

[19] 中华人民共和国国家知识产权局



[12] 发明专利申请公开说明书

[21] 申请号 200610080355.1

[51] Int. Cl.

G03F 7/20 (2006.01)

G03F 1/14 (2006.01)

G06T 15/00 (2006.01)

[43] 公开日 2006 年 11 月 1 日

[11] 公开号 CN 1854902A

[22] 申请日 2002.9.9

[21] 申请号 200610080355.1

分案原申请号 02822476.0

[30] 优先权

[32] 2001.9.12 [33] US [31] 09/954,721

[71] 申请人 麦克罗尼克激光系统公司

地址 瑞典泰比

[72] 发明人 托布乔恩·桑德斯特罗姆

马丁·奥尔森 珀·埃尔姆福斯

斯蒂芬·古斯塔夫森

[74] 专利代理机构 北京市柳沈律师事务所

代理人 黄小临 王志森

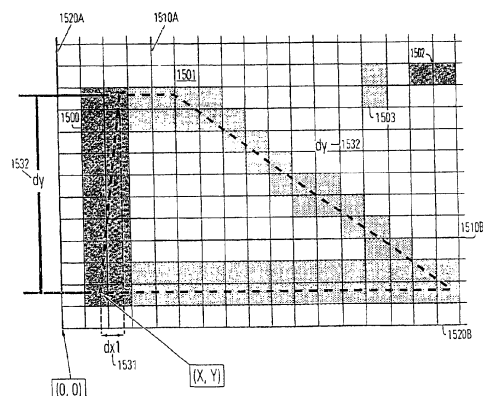
权利要求书 3 页 说明书 86 页 附图 50 页

[54] 发明名称

一种呈现的方法

[57] 摘要

本发明涉及使用子像素分辨率来呈现高精度图像，可适用于产生网线板和大面积掩模、直接写入图形和查看网线板或其他图形的工件。本发明的多个方面可以应用到 SLM 和扫描技术。本发明包括一种调制微镜阵列以建立和加强图像的方法。本申请公开了重叠区、保护区的使用、将多边形移动和重新定大小、使用两个分辨率级，表示区域、定义边、定位边和角。



1. 一种呈现的方法, 包括:
提供对应于存储器的较小寻址窗口;
提供较大寻址窗口, 包括较小寻址窗口和在较小寻址窗口之外的保护区, 所述保护区是可以寻址的, 但是不对应于存储器;
接收在所述较大寻址窗口中包括的多边形;
将所述多边形的至少部分呈现到所述较大寻址窗口中; 并且
向存储器写入用于在所述较小寻址窗口内的多边形的呈现部分的数据, 并且丢弃在所述较小窗口之外的多边形的呈现部分的数据。
2. 按照权利要求1的方法, 其中所述呈现步骤被执行而不用区别多边形的部分在所述较小寻址窗口内部或外部。
3. 按照权利要求1的方法, 其中这种方法可以被执行而不用将多边形截割以适合所述较小寻址窗口。
4. 按照权利要求1的方法, 还包括步骤: 过滤一组多边形, 以便多边形的至少一部分位于较小寻址窗口中。
5. 按照权利要求1的方法, 其中在向存储器控制器写入用于呈现部分的数据之前由过滤器执行所述丢弃所述数据, 因此降低了存储器带宽负荷。
6. 按照权利要求1的方法, 其中通过比较存储器地址信号与较小寻址窗口的有效地址并且使用比较结果来控制一个或多个存储器仲裁信号来执行所述丢弃所述数据。
7. 按照权利要求2的方法, 还包括步骤: 过滤一组多边形, 以便多边形的至少一部分位于较小寻址窗口中。
8. 按照权利要求3的方法, 还包括步骤: 过滤一组多边形, 以便多边形的至少一部分位于较小寻址窗口中。
9. 一种向较大寻址窗口呈现多边形的方法, 所述较大寻址窗口包括对应于存储器的较小寻址窗口和在较小寻址窗口之外的保护带, 所述保护带是可寻址的, 但是不对应于存储器, 所述方法包括:
接收在较大寻址窗口中包括的多边形;
重复选择多边形的一部分;
将所述部分转换为光栅化的表示数据;

处理所述光栅化的表示数据以便向存储器写入在较小寻址窗口内的所述光栅化的表示数据，并且不向存储器写入在较小寻址窗口外的所述光栅化的表示数据。

10. 按照权利要求 9 的方法，其中所述转换步骤被执行而不用区别多边形的部分在所述较小寻址窗口内部或外部。

11. 按照权利要求 9 的方法，其中这种方法可以被执行而不用将多边形截割以适合所述较小寻址窗口。

12. 按照权利要求 9 的方法，还包括：在写入到存储器之前，过滤光栅化表示数据，因此降低了存储器带宽负荷。

13. 按照权利要求 9 的方法，还包括：比较存储器地址信号与较小寻址窗口的有效地址，并且使用比较结果来控制一个或多个存储器仲裁信号。

14. 按照权利要求 9 的方法，还包括步骤：过滤一组多边形，以便多边形的至少一部分位于较小寻址窗口中。

15. 按照权利要求 10 的方法，还包括步骤：过滤一组多边形，以便多边形的至少一部分位于较小寻址窗口中。

16. 按照权利要求 11 的方法，还包括步骤：过滤一组多边形，以便多边形的至少一部分位于较小寻址窗口中。

17. 按照权利要求 9 的方法，还包括步骤：将一个几何形状分解为其大小适合保护区的一组多边形。

18. 按照权利要求 10 的方法，还包括步骤：将一个几何形状分解为其大小适合保护区的一组多边形。

19. 按照权利要求 11 的方法，还包括步骤：将一个几何形状分解为其大小适合保护区的一组多边形。

20. 一种器件，用于呈现在较大寻址窗口内包括的多边形，所述较大寻址窗口包括较小寻址窗口和较小寻址窗口外部的保护带，所述器件包括：

输入线；

连接到输入线的呈现器，用来接收在较大寻址窗口内包括的多边形和重复地将多边形的一部分转换为一个或更多寻址命令与数据；

存储器，与较小寻址窗口一致；

连接到呈现器的存储器控制器，用来处理寻址指令和数据，以将在较小寻址窗口内的数据写入存储器，并且丢弃在较小寻址窗口之外的数据部分。

21. 按照权利要求 20 的方法，其中呈现器用来转换所述部分而不区别在较小寻址窗口内外的部分。

22. 按照权利要求 20 的方法，其中所述器件不将所述多边形截割以适应在较小寻址窗口内。

23. 一种器件，用于向存储器写入在较大寻址窗口内包括的多边形的一个部分，所述较大寻址窗口包括较小寻址窗口和在较小寻址窗口之外的保护带，所述部分由一个或多个寻址指令和数据表示，所述器件包括：

输入线；

与较小寻址窗口一致的存储器；

连接到所述输入线和存储器的存储器控制器，用来参照较大寻址窗口来接收寻址指令和数据，向存储器写入在较小寻址窗口内的数据的部分，并丢弃在保护带内的数据的部分。

一种呈现的方法

本申请是发明名称为“用于高精度光刻技术的图形引擎”(申请号: 02822476.0; 申请日: 2002年9月9日)的申请的分案申请。

技术领域

本发明涉及一种呈现的方法。本发明可以应用于网线板(reticle)和大面积掩模的生产,图形的直接写入和网线板或其他带有图形的工件的检验。本发明的方面可以应用于 SLM 和扫描技术。

背景技术

用于半导体制造的设备包括用于在网线板和大面积掩模上产生图像、用于直接写入芯片的图形和用于检验带有图形的工件的写入器和检验设备。渐渐地,随着处理器、存储器电路和其他半导体取得越来越大的容量,芯片变得越来越复杂密集。特别是存储器电路和一般来说所有具有小图形尺寸的电路已经变得更为密集。用于这些电路的图形已经变得比电路更复杂,因为光接近度和激光接近度校正图形已经被增加到所述图形。响应于对在芯片上的较小图形和更紧密的临界尺寸的要求,所述设备和写入策略已经变得越来越复杂。多年前开发的用于满足半导体制造设备的需要的设备和写入策略的一个说明可以在 1996 年颁发给 Teitzel 等的美国专利第 5,533,170 中找到。

对于制造商的另一个挑战是补偿在工艺化学中的变化,诸如抗蚀剂对曝光辐射的响应中的变化。在不同类型和甚至不同批次的抗蚀剂对曝光的响应中出现变化。

当制造商努力与摩尔定律保持一致时,持续需要可以处理大量的几何图形和在工件上产生精确图形的写入器和检验设备。因此在产生所需要的精密图形的同时,需要补偿在抗蚀剂和其他处理变量中的变化的技术、方法和设备。

发明内容

本发明的方面可以被组合来使用例如 SLM 投射技术来处理大量的几何图形和产生在工件上的精确图形。本发明的其他方面便利了补偿在抗蚀剂响应和其他处理变量中的变化。

本发明的一个实施例是一个数据路径和多个处理器，它接受被分割的几何形状数据来作为输入，并且产生用于在微镜阵列中的微镜的驱动值来作为输出。在这个实施例中，所述方法包括：对应于从多个重叠曝光向工件的区域上的意欲的能量分布的图形而将工件映射到重叠区域中；使用用于像素图和用于对应于在像素图中的像素的微像素阵列的双分辨率存储器来将分割的多边形表达为灰度级像素值；根据重叠区域来计算用于灰度级像素值的曝光值；根据独立的微镜特征来勾画曝光值；并且输出所勾画的曝光值。根据用于印刷的遍次的数量，所述映射步骤可以产生 9 个重叠区域、81 个重叠区域或一些其他数量的重叠区域。所述计算步骤可以使用重叠区域来考虑从多个曝光产生的累积量。这种方法可以还包括在连接到一组镜子像素驱动器的至少一个分段缓冲器中缓冲所勾画的曝光值。所述缓冲器可以是双段的，第一段对应于表达引擎，第二段对应于微镜阵列的区域。缓冲器的第一段和第二段可以不同。可以使用多个缓冲器来收集数据以输出。缓冲器的输出可以到数模转换器。这个实施例的另一个方面是重叠区域考虑到被施加到工件的能量的脉冲到脉冲变化。在印刷的过程期间，可以调整重叠区域的照明以考虑到在工件上的抗蚀剂的影响，诸如抗蚀剂的老化、由多次曝光对抗蚀剂的启动和抗蚀剂的敏感度的变化，尤其是在工件的边或角。

第一实施例的各种子集合本身是有益的。重叠区域在面向标记的处理中的应用是有益的。将几何图形表达为在包括像素图和对应于在像素图中的像素的微像素阵列的双分辨率存储器中的灰度级像素值是有益的。独立地或组合地考虑到在标记之间的重叠、考虑到在多遍次之间的重叠和考虑到在曝光能量中的脉冲到脉冲变化的计算全是有益的。基于独立的微镜特征而不是微镜阵列的一般特征的曝光值的勾画是有益的。第一实施例的这些方面的成对组合也是有益的。

本发明的一个附加实施例是使用保护带来作为截割的替代。这包括：提供对应于存储器的较小寻址窗口；提供一个较大寻址窗口来建立在所述较小

寻址窗口周围的保护带，它是可以寻址的，但是不对应于存储器；接收在所述较大寻址窗口中包括的多边形；将所述多边形的至少部分呈现到所述较大寻址窗口中；并且在丢弃在所述较小窗口之外的多边形的呈现部分的同时向存储器写入用于在所述较小寻址窗口内的多边形的呈现部分的数据。这个实施例的另一个方面是所述呈现步骤被执行而不用区别多边形的部分在所述较小寻址窗口内部或外部。这种方法可以被执行而不用将多边形截割以适合所述较小寻址窗口。通过比较存储器地址与较小寻址窗口以使用比较结果来控制诸如写使能信号的一个或多个存储器仲裁信号，多边形的呈现部分的丢弃可以发生在数据被过滤器发送到存储器控制器之前、或者发生在存储器控制器中。这个实施例及其各个方面的进一步被增强可以通过包括过滤一组多边形的标记以便至少所接收和呈现的多边形的一部分位于所述较小寻址窗口内。

在替代实施例上的一个变化是向较大寻址窗口呈现多边形，所述较大寻址窗口包括对应于存储器的较小寻址窗口和在较小寻址窗口之外的保护带，所述保护带是可寻址的，但是不对应于存储器。这种方法可以包括：接收在较大寻址窗口中包括的多边形；重复选择多边形的一部分和将其转换为光栅化的表示数据；处理所述光栅化的表示数据以便在较小寻址窗口内的部分被写入存储器并且在那个窗口外的数据不被写入存储器中。这个变化的多个方面可以类似于在这个实施例上的第一变化。所述转换步骤可以被执行而不用区别在较小寻址窗口内外的多边形部分的区别。可以避免将多边形截割以适合所述较小寻址窗口。光栅化的表示数据可以在被写入到存储器控制器或存储器之前被过滤。或者，如上所述，它可以在存储器控制器中被过滤。输入的多边形可以被过滤，以便每个多边形的至少一部分位于较小寻址窗口内。这种变化可以还包括限制所接收或输入的多边形以便那些多边形足够小以适应在保护带内。

对应于前一个实施例的器件可以呈现在较大寻址窗口内包括的多边形，所述较大寻址窗口包括较小寻址窗口和较小寻址窗口外部的保护带。这个器件可以包括：连接到输入线的呈现器，用来接收在较大寻址窗口内包括的多边形和重复地将多边形的一部分转换为光栅化的表示数据；存储器，对应于较小寻址窗口。

连接到呈现器的存储器控制器被适配来处理光栅化的表示数据，并且将

在较小寻址窗口内的光栅化的表示数据写入存储器，并且丢弃在较小寻址窗口之外的光栅化的表示数据。这个器件的呈现器可以被适配来将多边形的部分转换为光栅化的表示数据而不区别在较小寻址窗口内外的部分。所述器件可以操作而不用将所接收的多边形截割以适应在较小寻址窗口内并且不要求那个所接收的多边形被包括在较小寻址窗口内。这个器件的另一个变化是用于向存储器写入在较大寻址窗口内包括的多边形的一部分的器件，所述较大寻址窗口包括较小寻址窗口和在较小寻址窗口之外的保护带，由光栅化的表示数据表示的部分和一个器件包括：输入线；对应于较小寻址窗口的存储器；连接到输入行和存储器的存储器控制器，用来引用较大寻址窗口来接收光栅化的表示数据、向存储器写入在较小寻址窗口内的光栅化的表示数据的部分、丢弃在保护带内的光栅化的表示数据的部分。这个变化的多个方面可以从或者前述方法或者这个实施例的其他变化来获得。

本发明的另一个实施例是使用至少两个分辨率级来表示区域的方法，包括：接收由边限定的多边形表示；重复选择多边形的一部分来用于呈现，所述部分对应于像素；根据多边形表示的边来表示在第一数据结构内的像素被填充、空或部分被填充。在通过被填充或空的子像素的阵列表示在第二数据结构中的部分填充的像素中，设置定义多边形表示的边的阵列和设置包括至少 3×3 子像素的阵列。或者，所述阵列可以包括 4×4 或 8×8 子像素。所述多边形可以是四边形、凸四边形、梯形或数据线。四边形或梯形表示可以具有 0 长度的一边。用于被填充的、空的或部分被填充的像素表示可以使用两个数据比特。所述方法可以还包括：保存对应于在阵列中的被填充的或空的子像素的部分被填充像素的总计。这些概述可以包括被填充或空的子像素的计数或子像素的加权评估。

本发明的一个相关实施例是使用至少两个分辨率级来表示区域的方法，包括：接收由边限定的多边形表示；重复选择通过多边形的一部分来用于呈现，所述部分对应于像素；根据多边形表示的边来将所述像素分类为被填充、空或部分被填充；通过被分配强度级而不是填充或空值的子像素的阵列来表示部分填充的像素。子像素的阵列限定多边形表示的边。这些阵列可以包括至少 3×3 子像素、 4×4 子像素、 8×8 子像素或 16×16 子像素。同上面一样，所述多边形具有几种不同的形式。将像素分类为被填充、空或部分被填充的所述像素图可以包括用于每个像素的两个比特。所述方法可以还包括：保存

对应于在阵列中子像素的强度级的部分被填充像素的总计。这些概述可以子像素强度级的概述或子像素的加权评估。

使用两个分辨率级来表示具有至少一个边的区域的数据结构实施例包括：至少一个存储器；存储在存储器中的至少一个像素图，将像素表示为被填充、空或部分被填充；对应于像素图的、存储在存储器中的至少一个子像素图，它通过被填充或空的子像素阵列来表示部分被填充的像素，所述子像素定义区域的边，子像素阵列包括至少 3×3 子像素；被填充或空的像素，它被表示而不使用子像素值来表示它们。这个实施例的另一个方面是它可以还包括概述在阵列中的被填充或空子像素的子像素概述。这个实施例可以还包括用于像素图和子像素阵列的独立可寻址的存储器。它可以还包括用于像素图和子像素概述图的独立可寻址存储器。

在这个实施例上的变化是使用两个分辨率级来表示至少一个梯形的数据结构，所述梯形具有第一和第三平行边以及第二和第四相对边，所述第一和第三平行边彼此平行并且与一个基准轴平行，所述数据结构包括：至少一个存储器；存储在存储器中的至少一个像素图，将像素表示为被填充、空或部分被填充；对应于在像素图的一个像素的、存储在存储器中的至少一个子像素阵列，它通过子像素阵列来表示在平行边和相对边上的像素。在这些阵列中的所述子像素可以是被填充的或空的。这些阵列包括至少 3×3 子像素、 4×4 子像素、 8×8 子像素或 16×16 子像素。这种数据结构可以还包括灰度值概述存储器，用于概述或者被填充或者空的子像素。所述概述可以基于被填充或空的子像素的计数或子像素的加权评估。所述子像素可以具有二进制值或可以另外具有中间强度值。像素图和子像素阵列可以被存储在独立可寻址的存储器中。梯形的平行边之一可以具有 0 长度。

本发明的多个方面的另一个实施例是一个协议，用于：使用两个分辨率级来发送表示具有多个边的多边形的图形数据，包括：通过像素阵列来表示多边形的部分，所述像素被分配被填充、空或部分被填充的一个值；通过子像素阵列来表示部分被填充的像素，所述子像素被分配被填充或空的一个值，用于限定边的至少一部分，所述阵列包括至少 3×3 个子像素；使用至少第一信道和独立的第二信道来发送像素阵列和多个子像素阵列的表示，第一信道用于像素阵列的表示，第二信道用于子像素的阵列。在这个实施例中，子像素阵列可以或者包括至少 4×4 、 8×8 或 16×16 个子像素。像素阵列的表示

可以被行程编码。所述多边形可以是梯形，所述梯形具有与基准轴平行的第一和第三平行边；多个沿着第一或第三平行边的部分被填充的像素可以由单个子像素阵列来表示。这个协议可以还包括：保存子像素概述图，用于总计在阵列中的被填充或空子像素。所述概述可以是在阵列中的被填充或空子像素的计数或子像素的加权评估。所述子像素除了被填充或空值之外也可以具有中间强度值。像素和子像素阵列可以被存储在独立可寻址的存储器中。类似地，像素和子像素概述图可以被存储在独立可寻址的存储器中。第一或第三平行边的相邻像素可以由同一子像素图来表示。第一或第三平行边可以被分配 0 长度。

本发明的另一个实施例是一种计算用于对应于多边形的边的部分的多值像素的值的方法，包括：提供子像素阵列；对应于具有特定方向的边而提供一组预先计算的子像素条形图；通过应用所述预先计算的子像素条形图而将多边形的边的部分表示为被填充或空的子像素；超级采样对应于像素的一组子像素，并且向所述像素分配一个灰度值。在这种方法中，预先计算的子像素条形图可以还被表示为填充条的阵列，应用预先计算的子像素图可以还包括：向对应于边或边的延长线与由子像素的阵列表示的区域的边界的交点的填充条应用偏移值。

使用子像素阵列来表示多边形边的至少一部分的方法的另一个方面是在具有边的一个区域内定义多边形的边的方法，所述区域被细分为子像素。这种方法包括：提供对应于多边形边和所述区域的边的可能相交和方向的多个预先计算的子像素条形图，其中所述可能相交被限于沿着所述区域的边的离散位置，所述可能方向被限于连接所述离散位置的方向；确定在多边形边的方向中、多边形边与区域的边的两个相交的三中之二；并且应用对应于在所述方向中的两个相交的三中之二的、预先计算的子像素条形图之一。在本发明的一个方面，所述区域被细分为不超过 256 个子像素，并且所述离散位置被限制为每个子像素不超过 65 个位置。或者，所述区域被细分为不超过 64 个子像素，并且所述离散位置被限制为每个子像素不超过 33 个位置。在另一个配置中，所述区域被细分为不超过 32×16 个子像素，并且沿着子像素的边存在不超过 17 个离散位置；或所述区域可以被细分为不超过 16×8 个子像素，子像素的一个边具有不超过 9 个离散位置。这个实施例的另一个方面是预先计算的子像素条形图可以被限于形成大约 45 度范围的一组可能的方向，并且

这个预先计算的子像素条形图的范围可以被转换为覆盖大约 180 度的范围。由多边形边相交的区域的边可以或者是相对边或相邻边。在这个实施例中的变化可能涉及使用多边形边和多边形边的延伸线来构造与区域的边的相交。多边形边的方向可以在本方法中被确定和利用，或者可以忽略它，因为两个相交限定多边形边。

子像素条形图选择是本发明的另一个实施例。一种选择子像素条形图以表示与一个区域相交的边的方法，所述区域具有至少第一和第二边，并且所述区域被细分为子像素，所述方法包括：构造在沿着第一和第二边的相交之间的线段，所述线段限定区域的第一和第二范围。其中所述相交限于沿着区域的边的离散位置；形成用于表示第一范围的子像素的候选组，所述在候选组中的子像素完全或部分地在第一范围内；确定在所述候选子组的面积覆盖和第一范围的面积覆盖之间的偏差；计算通过组合候选子组与用于其他分段的子像素条形图而形成的角；根据对所述偏差的确定与所述角的计算来在候选子组中选择；在存储器中存储所选择的子组。这个实施例还可以包括重复所述方法来用于限定沿着第一边的可能相交和淘汰冗余的组的一组线段。淘汰冗余组可以考虑到用于表示所述线段组的组的最大数量，或者它可以考虑到使用组来表示线段组的最大可接受误差。在这个实施例上的一个变化是一种预先计算用于对应于像素的子像素的子像素条形图以便限定多边形边的部分的方法，包括：沿着在一个像素对的第一边上的子像素之一的边来提供用于对应于多边形边的直线的的第一相交；沿着与第一边相对的、所述像素对的第二边来提供用于所述直线的可能的第二相交，其中可能的第一和第二相交限于离散位置；提供跨越预定角范围的、用于连接第一可能相交和第二可能相交的线段；选择用于限定由线段限制的范围的子像素条形图，其中子像素图的选择考虑到在子像素条形图的面积覆盖和在范围的面积覆盖之间的差异，并且进一步考虑到通过组合子像素条形图而形成的角的估计。由线段跨越的角的范围可以包括大约 45 度或可以包括大约 90 度。可以提供变换来将子像素图应用到跨越大约 180 度的可能线段方向的范围。

本发明的另一个应用是计算对应于一个角和多边形的第一边与第二边的相交的像素的值的方法，包括：提供存储器，包括子像素的第一阵列和子像素的第二阵列，第一和第二阵列都对应于一个像素；将第一和第二边扩展为第一和第二直线；对应于由第一直线限定的第一范围而将第一阵列的子像素

设置为被填充或空；对应于由第二直线限定的第二范围而将第二阵列的子像素设置为被填充或空；计算第一阵列和第二阵列的相交；超级采样在对应于一个像素的相交中的一组子像素，并且向所述像素分配灰度值。在这个实施例中，设置第一和第二阵列的子像素可以包括应用对应于第一和第二范围的、预先计算的子像素条形图。

一旦我写 SDU 和边位移权利要求，需要写一个 SDU 和边位移权利要求的概述。

附图说明

图 1A-B 是一对辐射吸收图，它们描述了作为降低在边上的曝光辐射的幅度的结果而将一个边移动小于一个像素或栅格图形。

图 2 是提供本发明的一个数据路径实现方面的概览的方框图。

图 3 是在簇协调处理、支持和掩模写入处理和呈现模块之间的分层关系的方框图概览。

图 4A-B 是一对用于图解呈现窗口、保护窗口、标记带、子带和与呈现相关联的其他元件的图。图 4C 是示出在调制器窗口、标记、带、子带和图像之间的内部关系的方框图。图 4D 是图解在全局和本地协调系统之间的关系 的向量图。

图 5A-B 是图解与一个像素相交的一个边的表示的一对子像素网格。

图 6 包括一系列像素网格和对应于像素之一的一个子像素网格。其中图解了向像素网格上映射一对几何图形。

图 7 是超集采样电路的方框图。

图 8-10 是几何形状、微像素高速缓存组和从那些几何形状产生的地址组的示例。

图 11A-D 图解了从一个像素和对应于所述像素的一个子像素阵列的一个角向一侧扩展一个边的四种情况。

图 12 图解了像素的水平和垂直对以及使用微像素条来呈现边的其他特征。

图 13A-D 图解了可以从一组微像素条呈现的不同方向的四个边。

图 14A-D 图解了使用一组微像素条来用于构造从一个角位移的一个边。

图 15A 图解了与呈现和保护带重叠的梯形的特征。图 15B-C 图解了准备

用于限定与像素相交的边的子像素条形图的界面。

图 16A-C 图解了构造在子像素网格中的角和计算对应于所述角的像素灰度值。图 16D 图解了将访问限定器应用到将一个几何图形覆盖在另一个上。

图 17 图解了用于边位移的椭圆结构图形。

图 18 和 19 图解了所谓的填充和滑动位移算法的操作。

图 20 是可以用于实现以多达 5×5 灰度值像素的相邻尺寸的边位移的、逻辑元件的硬件方框图。

图 21 描述了与一个标记重叠的 6 个调制器或呈现窗口。图 22 描述了一个标记和 9 个重叠子区。图 23A 将子区概念扩展到在多遍次中印刷。图 23B 图解了从四个曝光遍次产生的 81 个重叠子区。图 23C-D 是描述在标记、带和子带之间的关系的方框图。图 24A-B 图解了在标记的呈现或调制器窗口内的重叠子区以及对应于一些重叠子区的辐射量简档。

图 25 是一个方框图，其中包括向用于灰度级像素的多遍次补偿照明值的计算应用重叠区和能量变化处理。

图 26 是用于光栅化分割的几何形状的几个处理的方框图。

图 27 是图解镜子传递函数的应用的硬件方框图。

图 28 描述了使用本征矢量方法计算的一对基本函数。

图 29 是图解可能的缓冲配置的方框图。图 30 图解了数据从分割处通过安装进入 SLM 的流程。

图 31 是呈现模块的硬件方框图。图 32 是作为呈现模块的一部分的呈现处理器的硬件方框图。图 33 是作为呈现处理器的一部分的分割转换器的硬件方框图。图 34 是微像素高速缓存产生器的方框图。图 35 是微像素高速缓存缓冲器的方框图。图 36 是帧缓冲器接口和相关部件的方框图。

图 37 图解了在数据被写入存储器控制器之前实现的保护带过滤器的操作。图 38-39 提供了关于帧缓冲器接口的结构的进一步细节。

图 40 是提供关于预处理器核心的一种实现方式的细节的硬件方框图。

图 41 图解了当通过处理器时存储器块的处理。

图 42 是微像素高速缓存缓冲器的硬件方框图。

图 43 图解了在本发明的硬件实现方面的一个实施例中的存储器访问。

图 44 图解了表示一个像素的子像素的对称加权。图 45 图解了表示特定边方向的对称和非对称子像素图。

图 46-47 图解了在检测一个角中使用的 5×5 邻域的两级卷积。

图 48-51 图解了边位移的方面。图 52 是边位移的概览流程图。

图 53 是应用 Bresenham 算法的流程图。

图 54A-C 是多个 SLM 配置的示例。

图 55 图解了使用校正因数来补偿在从 SLM 向工件的投射中的最小缺陷和失真。

具体实施方式

参照附图来进行下面的详细说明。优选实施例的描述是为了说明本发明，而不是限制由权利要求限定的发明范围。本领域内的普通技术人员可以识别对于随后的说明的多种等同的改变。

新的芯片设计要求高精度光刻设备来用于掩模制造和直接写入。许多数据需要从几何域转换到由光刻设备可以使用的格式，以便投射期望的图像。数据转换技术可以与多种光刻投射设备组合。

光刻投射的一种变化形式使用微镜或具有独立像素的类似系统。微镜系统使用阵列，诸如 512×2048 的像素阵列，其中每个像素具有 65 个灰度值。使用 1 kHz 的脉冲速率的系统的这个尺寸要求每秒向微镜阵列装入大约 1 G 个数据像素。诸如 256×256 灰度级像素的较小阵列要求较小但仍然很大的数据吞吐量。诸如每个像素 257 灰度值的较大深度的灰度级要求一定程度上更大的数据吞吐量。一种替代的微镜系统可以使用扫过工件的微镜的较窄阵列，例如 1×512 、 2×512 或 4×512 微镜。

用于光刻投射的另一种技术涉及使用一个或多个扫描光束。或者所述光束可以类似于在电视屏幕上的图像而被系统地扫描以建立光栅图像，或者所述光束可以被向量扫描以建立独立的图形。所述光束可以是激光、电子、离子或粒子光束。更一般而言，可以使用任何辐射光束。按照本发明的方面光栅化的数据可以被游程编码或被压缩以用于扫描光束。

处理概览

光刻设备用于向对投射辐射敏感的表面投射图像。通常，抗蚀剂被辐射曝光。与投射图像正面或负面地相对应，被曝光的抗蚀剂被显影，并且抗蚀剂的区域被去除。图 1A 图解了抗蚀剂曝光的两个模式。独立的峰 101 表示

在均匀间隔的位置上的辐射的分布。对于微镜系统，这些位置可以表示独立的微镜像素元件。对于扫描光束系统，这些位置可以表示光束意欲扫描的网格位置。由抗蚀剂吸收的总能量是覆盖曝光区域的辐射分布的和。曲线 102 表示总的吸收辐射。抗蚀剂通常产生很高对比度的图像。具有大于门限 103 的总吸收辐射的抗蚀剂的区域当被显影时可能硬化，而具有小于所述门限的吸收辐射的抗蚀剂的区域可能在显影后被去除。在抗蚀剂表面建立的图形的宽度对应于沿着门限线 103 从总吸收辐射曲线 102 的一个交点到另一个交点的距离。图 1B 图解了通过降低在图形的一边上的辐射量来调整图形尺寸。在图 1B 中，当在最右位置中的辐射量被降低大约一半 106 时，边 104 被移动距离 105。在微镜系统中，可以调整独立的镜子元件以降低来自单个像素的辐射量。在一个扫描辐射系统中，可以在特定的位置降低扫描辐射的强度，或者在光束在其扫描中到达特定位置之前不久可以取消辐射。

系统架构概览

本发明的图像呈现引擎可以与分割引擎、光栅化引擎和驱动电路相结合地使用。图 2 提供了数据路径的概览。这个数据路径以预处理的几何数据开始来作为输入。预处理的几何数据可以是计算机辅助设计系统的输出。预处理可以降低分层或迭代信息和有效地使得几何表示流变平。数据获取 202 通常包括从辅助存储器获得预处理的几何数据。几何转换 203 是这样的处理：其中几何形状被转换为可呈现的定点几何形状 (RFG)。分割 204 是将几何形状划分为在微镜实现方式中对应于标记和标记的呈现窗口的不同窗口和子窗口的处理。分割引擎的输出是以一种或多种指定记录格式的几何数据。所述记录表示几何图形，诸如多边形和多边形组。将分割的数据表示为梯形是有用的，其中三角和矩形是梯形的子类。梯形的平行边之一可以具有 0 或接近 0 的长度，以便表示三角。分割的数据的另一种有用表示是作为三角或三角链。本发明的大部分方面等同地适合于梯形、矩形、三角或其他多边形或几何形状。多边形角的坐标可以被提供 7 比特或更多的子像素或半子像素分辨率，以便支持像素的 1/64 或 1/128 的精度或更大。可以根据图像投射技术的期望精度和特点来使用更高或更低比特的分辨率。

图像呈现引擎 210 包括多个部件。扩展 211 是在呈现之前扩展几何迭代的处理。可以将分割的几何形状接收作为迭代的 RFG，它具有重复的几何形

状或重复的几何形状组。扩展取消了 RFG 的编组，因此它们可以被独立的处理。呈现 212 是将包括可呈现定点几何形状的多边形转换为光栅化的图像的处理。在多个呈现处理器上执行呈现处理。超级采样 212 是采样微像素分辨率图像和计算灰度级像素值的处理。（在本文中，子像素和微像素一般指的是像素的同一细分。）下面讨论用于超级采样的替代加权方案。边位移 213 是缩放几何形状的处理，例如用于通过激光接近性校正（LPC）或通过光接近性校正（OPC）来补偿接近的和杂散的辐射。图像校正 214 是补偿在光路径、投射系统的级的布置或另一个特性中的非线性和小缺陷的处理。这可以包括非线性图像重新耦合。照明转换 215 考虑到下述因素：诸如在投射的区域之间的重叠、在曝光辐射中的变化和多遍写入。当投射系统使用微镜阵列时，镜子补偿 216 应用预先校准的因子以补偿独立镜子的特性。微镜补偿因子可以用于补偿对电压的不同响应、在工作周期的过程期间的响应的改变、在阵列中的死像素或微镜阵列的类似特征。在需要时和对于所使用的投射系统适当的情况下，可以向呈现引擎 210 增加另外的部件。

驱动电路 220 包括合成 221 和调制 222 处理。合成 221 是将来自几个呈现处理的结果组合到调制所响应的一个或多个数据流中的处理。使用合成器使得可以缩放呈现模块 330 的数量。例如，可以通过修改合成器参数来将呈现模块的数量从 10 提高到 12 而不改变对于调制系统的接口。在一种微镜系统中，可以使用一个数据流来进行调制，以便在使用辐射快速照射微镜阵列之前设置独立的微镜。在另一种微镜系统中，如果微镜用于扫描工件，则数据流的数量可以匹配微镜数量或微镜数量的因子。在传统的扫描系统中，数据流的数量可以匹配所使用的扫描光束的数量，调制 222 是将集中数据转换为投射系统的驱动值的处理。对于微镜系统，数模转换器可以用于产生模拟电压，所述模拟电压被施加到独立微镜元件上。对于扫描系统，驱动信号可以用于控制声光调制器，它调制电子、离子或粒子辐射的辐射光束或等同控制元件。

非线性变换可能要求向每个被再取样的像素应用像素再取样梯度。或者，用于每个像素的梯度可以被卷积核心取样以产生输出像素值。卷积核心的邻域将依赖于梯度的最大允许幅度。一像素梯度可以被 3×3 核心取样；两像素梯度可以被 5×5 核心取样。

投射系统通常也包括扫描 230 和网线板 240。扫描 230 携带图像信息来

通过对辐射曝光的网线板 240 的区域。网线板 240 是投射系统针对其工作的工件。

光栅簇概览

图 3 是提供数据处理的概览的方框图。写入器控制器 310 控制向工件的图像投射。写入器连接器 310 可以通过一个或多个接口连接以与簇协调处理 321 通信。在图 3 中, 描述了 Telnet 接口和写入器到光栅化器的接口 (WRI)。WRI 接口可以包括 TCP/IP 套接字和在工作在写入器控制 310 上的管理控制软件和簇协调处理 321 之间的接口。这个接口将光栅化处理与获取和计算在分割引擎中的数据以及曝光工件的处理同步。通过这个接口, 簇协调处理 321 可以向写入器控制 310 报告写入功能的级的位置和当前状态。簇协调处理 321 是依赖于呈现模块 330 来呈现 RFG 几何形状的一组处理 320 之一。簇级处理 320 支持多个簇 322、324。簇包括簇控制器 323, 它提供支持和掩模写入处理 323。簇还包括一个或多个呈现模块 331。簇控制器 323 可以被设计来支持多达 7 个呈现模块以与现有的 PCI 协议一致, 或者可以被设计用于实现期望吞吐量所需要的和与其他总线协议一致的更多或更少的模块和不同的总线结构。在图 3 中描述的任务的组织具有灵活性和可伸缩性。通过替代的硬件组织可以实现较少灵活性和更高速度的操作。

下面通过进一步说明其上可以实现本发明的各个方法的硬件来补充在簇控制器级的这个概览。

光栅化方法的概览

图 4C 提供了用于解释对于理解本发明的方面有益的术语的基础。SLM (空间光调制器) 在实现本发明的方面的一个实施例中是用于调制用于曝光的激光脉冲的微镜的矩阵阵列。标记 (例如独立的 431A-431E 或集体的 434A-434F) 表示 SLM 的单个曝光。带 (例如 432) 是重叠的标记的水平系列。水平可以或可以不是带的物理表示。水平可以被带的方向定义, 带的方向继而通过工件的 SLM 的扫描运动定义。标记区域包括重叠区。在带中的标记的高度可以相同或不同。标记的宽度也可以各标记彼此不同。在带中的标记的时间顺序最可能是依序的。它可以具有或者从左向右或从右向左的方向或两者。一个图像是带的垂直序列 (例如 432, 432A-432C)。图像的独立

的带可以或可以不具有相同的宽度。图像的带具有垂直重叠区域 433，这类类似于带的相邻标记。下面进一步讨论在标记之间的重叠。标记可以包括来自一个或多个调制器窗口 434A-434F 的数据。或者，调制器窗口可以包括跨越几个标记的整个带 432 或子带（例如 432A）。分区为多个呈现窗口用于实现在光栅化引擎的数据路径中的并行性。带可以被划分为两个或多个子带 432A-432C。子带不必重叠，因为它们被单一标记曝光印刷。在呈现过程中，如后所述，它们的扩展区重叠，因此用于产生子带的输入数据略微重叠。子带的划分可以对应于在标记中的调制器窗口之间的划分。在一个带内的子带从图像的一边到另一边具有相同的宽度，但是不必具有相同的高度。

图 26 提供了光栅化和呈现的进一步概览。它图解了参数文件（2611-2626）、支持软件（2630-2645）、参数安装、呈现模块的功能部件（2662-2668）。使用许多数据表或文件可以提高系统的灵活性。或者，优选地存储在表格中的数据值可以被编码为实现本发明的方面的软件、固件或硬件。硬编码通常降低灵活性，但是提高性能。

在图 26 中，掩模参数文件 2611 可以用于静态参数。可以逐个带地进行处理。处理关键信息可以逐个带不同。一些输入文件可以逐个带、逐个掩模部分和逐个工作地相同。可以通过平台处理器的 CAPS 软件处理来逐个掩模或逐个带地执行在系统中装入的参数的改变。本发明的另一个方面是可以在例如一个带内改变由系统使用的参数，因为写入处理达到工件的角，在此抗蚀剂被烘烤得比在工件的中心更热。

带首标文件 2612、窗口部分 2613 和命令参数 2614 包括对在一个带内的所有标记公共的信息。它描述了将 SLM 划分为多个窗口。影响 RASE 处理的参数是：子带的数量，它表示在 Y 轴上将 SLM 划分为多个调制器窗口。这个参数对应于在系统配置中的模块的数量。子带高度的表，它影响 RP（呈现处理器），所述 RP 使用子带高度来用于呈现地址产生处理和读出处理，子带高度的表也影响像素数据列高度参数。子带 Y 坐标偏移的表，它被几何形状预处理器用来给出独立子带的 Y 偏移。呈现窗口大小 X，它影响几何形状预处理器，所述几何形状预处理器使用 X 窗口尺寸来用于分割复杂几何形状。它也被用于防止 RP 的地址产生器的保护窗口删节。呈现窗口间距 X，它是在呈现窗口之间的距离。X 重叠区域大小可以从在 SLM 大小和呈现窗口间距之间的差来计算。X 间距被几何形状预处理器（PP）用于分割复杂几何图形和

对于提供到 RP 的几何形状进行坐标偏移计算。扩展区尺寸 X 和 Y，它被像素邻域操作（边位移）使用，并且影响在 PP 中的分割。呈现窗口尺寸减去扩展区等于调制器窗口尺寸。这个参数是 AP 的隐含部分，必须总是匹配 AP 的设计。调整参数文件 2621 包括几个数据类型。它一般包括对于在调整处理器中实现的处理的控制信息。这个参数部分包括区域代码的游程编码的位图。独立调制器窗口的区域代码的数量在一种实现方式中限于 128，但是 SLM 的区域代码的总数可以高达 512。每个游程记录具有两个值：重叠区 ID 和区域代码。重叠区 ID 的数量可以限于每个调制器窗口最大 15。文件的照明转换部分包括用于每个重叠区 ID 的一个传递函数表。微镜表部分包括用于每个镜子补偿系数 $[C_n]$ ($n = 1..4$) 的一个标度/偏移输入项 $[C_{n2}/C_{n1}]$ 。AP 参数文件的镜子函数部分包括用于镜子补偿计算的两个函数的函数表、用于每个参数 $C1..C4$ 的一组标度/偏移。镜子校准文件包括图像图，对于 SLM 的每个像素都有一个输入项，每个像素具有四个校准参数 ($C1..C4$)。关于镜子补偿函数的比特宽度和偏移的一般信息被存储在 AP 参数文件的镜子表部分中。这个文件由于大小的原因而是二进制的格式，这与基于文本的多数其他参数文件相反。向这些文件或这些文件的部分分配参数不是本发明的限制而是一个实施例的限制。

校正因子可以被存储在用于校正光路径的静态或系统特征的文件中，或可以实时地产生来用于系统的动态或随机特征，诸如级布置。

CPAS 软件 2630 从文件向呈现模块 2660 提供参数。逻辑块 2631-2645 对应于上述的参数文件。CPAS 软件实时修改在呈现模块中的参数以符合辐射曝光要求。在写入网线板或直接写入芯片期间，可以通过改变在呈现模块 2660 中的参数来处理至少三个抗蚀剂问题。在表面上抗蚀剂的烘烤不是完全均匀的。在许多情况下，表面的边或角比表面的中心烘烤得更快或更彻底。可以设置考虑到在被曝光的工件上的位置的参数以对应于工件的那个部分的烘烤特征。可以使用边位移和照明转换参数来响应于表面烘烤特征。接着，对曝光辐射的抗蚀剂响应依赖于曝光如何积累。即，辐射量对于抗蚀剂没有线性相加效应。许多抗蚀剂由其初始曝光敏化。A 辐射量后随 B 辐射量将产生比单个量 C 更大的响应，其中能量 $A+B=C$ 。通过使得 CAPS 软件在连续印刷遍次中向呈现模块安装适当的参数，可以考虑到抗蚀剂激活。另外，抗蚀剂老化可以影响对曝光辐射的响应。一些抗蚀剂与多遍次曝光网线板或芯片所

