基于AVS的软硬件协同可变长码解码器设计

刘 玮 陈咏恩 许苑丰
(同济大学通信软件及专用集成电路设计中心 上海 200092)

摘 要 提出一种基于软硬件协同方法的AVS可变长码解码器结构设计。定长码、指数哥伦布码及AVS视频标准特有的基于内容自适应二维可变长码(CA-2D-VLC)均可在该解码器上实现正确解析。通过对19张可变长码表的优化整合,提出一种新的码表设计方法。经验证,新码表相较使用原始码表可将硬件消耗降低30%以上。为确保整个系统设计的合理性和正确性,以RM52J为蓝本编写针对本解码器的验证器,通过对92个一致性测试码流序列解析对比,表明本设计满足AVS视频解码要求。

关键词 AVS,软硬件协同,可变长码,CA-2D-VLC
中图法分类号 TP368.1 **文献标识码** A

Hardware/Software Partitioning VLD Design Based on AVS

LIU Wei CHEN Yong-en XU Yuan-feng
(Communication Software & ASIC Design Center, Tongji University, Shanghai 200092, China)

Abstract Architecture with Hardware/Software co-design for AVS Variable Length Code(VLC) decoding was designed. Fixed Length Code, k-th Exp-Golomb Code and context-based adaptive 2D-VLC (CA-2D-VLC) Code were decoded correctly on the platform. New design method for VLC table was presented, which was the optimization of nineteen original tables. As a result of the logical design and large amount of instructions decrease, a reduction of more than 30% in hardware consumption was achieved. Verifier based on RM52J was proposed as well to validate the rationality and validity of the whole system. Tested by ninety-two streams, this design was proved to reach the requirement of AVS video decoding.

Keywords AVS, HW/SW co-design, Variable length code, CA-2D-VLC

AVS(Audio Video Coding Standard)是由我国AVS工作组提出并完成的具有自主知识产权的数字音视频编解码技术标准,是目前最先进的音视频压缩编解码标准之一,其视频编码效率达到目前国际通用标准MPEG-2的两倍以上,与H.264/AVC相当。它基于经典的混合编码框架,核心技术主要包括8×8整数变换、量化、帧内预测、1/4精度像素插值、特殊的帧间预测运动补偿、二维熵编码和去块效应环内滤波等。由于我国掌握AVS标准的主要知识产权[1],专利授权模式简单,因此AVS的产业化可以节省相当可观的专利费用。如果能够适时推出基于AVS的数字视频编解码芯片,就可望打破国外厂商对该产品的长期垄断,为我国数字电视等音视频产业和相关芯片产业提供跨越发展的技术源头。

视频解码器的结构一般包括VLD(Variable Length Code Decoder,可变长码解码器)、反扫描、反量化、反离散余弦变换、运动补偿和去块效应滤波6大部分,如图1所示。VLD作为整个解码器的第一级,它的解码速度、正确性及稳定性直接关系到其他模块的工作情况,在视频解码过程中占有至关重要的地位。AVS标准中定义的语法元素采用了定长码、指数哥伦布码和CA-2D-VLC(Context-based Adaptive 2D-VLC,基于内容的自适应二维可变长码)3种方法进行编码,因此为满足AVS视频解码的实时高速应用而设计一种高效合理的可变长解码器具有重大意义。

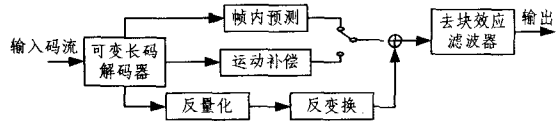

图1 视频解码器结构

由于编码效率的提高以高计算复杂度为代价,实时处理多维图像和视频信号需要操作大量数据流,因此这是传统的纯硬件结构所不能满足的。软件的灵活性和可编程性及硬件强大的数据处理能力,使得视频解码芯片设计方法向软硬件协同设计的方向发展。本文在深入研究AVS标准中采用的熵编码算法及数字视频解码芯片系统结构发展的基础上,设计出一种基于软硬件协同设计方法的AVS可变长码解码器结构。

1 AVS熵编码算法

可变长码又称哈夫曼码,其基本原理是使用较短的码字

到稿日期:2009-08-14 返修日期:2009-11-13

刘 玮(1982-),女,博士生,主要研究方向为H.264/AVS视频编解码算法及解码芯片设计等,E-mail:bowei0246@sina.com;陈咏恩(1946-),男,博士生导师,主要研究方向为通信中的信号处理等;许苑丰(1982-),男,硕士生,主要研究方向为H.264/AVS解码芯片设计等。

