[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04N 7/26 (2006.01)
H04N 7/24 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200710110192.1

[43] 公开日 2007 年 11 月 7 日

[11] 公开号 CN 101068364A

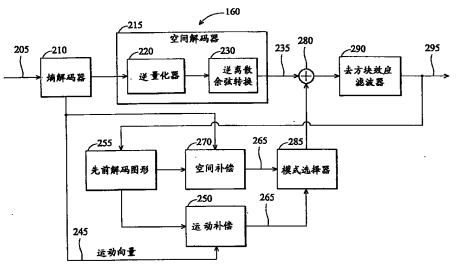
[22] 申请日 2007.6.18
[21] 申请号 200710110192.1
[30] 优先权
[32] 2006. 6.16 [33] US [31] 60/814,623
[71] 申请人 威盛电子股份有限公司
地址 中国台湾台北县
[72] 发明人 扎伊尔德·荷圣 库玛斯·萨伯丁
[74] 专利代理机构 北京市柳沈律师事务所
代理人 周少杰
权利要求书 3 页 说明书 24 页 附图 14 页

[54] 发明名称

视频编码器与图形处理单元

[57] 摘要

一示范性图形处理单元包含：一指令解码器与一视频处理单元。 该指令解码器，设置成解码多个去方块效应滤波器加速指令。 该去方块效应滤波加速器指令均与被一特定视频解码器所使用的一去方块效应滤波器相关。 该视频处理单元设置成接收由该去方块效应滤波器加速指令所编码的参数。 该视频处理单元还设置成由该接收参数判断多个第一像素数据源其中之一。 该视频处理单元还设置成由该接收参数判断多个第二像素数据源其中之一。 该视频处理单元还设置成从所判断的第一存储器源下载一第一像素数据方块。 该视频处理单元还设置成从所判断的第二存储器源下载一第二像素数据方块。



1 一种图形处理单元包含:

一解码器，设置成解码一第一与一第二去方块效应滤波器加速指令，该第一与第二去方块效应滤波加速器指令均与被一特定视频解码器所使用的一去方块效应滤波器相关；以及

一视频处理单元，设置成接收由该第一去方块效应滤波器加速指令所编码的第一参数，并判断由该接收第一参数所规范的第一存储器源，该第一存储器源位于该图形处理单元的多个存储器源其中之一，并接收由该第二去方块效应滤波器加速指令所编码的第二参数，并判断由该接收第二参数所规范的第二存储器源，该第二存储器源位于该图形处理单元的多个存储器源其中之一，

其中该视频处理单元还设置成从所判断的第一存储器源下载一第一像素数据方块，并将该去方块效应滤波器应用于该第一像素数据方块，并从所判断的第二存储器源下载一第二像素数据方块，并将该去方块效应滤波器应用于该第二像素数据方块。

2. 如权利要求 1 所述的图形处理单元，其中该多个存储器源包含在该图形处理单元内的一纹理高速缓存器与一执行单元。

3. 如权利要求 1 所述的图形处理单元，其中利用该第一存储器源与第二存储器源以实现一个接一个像素读取与写入。

4. 如权利要求 1 所述的图形处理单元，其中该第一与第二存储器源均为该图形处理单元内的一纹理高速缓存器。

5. 如权利要求 1 所述的图形处理单元，其中该第一存储器源是在该图形处理单元内的一纹理高速缓存器，而该第二存储器源是在该图形处理单元内的一执行单元。

6. 如权利要求 1 所述的图形处理单元，其中该第一与第二存储器源均为该图形处理单元内的一执行单元。

7. 如权利要求 1 所述的图形处理单元，其中该去方效应滤波器加速指令与 H.264 视频解码器所使用的滤波器相关。

8. 一种图形处理单元包含:

一视频处理单元，设置成应用于与一特定视频解码器相关的一去方块效

应滤波器;

一解码器，设置成对与该去方块效应滤波器相关的多个去方块效应滤波器加速指令解码;

一纹理滤波器单元，设置成提供像素数据至该视频处理单元，为该去方块效应滤波器所用；以及

一执行单元，设置成对像素数据执行一图形处理功能，

其中该视频处理单元还设置成接收由该各去方块效应滤波器加速指令编码的参数，并判断由该接收参数所规范的第一存储器源是对应于该纹理滤波器单元或是该执行单元，并判断由该接收参数所规范的第二存储器源是对应于该纹理滤波器单元或是该执行单元;

该视频处理单元还设置成从该第一存储器源下载一第一像素方块，与从该第二存储器源下载一第二像素方块，并应用该去方块效应滤波器至该第一像素数据方块与应用该去方块效应滤波器至该第二像素数据方块，依照所接收的参数。

9. 如权利要求 8 所述的图形处理单元，其中该视频处理单元还设置成应用于该去方块效应滤波器，依照至少一滤波器参数，而该执行单元还设置成计算该至少一滤波器参数，根据该第一像素数据方块。

10. 如权利要求 8 所述的图形处理单元，其中该去方效应滤波器加速指令与 H.264 视频解码器所使用的滤波器相关。

11. 如权利要求 8 所述的图形处理单元，其中由该接收参数所定义的第一存储器源对应该纹理滤波单元，而由该接收参数所定义的第二存储器源对应该执行单元，以实现一个接一个像素读取与写入。

12. 一种视频编码器包含:

多个执行单元指令，设置成计算与一像素数据方块及一特定视频编码规范相关的至少一去方块效应滤波器配置参数，且还设置成执行在一图形处理单元内的一着色执行单元;

多个去方块效应指令，设置成应用于与所计算的滤波器配置参数相符的一去方块效应滤波器，且还设置成执行在该图形处理单元内的一视频处理单元。

13. 如权利要求 12 所述的视频编码器，其中该去方效应滤波器加速指令与 H.264 视频解码器所使用的滤波器相关。

14. 如权利要求 12 所述的视频编码器，其中该至少一滤波器参数是边界强度、阿法(alpha)或贝塔(beta)。

15.如权利要求 12 所述的视频编码器，其中该多个去方块效应滤波器指令其中之一为该像素数据方块规范一第一存储器源，而其中该多个去方块效应滤波器指令其中的另一个为另一像素数据方块规范一第二存储器源，该第一和第二存储器源皆在图形处理单元内。

16. 如权利要求 15 所述的视频编码器，其中该第一与第二存储器源均为该图形处理单元内的一纹理高速缓存器。

17. 如权利要求 15 所述的视频编码器，其中该第一存储器源是在该图形处理单元内的一纹理高速缓存器，而该第二存储器源是在该图形处理单元内的一执行单元。

18. 如权利要求 15 所述的视频编码器，其中该第一与第二存储器源均为该图形处理单元内的一执行单元。

19. 如权利要求 12 所述的视频编码器，其中该去方效应滤波器加速指令与 H.264 视频解码器所使用的滤波器相关。

视频编码器与图形处理单元

技术领域

本发明关于影像压缩与解压缩，且尤其是关于具有影像压缩与解压缩特征的图形处理单元。

背景技术

个人计算机与消费性电子产品用于各种娱乐用品。这些娱乐用品可以大致区分为 2 类：使用计算机制图(computer-generated graphics)的那些，例如计算机游戏；与使用压缩视频数据流(compressed video stream)的那些，例如预录节目到数字式影音光盘(DVD)上，或由有线电视或卫星业者提供数字节目(digital programming)至一机顶盒(set-top box)。第 2 种亦包含编码模拟视频数据流，例如由一数字录影机(DVR，digital video recorder)所执行。

计算机制图通常由一图形处理单元(GPU，graphic processing unit)产生。图形处理单元是一种建立在计算机游戏平台(computer game consoles)与一些个人计算机上的一种特别微处理器。图形处理单元被最佳化为快速执行描绘三维空间基本物体(three-dimensional primitive objects)，例如三角形、四边形等。这些基本物体以多个顶点描述，其中每个顶点具有属性(例如颜色)，且可施加纹理(texture)至该基本物体上。描绘的结果是一二维空间像素阵列(two-dimensional array of pixels)，显示在一计算机的显示器或监视器上。

视频数据流的编码与解码牵涉到不同种类的运算，例如，离散余弦变换(discrete cosine transform)、运动估计(motion estimation)、运动补偿(motion compensation)、去方块效应滤波器(deblocking filter)。这些计算通常由通用中央处理器(CPU)结合特别的硬件逻辑电路，例如特殊应用集成电路(ASICs，application specific integrated circuits)，来处理。消费者因而需要多个运算平台以满足他们的娱乐需求。因而需要可以处理计算机制图与视频编码/解码的单一计算平台。

发明内容

在此公开的实施例提供一种用于视频压缩去方块效应的系统与方法。一示范性图形处理单元(GPU)包含：一指令解码器与一视频处理单元。该指令解码器，设置成解码多个去方块效应滤波器加速指令。该去方块效应滤波加速器指令均与被一特定视频解码器所使用的一去方块效应滤波器相关。该视频处理单元设置成接收由该去方块效应滤波器加速指令所编码的参数。该视频处理单元还设置成由该接收参数判断多个第一像素数据源其中之一。该视频处理单元还设置成由该接收参数判断多个第二像素数据源其中之一。该视频处理单元还设置成从所判断的第一存储器源下载一第一像素数据方块。该视频处理单元还设置成从所判断的第二存储器源下载一第二像素数据方块。

附图说明

图 1 是用于图形与视频编码和/或解码的一示范性运算平台的方块图。

图 2 是图 1 中该视频解码器 160 的方块图。

图 3 说明一 VC-1 滤波器的子方块像素设置。

图 4 是图 1 VC-1 回路内去方块效应滤波器硬件加速逻辑电路 400 的硬件描述虚拟码的列表。

图 5 是图 4 行加速逻辑电路 500 的硬件描述语言程序代码的列表。

图 6A-D 形成图 4、5 的行加速逻辑电路的一方块图。

图 7 H.264 回路内去方块效应滤波器硬件加速单元 700 的硬件描述虚拟码。

图 8A 与 8B 显示用于行加速逻辑电路 800 的硬件描述虚拟码。

图 9 是图 1 的图形处理单元 120 的数据流程图。

图 10 是 H.264 所用的 16x16 宏块的方块图。

【主要元件符号说明】

100～系统、110～通用 CPU、120～图形处理器(GPU)、130～存储器、140～总线、150～视频加速单元(VPU)、160～软件解码器、170～视频加速驱动器。

205～输入的比特流、210～熵解码器、215～空间解码器、220～逆量化器、230～逆离散余弦转换、235～图形、245～运动向量、250～运动补偿、255～先前解码图形、265～预测图形、270～空间补偿、280～加法器、290～去方块效应滤波器、295～解码图形。

310-320～两个邻近 4x4 子方块、330～垂直边界。

400～回路内去方块效应滤波器硬件加速逻辑电路、410～模块定义区段、420～迭代循环区段、430～测试垂直参数区段、440～比较循环参数与 3 区段、450～示例区段。

500～行加速逻辑电路、510～模块定义区段、520～像素值运算区段、530～比较循环参数与 3 区段、540～测试 DO_FILTER 区段、550～更新状态区段。

605-610-615-620～多工器、625-630-679～减法器、635-640-655-680～逻辑电路方块、645-650～加法器、660-665-670～寄存器、671～P4 寄存器输出、673～P5 寄存器输出。681～减法器、685～加法器。687-689-691-693～多工器、697～OR 门。

700～H.264 回路内去方块效应滤波器硬件加速单元、710～模块定义区段、720～迭代循环区段、730～测试垂直参数区段、740～撷取参数区段、750～示例区段。

800～行加速逻辑电路、810～模块定义区段、820～对应参数区段、830～像素计算区段。

910～指令流处理器、920～指令、930～指令数据、940～执行单元池、950～纹理滤波单元、960～纹理高速缓存器、970～后包装器。

具体实施方式

用于视频编码/解码的运算平台

图 1 是用于图形与视频编码和/或解码的一示范性运算平台的方块图。系统 100 包含一通用 CPU 110 (此后称为主处理器)、一图形处理器(GPU) 120、存储器 130 与总线 140。图形处理单元 120 包含一视频加速单元(VPU)150，其可加速视频编码和/或解码，将于后叙述。图形处理单元 120 的视频加速功能系可在图形处理单元 120 上执行的指令。

软件解码器 160 与视频加速驱动器 170 位于存储器 130 中，而至少一部分的解码器 160 与视频加速驱动器 170 在主处理器 110 上执行。通过一个由视频加速驱动器 170 提供的一主处理器接口 180，解码器 160 还发出给图形处理单元 120 的视频加速指令。如此一来，系统 100 通过发出视频加速指令给图形处理单元 120 的主处理器软件(host processor software)执行视频编码

和/或解码，图形处理单元 120 通过加速解码器 160 之一部分回应这些指令。

在一些实施例中，仅有一小部分的解码器 160 在主处理器 110 上执行，而大部分的解码器 160 由图形处理单元 120 执行，在驱动器极少超载之下。依此法，经常被执行的密集运算方块(computationally intensive blocks)被卸至图形处理单元 120，而更复杂的运算由主处理器 110 所执行。

在一些实施例中，由图形处理单元 120 内的 VPU 150 所实现的一个密集运算功能包含回路内去方块效应滤波器硬件加速逻辑电路(IDF，inloop deblocking filter hardware acceleration logic)400，亦称为回路内方块效应滤波器 400 或去方块效应滤波器 400。VPU 150 的一些实施例包含多个回路内去方块效应滤波器硬件加速逻辑电路的范例，例如，与不同编码标准如 VC-1 与 H.264 相符的滤波器。例如示于图 1 的实施例，其中 VPU 150 包含 H.264 回路内去方块效应滤波器硬件加速逻辑电路 170 与 VC- 回路内去方块效应滤波器硬件加速逻辑电路 400(稍后结合图 4 说明)。另一密集运算功能的范例系判定各滤波器的边界强度(BS，boundary strength)。