需要的时间相比较是短寿命的。随着抗蚀剂老化，它可能变得敏感度降低。可以通过使得 CAPS 软件根据抗蚀剂老化而向呈现模块安装适当的参数来考虑抗蚀剂老化。或者，可以通过在能量扩展补偿因数（ESCF）表中的值来处理这些问题，所述表格从几何形状扩展模块 2662 向照明转换模块 2666 传送。

替代截割的保护窗口

图 4 描述了使用保护区和保护窗口来呈现分割的几何形状的独立图形。在本图中描述的区域包括呈现窗口 401、保护窗口 402 和在呈现和保护窗口之间的保护区 403。呈现窗口是较小寻址窗口，保护窗口是较大寻址窗口。

图 4A 图解了在呈现处理中使用保护区和保护窗口来作为在分割处理中截割几何图形的替代方式。呈现窗口 401 对应于其中可以呈现几何形状以用于未来投射的存储器。呈现窗口 401 被保护窗口 402 围绕。保护区 403 是在呈现窗口 401 外部的保护窗口 402 的一部分。诸如多边形、梯形、矩形、四边形、三角形等的几何形状被接收来用于呈现。所述几何形状可以是凸的。落入呈现窗口 401 内的几何形状 404、即所谓的可呈现定点几何形状（RFG）被呈现到对应的存储器。跨越呈现窗口和保护区的几何形状 405 被部分地呈现到存储器。可以寻址保护带以用于呈现，但是数据不被写入存储器来用于投射图像。在呈现窗口之外的数据被丢弃。保护区可以不对应于存储器，同样呈现窗口不对应于存储器。地址在保护区的数据可以使用过滤器来迅速地处理，因为不需要存储器传送。在分割期间在呈现窗口的边界不截割跨越呈现窗口和保护区的几何形状。如下所述，可以构造这样的硬件，它同时呈现在呈现窗口内外的边。这个呈现可以不区别在呈现窗口内外的边。与仅仅呈现几何形状在呈现窗口内的部分相比较，呈现在呈现窗口外部的边和区域不会明显减慢所述跨越几何形状 405 的呈现。可以通过以两种方式限制几何形状来协助几何形状的呈现。首先，几何形状可以在尺寸上受限，以便没有几何形状可以落入呈现窗口 401 内和保护窗口 402 外。这样的几何形状可以适合于保护区内。第二，可以在呈现之前过滤几何形状，以便完全在呈现窗口 401 之外的几何形状都不对于呈现窗口处理。被呈现到保护区的数据可以在被存储在高速缓存缓冲器之前被过滤或丢弃，如下进一步所述，由此降低存储器带宽负荷。或者，当从微像素高速缓存向对应于呈现窗口的缓冲存储器转换几何形状时，寻址电路可以识别在保护区中的地址。这个寻址电路可以

将几何形状写入截断到呈现窗口。寻址电路可以比较地址信号与呈现窗口的有效地址，并且使用这个结果来控制写使能或其他存储器仲裁信号。或者，在保护区中开始或结束的固定长度填充指令可以被修改以匹配落入窗口内的指令的部分。落入保护窗口内的离散指令可以被地址处理电路忽略。实现保护带的寻址电路或存储器控制器可以与 FIFO（先入先出）存储器组合，如结合实现调整处理器的电路所述，以便实现负载平均化。在呈现过程中，重复选择几何形状的部分，并且产生微像素高速缓存，如下所述。所述微像素高速缓存是对应于那个部分的光栅化表示数据的一个替代集。微像素缓冲器是另一种表示。

呈现和保护窗口思想的一个扩展是扩展区。扩展区包括在呈现窗口 401 的每个边外部的一些像素。扩展区象呈现窗口那样对应于存储器。但是，在扩展区中的像素不用于投射图像。相反，它们建立像素的邻域，它可以用于在呈现窗口的边上卷积一个或多个像素。扩展区的大小应当支持在任何卷积中使用的像素的最大邻域。例如，如果使用 5×5 卷积过滤器或邻域或者两次应用 3×3 卷积过滤器，则扩展区应当为至少两个像素，以便在呈现区的边上的像素在每个方向上被两个像素围绕以建立用于卷积的 5×5 邻域。

呈现窗口配置

图 4B 还图解了被应用到微镜阵列的呈现窗口的几何形状。微镜阵列的两次闪光被描述为 460、470。阵列 460 在这个示例中被划分为四个呈现窗口，包括 451、461。所述阵列可以被水平、垂直或以这两种方式划分为适当数量的呈现窗口。一个 512×2048 阵列可以被划分为例如 10 个呈现窗口，以便向 10 个呈现处理器分布呈现工作。呈现窗口 451、461 被扩展区 452、462 围绕。呈现窗口 451 进一步被保护区 453 围绕。标记带 475 包括多个标记投射。子带 471、481 包括多个呈现窗口和它们的扩展区，所述扩展区扩展通过多个标记。由于子带和扩展区大于呈现窗口，因此用于寻址的原点可以为扩展区 465 的一个角。这个原点可以用于本地寻址、用于特定的呈现窗口及其扩展区或用于在遍及子带或标记带的全局寻址。本地寻址降低了几何边的内插所需要的比特的数量。用于寻址的另一个有益原点可以是保护区 463 的一个角。

图 4D 描述了从全局坐标向本地坐标的转换。在一种输入流中，以相对于当前带的软像素来给出坐标。一个软像素是一个微像素的一半。预处理器

的一个功能是准备用于呈现处理器的坐标，对于一个相对原点，转换所有的坐标。在这个附图中，梯形 404 被包含在呈现窗口 401 内，所述呈现窗口 401 被包含在保护窗口 402 中。可以通过这个附图中所示的向量操作来描述坐标变换。使得

$M = (M_x, M_y)$ = 当前调制器窗口相对带的原点。

$G = (G_x, G_y)$ = 当前保护窗口相对带的原点。

$V = (V_x, V_y)$ = 要变换的几何坐标。

$m = (m_x, m_y)$ = 当前调制器窗口相对保护窗口原点。

$v = (v_x, v_y)$ = 新几何坐标。

则

$$G = M - m$$

$$v = V - G$$

组合这些方程得到：

$$v_x = V_x - M_x + m_x$$

$$v_y = V_y - G_y$$

参数 M_x 、 m_x 和 G_y 可以被存储在寄存器中。 M_x 可以是 9 比特的无符号整数； $M_x = k * \text{SLM_PITCH} * 128$ ，其中 k 是标记编号， $k \geq 0$ 。 m_x 可以是 9 比特的带符号整数。 G_y 可以是 11 比特的带符号整数。注意可以以宏像素来表达参数值。

在一个替代实施例中，呈现窗口可能比微镜阵列更宽，甚至与整个子带一样宽。根据呈现引擎的性能和组织、所需要的吞吐量（例如对于大面积写入器）和被呈现输出的缓冲，一个单个呈现引擎或呈现引擎组可以产生标记的整个子带或带。一个宽的呈现窗口对于使用扫描辐射的投射系统或对于在微镜阵列的连续闪光之间具有实质重叠的基于微镜的系统可能尤其有用。在这个实施例中，对于连续闪光可以读出呈现窗口的部分。

用于增强的分辨率的像素和子像素

图 5A-B 描述了使用微像素或子像素阵列来表示被分配给像素的灰度值。上面，图 1A-B 图解了像素灰度值对于边位置的影响。图 5A-C 图解了表示 65 个灰度级（包括全白）的子像素。同样的原理适用于使用 5、10、17、26、

37、50、82、101、122、145、170、197、226、257 或一般而言 n 的平方加 1 个灰度级。类似的原理适用于使用 4、8、16、32、64、128 或 256 个灰度级。使用子像素的 8×8 网格来表示一个像素的很方便的。其他网格配置可以包括 3×3 子像素、 4×4 子像素、 16×16 子像素或更多。这些子像素可以被称为空的或填充的或称为接通或关闭。因为抗蚀剂可能对曝光具有正面或负面的响应，因此子像素的负面或正面表示是任意的约定。图 5A 图解了穿过像素中央的几何形状的垂直边。像素具有 8 行 501 和 8 列 502 个子像素。几何形状沿着像素的顶边距离 510 处相交 503。在这个示例中，几何形状边沿着像素的底边在相同的距离 511 相交 504。由几何形状边限制的几何区域精确地是像素的一半。这个特定边沿着在子像素之间的边界。根据阴影图约定，在边的左边加有阴影的子像素在由边限定的图形的外部或内部；在边的右边的白色子像素在图形的内部或外部。32 个加有阴影的子像素表示像素的区域的一半。对应于加有阴影的子像素的数量，可以向像素分配灰度值 32。图 5B 图解了几何形状的对角线边。在由 8 中之 4 的距离 530，这个边与上像素边的中间相交。这个边在距离 7 与下像素边 531 相交。几何形状边左边的区域的几何面积是像素的 68.75%。这个几何面积以阶梯样式由 44 个子像素表示，它具有与加有阴影的范围的几何面积精确匹配的面积。

可以通过使得一个几何形状边在子像素边界之间相交来进一步改进子像素的网格表示。方便的是，将一个子像素的边细分为由 2 的幂表示的多个增量，诸如 2、4、8、16、32 或 64 个增量，它们对应于每个子像素的 3、5、9、17、33 或 65 个可能的相交位置——如果子像素的两个边都算数的话。可以预先计算用于表示与在边界之间的子像素边相交的几何形状边的子像素的数量和模式，如下所述。

两级呈现

图 6 使用用于表示像素的灰度值的子像素阵列来图解了在一个像素网格上的两个几何形状。网格 610 包括 8 行 601 和 8 列 602 个像素。两个几何形状 611、610 被从几何域呈现到像素上。网格 620 指示独立的像素是白色 (W)、灰色 (G) 或黑色 (B)。像素 621 是黑色的；像素 622 是灰色的；像素 623 是白色的。两个比特的数据可以用于表示每个像素，00 = B、01 = W、10 = G 和 11 = 备用。像素 624 由子像素的网格 640 表示。在像素网格 620 中的为白

色或灰色的加有阴影的像素 641 在几何形状之一的内部。其他像素 642 在几何形状的外部。网格 630 指示独立像素的灰度值。例如，其中边将像素一分为二的、对应于几何形状 611 左侧边的像素具有 $1/2$ （或 $32/64$ ）的灰度值。在几何形状 611 的斜边上的像素 632 具有 $1/2$ 的灰度值。在几何形状的内部像素 633 具有灰度值 1，表示完全接通。像素 634 具有 $5/8$ 的灰度值，对应于在网格 640 中 40 个阴影子像素。在流水线或其他优化的结构中，独立的遍次可以提供对第一个像素图分辨率和对第二个更高分辨率的微像素高速缓存的访问。

像素的子像素表示可以对应于直接被映射到独立像素的存储器位置，诸如 32 或 64 比特字。或者，可以使用指针结构来将独立的灰度像素映射到存储子像素网格的位置。在任何一种情况下，仅仅需要对灰度像素、而不是对黑色或白色像素更新子像素网格。通过标注对应的像素阵列元素“B”，可以有效地消除子像素阵列，而不用改变独立子像素的值。下次使用子像素网格时，可以不用首先读出数据而向子像素网格写入值。

在一个替代实施例中，可以向子像素分配灰度值而不是二进制值。用于组合像素的一些逻辑操作将被需要替换为加法或减法运算。在子像素中的灰度值可以用于进一步改进分辨率。

具有总计高速缓存的两级

使用两级分辨率的这个方法和装置的一种增强方式是引入第三组存储器位置以总计子像素阵列或网格的灰度值。在诸如流水线结构的一些硬件实现方式中，可以当向子像素网格写入或记录数据时计算子像素网格的灰度值。灰度值可以是空的或被填充的子像素的数量计数，或者它可以是被填充或空的子像素的加权求和。加权对于微镜阵列或激光束扫描都是有益的，因为在投射光束的焦点处投射的辐射的强度大于与中心一段距离的强度。与多数辐射源相关联的辐射强度的高斯分布可以由子像素的加权和来更好地表示。图 44 图解了使用仅仅 10 个加权来表示在子像素的 8×8 阵列中曝光辐射的高斯或类似分布。固有地，高斯或类似分布是对称的，因此适用于像素的四个象限的加权可以取为相同的。对称性还允许分配单个加权或系数来覆盖编号子像素对，诸如标为 2、3、4、6、9 的子像素对。利用这种对称程度，可以通过在施加加权之前在所有四个象限内计数相同加权的 2 个像素来使得计算简

化。处理将从计数进行到应用加权。应当使用定点数值表示来计算加权和加权的和，并且具有双重性。随后加权的和可以在使用前舍入。所分配的加权以及甚至在本发明的一些实施例中使用的用于实现加权的门编程将是现场可编程的，使用诸如 Xilinx 的部分重新配置接口的逻辑。在图 44 中的手段可以良好地扩展到从更圆一些或更大的邻域采样。从更大邻域超级采样可以在正方形中进行，并且将在所述正方形的角中的加权设置为 0 以便有效地产生来自方形邻域的圆形采样。按照本发明，在超级采样上的许多变化是实际和现场可编程的。

使用第三组存储位置的一个替代方式是提高用于表示独立像素的比特的数量，以便可以由像素网格来表示灰度值。

图 7 描述了可以在读出时被应用或当写入到微像素高速缓存时被采用的一种硬件结构，用于总计像素的灰度值。图 7 图解了在 64 比特的微高速缓存中计数通/断微像素的数量，并且具有两个分辨率级。微像素高速缓存阵列 701 的比特被提供到四个计数器或求和电路 702。在一个替代实施例中，这些电路 702 可以向微像素施加加权方案。电路 702 的结果继而被一个或多个加法器 703 组合以产生用于微像素高速缓存的灰度值 713。当从微像素高速缓存读取时，两比特的 MUX 705 可以控制例如整个电路的所选择的输出。两个输入比特 704 来自像素图，它记录是否每个像素是黑色、白色或灰色。对于灰色像素，MUX 705 通过加法器 703 的结果。对于黑色或白色像素，MUX 705 通过对应于所有子像素的通或断的静态值 711、712。在一个替代实施例中，如果像素图 704 指示像素是黑色或白色的，则可以避免从微像素高速缓存获取或向微像素高速缓存写入。这降低了对于访问微像素高速缓存的要求。当向微像素高速缓存写入一个值时，这个电路可以被适配到超级采样。适配涉及使用求和硬件 701-703 和假定写入被执行，因为所述像素是灰色的。超级采样值将被写入到灰度值存储器。

边的并行处理

图 8 图解了在梯形几何图形上的微像素高速缓存产生器的操作。所述微像素高速缓存产生器包括两个 Bresenham 内插单元，它们并行工作在相对边 802、803 上。解释 Bresenham 呈现算法的一个参考文献是 F. S. Hill, Jr., Computer Graphics, ISBN 0-02-354860-6, pp. 428-433 (F. S. Hill, Jr., 计算机图

形学, ISBN 0-02-354860-6, 第 428-433 页)。Bresenham 算法使用整数值, 并且避免相乘。它是增量算法的一个示例, 其根据关于前一个像素的信息来计算在被呈现的直线上的每个像素的位置。Bresenham 是可以用于本发明的多种曲线和直线产生方法之一。在启动期间向两个 Bresenham 内插单元预先安装几何形状的相对边的坐标。在预先安装之后开始内插。内插从梯形的一个平行边到另一个重复。如果使用多边形来取代梯形, 则一般可以在所选择的坐标系统的轴 801 的方向上进行重复。它以几何形状的一个角开始, 并且沿着所述轴进行直到达到相对角。在内插期间, 对于每个内插步骤产生一个或多个微像素高速缓存组。这些高速缓存组表示通过几何形状的部分。所述微像素高速缓存组被存储在微像素高速缓存缓冲器中。在图 8 中描述了 5 个微像素高速缓存组 810-50。对于沿着内插轴 801 的特定坐标, 当两个内插单元都指示内插完成时, 产生地址组并存储在微像素高速缓存缓冲器中。这个地址组与自从组配最后地址组后所产生的独立微像素高速缓存一起形成微像素高速缓存组。在所述地址组中, 在一个高速缓存组 (例如 810) 中的第一边 (例如 811、821、831、841、851) 由地址 X1S 和 X1E 表示。在这种符号表示中, “X” 指示轴的方向, “1” 指示第一边, “S” 表示边的呈现部分的开始像素, “E” 表示边的呈现部分的结束像素。在所述地址组中, 第二边被地址 X2S 和 X2E 表示。在用于第一边和第二边的 X1 和 X2 地址之间的间隔表示可能不包括任何子像素信息的闭合数据的区域。当呈现具有与坐标系统平行的上下边的梯形时, 可以使用单个微像素高速缓存来表示在角 (811、813 和 851、853) 之间沿着所述上或下边 (812、852) 的所有像素。当所呈现的几何形状没有与坐标系统平行的上或下边时, 在边之间没有任何闭合区域。在下边之上和在上边之下, 闭合区域 (822、832、842) 被完全填充, 并且可以不使用微像素高速缓存而隐含地被表示, 或使用上述的两比特编码方案来被表示, 或者为了一致它可以被填充的微像素高速缓存来表示。

图 9 提供了呈现梯形的特殊情况, 其中在梯形中的两个点具有相同或接近相同的位置, 因此有效地将梯形缩为三角形。为了一致, 四个地址点可以被包括在由分割引擎提供的数据流中。在这个附图中再次表示 5 个微像素高速缓存组 910-50。内插轴 901 是垂直的。第一边 (911-951) 也是垂直的。第二边 (913-953) 是三角形的斜边。下面的闭合边 912 位于内插边之间。没有上面的边。微像素高速缓存组 920、930、940 包括完全填充的像素 922、932

和 942。在地址 X2S 和 X2E 之间的距离 903 对应于具有表示斜边的部分的子像素网格的像素的数量。

图 10A 表示另一种特殊情况，其中通过单个微像素高速缓存 1010 来表示几何形状。内插轴是 1001。象图 9 中一样，梯形已经被缩为三角形。左边延伸通过所有三个单元 1011、1012 和 1013。右边被包括在单元 1013 中，具有由 1003 指示的像素宽度。因为间隔 X1S...X1E 和 X2S...X2E 重叠，因此所述重叠表示其中微像素高速缓存以逻辑与运算来相结合的区域。还因为由单个微像素高速缓存组来表示整体几何形状，因此表示三角形底边的微像素高速缓存需要使用逻辑与运算与用于左边的部分的微像素高速缓存组合。

图 10B-D 图解了使用微像素高速缓存组发送的数据的部分。图 10B 描述了具有比顶部窄的底部的梯形。它适配在 7 x 7 像素网格中。通过并行处理这个梯形的上和下以及左和右边而产生的微像素高速缓存一般从上下开始，并且从下向上进行：

左处理器	高速缓存对	右处理器
L6.1	9	R6.7
L5.1	8	R5.7
L4.1	7	R5.6
L4.2	6	R4.6
L3.2	5	R3.6
L2.2	4	R2.6
L2.3	3	R2.5
L1.3	2	R1.5
T6.1	1	B1.3

使用诸如上面列出的顺序的固定顺序来处理边可以提高呈现逻辑的各个块的吞吐量。在一个替代实施例中，可以以不同的顺序来处理边，可以垂直定位几何形状的平行边，或者几何形状可以不是梯形。在这个示例中，上 (T) 和下 (B) 边是梯形的平行边。在第一重复中，使用与左边相交的像素、即顶部在像素 6.1 以及底部在像素 1.3 来对应于上和下边而呈现水平线。连续重复呈现左右边。在第二重复中，在行 1 中的右和左边完全被包括在单个像素中。行 2 需要两次重复、即第三和第四重复来产生微像素高速缓存，因为左和右

边的行 2 部分跨越两个像素。除了高速缓存的对，多边形解释包括产生地址组。所述地址组包括一个行坐标 (Y) 和四个列坐标，X1S、X1E、X2S 和 X2E，其中 S 指示边的开始，并且 E 指示其结尾。在 X1S、X1E 和 X2S、X2E 之间的距离限定每个行的左和右几何边位于何处。在 X1S、X1E 和 X2S、X2E 之间的距离表示闭合数据的区域或者上或下边的闭合部分。对于每行像素产生一个地址组：

Y	X1S	X1E	X2S	X2E
6	1	1	7	7
5	1	1	6	7
4	2	1	6	6
3	2	2	6	6
2	3	2	5	6
1	3	3	5	5

左边的 X1[S]tart 和 X1[E]end 的相对大小指示边当上升时向左倾斜。可以从表格推断通过左右处理器产生像素高速缓存的顺序。

图 10C-D 图解了在特殊情况中的地址组。图 10B 是垂直定位的小矩形。它的小指的是它是一个像素宽或更小。用于 Y 行 5 的地址组被图解。所有四个 X 参数具有相同的值。图 10C 是水平定位的小矩形。仅仅在 Y 行 5 中存在这个矩形的一个地址组。诸如小矩形的小几何形状的识别可以促进处理。

图 11A-D 帮助图解了在一个微像素高速缓存内的像素呈现和使用多边形边的延伸来开始应用 Bresenham 算法。每个图表示用于单个像素的 8 x 8 子像素网格。用于呈现的一个操作是确定边投影与像素的边的相交 (1102A、1102B、1112C 或 1112D)。在附图中提供了涉及梯形的四种情况。在图 11A 和 11B 中的梯形侧边的角大于从水平算起的 45 度。在图 11A 中，xTop 1101A 落到角 (Xm, Ym) 1103A 的右方。在图 11B 中，xTop 1101B 落到角 (Xm, Ym) 1103B 的左方。在图 11C 和 11D 中剩余的两种情况涉及从水平算起小于 45 度的角以及右/左 yTop 1111 到 yBottom 1112 关系。用于计算边投影与像素的边的相交的公式依赖于所图解的四种情况。对于超过 45 度的角度：

情况 A 和 B

$$xBottom = X_m - (dx1/dy) * Y_m$$

$$XTop = xBottom - (dx1/dy) * Micro\ pixelCacheWidth$$

情况 C

$$yBottom = Y_m - (dy/dx1) * X_m$$

$$yTop = yBottom + (dy/dx1) * MPCWidth$$

情况 D

$$Y_m + (dy/dx1) * MPCWidth$$

$$yBottom - (dy/dx1) * MPCWidth$$

使用 Bresenham 算法来呈现像素 (“MiPxCaW”)。使用微像素来呈现由一个边相交的像素。为了找到 MiPxCaW, 可以使用下面的算法:

- 1) 找到左下角所在的 MiPxCaW。
- 2) 找到从那个 MiPxCaW 的左下角到边的延长线与 MiPxCaW 的下边的交点的距离 (xBottom)。这是以作为与 7 个双重 (binal) 的定点坐标的宏像素为单位来提供的。这个距离是:

$$xBottom = X - \frac{dx1}{dy} Y$$

其中 (X, Y) 是从当前的 MiPxCaW 的左下角计算的角的坐标。

- 3) 使用 xBottom, 下一个交点被计算为:

$$xTop = xBottom + \frac{dx1}{dy} MiPxCaWinWidth,$$

- 4) 如果 $0 \leq xTop < MiPxCaWidth$, 则应当恰在当前那个之上的 MiPxCaW 中执行微像素呈现。
- 5) 如果 $xTop > MiPxCaWidth$, 则应当在恰在当前那个之上右侧的 MiPxCaW 中执行微像素呈现。
- 6) 如果 $xTop < 0$, 则应当在恰在当前那个之上左侧的 MiPxCaW 中执行微像素呈现。
- 7) 重复直到达到上一行。

不使用浮点数来表示 (dx1/dy), 可以将它们都乘以 dy 来得到整数。如果边的斜率小, 则可能必须向侧面走几步来获得在下一行上的下一个 MiPxCaW。如果在边和 x 轴之间的角限于大于 45 度, 则不是这样。可以通过将几何形状旋转 90 度来呈现较小的角。在接受大于 45 度的角度情况下用于 Bresenham 算法的代码为:

$$eLeft = dy * xBottom = dy * X - dx1 * Y;$$

$$eLeftShift = dx1 * MiPxCaWinHeight;$$

```
eLeftCond=dy*MiPxCaWinWidth;
```

```
xBottom=eLeft/dy;
```

```
eLeft+=eLeftShift;
```

```
while (eLeft>=eLeftCond){
```

```
    eLeft-=eLeftCond;
```

```
    MiPxCaWNumber+=1;
```

```
}
```

```
while (eLeft<0){
```

```
    eLeft+=eLeftCond;
```

```
    MiPxCaWNumber-=1;
```

```
}
```

```
xTop=eLeft/dy;
```

根据斜率是正的或负的，变量 MiPxCaWNumber 是应当被移到左或右的 MiPxCaW 的连续数目。

用于呈现左边的输入参数如下：

XBottom	边和 MiPxCaW 网格的下交点
XTop	边和 MiPxCaW 网格的上交点
CacheWinLeft	呈现开始的 MiPxCaW 的编号
StartCacheWin	在边右侧的第一个 MiPxCaW 的编号。这是在内部第一个要被填充的。这是一个返回参数。
StartRow	要呈现的最低微像素行。它被用在下边上。
StopRow	要呈现的最高微像素行。它被用在上边上。

在硬件中，有可能预先计算和在表格中存储在微像素级上的呈现。

用于呈现在 MiPxCaW 内的微像素的原理与开始用于找到 MiPxCaW 的原理相同。流程如下：

- 1) 找到 xBottom 的子微像素部分，并且使用它来作为开始值，并且将其称为 xBottomMi。
- 2) 填充通过其边进入 MiPxCaW 的微像素。
- 3) 使用 Bresenham 算法来计算边在下一个微像素线上通过哪个微像

素。一般， $MiPxHeight = \text{以定点单位的微像素的尺寸}$ 。

```
xBottomMi+=(((xTop-xBottom)/MiPxCaHeight)*MiPxHeight;
if(xBottomMi>=MiPxWidth){
    xBottomMi-=MiPxWidth;
miPxNumber++;
}
```

4) 关于 $MiPxCaW$ ，将其乘以 $MiPxCaHeight$ 以获得整数关系，下面给出用于 Bresenham 算法的代码。

```
xMiOffset=subMiPxBin(xBottom)*MiPxCaHeight;
shift=(xTop-xBottom)*MiPxHeight;
cond=MiPxWidth*MiPxCaHeight;
xMiOffset+=shift;
while (xMiOffset>=cond){
    xMiOffset-=cond;
    MiPxNumber++;
}
while (xMiOffset<0){
    xMiOffset+=cond;
    MiPxNumber--;
}
```

图 23 是描述应用 Bresenham 算法的一个实施例的流程图。在 5301 执行初始化步骤。在 5302、5303 和 5306 的测试将开始点划分为对应于图 11A-D 的情况。在方框 5305、5304、5307 和 5308 中执行用于这些情况的代码。在 5311 中设置用于所述算法的定位点。所述算法沿着在 5312-5321 中内插的边而重复。

在图 15A 中进一步描述了图 11A-D 的计算的参数。在这个附图中，在像素的网格 1501 上布置梯形 1500。给出了示例水平像素对 1502 和垂直像素对 1503。所述像素网格包括呈现窗口 1501A 和保护窗口 1520A。距离 $dx1$ 1531 被定义为在左边的下和上角的 x 坐标之间的差。距离 dy 1532 被定义为在梯形 1500 的上和下平行边之间的 y 坐标的差。在呈现窗口周围的扩展区未示出。角在 (X_m, Y_m) 。“ $dx1$ ”是在左边的上和下角之间的 x 坐标差 1531。类似

的量度“dx2”对应于在右边的上和下边之间的距离。“dy”是在梯形的平行上和下边之间的y坐标差1532。对于不同的几何形状，可能需要两个或更多的dy值。MPCWidth是微像素高速缓存的宽度（它等于像素的宽度）。应当选择适当的单位。

方便的是，一次呈现像素对，诸如水平像素对1502或垂直像素对1503。可以总是选择像素对以使得被呈现的边和边的任何投影在像素对的相对侧相交，而不是切除一个像素角。沿着左边1500，由于这个左边的方向而呈现水平像素对。在这个示例中，沿着上边1501和剩余的边呈现像素的垂直对。呈现包括与像素的相对侧相交的边的成对像素使之能够使用下述的表格查找方法。或者，可以使用或不使用像素对来直接应用Bresenham的或另一种内插算法。

均衡像素

用于实现本发明的方面的一种替代方式是使用均衡子像素并且使用或不使用预先计算的子像素条形图，如下所述。在用于呈现在左边的微像素的一种算法中，初始填充通过其被呈现的边与微像素的底部相交的微像素外加在相交的微像素右面（在几何形状内部）的所有微像素。于是，由被填充的微像素覆盖的区域一般与正确区域不同。为了补偿，可以计算这个差别，并且当总的面积误差大于一个微像素时，就增加或减去一个微像素。超过区域初始为0。对于每个MiPx行，它增加

$$A = \left[x_{BottomMi} + \left(x_{BottomMi} + \frac{dx1}{dy} MiPxHeight \right) \right] \frac{1}{2} MiPxWidth。$$

如果累积的超出区域大于一个MiPx，则应当移动边以补偿。对于正的和负的差别都执行这个补偿，并且加上（减去）均衡微像素，以便从不在一个MiPxCaW中偏离理想区域超过一个MiPx。超出区域可以对于每个新MiPxCaW被设置为0，或从微像素高速缓存的一行被携带到下一个。

上述用于计算A的公式基本上与前面的Bresenham算法相同。如果以硬件实现则两种内插流程可以被组合为一个单个流程。

使用预先计算的子像素图

另一种替换方式是使用预先计算的子像素来利用均衡子像素呈现在微像

素高速缓存内的边。图 12 图解了使用垂直像素对来呈现从水平起的小于 45 度的角，并且使用水平像素对来呈现大于 45 度的角。（可以使用像素对的任何一个方向来呈现精确的 45 度角。）垂直对沿着 X1-X2 轴（1211，1212）。水平对沿着 Y1-Y2 轴（1201，1202）。单像素或微像素高速缓存被粗线 1221 约束。单个子像素或微像素被标为 1225。阴影区域 1222 被填充以表示几何形状的内部（或外部）。阴影像素 1223 一行的长度是子像素或微像素条值。微像素条可以被用作连续执行用于呈现子像素高速缓存所需要的计算的替代方式。微像素条表示存储在表格中的预先计算的阴影。

可以使用左手和上几何边来构造微像素条的表格。左手边将对象的内部布置在边的右手侧。上边将对象的内部布置在边的下部。图 13B 和 13D 表示左手和顶部几何边。在传统的 x、y 坐标系统中，可以通过将左下手的角定位在或接近原点并且扫描通过从 90 到 45 度的角的范围来表示可能的角的范围。在图 13 中，通过组合反射、旋转和反转操作，用于图 13D 的微像素条可以被转换为其他图，所述反转如果需要的话被应用来将边的内部和外部保持在它们适当的位置。图 13D 是在 90 和 45 度之间的上部左手几何边。图 13C 是图 13D 通过 y 轴的反射外加反转。图 13B 是在 45 和 0 度之间的较浅的边角。通过 y 轴反射图 13D 然后逆时针旋转 90 度产生图 13B。通过 y 轴的进一步反射外加反转将图 13B 进一步变换为图 13A。在图 13A-13D 的四角布置中，需要两个操作，即一个反射外加旋转或反转来将一组微像素条值转换为方形的相邻角，并且需要四个操作来转换为方形的相对角。因此，可以通过将左下手角定位在原点并且扫描通过从 90 到 45 度的角范围来表示可能角的范围。

本发明的一个有益方面是使用对称子像素条形图。图 45 图解了这个对称性。在左手的子像素条形图对中，图 4501A 和 4502 是彼此的镜子图像。它们紧密适配。它们将一起覆盖微像素阵列的所有子像素，例如产生 64/64 的亮度。在右手的子像素条形图对中，图 4501B 和 4502 具有与 4501A 和 4502 相同的区域覆盖，但是产生了不同的合并结果。因为 4501B 和 4502 不对称，它们不能良好地适配在一起。存在一些重叠区域和其他间隙区域。它们一起覆盖少于所有子像素，例如 61/64。对于在图形中对角布置的几何形状，缺少对称性可以负面地影响系统的临界尺寸的统一和误差预算。

微像素条的表格仅仅包括某些角，因为几何形状边被限制为在特定的间隔相交子像素，如上所述。将沿着像素的边的离散位置使用作为交点是便利

使用具有可管理数量的输入项的表格的一种方式。在给定当前可以获得的存储器和处理器配置的情况下，对于离散位置，我们指的是 513 个位置或更少。进行下列限定是合理的：将子像素和离散位置的精度限制为 256 个子像素和每个子像素不超过 65 个位置、或限制为 64 个子像素和每个子像素不超过 33 个位置、或限制为 32×16 个子像素和沿着子像素的边不超过 17 个离散位置、或限制为 16×8 个子像素和沿着子像素的边不超过 9 个离散位置。在给定现有的存储器、处理器和 FPGA 技术的情况下，这些被认为是可能的替代配置。它们一般涉及 2 的幂，它外推到可以在诸如以后 5 年的不久的未来中可以实现配置。离散位置不是用来指可以使用 16 或 32 比特地址寻址的每个位置。使用离散位置限制了表格输入项的可能数量。当一个像素边具有 8 个子像素并且每个子像素具有 8 个增量时，它具有边相交的 64 或 65 个可能点。从原点，可以使用用于在 90 和 45 度之间的 64 个角（或 65 个角，包括两个末端）的表示来表示可能角的全范围。在 8×8 子像素网格中，可以将有效地跨越一对像素的 8 个微像素条表示为 5 比特值。

在我们的示例中，具有 45 到 90 度角的左手边可以相交在沿着 x 轴的 8 个不同的增量而不落入自原点一个全像素宽度距离范围内。可以选择微像素条的不同图形来表示通过原点的角而不是表示当它通过 x 轴的距离原点半个像素的相同的角。对于同一角需要微像素条的不同图形，以便实现在边布置中的子像素精度。为了说明，考虑 45 度边。如果这个边通过原点，则通过将 $32/64$ 的子像素加阴影来表示它。如果这个 45 度边向右移动一个全像素，则它以每行少于一个像素的比率被少了 8 个的像素来表示。对应于将边布置在沿着子像素边的 8 个不同的位置，在 32 和 25 个加有阴影的像素之间，可以存在在 45 度角上的 8 个变化。图 14A-14D 描述了使用一组微像素条来构造左手边的操作。因为角小于 45 度，因此使用一个垂直对的像素。在图 14A 中，被构造的边 1402 被整数个像素转换为 1401，以便它位于或接近（距离小于一个像素）一个角。在图 14B 中，根据下述增量来选择一组微像素条，以所述增量，被转换的边与在像素对的右和左侧上的子像素的边相交。接着，在图 14C 中，对应于从 1402 到 1401 的转换的一定数量或区域的像素 1412 被从微像素条减去。在图 14D 中，微像素条 1413 被移回靠着像素对顶部的位置，在此它们表示期望的边 1402。

子像素条的预先计算

图 15B-C 描述了用于预先计算被如上所述应用的子像素条的界面。两个附图的特征是相同的，除了所选择的标准和结果产生的微像素条配置。控件“产生 MiPxCaW 呈现表格” 1541 控制来自预先计算处理的输出。当选择此按键时，输出包括一个参数文件，它可以与其他参数文件合并，并且用于实现本发明的方面的系统中。复选框 1542 和 1543 选择如何表达边线段 1555 的交点。在图 15B 中，提供了 x 相交坐标。x1 (1551) 坐标是 20，它对应于在原点右面的一又四分之一的微像素 (20/16)。有效坐标在 1..255 的范围内，它对应于 16 个微像素的每个微像素的 16 个位置。x2 (1552) 坐标是 80，它对应于原点右面的 5 个微像素。如果选择 y 坐标选择 1543，则提供了在微像素显示的 y1 (1553) 和 y2 (1554) 边上的交点的值。提供了三个呈现选项。除非选择“呈现右边”选项 1544，所描述的边将是左边，并且几何形状 1556 的被填充部分在边 1555 的右面。

通过两个或可选择的三个标准来引导均衡像素的分布。对于在呈现时使用流水线运算来计算均衡像素的实现方式，优选前两个标准。首先，由包括均衡像素的微像素覆盖的区域应当与使用三角方法从 MiPiCa 的覆盖部分的角点计算的真正区域最多相差子像素的一半。控制按键 1545 “执行像素均衡”调用这个标准。第二，当通过两个边的逻辑“与”操作形成角时由微像素覆盖的区域应当与对于相交边可以使用三角方法计算的区域不应当相差超过 1 个子像素区域。控制按键 1546 “执行像素区域均衡”调用这个标准。信息窗口 1547 支持对于呈现实现的注解。当选择呈现按键 1548 时产生几个输出。使用所提供的坐标来以三角方法计算真实区域值 1561。在这个示例中它是边线段 1555 右面的区域。近似区域值 1562 是由微像素条 1556 覆盖的微像素的计数。近似区域误差 1563 是在真实区域 1561 和近似区域 1562 之间的差，它在这个示例中是 0。在这个附图中的灰度级值 1564 是冗余的。当系统一次产生在微像素条的整个组时使用最大表误差 1565。它通过指示在用于在所述组中的任何微像素条的真实区域和近似区域之间的最大差而提供了对于程序性能的校验。

确定分布的处理可以是任何种类的在角上的系统迭代、偏移和分布，由此评估对于每个可能组合的误差预算完成情况。按照第一标准来确定所需要的均衡像素的数量。然后，迭代程序尝试不同的可能角组合以找到以最小误

差满足第二标准的子像素配置。在微像素条的边和端之间的最小二方差可以用于测量误差。

在本实施例中的另一种方法是验证当从上部向下部和从下部向上部遍历 MiPiCa 时累积的误差（逐个条地增加误差）在任何点都不超过 1。

一般，选择微像素条组涉及选择偏好较小表尺寸或较紧的误差预算。一种方式是设置期望的表尺寸并找到对处理有最小误差的微像素条组。另一种方式是设置误差预算并产生导致最小可能表尺寸的微像素条组。还有一种手段是产生多组微像素条，并且在其他因素之中，特别根据表尺寸和误差分布的组合来选择一个。

比较图 15B-C 示出了依赖于所应用的标准 1545、1546 的、不同的结果微像素条组。在 1556A 和 1556B 之间的差别在行 0-1 和 2-3 中出现。

预先计算微像素条组的另一个方面可以涉及分层表格，其中包括用于不同角和网格相交偏移的微像素条组。通过使用具有两级的表系统来获得表尺寸的缩小，其中一个表具有用于角和相交偏移对的输入项（0..15，从高速缓存交点坐标（0..127）模 8 确定）。这个表格包括对第二表格的指针，所述第二表包括微像素条长度的组（对 8 x 8 子像素分辨率在每个组中为 8）。这个分层允许几种角/偏移组合共享一个微像素条组。因为条组表是较大的表，因此缩小了总的表尺寸。或者，可以构造支持角和偏移的附加组合的较大表，并且降低或消除了应用条对转换和旋转的需要。

Matlab 是用于找到可以由单个微像素条组表示的全部可能组合的有用工具。针对用于其他角/偏移对的分布而查看满足第一和第二标准的可能均衡像素分布。如果两个分布图形对于不同的角/偏移项匹配——这可以使用 Matlab 的独特函数来识别，则可以存储所述图形的一个拷贝，并且向两个输入项提供指向同一组微像素条的指针。

图 16A-C 图解了在像素或子像素网格的左下手区域中形成角。图 16A 图解了限定边的微像素条的图形。边标称地沿着 x 轴下降 8 中的 4 又 1/2 个增量。在像素的顶部 1601 和底部 1602 的边的交点是等同的。在这个示例中，64 个子像素的 36 个被加阴影。图 16B 图解了定义斜边的宏像素条的图形。在像素 1611 的顶部的边的交点是在像素的中间。在像素 1612 的底部的边的交点沿着 x 轴下降 4 又 1/2 个增量。加有阴影的子像素覆盖 64 个网格位置中的 36 个。在图 16C 中，在图 16A-B 中描述的一对边与前角相交。由浅灰来

指示对应于垂直边 1621 和水平边 1622 的子像素的图形。由逻辑与运算形成的两个像素图形的相交是深灰。在这个示例中的深灰阴影区域与理想值略微不同。

前面的附图图解了实现本发明的方面的替代实施例。本领域内的普通技术人员将认识到，在几何关系中的许多变化可以实现本发明。例如，用于使用辐射扫描来投射图像的器件的网格位置可以是矩形的。在扫描方向中具有子网格地址的网格位置可以被当作矩形像素或子像素。微镜阵列的独立微镜可以是矩形的、六边形的或非凸形的。一个子像素网格可以覆盖多个网格位置、微镜或像素，以适合于用于投射辐射以形成图像的器件的几何形状。从子像素网格总计像素的灰度级值应当对应于子像素到像素的映射。本领域内的普通技术人员将进一步认识到，两级分辨率的使用可以被适配到较高分辨率的子像素网格的灰度分级。当高分辨率的子像素网格包括用于独立子像素的灰度级信息时，用于形成角的边的相交将需要相加子像素的值，并且注意超过子像素的最大值的和，而不是使用逻辑操作。总计子像素网格的灰度级值将需要相加而不是计数阴影子像素。

边位移

图 17 按照本发明的方面来开始边位移的展示。边位移指的是通过将几何形状的边从图形的中心向外位移或向图形的中心内位移来增加或缩小（膨胀或侵蚀）所述几何形状。这与转换几何形状不同，因为几何形状的相对边向相反的方向移动。当边被位移时，理想地，在边之间的角将反映两个边的新相交。在实际的应用中，水平、垂直和 45 度边的位移是最重要的。与几何域相反不同，在呈现域中边位移使得设备操作员可以细调图形尺寸而不改变要印刷的几何文件。使用多个边位移参数来在单个工件上印刷的测试图形序列可以用于校准设备。测试图形的重复实例可以被包括在数据文件中。这个几何形状可以被分割或扩展。不同的边位移参数可以被应用到测试图形的不同实例。可以使用查看工具来从测试图形选择产生期望的图形尺寸的边位移参数组。

