[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04N 7/26 (2006.01)
H04N 7/30 (2006.01)
H04N 7/50 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200610157417.4

[43] 公开日 2008 年 6 月 11 日

[11] 公开号 CN 101198051A

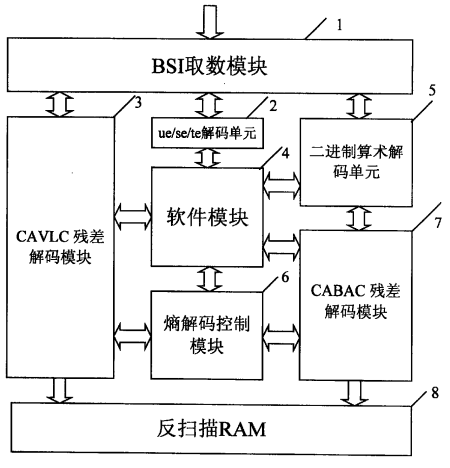
[22] 申请日 2006.12.7
[21] 申请号 200610157417.4
[71] 申请人 深圳艾科创新微电子有限公司
地址 518057 广东省深圳市南山区高新区科技中二路软件园一期 4 栋 406 室
[72] 发明人 何铁军 汤加跃 石 岭
权利要求书 5 页 说明书 14 页 附图 6 页

[54] 发明名称

基于 H.264 的熵解码器的实现方法及装置

[57] 摘要

本发明公开了一种专用于 H.264 解码 MAE 中对熵编码残差数据等进行解码的方法及装置，由于本发明所述方法采用了软硬件协同工作，将解码过程中占用大量资源的部分改用软件方法实现。 所述的软件程序方法是指利用嵌入式 cpu 代替硬件电路进行运算的方式，使得熵解码器在保证获得可靠的解码效率和解码质量的前提下，大大降低了结构的复杂度和计算复杂度，从而有效的解决了结构和效率之间的矛盾，顺利的实现了对 CAVLC 和 CABAC 熵编码码流的解码，所述该熵解码装置包括 BSI 取数模块、ue/se/te 解码单元、CAVLC 残差解码模快、软件模块、二进制算术解码单元、熵解码控制模块、CABAC 残差解码模块、反扫描 RAM。



1、基于 H. 264 的熵解码器的实现方法，其特征在于：该方法采用了软硬件协同工作，将解码过程中占用大量资源的部分改用软件程序方法实现，包括如下步骤：

步骤 1：熵解码控制模块（6）根据熵编码模式标志选择解码方式：即是对 CAVLC 编码码流进行解码还是对 CABAC 编码码流进行解码；

步骤 2：根据步骤（1）选定的解码类型以片（slice）数据为单位对 CAVLC 或者 CABAC 编码码流进行解码；

步骤 3：解码完成后，判断是否为 CAVLC 或者 CABAC 编码码流的最后一个片数据，若不是，返回步骤（2）继续对下一个片（Slice）的数据进行解码，否则，解码过程结束。

2、根据权利要求 1 所述的基于 H. 264 的熵解码器的实现方法，其特征在于：所述的将解码过程中占用大量资源的部分改用软件程序方法实现是指利用嵌入式 cpu 代替硬件电路进行运算的方式，而不是单纯的指用计算机软件实现熵解码中的某些计算功能。

3、根据权利要求 1 所述的基于 H. 264 的熵解码器的实现方法，其特征在于：该方法步骤（2）中所述的根据选定的解码类型以片数据为单位对 CAVLC 编码码流进行解码，其中对 CAVLC 编码码流的解码还包括如下的具体步骤：

步骤 2.1：熵解码控制模块（6）向软件模块（4）发出开始信号，开始求解与解码有关的变量和参数；

步骤 2.2：软件模块（4）收到来自熵解码控制模块（6）的开始信号后，经由 ue/se/te 解码单元（2）并最终通过 BSI 取数模块（1）得到所需的编码数据流，经解码得到所需的宏块参数以及除残差系数以外的其他语法元素；

步骤 2.3：熵解码控制模块（6）收到软件模块（4）发出的结束信号后，向

CAVLC 残差解码模块（3）发出解残差开始信号；

步骤 2.4：CAVLC 残差解码模块（3）收到熵解码控制模块（6）发来的开始信号后开始残差解码，并通过 BSI 取数据模块（1）获得解码所需的编码数据，得到亮度块和色度块的残差系数值；

步骤 2.5：判断解码的是否为片数据的最后一个宏块，如果不是，则返回到步骤（2.1）继续解下一个宏块，否则解码结束。

4、根据要求 1 所述的基于 H. 264 的熵解码器的实现方法，其特征在于：其特征在于：以片数据为单位对 CAVLC 编码码流进行解码是通过对组成片数据的每个宏块进行解码实现的，所述对单个宏块的解码包括如下的步骤：

步骤 1：由软件模块求解与宏块解码相关的变量和参数；

步骤 2：根据上述步骤（1）得到解码所需的当前宏块以及邻近宏块等解码所需的参数后，按照先亮度块后色度块的顺序对宏块进行解码；

步骤 3：对亮度块进行解码；以 4x4 块为基本的解码单位，即每次对一个 4x4 亮度块进行块解码，先对亮度块的直流(DC)系数进行解码，然后再对亮度块的交流系数(AC)进行解码；

步骤 4：解码完亮度块后，对色度块进行解码；以 4x4 块为基本的解码单位，按照先 Cb 块后 Cr 块的顺序，先解码色度块的直流系数，再解码色度块的交流系数；

步骤 5：重复上述步骤，继续解下一个宏块。

5、根据权利要求 1 所述的基于 H. 264 的熵解码器的实现方法，其特征在于：该方法步骤（2）所述的根据选定的解码类型以片数据为单位对 CABAC 编码码流进行解码，对 CABAC 编码码流的解码还包括如下的具体步骤：

步骤 2.1′：对 CABAC 解码所需的上下文模型进行初始化，并初始化概率

区间；

步骤 2.2′ ：熵解码控制模块（6）向软件模块（4）发出开始信号，开始对除残差系数以外的其他语法元素、运动向量 mvd_l0,mvd_l1 等解码，并求解残差解码中用到的宏块参数；

步骤 2.3′ ：软件模块（4）收到来自熵解码控制模块（6）的开始信号后，经由二进制算术解码单元（5）并最终通过 BSI 取数模块（1）得到所需的编码数据流，解码得到所需的宏块参数以及除残差系数以外的其他语法元素；

