(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111433741 B
(45)授权公告日 2024.12.20

(21)申请号 201880078648.7
(22)申请日 2018.11.15
(65)同一申请的已公布的文献号
申请公布号 CN 111433741 A
(43)申请公布日 2020.07.17
(30)优先权数据
17386048.7 2017.12.13 EP
(85)PCT国际申请进入国家阶段日
2020.06.04
(86)PCT国际申请的申请数据
PCT/EP2018/081444 2018.11.15
(87)PCT国际申请的公布数据
WO2019/115142 EN 2019.06.20
(73)专利权人 ARM有限公司
地址 英国剑桥
(72)发明人 姆布·埃约勒
奈杰尔·约翰·斯蒂芬斯
内尔·伯吉斯
格里戈里奥斯·马格克里斯
(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
专利代理师 林强
(51)Int.Cl.
G06F 9/30(2006.01)
(56)对比文件
US 2006259737 A1,2006.11.16
US 2015227367 A1,2015.08.13
US 6295597 B1,2001.09.25
审查员 陈雅灵

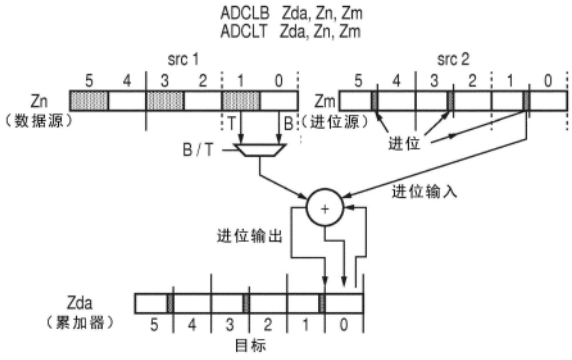
权利要求书4页 说明书17页 附图19页

(54)发明名称

数据处理设备和数据处理方法

(57)摘要

本公开提供了数据处理设备和数据处理方法。本公开描述了向量带进位加法指令,其使用目标向量寄存器的一些元素或谓词寄存器的对应字段,以提供对应于带进位加法运算结果的进位信息。这有助于加速涉及长整数值乘法的计算。



CN 111433741 B

1.一种数据处理设备,包括:

处理电路,用于执行数据处理;

指令解码电路,用于解码指令以控制所述处理电路执行所述数据处理;以及

多个向量寄存器,用于存储包括多个数据元素的向量操作数;其中:

所述指令解码电路响应于指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器和进位源向量寄存器的向量带进位加法指令,针对所述目标向量寄存器的每个数据元素对控制所述处理电路进行下述操作,其中,至少所述目标向量寄存器和所述进位源向量寄存器均指定包括至少一个数据元素对的操作数,每个数据元素对包括第一数据元素和第二数据元素,所述操作包括:

利用与第一源数据值、第二源数据值和进位值的加法的结果对应的值来更新所述目标向量寄存器的所述数据元素对的所述第一数据元素,所述第一源数据值获得自所述第一数据源向量寄存器的选定数据元素,所述第二源数据值获得自所述第二数据源向量寄存器的选定数据元素,所述进位值获得自所述进位源向量寄存器的对应数据元素对的所述第二数据元素;以及

利用与所述加法的进位输出对应的值来更新所述目标向量寄存器的该数据元素对的所述第二数据元素,

其中利用与所述加法的结果对应的值所更新的所述目标向量寄存器的所述数据元素对的所述第一数据元素和利用与所述加法的进位输出对应的值所更新的所述目标向量寄存器的所述数据元素对的所述第二数据元素是同一向量寄存器的数据元素。

2.根据权利要求1所述的设备,其中,所述第一数据源向量寄存器是与所述目标向量寄存器相同的寄存器,并且所述第一源数据值包括所述目标向量寄存器的所述数据元素对的所述第一数据元素的先前值。

3.根据权利要求1和2中任一项所述的设备,其中,每个数据元素对包括邻接数据元素对。

4.根据权利要求1所述的设备,其中,在所述目标向量寄存器和所述进位源向量寄存器中的数据元素对的数量至少为二时,这些数据元素对的所述第一数据元素和这些数据元素对的所述第二数据元素交错。

5.根据权利要求1所述的设备,其中,所述指令解码电路被配置为响应于所述向量带进位加法指令而控制所述处理电路进行下述操作:

从所述进位源向量寄存器的所述对应数据元素对的所述第二数据元素的最低有效位获得所述进位值,以及

利用与所述加法的所述进位输出对应的值来更新所述目标向量寄存器的所述数据元素对的所述第二数据元素的最低有效位。

6.根据权利要求5所述的设备,其中,响应于所述向量带进位加法指令,针对所述进位源向量寄存器和所述目标向量寄存器两者,所述第二数据元素的除了所述最低有效位以外的剩余位是未使用的。

7.根据权利要求1所述的设备,其中,每个数据元素包括2^N个位,并且所述第二源数据值包括2^N个位,其中N为整数。

8.根据权利要求1所述的设备,其中,所述第二数据源向量寄存器也指定包括至少一个

数据元素对的操作数,每个数据元素对包括第一数据元素和第二数据元素;

响应于所述向量带进位加法指令的第一变体,所述第二源数据值包括从所述第二数据源向量寄存器的对应数据元素对的所述第一数据元素获得的值;以及

响应于所述向量带进位加法指令的第二变体,所述第二源数据值包括从所述第二数据源向量寄存器的该对应数据元素对的所述第二数据元素获得的值。

9.根据权利要求1所述的设备,其中,响应于所述向量带进位加法指令的谓词变体,其中所述谓词变体与指定至少一个谓词指示的谓词值相关联并且每个谓词指示与所述目标向量寄存器的所述至少一个数据元素对之一对应,所述指令解码电路被配置为控制所述处理电路进行下述操作:

针对对应的谓词指示具有第一值的数据元素对,执行利用与所述加法的结果对应的值来对所述数据元素对的所述第一数据元素进行的所述更新,以及利用所述加法的所述进位输出来对所述数据元素对的所述第二数据元素进行的所述更新;以及

对于对应的谓词指示具有第二值的数据元素对,抑制利用与所述加法的结果对应的值来对所述数据元素对的所述第一数据元素进行的所述更新,以及利用所述加法的所述进位输出来对所述数据元素对的所述第二数据元素进行的所述更新。

10.根据权利要求1所述的设备,其中响应于所述向量带进位加法指令的加法变体,所述加法包括将所述第一源数据值、所述第二源数据值和所述进位值相加;以及

响应于所述向量带进位加法指令的减法变体,所述加法包括从所述第一源数据值减去所述第二源数据值,其中,所述进位值指示所述减法的借位值,并且所述进位输出包括所述减法的借位输出。

11.一种计算机可读存储介质,包括计算机程序,用于控制主机处理设备以提供用于执行目标程序代码指令的指令执行环境,该计算机程序包括:

指令解码程序逻辑,用于解码所述目标程序代码指令以控制处理程序逻辑执行数据处理;以及

向量寄存器数据结构,用于存储代表多个向量寄存器的数据,所述向量寄存器用于存储包括多个数据元素的向量操作数;其中:

所述指令解码程序逻辑响应于指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器和进位源向量寄存器的向量带进位加法指令,针对所述目标向量寄存器的每个数据元素对控制所述处理程序逻辑以更新所述向量寄存器数据结构来进行下述操作,其中,至少所述目标向量寄存器和所述进位源向量寄存器均指定包括至少一个数据元素对的操作数,每个数据元素对包括第一数据元素和第二数据元素,所述操作包括:

利用与第一源数据值、第二源数据值和进位值的加法的结果对应的值来更新所述目标向量寄存器的所述数据元素对的所述第一数据元素,所述第一源数据值获得自所述第一数据源向量寄存器的选定数据元素,所述第二源数据值获得自所述第二数据源向量寄存器的选定数据元素,所述进位值获得自所述进位源向量寄存器的对应数据元素对的所述第二数据元素;以及

利用与所述加法的进位输出对应的值来更新所述目标向量寄存器的该数据元素对的所述第二数据元素,

其中利用与所述加法的结果对应的值所更新的所述目标向量寄存器的所述数据元素

对的所述第一数据元素和利用与所述加法的进位输出对应的值所更新的所述目标向量寄存器的所述数据元素对的所述第二数据元素是同一向量寄存器的数据元素。

12.一种数据处理方法,包括以下步骤:

解码指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器和进位源向量寄存器的向量带进位加法指令,至少所述目标向量寄存器和所述进位源向量寄存器均指定包括至少一个数据元素对的操作数,每个数据元素对包括第一数据元素和第二数据元素;以及

响应于所述向量带进位加法指令的解码,针对所述目标向量寄存器的每个数据元素对控制处理电路进行下述操作:

利用与第一源数据值、第二源数据值和进位值的加法的结果对应的新值来更新所述目标向量寄存器的所述数据元素对的所述第一数据元素,所述第一源数据值获得自所述第一数据源向量的选定数据元素,所述第二源数据值获得自所述第二数据源向量寄存器的选定数据元素,所述进位值获得自所述进位源向量寄存器的对应数据元素对的所述第二数据元素;以及

利用与所述加法的进位输出对应的值来更新所述目标向量寄存器的所述数据元素对的所述第二数据元素,

其中利用与所述加法的结果对应的值所更新的所述目标向量寄存器的所述数据元素对的所述第一数据元素和利用与所述加法的进位输出对应的值所更新的所述目标向量寄存器的所述数据元素对的所述第二数据元素是同一向量寄存器的数据元素。

13.一种数据处理设备,包括:

处理电路,用于执行数据处理;

指令解码电路,用于解码指令以控制所述处理电路执行所述数据处理;

多个向量寄存器,用于存储包括多个数据元素的向量操作数;以及

多个谓词寄存器,包括用于存储谓词值的多个谓词字段,所述谓词值用于控制由所述处理电路执行的运算的掩码;其中:

所述指令解码电路响应于指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器、输入谓词寄存器和输出谓词寄存器的向量带进位加法指令,针对所述目标向量寄存器的给定数据元素控制所述处理电路进行下述操作:

利用与第一源数据值、第二源数据值和进位值的加法的结果对应的值来更新所述目标向量寄存器的所述给定数据元素,所述第一源数据值获得自所述第一数据源向量寄存器的对应数据元素,所述第二源数据值获得自所述第二数据源向量寄存器的对应数据元素,所述进位值获得自所述输入谓词寄存器的对应谓词字段;以及

利用与所述加法的进位输出对应的值来更新所述输出谓词寄存器的对应谓词字段。

14.一种计算机可读存储介质,包括计算机程序,用于控制主机处理设备以提供用于执行目标程序代码指令的指令执行环境,该计算机程序包括:

指令解码程序逻辑,用于解码所述目标程序代码指令以控制处理程序逻辑执行数据处理;以及

寄存器数据结构,用于存储代表多个向量寄存器和多个谓词寄存器的数据,所述向量寄存器用于存储包括多个数据元素的向量操作数,所述谓词寄存器用于存储用于控制由所

述处理程序逻辑执行的向量运算的掩码的谓词值;其中:

所述指令解码程序逻辑响应于指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器、输入谓词寄存器和输出谓词寄存器向量的带进位加法指令,针对所述目标向量寄存器的给定数据元素控制处理电路以更新所述寄存器数据结构来进行下述操作:

利用与第一源数据值、第二源数据值和进位值的结果对应的值来更新所述目标向量寄存器的所述给定数据元素,所述第一源数据值获得自所述第一数据源向量寄存器的对应数据元素,所述第二源数据值获得自所述第二数据源向量寄存器的对应数据元素,所述进位值获得自所述输入谓词寄存器的对应谓词字段;以及

利用与所述加法的进位输出对应的值来更新所述输出谓词寄存器的对应谓词字段。

15.一种数据处理方法,包括:

解码指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器、输入谓词寄存器和输出谓词寄存器的向量带进位加法指令,所述输入谓词寄存器和所述输出谓词寄存器选自用于存储谓词值的多个谓词寄存器,所述谓词值用于控制向量运算的掩码;以及

响应于所述向量带进位加法指令的解码,针对所述目标向量寄存器的给定数据元素控制处理电路进行下述操作:

利用与第一源数据值、第二源数据值和进位值的加法的结果对应的值来更新所述目标向量寄存器的所述给定数据元素,所述第一源数据值获得自所述第一数据源向量寄存器的对应数据元素,所述第二源数据值获得自所述第二数据源向量寄存器的对应数据元素,所述进位值获得自所述输入谓词寄存器的对应谓词字段;以及