上述的结构因而使下列运作有弹性：在主处理器 110 上对解码器 160 执行一些通过对宏块(marcoblock)执行一着色程序(shader program)的特殊功能(例如去方块效应或计算边界强度)；或在图形处理单元 120 上执行大部分的解码器 160，利用流水线(pipelining)与平行化(parallelism)。在一些解码器 160 在图形处理单元 120 上执行的实施例中，该去方块效应处理系该解码器 160 各态样间同步的线程(thread)。

图 1 中省略数个对于解释图形处理单元 120 的视频加速特征并非必要且本领域技术人员熟知的已知元件。

视频解码器

图 2 是图 1 中该视频解码器 160 的方块图。在图 2 中说明的特殊实施例，解码器 160 应用 ITU H.264 视频压缩规范。然而，本领域技术人员应当了解到图 2 的解码器 160 是一视频解码器的初步表示，该视频解码器亦说明类似于 H.264 的其他类型解码器的运作，例如 SMPTE VC-1 与 MPEG-2 规范。此外，尽管示为一图形处理单元 120 的一部分，本领域技术人员还应了解到在此公开的部分解码器 160 还实现于一图形处理单元之外，例如一独立存在的逻辑电路，特殊应用集成电路(ASIC)的一部分等。

输入的比特流 205 首先由一熵解码器(entropy decoder)210 所处理。熵编

码具有统计重复型(statistic redundancy)的优点：一些图样比其他图样更常出现，所以较常出现的就用较短的码代表。熵编码包含霍夫曼编码(Huffman coding)与运行长度编码(run-length encoding)。在熵编码之后，该数据由一空间解码器(spatial decoder)215 所处理，其具有下述优点，事实上，一图形中邻近的像素通常相同或相关，所以只要对差异编码即可。在此示范性实施例中，空间解码器 215 包含一逆量化器(inverse quantizer)220，与一逆离散余弦转换(IDCT)功能 230。IDCT 功能 230 的输出可视为一图形(235)，由数像素组成。

图形 235 被处理为较小的子区块，称为宏块。H.264 视频压缩规范使用 16x16 像素的宏块尺寸，而其他压缩规范可使用其他尺寸。图形 235 内的宏块与先前解码图项的信息结合，称为帧间预测(inter prediction)处理，或与图形 235 的其他宏块的信息结合，称为帧内预测(intra prediction)处理。该输入比特流 205，被熵解码器 205 解码，而依各类型的图形应用画面间或帧内预测。

当应用帧间预测时，熵解码器 210 产生一运动向量(motion vector)245 输出。运动向量 245 被用来暂时的编码，其具有下述优点，事实上，通常在一连串的图形中许多像素会有相同的值。从一图形到另一图形的改变系编码为运动向量 245。运动补偿方块 250 将一个或多个先前解码图形 255 结合运动向量 245 以产生一预测图形(265)。当应用帧间预测时，空间补偿方块 270 将得自邻近宏块的信息与图形 235 内的宏块结合以产生一预测图形(275)。

结合器 280 将图形 235 与模式选择器(mode selector)285 的输出相加。模式选择器 285 使用熵解码比特流以判定结合器 280 使用运动补偿方块 250 产生的预测图形(265)或使用空间补偿方块 270 所产生的预测图形(275)。

编码程序引起如在沿着宏块边缘的不连续以及沿着宏块内的子方块边缘不连续的伪像(artifact)。结果是在解码图框出现了“边缘”(edge)，而原本没有。去方块效应滤波器 290 系应用于由结合器 280 输出的结合图形，以移去这些边缘产物。储存由去方块效应滤波器产生的该解码图形 295 用来解码接下来的图形。

结合图 1 的讨论，部分解码器 160 在主处理器 110 上执行，而解码器 160 亦有由图形处理单元 120 提供视频加速指令的优点。尤其是，在一些实施例中，去方块效应滤波器 290 使用由图形处理单元 120 提供的一个或多个指令

用来实现使用相对低运算成本的滤波。

去方块效应滤波器 290 是一多抽头滤波器(multi-tap filter)，其基于邻近像素值调整子方块边缘的像素值。可依照解码器 160 施行的压缩规范使用去方块效应滤波器 290 的不同实施例。各规范使用不同的滤波器参数，例如子区块的尺寸、由该滤波运作更新的像素数目、该滤波器应用的频率(例如每 N 列或每 M 行)。此外，各规范使用不同滤波器长度结构。本领域技术人员应了解多抽头滤波器，在此不讨论特定单元的结构。

VC-1 去方块效应滤波器

由 VC-1 规范规定的去方块效应滤波器实施例将结合图 4 说明。首先，VC-1 滤波器的子方块像素安排将结合图 3 说明。

图 3 显示两个邻近 4x4 子方块(310,320)，定义为列 R1-R4 与行 C1-C8。这两个子方块间的垂直边界 330 系沿着行 C4 与 C5。该 VC-1 滤波器对每个 4x4 子方块运作。对于最左边的子方块，该 VC-1 滤波器检验在一预定列(3)中的一预定群像素(P1、P2、P3)。若该预定群像素达到一特定标准，则更新相同预定列中另一像素 P4。该标准是由该预定组中像素的计算与比较的特殊集合而定。本领域技术人员应了解到这些计算与比较还是为一组滤波抽头(a set of taps)，而详细的计算与比较将稍后结合图 5 讨论。更新值亦基于对预定群组中像素所执行的运算。

该 VC-1 滤波器以模拟方式处理最右边的子方块，判定像素 P6、P7、P8 是否达到一标准，若达到该标准则更新 P5。换言之，该 VC-1 滤波器为一预定列(R3)的一群预定像素-边缘像素 P4 与 P5-根据同一列中其他群预定像素的值计算数值，P4 的值根据 P1、P2、P3，而 P5 的值根据 P6、P7、P8。

该 VC-1 有条件的更新其余列的相同群预定像素，系根据为该预定列(R3)的预定群像素(边缘像素 P4、P5)所计算的值。如此一来，R1 中的 P4 基于 R1 中的 P1、P2、P3 更新了，然而仅有在 R3 中的 P4、P5 更新之后。同样地，R1 中的 P5 基于 R1 中的 P6、P7、P8 更新了，然而仅有在 R3 中的 P4、P5 更新之后。第 2 列与第 4 列亦以类似方式处理。

从另一方面来看，在一预定第三列的像素的一些像素被滤波或更新了，当在第三列的其他像素达到一标准时。该滤波器牵涉到对这些其他像素执行比较与计算。若在第三列的其他像素未达到该标准时，在其余列相应的各像素系以一模拟方式滤波，如上所述。在此公开的去方块效应滤波器 290 的一

些实施例使用一开创性技术，先对第三列滤波，接着再对其他列滤波。这些开创性的技术将结合第 4、5、6A-6D 图，更详细的说明。

尽管图 3 说明一列列的处理垂直边缘，本领域技术人员应可了解同一图旋转 90 度后还说明一行行处理水平边缘。本领域技术人员还了解到尽管 VC-1 使用四列中的第三列作为判定有条件更新其他列的预定列，在此公开的原则还应用至使用其他预定列的实施例(例如第一列、第二列等)，还应用至形成子方块列数目不同的其他实施例。同样地，本领域技术人员还了解到尽管 VC-1 检验邻近一组像素的值以设定欲更新像素的值，在此公开的原则还应用至其他像素已被检验且其他像素已设定的实施例。就一范例而言，可检验 P2 与 P3 以判定 P4 的更新值。另一范例，P3 可根据 P2 与 P4 的值设定。

图形处理单元 120 中的视频加速单元 150 为一回路内去方块滤波器(IDF，inloop deblockging filter)，例如由 VC-1 规范的回路内去方块效应滤波器，实现硬件加速逻辑电路。一图形处理单元指令实现此硬件加速逻辑电路，将于后说明。实现 VC-1 回路内去方块效应滤波器的已知方法系平行处理各列/行，因为相同像素计算是在一子方块的各列/行执行。此已知方法每周期对两个邻近的 4x4 子方块滤波，但需要一增加逻辑门数来执行。相对的，由 VC-1 回路内去方块效应滤波器硬件加速逻辑电路 400 所使用的开创性方法系先处理第三列/行像素，而若这些像素达到该所要求的标准，接着顺序处理剩下的那三列/行。此开创性方法比已知方法使用较少的逻辑门数，其复制各列/行的机能。VC-1 IFD 加速逻辑电路 400 的循序列处理每 4 个周期对两个邻近的 4x4 子方块滤波。此较长的滤波时间与图形处理单元 120 的指令周期一致，其中该已知方法较快速的滤波，事实上比所需求的速度还快，造成逻辑门数上的浪费。

图 4 是 VC-1 回路内去方块效应滤波器硬件加速逻辑电路 400 的硬件描述虚拟码的列表。虽非使用实际硬件描述语言(HDL，hardware description language)，例如 Verilog 与 VHDL 而使用一虚拟码，本领域技术人员应对这些虚拟码相当熟悉。这些人应可了解当以实际 HDL 描述时，这些程序代码应可被编译并接着合成为构成部分视频加速单元 150 的数逻辑门配置。这些人应当可了解到这些逻辑门可以各种技术实现，例如一特定应用集成电路(ASIC)、可编程门阵列(PGA)或现场可编程门阵列(FPGA)。

此程序代码的 410 段是模块定义(module definition)。VC-1 回路内去方

块效应滤波器硬件加速逻辑电路400有许多输入参数。要进行滤波的子方块由该方块参数(Block parameter)所规范。若垂直参数(Vertical parameter)为真(True)，则该加速逻辑电路400将方块参数视为4x8方块(参见图3)，并执行垂直边缘滤波。若垂直参数为假(False)，则该加速逻辑电路400将方块参数视为8x4方块(参见图3)，并执行水平边缘滤波。

程序代码的区段420开始一迭代循环(iteration loop)，设定该循环参数变量的值。第一次通过此循环时，循环参数设为3，故先处理第3行。后续的循环迭代设定循环参数为1、2与4。利用这些参数，VC-1 回路内去方块效应滤波器硬件加速逻辑电路400重复4次，每次处理8个像素，其中一行可为一水平列或一垂直行。每一列是由行加速逻辑电路500所处理(参见图5)。在一些实施例中，此行加速逻辑电路500以一HDL次模块实现，将结合图5说明。

区段430测试垂直参数以判定执行垂直或水平边缘滤波。根据该结果，行阵列变量的8个元素系自该4x8输入方块的列或8x4输入方块的行初始化。

区段440通过将循环参数(由区段420所设定)与3做比较判定该第3行是否处理。若循环参数为3，另两个控制变量，ProcessingPixel3 与FILTER_OTHER_3则设为真。若循环参数不为3，将ProcessingPixel3设为真。

区段450举例说明另一HDL模块，VC1_IDC_Filter_Line，该滤波器应用目前的行。(结合图3所述，该行滤波器基于邻近像素值更新边缘像素值。)提供至该子模块的参数包含该控制变量ProcessingPixel3、FILTER_OTHER_3与循环参数。在一实施例中，VC-1 回路内去方块效应滤波器硬件加速逻辑电路400有一额外输入参数，一量化值，而此量化参数还提供给该子模块。

在子模块处理该列的后，VC-1 回路内去方块效应滤波器 硬件加速逻辑电路400在区段420以一循环参数更新值继续该迭代循环。依此法，对输入方块的第3行应用该滤波器，接着第1行、第2行、第4行。

图5是行加速逻辑电路500的硬件描述语言程序代码的列表，其实现了上述的子模块。程序代码的区段510由一模块定义。行加速逻辑电路500有许多输入参数。将进行滤波的行定义为行输入参数。ProcessingPixel3是一输入参数，若该行为第行或第 3 列则通过较高层逻辑电路将其设为真。参数FILTER_OTHER_3 一开始由较高层逻辑电路设为真，而根据像素值由行加

速逻辑电路 500 调整。

区段 520 执行如 VC-1 所定的各种像素值运算。(因为该计算可以参考 VC-1 的规范理解，将不对这些运算作详细说明)。区段 530 测试由较高层 VC-1 回路内去方块效应滤波器硬件加速逻辑电路 400 所提供的 ProcessingPixel3 参数。若 ProcessingPixel3 为真，则区段 530 将一控制变量 DO_FILTER 初始化为一预设值，真。在区段 520 中间的运算的各种结果用来判定是否也要处理其他 3 行。若该像素运算结果表示不处理其他 3 行，则将 DO_FILTER 设为假。

若 ProcessingPixel3 为假，区段 540 使用输入参数 FILTER_OTHER_3(由较高层 VC-1 回路内去方块效应滤波器硬件加速逻辑电路 400 所设定)以设定 DO_FILTER 的值。若 DO_FILTER 为真，区段 550 测试该 DO_FILTER 变量并更新该行变量的该边缘像素 P4、P5(参见图 3)。

区段 560 测试该 ProcessingPixel3 参数，并适当更新 FILTER_OTHER_3。该 FILTER_OTHER_3 变量系用来传达此模块中不同范例的状态信息。若 ProcessingPixel3 为真，则区段 550 以 DO_FILTER 的值更新该 FILTER_OTHER_3 参数。此技术使得用来说明此模块的较高层模块(即 VC1_InloopFilter)提供由此例的 VC_1_INLOOPFILTER_LINE 低层模块所更新的 FILTER_OTHER_3 值至另一例的 VC_1_INLOOPFILTER_LINE。

本领域技术人员应了解到图 5 的虚拟码可以各种方式合成以产生实现行加速逻辑电路 500 的逻辑门布置。其中一种布置是在图 6A-D 中说明，他们一起构成行加速逻辑电路 500 的方块图。本领域技术人员应当对 VC-1 回路内去方块效应滤波器演算法及逻辑电路结构感到熟悉。因此，图 6A-D 的元件将不详述。而将选择详述行加速逻辑电路 500 的特征。