步骤 2.4′ ：熵解码控制模块（6）向 CABAC 残差解码模块（7）发出解残差开始信号；

步骤 2.5′ ：CABAC 残差解码模块（7）收到熵解码控制模块（6）发来的开始信号后开始残差解码，经由二进制算术解码单元（5）并最终通过 BSI 取数模块（1）得到所需的编码数据流，解码得到亮度（luma）和色度（chroma）块的残差系数值；

步骤 2.6′ ：判断解码的宏块是否为片（slice）的最后一个宏块，如果不是，则返回到步骤（2.1′ ）继续解下一个宏块，否则解码结束。

6、根据权利要求 1 所述的基于 H. 264 的熵解码器的实现方法，其特征在于：所述的以片数据为单位对 CABAC 编码码流进行解码是通过对组成片数据的每个宏块进行解码实现的，所述对单个宏块的解码包括如下的步骤：

步骤 1：对上下文模型和概率区间进行初始化；

步骤 2：开始对残差系数以外其他语法元素进行解码，并通过采用软件方法求解残差解码中要用到的宏块参数和其他参数；

步骤 3：按照先亮度后色度的顺序对残差宏块进行解码；

步骤 4：对亮度块的解码；以 4x4 块或者 8x8 块为基本单位，先对亮度直流

系数进行解码，再对剩余的亮度交流系数进行解码，完成对亮度块的解码过程；

步骤 5：对色度块的解码；按照先 Cb 后 Cr 块的顺序以 4x4 块为单位对两个色度块 Cb 和 Cr 块进行解码，先解 Cb 和 Cr 块的直流系数，然后再解 Cb 和 Cr 块的交流系数；

步骤 6：重复上述步骤，继续解下一个宏块。

7、基于 H.264 的熵解码器装置，该装置包括如下模块：

BSI 取数模块：负责从外部取得解码所需的码流数据，当接收到其他模块的取数请求时，返回相应长度的码流数据；

ue/se/te 解码单元：接收软件模块发送来的解码请求并返回解码后的参数值；

CAVLC 残差解码模快：以‘4X4 块’为单位，根据软件模块解码得到的相应参数，对亮度和色度的残差系数进行解码，解码完成后向熵解码控制模块发出结束信号，解码出来的残差系数存入反扫描 RAM；

软件模块：对残差解码需要的参数和变量进行解码，通过 ue/se/te 解码单元求解与 CAVLC 相关的参数，通过二进制算术解码单元求解与 CABAC 相关的参数，最终得到的参数再提供给 CAVCL 或 CABAC 残差解码模块使用；

二进制算术解码单元：在解码与宏块有关的参数时，当收到软件模块发送来的解码请求，经解码返回得到的二进制结果；另外在进行 CABAC 残差解码的时，当收到 CABAC 残差解码模快发送来的解码请求时，经解码返回得到的二进制结果；

熵解码控制模块：控制整个解码流程，先控制软件模块求解相关的参数，负责向 CAVLC 残差解码模块或者 CABAC 残差解码模块发出开始解码信号，当接收到解码结束信号后，将解码得到的相应残差系数写入反扫描 RAM；

CABAC 残差解码模块：以‘4X4’块或者‘8X8’块为单位，根据软件模块解码得到的相应参数，对亮度和色度的残差系数进行解码，解码完成后向熵解码控制模块发出结束信号，解码出来的残差系数存入反扫描 RAM；

反扫描 RAM：存储经 CAVLC 残差解码模快或者 CABAC 残差解码模块解码得到的残差系数。

基于H.264的熵解码器的实现方法及装置

技术领域

本发明属于集成电路数字多媒体处理技术领域，尤其涉及一种专为在视频通信的硬件MAE（media accelerator engine 多媒体加速引擎）上实现熵解码器的方法及装置。

背景技术

视频编解码技术是数字多媒体存储和传输应用的关键技术之一，视频编码的核心部分包括：预测编码，变换编码，熵编码。作为视频编码重要组成部分的熵编码，是利用去除视频数据冗余的方法达到对视频数据进行压缩的效果。

熵编码包括变长编码和算术编码两种编码方法。典型的变长编码包括：H.261、MPEG（Motion Picture Experts Group）—2中采用的H.263，MPEG-4中采用的3D-VLC编码，以及H.264/AVC（国际电信协会ITU-T的H.264视频编码建议或ISO/IEC的国际标准14496-10即MPEG-4标准的第十部分）中采用的CAVLC（Context-based Adaptive Variable Length Coding）编码。其中2D-VLC编码和3D-VLC编码采用单一码表进行编码，码表的体积较小，硬件实现比较简单；而CAVLC编码根据已编码的句法元素的情况，动态调整使用的码表，可以得到很高的压缩比，性价比更高。算术编码中最典型的是在H.264/AVC中采用CABAC算术编码，相对变长编码而言，由于可以对上下文进行建模，对信源符号出现的概率有更准确的估计，因此编码效率更高，但是相比变长编码，它的计算复杂度高，硬件实现复杂，因此目前采用不多。一般来说，CAVLC编

码多用于实时性要求较高的应用，如视频电话等。而 CABAC 虽然编码效率较高，编码的图像质量较好，但由于编码比较复杂，难以用于对速度要求较高的应用中，更多地用于数字多媒体的应用，如数字广播电视和数字视频存储等。

熵解码就是将熵编码编码过的视频数据准确可靠的还原，对于应用 H. 264/AVC 的视频编码标准进行编码的视频数据，针对存在的两种熵编码方式，熵解码器分别对经过 CAVLC 和 CABAC 编码过的视频数据进行解码。然而，由于 CAVLC 较以前的 2D-VLC 编码和 3D-VLC 编码复杂，CABAC 计算复杂度很高，因此要对同时支持两种熵编码方式的视频数据解码需要耗费相当的资源。在 IC 设计中，纯 ASIC 硬件实现也相当困难，也正是由于这个原因，当前的 H. 264 熵解码一般只是在计算机上采用纯软件的方式实现，少数硬件实现的熵解码器也只支持 CAVLC 编码方式，这样就限制了性能更好，压缩比更高的 CABAC 编码方法的应用。

发明内容

本发明所要解决的技术问题是如何采用一种简单有效的方法，在 MAE 上实现对 CAVLC 和 CABAC 的两种视频编码数据的熵解码，既能保证解码的质量和解码效率，结构简单又易于实现。

本发明的目的在于公开了一种专用于 H. 264 解码 MAE 中对熵编码残差数据等进行解码的方法及装置，由于本发明所述方法采用了软硬件协同工作，使得熵解码器在保证获得可靠的解码效率和解码质量的前提下，大大降低了结构的复杂度和计算复杂度，从而有效的解决了结构和效率之间的矛盾，顺利的实现了对 CAVLC 和 CABAC 熵编码码流的解码。