利用与所述加法的进位输出对应的值来更新所述输出谓词寄存器的对应谓词字段。

数据处理设备和数据处理方法

技术领域

[0001] 本技术涉及数据处理领域。

背景技术

[0002] 一些数据处理系统支持向量指令的处理,向量指令作用于或生成包含多个数据元素的向量操作数。通过支持响应于单一指令而处理多个不同的数据元素,相较于使用标量指令执行相同的操作,可以提高代码密度并可减少获取与解码指令的开销。

发明内容

[0003] 至少一些示例提供一种设备,包括:

[0004] 处理电路,用于执行数据处理;

[0005] 指令解码电路,用于解码指令以控制处理电路执行数据处理;以及

[0006] 多个向量寄存器,用于存储向量操作数,这些向量操作数包括多个数据元素;其中:

[0007] 该指令解码电路响应于指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器和进位源向量寄存器的向量带进位加法指令,针对该目标向量寄存器的每个数据元素对控制该处理电路,至少目标向量寄存器和进位源向量寄存器均指定包括至少一个数据元素对的操作数,每个数据元素对包括第一数据元素和第二数据元素:

[0008] 利用对应于第一源数据值、第二源数据值和进位值的加法的结果的值来更新该目标向量寄存器的该数据元素对的第一数据元素,该第一源数据值获得自第一数据源向量寄存器的选定数据元素,该第二源数据值获得自该第二数据源向量寄存器的选定数据元素,该进位值获得自进位源向量寄存器的对应数据元素对的第二数据元素;以及

[0009] 利用对应于该加法的进位输出的值来更新该目标向量寄存器的该数据元素对的第二数据元素。

[0010] 至少一些示例提供一种计算机,用于控制主机处理设备以提供指令执行环境以执行目标程序代码指令,该计算机程序包括:

[0011] 指令解码程序逻辑,用于解码该目标程序代码指令以控制处理程序逻辑执行数据处理;以及

[0012] 向量寄存器数据结构,用于存储数据,数据代表用于存储向量操作数的多个向量寄存器,这些向量操作数包括多个数据元素;其中:

[0013] 该指令解码程序逻辑响应于指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器和进位源向量寄存器的向量带进位加法指令,针对该目标向量寄存器的每个数据元素对控制该处理程序逻辑以更新该向量寄存器数据结构,至少该目标向量寄存器和该进位源向量寄存器均指定包括至少一个数据元素对的操作数,每个数据元素对包括第一数据元素和第二数据元素:

[0014] 利用对应于第一源数据值、第二源数据值和进位值的加法的结果的值来更新该目

标向量寄存器的该数据元素对的第一数据元素，该第一源数据值获得自该第一数据源向量寄存器的选定数据元素，该第二源数据值获得自该第二数据源向量寄存器的选定数据元素，该进位值获得自该进位源向量寄存器的对应数据元素对的第二数据元素；以及

[0015] 利用对应于该加法的进位输出的值来更新该目标向量寄存器的该数据元素对的第二数据元素。

[0016] 至少一些示例提供一种数据处理方法，包括以下步骤：

[0017] 解码指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器、以及进位源向量寄存器的向量带进位加法指令，至少该目标向量寄存器和该进位源向量寄存器均指定包括至少一个数据元素对的操作数，每个数据元素对包括第一数据元素和第二数据元素；以及

[0018] 响应于该向量带进位加法指令的解码，针对该目标向量寄存器的每个数据元素对控制处理电路：

[0019] 利用对应于第一源数据值、第二源数据值和进位值的加法的结果的新值来更新该目标向量寄存器的该数据元素对的第一数据元素，该第一源数据值获得自该第一数据源向量的选定数据元素，该第二源数据值获得自该第二数据源向量寄存器的选定数据元素，该进位值获得自该进位源向量寄存器的对应数据元素对的第二数据元素；以及

[0020] 利用对应于该加法的进位输出的值来更新该目标向量寄存器的该数据元素对的第二数据元素。

[0021] 至少一些示例提供了一种设备，包括：

[0022] 处理电路，用于执行数据处理；

[0023] 指令解码电路，用于解码指令以控制该处理电路执行该数据处理；

[0024] 多个向量寄存器，用于存储向量操作数，这些向量操作数包括多个数据元素；以及

[0025] 多个谓词寄存器，包括多个谓词字段以存储用于控制该处理电路所执行的运算的掩码的谓词值；其中：

[0026] 该指令解码电路响应于指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器、输入谓词寄存器与输出谓词寄存器的向量带进位加法指令，针对该目标向量寄存器的给定数据元素控制该处理电路：

