[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06K 9/00 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200580034470.9

[43] 公开日 2009 年 5 月 13 日

[11] 公开号 CN 101432760A

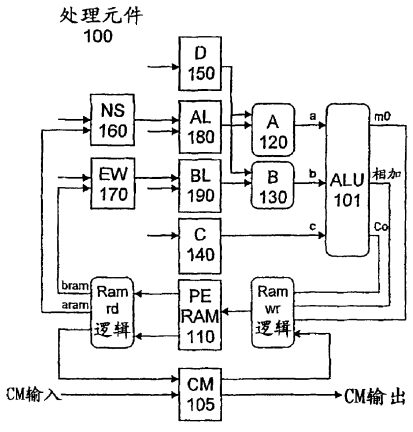
[22] 申请日 2005.8.8
[21] 申请号 200580034470.9
[30] 优先权
[32] 2004. 8.31 [33] US [31] 60/605,912
[86] 国际申请 PCT/US2005/028273 2005.8.8
[87] 国际公布 WO2006/026086 英 2006.3.9
[85] 进入国家阶段日期 2007.4.9
[71] 申请人 硅奥普迪思公司
地址 美国加利福尼亚
[72] 发明人 伍德罗·L·梅克
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 李 勇
权利要求书 4 页 说明书 17 页 附图 6 页

[54] 发明名称

管理位平面资源的方法和装置

[57] 摘要

管理位平面资源的系统和方法。 因为 SIMD 结构内的处理元件(PE)存储器的缺乏，确保其使用效率最优是非常重要的。 本发明公开了一种实时管理 PE 存储器的方法，从而能够获得接近 100％的使用效率。 所作用的方法允许在单个位平面增量内分配和重新分配存储器。 实时分配和重新分配连续发生，因此不能够保证 PE 存储器的邻接模块的可用性。 本发明所采用的方法可以使用分散的位平面，而不必花费执行时间来执行“无用集合”。



1、一种数字处理系统，包括:

a、一个处理元件阵列，具有可寻址数据存储装置，适于处理位平面数据;

b、一个包含存储物理地址的存储装置的分配表;

c、用于传递包含一个虚拟地址属性和一个尺寸属性的图像描述符的装置，所述虚拟地址和尺寸指定了分配表的位置范围，其中存储了包含一个图像的位平面的物理地址。

2、如权利要求 1 所述的系统，其中分配表中的存储位置数量超过包含有处理阵列的位平面数量。

3、如权利要求 1 所述的系统，其中所述图像描述符还包括一个符号属性，以表示该图像是否被作为有符号的数据被处理。

4、如权利要求 1 所述的系统，还包括一个用于控制处理器阵列内的位平面处理的序列发生器，所述序列发生器响应于至少一个图像描述符产生至少一个微指令序列，所述微指令包含响应于分配表中存储的物理地址而生成的至少一个位平面地址。

5、如权利要求 1 所述的系统，其中分配表被配置为提供多重同步访问。

6、如权利要求 1 所述的系统,其中所述阵列被分割为多个区段，每个区段包含多个位平面，并且其中分配表被分割为相应的区段，每个分配表区段包括用于存储属于一个阵列区段的物理地址的存储装置。

7、如权利要求 1 所述的系统，还提供了用于向一个图像分配位平面的第一分配装置，所述第一分配装置包括表示可用位平面的物理地址库，其中所述分配包括从该库中读取物理地址以及将该物理地址写入分配表。

8、如权利要求 7 所述的系统,其中所述库是一个 FIFO 存储器。

9、如权利要求 7 所述的系统,其中处理阵列被分割为多个区段，

每个区段包含多个位平面，并且其中该库被分割为相应的区段，每个库区段包括用于存储一个阵列区段的物理地址的存储装置。

10、 如权利要求 7 所述的系统，还包括用于向一个未增加的图像分配位平面的第二分配装置，所述图像没有被分配位平面，并且所述第二分配装置响应于尺寸属性为 0 的图像描述符并且进一步响应于被称为“新尺寸”的尺寸控制信号进行分配，所述新尺寸决定了要分配的位平面数量。

11、 如权利要求 7 所述的系统，还包括用于向一个部分增加的图像分配位平面的第三分配装置，所述图像具有分配给它的多个位平面，并且所述第三分配装置响应于一个图像描述符并且进一步响应于被称为“新尺寸”的尺寸控制信号来进行分配，以及分配额外数量的位平面，使图像增加了新尺寸数量的位平面。

12、 如权利要求 7 所述的系统，还包括一个记录板，所述记录板包含与分配表位置数量相等的存储器组成的阵列，其中每个存储器适用于表示相应的分配表位置是否被增加了。

13、 如权利要求 12 所述的系统，还包括检测装置，用于检测试图对已增加的分配表位置进行分配时的条件，并产生一个表示所述条件的信号，还进一步包括用于响应该信号来禁止分配、直到不再能检测到所述条件为止的响应装置。

14、 如权利要求 12 所述的系统，还包括检测装置，用于检测试图对已增加的分配表位置进行分配时的条件，并产生一个表示所述条件的信号，还进一步包括用于响应该信号来设定事件存储器的装置。

15、 如权利要求 12 所述的系统，其中分配表被划分为多个区段，每个区段包含多个存储位置，并且其中记录板被分割为相应的区段，每个记录板区段包含一个由适用于表示分配表相应区段内的位置分配状态的存储器组成的阵列。

16、 如权利要求 1 所述的系统，还提供了用于重新分配图像位平面的第一重新分配装置，所述第一重新分配装置包含表示可用位平面的一个物理地址库，并且其中所述重新分配包括从分配表中读取物

理地址并将该物理地址写入该库中。

17、 如权利要求 16 所述的系统，其中所述库是一个 FIFO 存储器。

18、 如权利要求 16 所述的系统，其中处理阵列被划分为多个区段，每个区段包含多个位平面，并且其中该库被划分为相应的区段，每个库区段包含用于存储一个阵列区段的物理地址的存储装置。

19、 如权利要求 16 所述的系统，还包括用于向一个增加的图像重新分配位平面的第二重新分配装置，所述第二重新分配装置响应于一个图像描述符并进一步响应于一个释放控制信号执行重新分配，所述释放控制信号表示该图像的所有位平面都将被重新分配。

20、 如权利要求 16 所述的系统，还包括用于向一个增加的图像重新分配位平面的第三重新分配装置，所述第三重新分配装置响应于一个图像描述符并进一步响应于被称为“新尺寸”的尺寸控制信号执行重新分配，并且重新分配多个位平面，使得图像增加了新尺寸数量的位平面。

21、 如权利要求 16 所述的系统，还包括一个记录板，所述记录板包括与分配表位置数量相等的存储器组成的阵列，其中每个存储器适用于表示相应的分配表位置是否被增加了。

22、 如权利要求 21 的系统，还包括检测装置，用于检测试图对未增加的分配表位置进行重新分配时的状态，并产生一个表示所述状态的信号，还进一步包括用于响应该信号来禁止重新分配、直到不再能检测到所述状态为止的响应装置。