基于 H. 264 的熵解码器的实现方法，其特征在于：该方法采用了软硬件协

同工作，将解码过程中占用大量资源的部分改用软件程序方法实现，包括如下步骤：

步骤 1：熵解码控制模块（6）根据熵编码模式标志选择解码方式：即是对 CAVLC 编码码流进行解码还是对 CABAC 编码码流进行解码；

步骤 2：根据步骤（1）选定的解码类型以片（slice）数据为单位对 CAVLC 或者 CABAC 编码码流进行解码；

步骤 3：解码完成后，判断是否为 CAVLC 或者 CABAC 编码码流的最后一个片数据，若不是，返回步骤（2）继续对下一个片（Slice）的数据进行解码，否则，解码过程结束。

基于 H. 264 的熵解码器的实现方法，所述的将解码过程中占用大量资源的部分改用软件程序方法实现，其特征在于：所述的软件程序方法是指利用嵌入式 cpu 代替硬件电路进行运算的方式，而不是单纯的指用计算机软件实现熵解码中的某些计算功能。

所述的基于 H. 264 的熵解码器的实现方法，其特征在于：该方法步骤（2）中所述的根据选定的解码类型以片数据为单位对 CAVLC 编码码流进行解码，其中所述的对 CAVLC 编码码流的解码还包括如下的具体步骤：

步骤 2.1：熵解码控制模块（6）向软件模块（4）发出开始信号，开始求解与解码有关的变量和参数；

步骤 2.2：软件模块（4）收到来自熵解码控制模块（6）的开始信号后，经由 ue/se/te 解码单元（2）并最终通过 BSI 取数模块（1）得到所需的编码数据流，经解码得到所需的宏块参数以及除残差系数以外的其他语法元素；

步骤 2.3：熵解码控制模块（6）收到软件模块（4）发出的结束信号后，向 CAVLC 残差解码模块（3）发出解残差开始信号；

步骤 2.4：CAVLC 残差解码模块（3）收到熵解码控制模块（6）发来的开始信号后开始残差解码，并通过 BSI 取数据模块（1）获得解码所需的编码数据，得到亮度块和色度块的残差系数值；

步骤 2.5：判断解码的是否为片数据的最后一个宏块，如果不是，则返回到步骤（2.1）继续解下一个宏块，否则解码结束。

所述的基于 H. 264 的熵解码器的实现方法，其特征在于：以片数据为单位对 CAVLC 编码码流进行解码是通过对组成片数据的每个宏块进行解码实现的，所述对单个宏块的解码包括如下的步骤：

步骤 1：由软件模块求解与宏块解码相关的变量和参数；

步骤 2：根据上述步骤（1）得到解码所需的当前宏块以及邻近宏块等解码所需的参数后，按照先亮度块后色度块的顺序对宏块进行解码；

步骤 3：对亮度块进行解码；以 4x4 块为基本的解码单位，即每次对一个 4x4 亮度块进行块解码，先对亮度块的直流(DC)系数进行解码，然后再对亮度块的交流系数(AC)进行解码；

步骤 4：解码完亮度块后，对色度块进行解码；以 4x4 块为基本的解码单位，按照先 Cb 块后 Cr 块的顺序，先解码色度块的直流系数，再解码色度块的交流系数；

步骤 5：重复上述步骤，继续解下一个宏块。

所述基于 H. 264 的熵解码器的实现方法，该方法步骤（2）所述的根据选定的解码方式以片数据为单位对 CABAC 编码码流进行解码，其特征在于：其中所述的对 CABAC 编码码流的解码还包括如下的具体步骤：

步骤 2.1′：对 CABAC 解码所需的上下文模型进行初始化，并初始化概率区间；

步骤 2.2′：熵解码控制模块（6）向软件模块（4）发出开始信号，开始对除残差系数以外的其他语法元素、运动向量 mvd_l0,mvd_l1 等解码，并求解残差解码中用到的宏块参数；

步骤 2.3′：软件模块（4）收到来自熵解码控制模块（6）的开始信号后，经由二进制算术解码单元（5）并最终通过 BSI 取数模块（1）得到所需的编码数据流，解码得到所需的宏块参数以及除残差系数以外的其他语法元素；

步骤 2.4′：熵解码控制模块（6）向 CABAC 残差解码模块（7）发出解残差开始信号；

步骤 2.5′：CABAC 残差解码模块（7）收到熵解码控制模块（6）发来的开始信号后开始残差解码，经由二进制算术解码单元（5）并最终通过 BSI 取数模块（1）得到所需的编码数据流，解码得到亮度（luma）和色度（chroma）块的残差系数值；

步骤 2.6′：判断解码的宏块是否为片（slice）的最后一个宏块，如果不是，则返回到步骤（2.1′）继续解下一个宏块，否则解码结束。

本发明所述的基于 H. 264 的熵解码器的实现方法，其特征在于：所述的以片数据为单位对 CABAC 编码码流进行解码是通过对组成片数据的每个宏块进行解码实现的，所述对单个宏块的解码包括如下的步骤：

步骤 1：对上下文模型和概率区间进行初始化；

步骤 2：开始对残差系数以外其他语法元素进行解码，并通过采用软件方法求解残差解码中要用到的宏块参数和其他参数；

步骤 3：按照先亮度后色度的顺序对残差宏块进行解码；

步骤 4：对亮度块的解码；以 4x4 块或者 8x8 块为基本单位，先对亮度直流系数进行解码，再对剩余的亮度交流系数进行解码，完成对亮度块的解码过程；

步骤 5：对色度块的解码；按照先 Cb 后 Cr 块的顺序以 4x4 块为单位对两个色度块 Cb 和 Cr 块进行解码，先解 Cb 和 Cr 块的直流系数，然后再解 Cb 和 Cr 块的交流系数；

步骤 6：重复上述步骤，继续解下一个宏块。

本发明还公开了一种基于 H.264 的熵解码器装置，该装置包括如下模块：

BSI 取数模块：负责从外部取得解码所需的码流数据，当接收到其他模块的取数请求时，返回相应长度的码流数据；

ue/se/te 解码单元：接收软件模块发送来的解码请求并返回解码后的参数值；

CAVLC 残差解码模快：以‘4X4 块’为单位，根据软件模块解码得到的相应参数，对亮度和色度的残差系数进行解码，解码完成后向熵解码控制模块发出结束信号，解码出来的残差系数存入反扫描 RAM；