表示出现频率较高的信息，而用较长的码字表示出现频率较低的信息，从而降低平均比特率。AVS采用了不同于其他视频编解码标准的特殊熵编码算法，即基于内容的自适应二维可变长码(CA-2D-VLC)。它是指数哥伦布码和二维可变长码的结合体，提供包含500多个码字的19张不同的码表进行编码，以提高编码效率。

AVS视频标准定义的语法元素如表1所列。其中位于码流条带层及以上的视频序列头层(sequence_header layer)、图像层(picture layer)等的语法元素，绝大多数使用定长码和0阶指数哥伦布码编码，使用CA-2D-VLC编码的是块层(block layer)的残差系数。

表1 AVS标准中语法元素定义

描述符	说明
b(8)	一个任意取值的字节
u(n)	n位无符号整数
f(n)	取特定值的连续n个比特
i(n)	n位整数，用高位在前的二进制补码表示
se(v)	用0阶指数哥伦布编码的有符号整数
ue(v)	用0阶指数哥伦布编码的无符号整数
ce(v)	用k阶指数哥伦布编码的残差系数，k=0～3

定长码的解码过程相对较为简单，只需按照给定的码字长度，从码流中截取相应位数的比特位，然后根据需要选择查表或直接返回截取的比特位值即可；0阶指数哥伦布码可根据公式解析码字，无需查表；而高阶指数哥伦布码则需要根据上下文信息调整阶数、变换码表，解析过程较为复杂。

1.1 指数哥伦布码

指数哥伦布码是一种具有规则结构的可变长码。码字由前缀和后缀两部分组成，其中前缀由 m 个前导"0"和一个分隔符"1"组成，后缀是 n 个任意值比特位，也是真正的信息位，如下所示：

$$\underbrace{0\cdots01}_{m}\ \underbrace{x_{n-1}x_{n-2}\cdots x_0}_{n}$$

对于 k 阶指数哥伦布码，$n=m+k$，则码字的长度 L 可用 m 和 k 表示：

$$L=m+n+1=2m+k+1 \tag{1}$$

k阶指数哥伦布码的结构如表2所列，其中信息位INFO定义为读取码流中连续 n 位二进制数的值：

$$INFO=\sum_{i=0}^{n-1}X_i\cdot 2^i \tag{2}$$

表2 k 阶指数哥伦布码

阶数	码字结构	取值范围	CodeNumber
k=0	1	0	2^m+INFO-2^0
	$01x_0$	1～2	
	$001x_1x_0$	3～6	
	……	……	
k=1	$1x_0$	0～1	2^{m+1}+INFO-2^1
	$01x_1x_0$	2～5	
	$001x_2x_1x_0$	6～13	
	……	……	
k=2	$1x_1x_0$	0～3	2^{m+2}+INFO-2^2
	$01x_2x_1x_0$	4～11	
	$001x_3x_2x_1x_0$	12～27	
	……	……	

AVS标准定义的语法元素中除了残差系数和定长码编码的语法元素外，其余均由无符号或有符号0阶指数哥伦布码编码。使用无符号方法编码的语法元素，其编码前原始值与CodeNumber相等；使用有符号方法编码的语法元素，其编码前原始值与CodeNumber的关系如式(3)所示，其中 v 表示编码信息原始值：

$$CodeNumber=\begin{cases}2|v|, & v\leqslant 0\\ 2|v|-1, & v>0\end{cases} \tag{3}$$

相应地，进行0阶指数哥伦布码解码时，第一步需要顺序读取码流中的前导"0"，直至找到分隔符"1"，并同时记录前导零的个数 m；第二步，继续顺序向后读取分隔符后 m 个比特位，即INFO的值；第三步，根据表2中对0阶指数哥伦布码的定义，计算CodeNumber的值。得到该值之后，依据不同语法元素在标准中定义为有符号还是无符号来计算其原始值。