图 17 图解了边位移的几何关系。用于边位移的参数包括 cdX 和 cdY 。 cdX 参数指示垂直边应当从几何形状的中心位移多远。正的 cdX 膨胀所述几何形状。负的 cdX 侵蚀它。 cdY 参数指示水平边应当被布置到距离几何形状的中

心多远。图 17 图解了被位移到新边位置 1712 的、与水平成大约 30 度角的旧边 1711。具有等于 cdX 和 cdY 的轴的椭圆 1701 被用作结构工具。新边位置 1712 与旧边 1711 平行，并且与旧边分离由椭圆结构工具限定的距离 1702。新边在与旧边相切的点接触结构工具。相切点的位置是 dX 、 dY 。位移 1703 的方向正交于旧边和新边。

$$\left(\frac{dX}{cdX}\right)^2 + \left(\frac{dY}{cdY}\right)^2 = 1 = z(dX, dY),$$

$$n \propto \nabla z \propto \left(\frac{dX}{cdX^2}, \frac{dY}{cdY^2}\right) \propto (fX, fY),$$

$$dX^2 = fX^2 \frac{cdX^4}{cdX^2 fX^2 + cdY^2 fY^2},$$

$$dY^2 = fY^2 \frac{cdY^4}{cdX^2 fX^2 + cdY^2 fY^2}$$

dX 和 dY 的计算是计算密集的。最小化边位移的计算要求的一种方式是在装入参数 cdX 和 cdY 的同时，向呈现引擎装入 dX 、 dY 、 fX 和 fY 的预先计算值。

边角检测

三种算法和变化方式提供了使用两级分辨率的呈现引擎中实现边位移的情况。所述算法被适配用于水平和垂直边的正交位移、在沿着水平或垂直边的方向上的斜位移、45 度边的正交位移。

卷积过滤器是用于检测边的角或方向的一种工具，以选择要应用的算法。考虑适用于像素的 3×3 邻域的下列卷积过滤器。

$$\begin{array}{ccc} & -1 & 0 & 1 \\ fX = & -2 & 0 & 2 \\ & -1 & 0 & 0 \end{array} \qquad \begin{array}{ccc} & 1 & 2 & 1 \\ fY = & 0 & 0 & 0 \\ & -1 & -2 & -1 \end{array}$$

为了应用这些卷积过滤器，过滤器系数的图形被布置在灰度像素值的图形上。对于每个单元计算过滤器系数和像素灰度值的乘积，并且将单元加在一起。对于每个过滤器计算独立的和。可以使用较大的邻域和使用非整数值来构造较好的卷积过滤器，诸如 Sobel 过滤器，或者使用 2 的平方根的两倍而不是整数 2 的过滤器。依赖于在计算要求和精度之间的大致折中，可以对过滤系数使用宽范围的近似。

边位移的结果

当使用对应于掩模图形的数据来测试 fY/fX 的反正切时，使用上面的过滤器来计算角。结果如下：

角	估计	标准偏差	平均差
90	90.000000	0.000000	0.000000
85	80.324390	11.137572	-4.675610
80	73.371079	12.643785	-6.628921
75	67.821833	12.267936	-7.178167
70	63.054278	10.983100	-6.945722
65	58.970741	9.160636	-6.069259
60	55.165984	7.048248	-4.834016
55	51.650496	4.790010	-3.349504
50	48.287420	2.442252	-1.712580
45	45.000000	0.000000	0.000000

填充算法

在图 18A-B 中描述了用于位移水平或垂直边的算法。每个附图包括 5 个像素 1801-05。图 18A 是在位移前；图 18B 是在位移后。被位移的边近乎垂直 1811、1812。在位移前，所述边与像素 1804 的一个角相交。因此，像素 1804 的灰度值是 58/64。所述边与像素 1803 的稍大角相交，因此像素 1803 的灰度值是 10/64。在这个示例中的边位移参数是左面的 15/64 的子像素。这种所谓的“填充”边位移算法用于与光栅坐标系统的轴之一接近平行的边。它用于将一个边移动 0-1 个像素。对于几何形状的膨胀，所述算法请求将最亮的灰色像素填充为 64 的白值，并且向第一黑色或灰色像素溢出任何多余的位移量。为了侵蚀几何形状，所述算法请求将最黑的灰色像素清空，并且从第一白色或亮灰色像素取得任何多余的位移量。下面的函数是实现这种算法的方式之一：

$$p'[i] = p[i] + \text{sgn}(cdX) \times \max \left(\begin{array}{l} 0, dX - (1 + \text{sgn}(cdX)) + \\ \text{sgn}(cdX)(p[i + \text{sgn}(cdX)\text{sgn}(fX)] + p[i + 2\text{sgn}(cdX)\text{sgn}(fX)]) \end{array} \right)$$

其中， $p'[i]$ 是像素 $p[i]$ 的结果亮度；

sgn 依赖于自变量是否为正或负而是 (1, 0, -1)；

cdX 是上述的输入参数;

dX 使用上面的公式被计算, 并且可以被预先计算;

fX 使用上面的公式被计算, 并且可以被预先计算。

在图 18A-B 的示例中, 期望像素位移 15/64。像素 1804 以 58/64 的值开始。像素 1804 被填充 15 个子像素之 6 的阴影。15 个中的剩余 9 个被加到像素 1803, 将其值从 10 提高到 19。边 1811 被位移到位置 1812, 它是在左面的 15/64 微像素。这种算法用于垂直和水平边的正交位移, 但是对于沿着近乎水平或垂直的边的位移效果不好。

滑动位移算法 – 第一实施例

图 19A-B 图解了滑动位移算法。对于沿着与被位移的边近乎平行的轴的位移, 可以使用下面的函数:

$$p'[i] = \frac{dX}{\max Val} p[i + \text{sgn}(cdX)\text{sgn}(fX)] + \frac{(\max Val - dX)}{\max Val} p[i]$$

其中, $\max Val$ = 被填充像素的最大亮度值 (在这个示例中是 64)。这种算法仅仅依赖于一个相邻像素的灰度值, 所述相邻像素依赖于参数 cdX 和 fX 的符号而在左或右。所述公式实际上计算在两个像素之间的加权平均, 其中加权系数的和为 1。

可以应用不同的加权系数以补偿当应用所述方法时观察到的误差。本图中图解的位移是 32/64 的膨胀。结果产生的加权是当前像素的亮度的一半加上其右手邻域的亮度的一半。边 1911 被向左 1912 位移。像素 1904 被分配像素 1904 和 1905 的平均值。它的值从 48 向 56 改变。像素 1903 保持不变, 因为像素 1903 和 1904 的加权的、预先膨胀的平均值和像素 1902 和 1903 的加权的、预先膨胀的平均值是 48。像素 1901 从 0 向 24 改变, 因为它具有像素 1902 的亮度值的一半。边 1911 通过这个处理向左位移 1/2 个像素。

滑动位移算法 – 第二实施例

用于滑动位移的替代公式是:

$$p'[i] = \max \begin{cases} 0 \\ \min \begin{cases} 64 \\ (1 - \frac{dX}{64})p[i] + \frac{dX}{64} * p[i + \text{sgn}(cdX)\text{sgn}(fX)] \end{cases} \end{cases}$$

其中 $p[i]$ 是中心像素。如果超过 64 个子像素表示单个像素，则相应地调整替代的 $\min$ 值。

通过滑动和填充的位移

图 48-49 中图解了在应用滑动和填充边位移中的变化方式。所使用的手段可以依赖于由梯度过滤器检测的边的角。在每种变化方式中，首先对沿着 x 或 y 轴的邻域的每个中心像素执行滑动操作。然后，对在相反方向中的邻域的中心像素进行填充操作。使用加权系数 1，但是可以修改它以成比例地提高或降低算法对位移参数的响应。

上述的填充和滑动算法的一种变化方式是使用是填充算法来用于所有的侵蚀，引入用于中心像素的加权因子，以便它仅仅与第一侵蚀方向中的其填充因子成比例地被侵蚀以及仅仅与第二侵蚀方向中的其填充因子的平方根成比例地被侵蚀。可以使用滑动算法用于几何形状的所有膨胀。当在 x 和 y 方向中都存在几个灰色像素时，所述填充算法趋向于不填满像素，使得滑动算法有吸引力。

45 度边算法

第三种算法对于 45 度边的位移作用良好。这种算法可以被应用到接近 45 的边。对于边是否足够接近 45 度以应用 45 度算法的两个测试是：

$$\text{abs}(\text{abs}(fX) - \text{abs}(fY)) * 32 < (\text{abs}(fX) + \text{abs}(fY)), \text{ 以及}$$

$$\text{abs}(\text{abs}(fX) - \text{abs}(fY)) * 8 < (\text{abs}(fX) + \text{abs}(fY))$$

其中 abs 是绝对值函数。这些测试被应用到用于检测边方向的梯度过滤器。因子 8 或 32 的选择确定在应用所述算法之前角必须如何接近 45 度。可以容易地使用其他因子或因子范围，包括 16、20 或 24 和 8-32 或 16-24。使用因子 8 对应于大约 45 度 ± 10 度。

如果边是 45 度，则距离它相交的像素的角的距离 D 可以被计算为：

$$D = \begin{cases} \sqrt{\frac{F}{\max Val}}, 0 \leq F \leq \max Val / 2 \\ \sqrt{2} - \sqrt{1 - \frac{F}{\max Val}}, \max Val / 2 \leq F \leq \max Val \end{cases}$$

其中 F 是阴影子像素的数量（例如，当 45 度边将像素平分为 2 时， $\max Val = 64$ 的 32）。一般，45 度边被位移的距离可以被计算为：

$$cdH = \frac{fX^2 cdX^2 \operatorname{sgn}(cdX) + fY^2 cdY^2 \operatorname{sgn}(cdY)}{\sqrt{(fX^2 + fY^2)(fX^2 cdX^2 + fY^2 cdY^2)}}$$

这个位移当然在与图 17 中的单位向量 $n(1703)$ 相同的方向中。如果 $cdH > 0$ 并且像素为空，或 $cdH < 0$ 并且像素为满，则我们必须查看对角邻域以看它们是否是灰色的，并且按照上述方程计算与角的距离。然后通过加或减 $\sqrt{2}$ 而获得当前的与角的距离，所述 $\sqrt{2}$ 是单个像素的对角距离。从角到边的新距离因此是：

$$D = D_{\text{邻居}} \pm \sqrt{2}$$

为了计算 45 度边的新填充因子，我们计算与像素角的距离为 $D + cdH$ 。则新的填充因子是：

$$\frac{F}{\max Val} = \begin{cases} (D + cdH)^2, & 0 \leq D + cdH \leq 1/\sqrt{2} \\ 1 - (\sqrt{2} - D - cdH)^2, & 1/\sqrt{2} \leq D + cdH \leq 1 \end{cases}$$

当然，存在用于 F 的 $\max(\max Val)$ 和 $\min(0)$ 。为了避免舍入误差，我们从中心像素和两个相邻像素取平均值 D 。

用于 45 度边位移的第二实施例类似于第一实施例。这种 45 度算法在 128×128 像素方块中变换被分类为 45 度边的灰度级虚拟边。所述边被从光栅域变换到几何域。在几何域中执行边位移，然后将边位移重新变换为灰度级填充值。定义 $GetDist$ 函数以将灰度级值转换为几何值。函数 $GetFill$ 将几何值转换为灰度级值。 $GetDist$ 将灰度级填充值转换为几何距离值：

填充 ≤ 0	距离 = 0
填充 $< \frac{64}{2}$	距离 = $\sqrt{\frac{\text{填充}}{64}} * 128$
填充 $> \frac{64}{2}$	距离 = $\left(\sqrt{2} - \sqrt{\frac{64 - \text{填充}}{64}} \right) * 128$
填充 ≥ 64	距离 = $128\sqrt{2}$

$GetFill$ 将距离值转换为灰度级填充值：

距离 ≤ 0	填充 = 0
距离 $< \frac{128}{\sqrt{2}}$	填充 = $\left(\frac{\text{距离}}{128} \right)^2 * 64$
距离 $> \frac{128}{\sqrt{2}}$	填充 = $\left(1 - \left(\sqrt{2} - \frac{\text{距离}}{128} \right)^2 \right) * 64$
距离 $\geq 128\sqrt{2}$	填充 = 64

所述 45 度算法使用环绕的 3x3 像素来计算中心像素位移。在 3x3 矩阵中的像素位置被表达为 $p[x][y]$ ，其中 $p[0][0]$ 是中心像素。所述算法的输入参数是 $sgnfX$ 、 $sgnfY$ 和 cdH 。

```

if (cdH>0) {
    if(p[0][0]<=64/2) {
        p[0][0]=a
    }
    elseif(p[0][0]>64/2) {
        p[0][0]=b
    }
    elseif(p[sgnfX][sgnfY]>0) {
        if(p[sgnfX][sgnfY]>=64/2) {
            p[0][0]=b
        }
        if(p[sgnfX][sgnfY]<64/2) {
            p[0][0]=c
        }
    }
}
elseif(cdH<0) {
    if(p[0][0]<64) {
        p[0][0]=e
    }
    elseif(p[-sgnfX][-sgnfY]<64) {
        p[0][0]=d
    }
}
}

```

在这个代码段中，($sgnfX, sgnfY \in \{1, -1\}$)。a 到 e 的值是：

$$a) = GF\left(\frac{GD(p[0][0]) + GD(p[sgnfX][0]) - \frac{128}{\sqrt{2}} + GD(p[0][sgnfY]) - \frac{128}{\sqrt{2}}}{3} + cdH\right)$$

$$b) = GF\left(\frac{GD(p[0][0]) + GD(p[-sgnfX][0]) + \frac{128}{\sqrt{2}} + GD(p[0][-sgnfY]) + \frac{128}{\sqrt{2}}}{3} + cdH\right)$$

$$c) = \text{GetFill}(\text{GetDist}(p[\text{sgn } fX][\text{sgn } fY]) - 128\sqrt{2} + cdH)$$

$$d) = \text{GetFill}(\text{GetDist}(p[-\text{sgn } fX][-\text{sgn } fY]) + 128\sqrt{2} + cdH)$$

$$e) = \text{GetFill}(\text{GetDist}(p[0][0]) + cdH)$$

其中 GF 是 GetFill 的缩写，并且 GD 是用于 GetDist 的缩写。

角检测和处理

边位移的一个附加、可选用的方面可以是角的特殊处理。所谓的 Forstner-Plessey-Harris 算法是用于检测角的一个基础。参见 VIGRA Computer Vision Library, template <...> void cornerResponseFunction at <http://kogs-www.information.uni-hamburg.de/~koethe/vigra/doc/cornerResponseFunction.html>; C. G. Harris and M. J. Stevens: "A Combined Corner and Edge Detector", Proc. of 4th Alvey Vision Conference, ed. by C. J. Taylor, pp. 147-151 (Manchester University, 31st August-2nd September 1988) (VIGRA 计算机视觉库, 在 <http://kogs-www.information.uni-hamburg.de/~koethe/vigra/doc/cornerResponseFunction.html> 的 template <...> void cornerResponseFunction; C. G. Harris 和 M. J. Stevens: "组合角和边检测器", C. J. Taylor 编辑的、第四次 Alvey 视觉会议会刊, 第 147-151 页 (Manchester 大学, 1988 年 8 月 31 日-9 月 2 日))。所述算法如下: 在给定的标度 s, 它计算如下的在每个像素的结构张量 (它是梯度乘积的平滑矩阵):

$$\begin{pmatrix} G^s(g_x^s g_x^s) & G^s(g_x^s g_y^s) \\ G^s(g_y^s g_x^s) & G^s(g_y^s g_y^s) \end{pmatrix} = \begin{pmatrix} A & C \\ C & B \end{pmatrix}$$

其中 G 表示在标度 s 的高斯平均值, g 和 g 是第一高斯导数, 并且相乘是逐个像素的。因此, 角响应可以被定义为:

$$CR = AB - C^2 - 0.0625(A+B)^2.$$

可以在设置门限后使用角响应 CR 来识别角像素。

在实际中, 我们使用 fX 和 fY 来估计 g 和下述形式的高斯过滤器:

$$\text{Gaussian} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

注意这个操作需要 5 x 5 的邻域。CR 的值接近角为高。门限用于选择角像素 (ALG2_PAR_HARRIS_LOW < CR < ALG2_PAR_HARRIS_HIGH)。或者, 角

检测可以使用 5×5 像素窗口来确定是否中心像素是目标角。用于一般边检测的两个 3×3 梯度过滤器被应用到在窗口中的每个可能位置，如图 46 所示。结果是具有 f_x 和 f_y 值的 3×3 矩阵。在每个位置，结果被求平方并且彼此相乘，产生每个位置三个值，如图 47 所示。对于每个值，上述的高斯过滤器被应用到图 47 中的矩阵。角响应 (CR) 被计算和与用于角检测的门限相比较。

$$CR = G(f_x^2) * G(f_y^2) - G(f_x * f_y)^2 - 0.0625 * (G(f_x^2) + G(f_y^2))^2$$

$$TBD < CR < TBD$$

边和角检测的另一种手段是最小单值段吸收核心 (SUSAN)，它已经被广泛地说明和包括在下述文献中：S. M. Smith, Method For Digitally Processing Images To Determine The Position Of Edges And/Or Corners Therein For Guidance Of Unmanned Vehicle, UK Patent 2272285 (15, January 1997) (S. M. Smith, 数字处理图像以确定其中的边和/或角的位置来用于引导无人汽车的方法，英国专利 2272285 (1997 年 1 月 15 日)); S. M. Smith, A New Class Of Corner Finder, in Proc. 3rd British Machine Vision Conference, pages 139-148(1992) (S. M. Smith, 新类型的角探测器，第三次英国机器视觉会议，第 139-148 页 (1992))。边位移的利用独立于所选择的特定边和角检测算法，除了更好的角和边检测方法提供膨胀或侵蚀几何图形的更好的基础。

角检测的另一种替代方式是在呈现过程期间记录角的建立。通过使用逻辑与运算相交两个边来建立角。在呈现多边形或组合多边形中的逻辑与运算的应用指示可能的角。上述的两比特像素编码保留了比特值“11”。这个值可以用于标记角操作而构造的灰色像素，这与几何图形的简单边相反。当连续的几何图形完全覆盖一个像素、将其亮度值驱至 maxVal 并且使得所述像素被标记为 W/B 时，将自动去除这个标记。当两个邻接的梯形建立先前两个角所在的一个连续边时，可能去除这个标记。所述标记可以被作为指示用于特殊情况处理的角或指示用于进一步测试的角。

边位移的流程图

图 52 是用于边位移的硬件实现的流程图。在这个实施例中，一旦计算完 dX 、 dY 和 cdH 值就计算位移计算，使得角检测算法可以与位移计算并行被赋值。在检测和判决树中， dX 、 dY 和 cdH 值的表格可以从软件预装，并且当 cdX 和 cdY 值改变时被重装。在所述附图中，输入的像素数据被缓冲 5201。

对应于所选择的邻域的大小，数据被提供为 5 行像素数据。数据送入延迟缓冲器 5211 和边卷积计算器 5212 两者。卷积的结果用于通过从表格 5221 提取一些数据来计算 dX、dY 和 cdH 5222。卷积结果也用于确定是否已经检测了角、45 度边或其它边 5223。延迟缓冲器 5211 使得 5 行像素数据和所计算的值 5222 组合以用于在并行处理中计算角位移 5231、45 度边位移 5232 和其它边位移 5233。复用器 5241 或其它逻辑选择器件使用检测结果 5223 来选择所计算的位移之一以应用或不选择。

边位移的结果

执行使用或不使用角检测而应用边位移算法的测试。测试图形是不同的。方形图形的 LR 或右下边类似于以 45 度削去右下角而剩下五边图形的方形。SQ 或方形图形包括一系列方形，其中原点沿着 x 和 y 轴移动半个像素。SRQot 或旋转的方形图形使用与 SQ 图形相同的方形，仅仅在 x 上移动并且在 y 上旋转 0-90 度。PlusRot 图形是被当作 SQRot 的一系列加号，具有 x 上的移动和旋转。这个图形将内角引入到在其他情况下仅仅具有外角的一组图形。用于每个测试图形的参数 cdX 和 cdY 不同，并且误差的范围在阴影子像素的数量与理想情况的差别上特征化。

在没有角检测的情况下，对于上述多个图形测试所述三边位移算法，并且得到不同的结果。对于 LR 文件，得到下面所示的统计数字，这仅仅用于通过不包括角的测试图形的那些行。因此，这些结果示出了纯边的算法行为。

(所包括的行是 $27+i+j*32$ ，其中 $i, j=0, 1, 2\dots$)。

CD X,Y	最小	最大	差别数量	平均	绝对平均	平方根
+16+16	-4	3	2480	-0.80	1.42	1.59
+32+32	-6	3	2528	-0.76	1.54	1.76
+64+64	-6	4	2384	-0.85	1.64	1.90
+128+128	-7	6	2264	-0.77	1.97	2.37
-16-16	-3	3	2376	-0.12	1.25	1.34
-32-32	-4	4	2136	-0.14	1.40	1.56
-64-64	-4	4	1768	-0.27	1.52	1.73
-128-128	-7	6	1808	-0.14	2.00	2.44
+32+96	-5	3	2440	-1.01	1.62	1.85

+32+96	-7	6	2392	-0.78	1.88	2.34
-32-96	-4	3	1816	0.04	1.53	1.69
-96-32	-7	6	2000	-0.25	1.88	2.29

对于 SQ 图形，达到下面的结果。因为在 SQ 中的边是 0 或 90 度，因此边角对于与理想图像的差别像素的数量没有影响。因此，在这个数据组中的平均表示角上的误差。

CD_X,Y	最小	最大	差别数量	平均	绝对平均	平方根
+16+16	-29	0	3418	-3.58	3.58	5.56
+32+32	-35	0	3708	-5.99	5.99	9.27
+64+64	-42	0	4396	-10.15	10.15	15.67
+128+128	-59	0	5872	-15.02	15.02	21.67
+32+96	-46	2	4306	-9.29	9.30	13.91
+96+32	-46	2	4330	-9.03	9.03	13.55
-16-16	-5	14	2172	-0.42	1.69	2.15
-32-32	-8	24	2084	-0.76	2.37	3.36
-64-64	-9	32	1736	-1.24	2.78	4.69
-128-128	-6	32	796	2.12	4.58	7.60
-32-96	-11	32	1604	-0.61	2.96	5.39
-96-32	-11	32	1604	-0.62	2.98	5.40

SQRot 图形的测试需要被重复，因为原始数据组包括非各向同性的 ED 的误差。下面的数据来自校正后的图像。因为边不是 0 或 90 度，因此在检测边方向中的误差影响差别像素的数量。因此，平均不代表角误差。

CD_X,Y	最小	最大	差别数量	平均	绝对平均	平方根
+16+16	-26	14	23971	-1.50	1.89	2.52
+32+32	-34	4	25265	-1.85	2.29	3.61
+64+64	-52	4	25316	-2.76	3.22	5.98
+128+128	-59	6	26286	-4.22	4.76	9.31
+32+96	-52	21	24888	-2.90	3.40	6.87
+96+32	-49	10	25347	-2.77	3.26	6.59
-16-16	-23	15	19867	-0.31	1.30	1.49

-32-32	-48	25	18432	-0.25	1.53	2.08
-64-64	-40	33	16749	-0.19	1.75	2.55
-128-128	-34	36	16501	0.23	2.21	3.14
-32-96	-25	39	18186	-0.09	1.87	2.93
-96-32	-51	35	18025	-0.09	1.86	2.93

与 SQRot 图形一样, PlusRot 图形的测试需要被重复, 因为原始数据组包括非各向同性的 ED 的误差。下面的数据来自校正后的图像。因为边不是 0 或 90 度, 因此在检测边方向中的误差影响差别像素的数量。因此, 平均不代表角误差。

CD_X,Y	最小	最大	差别数量	平均	绝对平均	平方根
+16+16	-40	20	24909	-1.83	2.29	3.18
+32+32	-44	20	25956	-2.56	3.06	4.75
+64+64	-56	11	25974	-4.13	4.61	8.00
+128+128	-60	9	27473	-7.07	7.53	13.02
+32+96	-51	25	26481	-4.56	5.12	9.60
+96+32	-52	10	26549	-4.24	4.81	8.94
-16-16	-38	33	21821	0.28	1.69	2.29
-32-32	-31	25	20642	0.75	2.33	3.63
-64-64	-45	43	19022	1.88	3.49	6.46
-128-128	-34	57	19093	4.29	5.76	10.48
-32-96	-32	49	19849	2.20	3.88	7.46
-96-32	-54	50	19703	2.24	3.95	7.49

对于上述的角检测 (非替代的 SUSAN 手段), 测试显示出一些改善。对于 LR 文件, 下面示出的统计数字仅仅基于那些不包括角的行。因此, 统计反映出纯边的行为。(所包括的行是 $27+i*j*32$, 其中 $i, j=0, 1, 2, \dots$ 。)

CD_X,Y	最小	最大	差别数量	平均	绝对平均	平方根
+16+16	-4	3	2112	-0.68	1.31	1.44
+32+32	-6	3	2112	-0.63	1.35	1.53
+64+64	-5	4	2136	-0.63	1.49	1.71
+128+128	-6	5	1960	-0.57	1.51	1.76

-16-16	-3	2	1920	0.02	1.16	1.22
-32-32	-4	4	1752	-0.01	1.25	1.38
-64-64	-4	4	1624	-0.14	1.50	1.70
-128-128	-6	4	1712	0.18	1.70	1.98
+32+96	-4	2	2056	-0.97	1.38	1.51
+32+96	-7	6	2112	-0.66	1.55	1.86
-32-96	-3	3	1544	0.32	1.36	1.49
-96-32	-6	5	1864	-0.06	1.59	1.87

对于 SQ 图形，在 SQ 中的边是 0 或 90 度，因此边角不影响差别像素的数量。因此，在应用角检测的情况下，所述平均表示在角的误差。

CD_X,Y	最小	最大	差别数量	平均	绝对平均	平方根
+16+16	-21	0	3474	-3.87	3.87	5.28
+32+32	-26	0	3916	-6.16	6.16	8.68
+64+64	-39	0	4828	-9.59	9.59	14.06
+128+128	-57	0	5748	-14.37	14.37	20.55
+32+96	-46	0	4684	-8.50	8.50	12.54
+96+32	-46	0	4748	-8.26	8.26	12.16
-16-16	-2	5	1202	1.47	1.85	2.10
-32-32	-3	9	1228	2.64	3.18	3.80
-64-64	-2	9	1288	2.97	3.64	4.24
-128-128	-1	0	252	-1.00	1.00	1.00
-32-96	-4	12	1218	2.30	3.19	4.07
-96-32	-4	12	1222	2.49	3.39	4.26

使用角检测的 SQRot 图形的结果是：

CD_X,Y	最小	最大	差别数量	平均	绝对平均	平方根
+16+16	-24	14	22797	-1.62	1.96	2.71
+32+32	-34	6	24128	-2.03	2.40	3.90
+64+64	-52	4	24768	-2.94	3.34	6.08
+128+128	-57	6	24807	-4.04	4.42	8.96
+32+96	-52	7	24237	-3.02	3.36	6.87

+96+32	-49	6	24676	-2.99	3.30	6.65
-16-16	-23	9	16820	-0.15	1.28	1.47
-32-32	-41	11	15835	-0.08	1.50	2.02
-64-64	-30	11	15774	-0.20	1.85	2.50
-128-128	-40	14	16095	-0.73	2.46	4.40
-32-96	-25	12	17184	-0.02	1.72	2.26
-96-32	-44	16	17125	-0.02	1.79	2.58

使用角检测的 PlusRot 图形的结果是:

CD_X,Y	最小	最大	差别数量	平均	绝对平均	平方根
+16+16	-42	20	25472	-1.73	2.63	3.57
+32+32	-44	19	26769	-2.42	3.68	5.53
+64+64	-56	26	27733	-3.73	5.26	8.47
+128+128	-60	30	26449	-6.23	7.22	12.60
+32+96	-51	19	27881	-4.12	5.25	9.37
+96+32	-52	18	28193	-3.92	5.05	8.86
-16-16	-37	31	19209	0.73	1.83	2.55
-32-32	-25	25	18648	1.40	2.62	4.10
-64-64	-41	45	19277	2.01	3.83	6.52
-128-128	-38	55	18717	2.36	6.13	10.74
-32-96	-32	49	20100	2.13	3.77	7.03
-96-32	-45	50	20133	2.14	3.81	6.87

边位移的一种硬件实现方式被图解在图 20 中, 并且被讨论如下。

硬件实现

图 20 描述了像素行延迟缓冲器, 包括例如 FIFO 和在 5 x 5 邻域上的像素延迟运算器。替代的实施例将获益于: 与传统的图像处理算法一致, 使用在卷积核心中的对称性来降低延迟元件的数量。在线 2001 上输入像素数据。延迟元件 2010、2020、2030、2040 和 2050 控制数据沿着线 2011、2012、2022、2032、2042 和 2052 传播到用于边位移的像素邻域运算器 2060。修改或位移的像素邻域被计算和输出 2070。调整处理器对在来自呈现处理器的输出数据

流中所接收的像素邻域上执行灰度分级的边位移。在一个实施例中，用于边位移的邻域尺寸是 5×5 灰度级像素，但是邻域大小是实现的细节，而不是对实现本发明的方面的方法的应用的限定。边位移的实现方式有利于并行性以容纳并行像素输入和输出数据流（每个周期 2 或 4 个像素）。输出数据图像的符号表示 $[\text{Inc } Y_0-2; \text{Inc } X_0-2]$ 指的是存在一个两像素线延迟和一个两像素延迟。

照明补偿

图 21 描述了所谓的标记重叠。呈现或调制器窗口可以覆盖一个或多个标记的部分。例如，标记 2101 被六个调制器窗口 2111-16 覆盖。在单遍次中，标记在它们的边缘 2102 附近重叠。重叠区 2103 是重叠区域。重叠分析也可以被应用到扫描辐射图像投影器，其中网线板区域被印刷，然后与该网线板的随后印刷区域重叠。如果从一条边向对边扫描网线板，则扫描系统的重叠图形可以包括在扫描带的所有侧上或仅仅在两侧上的重叠区。

用于重叠和能量变化补偿的重叠区

图 22 描述了在单遍次环境中的微镜标记的重叠子区。标记 2201 具有由与 8 个相邻标记 2211 的重叠而产生的 9 个重叠子区 2220，标记 2201 在 3×3 标记网格的中心。中心子区不与相邻标记重叠。沿着“+”图形的臂从中心子区向呈现区的边向外投射，存在与相邻标记曝光具有单次重叠的四个子区。在呈现区的四个角的每个中，存在与相邻曝光具有多次重叠的子区。对于由其它呈现子区的曝光围绕的矩形呈现区，角多次重叠子区的每个可以被曝光四次。这种 9 个子区配置可以被适配到多个曝光。四个交错的曝光可能产生呈现区的 81 个子区，如图 23B 所示。相同的子区划分可以容易地被应用到其它形状的呈现和保护区，诸如六边形或替代的三角形呈现和保护区。照明补偿是用于补偿在多遍次中曝光的区域之间的重叠的方法。这种方法以变化方式适用于微镜和扫描辐射束系统。多遍次使得在单个标记的角内的照明区的数量变为多倍。在图 23A 中，图解了标准两个遍次，当前遍次 2301 和来自另一个遍次的重叠标记 2311-14。区域代码 2320-26 在由多遍次曝光建立的许多重叠子区中。一种多遍次图形包括四个（移位）层。每个层由重叠的 SLM 图像组成，如图 23B 所示。最内的方形示出了通常的“标记”的几何图形。

每个这样的标记具有 81 个区域，其中每个是 4-7 个不同曝光脉冲的结果。区域代码算法的目的是将最后的剂量变化缩小到（大约）脉冲能量的变化的四分之一，并且提供剂量补偿来用于在重叠区中的缝合。在所述缝合区中，我们可以选择使用内插函数（以斜坡的形式）或在整个缝合区上的恒定值。

重叠区的一种使用是记录从标称曝光脉冲产生的理论曝光。图 24A-B 图解了用于标记 2301 的各种照明区域的照明剂量特征。在这个示例中，标记由 n 个调制器窗口 2302 覆盖，其中每个与标记一样宽。每个调制器窗口被分配 6 个照明子区。不与在单遍次中的其它曝光重叠的中心子区具有相对较高的剂量特征，如在图 24B 中的子区 0;4 指示。诸如 0;1 和 0;5 的单个重叠子区具有中间剂量特征。诸如 0;2 的多个重叠子区具有低的剂量特征。

重叠区的另一种使用是记录在闪光强度中的变化。曝光脉冲提供了在能量中的非确定变化或扩展。所述扩展相对于标称能量被表示为能量扩展。这尤其适合于激态原子脉冲激光器和其它可能在能量输出中具有大的脉冲到脉冲的变化的来源。如果由辐射光束提供的能量随着时间变化，例如当辐射源加热时，它也可以被应用到扫描辐射。照明区可以将关于来自重叠辐射的理想曝光的信息与关于在曝光特定工件期间产生的实际曝光能量的信息相结合。能量扩展补偿是通过调整在后续曝光遍次中的 SLM 调制值来补偿所述能量扩展的处理。在调制调整处理中对于照明值进行所述补偿。与标称调制相关联的补偿值是所谓的能量扩展补偿因子 (ESCF)。ESCF 表是表示用于一个特定调制器窗口的 ESCF 代码的表格，并且每个区域代码一个 ESCF 代码。ECT 阵列是一组 ESCF 表格，其中整个子带或带的每个调制器窗口一个表格。能量扩展补偿计算是产生 ESCF 表阵列的处理。这个处理可以在 Mercury 计算机集群或用于分割处理的其它主机中被执行，如下所述，或在模拟 Mercury 计算机集群实现方式的离线处理中被执行。

能量扩展补偿发生在整体光栅化引擎 (RASE) 的调整处理器块的照明转换处理中（图 31 中的 3141）。对于在标记中的每个调制器窗口，存在一组区域代码，每个区域代码表示一个窗口段。对于在带中的每个闪光脉冲，使用对于每个区域代码和对于每个脉冲独立的能量扩展补偿因子来补偿标记照明值。另外，通过装入用于在标记中的多个区域的不同的因子集，在一个标记中的所有区域可以被附加因子补偿。当补偿抗蚀剂和其它系统因子时这是所期望的，所述其它系统因子诸如角烘烤、启动和老化效应。所述因子可以以

下列方式提供二进制 2 补码定点编码:

$$f=(F+512)/512$$

其中 F 是二进制 ESCF 值 (-128..+127), f 是相对补偿因子 (0.7500..1.2481)。零 F 值表示 100%, 它表示“无补偿”。

再次地, 表示用于特定闪光脉冲的区域代码的一组 ESCF 被称为 ESCF 表 (ECT)。ESCF 表格可以包括用于沿 32 比特双字中排列的、每个位置的一字节输入项。ESCF 的长度可以被固定在 128, 而与所利用的区域代码的数量无关。

	比特 31..24	比特 23..16	比特 15..8	比特 7..0
dword 0	用于 AC 1 的 ESCF	用于 AC 2 的 ESCF	用于 AC 3 的 ESCF	用于 AC 4 的 ESCF
dword 1	用于 AC 5 的 ESCF	用于 AC 6 的 ESCF	用于 AC 7 的 ESCF	用于 AC 8 的 ESCF
	...	...	...	...
dword 31	用于 AC 125 的 ESCF	用于 AC 126 的 ESCF	用于 AC 127 的 ESCF	用于 AC 128 的 ESCF

对于要被写入的每个带, 需要一个 ECT 的阵列 (每个激光脉冲和每个调制器窗口一个), 称为 ECT 带阵列。所述阵列被划分为子带, 如图 23C 所示。产生 ECT 带阵列的处理被划分为子带, 因此对于每个 GLI 信道 (对应于一个子带), 产生 ECT 子带阵列来作为在分割引擎计算机集群中的独立结构, 如图 23D 所示。当在数据文件中表示 ECT 带时, 诸如当离线产生 ECT 带时, 在文件 1,1 到 1,n, 然后文件 2,1 到 2,n 等等中排序子带。

返回图 23A, 对于在特定调制器窗口中的每个区域代码, 存在一组能量测量源参考。在所述附图中, 标记 2301 是当前标记的曝光区域。标记 2311-14 是前一个遍次的四个重叠标记。调制器窗口 2302 以点虚线给出轮廓。在调制器窗口 (i) 中是各种区域代码 (在这个示例中的 i;0 到 i;8, 2320-2328)。因为这是两遍次示例, 因此在调制器窗口内存在 9 个区域代码。

在一个实施例中, 每个能量源测量参考被表示为在参数文件中的、具有两个指针和加权因子的记录。如果多个闪光影响曝光, 则所述指针指出 (多个) 源测量值, 加权因子表示来自在重叠区中的独立测量的影响。

从下面的表达式中计算 ESCF:

$$ESCF = \frac{1}{\sum_{j=1}^{j=m} (E(x + dx_j; y + dy_j) \cdot w_j)}$$

其中 x 是标记号，y 是带号，w_j 是能量加权因子，dx_j 和 dy_j 是相对标记坐标，m 是用于区域代码的源参考的数量。变量 dx_j 和 dy_j 是在从划分区域代码得到的范围(-1..+1)中的值，所述区域代码在多遍次曝光标记位置偏移之后。w_j 变量具有定点表示，它具有[0..32768]的整数间隔，其中 0 表示 0%的贡献，32768 表示 100%贡献。E 值也是具有范围[0..32768]的定点表示值，其中 32768 表示标称能量（100%），并且>32768 的值表示。

在下面的示例中，给出用于标记的源参考的完全集。扫描顺序是首先上升 X，然后上升 Y。在重叠区中，第一曝光标记具有 0,55 的加权，第二曝光标记具有 0,45 的加权。同一加权应用到垂直和水平重叠。当四个标记重叠时，按曝光的依次顺序，加权是(0,275;0,265;0,240;0,220)。能量测量已经得到了下面的结果，它们说明一个实施例。

E(0;0) E(1;0) E(0;1) E(1;1)
0,970 1,000 1,000 1,030

区域	相对 X	相对 Y	加权	加权	ESCF	ESCF 整数值
代码	指针	指针	实际值	整数值	实际值	
(1;0)	0	0	1,000	32768	1,031	+16
(1;1)	0	0	0,550	19661	1,017	+9
	+1	0	0,450	18022		
(1;2)	+1	0	1,000	32768	1,000	0
(1;3)	0	0	0,550	19661	1,017	+9
	0	+1	0,450	18022		
(1;4)	0	0	0,275	13107	1,002	+1
	1	0	0,265	9830		
	0	1	0,240	8847		
	1	1	0,220	8192		
(1;5)	+1	0	0,600	19661	0,987	-7
	+1	+1	0,400	18022		
(1;6)	0	1	1,000	32768	1,000	0

(1;7)	0	1	0,550	19661	0,987	-7
	+1	1	0,450	18022		
(1;8)	1	1	1,000	32768	0,971	-15

重叠区还可以考虑在理想和实际曝光中的、可以从图 1 看出的、在照明或曝光和边位置之间的非线性关系。

用于通过多遍次曝光补偿手段而获得的曝光能量变化校正的理论限制如下。假定以产生四次曝光辐射，并且最大能量在其误差规格内。如果 n 表示已经在同一点上发生了多少次曝光，则我们可以写出用于在 n 次后的能量的下列递归。

$$E_n = (n - E_{n-1})(1 + \delta) + E_{n-1}, \quad E_0 = 0;$$

上述表达式的解释如下。在了解在 $n-1$ 次闪光后的能量和 E_{n-1} 的情况下，我们将区域代码 (AC) 补偿因子设置为 $(n - E_{n-1})$ ，以便获得在 n 次激光闪光后的总剂量， n_s (标准剂量)。因为我们仅仅将激光控制为精度 $(1 + \delta)$ ，因此通过前一个给出的 (递归) 表达式来给出结果产生的剂量。

我们在下表中总结这个递归公式：

激光脉冲能量精度	$4 * (E_n - 4)$
10%	11.1%
5%	5.26%
2%	2.04%
1%	1.01%

在最后一列中的因子 4 被包括来突出以下事实：使用 4 遍次写入的改进是几乎四倍，但是不完全是。在此使用的表达式仅仅适用于与同一层中的其它区域不重叠的区域。

有益的区域代码补偿因子可以被计算如下。下面的符号表示如下：

a = 区域代码。范围(0,0)到(8,8)

l = 层。范围 1..4

s = 标记代码。范围(0,0)到.. (N,M) ，其中 $N \times M * 4 = 10^7$

ρ = 曝光补偿因子 (依赖于区域代码)，范围 $\approx 0.25-1$ (依赖于激光的脉冲能量稳定性)

$E(l,s)$ = 用于标记 s 和层 l 的所记录激光脉冲能量。

以与前一部分相同的方式，我们得到曝光补偿因子如下。对于每个区域（区域代码），先前进入那个区域代码（包括补偿因子）的总的激光脉冲能量和 K 倍的补偿因子的和应当等于 $1E_0$ 。 E_0 是标称激光脉冲能量。因子 K 是我们将在一个层内写入所述区域代码 a 的次数，并且依赖于 a 。因此：

$$K\rho_{a,l,s} + \sum \rho_{a',l',s'} E_{l',s'} = 1E_0$$

（注意在一些情况下 E_0 可以被设置为 1）。这再次是我们需要对于每个标记求解的一个递归方程。所述递归遍布在当前层中已经被写入的标记（和区域代码），并且随后遍布在下面的多个层中的标记。

实现方式

图 25 是用于照明转换的一个处理流程的方框图。这是照明转换的完全版本；部分实现方式也具有实用性，例如当仅仅使用一个遍次时，使用扫描辐射源或可以忽略在曝光能量中的脉冲到脉冲或遍次到遍次的变化。照明转换机制的任务是产生具有从输入的灰度级值（表示虚拟网格）计算的期望照明的、像素数据的图像。在虚拟网格和照明之间的关系是非线性的，这可以从图 25 看出。所述处理包括通过改变灰度级到照明的转换表来补偿重叠。这用于补偿在光致抗蚀剂中的非线性能量累积特性。所述处理也考虑了用于补偿在多遍次曝光的在前遍次中的闪光能量中的扩展或变化所需要的照明的补偿。

