(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 108140232 B
(45)授权公告日 2022.05.24

(21)申请号 201680045334.8
(22)申请日 2016.06.09
(65)同一申请的已公布的文献号
申请公布号 CN 108140232 A
(43)申请公布日 2018.06.08
(30)优先权数据
62/173,389 2015.06.10 US
62/173,392 2015.06.10 US
62/290,383 2016.02.02 US
62/290,389 2016.02.02 US
62/290,392 2016.02.02 US
62/290,395 2016.02.02 US
62/290,400 2016.02.02 US
62/293,145 2016.02.09 US
62/293,147 2016.02.09 US
62/293,908 2016.02.11 US
(85)PCT国际申请进入国家阶段日
2018.02.01
(86)PCT国际申请的申请数据
PCT/IL2016/050611 2016.06.09
(87)PCT国际申请的公布数据
WO2016/199151 EN 2016.12.15
(73)专利权人 无比视视觉技术有限公司
地址 以色列耶路撒冷
(72)发明人 丹尼尔·斯莱伯尼克
伊曼纽尔·西克苏
吉尔·伊斯雷尔·多贡
(74)专利代理机构 北京安信方达知识产权代理有限公司 11262
专利代理师 陆建萍 杨明钊
(51)Int.Cl.
G06T 1/20(2006.01)
(56)对比文件
US 6061477 A,2000.05.09
US 2010077177 A1,2010.03.25
CN 103854252 A,2014.06.11
CN 103810739 A,2014.05.21
CN 101194245 A,2008.06.04
US 2013108187 A1,2013.05.02
Lei Qu等."LittleQuickWarp: An ultrafast image warping tool".《Methods》.2014,第38-42页.
李旭东 等."非对称径向基函数与稳定边界图像变形算法".《计算机辅助设计与图形学学报》.2004,第16卷(第6期),第747-752页. (续)
审查员 于湃

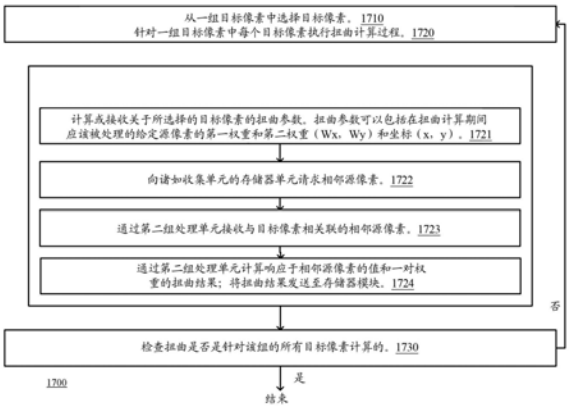
权利要求书7页 说明书57页 附图33页

(54)发明名称

用于处理图像的图像处理器和方法

(57)摘要

一种计算扭曲结果的方法,该方法可以包括:针对一组目标像素中的每个目标像素执行扭曲计算过程,该扭曲计算过程包括:通过处理单元阵列中的第一组处理单元接收与目标像素相关联的第一权重和第二权重;通过阵列中的第二组处理单元接收与目标像素相关联的相邻源像素的值;通过第二组基于响应于相邻源像素的值和一对权重计算扭曲结果;并且将扭曲结果提供给存储器模块。



CN 108140232 B

[转续页]

[接上页]

(56)对比文件

C.A.GLASEBY等.“A review of image-warping methods”.《Journal of Applied Statistics》.1998,第25卷(第2期),第155-171页.

K.T.Gribbon等.“A Real-time FPGA implementation of a Barrel Distortion Correction Algorithm with Bilinear Interpolation”.《Image and Vision Computing NZ》.2014,第408-413页.

Jose Martinez等.“FPGA-based Pipeline Architecture to Transform Cartesian Images into Foveal Images by Using a new Foveation Approach”.《IEEE Xplore》.2006,第1-10页.

Konstantis Daloukas等.“Fisheye Lens Distortion Correction on Multicore and Hardware Accelerator Platforms”.《IEEE Xplore》.2010,第1-10页.

Andrea Giachetti 等.“Real-Time Artifact-Free Image Upscaling”.《IEEE TRANSACTIONS ON IMAGE PROCESSING》.2011,第20卷(第10期),第2760-2768页.

Andrea.“Corrections to“Real-Time Artifact Free Image Upscaling””.《IEEE TRANSACTIONS ON IMAGE PROCESSING》.2012,第21卷(第4期),第2361页.

1.一种计算扭曲结果的方法,所述方法包括:

针对一组目标像素中的每个目标像素执行扭曲计算过程,所述扭曲计算过程包括:

通过处理单元的阵列中的第一组的处理单元接收包括与所述目标像素相关联的第一权重和第二权重的一对权重;

通过所述阵列中的第二组的处理单元接收与所述目标像素相关联的相邻源像素的值;

通过所述第二组基于响应于所述相邻源像素的值和所述一对权重来计算扭曲结果;以及

将所述扭曲结果提供给存储器模块;

其中,所述扭曲结果的计算包括:

通过所述第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值;

将所述第一差值提供给所述第二组的第二处理单元;

将所述第二差值提供给所述第二组的第三处理单元;以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

2.根据权利要求1所述的方法,其中,所述扭曲结果的计算包括在所述第二组中的处理单元之间转播所述相邻源像素中的一些的值。

3.根据权利要求1所述的方法,其中,所述扭曲结果的计算包括在所述第二组的处理单元之间转播通过所述第二组计算出的中间结果和所述相邻源像素中的一些的值。

4.根据权利要求1所述的方法,其中,所述扭曲结果的计算还包括:

通过所述第二组的第四处理单元响应于所述第一权重计算第一修改权重并且响应于所述第二权重计算第二修改权重;

将所述第一修改权重从所述第四处理单元提供给所述第二组的所述第二处理单元;

通过所述第二组的所述第二处理单元基于所述第一差值、第一相邻源像素和所述第一修改权重计算第一中间结果。

5.根据权利要求4所述的方法,其中,所述扭曲结果的计算还包括:

从所述第二组的所述第三处理单元向所述第二组的第六处理单元提供所述第二差值;

从所述第二组的第五处理单元向所述第二组的所述第六处理单元提供第二相邻源像素;以及

通过所述第二组的所述第六处理单元基于所述第二差值、所述第二相邻源像素和所述第一修改权重计算第二中间结果。

6.根据权利要求5所述的方法,其中,所述扭曲结果的计算还包括:

从所述第二组的所述第六处理单元向所述第二组的第七处理单元提供所述第二中间结果;

从所述第二组的所述第二处理单元向所述第二组的所述第七处理单元提供所述第一中间结果;以及

通过所述第二组的所述第七处理单元基于所述第一中间结果和所述第二中间结果来计算第三中间结果。

7.根据权利要求6所述的方法,其中,所述扭曲结果的计算还包括:

从所述第二组的所述第七处理单元向所述第二组的第八处理单元提供所述第三中间

结果；

从所述第二组的所述第六处理单元向所述第二组的第九处理单元提供所述第二中间结果；

从所述第二组的所述第九处理单元向所述第二组的所述第八处理单元提供所述第二中间结果；

从所述第二组的所述第四处理单元向所述第二组的所述第八处理单元提供所述第二修改权重；以及

通过所述第二组的所述第八处理单元基于所述第二中间结果和所述第三中间结果以及所述第二修改权重来计算所述扭曲结果。

8.根据权利要求1所述的方法，包括通过所述阵列并行地执行与目标像素子组相关联的多个扭曲计算过程。

9.根据权利要求8所述的方法，包括：从收集单元并行地提取与所述像素子组中的每个目标像素相关联的相邻源像素；其中，所述收集单元包括组相联缓存并且被布置为访问包括多个能够独立访问的存储体的存储器模块。

10.根据权利要求9所述的方法，包括：针对所述像素子组中的每个目标像素，接收第一扭曲参数和第二扭曲参数；其中，所述第一扭曲参数和第二扭曲参数包括所述第一权重和所述第二权重以及指示与所述目标像素相关联的相邻源像素的位置的位置信息。

11.根据权利要求10所述的方法，包括向所述收集单元提供所述像素子组中的每个目标像素的所述位置信息。

12.根据权利要求11所述的方法，包括通过所述收集单元将所述位置信息转换为所述相邻源像素的地址。

13.根据权利要求9所述的方法，包括：通过所述阵列中的第三组处理单元以及针对所述像素子组中的每个目标像素计算第一扭曲参数和第二扭曲参数；其中，所述第一扭曲参数和第二扭曲参数包括所述第一权重和第二权重以及指示与所述目标像素相关联的所述相邻源像素的位置的位置信息。

14.根据权利要求13所述的方法，从所述第三组到所述第一组感测所述第一权重和所述第二权重。

15.根据权利要求14所述的方法，包括向所述收集单元提供所述像素子组中的每个目标像素的所述位置信息。

16.根据权利要求15所述的方法，包括通过所述收集单元将所述位置信息转换为所述相邻源像素的地址。

17.一种用于计算扭曲结果的方法，所述方法包括：

通过处理单元的阵列中的第一组处理单元并针对像素子组中的每个目标像素同时接收第一权重和第二权重；

向收集单元同时提供针对所述像素子组中的每个目标像素的指示与所述目标像素相关联的相邻源像素的位置的位置信息；

通过所述阵列并从所述收集单元同时接收与像素子组中的每个目标像素相关联的相邻源像素；其中，所述阵列的不同组接收与所述像素子组中的不同目标像素相关联的相邻源像素；以及

通过所述阵列的所述不同组同时计算与所述不同目标像素相关的扭曲结果；

其中，所述扭曲结果的计算包括：

通过所述不同组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述不同组的第二处理单元；

将所述第二差值提供给所述不同组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

18.根据权利要求17所述的方法，包括：针对所述像素子组中的每个目标像素接收或计算第一扭曲参数和第二扭曲参数；其中，所述第一扭曲参数和所述第二扭曲参数包括所述第一权重和所述第二权重以及指示与所述目标像素相关联的相邻源像素的位置的位置信息。

19.一种图像处理器，所述图像处理器被配置为计算扭曲结果，所述图像处理器被配置为针对一组目标像素中的每个目标像素执行扭曲计算过程，所述扭曲计算过程包括：通过所述图像处理器的处理单元阵列中的第一组处理单元接收包括与所述目标像素相关联的第一权重和第二权重的一对权重；通过所述阵列中的第二组的处理单元接收与所述目标像素相关联的相邻源像素的值；通过所述第二组基于响应于所述相邻源像素的值和所述一对权重来计算扭曲结果；并且将所述扭曲结果提供给存储器模块；

其中，所述扭曲结果的计算包括：

通过所述第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述第二组的第二处理单元；

将所述第二差值提供给所述第二组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

20.一种图像处理器，所述图像处理器被配置为计算扭曲结果，所述图像处理器包括处理单元的阵列，所述阵列被配置为通过所述阵列中的第一组处理单元并针对像素子组中的每个目标像素同时接收第一权重和第二权重；向所述图像处理器的收集单元同时提供针对所述像素子组中的每个目标像素的指示与所述目标像素相关联的相邻源像素的位置的位置信息；通过所述阵列并从所述收集单元同时接收与像素子组中的每个目标像素相关联的相邻源像素；其中，所述阵列中的不同组接收与所述像素子组中的不同目标像素相关联的相邻源像素；并且通过所述阵列中的所述不同组同时计算与所述不同目标像素相关的扭曲结果；

其中，所述扭曲结果的计算包括：

通过所述不同组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述不同组的第二处理单元；

将所述第二差值提供给所述不同组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

21.一种非暂时性计算机可读介质，所述非暂时性计算机可读介质存储用于计算扭曲结果的指令，所述指令一旦被处理单元阵列执行则导致以下步骤的执行：针对一组目标像

素中的每个目标像素执行扭曲计算过程，所述扭曲计算过程包括：通过所述处理单元阵列中的第一组处理单元接收包括与所述目标像素相关联的第一权重和第二权重的一对权重；通过所述阵列中的第二组处理单元接收与所述目标像素相关联的相邻源像素的值；通过所述第二组基于响应于所述相邻源像素的值和所述一对权重来计算扭曲结果；以及将所述扭曲结果提供给存储器模块；

其中，所述扭曲结果的计算包括：

通过所述第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述第二组的第二处理单元；

将所述第二差值提供给所述第二组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

22.一种非暂时性计算机可读介质，所述非暂时性计算机可读介质存储用于计算扭曲结果的指令，所述指令一旦被处理单元阵列执行则导致以下步骤的执行：通过所述处理单元阵列中的第一组处理单元并针对像素子组中的每个目标像素同时接收第一权重和第二权重；向收集单元同时提供针对所述像素子组中的每个目标像素的指示与所述目标像素相关联的相邻源像素的位置的位置信息；通过所述阵列并从所述收集单元同时接收与所述像素子组中的每个目标像素相关联的相邻源像素；其中，所述阵列中的不同组接收与所述像素子组中的不同目标像素相关联的相邻源像素；以及通过所述阵列的不同组同时计算与所述不同目标像素相关的扭曲结果；

其中，所述扭曲结果的计算包括：

通过所述不同组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述不同组的第二处理单元；

将所述第二差值提供给所述不同组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

23.一种计算扭曲结果的方法，包括：

针对一组目标像素中的每个目标像素执行扭曲计算过程，所述扭曲计算过程包括：通过处理单元的阵列中的第一组处理单元接收包括与所述目标像素相关联的第一权重和第二权重的一对权重；通过所述阵列的第二组处理单元接收与所述目标像素相关联的相邻源像素的值；通过所述第二组基于响应于所述相邻源像素的值和所述一对权重计算扭曲结果；以及将所述扭曲结果提供给存储器模块；

通过数据处理器的阵列中的第一组数据处理器计算一组绝对差分和SAD；其中，所述一组SAD与源像素和目标像素子组相关联；其中，每个SAD是基于先前计算的SAD并基于当前计算的在另一个源像素与属于所述目标像素子组的目标像素之间的绝对差计算的；以及通过所述阵列中的第二组数据处理器响应于所述一组SAD的值确定所述目标像素子组中的最佳匹配目标像素；

其中，所述扭曲结果的计算包括：

通过所述第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述第二组的第二处理单元；

将所述第二差值提供给所述第二组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

24.一种计算扭曲结果的方法，包括：(a)针对一组目标像素中的每个目标像素执行扭曲计算过程，所述扭曲计算过程包括：通过处理单元阵列中的第一组处理单元接收包括与所述目标像素相关联的第一权重和第二权重的一对权重；通过所述阵列中的第二组处理单元接收与所述目标像素相关联的相邻源像素的值；通过所述第二组基于响应于所述相邻源像素的值和所述一对权重计算扭曲结果；并且将所述扭曲结果提供给存储器模块；以及(b)通过收集单元的输入接口接收用于检索多个被请求的数据单元的多个请求；通过包括多个条目的缓存存储器存储多个标签和多个缓存的数据单元；其中，每个标签与缓存的数据单元相关联并且指示存储器模块中的与所述缓存存储器不同的并存储该缓存的数据单元的一组存储器单元；通过比较器阵列在所述多个标签与多个被请求的存储器组地址之间同时进行比较以提供比较结果；其中，每个被请求的存储器组地址指示所述存储器模块中的存储所述多个被请求的数据单元中的被请求的数据单元的一组存储器单元；通过控制器基于所述比较结果将所述多个被请求的数据单元分类为存储在所述缓存存储器中的缓存的数据单元和未被缓存的数据单元；以及向争用评估单元发送关于缓存的数据单元和未被缓存的数据单元的信息；通过所述争用评估单元检查至少一个争用的发生；以及通过输出接口以无竞争的方式向所述存储器模块请求任何未被缓存的数据单元；

其中，所述扭曲结果的计算包括：

通过所述第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述第二组的第二处理单元；

将所述第二差值提供给所述第二组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

25.一种计算扭曲结果的方法，包括：(a)针对一组目标像素中的每个目标像素执行扭曲计算过程，所述扭曲计算过程包括：通过处理单元阵列中的第一组处理单元接收包括与所述目标像素相关联的第一权重和第二权重的一对权重；通过所述阵列中的第二组处理单元接收与所述目标像素相关联的相邻源像素的值；通过所述第二组基于响应于所述相邻源像素的值和所述一对权重来计算扭曲结果；并且将所述扭曲结果提供给存储器模块；和(b)操作包括数据处理器阵列的处理模块；其中，所述操作包括通过所述阵列中的数据处理器处理数据；其中，所述数据处理器阵列中的多个数据处理器中的每个数据处理器直接耦合到所述数据处理器阵列中的一些数据处理器，间接耦合到所述数据处理器阵列中的一些其他数据处理器，并且使用一个或更多个数据处理器中的一个或更多个转播信道在所述数据处理器的转播端口之间转播数据；

其中，所述扭曲结果的计算包括：

通过所述第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述第二组的第二处理单元；

将所述第二差值提供给所述第二组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

26.一种计算扭曲结果的方法，其包括：(a)针对一组目标像素中的每个目标像素执行扭曲计算过程，所述扭曲计算过程包括：通过处理单元阵列中的第一组处理单元接收一对权重，所述一对权重包括与所述目标像素相关联的第一权重和第二权重；通过所述阵列中的第二组处理单元接收与所述目标像素相关联的相邻源像素的值；通过所述第二组基于响应于所述相邻源像素的值和所述一对权重来计算扭曲结果；并且将所述扭曲结果提供给存储器模块；和(b)配置包括多个可配置电路和多个微控制器的图像处理器；其中，所述多个可配置电路包括存储器电路和多个数据处理器；其中，配置所述图像处理器包括在每个可配置电路中存储高达有限量的配置指令；通过向所述多个可配置电路重复提供选择信息来通过所述多个微控制器控制所述多个可配置电路，所述选择信息用于通过每个可配置电路从所述有限量的配置指令中选择所选择的配置指令；

其中，所述扭曲结果的计算包括：

通过所述第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述第二组的第二处理单元；

将所述第二差值提供给所述第二组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

27.一种计算扭曲结果的方法，包括：(a)针对一组目标像素中的每个目标像素执行扭曲计算过程，所述扭曲计算过程包括：通过处理单元阵列中的第一组处理单元接收一对权重，所述一对权重包括与所述目标像素相关联的第一权重和第二权重；通过所述阵列中的第二组处理单元接收与所述目标像素相关联的相邻源像素的值；通过所述第二组基于响应于所述相邻源像素的值和所述一对权重来计算扭曲结果；并且将所述扭曲结果提供给存储器模块；和(b)操作图像处理器，其中，操作所述图像处理器包括提供图像处理器，所述图像处理器包括数据处理器阵列；包括多个存储体的存储器模块；缓冲单元；收集单元；和多个微控制器；通过所述多个微控制器、所述存储器模块的一部分和所述缓冲单元控制所述数据处理器阵列；通过所述缓冲单元从所述存储器模块中检索数据；通过所述缓冲单元将所述数据发送到所述数据处理器阵列；通过所述收集单元接收从所述存储器模块检索多个被请求的数据单元的多个请求；通过所述收集单元将所述多个被请求的数据单元发送到所述数据处理器阵列；

其中，所述扭曲结果的计算包括：

通过所述第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值；

将所述第一差值提供给所述第二组的第二处理单元；

将所述第二差值提供给所述第二组的第三处理单元；以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

28.一种计算扭曲结果的方法，其包括：(a)针对一组目标像素中的每个目标像素执行扭曲计算过程，所述扭曲计算过程包括：通过处理单元阵列中的第一组处理单元接收一对权重，所述一对权重包括与所述目标像素相关联的第一权重和第二权重；通过所述阵列中的第二组处理单元接收与所述目标像素相关联的相邻源像素的值；通过所述第二组基于响

应于所述相邻源像素的值和所述一对权重来计算扭曲结果;并且将所述扭曲结果提供给存储器模块;和(b)配置包括数据处理器的阵列、第一微控制器、缓冲单元和第二微控制器的图像处理器;其中,配置所述图像处理器包括:在数据处理器配置过程期间向所述阵列中的数据处理器提供数据处理器配置指令;在缓冲单元配置过程期间向所述缓冲单元提供缓冲单元配置指令;通过向数据处理器提供数据处理器选择信息来通过所述第一微控制器控制所述数据处理器的操作;通过所述数据处理器响应于所述数据处理器选择信息选择所选择的数据处理器配置指令,并且根据所选择的数据处理器配置指令执行一个或更多个数据处理操作;通过向所述缓冲单元提供缓冲单元选择信息来通过所述第二微控制器控制所述缓冲单元的操作;通过所述缓冲单元响应于所述缓冲单元选择信息的至少一部分选择所选择的缓冲单元配置指令,并且根据所选择的缓冲单元配置指令执行一个或更多个缓冲单元操作;其中,数据处理器选择信息的大小是数据处理器配置指令的大小的一部分;

其中,所述扭曲结果的计算包括:

通过所述第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值;

将所述第一差值提供给所述第二组的第二处理单元;

将所述第二差值提供给所述第二组的第三处理单元;以及

根据所述第一差值、第二差值和所述一对权重来计算扭曲结果。

用于处理图像的图像处理器和方法

[0001] 相关申请的交叉引用

[0002] 本申请要求以下专利的优先权:于2015年6月10日提交的美国临时专利序列号62/173,389;于2015年6月10日提交的美国临时专利序列号62/173,392;于2016年2月2日提交的美国临时专利序列号62/290,383;于2016年2月2日提交的美国临时专利序列号62/290,389;于2016年2月2日提交的美国临时专利序列号62/290,392;于2016年2月2日提交的美国临时专利序列号62/290,395;于2016年2月2日提交的美国临时专利序列号62/290,400;于2016年2月9日提交的美国临时专利序列号62/293,145;于2016年2月9日提交的美国临时专利序列号62/293,147;以及于2016年2月11日提交的美国临时专利62/293,908,这些专利申请全部内容通过引用并入本文。

[0003] 背景

[0004] 在过去几年中,基于摄像头的驾驶员辅助系统(DAS)已经进入市场,并且大力发展自动驾驶汽车。DAS包括车道偏离报警(LDW)、自动远光灯控制(AHC)、行人识别和前向碰撞报警(FCW)。这些驾驶员辅助系统可以使用从安装在车辆中的相机捕获的多个图像帧中检测到的多个块(patches)的实时图像处理。

[0005] 越来越需要提供用于支持DAS和/或自动驾驶汽车的高吞吐量小面积图像处理器(high throughput low footprint image processor)。

[0006] 概述

[0007] 提供了如权利要求和说明书中所示的系统、方法。

[0008] 可以提供任何权利要求中的任何主题的任何组合。

[0009] 可以提供在任何附图中和/或在说明书中公开的任何方法和/或方法步骤的任何组合。

[0010] 可以提供在任何附图中和/或在说明书中公开的任何单元、设备和/或组件的任何组合。这种单元的非限制性示例包括收集单元、图像处理器等。

[0011] 可以提供原始提交的权利要求1-17、18-19、20-21、25-42、53、75、97、98、99和109-114的方法和/或方法步骤的任何组合。

[0012] 可以提供原始提交的权利要求22、23、24、43、76-93和94-96中的任何图像处理器和/或图像处理器组件的任何组合。

[0013] 可以提供原始提交的权利要求22、23、24、43、76-93和94-96中所要求保护的任何图像处理器与原始提交的权利要求44-52中所要求保护的任何收集单元和/或原始提交的权利要求54中的任何处理模块的任何组合。

[0014] 根据本发明的实施例,可以提供一种计算扭曲结果(warp result)的方法,该方法可以包括针对一组目标像素中的每个目标像素执行扭曲计算过程(warp calculation process),该扭曲计算过程可以包括通过处理单元阵列中的第一组处理单元接收与目标像素相关联的第一权重和第二权重;通过该阵列的第二组处理单元接收与该目标像素相关联的相邻源像素的值;通过该第二组基于响应于该相邻源像素的值和一对权重来计算扭曲结果;并将该扭曲结果提供给存储器模块。

[0015] 扭曲结果的计算可以包括在第二组中的处理单元之间转播(relay)一些相邻源像素的值。

[0016] 扭曲结果的计算可以包括在第二组的处理单元之间转播通过第二组计算出的中间结果和一些相邻源像素的值。

[0017] 扭曲结果的计算可以包括:通过第二组的第一处理单元计算第一对相邻源像素之间的第一差值和第二对相邻源像素之间的第二差值;将该第一差值提供给该第二组的第二处理单元;并且将第二差值提供给第二组的第三处理单元。

[0018] 该扭曲结果的计算还可以包括:响应于第一权重通过第二组的第四处理单元计算第一修改权重;将来自第四处理单元的第一修改权重提供给第二组的第二处理单元;通过第二组的第二处理单元基于第一差值、第一相邻源像素和第一修改权重计算第一中间结果。

[0019] 该扭曲结果的计算还可以包括:从第二组的第三处理单元向第二组的第六处理单元提供第二差值;从第二组的第五处理单元向第二组的第六处理单元提供第二相邻源像素;以及通过第二组的第六处理单元基于第二差值、第二相邻源像素和第一修改权重计算第二中间结果。

[0020] 扭曲结果的计算还可以包括:从第二组的第六处理单元向第二组的第七处理单元提供第二中间结果;从第二组的第二处理单元向第二组的第七处理单元提供第一中间结果;并且通过第二组的第七处理单元基于第一中间结果和第二中间结果来计算第三中间结果。

[0021] 扭曲结果的计算还可以包括:从第二组的第七处理单元向第二组的第八处理单元提供第三中间结果;从第二组的第六处理单元向第二组的第九处理单元提供第二中间结果;从第二组的第九处理单元向第二组的第八处理单元提供第二中间结果;从第二组的第三处理单元向第二组的第八处理单元提供第二修改权重;以及通过第二组的第八处理单元基于第二中间结果和第三中间结果以及第二修改权重来计算扭曲结果。

[0022] 该方法可以包括通过阵列并行地执行与目标像素的子组相关联的多个扭曲计算过程。

[0023] 该方法可以包括:从收集单元并行地提取与像素的子组中的每个目标像素相关联的相邻源像素;其中,收集单元可以包括组相联缓存(set associate cache)并且可以被布置为访问可以包括多个可独立访问的存储体的存储器模块。

[0024] 该方法可以包括:针对像素的子组中的每个目标像素接收第一扭曲参数和第二扭曲参数;其中,第一扭曲参数和第二扭曲参数可以包括第一权重和第二权重以及指示与目标像素相关联的相邻源像素的位置的位置信息。

[0025] 该方法可以包括向收集单元提供像素的子组中的每个目标像素的位置信息。

[0026] 该方法可以包括通过收集单元将位置信息转换为相邻源像素的地址。

[0027] 该方法可以包括通过阵列的第三组处理单元以及针对像素的子组的每个目标像素计算第一扭曲参数和第二扭曲参数;其中,第一扭曲参数和第二扭曲参数可以包括第一权重和第二权重以及指示与目标像素相关联的相邻源像素的位置的位置信息。

[0028] 该方法从第三组到第一组感测第一权重和第二权重。

[0029] 该方法可以包括向收集单元提供像素的子组的每个目标像素的位置信息。

[0030] 该方法可以包括通过收集单元将位置信息转换为相邻源像素的地址。

[0031] 根据本发明的一个实施例，可以提供一种用于计算扭曲结果的方法，该方法可以包括通过处理单元阵列中的第一组处理单元并针对像素的子组的每个目标像素，同时接收第一权重和第二权重；向收集单元同时提供针对像素的子组中的每个目标像素的指示与所述目标像素相关联的相邻源像素的位置的位置信息；通过该阵列并从所述收集单元同时接收与像素的子组中的每个目标像素相关联的相邻源像素；其中该阵列的不同组接收与像素的子组中的不同目标像素相关联的相邻源像素；并且通过该阵列的不同组来同时计算与不同目标像素相关的扭曲结果。

[0032] 该方法可以包括：针对像素的子组中的每个目标像素接收或计算第一扭曲参数和第二扭曲参数；其中，该第一扭曲参数和第二扭曲参数可以包括第一权重和第二权重以及指示与目标像素相关联的相邻源像素的位置的位置信息。

[0033] 根据本发明的一个实施例，可以提供一种用于计算扭曲结果的方法，该方法可以包括：针对目标像素组中的每个目标像素子组重复以下步骤：通过处理单元阵列接收与目标像素的子组中的每个目标像素相关联的相邻源像素；以及通过该阵列计算针对来自目标像素的子组的目标像素的扭曲结果；其中，计算可以包括计算中间结果并且在阵列中的处理单元之间转播中间结果中的至少一些。

[0034] 阵列的每个处理单元可以直接耦合到阵列中的一组处理单元并且可以间接耦合到阵列中的另一组处理单元。术语“处理单元”和“数据处理器”可以以可互换的方式使用。

[0035] 根据本发明的实施例，可以提供可以被配置成计算扭曲结果的图像处理器，该图像处理器可以被配置为针对目标像素的组中的每个目标像素执行扭曲计算过程，该扭曲计算过程可以包括通过该图像处理器的处理单元阵列的第一组处理单元接收与目标像素相关联的第一权重和第二权重；通过阵列中的第二组处理单元接收与目标像素相关联的相邻源像素的值；通过第二组基于响应于相邻源像素的值和一对权重来计算扭曲结果；并且将该扭曲结果提供给存储器模块。

[0036] 根据本发明的实施例，可以提供一种图像处理器，该图像处理器可以被配置为计算扭曲结果，该图像处理器可以包括处理单元阵列，该处理单元阵列可以被配置为通过阵列中的第一组处理单元并针对像素的子组中的每个目标像素同时接收第一权重和第二权重；同时向该图像处理器的收集单元提供针对像素的子组中的每个目标像素的指示与该目标像素相关联的相邻源像素的位置的位置信息；通过该阵列并从该收集单元同时接收与像素的子组中的每个目标像素相关联的相邻源像素；其中，该阵列中的不同组接收与像素的子组中的不同目标像素相关联的相邻源像素；并且通过该阵列中的不同组来同时计算与不同目标像素相关的扭曲结果。

[0037] 根据本发明的实施例，可以提供一种图像处理器，该图像处理器可以被配置为计算扭曲结果，该图像处理器可以被配置为针对目标像素的组中的每个目标像素子组重复以下步骤：通过该图像处理器的处理单元阵列接收与该目标像素的子组中的每个目标像素相关联的相邻源像素；并且通过该阵列计算针对来自目标像素的子组中的目标像素的扭曲结果；其中，计算可以包括计算中间结果并且在阵列中的处理单元之间转播中间结果中的至少一些。

[0038] 根据本发明的实施例，可以提供一种用于计算视差的方法，该方法可以包括通过

数据处理器阵列的第一组数据处理器来计算一组绝对差和(SAD);其中,该一组SAD可以与源像素和目标像素的子组相关联;其中,每个SAD可基于先前计算的SAD并且基于当前计算的另一个源像素与属于目标像素的子组的目标像素之间的绝对差来计算;以及通过该阵列的第二组数据处理器响应于该一组SAD的值确定目标像素的子组中的最佳匹配目标像素。

[0039] 该一组SAD中的给定的SAD反映给定的矩形源像素阵列与给定的矩形目标像素阵列之间的绝对差;其中,该先前计算的SAD可以包括(a)第一先前计算的SAD,其反映在(i)与给定的矩形源像素阵列相差第一源像素列和第二源像素列的矩形源像素阵列与(ii)与给定的矩形目标像素阵列相差第一目标像素列和第二目标像素列的矩形目标像素阵列之间的绝对差;以及(b)第二先前计算的SAD,其反映了第一源列与第一源列之间的绝对差。

[0040] 对于给定的SAD-另一个源像素可以是第二源像素列的最低源像素,并且属于目标像素的子组的目标像素可以是第二目标像素列的最低目标像素。

[0041] 该方法可以包括:通过从第一先前计算的SAD中减去(a)第二先前计算的SAD和(b)在(i)可以位于第二目标像素列的顶部的目标像素与(ii)可以定位于第二源像素列的顶部的源像素之间的绝对差来计算中间结果;并且将在第二目标像素列的最低目标像素与第二源像素列的最低源像素之间的绝对差添加到中间结果,来计算给定的SAD。

[0042] 该方法可以包括:针对给定的SAD,在数据处理器阵列中存储第一先前计算的SAD、第二先前计算的SAD、可以定位于第二目标像素列的顶部的目标像素以及可以定位于第二源像素列的顶部的源像素。

[0043] 给定的SAD的计算可以在提取第二目标像素列的最低目标像素和第二源像素列的最低源像素之前进行。

[0044] 目标像素的子组可以包括可以顺序地存储在存储器模块中的目标像素;其中,一组SAD的计算可以在从存储器模块提取目标像素的子组之前进行。

[0045] 从存储器模块对目标像素的子组的提取可以通过可以包括内容可寻址存储器缓存的收集单元来执行。

[0046] 目标像素的子组属于可以包括多子组目标像素的一组目标像素;其中,该方法可以包括针对每个目标像素子组重复以下步骤:通过第一组处理单元计算每个目标像素子组的一组SAD;并且通过该阵列的第二组数据处理器响应于目标像素的每一个子组的一组SAD的值找出该一组目标像素中的最佳匹配目标像素。

[0047] 该方法可以包括:通过数据处理器阵列中的第一组数据处理器计算可以与多个源像素和多个目标像素子组相关联的多组SAD;其中,该多组SAD中的每个SAD可以基于先前计算的SAD和当前计算的绝对差而被计算出;以及通过该阵列的第二组数据处理器以及针对源像素响应于可以与源像素相关联的SAD的值找出最佳匹配目标像素。

[0048] 多组SAD可以包括SAD的子组,SAD的每个子组可以与多个源像素和多个目标像素子组中的目标像素的多个子组相关联。

[0049] 多个源像素可以属于矩形像素阵列的列并且可以彼此相邻。

[0050] 多组SAD的计算可以包括并行计算SAD的不同子组中的SAD。

[0051] 该方法可以包括以顺序方式计算属于SAD的相同子组中的SAD。

[0052] 多个源像素可以是一对源像素。

[0053] 多个源像素可以是四个源像素。

[0054] SAD的不同子组可以通过数据处理器阵列中的不同的第一子组数据处理器来进行计算。

[0055] 该方法可以包括以顺序方式计算属于SAD的相同子组中的SAD;并且顺序地将与SAD的相同子组中的不同SAD相关的目标像素提取到数据处理器阵列。

[0056] 根据本发明的实施例,可以提供一种图像处理器,其可以包括数据处理器阵列并且可以被配置为通过数据处理器阵列的第一组数据处理器计算一组绝对差和(SAD)来计算视差;其中,该组SAD可以与源像素和目标像素的子组相关联;其中,每个SAD可基于先前计算的SAD并基于当前计算的在另一个源像素与属于目标像素的子组中的目标像素之间的绝对差来进行计算;以及通过该阵列的第二组数据处理器响应于该一组SAD的值确定目标像素的子组中的最佳匹配目标像素。

[0057] 根据本发明的实施例,可以提供一种收集单元,其可以包括可以被布置为接收用于检索多个被请求的数据单元的多个请求的输入接口;可以包括多个条目的缓存存储器可以被配置为存储多个标签和多个缓存的数据单元;其中,每个标签可以与缓存的数据单元相关联并且可以指示存储器模块中的与缓存存储器不同的并且存储缓存的数据单元的一组存储器单元;可以被布置成在多个标签与多个被请求的存储器组地址之间同时进行比较以提供比较结果的比较器阵列;其中每个被请求的存储器组地址可以指示存储器模块中的存储多个被请求的数据单元中的被请求的数据单元的一组存储器单元;争用评估单元;控制器,其可以被布置成:(a)基于该比较结果将多个被请求的数据单元分类为可以存储在缓存存储器中的缓存的数据单元和未被缓存的数据单元;以及(b)向争用评估单元发送关于缓存的和未被缓存的数据单元的信息;其中,该争用评估单元可以被布置成检查至少一个争用的发生;以及可以被布置为以无争用的方式向存储器模块请求任何未缓存的数据单元的输出接口。

[0058] 比较器阵列可以被布置成在单个收集单元时钟周期期间在多个标签与多个被请求的存储器组地址之间同时进行比较;并且其中,该争用评估单元可以被布置成在单个收集单元时钟周期期间检查该至少一个争用的发生。

[0059] 争用评估单元可以被布置成响应于缓存存储器的新标签而重新检查至少一个争用的发生。

[0060] 收集单元可以被布置成以流水线方式操作;其中,流水线的每个阶段的持续时间可以是一个收集单元时钟周期。

[0061] 存储体的一行中的每组存储器单元出自多个可独立存取的存储体;其中,该争用评估单元可以被布置成当两个未被缓存的数据单元属于相同的存储体中的不同行时确定发生潜在的争用。

[0062] 缓存存储器可以是全相联存储器缓存(fully associative memory cache)。

[0063] 收集单元可以包括地址转换器,其可以被布置成将包含在多个请求中的位置信息转换为多个被请求的存储器组地址。

[0064] 多个被请求的数据单元可以属于数据单元的阵列;其中,位置信息包括在数据单元的阵列内的多个被请求的数据单元的坐标。

[0065] 争用评估单元可以包括多组节点;其中,每组节点可以被布置为评估在多个被请求的存储器组地址与多个标签中的标签之间的争用。

[0066] 根据本发明的实施例，可以提供一种用于对检索多个被请求的数据单元的多个请求进行响应的方法，该方法可以包括：通过收集单元的输入接口接收用于检索多个被请求的数据单元的多个请求；通过可包括多个条目的缓存存储器存储多个标签和多个缓存的数据单元；其中，每个标签可以与缓存的数据单元相关联并且可以指示存储器模块中的与缓存存储器不同的并且存储缓存的数据单元的一组存储器单元；通过比较器阵列在多个标签与多个请求的存储器组地址之间同时进行比较以提供比较结果；其中，每个被请求的存储器组地址可以指示存储器模块中的存储多个被请求的数据单元中的被请求的数据单元的一组存储器单元；通过控制器基于比较结果将多个被请求的数据单元分类为可以存储在缓存存储器中的缓存的数据单元和未被缓存的数据单元；并且向争用评估单元发送关于缓存的和未被缓存的数据单元的信息；通过争用评估单元检查至少一个争用的发生；以及通过输出接口以无争用的方式向存储器模块请求任何未被缓存的数据单元。