1.2 CA-2D-VLC

AVS标准对残差系数编码相对复杂些，采用了CA-2D-VLC算法，语法元素与编码生成的CodeNumber之间的关系用ce(v)表示。

编码时，首先以8×8像素块为单位对量化后的二维残差系数进行游程编码，并转换成一系列残差数据对(Run，Level)，其中Level为非零系数的值，而Run为Level之前连续零的个数；对非逃逸残差系数，根据上下文信息从19张变长码表中选择适当码表，构造码字；对逃逸残差系数，要根据规则确定指数哥伦布码的阶数，再构造出码字。也就是，通过选择合适的可变长码表将每对残差数据(Run，Level)对应一个CodeNumber值，然后该值再根据上下文信息用0～3阶指数哥伦布码进行编码。通常一个块的第一个码字选择0号码表，例如VLC0_Intra，VLC0_Inter或者VLC0_Chroma，均由当前块的类型决定，而下一个码字的码表选择由它之前被解码出的Level值决定。当一个块中最后一对(Run，Level)被编码结束后，添加一个EOB(End of Block)码。

19个残差系数变长码表，包括7个帧内预测编码亮度块码表，即VLC0_Intra，VLC1_Intra，VLC2_Intra，VLC3_Intra，VLC4_Intra，VLC5_Intra和VLC6_Intra；7个帧间预测编码亮度块码表，即VLC0_Inter，VLC1_Inter，VLC2_Inter，VLC3_Inter，VLC4_Inter，VLC5_Inter和VLC6_Inter；5个色度块码表，即VLC0_Chroma，VLC1_Chroma，VLC2_Chroma，VLC3_Chroma和VLC4_Chroma[2]。每一个码表都采用了确定阶数的指数哥伦布码。

相对于其他视频编解码标准中不同像素块需要使用不同码表的方法，AVS标准仅仅采用19张映射码表，充分利用了上下文的信息，并大大降低了码表占用的存储空间以及解码时频繁查表带来的损耗。

2 软硬件协同系统结构设计

软硬件分区的过程就是算法在软件和硬件实现之间折中的过程。软件见长于处理控制方面的操作，有较强的灵活性；而硬件则善于处理包含大量重复性计算和简单条件跳转的操作，且实现速度相较软件有很大提升[3]。

基于AVS标准的视频码流按层次结构进行组织，从上到下依次为序列层、图像层、条带层、宏块层和块层。语法元素解析就是按照该层次结构依次从AVS码流中读取相应位数的比特位，并解码获得所需语法元素值的过程。由于宏块层以上各层都以唯一码字来标识其开始，语法元素大多采用定长码编码方式及少部分变长码编码方式，无需大量数据操作，

复杂度也不高，并且需要对解析结果编辑形成控制信息格式，因此适用于软件方式实现。而位于宏块层的残差数据解析要复杂得多，包含大量频繁的码表变换、判断及查表操作，若用单纯的软件解码根本无法适应视频解码的实时性要求，反而用硬件方式实现非常适合。

本文所设计的基于 AVS 标准的可变长解码器，即是将码流宏块层以上部分的语法元素解析及系统控制分配给软件实现。其中系统控制包括对硬件各寄存器状态设置、在 RAM 上配置待解码流等，此外还要将已经解析的语法元素流作为驱动的一部分，按照一定格式用 DMA 控制器传输给硬件。硬件部分则依据软件输入的参数流开始宏块层语法元素的解析，并将结果输出到 RAM 存储。软硬件两大部分的功能设计及实现将在下面详细介绍。

3 软件设计与实现

根据上述软件部分所需实现的功能，把它划分为 3 大层次，分别是应用层、中间层和驱动层。

应用层包括主函数 main()、AVS 码流解析函数和初始化模块，即参数寄存器分配和硬件初始化；中间层是语法元素解析的主体部分；驱动层负责软硬件接口及输出硬件控制信息。详细划分如图 2 所示。

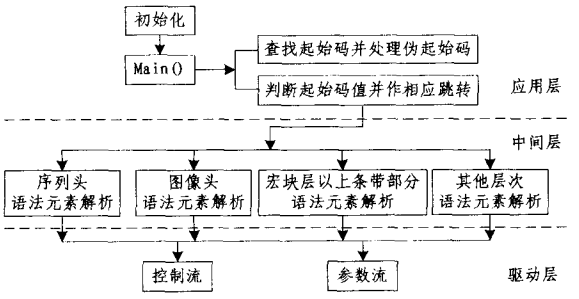

图 2 软件功能模块划分

3.1 应用层

3.1.1 初始化

初始化过程包括以下 5 个步骤：

1) 设置终端寄存器和矢量寄存器等关键寄存器。

2) 初始化内核工作区。

3) 创建初始化定义对象。

4) 启动生成的系统初始化程序，包括定时器、存储管理单元等。

5) 转向多任务进程，并开始第一个任务。

视频解码软件部分的算法采用 C 语言实现，在完成操作系统内核初始化之后，程序进入多任务状态，视频解码处理部分作为主任务开始执行。在程序编写时，使用类和结构体表示硬件系统内部的模块和寄存器，给类或结构体变量赋值实际就是将值传递给硬件寄存器的过程。