区域代码部分数据库 2511 包括一个或多个区域代码图像图和一个重叠区查找表。区域代码图包括用于在调制器或呈现窗口中的像素的输入项。所述输入项限定区域代码。在一个实施例中，所述区域代码的范围是 0..127。在单遍次的实施例中，可能使用少达 9 个的区域代码。区域代码的数量被适当地适配到从所有曝光遍次、重叠图形和特殊情况产生的照明子区的数量。区域代码图的处理可以考虑特殊情况，诸如在呈现窗口的端点的调制器窗口，这与在呈现窗口中央的调制器窗口、跨越呈现窗口的调制器窗口和用于在工件的边上应用的标记的调制器窗口不同。不同的区域代码图像图可以用于处理不同的情况，或者不同的索引可以用于对综合区域代码图的行或列给出地址。包括表面烘烤特征、抗蚀剂激活和抗蚀剂老化的抗蚀剂问题可能产生特殊适配的区域代码图像图。不管区域代码图像图如何存储，对于调制器或呈现窗口的特定像素或较大区域都可以查找区域代码。

区域代码 2522 可以作为重叠区查找表 2523 的索引。对于具有范围 0..127 的区域代码，重叠区域查找表将具有 128 个输入项。在重叠区查找表中的输入项可以是具有诸如 0..14 的范围的重叠区 ID (OZ ID) (2533)。可以根据照明表 2524 的期望数量来使用更宽或更窄范围的 OZ ID。OZ ID 用于驱动照明表选择器 2535。或者，可以直接用 OZ ID 2533 装入区域代码图像图，以便用于在调制器或呈现窗口中的像素的图像图输入项直接访问 OZ ID，而不用使用区域代码 2522 来访问查找表 2523 的中间步骤。

照明部分数据库 2512 包括用于可允许的灰度级值和用于有效 OZ ID 的表格，在这种情况下为 0..64 x 0..14 个输入项。对每个可允许灰度级值，将数据装入照明表 2524。所述数据提供了依赖于照明重叠区 ID 的、用于实现期望灰度级值的值范围（例如 0..14）。一个输入的灰度级值 2513 调用特定的照明表 2524，它使得可以获得一定范围的照明值 2534。照明表选择器 2535 使用 OZ ID 值 2533 来选择可应用的照明值。在所述照明表中的值可以考虑在来自多个曝光的剂量累积中的非线性，并且可以考虑在照明级中的非线性，所述照明级是产生可以被表示为全亮像素的相等划分的辐射剂量所需要的。可以通过将特殊适配的照明表 2524 存储在数据库 2512 中来处理抗蚀剂表面烘烤特征、抗蚀剂激活和抗蚀剂老化的问题。可以使用诸如 0..820 的宽范围的照明值 2544 来表达辐射剂量，所述辐射剂量是实现多种曝光重叠条件下的相对较小范围（例如 0..65）的灰度值的所需要的。

可以本身或与重叠转换组合而可能具有实用性的照明转换的另一个方面是能量变化补偿。当工件的部分在多遍次中向辐射曝光时，可以累积能量变化补偿因子的数据库 2541。例如，使用脉冲激态原子激光辐射源的微镜系统可以在能量输出中经历脉冲到脉冲的变化。这些变化可以被监控并记录。累积在诸如理想的 75-125% 的宽范围中的能量变化因子可以是有益的，所述范围足够宽以捕获多遍次中的累积变化。对于照明微镜系统的辐射源的每个脉冲，可以在对应于脉冲的标记位置的较大图内在逐个区域代码的基础上累积能量变化。然后，来自区域代码图像图 2521 的区域代码 2522 可以用于索引能量变化补偿因子表 2532。诸如 75-125% (384..639/512) 的补偿因子 2542 可以用于补偿来自理想能量曝光的可观察的变化。这个补偿因子 2542 可以 2545 乘以照明值 2544 来产生具有诸如 0..1023 的范围的、多遍次补偿照明值 2546。如果辐射剂量从一个遍次到下一个遍次缩放，例如在两遍次印刷处理

中缩放 80/20 或在四遍次处理中缩放 40/30/20/10，则这种能量变化补偿可能特别有益。多遍次缩放具有下述应用可能：通过向较小最大剂量应用从低到高的相同定标来改善特定动态范围的精度（例如 10 的能量剂量的 1024 个层级而不是 40 的能量剂量的）。

照明函数不仅校正抗蚀剂非线性，而且在基于 SLM 的部分相干成像的图形产生器中具有基本光学基础。理论研究已经显示，在最简单情况下，所述部分相干使得边位移成为灰度级的非线性函数，并且非线性依赖于照明 SLM 的光的空间分布，所谓照明系统的西格马值。从灰度值到边位移的 dx 的传递函数通常是在 g 和 $\sqrt{g}$ 之间的函数，其中 g 是在范围 0.00-1.00 中的灰度级。所述灰度级对于包括具有被设置为相同偏转的许多像素的大区域获得的灰度级。当多遍次（4 或更多）印刷时，照明函数的精确形状较不重要，可以以良好的精度来使用基于传递函数 $dx = g^{0.75}$ 的单个函数。

镜子补偿处理

镜子补偿函数将在照明转换函数中计算的照明值转换为镜子电压值。补偿函数使用来自存储在补偿系数图中的镜子校准文件的数据来计算 SLM 镜子的电压值。所述补偿系数图对在调制器窗口中的每个镜子包括一个输入项，具有四个系数（ $C1..C4$ ）。所述补偿系数 $C1..C4$ 每个经历一个定标/偏移机制，其中包括用 $Cn2$ 的二进制移位运算和偏移值 $Cn1$ 的相加。这些定标/偏移常数对于在调制器窗口中的所有镜子是公共的，并且从 AP 参数文件的镜子表格部分装入。从下面的用于镜子驱动电压的方程产生作为照明值 x 的函数的输出电压 U ：

$$C_{S1} = C_{11} + C_1 \cdot 2^{C_{12}}$$

$$C_{S2} = \text{round}((C_{21} + C_2 \cdot 2^{C_{22}}) \cdot x / 128)$$

$$C_{S3} = \text{round}((C_{31} + C_3 \cdot 2^{C_{32}}) \cdot F_3(x) / 128)$$

$$C_{S4} = \text{round}((C_{41} + C_4 \cdot 2^{C_{42}}) \cdot F_4(x) / 128)$$

$$U(x) = \text{round}(C_{S1} + C_{S2} + C_{S3} + C_{S4} / 128)$$

在一个实施例中，用于这些方程的参数范围是：

参数	范围
$C1..C4$	0..255
$C_{11}, C_{21}, C_{31}, C_{41}$	-4096..4095

C12,C22,C32,C42 0..7

补偿函数使用由查找表定义的两个原始函数。这些查找表从 AP 参数文件 2621 的镜子函数部分 2625 装入。在图 27 中示出了这些方程的一种硬件实现方式。

在图 27 中的硬件是上述方程的一种有效实现方式。系数 C1..C4 被存储在补偿系数图 2701 中,所述补偿系数图 2701 可以被存储在镜子校准文件 2626 中。对于每个像素可以存储多组系数,以便考虑镜子的老化、在镜子休止周期之间的时间、在镜子休止周期之间的镜子循环的数量或随着时间改变的独立微镜的其它特征。期望的照明值(x) 2702 是输入。计算块 2711-2713 实现了第一方程 $C_{s1}=C11+C1*2^{C12}$ 。系数 C1 被装入移位寄存器 2711 中。指数 C12 用于启动寄存器 2711 的移位功能,以较快地完成相乘和求幂。加法器 2712 将移位寄存器 2711 的结果和系数 C11 求和。溢出检测器 2713 响应于溢出范围的值。所述响应可以是错误或将结果设置为预定值,诸如数据路径的最大可允许值。计算块 2721-2723、2731-2733 和 2741-2743 类似地操作以部分地计算 C_{s2} 、 C_{s3} 和 C_{s4} 。计算这些值需要将块结果分别乘以期望的照明值(x)、乘以 $F3(x)$ 和乘以 $F4(x)$ 的乘法。乘法器 2753-2755 实现这些乘法。下移位和舍入逻辑单元 2761-2764 将所述结果定标为期望范围,其中保留了在计算初始阶段中的计算精度。逻辑单元 2761-2764 也实现除以 128,这可以被实现为移位寄存器操作。加法器 2765 组合结果 C_{s1} 、 C_{s2} 、 C_{s3} 和 C_{s4} 。溢出检测器 2766 类似于 2713 而校验结果,并且下移位和舍入逻辑单元 2768,定标、舍入和除以 128,以适当地产生在期望动态范围内的 $U(x)$ 。依赖于用于镜子补偿的函数和所使用的处理器的类型,不同的硬件逻辑将产生等同于在图 27 中的硬件逻辑单元的结果,其总是使用预先存储的系数和函数和数字逻辑来将期望的照明值转换为期望的调制或电压驱动器,以便产生驱动调制器或微镜的数字值。

校准程序的目标是:在给出 M (其中 M 可以是 10^6) 个微镜的情况下,找到对于所有镜子公共的反射系数范围和可以用于所有微镜的补偿例程中的有限(小)组的函数(优选的是两个或三个以方便存储)。即,在给出作为用于每个镜子的电压的函数的反射系数的情况下,找到可以由所有微镜获得的最大和最小(公共)反射系数。在那些值之间,对于所有镜子良好地定义倒数,即作为反射系数的函数的电压。除了可以被存储在表格中的一小组静态

函数之外，还使用用于每个镜子的 32 比特存储来找到那些倒数的适当表达式。对于具有公共白和黑级（至少由 CCD 看来）的阵列中的微镜，整个阵列的动态反射系数范围将由所有微镜可以获得的级来限定，它可以是多数微镜的动态范围的一半或更少。依赖于所需要的动态范围，无缺陷的一些微镜可以被排除使用或特殊地被对待，因为它们具有有限的动态范围。当镜子被特殊对待时，它们仍然被使用，并且在黑或白中具有较大的补偿误差。一旦选择了“黑”和“白”级，则在那个反射系数范围内校准独立的镜子。校准程序是上述引用的同时的专利申请的主题。

定义这些函数的一种手段是使用容易被傅立叶展开表示的函数，诸如基本函数 $\sin(\pi x)$ 和 $\sin(2\pi x)$ 。当使用这些基本函数时，可以期望类似于 $\sin(3\pi x)$ 的特征误差形式。得到所述函数的第二种方法是线性内插和使用特征函数的内插的混合。特征函数（实质上）是用于使用线性内插的内插误差的协方差矩阵的本征函数。用于 M 个镜子的协方差矩阵的直接计算需要 $M \times M$ 矩阵的对角化。或者，可以向 N 个正弦傅立叶分量投射 M 个函数，并且取代而计算 $N \times N$ 协方差矩阵的协方差。流程如下：1）在在端点内插反射函数和响应函数的直线之间的差别被扩展足够大量的傅立叶分量（例如 60 个正弦函数）；2）具有 M 个镜子（因此具有 M 个函数）并且扩展为 N 个分量向我们提供了具有 $N \times M$ 大小的矩阵 A ；3）现在通过选择具有最大本征值的（方阵） AA^t （ t 表示转置矩阵）的两个本征向量来选择基本函数。以这种方式获得的基本函数仍然是正弦类的，并且拟合数据而没有 $\sin(3\pi x)$ 形式的系统误差。图 28 的函数 2801-2802 图解了一对使用在傅立叶展开中的 60 个正弦项而产生的一对基本函数。这些基本函数是正弦类的，没有当如上所述使用其它基本函数时预期的系统误差。

用于上述方程的可用校准流程中存在傅立叶方法和最小二乘拟合。通过傅立叶方法，可以通过积分来得到系数，

$$c = \int_0^1 \sin(\pi x) e(x) dx = \sum_{n=1}^N w_n e(b_n) \sin(2\pi b_n) \sqrt{1-b_n}$$

其中 $e(x)$ 是在在端点内插数据的直线之间的差别， w_n 和 b_n 是来自求积法的加权和横坐标。也类似地对 $\sin(2\pi x)$ 积分。这个积分容易通过 Gauss-Chebyshev 求积分法来求解。在间隔中具有少达四个点的求积分法可以产生满意的结果。

在最小乘方拟合和傅立叶方法之间的差别是前者在端点是（设计上）精

确的，同时最小乘方拟合最小化了误差的偏差（至少当加权函数等于 1 时）。通过求解下式来得到校准系数 c_1 到 c_4 ：

$$Ac = Y$$

其中 A 是 4×4 矩阵，并且 Y 是 4×1 向量。矩阵的元素是

$$A_{ij} = \sum_m w(x_m) f_i(x_m) f_j(x_m)$$

并且

$$Y_i = \sum_m w(x_m) y_m f_i(x_m)$$

其中 Y 是在一些（规格化的）反射系数样值 x_m 上的电压。

所述函数的两个（ f_1 和 f_2 ）是常数函数和线性函数 $f(x)=x$ 。所使用的剩余两个是从 $\sin(x)$ 和（实质上的）本征函数推出的那些。如果将加权函数 $w(x)$ 被选择为 1，则我们将获得最小化偏差的校准系数（ c ）。如果我们也基本函数之中的两个选择为 $\sin(\pi x)$ 和 $\sin(2\pi x)$ ，则我们将获得很类似于傅立叶展开的解。在这两者之间的差别仅仅产生自下述要求：在傅立叶情况中，常数和线性函数用于精确地（在端点）内插入校准数据，而最小二算法自由地选择它们。结果，最小乘方拟合产生最小的平均误差，但是不保证在端点是精确的。

用于补偿的算法是很简单的，即

$$U(x) = c_1 + c_2 x + c_3 f_3(x) + c_4 f_4(x)$$

对于每个镜子存在四个特有系数和对于所有镜子共同的两个支持函数（ $f_3(x)$ 和 $f_4(x)$ ）。这两个函数可以被置入表格中来作为校准流程的结果。参数 x 在这种情况下被规格化到范围（0..1），虽然在最后的实现中可以使用其它的规格化方式。

有益的是，将镜子补偿系数所需要的存储量和带宽最小化到例如每个镜子 32 个比特。这将在补偿中引入“舍入”误差。例如，考虑被分别舍入为 9、9、7 和 7 比特的系数。舍入的进行首先通过下式将每组数字变换到范围 0..1 中：

$$X' = \frac{x - \min(x)}{\max(x) - \min(x)}, \quad (X' \text{ 属于 (闭合) 范围 } [0..1])$$

然后通过下式被删截到 N 比特的精度：

$$x_b = \text{Round}(X' 2^N) / 2^N (\max(x) - \min(x)) + \min(x)$$

函数“ROUND”仅仅舍入为最接近的整数。“ $1/2^N$ ”将结果移位回 $[0..1]$ 范围，并且最后的相乘恢复现在在 N 比特精度的原始校准参数。剩余的计算以浮点

(64 比特, IEEE) 精度进行。对于在此提供的方法, 存储用于补偿分量的 9, 9, 7, 7 (7 比特用于乘以表函数的常数) 比特不总是最佳选择。如果改变基本函数, 则另一种存储方式可能变得最佳。但是模拟显示: 以这种方式选择存储方式将产生充分精确的结果。

在 32 比特存储方式中, 可能也需要存储 7 比特的区域代码。测试显示存储分别具有 7, 7, 5, 6 比特精度的系数为区域代码流出空间, 仍然适合每个微镜 32 比特的存储。

替代的系数定标影响例如在图 27 中所需要的计算。通过乘以范围 (最大值-最小值) 来从存储值恢复系数可能显示计算太昂贵。如果所述范围被替换为超过所述范围的 2 的最接近倍数, 则可以通过简单的移位操作来恢复系数, 并且一定程度上牺牲精度。恢复系数的两种替代方式是: 1) 如前一样, 乘以定标值以恢复从最大值到最小值的所述范围; 2) 如上一样, 但是不包括独立微镜的最小和最大值。(这种手段的变化包括仅仅排除最小或仅仅最大值, 或排除独立微镜的多个最小或最大值。) 排除外围值具有以下效果: 值中的两个可能不落入范围 $0 \dots 1-2^{-n}$ 中。如果这样, 则被排除的值被分别存储为 0 和 2^{-n} 。第二流程可以向被排除的微镜 (例如, 在 1000 个微镜中的两个中) 的补偿误差引入实质的误差, 同时可能更有效地存储剩余的系数。下面的表格表示各种方法的模拟误差。第一种替代标度方法被称为 “all”, 并且第二种被称为 “loose2”。

方法	无舍入	最佳	All	Loose2
标准偏差	0.13%	0.23%	0.32%	0.25%

如表格中所示, “loose2” 平均起来几乎与 “最佳” 一样精确, 但是一些微镜具有达到 5% 的补偿误差, 而 “all” 以平均误差为代价将所有微镜的补偿误差保持为小于 1.5%。

可以根据系数 c1..c4 的分布来对于它们选择不同的定标方法。例如, 在模拟中, 多数时间下, c1 被正态分布, 因此排除最极端校准系数的 0.2% 不会大大改变范围。在同一模拟中, c4 具有一些极值, 并且排除它们看起来改善了剩余值的存储效率。

多 SLM 配置

可以使用大量的像素和高刷新率来制造 SLM 以便每秒打印许多像素。然

而，在同一系统中使用多个 SLM 是有益的。图 54A-C 示出了用于多个 SLM 的三种不同配置。在图 54 中，圆圈描述了在投影光学的工件上的光场。矩形对应于来自多个 SLM 的投射。它们可以或者被安装在诸如 PC 板或多芯片模块的单个支持件上，或者它们可以在不同的支持件上，并且通过镜子、棱镜、全息元件等将它们的光投射组合到投射透镜的场中。在所述附图中的平行细垂直线描述由独立的 SLM 印刷的条纹。在图 54A 中，投影被例如半透明镜子或其它分光镜或光束组合器光缝合在一起，以便它们建立在图像之间具有小重叠的连续区域。如上所述的重叠区的使用将在 SLM 图像之间的接缝结合在一起。在图 54B 中，在两个方向中分离图像。这种配置适合于将所有 SLM 安装到单个支持件上的实施例，因为在 SLM 之间的空间提供了用于驱动电子学器件和互连器件的空间。在图 54B 的配置中，SLM 在沿着带的不同位置，并且考虑到偏移而缓冲或产生数据。在图 54C 中，SLM 是印刷部分重叠带，因此在一个物理遍次中建立两个偏移遍次。在附图 54A-C 中的配置被以四个 SLM 的宽度的带偏移（即在写入路径之间的侧向运动）印刷，并且区域被印刷一次。在图 54C 中，所述偏移仅仅是一个 SLM 宽度，但是区域被印刷两次。

使用几个 SLM 的复杂性是精确的对齐以降低小图形误差和不可见的边界。图 55 示出了与由该级定义的理想几何网格 5501 相关联的两个 SLM 5503 和 5504 的图像。所述图示出了在 SLM 的像素和理想网格之间的放大对准误差。与理想网格相关联地提供要打印的图形的输入数据 5502。要由顶部 SLM 5503 打印的图形元素 5502 的部分相对于 SLM 的本地坐标系失准。考虑到这个失准而光栅化图形。对于一个或多个 SLM，一组失准参数——在这个情况下为一个旋转 5507 和两个转换 5505 和 5506——在印刷之前作为要应用到校正变换 2665 中的校正因子 2635 存储。

无法使用一个旋转和两个转换的简单方式来说明 SLM 图像 5504，因为它是失真的。在一个替代实施例中，通过每个 SLM 的失真和失准的图来实现在失准的表示中的较高精度。这对于图 54A-C 中的配置具有大价值，因为光场越大就越难建立无失真投影系统。

每个 SLM 图形的失准和/或失真的特征在于向在级上的参考点上投影部分图像和测量相对于参考点的位置。这通过自动校准程序来进行，所述自动校准程序测量 SLM 失准和/或失真，并且建立 SLM 失准数据组。

可以以三种不同的方式来按照 SLM 失准数据组进行数据变换。首先，可以在分割期间向数据应用校正。当数据被切割到呈现窗口时，涡系(vortex)的坐标值被校正失准，或当适用时被校正失真。可以进行变换的第二位置在光栅化步骤中，其中在建立位图期间或就在建立位图之前可以修改元素的坐标。第三种可能是最好在应用照明函数之前使用转换核心来卷积位图。参见图 26，2635。下面是在 x 方向中的小位移 dx 的转换核心。

$$\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 1-dx & dx \\ 0 & 0 & 0 \end{array}$$

对于小位移、一个像素的分割，图像质量的损失是可以忽略的。而且，卷积核心卷积只具有两个非 0 系数，并且可以容易地以硬件被实现。对于其它转换方向，核心也是简单的，并且将图像转换为在任何方向中的一个像素的一般硬件实现方式是简单和快速的。同一校正方法也中可以包括其它效果，诸如用于级位置误差和图像失真的校正。在图 26 的方框 2665 进行对于位图的校正，其可能包括透镜失真、多 SLM 对齐误差、级位置误差，并且在方框 2635 中存储校正因子或参数。在多个 SLM 的情况下，可能在方框 2635 中存在几组参数，并且方框 2635 包括对于失真控制的校正图。这些校正可以被应用到单一 SLM；它们不限于多 SLM 配置。

使用几个 SLM 的另一个方面是向数据路径增加另一级并行性的机会。在 55A 中的 SLM 可以被分割和光栅化为连续区域，但是每个 SLM 也可以独立于其它 SLM 地分割和/或光栅化。其中，每个 SLM 具有其本身的完整数据路径，并且在数据路径之间的最小通信可以保证它们同步。使用一种广泛并行计算方案，即使输入数据信道分离，系统也变得可以真正扩展到很高容量。使用分割、光栅化和 SLM 的完整写入通道可以被组合以建立可能需要的任何吞吐量。SLM 可以被集成到光投射系统中，并且保持电子自主。它们的组合能力与 SLM 的数量成比例。

有益硬件配置概览

图 29-42 是描述可以用于实现本发明的方面的、各种详细级别的方框图。图 30 是分割硬件 3001A-E、控制逻辑单元 3010、呈现模块 3011A-E、调度和控制逻辑单元 3020、缓冲器 3030A-B、数模转换器 (DAC) 和控制逻辑单元

3040 和微镜阵列的高层框图。分割可以 Mercury 处理器集群或任何其它适合的几何图形分割引擎来支持。所述集群可以是每个呈现模块一个分割处理器，如图所示。或者，分割可以是产生数据文件的离线处理。所谓的 FRAC-S 格式可以用于将数据从分割处理传送给呈现处理。被分割的数据的有益特征包括：所有被分割的几何图形可以小于一个矩形，所述矩形能够完全适配到在保护区中的任何位置；所有的分割几何图形可以小于一个矩形，所述矩形能够完全适配到在呈现窗口中的任何位置；所有的分割几何图形可以小于一个矩形，其边具有以像素测量的最大尺寸，诸如 255 或 253 个像素或者 511 或 509 个像素；所有的被分割几何图形必须至少 $1/2$ 像素高和 $1/2$ 像素宽。这些特征可以依赖于在特定实施例中使用的本发明的方面的组合而被修改。

SDU 执行一个处理，所述处理从逻辑角度来看是光栅域重新分割，用于将在分割/光栅域中的窗口划分转换为匹配 SLM 分区的窗口划分。从物理的角度来看，它也执行集中处理，但是从调制的角度来看，实际的集中不发生直到在 SLM 中。理论上，像素传送和数模转换可以发生在完全独立的器件中，所述器件不连接直到在 SLM 处，但是由于像素处理和调制器的不同定标属性，并且由于信号完整性和其它原因，数据被物理地集中在 SDU 中。

在第一实施例中，缓冲是有限的。图 29 描述的配置可变地轮询线输入，使用在呈现模块和 SDU 之间的一个或多个缓冲器，或者可以被适配来利用在 SDU 和 DAC 之间的一个或多个缓冲器。呈现模块 2911A-E 连接到 SRM 2921A-E。SRM 是 SDU 接收器模块，它向 SDU 提供接收来自 RM 的数据的接口。SLM 链路接口 (SLI) 将 SRM 连接到 SDU 2940。这个接口可能受到较弱信号完整性的影响。在图 29 中的配置上的逐个线的变化中，不使用缓冲器 2931，要求复杂协议需要用于线的确认/否认和发送-重发。SRM-SDU CABS 接口处理时钟域斜线与“等待直到可以获得所有线，然后进行”协议的同步。将在 RM 上的帧处理同步具有等待时间和闪光速率不确定性的显著损失。SDU 的输出是到连接到 SLM 的 DAC 对。

在第二实施例中，环形缓冲器连接到每个 SRM 2931。环形缓冲器总计带宽和大小与 SRM 的数量成线性比例。SRM 的数量大致上与像素的吞吐量成比例。环形缓冲器消除了在 SRM-SDU 之间的线同步和在 SRM-SDU 之间的帧同步的需要。如果所述缓冲器足够大，则它也将支持在曝光开始之前的完整带的光栅化。基于 SDRAM 的环形缓冲器已经在 SRM 的 RT-fract 侧上存在，

并且可以与其测试工具和接口一起复用所述设计。带宽和大小不是重要的，主要目的是将同步方法从逐个线改变为逐个带。SLM 链路接口在逐个带的基础上被同步。SRM-SDU CABS 接口不必处理除了基于逐个带之外的‘可用-进行’事件。这大大地提高了 SLM 链路接口和 SRM-SDU 接口的强壮性，并且去除了对于功能齐备的发送-再发 SRM 的设计和集成任务。对基础结构的可靠性要求被放松，其仅仅被回扫强度(retrace intensity)引导。在带级上处理所有例外使得 SRM-SDU 接口变为完全同步的系统。或者，缓冲器 2931 可以被布置在 SDU 和 DAC 之间。

在图 30 中描述了第三实施例。一个或多个 SDU 缓冲器 3030A-B 和相关的控制逻辑单元使得所使用的呈现 3011A-E 单元的数量独立于在微镜阵列 3050 中的段 3052A-D 的数量。有益的是，产生光栅化图像的处理器的数量独立于微镜段的数量。对于对具有特别复杂的几何图形的应用，可以提高分割和呈现单元的数量。当所述多个单元并行动作时，所涉及的单元越多，则可以获得越大的处理能力。对于对特别简单的几何图形的应用，分割和呈现单元的数量可以降低。调度和控制逻辑单元 3020 允许从呈现模块向缓冲器异步提供数据。

缓冲器本身被不同地分段以输入和读出。输入调制器窗口 3031A-E 对应于呈现单元。调制器窗口在这个图中被描述为相同大小，但是它们可以在大小上不同，并且可以在行和列两者之上划分存储器阵列。在缓冲器 3030A-B 中的存储器位置对应于在微镜阵列中的微镜或用于扫描系统的光栅化图像图。所述存储器可以是双端口的，以便可以装入来自呈现处理器的结果而不破坏读出以装入 DAC。在微镜闪光系统中，在每次闪光之前，镜子阵列被装入数据。在所图解的实施例中，128 个 DAC 3040 和控制逻辑单元用于向微镜阵列装入数据。在微镜阵列中或微镜阵列之外的复用电路可以从 DAC 向许多微镜分布模拟信号。8mhz 可以为大约 8,192 微镜执行 1khz 频率的转换。这 128 个 DAC 每个都可以处理 32 行 256 个微镜。通过 8 比特 DAC 来产生模拟电压以建立静电电荷，所述静电电荷偏转独立的微镜。在一些实施例中，更大的精度，诸如在模拟电压中的 10 或 14 比特可能是有益的。诸如下述微镜阵列可以被适当地用作投影器件：Micronic Laser System 公司共同拥有的申请，发明人 Torbjorn Sandstrom，“用于图形产生器的改善的调制器设计”，WO99/45441（优先权日为 1998 年 3 月 2 日）；或上述引用的同时提交的临时

申请。或者，可以使用基于微镜或调制的辐射的扫描器件。对于基于微镜的扫描器件，可以使用足够大组的 DAC 来在一个遍次中向所有微镜传送数据。对于调制辐射，每个辐射光束具有一个调制器是惯例。

可以使用一个、两个或多个缓冲器单元 3030A-B。从呈现模块异步接收数据和呈现模块的处理在时间上的变化最好被多个缓冲器单元支持。对于一些光刻设备，由于工件休止所在的移动级的惯性，微镜阵列的闪光不能在带的扫描期间被延迟。一些延迟可以在带之间被容纳。因此，DAC 的操作可以仅仅被延迟到这样的程度：段装入的顺序可以基于从呈现模块 3011 向调制器窗口 3031 写入数据的完成。如果任何呈现模块 3011 花费大于一个闪光周期来在缓冲器中装入其调制器窗口，则将引起数据错误并且可能浪费工件。使用两个或多个缓冲器 3030 允许在特定调制器窗口中的例外呈现周期的额外范围。

图 31 是呈现模块 3011 的方框图。在一个实现方式中，使用 5 个现场可编程门阵列（FPGA）来实现逻辑 3100、3110、3120A-B 和 3140。适合于逻辑实现的当前一代 FPGA 是 Xilinx Virtex XCV1600E-6 BG560 C FPGA。已经被声明的、但是尚在等待提供的 Xilinx-II XC2V 产品系列预期提供附加的功能，包括板上的 SRAM，它可以允许向较少的模块中并入功能块。或者，可以使用 RISC 处理器、ASIC 处理器或定制或半定制的处理器的 FPGA 的选择是方便的并降低了产品的研制时间和预先成本。多种其它处理器可以一样地用于实现本发明的方面。

RMPL-1 3300 的目的是提供到 PCI、CABS 和其它总线结构的接口。接口 3103 可用来现场配置 FPGA。可以重新配置或重新装入所有或部分的 FPGA。例如，可以现场重新配置和重新装入用于子像素加权的加权系数。这些接口允许测量板的工作温度。它们还支持对板所使用的时钟频率的编程。

几何图形预处理器概览

RMPL-2 3110 的目的是提供在几何图形输入光纤信道和呈现处理器之间的接口。主要任务是接收和缓冲复杂几何图形数据，将数据变平为原始几何图形，并且在请求下向呈现处理器输出数据。RMPL_2 因此包括：用于板上 256 M 字节 SDRAM 的接口；用于板上 2*8 M 比特 ZBT RAM 的接口；管理针对外部器件的 CABS 协议的链式访问总线结构（CABS）单元；所谓的几

何图形输入控制器 (GIC) 3113 的接收单元, 它从几何图形线接口 (GLI) 接收几何图形数据, 并且在 SDRAM 中存储数据; 预处理器 (PP) 3111, 它将几何图形数据变平并且执行坐标转换; 在 CABS、GIC、PP 和 SDRAM 之间的几何图形缓冲器接口 (GBI) 3112, 它还实现 SDRAM 控制器。RMPL-2 3110 的 CABS 块作为在内部块、零总线转变 (ZBT) SRAM 3115 和其它 FPGA 或更一般而言其它处理器或功能部件之间的接口。CABS 是用于器件内通信的一般总线结构。它基于器件的链状结构, 并且消除了一般多节点总线结构的板级的复杂性。在 CABS 上的采样可以是 25-50 MHz。它使用板级系统时钟来计时。CABS 块可以从存储器的引导区域连接到用于器件的默认配置的模式控制引脚。或者, 可以使用其它的总线结构和协议来实现本发明的方面。

GIC 块是在 GLI 链路和 GBI 之间的接口。它包括 FIFO 缓冲器, 用于将所接收的 32 比特的双字 (DS) 编组为 16 字的分段。一次发送一个分段到 GBI 以存储在 SDRAM 中。在 GLI 上的字可以以任何大小的分段到达。GBI 块作为在 GIC 和 PP 之间的接口。它包括逻辑单元, 用于从 GIC 接收数据分段, 将其存储在 SDRAM 中, 并且在请求下从 SDRAM 读取数据和向 PP 发送它。GBI 也可以处理来自 CABS 的读取和写入请求。在 CABS-GBI 接口, 数据一次以 $n \times 16$ 双字的分段长度被写入, 其中 n 是 1-64, 除了用于数据传送的使能信号之外, 不需要实际传送长度的指示。可以一次以 16 个双字的分段来读取数据。

几何图形预处理器 (PP) 块作为在 GBI 和 FPGA 3120A-B 之间的接口。预处理器将输入的分割数据流变平, 并且输出具有相对于当前的呈现窗口的坐标偏移的简单几何图形。分割的数据被从 GLI 信道接收并且被存储在几何图形缓冲存储器 (GBM) 3116 中。可以通过预处理器来获得原始数据模式以读出存储在 GBM 中的所有内容来用于测试目的。在图 40 中给出了关于预处理器核心的进一步细节。

存储器控制器包括 ZBT-SRAM 控制器和 SDRAM 顶部模块。ZBT-SRAM 控制器与诸如符合标准 PC 100 或 PC 133 的 256 MB 存储器的 SDRAM 连接。ZBT SRAM 器件是通过利用每个总线周期来提供最大系统吞吐量的同步 SRAM。如其名称所示, 不存在当在读取和写入周期之间切换时需要的转变周期。因此, 不存在浪费的周期, 并且系统实际提供所声明的带宽。呈现模块 ZBT SRAMS 可以获得休眠模式, 当启动该模式时将存储器置于低功率待

机模式中,以保留数据和忽略输入。这在重新配置 FPGA 期间可以是有益的。为了降低等待时间,ZBD-SRAM 控制器可以通过双频时钟来计时。SDRAM 顶部模块包括逻辑单元,用于产生用于执行存储器写入、读取、刷新和内置的自行测试所需要的信号。在 FPGA 中使用的各种 ZBT-SRAM 控制器,可以使用单个和双 32 和 64 比特的数据路径。例如,在预处理器 3111 和工作存储器 3115 之间的接口可以是一对独立的 32 比特数据路径。在呈现处理器 3121 和帧缓冲器之间的接口可以是一对 64 比特的数据路径。

呈现处理器概览

在图 31 中,指示了一对呈现处理器 3120A-B。这些呈现处理器的每个可以被实现在 FPGA 中,或者可以以 RISC 或 ASIC 处理器或者定制或半定制的处理器来被实现。可以使用两个处理器,以便一个从预处理器 3111 接收数据,而另一个向调整处理器 3141 写入数据。从预处理器 3111 到呈现处理器 3120A-B 的数据路径可以被复用器或类似的器件控制,或者呈现处理器可以忽略意欲到达互补处理器的信号的组。可以使用一个附加的呈现处理器。呈现处理器 3120A 被图解为具有到一对帧缓冲器 3122 的数据路径。这两个存储器被图解为具有分立的数据路径,即用于存储器访问的独立信道。一个用于像素图存储器,另一个用于灰度值超级采样中间存储。呈现处理器 3121 也被图解为具有到附加帧缓冲器 3123 的另一个独立数据路径或信道,所述附加帧缓冲器 3123 可以具有微像素帧缓冲器。一个或多个帧缓冲器 3123 可以用于微像素。在每个像素 64 个微像素的一种实现方式中,从处理器到帧缓冲器 3123 的数据路径可以是 64 比特宽。可以使用一对帧缓冲器 3123,以便可以在对另一个存储器执行读取、写入或读取/修改/写入的同时在一个帧缓冲器中执行存储器清除功能。在这个附图中未图解灰度值总计缓冲器。虽然这些存储器被图解为在 FPGA 外部,它们也可以被内部并入到适当的 FPGA、定制或半定制处理器中。

图 32 提供了关于可以用于实现本发明的方面的呈现处理器的功能块的附加细节。呈现处理器 3120 包括几个功能块。CABS 接口 3211 对于通过总线结构连接的系统的部件是共同的。分割几何图形转换器 3212 通过传送可呈现定点几何图形的数据路径而连接到 CABS 接口。转换器使用 50 MHz 的时钟频率来以每秒 50 百万的速度来提供单边角坐标。可以使用更快或更慢的时

钟速度来获得更快或更慢的吞吐量。转换器 3212 的输出是以角坐标几何图形格式。可以使用一对数据路径来向一对微像素高速缓存产生器 (MPCG) 3132L-R 提供数据。使用与多边形几何图形相结合的一对 MPCG 使得可以沿着多边形的右和左边来划分处理。可以根据诸如所使用的吞吐量和几何图形的设计特征来使用更少或更多的 MPCG。当多边形几何图形是梯形时,可以向不同的处理器分配多边形的顶部和平行的底部,例如在它们开始处理梯形的相对的、不平行的边之前。在图 33 中描述了关于转换器的附加细节。

图 32 的 MPCG 3213 使用 50 MHz 的时钟频率以每秒 50 百万 MiPxCaW 的速度提供微像素高速缓存组。可以使用更快或更慢的时钟速度来获得更快或更慢的吞吐量。MPCG 对向微像素高速缓存缓冲器 (MPCB) 3214 输出高速缓存组。在图 34 中描述了关于 MPCG 的附加细节。

微像素高速缓存缓冲器 3214 将包括单边信息、即梯形的左、右、上和下边的、被呈现的微像素高速缓存组转换为包括来自用于几何图形的一个到四个单边高速缓存的信息的、微像素高速缓存组。当几何图形包括被包含的上或下边线段时,上和下边的被包含段可以在计算角前后被通过。地址组也如上所述产生。地址组隐含地传送将被产生的微像素高速缓存的顺序。可以使用分立的数据路径来用于传送高速缓存组和地址组。这些高速缓存集体定义被呈现的多边形的边。在图 35 中描述了关于微像素高速缓存缓冲器 3214 的附加细节。

帧缓冲器接口 (FBIF) 3215 可以工作在三个阶段:呈现、读出和清除。在呈现阶段中,FBIF 从 MPCB 获取几何图形,并且将它们存储在一个或多个缓冲存储器中。在第一实施例中,唯一的存储器可以是微像素帧缓冲器 (MPFB) 3222。可以使用多个 MPFB 来允许在另一个 MPFB 上工作的同时清除一个 MPFB。这些存储器可以双端口的,并且可以具有用于存储器访问的独立信道。在第二实施例中,可以既有微像素帧缓冲器也有像素图帧缓冲器 (PMFB) 3223。可以有多个像素图存储器,它们最好带有独立的用于存储器访问的信道以便增强性能。在第三实施例中,可以有增加的灰度值帧缓冲器 (GVFB) 3224 来总计在微像素 (MPFB) 中的值。再次,可以使用多个灰度值存储器和独立的信道。在一个混合实施例中,PMFB 可以具有足够的值来作为一个 GVFB。帧缓冲存储器存储完整呈现窗口,它可以以 x、y 坐标来定位。这些存储器以不同的分辨率来存储被呈现的图像。对于所有像素来

说, 像素图 PMFB 存储器以两个比特来存储白色、黑色、灰色信息。微像素 MPFB 存储器存储对应于像素的独立微像素。例如, 8×8 、 16×16 、 32×32 或 64×64 微像素的阵列可以对应于一个像素。这些微像素可以具有黑色或白色值或者替代为灰度级值。组合微像素高速缓存的方式依赖于微像素是具有黑/白或灰色值。使用 MPFB 来利用灰色值 GVFB 存储器, 以便相加微像素阵列的赋值。微像素阵列可以在被读出或当它们被写入并且结果被存储在 GVFB 时被赋值为灰色值。

在读出阶段中, 对于每个像素报告灰色值。在第一实施例中, 仅仅使用 MPFB 存储器, 这涉及对所有的微像素阵列赋值。在第二实施例中, 使用 MPFB 和 PMFB 存储器, 一些像素位置在像素图中被记录为黑色或白色。系统可以将那些位置赋值为整数值, 诸如赋值为 0 或 64 或者赋值为 0 或 256, 而不用访问 MPFB 存储器, 有可能降低对 MPFB 存储器访问带宽的需要。在第三实施例中, 每次写入 PMFB 时, 向 GVFB 存储器赋值。可选地, 每次更新 PMFB 时, 向 GVFB 存储器赋值。在一种方式中, PMFB 计算以确定是否赋值一个白色或黑色值或利用 GVFB 值。在另一种方式中, GVFB 值被直接使用。来自 FBIF 3220 的灰度值被报告到 CABS 接口 3231 来作为光栅图像。

在清除阶段中, 一个或多个存储器被设置为黑色/白色, 以便可以在缩小的背景中呈现白色/黑色几何图形。在第一实施例中, MPFB 存储器被清除。在第二和第三实施例中, PMFB 被清除。对于一些类型的存储器, 清除较小的 PMFB 比 MPFB 或 GVFB 消耗更少的时间, 因为存储器的单个字可以表示低分辨率的多个像素。在第二和第三实施例中, 可以选用采取清除 MPFB 或 GVFB 存储器的更耗时的步骤, 因为像素图存储器控制是否使用其他的存储器来产生灰色值。即, 如果像素图存储器指示一个像素的值是黑色或白色的, 则在其他存储器中的任何灰色值可以被忽略, 因为其他存储器可能在先前的操作之后还没有被清除。对于支持在最小时钟周期中批量消除存储段的存储器配置, 用于清除 MPFB 或 GVFB 的开销不会使得处理变慢。

调整处理器概览

调整处理器 3140 经由 CABS 接口 3231 接收数据或从呈现处理器 3120A-B 接收另一种适当的总线结构数据。它可以以其 CABS 接口的频率的两倍来被定时。调整处理器包括用于边位移、照明补偿和镜子补偿的块。它

还可以包括用于校正在光学系统中的较小缺陷的逻辑单元。边位移如上所述是将曝光辐射的投影图像调整以调整线宽而不是调整曝光的抗蚀剂的处理的方法。照明补偿也如上所述是处理在单个或多个写入遍次中的重叠曝光和在期望的曝光辐射和实际的曝光辐射之间的能量差异的方法。镜子补偿将期望的照明值转换为用于独立镜子的驱动值。它也可以用于补偿当镜子变得老化时或在使用镜子的系统的休止周期之间变化的镜子响应。在使用微镜阵列和曝光辐射的闪光的系统中，镜子补偿可以将照明值转换为用于对独立微镜充电的 DAC 的输入。

调整处理器 3141 经由数据路径来访问系数存储器 3143。这个存储器保存用于镜子补偿的系数，并且也可以保存用于在照明补偿中使用的区域图的系数。ZBT-SRAM 存储器可以被用作外部存储器。或者，可以使用内部存储器。调整处理器还访问帧缓冲器存储器 3142。可以使用一个或多个 ZBT-SRAM 存储器来作为用于边位移的工作存储器，并且保存等待在一个或多个背板通道 3144-45 上输出的最后值。