[0067] 根据本发明的实施例，可以提供一种处理模块，其可以包括数据处理器阵列；其中，数据处理器阵列中的多个数据处理器中的每个数据处理器单元可以直接耦合到数据处理器阵列中的一些数据处理器，可以间接耦合到数据处理器阵列中的一些其他数据处理器，并且可以包括用于在数据处理器的转播端口之间转播数据的转播信道。

[0068] 多个数据处理器中的每个数据处理器的转播信道可以表现基本上为零的等待时间。

[0069] 多个数据处理器中的每个数据处理器可以包括核心；其中，核心可以包括算术逻辑单元和存储器资源；其中，多个数据处理器的核心可以通过可配置网络彼此耦合。

[0070] 多个数据处理器中的每个数据处理器可以包括可配置网络的多个数据流组件。

[0071] 多个数据处理器中的每个数据处理器可以包括可以直接耦合到第一组邻居(neighbor)的第一非转播输入端口。

[0072] 第一组邻居可以由数据处理器形成，该数据处理器可以位于距离数据处理器少于四个数据处理器的距离内。

[0073] 数据处理器的第一非转播输入端口可以直接耦合到第一组邻居的数据处理器的转播端口。

[0074] 数据处理器还可以包括第二非转播输入端口，其可以直接耦合到第一组邻居的数据处理器的非转播端口。

[0075] 数据处理器的第一非转播输入端口可以直接耦合到第一组邻居的数据处理器的非转播端口。

[0076] 第一组邻居可以由八个数据处理器组成。

[0077] 多个数据处理器中的每个数据处理器的第一转播端口可以直接耦合到第二组邻居。

[0078] 对于多个数据处理器中的每个数据处理器，第二组邻居不同于第一组邻居。

[0079] 对于多个数据处理器中的每个数据处理器，第二组邻居可以包括相比于属于第一组邻居的任何数据处理器更远离数据处理器的数据处理单元。

[0080] 除了多个数据处理器之外，处理器阵列还可以包括至少一个其他数据处理器。

[0081] 数据处理器阵列中的数据处理器可以按行和列进行布置。

[0082] 每行中的一些数据处理器可以以循环的方式彼此耦合。

[0083] 每行中的数据处理器可以由共享的微控制器来控制。

[0084] 多个数据处理器中的每个数据处理器可以包括配置指令寄存器;其中,指令寄存器可以被布置为在配置过程期间接收配置指令并且将配置指令存储在配置指令寄存器中;其中,给定行的数据处理器可以通过给定的共享微控制器来控制;其中,给定行中的每个数据处理器可以被布置为接收用于从给定的共享微控制器中选择所选择的配置指令的选择信息,并且在特定条件下配置数据处理器以根据所选择的配置指令进行操作。

[0085] 当数据处理器可以被布置为响应于选择信息时,该特定条件可以被满足;其中,当数据处理器可以被布置成忽略选择信息时,该特定条件可以不被满足。

[0086] 多个数据处理器中的每个数据处理器可以包括控制器、算术逻辑单元、寄存器文件和配置指令寄存器;其中,指令寄存器可以被布置成在配置过程期间接收配置指令并且将配置指令存储在配置指令寄存器中;其中,控制器可以被布置成接收用于选择所选择的配置指令的选择信息并且配置数据处理器以根据所选择的配置指令来操作。

[0087] 多个数据处理器中的每个数据处理器可以包括多达三个配置指令寄存器。

[0088] 根据本发明的实施例,可以提供一种用于操作可以包括数据处理器阵列的处理模块的方法;其中,该操作可以包括通过阵列中的数据处理器来处理数据;其中,数据处理器阵列中的多个数据处理器中的每个数据处理器单元可以直接耦合到数据处理器阵列中的一些数据处理器,可以间接耦合到数据处理器阵列中的一些其他数据处理器,以及使用一个或更多个数据处理器的一个或更多个转播信道在处理数据处理器的转播端口之间转播数据。

[0089] 根据本发明的实施例,可以提供一种图像处理器,其可以包括数据处理器阵列、第一微控制器、缓冲单元和第二微控制器;其中,阵列中的数据处理器可以被布置成在数据处理器配置过程期间接收数据处理器配置指令;其中,缓冲单元可以被布置成在缓冲单元配置过程期间接收缓冲单元配置指令;其中,第一微控制器可以被布置成通过向数据处理器提供数据处理器选择信息来控制数据处理器的操作;其中,数据处理器可以被布置成响应于数据处理器选择信息来选择所选择的数据处理器配置指令并且根据所选择的数据处理器配置指令来执行一个或更多个数据处理操作;其中,第二微控制器可以被布置为通过向缓冲单元提供缓冲单元选择信息来控制缓冲单元的操作;其中,缓冲单元可以被布置为响应于缓冲单元选择信息的至少一部分来选择所选择的缓冲单元配置指令并且根据所选择的缓冲单元配置指令来执行一个或更多个缓冲单元操作;并且其中,数据处理器选择信息的大小可以是数据处理器配置指令的大小的一部分。

[0090] 阵列中的数据处理器可以按照数据处理器组来进行布置,其中,不同的数据处理器组可以由不同的第一微处理器控制。

[0091] 数据处理器组可以是一行数据处理器。

[0092] 相同组的数据处理器中的数据处理器并行地接收相同的数据处理器选择信息。

[0093] 缓冲单元可以包括多个存储器资源组;其中,不同的存储器资源组可以被耦合到不同的数据处理器组。

[0094] 图像处理器可以包括第二微控制器;其中,不同的第二微控制器可以被布置来控制不同的存储器资源组。

[0095] 不同的存储器资源组可以是不同的移位寄存器组。

[0096] 不同的移位寄存器组可以被耦合到可以被布置成从存储器模块接收数据的多个缓冲器组。

[0097] 多个缓冲器组可以不受第二微控制器控制。

[0098] 缓冲单元选择信息选择在多个存储器资源组与多个数据处理器组之间的连接性。

[0099] 每个数据处理器可以包括算术逻辑单元和数据流组件;其中,数据处理器配置指令定义算术逻辑单元的操作码并且定义经由数据流组件到算术逻辑单元的数据流。

[0100] 图像处理器还可以包括可以包括多个存储体的存储器模块;其中,缓冲单元可以被布置成从存储器模块提取数据并且将数据发送到数据处理器阵列。

[0101] 第一微控制器共享程序存储器。

[0102] 每个第一微控制器可以包括存储第一指令地址、多个头指令和多个循环指令的控制寄存器。

[0103] 图像处理器可以包括可以耦合到缓冲单元的存储器模块;其中,存储器模块可以包括存储缓冲器、加载存储单元和多个存储体;其中,存储缓冲器可以由第三微控制器控制。

[0104] 存储缓冲器可以被布置成在存储缓冲器配置过程期间接收存储缓冲器配置指令;其中,第三微控制器可以被布置成通过向存储缓冲器提供存储缓冲器选择信息来控制存储缓冲器的操作。

[0105] 图像处理器可以包括可以由第三微处理器控制的存储缓冲器。

[0106] 第三微控制器、第一微控制器和第二微控制器可以具有相同的结构。

[0107] 根据本发明的实施例,可以提供一种图像处理器,其可以包括多个可配置电路和多个微控制器;其中,多个可配置电路可以包括存储器电路和多个数据处理器;其中,每个可配置电路可以被布置为存储高达有限量的配置指令;其中,多个微控制器可以被布置为通过向多个可配置电路重复提供选择信息来控制多个可配置电路,该选择信息用于通过每个可配置电路从有限量的配置指令中选择所选择的配置指令。

[0108] 多个可配置电路可以包括可以包括多个存储体的存储器模块;以及用于在存储器模块与数据处理器之间交换数据的缓冲单元。

[0109] 选择信息的大小不超过两个比特。

[0110] 根据本发明的实施例,可以提供一种用于配置可以包括多个可配置电路和多个微控制器的图像处理器的方法;其中,该多个可配置电路可以包括存储器电路和多个数据处理器;其中,该方法可以包括在每个可配置电路中存储高达有限量的配置指令;通过多个微控制器向多个可配置电路重复提供选择信息来控制多个可配置电路,该选择信息用于通过每个可配置电路从该有限量的配置指令中选择所选择的配置指令。

[0111] 根据本发明的实施例,可以提供一种用于操作图像处理器的方法,该方法可以包括:提供图像处理器,该图像处理器可以包括数据处理器阵列;可以包括多个存储体的存储器模块;缓冲单元;收集单元;和多个微控制器;通过多个微处理器、存储器模块和缓冲单元的一部分控制数据处理器阵列;通过缓冲单元从存储器模块中检索数据;通过缓冲单元将数据发送到数据处理器阵列;通过收集单元接收用于从存储器模块检索多个被请求的数据单元的多个请求;通过收集单元将多个被请求的数据单元发送到数据处理器阵列。

[0112] 根据本发明的实施例,可以提供一种用于配置图像处理器的方法,该图像处理器

可以包括数据处理器阵列、第一微控制器、缓冲单元和第二微控制器的阵列;其中,该方法可以包括在数据处理器配置过程期间向阵列中的数据处理器提供数据处理器配置指令;在缓冲单元配置过程期间向缓冲单元提供缓冲单元配置指令;通过第一微控制器向数据处理器提供数据处理器选择信息来控制数据处理器的操作;通过数据处理器响应于数据处理器选择信息选择所选择的数据处理器配置指令,并且根据所选择的数据处理器配置指令执行一个或更多个数据处理操作;通过第二微控制器向缓冲单元提供缓冲单元选择信息来控制缓冲单元的操作;通过缓冲单元响应于缓冲单元选择信息的至少一部分选择所选择的缓冲单元配置指令,并且根据所选择的缓冲单元配置指令执行一个或更多个缓冲单元操作;其中,数据处理器选择信息的大小可以是数据处理器配置指令的大小的一部分。

[0113] 根据本发明的实施例,可以提供一种非暂时性计算机可读介质,其存储用于对检索多个被请求的数据单元的多个请求进行响应的指令,该指令一旦被收集单元执行则导致执行以下步骤:通过收集单元的输入接口接收用于检索多个被请求的数据单元的多个请求;通过包括多个条目的缓存存储器存储多个标签和多个缓存的数据单元;其中,每个标签与缓存的数据单元相关联,并且指示存储器模块中的与缓存存储器不同的并存储缓存的数据单元的一组存储器单元;通过比较器阵列在多个标签与多个被请求的存储器组地址之间同时进行比较以提供比较结果;其中,每个被请求的存储器组地址指示存储器模块中的存储多个被请求的数据单元中的被请求的数据单元的一组存储器单元;通过控制器基于比较结果将多个被请求的数据单元分类为存储在缓存存储器中的缓存的数据单元和未被缓存的数据单元;并且向争用评估单元发送关于缓存的和未被缓存的数据单元的信息;通过争用评估单元检查至少一个争用的发生;以及通过输出接口以无争用的方式向存储器模块请求任何未被缓存的数据单元。

[0114] 根据本发明的实施例,可以提供一种存储用于操作处理模块的指令的非暂时性计算机可读介质,该指令一旦被处理模块执行则导致执行以下步骤:通过处理模块中的数据处理器阵列中的数据处理器来处理数据;其中,数据处理器阵列中的多个数据处理器中的每个数据处理器单元直接耦合到数据处理器阵列中的一些数据处理器,间接耦合到数据处理器阵列中的一些其他数据处理器,并且使用一个或更多个数据处理器的一个或更多个转播信道在数据处理器的转播端口之间转播数据。

[0115] 根据本发明的实施例,可以提供一种非暂时性计算机可读介质,其存储用于配置包括多个可配置电路和多个微控制器的图像处理器的指令;其中,多个可配置电路包括存储器电路和多个数据处理器,其中,该指令一旦被该图像处理器执行则导致执行以下步骤:在每个可配置电路中存储高达有限量的配置指令;通过该多个微控制器向多个可配置电路重复提供选择信息来控制多个可配置电路,该选择信息用于通过每个可配置电路从有限量的配置指令中选择所选择的配置指令。

[0116] 一种存储用于操作图像处理器的指令的非暂时性计算机可读介质,该图像处理器包括数据处理器阵列;包括多个存储体的存储器模块;缓冲单元;收集单元;和多个微控制器;其中,通过该图像处理器执行则导致执行以下步骤:通过该缓冲单元将数据发送到数据处理器阵列;通过收集单元接收用于从存储器模块检索多个被请求的数据单元的多个请求;通过收集单元将多个被请求的数据单元发送到数据处理器阵列。

[0117] 一种存储用于配置图像处理器的指令的非暂时性计算机可读介质,该图像处理器

包括数据处理器阵列、第一微控制器、缓冲单元和第二微控制器;其中,该多个可配置电路包括存储器电路和多个数据处理器,其中,该指令一旦被该图像处理器执行则导致执行以下步骤:在数据处理器配置过程期间向该阵列中的数据处理器提供数据处理器配置指令;在缓冲单元配置过程期间向缓冲单元提供缓冲单元配置指令;通过第一微控制器向数据处理器提供数据处理器选择信息来控制数据处理器的操作;通过数据处理器响应于数据处理器选择信息选择所选择的数据处理器配置指令,并且根据所选择的数据处理器配置指令执行一个或更多个数据处理操作;通过该第二微控制器向缓冲单元提供缓冲单元选择信息来控制缓冲单元的操作;通过缓冲单元响应于缓冲单元选择信息的至少一部分来选择所选择的缓冲单元配置指令,并且根据所选择的缓冲单元配置指令执行一个或更多个缓冲单元操作;其中,数据处理器选择信息的大小是数据处理器配置指令的大小的一部分。

[0118] 附图简述

[0119] 视为本发明的主题在说明书的结束部分中被特别指出并被清楚地要求保护。然而,当与附图一起阅读时,通过参考以下详细描述,可最好地理解本发明的关于操作的方法和组织以及其目的、特征和优点,其中:

[0120] 图1示出根据本发明的实施例的方法;

[0121] 图2示出根据本发明的实施例的图像处理器;

[0122] 图3示出根据本发明的实施例的图像处理器;

[0123] 图4示出根据本发明的实施例的图像处理器的一部分;

[0124] 图5示出根据本发明的实施例的时钟树;

[0125] 图6示出根据本发明的实施例的存储器模块;

[0126] 图7示出了根据本发明的实施例的在存储器模块的LSU与存储器模块的存储体之间的映射;

[0127] 图8示出根据本发明的实施例的存储缓冲器;

[0128] 图9和图10示出根据本发明的实施例的指令Row,Sel;

[0129] 图11示出根据本发明的实施例的缓冲单元;

[0130] 图12示出根据本发明的实施例的收集单元;

[0131] 图13是示出包括地址转换、缓存命中/未命中、争用和信息输出的过程的时序图;

[0132] 图14示出根据本发明的实施例的争用评估单元;

[0133] 图15和图16示出了根据本发明实施例的数据处理单元;

[0134] 图17示出根据本发明的实施例的扭曲计算方法;

[0135] 图18和图19示出了根据本发明的实施例的执行扭曲计算的数据处理器阵列;

[0136] 图20示出根据本发明的实施例的从各种数据处理器输出的扭曲参数;

[0137] 图21示出根据本发明的实施例的执行扭曲计算的数据处理器组;

[0138] 图22示出根据本发明的实施例的扭曲计算方法;

[0139] 图23示出根据本发明的实施例的处理单元的组;

[0140] 图24示出了第一子组源像素、第一子组目标像素、第二子组源像素和第二子组目标像素;

[0141] 图25示出具有中心像素SB的源像素的子组SG(B);

[0142] 图26示出了具有中心像素TB的目标像素(未示出)的对应子组TG(B);

[0143] 图27示出根据本发明的实施例的方法；

[0144] 图28示出了根据本发明实施例的通过DPA处理的八个源像素和三十二个目标像素；

[0145] 图29示出根据本发明的实施例的源像素的阵列；

[0146] 图30示出根据本发明的实施例的目标像素的阵列；

[0147] 图31示出了根据本发明的实施例的多组数据处理器DPU；

[0148] 图32示出了根据本发明的实施例的八组数据处理器DPU-每个组包括四个DPU;和

[0149] 图33示出根据本发明的实施例的扭曲计算方法。

[0150] 附图的详细描述

[0151] 在以下详细描述中,阐述了许多具体细节以便提供对本发明的透彻理解。然而,本领域技术人员将理解,可以在没有这些具体细节的情况下实践本发明。在其他情况下,没有详细描述公知的方法、过程和部件,以免模糊本发明。

[0152] 视为本发明的主题在说明书的结束部分中被特别指出并被清楚地要求保护。然而,当与附图一起阅读时,通过参考以下详细描述,可最好地理解本发明的关于操作的方法和组织以及其目的、特征和优点。

[0153] 应当理解,为了说明的简单和清楚,图中所示的元件不一定按比例绘制。例如,为了清楚起见,一些元件的尺寸可以相对于其他元件被放大。此外,在认为适当的情况下,参考数字可在图中重复以指示对应或类似的元件。

[0154] 因为本发明的所示实施例在很大程度上可以使用本领域技术人员已知的电子部件和电路来实现,所以细节将不会在比如上所述的必须考虑的那些在更大程度上进行解释,以便理解和领悟本发明的基本概念,并且以便不混淆或分散本发明的教导。

[0155] 说明书中对方法的任何引用应当经必要修改而应用于能够执行该方法的系统,并且应当经必要修改而应用于非暂时性计算机可读介质,该非暂时性计算机可读介质存储一旦由计算机执行就导致执行方法的指令。例如,原始提交的权利要求1-17、18、19、20-21和25-42以及97-99中的任何方法步骤可以由系统执行。在这个意义上,系统可以是图像处理器、收集单元或图像处理器的任何组件。可以提供一种存储指令的非暂时性计算机可读介质,一旦该指令被计算机执行就导致执行原始提交的权利要求1-17、18、19、20-21、25-42和97-99中的每一个的方法。

[0156] 说明书中对系统和任何其他组件的任何引用应当经必要修改而应用于可以由存储器设备执行的方法,并且应当经必要修改而应用于存储可以由存储器设备执行的指令的非暂时性计算机可读介质。例如,可以提供由权利要求44-52中任一项所描述的图像处理器执行的方法和/或方法步骤。例如,可以提供由权利要求76-93中任一项所描述的图像处理器执行的方法和/或方法步骤。

[0157] 说明书中对非暂时性计算机可读介质的任何引用应当经必要修改而应用于能够执行存储在非暂时性计算机可读介质中的指令的系统,并且应当经适当修改而应用于可以由计算机执行的方法,该计算机读取存储在非暂时性计算机可读介质中的指令。

[0158] 可以提供任何附图、说明书的任何部分和/或任何权利要求中列出的任何模块或单元的任何组合。特别是可以提供任何要求保护的特征的任何组合。

[0159] 像素可以是由相机获得的图像元素,可以是经处理的图像元素。

[0160] 术语“行(row)”和“行(line)”以可互换的方式使用。

[0161] 术语汽车被用作车辆的非限制性示例。

[0162] 为了简化说明,一些附图和一些以下文字包括数字示例(例如总线宽度、存储器行数,寄存器数量、寄存器长度、数据单元大小、指令大小、每单元或模块的组件数量、微处理器的数量、阵列中每行和/或每列的数据处理器的数量)。每个数字示例仅仅是一个非限制性示例。

[0163] 图1示出根据本发明的实施例的方法90。

[0164] 系统90可以是DAS,自动驾驶汽车控制模块的一部分等。

[0165] 系统90可以安装在汽车10中。系统90的至少一些部件在车辆内。

[0166] 系统90可以包括第一相机81、第一处理器83、存储单元85、人机接口86和图像处理器100。这些组件可以经由总线或网络82或通过任何其他布置而彼此耦合。

[0167] 系统90可以包括附加的相机和/或附加的处理器和/或附加的图像处理器。

[0168] 第一处理器83可以确定图像处理器100应当执行哪个任务并且指示图像处理器100相应地操作。

[0169] 要注意的是,图像处理器100可以是第一处理器83的一部分,并且它可以是任何其他系统的一部分。

[0170] 人机接口86可以包括显示器、扬声器、一个或更多个发光二极管、麦克风或任何其他类型的人机接口。人机接口可以与汽车驾驶员的移动设备进行通信,与汽车的多媒体系统等进行通信。

[0171] 图2示出根据本发明的实施例的图像处理器100。

[0172] 主端口101和从端口103提供在系统90的图像处理器100与任何其它组件之间的接口。

[0173] 图像处理器100包括:

[0174] 1)用于访问诸如存储单元85的外部存储器资源的直接存储器访问(DMA)。

[0175] 2)控制器,诸如但不限于标量单元104。

[0176] 3)标量单元(SU)程序存储器106。

[0177] 4)标量单元(SU)数据存储器108。

[0178] 5)存储器模块(MM)200。

[0179] 6)MM控制单元290。

[0180] 7)收集单元(GU)300。

[0181] 8)缓冲单元(BU)400。

[0182] 9)BU控制单元490。

[0183] 10)数据处理阵列(DPA)500。

[0184] 11)DPA控制单元590。

[0185] 12)配置总线130。

[0186] 13)多路复用器和缓冲器110、112、114、116、118、120。

[0187] 14)总线132、133、134、135、136和137。

[0188] 15)PMA状态和配置缓冲器109。

[0189] 图像处理器100还包括多个微控制器。为了简化说明,这些微控制器在图4中示出。

[0190] DMA 102耦合到多路复用器112、114和120。标量单元104耦合到缓冲器118和多路复用器112。缓冲器118被耦合到多路复用器116。缓冲器110耦合到多路复用器112、116和114。多路复用器112耦合到SU程序存储器106。多路复用器114耦合到SU数据存储器108。

[0191] 存储器单元200经由(单向)总线132耦合到收集单元300，经由(单向)总线134耦合到缓冲单元400并且经由(单向)总线133耦合到DPA500。

[0192] 收集单元300经由(单向)总线135耦合到缓冲单元400并经由(单向)总线137耦合到DPA 500。缓冲单元400经由(单向)总线136耦合到DPA 500。

[0193] 图像处理器的单元可以通过其他总线、通过额外的少数总线、通过互连和/或网络、通过其他宽度和方向性的总线等相互耦合。

[0194] 要注意的是，图2的收集单元300、缓冲单元400、DPA 500、存储器模块200、标量单元104、SU程序存储器106、SU数据存储器108和任何其他多路复用器和/或缓冲器可以以其他方式、通过附加的和/或其他的公共总线、网络、网格等彼此耦合。

[0195] 标量单元104可以控制图像处理器100的其他部件执行任务。标量单元104可以接收(例如来自图1的第一处理器83的)执行哪些任务的指令并且可以从SU程序存储器106提取相关指令。

[0196] 标量单元104可以确定SB控制单元290、BU控制单元490和DPA控制单元590内的微控制器将执行哪些程序。

[0197] 由SB控制单元290的微控制器执行的程序控制存储器模块200的存储缓冲器(未示出)。由BU控制单元490的微控制器执行的程序控制缓冲单元400。由DPA控制单元590的微控制器执行的程序控制DPA 500的数据处理单元。

[0198] 任何所述微控制器可以通过提供用于选择已经存储在受控模块或单元中的配置指令的短的选择信息(例如2-3比特、少于一个字节或比所选择的配置指令的比特数小的任何数目的比特)来控制任何模块或单元。这允许减少流量并执行快速的配置更改(因为配置更改可能需要在已存储在相关单元或模块中的不同配置寄存器之间进行选择)。

[0199] 应该注意的是，控制单元的数量和它们在图像处理器的组件之间的分配可能不同于图2中所示的那些。

[0200] 存储器模块200是图像处理器100的最高级的存储器资源。缓冲单元400和收集单元300是图像处理器100的较低级的存储器资源，并且可以被配置为从存储器模块200提取数据并将数据提供给DPA 500。DPA 500可以将数据直接发送到存储器模块200。

[0201] DPA 500包括多个数据处理器并被布置成执行计算任务，例如但不限于图像处理算法。图像处理算法的非限制性示例包括扭曲算法(warp algorithm)、视差等。

[0202] 收集单元300包括缓存存储器。收集单元300被配置为从DPA 500接收请求，以从缓存存储器或从存储器单元中提取多个数据单元(诸如像素)并提取被请求的像素。收集单元300可以以流水线的方式操作并且具有非常低的延迟(例如，一个(或少于五个或十个)时钟周期)的有限数量(例如三个)的流水线级。如下所示，收集单元也可以在附加模式下提取数据单元，同时使用存储器模块的地址生成器来提取信息。

[0203] 缓冲单元400被配置为充当存储器模块200与DPA 500之间的数据的缓冲器。缓冲单元400可以被布置为并行地向DPA 500的多个数据处理器提供数据。

[0204] 配置总线130耦合到DMA 102、存储器模块200、收集单元300、缓冲单元400和DPA

500。

[0205] DPA 500展示了可以支持并行和流水线实现的架构。它展示出灵活的连接性，能够将几乎每一个数据处理单元(DPU)连接到每一个DPU。

[0206] 图像处理器100的单元由紧凑型微处理器控制，该微处理器可以执行零延迟循环并且可以实现嵌套循环。

[0207] 图3示出了根据本发明的实施例的图像处理器100。

[0208] 图3提供了各种总线的宽度以及存储器模块200、收集单元300、缓冲单元400和DPA 500的内容的非限制性示例。

[0209] DPU 500可以包括6行6列的数据处理单元(DPU)510(0,0)-510(5,15)。

[0210] 配置总线130是32字节宽。

[0211] 总线132是8×64字节宽。

[0212] 总线134是6×128字节宽。

[0213] 总线135是2×128字节宽。

[0214] 总线137是2×16×16字节宽。

[0215] 总线133是6×2×16×16字节宽。

[0216] 总线136是2×16×16字节宽。

[0217] 存储器模块200被示出为包括地址生成器、6个加载存储单元、16个多端口存储器接口以及16个8个字节线的可独立访问的存储体。

[0218] 收集单元300包括缓存存储器，其包括18个寄存器，每个寄存器8字节。

[0219] 缓冲单元400包括6行4列16字节寄存器以及6行16列2:1多路复用器。

[0220] 图4示出根据本发明的实施例的图像处理器100的一部分。

[0221] 存储器模块200的两个存储缓冲器可以由SB控制单元290控制。SB控制单元290可以包括SB程序存储器292和SB微控制器291和292。SB程序存储器293存储要由SB微控制器291和292执行的指令。SB微控制器291和292可以通过(存储在配置寄存器298中的)信息(通过配置总线130和/或通过标量单元104)进行馈送，该信息指示要执行(存储在SB程序存储器293中的指令之中的)哪些指令。

[0222] 可以通过BU控制单元490控制缓冲单元400的不同的寄存器行。BU控制单元490可以包括BU程序存储器497、配置寄存器498和BU微控制器491-496。

[0223] BU程序存储器297存储要由BU微控制器491-496执行的指令。BU微控制器491-496可以通过指示执行(存储在BU程序存储器497中的指令中的)哪些指令的信息(存储在配置寄存器498中)进行馈送(通过配置总线130和/或通过标量单元104)。

[0224] DPA 500的不同行的DPU可以由DPA控制单元590控制。DPA控制单元590可以包括DPA程序存储器597、配置寄存器598和DPA微控制器591-596。

[0225] DPA程序存储器297存储要由DPA微控制器591-596执行的指令。DPA微控制器591-596可以通过指示执行(存储在DPA程序存储器597中的指令中的)哪些指令的信息(通过配置总线130和/或通过标量单元104)进行馈送。

[0226] 要注意的是，微控制器可以以其他方式分组。例如，可以有一个微处理器组、两个、三个或三个以上的微处理器组。

[0227] 图5示出根据本发明的实施例的时钟树。

[0228] 输入时钟信号2131被馈送到标量单元104。标量单元将clk_mem 2132发送到存储器模块的存储体610-625，并且将clk 2133发送到存储器模块200的缓冲单元400、收集单元300和加载存储单元(LSU)630-635。clk2133被转换为发送到DPA 500的dpa_clk 2134。

[0229] 存储器模块

[0230] 图6示出根据本发明的实施例的存储器模块200。图7示出根据本发明的实施例的存储器模块的LSU与存储器模块的存储体之间的映射。

[0231] 存储器模块200包括16个可独立访问的存储体M0-M15 610-625、6个加载存储单元LSU0-LSU5 630-625、大小地址生成器AG0-AG5 640-645以及两个存储缓冲器650和660。

[0232] 存储体M0-M15 610-625是八字节宽(每行有64比特)，并包含1K行以提供96KB的总存储容量。每个存储体可以包括(或可以被耦合到)多端口存储器接口，用于在发送到存储体的请求之间进行仲裁。

[0233] 在图6中，有四个客户端耦合到每个存储体(四个箭头)，且多端口存储器接口必须在出现在这四个输入端上的访问请求之间进行仲裁。

[0234] 多端口存储器接口可以应用任何仲裁方案。例如，它可以应用基于优先级的仲裁。

[0235] 每个LSU可以从地址生成器中选择6个地址中的一个，并且连接到4个存储体，并且每次访问可以访问(来自2个存储体的)16个字节，使得6个LSU一次可以访问16个存储体中的12个。

[0236] 图7示出了不同值的控制信号SysMemMap之间的映射与LSU 630-635与存储体M0-M15 610-625之间的映射。

[0237] 图6示出了存储器模块200通过八个八字节宽总线(总线132的部分)将数据单元输出到收集单元，并且经由六个十六字节宽总线(总线134的一部分)将数据单元输出到缓冲单元。

[0238] AG0-AG5 640-645中的每个地址生成器可以通过使用以下变量和寄存器来实现四维(4D)迭代器：

[0239] Baddr定义了存储体中的基地址。

[0240] 'W'方向-变量wDepth定义W方向上一个步长的字节距离。变量wCount定义W计数器的最大值-达到此值时，zCounter递增，并清除wCounter。

[0241] 'Z'方向-zArea定义Z方向一个步长的字节距离，变量zCount定义Z计数器最大值。达到此值时，X计数器递增，并清除Z计数器。

[0242] 'X'方向-变量xStep定义步长(可以是1、2、4、8或16字节)。变量xCount定义下一个'Y'之前的X计数器最大值

[0243] 'Y'方向-变量stride定义了连续“行”的起点之间的字节距离。变量yCount定义Y计数器的最大值。

[0244] 当所有计数器达到其最大值时，会产生停止条件。

[0245] 生成的地址是:Addr＝BAddr+wCount*wDepth+xcounter*xstep+ycounter*Stride+zcounter*Area。

[0246] 变量存储在可以通过配置总线130配置的寄存器中。

[0247] 下面是地址生成器配置映射的示例：

位移	大小	寄存器	说明
0x400	4B		保留
0x404	4B	ag_AddReg	基地址和当前地址
0x408	4B	ag_wCount	W 方向最大计数器
0x40C	4B	ag_wDepth	W 方向步长（深度）
0x410	4B	ag_zCount	Z 方向最大计数器
0x414	4B	ag_xzCount	X 和 Z 方向最大计数器
0x418	4B	ag_zArear	Z 方向步长（区域）
[0248] 0x41C	4B	ag_xCount	X 方向最大计数器
0x420	4B	ag_xStep	X 方向步长（模 8）
0x424	4B	ag_yCount	Y 方向最大计数器
0x428	4B	ag_Stride	Y 方向步长（stride）
0x42C	4B	ag_DlySz	[7 : 0]延迟: 哑元数访问真正的之前 [9 : 8]大小: 0 : 2B，1 : 4B，2 : 8B，3 : 16B
0x430	16B		保留

[0249] 存储也可以以不同于2、4、8和16字节的大小进行写入。

[0250] 每个LSU可以往来于所连接的4个存储体中的2个存储体(16个字节)执行加载/存储操作。

[0251] 被访问的存储体可以依赖于所选择的映射(例如参见图7)和地址。

[0252] 待存储的数据被准备于在下面描述的存储缓冲器650和660中的一个中。

[0253] 每个LSU可以选择从6个地址生成器AG0-AG5中的一个生成的地址。

[0254] 加载操作

[0255] 从存储体读取的数据被存储在加载存储单元的缓冲器(未示出)中,然后(经由总线134)被传送到缓冲单元400。该缓冲器有助于避免由于存储体上的争用而造成的停顿。

[0256] 存储操作

[0257] 要存储在存储体中的数据被准备于存储缓冲器650和660中的一个中。有2个存储缓冲器(存储缓冲器0 650和存储缓冲器1 660)。每个存储缓冲器可以请求在一个和四个16

字节的字之间写入LSU之一。

[0258] 每个LSU因此可以获得多达8个同时的请求，并以预定顺序一个接一个地授予：1)存储缓冲器0-字0，2)存储缓冲器0-字1，...4)存储缓冲器1-字0，...，8)存储缓冲器1-字3。

[0259] 当存储缓冲器被配置为以条件存储模式操作时，存储缓冲器可以忽略(不发送到存储体)或处理(发送到存储体)数据。

[0260] 当存储缓冲器被配置成以分散模式操作时，可以将由其接收的数据单元的一部分视为与剩余的数据单元的存储相关联的地址。

[0261] LSU运行优先级。存储操作优先于加载操作，使得存储将不会由于与加载争用而产生停顿。因为加载操作使用缓冲器，因此通常争用会被抑制(swallowed)而不产生停顿状况。

[0262] 存储缓冲器

[0263] 存储缓冲器650和660由存储缓冲器微控制器291和292控制。

[0264] 在配置过程中，存储缓冲器650和660中的每一个接收(并存储)三个配置指令(sb_instr[1]-sb_instr[3])。不同存储缓冲器的配置指令(也称为存储缓冲器配置指令)可以彼此不同或者可以相同。

[0265] 在配置过程中，每个存储缓冲器微控制器接收由每个存储缓冲器微控制器执行的指令的地址。首个和最后一个PC指示要从存储缓冲器程序存储器293读取的首个和最后的指令。在以下配置示例中，还定义了每个存储缓冲器微控制器的程序存储器的位置：

[0266] 存储缓冲器配置

地址	大小	寄存器
0x0004_4900	16B	存储缓冲器 0
0x0004_4900	4B	sb_Instr [0] - 哑元 - NOP
0x0004_4904	4B	sb_Instr [l]
0x0004_4908	4B	sb_Instr [2]
0x0004_490C	4B	sb_Instr [2]

[0267]

0x0004_4910	16B	存储缓冲器 1
0x0004_4910	4B	sb_Instr [0] - 哑元 - NOP
0x0004_4914	4B	sb_Instr [l]
0x0004_4918	4B	sb_Instr [2]
0x0004_491C	4B	sb_Instr [3]
0x0004_4920		保留

[0268] 存储缓冲器微控制器配置映射：

[0269] 0x0004_4940 32B sb_uCPM[0][0:15]用于uC0的程序存储器

[0270] 0x0004_4960 32B sb_uCPM[1][0:15]用于uC1的程序存储器

[0271] 0x0004_4980 4B sb_FLIPC0用于SB uC0的首个和最后一个PC

[0272] 0x0004_4984 4B sb_FLIPC1用于SB uC1的首个和最后一个PC

[0273] 0x0004_4988 保留

[0274] 存储缓冲器的微控制器指令可以是执行指令或do循环指令。他们有以下格式：

[0275] 执行指令：

[0276] [8:0]RLC:重复指令计数器:1...495:立即-496...511:间接计数器寄存器

[0277] [10:9]sel:存储缓冲器指令选择。触发器在非零时存储

[0278] [13:11]行:DPA行选择

[0279] [14]保留

[0280] [15]类型0

[0281] Do循环：

[0282] [8:0]RLC:重复循环计数器:0...495:立即-496...511:间接计数器寄存器

[0283] [13:9]长度:循环长度:1...16

[0284] [14]模式:0:计数循环,1:计数周期

[0285] [15]类型1

[0286] 图8示出根据本发明的实施例的存储缓冲器660。

[0287] 存储缓冲器660具有四个多路复用器661-664、四个缓冲器字0-字3671-674和四个多路分解器681-684。

[0288] 缓冲器字0 671耦合在多路复用器661和多路分解器681之间。缓冲器字1 672耦合在多路复用器662和多路分解器682之间。缓冲器字2 673耦合在多路复用器663和多路分解器683之间。缓冲器字3 674耦合在多路复用器664和多路分解器684之间。

[0289] 多路复用器661-664中的每一个具有用于接收总线133的不同线路的四个输入端并且由控制信号Row,Sel来控制。

[0290] 多路分解器681-684中的每一个具有六个输出端,用于向LSU0-LSU5中的任一个提供数据,并由控制信号En,LSU控制。

[0291] 存储缓冲器配置指令控制存储缓冲器的操作,且甚至生成命令Row,Sel和En,LSU。

[0292] 配置指令格式示例提供如下：

[0293] 存储缓冲器指令编码

[2:0] lsu0 通过 LSU # lsu0 写入字 0

[3] En0 写入字 0

[4]有条件的

[5]分散

[8:6] lsu 1 通过 LSU # lsu 1 写入字 1

[9] Enl 写入字 1

[10] 有条件的

[11] 分散

[0294] [14:12] lsu2 通过 LSU # lsu2 写入字 2

[15] En2 写入字 2

[16]有条件的

[17]分散

[20:18] lsu 3 通过 LSU # lsu3 写入字 3

[21] En3 写入字 3

[22]有条件的

[23]分散

[28:24] sel 数据选择

[0295] 称为“数据选择”的五个位实际上是指令Row,Sel,并在图9和图10中被示出。0到28之间的值映射到DPA 500的DPU的不同端口。在图9中,D#和E#分别表示DPU[row,#]的输出D和E,其中'row'是当前所选指令的行选择,且D'#是DPU[row+1,#]的输出D。