3.1.2 去伪起始码

AVS 标准中定义的起始码是一组特定的比特串，由 23 比特 "0"值、1 比特 "1"值和 8 比特的起始码值构成。起始码值标识码流中的特殊位置，比如图像头、条带头等。由于 AVS 大量采用指数哥伦布码，其结构如 1.1 节所述，若待编码的连续两个指数哥伦布码都较长，则有可能出现前一个数据的后缀和后一个数据的前缀的"0"加起来共 23 比特，也就是和起始码的前缀相同，从而引入伪起始码。

当码流中出现了字节对齐的 0x000001 时，如果接下来的一个字节是 B0(视频序列起始码)、B2(用户数据起始码)、B3(I 帧图像起始码)或 B6(P/B 帧图像起始码)，解码器就会把它误当成图像的头文件，使接下来的信息全部错位。如果伪起始码出现在 B 帧，只会影响当前图像；如果出现在 P 帧，会影响 2～3 个图像；如果出现在 I 帧，则可能会影响整个图像。

为避免以上情况发生，AVS 编码时每写入一位，如果该位是一个字节的第二最低有效位，则检测这位之前写入的 22 位。如果都是"0"，在该位之前插入"10"。相应地，解码时每读入一个字节，检查前面读入的两个字节和当前字节。如果这 3 个字节构成比特串'0000 0000 0000 0000 0000 0010'，则丢弃当前字节的最低两个有效位[4]。

本文设计的解码器软件部分中，去伪起始码操作即是基于以上的理论和处理方法，通过按位移动指针，探测填充位"10"。如果判断是填充位，则将指针自动后移两位，继续读取码流，从而实现丢弃填充位的功能。同时对已经读取的码流视为普通视频信息码，而非起始码进行处理，从而消除了起始码竞争。

3.2 中间层

应用层完成了视频码流读入、起始码正确定位及起始码值判别。中间层是循环解析视频码流各个层次语法元素的主体。宏块层以上所有用定长码或指数哥伦布码编码的语法元素都在该层得到解析处理，并且解析结果以一定格式存储于寄存器中，供驱动层使用。

根据 AVS 标准，可变长码的码长不大于 32 比特。因此，程序设计时定义了一个 32 位的 cache 变量，用于临时存储读取的视频码流。定长码解析时，只需从高位起顺序取出 cache 中相应的位数存入定义的对应变量即可。cache 中剩余比特位全部左移，同时有新的视频码流从 RAM 中读取并补充进来，再继续解析下一个语法元素。

此外，由于宏块层以上采用指数哥伦布码编码的语法元素都为 0 阶，仅存在是否有符号的差别，因此需要设计 0 阶指数哥伦布码的解析函数。基于 1.1 节所述解码理论方法，得到如图 3 所示的 0 阶指数哥伦布码解码过程图。该循环被多次调用，直至解析完所有宏块层以上 0 阶指数哥伦布码[5]。

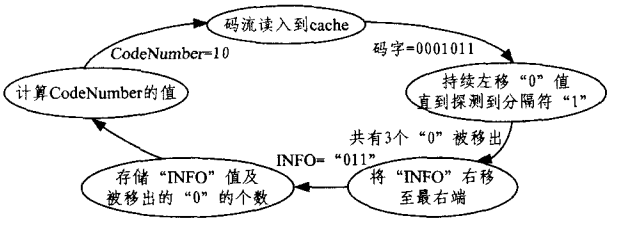

图 3 0 阶指数哥伦布码解码过程图

3.3 驱动层

中间层将解析得到的语法元素值分别存放于对应名称的变量中，驱动层则把这些值重新整合成为硬件可理解的格式，即进行数据格式转换。一些系统控制参数也以同样的方式按格式写入表示寄存器的变量中。然后进行一次拷贝，即生成了驱动硬件动作的控制流和参数流，传送到硬件内部[6]。下面以状态控制寄存器设置为例做具体说明。用 32bit 的变量 reg_ctrl 代表寄存器，结构体 pStatus 的成员变量 encode_de-