像素输出控制器（POC）提供格式转换以将数据适配到物理层。

几何图形预处理器

在几何图形预处理器以所谓的 FRAC-S 格式来接收几何图形说明。下面是所示的 FRAC-S 流的 BNF 语法。FRAC-S 流可以被存储在包括各种处理相关首标参数的、所谓的 FRAC 文件中。这些参数一旦被装上就可以通过控制寄存器被几何图形预处理器获得。

```

FRAC_S_FILE ::=          SUBSTRIP
                        [SUBSTRIP]*

SUBSTRIP ::=             <SUBSTRIP_START>
                        SUBSTRIP_BODY
                        <SUBSTRIP_END>

SUBSTRIP_BODY ::=        RENDERING_WINDOW
                        [RENDERING_WINDOW]*

```

```

RENDERING_WINDOW ::= <RENDERING_WINDOW_START>
                    [REND_WIN_BODY]*
                    <RENDERING_WINDOW_END>

```

```

REND_WIN_BODY ::=  GEOMETRY
                   [GEOMETRY]*
                   [LAYER]*

```

```

LAYER ::=          <BEGIN_LAYER>
                  [GEOMETRY]*
                  <END>

```

```

GEOMETRY ::=      <RECTANGLE>
                  |
                  |   <SMALL_RECTANGLE>
                  |
                  |   <REPEAT_RECTANGLE_X>
                  |
                  |   <REPEAT_RECTANGLE_XY>
                  |
                  |   <TRAPEZOID>
                  |
                  |   <REPEAT_TRAPEZOID_X>
                  |
                  |   <REPEAT_TRAPEZOID_XY>
                  |
                  |   <BEGIN_X_REPEAT>[GEOMETRY]*
                  |
                  |   <END>
                  |
                  |   <BEGIN_XY_REPEAT>[GEOMETRY]*
                  |
                  |   <END>
                  |
                  |   <BEGIN_Y_REPEAT>[GEOMETRY]*
                  |
                  |   <END>
                  |
                  |   <BEGIN_INSERT>[GEOMETRY]*<END>

```

符号[x]*表示“x的0次或更多的出现”。以这种格式的输入流被转换为在这个格式的子集中的输出流，其中诸如分层和重复的说明的复杂几何图形说明被简化。在输出流中的记录类型是矩形、小矩形、梯形、开始层和结束层。在一种实现方式中，重复级的数量是5。一次预处理一个呈现窗口。可以以

大于子像素或微像素的精度来处理数据。所谓的软像素是微像素的一半。

实时几何图形预处理器的处理速率部分依赖于几何图形的复杂程度。对于具有每秒 1,000 的闪光速率和 2048 x 512 像素（微镜）的阵列的闪光曝光辐射系统，整个系统每秒处理 104.8×10^6 个像素。对于半导体的金属层图形，这对应于每秒 7,000,000 个几何图形的平均几何图形速率。使用每个几何图形四个记录的平均值，所需要的记录输出速率将是每秒 28,000,000 个记录。

所述几何图形预处理器连接到三个其他模块，即它从其读出 FRAC-S 数据的 SDRAM、用于暂时存储的 ZBT RAM 和处理数据寻址和帧化的 CABS 单元。

PP 的整体操作如下：

- 1) 在 RESET, PP 等待 STRIP_SYNCN 事件。
- 2) 在 STRIP_SYNC, 前两个呈现窗口被读入，被变换为对应的 FRAC-L 序列，并且被发送到两个呈现处理器。所述预处理器然后等待 RP_DONE 事件。
- 3) 在 RP_DONE, 下一个呈现窗口被读入、处理和发送到第一呈现处理器。所述预处理器然后等待新的 RP_DONE 事件。
- 4) 在 RP_DONE, 下一个呈现窗口被读入、处理和发送到第二呈现处理器。所述预处理器然后等待新的 RP_DONE 事件，并且进行到 3)。

注意，从接收到 STRIP_SYNC 事件时直到遇到一个 SUBSTRIP_END 记录，FRAC-S 数据被预先装到本地输入高速缓存中。这有效地最小化了从 SDRAM 访问带来的等待时间。以双字的连续分段执行 FRAC-L 输出。因此，预处理器向内部缓冲器中存储输出记录，直到已经存储至少 10 个完整的 FRAC_L 块、或已经遇到一个 RENDERING_WINDOW 结束块。这种 10 个块的规则意味着到 RP 的平均分段长度将是大约 32 个双字。在每个呈现窗口的尾部，发送一个结尾的 END 记录。预处理器记录 LAYER 记录参数如下：层号应当以连续的顺序，开始于用于第一 LAYER 记录的 1、用于下一个的 2，等等。如果这种排序失败，则发出一个错误。并且，如果层操作 = IGNORE，则丢弃整个层。

下面示出了各种 FRAC 记录及其可以在一个呈现窗口内出现的对应块结构。对于分层结构，{G1..Gn} 表示在分层内的解析几何图形。即，块总是封闭最外的重复级，并且可以包括任何数量的下伏级。

FRAC 序列	块结构
<RECTANGLE>	RECT(x,y,dx,dy)
<SMALL_RECTANGLE>	SMALL_RECT(x,y,dx,dy)
<TRAPEZOID>	TRAP(x,y,dx,dy,dx1,dx2)
<REPEAT_RECTANGLE_X>	REPEAT_X(xrep,xdist,RECT(x,y,dx,dy));
<REPEAT_RECTANGLE_XY>	REPEAT_XY(xrep,xdist,yrep,ydist, RECT(x,y,dx,dy))
<REPEAT_TRAPEZOID_X>	REPEAT_X(xrep,xdist,TRAP(x,y,dx,dy, dx1,dx2))
<REPEAT_TRAPEZOID_XY>	REPEAT_XY(xrep,xdist,yrep,ydist, TRAP(x,y,dx,dy,dx1,dx2))
<BEGIN_LAYER>	LAYER(layer_no,oper)
<END>	END
<BEGIN_X_REPEAT>	
[GEOMETRY]*	
<END>	REPEAT_X(xrep,xdist,{G1..Gn})
<BEGIN_Y_REPEAT>	
[GEOMETRY]*	
<END>	REPEAT_Y(yrep,ydist,{G1..Gn})
<BEGIN_XY_REPEAT>	
[GEOMETRY]*	
<END>	REPEAT_XY(xrep,xdist,yrep,ydist,{G1..Gn})
<BEGIN_INSERT>	
[GEOMETRY]*	
<END>	INSERT({xoffs1..xoffsm},{G1..Gn})

```

<SUBSTRIP_START>          -
<SUBSTRIP_END>            -
<RENDERING_WIN_START>     -
<RENDERING_WINDOW_END>    REND_WIN_END(CPC)

```

对于每个呈现窗口，将可交换地使用两个存储器区域。到达的几何图形块在被穿过的同时被暂时地逐个存储在当前存储区域中。如果在随后的呈现窗口（RW）中不需要一个块，则在完全穿过之后将其从存储器去除。进一步需要的块在存储器中排队以用于随后的 RW。在示例中，来自第一 RW 的一个 REPEAT_X 4103 和 REPEAT_Y 4105 块保持被保存 4113、4115。对于下一个 RW，来自前一个 RW 的 REPEAT_X 块 4113 与来自输入的 FRAC-S 流的一个 INSERT 块 4142、4122 被保存 4131。

穿过一个块依赖于由块表示的几何图形的类型而不同。表示简单几何图形的块可以被直接转换为 FRAC-L 记录，而重复的几何图形块需要被递归地穿过。穿过程序如下：

- 1) 如果块类型是 {RECT, SMALL_RECT, TRAP, LAYER, END, REND_WIN_END}，则将所述块转换为记录序列。
- 2) 如果块类型是 {REPEAT_X, REPEAT_Y, REPEAT_XY, INSERT}，则：
 - 将开始地址和重复参数推到堆栈上；
 - 在上面的 1 开始递归地穿过子块；
 - 当进行完所有的重复时，将堆栈上托到前一个级。

图 40 是预处理器核心的方框图。FRAC-S 输入缓冲器 4101 避免或最小化由于在流程控制机制中的传播时间而导致的溢出。通过将 DRQ 信号设置为高来请求新的 FRAC-S 记录。对于其中 DAV 信号被设置为高的每个周期，一个新的记录被写入缓冲器中。只要 CTS 输入信号为高，则记录被从缓冲器读出并且被发送到解析器。将 CTS 设置为低就停止输出流。所接收的每个呈现窗口与存储在 RENDERING_WINDOW_END 记录中的循环冗余检验（CRC）代码相关联。CPC 控制块 4102 重新计算这个校验和，并且如果所述和与基准值不同则发出错误信号。在有错误信号的情况下，可以使用各种恢复方案。

解析器模块 4103 从缓冲器读取 FRAC-S 记录，并且将记录编组为几何图形块。当将 get_block 信号设置为高时，解析器输出将被存储在存储器中的一

个整块。重复的几何图形（即 REPEAT_RECTANGLE 和 REPEAT_TRAPEZOID）被转换为 REPEAT 块以简化进一步的处理。

如果解析器 4103 由于在 FRAC-S 流语法中的错误而不能产生块，则断言例外信号 syntax_err，并且可以放弃在当前呈现窗口中的操作。语法误差包括下列：没有 <END> 的 <BEGIN_REPEAT>、<BEGIN_INSERT> 或 <BEGIN_LAYER>，或者没有 <BEGIN_REPEAT>、<BEGIN_INSERT> 或 <BEGIN_LAYER> 的 <END>；在 REPEAT 或 INSERT 块中的 <BEGIN_LAYER>；或者在层说明中的 <BEGIN_LAYER>。

存储器 4104 包括被组织为 32 比特字的两个 ZBT RAM 4105A-B，它们被如上所述使用。存储器控制 4106 控制是否应当从存储器或从解析器读出新块。它选择要读写的存储区域和地址，并且执行读/写操作。为了降低存储器读取的等待时间，可以使用双时钟频率来定时到 ZBT SRAM 模块的接口。

穿过和产生模块 4107 穿过在存储器中的几何图形块，并且产生原始的几何图形（形状）。堆栈 4108 用于处理多个分层级。当进入一个新级（即 REPEAT..END 声明）时，重复参数被推到堆栈上，并且堆栈指针被递增。当完成了在当前级上的所有重复时，堆栈 4108 被上托到前一级。如果坐标加法器块拒绝几何图形，则也上托堆栈，并且当前块将被保留在存储器中来用于下一个 RW。堆栈处理器 4108 包括堆栈读/写机制和用于在重复循环中递增坐标的逻辑单元。它控制是否存储当前块、跳回循环的开始处或进行下一个指令。坐标加法器 4109 将存储在堆栈中的、累加的 $\sum dx$ 和 $\sum dy$ 增量加到到达的几何图形的当前 x 和 y 坐标。如果一个几何图形落到当前的呈现窗口坐标之外，则设置拒绝信号以指示应当上托这个重复级。

分割的几何图形转换器

图 33 是分割的几何图形转换器（FCON）的方框图。分割的几何图形转换器 3212 包括两个主块 3321、3323 和 FIFO 存储器 3322。FRAC-L 记录处理器块 FLRH 3321 作为到 CABS 块 3211 的接口。它的主要任务是从 FRAC-L 记录检索几何图形信息。FRAC 记录被从 CABS 接收为 32 比特的字。分析几何图形的类型信息并且被存储在记录中。几何图形到角转换器 GCCON 3323 将几何图形的 FRAC 表示转换为几何图形角表示。它也将数据划分为左和右数据流。在 FLRH 3321 中包括的 FIFO 存储器 3322 保证 CABS 数据传送均匀

地流动。这个 FIFO 存储器可以包括多达 255 个 FRAC-L 字。FCON 将使用控制信号来指示何时缓冲器几乎满或需要被重新填充。

在 GCCON 块 3323 中,几何图形开始坐标和长度规格被转换为几何图形角说明,并且被划分为上、左和下、右坐标对。为了简化下游处理块的功能,对于每个新的几何图形,首先方便地发送上和下几何图形信息。因此,即使几何图形是三角形也存在顶部信息。如果几何图形边坐标位于保护窗口的外部或在顶部或左边界上,则将启动 FCON 错误指示。

微像素高速缓存产生器

图 34 是图解呈现 212 的一个实施例的方框图,它是在图 2 中的呈现引擎的部分。子像素阵列或网格有时被称为微像素高速缓存。微像素高速缓存产生器包括四个块:启动;呈现;表查找;转换。输入 3401 可以是可呈现的定点几何图形,或更一般而言是要呈现的多边形。启动块 3402 计算用于将 Bresenham 呈现算法应用到多边形的开始参数。Bresenham 算法的应用结合附图 8-11 被说明。呈现块 3403 执行 Bresenham 算法。表查找块 3404 包括查找表,用于将微像素高速缓存坐标转换为子像素图像。这个处理结合附图 12-14 被说明。最后的块转换 3405 将用于子像素条值的值转换为在子像素网格中的阴影子像素。在一些实例中,为在子像素网格中的子像素加阴影,需要在查找表中值的旋转或反转。如果必要的话,可以使用触发器阵列在硬件中执行从查找表的子像素条值的转换和条的旋转,所述触发器可以在水平与垂直两个方向上寻址。来自微像素高速缓存产生器的输出 3406 是微像素高速缓存组。

微像素高速缓存缓冲器

微像素高速缓存缓冲器 3214 包括 5 个主要块,如图 35 所示。左(L) 3512 和右(R)3514 FIFO 存储器从两个 MPCG 3213L-R 接收 MiPxCaW 信息。对于每个新的几何图形接收的第一对 MiPxCaW——它们包括上和下边信息——被存储在上(T) 3511 和下(B) 3513 FIFO 中。四个 MiPxCaW FIFO 连接到组合的复用器和逻辑与功能 3521,产生最后的被呈现 MiPxCaW。存在一个微像素高速缓存组产生器 MPCSG,它被划分为三个块 3501、3502、3504,用于产生用于控制逻辑与功能 3521 的微像素高速缓存组和由帧缓冲器

接口 FBIF 3215 使用的信息, 所述帧缓冲器接口 FBIF 3215 描述在输出上如何组织被呈现的 MiPxCaW。

帧缓冲器接口

图 36 是帧缓冲器接口 3220 的一个实施例的方框图。如图所示, 处理流从微像素高速缓存产生器 3213 流向微像素高速缓存缓冲器 3214 并且流到 FBIF 3220 上。FBIF 的第一元素是保护区过滤器 3641, 它是上述的保护区的一种硬件实现方式。图 37 图解了这个过滤器的操作。考虑几何图形 3763。它跨越在呈现窗口和保护区 3762 之间的边界 3761。在保护区内的左和底边的部分以阴影指示。保护区过滤器 3641 接收输入数据组 3751, 并且产生输出数据组 3752。对于在网格的行 6 中的图形 3763 的部分, 数据 3751 包括在列 2 中的边开端、在列 3 中的边尾端和用于像素 6,2 和 6,3 的微像素高速缓存。过滤器检测 6,2 和 6,3 在保护区中。它产生新的地址组和新的微像素高速缓存 3752。行 6 边线段的开端和尾端被设置为 6,4, 并且单个、白色微像素高速缓存 MiPxCaW_X4 替换微像素高速缓存的输入对。接口 3643、3642 控制对于相应的存储器 3223、3222 的访问。读出逻辑单元 3644 从相应的存储器向 CABS 3231 接口传送值。所述逻辑单元依赖于存储器的实施例, 如上所述。在图 7 中描述了一种读出逻辑单元。在图 38-39 中示出了一个实施例的进一步的细节。

图 38 和 39 是使用 MPCB 存储器 3222 而不是 PMCB 存储器 3223 的 FBIF 3220 的第一实施例的方框图。为吞吐量而提供两个子单元 3803A-B, 它们以如下所述的交错方式操作以有效地利用带宽。可以根据子单元的性能和期望的吞吐量而使用更少或更多的子单元。如上所述, 第二代 FBIF 具有三个阶段: 1) 呈现阶段; 2) 读出阶段; 3) 清除阶段。调度块 3802 调度这 3 个阶段。在一个实施例中, 作为读出阶段的一部分, 执行超级采样。(在另一个实施例中, 可以在每次写入微像素阵列时进行超级采样)。恰好在数据被写入 3821 到 CABS_代理 3801 中之前执行超级采样。CABS_代理 3801 可以放弃在所有块中的当前操作。当设置放弃信号 3825 时, FBIF 将放弃当前的功能直到: MPCB 3214 以信号表示新的数据, 然后 FBIF 3220 在呈现阶段中开始并且当前的存储内容用于新的呈现操作中; 或者 CABS 以 send_frame_cabs 信号 3822 来命令 FBIF 读出数据。一般, FBIF 块可以被双定时。

在图 38 和 39 中,调度器 3802 调度三个阶段来用于在 4 阶段交错调度中的两个子单元:

处理器	缓冲器	阶段 1	阶段 2	阶段 3	阶段 4
RP A	MPFB A1	清除	呈现	读出/超级 采样	空闲
	MPFB A2	空闲	清除	呈现	读出/超级 采样
RP B	MPFB B1	读出/超级 采样	空闲	清除	呈现
	MPFB B1	呈现	读出/超级 采样	空闲	清除

调度器 3802 向子单元 3903A-B 发送信号 3827 呈现、清除和读出。当子单元 8303A-B 接收到来自 MPCB 3214 和 CABS_代理 3801 的数据和控制 3822、3823、3824 时,它们与它们所在的阶段一致地响应。调度器使用来自其他逻辑单元的信号。从 MPCB 3214,它使用 downward_ready 和 WEND。从 CABS_代理 3801,它使用 send_frame_CABS。从子单元,它使用 frame_done_CABS、clear_done。如果接收到一个放弃信号,则调度器进入阶段 1,并且子单元 A 等待 downward_ready 以开始呈现;同时,在子单元 B 中开始清除存储器处理。如果接收到放弃,则当前在 FBIF 中处理的所有数据将被丢失,但是存储器的内容将不由于放弃而被改变。

控制逻辑 3904 也与其所在的阶段一致地响应。在呈现阶段期间,downward_ready 和 upward_req 信号控制从 MPCB 3214 到 FBIF 3220 的数据流。当 MPCB 信号 Downward_ready 激活以指示数据可用时,呈现阶段开始。FBIF 将接收数据,直到没有用于当前呈现窗口的数据,并且 WEnd 信号变得激活。只要 Downward_ready 是激活的,则存在从 MPCB 块接收的 MiPxCaW。如果 Downward_ready 是非激活的并且未接收到 WEnd 信号,则存在用于这个呈现窗口的更多的 MiPxCaW,但是在此时 MPCB 缓冲器为空。只要 FBIF 准备好接收新的数据,则它将 upward_req 保持激活。当 WEnd 信号变为激活时,呈现阶段结束。FBIF 然后向 CABS_代理发送 frame_ready_CABS,因此

CABS_代理可以获取数据。Downward_ready 仅仅在在激活的 WEnd 或 GEnd 脉冲之后才变得非激活。在激活的 WEnd 或 GEnd 脉冲后，在 FBIF 中的流水线被清空。当新的多个 MiPxCaW 来自新几何图形的 MPCB 到来时，在流水线中将不存在任何来自旧几何图形的未处理的旧 MiPxCaW。当地址组被处理时，控制块检查在存储器中的地址是否需要被增加或降低，并且向地址产生器提供上或下计数命令。当白色 MiPxCaW 被呈现时，数据仅仅被写入到存储器中，不执行任何读取。

使用 send_frame_CABS 来从 CABS_代理 3801 调用读出阶段。数据然后被发送到 CABS_代理直到帧缓冲器被清空。一旦最后的帧数据被发送到 CABS_代理，则 FBIF 将 send_frame_CABS 信号设置为激活，同样在读出阶段中，控制器 3904 向清除和读出地址产生器 3902 和数据处理器 3905 提供开始信号。数据经由超级采样块 3901 被写入 CABS 接口 3801。

清除阶段的从调度器 3802 被调用的。控制器向清除和读出地址产生器 3902 发送开始信号，清除和读出地址产生器 3902 产生存储器的所有地址。所有的存储器位置将被写入用于黑色的 0。

数据处理器 3905 包括用于识别三个阶段并且据此动作的逻辑单元。在呈现阶段期间，MiPxCaW 来自其接口。只有 MiPxCaW 是灰色时，新的 MiPxCaW 才到来。如果 MiPxCaW 是白色的，则这作为地址组的隐含信息，并且不沿着数据路径发送任何微像素高速缓存。控制块 3904 告诉数据处理器 3905 当前数据是否是白色或灰色的。数据处理器使用上述的逻辑单元来执行一个或多个逻辑或/与非运算。一般，读取存储器以看所存储的数据是否是白色、灰色或黑色的；执行逻辑或/与非，并且新的数据被写入存储器中。为了当读取来自存储器的数据时补偿在 ZBT 存储器 3805 和在 ZBT 存储器控制器 3804 中的流水线中的延迟，在数据路径中和在数据处理器中的控制路径中存在延迟函数 3906。在读出阶段期间，延迟被禁止。在清除阶段期间，数据处理器向存储器写入 0。

地址产生器 3903 响应于执行的当前状态。在呈现阶段中，通过从 MPCB 向 FBIF 发送的地址组来提供地址。控制块读取所述地址组，然后它向地址产生器发送关于下列的信息：关于在窗口中的当前位置的信息；来自 MPCB 的地址组可以被定位在上/下或在下/上方向。控制块告诉地址产生器是否它应当递增或递减计数。一个完全呈现窗口是 $\text{Rend_win_ln_x} * \text{Rend_win_ln_y}$

MiPxCaW。呈现窗口以逐个行的方向被存储在帧存储器中。例如， $\text{Rend_win_ln_y} = 400$ 或 $\text{Rend_win_ln_x} = 200$ 。

地址 0 1 2 3 4 5 6 7 8 9 10 11 ... 199 200 201 202 ...										
数据	行 1					行 2				

在呈现阶段中，当呈现灰色数据时，每个第二时钟周期产生一个新的地址。当呈现白色数据时，数据仅仅被写入存储器中并且不被读取，因此每个时钟周期产生一个新的地址。控制信号 Rend_win_offs_x ， Rend_win_offs_y 包括关于在保护窗口中的呈现窗口的位置的信息，并且 Rend_win_offs_x 和 Rend_win_offs_y 告知关于呈现窗口的大小。这些用于其确定 MiPxCaW 是否位于保护窗口中或在呈现窗口中。在保护和呈现窗口中的所有的 MiPxCaW 被 FBIF 等同地处理。但是如果地址产生器检测到在呈现窗口之外的地址，则存储器使能信号将不是激活的，并且因此对应的 MiPxCaW 将不被写入存储器中。在读出和清除阶段期间，不使用这个逻辑块。

在呈现阶段期间不使用清除和读出地址产生器 3902。在读出阶段期间，数据被逐个行地写入到 CABS 接口，即用于两个连续数据的地址增量将是 Rend_win_ln_x ，除了当新行开始的时候。当新的行开始时， $\text{current_line_pointer}$ （当前行指针）将被递增 1。在消隐情况下的两个连续行之间，地址将按照消隐参数而被停止，参见下面的示例。对于参数 $\text{Rend_win_ln_y} = 400$ 、 $\text{Rend_win_ln_x} = 200$ 和消隐 = 2:

地址	0	1	2	3	...	199	199	199	200	...
行	行 1						消隐		行 2	
数据	D0	D1	D2	D3	...	D199	D199	D199	D200	...

在读出阶段，对于 100 M 像素/秒的读出速率，每个时钟周期产生一个新的地址，并且对于 50 M 像素/秒的读出速率，每两个时钟周期产生一个地址。为了以交错模式产生地址，在两个子单元发挥作用的情况下，一个偏移将被加到地址： $\text{Address_1} = \text{adr_gen}$ ； $\text{Address_2} = \text{adr_gen} + \text{rend_win_length_x}/2$ 。在交错模式中，存储器访问从地址 1 开始，在地址 1 和地址 2 之间交错。如果使用扩展区，则地址 2 将具有下列表达式： $\text{Address_2} = \text{adr_gen} + \text{rend_win_length_x}/2 - 2$ 。在清除阶段中。产生用于清除的、

从 0-256k 的所有存储器地址。(对于允许批清除功能的存储器设计可能不需要这个功能)。

地址延迟逻辑单元 3906 在呈现阶段期间被使能。旧的存储数据需要被读取以便使用新的 MiPxCaW 数据来执行或/与非。为了补偿在读取处理中在 ZBT 存储器和在 ZBT 存储器控制器中的流水线中的延迟,需要延迟数据和地址两者。地址延迟块延迟地址。数据的延迟被集成在数据处理器中。地址延迟逻辑单元在读出和清除阶段期间被禁止。

在这个实施例中,超级采样逻辑单元 3901 在读出阶段期间被使用,如图 7 所示。

FBIF 的第二和第三实施例分别利用被适配来具有 PMFB 和 GVFB 存储器的逻辑单元。如图 32 所示,多个像素图 3223 和灰度级 3224 存储器会是有益的。从下面的交错讨论来看,增加的存储器和用于存储器访问的独立信道的影响是明显的。像素图 PMFB 存储器作为访问限定器,用于限定访问用于特定像素的微像素存储器的需要。用于像素图符号修改的真值表依赖于所需要的呈现运算符,它可以是“或”或“与非”。在表格中的“*”表示任何符号值,表示“无关”或“通配符”。

或运算

之前

修改

之后

B	*	B	G	W	G
G	W	B	B	*	G
G	W	B	G	W	G

与非运算

之前

修改

之后

W	*	W	G	B	G
G	W	B	B	*	G
G	B	W	G	B	G

访问限制器可以是写或者修改写,所述写表示单个写操作,所述修改写表示读取-修改-写操作。对于或运算,当一个像素的之后值是 W 或 B 时,微像素阵列不需要被读取或写入。相反,像素图 PMFB 存储器被设置为适当的值。当之后值是灰色并且之前或修改值是黑色时,则单个写或根本没有写将产生在微像素 MPFB 存储器中的正确灰色值。当一个像素具有作为灰色的之前和修改值的时候,微像素阵列需要被读取、通过对之前和修改值应用逻辑或或者与非运算来被进行运算,随后向 MPFB 写入结果产生的值。本质上,

同一访问限制器可以向或运算一样被应用到与非运算。图 16D 图解了访问限制器的操作。矩形是像素的在前值。平行四边形是像素的修改值。阴影图指示在修改运算符之后的结果产生的像素图值（黑色，白色，灰色）。它示出了在应用逻辑或运算来使用平行四边形修改矩形之后微像素 MPFB 存储器进行了单个写入操作、读取-修改-写入操作、或者不被改变。

用于第二实施例的交错图是：

处理器	缓冲器	阶段 1	阶段 2
RP A	MPFB A	呈现	读出/超级采样
	PMFB A	呈现	读出/清除
RP B	MPFB B	读出/超级采样	呈现
	PMFB B	读出/清除	呈现

上述的第三实施例将涉及每次执行写入或读取-修改-写入操作时的超级采样。一个或多个灰色 GVFB 总计存储器将保存结果。所述第三实施例的两种变化形式使用在同一或独立数据路径上的像素图和灰度值存储器。第一交错图用于两个共享的 PMFB 存储器（在同一数据路径上、在同一存储器中的两个缓冲器）和两个共享的 GVFB 存储器（在同一存储器和数据路径中的两个缓冲器）。（在逐个行基础上交错的子阶段，每个读出行一组四个子阶段，对于整个阶段 512 个读出行）

		阶段 1		阶段 2	
处理器	缓冲器	子阶段 1	子阶段 2..4	子阶段 1	子阶段 2..4
RP A	MPFB A	空闲（que end）	呈现	空闲（que end）	呈现
	PMFB A1	-	呈现	读出/清除	-
	PMFB A2	读出/清除	-	-	呈现
	GVFB A1	-	超级采样	读出	-
	GVFB A2	读出	-	-	超级采样
RP B	MPFB B	空闲（que end）	呈现	空闲（que end）	呈现
	PMFB B1	-	呈现	读出/清除	-
	PMFB B2	读出/清除	-	-	呈现

	GVFB B1	-	超级采样	读出	-
	GVFB B2	读出	-	-	超级采样

Que end 表示在 MPFB 代理之前的 FIFO 缓冲器可以仍然被清空,即使 PM 代理正在忙于读取。这提供了负荷共享。

在第三实施例的第二种变化方式中,使用两个独立的 PMFB 和两个独立的 GVFB。每个存储器具有其本身的数据路径或用于存储器访问的独立信道。

处理器	缓冲器	阶段 1	阶段 2
RP A	MPFB A	呈现	呈现
	PMFB A1	呈现	读出/清除
	PMFB A2	读出/清除	呈现
	GVFB A1	超级采样	读出
	GVFB A2	读出	超级采样
RP B	MPFB B	呈现	呈现
	PMFB B1	呈现	读出/清除
	PMFB B2	读出/清除	呈现
	GVFB B1	超级采样	读出
	GVFB B2	读出	超级采样

在第二和第三实施例中,读出和清除阶段被组合。当读出 PMFB 和 MPFB 存储器的时候,像素图被清除。这迅速地发生,因为像素图以仅仅两个比特表示像素。不必有分立的读出和清除阶段。

图 42 描述了在第二实施例中的帧缓冲器接口的大致方框结构。所述结构视图与功能视图相反,给出了如何合并交错和流水线算法以便加速执行。离大数据总线并且在不互连的几个较小逻辑块中并行处理数据是有益的。将数据划分为几个较小块便于在高速数字器件中的物理实现。块划分适用于流水线操作。在产生地址族 4201 和微像素高速缓存 4202 中的 MPCB 3214 的功能如上所述。保护区过滤器作为保护区的一个实施例是 3641。像素图帧缓冲器接口 3643 访问 PMFB 存储器 3223 (见图 36)。微像素帧缓冲器接口 3642 访问 MPFB 存储器 3222 (也参见图 36)。

在这种配置中,保护区过滤器块 3641 与 MPCB(微像素高速缓存缓冲器) 3214、像素图帧缓冲器访问序列产生器 (PMFB AccSeqGen) 4221、微像素帧缓冲器 AccSeqGen 4222 和 MiPxCaW 缓冲器 4223 共享信号。需要下面的信

号来确定 MiPxCaW 是否位于呈现窗口内：Interior_info、MPCS_Y、MPCS_X1S、MPCS_X1E、MPCS_X2S 和 MPCS_X2E 是在 x 和 y 方向中相对于保护窗口的左角的坐标；Rend_win_offs_x 和 Rend_win_offs_y 指示呈现窗口的原点位置；Rend_win_ln_x 和 Rend_win_ln_y 指示呈现窗口的尺寸；一对交换信号（Downward_Ready/Up_Request），以便仲裁在这个块和 MPCB（微像素高速缓存缓冲器）4202 之间的数据流。

像素图帧缓冲器访问序列产生器 4221 与保护区过滤器 3641 和 PMFB 代理 4231 共享信号。它接收下列内容：来自保护区过滤器的(X, Y, interior_info)的地址组；一对交换信号（Downward_Ready/Up_Request），它们仲裁在这个块和保护区过滤器之间的数据流；地址和字节使能，对应于要在 PMFB 代理中处理的像素图组；用于指示存在可用于处理的数据的信号（PMFB AccSeqGen ->PMFB 代理）；用于指示 PMFB 代理准备好从 PMFB AccSeqGen 接收数据的信号。

PMFB 代理 4231 一次读取一个地址组（AS），并且产生 PMFB（像素图帧缓冲器）访问的序列。对于 X1S、X1E、X2S、X2E、Y 的每个 AS，将产生 PMFB 访问的序列。PMFB 访问包括通过字位置 [(0...Xsize/4-1), (0...Ysize/4-1)]和组编号[0...3]寻址的 PM 组访问。所述组编号将在 PMFB 代理中被用作字节使能信号。

图 43 图解了可以用于实现本发明的方面的一个存储器结构。在这个结构中，像素图符号具有 W = 白色、G = 灰色、B = 黑色的值，像素图组包括被排列为 8 比特字的四个像素图符号，一个像素图字包括四个像素图组，它被布置为一个 32 比特的字。在这个存储器结构中支持下面的 RMW 程序：读取 8 比特字（丢弃 24 个剩余比特），使用掩模（字节使能）来写入，读取 32 比特字，写 32 比特字。再次参见图 43，假定行 8 被 PMFB AccSeqGen 4221 处理。将产生三个存储器访问的序列。

	序列 1	序列 2	序列 3
PM 组	GGWW	WWWW	GGBB
字节使能	0	0	0
对应的存储器字位置	(1, 2)	(2, 2)	(3, 2)

PMFB 代理 4231 构造一个一般像素图容器来处理 PM 组或 PM 字的中间

存储。所述容器包括一个到四个符号组和一个字地址。所述容器具有一个像素图组输入项：

	组阵列	组有效	运算符
组 0	SSSS	v	op
组 1	SSSS	v	op
组 2	SSSS	v	op
组 3	SSSS	v	op

s = 像素图符号[W, G, B]

v = 有效状态[真，伪]

op = 运算符[或，与非]

示例：

	组阵列	组有效	评述
组 0	----	伪	在组 0 上无修改
组 1	----	伪	在组 1 上无修改
组 2	WWWG	伪	v
组 3	----	伪	v

	组阵列	组有效	评述
组 0	----	伪	在组 0 上无修改
组 1	WWGG	真	以符号 WWGG 修改组 1
组 2	WWWG	真	以符号 WWWG 修改组 2
组 3	----	伪	V

PMFB 代理 4231 与 ZBT 控制器 3804A、PMFB-AccSeqGen 4231 和访问限制器 FIFO 4232 共享信号。在 PMFB AccSeqGen 和 PMFB 代理之间的接口如上所述。在 AccQua FiFo 4232 和 PMFB 代理 4231 之间的接口包括向 FiFo

发送访问限制器。为了使得可以写入新的数据，需要一个写使能以及 FiFo 状态标志，例如 `fifo_full` 或 `fifo_overrun`。在 MPFB 代理 4231 和 ZBT 控制器 3804A 之间的接口包括：地址、数据输入、数据输出、读写和字节使能信号。另外，需要在一般容器中包括的呈现窗口逻辑运算符（RWLOper）来确定使用那个逻辑运算符来执行呈现处理。

MPFB AccSeqGen 4222 用于产生对应于在 MiPxCaW 缓冲器 4223 中存储的 MiPxCaW 的地址。从 Y 值、X1S、X1E、X2S 和 X2E 向外产生地址。S 指示开始，E 指示结束。下面是地址组的逻辑规则：

$X1S < X1E$: MiPxCaW 被存储在 MPCB 中，其中在从 X1S 到 X1E 的区域中首先出现最低地址数据。（左边）

$X1S > X1E$: MiPxCaW 被存储在 MPCB 中，并且在从 X1S 到 X1E 的区域中首先出现最高地址数据。（左边）

$X2S < X2E$: MiPxCaW 被存储在 MPCB 中，其中在从 X2S 到 X2E 的区域中首先出现最低地址数据。（右边）

$X2S > X2E$: MiPxCaW 被存储在 MPCB 中，其中在从 X2S 到 X2E 的区域中首先出现最高地址数据。（右边）

而且，这个块必须向上、即向保护区过滤器指示：一旦它已经向 MPFB 代理 4233 发送了对应于前一个接收的地址组的最后一个地址，它就准备好接收一个新的地址组。

MPFB AccSeqGen 4222 具有与保护区过滤器 3641 和与微像素帧缓冲器代理 4233 的公共接口。保护区过滤器根据请求发送地址组：X1S、X1E、X2S、X2E 和 `interio_info` 标记。一对交换信号（`Downward_Ready/Up_Request`），仲裁在这个逻辑块和保护区过滤器之间的数据流。在 MPFB 代理和 MFB 访问序列产生器块 4222 之间的接口，传送要用于对应的 MiPxCaW（存储在 MiPxCaW 缓冲器中）的地址以及数据流仲裁信号。

MiPxCaW 缓冲器 4223 仅仅是一个 FIFO 缓冲器。它容许在这个设计所需要的高时钟频率（100 MHz），并且一般是 FloorPlanner 友好的。需要提供四个 DPRAM 块来存储最大的几何图形（253 MiPxCaW），即 64 比特的字被划分为四个 16 比特的字。MiPxCaW 缓冲器 4223 具有与保护区过滤器和 MPFB 代理的公共接口。在保护区过滤器和这个块之间的接口，需要到缓冲器的 MiPxCaW 流和写使能信号来提供 FiFo。在 MPFB 代理和这个块之间的

接口，需要来自缓冲器的 MiPxCaW 流和读使能信号来从 FiFo 获取数据。

MP 帧缓冲器代理 4233 在逐个 MiPxCaW 的基础上对 MPFB 3222 执行操作，即一次一个 64 比特的字。所述代理根据访问限制器 AccQua 的值来执行写入或读取-修改-写入操作。用于 MiPxCaW 修改的真值表依赖于逻辑呈现运算符，它可以是“或”或者“与非”。如果用于一个给定对（地址，MiPxCaW）的 AccQua 是写入，则 MiPxCaW 仅仅需要被写入对应的地址。但是如果 AccQua 是修改-写入，则必须首先从由所述地址指示的位置获取存储在 MPFB 中的值。其后，可以按照呈现窗口运算符来修改数据。但是，在等待要被检索的数据的同时必须在队列中存储数据和地址。在所述代理向 ZBT 控制器 3804 发送地址和控制信号的时刻和在 Dout 总线上可以获得对应数据的时刻之间存在一个等待时间（大约 10 个时钟周期）。这意味着如果例如在 MPFB 中先后写入两个重叠几何图形，则会发生高速缓存一致性争用。这可以使用监控用于访问存储器的地址的 CAM 结构来解决。一种可能的实现方式是：如果一个地址在 CAM 结构中出现两次，则存储最后 10 个地址以发出匹配指示符。这意味着在操作中的暂时中断直到匹配指示器消失。

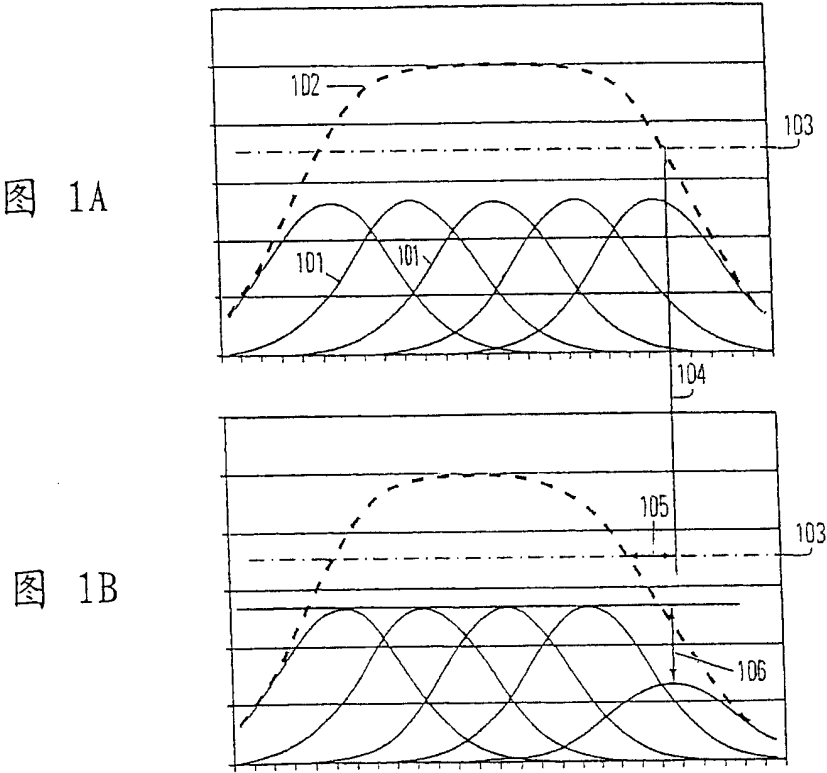
微像素帧缓冲器代理 4233 逻辑块具有与 AccQua FiFo 4232、MPFB AccSeqGen 4222、MiPxCaW 缓冲器 4223 和 ZBT 控制器 3804B 的公共接口。对于 FiFo、缓冲器和地址产生器公共的读取使能信号触发释放新的 AccQua、地址和 MiPxCaW。在 FiFo 和这个块之间的接口，存在 AccQua 和 FiFo 状态指示符，诸如 FiFo 空标记。在 MPFB AccSeqGen 和这个块之间的接口，存在一个地址总线 and 一对仲裁信号，例如准备好/进行。在 MiPxCaW 缓冲区和这个块之间的接口，存在 MiPxCaW 总线 and 一对仲裁信号，例如准备好/进行。有可能所述这对仲裁信号与在这个块和 MPFB AccSeqGen 之间使用的相同。最后，对于 PMFB 代理，在 ZBT 控制器和这个块之间的接口如上所述。

读出逻辑单元 3644 具有与帧缓冲器（经由 ZBT 控制器）和 CABS 总线的公共接口。

定时策略需要使用在 100 MHz 的双频的 clk_c2 时钟来定时 FBIF 的逻辑块。读出逻辑单元也使用 clk_c1 时钟来被定时，所述 clk_c1 时钟具有 50 MHz 的基本频率。可以使用两个时钟频率来驱动使用 clk_c1 时钟的 CABS 接口和使用 clk_c2 时钟的 ZBT 控制器。两个时钟产生具有相同的 DLL，这意味着应当可以忽略变形。即，两个时钟可以被考虑为同步时钟。每当在那两个时

钟域中进行过渡时，可以考虑两种手段：或者仅仅使用 clk_c2 时钟并当需要时使用多个时钟周期路径，或者使用 clk_c2 和 clk_c1 两者。

虽然已经参照上面详细所述的优选实施例和示例公开了本发明，但是可以理解，这些示例意欲是说明性的而不是限定性的。可以认识到，本领域的技术人员可以容易地进行修改和组合，所述修改和组合将在本发明的精神和所附的权利要求的范围内。