[0296] 缓冲单元

[0297] 图11示出根据本发明的实施例的缓冲单元400。

[0298] 缓冲单元400包括统一标记为402的读取缓冲器(RB)、寄存器文件(RF)404、缓冲单元内部网络408、多路复用器控制电路471-476、输出多路复用器406、BU配置寄存器401(1)-401(5)(每个存储两个配置指令)、历史配置缓冲器405和BU读取缓冲器配置寄存器403。

[0299] BU微控制器可以为每一行选择读取(401(1)-401(5)中的每个BU配置寄存器中存储的两个配置指令中的)哪个配置指令。

[0300] 有六行多路复用器,且它们包括多路复用器491(0)-491(15)和491'(0)-491'(15)、多路复用器492(0)-492(15)和492'(0)-492'(15)、多路复用器493(0)-493(15)和

493’(0)-493’(15)、多路复用器494(0)-494(15)和494’(0)-494’(15)和多路复用器495(0)-495(15)和495’(0)-495’(15)。

[0301] 为了简化说明,图11仅示出了多路复用器控制电路471和476。

[0302] 缓冲单元内部网络408将读取缓冲器402耦合到寄存器文件404。

[0303] 第一行的四个读取缓冲器415、416、417和417(经由缓冲单元内部网络408)耦合到第一行的四个寄存器R3 413、R2 412、R1 411和R0 410。

[0304] 第二行的四个读取缓冲器425、426、427和427(经由缓冲单元内部网络408)耦合到第二行的四个寄存器R3 423、R2 422、R1 421和R0 420。

[0305] 第三行的四个读取缓冲器435、436、437和437(经由缓冲单元内部网络408)耦合到第三行的四个寄存器R3 433、R2 432、R1 431和R0 430。

[0306] 第四行的四个读取缓冲器445、446、447和447(经由缓冲单元内部网络408)耦合到第四行的四个寄存器R3 443、R2 442、R1 441和R0 440。

[0307] 寄存器文件的不同行和多路复用器的相应行由491-495中不同的BU微控制器控制。

[0308] 还通过DPU微控制器控制不同行的多路复用器。特别地,591-596的每个DPU微控制器控制相应的DPU行,并且将控制指令(MuxCtl)发送到相应的多路复用器行(经由多路复用器控制电路471-476)。每个多路复用器控制电路存储最后(例如十六个)MuxCtl指令(指令历史),并且历史配置缓冲器405存储用于确定将哪个MuxCtl指令发送到多路复用器行的选择信息。

[0309] 多路复用器控制电路471控制第一行的多路复用器并包括用于存储从DPU微控制器491发送的MuxCtl指令的FIFO 481(1),并且包括控制多路复用器481(2)以选择从FIFO 481(1)中取出哪个存储的MuxCtl指令并发送到包括多路复用器491(0)-491(15)和491’(0)-491’(15)的第一行的多路复用器。

[0310] 多路复用器控制电路476控制第六行多路复用器并包括用于存储从DPU微控制器496发送的MuxCtl指令的FIFO 486(1)并包括控制多路复用器486(2)以选择从FIFO 482(1)中取出哪个存储的MuxCtl指令并发送到包括多路复用器495(0)-495(15)和495’(0)-495’(15)的第六行的多路复用器。

[0311] 寄存器文件可以由BU微控制器控制。由寄存器文件执行的操作可以包括(a)从最高到最低有效位的移位,其中移位的跳跃是2字节(或任何其他值)的幂,(b)从读取缓冲器加载一个或两个寄存器。文件寄存器中的内容可以被操纵。例如,内容可以被交织和/或隔行扫描。以下提供的指令集中提供了一些示例。

[0312] 缓冲配置映射包括配置缓冲器的地址,用于存储用于缓冲单元的配置指令并存储由缓冲单元微控制器(BUuC0-BuuC5)提取哪些命令的指示。后者被称为的BU指令对的第一个和最后一个PC(每个三十二位MuxConfig指令包括两个单独的缓冲单元配置指令):

[0313] 缓冲单元配置映射

[0314]

地址	大小	寄存器
0x0004_4200	256B	缓冲单元
0x0004_4200	128B	bu_uCPM [0:63] 微控制器程序存储器
0x0004_4280	64B	bu_uCCounters [0:15] BU 微控制器间接计数器
0x0004_42c0	4B	bu_MuxConfig0 [31:0]
0x0004_42c4	4B	bu_MuxConfig1 [31:0]
0x0004_42c8	4B	bu_MuxConfig2 [31:0]
0x0004_42cc	4B	bu_MuxConfig3 [31:0]
0x0004_42d0	4B	bu_MuxConfig4 [31:0]
0x0004_42d4	4B	bu_MuxConfig5 [31:0]

[0315]

0x0004_42d8	4B	bu_history [23:0]
0x0004_42dc	4B	bu_RBSrcCnf [23:0]
0x0004_42E0	4B	bu_FLIPC0 BUuC0 的第一个和最后一个 PC
0x0004_42E4	4B	bu_FLIPCl BUuCl 的第一个和最后一个 PC
0x0004_42E8	4B	bu_FLIPC2 BUuC2 的第一个和最后一个 PC
0x0004_42EC	4B	bu_FLIPC3 BUuC3 的第一个和最后一个 PC
0x0004_42F0	4B	bu_FLIPC4 BUuC4 的第一个和最后一个 PC
0x0004_42F4	4B	bu_FLIPC5 BUuC5 的第一个和最后一个 PC

[0316] 由BU微控制器执行的指令包括用于循环控制或指令重复的位[0:8],并且包括包含控制指令执行的值的[9:13]位。对于寄存器文件命令和do循环命令都是如此。

[0317] 指令编码:

[0318] RFL代表寄存器文件行并且RBL代表读取缓冲器行。

[0319] RRR[r/r1:r0]代表编号“r”(16个字节)/从寄存器r1到t0的寄存器(RFL或RBL)。RRR[r][b/b1:b0]表示编号“r”字节b/从字节b1到b0的寄存器(RFL或RBL)。

[0320] RF指令:(每个周期,BU微控制器可以将位9-14发送到BU)

[0321] [8:0]RIC:重复指令计数器:1...495:立即-496...511:间接计数器寄存器

[0322] [11:9]移位:0:NOP,1:1B,2:2B,3:4B,4:8B,5:1R,6:2R,7:1L,8-15:Res

[0323] [14:12]加载:

[0324] 0:NOP

[0325] 1:单精度型A:RFL[3]<=RBL[0]

[0326] 2:单精度型B:RFL[2]<=RBL[0]

[0327] 3:双精度型:RFL[2]<=RBL[0],RFL[3]<=RBL[1]

[0328] 4:交错字节型:RFL[2:3]<={RBL[1][15],RBL[0][15],…,RBL[0][0]}

[0329] 5:交错短整型:RFL[2:3]<={RBL[1][15:14],RBL[0][15:14],…,

[0330] RBL[1][3:2],RBL[0][3:2],RBL[1][1:0],RBL[0][1:0]}

[0331] 6:交错字节型&0:RFL[2:3]<={0,RBL[0][15],…,0,RBL[0][0]}

[0332] 7:交错短整型&0:RFL[2:3]<={0,RBL[0][15:14],…,0,RBL[0][1:0]}

[0333] [15]类型:0

[0334] Do循环:

[0335] [8:0]RLC:重复循环计数器:0...495:立即-496...511:间接计数器寄存器

[0336] [13:9]长度:循环长度:1...16

[0337] [14]模式:0:计数循环,1:计数周期

[0338] [15]类型1

[0339] 读取缓冲区加载。

[0340] 从LSU的RB加载操作是由配置控制的和由其状态和相关LSU的状态自触发的。BU读取缓冲器配置寄存器(也被称为RBSrcCnf)403指定从哪个LSU加载的每个RB线路。

[0341] 存储在BU读取缓冲器配置寄存器403中的配置指令具有以下格式:

[0342] [3:0]读取缓冲器0LSU源:

[0343] [7:4]读取缓冲器1LSU源

[0344] [11:8]读取缓冲器2LSU源

[0345] [15:12]读取缓冲器3LSU源

[0346] [19:16]读取缓冲器4LSU源

[0347] [23:20]读取缓冲器5LSU源

[0348] “读取缓冲器”是指读取缓冲器的行。每个读缓冲器行的四个位可以具有以下含义:0-7:无负载,8:LSU0、9:LSU1、10:LSU2、11:LSU3、12:LSU4、13:LSU5、14:GU0、15:GU1(GUI交换8字节加载为d8..d15,d0..d7代替d0..d15或在短整型模式下的GU最后8个短整型输出)。

[0349] 多路复用器配置

[0350] 多路复用器的配置如下所示(这个示例涉及第一行,并且我的指令MuxCtl表示为MuxCt10):

[0351] MuxCt10[Row][4:0]:BRfpSelB://[0-7]:BSelB0,8-19:{RSelB0,BSelB0},31:fpB0,

[0352] MuxCt10[Row][7:5]:BSelAb//Reg[4][BSelAb*2+31]

[0353] MuxCt10[Row][8]:fpA0//MuxA(col<<2)的浮点模式

[0354] MuxCtl0[Row][11:9]:BSelBb//Reg[5][BSelAb*2+31]

[0355] MuxCtl0[Row][15:12]:保留

[0356] MuxCtl通过以下方式操作Muxes(选择寄存器文件的寄存器和DPU的端口):

[0357] PxlA[Row,Col]＝!fpA?Reg[Row][Col*2]:REG[Row][Col*4]

[0358] PxlFpB[Row,Col]＝Reg[Row][Col*4+2]

[0359] PxlB[Row,Col]＝BSelB＝＝0x1f?

[0360] PxlFpB[Row,Col]:

[0361] BSelB<8?

[0362] REG[Row][BSelB+Col*2]:

[0363] Reg[((BSelB-8)/2+Row)%6][(BSelB&1)+Col*2]

[0364] PxlAb[Row,0]＝Reg[4][BSelAb*2+32]

[0365] PxlBb[Row,0]＝Reg[5][BSelBb*2+32]

[0366] 可以通过读取存储每行的多路复用器的选择信息的四位的历史配置缓冲器405的内容来完成(由FIFO 471-476)对历史的选择。

[0367] 收集单元

[0368] 图12示出根据本发明的实施例的收集单元300。

[0369] 收集单元300包括输入缓冲器301、地址转换器302、缓存存储器303、地址标签比较器304、争用评估单元306、控制器307、存储器接口308、迭代器310和配置寄存器311。

[0370] 收集单元300被配置为通过包括16个8字节的寄存器的全关联缓存存储器(CAM)303从八个存储体MB0..MB7或MB8..MB15收集高达16个字节或短整型(short)像素,这取决于配置寄存器311的位gu_Ctrl[15]。像素地址或坐标是根据配置寄存器311的模式gu_Ctrl[3:0]从阵列产生或接收的。

[0371] 收集单元300可以使用指示存储体内的存储体编号和行的地址(二联体(duplets))来访问存储器模块200的存储体。

[0372] 收集单元可以接收或生成代替地址的表示图像中被请求的像素的位置的X和Y坐标。收集单元包括用于将X,Y坐标转换为地址的地址转换器302。

[0373] 迭代器310可以以两种模式之一操作-(a)仅内部迭代器和(b)使用存储器模块的地址生成器。

[0374] 当在模式(a)中操作时,迭代器310可以使用以下控制参数(其存储在配置寄存器311中)生成十六个地址:

[0375] 1)AddBase-16个基地址。

[0376] 2)AddStep-迭代器地址步长。

[0377] 3)xCount:stride之前的最大步数计数器。stride从前一步或基础坐标进行。

[0378] 4)AddStride:stride(或y步长)。

[0379] 在模式(b)下操作时,迭代器将AddBase提供给地址生成器,且地址生成器使用该地址执行迭代。

[0380] 例如,在视差计算期间,迭代器模式可能是有用的,其中收集单元可以从源图像和目标图像(特别是彼此接近的像素)检索数据单元。

[0381] 另一种操作模式包括接收被请求的数据的地址(地址可以是地址转换器302要转

换的X,Y坐标或诸如二联体的存储器地址),检查被请求的数据单元是否存储在缓存存储器中,且如果不是,则从存储器模块200提取数据单元。

[0382] 另一种操作模式包括接收被请求的数据单元的地址并将地址转变成更多被请求的地址,然后从缓存存储器303或从存储器单元中提取更多被请求的地址的内容。例如,在扭曲计算期间,这种操作模式可能是有用的,其中收集单元可以接收像素的地址并获得像素和几个其他相邻的数据单元。

[0383] 缓存存储器303存储为二联体的标签,并且这些标签用于确定被请求的数据单元是否在缓存存储器303内。这些二联体也用于检测争用-当在同一个周期内,多个被请求的数据单元驻留在同一个存储体的不同行中。

[0384] 图13是示出包括地址转换、缓存命中/未命中、争用和信息输出的过程的时序图。

[0385] 坐标(X,Y)在周期0中被接受或生成,在周期1中被转换为二联体。存储体访问是在同一周期内进行计算,以用于在周期2中执行访问。如果几个坐标在不同地址处寻址同一个存储体(如本示例中的情况如此),则存在争用,相应的像素提取被延迟到下一个周期,并且在周期1中被断言停止。导致争用的坐标在周期1中退出,并在周期2中访问存储体。坐标和像素之间的延迟是5个周期+停止周期的数量。在极端情况下,访问16个像素会导致15个停止周期。对于扭曲操作,由于所访问的像素彼此接近,16个像素以平均1.0-1.4个周期进行提取,这取决于扭曲的类型。

[0386] 配置映射

[0387]

地址	大小	寄存器
0x0004_4300	2B	gu_Ctrl 控制寄存器（配置寄存器 311）

[3:0]: 模式

0: 使用 DPU 的 16 个坐标

1: 使用来自 DPU 的 8 个坐标并生成(x+1,y)坐标

2: 使用来自 DPU 的 8 个坐标并生成(x,y+1)坐标

3: 使用来自 DPU 的 4 个坐标并生成（x+1，y）（x，y+1）&（x+1，y+1）坐标

4: 使用来自 DPU 的 4 个坐标并生成（x+1，y）...（x+2，y）坐标

5..7: 保留

8: 使用迭代器

9: 使用具有（LSU 的）地址生成器 5 的迭代器

[6:4]: NoValXY 0:16，1:14，2:12，3:10，4: 8

[7]保留

[11:8]收集历史选择

[12]短模式 - 传送 16 位数据。地址（X 坐标）必须对齐到 2 个字节。0: 字节型 - 1: 短整型（16 位）

[14:13]: 保留

[15]: 存储体切换控制:

0: 使用存储体 0-7

1: 使用存储体 8-15

0x0004_4310	4B	gu_Stride 地址 stride= M<<（E+4）
[0388]		[3:0]尾数'M'
		[7:4]指数'E'
0x0004_4314	2B	gu_AddStep 地址步长
0x0004_4316	2B	gu_AddStride 地址每步计数
0x0004_4318	4B	gu_xCount X 上的最大步长计数器
0x0004_4320	32B	gu_AddBase 16 个基地址，每个迭代器 16 位

[0389] 返回参考图12:输入缓冲器301耦合到地址转换器302。地址标签比较器304接收来自缓存存储器303和来自地址转换器302(如果需要地址转换)或来自输入缓冲器301的输入。地址标签比较器304向控制器307以及争用评估单元306和存储器接口308发送指示比较的输出信号(诸如缓存未命中、缓存命中以及如果命中,命中发生何处)。迭代器310耦合到输入缓冲器301和存储器接口308。

[0390] 输入接口(诸如输入缓冲器301)被布置为接收用于检索多个被请求的数据单元的多个请求。

[0391] 缓存存储器303包括存储多个标签(每个标签可以是二联体)和多个缓存的数据单元的条目(诸如十六个条目或行)。

[0392] 每个标签与缓存的数据单元相关联,并且表示存储器模块(诸如存储体)中的与缓存存储器不同并且存储缓存的数据单元的一组存储器单元(诸如行)。

[0393] 地址标签比较器304包括比较器阵列,该比较器阵列被布置成在多个标签和多个被请求的存储器组地址之间同时进行比较以提供比较结果。

[0394] 地址标签比较器包括K×J个节点-覆盖每对标签和被请求的存储体地址。第(k,j)个节点304(k,j)将第k个请求的地址与第j个标签进行比较。

[0395] 如果例如某个被请求的存储体/行地址不匹配任何标签,则地址标签比较器304将发送未命中信号。

[0396] 控制器307可以被布置为(a)基于比较结果将多个被请求数据单元分类为存储在缓存存储器303中和未被缓存的数据单元(未存储在高速缓存存储器303中);并且(b)当至少存在未被缓存的数据单元时,向争用评估单元306发送关于它的信息。

[0397] 争用评估单元306被布置成检查至少一个争用的发生。

[0398] 存储器接口308被布置为以无争用的方式向存储器模块请求任何未被缓存的数据

单元。

[0399] 当一个或更多个未被缓存的数据单元被收集单元检索时，它们被存储在缓存存储器中。在检测到争用的给定周期之后的周期中，地址标签比较器304可以从输入缓冲器或从地址转换器接收来自给定周期的相同请求的数据单元。现在存储在缓存存储器303中的先前未被缓存的数据单元将更改地址标签比较器304所做的比较结果，并且将从存储器模块仅检索在前面的迭代中未被请求的未被缓存的存储器数据单元。

[0400] 争用评估单元可以包括多组节点。图14中提供了示例。

[0401] 节点组的数量是收集单元可以同时访问的存储体的最大数目(例如八个)。

[0402] 每组节点被布置成评估与单个存储体相关的争用。例如，在图14中有八组节点-用于在十六个被请求的地址和八个存储体的行之间进行比较。

[0403] 第一组节点(305(0,0)-306(0,15))的节点彼此串联连接。第一组节点的最左端节点306(0,0)接收输入信号(被使用，存储体0，行)以及信号(有效0，地址0)。

[0404] 地址0是(数据单元的)16个被请求的地址中的第一个地址，并且有效0指示第一个被请求的地址是否有效-第一个请求的地址是指缓存的数据单元(无效)或者是指未被缓存的数据单元(有效)。

[0405] 输入信号(被使用，存储体0，行)表示存储体0是否被使用以及该组节点的第一个节点所请求的行。馈送到最左端节点306(0,0)的输入信号(被使用，存储体0，行)指示存储体0未被使用。

[0406] 例如，如果最左端节点306(0,0)(或任何其他节点)确定应该使用先前未使用的存储体(当前未与任何未被缓存的数据单元相关联)(用于检索与节点关联的有效数据单元地址节点)，则节点更改信号(被使用，存储体0，行)以指示存储体被使用-并且还将“行”更新为由节点请求的行。

[0407] 如果第一组节点(在节点306(0,1)-306(0,15)中的)的任何节点接收到指示存储体0的有效地址，则该节点在所请求的地址的行与通过(被使用，存储体0，行)所指示的行之间进行比较。如果行值不匹配，则节点输出争用信号。

[0408] 同一个过程由任何一组节点同时执行。

[0409] 如果有J个标签，则有J个串联连接的节点组，并且每个组可能包含K个节点。

[0410] 因此，该组的每个节点被布置为(a)接收指示该组的任何先前节点是否请求访问通过二联体(有效，地址)标识的存储体的访问请求指示(例如信号(被使用，存储体，行)和(b)更新访问请求指示，以指示该组是否正在请求访问其相应的存储体。

[0411] 数据处理阵列(DPA)500。

[0412] DPA500包括96个DPU，被布置成六行(行)，而每行包括十六个DPU。

[0413] DPU的每一行可以由单独的DPA微控制器控制。

[0414] 图15和图16示出根据本发明的实施例的DPU 510。

[0415] DPU 510包括：

[0416] 1)算术逻辑单元(ALU)540

[0417] 2)包括十六个寄存器550(0)-550(15)的寄存器文件550。

[0418] 3)两个输出多路复用器MuxD 534、MuxE 535。

[0419] 4)多个输入多路复用器MuxIn0 570、MuxIn1 571、MuxIn2 572、MuxIn3 573、

MuxA561、MuxB 562、MuxCl 563、MuxCh 564、MuxF 526和MuxG 527。

[0420] 5)内部多路复用器MuxH 529和MuxG'528。

[0421] 6)触发器565和566。

[0422] 7)寄存器RegA 531、RegB 532、RegCl 533和RegCh 534。

[0423] 上面提到的多路复用器是数据流组件的非限制性示例。

[0424] 每个输入多路复用器耦合到DPU 510的输入端口,并且可以耦合到其他DPU,耦合到收集单元300,耦合到存储器模块200,耦合到缓冲单元400或耦合到DPU的输出端口。

[0425] 输入多路复用器MuxA 561、MuxB 562、MuxCl 563、MuxCh 564和MuxH 529还包括耦合到总线581的输入端。总线581还耦合到MuxIn0 570、MuxIn1 571、MuxIn2 572、MuxIn3 573的输出端。

[0426] 寄存器RegA 531、RegB 532、RegCl 533和RegCh 534连接在输入多路复用器MuxA 561、MuxB 562、MuxCl 563、MuxCh 564(分别为每个多路复用器一个寄存器)和ALU 540之间,并向ALU馈送数据。

[0427] 使用其中一组多路复用器可接收另一组的输出的两组多路复用器增加提供馈送给ALU 540的数据的源的数量。

[0428] ALU 540的输出端耦合到寄存器文件550的输入端。第一寄存器文件Reg0 550(0)还作为输入端连接到ALU 540。

[0429] 寄存器文件550的输出端耦合到输出多路复用器MuxD 534和MuxE 535。输出多路复用器MuxD 534、MuxE 535的输出端分别耦合到输出端口D(521)和E(522),并且(经由MuxG'528)耦合到输出端口G 523和触发器566。

[0430] 寄存器RegH 539连接在MuxH 529和MuxG 527之间。MuxF 526直接连接到端口F 522和触发器565,从而提供低延时转播信道。MuxG 527耦合到MuxG'528。

[0431] MuxIn0、MuxIn1、MuxIn2、MuxIn3可以实现到其他DPU的短路由:(a)MuxIn0、MuxIn1从8个DPU的D输出端获得输入,(b)MuxIn2、MuxIn3从相同的8个DPU的E输出端获得输入。

[0432] 其他五个输入多路复用器MuxA、MuxB、MuxCl、MuxCh和MuxG可以实现以下路由:

[0433] MuxA可以从缓冲单元获得其输入,并从MuxIn0..MuxIn3获得输入。

[0434] MuxB、MuxCl和MuxCh中的每一个可以从缓冲单元MuxIn0..MuxIn3并从寄存器文件的内部寄存器(例如寄存器文件的R14或R15)获得其输入。

[0435] 大部分ALU操作产生一个短整型(Out0)结果。某些操作会生成一个字结果或两个短整型结果({Out1,Out0})。输出被存储在寄存器文件550中的恒定位置:R(0)<=Out0,并且R(1)<=Out1(用于产生2个短整数的操作)。

[0436] 同一行中的DPU 510和其他DPU由共享行PDA微控制器控制,该共享行PDA微控制器生成用于在存储在DPU中的配置指令(参见配置寄存器511)之间进行选择的选择信息流。

[0437] 配置寄存器(也称为dpu_Ctrl)511可以存储以下内容:

[0438]

地址	大小	寄存器	
0x000	4B	dpu_Ctrl	
0x004	24B	保留	
0x01c	4B	{dpu_Reg [15]， dpu_Reg [14]}	
0x020	4B	dpu_Inst [0] L	（哑元 NOP）
0x024	4B	dpu_Inst [0] H	（哑元 NOP）
0x028	4B	dpu_Inst [l] L	
0x02c	4B	dpu_Inst [l] H	
0x030	4B	dpu_Inst [2] L	
0x034	4B	dpu_Inst [2] H	
0x038	4B	dpu_Inst [3] L	
0x03c	4B	dpu_Inst [3] H	

[0439]

[0440] 四个配置寄存器511(1)-511(3)被称为dpu_Inst[0]-dpu_Inst[1]。

[0441] 如上所述,DPA 500的每个DPU直接耦合到PMA的一些DPU,并直接耦合(经由一个或更多个中间DPU耦合)到数据处理器阵列的一些其他数据处理器。每个DPU具有一个转播信道(端口F和G之间),用于在DPU的转播端口(端口F和端口G)之间转播数据。这样简化连接并减少连接,同时提供足够的连接性和灵活性,以便以高效的方式执行图像处理任务。

[0442] 多个数据处理器中的每个数据处理器的转播信道(尤其是端口F和端口G的输出G之间的路径)基本上呈现零等待时间。这允许使用PDU作为零等待时间转播信道,从而间接地在DPU之间进行耦合,并且还允许通过使用不同DPU之间的转播信道来将数据广播到多个DPU。

[0443] 再参考图16,每个DPU包括一个核心。核心包括ALU 540和诸如寄存器文件550的存储器资源。多个DPU的核心通过可配置网络相互耦合。可配置网络包括数据流组件,例如多路复用器MuxA-MuxCh、MuxD-MuxE、MuxIn0-MuxIn3、MuxF-MuxH和MuxG'。这些数据流组件可以被包含在DPU内(如图16所示),但是可以至少部分地位于DPU之外。

[0444] DPU可以包括直接耦合到第一组邻居的非转播输入端口。例如-非转播输入端口可以包括输入端口A、B、Cl、Ch、In0、In1、In2和In3。它们与第一组邻居的连接性在下面的例子中列出。第一组邻居可以包括例如八个邻居。

[0445] 第一组邻居由位于距DPU小于四个DPU的距离(循环距离)内的DPU形成。距离和方向是循环的。例如,MuxIn0耦合到DPU的D端口,它们是:(a)在同一行中但向左一列(D(0,-1)),(b)在同一行中但向右一列(D(0,+1)),(c)在同一列但在上一行(D(-1,0)),(d)上面的一行且向左一列(D(-1,-1)),(e)上面的一行且向右一列(D(-1,+1)),(f)上面两行且同一

列(D(-2,0)),(g)上面两行且向左一列(D(-2,-1)),(h)上面两行但向右一列(D(-2,+1))。

[0446] 数据处理器的第一非转播输入端口可以直接耦合到第一组邻居的数据处理器的转播端口。例如,参见端口A,其直接耦合到同一DPU的F端口,并且耦合到同一行但是向左一列(F/Fd(0,-1))的DPU的F端口。

[0447] 第一转播端口(例如端口G和F)可以直接耦合到第二组邻居。例如,输入多路复用器F(耦合到端口F)可以耦合到DPU的G端口的G输出端和延迟G输出端(Gd),其是(a)下一行并且在同一列(G/Gd(+1,0)),(b)下两行并在同一列G/Gd(+2,0),(c)下三行且在同一列(G/Gd(+3,0)),(d)同一行但向左一列(G/Gd(0,+1)),(e)相同的行但向左两列G/Gd(0,+2),(f)同一行但向左四列(G/Gd(0,+4)),和(g)同一行但向左八列(G/Gd(0,+8))。

[0448] 在以下配置示例中,提供了PMA和DPU微控制器配置寄存器598(包括寄存器p_FLIP0-p_FLIPS5-每个DPU微控制器一个)的不同配置缓冲器的位置:

地址	大小	寄存器	
0x0004_4000	256B	p_uCPM [0:127]	DPA 微控制器程序存储器

[0449]

0x0004_4100	4B	PmaCSR	PMA 控制寄存器
0x0004_4118	4B	p_FLIPC0	DPA uC0 的首个和最后一个 PC
0x0004_411C	4B	p_FLIPCl	DPA uCl 的首个和最后一个 PC
0x0004_4120	4B	p_FLIPC2	DPA uC2 的首个和最后一个 PC
0x0004_4124	4B	p_FLIPC3	DPA uC3 的首个和最后一个 PC

[0450]

0x0004_4128	4B	p_FLIPC4	DPA uC4 的首个和最后一个 PC
0x0004_412C	4B	p_FLIPC5	DPA uC5 的首个和最后一个 PC

[0451] 配置寄存器511(0)-511(3)可以存储多达四个配置指令。配置指令可以是64位长并且可以通过一个或两个读取操作来读取。

[0452] 配置指令控制多路复用器选择(A、B、C、D、E、F和G),并且寄存器文件550移位:

[0453] 1)按步骤1移位:对于[0,15]中的n:对于(i=15;i>0;i--)Ri<=R(i-1)。

[0454] 2)按步骤2移位:对于[0,2,4...14]中的n:对于(i=7;i>0;i--){R(2i+1),R(2i)}<={R(2i-1),R(2i-2)}

[0455] 这些字段可以在不同的时刻应用:

[0456] 1)输入多路复用器A、B、C、F和输出多路复用器G控制不被延迟。

[0457] 2)ALU控制延迟一个时钟周期。

[0458] 3)输出多路复用器D和E控制延迟两个时钟周期。

[0459] 下表描述了DPU配置指令的不同字段:

	4:0	输入 A 选择
	9:5	输入 B 选择
	14:10	输入 Cl 选择
	19:15	输入 Ch 选择
[0460]	23:20	输出 D 选择
	27:24	输出 E 选择
	31:28	输入/输出 F 选择
	36:32	输出 G 选择
	41:37	RegOpCode:

40:37 移位目的地（视为寄存器文件）:

移位操作中最后一次更改寄存器的位置

41 移位步骤:

0: 移位 1（短整型结果）

1: 移位 2（用于整数型结果）

42 将 ALU 输出交错写入 R0 和 R1

44:43 wrConst: 0: NOP; 1: Regl5 <= ALU 低; 2: Regl5: ALU 高; 3: {Regl4, Regl5} <= ALU

56:45 ALU 控制:（AluCtl_t）

[5:0]: DPU ALU 操作码（DpuAluOpCode_t）

[0461] [7:6]: 后置修饰符（Post Modifier）: 对 FP 操作进行四舍五入，对其他进行后置移位

四舍五入模式: 0: Round，l: Int，2: Floor，3: Ceil

后移位: 左移 0、1、2 或 3 位

[8]: 向量（vectorial）: 0: 规则（regular）– 1: 向量

[9]: mode_a: 0: 无符号 – 1: 有符号

[10]: mode_b: 0: 无符号– 1: 有符号

[11]: acc: 累加器模式: 选择 Cacc 并跳过寄存器 C

57: wrReg1: 从 ALU 写出 Reg1 高电平。

63:58 保留

[0462] MuxIn0-MuxIn3中的每一个都连接到多个总线上-如下所示:MuxIn0:

[0463] ○MuxIn0[0]<=D(0,-1)

[0464] ○MuxIn0[1]<=D(0,+1)

[0465] ○MuxIn0[2]<=D(-1,0)

[0466] ○MuxIn0[3]<=D(-1,-1)

[0467] ○MuxIn0[4]<=D(-1,+1)

[0468] ○MuxIn0[5]<=D(-2,0)

[0469] ○MuxIn0[6]<=D(-2,-1)

[0470] ○MuxIn0[7]<=D(-2,+1)

[0471] MuxIn1：

[0472] ○MuxIn1[0]<＝D(0,-1)

[0473] ○MuxIn1[1]<＝D(0,+1)

[0474] ○MuxIn1[2]<＝D(-1,0)

[0475] ○MuxIn1[3]<＝D(-1,-1)

[0476] ○MuxIn1[4]<＝D(-1,+1)

[0477] ○MuxIn1[5]<＝D(-2,0)

[0478] ○MuxIn1[6]<＝D(-2,-1)

[0479] ○MuxIn1[7]<＝D(-2,+1)

[0480] MuxIn2：

[0481] ○MuxIn2[0]<＝E(0,-1)

[0482] ○MuxIn2[1]<＝E(0,+1)

[0483] ○MuxIn2[2]<＝E(-1,0)

[0484] ○MuxIn2[3]<＝E(-1,-1)

[0485] ○MuxIn2[4]<＝E(-1,+1)

[0486] ○MuxIn2[5]<＝E(-2,0)

[0487] ○MuxIn2[6]<＝E(-2,-1)

[0488] ○MuxIn2[7]<＝E(-2,+1)

[0489] MuxIn3：

[0490] ○MuxIn3[0]<＝E(0,-1)

[0491] ○MuxIn3[1]<＝E(0,+1)

[0492] ○MuxIn3[2]<＝E(-1,0)

[0493] ○MuxIn3[3]<＝E(-1,-1)

[0494] ○MuxIn3[4]<＝E(-1,+1)

[0495] ○MuxIn3[5]<＝E(-2,0)

[0496] ○MuxIn3[6]<＝E(-2,-1)

[0497] ○MuxIn3[7]<＝E(-2,+1)

[0498] 输入多路复用器MuxA、MuxB、MuxC、MuxF、输出多路复用器MuxD、MuxE、输出F、延时输出Fd、输出G和延时输出Gd提供本段列出的连接性。符号X(N,M)表示DPU的输出X(row+N％6,col+M％16)。每行的DPU以循环方式彼此连接，并且每列的DPU以循环方式彼此连接。应该注意的是，下面列出的各种多路复用器具有多个(例如十六个)输入端，以及下面列表提供了与这些输入中的每一个的连接。例如A[0]-A[15]是MuxA的十六个输入。在下文中，6％表示模6操作，以及16％表示模16操作。R14和51R是寄存器文件的最后两个寄存器。

[0499] •输入A多路复用器：

[0500] ○A[0]<＝0

[0501] ○A[1]<＝nu_Px1A

[0502] ○A[2]<＝D(0,0)

[0503] ○A[3]<＝E(0,0)

[0504] ○A[4]<＝MuxIn0

[0505] ○A[5]<＝MuxIn1

[0506] ○A[6]<＝MuxIn2

[0507] ○A[7]<＝MuxIn3

[0508] ○A[8]<＝PxlAb

[0509] ○A[9]<＝PxlBb

[0510] ○A[10]<＝0

[0511] ○A[11]<＝F(0,0)

[0512] ○A[12]<＝F/Fd(0,-1)

[0513] ○A[13]<＝R15

[0514] ○A[14]<＝R14

[0515] ○A[15]<＝PxlBi

[0516] •输入B多路复用器：

[0517] ○B[0]<＝0

[0518] ○B[1]<＝nu_PxlB

[0519] ○B[2]<＝D(0,0)

[0520] ○B[3]<＝E(0,0)

[0521] ○B[4]<＝MuxIn0

[0522] ○B[5]<＝MuxIn1

[0523] ○B[6]<＝MuxIn2

[0524] ○B[7]<＝MuxIn3

[0525] ○B[8]<＝PxlAb

[0526] ○B[9]<＝PxlBb

[0527] ○B[10]<＝0

[0528] ○B[11]<＝F(0,0)

[0529] ○B[12]<＝F/Fd(0,-1)

[0530] ○B[13]<＝R15

[0531] ○B[14]<＝R14

[0532] ○B[15]<＝PxlBi

[0533] •输入Cl多路复用器：

[0534] ○Cl[0]<＝0

[0535] ○Cl[1]<＝nu_PxlA

[0536] ○Cl[2]<＝D(0,0)

[0537] ○Cl[3]<＝E(0,0)

[0538] ○Cl[4]<＝MuxIn0

[0539] ○Cl[5]<＝MuxIn1

[0540] ○Cl[6]<＝MuxIn2

[0541] ○Cl[7]<＝MuxIn3

[0542] ○Cl[8]<＝PxlAb

[0543] ○Cl[9]<＝PxlBb

[0544] ○Cl[10]<=0

[0545] ○Cl[11]<=F(0,0)

[0546] ○Cl[12]<=F/Fd(0,-1)

[0547] ○Cl[13]<=R15

[0548] ○Cl[14]<=R14

[0549] ○Cl[15]<=PxlBi

[0550] •输入Ch多路复用器:

[0551] ○Ch[0]<=0

[0552] ○Ch[1]<=nu_PxlB

[0553] ○Ch[2]<=D(0,0)

[0554] ○Ch[3]<=E(0,0)

[0555] ○Ch[4]<=MuxIn0

[0556] ○Ch[5]<=MuxIn1

[0557] ○Ch[6]<=MuxIn2

[0558] ○Ch[7]<=MuxIn3

[0559] ○Ch[8]<=PxlAb

[0560] ○Ch[9]<=PxlBb

[0561] ○Ch[10]<=0

[0562] ○Ch[11]<=F(0,0)

[0563] ○Ch[12]<=F/Fd(0,-1)

[0564] ○Ch[13]<=R15

[0565] ○Ch[14]<=R14

[0566] ○Ch[15]<=PxlBi

[0567] •输出D多路复用器:{Regs[0..15]}

[0568] •输出E多路复用器:{Regs[0..15]}

[0569] •输入F多路复用器:

[0570] ○F[0]:使用在dpu_CSR中定义的F多路复用器

[0571] ○F[1]<=G/Gd(+1,0)

[0572] ○F[2]<=G/Gd(+2,0)

[0573] ○F[3]<=G/Gd(+3,0)

[0574] ○F[4]<=G/Gd(0,+1)

[0575] ○F[5]<=G/Gd(0,+2)

[0576] ○F[6]<=G/Gd(0,+4)

[0577] ○F[7]<=G/Gd(0,+8)

[0578] •输出F:F多路复用器输出

[0579] •输出Fd:F锁存@clk

[0580] •输出G多路复用器:

[0581] ○G[0]<=使用dpu_CSR中定义的G多路复用器

[0582] ○G[1]<=D

[0583] ○G[2]<=E

[0584] ○G[3]<=F

[0585] ○G[4]<=D(+1,0)

[0586] ○G[5]<=E(+1,0)