code 和 start_stop 分别表示编码/解码和开始/停止状态。另一个结构体 pStreaminfo 的成员变量 stream_type，stream_size，stream_add，stream_offset 分别表示待解码流的类型、大小、存放基地址和偏移地址，则对该寄存器各位写入示例可表示为

```
reg_ctrl=0x00000000
        | ((M_01BIT&pStatus->encode_decode) << 31)
        | ((M_01BIT&pStatus->start_stop) << 30)
        | ((M_02BIT&pStreaminfo->stream_type) << 28)
        | ((M_08BIT&pStreaminfo->stream_size) << 20)
        | ((M_04BIT&0) << 16)
        | ((M_08BIT&pStreaminfo->stream_add) << 8)
        | ((M_08BIT&pStreaminfo->stream_offset)<< 0)
```

上式中 M_0XBIT 是一个特别定义的宏，可以通过和它进行“与”运算来取低位 Xbit 的值。通过用类似的方法设置其他寄存器，就可完成数据的格式转换，使之适应硬件对输入数据的要求。

4 硬件设计与实现

当操作中包含大量重复、迭代计算或者简单的条件跳转时，通过相对简单的硬件结构即可达到很高的处理速度。因此在本设计中，使用硬件来解析位于宏块层的用 ce(v)描述的语法元素及剩余的少量用定长码和指数哥伦布码编码的语法元素。解析程序用本设计开发平台适用的汇编语言编写，编译后生成的二进制文件下载到硬件平台运行。

4.1 CA-2D-VLC 解码过程

相对于 CA-2D-VLC 码的编码过程，当从码流中解析残差数据对(Run，Level)时，具体方法如下，同时也是本文提出的可变长码表设计思想及优化的理论基础。

如果 trans_coefficient 小于 59：

1)如果 CurrentVLCTable 中包含此 trans_coefficient 值的索引项，则以 trans_coefficient 为索引查 CurrentVLCVTable，得到量化系数和游程，把它们分别放入 Level 数组和 Run 数组；

2)如果表中不包含此 trans_coefficient 值的索引项，则以(trans_coefficient-1)为索引查 CurrentVLCTable，得到量化系数和游程，并将量化系数符号取反后放入 Level 数组，将游程放入 Run 数组。

如果 trans_coefficient 大于或等于 59：

1)将(trans_coefficient-59)/2 放入 Run 数组；

2)由 CurrentVLCTable 查表得到 MaxRun。如果(trans_coefficient-59)/2 大于 MaxRun，RefAbsLevel 等于 1；否则以(trans_coefficient-59)/2 为索引查 CurrentVLCTable 得到 RefAbsLevel；

3)如果 trans_coefficient 是奇数，将－(RefAbsLevel＋escape_level_diff)放入 Level 数组；否则，将(RefAbsLevel＋escape_level_diff)放入 Level 数组[2,7]。

4.2 码表设计与优化

如 4.1 节所述，使用 CA-2D-VLC 编码的语法元素在解析时需要存储码表及进行大量查表操作，同时会对指数哥伦布码的解码操作产生更多的硬件消耗。因此，码表的合理设计和优化对于降低硬件消耗是非常必要的。相对于数据对(Run，Level)的生成过程，码表设计分以下几个方面：

1)当 trans_coefficient 小于 59 并且处在当前码表中时，则新设计码表与原码表保持一致；

2)当 trans_coefficient 小于 59 但不处于当前码表中时，则 Level 和 Run 的值不能够直接从码表中查得，之后的减法及符号取反操作会带来额外的硬件消耗。因此，码表的第一个优化方案就是将未出现在原码表中的 trans_coefficient 值也包含进新码表中，同时包括对应取反的 Level 值。原始码表如图 4 所示，新码表以该码表为例做出优化与整合，如图 5 所示。经过这种新的设计之后，数据对(Run，Level)通过一次查表操作即可获得。

