中图法分类号: TN919.81 **文献标志码**: A **文章编号**: 1006-8961(2011)02-0202-07
论文索引信息: 陶为, 唐建, 郭立. CAVLC 编码器的高速度、低复杂度结构设计[J]. 中国图象图形学报,2011,16(2):202-208

CAVLC 编码器的高速度、低复杂度结构设计

陶为, 唐建, 郭立
(中国科学技术大学电子科学与技术系,合肥 230027)

摘 要: 变长编码是视频编码中的关键技术之一,计算量大。提出一种应用于 H.264/AVC 的高速有效的 CAVLC 编码器结构。采用基于预处理的查找方法,改进了查找表结构;利用算术计算替代查找表,且通过展开和共享技术对算术表达式进行优化;其他方面也做了面积上的优化。实验结果表明,在 133 MHz 的频率约束下,采用 SMIC 0.18 μm CMOS 工艺进行逻辑综合,所需的逻辑门数为 8 723,能满足高清视频 1 920×1 088 – 30 帧/s 实时编码吞吐量的要求,具有实际应用价值。

关键词: H.264/AVC; 基于上下文的自适应变长编码 CAVLC; VLSI 结构

High speed and low cost architecture of CAVLC encoder

Tao Wei, Tang Jian, Guo Li
(*Department of Electronic Science and Technology, University of Science and Technology of China, Hefei* 230027 *China*)

Abstract: Variable Length Coding is one of the key technologies in video coding with large computation requirement. A high speed and efficient VLSI architecture for H.264/AVC CAVLC is proposed. A new look-up table algorithm based on the pre-processing greatly optimizes look-up table structure. An arithmetic compute method is exploited to replace look-up table, and the arithmetic expression is optimized with technology of unfolding and share. Some area optimization is done in the other part. Experimental result shows that for logic synthesis, the hardware cost of the proposed design is 8 723 logic gates by using SMIC 0.18 μm CMOS technology at the clock frequency constraint of 133 MHz. The new architecture can meet the requirement for the real-time processing for High-definition 1 920×1 088 – 30 fps video with less hardware cost. So it has practical application value.

Keywords: H.264/AVC; context-based adaptive variable length coding(CAVLC); VLSI architecture

0 引 言

H.264/AVC 是 ITU-T 和 MPEG 组织共同推出新一代视频压缩标准,其压缩率较以往的视频标准有显著提高,得到广泛应用。H.264 标准编解码运算变得更加复杂,随着实时、高清视频的需求日益广泛,对高清、高保真编解码实现方案的硬件实现及优化研究具有很高的实用价值。

H.264 中的熵编码主要包括基于上下文的自适应二进制算术编码(CABAC)和基于上下文的自适应二进制变长编码(CAVLC)。其中 CAVLC 是 H.264标准的基本熵编码方法,计算量大,被基本档次和扩展档次所采用。它主要用于对亮度和色度残差变换、量化后的系数进行编码。

近年来,对于 CAVLC 主要有如下几种设计方案[1-5],主要可归纳为针对结构的优化和针对查找表(LUT)的优化。文献[1]采用直接映射的方式,未对算法进行优化,造成硬件资源的浪费,系统时钟频率仅能达到 32 MHz;文献[2]采用了两级流水线实现了对两个 4×4 块的并行处理,虽然编码速率达 100 MHz,可实现高清视频实时处理,但硬件资源消

收稿日期:2009-08-21;**修回日期**:2009-10-27

第一作者简介:陶为(1986—),男。中国科学技术大学电子科学与技术系电路与系统专业硕士研究生,主要研究方向为视频编解码的超大规模集成电路设计。E-mail: zytttw@mail.ustc.edu.cn。

耗达 17.6 K 门;文献[3]采用边扫描边编码的方式,去除统计过程中的存储部件,码字输出部分采用状态机输出。虽然资源消耗少,但吞吐量小,只能用于移动环境下低比特率处理;文献[4]虽然采用了在线计算的方法减少查找表的数目,优化了面积,但未对关键路径优化,只能满足 QCIF/10 帧/s。

