(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 108139910 A
(43)申请公布日 2018.06.08

(21)申请号 201680060280.2

(22)申请日 2016.09.14

(30)优先权数据
1518735.4 2015.10.22 GB

(85)PCT国际申请进入国家阶段日
2018.04.13

(86)PCT国际申请的申请数据
PCT/GB2016/052837 2016.09.14

(87)PCT国际申请的公布数据
WO2017/068318 EN 2017.04.27

(71)申请人 ARM有限公司
地址 英国剑桥

(72)发明人 贾科莫·加布利尔
奈杰尔·约翰·斯蒂芬斯

(74)专利代理机构 北京东方亿思知识产权代理有限责任公司 11258
代理人 林强

(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/38(2006.01)

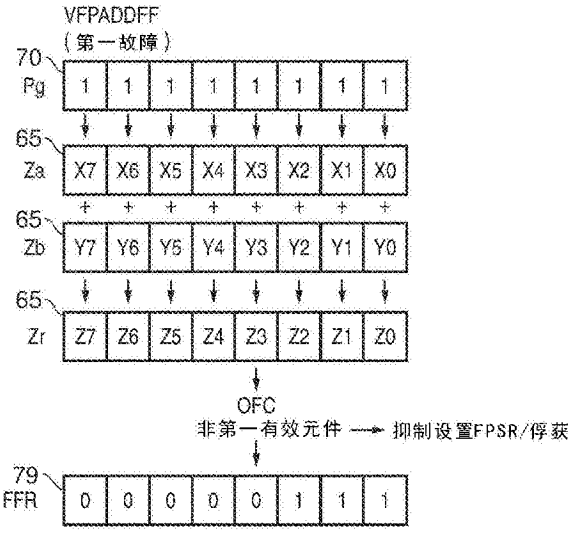
权利要求书3页 说明书12页 附图10页

(54)发明名称

处理用于向量算术指令的例外状况

(57)摘要

处理电路30、35支持规定至少第一输入向量的第一类型的向量算术指令。当针对算术运算侦测到至少一个例外状况时，处理电路30、35执行至少一个响应动作，该算术运算针对预定序列中的第一输入向量的第一有效数据组件执行。当针对预定序列中除第一有效数据组件之外的给定有效数据组件侦测到至少一个例外状况时，处理电路30、35抑制至少一个响应动作且存储组件识别信息，该组件识别信息识别哪个数据组件是触发例外状况的给定有效数据组件。这可用来降低用于追踪例外状况的出现和/或支持向量指令的推测执行的硬件资源量。



CN 108139910 A

1.一种设备,包括:

处理电路,用于响应于规定包括多个数据组件的至少第一输入向量的第一类型的向量算术指令,执行针对该第一输入向量的至少一个有效数据组件的算术运算;

其中该第一输入向量的数据组件具有预定序列;

当针对该算术运算侦测到至少一个例外状况时,该处理电路被配置为执行至少一个响应动作,该算术运算针对该预定序列中的第一有效数据组件执行;以及

当针对该算术运算侦测到该至少一个例外状况时,该处理电路被配置为抑制该至少一个响应动作且存储组件识别信息,该组件识别信息识别该第一输入向量的哪个数据组件是给定有效数据组件,该算术运算针对该预定序列中除该第一有效数据组件之外的该给定有效数据组件执行。

2.如权利要求1所述的设备,其中该至少一个响应动作包括更新状态寄存器以指示侦测到针对该第一有效数据组件的该至少一个例外状况。

3.如权利要求2所述的设备,其中该状态寄存器包括一个或多个例外状况指示位的组,该组例外状况指示位用于指示一个或多个对应类型的例外状况的出现,其中该组例外状况指示位对于每个类型的例外状况包括单一位。

4.如任意前述权利要求所述的设备,其中该至少一个响应动作包括触发例外处理常用程序的执行。

5.如权利要求4所述的设备,包括配置寄存器以存储配置信息,该配置信息规定是否触发用于一个或多个类型的例外状况的该例外处理常用程序的执行;以及

当该配置信息规定应针对该侦测到的类型的例外状况触发该例外处理常用程序时,该至少一个响应动作包括触发该例外处理常用程序的执行。

6.如任意前述权利要求所述的设备,其中在已经解决用于决定是否应处理该一个或多个有效数据组件的有关状况之前,该处理电路被配置为执行用于该第一输入向量的一个或多个有效数据组件的该算术运算。

7.如任意前述权利要求所述的设备,其中该第一输入向量与掩码有关,该掩码指示该第一输入向量的哪些数据组件是有效数据组件。

8.如权利要求7所述的设备,其中该处理电路响应于至少一个其他指令以基于该组件识别信息产生用于随后尝试以执行该第一类型的向量算术指令的新第一输入向量及新掩码中的至少一者。

9.如权利要求8所述的设备,其中该处理电路响应于该至少一个其他指令以产生该新掩码,其中该给定有效数据组件指示为该预定序列中的该第一有效数据组件。

10.如权利要求8所述的设备,其中该处理电路响应于该至少一个其他指令以产生该新第一输入向量,其中该新第一输入向量的该第一有效数据组件具有与该第一输入向量中的该给定有效数据组件相同的值。

11.如任意前述权利要求所述的设备,其中该组件识别信息包括组件识别掩码,该组件识别掩码包括:一个或多个指示,该指示具有用于在该预定序列中的该给定有效数据组件之前的至少一个数据组件的第一值;以及一个或多个指示,该指示具有用于在该预定序列中的该给定有效数据组件及任何后续有效数据组件的第二值。

12.如任意前述权利要求所述的设备,其中在该预定序列中的该第一有效数据组件包

括该第一输入向量的最低有效的有效数据组件。

13.如任意前述权利要求所述的设备，其中响应于第二类型的向量算术指令，该处理电路被配置为响应于侦测针对该算术运算的该至少一个例外状况执行该至少一个响应动作，该算术运算针对该第一输入向量的任何有效数据组件执行。

14.如权利要求13所述的设备，其中该第一类型及第二类型的向量算术指令包括不同运算码，或包括规定该向量指令是该第一类型还是第二类型的字段。

15.如权利要求13所述的设备，其中该第一类型的向量算术指令包括在第一模式的该处理电路中执行的向量算术指令；以及

该第二类型的向量指令包括在第二模式的该处理电路中执行的向量算术指令。

16.如权利要求13至15中任一项所述的设备，其中响应于该第二类型的向量算术指令，该至少一个响应动作包括更新状态寄存器以提供针对该算术运算侦测到的一个或多个类型的例外状况的指示，该算术运算针对该第一输入向量的任何有效数据组件执行。

17.如任意前述权利要求所述的设备，其中该算术运算包括浮点运算且该至少一个例外状况包括下列的至少一者：

在该算术运算中产生的浮点值的溢出；

在该算术运算中产生的浮点值的下溢；

在该算术运算中产生的浮点值是不准确的；

该算术运算对应于无效运算；

该算术运算包括零除；以及

到该算术运算的输入是非正规的。

18.如任意前述权利要求所述的设备，其中该算术运算包括饱和算术运算，且该至少一个例外状况包括在该饱和算术运算中产生的值的饱和度。

19.一种设备，包括：

用于响应于规定包括多个数据组件的至少第一输入向量的第一类型的向量算术指令，执行针对该第一输入向量的至少一个有效数据组件的算术运算的装置；

其中该第一输入向量的数据组件具有预定序列；

当针对该算术运算侦测到至少一个例外状况时，该用于执行的装置被配置为执行至少一个响应动作，该算术运算针对该预定序列中的第一有效数据组件执行；以及

当针对该算术运算侦测到该至少一个例外状况时，该用于执行的装置被配置为抑制该至少一个响应动作且存储组件识别信息，该组件识别信息识别该第一输入向量的哪个数据组件是给定有效数据组件，该算术运算针对该预定序列中除该第一有效数据组件之外的该给定有效数据组件执行。

20.一种数据处理方法，包括：

响应于规定包括多个数据组件的至少第一输入向量的第一类型的向量算术指令，执行针对该第一输入向量的至少一个有效数据组件的算术运算，其中该第一输入向量的数据组件具有预定序列；

当针对该算术运算侦测到至少一个例外状况时，执行至少一个响应动作，该算术运算针对该预定序列中的第一有效数据组件执行；以及

当针对该算术运算侦测到该至少一个例外状况时，抑制该至少一个响应动作且存储组

件识别信息，该组件识别信息识别该第一输入向量的哪个数据组件是给定有效数据组件，该算术运算针对该预定序列中除该第一有效数据组件之外的该给定有效数据组件执行。

21.一种在计算机可读存储介质上存储的计算机程序，当该计算机程序由数据处理设备执行时，提供虚拟机，该虚拟机提供对应于如权利要求1至18中任一项所述的设备的指令执行环境。

处理用于向量算术指令的例外状况

技术领域

[0001] 本技术关于数据处理的领域。更具体地，本技术关于处理向量算术指令。

背景技术

[0002] 一些数据处理设备可支持向量处理，其中可在向量的各数据组件上执行给定处理操作以产生结果向量的对应数据组件。这允许利用单一指令处理数个不同数据值，以减少处理给定数量的数据值所需的程序指令的数量。向量处理还可称为SIMD(单指令，多数据)处理。

发明内容

[0003] 至少一些示例提供一种设备，该设备包括：

[0004] 处理电路，用于响应于规定包括多个数据组件的至少第一输入向量的第一类型的向量算术指令，执行针对该第一输入向量的至少一个有效数据组件的算术运算；

[0005] 其中该第一输入向量的数据组件具有预定序列；

[0006] 当针对该算术运算侦测到至少一个例外状况时，该处理电路被配置为执行至少一个响应动作，该算术运算针对该预定序列中的第一有效数据组件执行；以及

[0007] 当针对该算术运算侦测到该至少一个例外状况时，该处理电路被配置为抑制该至少一个响应动作且存储组件识别信息，该组件识别信息识别该第一输入向量的哪个数据组件是给定有效数据组件，该算术运算针对该预定序列中除该第一有效数据组件之外的该给定有效数据组件执行。