[0587] ○G[6]<=NU_A

[0588] ○G[7]<=NU_B

[0589] ○G[8]<=MuxG,MuxG<=MuxInD0

[0590] ○G[9]<=MuxG,MuxG<=MuxInD1

[0591] ○G[10]<=MuxG,MuxG<=MuxInE0

[0592] ○G[11]<=MuxG,MuxG<=MuxInE1

[0593] ○G[12]<=PxlAb

[0594] ○G[13]<=PxlBb

[0595] ○G[14]<=PxlAbi

[0596] ○G[15]<=PxlBbi

[0597] •输出Gd:G锁存@clk

[0598] 输入(A,B,Cl,Ch,G)寄存器配置规范(配置位被存储在DPU的配置寄存器中,且用于控制DPU的各个组件)。

[0599] 以下示例列出了DPU配置指令中所包含的各个位的值。符号X(N,M)表示DPU的输出X(row+N%6,col+M%16)。

[0600] •输入A多路复用器:

[0601] ○A[0]<=0

[0602] ○A[1]<=nu_PxlA

[0603] ○A[2]<=D(0,0)

[0604] ○A[3]<=E(0,0)

[0605] ○A[4]<=MuxIn0

[0606] ○A[5]<=MuxIn1

[0607] ○A[6]<=MuxIn2

[0608] ○A[7]<=MuxIn3

[0609] ○A[8]<=nu_PxlAb

[0610] ○A[9]<=nu_PxlBb

[0611] ○A[10]<=nu_PxlAbi

[0612] ○A[11]<=F(0,0)

[0613] ○A[12]<=F/Fd(0,-1)

[0614] ○A[13]<=R15

[0615] ○A[14]<=R14

[0616] ○A[15]<=nu_PxlBbi

[0617] ○A[16]<=D(0,-1)

[0618] ○A[17]<=D(0,+1)

[0619] ○A[18]<=D(-1,0)

[0620] ○A[19]<＝D(-1,-1)

[0621] ○A[20]<＝D(-1,+1)

[0622] ○A[21]<＝D(-2,0)

[0623] ○A[22]<＝D(-2,-1)

[0624] ○A[23]<＝D(-2,+1)

[0625] ○A[24]<＝E(0,-1)

[0626] ○A[25]<＝E(0,+1)

[0627] ○A[26]<＝E(-1,0)

[0628] ○A[27]<＝E(-1,-1)

[0629] ○A[28]<＝E(-1,+1)

[0630] ○A[29]<＝E(-2,0)

[0631] ○A[30]<＝E(-2,-1)

[0632] ○A[31]<＝E(-2,+1)

[0633] •输入B多路复用器：

[0634] ○B[0]<＝0

[0635] ○B[1]<＝nu_PxlB

[0636] ○B[2]<＝D(0,0)

[0637] ○B[3]<＝E(0,0)

[0638] ○B[4]<＝MuxIn0

[0639] ○B[5]<＝MuxIn1

[0640] ○B[6]<＝MuxIn2

[0641] ○B[7]<＝MuxIn3

[0642] ○B[8]<＝nu_PxlAb

[0643] ○B[9]<＝nu_PxlBb

[0644] ○B[10]<＝nu_PxlAbi

[0645] ○B[11]<＝F(0,0)

[0646] ○B[12]<＝F/Fd(0,-1)

[0647] ○B[13]<＝R15

[0648] ○B[14]<＝R14

[0649] ○B[15]<＝nu_PxlBbi

[0650] ○B[16]<＝D(0,-1)

[0651] ○B[17]<＝D(0,+1)

[0652] ○B[18]<＝D(-1,0)

[0653] ○B[19]<＝D(-1,-1)

[0654] ○B[21]<＝D(-2,0)

[0655] ○B[22]<＝D(-2,-1)

[0656] ○B[23]<＝D(-2,+1)

[0657] ○B[24]<＝E(0,-1)

[0658] ○B[25]<＝E(0,+1)

[0659] ○B[26]<=E(-1,0)

[0660] ○B[27]<=E(-1,-1)

[0661] ○B[28]<=E(-1,+1)

[0662] ○B[29]<=E(-2,0)

[0663] ○B[30]<=E(-2,-1)

[0664] ○B[31]<=E(-2,+1)

[0665] •输入C1多路复用器：

[0666] ○C1[0]<=0

[0667] ○C1[1]<=nu_Pxl A

[0668] ○C1[2]<=D(0,0)

[0669] ○C1[3]<=E(0,0)

[0670] ○C1[4]<=MuxIn0

[0671] ○C1[5]<=MuxIn1

[0672] ○C1[6]<=MuxIn2

[0673] ○C1[7]<=MuxIn3

[0674] ○C1[8]<=nu_PxlAb

[0675] ○C1[9]<=nu_PxlBb

[0676] ○C1[10]<=nu_PxlAbi

[0677] ○C1[11]<=F(0,0)

[0678] ○C1[12]<=F/Fd(0,-1)

[0679] ○C1[13]<=R15

[0680] ○C1[14]<=R14

[0681] ○C1[15]<=nu_PxlBbi

[0682] ○C1[16]<=D(0,-1)

[0683] ○C1[17]<=D(0,+1)

[0684] ○C1[18]<=D(-1,0)

[0685] ○C1[19]<=D(-1,-1)

[0686] ○C1[21]<=D(-2,0)

[0687] ○C1[22]<=D(-2,-1)

[0688] ○C1[23]<=D(-2,+1)

[0689] ○C1[24]<=E(0,-1)

[0690] ○C1[25]<=E(0,+1)

[0691] ○C1[26]<=E(-1,0)

[0692] ○C1[27]<=E(-1,-1)

[0693] ○C1[28]<=E(-1,+1)

[0694] ○C1[29]<=E(-2,0)

[0695] ○C1[30]<=E(-2,-1)

[0696] ○C1[31]<=E(-2,+1)

[0697] •输入Ch多路复用器：

[0698] ○Ch[0]<＝0

[0699] ○Ch[1]<＝nu_PxlB

[0700] ○Ch[2]<＝D(0,0)

[0701] ○Ch[3]<＝E(0,0)

[0702] ○Ch[4]<＝MuxIn0

[0703] ○Ch[5]<＝MuxIn1

[0704] ○Ch[6]<＝MuxIn2

[0705] ○Ch[7]<＝MuxIn3

[0706] ○Ch[8]<＝nu_PxlAb

[0707] ○Ch[9]<＝nu_PxlBb

[0708] ○Ch[10]<＝nu_PxlAbi

[0709] ○Ch[11]<＝F(0,0)

[0710] ○Ch[12]<＝F/Fd(0,-1)

[0711] ○Ch[13]<＝R15

[0712] ○Ch[14]<＝R14

[0713] ○Ch[15]<＝nu_PxlBbi

[0714] ○Ch[16]<＝D(0,-1)

[0715] ○Ch[17]<＝D(0,+1)

[0716] ○Ch[18]<＝D(-1,0)

[0717] ○Ch[19]<＝D(-1,-1)

[0718] ○Ch[21]<＝D(-2,0)

[0719] ○Ch[22]<＝D(-2,-1)

[0720] ○Ch[23]<＝D(-2,+1)

[0721] ○Ch[24]<＝E(0,-1)

[0722] ○Ch[25]<＝E(0,+1)

[0723] ○Ch[26]<＝E(-1,0)

[0724] ○Ch[27]<＝E(-1,-1)

[0725] ○Ch[28]<＝E(-1,+1)

[0726] ○Ch[29]<＝E(-2,0)

[0727] ○Ch[30]<＝E(-2,-1)

[0728] ○Ch[31]<＝E(-2,+1)

[0729] •输出G多路复用器：

[0730] ○G[0]<＝使用dpu_CSR中定义的G多路复用器

[0731] ○G[1]<＝D

[0732] ○G[2]<＝E

[0733] ○G[3]<＝F

[0734] ○G[4]<＝D(+1,0)

[0735] ○G[5]<＝E(+1,0)

[0736] ○G[6]<＝NU_A

[0737] ○G[7]<＝NU_B

[0738] ○G[8]<＝MuxG,MuxG<＝MuxInD0

[0739] ○G[9]<＝MuxG,MuxG<＝MuxInD1

[0740] ○G[10]<＝MuxG,MuxG<＝MuxInE0

[0741] ○G[11]<＝MuxG,MuxG<＝MuxInE1

[0742] ○G[12]<＝PxlAb

[0743] ○G[13]<＝PxlBb

[0744] ○G[14]<＝PxlAbi

[0745] ○G[15]<＝PxlBb

[0746] ○G[16]<＝D(0,-1)

[0747] ○G[17]<＝D(0,+1)

[0748] ○G[18]<＝D(-1,0)

[0749] ○G[19]<＝D(-1,-1)

[0750] ○G[21]<＝D(-2,0)

[0751] ○G[22]<＝D(-2,-1)

[0752] ○G[23]<＝D(-2,+1)

[0753] ○G[24]<＝E(0,-1)

[0754] ○G[25]<＝E(0,+1)

[0755] ○G[26]<＝E(-1,0)

[0756] ○G[27]<＝E(-1,-1)

[0757] ○G[28]<＝E(-1,+1)

[0758] ○G[29]<＝E(-2,0)

[0759] ○G[30]<＝E(-2,-1)

[0760] ○G[31]<＝E(-2,+1)

[0761] 输入(A,B,Cl,Ch,G)具有到(MuxIn0,...,MuxIn3)多路复用和两个附加输入。为了保存配置位,至(A,B,Cl,Ch,G)的D(N,M)和E(N,M)配置位的(MuxIn0,...,MuxIn3)的动态资源分配可以被使用。可以如下进行分配:如果输入(A,B,Cl,Ch,G)配置中的1个或2个具有D(N,M)形式,则第一个输入分配MuxIn0,且第二个(如果存在)分配MuxIn1。为了方便起见,输入由inputD1和inputD2表示,并且配置分别由D1(N,M)和D2(N,M)表示。MuxIn0和MuxIn1多路复用器的控制可分别根据D1(N,M)和D2(N,M)。另外,inputD1和inputD2多路复用控制MuxIn0、MuxIn1。如果输入(A,B,Cl,Ch,G)配置中的1个或2个具有E(N,M)形式,则对MuxIn2和MuxIn3应用相同的动态分配。

[0762] 例如:

[0763] 假设有以下配置:

[0764] ••A＝nu_PxlA

[0765] ••B＝D(0,-1)

[0766] ••Cl＝E(0,-1)

[0767] ••Ch＝D(-1,0)

[0768] ••G＝MUXIn0

[0769] 然后,多路复用器的控制将如下:

[0770] • • MuxIn0＝D(0,-1)

[0771] • • MuxIn1＝D(-1,0)

[0772] • • MuxIn2＝E(0,-1)

[0773] • • muxA＝nu_Px1B

[0774] • • muxB＝MuxIn0

[0775] • • muxCl＝MuxIn2

[0776] • • muxCh＝MuxIn1

[0777] • • muxG＝MuxIn0

[0778] ALU操作码

[0779] 整数运算

0. nop

1. addc: {Out1, Out0} <= A + B + C

addc (v): Out0 <= A1 + B1 + Cl; Out1 = Ah + Bh + Ch

[0780] 2. adds: Out0 <= (A + B) >> C; Out1 <= (A + B) [C-1:0]

3. addl: {Out1, Out0} <= {B, A} + C

addl (v): Out0 <= A + Cl; Out1 <= B + Ch

4. addrv: {Out1, Out0} <= A1 + B1 + Ah + Bh + C

5. add4: {Out1, Out0} <= A + B + Cl+ Ch

6. subb: {Out1, Out0} <= A- B +C

subb (v): Out0 <= Al - Bl + C1; Out1 = Ah - Bh + Ch

7. subc: {Out1, Out0} <= A + B - C

subc (v): Out0 <= A1 + B1 – C1; Out1 = Ah + Bh - Ch

8. subl: {Out1, Out0} <= {B, A} - C

subl (v): Out0 <= A – C1; Out1 <= B - Ch

9. subrv: {Out1, Out0} <= A1 + B1 + Ah + Bh - C

10. mac: {Out1, Out0} <= A * B + C

mac (v): Out0 <= A1 * B1 + C1; Out1 = Ah * Bh + Ch

11. macs: {Out1, Out0} <= A * B - C

macs (v): Out0 <= A1 * B1 – C1; Out1 = Ah * Bh - Ch

[0781]

12. macrv: {Out1, Out0} <= (Al * B1) + (Ah * Bh) + C

13. shift: {Out1, Out0} <= (A<<B) >>C

shift (v): Out0 <= (A1<<B1)>>C1; Out1 = (Ah<<Bh)>>Ch

14. shiftrl: {Out0, Out1} <= {B, A}>>C

shiftrl (v): Out0 <= A>>C1; Out1 <= B>>Ch

15. shiftll: {Out0, Out1} <= {B, A}<< C

shiftll (v): Out0 <= A<<C1; Out1 <= B<<Ch

16. absd: {Out1, Out0} <=|A-B| + C

absd (v): Out0 <=|A1- B1| + C1; Out1 = |Ah-Bh| + Ch

17. absdrv: {Out1, Out0} <= |A1 – B1| + |Ah - Bh| + C

18. absddrv: {Out1, Out0} <= |A1 – B1| - |Ah - Bh| + C

19. min: {Out1, Out0} <= min ({B (val), A (Idx) }, C ({val,

Idx}))--> 输出: Idx min

20. equalone: {Out1, Out0} <= min ({B (val), A (Idx)}, C ({val, Idx})) --> 输出: 第一操作数

21. equaltwo: {Out1, Out0} <= min ({B (val), A (Idx)}, C ({val, Idx})) --> 输出: 第二操作数

22. lessc: {Out1, Out0} <= (A <B) + C

lessc (v): Out0 <= (Al < Bl) + Cl; Out1 = (Ah < Bh) + Ch

23. lesseqc: {Out1, OUt0} <= (A <= B) + C

lesseq (v): Out0 <= (A1<= B1) + C1; Out1 = (Ah <= Bh) + Ch

[0782]

24. equalc: {Out1, Out0} <= (A == B) + C

equalc (v): Out0 <= (A1 == B1) + C1; Out1 = (Ah == Bh) + Ch

25. nequalc: {Out1, Out0} <= (A! = B) + C

nequalc (v): Out0 <= (A1! = B1) + C1; Out1 = (Ah! = Bh) + Ch

26. lesscrv: {Out1, Out0} <= (A1 <B1) + (Ah <Bh) + C

27. lesseqcrv: {Out1, OUt1} <= (A1 <= B1) + (Ah <= Bh) + C

28. equalcrv: {Out1, Out0} <= (A1 == B1) + (Ah == Bh) + C

29. nequalcrv: {Out1, Out0} <= (A1! = B1) + (Ah! = Bh) + C

[0783] 浮点运算

[0784] 转换中的C指数偏置参数可以看作是一个7位有符号整数(C的其他25位可以忽略，且符号位扩展)

30. short2fp: {Out1, Out0} <= A * 2 ** C

31. word2fp: {Out1, Out0} <= {B, A} * 2 ** C

[0785]

32 lessf {Out1, Out0} <= {B, A} <C

33 lesseqf {Out1, Out0} <= {B, A} <= C

34. fp2word: {Out1，Out0} <= {B，A} * 2 ** C

35. mulf: {Out1，Out0} <= {B，A} * C

[0786] 36. addf: {Out1，Out0} <= {B，A} + C

37. subf: {Out1，Out0} <= {B，A} - C

38. int_div: {Out1，Out0} <= 1/A

[0787] 加法运算

39. or {Out0，Out1} <={B，A} | Cn

40. xor {Out0，Out1} <={B，A} ^ AC

41. and {Out0，Out1} <={B，A} &C

42. equal32 {Out1，Out0} <={B，A} == C

43. nequal32 {Out1，Out0} <={B，A}! = C

44. less32 {Out1，Out0} <={B，A} <C

45. lesseq32 {Out1，Out0} <= {B，A} <= C

46. abs32 {Out1，Out0} <= | {B，A} - C |

47. max32 {Out1，Out0} <=max ({B，A}，C) --> 输出: 32max

[0788]

48. minf {Out1，Out0} <=min ({B，A}，C) --> 输出: fp min

49. maxf {Out1，Out0} <=max ({B，A}，C) --> 输出: fp max

50. mac4 {Out1，Out0} <= (A1 * B1) + (Ah * Bh) + (C0 * C2) + (C1* C3) + acc

mac4 (v) Out0 <= (Al * B1) + (Ah * Bh) + acc0; Out1 = (C0 * C2) + (C1 * C3) + acc1

51. shift_sat {Out1，Out0} <= Sat ({B，A}，C): rnd = byteMode: 0 shortMode = 1 | notRELU: 2

52. add_sat {Out1，Out0} <= Sat ({A + B}，C)

add_sat (v) {Out1，Out0} <= Sat ({Al + Bl}，C)

[0789] DPU微控制器指令编码

[0790] 执行指令：

[0791] [8:0]rc:重复计数器：

[0792] [0..495]:立即值

[0793] [496..511]:计数器[rc-496]

[0794] [10:9]sel:DPU指令选择

[0795] [11]收集:激活收集单元(只有微控制器0)

[0796] [14:12]保留

[0797] [15]类型0

[0798] Do循环:

[0799] [8:0]RLC:重复循环计数器:

[0800] [0..495]:立即值

[0801] [496..511]:计数器[rc-496]

[0802] [13:9]长度:循环长度:[1..32]。

[0803] [14]模式:0:计数循环,1:计数周期

[0804] [15]类型1

[0805] DPU可以一次配置一个(每个DPU具有唯一的单播地址)或者可以以广播模式配置-存在可以反映共享行和/或列的DPU的行和/或列的地址且这允许广播配置信息。

[0806] 数据处理阵列:线性映射(编程单个DPU配置寄存器)

[0807]

地址	大小	寄存器
0x0004_6000	6KB	Ad [31:13] = 0x00001 Add[5:0]: DPU 内部映射 Add [9:6] = Col Add [12:10] = Row

0x0004_6000	1KB	DPU Row 0
0x0004_6000	64B	DPU（0，0）
0x0004_6040	64B	DPU（0，1）
...		
0x0004_63C0	64B	DPU（0，15）
0x0004_6400	1KB	DPU Row1
0x0004_6800	1KB	DPU Row 2
0x0004_6C00	1KB	DPU Row 3
0x0004_7000	1KB	DPU Row 4
0x0004_7400	1KB	DPU Row 5
...		
0x0004_77C0	64B	DPU（5，15）

[0808]

[0809] 数据处理阵列:广播映射(同时编程多个DPU的配置寄存器)

地址	大小	寄存器
0b0001_rrrr_rrcc_cccc	64KB	Add [31:28] = 0x1 => DPA 广播映射
cccc_cccc_cc00_0000		

[0810]

Add [27:22]: 行启用（每行 1 位）

Add [21:6]: 列启用（每列 1 位）

Add [5:0]: DPU 配置地址

[0811] 应该注意的是,任何图像处理算法都可以由图像处理器以迭代方式执行。有关一些像素的结果由DPA 500处理。有些结果可能被存储在DPA中一段时间,然后被发送到存储器模块。通常基于PMA的存储器资源的大小以及由DPA在某个任务期间处理的源像素或目标像素的数量设置一定的时间段。一旦再次需要这些结果,它们可能会从存储器模块中被提取。例如,当DPA 500执行关于源图像的某些源像素的计算时,可以将这些结果存储一段时间(例如,当执行与相邻源像素有关的计算时),然后将其发送到存储器。当进一步需要结果时,可以从存储器模块提取它们。

[0812] 扭曲计算

[0813] 由于各种原因,扭曲计算可能被应用。例如,补偿图像采集的不平衡。

[0814] 扭曲计算可以由DPA 500执行。

[0815] 根据本发明的实施例,针对一组目标像素中的每个目标像素(目标图像的像素)应用扭曲计算。目标像素组可以包括整个目标图像或目标图像的一部分。通常,目标图像被虚拟分割成多个窗口,且每个窗口是一组目标像素。

[0816] 扭曲计算可以接收或者可以计算相应的一组源像素。相应组的源像素中的源像素在扭曲计算期间被处理。通常将相应组的源像素的选择馈送到PMA,并且可以取决于例如期望的扭曲函数。

[0817] 通过对与目标像素相关联的相邻源像素应用权重(Wx,Wy)来计算目标像素的扭曲值。至少一个相邻源像素的权重和坐标(x,y)以扭曲参数(X',Y')进行定义。

[0818] 图17示出根据本发明的实施例的方法1700。

[0819] 方法1700可以从步骤1710开始,从一组目标像素中选择目标像素。所选择的目标像素将被称为"目标像素"。

[0820] 步骤1710之后可以是步骤1720,针对一组目标像素中的每个目标像素执行扭曲计算过程,其包括:

[0821] 1)计算(1721)或接收关于所选择的目标像素的扭曲参数。扭曲参数可以包括在扭曲计算期间应该被处理的给定源像素的第一权重和第二权重(Wx,Wy)和坐标(x,y)。第一权重和第二权重由处理单元阵列(DPA)中的第一组处理单元(DPU)接收。

[0822] 2)向诸如收集单元的存储器单元请求(1722)相邻源像素(其包括给定的源像素)。收集单元可以在各种操作模式下接收4个坐标并将它们转换为16个源像素-四个相邻源像素组。

[0823] 3)通过第二组处理单元接收(1723)与目标像素相关联的相邻源像素。

[0824] 4)通过第二组处理单元计算(1724)响应于相邻源像素的值和一对权重的扭曲结果;向存储器模块提供扭曲结果。

[0825] 步骤1721、1722、1723和1724可以以流水线方式执行。

[0826] 参考图18,第一组处理单元被表示为505并且可以包括DPA 500的上部第一行中的最左边的四个DPU。第二组处理单元被表示为501并且可以包括DPA 500的最右边的两列。

[0827] 在步骤1720之后是步骤1730,检查扭曲是否是针对该组的所有目标像素计算的。如果没有,则结束扭曲计算。

[0828] 步骤1726可以包括在第二组中的处理单元之间转播一些相邻源像素的值。

[0829] 图18和图19示出DPU(0,4)的输出信号(组504的X')被发送到DPU(0,15),然后被转播到DPU(1,15)。应该注意,在图18和图19中,PMA并行计算四个像素的扭曲函数:

[0830] 1)DPU(0,3)、DPU(1,3)和组501中的DPU涉及计算第一像素的扭曲。

[0831] 2)DPU(0,2)、DPU(1,2)和组502的DPU涉及计算第二像素的扭曲。

[0832] 3)DPU(0,1)、DPU(1,1)和组503的DPU涉及计算第三像素的扭曲。

[0833] 4)DPU(0,0)、DPU(1,0)和组504的DPU涉及计算第三像素的扭曲。

[0834] 步骤1726可以包括转播由第二组计算的中间结果和第二组的处理单元之间的一些相邻源像素的值。

[0835] 图20示出了从组505和506的DPU(510(0,0)-510(0,3)和510(1,0)-510(1,3))发送扭曲参数(组501-504的X'和组501-504的Y'))至组501、502、503和504。

[0836] 图21和图22示出根据本发明实施例的由组501的DPU 510(0,15)-510(3,15)和DPU

510(0,14)-510(5,14)执行的扭曲计算。

[0837] 图21的扭曲计算包括以下步骤(其中一些相互并行地执行)。步骤1751-1762也在图22中示出。

[0838] 由第二组的第一处理单元(DPU 510(5,14))计算(1751)在第一对相邻源像素之间的第一差值(P0-P2)和在第二对相邻的源像素之间的第二差值(P1-P3)。

[0839] 将第一差值提供(1752)到第二组的第二处理单元(DPU 510(1,14)),并将第二差值提供给第二组的第三处理单元。

[0840] 由第二组的第四处理单元(DPU 510(1,15))响应于第一权重计算(1753)第一修改权重Wy'。

[0841] 将第一修改权重从第四处理单元提供(1754)给第二组的第二处理单元(DPU 510(1,14))。

[0842] 由第二组的第二处理单元基于第一差值(P0-P2)、第一相邻源像素(P0)和第一修改权重(Wy')计算(1755)第一中间结果(Var0)。Var0＝(P0-P0)*Wy'-P0。

[0843] 从第二组的第三处理单元向第二组的第六处理单元(DPU 510(0,15))提供(1756)第二差值(P1-P3)。

[0844] 从第二组的第五处理单元(DPU 510(0,14))向第二组的第六处理单元(DPU 510(2,14))提供(1757)第二相邻源像素(P1)。

[0845] 通过第二组的第六处理单元基于第二差值、第二相邻源像素和第一修改权重计算(1758)第二中间结果Var1。Var1＝(P1-P3)*Wy'-P1。

[0846] 将来自第二组的第六处理单元的第二中间结果Var1提供(1759)到第二组的第七处理单元(DPU 510(2,15)),并且将第一中间结果Var0从第二组的第二处理单元提供到第二组的第七处理单元。

[0847] 通过第二组的第七处理单元计算(1760)相对于第一中间结果和第二中间结果的第三中间结果Var2。Var2＝Var0-Var1。

[0848] 将第三中间结果从第二组的第七处理单元的提供(1761)到第二组的第八处理单元(DPU 510(3,15))。将第二中间结果从第二组的第六处理单元提供给第二组的第九处理单元(DPU 510(3,14))。

[0849] 将第二中间结果从第二组的第九处理单元提供(1762)给第二组的第八处理单元。将第二修改权重(Wx')从第二组的第三处理单元提供给第二组的第八处理单元。

[0850] 通过第二组的第八处理单元基于第二中间结果和第三中间结果以及第二修改权重来计算(1763)扭曲结果。Warp_result＝Var2*Wx'+Var1。

[0851] 如图20所示,DPU 510(5,14)可以从收集单元接收像素P0、P1、P2和P3。当DPA 500一次处理四个像素时,组501、502、503和504从收集单元(并行)接收16个像素。

[0852] 应该注意的是,DPA 500还接收(例如,来自收集单元的)与每个像素相关的扭曲参数X',Y'。

[0853] 根据本发明的一个实施例,每个像素的扭曲参数可以通过DPA的DPU计算-例如,当扭曲参数可以通过诸如多项式的数学公式表示时。

[0854] 图23示出了计算X'和Y'的一组DPU 507,并且这些计算出的X'和Y'可以被馈送到组505和506。

[0855] 应该注意,图18至图22仅示出了非限制性分组方案。扭曲计算可以由其他形状和尺寸的DPU组执行。

[0856] 视差

[0857] 视差计算旨在为源像素找到最佳匹配目标像素。可以针对源图像中的所有源像素以及针对目标图像的所有目标像素来执行搜索-但是这不一定是这样,并且视差可以仅应用于源图像的一些源像素上和/或目标图像的一些目标像素上。

[0858] 视差计算不仅仅比较单个源像素与单个目标像素之间的差值,还将源像素的子组与目标像素的子组进行比较。比较可以包括计算诸如在源像素和对应目标像素之间的绝对差和(SAD)之类的函数。

[0859] 源像素可以位于源像素子组的中心,并且目标像素可以位于目标像素的子组的中心。源像素和/或目标像素的其他位置可以被使用。

[0860] 源像素的子组和目标像素的子组可以是矩形形状(或可以具有任何其它形状),并且可以包括N行和N列,而N可以是可以超过三的奇正整数。

[0861] 大部分的视差计算可能受益于以前的计算机视差计算。这些示出在图24和25中被提供。

[0862] 图24示出了5×5源像素S(1,1)-S(5,5)的第一子组1001、5×5目标像素T(1,1)-T(5,5)的第一子组1002、5×5源像素S(1,2)-S(5,6)的第二子组1003和5×5目标像素T(1,2)-T(5,6)的第二子组1004。

[0863] 源像素S(3,3)和S(3,4)位于源像素的第一子组1001和第二子组1003的中心。目标像素T(3,3)和T(3,4)位于目标像素的第一子组1002和第二子组1004的中心。

[0864] 与S(3,3)和T(3,3)相关的SAD等于:

[0865] SAD(S(3,3),T(3,3))=SUM(|S(i,j)-T(i,j)|)-针对在1和5之间的索引i和j。

[0866] 与S(3,4)和T(3,4)相关的SAD等于:

[0867] SAD(S(3,4),T(3,4))=SUM(|S(i,j)-T(i,j)|)-针对2和6之间的索引i和针对在1和5之间的索引j。

[0868] 假设SAD是从左到右计算的。在这个假设下-SAD(S(3,4),T(3,4))的计算可能受益于SAD(S(3,3),T(3,3))的计算。

[0869] 特别地:SAD(S(3,4),T(3,4))=SAD(S(3,3),T(3,3))-SAD(源像素和目标像素的第一子组的最右列)+SAD(源像素和目标像素的第二子组的最左列)。

[0870] 由于源图像和目标图像是二维的,并假设源像素从左到右(每片)和从上到下扫描-那么SAD的计算就更有效率。

[0871] 图25示出了具有中心像素SB的源像素的子组SG(B)。图26示出了具有中心像素TB的目标像素(未示出)的对应子组TG(B)。

[0872] 针对在SB的行的上面的行的源像素以及针对位于SB左侧且在同一行上的像素计算SUD。

[0873] 像素SA是子组SG(A)的中心并且是像素SB的左边邻居。目标像素TA是像素SB的左边邻居,并且是子组TG(A)的中心。

[0874] 像素SC是子组SG(C)的中心并且是像素SB的上方邻居。目标像素TC是像素SB的上方邻居并且是子组TG(C)的中心。

[0875] SG(A)的最左边的列表示为1110。SG(C)的最右边的列表示为1114。SG(B)的当前最右边的列表示为1115。SG(B)的最右边的最低像素(也被称为新源像素NSP)被表示为1116。在SG(B)的当前最右列顶部的旧像素(属于SG(C))(也称为旧源像素NSP)被表示为1112。

[0876] TG(A)的最左边的列表示为1110'。TG(C)的最右边的列表示为1114'。TG(B)的当前最右边的列表示为1115'。TG(B)的最右边的最低像素(也称为新目标像素NTP)被表示为1116'。在TG(B)的当前最右列顶部的旧像素(属于TG(C))(也称为旧目标像素NTP)被表示为1112'。

[0877] 计算(SB,TB)的SAD可以等于：

[0878] SAD(SA,TA)。

[0879] -SAD(SG(A)最左边的列,SG(B)最左边的列)。

[0880] +SAD(SG(C)的最右边的列,TG(C)的最右边的列)。

[0881] +SG(B)和TG(B)的最右边的最低的源像素和目标像素的绝对差。

[0882] -SG(C)和TG(C)最右边的列的最上面的源像素和目标像素的绝对差。

[0883] 图27示出根据本发明的实施例的方法2600。

[0884] 方法2600可以开始于步骤2610,选择源像素并选择目标像素的子组。目标像素的子组可以是目标图像的一部分的整个目标图像。

[0885] 步骤2610之后可以是步骤2620,通过数据处理器阵列的第一组数据处理器计算一组绝对差和(SAD)。

[0886] 该组SAD与包括在步骤2610中选择的目标像素的目标像素的子组和源像素相关联。根据(相同)源像素和目标像素的子组中的不同目标像素计算该组的不同SAD。

[0887] 计算相同源像素的该组SAD减少了被提取到DPA的数据量。

[0888] 目标像素的子组可以包括被顺序地存储在存储器模块中的目标像素。在计算该组SAD之前,从存储器模块中提取目标像素的子组。从存储器模块提取目标像素的子组是通过包括内容可寻址存储器缓存的收集单元执行的。

[0889] 每个SAD是基于先前计算的SAD以及基于当前计算的其他源像素与属于目标像素的子组的其他目标像素之间的绝对差来计算的。图25提供了这样的计算的示例。

[0890] 在步骤2620之后可以是步骤2630,通过该阵列中的第二组数据处理器响应于该组SAD的值找到目标像素的子组中的最佳匹配目标像素。

[0891] 步骤2620和2630可以包括在数据处理器阵列中存储计算结果-整个矩形像素阵列的SAD,列的SAD等等。应该注意的是,每个DPU的寄存器文件的深度可以足够长,以存储先前矩形阵列的最右边的列的SAD。例如-如果SG(A)中有15列,那么DPU的寄存器文件550应至少为15。

[0892] 在存储先前的SAD之后,则对于给定的SAD,存储第一先前计算的SAD、第二先前计算的SAD、位于第二目标像素列顶部的目标像素和位于第二源像素列顶部的源像素。

[0893] 参考步骤2620-第一先前计算的SAD可以反映以下两项之间的绝对差:(i)与给定的矩形源像素阵列相差第一源像素列和第二源像素列的矩形源像素阵列,以及(ii)与给定的矩形目标像素阵列相差第一目标像素列和第二目标像素列的矩形目标像素阵列。例如-SAD(SA,TA)。

[0894] 第二先前计算的SAD可以反映在第一源列与第一源列之间的绝对差。例如-SAD(SG

(A)的最左边一列,SG(B)的最左边一列)。

[0895] 步骤2620可以包括:

[0896] 1)通过从第一先前计算的SAD(例如SAD(SA,TA))中减去(a)先前计算的第二SAD(例如-SAD(SG(A)的最左列,SG(B)的最左列),以及(b)以下两项之间的绝对差:(i)位于第二目标像素列的顶部上的目标像素,和(ii)位于第二源像素列的顶部上的源像素(例如-OSP 1112与OTP 1112'之间的绝对差)。

[0897] 2)向中间结果添加第二目标像素列的最低目标像素与第二源像素列的最低源像素之间的绝对差(例如-NSP 1116与NTP 1116'之间的绝对差)。

[0898] 应当注意的是,找到最佳匹配目标像素可涉及迭代过程,并且可以执行步骤2610、2620和2630的多次重复-针对不同像素的子组并且通过比较这些多次迭代的结果-目标像素的组中的最佳匹配目标像素可以被找到。

[0899] 还要注意的是,处理单元阵列可以并行执行多个视差计算(针对不同的源像素和/或针对不同的目标像素)。

[0900] 图28示出了根据本发明实施例的由DPA处理的八个源像素和三十二个目标像素。图29示出根据本发明的实施例的源像素阵列。图30示出根据本发明的实施例的目标像素阵列。

[0901] 计算出与以下像素相关的SAD:源像素(SP0,SP1,SP2和SP3)和(SP'0,SP'1,SP'2和SP'3)、4×8目标像素(包括TP0,TP1,TP2和TP3的最左列)以及另外的4×8目标像素(包括TP'0,TP'1,TP'2和TP'3的最左列)。

[0902] 源像素SP0,SP1,SP2和SP3属于同一列,它们的SAD以流水线方式计算:

[0903] 1)针对SP0和某个目标像素计算SAD。

[0904] 2)在计算SP1和某个目标像素的SAD时使用先前的计算。

[0905] 3)在计算SP2和某个目标像素的SAD时使用先前的计算。

[0906] 4)在计算SP3和某个目标像素的SAD时使用先前的计算。

[0907] 在计算源像素SP0、SP1、SP2和SP3的SAD的同时、PMA还计算SP'0、SP'1、SP'2和SP'3的SAD。SP'0、SP'1、SP'2和SP'3属于同一列、且它们的SAD以流水线方式计算:

[0908] 1)针对SP'0和某个目标像素计算SAD。

[0909] 2)在计算SP'1和某个目标像素的SAD时使用先前的计算。

[0910] 3)在计算SP'2和某个目标像素的SAD时使用先前的计算。

[0911] 4)在计算SP'3和某个目标像素的SAD时使用先前的计算。

[0912] DPA 500可以并行地计算每个源像素和多个其他目标像素的SAD。

[0913] 例如,假定4×8目标像素的第一行包括TP0和七个移位目标像素(TP0、Ts1P0、Ts2P0、Ts2P0、Ts3P0、Ts4P0、Ts5P0、Ts6P0、Ts7P0),则SP0的SAD的计算可以包括针对SP0和TP0、Ts1P0、Ts2P0、Ts2P0、Ts3P0、Ts4P0、Ts5P0、Ts6P0、Ts7P0中的每一个计算SAD。

[0914] 计算任何SAD时,都需要计算新像素的绝对差。图29示出了四个新的源像素NS0、NS1、NS2和NS3(用于计算与SP0、SP1、SP2和SP3以及仅一个目标像素列相关的SAD)。

[0915] 图30示出了32个新的目标像素:

[0916] 1)用于计算SP0与八个不同的目标像素-NT0、Ns1T0、Ns2T0、Ns3T0、Ns4T0、Ns5T0、Ns6T0、Ss7T0的SAD的新目标像素。

[0917] 2)用于计算SP1和八个不同的目标像素-NT1、Ns1T1、Ns2T1、Ns3T1、Ns4T1、Ns5T1、Ns6T1、Ss7T1的SAD的新的目标像素。

[0918] 3)用于计算SP2与八个不同的目标像素-NT2、Ns1T2、Ns2T2、Ns3T2、Ns4T2、Ns5T2、Ns6T2、Ss7T2的SAD的新目标像素。

[0919] 4)用于计算SP3与八个不同的目标像素-NT3、Ns1T3、Ns2T3、Ns3T3、Ns4T3、Ns5T3、Ns6T3、Ss7T3的SAD的新目标像素。

[0920] 图31示出了八个DPU的组1131、1132、1133、1134、1135、1136、1137和1138-每个组包括4个DPU。

[0921] 每个组1131、1132、1133和1134计算像素SP0、SP1、SP2和SP3的SAD-但针对不同的目标像素(TP0、TP2、TP3和TP4)。

[0922] 每个组1135、1136、1137和1138计算像素SP'0、SP'1、SP'2和SP'3的SAD-但针对不同的目标像素(TP0、TP2、TP3和TP4)

[0923] 像素组1140对由组1131-1138计算出的SAD执行最小化操作。

[0924] 相应地,方法2600可以包括由数据处理器阵列的第一组数据处理器计算与多个源像素和目标像素的多个子组相关联的多组SAD;其中,多组SAD中的每个SAD是基于先前计算的SAD和当前计算的绝对差计算的;以及通过该阵列中的第二组数据处理器并针对源像素,响应于与源像素相关联的SAD的值找到最佳匹配目标像素。

