiments have been performed on the un-optimized encoder and the optimized encoder and the results are presented and analysed.

Key words: Video coding; H.264; DSP; Optimization

1 引言

H.264是ISO/IEC和ITU-T共同制定的新一代视频编码国际标准,它和以往的视频编码标准类似,也是基于块匹配的混合编码框架,基本算法仍然是通过帧间预测和运动补偿来消除视频序列的时间冗余,通过变换编码消除频域冗余。但在技术细节上引入了多种新技术,如帧内预测,子象素运动估计,多尺寸运动补偿,多参考帧运动估计等[1]。正是因为这些新技术所产生的效果累积,导致H.264编码性能得到极大的提高,在相同的重建图像质量下,它有更高的压缩比、更好的IP和无线网络信道的适应性。

H.264编码性能的提高是以编码复杂度大大增加为代价的,它的运算量是以往视频编码标准的好几倍,因此,选择快速、稳定的处理器作为信号处理平台是H.264获得广泛应用的关键。目前数字信号处理器(DSP)的高速发展,为实现H.264的高效视频编解码提供了可能性。TI公司推出的TMS320DM642

收稿日期:2006-04-21
作者简介:彭超男(1980-),女(汉族),湖南娄底市人,北京新岸线软件科技有限公司.
E-mail: pengchaonan@sohu.com

(以下简称DM642)是一款针对多媒体处理领域应用的DSP,它是基于C64x的CPU,采用TI第二代高性能超长指令字(VLIW)结构VelociT1.2TM,主频600MHz,峰值处理速度达到4800MIPS[2],这样强大的运算能力,使得DM642可支持实时的视频处理。因此,本文选择它作为H.264编码器的硬件平台。

2 H.264编码器在DM642上的优化过程

目前很多H.264的参考源代码都不能满足实时应用提出的25帧/秒的需要,难于应用。因此,如何进行优化,找出优化重点,提出一个好的优化方案是提高编码速度的关键问题。

一般来说,提高软件运行速度的途径有两种:

(1)优化软件运行环境,并选择好的编译器。

(2)手动优化

第一种是通过提高处理器的运算速度和性能,选择好的编译优化器,缩短代码长度,以使软件运行速度得到很大的提高。CCS自带的C编译器效率可达70%-80%的汇编语言级C编译器,编译器的优化选项具有4个不同的优化级别。

编译器的自动优化毕竟是有限的,它更不可能从算法上去优化,如果结合手动优化,往往会产生意想不到的效果。手动优化的方法有很多,例如优化算法,优化循环,优化存储器的访问,手工汇编等。另外,C6000独有的汇编优化器可以使开发者采用线性汇编语言得到近似标准汇编的性能。因此,降低了汇编优化的开发难度。

本文将针对上述2种优化途径从算法优化,编译选项,C代码优化,线性汇编等方面对编码器优化进行讨论。

2.1 优化平台的选择

目前H.264有三大开源代码:JM、X264和T264,其中,JM是H.264的官方测试源码,实现了H.264所有的特性,学术研究的算法很多都是在JM基础上实现并和JM源码进行比较。但其程序结构冗长,只考虑引入各种新特性以提高编码性能,其编码复杂度极高,不宜实用。因此,本文没有考虑它。

X264是网上自由组织联合开发的兼容H.264标准码流的编码器,它注重实用,无自己的解码器。和JM相比,在不明显降低编码性能的前提下,努力降低编码的计算复杂度,代码结构相对简单。

T264是中国视频编码自由组织联合开发的H.264编解码器,编码器输出标准的H.264码流,T264的整体结构取自于X264,但是对其中的运动估计等部分进行了一些改动,所以在相同条件下编码速度比X264高出2—3倍左右,编码性能比X264略有下降。

考虑到T264编码速度快,代码运行效率高,编码性能细微的劣化可以忽略,在此基础上做进一步的优化实现起来会方便许多。因此,本文选择T264作为软件优化平台。

2.2 项目级优化

T264开源代码是一个可移植性很高的程序,已经搭建好了在CCS中的开发环境,并且在程序中尽量使用了高效的表达式。CCS的开发环境中提供了很多编译选项,这些选项控制着编译器的操作[3]。其中,三级优化选项-o3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,如软件流水,循环展开和SIMD(单指令多数据流)等。本课题选用-o3选项。

项目级优化后,启用了-o3(T264lib.pjt),-o1(T264.pjt),-g,-pm优化选项和-mv6400编译选项,实验发现启动编译优化选项对代码运行效率影响很大,优化后编码器的编码速度提高了近6倍。