软件模块：对残差解码需要的参数和变量进行解码，通过 ue/se/te 解码单元求解与 CAVLC 相关的参数，通过二进制算术解码单元求解与 CABAC 相关的参数，最终得到的参数再提供给 CAVCL 或 CABAC 残差解码模块使用；

二进制算术解码单元：在解码与宏块有关的参数时，当收到软件模块发送来的解码请求，经解码返回得到的二进制结果；另外在进行 CABAC 残差解码的时，当收到 CABAC 残差解码模快发送来的解码请求时，经解码返回得到的二进制结果；

熵解码控制模块：控制整个解码流程，先控制软件模块求解相关的参数，负责向 CAVLC 残差解码模块或者 CABAC 残差解码模块发出开始解码信号，当接收到解码结束信号后，将解码得到的相应残差系数写入反扫描 RAM；

CABAC 残差解码模块：以‘4X4’块或者‘8X8’块为单位，根据软件模块解

码得到的相应参数，对亮度和色度的残差系数进行解码，解码完成后向熵解码控制模块发出结束信号，解码出来的残差系数存入反扫描 RAM；

反扫描 RAM：存储经 CAVLC 残差解码模快或者 CABAC 残差解码模块解码得到的残差系数。

本发明的显著有益效果在于：在 CAVLC 和 CABAC 的解码过程中，尤其是在对 CABAC 码流的解码过程中，CABAC 解码本身就是一个复杂的过程，而且在对残差以外的其他语法元素进行解码时，CABAC 的编码方式决定了解码的复杂性，伴随而来的是计算的复杂度，在 ASIC 设计中，如果完全采用纯硬件实现，这不仅将会占用大量的硬件资源，而且势必会大大增加设计的难度。

本发明提出了一种专用于 H.264 解码 MAE 中对熵编码残差数据等进行解码的方法，由于采用了软硬件协同工作，将解码过程中占用大量硬件资源的部分改用软件方式实现，使得熵解码器在保证获得可靠的解码效率和解码质量的前提下，大大降低了结构的复杂度和计算的复杂度，从而有效的解决了速度和效率之间的矛盾，顺利的实现了对 CAVLC 和 CABAC 熵编码码流的解码。而且，由于采用本发明所述的解决方案，使得熵解码器可以应用于实际的电路设计中，由于支持 CABAC 的解码，可以加快 H. 264 在多媒体视频方面的应用，可以用于标清的数字存储和其他的实时应用，如视频存储、VOIP、可视电话等。

附图说明

图 1 为本发明熵解码器的硬件结构实现原理图；

图 2 为熵解码模式选择的原理图；

图 3 为对一个宏块的数据进行 CAVLC 解码的方法流程图；

图 4 为对一个 4x4 子块数据块进行 CAVLC 解码的方法流程图；

图 5 为对一个宏块的数据进行 CABAC 解码的方法流程图；

图 6 为对一个 4x4 子块或者 8x8 子块数据进行 CABAC 解码的方法流程图；

图 7 为对 4x4 子块中的参数的举例说明附表；

具体实施方式

下面参考说明书附图，具体说明本发明的实现过程。如附图1所示：本发明所述的基于H.264的熵解码器装置，该装置包括如下模块：

BSI 取数模块 1：负责从外部取得解码所需的码流数据，当接收到其他模块的取数请求时，返回相应长度的码流数据；

ue/se/te 解码单元 2：接收软件模块发送来的 ue, se 或者 te 解码请求，并返回解码后的参数值；

CAVLC 残差解码模快 3：以‘4X4 块’为单位，根据软件模块解码得到的相应参数，对亮度和色度的残差系数进行解码，解码完成后向熵解码控制模块发出结束信号，解码出来的残差系数存入反扫描 RAM；

软件模块 4：对残差解码需要的一些全部和局部参数和变量进行解码，通过 ue/se/te 解码单元求解与 CAVLC 相关的参数，通过二进制算术解码单元求解与 CABAC 相关的参数，最终得到的参数再提供给 CAVCL 或 CABAC 残差解码模块使用；

二进制算术解码单元 5： 除了软件模块在解码与残差相关的参数时需要借助该模块以外，在进行 CABAC 残差解码的时候，CABAC 残差解码模快向该模块发送解码请求，经解码，返回得到二进制结果；

熵解码控制模块 6：控制整个解码流程，先控制软件模块求解相关的参数，负责向 CAVLC 残差解码模块或者 CABAC 残差解码模块发出开始解码信号，当接收到解码结束信号后，将解码得到的相应残差系数写入反扫描 RAM；

CABAC 残差解码模块 7：以‘4X4’块或者‘8X8’块为单位，根据软件模块

解码得到的相应参数，对亮度和色度的残差系数进行解码，解码完成后向熵解码控制模块发出结束信号，解码出来的残差系数存入反扫描 RAM.

反扫描 RAM8：根据 CAVLC 残差解码模快或者 CABAC 残差解码模块发来的信号，存储经 CAVLC 残差解码模快或者 CABAC 残差解码模块解码得到的残差系数。

熵解码首先要确定编码码流的编码方式，编码方式的确定是依据编码模式标志得到，如附图 2 所示，当编码模式标志 entropy_mode_flag=0 时表示编码方式为 CAVLC 编码；当编码模式标志 entropy_mode_flag=1 时表示编码方式为 CABAC 编码，根据判断出的编码方式选择相应的解码方法。

下面结合附图 1 说明熵解码器的解码过程：

步骤 1：确定了码流的编码方式之后，熵解码控制模块 6 根据编码方式决定解码的类型：即是对 CAVLC 编码码流解码还是对 CABAC 编码码流解码；如果是解 CAVLC 编码码流，则进入步骤 2 执行，如果是解 CABAC 编码码流，则进入步骤 3 执行；

步骤 2：进行 CAVLC 解码；

步骤 3：进行 CABAC 解码；

步骤 4：步骤（2）或步骤（3）完成后，返回到步骤（1），继续解下一个片（slice）的数据。

如附图 1 所示，上述步骤（2）所述的进行 CAVLC 解码的具体步骤为：

步骤 2.1：熵解码控制模块 6 向软件模块 4 发出开始信号，求解与解码有关的非残差语法元素，如 mb_type,sub_mb_type，运动向量 mvd_l0,mvd_l1 等；

步骤 2.2：软件模块 4 收到来自熵解码控制模块 6 的开始信号后，经由 ue/se/te 模块 2 并最终通过 BSI 取数模块 1 得到所需的编码数据流，解码得到所需的除宏块参数以及残差系数以外的其他语法元素；

