(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102572437 A
(43)申请公布日 2012.07.11

(21)申请号 201210034976.1

(22)申请日 2012.02.16

(71)申请人 复旦大学
地址 200433 上海市杨浦区邯郸路 220 号

(72)发明人 范益波 沈沙 沈蔚炜 曾晓洋

(74)专利代理机构 上海正旦专利代理有限公司 31200
代理人 陆飞 盛志范

(51)Int.Cl.
H04N 7/26(2006.01)

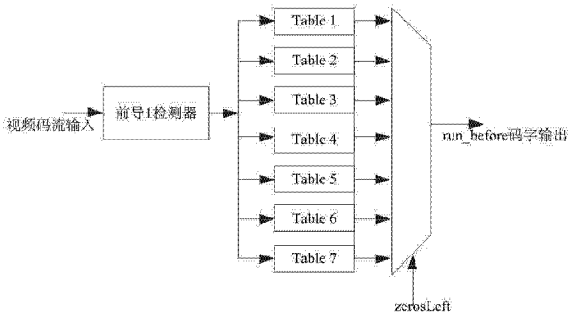
权利要求书 3 页 说明书 6 页 附图 1 页

(54)发明名称

一种快速解码CAVLC run_before码字的硬件实现方法

(57)摘要

本发明属于数字视频信号编解码技术领域，具体为一种快速解码 CAVLC run_before 码字的硬件实现方法。本方法采用码表分拆和码字合并这两种方法，在一个时钟周期内最多可以解码 14 个值为 0 的 run_before 码字，或两个值不等于 0 的 run_before 码字。输入的视频码流首先经过前导 1 检测器来检测带解码的 run_before 码字是否为零，前导 1 检测器最多一次可以检测 14 位前导 1。当前 run_before 码字的值不为零，如果我们将当前解码位置之前所包含的 0 的总个数记为 zerosLeft，根据 zerosLeft 的值可以将 run_before 码字分为 7 种类型，将原标准中规定的 run_before 码表分解为 7 个子码表，针对每一种类型分别进行解码。本发明可以大大提高硬件解码器的数据处理能力。



CN 102572437 A

1. 一种快速解码 CAVLC run_before 码字的硬件实现方法，其特征在于输入的视频码流首先经过前导 1 检测器检测带解码的 run_before 码字是否为零，前导 1 检测器最多一次可以检测 14 位前导 1；将当前解码位置之前所包含的 0 的总个数记为 zerosLeft，根据 zerosLeft 的值将 run_before 码字分为 7 种类型，即将原标准中规定的 run_before 码表分解为 7 个子码表，针对每一种类型分别进行解码；具体步骤为：

（1）码表分拆和码字合并；

（2）检测前导 1 的个数；

（3）根据 zerosLeft 的值从 7 个码表的输出选择出正确的值；

（4）更新 zerosLeft 变量，即每次解码完 run_before 码字后对 zerosLeft 变量进行更新，在下次解码时使用更新后的值；

其中，所述码表分拆和码字合并，是将 H.264 视频标准中规定用于解码 run_before 码字的码表分拆和合并如下表 2—表 8 所示的 7 个表：

表 2：针对 zerosLeft>6 时的码表

run_before	zerosLeft>6
0	111
1	110
2	101
3	100
4	011
5	010
6	001
7	0001
8	00001
9	000001
10	0000001
11	00000001
12	000000001
13	0000000001
14	00000000001

表 3：针对 zerosLeft=6 时的码表

	Run_before0					
run_before1	6	5	4	3	2	1
0	100-	1011	0101	01111	00111	00011
1	-	1010	01001	01110	00110	00010
2	-	-	01000	01101	00101	000011
3	-	-	-	01100	001001	000010
4	-	-	-	-	001000	000001
5	-	-	-	-	-	000000

表 4：针对 zerosLeft=5 时的码表

run_before1	Run_before0				
	5	4	3	2	1
0	000	0011	0101	01111	1011
1	-	0010	01001	01110	1010
2	-	-	01000	01101	1001
3	-	-	-	01100	10001
4	-	-	-	-	10000