本领域技术人员应了解到，VC-1 回路内去方块效应滤波器所牵涉到的运算包含下列，其中 P1-P8 系指像素在被处理的列/行中的位置。

A0 = (2*(P3 - P6) - 5*(P4 – P5) + 4) >> 3

A1 = (2*(P1 - P4) - 5*(P2 - P3) + 4) >> 3

A2 = (2*(P5 - P8) - 5*(P6 - P7) + 4) >> 3

clip = (P4 – P5)/2

前 3 个运算中的每一个牵涉到 3 个减法、2 个乘法、1 个加法与 1 个右移。图 6A 中的行加速逻辑电路 500 的一部分使用共用逻辑电路循序计算 A0、

A1、A2，而非为了 A0、A1、A2 使用特定独立逻辑电路方块。藉由避免逻辑电路方块重复，利用多工器循序处理各输入，减少了逻辑门和/或功率消耗。

多工器 605、610、615 与 620 用来从像素寄存器 P1-P8 在不同时序周期选择不同的输入，而这些输入被提供给各共用逻辑电路方块。逻辑电路方块 625 与 630 各执行一减法。逻辑电路方块 635 通过执行左移 1 位实现乘以 2。乘以 5 由左移 1 位(640)所实行，后面接一加法器 645。加法器 650 将左移器 635 的输出、一常数 4 与 645 输出的负数加在一起。最后，逻辑电路方块 655 执行右移 3 位。

在第 1 时序周期，一输入 T = 1 被提供至各多工器 605、610 与 615，而计算 A1 的值并存在寄存器 660。在第 2 时序周期，一输入 T = 2 被提供至各多工器 605、610 与 615，而计算 A2 的值并存在寄存器 665。在第 3 时序周期，一输入 T = 3 被提供至各多工器 605、610 与 615，而计算 A0 的值并存在寄存器 670。存在寄存器 660、665、670 的值 A1、A2、A3 将被图 6B 的部分行加速逻辑电路 500 所使用，将于后说明。P4 寄存器(671)的输出与 P5 寄存器(673)的输出将被第 6C 图的部分行加速逻辑电路 500 所使用，将于后说明。

本领域技术人员还应了解在 VC-1 回路内去方块效应滤波器所牵涉到后叙的额外运算:

```
D = 5*((sign(A0) * A3) - A0)/8
if (CLIP > 0)
{
   if (D < 0)
      D = 0
   if (D > CLIP)
      D = CLIP
}
else
{
   if (D > 0)
      D = 0
   if (D < CLIP)
```

D = CLIP

}

图 6B 的部分行加速逻辑电路 500 从图 6A 的部分行加速逻辑电路 500 接收输入，并计算 D(675)。再次参照图 6A，CLIP(677)被如下产生：像素 P4 与 P5 由逻辑电路方块 679 相减，该结果由逻辑电路方块 680 右移(整数除以 2)以产生 CLIP 677。回到图 6B， A1 可在第一周期自寄存器 660 取得， A2 可在第二周期自寄存器 665 取得，A0 可在第三周期自寄存器 670 取得。因而，在第四周期，第 6 图的部分行加速逻辑电路 500 根据上述的方程序计算 D(675)。

行加速逻辑电路 500 利用(675)以更新 P4、P5 的像素位置。尤其是，P4=P4-D 而 P5=P5+D。尽管图 6A、6B 先前结合单一列/行(例如单一组像素位置 P0-P8)说明，一子区块第 3 列/行的运算会影响该子区块其他 3 列/行的行为。行加速逻辑电路 500 利用一开创性方法实现此行为。当独立滤波运算从最前面开始-平行地-完成，结合图 6A、6B 的说明，示于图 6C、6D 的部分行加速逻辑电路 500 有条件的选择要更新的位置。换言之，VC-1 回路内去方块效应滤波器硬件加速逻辑电路 400 判定是原本的值被写回或新的值被写回。相对地，一已知方法，一 VC-1 回路内去方块效应滤波器使用循环，所以独立滤波运算有条件地执行。

如先前说明的，图 4 解释行加速逻辑电路 500 的虚拟码在一循环内如此运作：在一重复区段 420 中出现了实例区段(instantiation section)450。此外行加速逻辑电路 500 的示例使用 2 个参数，ProcessingPixel3 与 FILTER_OTHER_3。用行加速逻辑电路 500 的这些参数如下执行像素 P4、P5 有条件的更新。参见第 6C 图，寄存器 P4 写入减法器 681 的结果，其中减法器 681 有一输入为 P4(671)，为 0 或 D(675)，依 DO_FILTER(683)的值而定。同样地，寄存器 P5 写入加法器 685 的结果，其中加法器 685 有一输入为 P5(673)，为 0 或 D(675)，依 DO_FILTER(683)的值而定。因而，P4 的更新值为原本的 P4 值(若 DO_FILTER 为假)，或 P4-D。同样地，P5 的更新值为原本的 P5 值(若 DO_FILTER 为假)，或 P5+D。

本领域技术人员应当了解到，当处理一子方块第 3 列时，以 P4-D 更新 P4 的标准为：

((ABS(A0) < PQUANT) OR (A3 < ABS(A0)) OR (CLIP != 0)

DO_FILTER 683 由第 6D 图中检验这些条件的部分行加速逻辑电路 500 所计算。多工器 687 提供一输入至 OR 门 697，若 ABS(A0) < PQUANT 则选择一真输出，其他则为假。多工器 689 提供另一输入至 OR 门 697，若 A3<ABS(A0)则选择一真输出，其他则为假。多工器 691 提供另一输入至 OR 门 697，若 CLIP! =0 则选择一真输出，其他则为假。

DO_FILTER 683 是由多工器 693 所提供，其利用控制输入 Processing_Pixel_3(695)以选择输出 OR 门 697 的输出或输入信号 FILTER_OTHER_3(699)。输入 Processing_Pixel_3(695)与 FILTER_OTHER_3(699)先前结合图 4 与举例说明行加速逻辑电路 500 的较高层 VC-1 回路内去方块效应滤波器硬件加速逻辑电路 400 的虚拟码已说明过了。回到图 4，当处理第 3 行/列时(第 1 圈)，Processing_Pixel_3(695)设为真，其他则为假。基于关于 PQUANT、ABS(A0)、CLIP 的条件，记录一中间变量 DO_FILTER，不论 P4/P5 是否更新。最后 FILTER_OTHER_3(699)的值系设自该中间变量 DO_FILTER。图 6C、6 的逻辑电路部分的行加速逻辑电路 500 的结果是为，每 4 个周期，在 4 邻近列/行的 P4、P5 的像素位置设为滤波后的值(根据 A0-A3、PQUANT、CLIP 等变量)或再次写入其原来的值。

该 VC-1 去方块效应加速单元 400 开创性地采用平行与循序的结合，如前所述。平行处理提供较快速的执行并减少延迟。尽管平行化增加了逻辑门数，但增加量被前述的循序处理所抵消。没有使用前述循序处理的已知方法徒增逻辑门数。

H.264 去方块效应滤波器

由 VC-1 所规范的去方块效应滤波器(IDF)的一实施例已如上所述。图形处理单元 120 的一些实施例包含一用于 H.264 去方块效应的硬件加速单元。本领域技术人员应当对 H.264 回路内去方块效应滤波器相当熟悉，故仅对该滤波操作简单概述。H.264 回路内去方块效应滤波器是一条件滤波器(conditional filter)，应用于所有图形的 4X4 方块边缘，除非 Disable_Deblocking_Filter_IDC 被定义为该边缘。该滤波器系循序的应用于所有的宏块以增加宏块位址。对每个宏块，垂直边缘先从左滤到右，接着水平从上到下的滤波(VC-1 应用相反的顺序)。从而使用来自目前宏块上面与左边的宏块及先前滤过的宏块取样值，且可能再次滤波。

H.264 回路内去方块效应滤波器硬件加速单元 700 的一些先进特色将结

合图 7 的硬件描述虚拟码说明。虽非使用实际硬件描述语言(HDL，hardware description language)，例如 Verilog 与 VHDL 而使用一虚拟码，本领域技术人员应对这些虚拟码相当熟悉。这些人应可了解当以实际 HDL 描述时，这些程序代码应可被编译并接着合成为构成部分视频加速单元 150 的数逻辑门配置。这些人应当可了解到这些逻辑门可以各种技术实现，例如一特定应用集成电路(ASIC)、可编程门阵列(PGA)或现场可编程门阵列(FPGA)。

此程序代码的 710 段是模块定义(module definition)。H.264 回路内去方块效应滤波器硬件加速逻辑电路 700 有许多输入参数。要进行滤波的子方块是由该方块参数(Block parameter)所规范。若垂直参数(Vertical parameter)为真(True)，则该加速逻辑电路 700 将方块参数视为 4x8 方块，并执行垂直边缘滤波。若垂直参数为假(False)，则该加速逻辑电路 700 将方块参数视为 8x4 方块(参见图 3)，并执行水平边缘滤波。

程序代码的区段 720 开始一迭代循环(iteration loop)，设定该循环参数变量的值。利用这些参数，H.264 回路内去方块效应滤波器硬件加速逻辑电路 700 重复 4 次，每次处理 8 个像素，其中一行可为一水平列或一垂直行，依垂直参数而定。将于下详述，各行是由行加速逻辑电路 800 执行 2 次(参见第 8 图)。

区段 730 测试垂直参数以判定执行垂直或水平边缘滤波。根据该结果，行阵列变量的 8 个元素被自该 4x8 输入方块的列或 8x4 输入方块的行初始化。当示例时(when instantiated)，区段 730 的这些码结合区段 720 的迭代码变成多工与比特放置(bit-positioning)逻辑电路(有时也称为重组逻辑电路，swizzling logic)，其依程序代码所描述的，从存储器的输入方块中移动比特至 P 寄存器中适当的比特位置。应注意到区段 720、730 中的程序代码与图 4 中用于 VC-1 去方块效应滤波器 400 的模拟码(analogous)相同。该选择的结果是，单个多工/重组逻辑电路方块被产生且用于 H.264 回路内去方块效应滤波器逻辑电路 700 与 VC-1 回路内去方块效应滤波器逻辑电路 400。

区段 750 从由图形处理单元 120 所提供的 H.264 指令内所含的信息撷取使用于实际滤波器的参数。bS(边界强度)与 chromaEdgeFlag 参数被使用于 H264 回路内去方块效应滤波器且为本领域技术人员所熟悉。参数 indexA 与 indexB 对应 H.264 回路内去方块效应滤波器所使用的阿法(alpha)与贝塔(beta)参数，其也为本领域技术人员所熟悉。

图形处理单元 120 的一开创性特征为 indexA、indexB、bS 参数不由 H.264 回路内去方块效应滤波器硬件加速逻辑电路 700 所计算，而由图形处理单元 120 内的一执行单元 940 所计算(稍后结合图 9 说明)。通过使用 EU 指令以实现 bS、indexA、indexB 的计算，可利用图形处理单元执行单元 940 的运算功率与通用，增强回路内去方块效应滤波器硬件加速逻辑电路 700。这样的选择避免了回路内去方块效应滤波器硬件加速逻辑电路 700 中额外的、可能复杂的逻辑电路。在另一实施例中，参数 bS、indexA、indexB 是由主处理器 110(参见图 1)上所执行的程序代码所运算。

区段 750 举例说明另一 HDL 模块，H264_Deblock_Filter_Line，该滤波器应用目前的行。提供至该子模块的参数包含撷取自上述 EU 指令的控制变量，与 LeftTop 参数。逻辑电路 700 的一开创性特征是呼叫该行滤波器两次，每次呼叫仅更新一半的像素，其中被更新的一半是由 LeftTop 参数所标示。此设计衡量节省了逻辑门数但需要更多的时脉周期。本领域技术人员应当了解到如何以不同参数值示例该滤波行模块以产生两不同的逻辑电路方块，具有如该像素方块不同两半的输入。

在子模块处理该列的后，硬件加速逻辑电路 700 在区段 720 以一循环参数更新值继续该迭代循环。依此法，对输入方块的第 1-4 行应用 H264_Deblock_Filter_Line。

图 8A、B 显示用于行加速逻辑电路 800 的硬件描述虚拟码，其实现了上述的 H264_Deblock_Filter_Line 次模块。如图 8A 所示，该行模块 800 被分成模块定义区块 810、对应参数区块 820 与计算像素区块 830。本领域技术人员应可从第 8A 的程序代码中理解模块定义区块 810，将不再解释。对应参数区块 820 呼叫其他两个子程序(结合图 8B 说明)以对应由 H.264 回路内去方块效应滤波器硬件加速逻辑电路 700 所提供的参数 IndexA、IndexB 至阿法与贝塔参数。

阿法与贝塔，以及 ChromaEdge flag，接着被区段 830 使用以通过基于阿法、贝塔、ChromaEdge 与邻近的像素值计算新的像素值确实的应用该滤波器。未显示该区段实际的虚拟码，因为本领域技术人员应知道如何实现用于单一行的去方块效应滤波器，如 H.264 规范中所述。

行加速逻辑电路 800 的开创性特征更示于逻辑区段，getAlphaBeta 850 与 getThreshold 870，示于图 8B。这些逻辑区段对应图 8A 的对应参数区段

820 所使用的子程序。如图 8B 中可看到的程序代码，只读存储器(ROM，read only memory)表系用来自 IndexA 与 IndexB 对应相对的阿法与贝塔值。同样地，ROM 表是用来计算该临界值。

在图形处理单元 120 的一些实施例中，其中上述 H.264 去方块效应功能通过图形处理单元指令实施。将结合图 10 更详细的说明图形处理单元 120，强调图形处理单元指令的特殊选择以实施 H.264 去方块效应加速。

图形处理器

多重去方块效应指令的原理

图形处理单元 120 的指令集包含在软件里执行的部分解码器 160 可用来加速一去方块效应滤波器。在此说明一开创性技术提供不只一个的多重图形处理单元指令以加速特定去方块效应滤波器。回路内去方块效应滤波器 290 原本就是循序的，因而一特定滤波器必须以一定次序对像素滤波(例如 H.264 规定从左到右接着从上到下)。因而，先前滤过的或更新过的像素在滤后面像素时被拿来作为输入。主处理器处理储存在已知存储器的像素值，这使得像素一个接一个读取、写入。然而，这循序的本质当回路内去方块效应滤波器 290 使用一图形处理单元加速部分滤波处理时无法适当配合。已知图形处理单元将像素储存在一纹理高速缓存器(texture cache)，而该图形处理单元管线设计不遵从一个接一个(back-to-back)读取、写入纹理高速缓存器。