步骤 2.3：熵解码控制模块 6 收到软件模块 4 发出的结束信号后，向 CAVLC 残差解码模块 3 发出解残差开始信号；

步骤 2.4：CAVLC 残差解码模块 3 收到熵解码控制模块 6 发来的开始信号后开始残差解码，并通过 BSI 取数据模块 1 获得解码所需的编码数据，最终解码得到亮度和色度块的残差系数值；

步骤 2.5：如果解码的不是片（slice）的最后一个宏块，则返回到步骤 2.2，继续解下一个宏块，否则结束。

如附图 1 所示，步骤（3）所述的对 CABAC 解码的具体步骤为：

步骤 3.1：对 CABAC 解码所需的上下文模型进行初始化，并初始化概率区间；

步骤 3.2：熵解码解码控制模块 6 向软件模块 4 发开始信号，开始对残差系数以外其他语法元素如 mb_type,sub_mb_type,运动向量 mvd_l0,mvd_l1 等解码，并求解残差解码中要用到的宏块参数和其他参数；

步骤 3.3：软件模块 4 收到来自熵解码解码控制模块 6 的开始信号后，经由二进制算术解码单元 5 并最终通过 BSI 取数模块 1 得到所需的编码数据流，解码得到所需的宏块参数以及除残差系数以外的其他语法元素；

步骤 3.4：熵解码控制模块 6 向 CABAC 残差解码模块 7 发出解残差开始信号；

步骤 3.5：CABAC 残差解码模块 7 收到熵解码控制模块 6 发来的开始信号后开始残差解码，同样经由二进制算术解码单元 5 并最终通过 BSI 取数模块 1 得到所需的编码数据流，最终解码得到亮度和色度块的残差系数值；

步骤 3.6：如果解码的不是片（slice）的最后一个宏块，回到步骤 3.2，否则结束。

下面参考图 3 说明对一个片数据的宏块进行 CAVLC 解码的流程：

步骤 1：由软件模块完成求出与宏块解码相关的变量和参数，如 mb_type、mb_skip_flag、sub_mb_type、mvd_l0、mvd_l1、ref_idx_l0、ref_idx_l1、mb_qp_delta、intra_chroma_pred_mode 等；

步骤 2：由步骤 1 得到解码时所需的当前宏块以及邻近宏块等解码所需的参数后，由于一个宏块包含一个 16x16 的亮度块和两个 8x8 的色度块 Cb 和 Cr，所以下面按照先亮度后色度的顺序对宏块进行解码。

对亮度块进行解码：

进行亮度块解码时，由于亮度块包含 16x16 个像素点，解码时以 4x4 块为基本的解码单位，即每次对一个 4x4 块解码，共分 16 次完成一个 16x16 亮度块的解码。在每个 16x16 亮度块中，每个 4x4 块有一个直流系数，16 个 4x4 亮度块共有 16 个直流系数；解码时首先对 16 个亮度的直流(DC)系数进行解码，然后对 16 个 4x4 亮度块的交流系数(AC)进行解码，每个 4x4 交流系数块有 15 个交流系数。

对色度块进行解码：

解码完 16x16 亮度块后，接着对两个 8x8 色度块解码，同样的以 4x4 块为基本的解码单位，同样先解码直流系数。每个 8x8 色度块有 4 个直流系数，分别完成对 Cb 和 Cr 块的直流系数的解码后，再对 Cb 和 Cr 块的交流系数进行解码，对两个色度块的解码顺序都是先 Cb 块后 Cr 块。

步骤 3：完成对 16x16 亮度块和 Cb 及 Cr 色度块的解码之后，一个宏块的数据就解码完了，可以继续下一个宏块的解码。

下面参考图 4 以对一个亮度 4x4 块或色度 4x4 块残差系数进行 CAVLC 解码说明 CAVLC 残差解码的过程：

步骤 401：确定所要解码的系数是亮度系数还是色度系数，是直流系数还是交流系数；

步骤 402：如果色度直流系数为 1，参数 nC=1，是对色度直流系数进行解码，否则 nC=（nA+nB）/2，其中 nA，nB 分别表示解码残差块的左边和正上方参考块中非零系数的个数。

步骤 403：由于 CAVLC 是变长编码，采用的不是固定的码表，因此，根据步骤 2 得到的 nC 的值决定着进行解码时采用哪一个码表，以及是采用变长码表还是定长码表或者是否使用码表。

步骤 404：查表求解 numtrailingones，其中 numtrailingones 表示拖尾系数，即从最后一个高频残差系数开始往前连续遇到的 1 和-1 的个数(1 和-1 之间可以包括 0)，numtrailingones 的值不超过 3。numtrailingones 的作用在于，在进行编码的时候，对于 4x4 块中高频部分的系数多为 1，0，-1 的情况，编码的时候，可以只记录 1，-1 的个数以及符号位即可，而无需像其他的残差系数一样要对整个系数值 level 进行编码，提高了编码效率。

步骤 405：对拖尾系数 trailingones 部分的系数符号进行解码，得到 trailingones 部分的幅值 level 的值，1 或者-1。

步骤 406：解除拖尾系数之外的其他非零系数的值 level，加上步骤 5 中得到的拖尾系数部分的 level 值，得到 4x4 块中的非零系数值 level[i]，这里的 I 表示幅值 level 的序号。

步骤 407：根据相应的码表求得 TotalZeros 的值，TotalZeros 表示 4x4 中最后一个非零系数前零的个数。

步骤 408：求各个非零系数得 run 的值，run 表示各非零系数前零的个数，分别用 run[i]表示，其中 i 是非零系数 level 的序号。

步骤 409：结束：根据前面得到的 level[i]，run[i]的值从低频部分第一个系数就可以恢复 16 个 4x4 块的残差系数值。

下面举例说明 numtrailingones，numcoeff，TotalZeros，level[j]，run[j]的含义：如附图 7 所示为一个 4x4 块的 16 个残差系数：这里 numtrailingones=3，numcoeff=6，TotalZeros=10，level[j]，run[j]的值已经在图中注明。其中的符号 i 表示 4x4 子块中残差系数的序号，level[j]，run[j]中的符号 j 表示非零系数 level[j]和零行程 run[j]的序号。

下面参考图 5 说明对一个片数据中的一个宏块进行 CABAC 解码的流程：

步骤 1：对上下文模型和概率区间进行初始化；CABAC 是自适应的二进制算术编码，所以无论是在解码还是编码时对它的上下文模型和概率区间都必须先进行初始化，才能保证解码时和编码时使用的是相同的上下文模型和概率区间；