表 5 :针对 zerosLeft=4 时的码表

run_before1	Run_before0			
	4	3	2	1
0	000	0011	011	1011
1	-	0010	0101	1010
2	-	-	0100	1001
3	-	-	-	1000

表 6 :针对 zerosLeft=3 时的码表

run_before1	Run_before0		
	3	2	1
0	00	011	101
1	-	010	1001
2	-	-	1000

表 7 :针对 zerosLeft=2 时的码表

run_before1	Run_before0	
	2	1
0	00	011
1	-	010

表 8 :针对 zerosLeft=1 时的码表

run_before	
0	1
1	0

。

2. 根据权利要求 1 所述的快速解码 CAVLC run_before 码字的硬件实现方法，其特征在于所述检测前导 1 的个数的步骤为 :在解码 run_before 码字时，首先检测当前码流中前导 1 的个数 ;前导 1 检测器最多可以一次检测 14 个前导 1 ;

如果输入的视频码流第一个比特为 0，那么前导 1 的个数为零，说明当前 run_before 的值大于零，此过程结束，进入步骤 (3) ;否则，根据前导 1 的个数同时解码多个 run_before

码字，这些 run_before 码字的解码值都是 0；将前导 1 的个数记为 N，如果 zerosLeft>6 且 N ≥ 3，那么包括当前码字在内的连续 N/3 个 run_before 码字的值都是 0；如果 2<zerosLeft<6 且 N ≥ 2，那么包括当前码字在内的连续 N/2 个 run_before 码字的值都是 0；如果 zerosLeft ≤ 2，那么包括当前码字在内的连续 N 个 run_before 码字的值都是 0；上述计算中，如果 N 不是 3 或 2 的整数倍，N/3 和 N/2 取其运算结果的整数部分。

3. 根据权利要求 2 所述的快速解码 CAVLC run_before 码字的硬件实现方法，其特征在于所述根据 zerosLeft 的值从 7 个码表的输出选择出正确的值的具体步骤为：

zerosLeft 的取值范围分为 7 种：大于 6，6，5，4，3，2，1；对应的码表分别为表 2、表 3、表 4、表 5、表 6、表 7、表 8；根据当前 zerosLeft 的值，选择相应的码表输出作为最终的解码结果，每查询一次表 2 或表 8，得到一个 run_before 码字的值，每查询一次表 3、表 4、表 5、表 6 或表 7，得到两个 run_before 码字的值。

4. 根据权利要求 3 所述的快速解码 CAVLC run_before 码字的硬件实现方法，其特征在于所述更新 zerosLeft 变量的具体步骤为：

如果当前 zerosLeft 等于 1 或者大于 6，根据步骤（3）中解码得到当前 run_before 码字的值，zerosLeft 变量减去当前 run_before 码字在解码后得到的值，即为新的 zerosLeft 变量；如果当前 zerosLeft 等于 2、3、4、5、6 中任意一个值，根据步骤（3）中得出的两个 run_before 码字解码后的值，zerosLeft 变量减去这两个码字值的和，即为新的 zerosLeft 变量。

一种快速解码 CAVLC run_before 码字的硬件实现方法

技术领域

[0001] 本发明属于数字视频信号编解码技术领域，具体涉及一种解码 CAVLC run_before 码字的硬件实现方法。

背景技术

[0002] H.264/AVC(Advanced Video Coding) 由国际电信组织(ITU)和运动图像专家组(MPEG)联合制定而成的国际视频编码标准，目前已经在多媒体音视频领域得到了广泛的应用。H.264/AVC 中规定其熵编码可以采用两种方式：下文自适应可变长编码(CAVLC)和上下文自适应算术编码 (CABAC)。CABAC 具有较高的编码效率，但是编解码的复杂度也大大增加，而 CAVLC 在编码效率和复杂度上有着较好的均衡。与前一代视频标准如 MPEG-1, MPEG-2 相比，CAVLC 的编码效率有着显著提升。

[0003] 视频码流中的视频头信息和预测信息一般采用较低复杂度的定长码或指数哥伦布码，而残差系数的信息占据了视频码流的绝大部分，这部分信息的编码需要采用更高效的 CAVLC 编码方式。CAVLC 的码字总共有 5 类：

