AVS 变长解码的 DSP 优化

于新涛[1],张宏波[1],马 磊[1,2]
YU Xin-tao[1],ZHANG Hong-bo[1],MA Lei[1,2]

1.山东大学 信息科学与工程学院,济南 250100
2.山大鲁能信息科技有限公司,济南 250100
1.School of Information Science and Engineering,Shandong University,Ji'nan 250100,China
2.Shanda Luneng Information Technology Co.,Ltd,Ji'nan 250100,China
E-mail:yxt433@126.com

YU Xin-tao,ZHANG Hong-bo,MA Lei.Optimization of AVS variable length decoding on DSP.Computer Engineering and Applications,2008,44(20):93-95.

Abstract: The part of AVS variable length decoding is optimized based on DM642.The accessing method is changed according to the property of Exp-Golomb code.By altering the loop structure and assigning the registers appropriately,the compiler can process an effective software pipeline.The executive time is deduced by 30% after optimization,which satisfies the requirements of real-time decoding of standard definition video sequences.

Key words: Audio Video coding Standard(AVS);variable length decoding;Exp-Golomb;software pipeline

摘 要:在 DM642 上对 AVS 变长解码部分进行了优化。针对指数哥伦布码的特性调整了存储器访问的方式,并通过对循环结构的调整及寄存器资源的合理分配使编译器能够进行高效的软件流水编排。经过优化后代码执行时间降低了 70%以上,达到了标清尺寸实时解码要求。

关键词:AVS;变长解码;指数哥伦布码;软件流水
DOI:10.3778/j.issn.1002-8331.2008.20.029 **文章编号:**1002-8331(2008)20-0093-03 **文献标识码:**A **中图分类号:**TP301.6

1 引言

AVS 标准(Audio Video coding Standard)是我国制定的拥有自主知识产权的音视频编解码标准,在 IPTV、视频监控等领域有着广泛的应用。AVS 编码效率是 MPEG-2 的 2~3 倍,优于国际上的 MPEG-4 AVC/H.264,而且方案简洁,复杂度低于 H.264,在高清部分处于领先地位。AVS 基本类标准于 2006 年 2 月正式成为中国国家标准[1],目前处于产业化阶段,其中 AVS 标清/高清芯片的研发与实现是其中重要的工作。

本文在 TI 公司的 DM642 平台上对 AVS 解码器中的变长解码部分进行了优化,并且给出了 Emulator 上的仿真结果。

2 AVS 变长解码的主要流程

在 AVS 变长解码[2]中,采用了基于指数哥伦布的自适应变长编码技术。对于宏块编码数据,解码器首先用 0 阶、1 阶、2 阶或 3 阶指数哥伦布码进行 ce(v)解析,然后根据所得语法元素查表得到量化系数值(Level)和游程(Run)。

2.1 指数哥伦布码及 ce(v)语法元素解析

指数哥伦布码的比特串分为前缀和后缀两部分。前缀由 m 个"0"和 1 个分隔符"1"组成,后缀有 $m+k$ 个比特,记作 databits。其中 k 为阶数。对于 k 阶指数哥伦布码,根据下式进行解析:

$$CodeNum=2^{m+k}-2^{k}+databits \quad (1)$$

在 AVS 标准中规定了 19 个变长码表[1],不同的码表决定了 ce(v)所用的指数哥伦布码的阶数。

2.2 变长码解码

由 *CodeNum* 做索引生成 *Level*、*Run* 的过程如图 1 所示。

从码流中读取并解析 *CodeNum* 得到变换系数(*trans_coefficient*)。如果变换系数的值小于 59,则以该系数为索引查表求得 *Level*、*Run* 的值并分别存入 *buff_level*、*buff_run* 数组中。

否则,需要解析下一个 *CodeNum*,得到转逸系数差值(*escape_level_diff*)。

由变换系数确定 *Run* 的值:

$$Run=\frac{trans\_coeff-59}{2} \quad (2)$$

Level 的模值由转逸系数和 *RefAbsLevel* 确定,*Level* 的符号由变换系数的奇偶决定:

$$level=\begin{cases}-(RefAbsLevel+escape\_level\_diff)(trans\_coeff\text{为奇数})\\RefAbsLevel+escape\_level\_diff(trans\_coeff\text{为偶数})\end{cases} \quad (3)$$

