基于多级查找表的 VLD 设计及其状态机的优化 [1]

黎 文 李蜀雄 朱维乐

(电子科技大学电子技术系 成都 610054)

摘 要 采用多级查找表的 VLD 方法具有快速、节省存储器空间等特点，因而在各种数字视频和图像解码器中得到了广泛的应用. 本文详细地讨论了多级查找表及其状态机的设计和优化，并推导出了求 LUT 表项数和 FSM 状态数的具体公式，文中以 MPEG1 为数值样例，证明了该算法的正确性. 利用本文的结果，可以在硬件设计时充分地节约表项资源.

关键词 变长解码，有限状态机，MPEG1, 数字视频

中图号 TN941, O224

1 引 言

Huffman 编码是一种无失真的最佳不等长编码方法，它将短码字分配给出现概率大的符号，而长码字分配给出现概率小的符号. Huffman 码与游程码的结合是一种高效的近似最优的熵编码技术，因而在数字视频和图像处理中有着广泛的应用。而且，解 Huffman 码的变长解码器 (VLD) 是各种图像编解码标准如 H.261 , H.263 , JPEG , MPEG1 , MPEG2 的技术核心之一，对硬件设计更极为重要 [1-4] .

在做变长解码时，设 Huffman 码的比特数最长为 k , 则若每次查一比特，那么最多需查 k 次，使得解码速度太慢；而若每次查 k 比特，那么所需的 LUT 表 (Look-Up Table) 将太大，达到 2^k 个表项，极大地浪费了存储器空间和硬件资源. 为了解决这两个矛盾，近年提出了一种既快速又节约存储器空间的基于多级查找表的 VLD 方法 [1,2] . 本文根据文献 [1,2] 的 VLD 方法，详细地讨论了多级查找表及其状态机的设计和优化. 利用本文推导出的结果，我们可以很方便地设计出状态数和表项数均较优的多级查找表，从而在硬件设计中充分地节约资源，同时提高 VLD 的解码速度，降低大规模芯片的生产成本.

2 采用多级查找表的 VLD 方法

采用多级查找表的 VLD 技术是基于对码字进行排序 (ordering) 和分割 (partitioning) 、簇化 (clustering) 而实现的. 排序和簇化的原则是: (1) 出现概率大的码字的查表次数少，出现概率小的码字的查表次数多；(2) 对比特数大的码字的查找速度也能加快. 该原则可以通过对码字按给定的长度进行分割、簇化成含多个比特的比特组，解码时每次查找一个比特组来实现 [2] . 下面以一具体例子来简要说明多级 LUT 表的设计.

图 1 为一典型的 Huffman 码表所对应的二叉树，从图中我们可以看出：二叉树被相互之间距离为 L_m 分割线 x-x , y-y , z-z 分割成不同的具有单一根节点的子树 (a, b, c, d, e, f, g) , 每一子树称为一簇，容易证明每一簇的长度 $L \le L_m$. 现在我们给每一簇分配一个 LUT 表，表中的每一项代表一个叶节点 (码字) 或下一级子树的根节点. 各级 LUT 表共同组成一个总的 LUT 表. 在 LUT 表中，每一表项包含 3 个信息：标志位；耗用比特数 / 待查比特数；游程码 / 下一级表的起始地址. 若标志位为 0 , 则表示该项为一叶节点，第 2 个信息为解该码字时本级 LUT 表所耗用的比特数，第 3 个信息则为所解出的具体的游程码；若

[1] 1998-11-11 收到，2000-01-14 定稿

标志位为 1，则表示该项为一下一级子树的根节点，当前仍然没有查到具体码字，需转到该子树所对应的下一级 LUT 表中继续查找，第 2 个信息为解下一级子树所需要的比特数 L，即下一次要查找的比特数，第 3 个信息则为下一级 LUT 表的入口地址。