1. Coeff_token：本码字代表非零系数的数目和拖尾 1 的个数(TrailingOnes)。

[0004] 2. Sign_trail：本码字是拖尾 1 的符号。每一个符号编码为一比特，0 表示正 1，1 表示负 1。

[0005] 3. Levels：本码字是指 TrailingOnes 外的其它的非零系数的幅值。

[0006] 4. Total_zeros：它编码在以反 Zig-Zag 扫描顺序时第一个非零系数后的总的零系数的个数。

[0007] 5. Run_before：本码字是指每一个非零系数前的零的个数。

[0008] 对于硬件解码而言，计算非零系数的幅值和 run_before 码字是整个 CAVLC 解码过程中最耗费时间的过程。对于一个长度和高度都为 4 个像素的图形块，如果采用 CAVLC 编码方式，有些码字如 Coeff_token、sign_trial 和 Total_zero 在解码过程中只需要一次解码，而 run_before 码字最多可以达 14 个。为了加快硬件处理的速度，有必要针对 run_before 码字提出一种快速的解码方法。

发明内容

[0009] 本发明的目的在于提出一种适用于 H.264 视频规范所规定的 CAVLC 解码过程的快速解码 run_before 码字的硬件实现方法。

[0010] 本发明提出的快速解码 run_before 码字的硬件实现方法，可以在一个时钟周期内最多完成两个 run_before 码字的解码。整个码字解码的硬件框图如图 1 所示。本方法采用了码表分拆和码字合并这两种方法，在一个时钟周期内最多可以解码 14 个值为 0 的 run_before 码字，或两个值不等于 0 的 run_before 码字。输入的视频码流首先经过前导 1 检测器来检测带解码的 run_before 码字是否为零，前导 1 检测器最多一次可以检测 14 位前导 1。如果当前 run_before 码字的值不为零，我们将当前解码位置之前所包含的 0 的总

个数记为 zerosLeft，根据 zerosLeft 的值可以将 run_before 码字分为 7 种类型，即将原标准中规定的 run_before 码表分解为 7 个子码表，针对每一种类型分别进行解码。每个时钟周期内最多可以解码 2 个 run_before 码字。

[0011] CAVLC 定义了多种码字，在编码一个 4x4 残差数据块时，run_before 码字最多可出现 14 次，因此这一硬件实现方法可以大大提高了硬件解码器的数据处理能力。

[0012] 具体的解码过程分为如下四个步骤：

（1）码表分拆和码字合并。

[0013] （2）检测前导 1 的个数。

[0014] （3）根据 zerosLeft 的值从 7 个码表的输出选择出正确的值。

[0015] （4）更新 zerosLeft 变量。每次解码完 run_before 码字后对 zerosLeft 变量进行更新，在下次解码时需要使用更新后的值。

[0016] H.264 视频规范中指定了一个用于解码 run_before 码字的码表，此码表如表 1 所示。本发明将此码表分解为 7 个较小的码表，如表 2—表 8 所示，并对其中的 5 个码表（表 3、表 4、表 5、表 6、表 7）进行码字合并，使得其能够在一个时钟周期内完成两个码字的解码。

[0017] CAVLC 定义了 6 种码字，其中非零系数幅值的码字和 run_before 码字出现的频率最高，在编码一个 4x4 残差数据块时 run_before 码字最多可出现 14 次，因此这一硬件实现方法可以大大提高硬件解码器的数据处理能力。

[0018] 本发明的有益效果：

现有方法在一个时钟周期内只能解码一个 run_before 码字，而本发明中描述的方法可以将速度加倍，在一个时钟周期内可以解码得到多个 run_before 码字的值，从而提高了 CAVLC 整体的解码速度。适用于各种具有 H.264 视频解码功能的电子设备。