[0008] 至少一些示例提供一种设备，该设备包括：

[0009] 用于响应于规定包括多个数据组件的至少第一输入向量的第一类型的向量算术指令，执行针对该第一输入向量的至少一个有效数据组件的算术运算的装置；

[0010] 其中该第一输入向量的数据组件具有预定序列；

[0011] 当针对该算术运算侦测到至少一个例外状况时，该用于执行的装置被配置为执行至少一个响应动作，该算术运算针对该预定序列中的第一有效数据组件执行；以及

[0012] 当针对该算术运算侦测到该至少一个例外状况时，该用于执行的装置被配置为抑制该至少一个响应动作且存储组件识别信息，该组件识别信息识别该第一输入向量的哪个数据组件是给定有效数据组件，该算术运算针对该预定序列中除该第一有效数据组件之外的该给定有效数据组件执行。

[0013] 至少一些示例提供一种数据处理方法，该数据处理方法包括：

[0014] 响应于规定包括多个数据组件的至少第一输入向量的第一类型的向量算术指令，执行针对该第一输入向量的至少一个有效数据组件的算术运算，其中该第一输入向量的数据组件具有预定序列；

[0015] 当针对该算术运算侦测到至少一个例外状况时，执行至少一个响应动作，该算术运算针对该预定序列中的第一有效数据组件执行；以及

[0016] 当针对该算术运算侦测到该至少一个例外状况时，抑制该至少一个响应动作且存储组件识别信息，该组件识别信息识别该第一输入向量的哪个数据组件是给定有效数据组件，该算术运算针对该预定序列中除该第一有效数据组件之外的该给定有效数据组件执行。

[0017] 至少一些示例提供一种在计算机可读存储介质上存储的计算机程序，该计算机程序当由数据处理设备执行时，提供虚拟机，该虚拟机提供对应于上述设备的指令执行环境。

附图说明

[0018] 本技术的其他方面、特征及优势将在示例的以下详细说明中显而易见，该详细说明将结合附图来阅读，其中：

[0019] 图1示意性地示出了支持向量处理的数据处理设备的示例；

[0020] 图2示出了浮点状态寄存器及浮点控制寄存器的示例；

[0021] 图3示出了向量算术指令的非第一故障(non-first-faulting)形式的示例；

[0022] 图4示出了向量算术指令的第一故障(first-faulting)形式的示例，针对此形式，出现用于向量的第一有效组件的例外状况；

[0023] 图5示出了向量算术指令的第一故障形式的示例，针对此形式，出现用于不是第一有效组件的组件的例外状况；

[0024] 图6示出了产生用于随后尝试以执行第一故障向量算术指令的新掩码的示例；

[0025] 图7示出了产生用于随后尝试以执行第一故障向量算术指令的新输入向量的示例；

[0026] 图8示出了用于非推测执行向量算术指令的伪码的示例；

[0027] 图9示出了用于推测执行向量算术指令的伪码的示例；

[0028] 图10示出了用于多次执行向量算术指令直至已处理全部有效组件的循环的示例；

[0029] 图11示出了处理向量算术指令的方法的流程图；

[0030] 图12示出了用于浮点状态寄存器的替代布置；

[0031] 图13示出了用于使用图12的状态寄存器布置处理向量算术指令的推测执行的技术；以及

[0032] 图14示出了虚拟机实施方式。

具体实施方式

[0033] 现在将描述一些具体示例。应了解，本发明不限于这些特定示例。

[0034] 处理电路可支持至少第一类型的向量算术指令，该类向量算术指令规定包括多个数据组件的至少第一输入向量。响应于第一类型的向量算术指令，该处理电路可执行用于第一输入向量的至少一个有效数据组件的算术运算。有时，可侦测到用于算术运算的例外状况，该算术运算针对第一输入向量的一个或多个有效数据组件执行。例如，例外状况可指示结果超出可由结果值表示的范围，或已经出现错误。

[0035] 可认为第一输入向量的数据组件具有预定序列。当侦测到用于算术运算的至少一个例外状况时，该算术运算针对在序列中的第一有效数据组件执行，则该处理电路可执行至少一个响应动作。然而，当侦测到用于算术运算的至少一个例外状况时，该算术运算针对

在序列中除第一有效数据组件之外的给定有效数据组件执行，则该处理电路可抑制至少一个响应动作且存储数据识别信息，该数据识别信息识别第一输入向量中的哪个数据组件是对其侦测到例外状况的给定有效数据组件。

[0036] 此方法可提供若干优点。首先，追踪已经侦测到的例外状况，且若必要，则执行至少一个响应动作，可需要在处理电路中提供某些资源。提供分别用于正在执行处理的各向量通道的这类资源，就电路区域而言及就管理这类资源的额外负担而言是高成本的。实际上，例外状况可比较少见且由此该额外负担是不合理的。通过当执行用于序列中的第一有效数据组件的例外状况时执行响应动作，但若其他数据组件触发该例外状况则抑制此响应动作，可利用对应于单一数据组件的单一组资源有效管理例外状况处理，而不需要复制用于向量的各数据组件的这些资源。

[0037] 当序列中后续组件触发例外状况时，组件识别信息用于利用故障组件再启动向量算术指令的执行，该故障组件现为第一有效数据组件。以此方式，执行算术指令的重复迭代可利用任何例外状况通过向量组件逐渐取得前向进度，正在使用用于各路径上的第一有效数据组件的单一组资源追踪该任何例外状况，从而消除对每向量通道多组资源的需求。

[0038] 例如，当侦测到用于第一有效数据组件的例外状况时，至少一个响应动作可包括更新状态寄存器以指示出现哪一个或多个例外状况。例如，状态寄存器可包括一组位，该组位指示不同类型例外状况且该响应可包括设定用于侦测到的类型的例外状况的适当位。通过执行用于第一有效数据组件而非触发例外状况的其他组件的响应，单一组例外状况指示位可以在完整向量的数据组件间共享(每种类型例外状况一个位)，且并非必需提供每通道多组状态位以追踪针对每个数据组件出现的例外状况。这可大幅度降低硬件的复杂性，不仅仅允许使用较小状态寄存器，而且降低状态标记的数量，该状态标记沿处理电路中的数据路径传送。

[0039] 此外，响应动作可包括触发例外处理常用程序的执行。例如，当某些例外状况出现时，则这可触发到操作系统或其他控制程序的俘获(trap)，该系统采取措施以处理例外状况。因此，若侦测到对于序列中第一有效数据组件而非对于其他组件的例外状况，则可执行例外处理常用程序。

[0040] 在一些实施方式中，若侦测到用于序列中第一有效数据组件的某些例外状况，则某些例外状况通常触发例外处理常用程序。或者，可提供配置寄存器以规定配置信息，该配置信息指示是否触发用于不同类型的例外状况的例外处理程序的执行。在此情形中，仅当在配置寄存器中的配置信息规定应触发用于侦测到的类型的例外状况的例外处理常用程序时，则响应可包括触发例外处理常用程序的执行。

[0041] 一些系统可支持向量运算的推测执行，其中给定状况决定是否需要处理向量的某些组件，但在已实际解决有关状况之前可执行此向量指令。在一些情形中，有关状况可甚至取决于执行用于向量的一些组件的向量指令的结果，以决定是否应实际处理其他组件。允许此推测执行可使与系统相比向量化代码更为有效或实际，该系统需要精确知道在执行指令之前处理哪些组件。例如，当向量处理用于并行处理相同程序循环的不同迭代及向量操作数的不同组件时这通常是有用的，向量操作数的不同组件对应于在相同循环的不同迭代中使用或产生的值。

[0042] 然而，若向量算术指令的推测执行是可能的，当随后证实一旦已经解决有关条件，

则不应处理第一输入向量的一些数据组件时，则一个结果可为推测地处理第一输入向量的一些数据组件。若这些组件导致正在侦测的例外状况，则当实际上甚至不应首先执行此操作时，这导致采取响应动作(例如，更新状态寄存器或执行例外处理常用程序)。不期望引起归因于向量处理的不正确推测通道的这类副效应。

[0043] 一种方法可用以维持两种不同版本的状态寄存器，一种推测版本状态寄存器及一种非推测版本状态寄存器，且一旦已经确定处理的推测执行通道，则随后基于推测版本更新非推测版本。然而，就硬件而言此举是更为高成本，由于需要额外寄存器以追踪推测及非推测版本状态寄存器，以及一旦已经解决推测，则需要待执行的额外指令以解决经确定版本的状态指示。通过使用上文论述的技术，可避免此增加的额外负担，因为一般而言，在序列中的第一有效数据组件将为处理的非推测执行通道，且由此响应于针对此组件侦测到的例外状况执行响应动作是安全的。针对输入向量的其他组件，即使这些组件产生例外状况，这也不触发响应动作，由此若随后证实处理的这些通道是误推测，则不产生不利副效应。因此，上述技术还有助于使推测执行向量运算的处理更为有效。

[0044] 可以不同方式决定输入向量的有效组件。在一些情形中，可认为输入向量的所有组件是有效组件，由此不存在定义哪些数据组件是有效或非有效的任何数据。即，输入向量是非述语向量。在此情形中，第一有效组件可仅为向量的第一组件(例如，若将预定序列处理为从最低有效组件延伸至最高有效组件，则为最低有效组件)。

[0045] 然而，更灵活技术可为将输入向量与掩码关联，该掩码指示哪些数据组件是有效或非有效数据组件。在此情形中，第一有效数据组件可为除输入向量的第一组件之外的组件。

[0046] 若侦测到用于在序列中除第一有效数据组件之外的给定有效数据组件的例外状况，则组件识别信息可用于修改有关用于后续尝试的向量算术指令的掩码且执行指令或修改用于后续尝试的输入向量本身。可更新掩码使得触发例外状况的给定组件成为在序列中的第一有效数据组件(例如，在前组件现可指示为非有效)，或可修改在第一输入向量中的组件的位置使得目前新的第一输入向量的第一有效数据组件具有用于先前尝试以执行该指令的如在第一输入向量中的给定有效数据组件的相同值。举例而言，程序设计器可包括在围绕向量算术指令的循环中的一个或多个指令，以检查组件识别信息来了解是否成功处理全部组件，该组件识别信息由一个用以执行向量算术指令的尝试产生，且若识别给定有效数据组件的组件识别信息已触发例外状况，则在循环返回以用于另一尝试之前可修改掩码或输入向量。在随后尝试中，给定有效数据组件现在可为第一有效数据组件且由此若例外状况仍产生，则可采取响应动作以允许处理例外状况。因此，指令可使用用于处理例外状况的仅单一组资源通过输入向量的组件逐步取得前向进度。