23、 如权利要求 21 的系统，还包括检测装置，用于检测试图对未增加的分配表位置进行重新分配时的状态，并产生一个表示所述状态的信号，还进一步包括用于响应该信号来设定事件存储器的装置。

24、 如权利要求 21 的系统，其中分配表被划分为多个区段，每个区段包含多个存储位置，并且其中该记录板被划分为相应的区段，每个记录板区段包含一个由适用于表示分配表相应区段内的位置分配状态的存储器组成的阵列。

25、 如权利要求 1 所述的系统，还包括用于处理指令的处理装置，所述指令包括至少一个图像描述符，并且还包括一个被称为“新尺寸”的尺寸控制，以决定作为处理该指令的结果而使图像增加的位平面的数量。

26、 如权利要求 25 所述的系统，还包括用于用图像描述符的所存储的拷贝中的新尺寸值来替代尺寸属性的替代装置。

27、 如权利要求 1 所述的系统，还包括用于处理指令的处理装置，所述指令包括至少一个图像描述符，并且还包括至少一个释放控制，来决定作为处理该指令的结果而使图像增加的所有位平面都被重新分配。

管理位平面资源的方法和装置

技术领域

本发明涉及 SMID 并行处理，尤其涉及位平面资源的分配。

背景技术

采用并行度最高的并行处理结构是遵循单指令多数据（SMID）方法及使用最简单可行的处理元件（PE）结构：一个单比特运算处理器。因为每个 PE 的处理输出能力很低，PE 逻辑的简单性支持用大量的 PE 来构造处理器阵列。通过把这样大量的 PE 组合成 SIMD 处理器阵列，能够获得非常高的处理输出。

位串行 SIMD 结构的一种变体是将 PE 连接成 2 维网格，每个 PE 与阵列内其北面、南面、东面和西面的 4 个相邻 PE 形成联通。2 维结构适用于，但并不限于，处理具有 2 维结构的数据，例如图像像素数据。

发明内容

本发明的一个方面提供了一种数字数据处理系统，其包括：

- 一个处理元件阵列，其具有可寻址数据存储装置，适用于处理位平面数据；
- 一个分配表，其包括用于存储物理地址的存储装置；
- 用于传送一个包含虚拟地址属性和尺寸属性的图像描述符的装置，所述虚拟地址和尺寸规定了分配表位置的范围，其中存储了包含一个图像的位平面的物理地址。

本发明的实施例的其他方面和优点的进一步细节将在后面结合附图描述。

附图说明

附图中：

图 1 示出了一个示例性处理元件（PE）的示意图。

图 2 是处理元件阵列的图示。

图 3 示出了构成处理元件组（PEG）的 PE 阵列的示意图。

图 4 是一个 PEG 的示意图。

图 5 是一个 SIMD 阵列处理器的示意图。

图 6 示出了分层次的阵列序列发生器的各级示意图。

图 7 示出了一个图像描述符的组成。

图 8 示出了一个位平面管理元件的图示。

图 9 示出了分配控制信号的表格。

图 10 示出了分配表内的所分配的图像的图示。

图 11 示出了与分配相关的事件信号的表格。

图 12 示出了基本序列发生器产生的分配核查信号的表格。

具体实施方式

本发明涉及数字数据的并行处理，尤其涉及数字图像像素数据。尽管本文公开的实施例涉及了图像像素数据的特定情况，需要理解的是可以用任意数字数据替换像素数据，并没有偏离本发明的范围和主旨。

本发明是并行处理器的一部分。该处理器包括一个处理元件（PE）的阵列，序列控制逻辑，以及像素输入/输出逻辑。该结构是单指令多数据（SIMD），其中单指令流控制所有 PE 的执行，并且所有 PE 同时执行各个指令。PE 阵列被称为 SIMD 阵列，整个并行处理器被称为 SIMD 阵列处理器。

SIMD 阵列是网格连接的处理元件阵列。每个处理元件（PE）包含存储器、寄存器和处理 1 位数据的计算逻辑。在本发明的一个示例性实施例中，阵列包括 48 行和 64 列 PE。SIMD 阵列构成了 SIMD 阵列处理器逻辑的主要部分，并执行基本上所有的像素数据计算。

每个 PE 100 包括存储器、寄存器和处理 1 位数据的计算逻辑。尤其参看图 1，示例性的 PE 100 包含处理 1 位数据的 PE RAM 110，ALU 101，逻辑模块 A 120，B 130 以及寄存器 C 140，D 150，NS 160，EW 170，AL 180，BL 190 和 CM 105。

PE RAM 110 对于每个 PE 是 1 个比特宽，存储由 PE 处理的像素数据。多位像素值用 RAM 110 内存储的多个比特表示。通过依次处理操作数像素的相应比特来执行多位操作数的运算。在这个示例性实施例中，RAM 110 在每个周期中执行最多 2 次读取和 1 次写入。读取和写入的组合是通过使用包括 2 组双端口 RAM（每组 1 个读取，1 个写入）的“分段”RAM 来实现的。对于每个 PE 的全部 256 个比特，这两组的深度分别是 128 位。只要源操作数是在相对的段中，分段 RAM 方法允许每个周期中的 2 次读取和 1 次写入。另一个实施例中，一个 256 位的 3 端口 RAM 提供了相同的访问，而不用考虑源操作数的设置。其他实施例也可以采用其他多重访问方法（例如，1 次读取和 1 次写入），或者在每个周期中提供单次读取或写入访问。

如图 2 所示，一个示例性的 PE 阵列 1000 包括 48 行和 64 列 PE。像素序号是从该阵列的西北角的 0，0 到东南角的 47，63。

处理过程中，阵列的所有 PE 同时执行每个操作步骤。每次读取或写入一个操作比特，在 PE 寄存器之间每移动一个比特，阵列的每个 PE 同时执行每个 ALU 输出。在说明这种操作模式时，总体考虑相应的图像比特是比较有利的。相应图像比特的阵列大小的集合被称为一个“位平面”。根据（串行）指令流的观点，SIMD 阵列操作被模块化为位平面操作。

这个示例性实施例中的每个指令包含指引流程或处理位平面的指令。单指令可包含多个指令字段，1 个字段用于各个寄存器资源，1 个字段用于 PE RAM 写入端口，并且一个附加字段用于 ALU 101 的控制处理。这种方法是阵列指令的常规微指令实现方式，为单个处理周期提供了阵列控制。

示例性的 PE 阵列 1000 在实现中是分层次的，PE 被分成 PE 组（PEG）。每个 PEG 200 包含 64 个 PE，表示一个 8×8 阵列区段。因此 48×64 PE 阵列 1000 由 6 行 PEG 实现，每行有 8 个 PEG。每个 PEG200 被耦合到其相邻的 PEG，使得在 PEG 边界处形成了 PE 至 PE 的联通。这种耦合是无缝的，因此对于位平面的操作，PEG 分区