针对 LUT 优化方面,文献[5]利用分裂和共享技术降低 LUT 的大小,码字中的前缀和后缀可共享同一个 LUT,但其地址产生电路复杂,致使实际面积仍然很大。目前尚未有对速度和面积较好折衷的方案报道。

从算法和结构上综合考虑,提出一种高速度、低复杂度的 CAVLC 编码器结构,实现速度和面积很好折衷。通过基于预处理的查找表方法,改进查找表结构,可以快速得到相应的码长和码字;对于非零系数 Level 的编码,利用展开和共享技术,减小关键路径,面积得到有效的降低;码字拼接模块里面也对面积进行优化。

实验结果表明,提出的方法对于减少占用芯片面积是非常有效的,且能满足高清 HDTV(1 920 × 1 088 – 30 帧/s)实时编码要求。

1　CAVLC 基本原理

CAVLC 主要用于对亮度和色度残差变换、量化后的系数进行编码。CAVLC 编码器结构如图 1 所示,主要由 5 个模块组成:输入 RAM 模块、扫描与统计模块、编码模块、码字拼接模块、控制模块。

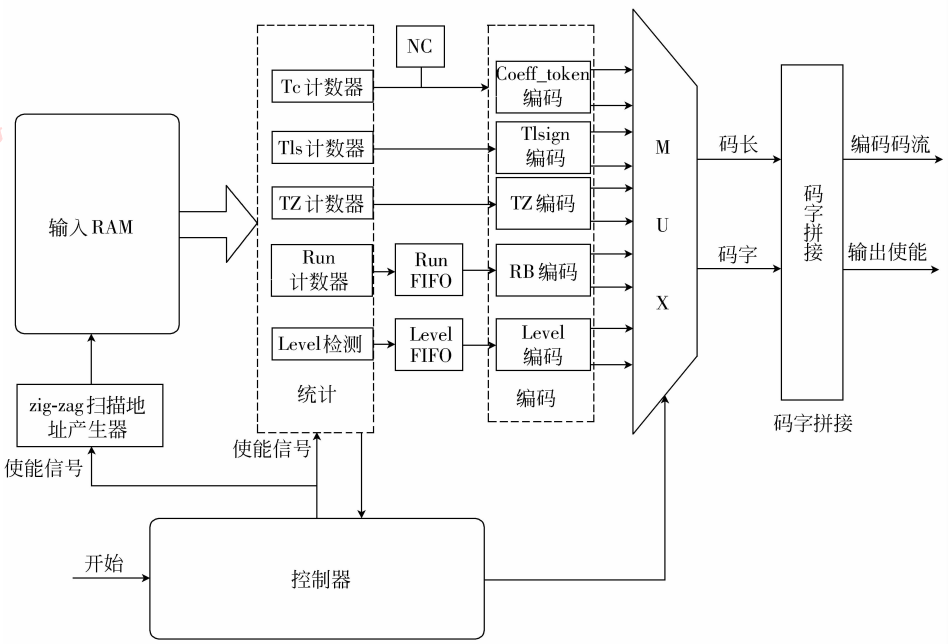

图 1　CAVLC 编码器结构框图

Fig. 1　Architecture of CAVLC encoder

量化后的数据要经过图 2 所示的 zig-zag 扫描。由于较大的非零系数主要集中在低频附近,高频部分的非零系数大部分是 +1/ –1;相邻 4 × 4 块的非零系数的个数是相关的。因此经过 zig-zag 扫描后,CAVLC 就可以利用相邻已编码的符号所提供的相关性,为待编码的符号选择合适的上下文模型,从而降低了符号间的冗余度。

编码时,当开始信号有效,控制模块发出扫描与统计模块使能信号,输入 RAM 中数据按逆向 zig-zag 扫描顺序读出,统计出 5 个编码元素(非零系数个

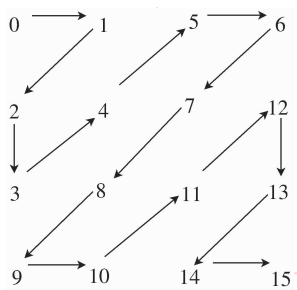

图 2　Zig-zag 扫描

Fig. 2　Zig-zag scanning

数 Tc(TotalCoeff),拖尾系数个数 Tls(Trailingones,即 +/-1),除拖尾系数外的非零系数(Level),最后一个非零系数前零的个数 TZ(TotalZero),每个非零系数前零的个数 RB(RunBefore))的值;编码模块根据查找表得出相应的码字和码长;码字拼接模块根据码长拼接相应码字,并以 32 位输出。具体的统计过程如下:

1)如果输入是非零系数,则 Tc 计数器加 1,并作标记,表示已出现第一个非零系数。

2)如果输入是(托尾系数)+1/-1,则 Tls 计数器加 1。

3)如果是其他非零系数,或 Tls 等于 3 后的 +1/-1,存入相应的非零系数 Level FIFO 中。

4)对 TZ 和 RB 统计,在非零系数开始计数,同时置 RB 计数器为 0,在每个零系数来时对 TZ 和 RB 计数,在下个非零系数时,输出 RB 计数器的值。

具体地,CAVLC 的编码模块工作过程如下:

1)Coeff_token 编码:对 Tc 和 Tls 进行联合编码,Tc 的取值为 0 到 16,Tls 的取值为 0~3。编码过程中根据码表选择参数(*NC*)来选择 6 个码表之一进行编码。其中 *NC* 值依赖于当前块上面的 4×4 块的非零系数和左边的 4×4 块的非零系数,正是基于上下文自适应的体现。

2)Tlsign 编码:对 Tls 符号进行编码,正 1 用 0 表示,负 1 用 1 表示,按照逆向 zig-zag 扫描。

3)Level 编码:对除拖尾系数之外的非零系数编码,按照逆向 zig-zag 扫描。

4)TZ 编码:对最后一个非零系数前零的个数进行编码,此非零系数指的是按正向扫描的最后一个非零系数。

5)RB 编码:按照逆向 zig-zag 扫描顺序,对每个非零系数前零的个数编码。有两种情况不需要编码:一种是最后一个非零系数,另一种是没有剩余的零需要编码。

2 算法和架构

由于 CAVLC 中的编码模块最为复杂,耗时较多,含有诸多码表,主要是通过 LUT 实现。为了提高速度和减小面积,结合编码特点,提出一种基于预处理的查找方法,可以快速得到相应的码长和码字;对于非零系数 Level 编码,算术计算代替直接查找表,且利用展开和共享技术对逃脱码下的编码进行优化,减小关键路径,提高编码速度;码字拼接模块主要通过优化桶型移位寄存器及取模计算来减小面积。下面就 3 个方面具体阐述。

2.1 Coeff_token 编码

Coeff_token 编码是通过查找表实现,亮度部分有 3 个变长码表(VLCT)VLCT_0 - VLCT_2 和一个定长码表,色度部分有一个 VLCT[6]。

对于每一个 CAVLC 元素编码,需要两个值来表征:码长 CL(codelength)和码字 CW(codeword)。表 1 所示为亮度码表 VLCT_0[6-7],由表 1 可知,若采用直接映射查找表则需要很大存储空间。例如,Tc=5,Tls=0 时,对应的码字为 00000000111,码长大小为 11,则分别需 11 bit 和 4 bit。文献[5]提出一种降低 LUT 的方法,将码字按前缀和后缀来存储。其中,前缀值即首个 1 前面零的个数,后缀用 4 bit 表示。由于前缀小于等于 12,也可以用 4 bit 表示,故 LUT 大小降为 4×16。但其地址产生电路比较复杂,而且没有解决码长的查找问题。

表 1 标准中的 VLCT_0 码表结构

Tab.1 VLCT_0 in the standard

Tc	Tls			
	0	1	2	3
0	1			
1	000101	01		
2	00000111	000100	001	
3	000000111	00000110	0000101	00011
4	0000000111	000000110	00000101	000011
…	…	…	…	…

表 2 给出了本文提出的码表结构,通过对 coeff_token 码表分析,改变码表结构。首先按码长长度对码表进行分组排列,同码长的码字归属同一组,但仍然不能由输入 Tc 和 Tls 得出该组码长的信息。进一步分析,提出一种基于预处理的方法来判断码长。通过先求出“差值”,再由差值来判断码长。其中,“差值”等于 Tc - Tls。

