

文章编号: 1000-5013(2012)05-0499-04

基于四叉树的嵌入式平台 Huffman 解码优化

鲁云飞¹, 何明华²

(1. 福州大学 电气工程与自动化学院, 福建 福州 350108;
2. 福州大学 物理与信息工程学院, 福建 福州 350108)

摘要: 考虑到嵌入式设备资源的有限性,提出一种基于四叉树的 Huffman 解码优化算法. 解码过程中,先将 Huffman 码表表示成四叉树结构,据此重建为一维数组,并充分利用数值计算代替判断与跳转操作. 为测试本算法解码性能,将其应用于嵌入式 MP3 实时解码中,结果表明本算法内存损耗小,解码速率快,算法复杂度低,相比于其他优化算法,更适合应用于嵌入式设备中.

关键词: 嵌入式; 四叉树; Huffman 解码; 解码优化; MP3 音频

中图分类号: TP 391 **文献标志码:** A

Huffman 编码是一种高效的无失真信源熵编码,作为无损信源压缩的重要方法,该编码算法在文本、音频、图像、视频等多种多媒体数据压缩中得到广泛应用^[1-2]. Huffman 解码的实现分硬件实现和软件实现. 其中,硬件实现需采用专用集成电路或芯片,解码效率虽高,但解码成本也高,系统硬件依赖性强,解码灵活度小;而软件实现则依靠处理器采用软件编程为手段,虽效率较前者稍低,但系统开发成本小,灵活度大. 传统的 Huffman 解码软件实现算法主要有顺序搜索法、二值树法、查表法等^[1-5]. 但这些算法分别存在搜索时间长、解码速率低及内存损耗大等问题,因此,不少学者以此为基础,提出了一系列的改进算法. Hashemian 及董培良等(以下简称 Hashemian 等算法)提出每次从码流中读取 r 比特码元,减少了解码效率对码长的依赖^[2-4]; Aggarwal 等^[5] 基于特征分类方法提出解码思想,解决了解码效率对码长的依赖问题; Lee 等^[6] 提出了改进二值解码算法(以下简称 Lee 算法),用数值计算代替每步解码时的判断与跳转操作,其解码效率在一定程度上得到提高. 然而,随着生活质量的提高与生活节奏的加快,人们对娱乐设备的要求也越来越高,即要求设备便携的同时还能提供高质量与多功能的服务^[6-8]. 这就需要在不影响功能效果的前提下尽量减小单个功能模块的内存损耗,充分利用嵌入式设备的有限存储空间和能源供给. 基于此,本文结合 Hashemian 等算法中的多位操作及 Lee 算法中的数值计算等优点,提出了基于四叉树的 Huffman 解码优化算法.

1 基于四叉树的 Huffman 解码算法

1.1 Huffman 码表四叉树形结构的建立

常规的多媒体数据解码时,其 Huffman 码多采用二值 Huffman 树表示. 图 1 为二值 Huffman 树,即将表 1 中的 Huffman 码字用二值树表示时的树形图. 为方便算法阐述,表 1 所示的码表来源于 MPEG1 音频数据 layer 3 的部分标准 Huffman 码表^[9]. 如图 1 所示,从根节点出发,每次从码流中读取 1 bit 的码元,根据这个数据的数值进行跳转,若为 1 则跳转到右节点,反之则跳转到左节点;以此类推,直到跳转至二叉树的叶节点,即可得到码字所代表的字符^[10].

从图 1 可看到:二值 Huffman 树对码流一位一位进行操作,其码字的搜索次数多,搜索深度广,因而解码效率低. 为提高 Huffman 解码过程中对码字的搜索速度,采用 Hashemian 等算法中的多位操作

收稿日期: 2012-01-22

通信作者: 何明华(1971-),男,教授,博士生导师,主要从事嵌入式系统与系统级芯片设计的研究. E-mail: mhhe@fzu.edu.cn.