是不可见的。需要注意的是这些阵列尺寸仅是示例性的，使用其他尺寸的阵列也没有偏离本发明的范围。

示例性的 PEG 200 包括一个 64 位宽的多重访问 PE RAM 210，PEG 控制逻辑 230，以及构成 PE 阵列 202 中的 64 个 PE 的寄存器和计算逻辑。PE RAM 210 的每个位片被耦合到 64 个 PE 中的一个上，为每个 PE 提供一个有效 1 位宽的 PE RAM 110。

每个示例性的 PEG 除了与北面、南面、东面和西面的相邻 PEG 相联通外，其还包括一个 8 比特输入和输出路径，用于将像素数据移入和移出 PE 阵列 202。CM 寄存器平面在输入和输出过程中处理位平面数据。数据被移入和移出位平面形式的 PE 阵列 1000。

上述 PE 阵列提供了执行像素数据运算的计算逻辑。为了执行这些操作，PE 阵列需要一个指令源，支持像素数据移入和移出阵列。

图 5 示出了一个示例性的 SIMD 阵列处理器 2000。SIMD 阵列处理器 2000 包括阵列序列发生器 300，用于向 PE 阵列 1000 提供指令流。还提供了一个像素 I/O 单元 400，用于控制像素数据移入和移出 PE 阵列 1000。总的来说，这些单元包含一个 SIMD 阵列处理器 2000。

SIMD 阵列处理器 2000 可用于执行阵列大小的图像部分的算法。此处理器可以在一个集成电路设备上实现，或者作为单个设备上的一个较大系统的一部分。另一种实施方式中，SIMD 阵列处理器 2000 是系统控制处理器（本文中简称为 CPU）的从属设备。SIMD 阵列处理器 2000 和 CPU 之间的接口通过 CPU 为示例性 SIMD 阵列处理器 2000 提供初始化和控制。

像素 I/O 单元 400 通过称为“Img Bus”的图像总线控制像素数据在 PE 阵列 1000 和外部存储器之间的移动。像素数据的移动和 PE 阵列计算同时进行，从而提供了更高的像素数据处理的吞吐量。像素 I/O 单元 400 执行图像数据在像素格式和位平面格式之间的转换。Img Bus 数据是像素格式的，PE 阵列数据是位平面格式的，这两种格式之间的数据转换由像素 I/O 单元 400 执行，并作为 I/O 处理的一部分。

SIMD 阵列处理器 2000 处理称作“子帧”的阵列大小的部分中的

图像数据。典型的情况下，待处理的图像帧比 PE 阵列 1000 的尺寸要大得多。图像帧的处理是通过依次处理子帧图像部分来实现的，直到图像帧被全部处理。

对 PE 阵列 1000 处理子帧的控制是通过分层次设置序列发生器单元来提供的（图 6）。这些单元包括程序序列发生器 330，其对应用程序排序，并向基本序列发生器发生 340 分配图像操作（也称为“基元”），基本序列发生器 340 对每个基本操作进行排序，向覆盖（overlay）单元 350 分配单个位操作，覆盖单元 350 把位操作微指令字段安排成适当的管道级，从而解决了任何指令冲突。覆盖单元 350 的输出是微指令流，其对于 PE 阵列提供时钟到时钟的控制，用于执行子帧操作。

程序序列发生器 330 实现了一个看起来很常规的机器语言指令集，其提供 SIMD 内核的序列控制和各种支持操作，并向基本序列发生器 340 分配“基本操作”形式的指令。基本操作是对于连续存储器地址序列重复的特定微指令（或一个微指令和适当的派生物），即对于各个操作数比特（平面）重复。基本阵列操作，例如一个图像 ADD，可以用单个基本操作来实现。

基本模型将一个图像操作数定义为位平面集合，它从起始地址开始，在 PE RAM 内从起始平面开始延伸一定数量的位平面。包含图像的位平面数量被称为图像“尺寸”。每个图像还具有一个“符号”属性，其表示各个像素值的位模式应被解释为有符号数据（符号=1）还是无符号数据（符号=0）。这些属性总称为图 7 中的“图像描述符”。需要注意的是这些实施例中给出的尺寸只是示例性的，可以采用任何不同的尺寸来缩放，这并没有偏离本发明的范围。

图像描述符被存储（用于处理）在程序序列发生器 330 的寄存器组内，并通过程序指令内的寄存器号进行索引。在执行图像操作数的运算之前，可能需要向处理器寄存器加载操作数描述符。程序序列发生器 330 通过提供 3 个图像操作数（2 个源操作数，1 个目标操作数）的描述符信息以及 PE 阵列 1000 所使用的一个 24 位基本指令字向基本序列发生器 340 分配基本操作。

基本序列发生器340的基本操作序列用两个源操作数图像的相加来表示，其结果被存储在一个目标操作数图像内。3 个操作数中的每个操作数用一个图像描述符表示。第一个位操作向每个源操作数添加比特 0，把结果存储在目标操作数的比特 0 内。从相应图像描述符的基地址值导出每个图像操作数位平面的地址。基本序列发生器 340 向覆盖单元 350 分配一个位操作，其包括 3 个操作数位平面地址和 24 位基本指令字。当基本序列发生器 340 工作时，各个描述符的地址字段被递增，依次表示下一位（比特 1 等）的操作数地址。当已经处理了和目标尺寸相对应的位个数之后，结束运算序列。当目标尺寸超过了源操作数的尺寸时，源描述符的符号属性被用于确定对于该源操作数是符号扩展还是零填充。

覆盖单元 350 处理位操作，产生将要由阵列执行的 PE 微指令字。位操作包括必须安排在适当的管道间隙内的微指令。覆盖单元 350 执行已安排的任务，并且解决内部微指令冲突。PE 微指令从覆盖单元 350 被分配到 PE 阵列 1000 的 PEG 用于执行。

对于编程者来说，图像被模块化为 PE RAM 内的位平面连续序列。图像描述符 600 提供一个基础地址和一个表示图像从基地址向上延伸并占用了连续地址的尺寸属性。在示例性的实施例中，只给出了部分图像。

图像在算法执行过程中不停来往，因此很难设置不采用无用集合的最大 PE RAM。编程者和软件工具编写者会遇到的困难是为了将分散的位平面收集到可用的图像模块中，PE RAM 使用效率不高（由于可用存储器的不可使用的小模块的增加），或者执行时间发生损失。由于 PE RAM 和执行时间都是 SIMD 芯片编程者的重点考虑方面，不具有无用集合的支持最优 PE RAM 的装置被结合到本发明的 SIMD 芯片中。

图像描述符 600 包括一个 9 位基地址元件来描述图像的基地址。基地址值跨过 512 个位平面的空间，低 256 位是 PE RAM 部分 0，高 256 位是部分 1。这是一个虚拟空间，因为示例性实施例中的可用物理

存储器对于各个部分是128个PE RAM平面。在此虚拟空间内，图像用连续的位平面模块表示。虚拟空间的各个部分在给定时间可以增加到128个位置。

PE RAM平面根据需求被分配到图像。在给定的时刻，图像应该被分配到表示图像像素的数值范围所需的最小数量的位平面。操作目标图像通常需要调整尺寸来容纳操作结果。这种调整是通过指定目标操作数的尺寸控制值“Newsize（新尺寸）”来实现的。源图像操作数也称为“Free'd”，即整体重新分配。稍后本发明将详细分配和重新分配控制。