[0925] 多组SAD可以包括SAD的子组,SAD的每个子组与多个源像素和目标像素的多个子组中的目标像素的多个子组相关联。例如,组1131-1138计算不同的SAD子组。

[0926] 多个源像素可以属于矩形像素阵列的列并且彼此相邻。

[0927] 计算多组SAD可以包括并行计算不同SAD子组的SAD。

[0928] 计算可以包括以顺序方式计算属于相同SAD子组的SAD。

[0929] 以下文字说明了根据本发明的实施例的一些PMA状态和配置缓冲器。

[0930] 这些状态和配置缓冲器109包括PMA控制状态寄存器,PMA暂停启用控制寄存器和PMA暂停事件状态寄存器。

[0931] 控制寄存器可以允许例如标量单元确定图像处理器的预定操作时间段。另外地或替代地,标量单元可以暂停图像处理器(而不更改PMA的状态)并且编程程序处理器,将控制信号发送到程序处理器并且从相同的点恢复图像处理器的操作(除了由标量单元引入的更改)。

[0932] PMA控制状态寄存器(p_PmaCsr)

[2:0]	lsuAddSel [0]	地址选择 LSU0

[0933]

[5:3]	lsuAddSel [l]	地址选择 LSU1
[8:6]	lsuAddSel [2]	地址选择 LSU2

[11:9] lsuAddSel [3] 地址选择 LSU3

[14:12] lsuAddSel [4] 地址选择 LSU4

[17:15] lsuAddSel [5] 地址选择 LSU5

[23:18]保留

[0934] [26:24] agStopSel AGU 生成的停止条件选择

[27] XorParity 在启用时写入同时反转奇偶校验（parity）

[29:28] sysMemMap 系统存储器映射

[30] proEn 程序启用

[31] suspCntEn 挂起计数器启用

[0935] PMA暂停启用控制寄存器(HaltOnEvent)

[15:0] mbParityEn 存储体[15:0]奇偶校验错误暂停启用

[16] spmparityEn SU 程序存储器奇偶校验错误暂停启用

[17] sdmparityEn SU 数据存储器奇偶校验错误暂停启用

[18] dmaIntEn DMA 中断暂停启用

[0936] [19] suRdErrEn 标量单元读取错误暂停启用

[20] suDivZeroEn 标量单元除以零暂停启用

[21] SuEnEn 标量单元中断暂停启用

[22] suHaltEn 标量单元暂停启用

[31:23]保留

[0937] PMA暂停事件状态寄存器(HoeStatus)

[15:0] mbParityErr 存储体[15:0]奇偶校验错误

[16] spmparityErr SU 程序存储器奇偶校验错误

[0938]

[17] sdmparityErr SU 数据存储器奇偶校验错误

[18] dmaInt DMA 中断

[19] suRdErr 标量单元读取错误

[20] suDivZero 标量单元除以零

[0939] [21] suInt 标量单元中断

[22] SuHalt 标量单元暂停

[31:23]保留

[0940] 挂起-恢复-事件计数器&递增。

[0941] 此功能启用挂起操作,更改某些配置而不清空计算流水线,并恢复操作。它通过以下寄存器实现:(a)挂起计数器启用控制位(p_PmaCsr中的suspCntEn),(b)挂起计数器(p_SuspCnt)和(c)挂起重置控制(p_RstCtl)。

[0942] 当启用(suspCntEn=1)时,挂起计数器倒计时。当达到零时,PMA挂起操作(保持停止状态),直到suspCntEn重置,或者p_SuspCnt用新值(!=0)写入。在停止期间,标量单元可以重新配置PMA(指令,常量...)。当重置suspCntEn或写入p_StallCnt时,PMA将用新配置恢复其操作。p_RstCtl定义在恢复时哪些功能被重置。

[0943] 可以重置的特征是:

[0944] 1)DPU微控制器。

[0945] 2)DPA程序存储器。

[0946] 3)BU程序存储器。

[0947] 4)SB程序存储器。

[0948] 5)地址生成器。

[0949] 6)BU读取缓冲器。

[0950] 7)GU迭代器(1位)

[0951] 在挂起时重置控制

[0952]

字段	名称	说明
[5:0]	addGenRst	在挂起时重置地址生成器
[5:0]	nuRdBufRst	在挂起时清空读取缓冲器线
[5:0]	nuUcRst	在挂起时重置 BU 微控制器

[0953]

[5:0]	dpuUcRst	在挂起时重置 DPU 微控制器
[1:0]	sbUcRst	在挂起时重置存储缓冲器微控制器

[0954] 事件计数器p_EventCnt

[0955] 一个简单的计数器,用DPU时钟计数(在p_stall期间不计数)。计数器是通过配置预设的。只要计数器为空,就会向标量单元发出一个事件信号。这个计数器通过配置总线是可读的。

[0956] 挂起和事件增量寄存器p_SuspEventInc

[0957] 低的一半(15..0)用于递增挂起计数器,在其正常递减期间和同时递增。高的一半(31..16)也同时递增事件计数器。

[0958] 图33示出根据本发明的实施例的方法3300。

[0959] 方法3300可以从步骤3310开始,从源像素的组中选择源像素。所选择的源像素将被称为“源像素”。

[0960] 在步骤3310之后可以是步骤3320,针对源像素的组中的每个源像素执行扭曲计算过程,包括:

[0961] 1)计算(3321)或接收关于所选择的源像素的扭曲参数。扭曲参数可以包括在扭曲计算期间应当处理的给定的目标像素的第一权重和第二权重(Wx,Wy)和坐标(x,y)。第一权重和第二权重由处理单元阵列(DPA)中的第一组处理单元(DPU)接收。

[0962] 2)向诸如收集单元的存储器单元请求(3322)相邻目标像素(其包括给定的目标像素)。收集单元可以在各种操作模式下接收4个坐标并将它们转换为16个目标像素-四组相邻目标像素。

[0963] 3)由第二组处理单元接收(3323)与源像素相关联的相邻目标像素。

[0964] 4)由第二组处理单元响应于相邻目标像素的值和一对权重计算(3324)扭曲结果;向存储器模块提供扭曲结果。

[0965] 步骤3321、3322、3323和3324可以以流水线方式执行。

[0966] 步骤3320之后是步骤3330,检查扭曲是否是针对该组的所有源像素计算的。如果不是-结束扭曲计算。

[0967] 步骤3326可以包括在第二组的处理单元之间转播一些相邻目标像素的值。

[0968] 对术语“包括”(comprise)、“包括(comprises)”、“包括(comprising)”、“包含(including)”、“可包含(may include)”以及“包含(includes)”中的任何一个的任何引用可以应用于术语“由.....构成”(consists)、“由.....构成(consisting)”、“以及基本上由.....构成(and consisting essentially of)”。例如-任何描述步骤的方法都可以包括比图中所示的步骤更多的步骤,只是图中所示的步骤或者基本上仅在图中示出的步骤。这同样适用于设备、处理器或系统的组件以及存储在任何非暂时性计算机可读存储介质中的指令。

[0969] 本发明也可以在用于在计算机系统上运行的计算机程序中实施,该计算机程序至少包括用于在可编程装置(诸如计算机系统)上运行时执行根据本发明的方法的步骤、或者使得可编程装置能够执行根据本发明的设备或系统的功能的代码部分。计算机程序可以致使存储系统将硬盘驱动器分配给硬盘驱动器组。

[0970] 计算机程序是诸如特定应用程序和/或操作系统的指令列表。计算机程序可以例如包括以下各项中的一个或多个:子例程、函数、过程、对象方法、对象实现、可执行应用程序、小应用程序、小服务程序、源代码、目标代码、共享库/动态加载库和/或被设计用于在计算机系统上执行的其它指令序列。

[0971] 计算机程序可以内部地存储于非暂时性计算机可读介质上。所有或一些计算机程序可以设置在永久地、可移除地或远程地耦合到信息处理系统的计算机可读介质上。计算机可读介质可以包括例如但不限于任何数量的以下各项:磁存储介质,包括硬盘和磁带存储介质;光存储介质,诸如光盘介质(例如,CD-ROM、CD-R等)和数字视频盘存储介质;非易失

性存储器存储介质，包括基于半导体的存储器单元，诸如闪存存储器、EEPROM、EPROM、ROM；铁磁数字存储器；MRAM；易失性存储介质，包括寄存器、缓冲器或缓存、主存储器、RAM等。

[0972] 计算机进程通常包括执行(运行)程序或程序的一部分、当前程序值和状态信息以及由操作系统用以管理进程的执行的资源。操作系统(OS)是管理计算机资源共享、并为程序员提供用于访问这些资源的接口的软件。操作系统处理系统数据和用户输入，并通过分配和管理作为对系统的用户和程序的服务的任务和内部系统资源进行响应。

[0973] 计算机系统可以例如包括至少一个处理单元、相关联的存储器和多个输入/输出(I/O)设备。当执行计算机程序时，计算机系统根据计算机程序处理信息，并且经由I/O设备产生所得到的输出信息。

[0974] 在前述说明书中，已经参考本发明的实施例的具体示例描述了本发明。然而，显然，在不脱离如所附权利要求中阐述的本发明的更广泛的精神和范围的情况下，可以在其中进行各种修改和改变。

[0975] 此外，说明书和权利要求中的术语“前”、“后”、“顶部”、“底部”等，如果有的话，用于描述目的，而不一定用于描述永久相对位置。应当理解，这样使用的术语在适当的情况下是可互换的，使得本文所描述的本发明的实施例例如能够在除了本文所示出或另外描述的那些方向之外的其它方向上操作。

[0976] 本文讨论的连接可以是适合于例如经由中间设备往来于相关节点、单元或设备传送信号的任何类型的连接。因此，除非暗示或另外陈述，否则连接可以例如是直接连接或间接连接。可以参照单个连接、多个连接、单向连接或双向连接来图示或描述连接。然而，不同的实施例可以改变连接的实现。例如，可以使用单独的单向连接而不是双向连接，反之亦然。而且，多个连接可以用串行或以时分复用方式传送多个信号的单个连接代替。同样，携带多个信号的单个连接可以被分离成携带这些信号的子组的各种不同的连接。因此，传送信号有很多选择。

[0977] 虽然在实施例中已经描述了具体的导电类型或电势的极性，但是应该理解，导电类型和电势的极性可以颠倒。

[0978] 本文描述的每个信号可以被设计为正逻辑或负逻辑。在负逻辑信号的情况下，在逻辑真状态对应于逻辑电平零的情况下，信号是低电平有效的。在正逻辑信号的情况下，在逻辑真状态对应于逻辑电平的情况下，信号为高电平有效。注意，本文描述的任何信号可以被设计为负逻辑信号或正逻辑信号。因此，在替代实施例中，被描述为正逻辑信号的那些信号可以被实现为负逻辑信号，而被描述为负逻辑信号的那些信号可以被实现为正逻辑信号。

[0979] 此外，当涉及将信号、状态位或类似装置呈现为其逻辑真或逻辑假的状态时，本文使用术语“断言”或“置位”和“否定”(或“无效”或“清除”)分别用于其逻辑真或逻辑假状态。如果逻辑真状态是逻辑电平1，则逻辑假状态是逻辑电平0。如果逻辑真状态是逻辑电平0，则逻辑假状态是逻辑电平1。

[0980] 本领域技术人员将认识到，逻辑块之间的边界仅仅是说明性的，并且可替代实施例可以合并逻辑块或电路元件，或者对各种逻辑块或电路元件施加功能的替代分解。因此，应当理解，本文所描述的体系结构仅仅是示例性的，并且实际上可以实施实现相同功能的许多其它体系结构。

[0981] 实现相同功能的部件的任何布置被有效地“关联”，使得实现期望的功能。因此，本文组合以实现特定功能的任何两个部件可以被看作彼此“相关联”，使得实现期望的功能，而与体系结构或中间部件无关。同样地，如此关联的任何两个组件也可以被视为彼此“可操作地连接”或“可操作地耦合”以实现期望的功能。

[0982] 此外，本领域技术人员将认识到上述操作之间的边界仅是说明性的。多个操作可以组合成单个操作，单个操作可以分布在附加操作中，并且操作可以在时间上至少部分地重叠地执行。此外，可替代实施例可以包括特定操作的多个实例，并且在各种其它实施例中可以改变操作的顺序。

[0983] 还例如，在一个实施例中，所示示例可以被实施为位于单个集成电路上或在同一设备内的电路。可替代地，该示例可实施为以合适方式彼此互连的任何数目的单独集成电路或单独设备。

[0984] 还例如，示例或其部分可以实施为物理电路的软件或代码表示，或者可以实现为可转换成物理电路的逻辑表示的软件或代码表示，诸如以任何适当类型的硬件描述语言实施。

[0985] 此外，本发明不限于在非可编程硬件中实施的物理设备或单元，而是还可以应用在能够通过根据适当的程序代码进行操作来执行期望的设备功能的可编程设备或单元中，例如大型机、小型计算机、服务器、工作站、个人计算机、记事本、个人数字助理、电子游戏、汽车和其他嵌入式系统、蜂窝电话和各种其他无线设备，在本申请中通常称为“计算机系统”。

[0986] 然而，其它修改、变化和替代也是可能的。因此，说明书和附图被认为是说明性的而不是限制性的。

[0987] 在权利要求中，置于括号之间的任何附图标记不应被解释为限制权利要求。词“包括”不排除权利要求中列出的那些之外的其他元素或步骤的存在。此外，如本文所使用的术语“一(a)”或“一个(an)”被定义为一个或多于一个。此外，在权利要求中使用诸如“至少一个”和“一个或更多个”之类的引导性短语不应被解释为暗示通过不定冠词“一(a)”或“一个(an)”引入另一个权利要求要素将包含该引导的权利要求要素的任何特定权利要求限制于仅包含一个该要素的发明，即使当相同的权利要求包括引导性短语“一个或更多个”或“至少一个”和不定冠词诸如“一(a)”或“一个(an)”时。这同样适用于定冠词的使用。除非另有说明，否则诸如“第一”和“第二”的术语用于任意地区分此类术语所描述的元素。因此，这些术语不一定旨在指示这些元素的时间或其他优先级。某些措施在相互不同的权利要求中引用的不争事实并不指示这些措施的组合不能有利地被使用。

[0988] 虽然本文已经示出和描述了本发明的某些特征，但是本领域普通技术人员将想到许多修改、替换、改变和等同物。因此，应当理解，所附权利要求旨在覆盖落入本发明的真实精神内的所有这样的修改和改变。

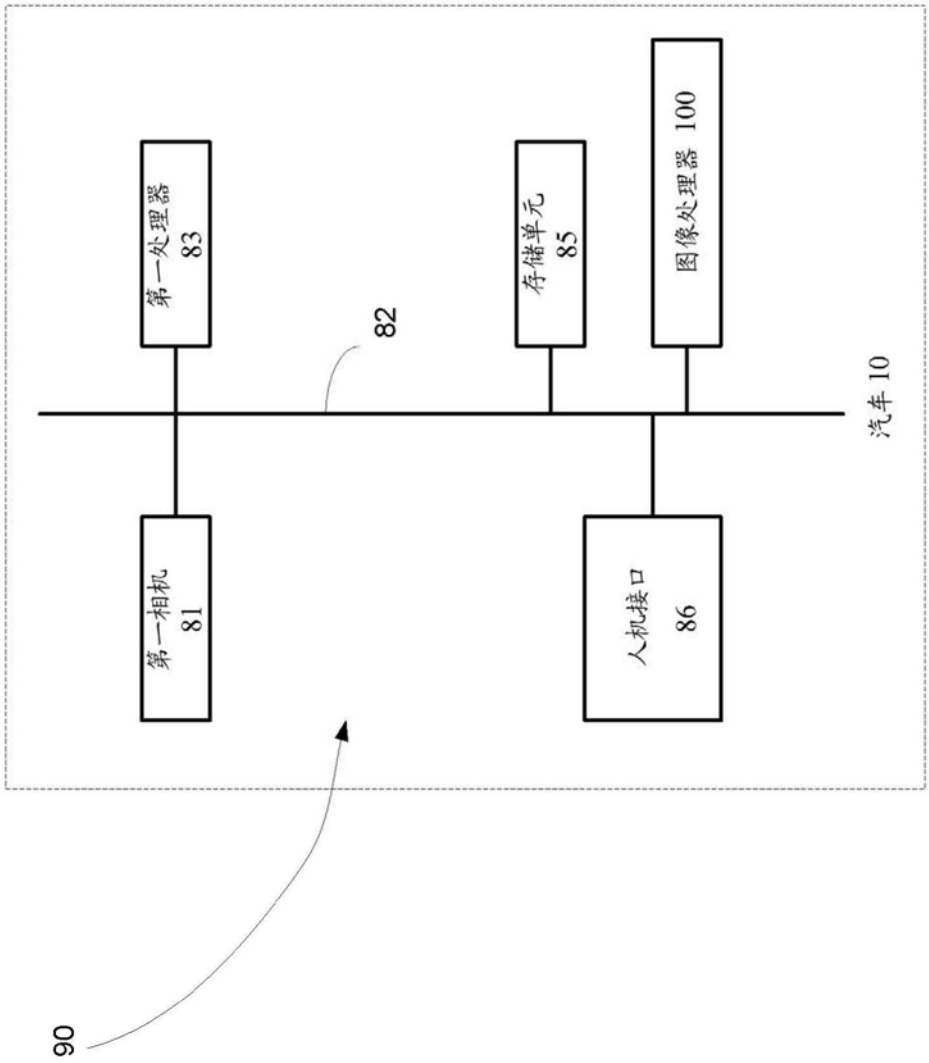


图1

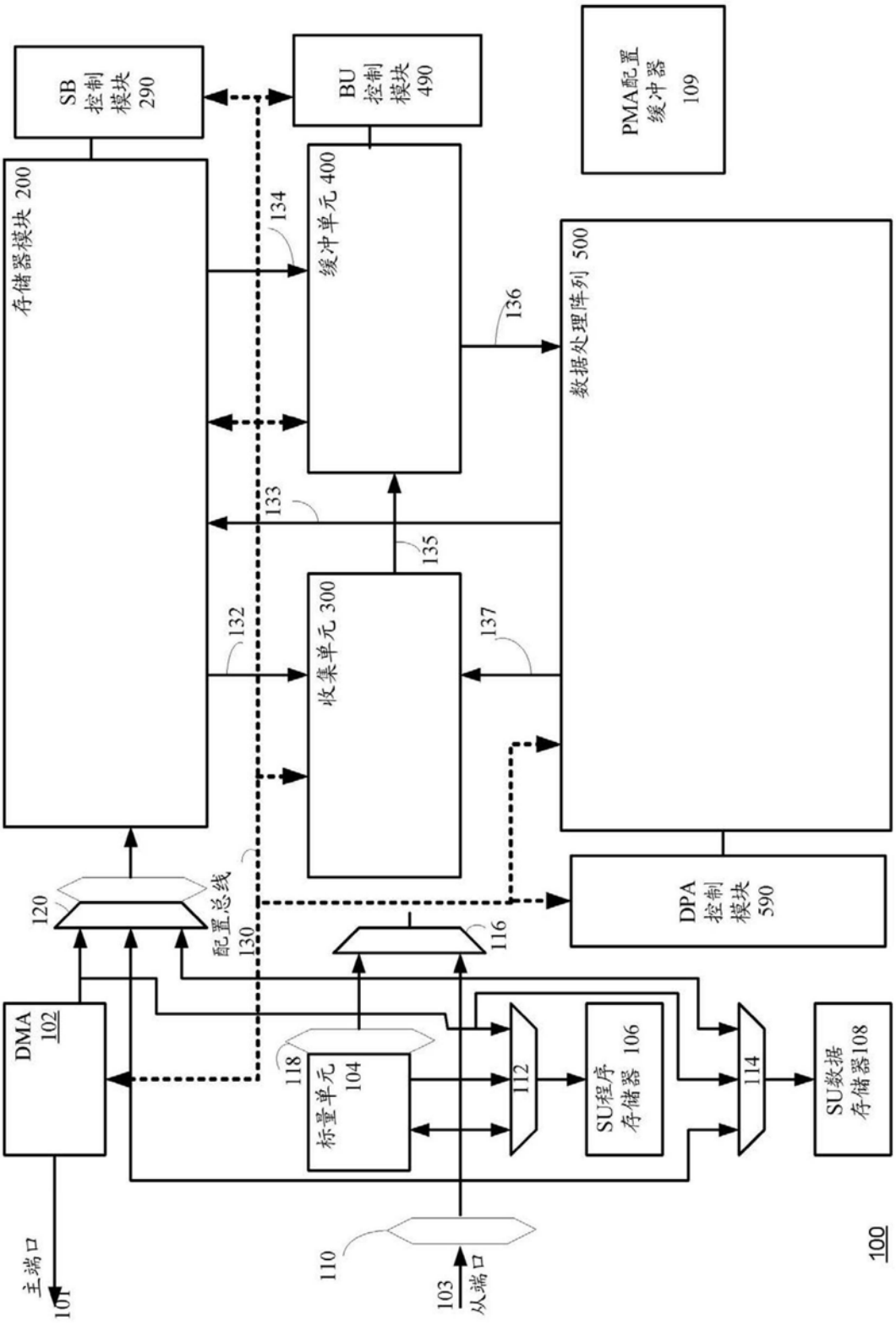

SB 控制模块 290
BU 控制模块 490
PMA配置缓冲器 109
存储器模块 200
缓冲单元 400
数据处理阵列 500
收集单元 300
DPA 控制模块 590
配置总线
DMA 102
标量单元 104
SU程序存储器 106
SU数据存储器108
主端口
从端口
100
103
107
110
112
114
116
118
120
130
132
133
134
135
136
137

图2

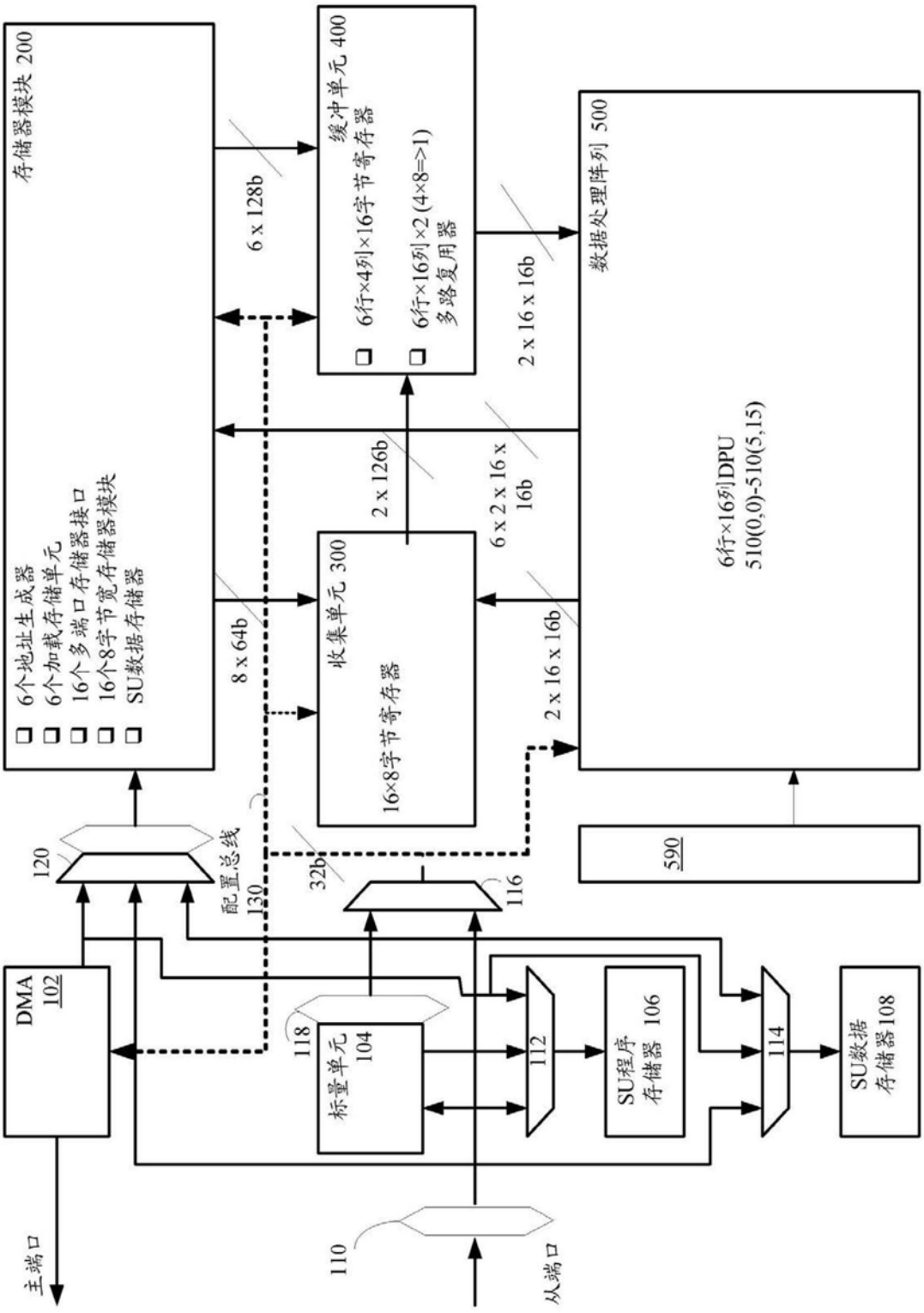

存储器模块 200
6个地址生成器
6个加载存储单元
16个多端口存储器接口
16个8字节宽存储器模块
SU数据存储器
缓冲单元 400
6行×4列×16字节寄存器
6行×16列×2 (4×8=>1)
多路复用器
数据处理阵列 500
6行×16列DPU
510(0,0)-510(5,15)
收集单元 300
16×8字节寄存器
6 x 128b
2 x 16 x 16b
2 x 126b
6 x 2 x 16 x 16b
8 x 64b
2 x 16 x 16b
590
120
配置总线
130
32b
116
DMA
102
118
标量单元
104
112
SU程序
存储器 106
114
SU数据
存储器108
110
主端口
从端口

图3

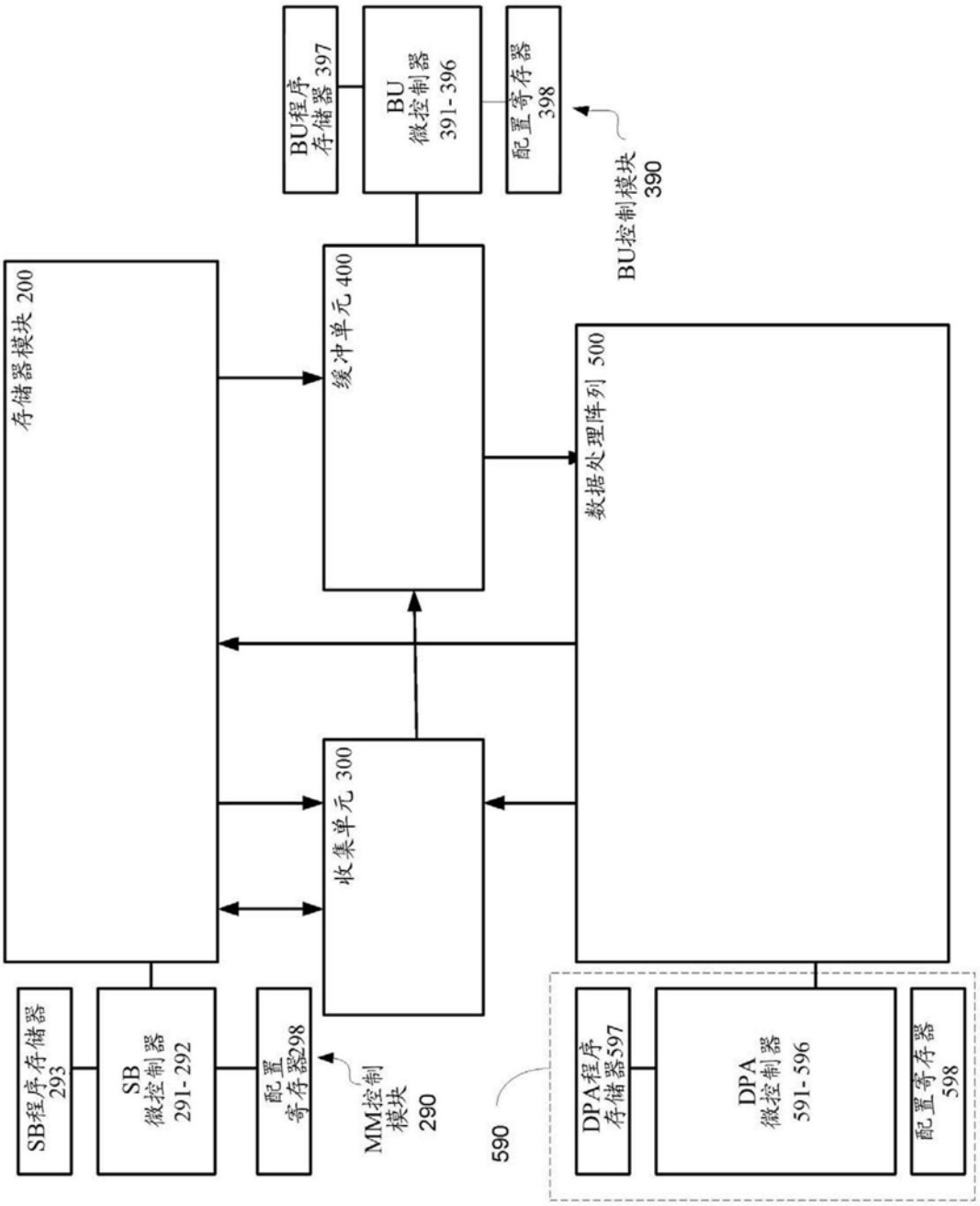

BU程序存储器 397
BU微控制器 391-396
配置寄存器 398
BU控制模块 390
存储器模块 200
缓冲单元 400
数据处理阵列 500
收集单元 300
SB程序存储器 293
SB微控制器 291-292
配置寄存器298
MM控制模块 290
590
DPA程序存储器597
DPA微控制器 591-596
配置寄存器 598

图4

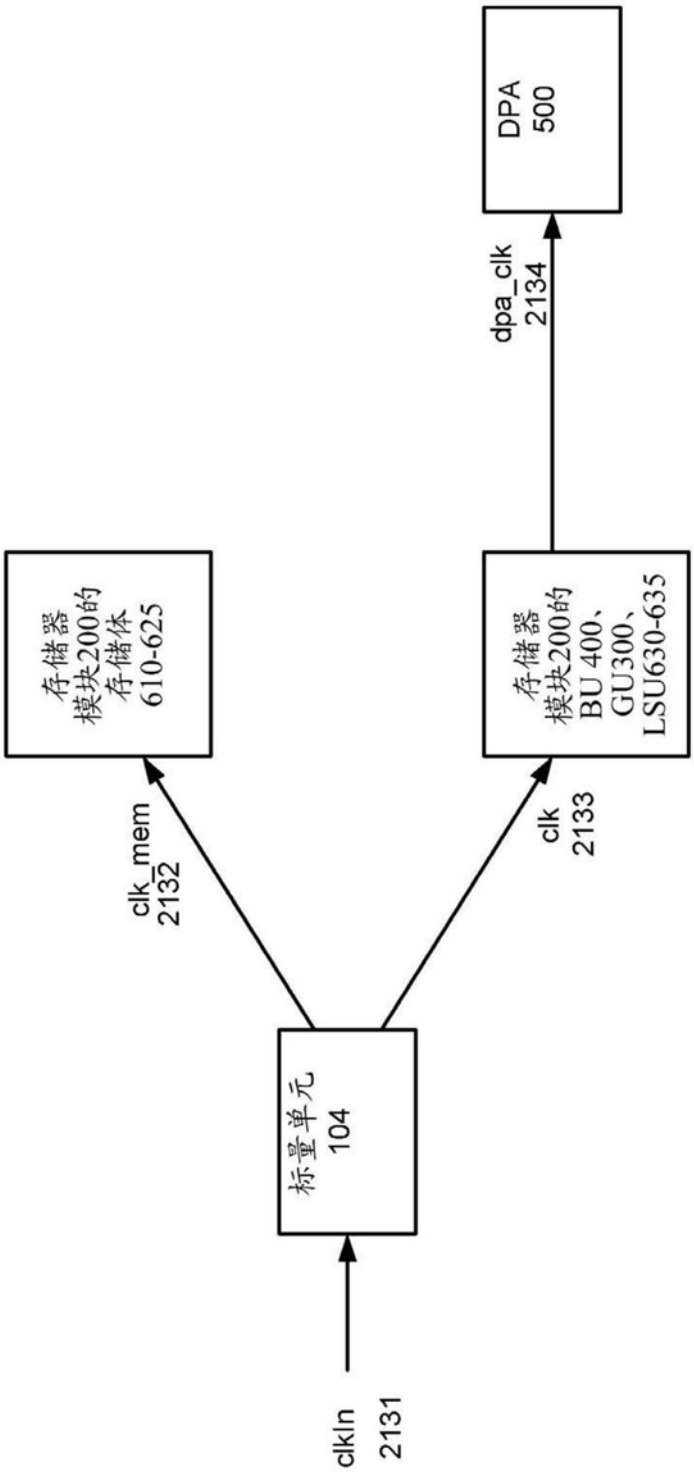

DPA
500
dpa_clk
2134
存储器
模块200的
存储体
610-625
存储器
模块200的
BU 400、
GU300、
LSU630-635
clk_mem
2132
clk
2133
标量单元
104
clkIn
2131

图5

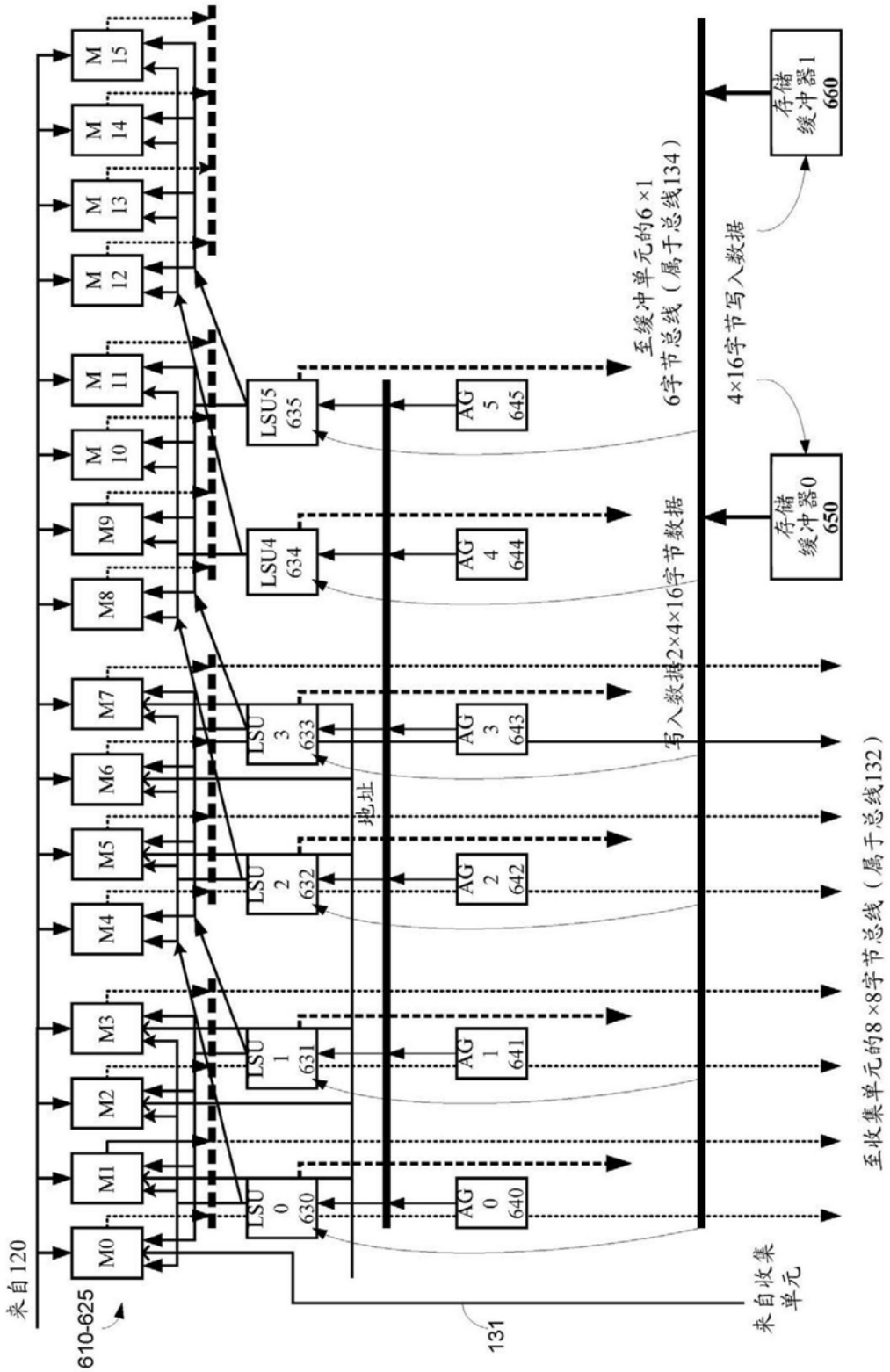

来自120
610-625
M0
M1
M2
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
M13
M14
M15
LSU 0 630
LSU 1 631
LSU 2 632
LSU 3 633
LSU4 634
LSU5 635
AG 0 640
AG 1 641
AG 2 642
AG 3 643
AG 4 644
AG 5 645
地址
131
来自收集单元
至缓冲单元的6×1 6字节总线（属于总线134）
4×16字节写入数据
写入数据2×4×16字节数据
存储缓冲器0 650
存储缓冲器1 660
至收集单元的8×8字节总线（属于总线132）