[0047] 在基于组件识别信息更新第一输入向量的情形中，这可通过转换第一输入向量达成，使得给定有效数据组件成为第一有效数据组件(且在给定有效数据组件之后的任何组件还可沿在对应组件中的位置移动至给定有效数据组件)。或者，若向量运算指令在利用从高速缓存或存储器加载的数据填充第一输入向量(或用于产生第一输入向量的较早向量)的较早向量加载指令之后，则可通过将向量加载指令的地址调整达对应于在如由组件识别信息指示的原始向量中的给定有效数据组件位置的量，且重复向量加载指令以及向量算术指令本身来有效更新第一输入向量。以此方式，下次执行向量算术指令时，向量加载指令将

使数据加载对应于第一有效组件的向量中的位置中，该数据对应于先前故障数据组件。

[0048] 可以不同方式表示组件识别信息。在一些情形中，该组件识别信息单纯为给定有效数据组件的组件数量的指示，该给定有效数据组件触发例外状况。然而，特别有用的表示可用以提供组件识别掩码，该组件识别掩码包括：一个或多个指示，该一个或多个指示具有用于在序列中的给定有效数据组件前的至少一个数据组件的第一值；及一个或多个指示，该一个或多个指示具有用于序列中的给定有效数据组件及任何后续有效数据组件的第二值。因此，组件识别掩码可基本上将向量分为已经处理且不具有故障的部分以及可仍导致故障的部分。此组件识别掩码可启用用于随后尝试的新掩码以更有效执行产生的指令。此外，由于存在正在执行以实施一系列数据处理运算的若干成功指令，此类型的组件识别掩码是有用的；及若这些指令的任一者遇到用于给定组件的例外状况，则期望停止在随后指令中正在执行的处理的相应通道。因此，上文论述的类型的组件识别掩码可用于产生还用于随后指令的掩码。

[0049] 用于输入向量的组件的预定序列可为任意序列。然而，通常在序列中的第一有效数据组件为第一输入向量的最低有效的有效数据组件可为方便的。在序列中的随后组件则可对应于输入向量的数据组件，以增加有效值直至最高有效的有效数据组件。此方法可最佳映射至其中实际写入向量化代码的方式。然而，可使用其他序列，诸如起始于最高有效的有效数据组件并结束于最低有效组件的序列。

[0050] 若在序列中除第一有效数据组件之外的超过一个有效数据组件遇到例外状况，则在一些情形中，组件识别信息可识别这些组件中的每一个。或者，组件识别信息可仅识别触发例外状况的序列中的下一个组件(在第一有效数据组件之后)，且即使后续组件还可导致例外状况，该组件识别信息还不可识别任何其他组件。

[0051] 在一些实施方式中，该处理电路可通常以上文论述的方式处理向量算术指令，其中仅当第一有效数据组件触发例外状况时采取响应动作。

[0052] 然而，其他实施方式还可支持第二类型的向量算术指令，针对此第二类型的向量算术指令，该处理电路响应于侦测用于该向量的任何有效数据组件的例外状况执行至少一个响应动作，无论是否侦测到用于第一有效组件或后续组件的例外状况。第一及第二类型的向量算术指令可为完全不同的指令(例如，具有不同运算码)，或可对应于具有字段的通用指令运算码，在指令编码时该通用指令运算码规定该指令是第一类型还是第二类型。或者，第一及第二类型的向量算术指令具有相同编码，但第一类型的向量算术指令可为在第一模式的处理电路中执行的向量算术指令，而第二类型的向量指令可为在第二模式的处理电路中执行的向量算术指令。例如，存在规定该处理电路当前是第一模式还是第二模式的控制寄存器。

[0053] 提供第一及第二类型两者的向量算术指令可用于启用程序设计器以取决于待执行的代码性质在替代类型的指令的间选择，其中仅当第一有效组件遇到例外状况时第一类型的向量算术指令触发响应动作，而第二类型可触发用于任何有效组件的响应动作。例如，若代码需要如上文论述的向量运算的推测执行，则可选择第一类型以避免响应于向量通道不经意地触发副效应，该向量通道最终不需要以任何方式执行。在另一方面，针对非推测代码，可选择第二类型的向量算术指令以改善性能，由于第二类型的向量算术指令较不可能需要若干迭代以通过向量的各组件取得前向进展。因此，提供两种类型的指令可提供在正

确行为与性能间的更佳平衡。

[0054] 在一些实施方式中，针对第二类型的向量算术指令，侦测到对于第一输入指令的任何组件的例外状况的响应可包括记录对哪些组件触发例外状况及侦测到哪些例外类型的精确指示。然而，这可能需要大量额外负担。实际上，例外状况可为少见的且由此该额外负担可为不合理的。实际上，第二类型的向量算术指令可触发响应动作，该响应动作包括更新状态寄存器以提供任何类型的例外状况的指示，该例外状况针对输入向量的任何有效数据组件出现，而不区分哪些特定组件触发哪些特定类型的例外状况。在众多情形中，这足够允许例外处理常用程序(例如，诸如操作系统)以决定如何解决该例外状况。

[0055] 该例外状况可为由算术运算触发的任何类型的状况，该算术运算指示需要进一步研究的一些异常结果或性质。本技术特别用于其中算术运算包括浮点运算的指令。用于浮点算术的IEEE 754标准规定数个例外状况，应针对浮点运算追踪这些例外状况。例如，这些例外状况可包括下列情况的一个或多个者：溢出、下溢、不准确、无效运算或零除。可针对浮点运算而追踪的另一类型的例外状况可为至给定算术运算的输入是否是非正规的(即，其有效数字起始于至少一个前导零，而非起始于1，如对于正规浮点值而言)。由于存在待追踪的相对大量的例外状况，记录用于向量的每个通道的这些例外状况应是非常高成本的，且由上文论述的技术可通过允许单一组位在向量通道间共享来大幅度节省硬件资源，一个位用于每一类型的例外状况。

[0056] 本技术还可用于非浮点运算。例如，针对整数运算，一些指令可执行饱和算术运算，其中限制算术运算的结果以位于某些最小及最大界限的范围中。在此情形中，当在饱和算术运算中产生的值的饱和度出现时，可产生例外状况。即，若算术运算产生大于饱和度的最大界限或小于最小界限的结果，则可触发例外状况。

[0057] 例如，响应于向量算术指令执行的算术运算可为加法、减法、乘法、除法、乘法-加法、乘法-减法、平方根等等。算术运算还可为用以将给定数据值的表示转换为不同形式的转换运算，诸如在浮点值的不同精确度间转换或在浮点值与整数值间转换。针对一些指令(诸如转换指令)，第一输入向量可仅为该指令的输入向量。针对其他指令，诸如加法、减法或乘法指令，还存在第二输入向量，该第二输入向量的数据组件将与第一输入向量的数据组件合并。一些指令甚至可满足三个或三个以上输入向量(例如，乘法-加法)。

[0058] 本技术还可使用虚拟机实施。虚拟机可为程序，当由主机设备执行时该虚拟机提供用于执行指令的指令执行环境，使得该等主机设备由程序设计师的观点显示，如同其具有上文论述的电路。主机处理器实际上本身不需要具有该电路，而是虚拟机的代码控制主机硬件以执行如同提供此电路的指令。例如，虚拟机可为在存储媒体上存储的计算机程序。该存储媒体可为非暂时性的。

[0059] 图1是其中可采用所述实施例的技术的系统的方块图。在图1中所示的示例中，该系统采取管线式处理器的形式。通过撷取电路10从指令高速缓存15(通常经由一个或多个其他阶层的高速缓存，诸如2阶高速缓存50耦接至存储器55)来撷取指令，其中经由解码电路20自撷取电路传递这些指令，此解码电路20解码各指令以产生适当控制信号，该控制信号用于控制在管线式处理器中的下游执行资源以执行该指令所需的运算。将形成经解码指令的控制信号传递至发布阶段电路25，用于发布至管线式处理器中的一个或多个执行管线30、35、40、80。可将执行管线30、35、40、80共同地认为形成处理电路。

[0060] 发布阶段电路25已存取寄存器60，其中存储由运算所需的数据值。更具体地，用于向量运算的源操作数可存储在向量寄存器65中，且用于纯量运算的源操作数可存储在纯量寄存器75中。此外，一个或多个述语(掩码)可存储在述语寄存器70中，当执行某些向量运算时用作用于处理的向量操作数的数据组件的控制信息。一个或多个纯量寄存器还可用于存储用以导出此控制信息的数据值，此控制信息在某些向量运算的执行期间使用。

[0061] 寄存器60还可包括用以提供各种控制信息的数个控制寄存器76，该控制信息诸如用于控制处理管线的操作的配置信息，或指示在处理或指令结果性质期间产生的状况的状态信息。例如，控制寄存器76可包括浮点配置寄存器(FPCR)77、浮点状态寄存器(FPSR)78及第一故障寄存器(FFR)79，这些寄存器将在下文更详细描述。

[0062] 源操作数及任何有关控制信息可经由路径47路由至发布阶段电路25，使得可将其以及识别待执行以实施各经解码指令的运算的控制信号发送至适当执行单元。假设图1示出的各执行单元30、35、40、80是用于在向量操作数上运算的向量处理单元，但若期望处理由设备支持的任何纯量运算，则可提供分离的执行单元(未示出)。

[0063] 考虑到各向量运算，例如，可将算术运算以及所需源操作数(及诸如述语的任何控制信息)转发到算术逻辑单元(ALU)30，以启用在那些源操作数上执行的算术或逻辑运算，其中该结果值通常输出为目的操作数，该目的操作数用于存储在向量寄存器组65的规定寄存器中。

[0064] 除ALU 30外，可提供浮点单元(FPU)35以用于响应于经解码的浮点指令执行浮点运算，且可提供向量置换单元80以用于在向量操作数上执行某些置换运算。此外，加载/存储单元(LSU)40用于执行加载运算以将数据值从存储器55(经由数据高速缓存45及任何中间其他阶层的高速缓存，诸如2阶高速缓存50)加载至寄存器组60中的规定寄存器中，且用于执行存储运算以将数据值从那些寄存器存储回存储器55。应了解，还可提供未在图1中示出的其他类型的执行单元。

[0065] 图1示出的系统可为有序处理系统，其中按程序顺序执行指令序列；或者该系统可为无序系统，允许其中出于尝试改善性能的目的将各种指令执行以重新排序的顺序。本领域技术人员应理解，在无序系统中，可提供额外结构(未在图1中明确示出)，例如将由指令规定的架构寄存器映射至来自寄存器组60中的一组实体寄存器的实体寄存器(该组实体寄存器通常大于架构寄存器的数量)的寄存器重命名电路，由此实现移除某些风险，促进无序处理的更多使用。此外，通常可提供重排序缓冲器以保持追踪无序执行，且允许按序确认各指令的执行结果。