虚拟空间通过查询表管理，即图8中的分配表700。在这个示例性实施例中，分配表700是一个512深的多端口RAM，其为每个9比特图像描述符基地址值提供物理PE RAM地址的查询。位平面分配到图像是通过从“比特池”中读取物理地址，并将其写入一个分配表来完成的。为了重新分配位平面，该地址从分配表700中读出，并被写入比特池。

共有两个比特池730和740，每个区段一个。在这个特定实施例中，每个比特池是一个128深的FIFO，其在任意给定时间包含所有可以用于分配的PE RAM地址。

如图8所示，一个512位的记录板寄存器，划分成记录板750和记录板760，用于表示哪个分配表位置在增长。记录板被基本序列发生器340用来加强分配和重新分配的顺序相关性，以及检测分配误差条件。与分配相关的误差被一个事件寄存器跟踪，向编程者提供一个报告。

在执行应用程序之前，分配逻辑应该处在一个“清除”状态。比特池应该用256PE RAM地址来初始化，记录板应该全都是0。当这两个条件都满足时，就存在了“清除”状态。

指定了一个阵列操作的程序序列发生器指令确定了2个源操作数和1个目标操作数。源操作数可以是或者不是图像操作数，但阵列操作的目标操作数必须是一个图像操作数。除了操作数，指令还规定了

如图 9 所示的分配控制。

指令的分配控制字段允许编程者通过Newsize直接确定执行操作所得到的目标操作数的尺寸。Newsize 会使得目标操作数变大（递增分配）、变小（递增重新分配）或者保持与当前目标操作数的相同尺寸。

Free1 和 Free2 控制允许编程者指定要重新分配的第一个源操作数（Free 1 有效）和/或要重新分配的第二个源操作数（Free 2 有效）。当 Free 控制被用于执行阵列操作之后，该控制会导致重新分配全部相应的操作图像。

图像存储器的管理直接受编程者的控制。编程者确定用于管理一个图像的分配表 700 内的地址跨度。编程者确定何时和如何增加虚拟地址空间区域。编程者通过生成适当的图像描述符数值来表示图像，并通过使用分配控制字段用于图像操作。

为了更好地理解本发明的其他细节，使用下列词汇表:

Free 1 和 Free 2 –用于 Src1 和 Src2 操作数的重新分配控制信号

Src1 和 Src2 – “源 1”和“源 2”， 用于阵列操作的两个源操作数，或输入

Dest –用于阵列操作的 “目标”操作数，或输出

Zend – 目标基地址+Newsize，在乘法序列过程中，确定何时停止生成位操作和分配位平面

Alloc.oldsize_crt – 递减计数器，在基本序列开始时加载初始 Dest 操作数尺寸值

Dest_in – 从程序序列发生器传到基本序列发生器的信号，其传送 Dest 操作数图像描述符

Alloc_in – 从程序序列发生器传到基本序列发生器的信号，其传送 Free 1，Free 2 和 Newsize 值

Alloc_en_reg – 保持分配使能信号的寄存器

Wadr_out –从程序序列发生器传到覆盖单元的信号，其传送位平面操作的写地址

Wadr_sel – 选择 Wadr_out 值的源的选择信号

Deall_en_reg – 保存重新分配使能信号的寄存器

Alloc.deall_wr0 或 Alloc.deall_wr1 – 把位平面地址分别写入(推入)比特池 0 和比特池 1 的写选通

Aadr – Bit_Op_Reg1 的 PE RAM 地址字段，在序列过程中依次提供 Src1 操作数虚拟地址

Badr – Bit_Op_Reg1 的 PE RAM 地址字段，在序列过程中依次提供 Src2 操作数虚拟地址

Wadr – Bit_Op_Reg1 和 Bit_Op_Reg2 的 PE RAM 地址字段，在序列过程中依次提供 Dest 操作数虚拟地址

Aadr_out – 从基本序列发生器传到覆盖单元的信号，其提供位操作输出的 Src1 操作数物理地址分量给覆盖单元

Badr_out – 从基本序列发生器传到覆盖单元的信号，其提供位操作输出的 Src2 操作数物理地址分量给覆盖单元

Wadr_out – 从基本序列发生器传到覆盖单元的信号，其提供位操作输出的 Dest 操作数物理地址分量给覆盖单元

Free_1_reg – Src1 操作数的重新分配任务描述符；该名称的字段被同时包括在基本寄存器和分配任务寄存器内

Free_2_reg – Src2 操作数的重新分配任务描述符；该名称的字段被同时包括在基本寄存器和分配任务寄存器内

Free_3_reg – Dest 操作数的重新分配任务描述符；该名称的字段被同时包括在基本寄存器和分配任务寄存器内

图 10 给出了图像存储器的一个实例。该实例中，图像 A 和 B 是分别存储在字段 0 和字段 1 中的 8 比特图像。这两个图像都被完全增加，如阴影表示。假设 A 是无符号，B 有符号，该描述符应该是:

A=（0<<15）|(8<<9)|0 = 0 × 1000

B=（1<<15）|(8<<9)|256 = 0 × 9100

现在，假设 A 和 B 被相加以生成图像 C，必须为 C 选择某个位置。必须有足够大的连续空间来容纳 C 的中间值，最好足够大以能够

容纳其有效跨距内的所有数值 C。所示实施例中，编程者选择一个未使用的分配表位置模块给 C 使用，从地址 10 开始，并扩展要由 C 所使用的 10 个比特。

关于这一点，C 是一个新图像，因此没有增加（即没有给 C 分配位平面）。假设 C 是一个有符号的图像，C 的描述符是:

C=（1<<15）|(0<<9)1|0 = 0 × 800a

对于这个实例，上述描述符 A、B 和 C 是在预备操作中置于寄存器组 331 内的值。现在编程人员必须基于操作结果而决定 C 的尺寸。在这个实例中，A 和 B 有不同的信号属性，而尺寸都是 8 位。

假定，操作结果是 10 位，从而容纳像素 A 和 B 之间所有可能的和。如果信号属性相同，则结果尺寸很可能是 9 位。然而，编程人员基于他/她对操作结果的认识，可能选择 8 位结果，或者简单地保持结果的最低 8 位。通过 Newsize 控制，程序人员能够规定 C 的结果尺寸。

假设要得到的是 10 位结果，并且 A 和 B 没有重新分配（释放），这个操作的分配控制应该是:

分配控制=(10<<2)(0<<1)|0=0×28

这个操作的结果是 C 图像增加了 10 个位平面。程序人员通过把该操作的 Newsize 设定为 10 来确定该结果。程序序列发生器 330 根据新的尺寸属性来自动更新 C 的描述符作为阵列操作的结果，因此现在 C 描述符的值为:

C=(1<<15)|(10<<9)|10=0×940a

由此，编程人员为每个阵列操作调整目标操作数的尺寸，使得该图像是存储操作结果的像素值所需要的最小尺寸。通过用 Newsize 来更新目标操作数寄存器，编程人员不需要由于描述符的数值变化而重新载入描述符。

在这个实例中，C 的初始尺寸是 0。在许多情况下，一个目标操作数将有一个非零尺寸，并且将根据阵列操作来调整尺寸。在所有情况下，Newsize 控制确定操作结果的尺寸，并且目标操作数的尺寸属性（在寄存器组 331 中）被 Newsize 值所替代。如果 Newsize 大于目

标值，则发生一个增加分配，其中分配位平面以增加图像的尺寸。如果 Newsize 小于目标值，则发生一个递增重新分配，其中重新分配位平面以减少图像的尺寸。

当不再需要一个图像时，它必须被重新分配。通常使用 Free1 或 Free2 分配控制来完成重新分配。一个释放控制被施加于访问图像的最后操作。一个释放控制只被施加于源操作数。这意味着图像的最后访问被作为源操作数，否则将产生一个值（写入）并且不会被使用（读取）。使用释放控制也可有助于确保在第一时机进行重新分配，从而使得重新分配位平面尽快可用。

当应用一个释放控制时，编程人员必须确保采用了一个完整的描述符。例如，一个图像的最后访问可以是该图像的一部分而并非是整个图像。如果释放控制被施加于一个图像部分，则部分图像会保持被分配。一种解决方案是将该释放控制延迟成一个分离控制，它对整个图像进行释放控制。另一种解决方案通过使用操作来释放图像部分，然后产生一个表示其余位平面的描述符，并且对图像的该部分进行释放操作。

有多种方法来破坏图像分配逻辑的状态。再来看图 10 中的实例，可以看出图像 C 没有被增加，但是却使用了一个尺寸为 10 位的无效描述符。在这种情况下，Newsize 和尺寸相匹配，因此没有出现分配或重新分配。在这种情况下，在分配表 700 中将使用 10 到 19 位之间的物理地址来执行操作。如果这些地址属于其它图像，该位平面数据将被破坏。如果这些地址全部在一个比特池中，并且图像 C 从来没有被重新分配，那么编程人员将会侥幸避免此失误。访问没有增加的分配表地址将不会被 SIMD 内核所标记。

继续该实例，如果编程人员随后重新分配图像 C，则会错误地发送地址到比特池。这会使得在比特池中复制值，导致图像间串扰。这也可能会导致比特池的溢出，因为有额外地址被写入。并且，未增加分配表位置的重新分配会被检测到，并被标记为一个重新分配的冲突。

如果一个图像不止一次被释放，将产生一个重新分配冲突并且由

于额外地址被写入比特池而导致最终比特池溢出。如果不能释放一个图像，由于数位没有被替换而连续分配，将最终导致数据库下溢。如果在没有图像重新分配的情况下再次使用该图像的分配表空间（也就是新的位平面被分配到相同的空间），将产生一个分配冲突事件。

可以列举导致错误发生的其它情况。简而言之，编程人员必须确保图像描述符不会与分配表 700 的实际状态相冲突。他/她必须确定分配表入口的增加先于作为源操作数被使用，以及操作表入口在最终使用后被重新分配。

一个图像属于一个区段或另一个区段；在示例性实施例中图像没有跨过区段的边界。图像描述符地址属性的高位（位 8）表明图像相对于分配表 700 和 PE RAM 的区段。分配表区段被卷绕，使得超过区段末端的一个图像块被卷绕回相同区段的开始位置处。

因为比特池的初始内容为编程人员所控制，SIMD 内核可以具有一定程度的容错量。尤其，如果一个单个 PE RAM 平面发生错误，可以通过从所有比特池中去掉相应的位平面地址来将其分离。这留给编程人员更少的位平面，这可以通过在所涉及的应用程序或软件工具来完成。

如果基本指令并未确定写入到 PE RAM，则不会执行递增分配和重新分配。换句话说，如果目标操作数没有被写入，在示例性实施例中将不会发生分配和重新分配。

在示例性实施例中，物理上存在 256 位平面的 PE RAM 用来存储有效图像。这些 256 位平面被分成 128 位平面区段。每个区段的 128 位平面由 128 个地址来表示。可用于分配的地址置驻留在 FIFO 比特池中。有 2 个比特池，即比特池 0（730）和比特池 1（740），每个 PE RAM 区段使用一个比特池。位平面分配给图像包括从适当的比特池中读取位平面。为了进行重新分配，位平面地址被写回到比特池中。

通过查询分配表 700，将图像描述符位平面地址映射到物理 PE RAM 地址。分配表 700 是一个 512 位深、8 位宽的 5 端口 RAM（4 个读取端口，1 个写入端口）。图像描述符的基地址和尺寸属性描述

了分配表 700 中的分配范围，该分配表包含图像操作数的 PE RAM 地址。由于位平面被分配给一个图像，它们被写入到分配表 700 中的相应位置。由于位平面被重新分配，它们被从分配表 700 中读出并写回到适当的比特池中。因此，在位平面的分配和重新分配过程中，使用分配表的读取端口和写入端口中之一。其它三个读取端口被用于在操作过程中查询记录板 750、记录板 760 和目标操作数的 PE RAM 地址。

在分配表 700 中，区段 0（710）提供 PE RAM 区段 0 的地址映射，而区段 1（720）提供 PE RAM 区段 1 的映射。图像不能跨越区段；每个图像被完全存储在一个区段内。

分配表 700 所包含的位置数量是位平面数量的两倍。这使得分配表位置中可以使用“孔洞”，而不会浪费物理位平面地址。

应该指出，在执行过程中比特池中的位平面地址混杂会导致地址序列内的大量混乱。因此，分配给一个图像的物理地址可以是 PE RAM 中位平面的一个近似随机集合。这点不会产生影响，因为地址是从分配表 700 中依次读取的（也就是“虚拟”地址序列）。

序列操作过程中，基本寄存器 342 的 Src1、Src2 和 Dest 操作数地址字段被提供到 Bit Op1 寄存器 343 的 Aadr、Badr 和 Wadr 字段。Aadr、Badr 和 Wadr 字段提供在分配表 700 中 3 个操作数的查询地址。从分配表 700 中读取的物理地址包括输出给覆盖单元 500 的 Aadr_out、Badr_out 和 Wadr_out。由于分配表 700 是一个同步 RAM，此读取提供了与其余位操作的地址数据同步的单个时钟延迟，该延迟分段通过 Bit Op2 寄存器 344。

启动时，比特池必须用物理位平面地址进行初始化。比特池 0 730 必须加载地址 0 ...127，比特池 1 740 必须加载地址 128...255。

只有当图像是目标操作数时，才执行对图像的位平面分配。基本序列发生器 340 接收一个目标操作数的当前尺寸（Dest_in[14:9]）和一个新尺寸（Alloc_in[7:2],也就是 Newsize）。如果 Newsize 的数值大于当前尺寸，就为每个超过当前尺寸的位平面进行位平面分配，从而将目标图像的尺寸调整为一个新的尺寸值。