步骤 2：开始对残差系数以外其他语法元素，如 mb_type,sub_mb_type,运动向量 mvd_l0,mvd_l1 等解码，并求解残差解码中要用到的宏块参数，由于这一部分需要占用较多地硬件资源，所以这里采用软件的方式完成；

步骤 3：下面对一个残差宏块进行解码；一个宏块包含一个 16x16 的亮度块和两个 8x8 的色度块 Cb 和 Cr，此处和 CAVLC 的解码顺序一样，仍然按照先亮度后色度的顺序对宏块进行解码；

步骤 4：以 4x4 块或者 8x8 块为基本单位对亮度进行解码。如果有直流系数，先对亮度直流系数进行解码，一共 16 个亮度 dc 系数；然后以 4x4 块为基本单位，对剩余的亮度交流系数进行解码，共 16 个 4x4 交流系数块；如果只有交流系数，直接以 4x4 块或者 8x8 块为基本单位对交流系数进行解码。这样就完成了亮度块的解码过程；

步骤 5：按照先 DC 系数后 AC 系数先 Cb 后 Cr 块的顺序对两个色度块 Cb 和 Cr 块进行解码，先解 Cb 和 Cr 块的 DC 系数，然后再解 Cb 和 Cr 块的 AC 系数，每个 8x8 色度块一共得到 4 个 DC 系数和 60 个 AC 系数；

步骤 6：继续解下一个宏块。

下面参考图 6 以解一个 4x4 残差数据块为例来说明 CABAC 的解码过程：

步骤 601：根据标志 coded_block_flag 的值决定是否需要对该 4x4 块进行解码。如果 coded_block_flag=0，表示该 4x4 块中系数全为零，所以不必进行解码，当 coded_block_flag=1 时进行下一步；

步骤 602：计算 4x4 块中非零系数的幅值 coeffLevel[i]，从低频到高频的顺序，以 i 表示系数的标号；

步骤 603：计算非零系数的 coeffLevel[i]的绝对值 abs（coeffLevel[i]）；

步骤 604：计算非零系数的符号 sign；

步骤 605～606：由上面的得到的非零系数的符号决定非零系数的值；

步骤 607：根据已有的残差系数的大小和位置信息，加上符号信息，对 4x4 块的 16 个残差系数进行恢复，得到 4x4 块的 16 个残差系数的值；

本发明所说的软件方式，是指利用嵌入式 cpu 代替硬件电路进行运算的方式，而不是单纯的指用计算机软件实现熵解码中的某些计算功能。

本发明所用的实例只是用来对上述解码器中所支持的解码功能的一种解释说明，并不代表本发明所要保护的范围仅限于此，凡采用本发明中所提及的方法和结构，或者采用对某些部分相同功能的替代方案，均在本发明所要求的保护范围，从事本行业的技术人员对此应予理解。