作者简介:于新涛(1981-),男,硕士研究生,主要研究领域为视频编解码与 DSP 应用;张宏波(1984-),男,硕士研究生,主要研究领域为视频编解码;马磊(1960-),男,教授,研究生导师,主要研究领域为信号处理及嵌入式开发。

收稿日期:2007-10-09 **修回日期:**2008-01-21

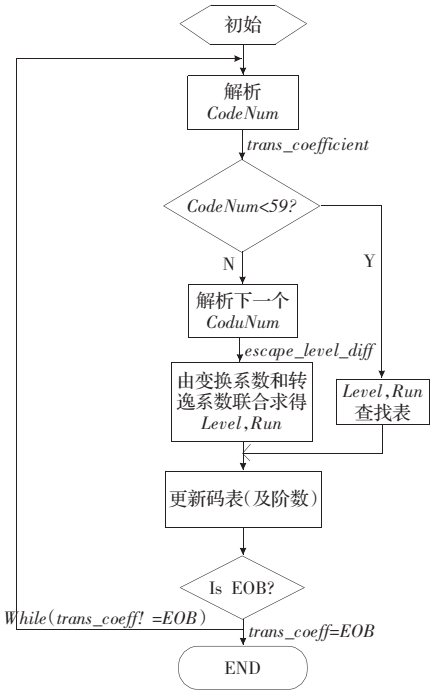

图 1 宏块系数变长码解码的流程图

其中 *RefAbsLevel* 与式(2)中求得的 *Run* 有关,当 *Run* 大于 *MaxRun* 时,*RefAbsLevel* 等于 1; 否则以 *Run* 为索引查表求得 *RefAbsLevel*。

保存 *Level*、*Run* 之后,更新码表的表号,继续解析下一个块系数,直到变换系数值等于 EOB 时跳出循环。

3 AVS 变长解码的 DSP 实现与优化

3.1 DSP 实现平台

变长解码所采用的 DSP 平台为 TI 公司的 TMS320DM642/C64x 系列芯片,具有第二代高性能的超长指令字结构 VelociTI.2, 在 8 个功能单元里扩展了 88 条新的指令以增强其在视频/图像应用中的性能,并提高了视频处理的并行性。

将 AVS 工作组提供的参考代码 rm52j 移植到 DM642 上,并通过以太网口传送解码输出图像。在优化的过程中,主要是发挥 C64x 芯片的并行性特点对数据进行处理, 并且通过对数据存储结构以及程序架构的调整,使编译器更好地进行软件流水,以提高解码的效率。

3.2 DSP 优化[3]

3.2.1 指数哥伦布解码的优化

对指数哥伦布解码部分的优化主要基于两点:(1)存储器访问的优化;(2)使用内联指令(intrinsic)。

在 AVS 工作组参考代码中, 指数哥伦布解码部分的码流读取是逐比特进行的。而对 C6000 的存储器进行访问是很费时的,要提高数据处理效率,应使 1 条 Load/Store 指令能够访问多个数据。所以,在 DSP 程序中,对变长解码部分的码流读取部分进行了改进。对于每个要解码的系数, 一次读入 32 bit 到寄存器中进行处理(Little Edian 模式下还要进行 4 字节倒序的操作)。使用内联指令 _lmbd(0x1,buff_bits)来判断前缀"0"的个数,然后通过移位操作读取 databits,按照公式(1)进行指数哥伦布解析。使用内联指令加以改编,可以大幅度减少循环体执行的指令周期数。

3.2.2 变长解码部分软件流水的优化

软件流水线技术用来对一个循环结构的指令进行调度安排,使之成为多重迭代循环并行执行。在编译代码时,可以选择编译器的-o2 或-o3 选项, 使编译器将根据程序尽可能地安排软件流水线。

使用软件流水线还有下面几点限制:

(1)循环结构不能包含代码调用。

(2)循环结构不能包含 break,if 语句不能嵌套,条件代码应当尽量简单。

(3)循环结构中不要包含改变循环计数器的代码。

(4)循环体代码不能过长,因为寄存器的数量有限,应该分解为多个循环。

为了使编译器能够实现高效的软件流水编排,本文做了以下优化:

(1)把指数哥伦布解码部分的函数调用内联函数代替直接写入主函数中。

(2)减少冗余计算。对于帧内、帧间或亮度块的系数,提取其循环的公共操作如计算哥伦布阶数等, 放到循环外部执行。一方面避免了重复的运算,另一方面减少了循环内部的代码长度,有利于软件流水的编排。

(3)改变程序结构,拆分 for 循环。

由于亮度系数的复杂性以及块系数个数的不确定性,对变长解码流程中的循环往往难以进行流水编排。造成流水失败的原因有以下几种:

图 1 所示的变长码解码的循环中,判断变换系数是否小于 59 处出现了 if…else…分支语句。由于每个跳转指令有 5 个延迟间隙,使得程序执行时间延长;另外,循环内跳转也使软件流水受到阻塞。

同时,由于循环中的代码过长,而且在码表切换等处多次用到条件寄存器,使得循环体内寄存器不够分配,?从而导致系统编译器无法实现循环的 pipeline。

对于大部分的 AVS 编码码流, 其宏块残差系数中变换系数小于 59 的比例占了 90%以上,如表 1 所示。

表 1 不同码流中亮度块(帧内、帧间)系数<59 的概率统计

测试序列	Intra		Inter	
	<59	≥59	<59	≥59
football	94.21%	5.79%	93.77%	6.23%
cctv_live	97.24%	2.76%	96.65%	3.35%
mobile	96.86%	3.14%	96.90%	3.10%
Movie_1800k	98.52%	1.48%	97.57%	2.43%

当变换系数大于等于 59 时, 计算其对应的 *Level*、*Run* 以及更新码表的操作都要比小于 59 时的情况复杂得多。根据这一特点,本文把 *trans_coeff*<59 的情况单独拆分出来,即从复杂循环中拆分出一个较小的简单循环。用伪代码表示如下:

```
do
{
  for(;trans_coeff<59 && !=EOB;)
  {
    Loop:trans_coeff
  }
  Processing:escape_level_diff
}while(trans_coeff!=EOB)
```

(4)用逻辑判断语句替代 if…else…语句,减少跳转语句。把

if 结构用条件运算表达式进行改写,最后使循环可以 pipeline。

(5)减少中间变量。避免对存储器的冗余存取,减少寄存器的使用数目,利于对循环做出更进一步的优化。

(6)利用数据打包、解包减少存取时间。*Level*、*Run* 分别保存在两个数组中,增加了寄存器的存储时间。现将二者打包存放在一个 int32 型的数组中,在反扫描反量化时再进行解包。经测试,存储所用时间有所降低,而后续进程中(反扫描部分)读取所耗费的时间基本不变。

经过上述调整,可以使编译器对占变换系数绝大部分比例的小于 59 循环内进行软件流水,提高了解码效率。

4　实验结果及结论

本文在 CCS2.21 环境下对 AVS 变长解码部分的程序进行优化,并且在 XDS510 Emulator 上进行了仿真,测试序列选用了从 1 Mb/s 到 15 Mb/s 的不同码率的标清尺寸 AVS 码流。

表 2 记录了 football 序列(704×480,码率 10.88 Mb/s,50 帧)各优化阶段的变长解码函数占用时间。

表 2　各阶段优化后的变长解码部分函数时间

STEP	CPU 时钟周期/Clk	函数时间/ms
1.使用-O1 优化	406,252,284	677.1
2.指数哥伦布优化	300,578,625	500.9
3.使用-O3 编译选项	260,613,528	434.4
4.使循环 pipeline	196,977,493	328.3
5.其他软件流水优化	119,342,005	198.9

通过 3.2.1 中的方法对哥伦布解码部分的优化(STEP-2),使得解码器对变长编码数据的处理效率大为提高。再经过 3.2.2 中的(1)(2)处理,使循环满足使用软件流水的条件,对程序进行 O3 级别的优化(STEP-3)。此时观察.asm 反馈信息,得知编译器多次尝试软件流水均未成功。通过 3.2.2 中(3)(4)的方法调整循环结构,可以实现流水编排(STEP-4)。(5)(6)进一步降低了循环体内部的数据存取时间,使编译器做出了更好的优化。