由表 2 可以看出:

1)按一般的同码长分组,分成许多等长小码表。但是,不能快速有效地从输入得出该组码长。

2)在相同的“差值”对应的码字中,大多情况下,Tls 为 3 对应的码长比 Tls 为其他值对应码长小 1,这也与给大概率字符赋予短码字相符。

3)在相同的“差值”里,大多数情况下,有效码

字与拖尾系数有 CW = ~Tls。因此,可用算术计算的方法来替代部分 LUT,大大减少 LUT 大小。

表 2　提出的 VLCT_0 码表结构

Tab. 2　Proposed VLCT_0

Tls	Tc	码字	码长	Tc - Tls
0	0	1	1	0
1	1	01	2	
2	2	001	3	
3	3	00011	5	
3	4	000011	6	1
0	1	000101		
1	2	000100		
2	3	0000101	7	1
3	5	0000100		2
0	2	00000111	8	2
1	3	00000110		
2	4	00000101		
3	6	00000100		3
0	3	000000111	9	3
1	4	000000110		
2	5	000000101		
3	7	000000100		4
0	4	0000000111	10	4
1	5	0000000110		
2	6	0000000101		
3	8	0000000100		5
…	…	…		

通过新的算法可将原先的码字和码长都需要通过 Tc 和 Tls 查找得出的查找方式,变为以“差值”为主来查找码长,对应里面的码字由算术计算代替 LUT 的查找方式。

2.2　非零系数 Level 的编码

Level 编码是 CAVLC 中最复杂的模块,为了提高编码效率,CAVLC 采用 7 个变长码表[6]。记 $n \in \{0,1,2,3,4,5,6\}$ 代表 Level 编码中变长码表的索引。对每个码表的选择是基于上下文自适应的,依赖于非零系数的个数,拖尾系数和已编码的 level 值。码表索引的初始化及更新语法如下:

```
n = (Tc > 10&&Tls < 3) ? 1:0;
if(abs(level[i]) > (3≪(n - 1))&&n < 6)
n++;
```

Level 编码包括常规编码和逃脱码编码(escape code)。逃脱码是指发生概率小的字符,其编码用长码字表示。标准[6]规定逃脱码下,码长为 28,前缀长度为 15。但随着像素位宽增大,经 DCT 变换后的残差系数明显增大。标准采纳 Au[7]的建议,扩展了逃脱码下的编码方案,最大码长将超过 28,前缀长度也可能大于 15,使其实现复杂度变大。

如果采用直接映射查表的方法需要占用大量的 ROM 资源。通过算术计算得到相应的码字和码长,Level 码字可以表示为如下形式:00…01xx…xs,由前缀(level_prefix)和可能有的后缀(level_suffix)构成。其中首个 1 前面的所有 0 为前缀,xx…xs 为后缀,且 s 代表 level 的正负,若是正数,s 为 0;反之,s 为 1。对一个 level,码字可由以下几个参数确定:前缀值 l_{pfx}、后缀值 x_{sfx}、后缀长度 l_{sfx}。

记 x 为编码的 level 值。为避免正负动态范围,用的幅值(x)的绝对值 x'代替 x。即

$$x' = \begin{cases} x & x \geqslant 0 \\ -x & x < 0 \end{cases} \tag{1}$$

首先给出后缀长度 l_{sfx}的表达式,它可通过前缀值以及码表索引值计算得到。有两种情况:

使用 0 号码表时($n=0$),后缀长度的表达式为

$$l_{sfx} = \begin{cases} l_{pfx} - 3 & l_{pfx} \geqslant 15 \\ 4 & l_{pfx} = 14 \\ 0 & \text{其他} \end{cases} \tag{2}$$

使用 1-6 号码表时,

$$l_{sfx} = \begin{cases} l_{pfx} - 3 & l_{pfx} \geqslant 15 \\ n & \text{其他} \end{cases} \tag{3}$$