附图说明

[0019] 图 1 ：run_before 码字的硬件解码框图。

具体实施方式

[0020] 下面结合附图对本发明做进一步的描述。

[0021] 本发明所述的快速解码 run_before 码字的具体实施方式如下：

（1）码表分拆和码字合并。

[0022] H.264 视频标准中规定用于解码 run_before 码字的码表如表 1 所示。查询一次此码表只能得到一个 run_before 码字的值。为了加速此解码过程，我们将此码表按照 zerosLeft 的值分拆成 7 个码表。表 2 是针对 zerosLeft>6 时的码表。表 3 是针对 zerosLeft=6 时的码表。表 4 是针对 zerosLeft=5 时的码表。表 5 是针对 zerosLeft=4 时的码表。表 6 是针对 zerosLeft=3 时的码表。表 7 是针对 zerosLeft=2 时的码表。表 8 针对 zerosLeft=1 时的码表。其中表 3、表 4、表 5、表 6、表 7 采用了码字合并的技巧，每次查询这 5 个表可以得到 2 个 run_before 码字。而表 2 和表 8，则是从表 1 中直接分拆得到，每次查询表 2 和表 8，只能得到 1 个 run_before 码字。

[0023] （2）检测前导 1 的个数。

[0024] 在解码 run_before 码字时，首先检测当前码流中前导 1 的个数。本发明中的前导

1 检测器最多可以一次检测 14 个前导 1。如果输入的视频码流第一个比特为 0，那么前导 1 的个数为零，说明当前 run_before 的值大于零，此过程结束，进入步骤 (3)。否则，根据前导 1 的个数可同时解码多个 run_before 码字，这些 run_before 码字的解码值都是 0。我们将前导 1 的个数记为 N，如果 zerosLeft>6 且 N ≥ 3，那么包括当前码字在内的连续 N/3 个 run_before 码字的值都是 0；如果 2<zerosLeft<6 且 N ≥ 2，那么包括当前码字在内的连续 N/2 个 run_before 码字的值都是 0； 如果 zerosLeft ≤ 2，那么包括当前码字在内的连续 N 个 run_before 码字的值都是 0。上述计算中，如果 N 不是 3 或 2 的整数倍，N/3 和 N/2 取其运算结果的整数部分。本发明中 N 最大值为 14，所以前导 1 检测器最多一次可以解码 14 个值为零的 run_before 码字。

[0025] (3) 根据 zerosLeft 的值从 7 个码表的输出选择出正确的值。

[0026] zerosLeft 的取值范围可以分为 7 种：大于 6, 6, 5, 4, 3, 2, 1。此时对应的码表分别为表 2、表 3、表 4、表 5、表 6、表 7、表 8。根据当前 zerosLeft 的值，选择相应的码表输出作为最终的解码结果。每查询一次表 2 或表 8，可以得到一个 run_before 码字的值，而每查询一次表 3、表 4、表 5、表 6 或表 7，可以得到两个 run_before 码字的值。

[0027] (4) 更新 zerosLeft 变量。

[0028] 如果当前 zerosLeft 等于 1 或者大于 6，根据步骤 (3) 中解码得到当前 run_before 码字的值，zerosLeft 变量减去当前 run_before 码字在解码后得到的值，即为新的 zerosLeft 变量；如果当前 zerosLeft 等于 2、3、4、5、6 中任意一个值，根据步骤 (3) 中得出的两个 run_before 码字解码后的值，zerosLeft 变量减去这两个码字值的和，即为新的 zerosLeft 变量。

[0029] 经过上述四个步骤，即可在一个时钟周期内解码两个 run_before 码字。此发明中提出的硬件架构可以大大加快 H.264 CAVLC 的解码速度。

[0030] 表 1：H.264 标准中规定用于解码 run_before 码字码表

run_before	zerosLeft						
	1	2	3	4	5	6	>6
0	1	1	11	11	11	11	111
1	0	01	10	10	10	000	110
2	-	00	01	01	011	001	101
3	-	-	00	001	010	011	100
4	-	-	-	000	001	010	011
5	-	-	-	-	000	101	010
6	-	-	-	-	-	100	001
7	-	-	-	-	-	-	0001
8		-	-	-	-	-	00001
9	-	-	-	-	-	-	000001
10	-	-	-	-	-	-	0000001
11	-	-	-	-	-	-	00000001
12	-	-	-	-	-	-	000000001
13	-	-	-	-	-	-	000000000
14	-	-	-	-	-	-	000000000