Run	EOB	Level＞0						RefAbsLevel
	8	1	2	3	4	5	6	
0	-	0	4	15	27	41	55	7
1	-	2	17	35	-	-	-	4
2	-	6	25	53	-	-	-	4
3	-	9	33	-	-	-	-	3
4	-	11	39	-	-	-	-	3
5	-	13	45	-	-	-	-	3
6	-	19	49	-	-	-	-	3
7	-	21	51	-	-	-	-	3
8	-	23	-	-	-	-	-	2
9	-	29	-	-	-	-	-	2
10	-	31	-	-	-	-	-	2
11	-	37	-	-	-	-	-	2
12	-	43	-	-	-	-	-	2
13	-	47	-	-	-	-	-	2
14	-	57	-	-	-	-	-	2

图 4 原始码表

trans_coefficient (Index)	标志 31～24位	Level绝对值 23～16位	有符号Level值 15～8位	Run值 7～0位
0x0000	下一码表号及出错标志	1	1	0
0x0001		1	-1	0
⋮		⋮	⋮	⋮
0x0039		1	1	14
0x003A		1	-1	14
>=59		RefAbsLevel	有符号RefAbsLevel	(Index-59)/2
0x003B	⋮	7	-7	0
0x003C		7	7	0
⋮		⋮	⋮	⋮
0x00B9		1	-1	63
0x00BA		1	1	63

图 5 新码表设计图

3) 当 trans_coefficient 大于或等于 59 时，Run 值需要经过进一步算术运算才能获得，因此码表的第二个优化方案就是直接包含 Run 值的最终计算结果，使得在此种情况下 Run 值也可以通过一次查表操作获得。如图 5 所示，第 5 列就是(trans_coefficient-59)/2 的直接计算结果，即此时 Run 的值。RefAbsLevel 的符号是由 trans_coefficient 值的奇偶性决定的，设计时对其进行了反向判别。即如果第 3 列与第 4 列值相同，则此时的 trans_coefficient 值为偶，反之为奇数。查表得到的 RefAbsLevel 将用于最终计算 Level 的值。

4) 与原始码表不同的是，新码表并未包含 EOB 码，而是在程序编写时单独添加了一条判别语句。这不但易于实现，而且节省了分配给原始码表中每张码表 EOB 码的存储空间。

4.3 硬件解析结果

根据 AVS 视频标准的描述，宏块层语法元素解析时会用到本设计中软件部分的解析结果，用作判断条件或计算时的中间量，这些值可以从驱动层传递的参数流中获取。硬件解析结果的输出与软件部分处理类似，即生成涵盖所有解析结果和有用的控制信息的信息流，并以一定格式输出。

5 实验结果与验证

为验证系统结构的合理性和算法的正确性，用C语言和C++语言混合编程搭建了完整的软件模拟平台C-model，作为仿真验证平台。硬件设计中所有的结构，包括寄存器和内部总线等在C-model上都用对应的变量、结构体和类来表示。C-model通过完整的逻辑关系联系覆盖了所有硬件内部实体功能和动作。此外，C-model提供了各种信号量用于仿真时的时序测试。所以，之前的软件和硬件设计只有通过了C-model平台的测试才能保证其合理性。

另一方面，为了进一步确保输出结果的正确性，在参考AVS官方软件解码程序RM52J的基础上改写了纯软件视频解码验证器，并把解码出的结果进行格式变换，生成要求格式的信息流。验证器输出的信息流与C-model生成的信息流格式一致，在输入同一AVS原始码流文件前提下，若对比两者输出的信息流内容完全一致，则证明解码是正确的。

由于本文设计的可变长解码器基于软硬件协同方法，因此软件部分的解析与控制信息设置是否正确直接关系到最后的输出结果。因此，在程序编写时使用条件编译的方式添加打印语句，用于查看控制台下打印结果，以确保软件解析准确无误。

经过用AVS工作组发布的92个一致性测试码流序列对本文设计的解码器进行测试后，确认本可变长解码器输出的信息流与相应验证器输出完全一致。

同时，为验证系统性能，在C-model中添加了测试代码，用于统计硬件运行周期数，以记录硬件消耗。由于在硬件内存允许范围内采用了新的码表格式设计，如4.2节所述，大量的算术运算可以通过一次查表操作实现，使得硬件损耗的周期数相较采用原始码表时减少了30%以上。

结束语 通过剖析AVS视频标准中对语法元素的定义及AVS码流构成模式，充分挖掘软硬件协同处理方法应用于视频解码器的优势，设计出可同时用于解析定长码、指数哥伦布码和CA-2D-VLD码的可变长码解码器。