基金项目: 福建省科技重大专项(2009HZ0007-1)

表 1 MP3 部分标准 Huffman 码表

Tab.1 Part of the MP3 Huffman code table

码字	1	0100	0101	0110	0111	00100	00101
字符	S ₁	S ₂	S ₃	S ₄	S ₅	S ₆	S ₇

思想,以缩短解码过程中对码字的搜索深度.然而,由于 Huffman 码为单义码,且为前缀码,码中无任何码字为其他码字的前缀,所以会出现相同前缀的码字解出相同结果的情况.为尽量减小这种情况所带来的大量内存冗余,提高嵌入式设备内存使用效率,本算法一次性对码流的 2 位码元进行操作,将标准 Huffman 码表重建成四叉树结构,如图 2 所示.图 2 中:根节点表示 Huffman 树的开始;叶子节点表示某个码字的结束,其处存储着该码字所代表的字符;中间节点对应码字的码元;空节点表示该节点为无意义节点.

从图 2 可看出:该四叉码树同样由父节点及子节点组成,每个节点对应 2 个码元,且每个节点下面最多有 4 个子节点,从左到右分别代表读入的码字为“00”~“11”的搜索路径.因而解码时,每经过四叉 Huffman 树一个节点,相当于从码流中读取 2 个比特码元进行 Huffman 码字匹配,故理论上对每个 Huffman 码字解码的搜索深度及搜索时间最多可减少为原二叉树的 1/2.

由于一次性只对码流的 2 位码元进行操作,当码长为 2 的整数倍时,最后一组剩余的码元个数为 2,因而该组码元即与四叉树中节点是一一对应关系,如图 2 中的 S₂~S₅;当码长不是 2 的整数倍时,最后一组剩余的码元个数为 1,此时该组码元对应两个具有相同前缀的节点,即如待解的 Huffman 码字长度为 L,M=L mod N(N 为单次处理码元的位数),则需要填补(2^M-1)个节点,各节点数值大小从左到右,从上到下依次递增“01”,如图 2 中的 S₁,S₆及 S₇.从图 2 可知:本四叉 Huffman 树的冗余度较小.

1.2 四叉 Huffman 码树信息的存储

四叉 Huffman 树重建完后,还需根据其结构将所带数据信息存储起来.为进一步提高解码速度,减少二值 Huffman 解码算法中大量“比较跳转”操作所消耗的时间,可采用 Lee 等数值运算的方法,将图 2 所示的四叉树信息存储成一维数组形式以供解码时使用,所建一维数组 Array[]如表 2 所示.

表 2 所建一维数组

Tab.2 Single dimensional array constructed

参数	层数													
	1				2				3					
index	0	1	2	3	6	8	9	10	11	12	13	14	15	
F	0	0	1	1	0	1	1	1	1	1	1	1	1	
el	2	2	1	1	2	2	2	2	2	1	1	1	1	
rv	4	7	S ₁	S ₁	6	S ₂	S ₃	S ₄	S ₅	S ₆	S ₆	S ₇	S ₇	

在重建数组时,将四叉树中的节点按照从上到下、从左到右的顺序排列后,依次分配给数组中的元素,因而,数组元素的索引值 index 随着节点数目的增加而增加.除此之外,数组 Array[]中的每个节点包含 3 个变量:F,el,rv.其中:F 用以表示该节点是否为叶节点,若 F 为 0,则该节点为中间节点,否则为叶子节点;el 用于表示中间节点中的有效码元个数,且 el 的取值为 1~2;rv 用于表示中间节点到其子节点 0 的偏移距离.如表 2 所示,数组中的每个元素代表 1 个节点,且第 1 行的数据表明了节点在四叉树中所属的层数,表中前 4 个元素对应树的第 1 层节点,后 5 个对应树的第 2 层,最后 4 个则对应树的第 3 层.以四叉树中第 2 层左起第 3 个节点为例,其对应表 2 中索引 Index 值为 6 的元素,且其为中间元素,有效码元个数为 2,到其 0 子节点的偏移距离为 6,即索引值为 12 处的元素.

1.3 码流的 Huffman 解码

利用重建好的 Huffman 码四叉树型结构及其对应的一维数组,对读入的码流进行相应 Huffman

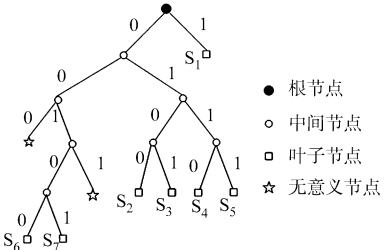


图 1 二值 Huffman 树

Fig.1 Binary Huffman tree

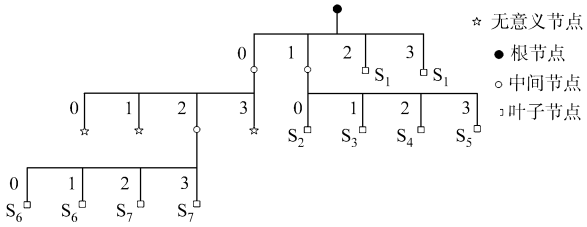


图 2 四叉 Huffman 树

Fig.2 Quad Huffman tree

解码,其流程如图 3 所示. 具体解码过程如下:

- 1) 先初始化查找表的基地址;
- 2) 从输入的码流中读取 2 bit 码元,并将新读入码元值 new_2digit 与初始基地址(即初始索引值 Index)之和作为新索引地址值 Index,从而得到当前查找节点的地址;
- 3) 从数组中该节点的 F 值判断该节点是否为叶子节点,若是,则终止本次搜索,并将该节点数组中存储的码字 Array[\cdot].rv 读取出来;否则,将该节点数组中存储的偏移值 rv 读出来,并与过程 2) 所得新索引值 Index 相加,再转到步骤 2) 继续搜索,直至遇到叶子节点,则该分支码字搜索结束;
- 4) 依次递推搜索直至将输入码流解码完毕;
- 5) 将搜索出来的码字依次排列起来,即为输入 Huffman 码流所对应解码后的码字.

当对一输入 Huffman 码流解码连续出现多个中间节点时,可将图 3 所示流程中计算下一分支地址索引值的过程直观地归纳为: $index := index + Array[index].rv + new_2digit$. 以防止索引值多次叠加时出错,以输入 Huffman 码流“010100101”解码为例,其解码结果为“S₃S₇”,在码字“S₇”所对应的码元解码时就连续出现对 3 个中间节点操作的情况.

2 实验结果及算法性能分析

根据以上四叉树 Huffman 解码优化算法原理,以软件编程方式实现其解码过程,并将其应用到嵌入式 MP3 解码中,以测试该算法解码性能. 为便于各算法性能的比较,分别以 W1, W2, W3 表示 Lee 算法(改进二值法)、Hashemian 等算法(以一次性操作 3 位即八叉树来代替)和本文提出的基于四叉树的 Huffman 解码算法. 以 5 个不同码率的 MP3 文件作为测试文件,测试表明算法 W1, W2, W3 消耗的内存分别为 0.62, 1.23, 0.75 KB,而所需解码时间以及解码指令数情况如表 3 所示.

由测试结果可知:若以 W1 算法的 Huffman 解码算法为基准, W3 算法内存损耗增加少,约为 W1 算法的 21%,而 W2 算法所耗内存增量,约为 W1 算法的 98%,换言之, W2 算法带来了很大的内存冗余. 根据表 3 数据可知:当对不同码率的文件进行测试时,所需的 Huffman 解码时间及指令数都各有不同. 就解码时间而言,相比于 W1 算法, W3 算法节省了约为 47.5%,而 W2 算法节省的略多于 W3 算法,即约为 W1 算法的 62.6%,也即 W2 算法的解码速度略快于 W3 算法;就解码所需指令数而言,相比于 W1 算法, W3 算法总指令数约为 W1 算法的 57%,而 W2 算法约为 W1 算法的 44%,即 W3 算法的指令数略多于 W2 算法.

表 3 3 种 Huffman 解码算法解码时间及算法指令数

Tab. 3 Three different Huffman decode algorithm decoding time and instruction number

文件	码率/kbit · s ⁻¹	解码时间/ms			指令总数/10 ⁷		
		W1	W2	W3	W1	W2	W3
Test1. MP3	141	8 768	3 487	4 828	1.11	0.49	0.63
Test2. MP3	160	8 584	3 340	4 438	1.01	0.44	0.58
Test3. MP3	192	10 432	3 812	5 344	1.18	0.52	0.67
Test4. MP3	224	11 587	3 922	5 781	1.20	0.53	0.68
Test5. MP3	320	6 048	2 448	3 453	0.71	0.31	0.41

3 结论

Lee 的改进二值法虽内存损耗小,但解码所耗时间长,算法复杂度高,因而算法的解码实时性较差,不适合应用于嵌入式设备多媒体应用程序开发当中. Hashemian 等算法的 Huffman 解码效率较好,实