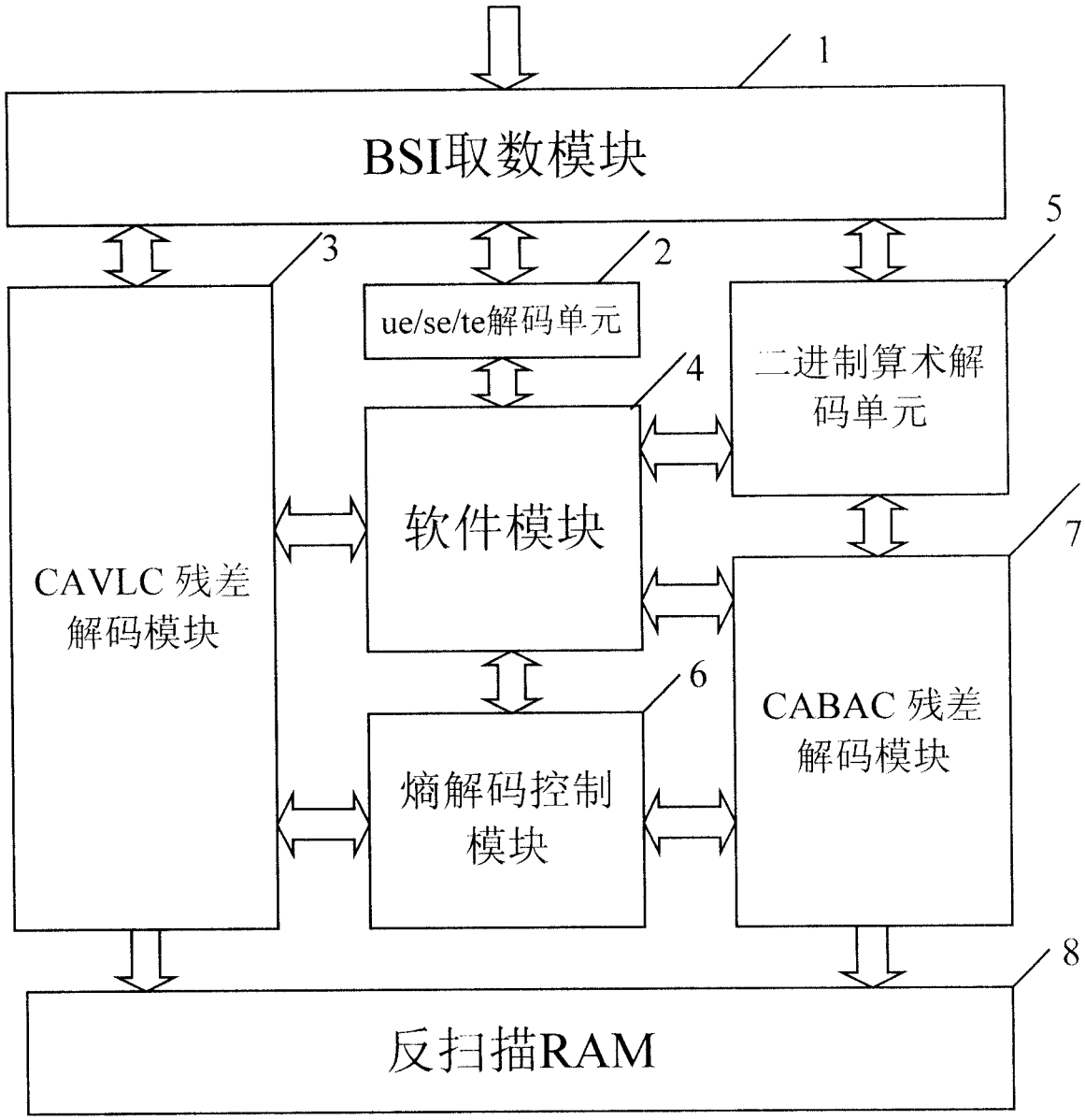

1
BSI取数模块
5
3
2
ue/se/te解码单元
二进制算术解码单元
4
软件模块
CAVLC 残差解码模块
7
6
CABAC 残差解码模块
熵解码控制模块
8
反扫描RAM

图 1

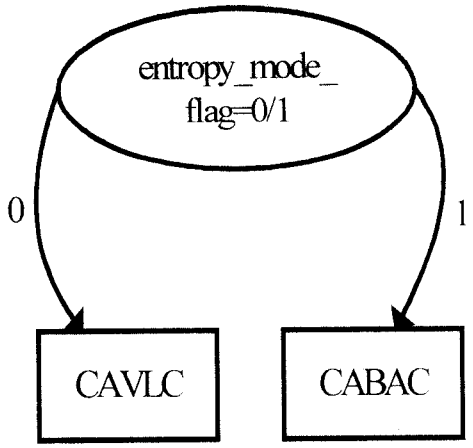

entropy_mode_
flag=0/1
0
1
CAVLC
CABAC

图 2

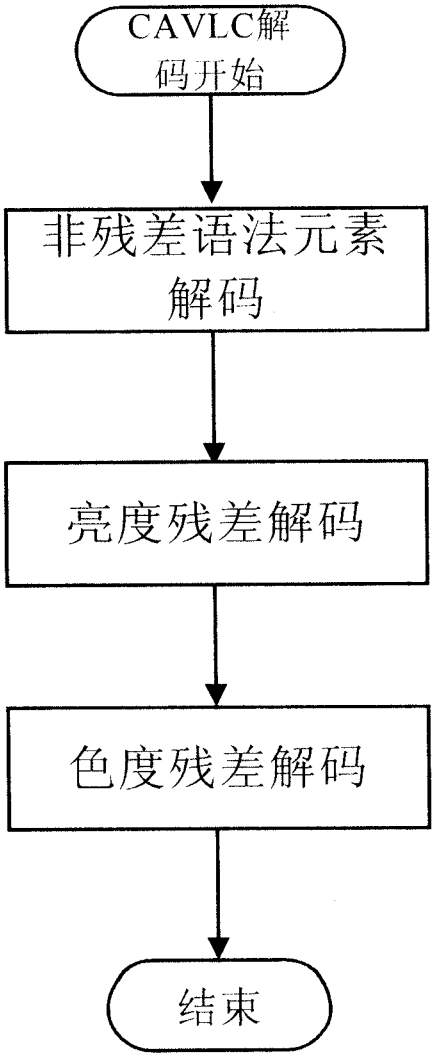

CAVLC解码开始
非残差语法元素解码
亮度残差解码
色度残差解码
结束

图 3

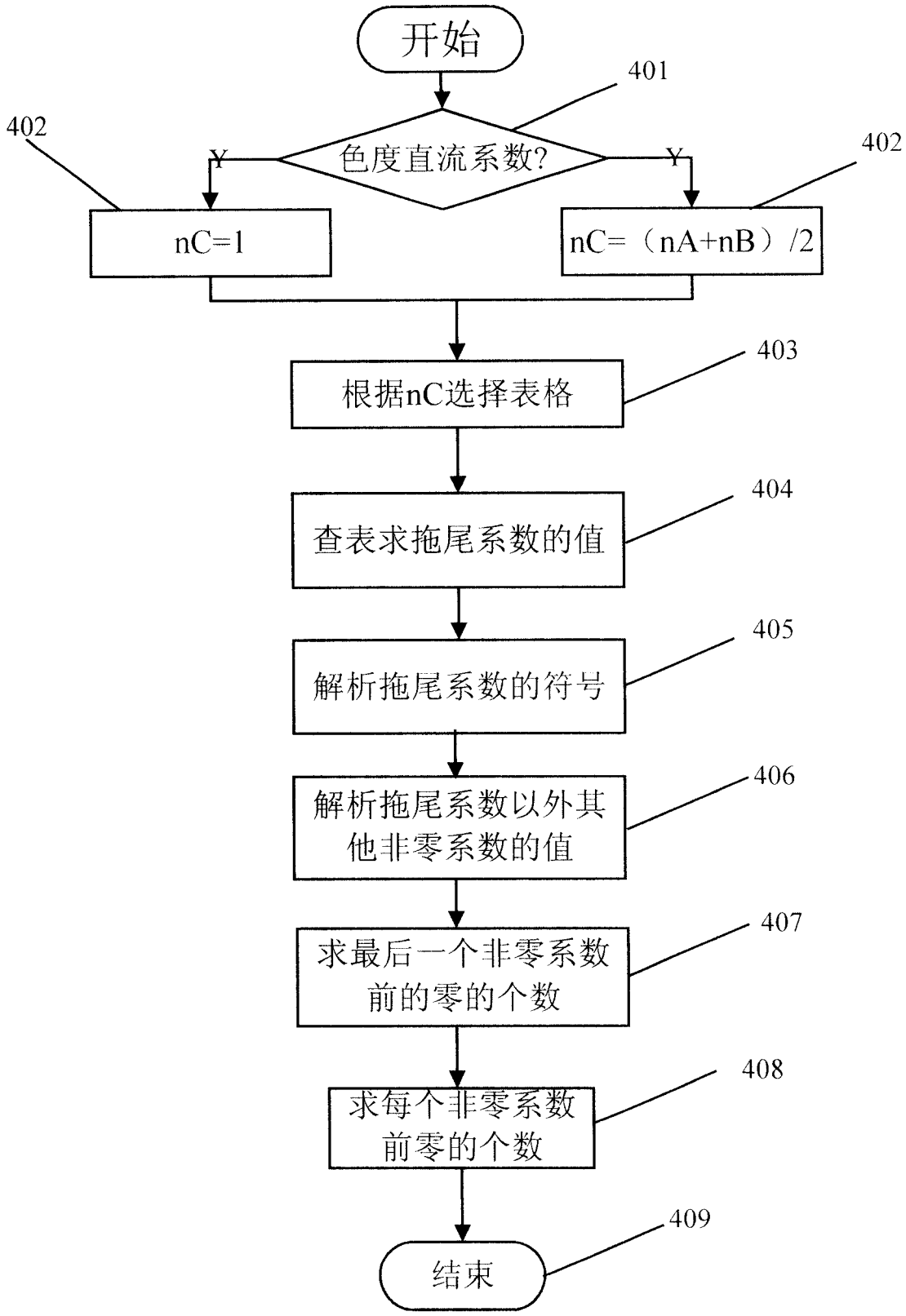

开始
401
色度直流系数?
Y
Y
402
402
nC=1
nC=（nA+nB）/2
403
根据nC选择表格
404
查表求拖尾系数的值
405
解析拖尾系数的符号
406
解析拖尾系数以外其他非零系数的值
407
求最后一个非零系数前的零的个数
408
求每个非零系数前零的个数
409
结束