针对不同编码方式的特点，详细阐述了对应的软件和硬件解析方法及设计流程。通过分析AVS可变长码表的构成，找出相应解码规则，设计新的码表并减少了存储空间和硬件损耗。经测试，本设计在合理性、正确性及性能等方面均达到了AVS视频解码要求，可用于解析AVS标准基准档次2.0～6.2级别的视频流。

参 考 文 献

[1] 彭聪. 多模数字视频解码SOC芯片设计及研究[D]. 北京：中国科学院计算机技术研究所，2006

[2] 数字音视频编码技术标准工作组(AVS). 高级音视频编码，第二部分：视频[S]. 中国，2004

[3] Sheng Bin, Gao Wen, Xie Don, et al. An Efficient VLSI Architecture of VLD for AVS HDTV Decoder[J]. IEEE Trans. on Consumer Electronics, 2006, 52: 696-701

[4] 王智. AVS视频编码器中熵编码的研究[D]. 哈尔滨：哈尔滨工业大学，2007

[5] Liu Wei, Chen Yong-en. VLD Design for AVS Video Decoder [C]// IEEE Proceedings of Second International Workshop on Knowledge Discovery and Data Mining. WKDD, 2009: 648-651

[6] 周密. AVS解码器中间件及驱动程序的设计与实现[D]. 上海：同济大学中德学院，2009

[7] Wu Di, Gao Wen, Hu Mingzeng, et al. An Exp-Golomb encoder and decoder architecture for JVT/AVS[C]// Proc. 5th International Conference on ASIC. 2003: 910-913

(上接第290页)

由于笛卡尔-球空间坐标变换的角度θ和ϕ通过RGB比例得到，因此在分母趋近零时，变换呈现退化现象。例如R=G=B=0的纯黑色在球空间内$r=0$，θ和ϕ可以在值域内任意取值，因此图像中颜色近黑的部分在球空间角度灰度图中噪点激增，很难与原始灰度本身较大的部分区分开，造成后续分割过程的欠分割和无分割，如图17中的圆圈标记。

结束语 针对原始DCNON网络耦合结构复杂、运算量庞大的不足，根据原网络的动力学模型提出了基于衍射扩散检测环结构的改进的DCNON网络分割算法；给出算法流程，分析算法的特点，并与原始分割算法加以比较；将改进的DCNON网络分割算法与色彩的笛卡尔-球空间坐标变换相结合，分析笛卡尔-球空间坐标变换的特点，提出一种混合的彩色图像分割算法；分别对改进的DCNON网络与混合分割算法进行仿真，结果表明改进的DCNON网络分割算法能有效地压缩运算时间，混合算法分割图像不受光照、阴影和纹理等因素的干扰，能得到较好的分割结果。

参 考 文 献

[1] von der Malsburg C. The correlation theory of brain function, Gottingen [R]. 81-2, Max-Planck-Institute for Biophysical Chemistry Internal Report, 1981

[2] von der Malsburg C, Schneider W. A neural cocktail-party processor[J]. Biological Cybernetics, 1986, 54: 29-40

[3] Wang D L, Terman D. Locally excitatory globally inhibitory oscillator networks[J]. IEEE Transaction on Neural Networks, 1995, 6: 283-286

[4] Wang D L, Terman D. Images segmentation based on oscillatory correlation[J]. Neural Computation, 1997, 9: 1623-1626

[5] von der Malsburg C, Schneider W. A neural cocktail-party processor[J]. Biological Cybernetics, 1986, 54(1): 29-40

[6] Hu Hai. Representing RGB in sphere coordinate [A]// Proceedings of the 2008 2nd International Symposium on Information Technologies and Applications in Education [C]. (ISITAE). 2008: 132-136

[7] Chen Ke, Wang De-liang. A dynamically coupled neural oscillator network for image segmentation [J]. Neutal Networks, 2002, 15: 423-439

[8] 段华. 室外移动机器人视觉导航关键技术研究[D]. 南京：南京航空航天大学，2007

[9] Chaabane S B, Sayadi M, Fnaiech F, et al. Color Image Segmentation Using Automatic Thresholding and the Fuzzy C-Means Techniques[C]// Ajaccio. May 2008: 857-861

[10] Grant R N, Green R D, Clark A J. HLS Distorted Colour Model for Enhanced Colour Image Segmentation[C]// Christchurch. Nov. 2008: 1-6