接下来给出后缀值 x_{sfx}的推导。记 x'_{sfx} = xx…x 有

$$x_{sfx} = x'_{sfx} \ll 1 + sign \tag{4}$$

式中 $sign$ 表示前文中 level 码字结构 xx…xs 中的“s”。

$$x'_{sfx} = \begin{cases} x' - l_{pfx} \cdot 2^{(n-1)} & l_{pfx} \leqslant 15 \\ x' - 15 \cdot 2^{(n-1)} - 2^{l_{pfx}-3} - 2^{11} & l_{pfx} \geqslant 16 \end{cases} \tag{5}$$

由此可知,参数前缀值 l_{pfx}很重要,直接关系另两个参数的求解。在非逃脱码的情况下如下式

$$l_{pfx} = x' \gg (n-1) \tag{6}$$

逃脱码情况下比较复杂,与已求前缀值和当前编码的 Level 值大小有关,算法框图如图 3 所示。

图 3 中,$l_{esc} = x' - \text{escape}$。escape 为每个码表里有对应的逃脱码,$x'$大于该值的编码就称为逃脱码编码,小于则为常规编码。由上述可知,在逃脱码下 level 编码变得相当复杂,尤其是其前缀值的计算,如果直接将其映射成硬件电路,迭代和反馈必将给速度带来负担。通过对其进行变换,计算出每种逃脱码下每次迭代对应的前缀值,将反馈展开。这样

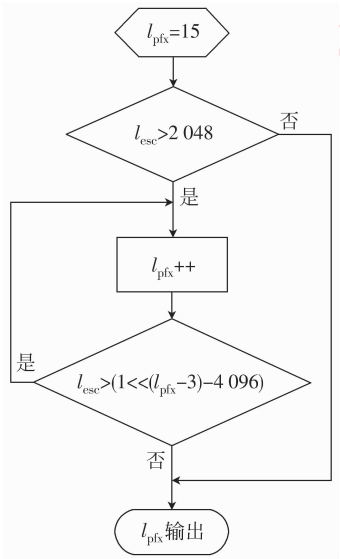

图 3 逃脱码下前缀值的计算

Fig. 3 Computation of l_{pfx} in the escape code

通过输入直接得出最终的前缀值,去除了迭代过程,提升了编码速度。

到此,对于给定码表索引 n,根据上述公式就能求出 level 编码中 3 个重要的参数。进一步,可以根据已知三要素推导出码字和码长的表达式。

码长的求解如式(8)所示,在非逃脱码下,码长可表示为前缀值加后缀长度再加 1;在逃脱码下,用变量 $\Delta_{pfx} = l_{pfx} - 15$ 表示在逃脱码下前缀值大于 15 时前缀值的增加量;前缀值等于 15 时,该值为 0。这样,通过表达式共享,将标准中两次的方案整合到一起。

$$CL = \begin{cases} l_{pfx} + n + 1 & \text{非逃脱码} \\ 28 + (\Delta_{pfx} \ll 1) & \text{逃脱码} \end{cases} \tag{7}$$

对于码字的求解,同样类似于码长的求解,由下式给出。

$$CW = \begin{cases} 1 \ll (l_{pfx} + 1) + x_{sfx} & \text{非逃脱码} \\ 1 \ll (12 + \Delta_{pfx}) + x_{sfx} & \text{逃脱码} \end{cases} \tag{8}$$

2.3 码字拼接和输出

CAVLC 码字拼接和输出架构如图 4 所示。首先,根据码长对码字进行左对齐处理。然后,在码长的控制下,左移桶型移位寄存器(LBS)将每次编码产生的码字进行拼接,寄存器 W1 中存放的是已拼接的码字,但没有输出。右移桶型移位寄存器(RBS)则将拼接好的码字进行分割成 32 比特输出至寄存器 W2。

文献[8]采用 3 个桶型移位寄存器(BS)结构,该结构虽然速度快,然而 BS 占据面积非常大。文献[9]用加法器替代了文献[8]中的 BS,面积上有所减少。用一个 LBS 和一个 RBS 替代两个 BS,但两者的面积相当于一个 BS。另外,LBS 移位的控制部分位宽减少到 5 位。对于 RBS 移位的控制,是利用累加码长,再与 32 取模得出将要移的位数。文