图 4

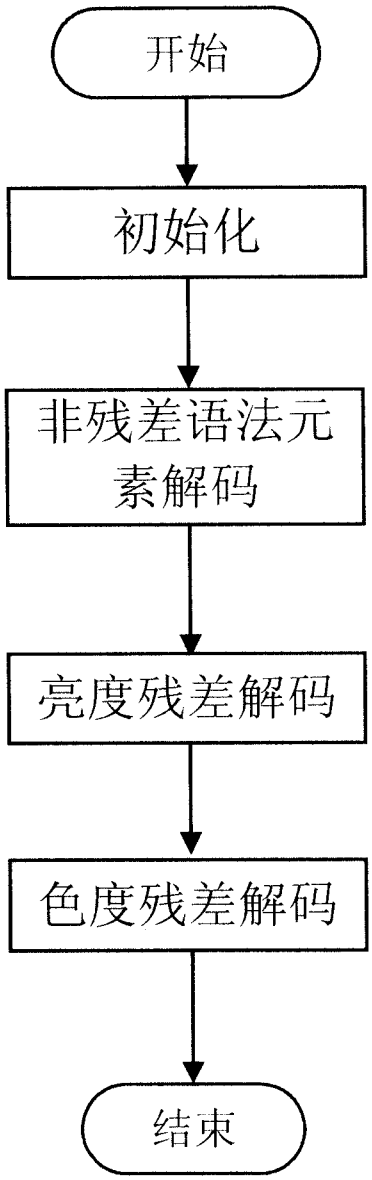


图 5

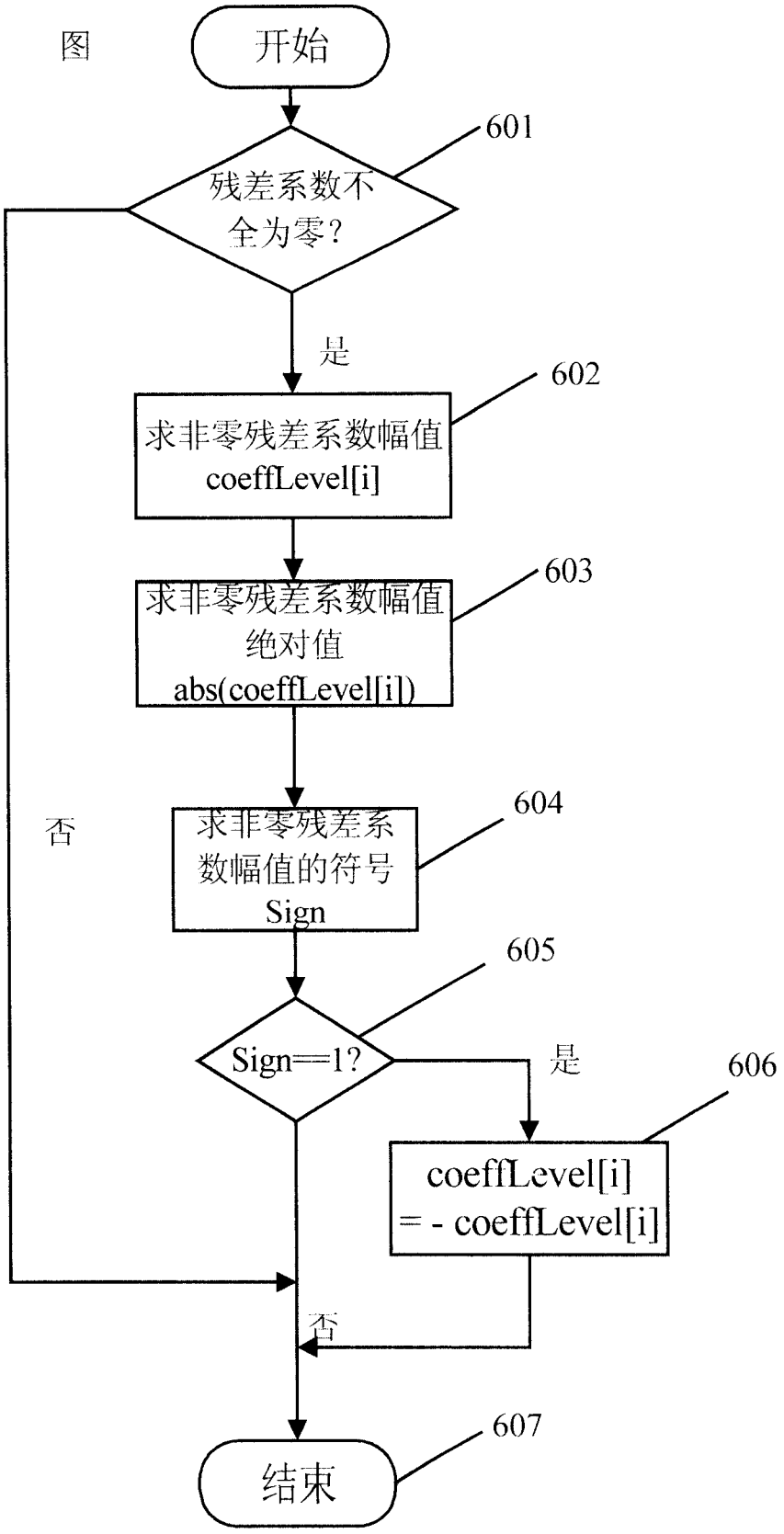


图
开始
601
残差系数不全为零?
是
602
求非零残差系数幅值 coeffLevel[i]
603
求非零残差系数幅值绝对值 abs(coeffLevel[i])
否
604
求非零残差系数幅值的符号 Sign
605
Sign==1?
是
606
coeffLevel[i] = - coeffLevel[i]
否
607
结束

图 6

序号 i	0	1	2	3	4	5	6	7
系数	0	11	0	0	-5	2	0	-1
非零系数 level[j]		level[0]			level[1]	level[2]		level[3]
零行程 run[j]		1			2	0		1
序号 i	8	9	10	11	12	13	14	15
系数	0	0	1	0	0	0	1	0
非零系数 level[j]			level[4]				level[5]	
零行程 run[j]			2				3	

图 7