图6

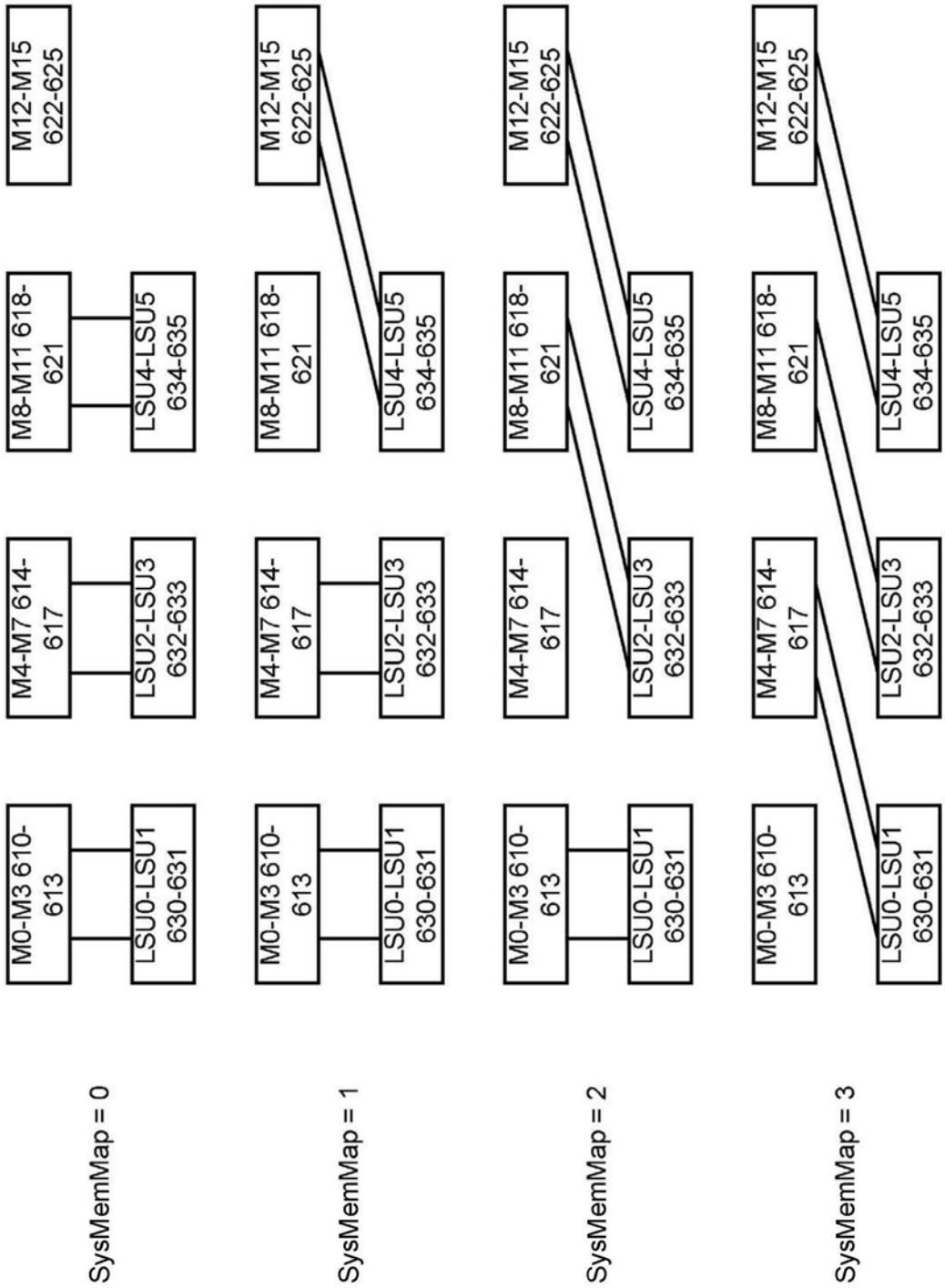

M12-M15 622-625
M8-M11 618-621
LSU4-LSU5 634-635
M4-M7 614-617
LSU2-LSU3 632-633
M0-M3 610-613
LSU0-LSU1 630-631
SysMemMap = 0
SysMemMap = 1
SysMemMap = 2
SysMemMap = 3

图7

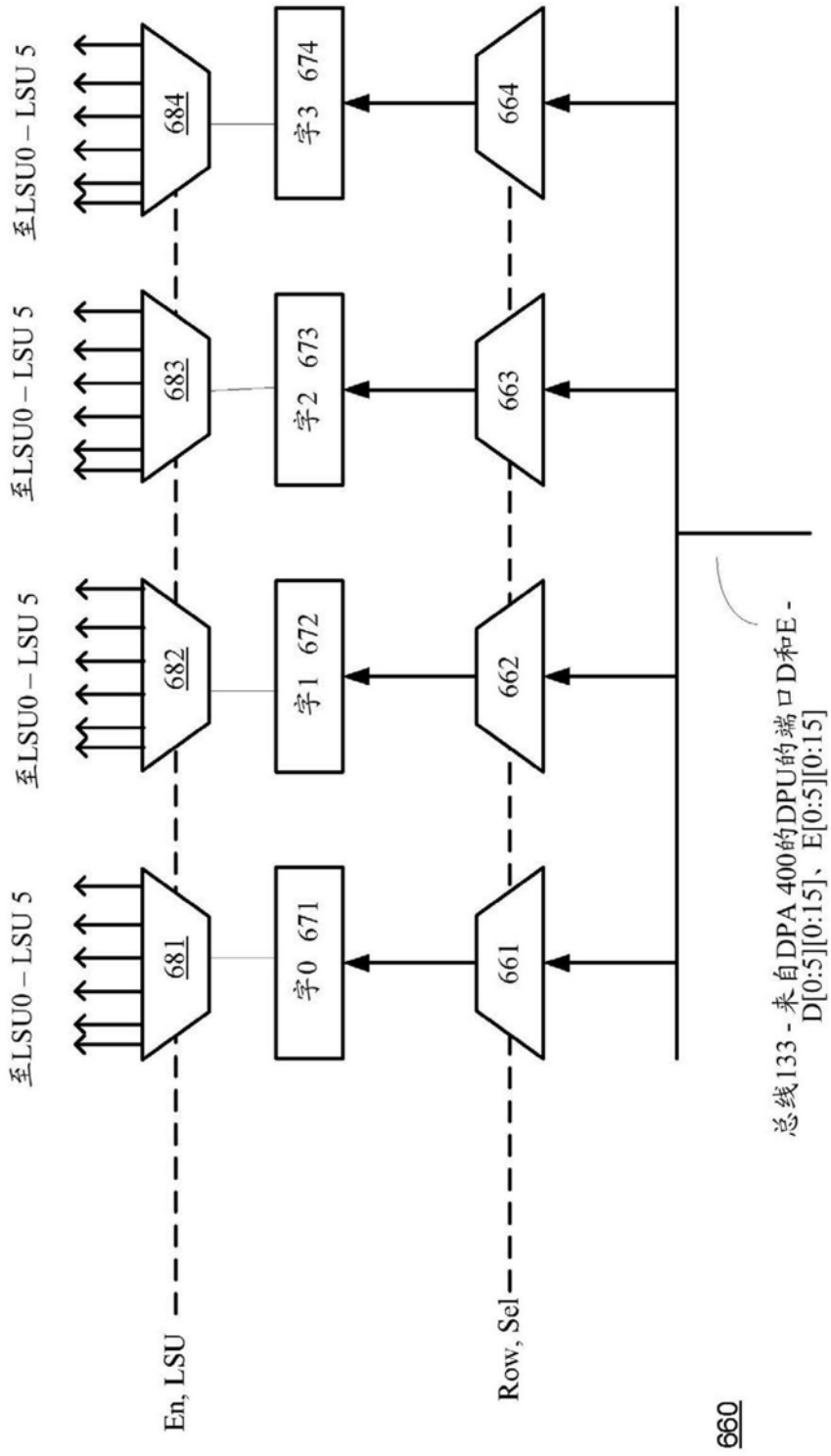

至LSU0 – LSU 5
至LSU0 – LSU 5
至LSU0 – LSU 5
至LSU0 – LSU 5
681
682
683
684
字0 671
字1 672
字2 673
字3 674
661
662
663
664
En, LSU
Row, Sel
660
总线133 - 来自DPA 400的DPU的端口D和E - D[0:5][0:15]、E[0:5][0:15]

图8

数据选择编码 - 第一部分

D # 和E # 表示DPU [row, #]的输出D和E，其中'row'是来自当前所选指令的行选择，且D' # 是DPU [row + 1, #] 的输出D

```
 0: [D0,D1,..,D7] [D8,D9..,D15] [E0,E1,..,E7] [E8,E9..,E15]
 1: [D0,E0,D1,E1,..,D3,E3] [D4,E4,D5,E5,..,D7,E7] [D8,E8,D9,E9,..,D11,E11] [D12,E12,D13,E13,..,D15,E15]
 2: [D0,D2,..,D14] [D1,D3..,D15] [E0,E2,..,E14] [E1,E3..,E15]
 3: [D0,E0,D2,E2,..,D6,E6] [D8,E8,D10,E10,..,D14,E14] [D1,E1,D3,E3,..,D7,E7] [D9,E9,D11,E11,..,D15,E15]
 4: [D0,D1,..,D7] [D8,D9..,D15] [D'0,D'1,..,D'7] [D'8,D'9..,D'15]
 5: [D0,D'0,D1,D'1,..,D3,D'3] [D4,D'4,D5,D'5,..,D7,D'7] [D8,D'8,D9,D'9,..,D11,D'11] [D12,D'12,D13,D'13,..,
 6: [D0,D2,..,D14] [D1,D3..,D15] [D'0,D'2,..,D'14] [D'1,D'3..,D'15]
 7: [D0,D'0,D2,D'2,..,D6,D'6] [D8,D'8,D10,D'10,..,D14,D'14] [D1,D'1,D3,D'3,..,D7,D'7] [D9,D'9,D11,D'11,..,
 8: [D4,D5,..,D11] [D12,D13,..,D15,E0,...E3] [E4,E5,..,E11] [E12,E13..,E15, D0,D1..D3]
 9: [D2,E2,D3,E3,..,D5,E5] [D6,E6,D7,E7,..,D9,E9] [D10,E10,D11,E11,..,D13,E13] [D14,E14,D15,E15,D0,E0,D1,E
10: [D8,D10,..,D14,D1,D3..,D7] [D9,D11,..,D15,E0,E2..,E6] [E8,E10,..,E14,E1,E3..,E7] [E9,E11,..,E15,D0,D2.
11: [D4,E4,D6,E6,..,D10,E10] [D12,E12,D14,E14,D1,E1,D3,E3] [D5,E5,D7,E7,..,D11,E11] [D13,E13,D15,E15,D0,E0
12: [D4,D5,..,D11] [D12,D13,..,D15,D'0,...D'3] [D'4,D'5,..,D'11] [D'12,D'13..,D'15, D0,D1..D3]
13: [D2,D'2,D3,D'3,..,D5,D'5] [D6,D'6,D7,D'7,..,D9,D'9] [D10,D'10,D11,D'11,..,D13,D'13]
[D14,D'14,D15,D'15,D0,D'0,D1,D'1]
14: [D8,D10,..,D14,D1,D3..,D7] [D9,D11,..,D15,D'0,D'2..,D'6] [D'8,D'10,..,D'14,D'1,D'3..,D'7]
[D'9,D'11,..,D'15,D0,D2..,D6]
15: [D4,D'4,D6,D'6,..,D10,D'10] [D12,D'12,D14,D'14,D1,D'1,D3,D'3] [D5,D'5,D7,D'7,..,D11,D'11]
[D13,D'13,D15,D'15,D0,D'0,D2,D'2]
```

图9

数据选择编码 - 第二部分
D # 和E # 表示DPU[row， #]的输出D和E，其中'row'是来自当前所选指令的行选择，且D' # 是DPU [row + 1， #] 的输出D

```
16: [Dl0,Dl1,..,Dl7,..,Dl15] [El0,El1,..,El7..,El15] [Dh0,Dh1,..,Dh7,..,Dh15] [Eh0,Eh1,..,Eh7..,Eh15]
 17: [Dl0,El0,Dl1,El1,..,..,Dl7,El7] [Dl8,El8,Dl9,El9,..,Dl15,El15] [Dh0,Eh0,Dh1,Eh1,..,..,Dh7,Eh7]
[Dh8,Eh8,Dh9,Eh9,..,Dh15,Eh15]
 18: [Dl0,Dl2,..,Dl14,Dl1,Dl3..,Dl15] [El0,El2,..,El14,El1,El3..,El15] [Dh0,Dh2,..,Dh14,Dh1,Dh3..,Dh15]
[Eh0,Eh2,..,Eh14,Eh1,Eh3..,Eh15]
 19: [Dl0,El0,Dl2,El2,..,Dl14,El14] [Dl1,El1,Dl3,El3,..,Dl15,El15] [Dh0,Eh0,Dh2,Eh2,..,Dh14,Eh14]
[Dh1,Eh1,Dh3,Eh3,..,Dh15,Eh15]
 20: [Dl0,Dl1,..,Dl7,..,Dl15] [D'l0,D'l1,..,D'l7..,D'l15] [Dh0,Dh1,..,Dh7,..,Dh15] [D'h0,D'h1,..,D'h7..,D'h15
 21: [Dl0,D'l0,Dl1,D'l1,..,..,..,Dl7,D'l7] [Dl8,D'l8,Dl9,D'l9,..,Dl15,D'l15] [Dh0,D'h0,Dh1,D'h1,..,..,..,Dh7,D'h7]
[Dh8,D'h8,Dh9,D'h9,..,Dh15,D'h15]
 22: [Dl0,Dl2,..,Dl14,Dl1,Dl3..,Dl15] [D'l0,D'l2,..,D'l14,D'l1,D'l3..,D'l15] [Dh0,Dh2,..,Dh14,Dh1,Dh3..,Dh15]
[D'h0,D'h2,..,D'h14,D'h1,D'h3..,D'h15]
 23: [Dl0,D'l0,Dl2,D'l2,..,Dl14,D'l14] [Dl1,D'l1,Dl3,D'l3,..,Dl15,D'l15] [Dh0,D'h0,Dh2,D'h2,..,Dh14,D'h14]
[Dh1,D'h1,Dh3,D'h3,..,Dh15,D'h15]
 24: [Dl4,Dl5,..,Dl15,El0,...El3] [El4,El5,..,El15, Dl0,Dl1..Dl3] [Dh4,Dh5,..,Dh15,Eh0,...Eh3] [Eh4,Eh5,..,Eh
Dh0,Dh1..Dh3]
 25: [Dl2,El2,Dl3,El3..,Dl9,El9] [Dl10,El10,Dl11,El11,..,Dl15,El15,Dl0,El0,Dl1,El1]
     [Dh2,Eh2,Dh3,Eh3..,Dh9,Eh9] [Dh10,Eh10,Dh11,Eh11,..,Dh15,Eh15,Dh0,Eh0,Dh1,Eh1]
 26: [Dl8,Dl10,..,Dl14,Dl1,Dl3..,Dl15,El0,El2..,El6] [El8,El10,..,El14,El1,El3..,El15,Dl0,Dl2..,Dl6]
     [Dh8,Dh10,..,Dh14,Dh1,Dh3..,Dh15,Eh0,Eh2..,Eh6] [Eh8,Eh10,..,Eh14,Eh1,Eh3..,Eh15,Dh0,Dh2..,Dh6]
 27: [Dl4,El4,Dl6,El6..,Dl14,El14,Dl1,El1,Dl3,El3] [Dl5,El5,Dl7,El7..,El15,Dl0,El0,Dl2,El2]
     [Dh4,Eh4,Dh6,Eh6..,Dh14,Eh14,Dh1,Eh1,Dh3,Eh3] [Dh5,Eh5,Dh7,Eh7..,Eh15,Dh0,Eh0,Dh2,Eh2]
 28: [Dl4,Dl5,..,Dl15,D'l0,...D'l3] [D'l4,D'l5,..,D'l15, Dl0,Dl1..Dl3] [Dh4,Dh5,..,Dh15,D'h0,...D'h3]
[D'h4,D'h5,..,D'h15, Dh0,Dh1..Dh3]
 29: [Dl2,D'l2,Dl3,D'l3..,Dl9,D'l9] [Dl10,D'l10,Dl11,D'l11,..,Dl15,D'l15,Dl0,D'l0,Dl1,D'l1]
     [Dh2,D'h2,Dh3,D'h3..,Dh9,D'h9] [Dh10,D'h10,Dh11,D'h11,..,Dh15,D'h15,Dh0,D'h0,Dh1,D'h1]
 30: [Dl8,Dl10,..,Dl14,Dl1,Dl3..,Dl15,D'l0,D'l2..,D'l6] [D'l8,D'l10,..,D'l14,D'l1,D'l3..,D'l15,Dl0,Dl2..,Dl6]
     [Dh8,Dh10,..,Dh14,Dh1,Dh3..,Dh15,D'h0,D'h2..,D'h6] [D'h8,D'h10,..,D'h14,D'h1,D'h3..,D'h15,Dh0,Dh2..,Dh6]
 31: [Dl4,D'l4,Dl6,D'l6..,Dl14,D'l14,Dl1,D'l1,Dl3,D'l3] [Dl5,D'l5,Dl7,D'l7..,D'l15,Dl0,D'l0,Dl2,D'l2]
     [Dh4,D'h4,Dh6,D'h6..,Dh14,D'h14,Dh1,D'h1,Dh3,D'h3] [Dh5,D'h5,Dh7,D'h7..,D'h15,Dh0,D'h0,Dh2,D'h2]
```

图10

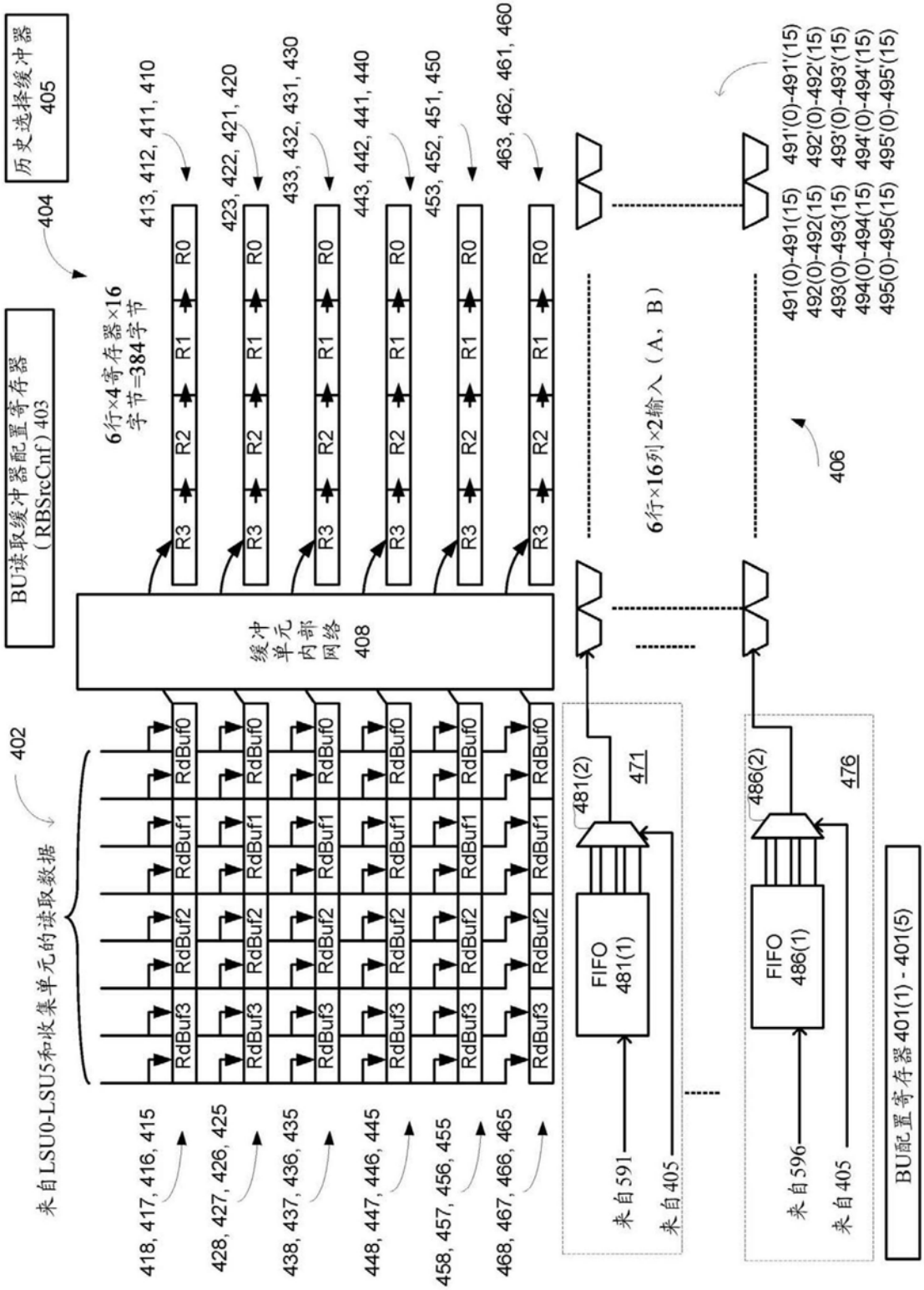

历史选择缓冲器
405
BU读取缓冲器配置寄存器
(RBSrcCnf) 403
404
6行×4寄存器×16
字节=384字节
413, 412, 411, 410
423, 422, 421, 420
433, 432, 431, 430
443, 442, 441, 440
453, 452, 451, 450
463, 462, 461, 460
R3
R2
R1
R0
缓冲
单元
内部
网络
408
402
来自LSU0-LSU5和收集单元的读取数据
RdBuf3
RdBuf2
RdBuf1
RdBuf0
418, 417, 416, 415
428, 427, 426, 425
438, 437, 436, 435
448, 447, 446, 445
458, 457, 456, 455
468, 467, 466, 465
6行×16列×2输入（A，B）
406
491(0)-491(15)
492(0)-492(15)
493(0)-493(15)
494(0)-494(15)
495(0)-495(15)
491'(0)-491'(15)
492'(0)-492'(15)
493'(0)-493'(15)
494'(0)-494'(15)
495'(0)-495'(15)
481(2)
471
FIFO
481(1)
来自591
来自405
486(2)
476
FIFO
486(1)
来自596
来自405
BU配置寄存器401(1) - 401(5)

图11

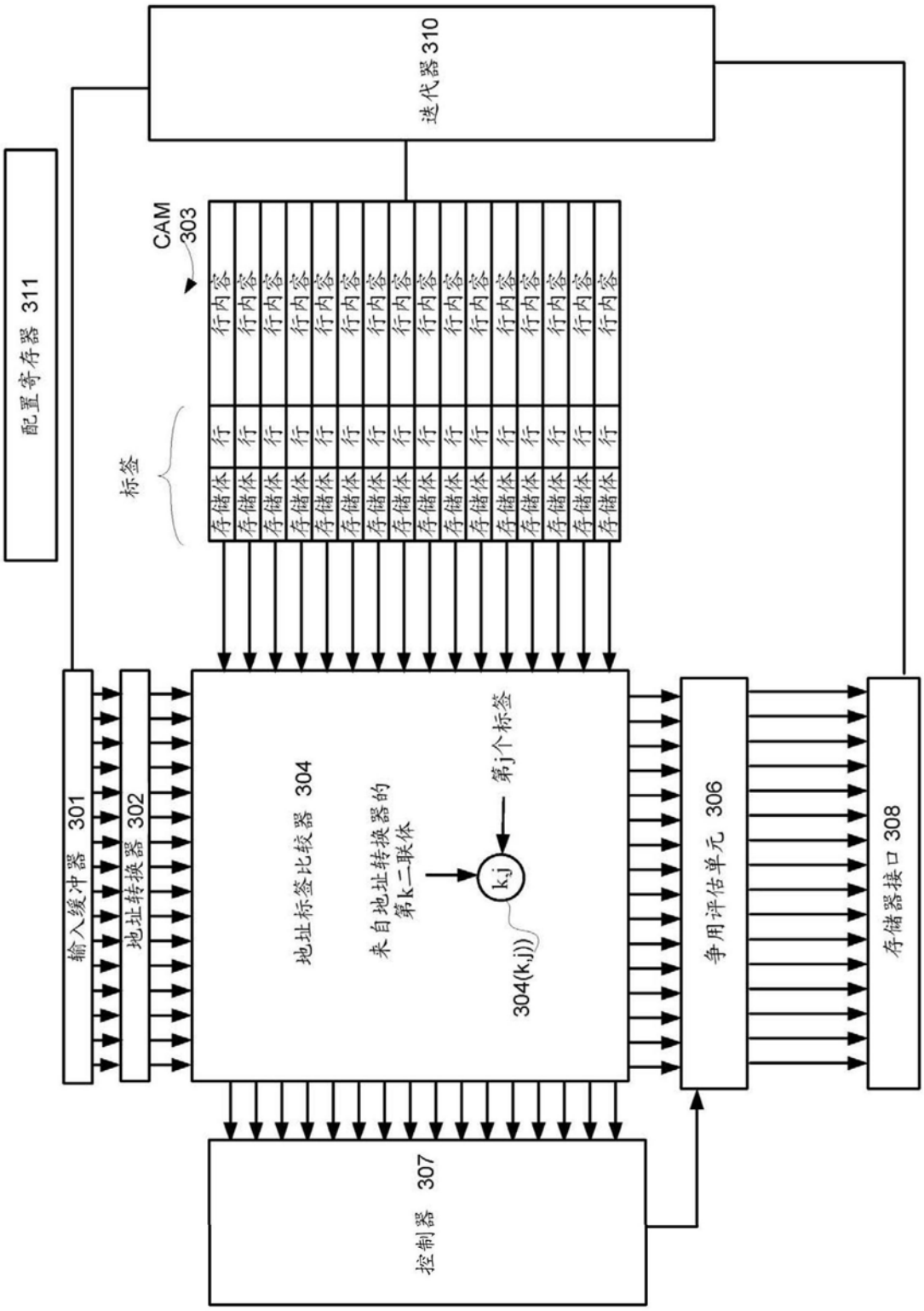

迭代器 310
配置寄存器 311
CAM
303
标签
存储体
行
行内容
输入缓冲器 301
地址转换器 302
地址标签比较器 304
来自地址转换器的
第k二联体
第j个标签
k,j
304(k,j))
争用评估单元 306
存储器接口 308
控制器 307

图12

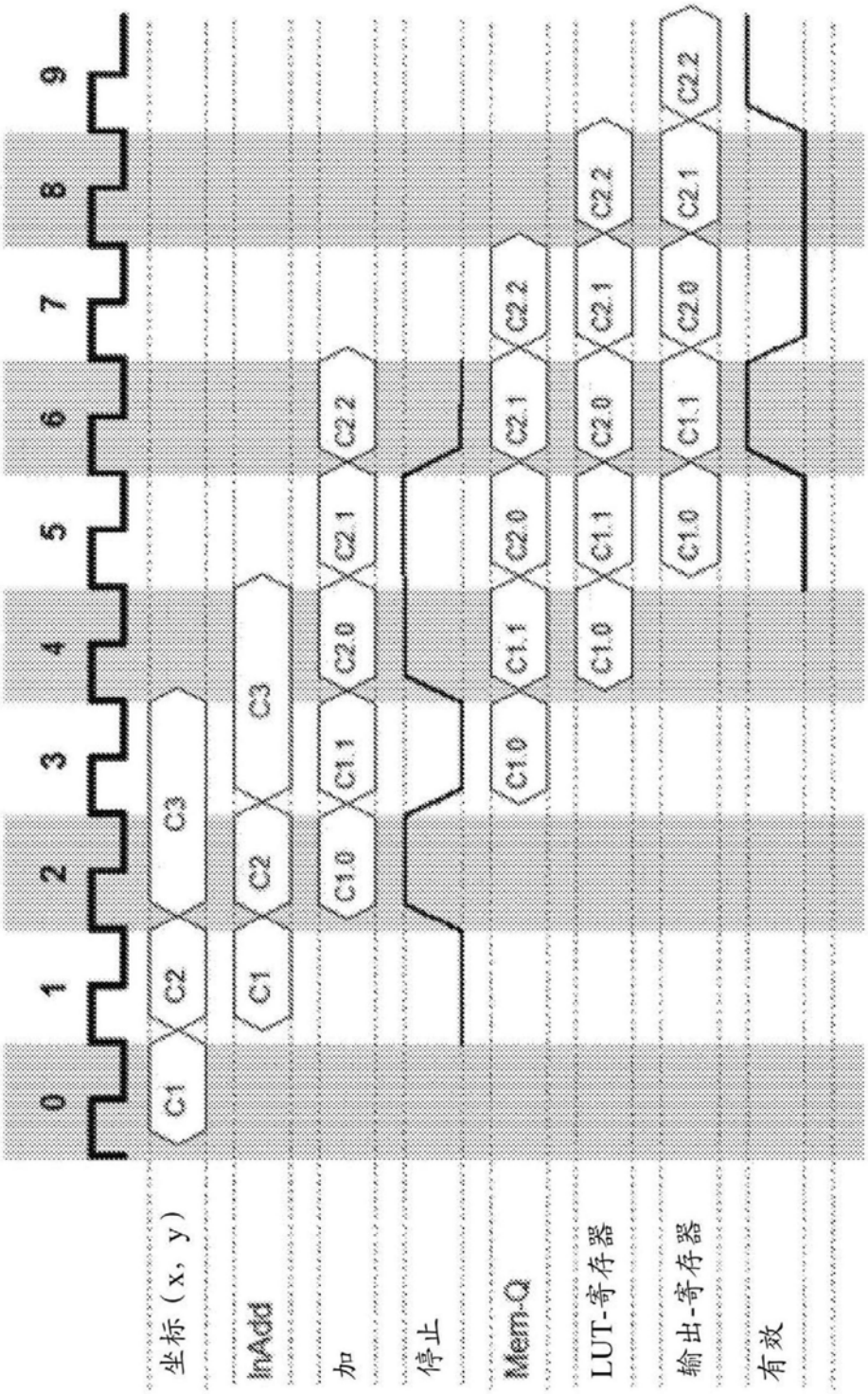

0
1
2
3
4
5
6
7
8
9
坐标（x，y）
C1
C2
C3
InAdd
C1
C2
C3
加
C1.0
C1.1
C2.0
C2.1
C2.2
停止
Mem-Q
C1.0
C1.1
C2.0
C2.1
C2.2
LUT-寄存器
C1.0
C1.1
C2.0
C2.1
C2.2
输出-寄存器
C1.0
C1.1
C2.0
C2.1
C2.2
有效

图13

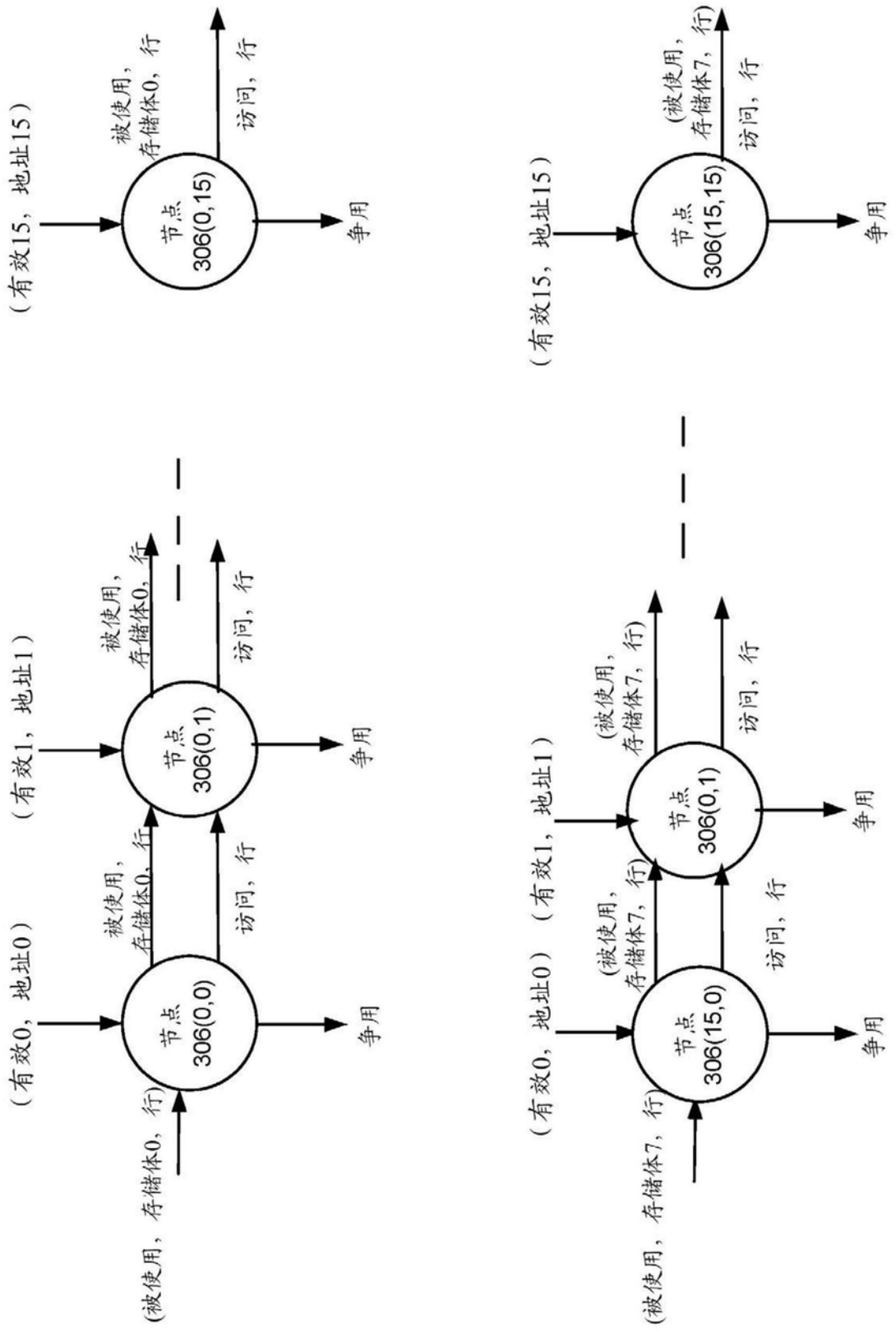

(有效0，地址0)
(有效1，地址1)
(有效15，地址15)
(被使用，存储体0，行)
(被使用，存储体7，行)
节点
306(0,0)
306(0,1)
306(15,0)
306(15,15)
被使用，存储体0，行
(被使用，存储体7，行)
访问，行
争用

图14

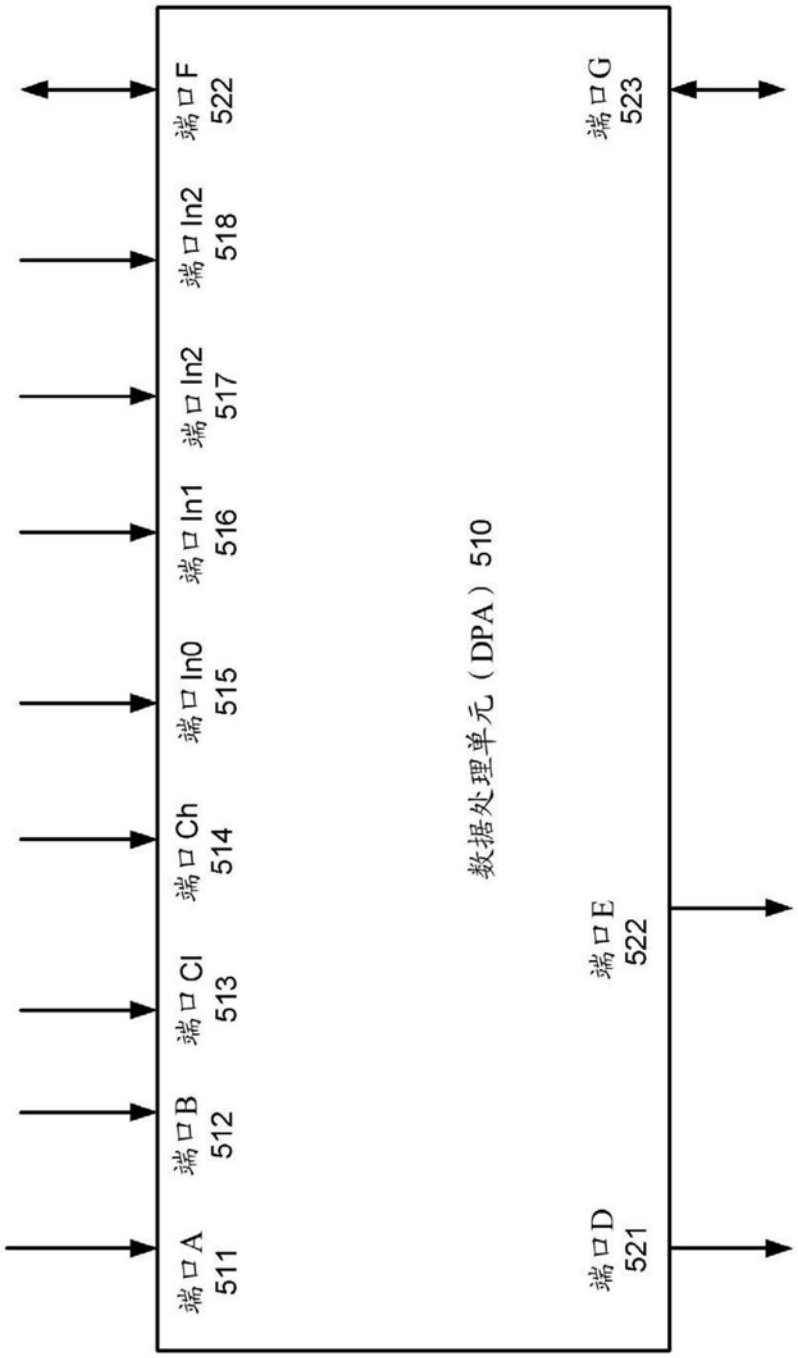

数据处理单元（DPA）510
端口A 511
端口B 512
端口Cl 513
端口Ch 514
端口In0 515
端口In1 516
端口In2 517
端口In2 518
端口F 522
端口G 523
端口D 521
端口E 522