[0066] 在描述的实施例中，布置图1的电路以在向量寄存器65中存储的向量操作数上执行向量运算，其中向量操作数包括多个数据组件。针对在此向量操作数上执行的某些向量运算(诸如算术运算)，可将所需运算并行(或迭代)应用至向量操作数中的各种数据组件。述语信息(还称为掩码)可用于识别在向量中的哪些数据组件是用于特定向量运算的有效数据组件，并因此该数据组件是运算将要应用到的数据组件。

[0067] 图2更详细示出了FPSR 77及FPSR 78的示例。如图2中所示，FPSR78可包括数个例外指示位85，这些例外指示位各自对应于不同类型的例外状况且指示自从最终清除位85起，出现对应类型的至少一个例外状况。在此示例中，在FPSR 78中指示的例外状况包括：

[0068] ●QC：整数饱和，若饱和整数运算结果超过最大界限或小于用于饱和度的最小界

限，此例外状况则出现；

[0069] ●IDC：非正规输入，若至浮点运算的输入值是非正规，此例外状况则出现；

[0070] ●IXC：不准确，若浮点运算结果不能使用针对输出规定的浮点格式准确表示，此例外状况则出现；

[0071] ●UFC：下溢，当浮点运算结果小于可使用针对输出规定的浮点格式表示时，此例外状况出现；

[0072] ●OFC：溢出，若浮点运算结果大于可使用针对输出规定的浮点格式表示，此例外状况则出现；

[0073] ●DZC：零除，若尝试零除，此例外状况则出现；

[0074] ●IOC：无效运算，若尝试执行无效运算，此例外状况则出现。

[0075] 应了解，还可指示其他类型的例外状况。在某种意义上累计例外指示位85，这些例外指示位一旦设定将保持设定，直至由处理管线执行的指令明确清除FPSR 78的位，或整体清除FPSR为止。因此，当出现例外状况导致设定对应位时，不触发例外状况的指令将不清除对应位。

[0076] FPCR 77包括数个例外掩码位87，各位对应于在FPSR 78中指示的类型的例外状况的一者。例外掩码位控制在FPSR 78中的各类型的例外状况是否将触发至操作系统以处理该类型例外的俘获。例如，在图2的情形中，该系统当前经配置使得零除或无效运算例外状况将触发对操作系统的俘获，而其他种类的例外状况将不会如此触发。应了解，未在图2中示出了的其他信息还可存储在FPCR及FPSR 78中。

[0077] 在此实施例中，提供在用于给定向量运算的全部向量通道中共享的单一FPSR 78，该FPSR 78具有每种类型的例外的单一例外指示位85。因此，FPSR 78实际上是纯量FPSR且不包括指示分别用于各向量组件的例外的每通道位。

[0078] 图3示出了由浮点单元35支持的非第一故障形式的向量算术指令的示例。在此示例中，待执行的算术运算是浮点加法运算。该指令将在向量寄存器65中存储的一对向量Za、Zb，及在述语寄存器70中存储的述语Pg作为输入，述语Pg提供识别输入向量Za、Zb的哪些组件是有效的述语值(或掩码)。在图3的示例中，掩码指示Za及Zb的组件0至3是有效的且组件4至7是非有效的。因此，响应于向量算术指令，浮点单元35执行在寄存器Za、Zb的通道0至3中的对应组件X、Y的浮点加法以在结果寄存器Zr中产生对应结果组件Z0至Z3。在另一方面，在结果寄存器的非有效通道中，组件Z4至Z7采用独立于针对那些通道另外执行的加法结果的值。例如，Z4至Z7可设定为诸如零的预定值，可直接映射至在Za、Zb的对应通道中的输入值的一者，或可保持先前存储在结果寄存器Za的对应部分中的值。因此，仅将有效通道设定为取决于算术运算结果的值，该算术运算在输入的对应组件上执行。

[0079] 针对非第一故障形式的向量算术指令，若出现对于向量的任何有效通道的任何例外状况，则设定在FPSR 78中的对应位。有效地，可使用OR运算合并由各有效通道触发的例外状况的指示以由此设定FPSR 78。例如，在图3中，组件Z0触发不准确例外IXC而组件Z3触发溢出例外PFC，且由此可在FPSR 78中设定对应此等类型的例外的位85。若FPCR77指示此等类型的例外应触发对操作系统的俘获，则实施该俘获且该操作系统可执行用于处置这些类型的例外的例外处理常用程序。应注意，针对在图3中示出的非第一故障形式指令，FPSR不区分哪些特定组件触发该例外状况。

[0080] 图4示出了不同第一故障形式的向量算术指令。算术运算本身是与图3相同，其中执行用于各通道的输入向量Za、Zb中的对应组件的浮点加法，对于该各通道的在述语寄存器Pg中的掩码的对应位是1。此时掩码的所有组件为一，且由此针对此示例所有该等组件是有效的。在此示例中，在图4中示出的第一故障形式的指令具有与在图3中示出的非第一故障形式的指令相比不同的运算码。或者，不同形式的指令可共享相同运算码，但可包括规定正在执行哪种形式的指令的字段。此外，在一些情形中，非第一故障形式及第一故障形式的指令的编码可为完全相同，但该指令是被视为第一形式还是第二形式可取决于处理电路的当前操作模式。例如，在FPCR 77或其他控制寄存器76的一者中可存在位，该控制寄存器76设定处理器模式以指示向量算术运算应该根据图3的非第一故障形式还是图4的第一故障形式执行。

[0081] 针对在图4中示出的第一故障形式的指令，与图3中非第一故障形式相比不同地处理例外状况。如在图4中示出，若出现对于算术运算的例外状况，该算术运算在向量的第一有效组件(在此示例中为最低有效组件X0/Y0/Z0)中执行，则采用适当例外处理响应。举例而言，这可包括设定在FPSR 78中的对应位；及取决于FPCR是否实现俘获此类例外，视需要俘获该操作系统。即使侦测到用于向量的其他组件的例外状况，未采取针对此举的响应且不基于任何例外状况来更新FPSR 78，出现用于除向量的第一有效组件之外的任何组件的该等例外状况。举例而言，在图4中，尽管针对包括对应于组件X3/Y3/Z3的通道侦测到溢出，还不设定FPSR 78的溢出位。

[0082] 在另一方面，如在图5中示出，针对第一故障形式的指令，若第一有效组件不触发例外状况，则抑制故障处理响应。因此，FPSR 78保持未经修改且不存在对操作系统的俘获。然而，若对于在向量中不为第一有效组件的组件的算术运算触发例外状况，则如在图5中示出，更新第一故障寄存器79以记录哪个组件触发该例外。如在图5中示出了，第一故障寄存器可对应于故障掩码，其中与触发故障的组件3相比较低有效的组件0至2指示为具有位值1，且针对触发故障的组件3及任何后续组件，在FFR 79中的掩码位为0。

[0083] 若需要，FFR 79提供信息，该信息由后续指令使用以设定用于另一尝试以执行向量算术指令的掩码或输入向量。例如，如在图6中示出，FFR 79可与先前掩码Pg合并以产生新掩码Pg'，该新掩码Pg'表示可仍触发尚未处理的例外状况的剩余组件。举例而言，此举可由清除等于1的原始掩码Pg的任何位决定，FFR 79的对应位还为1。若原始掩码Pg均为1，则产生新掩码的另一方式可为自原始掩码Pg减去FFR 79。

[0084] 或者，如在图7中示出，FFR 79可用于修改用于向量算术指令的输入向量Za，以便将在FFR 79中由零指示的组件X3至X7再次加载对应于输入向量的第一有效组件的向量位置0至4。可执行用于任何其他输入向量Zb的相似运算，且随后重复向量算术指令使得目前组件X3将是第一有效组件，且若此举触发例外状况，则可将此举记录在FPSR中，且若需要的话则触发此举。

[0085] 因此，使用向量算术指令的软件可提供围绕向量算术指令的循环，该向量算术指令检查FFR 79以指示正确执行且无例外状况的哪些组件且若需要重复起始于有效组件的指令，该有效组件先前触发例外状况。在下文图9中示出了此循环的示例。

[0086] 此方法具有若干优点。首先，此技术实现精确记录对于向量的各组件出现的例外状况，包括追踪触发该例外状况的哪个特定组件，使用指示在FPSR 78中的位的仅单一组例

外，以避免需要提供用于各向量通道的位85的分离复本，由于此举不仅需要较大FPSR 78而且需要沿管线向下的浮点例外标记的多个复本，所以当向量变大时该分离复本是非常高成本的。

[0087] 此外，来自指令的第一故障可用于支持推测执行向量运算。这关于图8及图9更详细地进行了描述。图8示出非推测执行给定向量算术指令的示例，而图9示出了推测执行该指令的示例。

[0088] 若给定组的处理运算需要应用至在数据值数组中的各个值，以产生用于数组各个值的对应结果，则向量指令可为特别有用。通常，可通过提供向量加载指令向量化代码，该向量加载指令将数个待处理的数据值从存储器加载到向量寄存器中，并随后在执行向量存储指令以将结果值存储回存储器之前，一个或多个向量算术指令以一些方式处理经加载向量的各数据组件以产生结果。然而，待处理的数据值的总数通常可能不是由处理电路支持的向量长度的准确倍数。通常，向量化代码可包括循环，该循环经由包括向量加载指令及数个向量算术指令的一组指令重复迭代，其中各循环迭代处理对应于向量长度的给定区块数据值。通常，可在循环中定义某种停止条件以检查是否已经处理全部所需数据值；且若已处理，则中断该循环。可分别针对当前正在处理的向量的各组件检查停止条件，且若决定停止条件满足组件的一者，则这可指示实际上不需要处理向量的后续组件。

[0089] 针对一些向量指令，停止条件可独立于该向量指令的结果。如图8示出，针对产生基于输入向量Za、Zb的结果向量Zr的给定向量加法指令，对应停止条件可取决于独立于加法结果的一些其他向量Zc。在此情形中，由于已知应无关该条件的结果处理其组件，可认为向量加法指令是非推测的。