在基本寄存器 342 的加载过程中，Newsize 值被载入到基本寄存器 Dest_size 字段以用于对基本序列进行排序。Dest_in 尺寸值被载入到 Alloc.oldsize_cnt 计数器。在对基本序列进行排序的过程中，Dest_size 和 Alloc.oldsize_cnt 值被递减计数。一旦 Alloc.oldsize_cnt 值达到 0，便开始位平面分配。而 Dest_size 继续递减计数（并且 Alloc.oldsize_cnt 是 0），位平面被分配到目标操作数图像。

如果是一个乘法操作，由于操作是多通道的，分配将更加复杂。基本寄存器 342 的 Zend 字段跟踪位平面的分配到乘法目标操作数。每次当对目标操作数的写入超过 Zend 值时，就执行一次分配并且递增 Zend。

基本序列发生器 340 通过设定 Alloc_en_reg 来请求分配。寄存器用 Bit Op1 寄存器 343 进行分段。响应有效的 Alloc_en_reg，从适当的比特池中读取一个位平面地址，并且将位平面地址写入到分配表 700 内由 Bit Op1 寄存器 343 的 Wadr 字段给出的地址上。

如果位操作没有发生位平面分配，则从分配表 700 中读取 Wadr_out 值。然而，如果出现分配，则从比特池中产生一个新的位平面地址而不是分配表输出，并将其提供给 Wadr_out。从比特池中得到的新的位平面被载入到 Bit_pool_wadr 寄存器，使得与 Bit Op2 寄存器命令同步提供。对应于 Alloc_en_reg 值来设定 Bit Op2 寄存器 344 的一个字段 Wadr_sel，从而在 Bit_pool_wadr 值和分配表值之间进行适当选择，用来计算 Wadr_out。

对一个操作的任何操作数都可以重新分配位平面。分配控制 900 包括 2 个单个的位字段，其中的每个字段都指示源操作数的“释放”或重新分配。此外，如果 Newsize 值小于 Dest_in 尺寸值，则表示进行递增重新分配。一个源操作数的释放意味着由该源操作数描述符表示的所有位平面的重新分配。然而，目标操作数位平面的递增重新分配意味着位平面的重新分配超过了 Newsize 值。

位平面的重新分配与基本操作同时进行，尽管与分配不同，重新分配并没有作为基本操作的一部分。一个分离的重新分配序列发生器

为重新分配任务进行排序。重新分配任务被写入一个深度为 8 位的FIFO，称为重新分配队列。当该重新分配序列发生器完成了一个重新分配任务，便从该重新分配队列中读取出一个新的任务。

排列重新分配任务的过程从加载基本寄存器(342)Free_1_reg、Free_2_reg 和 Free_3_reg 字段开始。将 Src1 和 Src2 地址和尺寸字段（低 15 位）加载到 Free_1_reg、Free_2_reg 寄存器。当相应的分配控制输入字段（Free1，Free2）有效时，上述每个寄存器被设置为高位。只有设定为高位时，每一个寄存器都标记为重新分配。Free_3_reg 被设定为代表执行递增重新分配的任务值。如果表示重新分配，则更高位被设定，地址字段被设定为 Dest_in 地址值与 Newsize 值的和，并且尺寸字段被设定为 Dest_in 地址值与 Newsize 值的差。

基本寄存器 Free_1/2/3_reg 字段被写入到一个分配任务寄存器的相应 Free_1/2/3_reg 字段。这个载入是一同进行的，即所有 3 个寄存器被同时加载。只有在当前 Alloc Free_1/2/3_reg 任务全部已经排队的情况下，才允许把新任务从基本寄存器 342 载入到分配任务寄存器（分配逻辑 341）。并且，直到已经生成了基本操作的第一位操作时（防止出现不依照顺序的重新分配），基本存储器 Free_1/2/3_reg 的任务才可以被载入到分配任务存储器。

从 Alloc Free 寄存器将重新分配任务写入到重新分配队列。对于每个具有有效的最高位（MSB）的空闲寄存器，向队列写入一个任务。因为有 3 个空闲寄存器，这个过程要进行 3 个周期才能完成。当任务被写入队列时，被读出任务的寄存器的 MSB 位被清除，以表示寄存器内不再有一个有效的任务。

如果重新分配队列已满，则保持任务的顺序，直到队列不再排满为止。如果需要，为了允许完成排序，将保持基本任务的排序，以使得基本寄存器 Free 字段值可以被传输到 Alloc Free 寄存器。

重新分配任务用重新分配序列发生器进行排序。一个重新分配任务由基地址和尺寸来表示。基地址递加，同时尺寸递减。对于每次计数，进行一次重新分配。

通过设置 Dell_en_reg 以及向 Dell_addr_reg 载入用于重新分配的位平面地址，来请求一个位平面重新分配。一旦（重新分配序列发生器的）尺寸计数器达到 1，便从一个可用的重新分配队列中读取出一个新的重新分配任务。如果没有有效的重新分配任务，则不会发生重新分配。

一个有效的 Dell_en_reg 使得读取 Dell_addr_reg 所提供的地址处的分配表 700。根据该区段，Alloc.deall_wr0 或 Alloc.deall_wr1 被设置为在下一周期内将分配表读取数据推入适当的 FIFO（比特池 0（730）或比特池 1（740））。

分配和重新分配过程的不同之处在于，分配任务作为操作的一部分来执行，而重新分配任务被排序，从而具有独立序列。由于重新分配任务的排序，可以在顺序之外执行分配任务。也就是说，位平面的重新分配可以由一个操作进行排序，当下一次操作试图分配相同的位平面时，其仍然保持在队列内。从编程人员的角度来说，位平面是可用的，但在执行时它们有时会不能用。

对于编程人员或软件撰写者来说，进行分配和重新分配能力分析以判断分配任务能何时发生会是一个困难并难以接受的负担。由于必须知道程序内每个有效操作在程序之外的执行效果，独立程序内的码序列封装几乎是不可行的。

强制执行分配和重新分配指令的装置被包含在基本定序器 340 中。该指令执行装置是包含由 512 个寄存器组成的阵列的记录板，每个寄存器对应一个分配表位置。当发生一个分配时，设置对应于分配表地址的记录板寄存器。当发生重新分配时，记录板寄存器被清除。

如果为一个分配表地址请求一个分配，并设置了相应的记录板存储器，如果有重新分配任务等待处理，就存在一个顺序之外的分配状态。对应这个状态，保持基本定序器 340，以允许执行该待处理的重新分配任务，然后在进行分配操作时执行此重新分配任务。如果记录板仍然表示顺序之外的分配没有待处理的重新分配任务，或者如果记录板表示顺序之外的重新分配任务，则表示有一个分配错误。在这种

情况下，判定事件信号并且执行操作。

基本序列发生器分配逻辑 341 不能防止上述类型的错误，但是至少它们能够检测出来这些错误，并在事件寄存器中作为被记录的事件进行传递。基本定序器 340 记录 7 个事件，只有一个事件和分配有关。图 11 对这些事件进行了描述。