图15

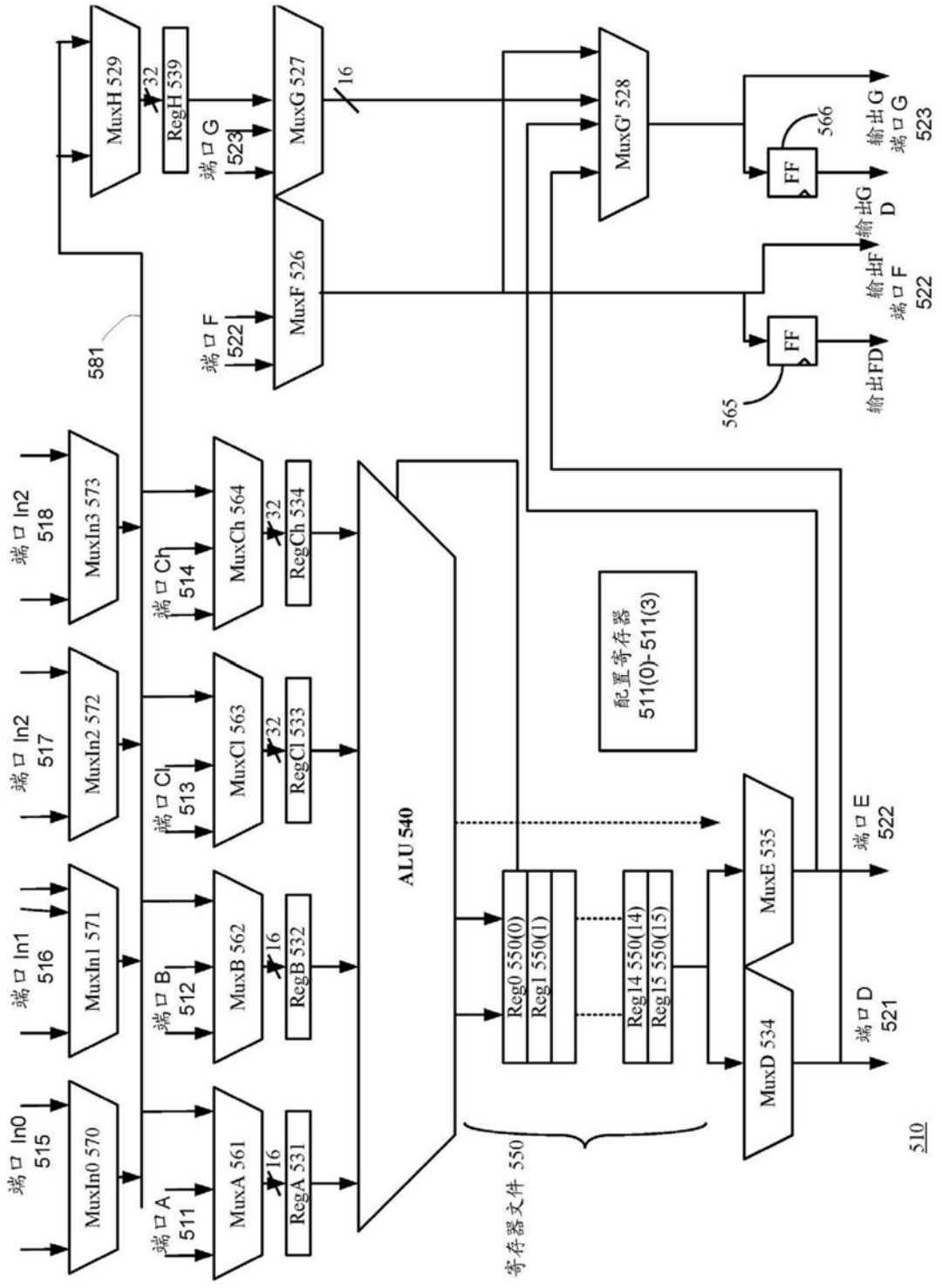

MuxH 529
32
RegH 539
端口 G
523
MuxG 527
16
MuxG' 528
FF
566
输出G
端口 G
523
输出G
D
581
MuxF 526
端口 F
522
输出F
端口 F
522
FF
输出FD
565
端口 In2
518
MuxIn3 573
端口 Ch
514
MuxCh 564
32
RegCh 534
端口 In2
517
MuxIn2 572
端口 Cl
513
MuxCl 563
32
RegCl 533
端口 In1
516
MuxIn1 571
端口 B
512
MuxB 562
16
RegB 532
端口 In0
515
MuxIn0 570
端口 A
511
MuxA 561
16
RegA 531
ALU 540
配置寄存器
511(0)-511(3)
Reg0 550(0)
Reg1 550(1)
Reg14 550(14)
Reg15 550(15)
寄存器文件 550
MuxE 535
MuxD 534
端口 E
522
端口 D
521
510

图16

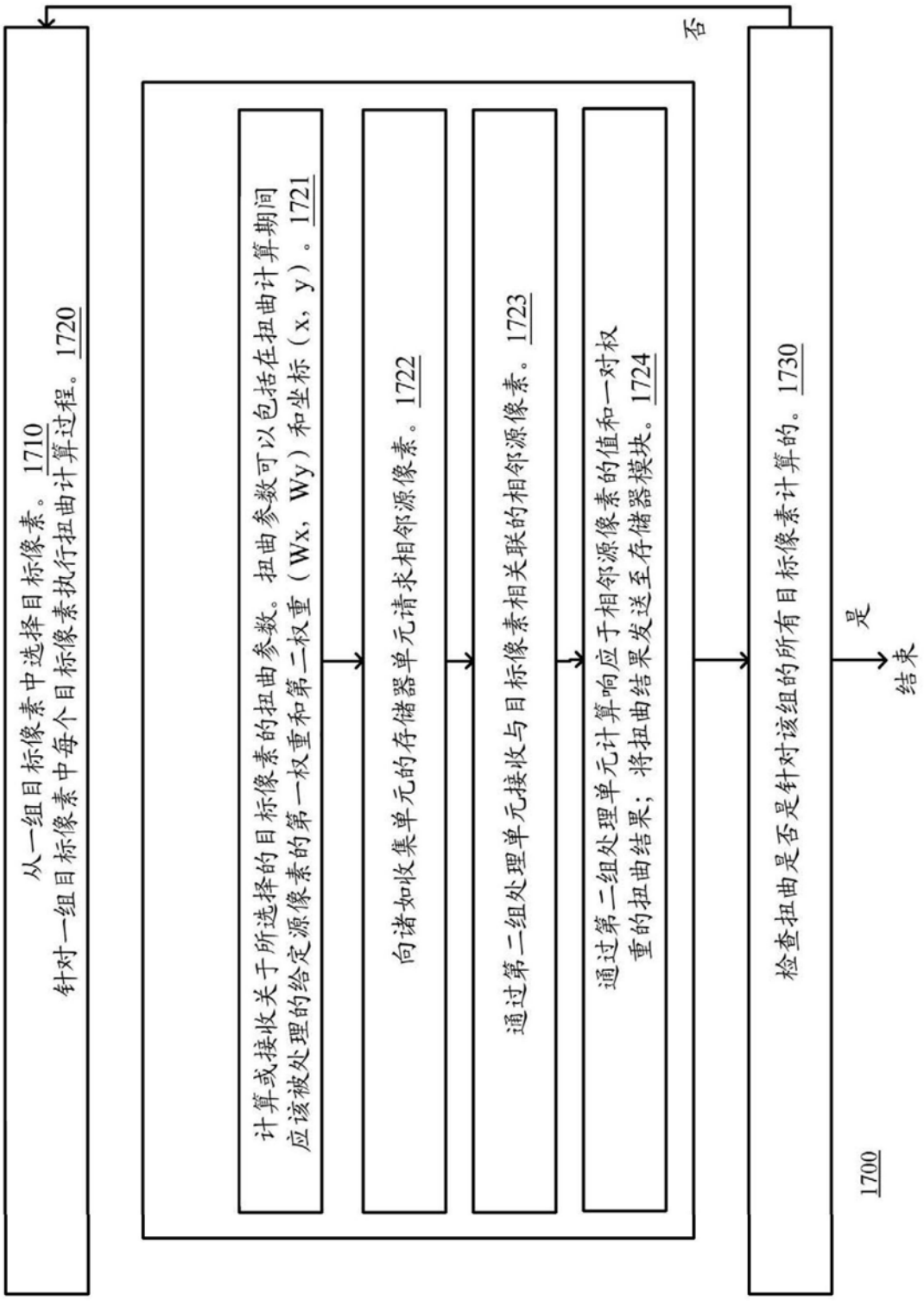

从一组目标像素中选择目标像素。1710
针对一组目标像素中每个目标像素执行扭曲计算过程。1720
计算或接收关于所选择的目标像素的扭曲参数。扭曲参数可以包括在扭曲计算期间应该被处理的给定源像素的第一权重和第二权重（Wx，Wy）和坐标（x，y）。1721
向诸如收集单元的存储器单元请求相邻源像素。1722
通过第二组处理单元接收与目标像素相关联的相邻源像素。1723
通过第二组处理单元计算响应于相邻源像素的值和一对权重的扭曲结果；将扭曲结果发送至存储器模块。1724
检查扭曲是否是针对该组的所有目标像素计算的。1730
否
是
结束
1700

图17

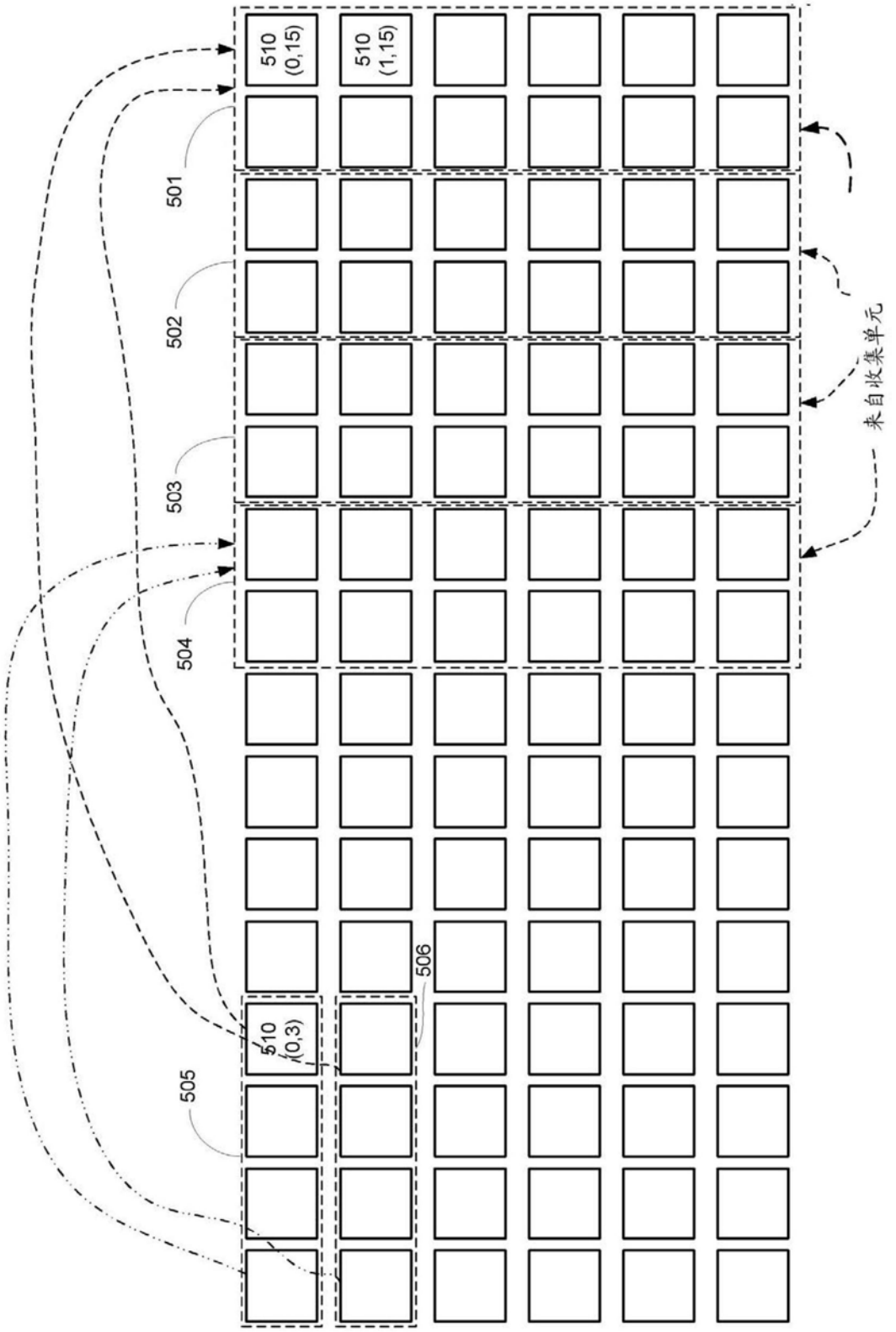


图18

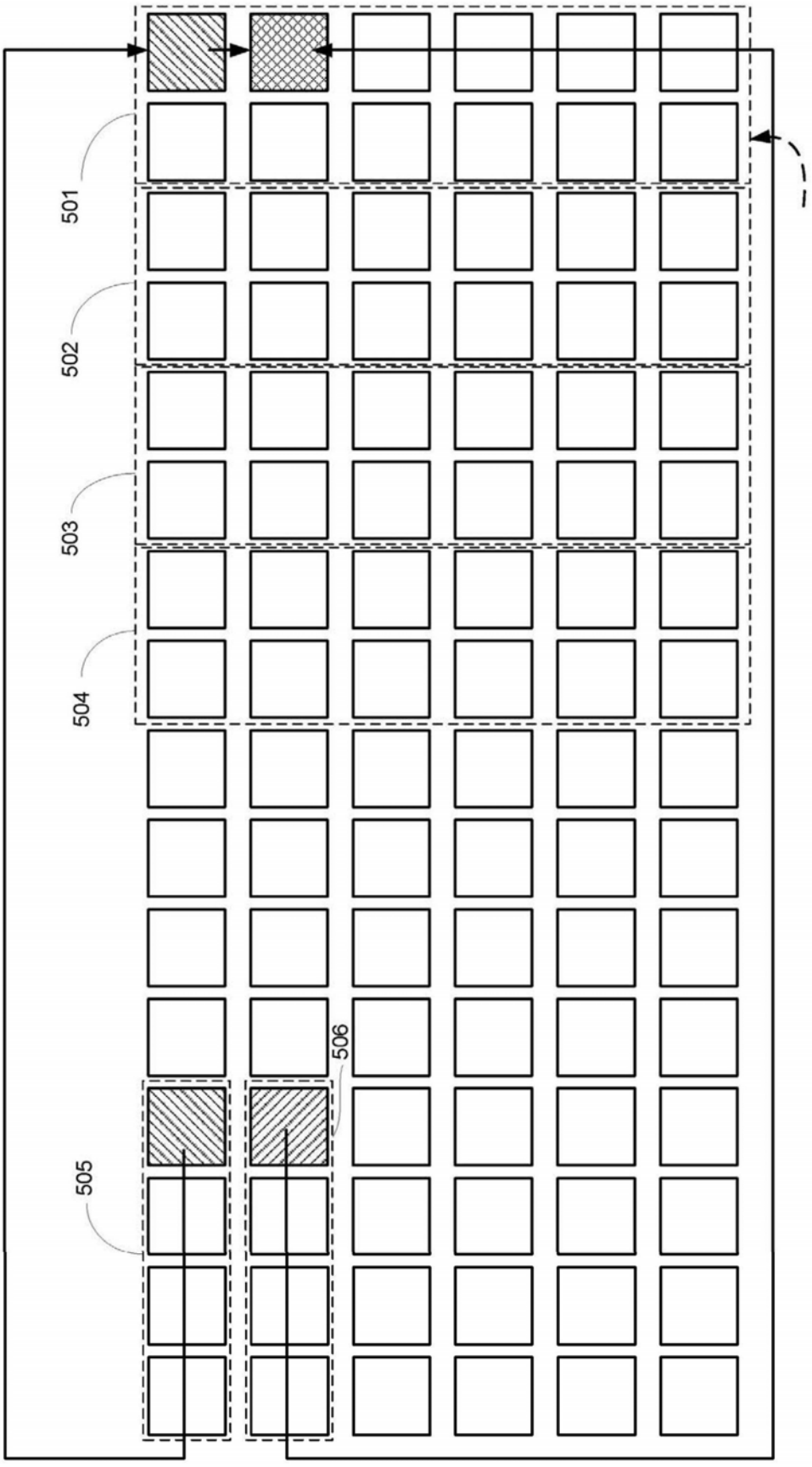


图19

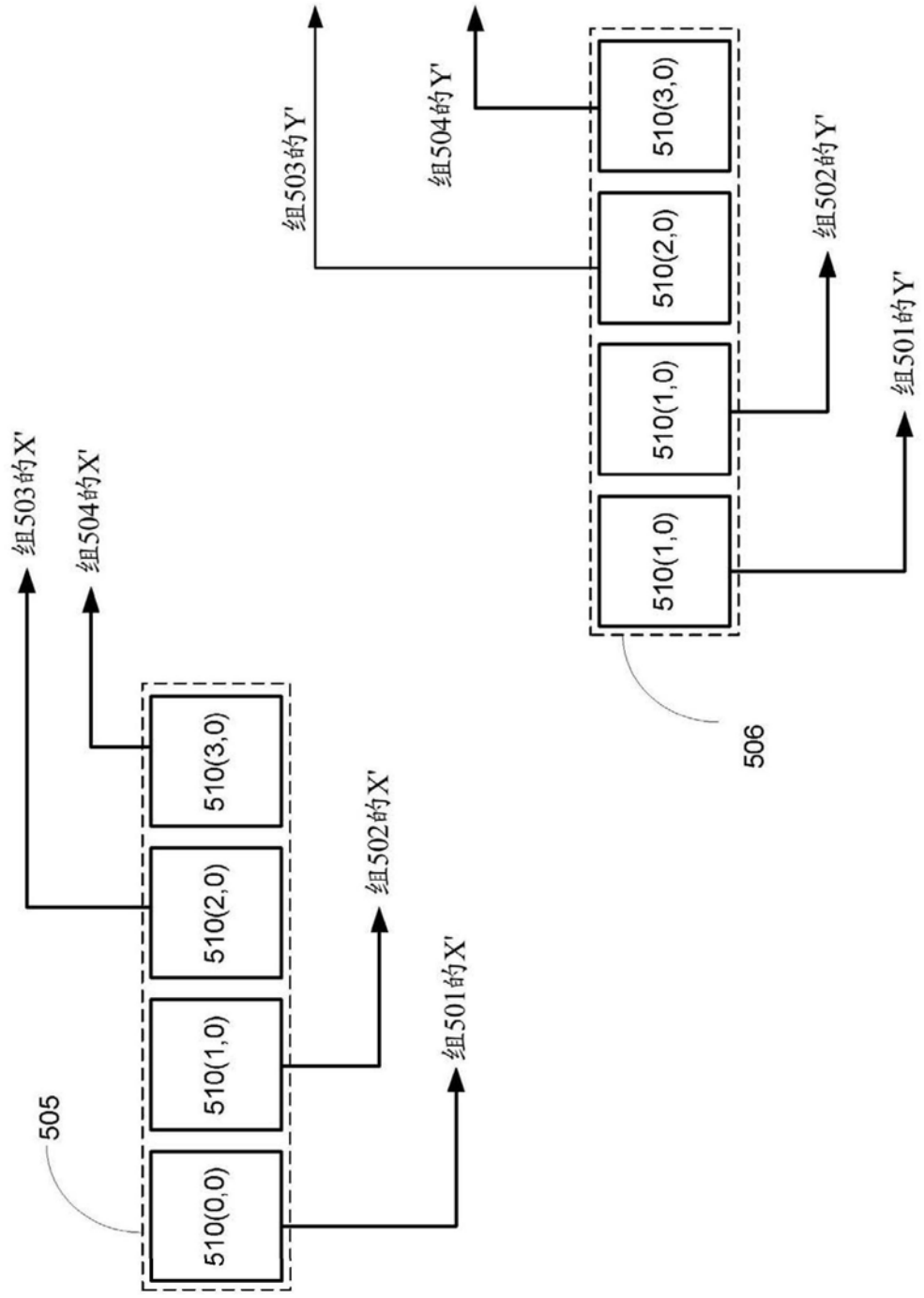

组503的Y'
组504的Y'
510(3,0)
510(2,0)
组502的Y'
510(1,0)
组501的Y'
510(1,0)
506
组503的X'
组504的X'
510(3,0)
510(2,0)
组502的X'
510(1,0)
组501的X'
505
510(0,0)

图20

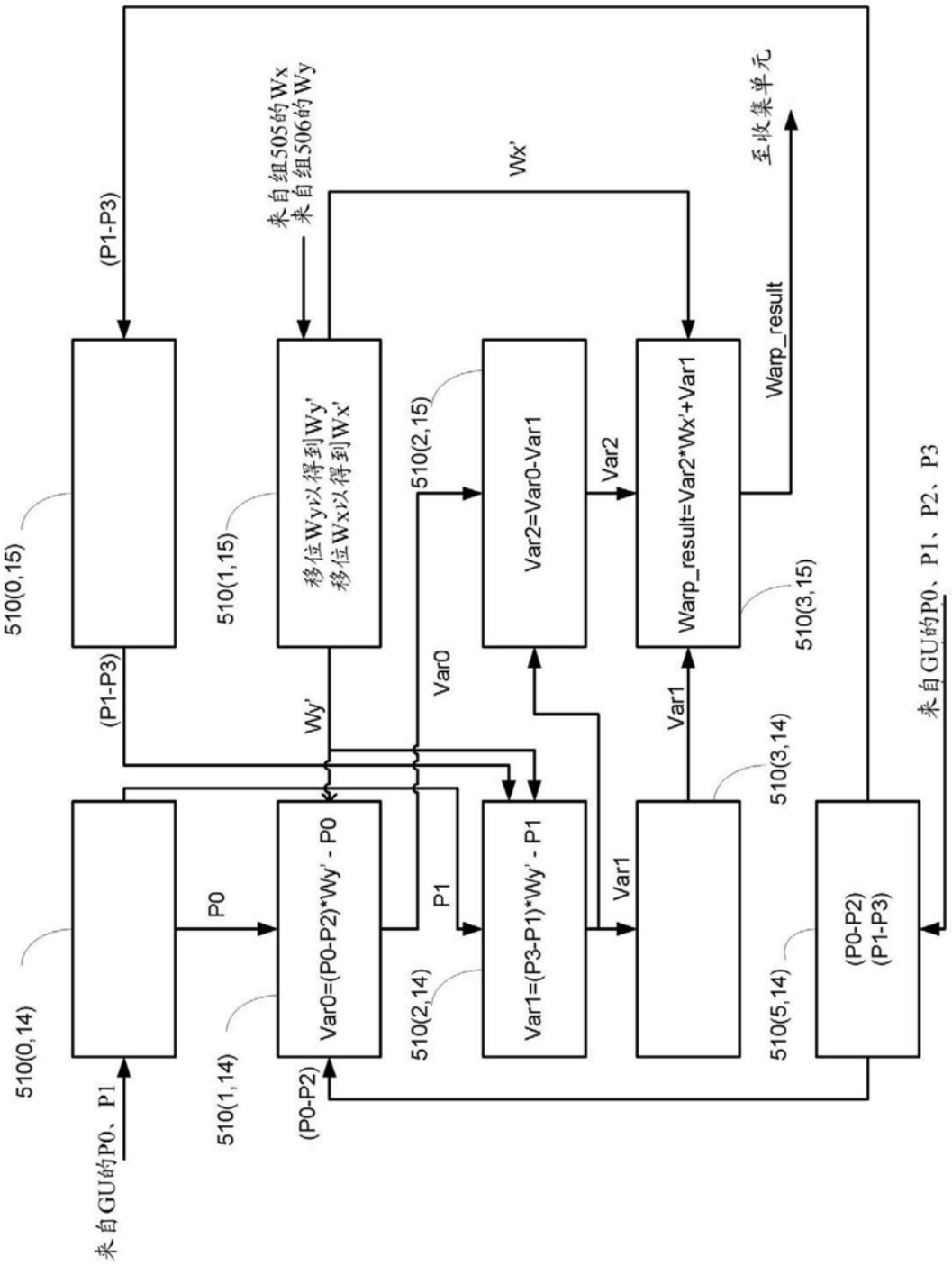

来自GU的P0、P1
来自组505的Wx
来自组506的Wy
至收集单元
(P1-P3)
(P1-P3)
(P0-P2)
(P0-P2)
(P1-P3)
来自GU的P0、P1、P2、P3
Wx'
Wy'
Var0
Var1
Var2
P0
P1
Warp_result
510(0,15)
510(1,15)
510(2,15)
510(3,15)
510(0,14)
510(1,14)
510(2,14)
510(3,14)
510(5,14)
移位Wy以得到Wy'
移位Wx以得到Wx'
Var2=Var0-Var1
Warp_result=Var2*Wx'+Var1
Var0=(P0-P2)*Wy' - P0
Var1=(P3-P1)*Wy' - P1

图21

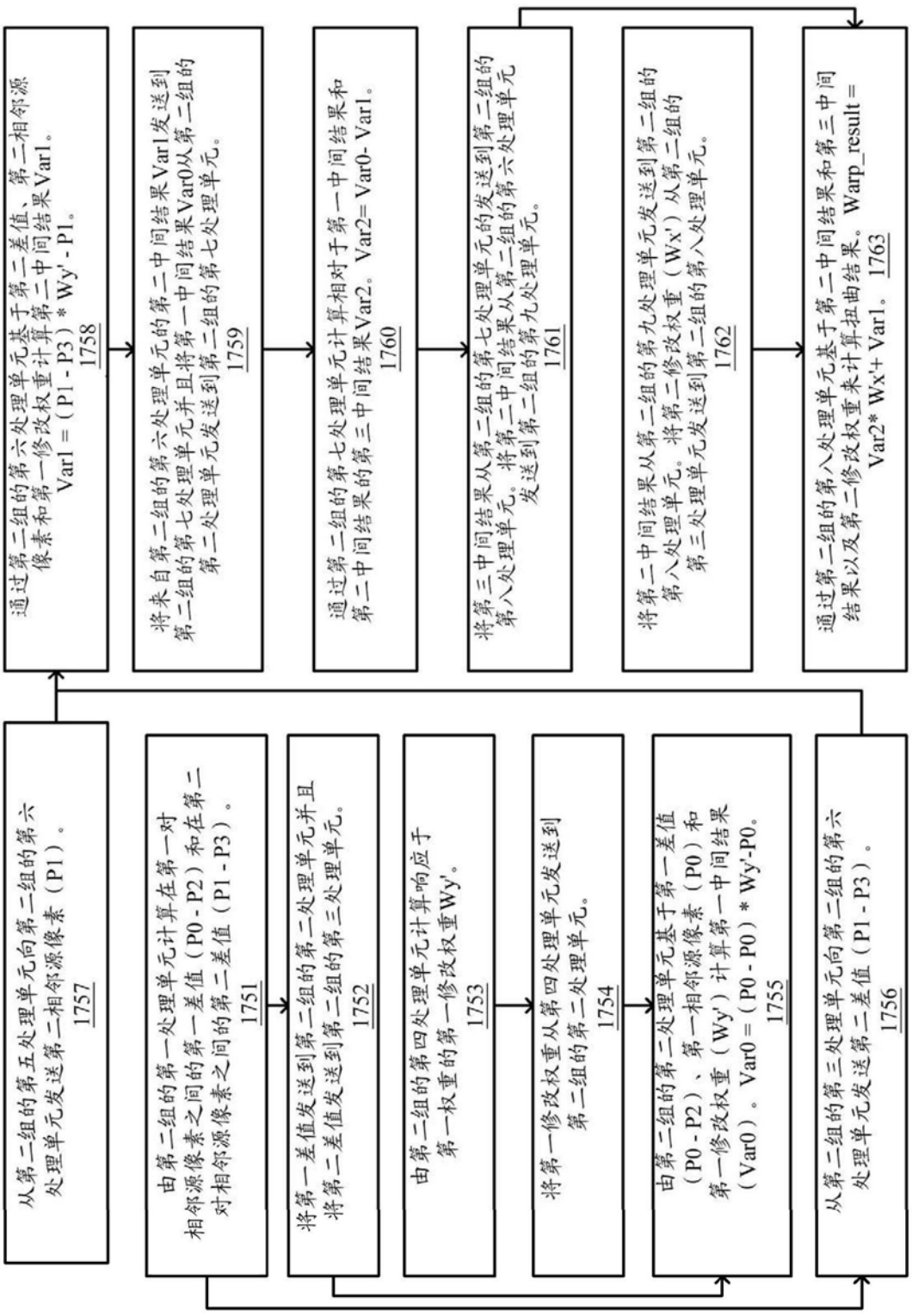

从第二组的第五处理单元向第二组的第六处理单元发送第二相邻源像素（P1）。
1757
由第二组的第一处理单元计算在第一对相邻源像素之间的第一差值（P0 - P2）和在第二对相邻源像素之间的第二差值（P1 - P3）。
1751
将第一差值发送到第二组的第二处理单元并且将第二差值发送到第二组的第三处理单元。
1752
由第二组的第四处理单元计算响应于第一权重的第一修改权重Wy'。
1753
将第一修改权重从第四处理单元发送到第二组的第二处理单元。
1754
由第二组的第二处理单元基于第一差值（P0 - P2）、第一相邻源像素（P0）和第一修改权重（Wy'）计算第一中间结果（Var0）。Var0 =（P0 - P0）* Wy'-P0。
1755
从第二组的第三处理单元向第二组的第六处理单元发送第二差值（P1 - P3）。
1756
通过第二组的第六处理单元基于第二差值、第二相邻源像素和第一修改权重计算第二中间结果Var1。
Var1 =（P1 - P3）* Wy' - P1。
1758
将来自第二组的第六处理单元的第二中间结果Var1发送到第二组的第七处理单元并且将第一中间结果Var0从第二组的第二处理单元发送到第二组的第七处理单元。
1759
通过第二组的第七处理单元计算相对于第一中间结果和第二中间结果的第三中间结果Var2。Var2= Var0- Var1。
1760
将第三中间结果从第二组的第七处理单元的发送到第二组的第八处理单元。将第二中间结果从第二组的第六处理单元发送到第二组的第九处理单元。
1761
将第二中间结果从第二组的第九处理单元发送到第二组的第八处理单元。将第二修改权重（Wx'）从第二组的第三处理单元发送到第二组的第八处理单元。
1762
通过第二组的第八处理单元基于第二中间结果和第三中间结果以及第二修改权重来计算扭曲结果。Warp_result = Var2* Wx'+ Var1。
1763

图22

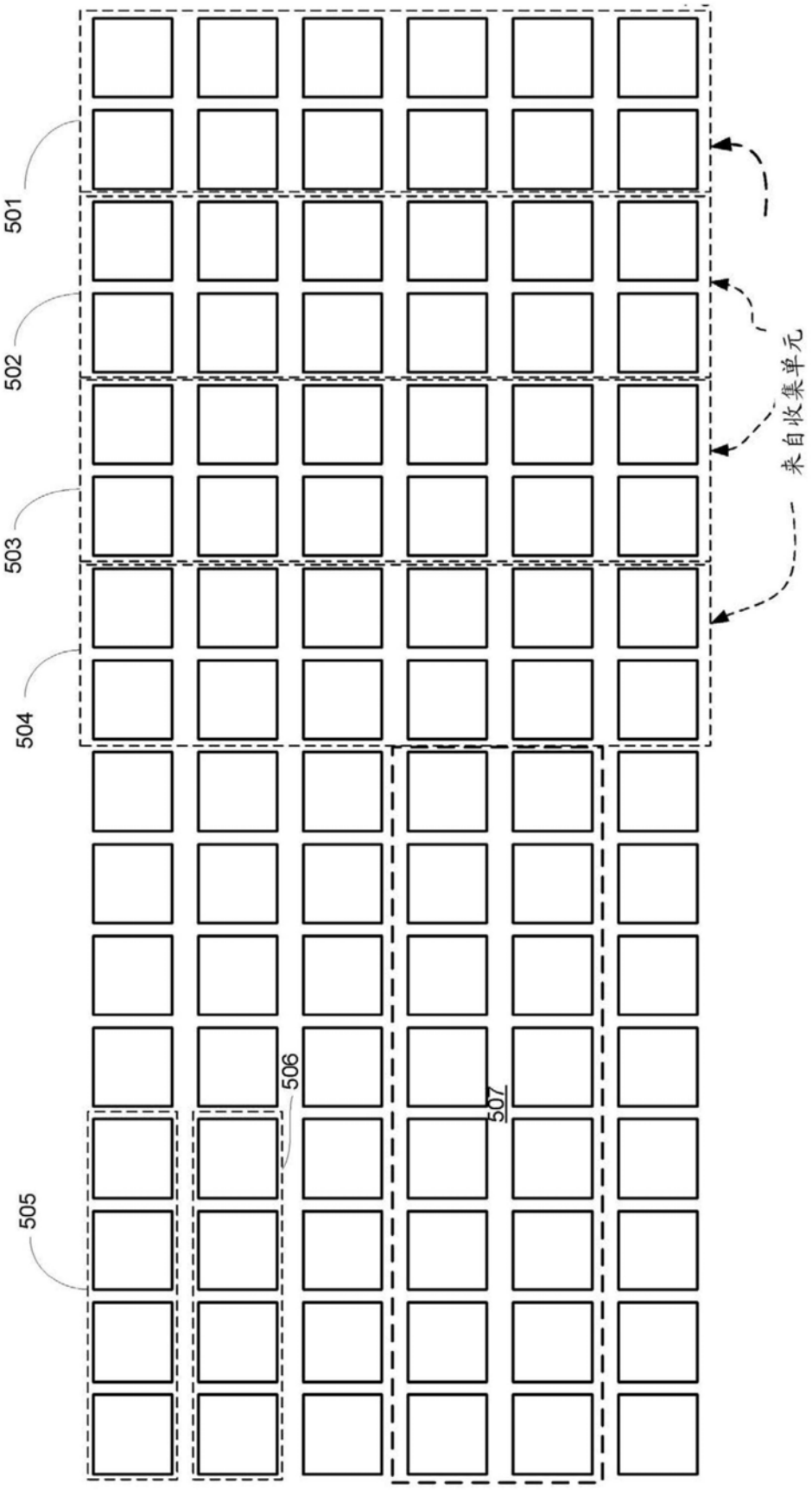


图23

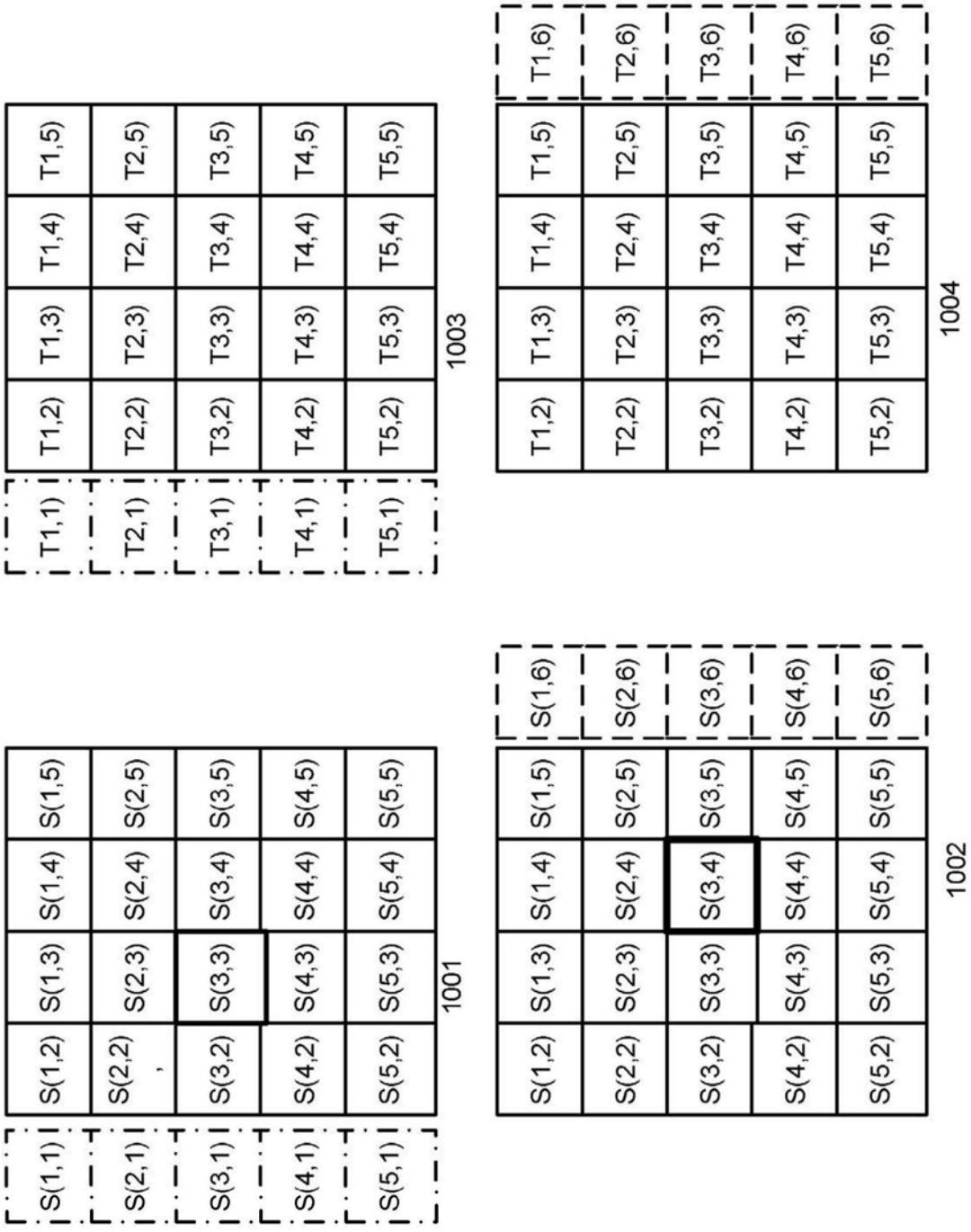

S(1,1) S(1,2) S(1,3) S(1,4) S(1,5)
S(2,1) S(2,2) S(2,3) S(2,4) S(2,5)
S(3,1) S(3,2) S(3,3) S(3,4) S(3,5)
S(4,1) S(4,2) S(4,3) S(4,4) S(4,5)
S(5,1) S(5,2) S(5,3) S(5,4) S(5,5)
1001
S(1,6) S(1,5) S(1,4) S(1,3) S(1,2)
S(2,6) S(2,5) S(2,4) S(2,3) S(2,2)
S(3,6) S(3,5) S(3,4) S(3,3) S(3,2)
S(4,6) S(4,5) S(4,4) S(4,3) S(4,2)
S(5,6) S(5,5) S(5,4) S(5,3) S(5,2)
1002
T1,1) T1,2) T1,3) T1,4) T1,5)
T2,1) T2,2) T2,3) T2,4) T2,5)
T3,1) T3,2) T3,3) T3,4) T3,5)
T4,1) T4,2) T4,3) T4,4) T4,5)
T5,1) T5,2) T5,3) T5,4) T5,5)
1003
T1,6) T1,5) T1,4) T1,3) T1,2)
T2,6) T2,5) T2,4) T2,3) T2,2)
T3,6) T3,5) T3,4) T3,3) T3,2)
T4,6) T4,5) T4,4) T4,3) T4,2)
T5,6) T5,5) T5,4) T5,3) T5,2)
1004