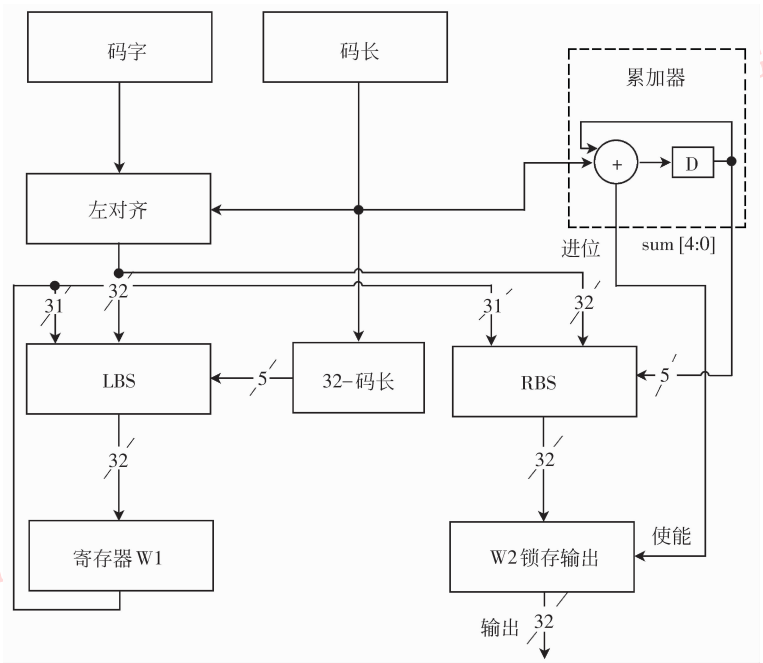

图 4 CAVLC 码字拼接和输出结构

Fig. 4 Architecture of CAVLC codeword pack and output

献[9]利用了两个加法器,一个 D 触发器,一个非门;文本结构利用一个 6 位的加法器电路替代,只需取输出和的低 5 位即可实现取模计算,利用最高位是否为 1,判断是否输出 W2。

3　实验结果及分析

为了对提出的结构进行验证,采用 verilog HDL 进行 RTL 级设计,用 Modelsim SE6.0 进行功能仿真,并与 H.264/AVC 软件模型 JM14.1[10] 比对。最后在 SMIC 0.18 μm CMOS 工艺下用 synopsys 综合工具 DC 进行逻辑综合,并通过 FPGA 验证。

表 3 所示为 3 种不同类型的 QCIF 测试序列在不同的量化参数(QP)下的编码所需要的时钟周期数。对于高清视频 1 920 × 1 088 – 30 帧/s,在 133 MHz时钟下,每个 MB 编码理论上可得到的时钟数可计算如下:

$$f_{MB} = 133 \times 10^6 \Big/ \frac{1\,920 \times 1\,088 \times 30}{16 \times 16} = 543$$

测试结果表明,编码一个所宏块(MB)需要的平均时钟周期数约为 300 左右,小于理论计算值 543。

表 3　每个宏块编码所需要的平均时钟周期

Tab.3　Average coding cycles of one macroblock

QP	Forman	Mobile calendar	Stefan
12	420	498	450
24	264	364	327
36	185	244	228
48	104	186	177
平均	243	323	296

表 4 所示为提出的结构在不同频率约束下与文献[9]和文献[2]在硬件资源消耗方面的比较。可以看出,与其他结构相比,本文提出的结构的综合结果有明显的优势。在 Coeff_token 编码中,本文提出的算法通过预先求差值运算之后再查表,使得面积得到明显减少,仅添加一个加法器,可以在一个时钟周期内完成;在 level 编码中,通过算术计算代替查找表,针对逃脱码进行优化,减小关键路径,提升了编码速度。

表 5 给出了文本结构与其他几种实现结构整体上的资源与速度的比较。由于不同的工艺和平台综合得出的电路的速度和面积会有不同,表中分别标明不同工艺的参数。与文献[3]相比,该文虽更具有面积上的优势,但其吞吐量小,不适应高清视频编码应用。与文献[9]相比,本文提出的结构在相同的时钟频率下,所消耗的资源数大约减少 33%,实现了面积和速度的有效折衷。