[0027] 利用对应于第一源数据值、第二源数据值和进位值的加法的结果的值来更新该目标向量寄存器的该给定数据元素，该第一源数据值获得自该第一数据源向量寄存器的对应数据元素，该第二源数据值获得自该第二数据源向量寄存器的对应数据元素，该进位值获得自该输入谓词寄存器的对应谓词字段；以及

[0028] 利用对应于该加法的进位输出的值来更新该输出谓词寄存器的对应谓词字段。

[0029] 至少一些示例提供了一种计算机程序，用于控制主机处理设备以提供指令执行环境以执行目标程序代码指令，计算机程序包括：

[0030] 指令解码程序逻辑，用于解码该目标程序代码指令以控制处理程序逻辑执行数据处理；以及

[0031] 寄存器数据结构，用于存储数据，该数据代表用于存储包括多个数据元素的向量操作数的多个向量寄存器和用于存储用于控制该处理程序逻辑所执行的向量运算的掩码的谓词值的多个谓词寄存器；其中：

[0032] 该指令解码程序逻辑响应于指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器、输入谓词寄存器和输出谓词寄存器的向量带进位加法指令，针对该目标向量寄存器的给定数据元素控制该处理电路以更新该寄存器数据结构：

[0033] 利用对应于第一源数据值、第二源数据值和进位值的结果的值来更新该目标向量寄存器的给定数据元素，该第一源数据值获得自该第一数据源向量寄存器的对应数据元素，该第二源数据值获得自该第二数据源向量寄存器的对应数据元素，该进位值获得自该输入谓词寄存器的对应谓词字段；以及

[0034] 利用对应于该加法的进位输出的值来更新该输出谓词寄存器的对应谓词字段。

[0035] 至少一些示例提供一种数据处理方法，包括以下步骤：

[0036] 解码指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器、输入谓词寄存器和输出谓词寄存器的向量带进位加法指令，该输入谓词寄存器和该输出谓词寄存器选自用于存储用于控制向量运算掩码的谓词值的多个谓词寄存器；以及

[0037] 响应于该向量带进位加法指令的解码，针对该目标向量寄存器的给定数据元素控制处理电路：

[0038] 利用对应于第一源数据值、第二源数据值和进位值的加法的结果的值来更新该目标向量寄存器的给定数据元素，该第一源数据值获得自该第一数据源向量寄存器的对应数据元素，该第二源数据值获得自该第二数据源向量寄存器的对应数据元素，该进位值获得自该输入谓词寄存器的对应谓词字段；以及