在此公开图形处理单元 120 的一些实施例提供多重图形处理单元指令，其可一起用来加速一特定去方块效应滤波器。其中一些指令把纹理高速缓存器当像素数据源，而一些指令使用图形处理单元执行单元作为数据源。回路内去方块效应滤波器 290 适当的结合使用这些不同的图形处理单元指令以实现一个接一个读取、写入像素。接下来概要说明流经图形处理单元 120 的数据，再接着解释由图形处理单元 120 提供的去方块效应加速指令，与回路内去方块效应滤波器 290 运用这些指令。

图形处理单元流

图 9 是图形处理单元 120 数据流的图，其中指令流是由图 9 左边的箭头，而影像或图形流是由右边的箭头表示。图 9 省略了数个本领域技术人员已知的元件，这些对解释图形处理单元 120 的回路内去方块效应特征非必要。一指令流处理器 910 从一系统总线(未示)接收一指令 920，并解码该指令，产生指令数据 930，例如顶点数据。图形处理单元 120 支援一已知图形处理指

令，以及加速视频编码和/或解码的指令。

已知图形处理指令牵涉到如顶点着色(vertex shading)、几何着色(geometry shading)、像素着色(pixel shading)等难题。因此，指令数据 930 系应用于着色器执行单元(shader execution units)的池(pool)940。执行单元 940 必要使用一纹理滤波单元(TFU，texture filter unit)950 以施加一纹理至一像素。纹理数据被高速缓存自纹理高速缓存器 960，其在主存储器(未示)后面。

一些指令送给视频加速器 150，其运作将于后说明。产生的数据接着由后包装器(post-packer 970)处理，其压缩该数据。在后处理(post-processing)之后，由视频加速单元所产生的数据被提供给执行单元池(execution unit pool)940。

视频编码/解码加速指令的执行，例如前述的去方块效应滤波指令，在许多方面与前述的已知图形指令不同。首先，视频加速指令是由视频加速单元 150 执行，而非着色器执行单元。其次，视频加速指令不使用其纹理数据。

然而，视频加速指令所使用的影像数据与图形指令所使用的纹理数据均为 2 维阵列。图形处理单元 120 同样利用此优点，使用纹理滤波单元 950 下载给视频加速单元 150 的影像数据，因而使纹理高速缓存器 960 高速缓存一些由视频加速单元 150 运作的影像数据。因此，示于图 9，视频加速单元 150 是位于纹理滤波单元 950 与后包装器 970 之间。

纹理滤波单元 950 检验从指令 920 撷取的指令数据 930。指令数据 930 更提供纹理滤波单元 950 纹理高速缓存器 960 内想要的影像数据的座标。在一实施例中，这些座标标明为 U、V 对，本领域技术人员应对此熟悉。当指令 920 是一视频加速指令时，所撷取的指令数据更命令纹理滤波单元 950 略过纹理滤波单元 950 内的纹理滤波器(未示)。

依此法，纹理滤波单元 950 系受操纵为视频加速指令去下载影像数据给视频加速单元 150。视频加速单元 150 从数据路径上的纹理滤波单元 950 接收影像数据，与命令路径上的命令数据 930，并根据命令数据 930 对该影像数据执行一运作。由视频加速单元 150 所输出影像数据系回馈给执行单元池 940，在由后包装器 970 处理之后。

去方块效应指令

在此叙述的图形处理单元 120 的实施例，提供 VC-1 去方块效应滤波器与 H.264 去方块效应滤波器硬件加速。VC-1 去方块效应滤波器是由一图形

处理单元指令("IDF_VC-1")加速，而 H.264 去方块效应滤波器由三个图形处理单元指令("IDF_H264_0"、 "IDF_H264_1"、 "IDF_H264_2")加速。

如先前说明的，各图形处理单元指令系解码且分析(parsed)为指令数据930，其可视为各指令的特定参数集，示于表 1。IDF_H264_x 指令共用一些共用参数，而其他的为各指令独有的。本领域技术人员应了解到这些参数可以使用各种操作码(opcode)与指令格式编码，所以这些议题将不在此讨论。

表 1: IDF_H264 指令的参数

参数	大小	运算元	叙述
FieldFlag (Input)	1-比特		若 FieldFlag == 1 则 Field Picture, 其他 Frame Picture
TopFieldFlag (Input)	1-比特		若 TopFieldFlag == 1 则 Top-Field-Picture, 其他 Bottom-Field-Picture 若设定了 FieldFlag.
PictureWidth (Input)	16-比特		例如, 用于 HDTV 的 1920
PictureHeight (Input)	16-比特		例如, 用于 30P HDTV 的 1080
YC Flag	1-比特	Control-2	Y 平面 or 色度平面
Field Direction	1-比特	Control-1	
CBCR Flag	1-比特	Control-1	Cb 或 Cr
BaseAddress (Input)	32-比特无符号的		用于 IDF_H64_0 与 IDF_H64_0:纹理存储器中的子方块基本位址

BlockAddress (Input)	13.3 格式，省略分数部分	SRC1[0:15] = U SRC1[31:16] = V	用于 IDF_H64_0: 整个子方块(关于基本位址)的纹理座标
			For IDF_H64_1: 剩下的子方块(关于基本位址)的纹理座标
			在 IDF_H64_2 未使用
DataBlock1	4x4x8-比特		在 IDF_H64_0 未使用
		SRC2[127:0]	用于 IDF_H64_1: 子方块的上半或左半部，根据依 Control 2 参数编码的 FilterDirection
		SRC2[127:0]	用于 IDF_H64_2: 第一(偶数)寄存器对
DataBlock2	4x4x8-比特		在 IDF_H64_0 o 或 IDF_H64_1 中未使用
		SRC2[255:128]	用于 IDF_H64_2: 第二(奇数)寄存器对
Sub-block (Output)	128-比特		去方块效应的 8x4x8-bit 子方块(128-比特)

结合使用许多输入参数以判定由纹理滤波单元 950 所撷取的 4x4 方块位址。BaseAddress 参数指出在纹理高速缓存器中该纹理数据的起点。将此区域内左上方块座标给 BaseAddress 参数。PictureHeight 与 PictureWidth 输入参数系用来判断该方块的范围，即左下方座标。最后，视频图形可为逐行扫描(progessive)或隔行扫描(interlace)。若为隔行扫描，其是由两个方向组成(上方与下方)。纹理滤波单元 950 使用 FieldFlag 与 TopFieldFlag 以适当处理隔行扫描影像。

去方块效应 8x4x8 比特输出被提供给一目标寄存器，且还写回执行单元池 940。将去方块效应输出写回执行单元池 940 是一“位置修改(modify in

place)”运作，在某些解码器的实现中是必要的，例如 H.264 其中方块中的像素值，右边与下方，是依先前的结果所计算。然而 VC-1 解码器不像 H.264 有此限制关系。在 VC-1 中，对每个 8x8 边界(先垂直再水平)滤波。所有的垂直边缘可以因而实质上平行地执行，4x4 边缘稍后滤波。可以利用平行化因为仅有两个像素(一个边缘一个)被更新，而这些像素不用来计算其他边缘。既然去方块效应数据是写回执行单元池 940 而非纹理高速缓存器 960，提供了不同的 IDF_H264_x 指令，这子方块从不同位置被撷取。这可在表 1 中看到，在 BlockAddress 的叙述中，Data Block 1 与 Data Block 2 参数。IDF_H264_0 指令从纹理高速缓存器 960 撷取整个 8x4x8 比特子方块。IDF_H264_1 指令从纹理高速缓存器 960 撷取半个子方块并从执行单元池 940 撷取半个。

随解码器 160 而变的 IDF_H264_x 指令的功用将结合第 8 图详述。接下来叙述在供应像素数据给视频加速单元 150 前，纹理滤波单元 950 与执行单元池 940 转换所撷取的像素数据的处理。

影像数据的转换

上述的指令参数，提供欲从纹理高速缓存器 960 或从执行单元池 940 解取的子方块位址的座标给纹理滤波单元 950。影像数据包含亮度(Y)与色度(Cb,Cr)平面。一 YC 标志输入参数定义要处理 Y 平面或是 CbCr 平面。

当处理亮度(Y)数据时，如 YC 标志参数所标示的，纹理滤波单元 950 撷取该子方块并提供该 128 比特作为 VC-1 回路内去方块效应滤波器 硬件加速逻辑电路 400 的输入(例如图 4 的 VC-1 加速器范例的方块输入参数)。所产生的数据被写入目标寄存器作为一 4 组-寄存器(register quad，即，DST、DST+1、DST+2、DST+3)。

当处理色度数据时，如 YC 标志参数所标示的，Cb 与 Cr 方块将由 VC-1 回路内去方块效应滤波器 硬件加速逻辑电路 400 连续地处理。所产生的数据被写入纹理高速缓存器 960。在一些实施例中，此写入动作在各周期中发生，每个周期写入 256 比特。

一些视频加速单元实施例使用隔行扫描 CbCr 平面，各存为一半宽度与一半长度。在这些实施例中，纹理滤波单元 950 为视频加速单元 150 将 CbCr 子方块数据解隔行扫描至用来沟通纹理滤波单元 950 与视频加速单元 150 的一缓冲器。尤其是，纹理滤波单元 950 将 2 个 4x4 Cb 方块写入该缓冲器，接着将 2 个 4x4 Cr 方块写入该缓冲器。8x4 Cb 方块首先由 VC-1 回路内去

方块效应滤波器硬件加速逻辑电路400处理，所产生的数据写入纹理高速缓存器960。接着，8x4 Cr方块由VC-1 回路内去方块效应滤波器硬件加速逻辑电路400处理，所产生的数据写入纹理高速缓存器960。视频加速单元150使用CbCr标志参数以管理此循序处理。

软件解码器使用去方块效应指令

结合先前图1的说明，解码器160在主处理器110上执行但也利用图形处理单元 120所提供的视频加速指令。尤其是H.264回路内去方块效应滤波器290的实施例使用特定IDF_H264_x结合以处理边缘，依H.264所规定的次序，从纹理高速缓存器960撷取一些子方块并从执行单元池940撷取另一些。在适当结合之下，这些IDF_H264_x指令实现一个接一个像素读取与写入。

图10是用于H.264的16x16宏块的方块图。这宏块切割成16个4x4子方块，每个均将进行去方块效应。图10中的4个子方块可依列与行定义(例如R1，C2)。H.264定义先处理垂直边缘在处理水平边缘，如图10所示的边缘顺序(a-h)。

因此，该去方块效应滤波器是应用于一对子方块间的边缘，子方块对依此次序滤波:

edge a=[block to left of R1,C1] | [R1,C1] ; [block to left of R2,C1] | [R2,C1] ;

[block to left of R3,C1] | [R3,C1]; [block to left of R4,C1] | [R4,C1]

edge b=[R1,C1] | [R2,C2] ; [R2,C1] | [R2,C2] ;

[R3,C1] | [R3,C2] ; [R4,C1] | [R4,C2] ;

edge c=[R1,C2] | [R2,C3] ; [R2,C2] | [R2,C3] ;

[R3,C2] | [R3,C3] ; [R4,C2] | [R4,C3] ;

edge d=[R1,C3] | [R2,C4] ; [R2,C3] | [R2,C4] ;

[R3,C3] | [R3,C4] ; [R4,C3] | [R4,C4] ;

edge e=[block to top of R1,C1] | [R1,C1] ; [block to top of R1,C2] | [R1,C2] ;

[block to top of R1,C3] | [R1,C3]; [block to top of R1,C4] | [R1,C4]

edge f=[R1,C1] | [R2,C1] ; [R1,C2] | [R2,C2] ;

[R1,C3] | [R2,C3]; [R1,C4] | [R2,C4]

edge g=[R2,C1] | [R3,C1] ; [R2,C2] | [R3,C2] ;
[R2,C3] | [R3,C3]; [R2,C4] | [R3,C4]
edge h=[R3,C1] | [R4,C1] ; [R3,C2] | [R4,C2] ;
[R3,C3] | [R4,C3]; [R3,C4] | [R4,C4]

对于第 1 对子方块，均下载自纹理高速缓存器 960，因为还没有像素因应用滤波器而被改变。尽管第 1 垂直边缘(a)的滤波器可以改变(R1，C1)的像素值，第 2 列垂直边缘实际上与第 1 列垂直边缘共用所有像素。因此，第 2 对子方块(边缘 b)还下载自纹理高速缓存器 960。既然两相邻列间的垂直边缘不共用像素，第 3 对(边缘 c)与第 4 对(边缘 d)子方块也相同。

由回路内去方块效应滤波器 290 所发出的特定 IDF_H264_x 指令判定要从那个位置下载像素数据。由回路内去方块效应滤波器 290 所使用的 IDF_H264_x 指令处理第 1 垂直边缘(a)的次序为:

IDF_H264_0 SRC1=address of (R1,C1);
IDF_H264_0 SRC1=address of (R2,C1);
IDF_H264_0 SRC1=address of (R3,C1);
IDF_H264_0 SRC1=address of (R4,C1);

接下来，回路内去方块效应滤波器 290 处理第 2 垂直边缘(b)，从(R1，C2)开始。在定义为(R1，C2)8x4 子方块内最左边 4 个像素与(R1，C1)子方块最右边的像素重迭。这些由(R1，C1)的垂直边缘滤波器所处理，还能更新，之重迭像素系因而被读自执行单元池 940 而非纹理高速缓存器 960。然而，在(R1，C2)子方块最右边的 4 个像素还没被滤波，因而读自纹理高速缓存器 960。子方块(R2，C2)到(R4，C2)也相同。回路内去方块效应滤波器 290 藉由命令下面 IDF_H264_x 的顺序以处理第 2 垂直边缘(b)，以完成此结果:

IDF_H264_1 SRC1=address of (R1,C2);
IDF_H264_1 SRC1=address of (R2,C2);
IDF_H264_1 SRC1=address of (R3,C2);
IDF_H264_1 SRC1=address of (R4,C2);

当处理第 3 垂直边缘(c)时，从(R1，C3)开始。在(R1，C3)8x4 子方块内最左边 4 个像素与(R1，C2)子方块最右边的像素重迭，因而要读自执行单元池 940 而非纹理高速缓存器 960。然而，在(R1，C2)子方块最右边的 4 个像素还没被滤波，因而读自纹理高速缓存器 960。子方块(R1，C2)到(R4，C2)

也相同。最后一垂直边缘(d)会发生类似的情形。因此，回路内去方块效应滤波器 290 藉由命令下面 IDF_H264_x 的顺序以处理剩下 2 垂直边缘 c 与 d:

IDF_H264_1 SRC1=address of (R1,C3);

IDF_H264_1 SRC1=address of (R2,C3);

IDF_H264_1 SRC1=address of (R3,C3);

IDF_H264_1 SRC1=address of (R4,C3);

IDF_H264_1 SRC1=address of (R1,C4);

IDF_H264_1 SRC1=address of (R2,C4);

IDF_H264_1 SRC1=address of (R3,C4);

IDF_H264_1 SRC1=address of (R4,C4);

接着处理水平边缘(e-h)。此时，去方块效应滤波器已应用于宏块中的每个子方块，因而每个像素可能已更新。因此，送去进行水平边缘滤波的各子方块系读自执行单元池 940 而非纹理高速缓存器 960。因此，回路内去方块效应滤波器 290 藉由命令下面 IDF_H264_x 的顺序以处理水平边缘:

IDF_H264_2 SRC1=address of (R1,C1);

IDF_H264_2 SRC1=address of (R2,C1);

IDF_H264_2 SRC1=address of (R3,C1);

IDF_H264_2 SRC1=address of (R4,C1);

IDF_H264_2 SRC1=address of (R1,C2);

IDF_H264_2 SRC1=address of (R2,C2);

IDF_H264_2 SRC1=address of (R3,C2);

IDF_H264_2 SRC1=address of (R4,C2);

IDF_H264_2 SRC1=address of (R1,C3);

依此法，复杂的滤波运作系通过图形处理单元指令集所实施。整个去方块效应滤波运作通常太复杂而难以实现为单一指令滤波器。例如，H.264 滤波器太复杂了，其包含水平路径与垂直路径。此外，方块尺寸亦相当大。因此，与其建构硬件管理滤波器的控制方面，不如依序结合各单一指令(例如，巨集)，于是这些指令序列就被用来处理 4x4 方块。这使得可使用执行单元资源，其已齐备，因而将在回路内去方块效应滤波器中复杂控制结构的需求降到最低，如此一来可减低回路内去方块效应滤波器单元中硬件与存储器的需求。另一方面，在去方块效应滤波器 290 中实现这些滤波指令而不是通过

在执行单元上执行的指令实现是有好处的，因为该滤波包含一些数量运作(scalar operation，例如数据重组、查表、条件滤波)，这对以向量为基础的执行单元来说是没有效率的。

任何程序说明或流程图中的方块应被理解为表示模块、区段或部分程序代码，其包含用于实现特定逻辑电路功能或程序中的步骤的一个或多个可执行的指令。软件领域的普通技术人员应当了解到，其他的实现方法亦包含于所公开的范围内。在其他的实现方法中，各功能可不依所示或公开的顺序执行，包含实质上同步进行或逆向进行，依所涉之功能而定。

在此公开的系统与方法可以软件、硬件或其结合实现。在一些实施例中，该系统和/或方法系以存在存储器中的软件实现，且由位于一计算装置中的适当处理器所执行(包含而不限于一微处理器、微控制器、网路处理器、可重新装配处理器、可扩充处理器)。在其他实施例中，该系统和/或方法是用逻辑电路实现，包含而不限于一可编程逻辑器件(PLD，programmable logic device)、可编程门阵列(PGA，programmable gate array)、现场可编程门阵列(FPGA，field programmable gate array)或特定应用电路(ASIC)。在其他实施例中，这些逻辑叙述是在一图形处理器或图形处理单元(GPU)完成。

在此公开的系统与方法可被嵌入任何计算机可读介质而使用，或连结一指令执行系统、设备、装置。该指令执行系统包含任何以计算机为基础的系统、含有处理器的系统或其他可以从该指令执行系统撷取与执行这些指令的系统。所公开的文字“计算机可读介质(computer-readable medium)”可为任何可以容纳、储存、沟通、传递或传送该程序作为使用或与该指令执行系统连结的工具。该计算机可读介质可为，例如(非限制)为基于电子的、有磁性的、光的、电磁的、红外线的或半导体技术的一系统或传递介质。

使用电子技术的计算机可读介质的特定范例(非限制)可包含：具有一条或多条电性(电子)连接的线；一随机存取存储器(RAM，random access memory)；一只读存储器(ROM，read-only memory)；一可擦除可编程只读存储器(EPROM 或快闪存储器)。使用磁技术的计算机可读介质的特定范例(非限制)可包含：可携带计算机磁盘。使用光技术的计算机可读介质的特定范例(非限制)可包含：一光纤与一可携带只读光盘(CD-ROM)。

虽然本发明在此以一个或更多个特定的范例作为实施例阐明及描述，不过不应将本发明局限于所示的细节，然而仍可在不背离本发明的精神下且在

请求保护的范围均等的领域与范围内实现许多不同的修改与结构上的改变。因此，最好将所附上的权利要求书广泛地且以符合本发明领域的方法解释，在随后的请求保护的前提出此声明。

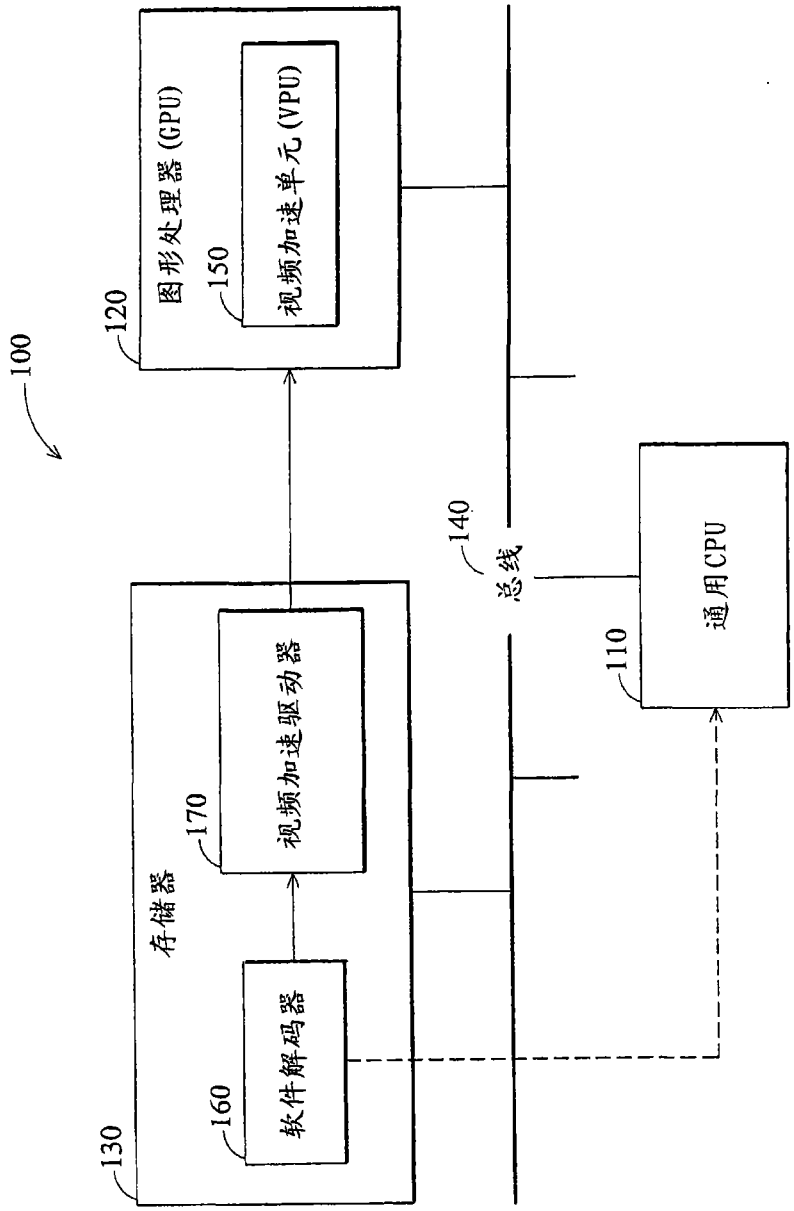

100
120
图形处理器 (GPU)
150
视频加速单元 (VPU)
140
总线
110
通用CPU
130
存储器
170
视频加速驱动器
160
软件解码器

图 1

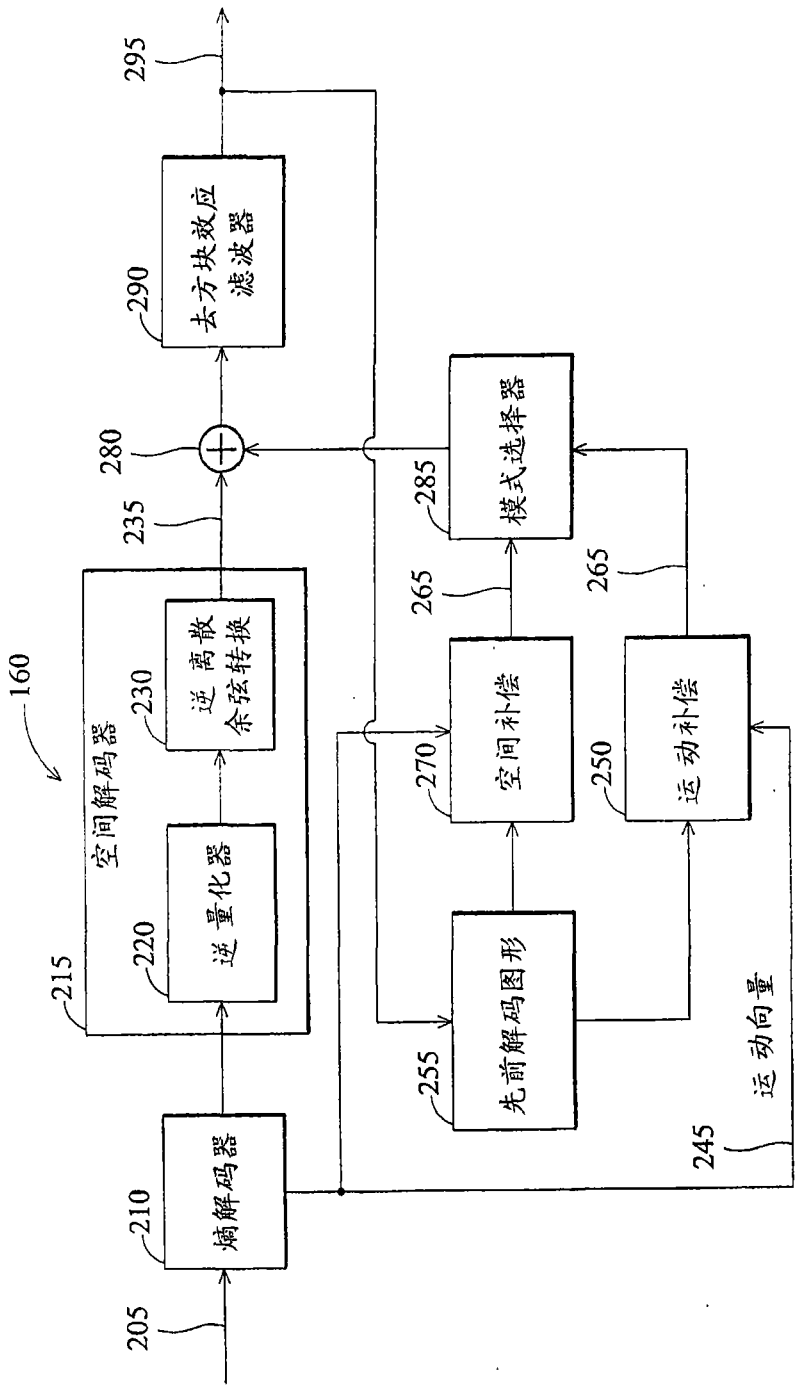

160
空间解码器
215
205
熵解码器
210
逆量化器
220
逆离散余弦转换
230
235
280
去方块效应滤波器
290
295
先前解码图形
255
空间补偿
270
运动补偿
250
245
运动向量
265
265
模式选择器
285