2.3 分析代码性能确定优化对象

通常一个程序的80%的运行时间集中在10%的代码上。而其他代码仅占用了整体运行时间的20%左右。因此,找出这10%的代码进行重点优化才能做到有的放矢。在开展下一步优化前,先用CCS自带的profiler工具(load6x)进行代码剖析,获得每个函数的具体运行情况。在DOS状态下运行:load6x -g T264.out,剖析结果存储在T264.vaa内,从剖析结果可看出,耗时的模块主要有运动搜索,变换,量化和插值等。其中,运行次数多,耗时多的函数集中在搜索T264_search(),求块的SAD值(T264_satd_i16x16_u_c,T264_sad_u_16x16_c,T264_sad_u_4x4_c等),1/2和1/4象素插值(interpolate_halfpel_hv_c,interpolate_halfpel_h_c,T264_pia_u_c,interpolate_halfpel_v_c等),量化/反量化(quant4x4_

c, iquant4x4_c), 运动矢量预测(get_pmv, T264_predict_mv), DCT/IDCT(dct4x4_c, idct4x4_c), 计算/补偿残差(expand8to16sub_c, contract16to8add_c)等。

因此,根据上述分析结果可以初步确定要优化的函数模块如下:运动搜索模块,特别是搜索算法的优化;预测编码过程中求SAD(SATD)的模块,DCT/IDCT及量化模块,象素插值模块等。接下来将针对这几个模块进行重点优化。

2.4 运动搜索算法优化

基于块匹配的快速运动搜索算法是影响整个编码速度和质量的关键,目前用得最多,搜索速度最快的是钻石搜索算法,本课题的参考源代码选的是小钻石搜索算法,本文对该算法的实现进行了一些细节上的改进。

目前,人们提出了各种提高运动搜索速度的方案,其基本原理是当误差值小于某一门限时,把当前位置作为最佳匹配块位置,不必再计算其他的搜索点,提前结束搜索过程,即所谓的"早中止 early stop"策略。这种方法是在牺牲部分视频质量条件下来提高运动估计的速度。因此,如何设定误差门限是该方法需优先解决的问题。

T264参考源代码中用的门限值为搜索块内的象素点数,通过对大量不同标准序列测试发现该门限值设得太小,未能发挥"早中止"的优势。在优化过程中对门限值的设置进行了改进。采用Libo Yang等提出的可变块大小早中止算法。他们在[4]中提出,如果量化参数QP=32,则运动搜索过程中的门限值根据块大小分别为:2500(16x16)、1500(16x8)、1400(8x16)、920(8x8)、625(8x4)、570(4x8)、500(4x4)。另外,不同的QP对应不同的门限值。根据率失真准则:$J = SAD + \lambda R$,其中,R是编运动矢量差所需的bits数,J与λ有关,λ是根据QP从查找表中获得的,因此需根据QP值来调整门限。把QP=32时的门限值作为基本门限T_0,QP改变时,门限值准则调整为:

$$T_{QP} = T_0(\lambda_{QP} - \lambda_{32}) * 10$$

对钻石搜索算法的门限值准则调整后,编码速度约提高26%,当然图像质量略有下降。

2.5 优化C代码

(1)使用intrinsics(内联函数)

C6000编译器提供了许多内联函数,这些函数直接与C6000的汇编指令映射,可快速优化C代码。优化过程中用到的内联函数包括:_abs(),_amem4(),_amem4_const(),_pack2(),_packl4(),_amemd8(),_amemd8_const(),_min2(),_max2(),_dotpu4()等,调用这些内联函数大大减少了函数执行时间。

(2)宽内存数据访问和数据打包处理

C6000访问存储器很费时,为了提高DSP的内存数据吞吐率,可用单指令访问内存中连续的多个短字长数据,当程序要对一连串短数据进行操作时,把数据进行打包处理,这样可以减少对内存的访问,打包操作的指令在一个指令周期内可并行处理一组数据,从而提高运行速度。

(3)软件流水

软件流水是一种用于安排循环内指令运行方式,使循环的多次迭代能并行执行的一种技术。编译C代码时,使用-o3选项,编译器能从程序中收集信息,尝试对程序循环实现软件流水。在C代码中,用户不用直接参与软件流水的实现,只要向编译优化器提供信息,使之能较好地编排软件流水就行。因此为改善软件流水,优化过程中主要利用特定的关键字和指令向编译器提供优化信息。如:使用关键字restrict消除数据间的相关性;使用并行代码,尽可能把长的有依赖的代码链分解成几个可以在流水线执行单元中并行执行的没有依赖的代码链;用宏指令#pragma MUST_ITERATE告诉编译器有关循环迭代次数的信息等。编译器会根据这些信息安排指令并行,进行软件流水。