除了上述提到的情况，会产生 2 个核查信号。每个核查信号适用于一个区段，并且当相应的比特池已满并且相应的半个记录板被清除（没有被分配的位）时，该核查信号有效。这些信号通过事件寄存器传送到程序序列发生器 300，此时它们是可用的。核查信号为编程人员提供了完整性检查，他/她在应用程序中期望进行此检查的停顿点处检查分配被“清除”。与其它事件信号不同，该核查信号在程序序列发生器的事件寄存器中不是“停滞”的，即它们被持续更新，一旦设定后不必保持判定结果。

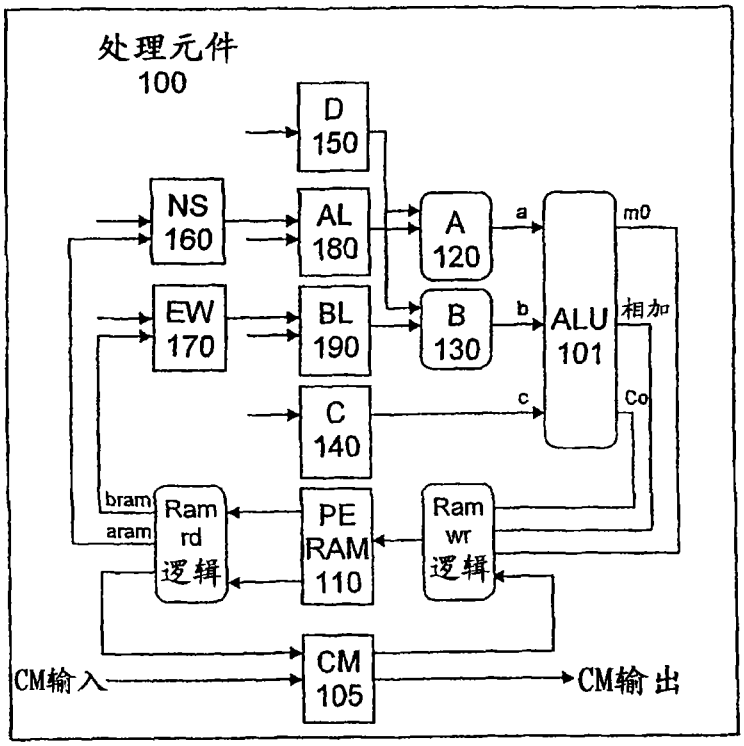

处理元件
100
D
150
NS
160
AL
180
A
120
a
m0
EW
170
BL
190
B
130
b
ALU
101
相加
C
140
c
Co
bram
aram
Ram
rd
逻辑
PE
RAM
110
Ram
wr
逻辑
CM
105
CM输入
CM输出

图1

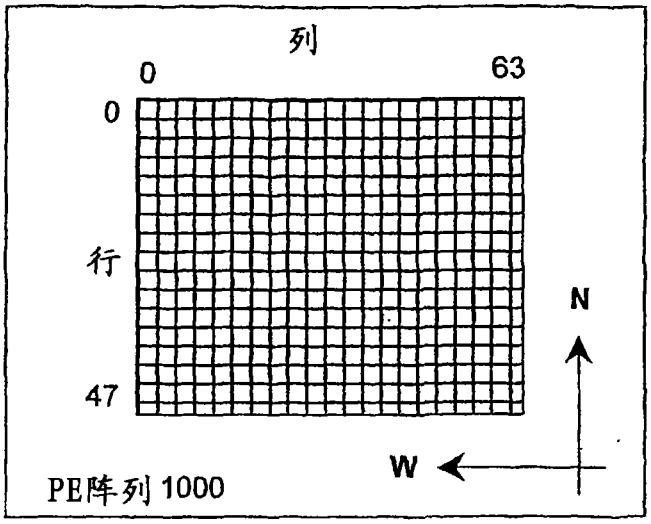


图 2

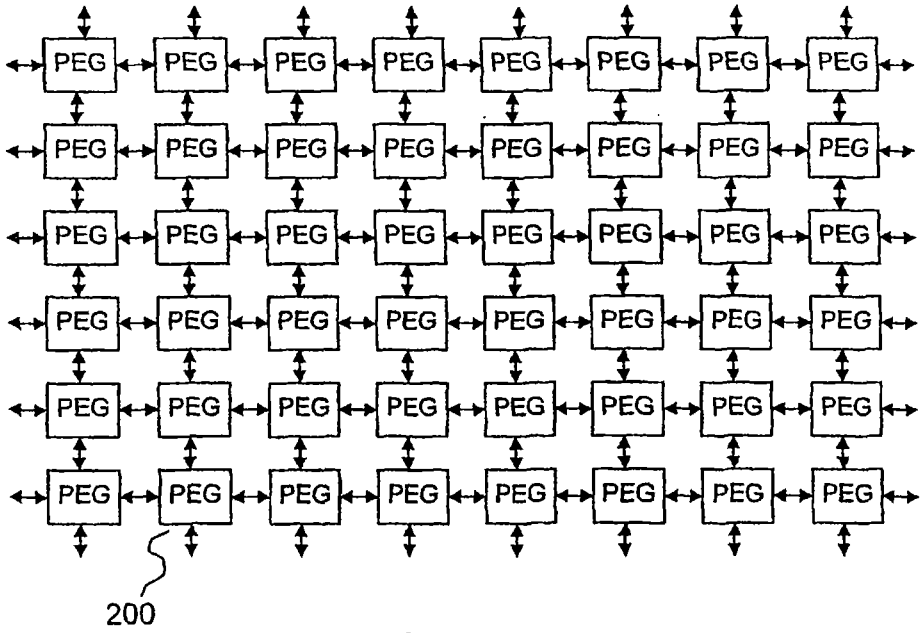


图 3

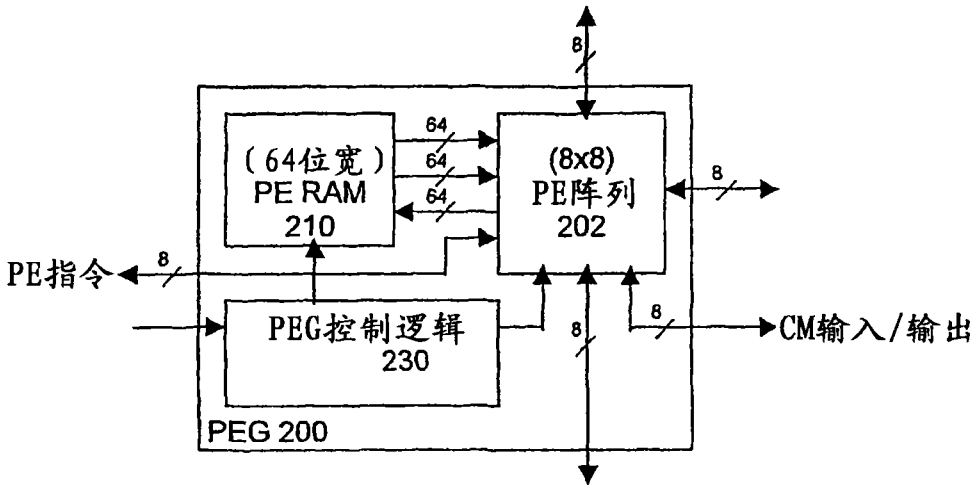

8
(64位宽)
PE RAM
210
64
64
64
(8x8)
PE阵列
202
8
PE指令
8
PEG控制逻辑
230
8
8
CM输入/输出
PEG 200