[0039] 利用对应于该加法的进位输出的值来更新该输出谓词寄存器的对应谓词字段。

附图说明

[0040] 根据下文示例说明，并结合附图来阅读时，本技术的进一步的方面、特征和优点将变得显而易见，在附图中：

[0041] 图1示意性地示出了支持执行向量指令的数据处理设备的示例；

[0042] 图2示出了具有谓词的向量操作的示例；

[0043] 图3和图4示出了根据第一示例的向量带进位加法指令的加法和减法变体；

[0044] 图4A示出了向量带进位加法指令的谓词变体；

[0045] 图5是示出处理向量带进位加法指令的方法的流程图；

[0046] 图6示出了根据Karatsuba算法的长整数值的乘法的向量化的示例；

[0047] 图7示出了使用64位向量元素大小对两个256位进行乘法运算时要相加的部分乘积的示例；

[0048] 图8示出了包括图3示出的向量带进位加法指令的示例指令序列；

[0049] 图9示出了使用向量带进位加法指令的指令序列的第二示例；

[0050] 图10A至图10I示出了图9的指令序列的处理的系列图；

[0051] 图11示出使用谓词寄存器传输针对加法或减法的进位值的向量带进位加法指令的第二示例；

[0052] 图12是根据图11的示例示出向量带进位加法指令的处理的流程图；以及

[0053] 图13例示了可使用的仿真器示例。

具体实施方式

[0054] 一些处理工作负载可要求对非常大的整数值(诸如1024位、2048位或4096位值)应用数学运算。例如,密码学工作负载可依赖对大数进行因子分解的难度,且因此可需要将这种大数相乘以加密或解密信息。例如在RSA 2048中,存在许多2048位数字的循序乘法以产生4096位乘积。一些财务处理应用也需要处理长整数值。若使用标准教科书方法将一个操作数的每一位乘以另一操作数的每一位,则执行乘法的复杂度通常将随着位数的平方值而放大。然而,已知一些减少复杂度的算法,使得计算成本随着位数放大的比例较慢。例如,Karatsuba算法将要相乘的长数分成较小的部分以单独相乘,这能够在所需的相乘数量以及一些额外的加法、减法或移位运算之间进行权衡。因此,长整数的乘法,可被分解成数个较小的乘法以及数个加法、减法与移位。可使用向量指令以加速这种计算,此系通过将不同的子乘法映射到不同的向量道(vector lane)上,使得单一指令可控制数个独立子乘法的部分的计算。在此情况中,可将多数工作负载有效率地运用在执行各种乘法的部分乘积的一系列的加法或减法中。然而,这在管理部分乘积加法之间的进位信息的传输上产生了设计挑战,因为来自一个加法的进位信息可需要被输入至对均等有效值(values of equivalent significance)执行的下一个加法中,且因此可需要将来自一个指令的进位信息保存作为对于另一个指令的输入。

[0055] 一个方法可以有效地减少正在每个向量道内处理的实数据的元素大小,并将每个道顶端处的一个或更多个位保存用于存储来自此道中加法的进位信息,以让后续的指令可读取此进位信息。然而,发明人认识到在使用此方法时,因为需要在每个道内包括额外的进位信息,可能需要许多额外指令以解封输入数据向量(具有完全占用向量道的元素)以及调整这些分量的大小以由减少的有效数据元素大小将它们分散到多个向量道。再者,若输入数据包括多个具有给定元素大小的元素,则分散到具有保留给进位的空间的道时,相同量的输入数据将需要较大的向量道总数。例如,4个64位元素的输入向量,在由56位实际数据与8位保留给进位之下分散到向量道时,将需要分配5个道以容纳相同位数(256位)的输入数据,由于4道56位的每道仅提供224位且因此需要第五道以容纳最后32位。由于要计算的部分乘积数随着向量分量数的平方放大,这可显著增加所需要的乘法与部分乘积加法的数量,而可进一步降低性能。

[0056] 下文讨论的技术,提供用于实施向量带进位加法(add-with-carry)指令的技术,使得不需要解封输入数据,因为指令可直接对占用整体向量道的源数据元素进行运算。

[0057] 在一个方法中,向量带进位加法指令指定目标向量寄存器、第一数据源向量寄存器和第二数据源向量寄存器以及进位源向量寄存器。至少目标向量寄存器和进位源向量寄存器每个均指定包括至少一个数据元素对的操作数,其中每个数据元素对包括第一数据元素和第二数据元素。响应于指令,指令解码器对目标向量寄存器的每个数据元素对控制处理电路,以利用对应于第一源数据值、第二源数据值和进位值的加法结果的值来更新目标向量寄存器的数据元素对的第一数据元素,第一源数据值获得自第一数据源向量寄存器的选定数据元素,第二源数据值获得自第二数据源向量寄存器的选定数据元素,进位值获得自进位源向量寄存器的对应数据元素对的第二数据元素。再者,利用对应于加法的进位输出的值来更新目标向量寄存器的数据元素对的第二数据元素。

[0058] 因此,目标向量寄存器中的每个元素对的第二数据元素被保留以提供加法的进位

输出,且类似地,进位源向量寄存器内的每对的第二数据元素被用于提供加法的进位输入。因此,目标向量寄存器和进位源向量寄存器具有对应的布局,且实际上通常在实际使用这些指令时,由向量带进位加法指令的一个实例所产生的目标向量寄存器,可用作针对向量带进位加法指令的后续实例的进位源向量寄存器。

[0059] 此方法是与直觉相反的,因为这有效地将可用于携带源数据和加法结果值的数据元素数减半,这相反于向量处理器中通过尽可能多地增加向量寄存器文件的利用来改进计算的效率的常见设计原则。换言之,使用一半的向量道来提供进位值,看起来会浪费向量处理器和向量寄存器文件的容量,并将使处理给定数据元素量所需的指令数增加一倍。