关于采用多级查找表的 VLD 的基本构架可查参考文献 [1,2]。表 1 列出了图 1 所对应的多级查找表。

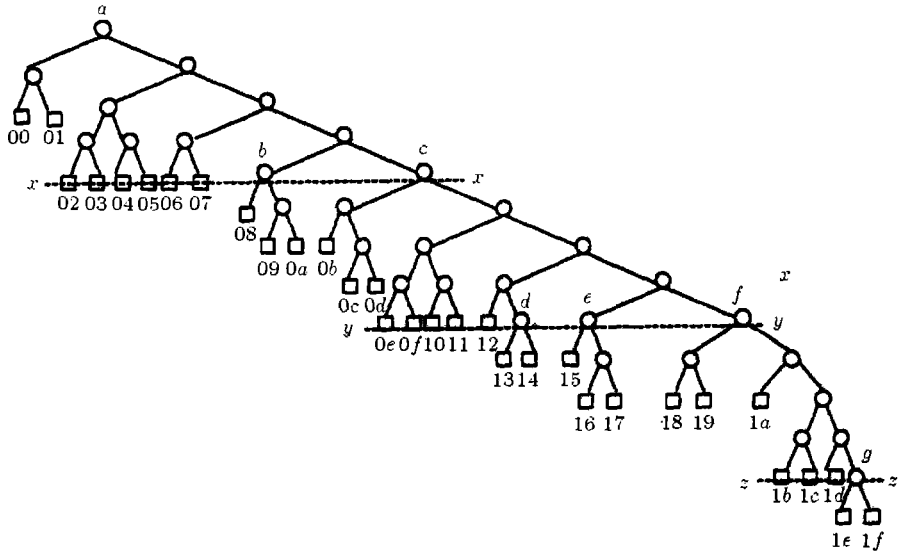

图 1 Huffman 码二叉树

表 1 Huffman LUT 表

0/01 00	0/01 00	0/01 00	0/01 00	0/01 01	0/01 01	0/01 01	0/01 01
0/11 02	0/11 03	0/11 04	0/11 05	0/11 06	0/11 07	0/01 10	0/11 14
0/00 08	0/00 08	0/01 09	0/01 0a	0/01 0b	0/01 0b	0/01 0b	0/01 0b
0/10 0c	0/10 0c	0/10 0d	0/10 0d	0/11 0e	0/11 0f	0/11 10	0/11 11
0/11 12	1/00 24	1/01 26	1/11 2a	0/00 13	0/00 14	0/00 15	0/00 15
0/01 16	0/01 17	0/01 18	0/01 18	0/01 18	0/01 18	0/01 19	0/01 19
0/01 19	0/01 19	0/01 1a	0/01 1a	0/01 1a	0/01 1a	0/11 1b	0/11 1c
0/11 1d	1/00 1e	1/00 1f					

3 LUT 表项的估计及简化

文献 [2] 对二叉树的任意两分割线之间的路径的长度 L_m 究竟应该选择什么样的值才能使 LUT 表的表项最少没有讨论，鉴于实际设计电路时该问题的重要性，下面将详细讨论这个问题。

首先计算 LUT 表的级数. 设 Huffman 码表中码字的最大比特数为 K ， $L_{\max}$ 表示 LUT 表的级数，则

$$L_{\max}=\frac{K+L_m-1}{L_m} \tag{1}$$

这里的除法为整除.

现在设 Huffman 码表中，长度为 k 比特的码字有 n_k 个， Huffman 码字的最大比特数为 K ，同时设 c_k 为一个 k 比特的码字. 则在比特数为 $i(k<i<K)$ 的二进制码字中，理论上有 2^{i-k} 个码字的根为 c_k ，根据异字头码的要求，这 2^{i-k} 个码字不能用作变长码. 故在比特数为 $i(i>2)$ 的二进制码字中，总共有 $\sum_{k=1}^{i-1}n_k2^{i-k}$ 个码字不能用，而可用码字的个数为

$$N_i=2^i-\sum_{k=1}^{i-1}n_k2^{i-k},\qquad 1<i\le K \tag{2}$$

设 N_i 中有 p 个码字是 Huffman 码表中的变长码，那么每一个码字至少应占一个表项，而另外 N_i-p 个码字可能是其余码字的根，若 i 为 L_m 的整数倍，则 VLD 遇到该码字时需要转移到下一级 LUT 表，故应占一个表项. 同时，我们认为当 $i>K$ 时， $N_i=0$.