表 4　CAVLC 中模块综合结果比较

Tab.4　Comparison of synthesis result of modules in CAVLC

子模块	逻辑门数			
	本文 133 MHz	本文 100MHz	文献[7] 133 MHz	文献[2] 100 MHz
Coeff-Token 编码	561	531	917	864
Level 编码	876	827	961	1 012
TZ 编码	427	397	598	646
RB 编码	223	204	289	263
统计及缓存	4 480	4 193	8 795	12 283

表 5　本文结构与其他实现结构比较

Tab.5　Comparison of proposed design with others

	本文结构	文献[7]	文献[2]	文献[3]	文献[4]
工艺	0.18	0.18	0.18	FPGA	0.35
门数	8 723	13 114	17 635	6.8 K	9 171
速度/MHz	133	133	100	50	66
性能/(帧/s)	HD1080 30	HD1080 30	HD1080 30	CIF/QCIF 30	QCIF 10

4　结　论

提出了一种应用于 H.264/AVC 的高速度、低复杂度的 CAVLC 编码器结构。从算法和结构上综合考虑,通过分析 CAVLC 中码表的特性,力求将硬件复杂度降低。主要采取以下方法:

1)对于 LUT 优化,基于预处理的查找法改进了查找表结构,算术计算则消除了部分查找表。

2)对于 Level 编码,算术计算代替直接查找表,利用展开和共享等技术降低了逃脱码下硬件实现复杂度。

3)对于码字拼接模块,通过减少桶型移位寄存器个数以及其位宽,巧妙地利用了最高位实现取模等技术,使得该模块的面积得到有效降低。

实验结果表明,本文的设计可以在满足 HD1 920 × 1 088 – 30 帧/s 下,明显地减少硬件资源消耗。提出的架构是有效的,在高清、高保真视频编码器的设计中将有广泛地应用前景。

参考文献(References)

[1] Amer Ihab, Badawy Wael, Jullien Graham. Towards MPEG-4 part 10 system on chip: a VLSI prototype for context based adaptive variable length coding (CAVLC)[C]//IEEE Workshop on Signal Processing Systems, Austin, Texas, USA: IEEE Press, 2004: 275-279.

[2] Chen Tungchien, Huang Yuwen, Tsai Chuanyung, et al. Architecture design of context-based adaptive variable-length coding for H.264/AVC[J]. IEEE Transactions on Circuits and Systems II, 2006, 53(9): 832-836.

[3] Rahman Choudhury A, Badawy Wael. CAVLC encoder design for real-time mobile video applications[J] IEEE Transactions on Circuits and Systems II: Express Briefs, 2007, 54(10): 873-877.

[4] Lai Yeongkang, Chou Chihchung, Chung Yuchieh. A simple and cost effective video encoder with memory-reducing cavlc[C]// IEEE International Symposium on Circuits and Systems. Washington DC, USA: IEEE Press, 2005: 432-435.

[5] Rahman C A, BadawY W. An area efficient real-time CAVLC ip-block towards the H.264/AVC Encoder [C]//IEEE Workshop on Signal Processing Systems Design and Implementation. Washington DC, USA: IEEE Press, 2006: 368-371.

[6] Draft ITU_T Recommendation and Final Draft Internaltional Standard of Joint video Specification (ITU-T Rec. H.264/ISO/IEC 14496-10 AVC)[EB/OL]. In Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G053, 2005.

[7] Au James. Complexity reduction of CAVLC[EB/OL]. (2002-07-22)[2008-07-01]. http://wftp3.itu.int/av-arch/jvt-site/2002_07_Klagenfurt/JVT-D034.doc.

[8] Lei Shawmin, Sun Mingting. An entropy coding system for digital HDTV application [J]. IEEE Transactions on Circuits and Systems for Video Technology, 1991, 1(1): 147-155.

[9] Hu Hongqi, Sun Jingnan, Xu Jiadong. High performance architecture design of CAVLC encoder in h.264/AVC[C]// Proceedings of the 2008 Congress on Image and Signal Processing. Washington DC, USA: IEEE Press, 2008: 613-616.

[10] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG. JVT Reference Software, version JM14.1[CP/OL] 2005-09-26/2008-07-01. http://iphome.hhi.de/suehring/tml/download/old_jm/.