图 2

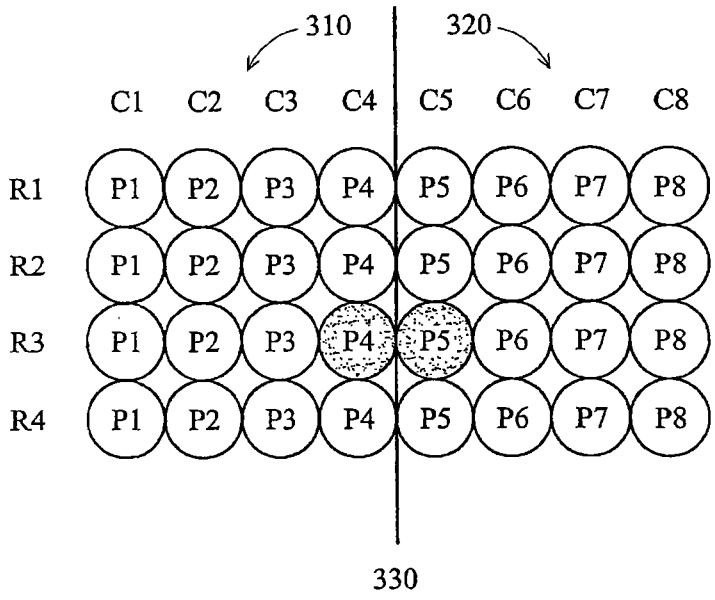


图 3

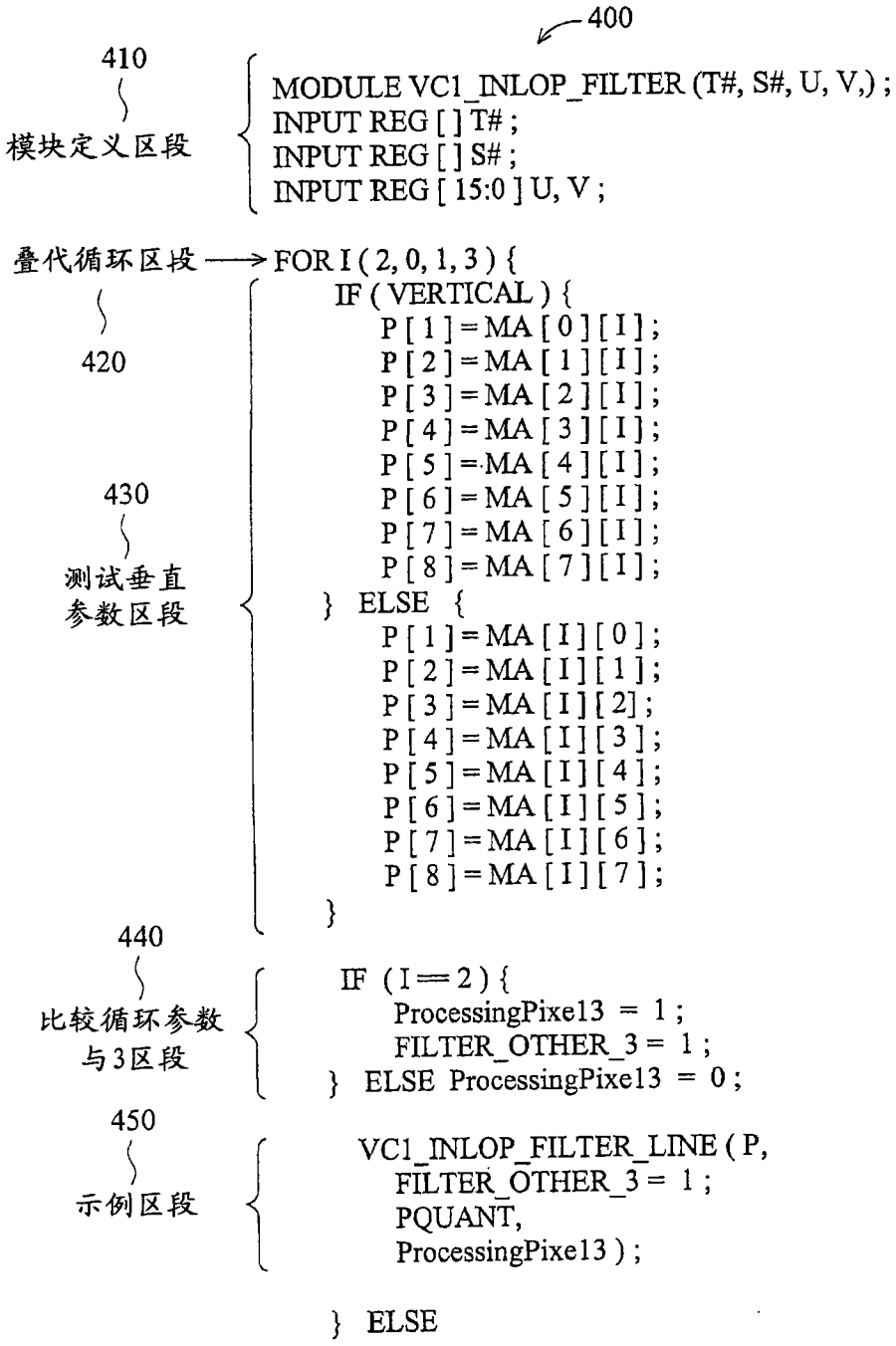

400
410
模块定义区段
MODULE VC1_INLOP_FILTER (T#, S#, U, V,) ;
INPUT REG [ ] T# ;
INPUT REG [ ] S# ;
INPUT REG [15:0] U, V ;
叠代循环区段
FOR I (2, 0, 1, 3) {
420
IF (VERTICAL) {
P [1] = MA [0] [I] ;
P [2] = MA [1] [I] ;
P [3] = MA [2] [I] ;
P [4] = MA [3] [I] ;
P [5] = MA [4] [I] ;
430
P [6] = MA [5] [I] ;
P [7] = MA [6] [I] ;
测试垂直
参数区段
P [8] = MA [7] [I] ;
} ELSE {
P [1] = MA [I] [0] ;
P [2] = MA [I] [1] ;
P [3] = MA [I] [2] ;
P [4] = MA [I] [3] ;
P [5] = MA [I] [4] ;
P [6] = MA [I] [5] ;
P [7] = MA [I] [6] ;
P [8] = MA [I] [7] ;
}
440
比较循环参数
与3区段
IF (I == 2) {
ProcessingPixel3 = 1 ;
FILTER_OTHER_3 = 1 ;
} ELSE ProcessingPixel3 = 0 ;
450
示例区段
VC1_INLOP_FILTER_LINE (P,
FILTER_OTHER_3 = 1 ;
PQUANT,
ProcessingPixel3) ;
} ELSE

图 4

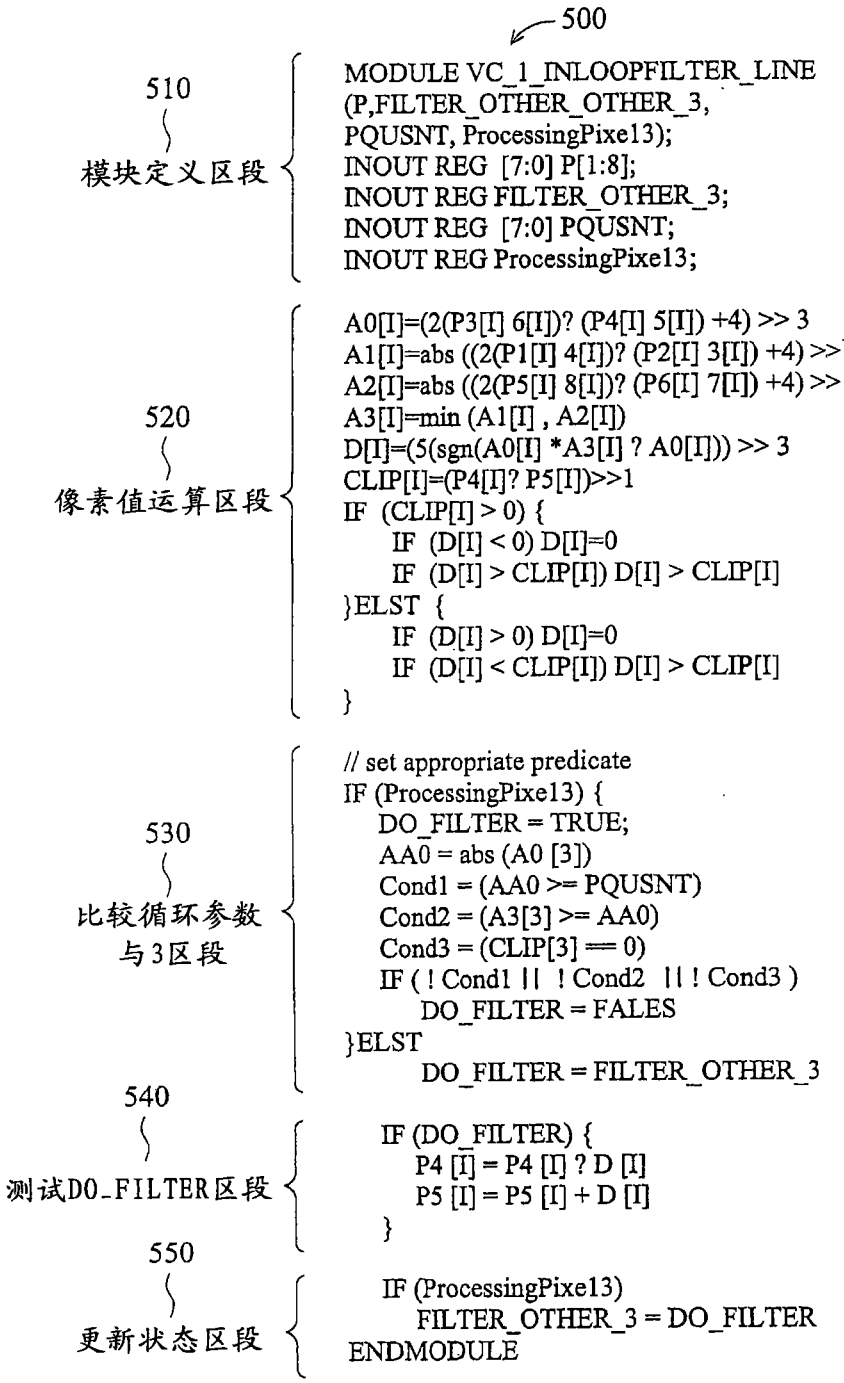

500
510
模块定义区段
MODULE VC_1_INLOOPFILTER_LINE
(P,FILTER_OTHER_OTHER_3,
PQUSNT, ProcessingPixe13);
INOUT REG [7:0] P[1:8];
INOUT REG FILTER_OTHER_3;
INOUT REG [7:0] PQUSNT;
INOUT REG ProcessingPixe13;
520
像素值运算区段
A0[I]=(2(P3[I] 6[I])? (P4[I] 5[I]) +4) >> 3
A1[I]=abs ((2(P1[I] 4[I])? (P2[I] 3[I]) +4) >> 3)
A2[I]=abs ((2(P5[I] 8[I])? (P6[I] 7[I]) +4) >> 3)
A3[I]=min (A1[I] , A2[I])
D[I]=(5(sgn(A0[I] *A3[I] ? A0[I])) >> 3
CLIP[I]=(P4[I]? P5[I])>>1
IF (CLIP[I] > 0) {
IF (D[I] < 0) D[I]=0
IF (D[I] > CLIP[I]) D[I] > CLIP[I]
}ELST {
IF (D[I] > 0) D[I]=0
IF (D[I] < CLIP[I]) D[I] > CLIP[I]
}
530
比较循环参数
与3区段
// set appropriate predicate
IF (ProcessingPixe13) {
DO_FILTER = TRUE;
AA0 = abs (A0 [3])
Cond1 = (AA0 >= PQUSNT)
Cond2 = (A3[3] >= AA0)
Cond3 = (CLIP[3] == 0)
IF (! Cond1 || ! Cond2 || ! Cond3)
DO_FILTER = FALES
}ELST
DO_FILTER = FILTER_OTHER_3
540
测试DO_FILTER区段
IF (DO_FILTER) {
P4 [I] = P4 [I] ? D [I]
P5 [I] = P5 [I] + D [I]
}
550
更新状态区段
IF (ProcessingPixe13)
FILTER_OTHER_3 = DO_FILTER
ENDMODULE

图 5

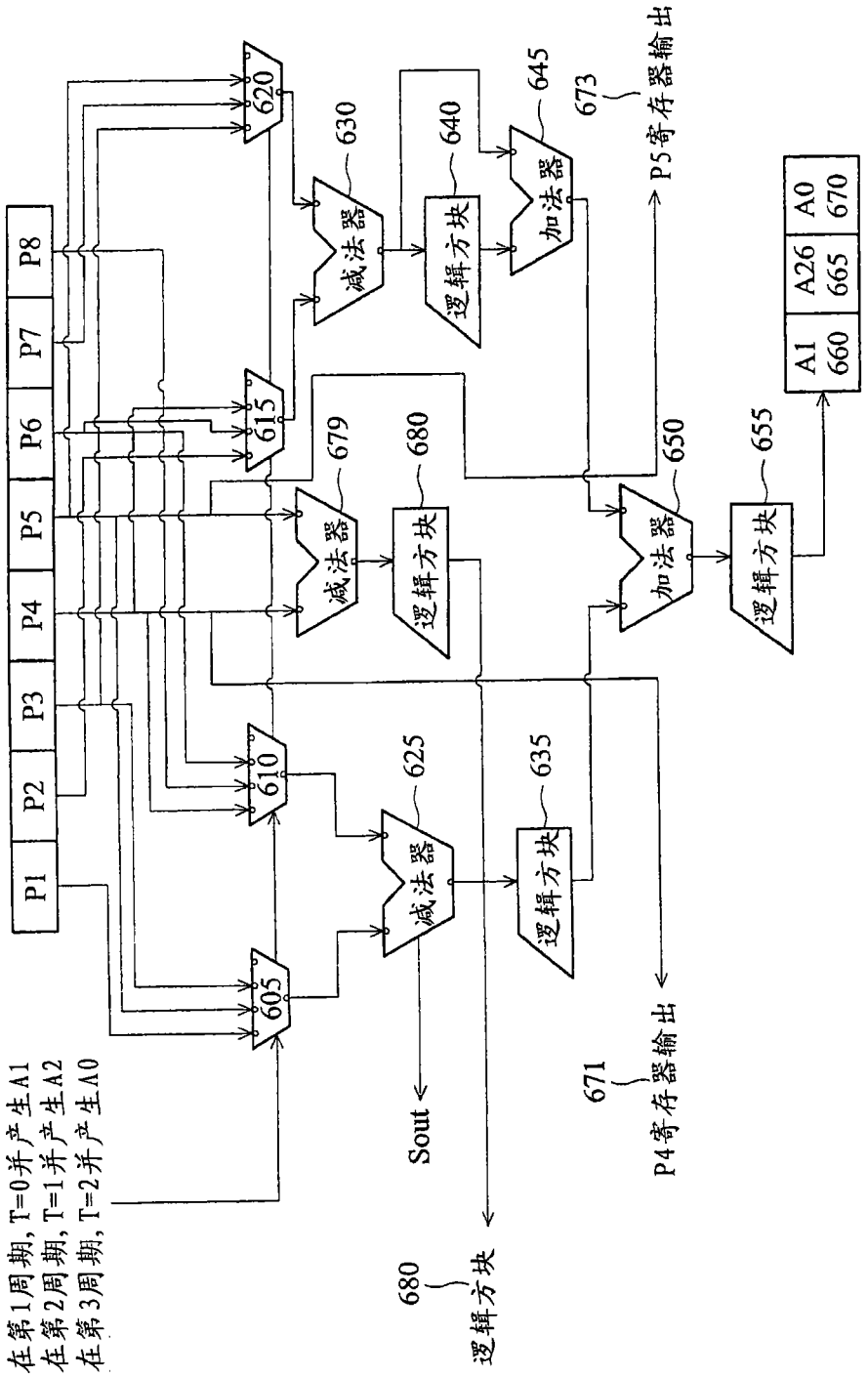

在第1周期，T=0并产生A1
在第2周期，T=1并产生A2
在第3周期，T=2并产生A0
P1
P2
P3
P4
P5
P6
P7
P8
605
610
615
620
625
630
635
640
645
650
655
660
665
670
673
679
680
671
减法器
逻辑方块
加法器
A1
A26
A0
Sout
P4寄存器输出
P5寄存器输出
680
逻辑方块