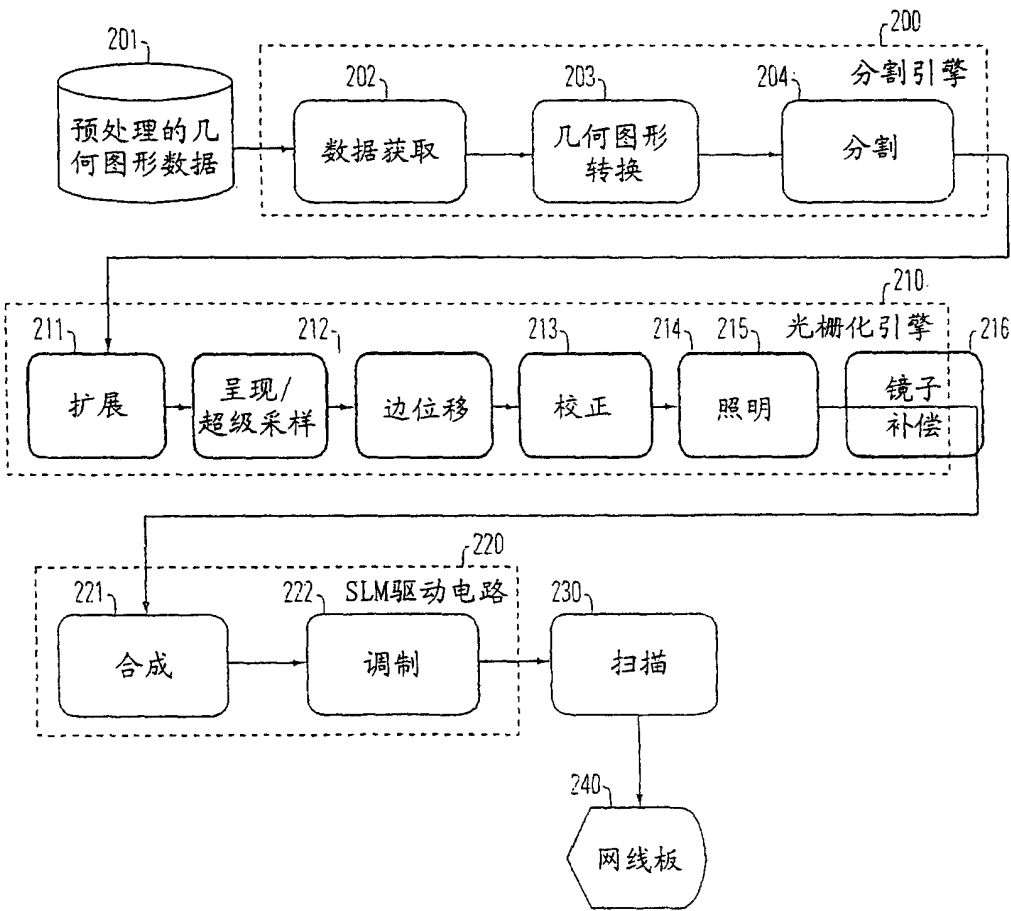


图 2

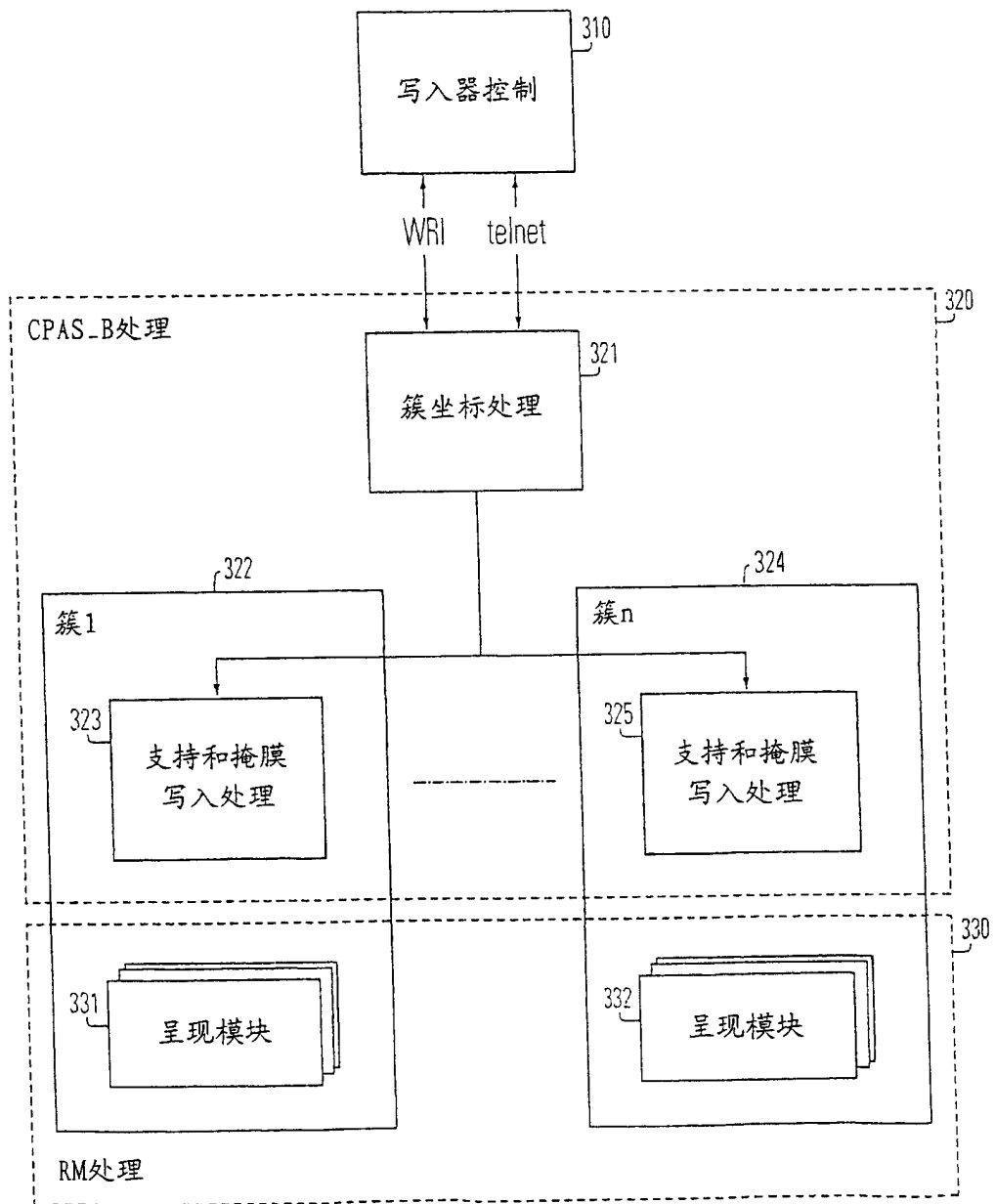


图 3

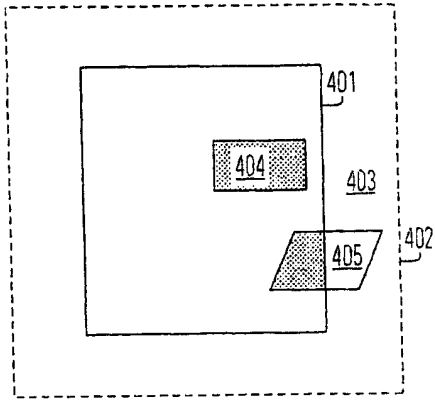


图 4A

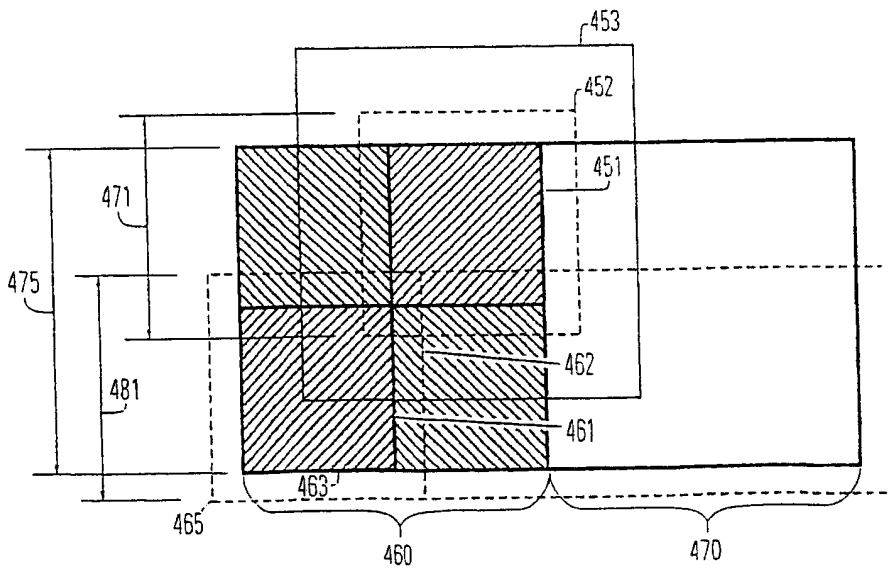


图 4B

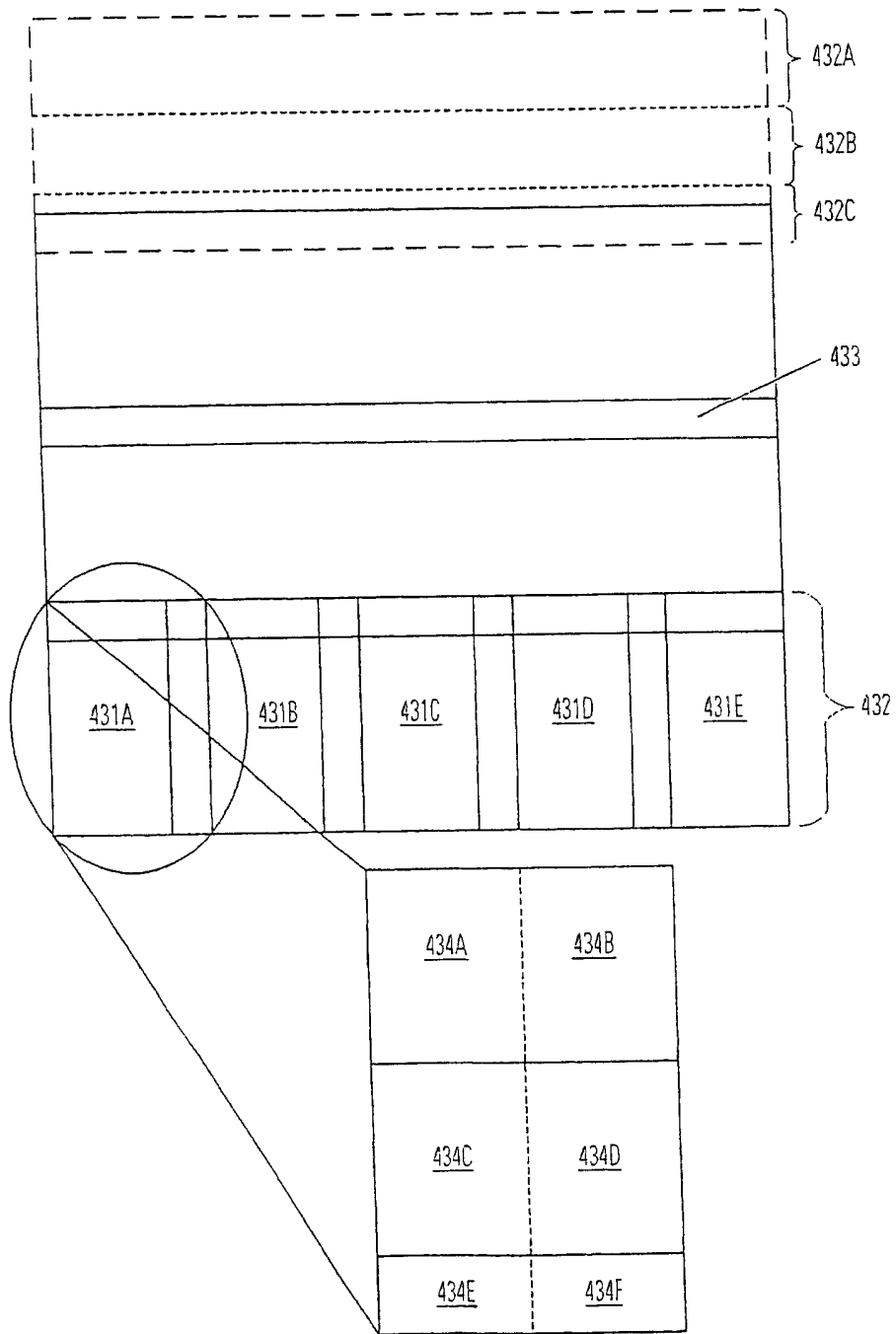


图 4C

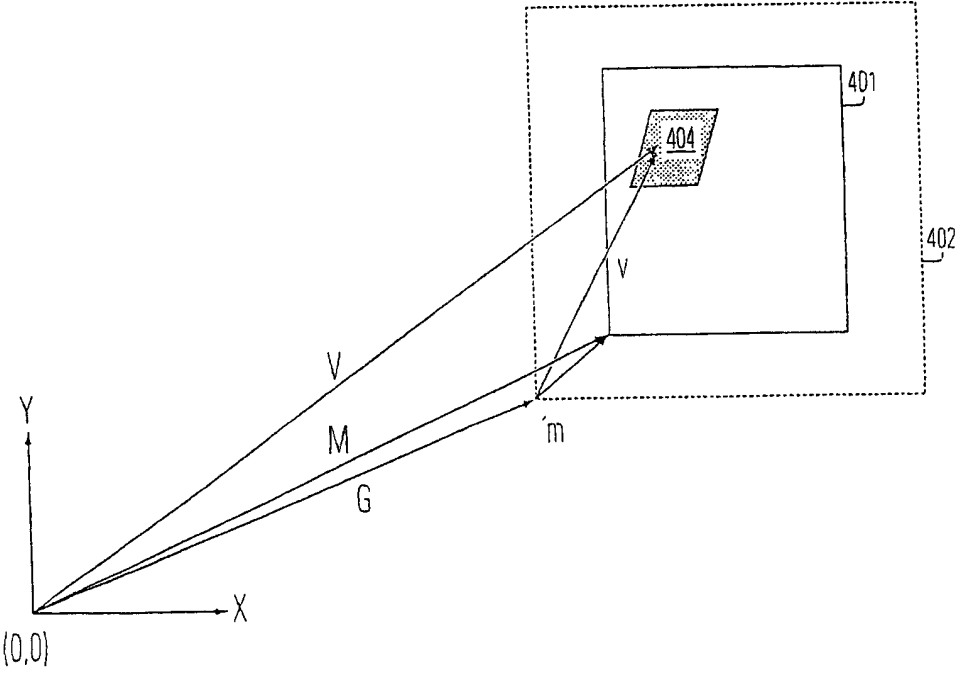


图 4D

图 5A

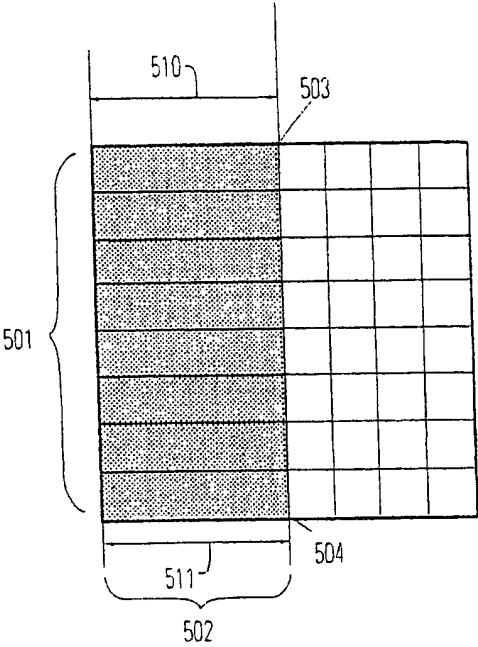
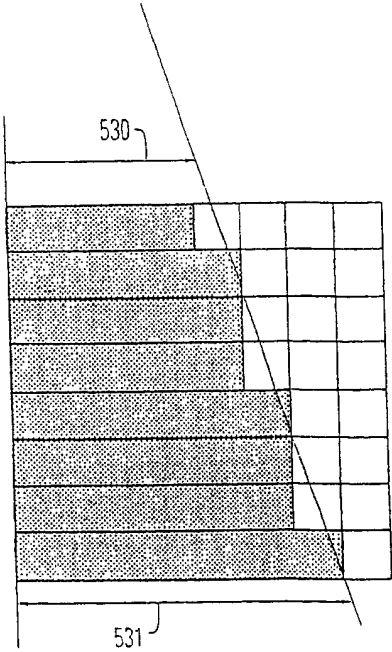


图 5B



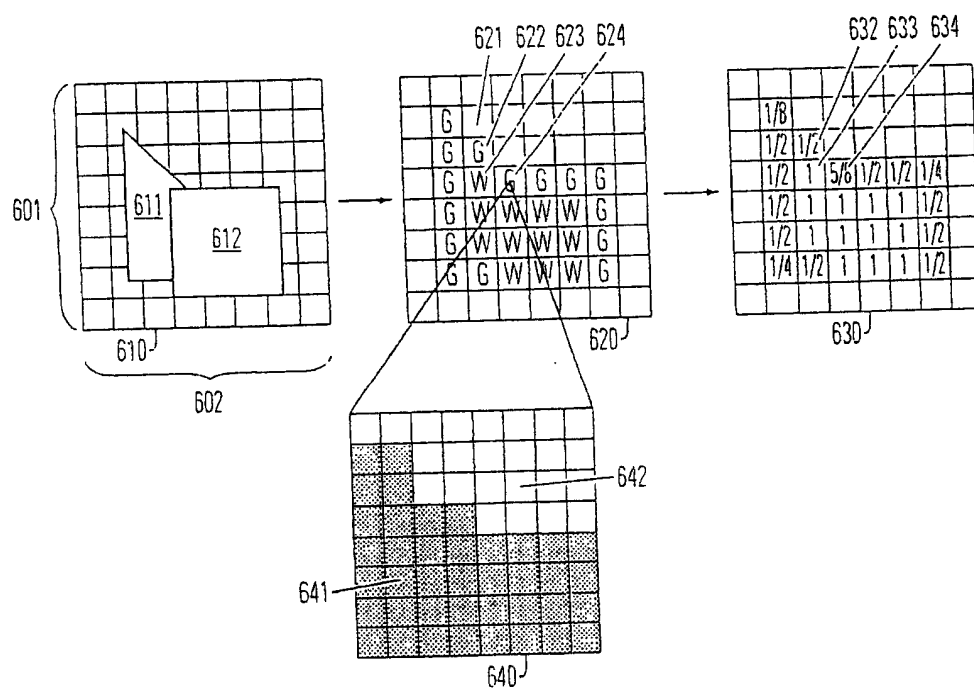


图 6

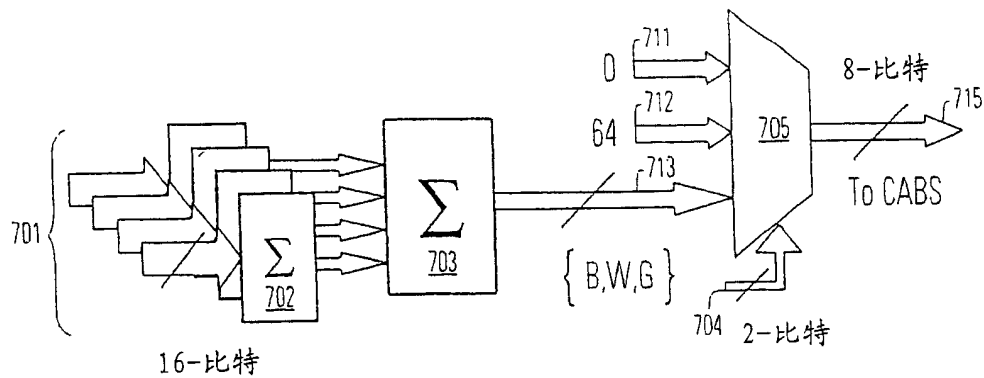


图 7

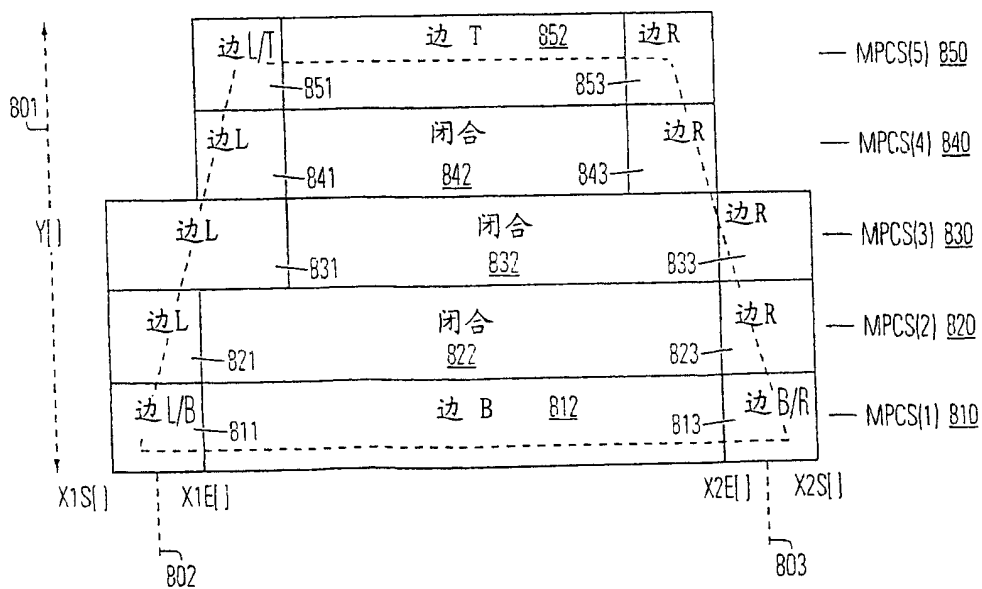


图 8

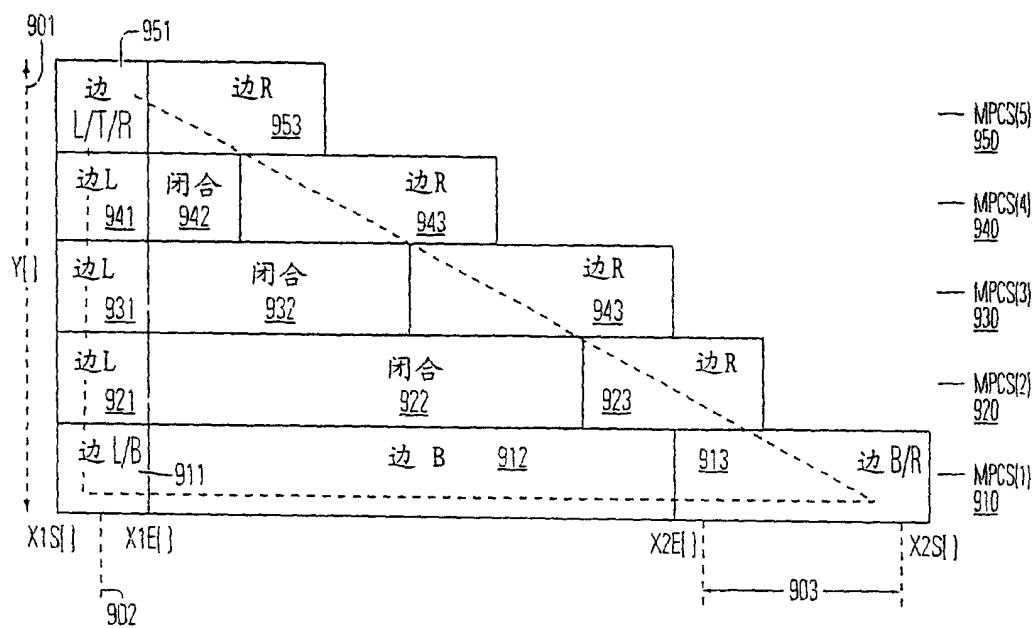


图 9

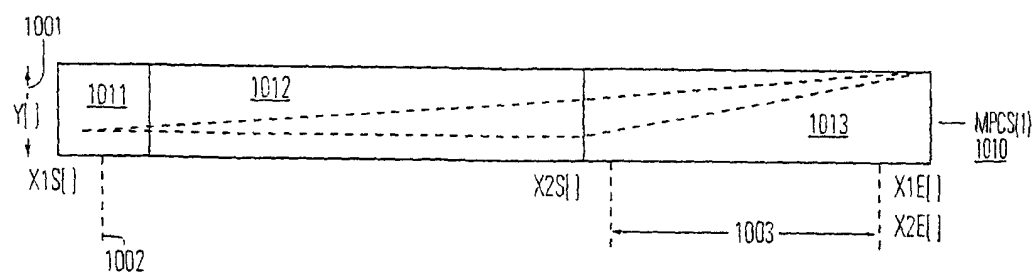


图 10A

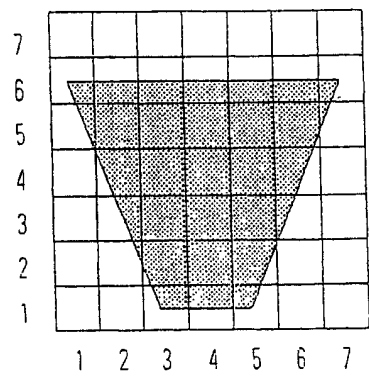


图 10B

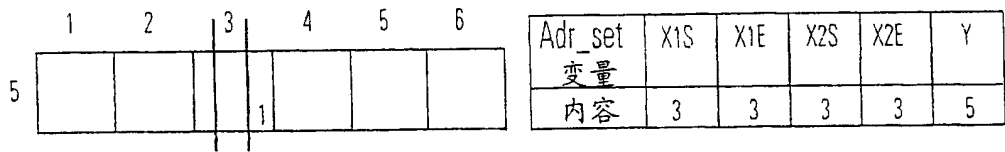


图 10C

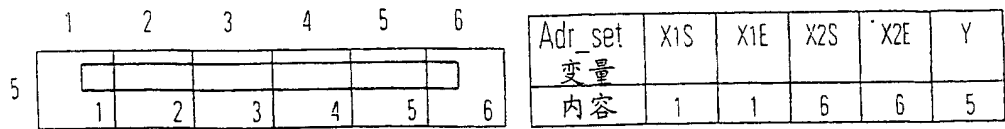


图 10D

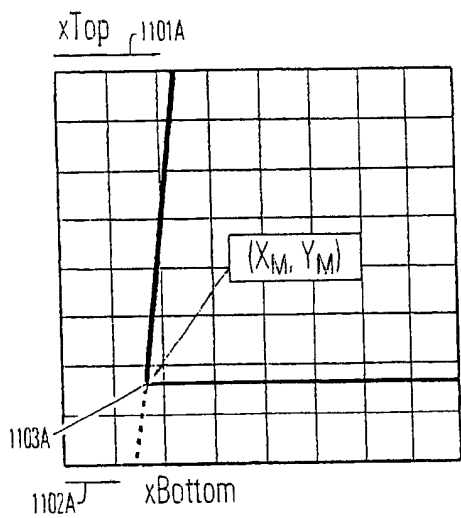


图 11A

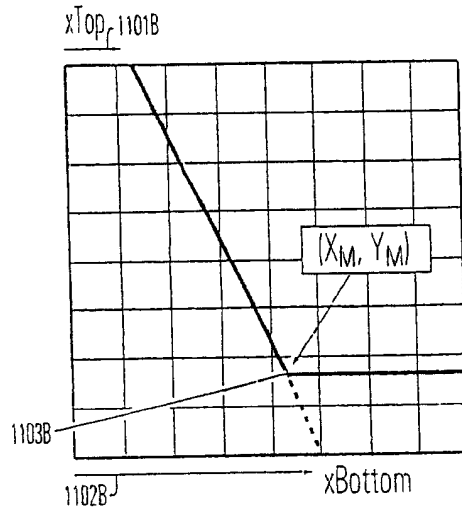


图 11B

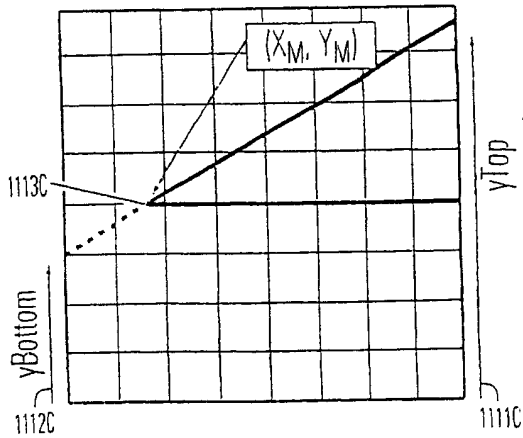


图 11C

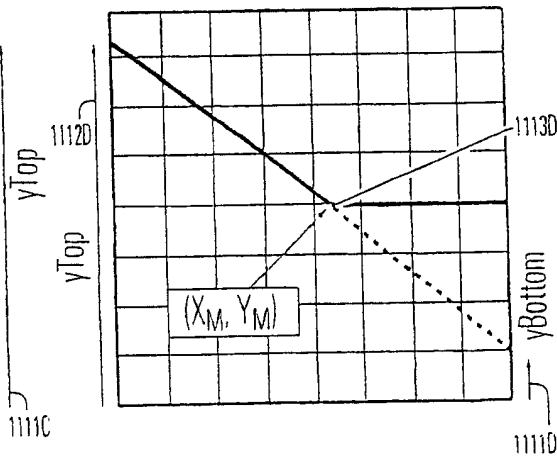


图 11D

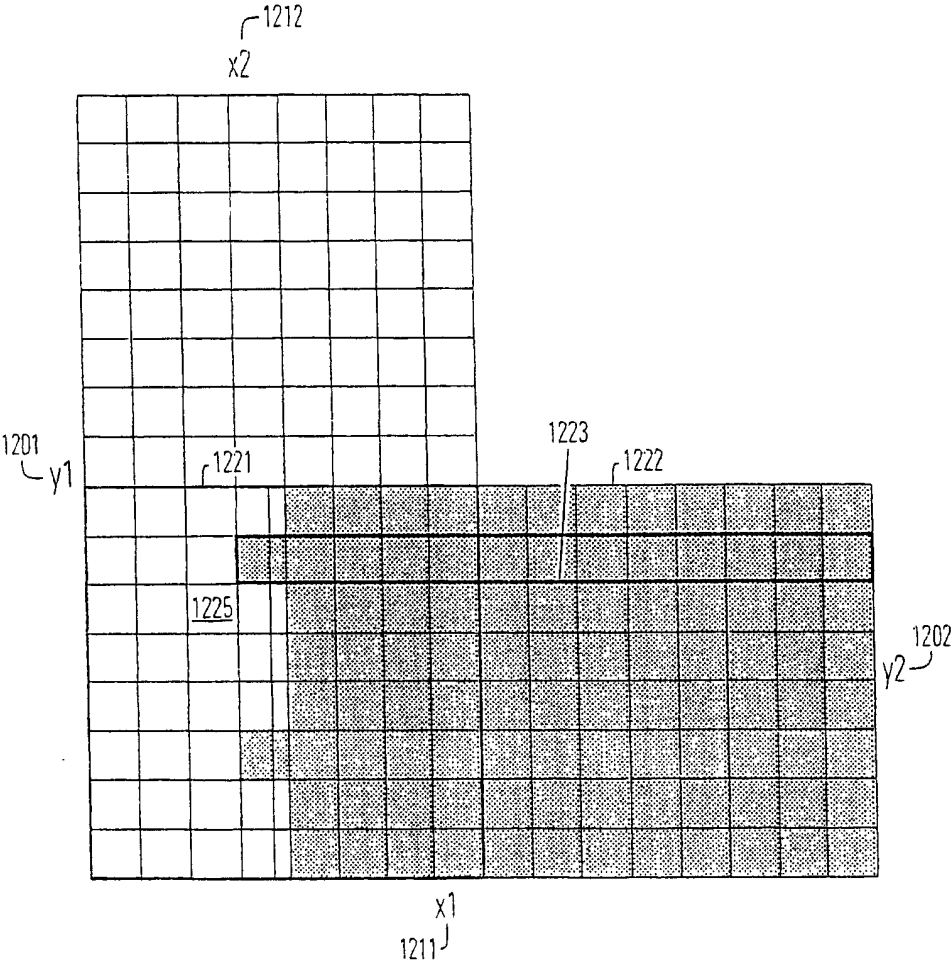


图 12

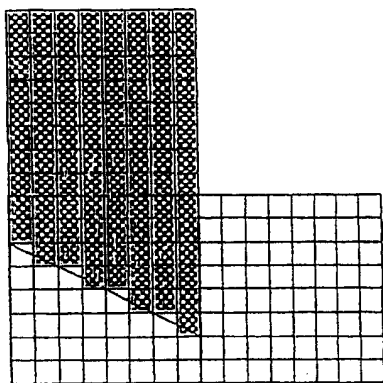


图 13A

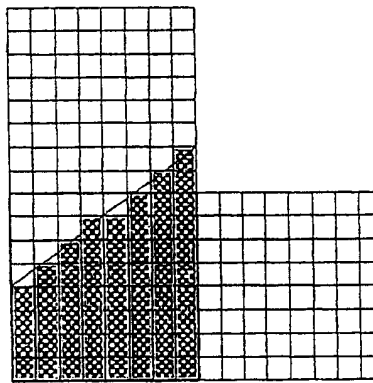


图 13B

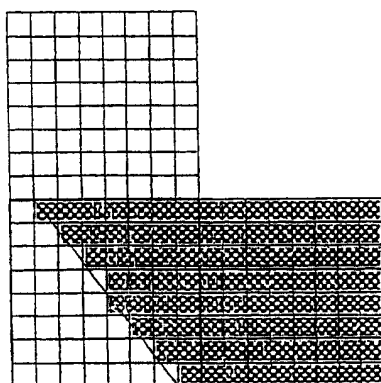


图 13C

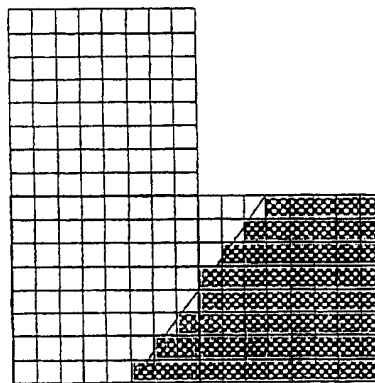


图 13D

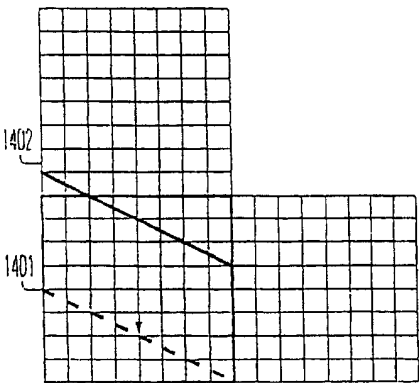


图 14A

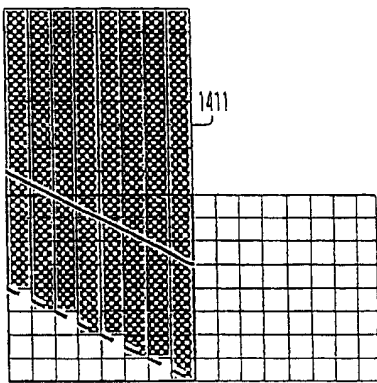


图 14B

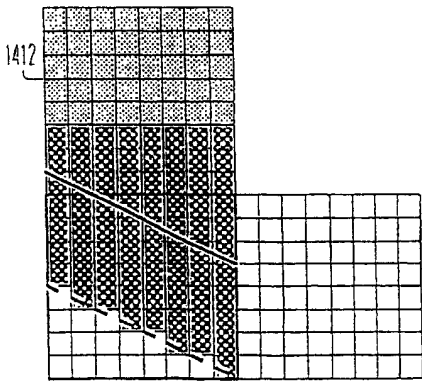


图 14C

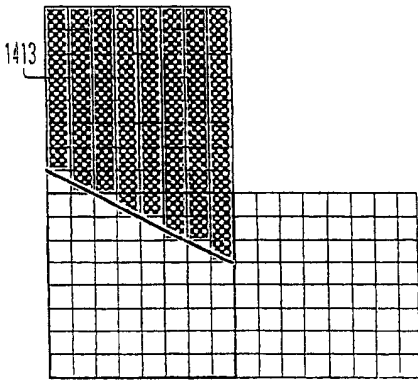


图 14D

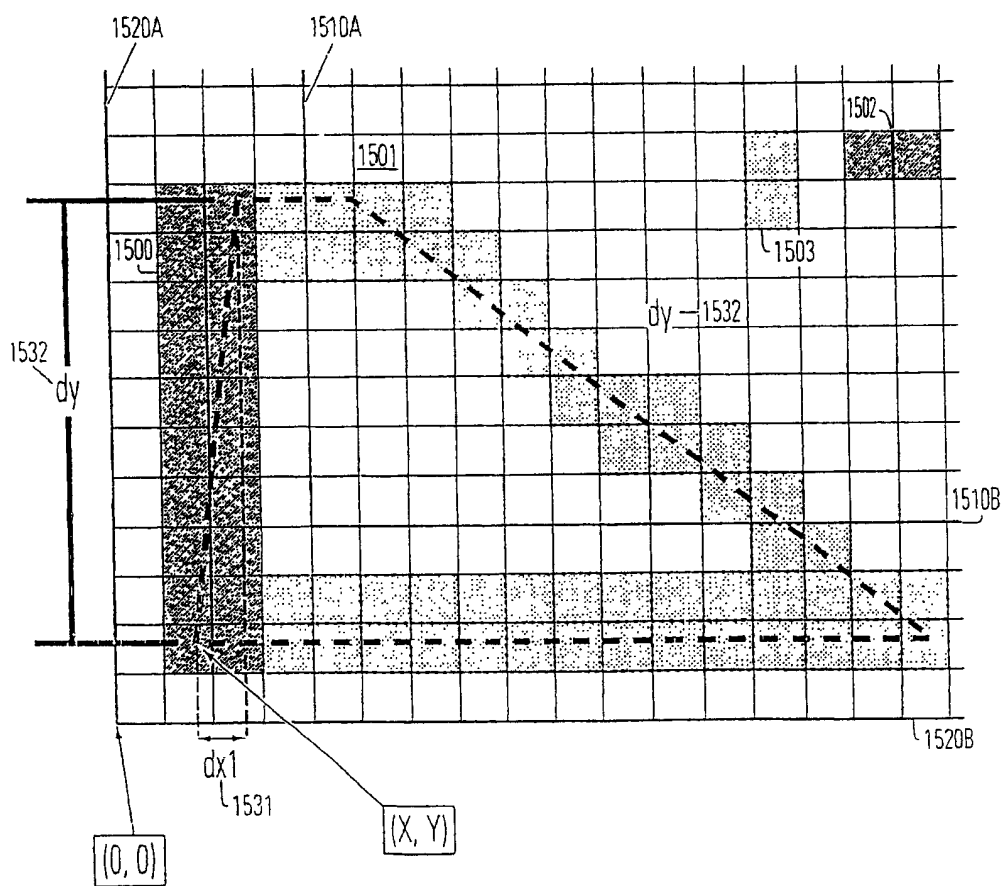
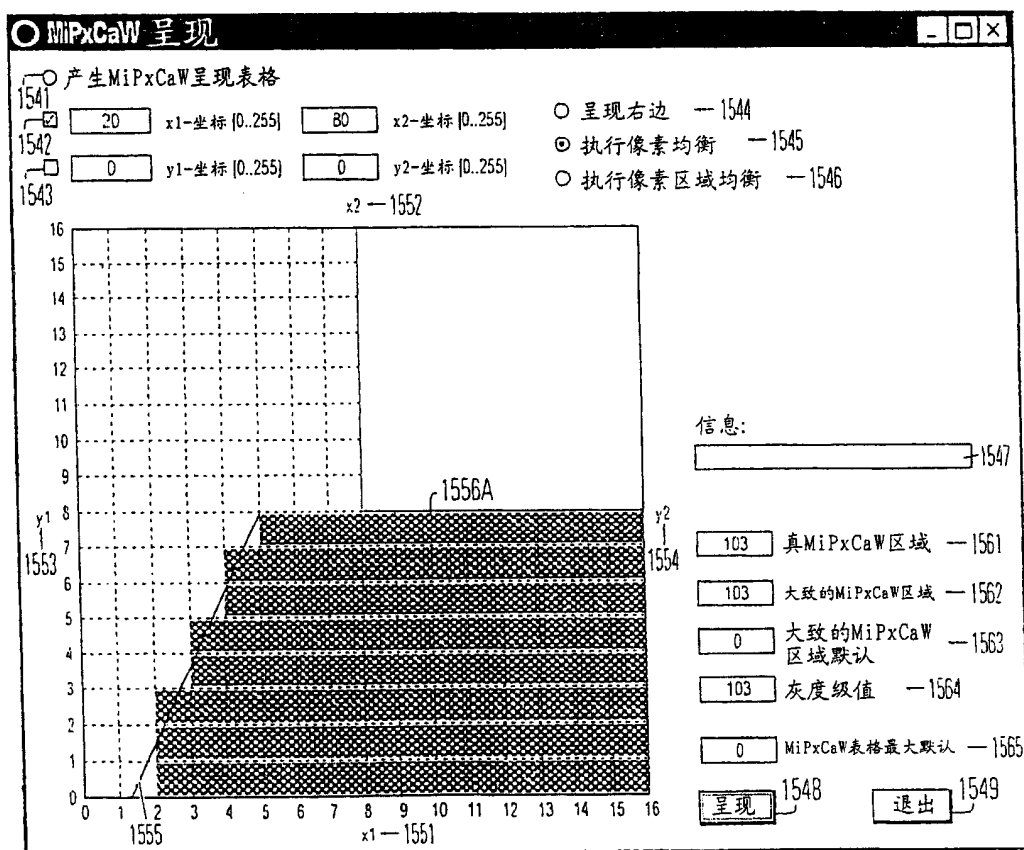