图4

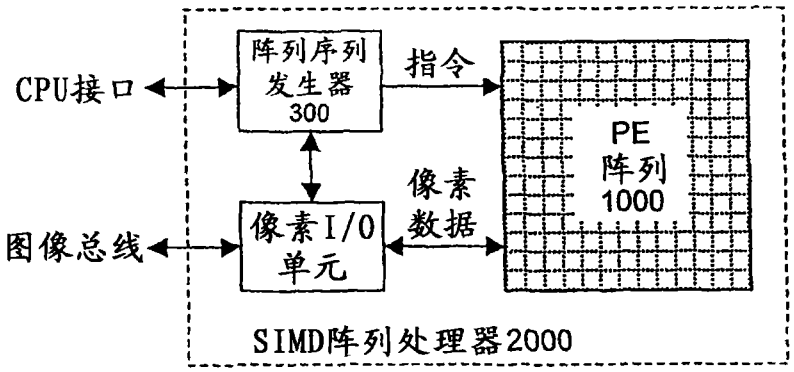

CPU接口
阵列序列
发生器
300
指令
PE
阵列
1000
像素
数据
图像总线
像素I/O
单元
SIMD阵列处理器2000

图5

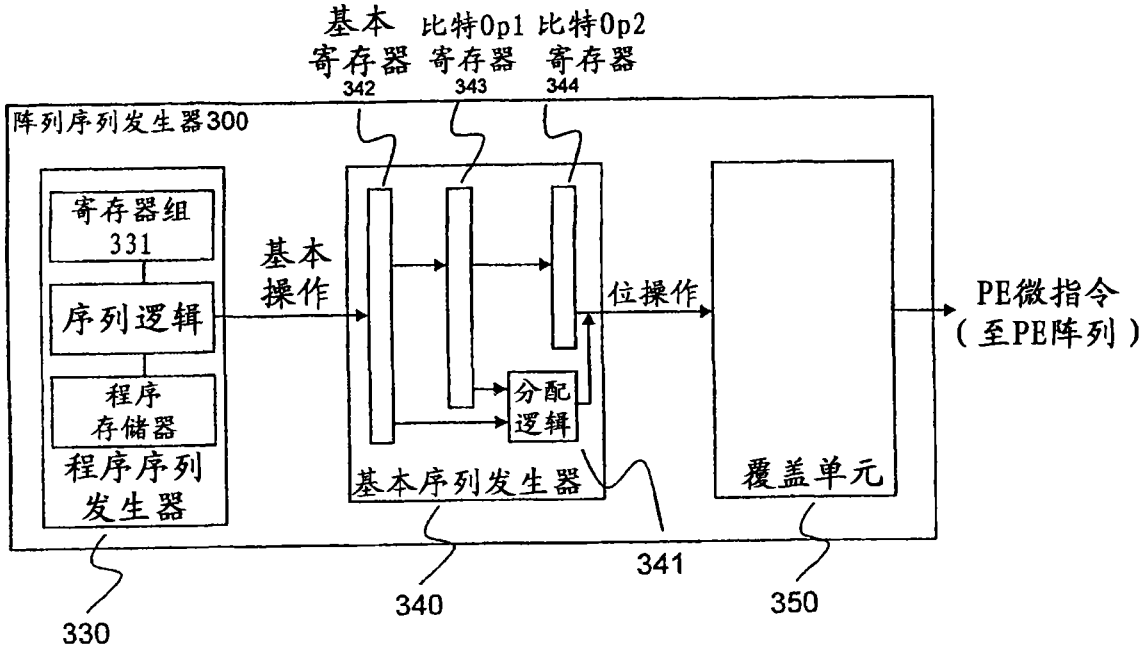


图6

图像描述符600

15	14-9	8-0
符号	尺寸	基地址

图7

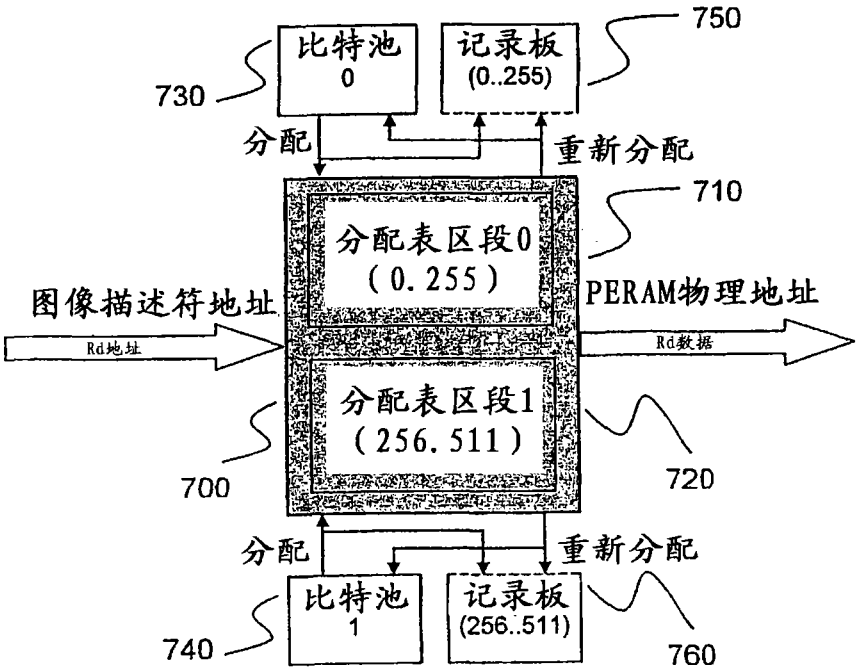


图8

分配控制 900

7-2	1	0
Newsize	Free2	Free1

图9

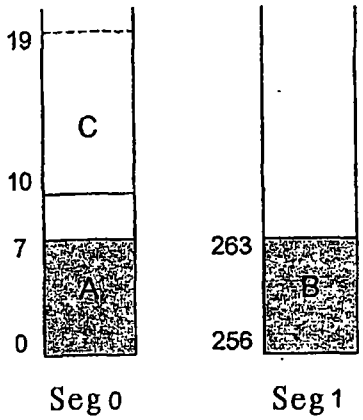


图10

事件信号

事件比特	事件描述
0	比特池0下溢
1	比特池0上溢
2	比特池1下溢
3	比特池1上溢
4	记录板违反：分配
5	记录板违反：重新分配
6	

图11

核查信号

区段	核查信号
0	Prim_Audit_out[0]
1	Prim_Audit_out[1]

图12