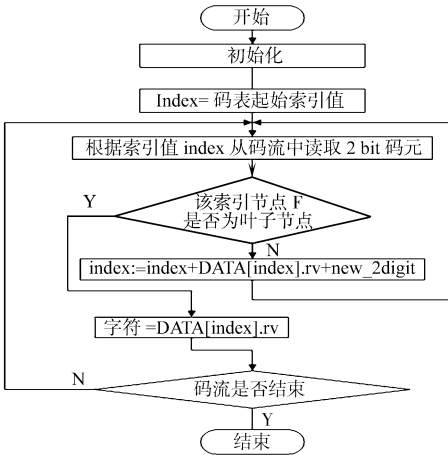


图 3 基于四叉树结构的 Huffman 解码流程
Fig. 3 Huffman decode flow base on quad-tree

时性较高,但其内存损耗大,因而对于内存有限的嵌入式开发并不合适. 所提出的基于四叉树的 Huffman 解码算法综合了以上两类算法的优点. 相比于 Lee 的改进二值法,在内存损耗增加不明显的情况下,显著提高了 Huffman 的解码性能;相比于 Hashemian 等算法,其解码性能虽有所降低,但却能明显改善系统的内存冗余,提高内存的利用率,符合嵌入式开发的实时性与内存低损耗性要求. 因此,可更灵活地移植到嵌入式环境下多种多媒体数据解码中,具有一定的普适性.

参考文献:

[1] 倪昕,王维东,刘鹏,等. 媒体处理器视频哈夫曼解码快速算法[J]. 浙江大学学报:工学版,2007,41(12):2036-2039.

[2] 董培良,俞日龙,廖天康,等. 一种快速霍夫曼解码算法及其软硬件实现[J]. 复旦学报:自然科学版,2002,41(2):165-169.

[3] HASHEMIAN R. Memory efficient and high speed search Huffman coding[J]. IEEE Trans on Communications, 1995,43(10):2576-2581.

[4] HASHEMIAN R. Condensed table of Huffman coding, a new approach to efficient decoding[J]. IEEE Transactions on Communications,2004,52(1):6-8.

[5] AGGARWAL M,NARAYAN A. Efficient huffman decoding[C]// International Conference on Image Processing. Vancouver:[s. n.],2000:936-939.

[6] LEE J S,JEONG J H,CHANG T G. An efficient method of Huffman decoding for MPEG-2 AAC and its performance analysis[J]. IEEE Trans on Speech and Audio Processing,2005,13(6):1206-1209.

[7] WANG Sung-wen,WU Ja-ling,CHUANG Shang-chih. Memory efficient hierarchical lookup tables for mass arbitrary-side growing huffman trees decoding[J]. IEEE Trans on Circuits and Systems for Video Technology,2008,18(10):1335-1346.

[8] PHAM H A,BUI V H,DINH-DUC A V. An adaptive huffman decoding algorithm for MP3 decoder[C]// Fifth IEEE International Symposium on Electronic Design, Test & Applications. Washington D C:IEEE Computer Society,2010:153-157.

[9] ISO/IEC. ISO/IEC 11172-3—1994 Information technology: Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s; Part 3: Audio[S]. Geneva:ISO/IEC,1994.

[10] 朱坤旺,傅文渊,凌朝东. 低功耗 H. 264 Baseline 解码 IP 核设计[J]. 华侨大学学报:自然科学版,2011,32(3):280-283.

Embedded Platform Huffman Optimization Decoding
Algorithm Base on Quad-Tree

LU Yun-fei¹, HE Ming-hua²

(1. College of Electric Engineering and Automation, Fuzhou University, Fuzhou 350108, China;
2. College of Physics and Telecommunication Engineering, Fuzhou University, Fuzhou 350108, China)

Abstract: Considering the limitation of embedded system resources, a Huffman decoding optimization algorithm based on the quad tree is proposed in this paper. In this process, the Huffman code table is expressed as quad tree structure at first, and according to which a one-dimensional array is reconstructed, then make full use of numerical calculation instead of judgment and jump operation. In order to test the decoding performance, the method is applied to the embedded real-time MP3 decoding. The results show that the algorithm memory loss is small, decoding speed is rapid and its complexity is low, compared to other optimization algorithms, this algorithm is more suitable for application in embedded devices.

Keywords: embedded; quad-tree; Huffman decoding; decoding optimization; MP3 audio