[0090] 在其他方面，如图9所示，针对其他示例，停止条件可取决于向量加法指令的结果；举例而言，在此情形中，停止条件检查由向量加法指令产生的结果向量的任何组件j是否满足一些条件，且若满足则中断此循环。若结果向量Zr的组件j中的一者满足该条件，则这将指示不应已处理任何后续组件j+1、j+2等等。替代方法应执行进一步指令，这些指令在实际非推测地执行向量运算之前检查Zr值为多少，但这需要额外指令及更不易于向量化代码。通过在已知是否实际需要全部其组件之前，允许推测地执行向量加法指令，可改善性能。然而，这可随后导致一情况，其中若执行非第一故障形式指令，则可基于例外状况更新FPSR 78，该例外状况针对向量处理的推测通道出现，随后证实不需要该向量处理。在此情形中，推测执行可导致不需要的副效应，该副效应指示假例外并可能非必要地俘获至该操作系统。

[0091] 此情况可通过实际执行如关于图4及图5论述的第一故障形式的指令来避免。向量的第一有效组件一般是非推测组件，由于即使该组件满足停止条件，其仍为有效结果。因此，在响应于针对第一有效组件触发的例外状况触发故障处理响应时，不存在问题。若任何后续组件还触发例外状况，则这不触发响应；且若证实这些组件是非推测，则这不导致任何不需要的副效应。为了当未满足停止条件时允许正确进行该指令，如在图9中所示，若FFR 79指示针对向量的任何更多组件存在待执行的任何剩余运算，则设定围绕向量算术指令的循环以再次尝试该指令。举例而言，在各个尝试执行向量加法指令102之前，指令100可设定FFR的所有位。若已解决该状况，且若需要，则中断该循环(指令104)，随后若未满足该状况，则指令106修改由向量算术指令102使用的掩码Pk以指示为有效，该等剩余组件起始于触发

该例外的组件。此举基于如图6的示例中的先前掩码70及FFR 79来决定。若存在仍待处理的至少一个剩余组件，即，若新掩码具有除其中全部位为零的值以外的任何值，分支指令108则分支回到指令100。

[0092] 图10示出了以此方式围绕向量算术指令的多个迭代的循环的示例。在第一迭代上，未发生用于第一有效组件的例外，在此示例中第一有效组件是向量索引[1]，但于向量索引[2]侦测到用于下一有效组件的例外。由于组件[2]不为第一有效数据组件，所以未采取例外响应，且实际上更新FFR 79以清除对应于组件[2]及[3]的位。FFR 79随后用于产生对于围绕该循环的第二迭代的新述语，其中此时第一有效组件是通道[2]，且由此此时例外触发设定在FPSR 78中的标记，且若FPCR 77需要则触发对操作系统的俘获。相似地，若需要则可执行此指令的进一步迭代以处理在序列的后续组件中的进一步例外。

[0093] 实际上，浮点例外是较为少见的且通用软件非常少见地依赖于浮点状态标记值，由此当浮点例外产生时引入一些串行化的解决方法就性能而言是完全可以接受的。当需要推测执行向量处理时，或若精确追踪向量的哪些特定组件触发例外状况非常重要，则程序设计器选择第一故障形式的指令。在此情形中，通过通过指令的多个循环迭代以处理在连续有效组件上逐步产生的例外，精确例外处理是可能的，且可避免由非推测通道触发的假例外处理。

[0094] 在另一方面，若当已经得知需要处理全部有效组件而正在非推测地执行向量运算时，则可选择非第一故障形式的指令以改善性能，且避免需要提供围绕该向量指令的循环的额外指令。相似地，若不需要精确识别触发例外的特定通道，则可选择非第一故障形式的指令以仅在FPSR 78中记录针对任何通道出现的例外类型，而无需记录触发该例外的特定通道。

[0095] 当上文示例显示其中向量算术指令是加法指令的情形时，应了解，可执行相似技术至不同类型的算术运算(包括浮点运算及整数运算)的范围。

[0096] 图11是示出了处理向量算术指令的示例的流程图。在步骤200中，处理管线侦测当前指令是否为向量算术指令，且若不是，则以一些其他方式处理该指令。当执行向量算术指令时，则在步骤202中管线决定指令类型。若该指令是第一类型(第一故障形式的指令)，则在步骤204中执行用于输入向量的各有效组件的对应算术运算以产生结果向量的对应结果组件。在步骤206中，决定是否侦测到用于向量的第一有效组件的例外状况。若侦测到，则在步骤208中，更新FPSR 78以指示出现例外状况；且在步骤210中，检查FPCR 77以决定是否当前实现了用于侦测到的例外状况的对操作系统的俘获；且若如此，则触发俘获。在另一方面，若未出现用于第一有效组件的例外状况，则在步骤212中，该处理电路决定是否出现用于不为第一有效组件的另一有效组件的例外状况。若出现，则在步骤214中，针对有效组件及任何后续组件清除FFR 79的一个或多个位，该有效组件触发例外状况。在另一方面，若未出现用于任何组件的例外状况，则跳过步骤214。随后，停止处理当前向量算术指令。如上文所论述，其他指令可使用FFR 79以修改至向量算术指令的掩码或输入向量，且若尚未利用完全处理的例外状况来处理所有组件，则随后分支返回以再次尝试向量算术指令。

[0097] 在其他方面，若当前向量算术指令是第二类型(非第一故障形式的指令)，则在步骤220中，执行针对各有效组件的算术运算以产生结果向量的对应结果组件。在步骤222中，更新FPSR 78以指示对于向量的任何有效组件出现的任何例外状况。这不区分触发该例外

的特定组件。在步骤224中，该处理电路再次检查FPCR以决定针对出现的任何例外状况是否需要对操作系统的俘获，且若需要，则俘获至该操作系统。指令的处理结束。

[0098] 图12及13示出了用于处理推测更新至浮点例外标记的替代方法。除了提供如上文论述的第一及第二形式的向量算术指令，可提供一类向量算术指令，该类向量算术指令记录针对在第一版本的FPSR(FPSR1 78-1)中的任何向量通道出现的例外状况，该版FPSR对应于向量指令的推测处理。一旦已经解决控制该推测是否正确的有关状况，则可执行确定指令以将用于任何正确推测通道的浮点例外标记复制至第二版本的浮点状态寄存器FPSR2 78-2，该版本的寄存器表示FPSR的非推测状态(或经确定版本)(参见图13)。例如，确定指令可利用OR运算合并FPSR 1 78-1与FPSR 2的先前值以产生用于FPSR 2的更新状态。若需要，合并指令可随后用于将状态标记的向量视图转换为在FPSR 78中的常规纯量视图，其中每类型例外的单一标记在整个向量间共享，该标记指示针对向量整体出现的例外。例如，合并指令可OR在FPSR2 78-2的各通道中的用于给定例外的对应标记以产生总体标记，该总体标记指示针对向量整体是否出现该类型的例外。

[0099] 图12示出了用于FPSR 1 78-1及FPSR 2 78-2的示例性布置，FPSR 178-1及FPSR 2 78-2用于在字节宽度标记组件中捕获浮点及整数饱和状态，该标记组件对准导致浮点例外或整数饱和出现的任何数据组件的最低有效字节。由于最小浮点组件通常为32位宽而整数组件可为8位宽，这表示浮点状态标记85仅出现在数量4的倍数的标记组件中，然而整数饱和标记QC出现在各字节中。不需要由硬件存储未使用的位，且当读取时这些位返回零。在寄存器中的位可接受在有效向量组件上的Shoji浮点或饱和整数运算的副效应的一者。在图12中示出FPSR 1及FPSR 2的最低有效32位，但这可重复直至最大向量长度。

[0100] 图14示出了可使用的虚拟机实施方式。尽管前文描述的实施例根据用于操作支持相关技术的具体处理硬件的设备及方法来实施本发明，但还有可能提供硬件组件的所谓虚拟机实施方式。这些虚拟机实施方式在主机处理器530上执行，该主机处理器530执行支持虚拟机程序510的主机操作系统520。通常，需要大型高效处理器以提供按合理速度执行的虚拟机实施方式，但此方法在某些环境中可为合理的，诸如当需要执行另一处理器本机的代码以获得兼容性时，或出于再使用的原因。虚拟机程序510向应用程序500提供应用程序编程接口，该接口与将由实际硬件提供的应用程序编程接口相同，该实际硬件是通过虚拟机程序510模型化的装置。由此，程序指令(包括上述对存储器存取的控制)可通过使用虚拟机程序510而在应用程序500内执行，以模型化指令与虚拟机硬件的交互。

[0101] 在本申请中，词语“被配置为”用于表示设备的组件具有能够进行经定义的操作的配置。在此上下文中，“配置”表示硬件或软件互连的布置或方式。举例而言，设备可具有提供经定义的操作的专用硬件，或处理器或其他处理装置可经程序化以执行功能。“被配置为”不暗示需要以任何方式改变设备组件以便提供经定义的操作。

[0102] 尽管已在本文中参考附图详细描述了本发明的说明性实施例，但应理解，本发明不限于那些精确实施例，且本领域技术人员可在本发明中实现各种变化及修改，而不偏离如由所附权利要求所定义的本发明的范围及精神。