表 2 :针对 zerosLeft>6 时的码表(Table 1)

run_before	zerosLeft>6
0	111
1	110
2	101
3	100
4	011
5	010
6	001
7	0001
8	00001
9	000001
10	0000001
11	00000001
12	000000001
13	0000000001
14	00000000001

表 3 :针对 zerosLeft=6 时的码表(Table 2)

run_before1	Run_before0					
	6	5	4	3	2	1
0	100-	1011	0101	01111	00111	00011
1	-	1010	01001	01110	00110	00010
2	-	-	01000	01101	00101	000011
3	-	-	-	01100	001001	000010
4	-	-	-	-	001000	000001
5	-	-	-	-	-	000000

表 4 :针对 zerosLeft=5 时的码表(Table 3)

run_before1	Run_before0				
	5	4	3	2	1
0	000	0011	0101	01111	1011
1	-	0010	01001	01110	1010
2	-	-	01000	01101	1001
3	-	-	-	01100	10001
4	-	-	-	-	10000

表 5 :针对 zerosLeft=4 时的码表(Table 4)

run_before1	Run_before0			
	4	3	2	1
0	000	0011	011	1011
1	-	0010	0101	1010
2	-	-	0100	1001
3	-	-	-	1000

表 6 :针对 zerosLeft=3 时的码表(Table 5)

run_before1	Run_before0		
	3	2	1
0	00	011	101
1	-	010	1001
2	-	-	1000

表 7 :针对 zerosLeft=2 时的码表(Table 6)

run_before1	Run_before0	
	2	1
0	00	011
1	-	010

表 8 :针对 zerosLeft=1 时的码表(Table 7)

run_before	
0	1
1	0

。

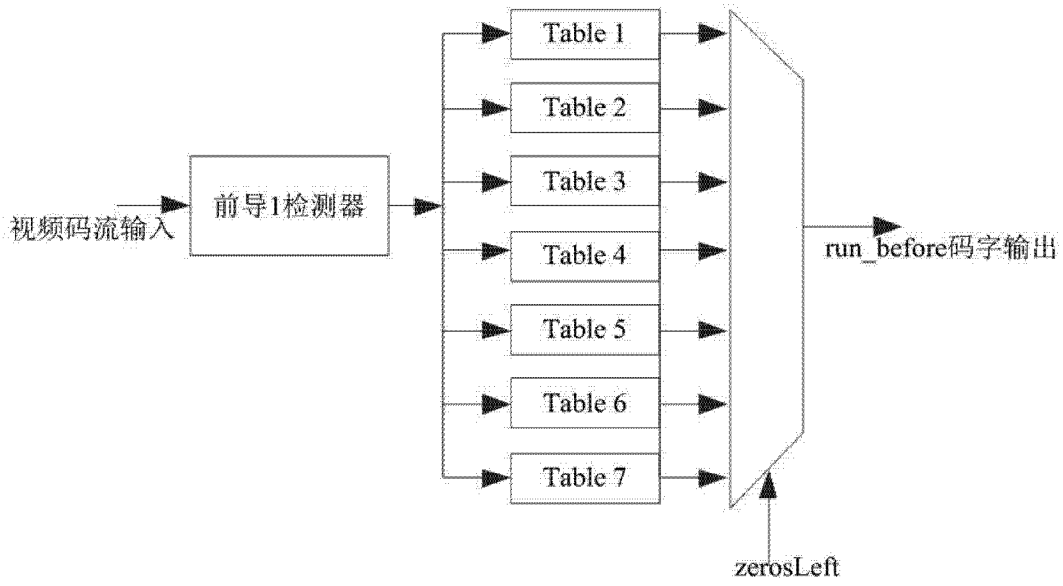


Table 1
Table 2
Table 3
Table 4
Table 5
Table 6
Table 7
前导1检测器
视频码流输入
run_before码字输出
zerosLeft

图 1