图 6A

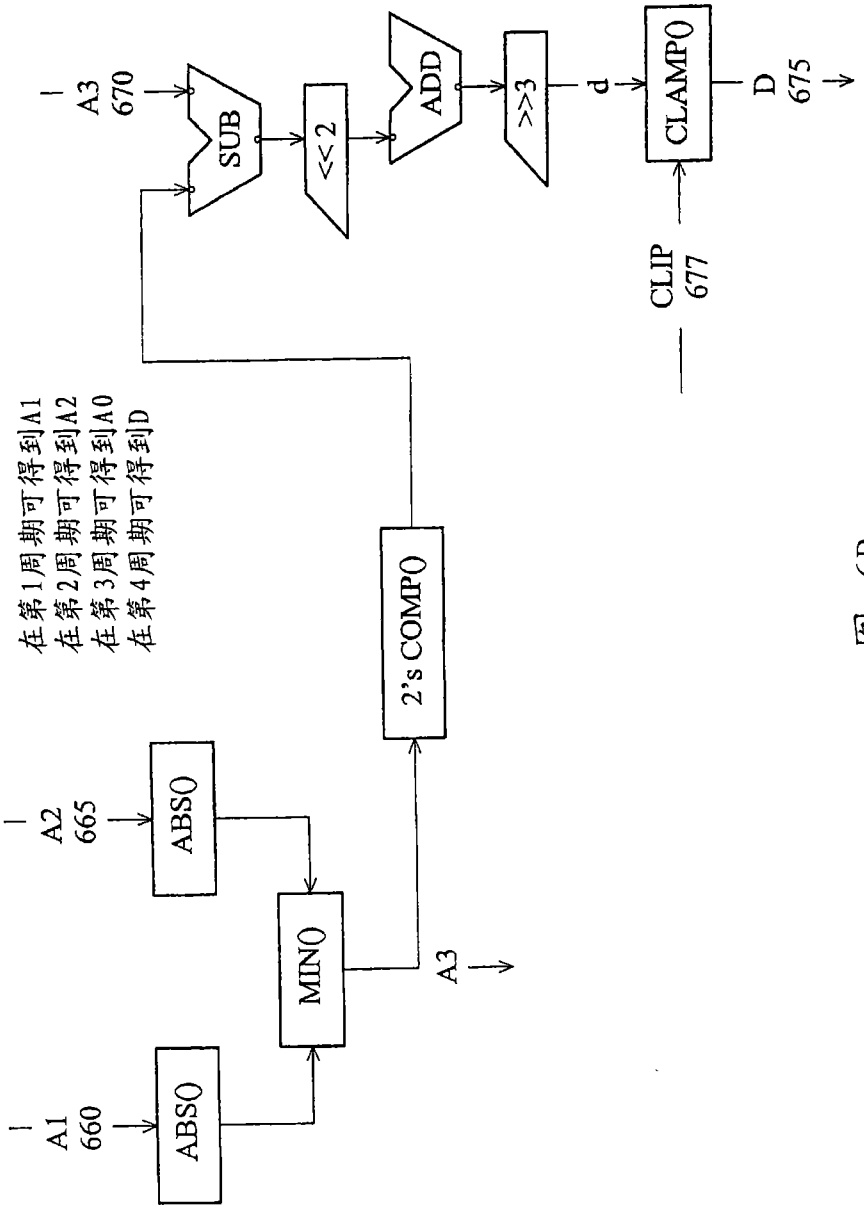

A1
660
A2
665
A3
670
ABS0
ABS0
MIN0
2's COMP0
SUB
<<2
ADD
>>3
d
CLAMP0
D
675
CLIP
677
A3
在第1周期可得到A1
在第2周期可得到A2
在第3周期可得到A0
在第4周期可得到D

图 6B

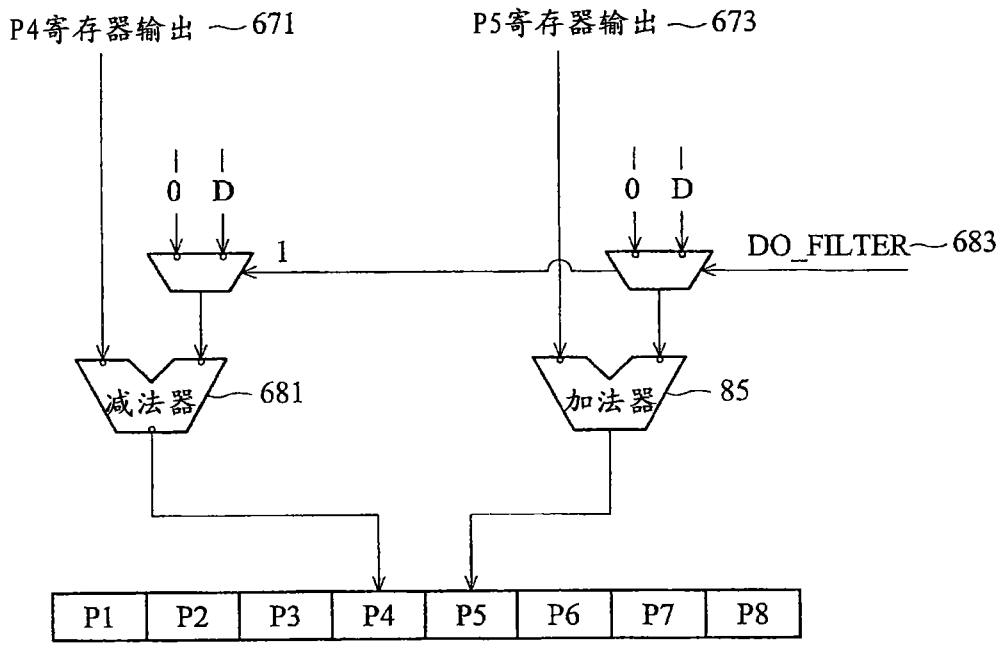

P4寄存器输出
671
P5寄存器输出
673
0
D
0
D
1
DO_FILTER
683
减法器
681
加法器
85
P1
P2
P3
P4
P5
P6
P7
P8

图 6C

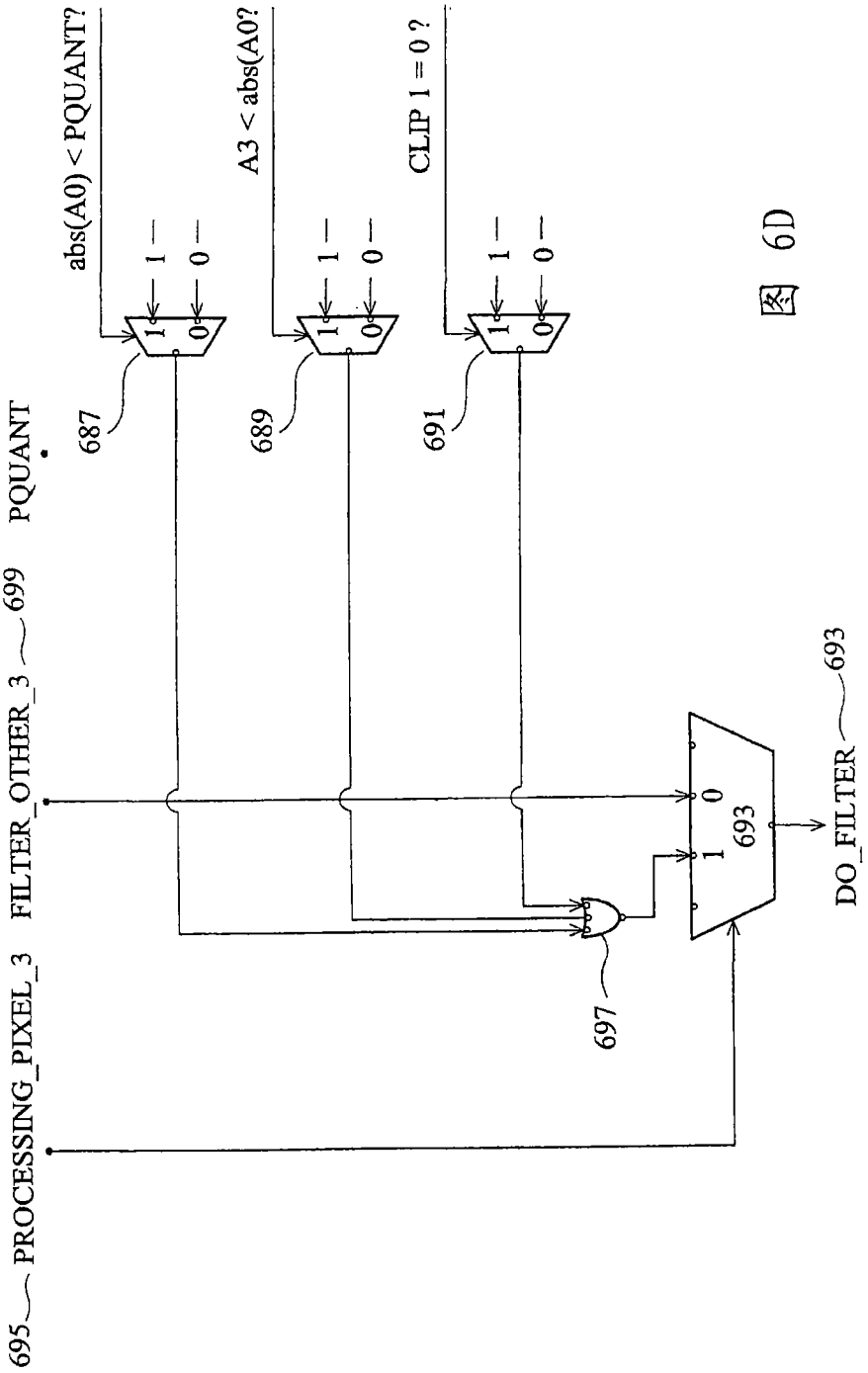

abs(A0) < PQUANT?
A3 < abs(A0?
CLIP 1 = 0 ?
687
689
691
697
693
DO_FILTER
695 PROCESSING_PIXEL_3
FILTER_OTHER_3 699
PQUANT
1
0