我们先近似地认为 VLD 在查表时，每次耗用或待查的比特数均为 L_m ，则第 L 级 LUT 表的表项数为

$$\mathrm{Item}_L=N_{LL_m}+\sum_{i=1}^{L_m-1}2^in_{LL_m-i} \tag{3}$$

因为对于比特数为 n_{LL_m-i} 的码字，在 LUT 表中要占用 2^i 个表项. 如图 1 中码字 00 为 2bit，在表 1 中每次移入 4bit ，则以 00 为根的 4 个码字 0000 ， 0001 ， 0010 和 0011 要占用前 4 个表项.

由此，我们将每一级表的表项相加，即可得到总的表项数为

$$\mathrm{Item}=\sum_{L=1}^{L_{\max}}\mathrm{Item}_L \tag{4}$$

但是， (4) 式的值只是近似值. 若 Huffman 码表的设计严格地按照码字向树的一边靠，节点向树的另一边靠的原则，则在 VLD 的 LUT 表的设计中，从上一级表到下一级表的转换中，其待查比特数不一定是 L_m(如表 1 中的 0eH 项，其待查比特数为 2) ，此时则可节省 LUT 表项. 节省的项数推导如下：

设在第 L 级表项中，按照推 (3) 式时的假设，对于待查比特数为 $i(1\le i\le L_m)$ 的码字，每一个码字可以节约的表项为 $2^{L_m-i}-1$ 个.

因为在 $L-1$ 级 LUT 表的任一转移项中，其待查比特数若为 i ，则在第 L 级 LUT 表中可查 2^i 个码字，每个码字的比特数为 $(L-1)L_m+i$. 因而我们可由比特数为 $(L-1)L_m+i$ 的码字数，求出 $L-1$ 级 LUT 表中待查比特数为 i 的转移项数为

$$\frac{n_{(L-1)L_m+i}+2\varphi_{i-1}}{2^i}\qquad(1\le i\le L_m)$$

这里的除法为整除， φ_{i-1} 表示比特数为 $(L-1)L_m+i-1$ 的，不能节约表项的码字的个数.

对比特数为 $(L-1)L_m+i-1$，不能节约表项的码字，则应在比特数为 $(L-1)L_m+i$ 的码字中计算 2 倍，因为下一级码字中有 2 个是以上一级的某个码字为根．故 φ_i 的递推公式为

$$\begin{aligned}&\varphi_i=(n_{(L-1)L_m+i}+2\varphi_{i-1})\%2^i, \qquad 1\le i\le L_m \\ &\varphi_0=0\end{aligned} \tag{5}$$

这里，% 表示取余运算．故我们设在比特数为 $(L-1)L_m+i$ 的码字中，能节约表项的码字的个数为 $\psi_{(L-1)L_m+i}$，则

$$\psi_{(L-1)L_m+i}=\frac{n_{(L-1)L_m+i}+2\varphi_{i-1}}{2^i}\times 2^i, \qquad 1\le i\le L_m \tag{6}$$

这里除法为整除。

综上可得，比特数为 $(L-1)L_m+i$ 的码字总共可节约的表项数为

$$\xi_{(L-1)L_m+i}=\psi_{(L-1)L_m+i}(2^{L_m-i}-1), \qquad 1\le i\le L_m \tag{7}$$

则第 L 级 LUT 表能节约的表项数为

$$T_L=\sum_{i=1}^{L_m-1}\xi_{(L-1)L_m+i} \tag{8}$$

故 LUT 表总共能节约的表项数为

$$T=\sum_{L=2}^{L_{\max}}T_L \tag{9}$$

由 (4) 式和 (9) 式可得总的 LUT 表项数为

$$\text{Item}_{\text{sum}}=\text{Item}-T \tag{10}$$

根据 (10) 式，我们可以很方便地针对某一固定的 Huffman 码表，计算出不同的 L_m 所对应的 LUT 表项数，从而寻找到最优的 L_m，使得 LUT 的表项数最少．

下面我们计算一下在变长解码时的平均查表次数．设在 Huffman 码表中，长度为 k 比特的码字总共出现的概率为 $p_k(1\le p_k\le K)$，显然

$$p_1\ge p_2\ge\cdots p_i\ge\cdots\ge p_K \quad \text{且} \quad \sum_{k=1}^{K}p_k=1$$