图 15A



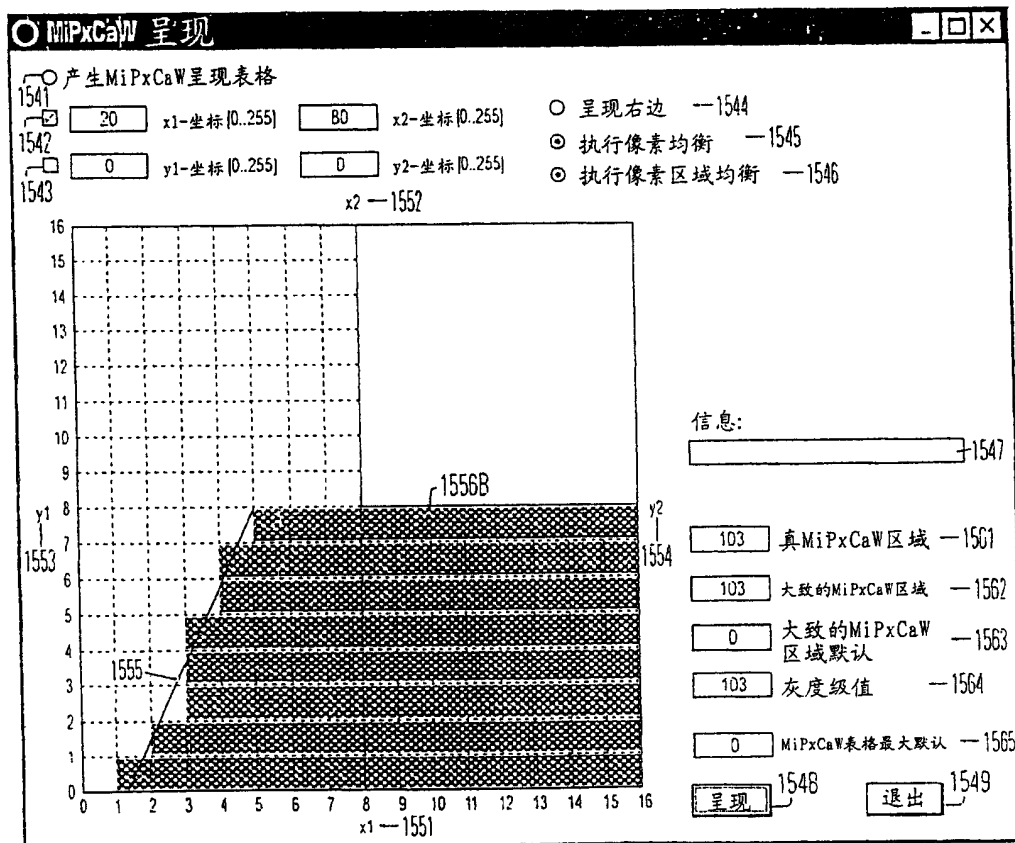


图 15C

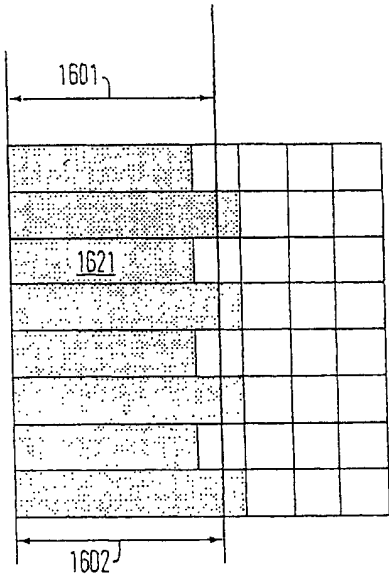


图 16A

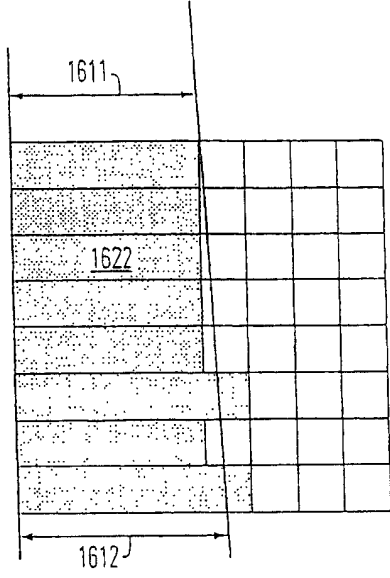


图 16B

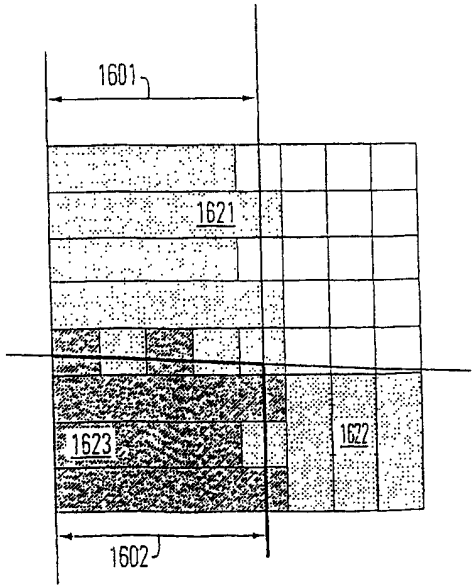


图 16C

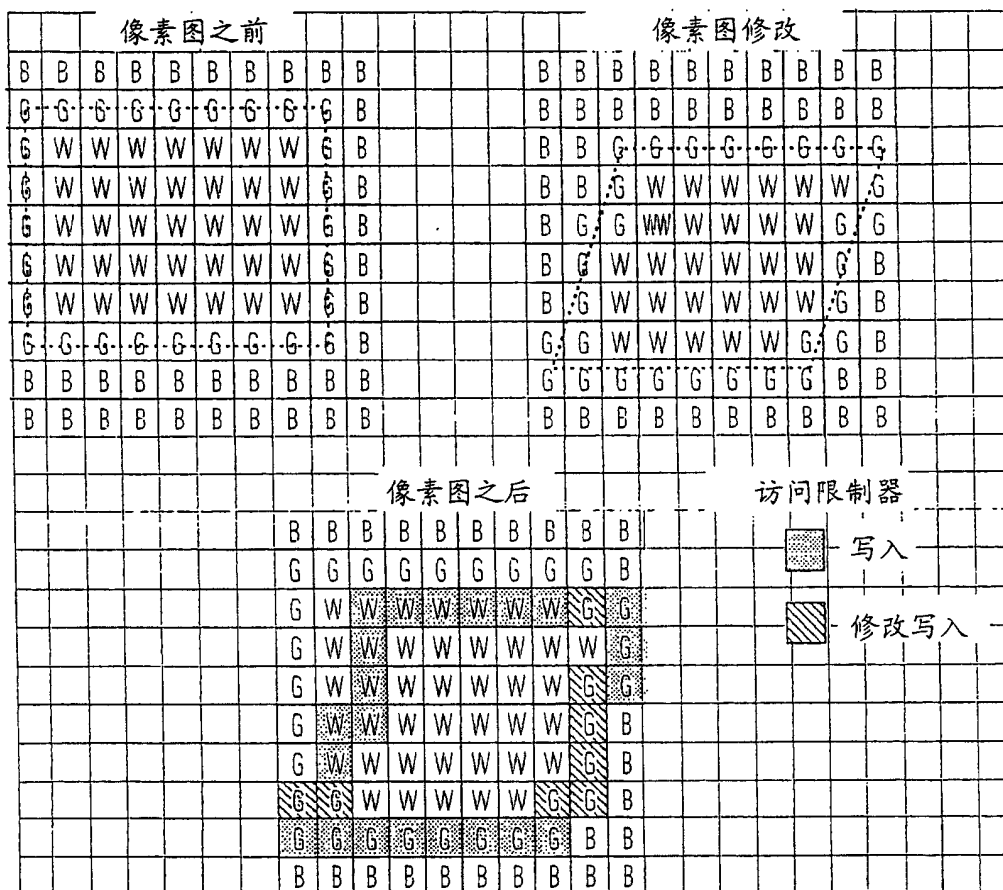


图 16D

图 17

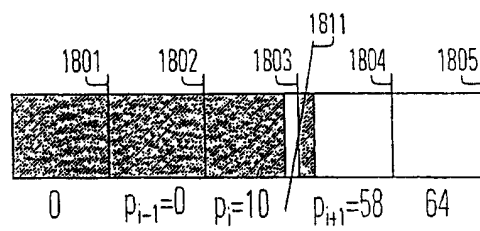
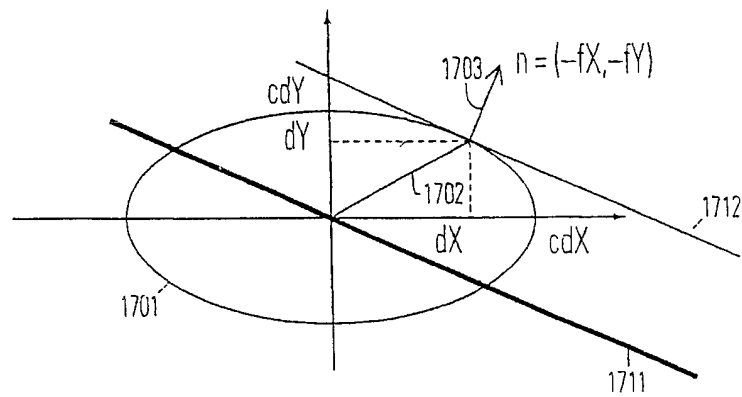