图 6D

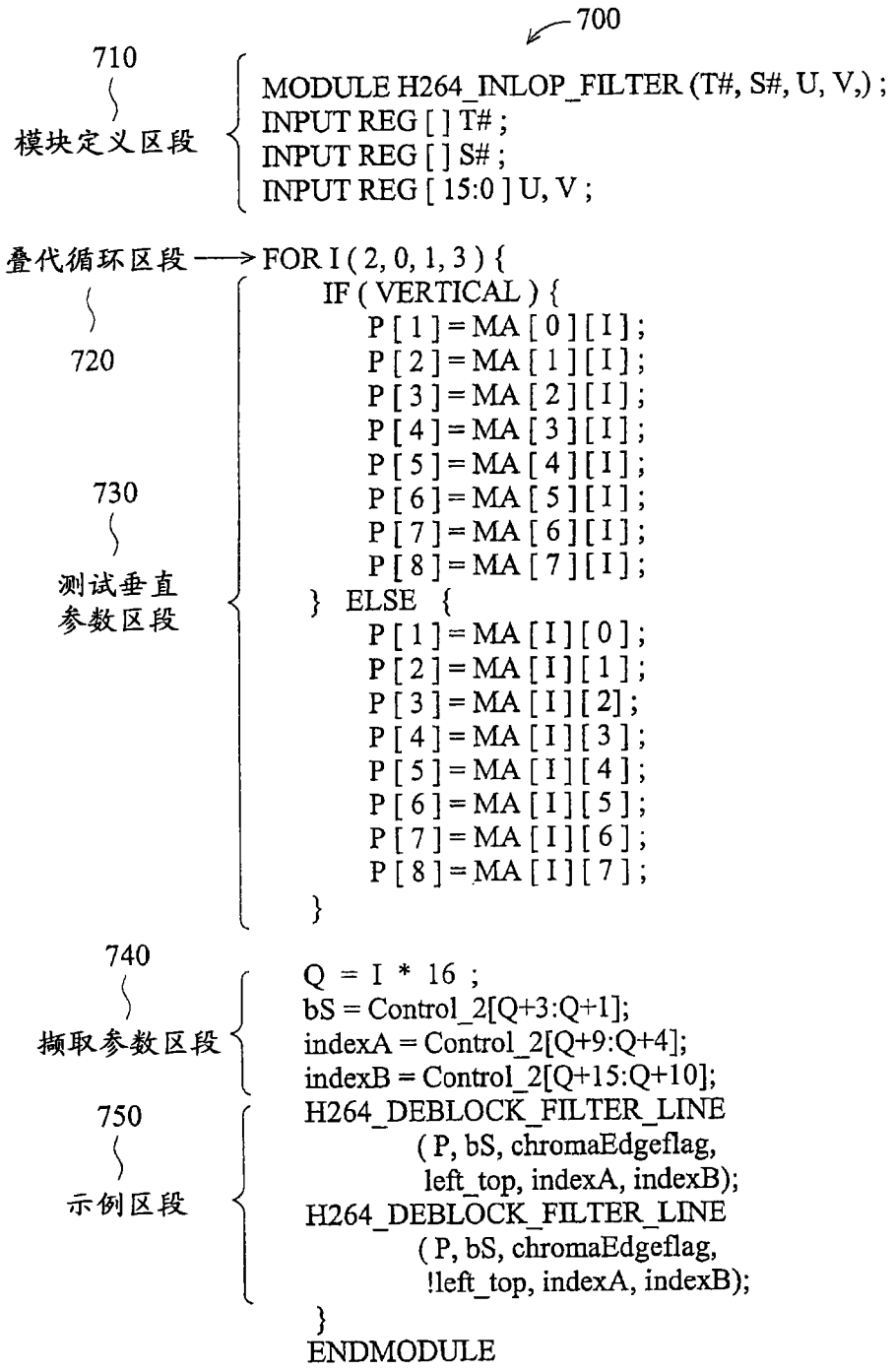

700
710
模块定义区段
MODULE H264_INLOP_FILTER (T#, S#, U, V,) ;
INPUT REG [ ] T# ;
INPUT REG [ ] S# ;
INPUT REG [15:0] U, V ;
叠代循环区段
FOR I (2, 0, 1, 3) {
720
730
测试垂直参数区段
IF (VERTICAL) {
P [1] = MA [0] [I] ;
P [2] = MA [1] [I] ;
P [3] = MA [2] [I] ;
P [4] = MA [3] [I] ;
P [5] = MA [4] [I] ;
P [6] = MA [5] [I] ;
P [7] = MA [6] [I] ;
P [8] = MA [7] [I] ;
} ELSE {
P [1] = MA [I] [0] ;
P [2] = MA [I] [1] ;
P [3] = MA [I] [2] ;
P [4] = MA [I] [3] ;
P [5] = MA [I] [4] ;
P [6] = MA [I] [5] ;
P [7] = MA [I] [6] ;
P [8] = MA [I] [7] ;
}
740
撷取参数区段
Q = I * 16 ;
bS = Control_2[Q+3:Q+1];
indexA = Control_2[Q+9:Q+4];
indexB = Control_2[Q+15:Q+10];
750
示例区段
H264_DEBLOCK_FILTER_LINE
(P, bS, chromaEdgeflag,
left_top, indexA, indexB);
H264_DEBLOCK_FILTER_LINE
(P, bS, chromaEdgeflag,
!left_top, indexA, indexB);
}
ENDMODULE

图 7

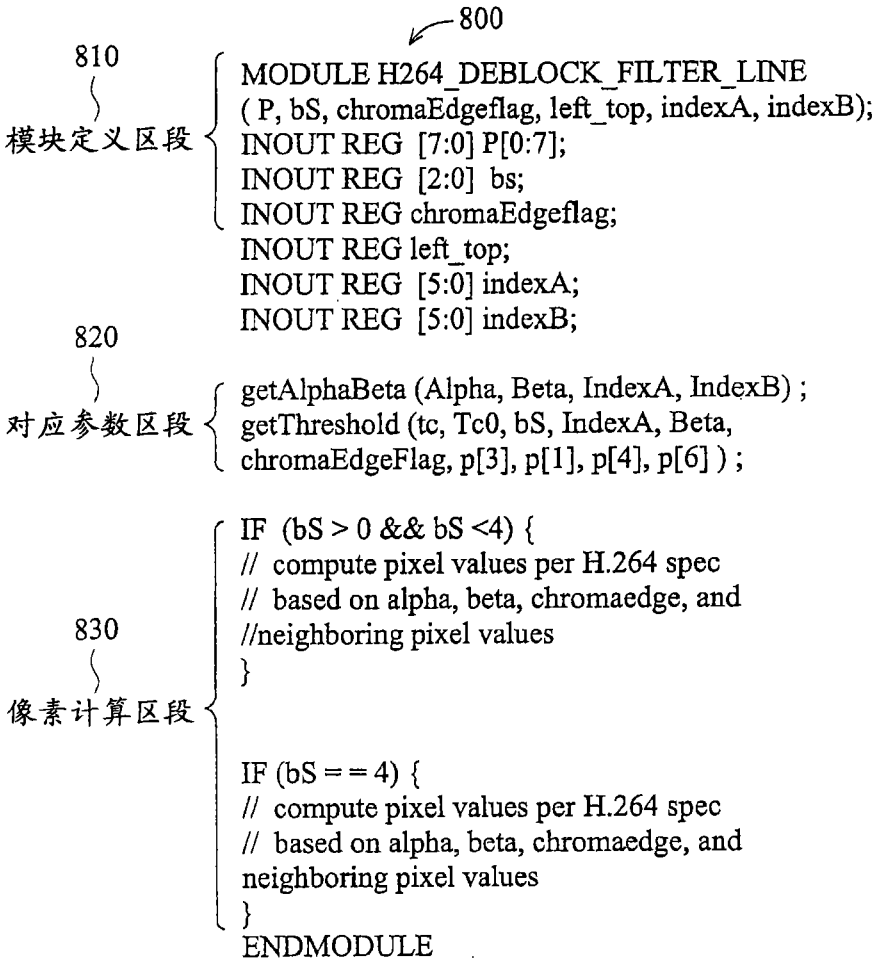

800
810
模块定义区段
MODULE H264_DEBLOCK_FILTER_LINE
(P, bS, chromaEdgeflag, left_top, indexA, indexB);
INOUT REG [7:0] P[0:7];
INOUT REG [2:0] bs;
INOUT REG chromaEdgeflag;
INOUT REG left_top;
INOUT REG [5:0] indexA;
INOUT REG [5:0] indexB;
820
对应参数区段
getAlphaBeta (Alpha, Beta, IndexA, IndexB) ;
getThreshold (tc, Tc0, bS, IndexA, Beta,
chromaEdgeFlag, p[3], p[1], p[4], p[6]) ;
830
像素计算区段
IF (bS > 0 && bS <4) {
// compute pixel values per H.264 spec
// based on alpha, beta, chromaedge, and
//neighboring pixel values
}
IF (bS = = 4) {
// compute pixel values per H.264 spec
// based on alpha, beta, chromaedge, and
neighboring pixel values
}
ENDMODULE

图 8A

```
MODULE getAlphaBeta (Alpha, Beta, IndexA,
IndexB)
OUTPUT REG [ 7:0 ] Alpha;
OUTPUT REG [ 7:0 ] Beta;
INPUT    REG [ 5:0 ] IndexA;
INPUT    REG [ 5:0 ] IndexB;

ROMTable  [ 7:0 ]  TableAlpha [ 0:255 ];
ROMTable  [ 4:0 ]  TableBeta [ 0:255 ];

Alpha = TableAlpha [ IndexA ];
Beta   = {3'b0, TableBeta [ IndexB ]}; // zero-extend
ENDMODULE
```

850

```
MODULE getThreshold (tc, Tc0, bS, IndexA, Beta,
chromaEdgeFlag, p0, p2, q0, q2)
OUTPUT REG [ 4:0 ] tc;
OUTPUT REG [ 4:0 ] Tc0;
INPUT    REG [ 2:0 ] bS;
INPUT    REG [ 5:0 ] IndexA;
INPUT    REG [ 7:0 ] Beta;
INPUT    REG chromaEdgeFlag;
INPUT    REG [ 7:0 ] p0, p2, q0, q2;
WIRE [ 1:0 ] t0;
WIRE apt, aqt;
ROMTable  [ 4:0 ]  TableThr [ 1:3 ] [ 0:51 ];
REG [ 4:0 ] Tc0;

Tc0 =  TableThr [bS] [IndexA];
apt =  ( abs (p2-p0) < Beta ? 1 : 0 );
apt =  ( abs (q2-q0) < Beta ? 1 : 0 );
IF (chromaEdgeFlag) t0 = 1;
ELSE t0 = apt +aqt;
Tc = Tc0 + t0;
ENDMODULE
```

870

图 8B

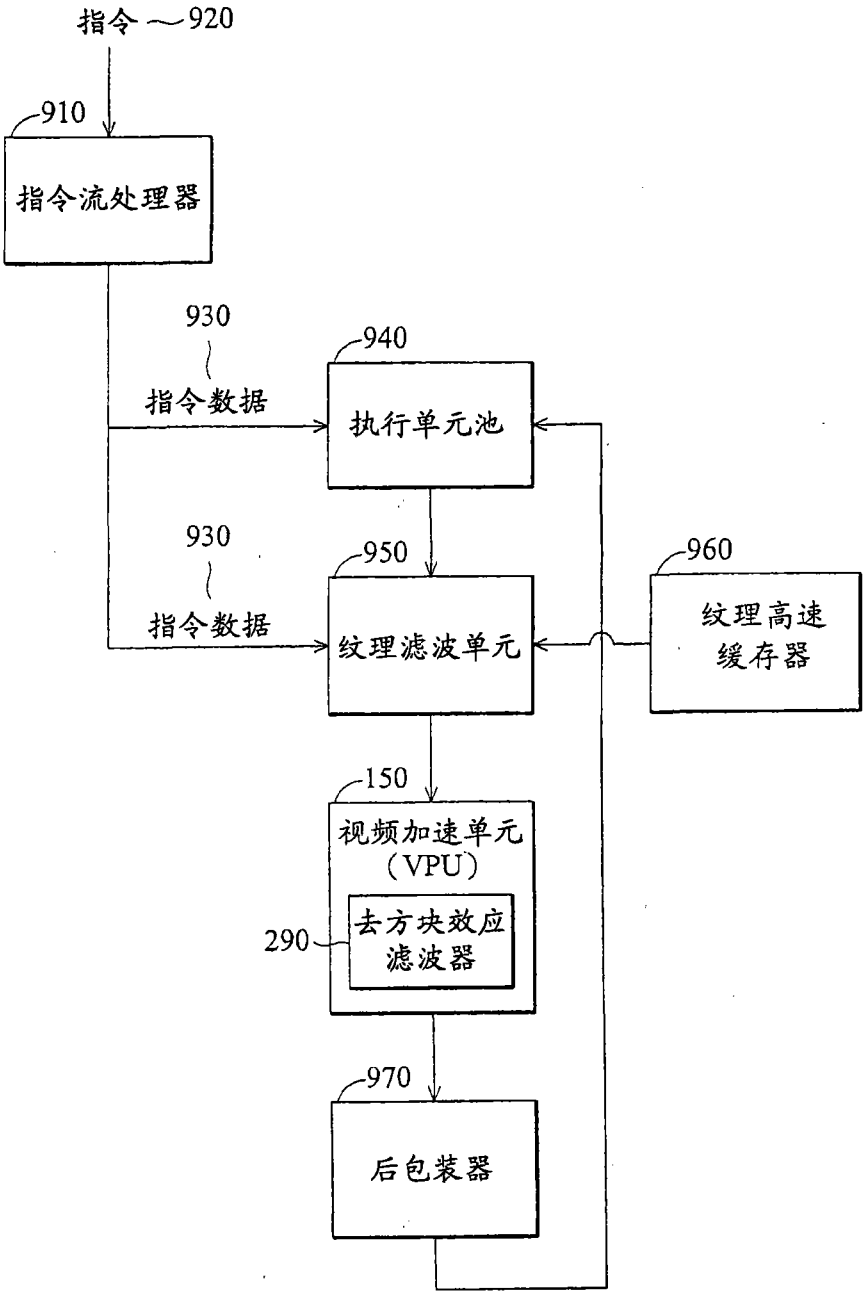

指令
920
910
指令流处理器
930
指令数据
940
执行单元池
930
指令数据
950
纹理滤波单元
960
纹理高速缓存器
150
视频加速单元（VPU）
290
去方块效应滤波器
970
后包装器

图 9

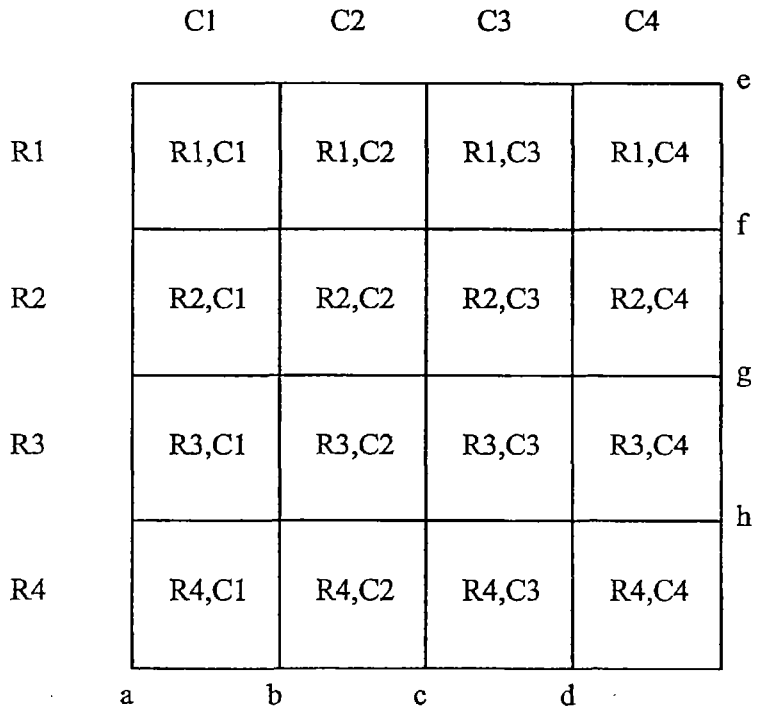

C1
C2
C3
C4
R1
R2
R3
R4
R1,C1
R1,C2
R1,C3
R1,C4
R2,C1
R2,C2
R2,C3
R2,C4
R3,C1
R3,C2
R3,C3
R3,C4
R4,C1
R4,C2
R4,C3
R4,C4
a
b
c
d
e
f
g
h

图 10