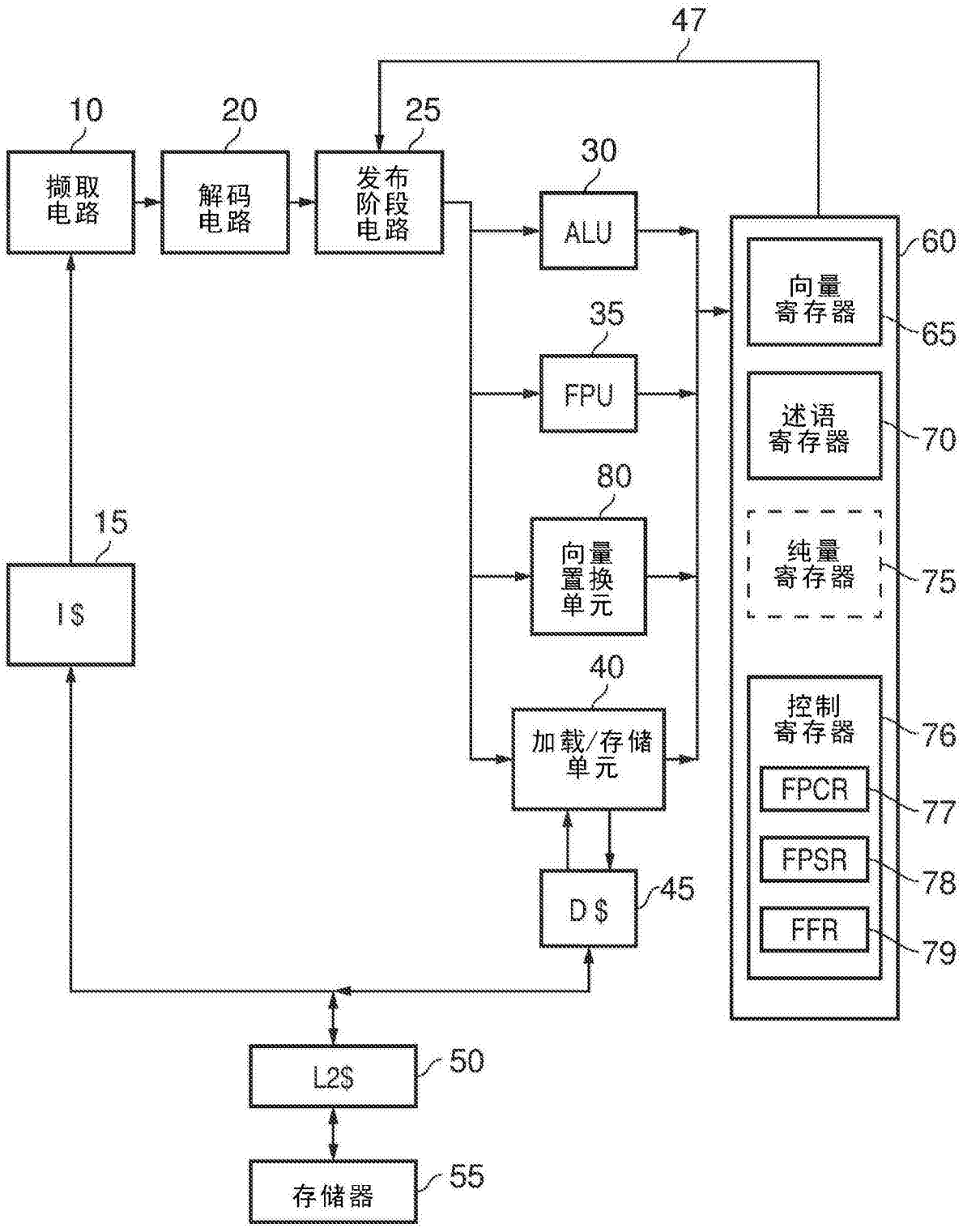

47
10
20
25
30
撷取电路
解码电路
发布阶段电路
ALU
60
向量寄存器
65
35
FPU
述语寄存器
70
80
向量置换单元
15
纯量寄存器
75
I$
40
加载/存储单元
控制寄存器
76
FPCR
77
FPSR
78
FFR
79
D$
45
L2$
50
存储器
55

图1

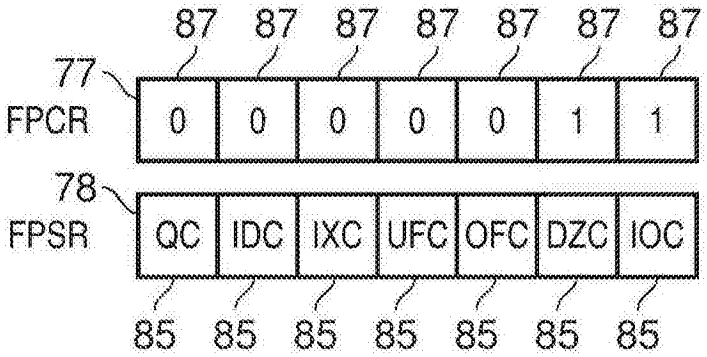

87 87 87 87 87 87 87
77
FPCR
0 0 0 0 0 1 1
78
FPSR
QC IDC IXC UFC OFC DZC IOC
85 85 85 85 85 85 85

图2

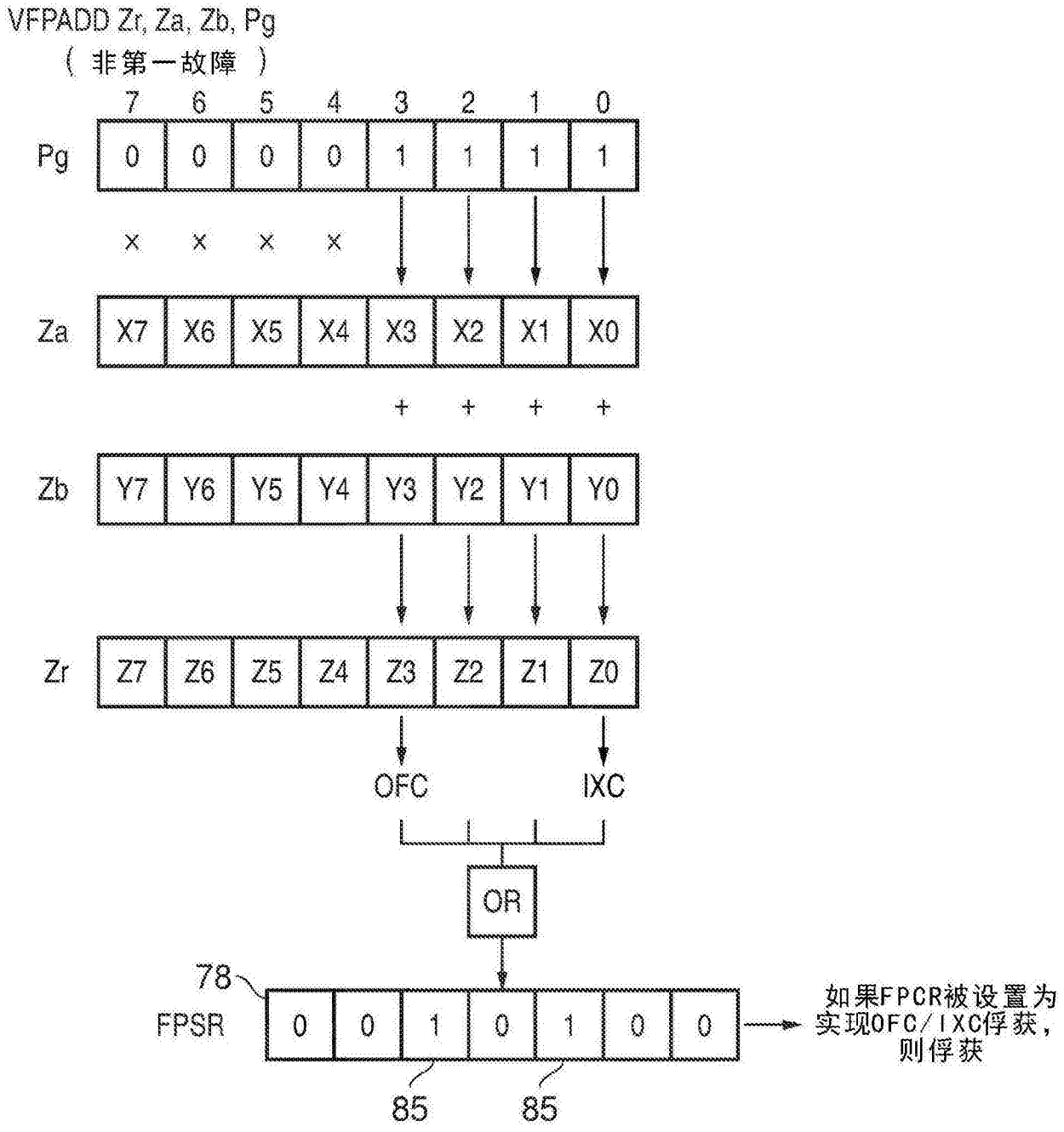

VFPADD Zr, Za, Zb, Pg
（ 非第一故障 ）
7 6 5 4 3 2 1 0
Pg
0 0 0 0 1 1 1 1
× × × ×
Za
X7 X6 X5 X4 X3 X2 X1 X0
+ + + +
Zb
Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
Zr
Z7 Z6 Z5 Z4 Z3 Z2 Z1 Z0
OFC
IXC
OR
78
FPSR
0 0 1 0 1 0 0
85
85
如果FPCR被设置为
实现OFC/IXC俘获，
则俘获