(4)某些函数实现过程优化

从代码剖析的结果看,有些函数完成的功能很简单,但调用次数颇多,在这些函数上运行耗费的总时间不少,但它们的运行效率低,优化的余地比较大。比如:T264_median(求3个数的中值)修改了其实现过程,简化了求中值时的比较步骤,同时将它改成内联函数,减少函数调用的开销。array_non_zero_count函数将for循环里取数组元素的部分,改成用指针访问且每取完一个值指针自动指向下一个内存单元,这样节省了大量的寻址时间。

2.6 线性汇编级优化

线性汇编是C6000特有的一种编程语言,介于

高级语言和汇编语言之间。在线性汇编代码中不需给出汇编代码必须指出的所有信息,如使用的寄存器,指令使用的功能单元等。汇编优化器会根据代码的情况确定这些信息,并安排软件流水。

本课题将部分耗时的函数用线性汇编进行了改写,包括计算SAD,DCT/IDCT,量化,插值等,下面以计算16x16亮度块的SAD为例来说明线性汇编的实现过程。

计算SAD值是H.264编码过程中一个很重要的步骤,无论是帧内模式选择还是帧间预测中的运动搜索都需要频繁调用该模块,计算SAD的公式为:

$$SAD = \sum_{M}\sum_{N} | S(x,y) - S(x',y') |$$

求2个16x16亮度块的差的绝对值之和时,考虑到DM642有数据打包处理指令:

LDDW:一次从内存中读出64位,即8个相邻的亮度值

SUBABS4:一次计算4个亮度值的差的绝对值

DTOPU4:求2个4字节向量的点积

亮度值都是不大于128的无符号单字节数据,每对亮度值的差的绝对值仍是8bits单字节数据,这样SUBABS4计算出来的结果与0x01010101点积得到的刚好是4对亮度值的差的绝对值之和。为了能够更好的运用流水线机制和充分利用DM642的通用寄存器组,采用连续读取两组数据的方法进行处理。线性汇编代码经汇编优化器优化后,该函数运算速度提升了4倍多。

2.7 存储器分配的优化

H.264编码过程中反复从外存进行数据和程序的搬运,CPU等待耗费的时间非常多,因此需要对存储器等系统资源的分配进行优化,以减少CPU的等待时间。存储空间分配的基本原则是尽可能避免CPU对片外数据的直接访问,尽可能把执行次数多和耗费时钟周期较大的段放在片内数据(程序)存储器,尽可能减少Cache不命中(Miss)的出现次数,尽量使调入Cache的代码和数据能被最大限度的重复利用,从而大幅度提高CPU的利用率。

DM642具有L1和L2两级缓存结构,CPU访问L1的速度最快,但不能寻址,用户无法进行操作和分配。它包括相互独立的16KB一级程序缓存L1P和16KB的一级数据缓存L1D。L2为256 KB,是L1与外存的中间层,可同时存储程序和数据,它有5种配置模式:

全部配置成内部SRAM映射到地址空间

224K RAM + 32K L2cache

192K RAM + 64K L2cache

128K RAM + 128K L2cache

256K L2cache

一般来说,对内存的访问,片内SRAM最快,片外SDRAM次之,cache越大,系统访问片外SDRAM的速度越快,但是同时片上SRAM资源越小。因此如何采用合理的SRAM和cache分配策略来组织程序和数据,对提高编码器的运行速度很重要。

T264参考代码中,把L2全部配置成256KB cache,程序和数据都放在片外SDRAM,片内没有SRAM,编码时全靠通过L2 cache从片外SDRAM中读取代码和数据。优化后,把L2配置为Cache和SRAM混合使用,各128KB[5]。

为了尽可能提高cache的命中率,需调整数据和代码在内存中的分配。把反复调用的程序段放在片内程序存储区中,如DCT/IDCT,运动搜索,计算SAD,插值等模块;把频繁用到的数据段放在片内数据存储器中,如全局变量,静态变量,量化和反量化表,熵编码表,当前块,参考帧搜索窗内的数据等;把不常用到的程序和数据段及占用空间较大的数据分配到片外存储器中。在C代码内用#pragma CODE_SECTION或#pragma DATA_SECTION,在线性汇编代码内用汇编伪指令.sect把这些需调整的部分定义成新的代码段或数据段,同时修改cmd文件,将这些新定义的段正确分配到地址空间中去。

另外,在代码中选择合适的数据类型以减少程序对内存带宽的需求,如将运动矢量、参考帧和QP等数据设置为short型[6]。

3 优化后的综合结果

本课题对优化前后的编码器进行了大量实验,实验前先固定一组编码参数,本文主要是研究Baseline档次下的编码,修改T264的编码配置文件(enconfig.txt)把编码器的主要参数设置为:

测试序列为标准QCIF序列,运动搜索的最大范围为±16,一共编100帧,I帧间隔为10,不编B帧,即IPPPP...编码模式,参考帧个数为1,量化参

数QP为26,不启用环路滤波,帧间编码时允许使用所有的7种分块方式,P宏块允许帧内预测,1/4象素运动估计,使用钻石搜索算法,使用SAD作为块匹配准则,不使用CABAC熵编码方案。

3.1 优化后的编码速度和性能对比

实验过程中对各种不同的标准序列分别进行测试,优化前后的编码速度及编码性能对比表格如下:

表1 编码器的实验结果

测试序列	特点	条件	编码速度(fps)	平均PSNR		
				y	u	v
Forman	运动剧烈 有场景切换	优化前	2.97	36.71	39.99	41.95
		优化后	27.68	36.64	39.97	41.90
Mother and daughter	背景简单,基本不变 画面人物运动幅度不大	优化前	3.34	37.68	41.00	40.89
		优化后	29.59	37.64	40.92	40.89
News	背景变化 画面人物运动幅度不大	优化前	3.39	38.09	40.69	41.25
		优化后	30.08	38.04	40.67	41.23
Container	背景简单 画面景物缓慢运动	优化前	3.46	37.58	42.04	41.74
		优化后	30.40	37.56	42.04	41.74

注意:这里的编码速度包含了每编完一帧后计算PSNR的过程,该计算过程只是为了方便了解优化对编码性能的影响,一旦编码器投入使用后,在实际编码过程中不需要计算PSNR,且该计算过程对编码器的编码性能没有任何影响,只会增加编码过程的运算量影响编码速度。因此,编码器的真实速度应该是去掉PSNR计算后的编码速度。去掉PSNR计算过程后,foreman QCIF序列的编码速度为34.75fps, Mother and daughter QCIF序列为37.88fps,完全达到实时要求。

3.2 优化前后的图像对比

视频原始图像,优化前和优化后的图像质量对比如下:

图1 Foreman序列优化前后I帧图像质量对比

图2 Foreman序列优化前后P帧图像质量对比

4 结束语

本文充分利用DM642的结构特点,采取了一系列的优化措施,优化后编码速度得到很大的提高,已经实现对QCIF格式视频的实时编码。今后将对运动搜索和模式选择等模块提出相应的快速算法,提高某些模块线性汇编的效率,同时将更多的函数用线性汇编改写,最终实现CIF视频的实时编码。

参考文献

[1] Iain E. G. Richardson. H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia [M]. Chichedter: John Wiley & Sons, 2003.

[2] TMS320DM642 Technical Overview Texas Instrument Inc. [Z] USA, Sep. 2002.

[3] 李方慧,王飞,何佩琨. TMS320C6000系列DSPs原理与应用[M]. 北京:电子工业出版社,2003.

[4] Libo Yang, Keman Yu, Jiang Li, etc. An Effective Variable Block-Size Early Termination Algorithm for H.264 Video Coding [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2005,15(6).

[5] 刘方. H.264编码器在TMS320DM642上的存储空间分配方案[A]. 章毓晋. 第十二届全国图像图像学学术会议(NCIG'2005)论文集[C]. 北京:清华大学出版,2005年10月

[6] 何莉莉. 基于TMS320DM642的H.264视频编码器设计[J]. 西华大学学报,2005,24(5).

(责任编辑:王 谦)

(上接第58页)

识,不过,即使最大输出功率的晶体管也不能提供像电子管放大器提供给雷达系统那样的几十或几百千瓦的功率。射频晶体管货源充足而且便宜,射频功率的高电平可以借助大量固态放大模块的功率输出相加而获得。以此方式取得高功率放大,其可靠性远超过真空管放大器的设计。本文中功率合成器采用了径向传输线结构,从合成器测试结果可以看出,该功率合成器的研制是成功的,取得了预期的效果。

参考文献

[1] Foti, S R Flam, W Scharpf. 60-Way Radial Combiner Uses No Isolators. Microwaves & RF, July 1984, p96.

[2] R P Flam, J P Macgahan. Radial power combiners for solid-state power amplifiers.

[3] Marcuvitz, N Handbook. IEEE Electromagnetic Waves Series Volume 21, Peter Peregrinus Ltd London, 1986, p29.

[4] Williamson. Radial Line/Coaxial Line Stepped Junction. IEEE Trans Microwave Theory Tech, Vol MTT-33 No1, January 1985, p56.

[5] 邸英杰,章日荣. 同轴-径向波导接头分析与设计. 电波科学学报.

(责任编辑:宋金宝)