图 18A

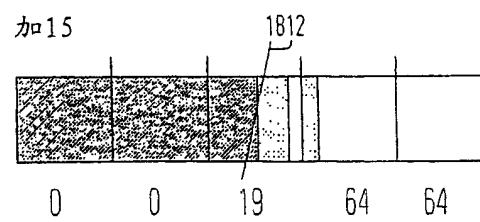


图 18B

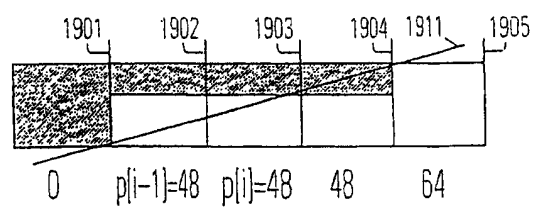


图 19A

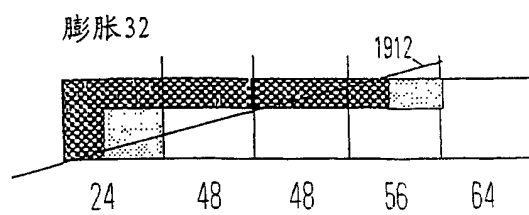


图 19B

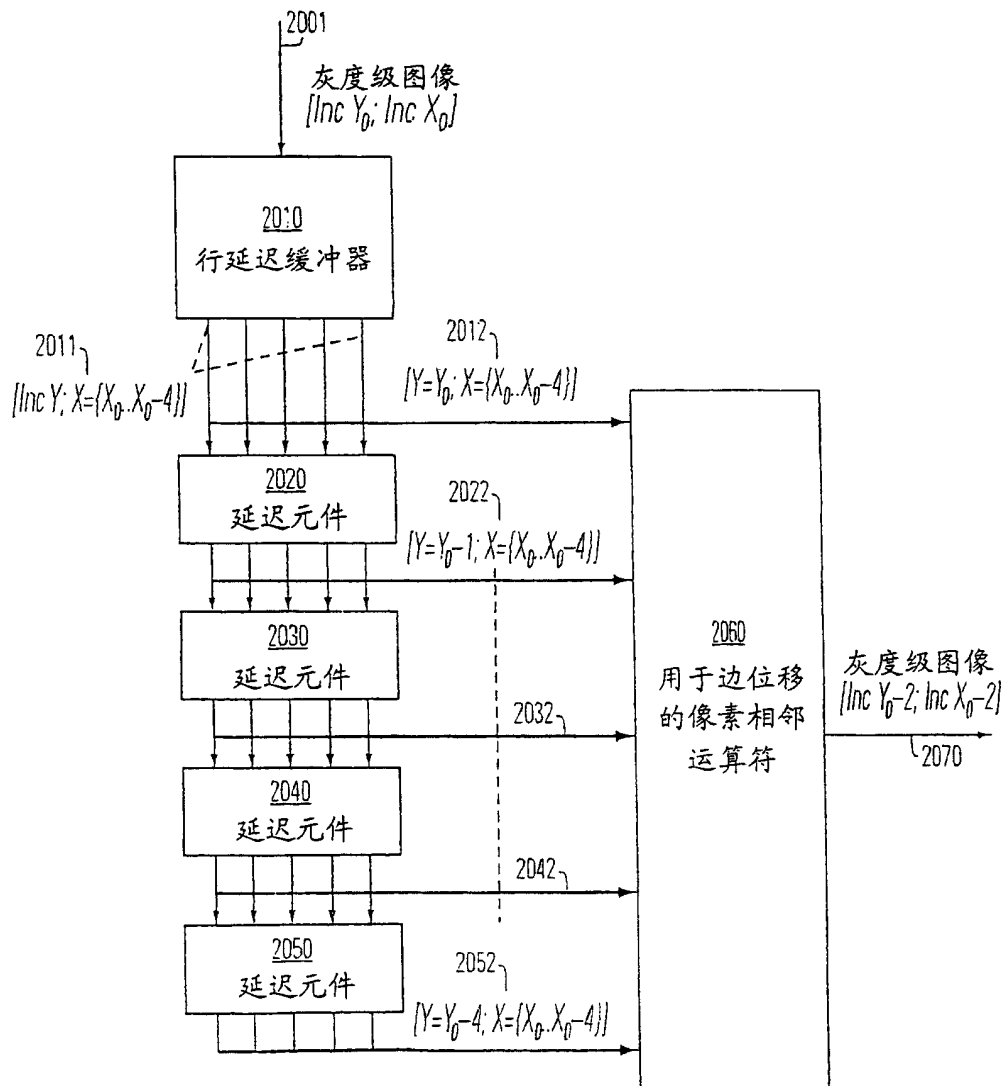


图 20

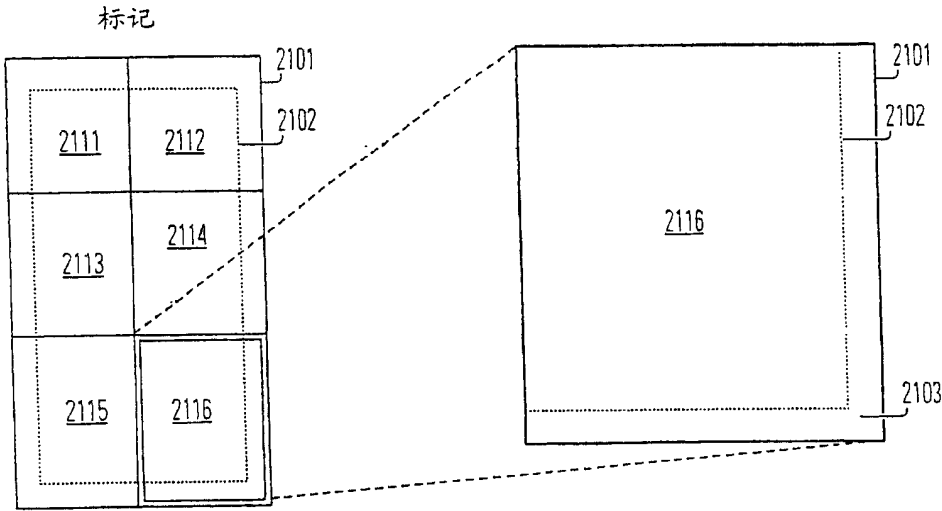


图 21

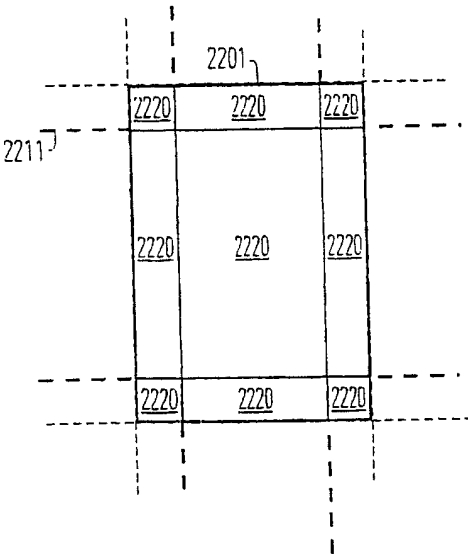


图 22

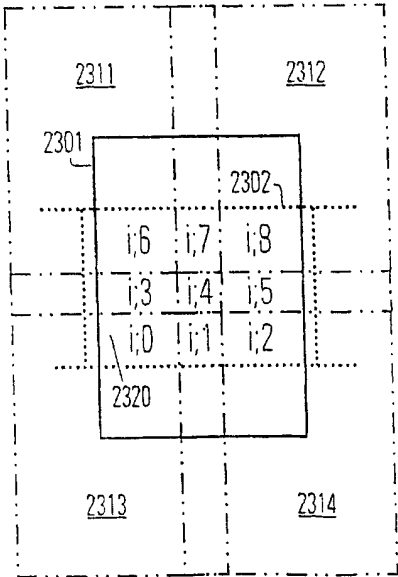


图 23A

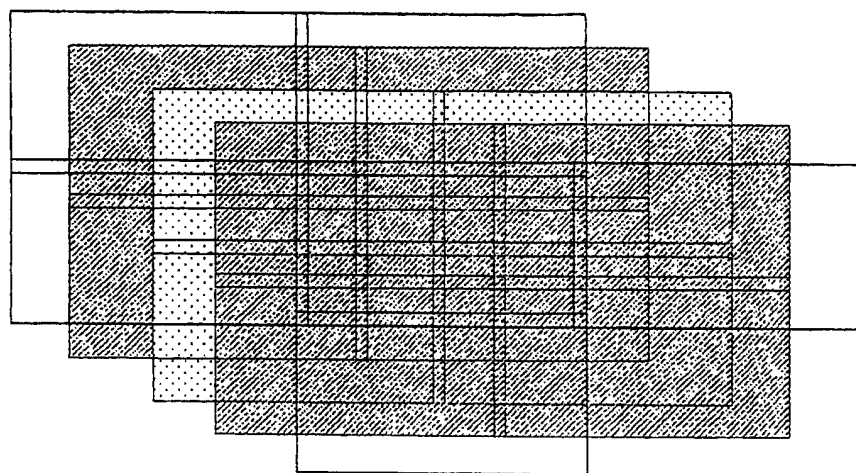


图 23B

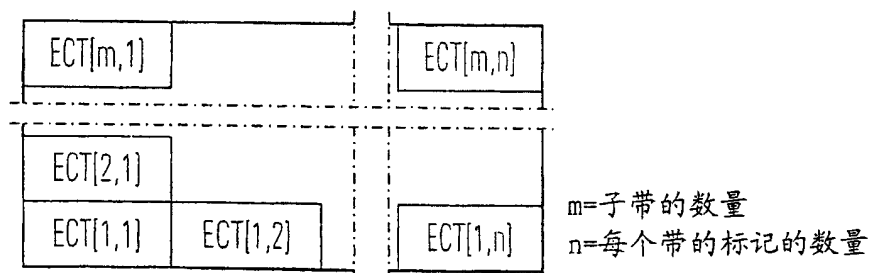


图 23C



图 23D

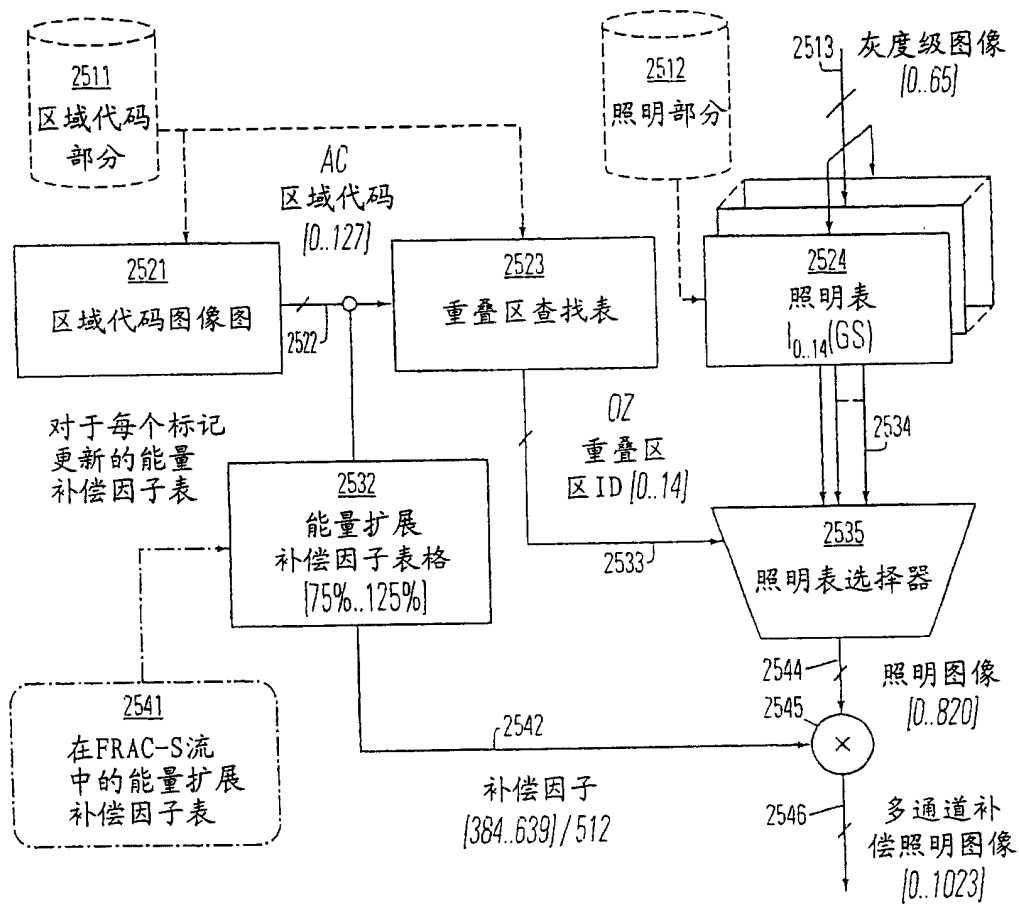


图 25

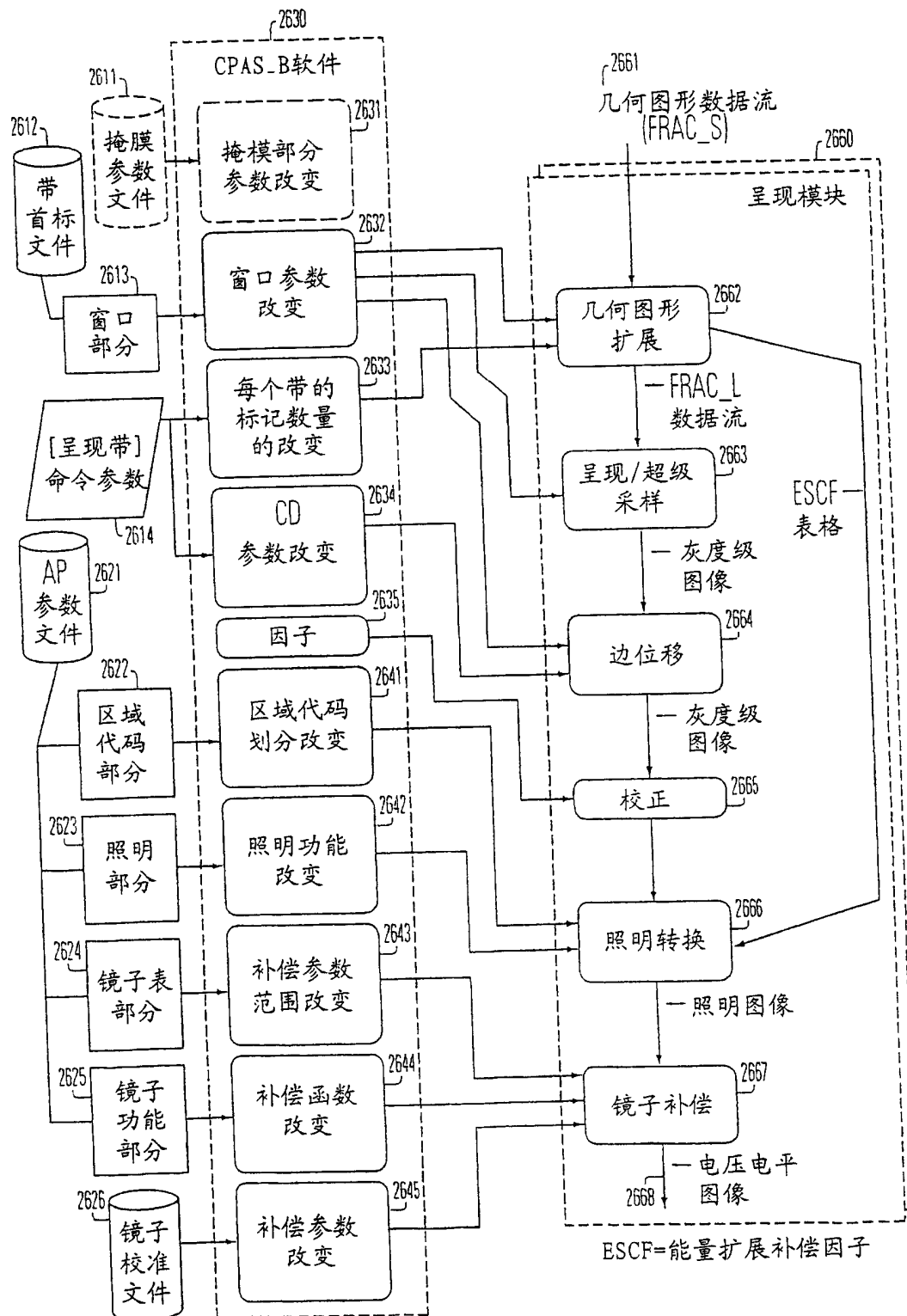


图 26

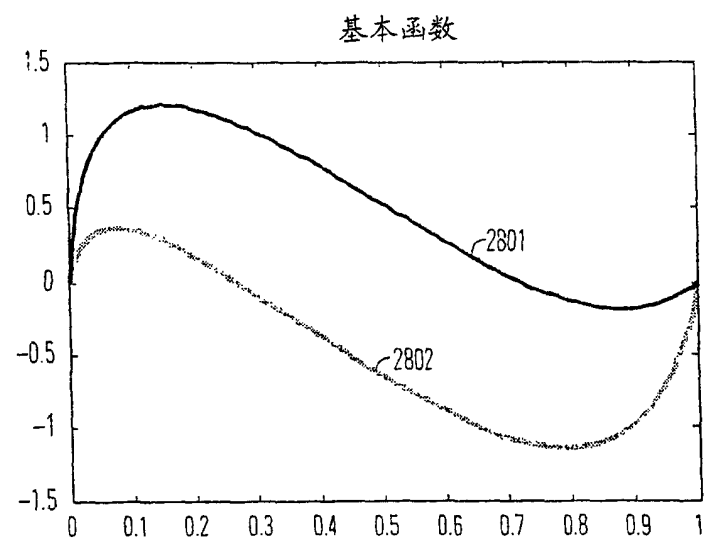


图 28

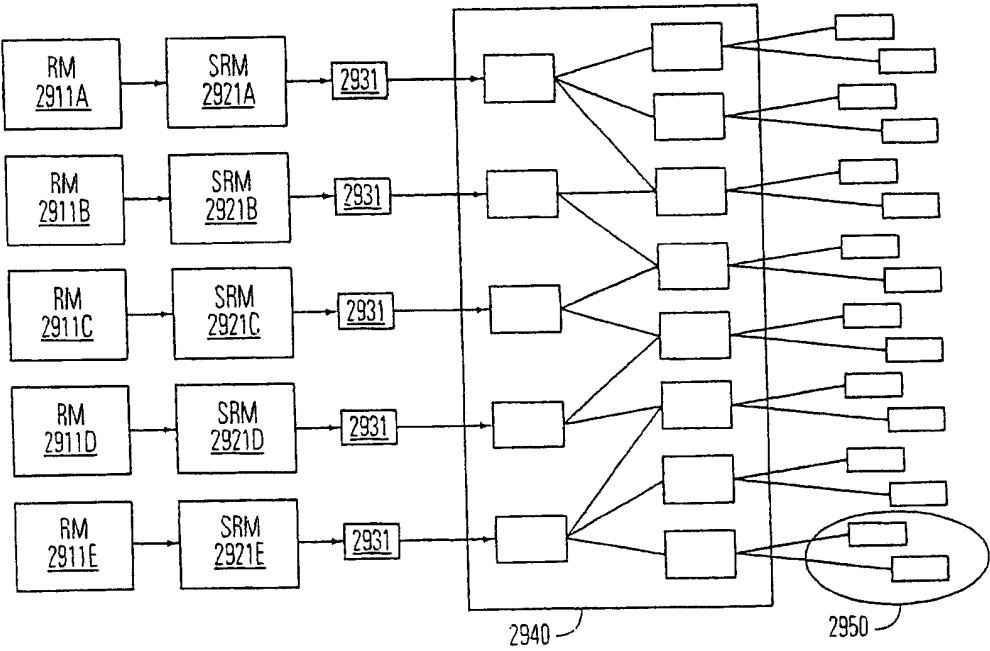


图 29

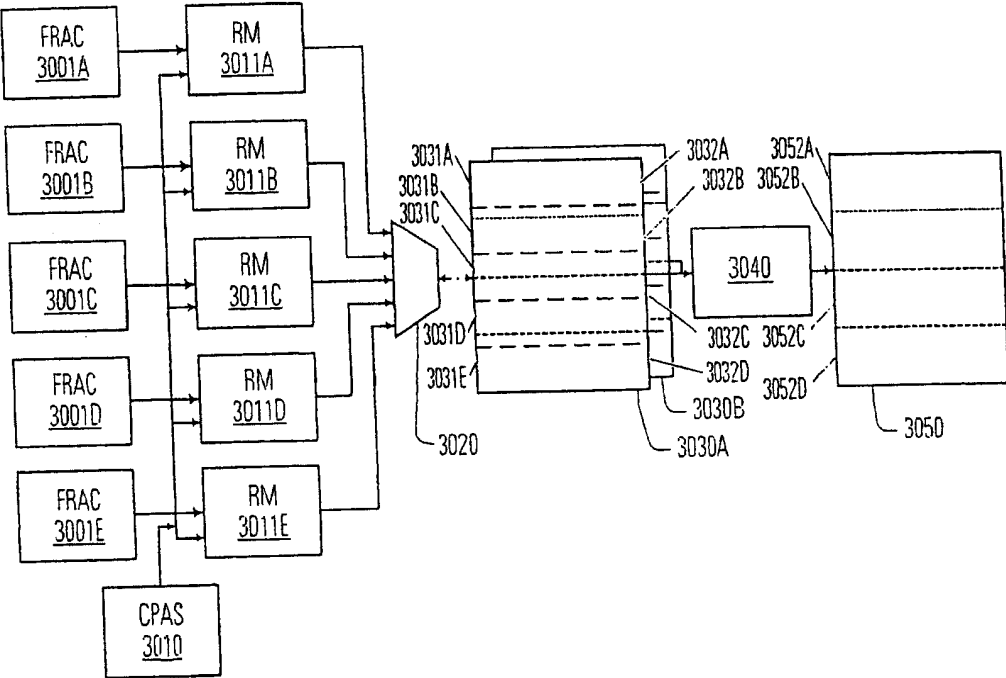


图 30

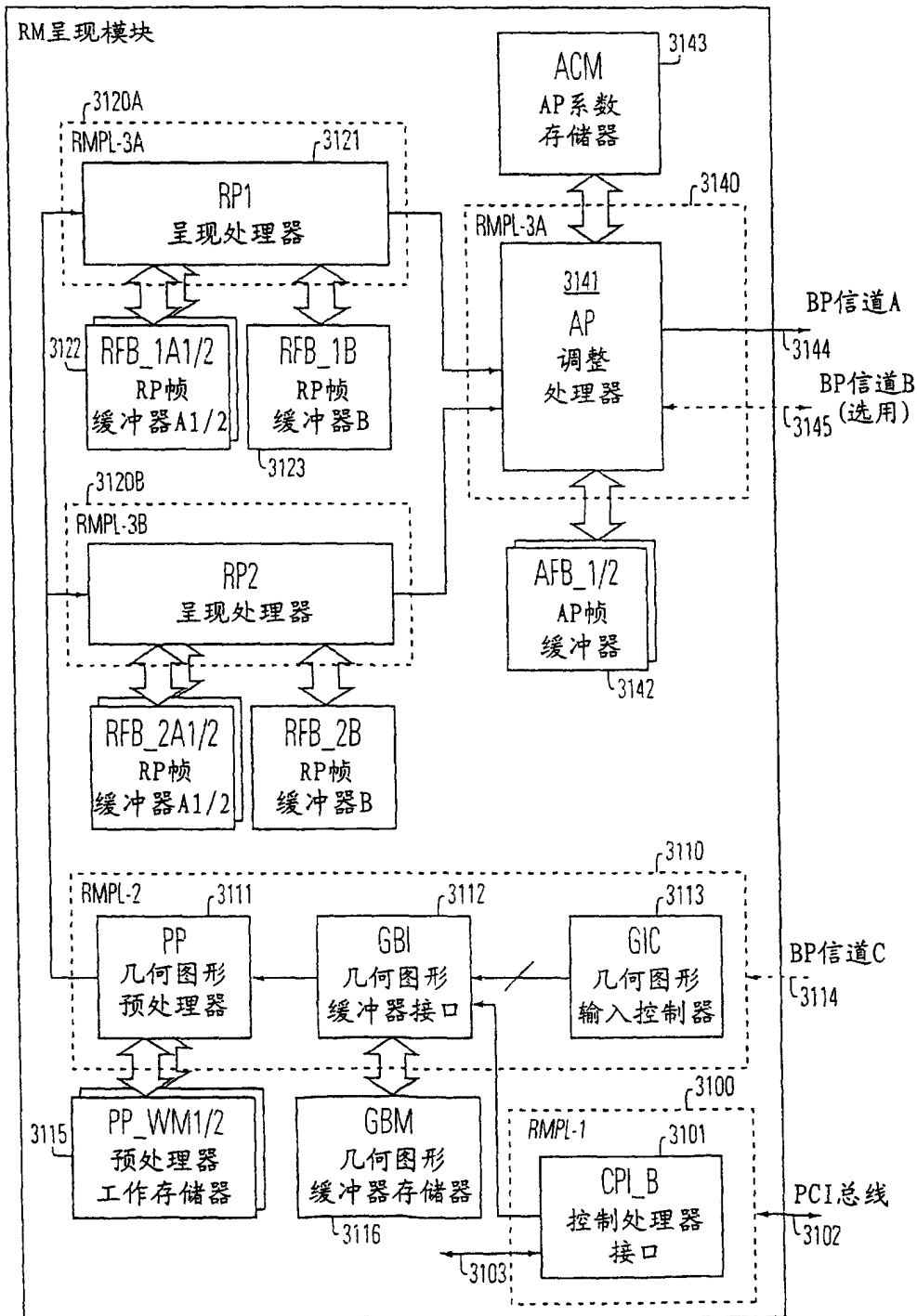


图 31

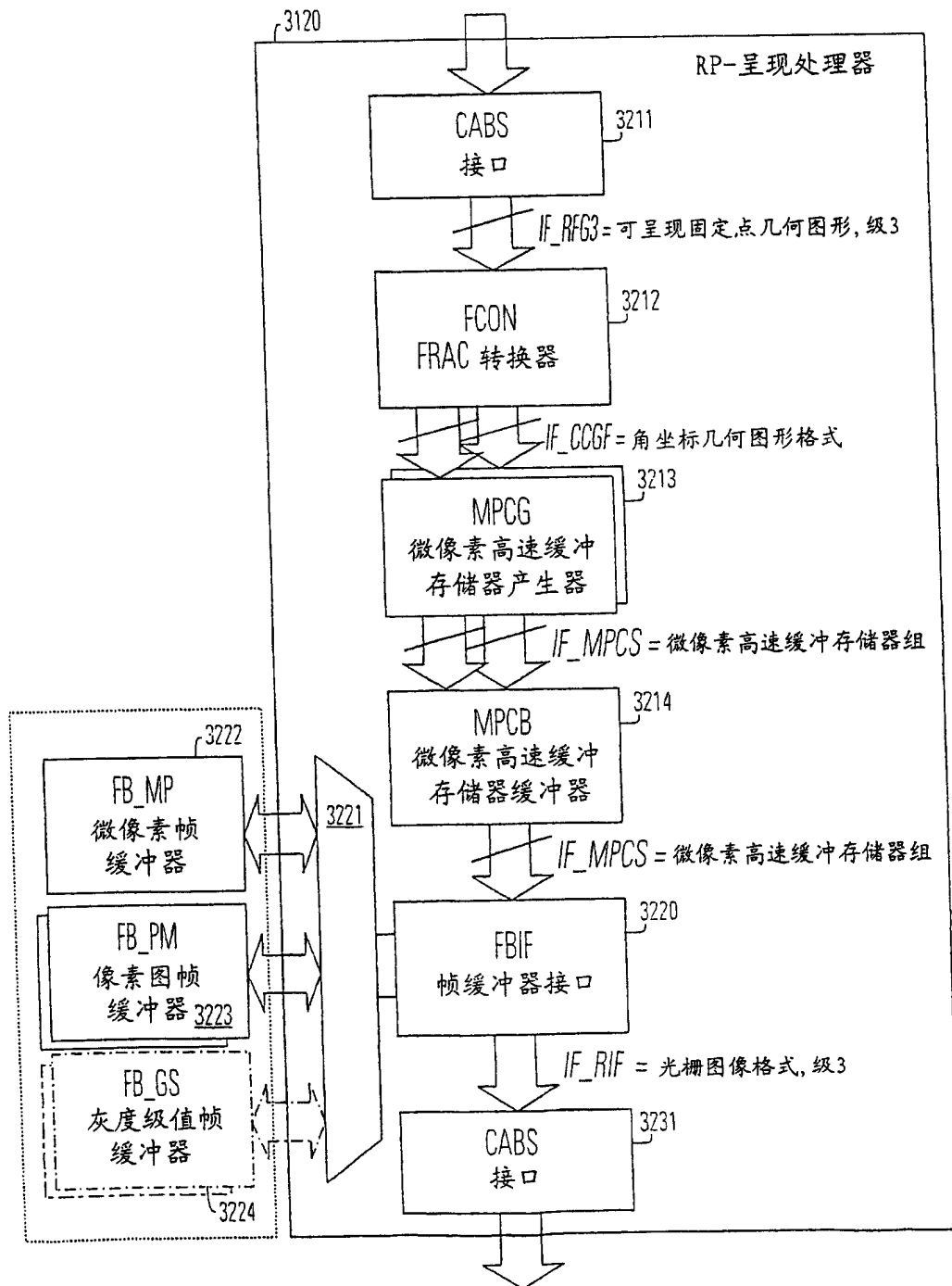


图 32

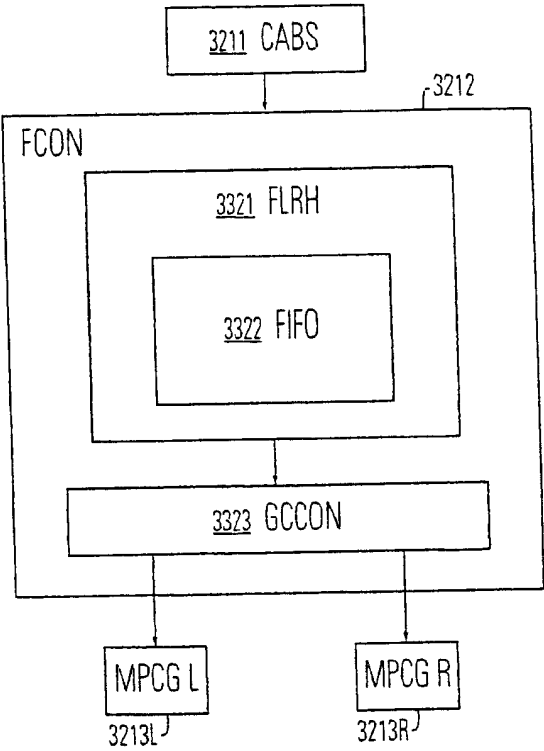


图 33

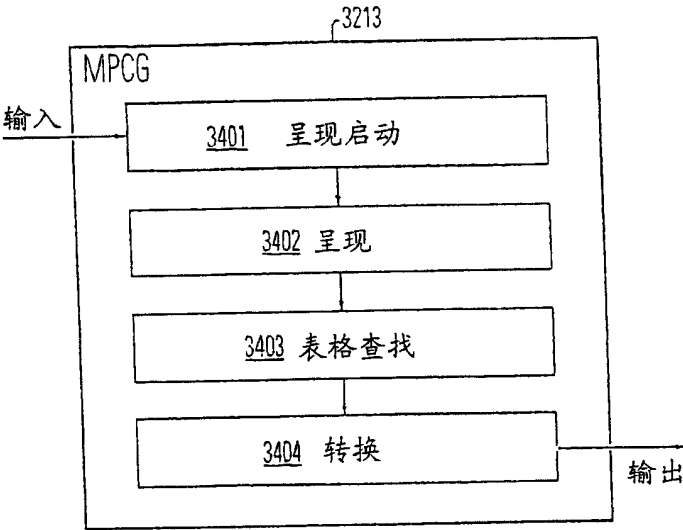


图 34

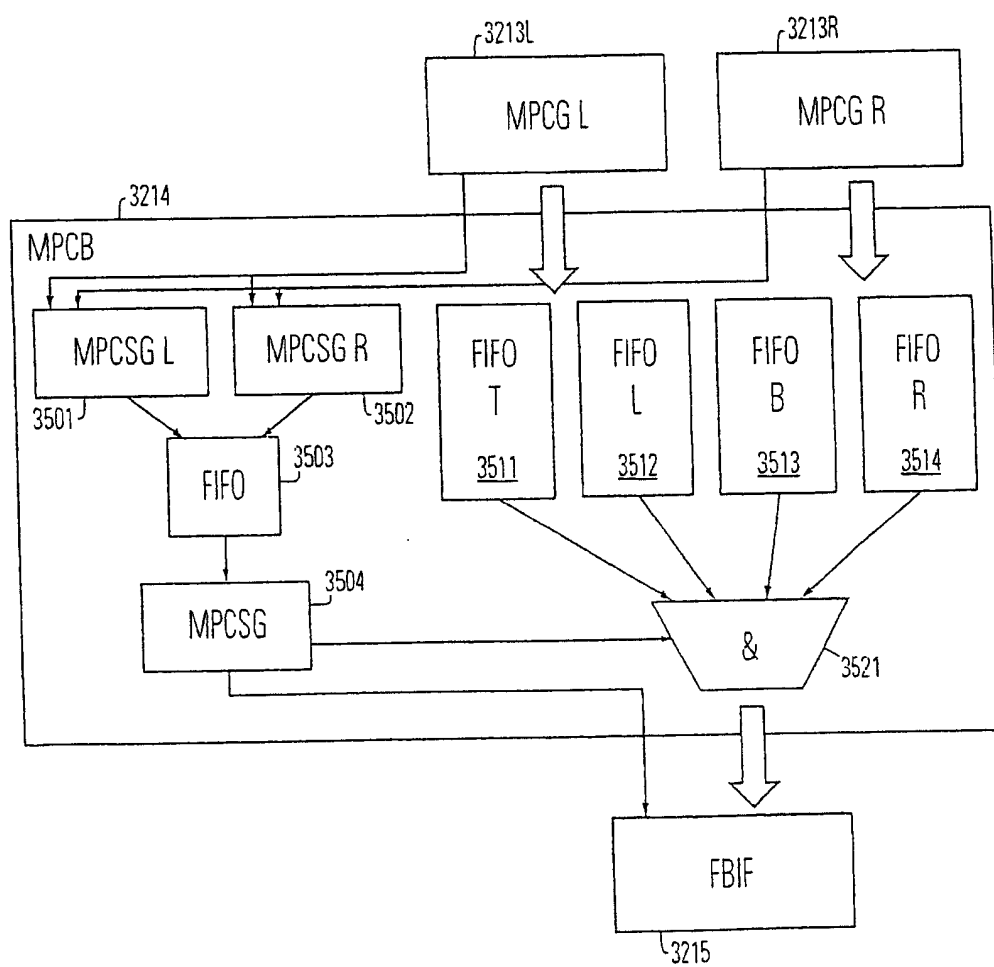


图 35

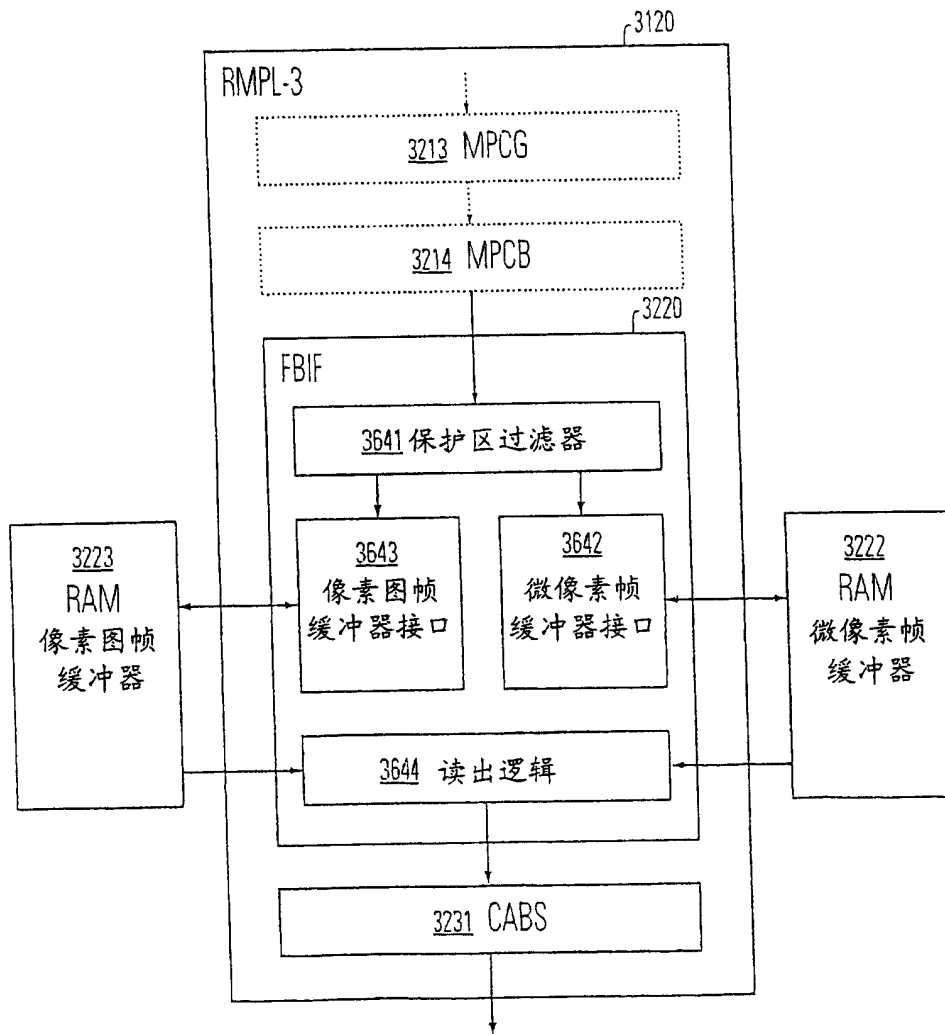


图 36

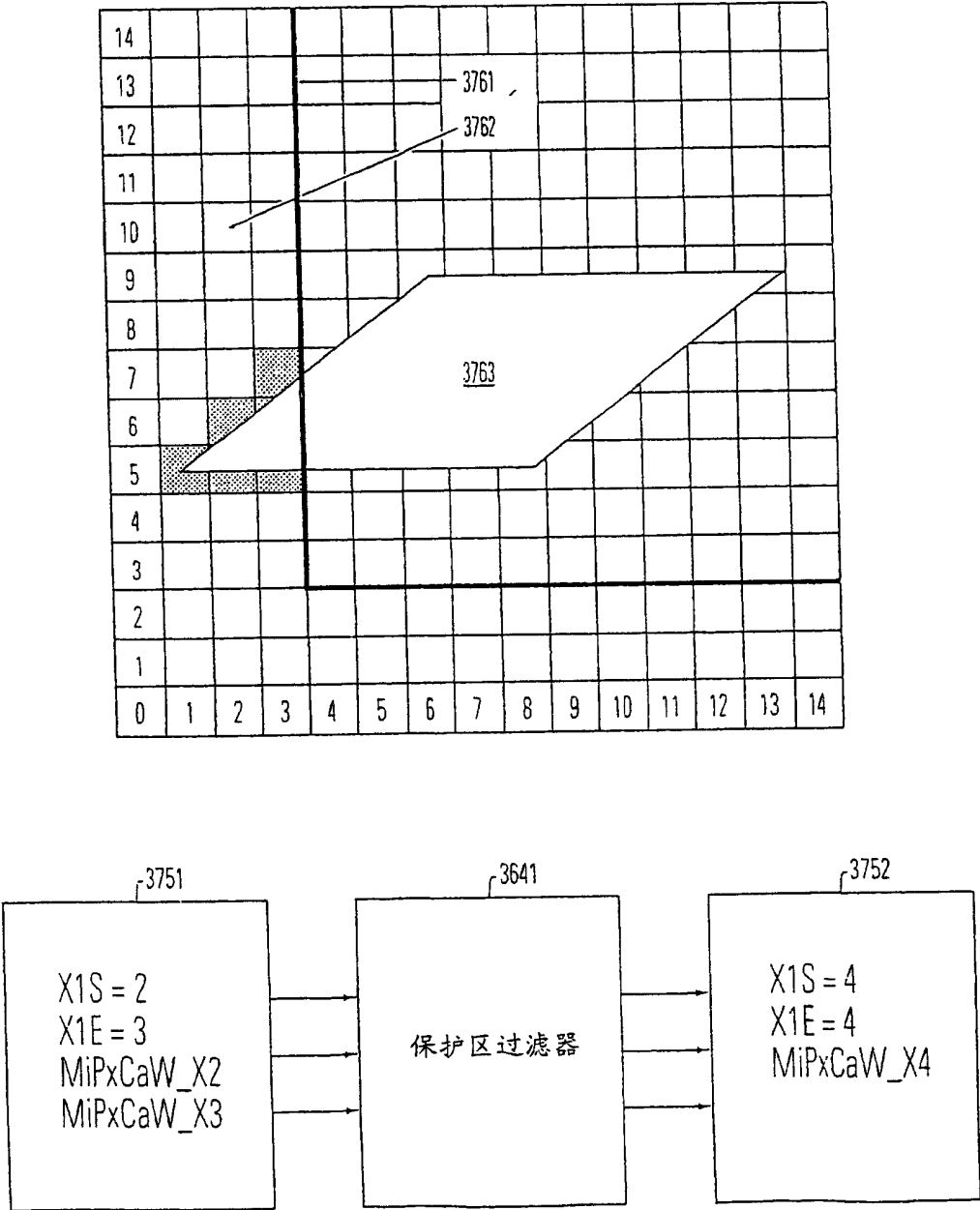


图 37

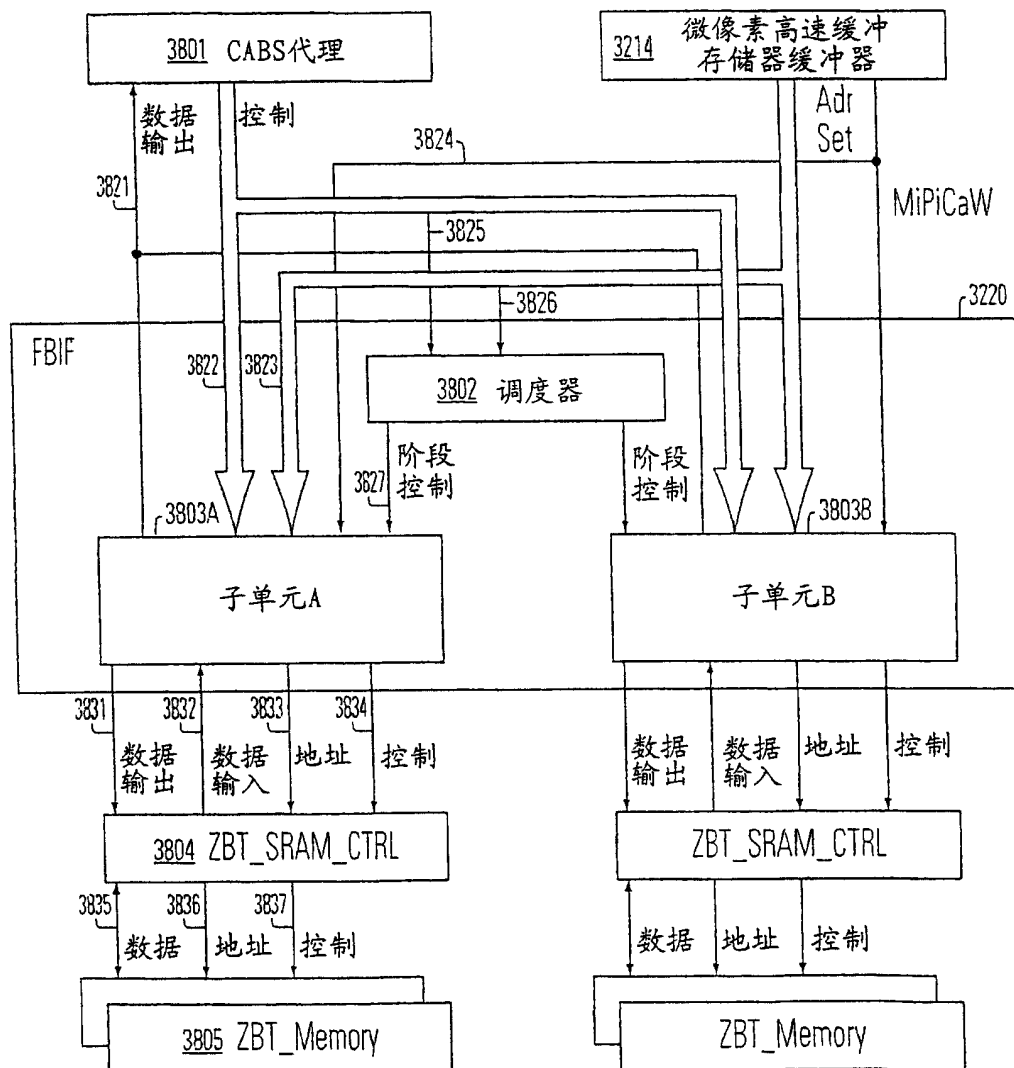


图 38

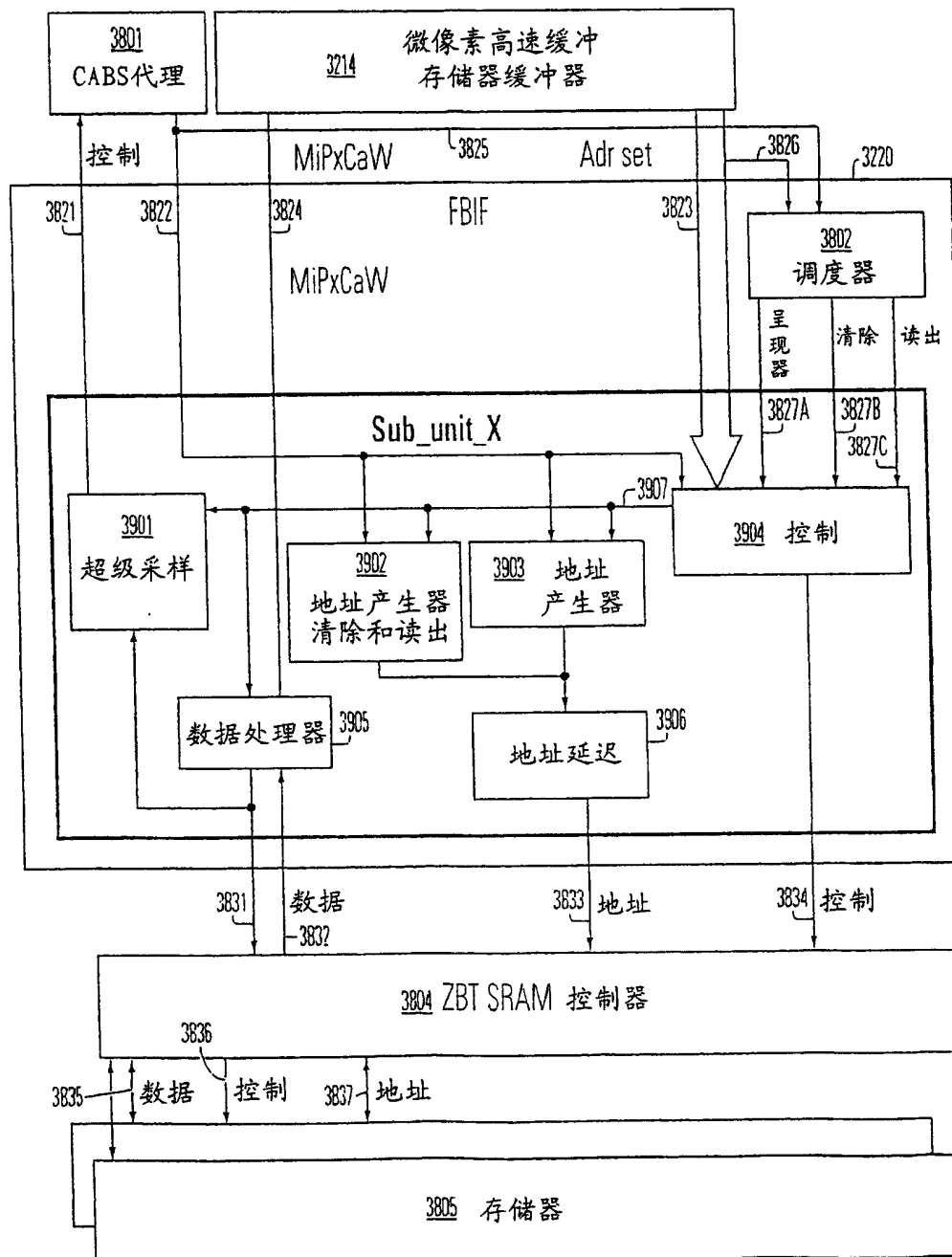


图 39

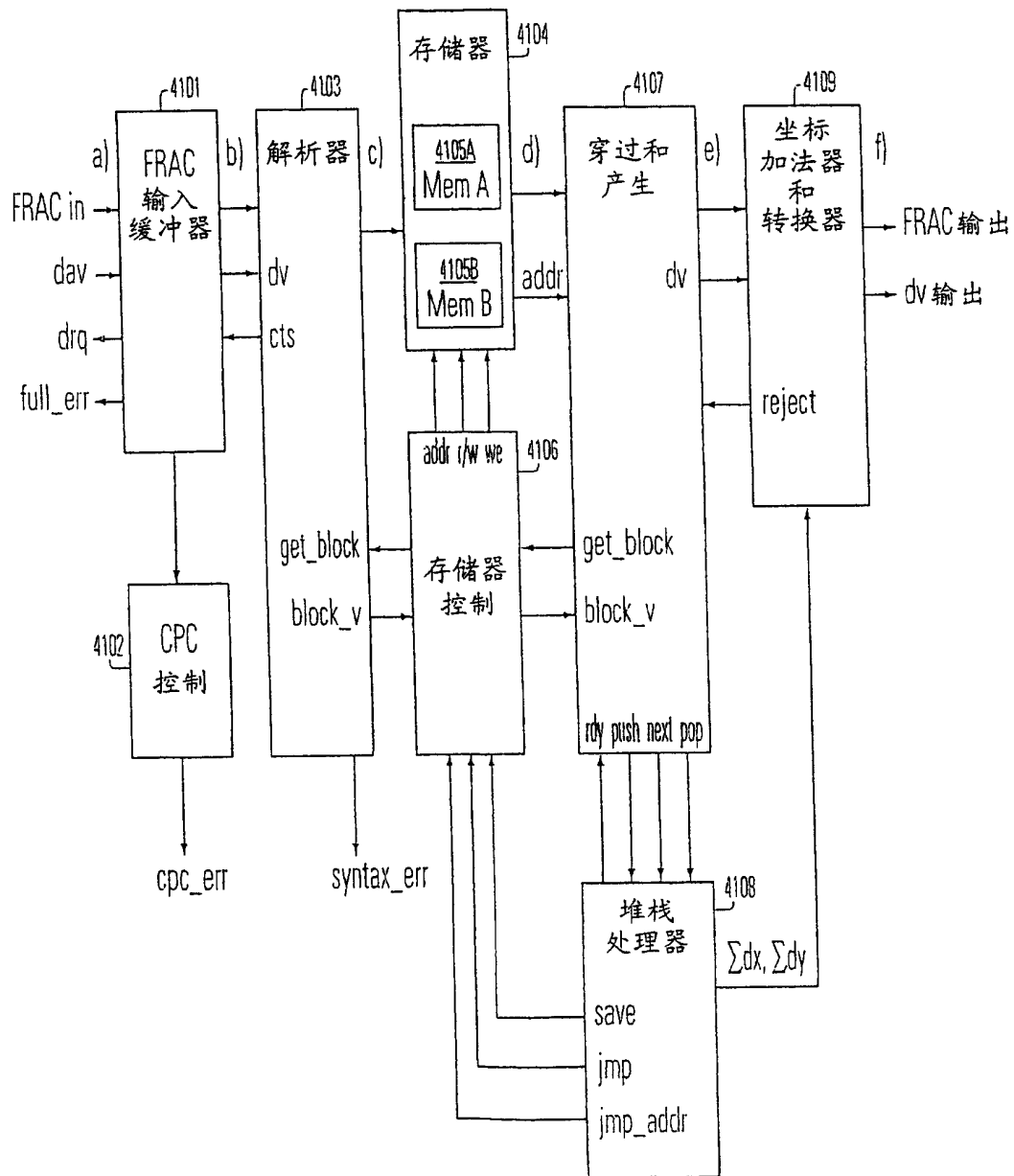


图 40

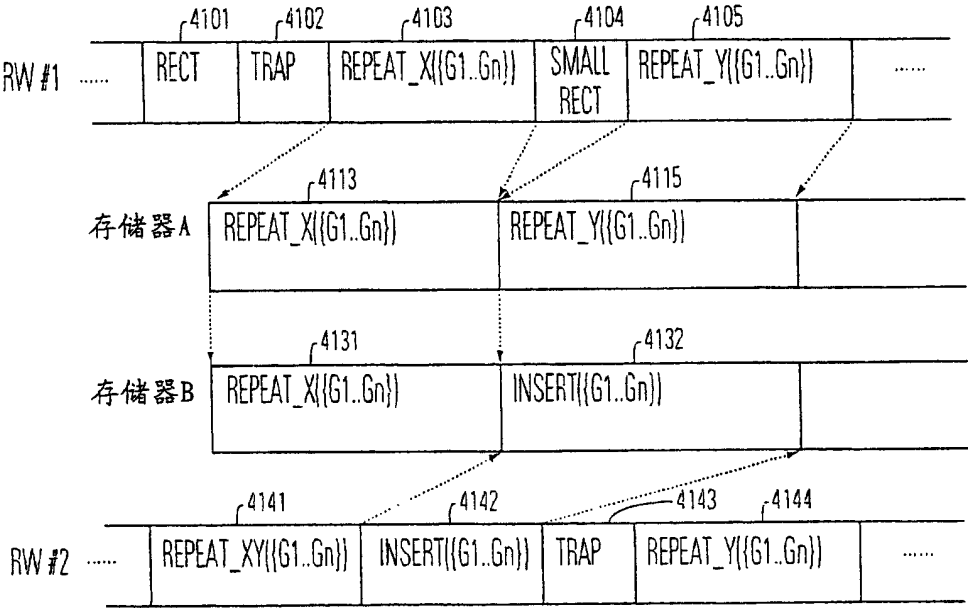


图 41

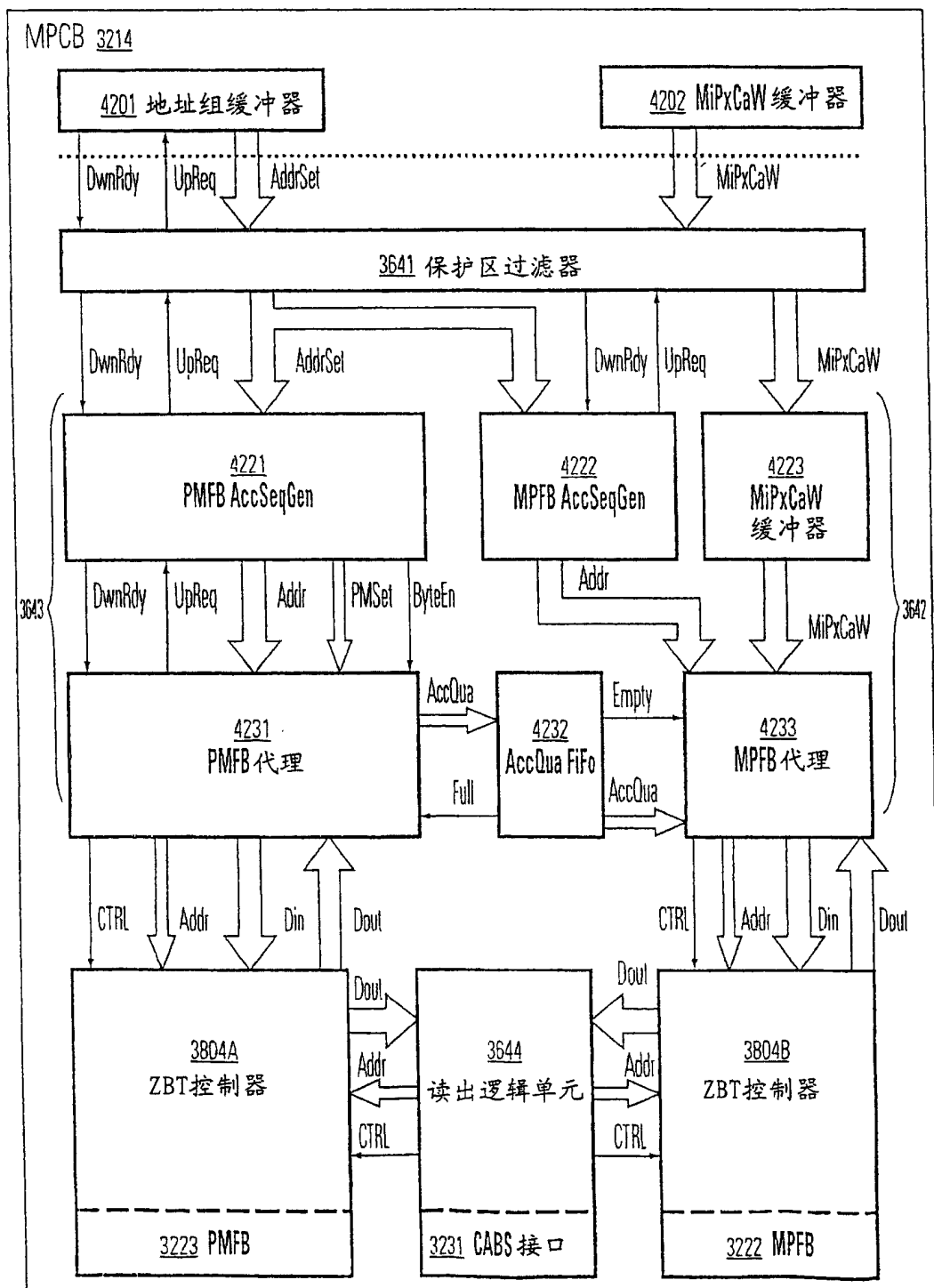


图 42

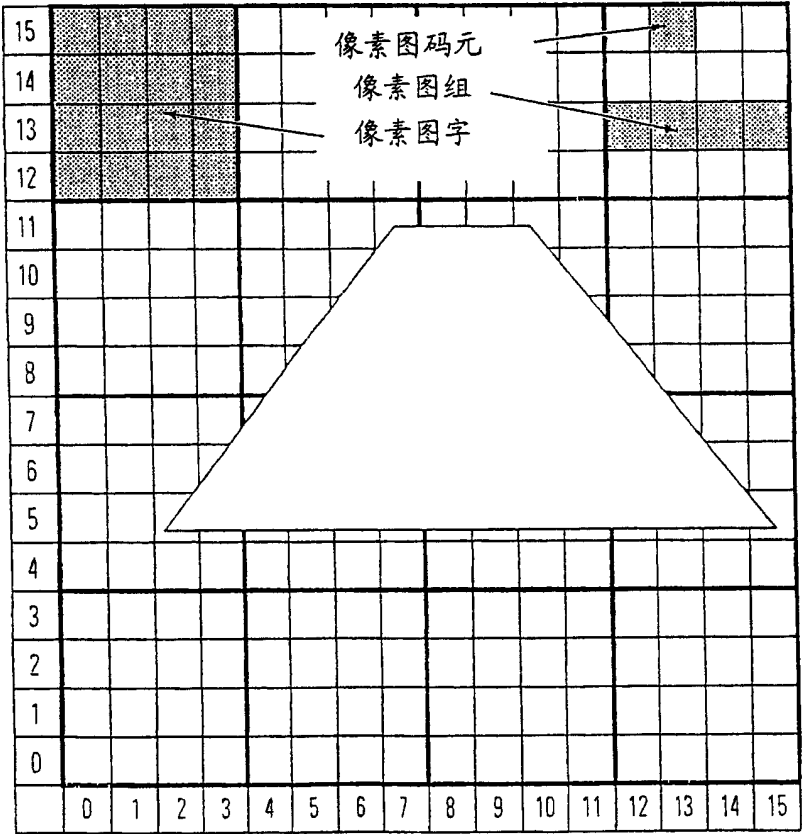


图 43

	4	7	9	10
	3	6	8	9
	2	5	6	7
	1	2	3	4

图 44

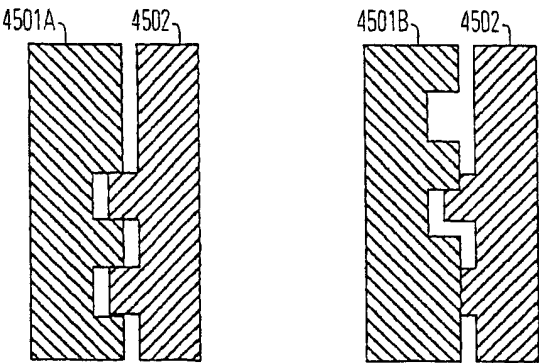


图 45

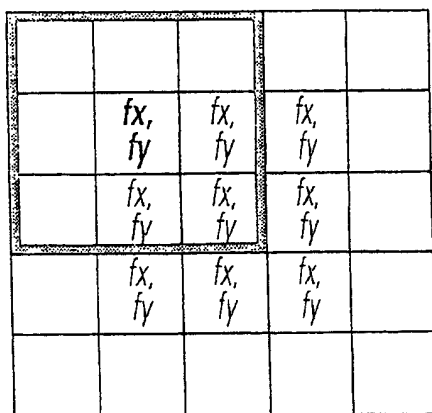


图 46

fx^2 fy^2 $fx*fy$	fx^2 fy^2 $fx*fy$	fx^2 fy^2 $fx*fy$
fx^2 fy^2 $fx*fy$	fx^2 fy^2 $fx*fy$	fx^2 fy^2 $fx*fy$
fx^2 fy^2 $fx*fy$	fx^2 fy^2 $fx*fy$	fx^2 fy^2 $fx*fy$

图 47

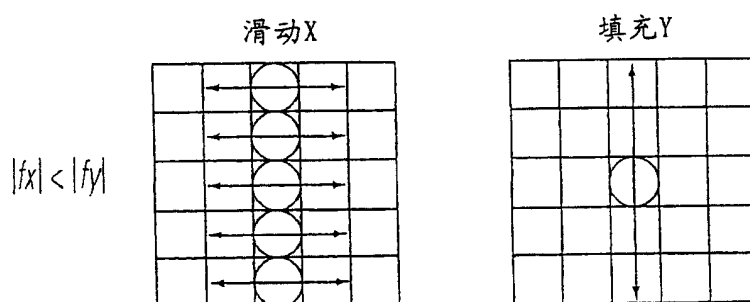


图 48

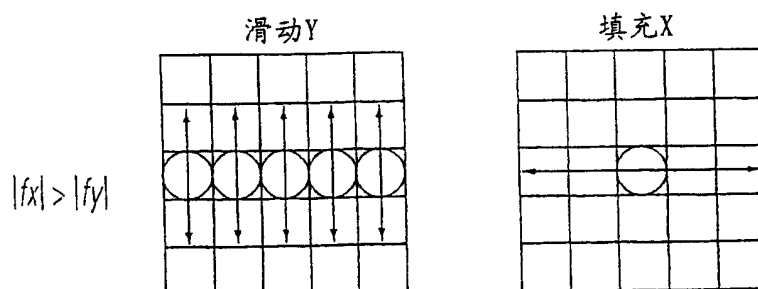


图 49

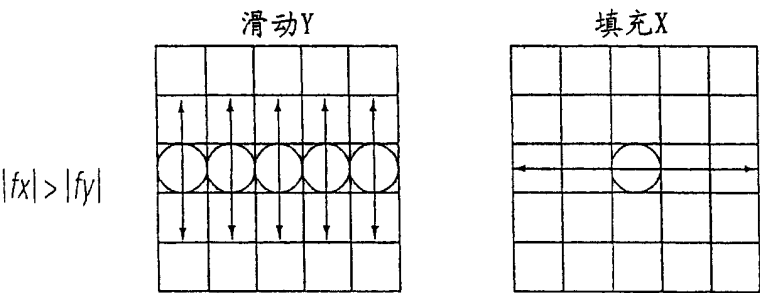


图 50

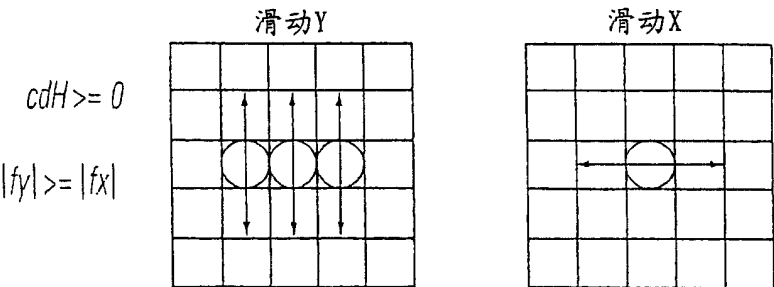


图 51

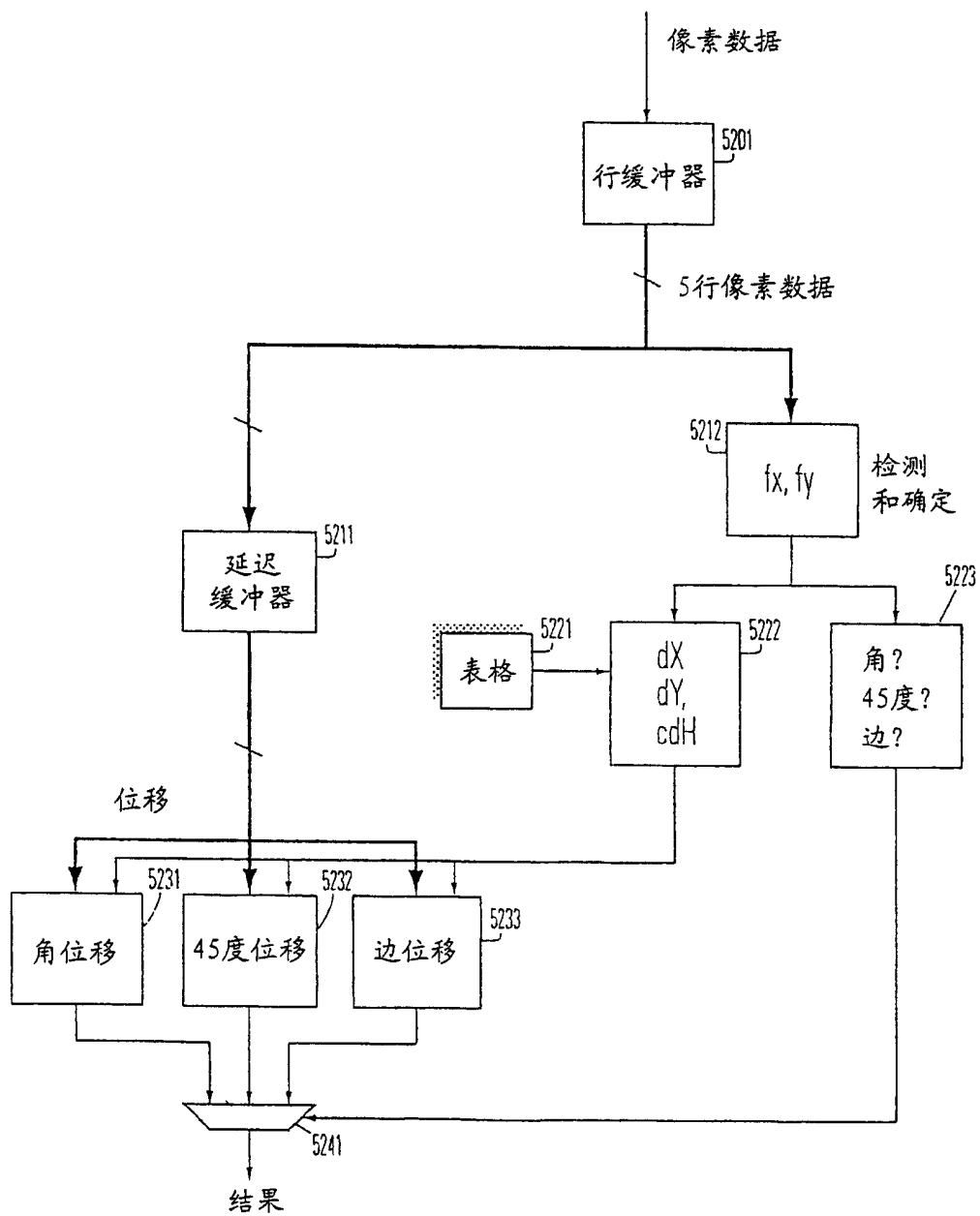


图 52

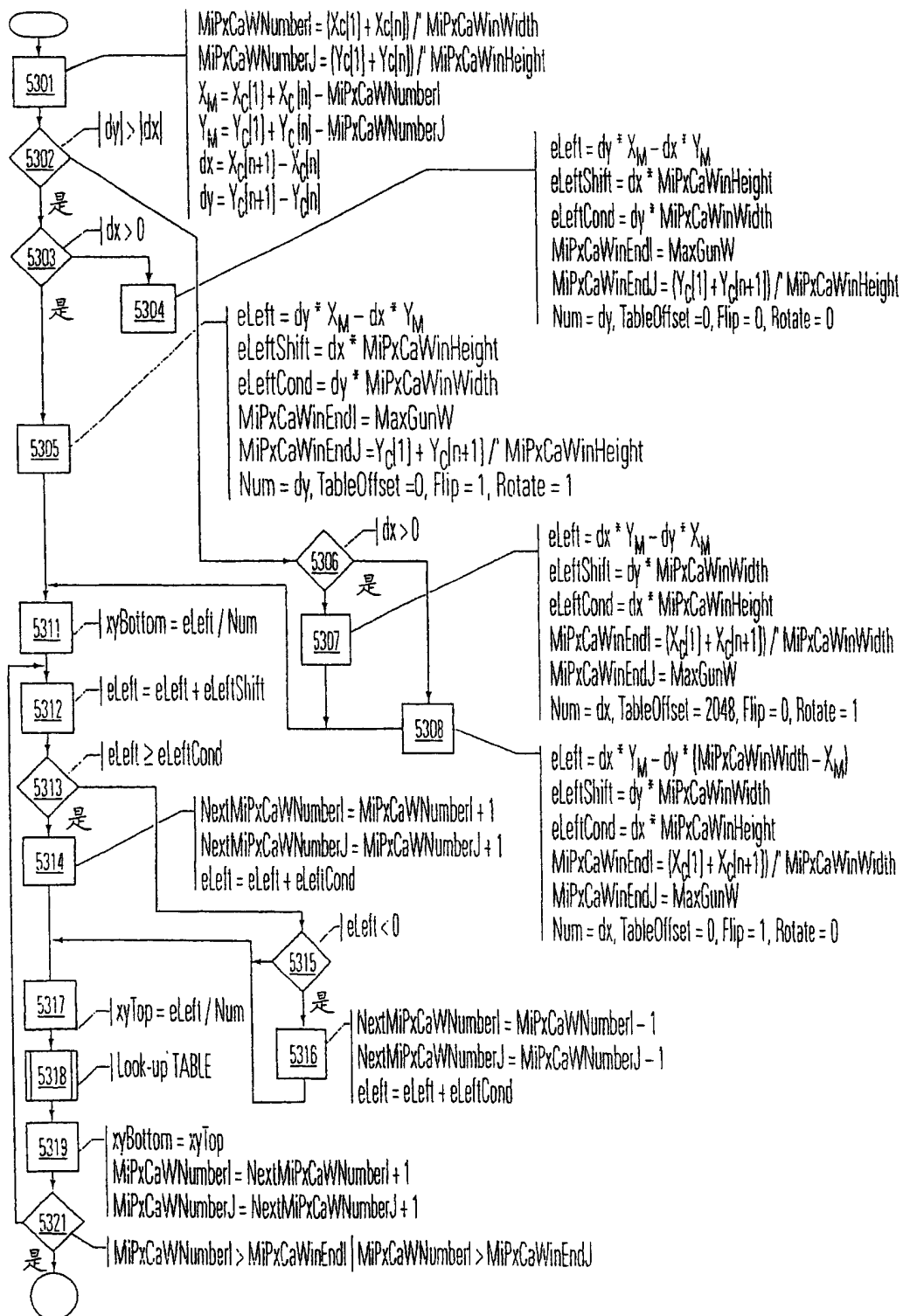


图 53

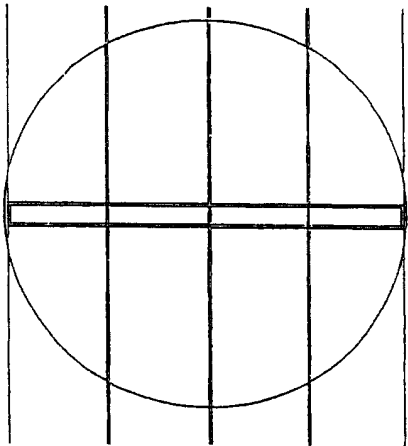


图 54A

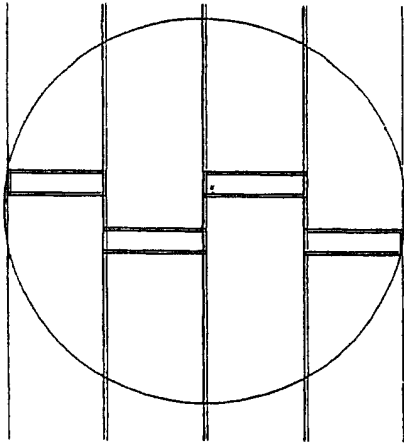


图 54B

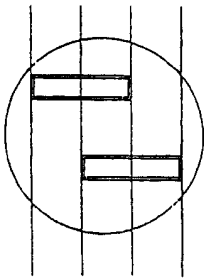


图 54C

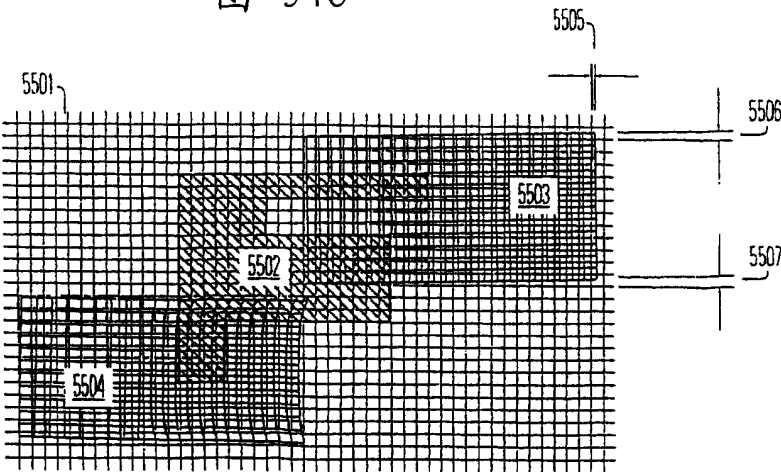


图 55