变长解码的优化为解码器最终达到实时解码奠定了基础。对于当前网络带宽条件下所能传送的大部分码流,变长解码占解码理想总时间的比例都能够控制在 2%以下, 达到了预期的目标。同时,对于更高码率的码流,仍需要进一步地优化,以适应未来 IPTV 技术的发展需求。

表 3　不同码率测试序列变长解码所占比例

测试序列	NSCC_fld	Movie	CCTV_live	Football
帧数	80	88	86	50
尺寸	720×576	720×576	720×576	702×480
码率/(Mb/s)	1.47	1.76	1.39	10.88
解码达实时(30 f/s)所需总时间/ms	2 667	2 933	2 867	1 667
变长解码时间/ms	79.5	121.9	91.6	198.9

参考文献:

[1] AVS 工作组.GB/T200090.2-2006 信息技术——先进音视频编码 第 2 部分:视频[S].2006.

[2] 陈光法,姚立敏,虞露.AVS 熵解码与 DSP 实现[J].电视技术,2004(10):43-46.

[3] 李方慧.TMS320C6000 系列 DSPs 原理与应用[M].2 版.北京:电子工业出版社,2005.

[4] 杨阳.基于 OMAP 平台的 AVS 解码实现[J].电子设计应用,2006(4):90-93.

[5] 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.

(上接50 页)

表 3　仿真结果

仿真	最优解	达到最优解						未达到最优解			
		迭代次数		用时/s		达到最优解次数		最差次优解		最好次优解	
		IQEA	HQA	IQEA	HQA	IQEA	HQA	IQEA	HQA	IQEA	HQA
Car1	7 038	8.70	7.80	1.24	0.23	20	20	–	–	–	–
Car2	7 166	94.53	9.95	17.75	0.35	17	20	7 376	–	7 226	–
Car3	7 312	114.00	23.93	20.20	0.83	2	15	7 491	7 399	7 328	7 366
Car4	8 003	86.72	19.80	19.95	0.76	18	20	8 008	–	8 008	–
Car5	7 720	90.29	10.89	14.30	0.31	7	19	7 768	7 727	7 732	7 727
Car6	8 505	65.23	8.79	8.22	0.22	13	19	8 570	8 570	8 570	8 570
Car7	6 590	15.16	5.95	1.50	0.12	19	20	6 645	–	6 645	–
Car8	8 366	35.55	11.42	4.06	0.29	20	19	–	8 424	–	8 424

项指标上突出,就得在别的指标上付出代价,相对而言,HQA 性能最好,各项指标都比较稳定。

5　结束语

采用具有并行性的量子比特的编码方式,借鉴微粒群算法的搜索特性,引进经典进化计算的优化理念,合成了一种新型的自适应算法——混合量子算法。从原理上说明了该算法在寻优方面的可用性,以及在组合优化问题中的实用性,并将其应用于置换 flow shop 问题,其测试结果表明该算法具有种群小、迭代次数少、用时少等优点,并且克服了 QEA 的不足之处,具有实用价值。

参考文献:

[1] 陆晓亮,胡苏太.量子计算机的发展现状和趋势[J].高性能计算发展与应用,2006(1):7-11.

[2] Kuk-Hyun Han,Jong-Hwan Kim.Genetic quantum algorithm and its application to combinatorial optimization problem[C]//Proceedings of the 2000 IEEE Congress on Evolutionary Computation,2000:1354-1360.

[3] 麦克维克斯.现代启发式方法[M].曹宏庆,译.北京:中国水利水电出版社,2003:147-149.

[4] 云庆夏.进化算法[M].北京:冶金工业出版社,2000:148-151.

[5] 杨淑媛,刘芳,焦李成.量子进化策略[J].电子学报,2001,29(12):1873-1877.

[6] 曾建潮,介婧,崔志华.微粒群算法[M].北京:科学出版社,2004:3-4.