图24

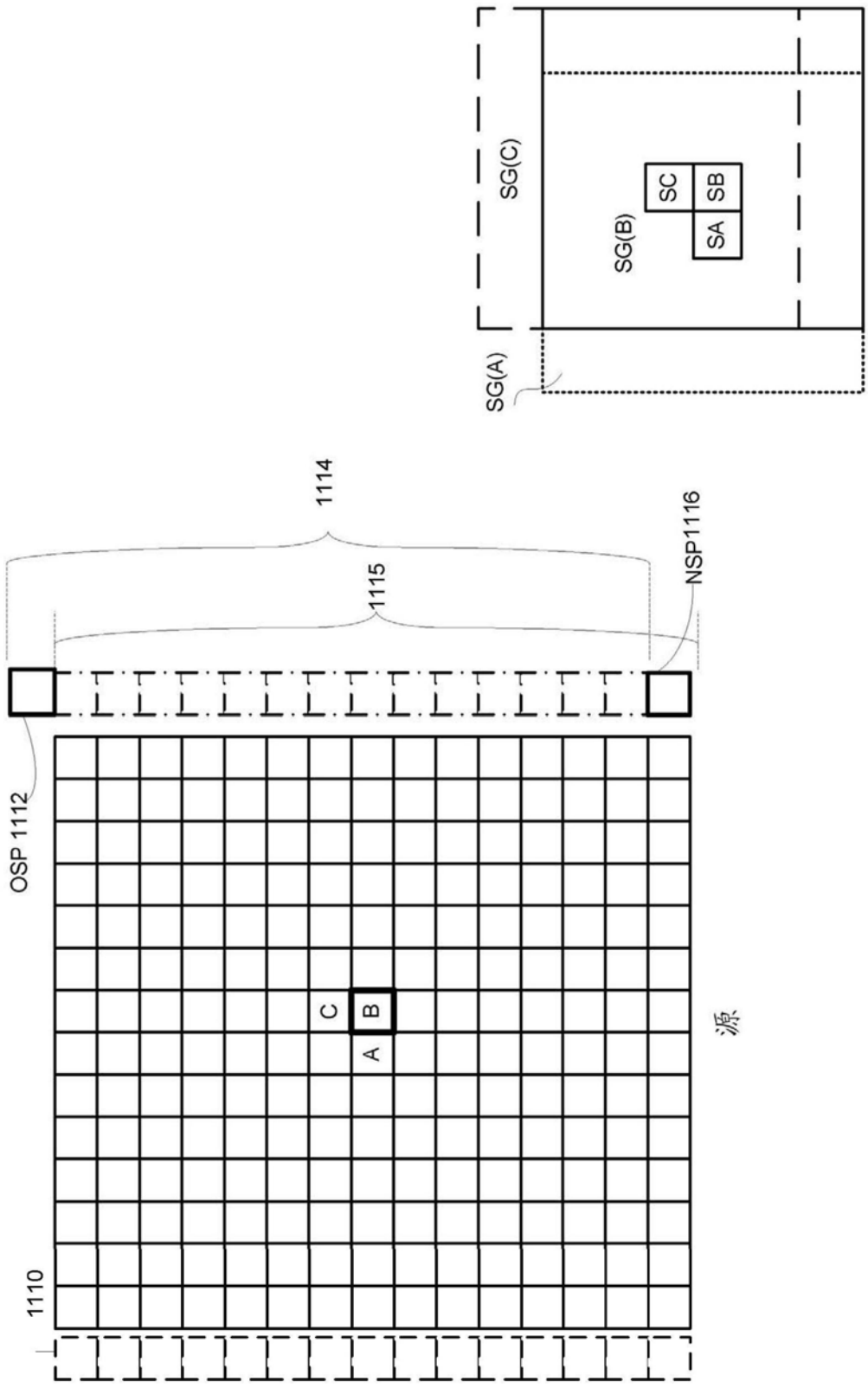


图25

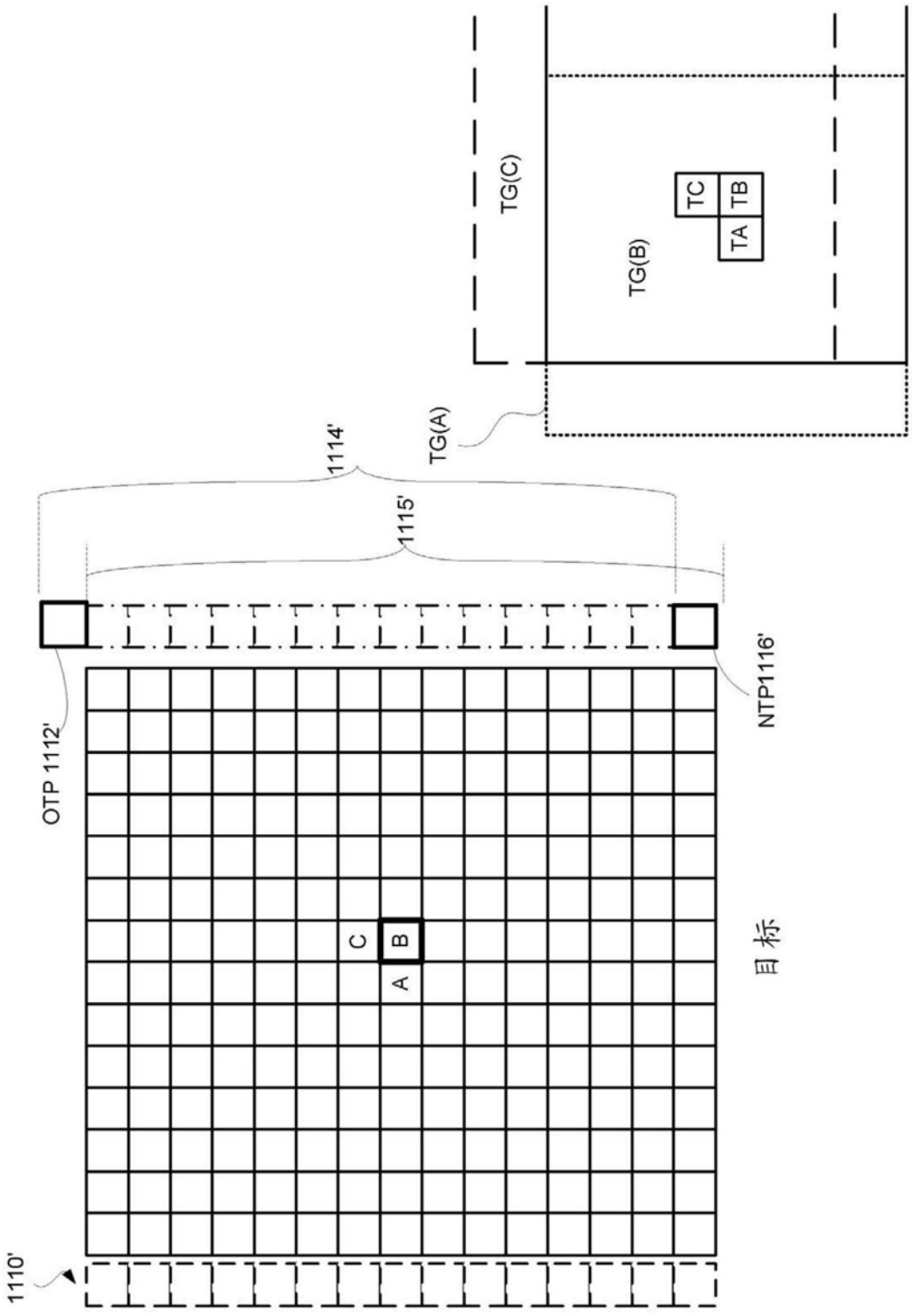


图26

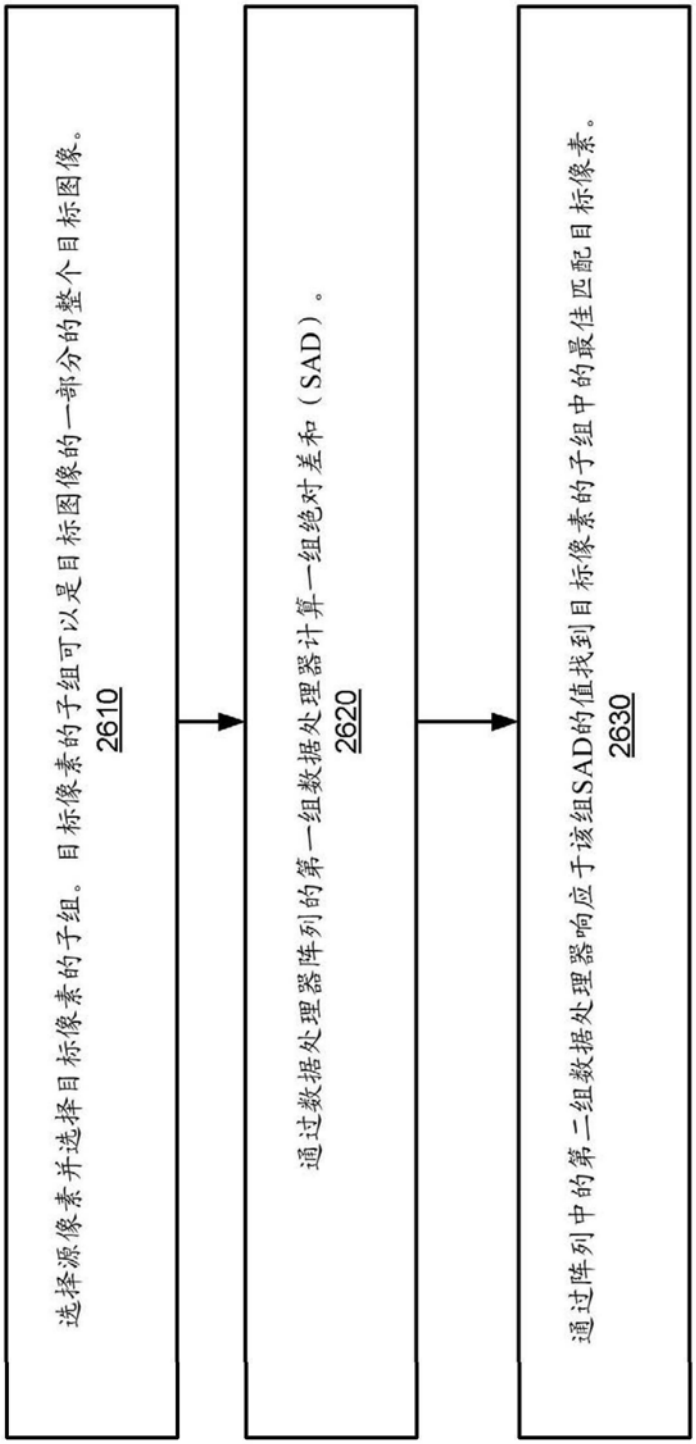

选择源像素并选择目标像素的子组。目标像素的子组可以是目标图像的一部分的整个目标图像。
2610
通过数据处理器阵列的第一组数据处理器计算一组绝对差和（SAD）。
2620
通过阵列中的第二组数据处理器响应于该组SAD的值找到目标像素的子组中的最佳匹配目标像素。
2630
2600

图27

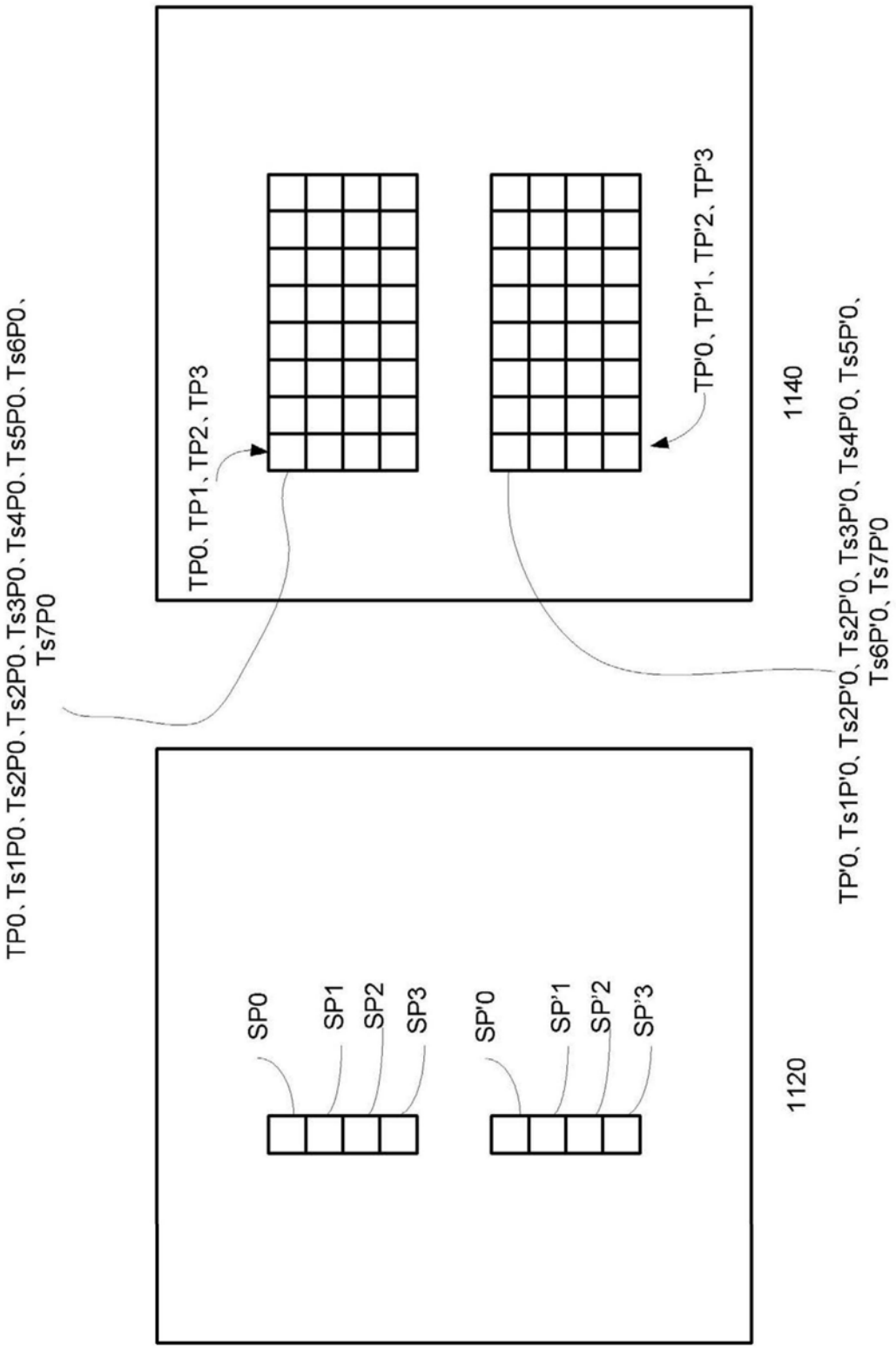

TP0、TP1、TP2、TP3
TP'0、TP'1、TP'2、TP'3
1140
TP0、Ts1P0、Ts2P0、Ts2P0、Ts3P0、Ts4P0、Ts5P0、Ts6P0、Ts7P0
TP'0、Ts1P'0、Ts2P'0、Ts2P'0、Ts3P'0、Ts4P'0、Ts5P'0、Ts6P'0、Ts7P'0
SP0
SP1
SP2
SP3
SP'0
SP'1
SP'2
SP'3
1120

图28

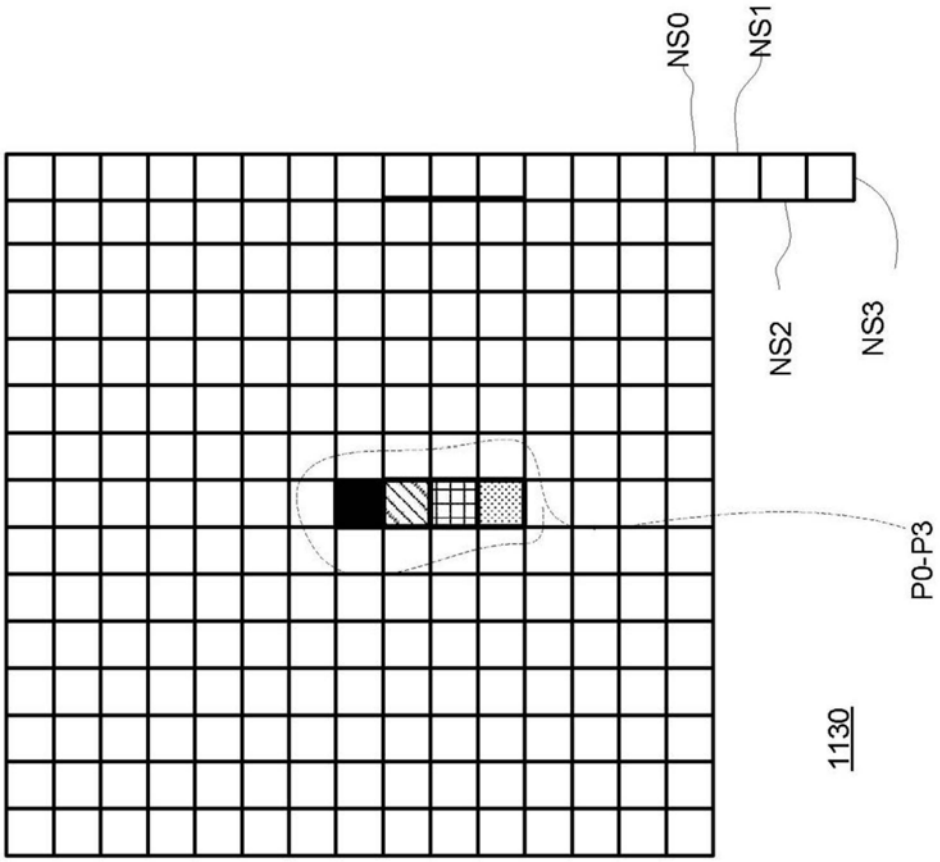


图29

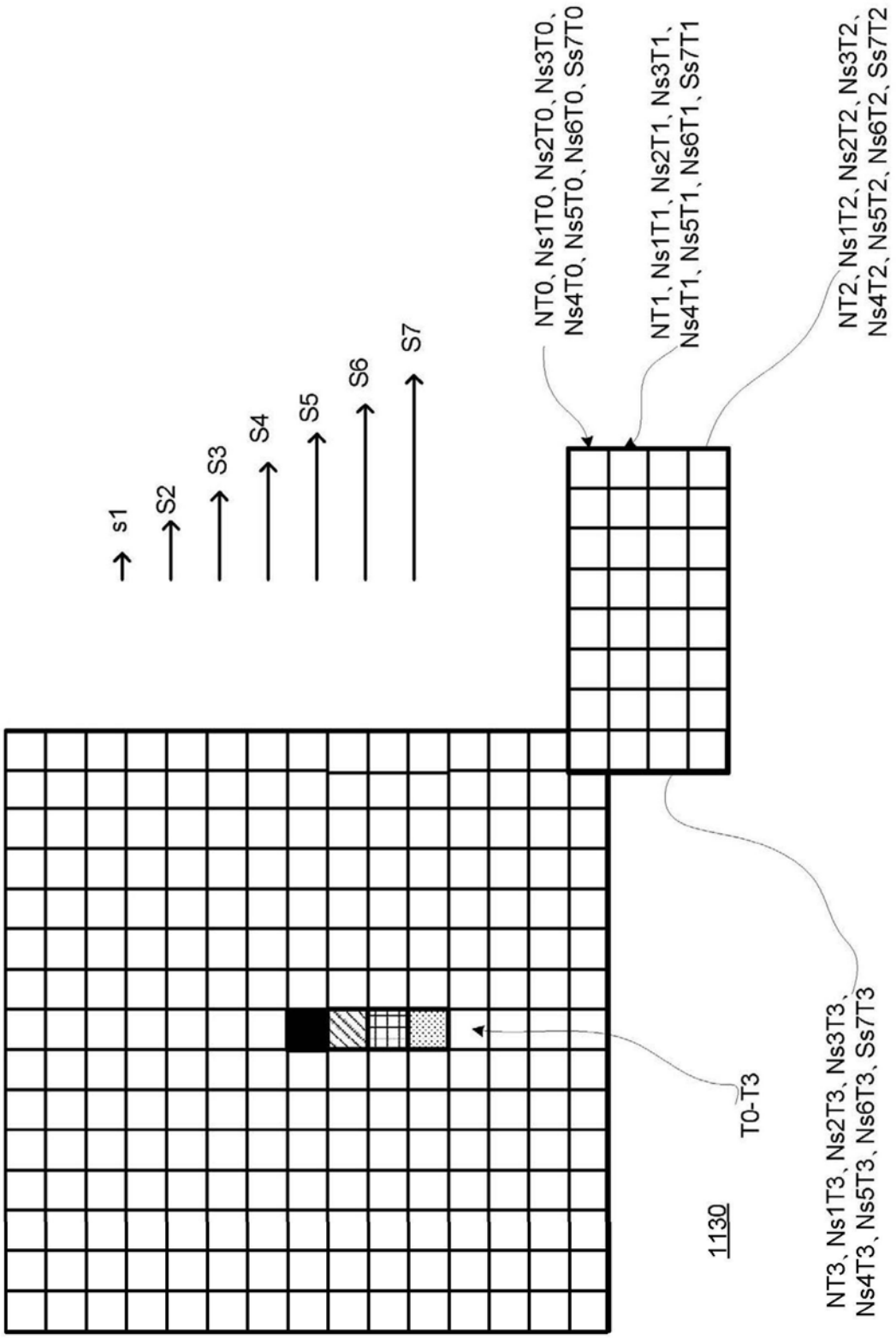

s1
S2
S3
S4
S5
S6
S7
NT0、Ns1T0、Ns2T0、Ns3T0、
Ns4T0、Ns5T0、Ns6T0、Ss7T0
NT1、Ns1T1、Ns2T1、Ns3T1、
Ns4T1、Ns5T1、Ns6T1、Ss7T1
NT2、Ns1T2、Ns2T2、Ns3T2、
Ns4T2、Ns5T2、Ns6T2、Ss7T2
T0-T3
1130
NT3、Ns1T3、Ns2T3、Ns3T3、
Ns4T3、Ns5T3、Ns6T3、Ss7T3

图30

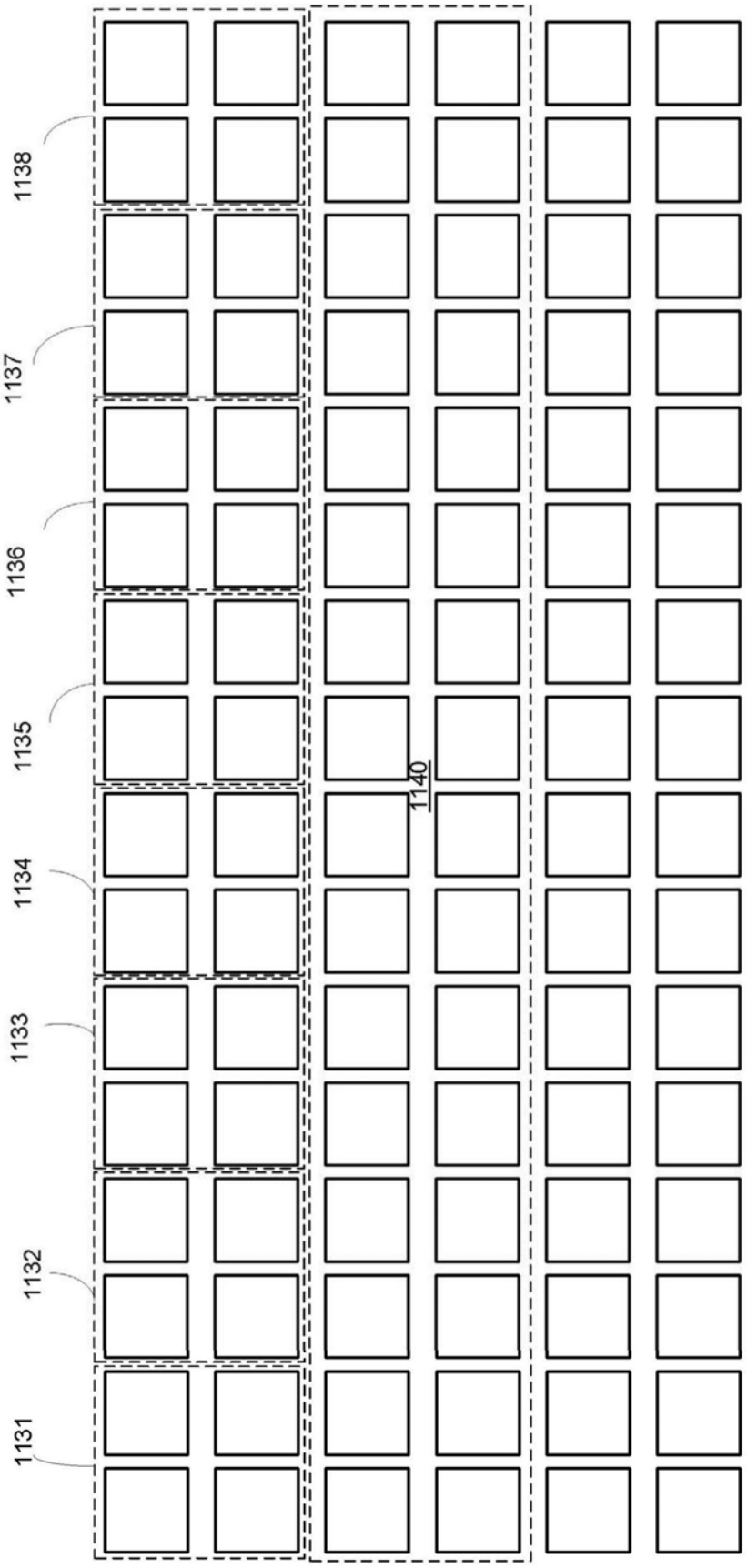


图31

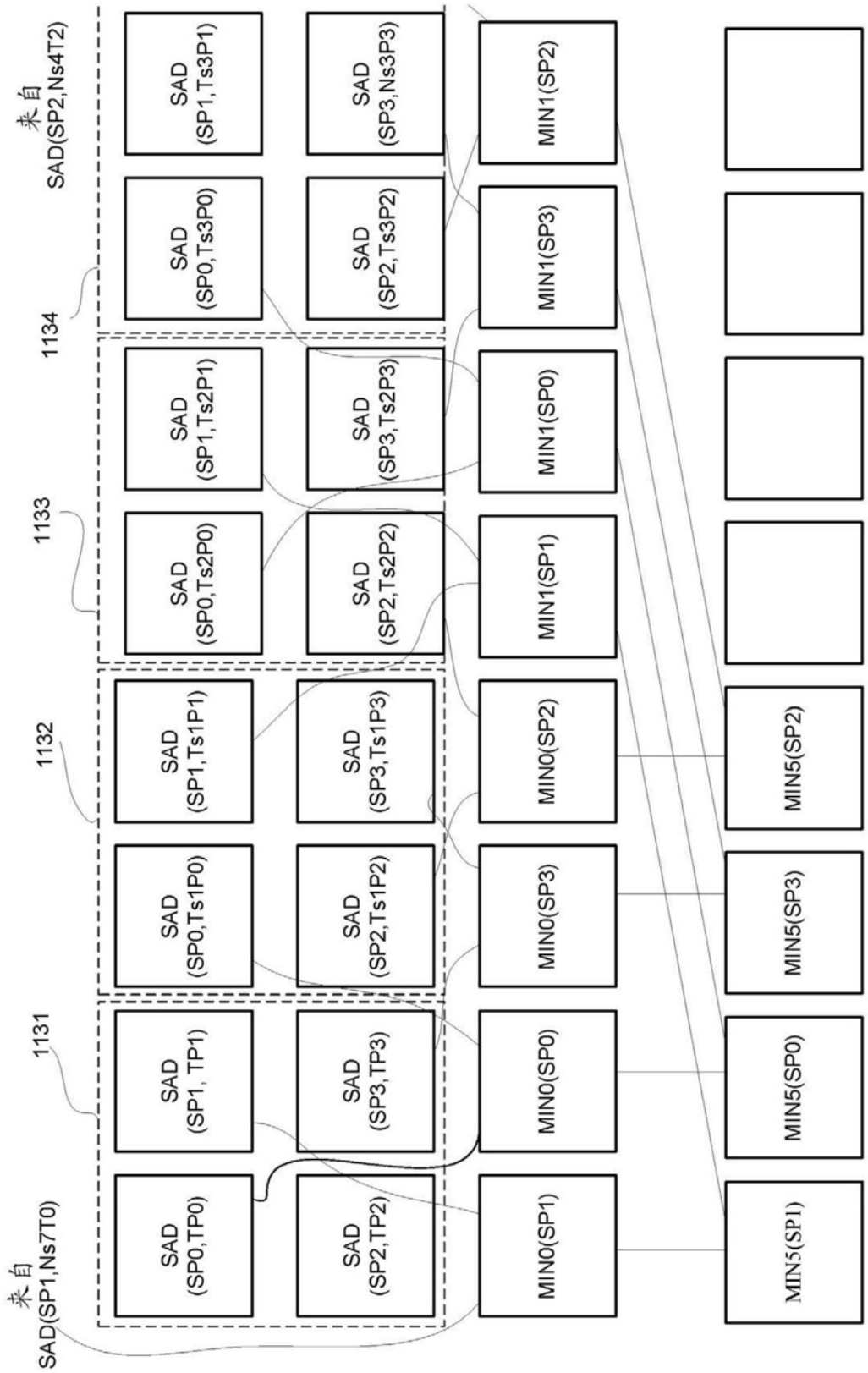

来自
SAD(SP2,Ns4T2)
1134
1133
1132
1131
来自
SAD(SP1,Ns7T0)
SAD (SP1,Ts3P1)
SAD (SP3,Ns3P3)
SAD (SP0,Ts3P0)
SAD (SP2,Ts3P2)
SAD (SP1,Ts2P1)
SAD (SP3,Ts2P3)
SAD (SP0,Ts2P0)
SAD (SP2,Ts2P2)
SAD (SP1,Ts1P1)
SAD (SP3,Ts1P3)
SAD (SP0,Ts1P0)
SAD (SP2,Ts1P2)
SAD (SP1, TP1)
SAD (SP3,TP3)
SAD (SP0,TP0)
SAD (SP2,TP2)
MIN1(SP2)
MIN1(SP3)
MIN1(SP0)
MIN1(SP1)
MIN0(SP2)
MIN0(SP3)
MIN0(SP0)
MIN0(SP1)
MIN5(SP2)
MIN5(SP3)
MIN5(SP0)
MIN5(SP1)

图32

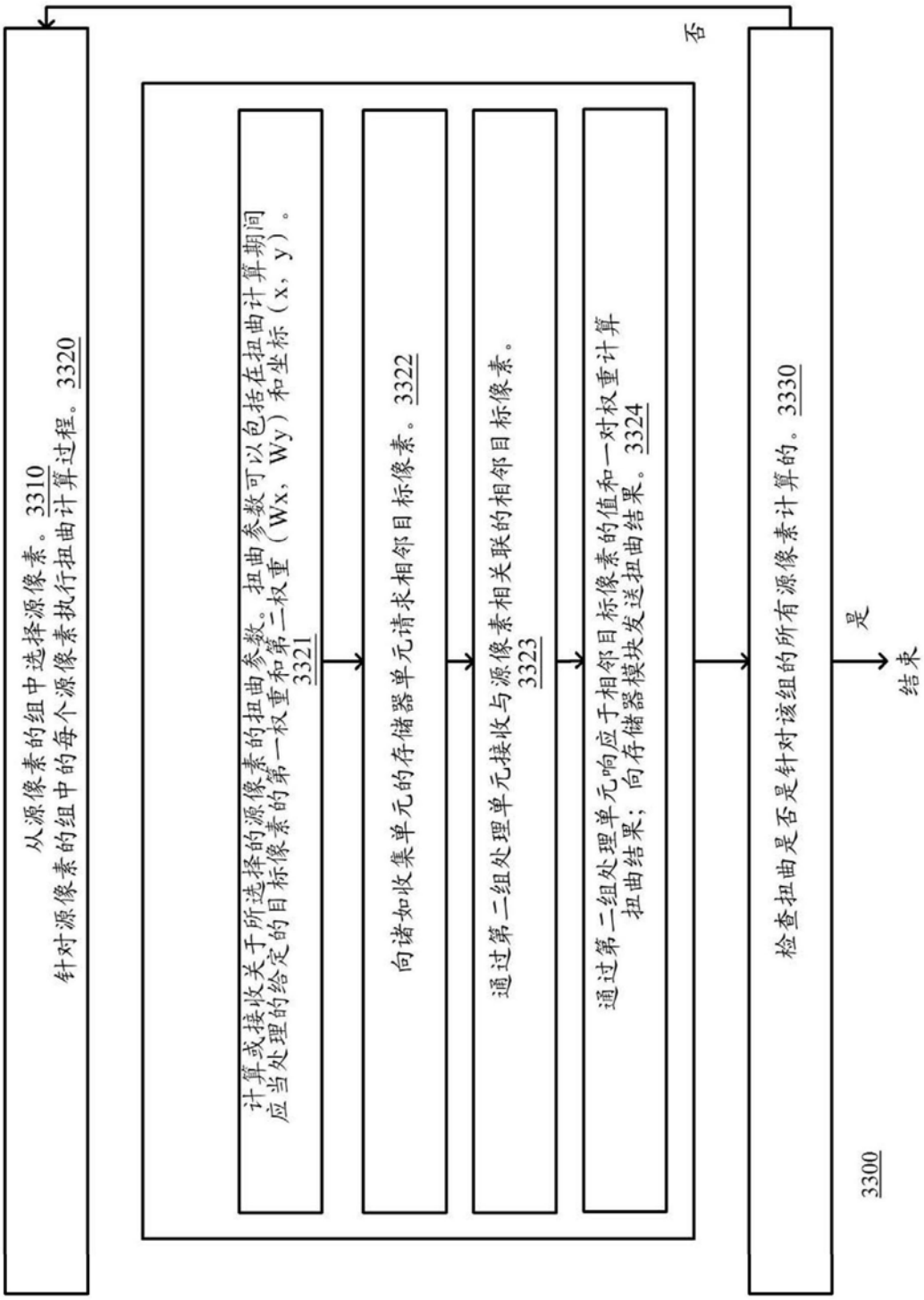

从源像素的组中选择源像素。3310
针对源像素的组中的每个源像素执行扭曲计算过程。3320
计算或接收关于所选择的源像素的扭曲参数。扭曲参数可以包括在扭曲计算期间应当处理的给定的目标像素的第一权重和第二权重（Wx，Wy）和坐标（x，y）。3321
向诸如收集单元的存储器单元请求相邻目标像素。3322
通过第二组处理单元接收与源像素相关联的相邻目标像素。3323
通过第二组处理单元响应于相邻目标像素的值和一对权重计算扭曲结果；向存储器模块发送扭曲结果。3324
检查扭曲是否是针对该组的所有源像素计算的。3330
否
是
结束
3300

图33