图3

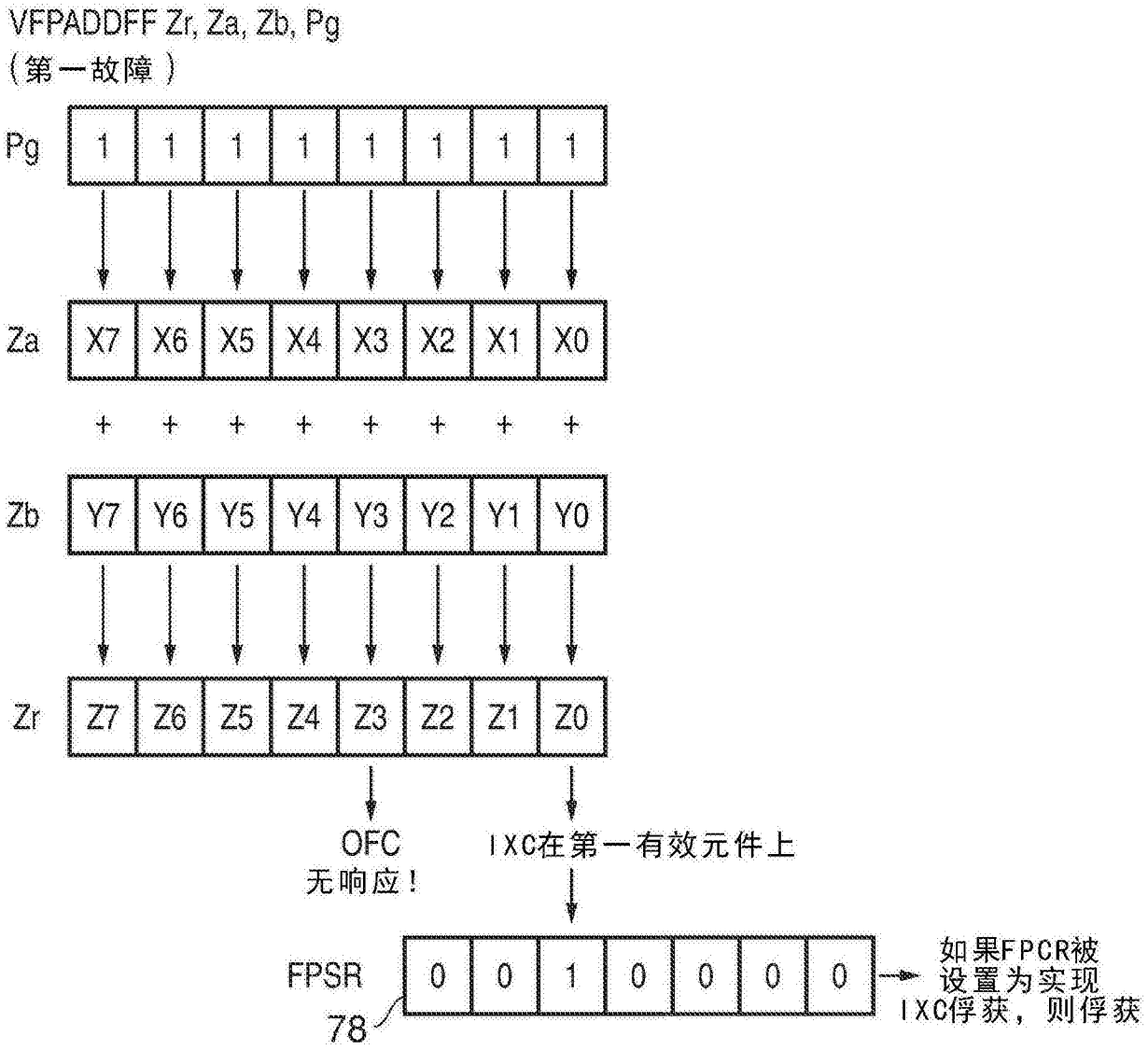

VFPADDFF Zr, Za, Zb, Pg
（第一故障）
Pg
1 1 1 1 1 1 1 1
Za
X7 X6 X5 X4 X3 X2 X1 X0
+ + + + + + + +
Zb
Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
Zr
Z7 Z6 Z5 Z4 Z3 Z2 Z1 Z0
OFC
无响应！
IXC在第一有效元件上
FPSR
78
0 0 1 0 0 0 0
如果FPCR被
设置为实现
IXC俘获，则俘获

图4

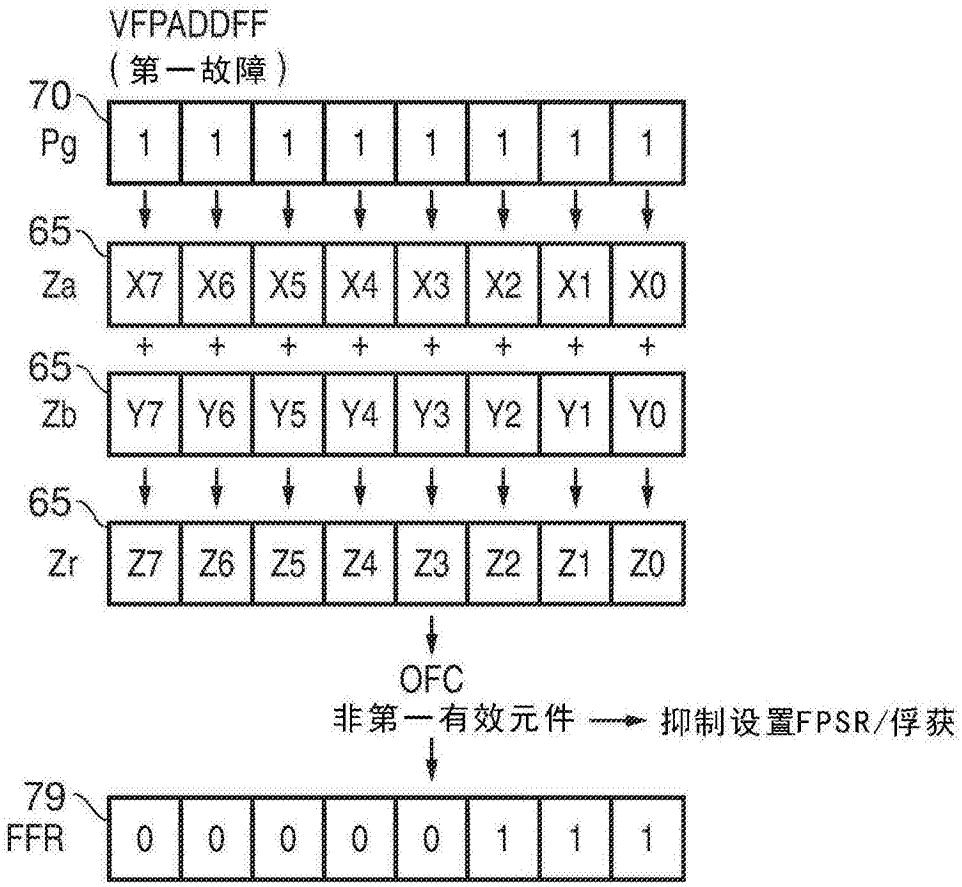

VFPADDFF
(第一故障)
70
Pg
1 1 1 1 1 1 1 1
65
Za
X7 X6 X5 X4 X3 X2 X1 X0
+ + + + + + + +
65
Zb
Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
65
Zr
Z7 Z6 Z5 Z4 Z3 Z2 Z1 Z0
OFC
非第一有效元件 → 抑制设置FPSR/俘获
79
FFR
0 0 0 0 0 1 1 1

图5

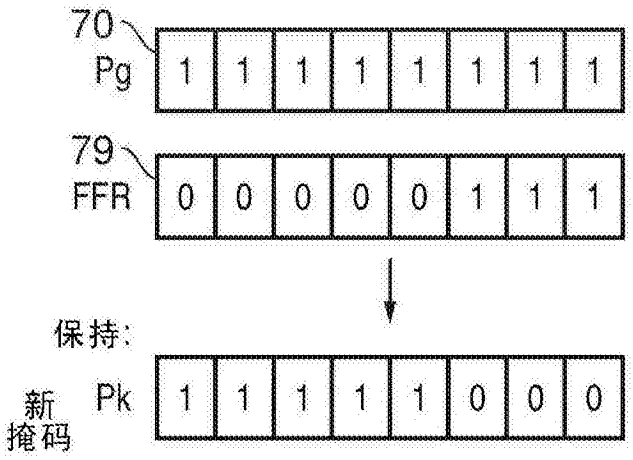

70
Pg
1 1 1 1 1 1 1 1
79
FFR
0 0 0 0 0 1 1 1
保持:
新
掩码
Pk
1 1 1 1 1 0 0 0

图6

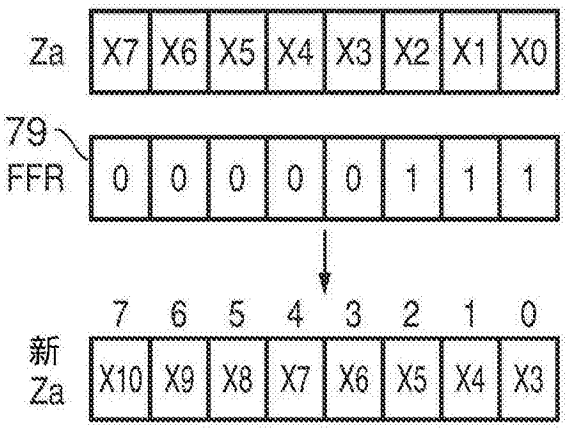


图7

```
non-speculative:    ⋮
        while (1) {
        VFPADD Zr, Za, Zb, Pk
        IF cond (Zc [ j ]) = TRUE, break
```

图8

```
        }
speculative:
        while (1) {
                Pk = Pg
retry:          SETFFR          100
        102     VFPADDFF Zr, Za, Zb, Pk
        104     IF cond (Zr [ j ]) = TRUE, break
        106     Pk = remain (Pg, FFR)
        108     BR retry IF Pk! = ALL FALSE
        }
```

图9

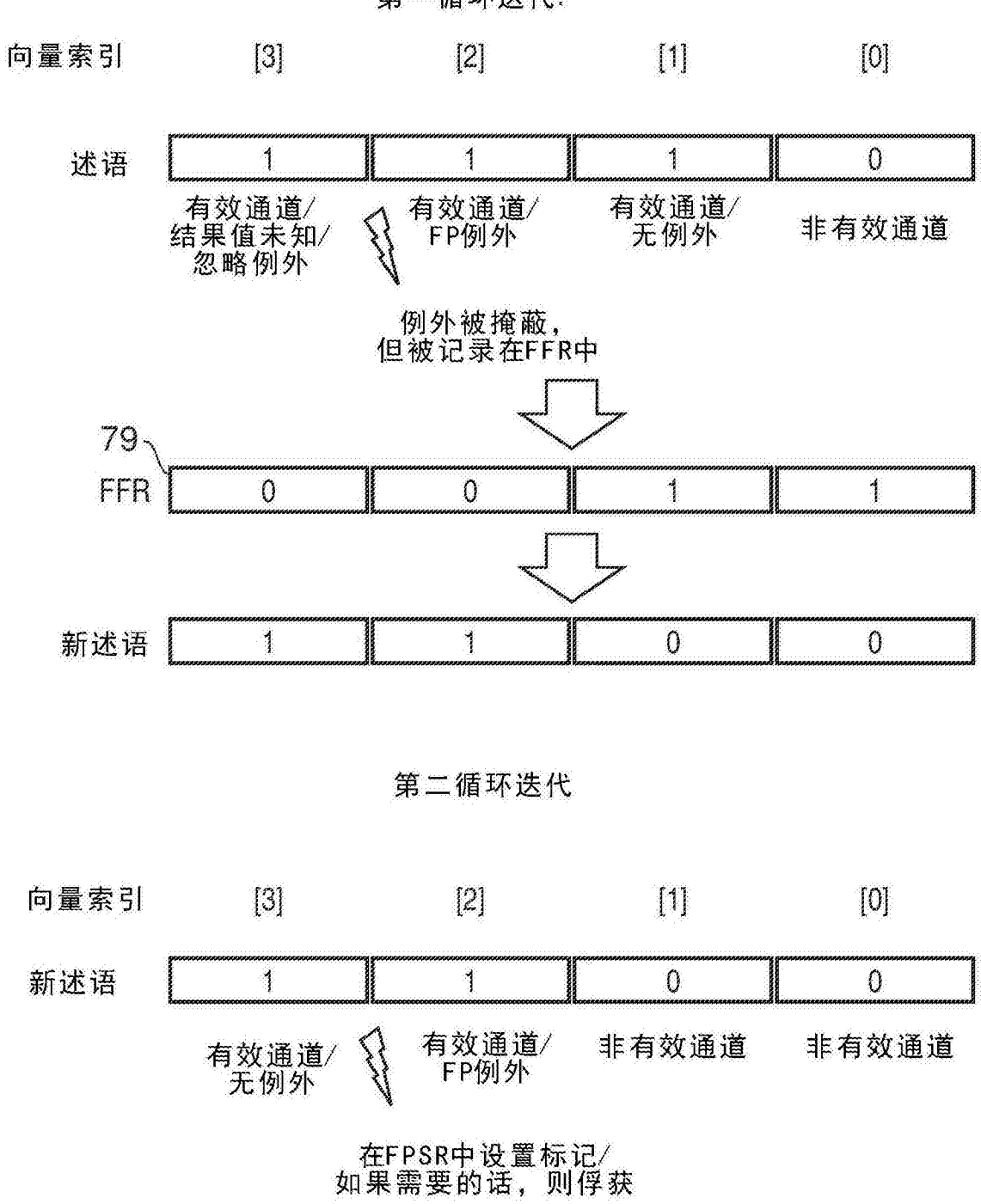

第一循环迭代:
向量索引
[3]
[2]
[1]
[0]
述语
1
1
1
0
有效通道/
结果值未知/
忽略例外
有效通道/
FP例外
有效通道/
无例外
非有效通道
例外被掩蔽,
但被记录在FFR中
79
FFR
0
0
1
1
新述语
1
1
0
0
第二循环迭代
向量索引
[3]
[2]
[1]
[0]
新述语
1
1
0
0
有效通道/
无例外
有效通道/
FP例外
非有效通道
非有效通道
在FPSR中设置标记/
如果需要的话,则俘获