则平均查表次数为

$$\text{Time}=\sum_{k=1}^{K}p_k(k/L_m+1)$$

这里的除法也表示整除．综合平均查找次数和 LUT 表项的长度，我们即可确定 VLD 时的最佳分割长度 L_m．

4 状态机的简化

在做 VLD 的硬件设计时，我们一般认为 LUT 表中的每一个表项均代表有限状态机 (FSM) 中的一个状态．但事实上，LUT 表中的某些状态是相同的，我们可以简化成一个状态．

考查表 1 中的前 4 项，我们发现每一表项中的 3 个信息均相同，都是耗用读入的 4 个比特的前 2 位，且解得的符号都是 00，故我们可以认为它们实际上是完成相同的功能，因而在逻辑上认为它们是同一个状态．事实上在 LUT 表中，若某一表项中的 3 个信息均与另一表项相同，则可认为这两个表项在逻辑上代表 FSM 的同一状态．

实际上，VLD 中 FSM 的状态由两部分构成：一部分是 Huffman 码代表的状态，另一部分为转移态．在 VLD 中，每一个 Huffman 码均代表一个状态，则 VLD 的总的状态数计算如下：

$$\mathrm{State} = \sum_{k=1}^{K} n_k + \sum_{L=1}^{L_{\max}-1} \mathrm{Jump}_L \tag{11}$$

其中 n_k 的定义与第 3 节相同，Jump_L 为第 L 级 LUT 表的转移态的个数，

$$\mathrm{Jump}_L = N_{LL_m} - n_{LL_m}$$

其中 N_{LL_m} 的表达式为 (2) 式．

5 实验结果及分析

根据本文的分析，我们在计算机上对 (10) 式和 (11) 式进行了仿真，首先针对本文第 1 节的图 1 和 MPEG1 中的 DCT 系数变长码表 (文献 [4] 之表 B.5c) 统计出不同的比特数所对应的码字个数 (见表 2)，然后分别计算出不同的 L_m 所决定的 $\mathrm{Item_{sun}}$ 和 State 值，结果如表 3 所示：

表 2　比特数及码字个数统计表

K	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
本文表 1 的 n_k	2	0	6	1	3	2	5	3	5	0	3	2			
MPEG1 的 n_k	2	1	2	3	5	4	8	0	8	0	16	16	16	16	16

表 3　LUT 表项数和 VLD 状态数表

L_m	2	3	4	5	6	7	8
本文表 1 的 $\mathrm{Item_{sun}}$	46	50	60	78	136	200	294
MPEG1 的 $\mathrm{Item_{sun}}$	186	166	166	176	204	304	536
本文表 1 的 State	44	43	38	36	36	36	35
MPEG1 的 State	168	161	137	142	134	135	117

从表 3 可以看出：当 $L_m = 4$ 时，对 Huffman 码图 1 计算出的 LUT 表项数为 60，状态数为 38，完全与表 1 的项数及状态数相符合，从而证明了 (10) 式和 (11) 式的正确性．

同时，我们可以看到当 L_m 等于 4 或 3 时，LUT 表项最少，而当 L_m 小于 3 和大于 4 时，LUT 表项数将会增大．造成这种现象的原因是：当 L_m 太小时，LUT 表中的转移项将随之增多；而当 L_m 太大时，LUT 表中信息相同的状态将随之增多．以上这两个原因均可导致 LUT 表项的增加．

我们在表 4 中列出了 $L_m = 4$ 时 MPEG1 中 DCT 系数 Huffman 码表的多级 LUT，表中第 3 个信息的前 2 位表示零游程，后 2 位表示电平，从表 4 中可以看到该 LUT 的表项为 158 项，而表 3 中的值为 166，这是因为 MPEG1 中的 Huffman 码字中，没有用到 0000 0000 0000 0000 码字或以它为根的码字，从而造成了计算上的误差，如果用上该码字，则可使表项降到 150 项。