图10

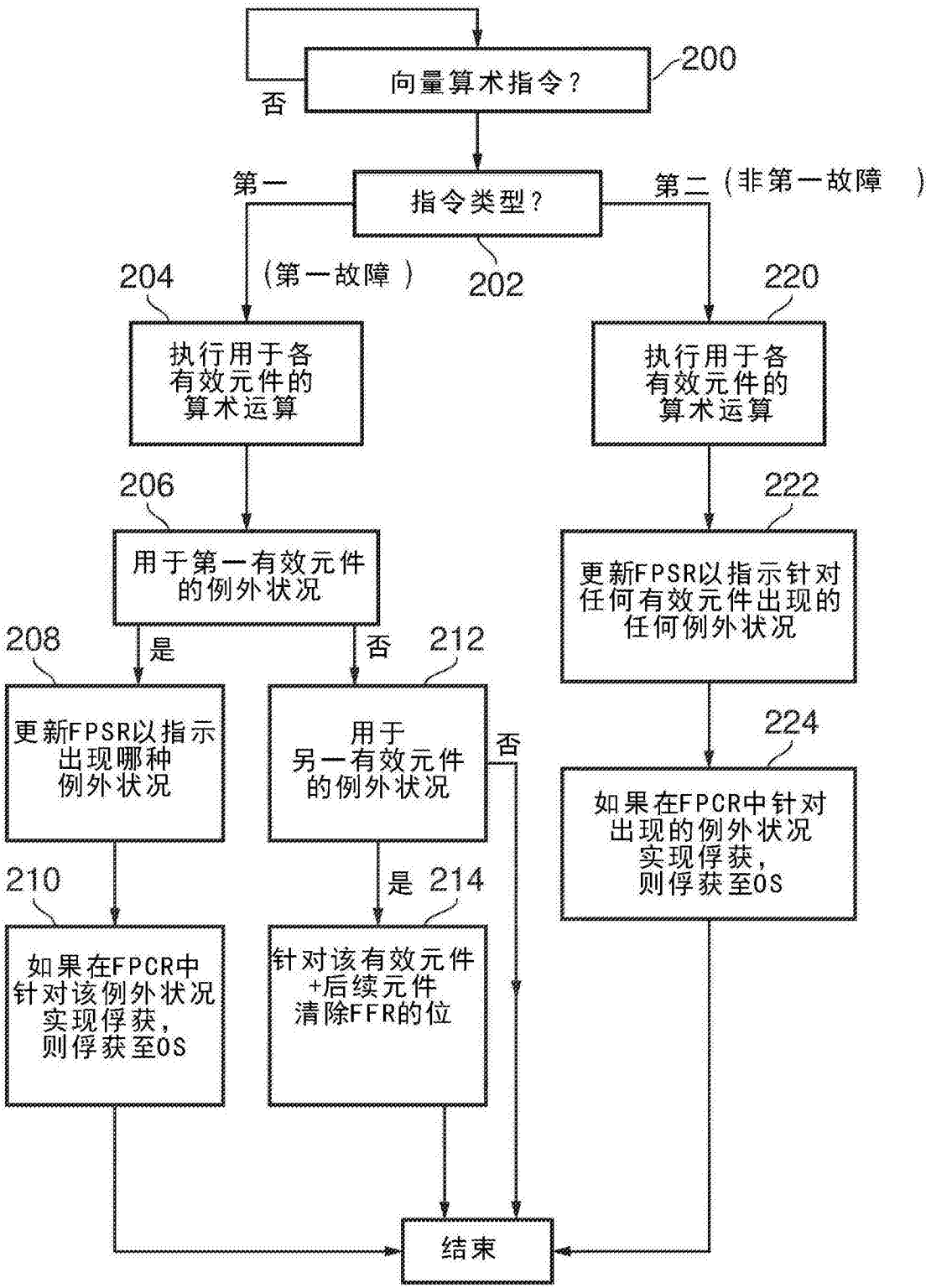
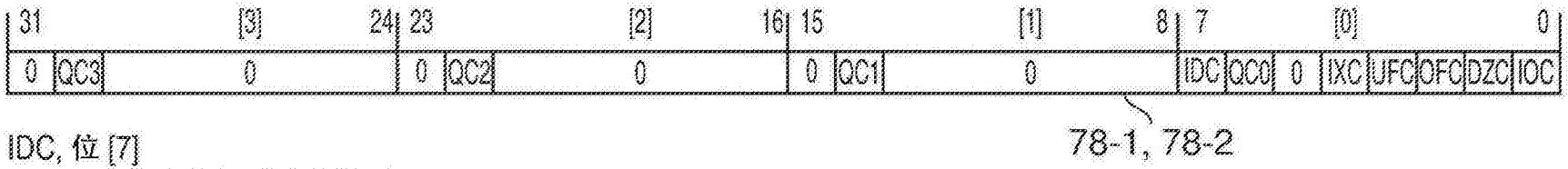

向量算术指令？
200
否
指令类型？
第一
第二（非第一故障）
202
204
（第一故障）
220
执行用于各有效元件的算术运算
执行用于各有效元件的算术运算
206
222
用于第一有效元件的例外状况
更新FPSR以指示针对任何有效元件出现的任何例外状况
208
是
否
212
更新FPSR以指示出现哪种例外状况
用于另一有效元件的例外状况
否
224
如果在FPCR中针对出现的例外状况实现俘获，则俘获至OS
210
是
214
如果在FPCR中针对该例外状况实现俘获，则俘获至OS
针对该有效元件+后续元件清除FFR的位
结束

图11



IDC, 位 [7]

FP输入不正规累计标记

QC*n*, 位 [6, 14, 22, 30]

在字节0、1、2和3上的整数饱和度累计标记

IXC, 位 [4]

FP不准确累计标记

UFC, 位 [3]

FP下溢累计标记

OFC, 位 [2]

FP溢出累计标记

DZC, 位 [1]

FP除以零累计标记

IOC, 位 [0]

FP无效运算累计标记

位 [5,8-13,15-21, 23-29, 31]

保留，RES0

图12

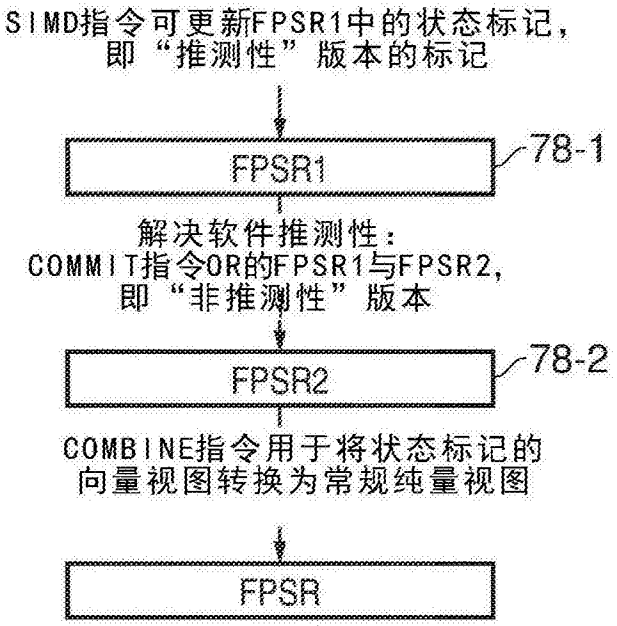


图13

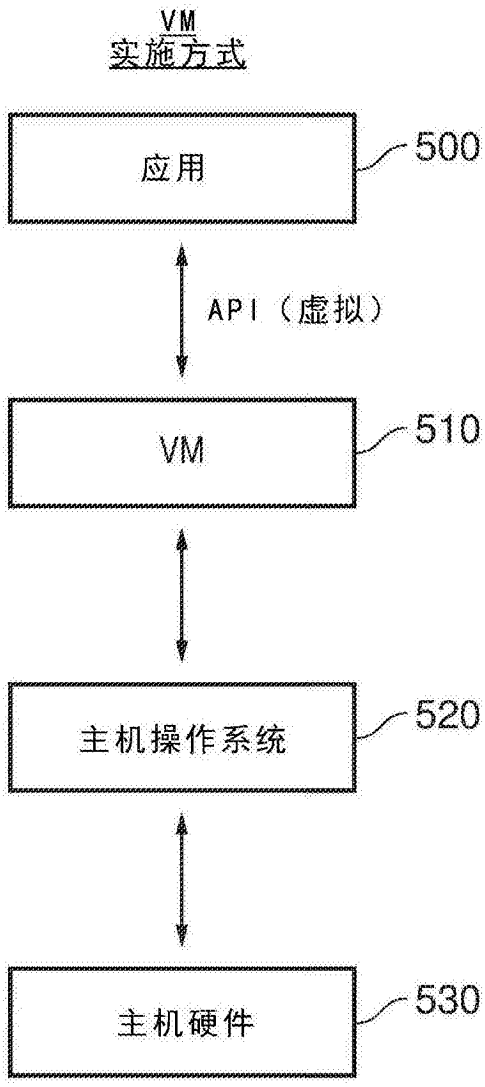


图14