表 4 MPEG1 的 DCT 系数变长码表的多级 LUT

1/11	1/01	1/11	1/00	0/11	0/10	0/10	0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/01
10	20	24	34	0002	0201	0101	0101	EOB	EOB	EOB	EOB	0001	0001	0001	0001
1/11	1/11	1/01	1/01	0/01	0/01	0/01	0/01	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10
36	46	56	5A	ESC	ESC	ESC	ESC	0202	0202	0901	0901	0004	0004	0801	0801
1/01	1/01	1/01	1/01	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/00	0/00	0/00	0/00
0701	0601	0102	0501	1301	0006	1201	1101	0302	0103	0005	1001	0003	0003	0003	0003
1/00	1/00	1/00	1/00	0/00	0/00	ERR	1/11	1/10	1/10	1/01	1/01	0/01	1/01	1/00	1/00
0003	0003	0003	0003	0401	0301		5E	6E	76	7E	82	86	8A	8E	90
1/00	1/00	1/00	1/00	0/00	1/00	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11
92	94	96	98	9A	9C	0011	0802	0403	0010	0204	0702	2101	2001	0009	1901
0/11	0/11	0/01	0/11	0/11	0/01	0/01	0/01	0/10	0/10	0/10	0/10	0/10	0/10	0/11	0/11
1801	0105	0303	0008	0602	1701	1601	0502	0007	0203	0104	1501	1401	0402	0118	0117
0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11	0/00	0/00	0/10	0/10
0116	0115	0603	1602	1502	1402	1302	1202	1102	3101	3001	2901	2801	2701	0040	0039
0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/01	0/01
0038	0037	0036	0035	0034	0033	0032	0114	0113	0112	0111	0110	0109	0108	0031	0030
0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/01	0/00	0/00
0029	0028	0027	0026	0025	0024	0023	0022	0021	0020	0019	0018	0017	0016	1002	0902
0/00	0/00	0/00	0/00	0/00	0/00	0/00	0/00	0/00	0/00	0/00	0/00	0/00	0/00		
0503	0304	0205	0107	0106	0015	0014	0013	0012	2601	2501	2401	2301	2201		

从表 3 和以上分析可以计算出：最佳 L_m 要比其他 L_m 至少节约 6% 的表项数，而相对于普通方法所需要的 2^{13} 或 2^{16} 个表项，更是节约了 $2^7 = 128$ 倍以上。

6 小 结

通过本文分析，我们利用 (10) 式和 (11) 式可以计算出不同的 L_m 所对应的 LUT 表项数和 VLD 的状态数，从而可在两者之间寻找到最佳的 L_m，使得 LUT 表项数和 VLD 的状态数均达到较为满意的值，充分节约硬件设计时的资源，最终降低大规模芯片的生产成本。我们同时发现，当 Huffman 码树的叶节点向一边靠，根节点向另一边靠，并且利用完码树中的每一级的全部有用的节点时，设计出的多级查找表的表项数将最少。利用以上观点，我们可以优化 Huffman 码表的设计，达到节约多级 LUT 表的目的。本文所述方法已成功地应用于我校所承担的 MPEG2 解码芯片优化设计的项目中，并达到了较为理想的效果。

参 考 文 献

[1] R. Hashemian, High speed search and memory efficient Huffman coding, IEEE Inter. Symp. Circuit Syst., Chicago, May 3-6, 1993, 287–290.

[2] R. Hashemian, Design and hardware implementation of a memory efficient Huffman decoding, IEEE Trans. on Consumer Elec., 1994, 40(3), 345–352.

[3] S. B. Choi, M. B. Lee, High speed pattern matching for a fast Huffman decoder, IEEE Trans. on Consumer Elec., 1995, 41(1), 97–103.

[4] ISO/IEC JTCI/SC29/WG11/ISO CD 11172-2, November, 1991.

THE OPTIMIZATION FOR VLD AND ITS STATE-MACHINE BASED ON MULTI-LEVEL LUT

Li Wen　Li Shuxiong　Zhu Weile

(*Dept. of Electronic Eng., UEST of China, Chengdu* 610054, China)

Abstract The multi-level LUT method for VLD is a fast, memory efficient method and is widely implemented in digital video and image decoders. This paper discusses the design and optimization for multi-level LUT and its state-machine in details, provides a formula for calculating the items of LUT and FSM states, and then proves the algorithm with an experiment of MPEG1. According to the result of this paper, the items in LUT can be fully saved in hardware design.

Key words VLD, FSM, MPEG1, Digital video

黎　文：男，1971 年生，博士生，主要从事图像处理、数字视频和 HDTV 等领域的研究.
李蜀雄：男，1971 年生，助教，主要从事图像处理、数字视频和 HDTV 等领域的研究.
朱维乐：男，1940 年生，教授，主要从事图像传输与处理、数字视频和 HDTV，信号处理等领域的研究与教学.