[0060] 然而,令人惊讶地,发明人认知到尽管可能需要更多的带进位加法指令以执行给定工作负载,但将进位放到每对元素的第二数据元素中的益处,在于第一数据元素可占用完整道大小,这表示不需要将来自输入向量的元素解封或调整大小,且因此向量带进位加法指令可直接对封装的源数据运算。实际上,对于涉及长整数乘法的工作负载,已发现到,即使需要额外的指令来补偿有效地减少的向量带进位加法指令所处理的元素数,但总体性能仍为较高的,因为(a)避免了解封/调整大小的开销,以及(b)通过前述道内进位方法避免了通过减少有效道大小所造成的部分乘积数的增加。因此,整体而言可改良性能。在任何情况中,在一些微型架构中,即使单个向量带进位加法指令在架构位准对一半的元素运算,结合以对所有元素运算的两个指令实例,可被"融合"在一起以作为单个微指令以被处理管线处理,或可被并行处理,以避免因为将每巨指令(macro-instruction)所处理的元素数减半所造成的显然的性能损失。这是可行的,例如若微型架构支持对结果总线进行多次同时写入。也存在为独立进位链形式的并行方法。

[0061] 在一些实施例中,可由指令与第一和第二源向量寄存器以及进位源向量寄存器分开而指定目标向量寄存器,以提供非破坏性的指令格式,这在指令执行后将所有源向量保存在寄存器文件中,由于结果被写入不同于所有源寄存器的寄存器。

[0062] 然而,在其他实施方式中,第一数据源向量寄存器可为与目标向量寄存器相同的寄存器,且对于每对,第一源数据值可包括目标向量寄存器的数据元素对的第一数据元素的先前值。实际上,累加运算(其中旧累加器值被加至第二源数据值和进位值,并写回同一寄存器)对许多涉及长整数乘法的工作负载(用于将乘法的各自部分乘积相加)是有用的,因此对于第一源数据向量寄存器,不必独立于目标向量寄存器器进行指定。通过将单个寄存器指定为目标向量寄存器和第一数据源向量寄存器两者,编码指令集架构中空间的指令可被节省以用于其他目的。破坏性指令格式(同一寄存器同时作为目标寄存器和一个源寄存器)的另一个优点为,由于有时可需要限制指令所需的源寄存器端口数,且在一些微型架构中,在具有合并谓词(其中目标寄存器的谓词道保存它们的先前值)的运算的情况中目标寄存器可已经需要被读取,仅具有两个进一步的寄存器存取(对于第二源向量寄存器和进位向量寄存器)可为有益的。

[0063] 向量操作数的向量元素大小和/或总和长度可改变。一些实施例可将元素大小和/或向量长度硬联机至特定的固定值。然而,其他系统可支持可变式向量元素大小和向量长度。对于一些向量元素大小或向量长度,向量可仅包括两个元素,且在此情况中目标向量寄存器与进位源向量寄存器可包括单个数据元素对,即一个第一数据元素和一个第二数据元素。

[0064] 然而,有用的是使指令解码电路与处理电路支持执行向量带进位加法指令的至少一个实例,目标向量寄存器和进位源向量寄存器包括多个数据元素对,即它们包括至少四个元素(第一与第二数据元素的每者的两个或更多个)。这使得系统能够支持响应于单个指令计算多个加法,这可有用于加速涉及长整数操作数的乘法的计算。例如,每个元素对可代表对Karatsuba算法的不同子乘法的部分乘积的加法。

[0065] 可由多种方式完成将数据元素对映射到向量操作数元素上。例如,在一个示例中,每对的第一元素可位于寄存器的一半中,而每对的第二元素可位于另一半中。例如,在具有4对第一和第二元素的示例中,这些元素可被由2d、2c、2b、2a、1d、1c、1b、1a的次序设置(其中1a为对应于第二元素2a的第一元素,且对于b、c、d对而言也是如此)。

[0066] 然而在一个实施方式中,每个数据元素对可包括一对邻接的数据元素。因此,在目标向量寄存器和进位源向量寄存器中的数据元素对的数量至少为两个时,数据元素对的第一数据元素可以与数据元素对的第二数据元素交错(interleaved)。例如,在4对第一元素和第二元素下,这些元素可被由2d、1d、2c、1c、2b、1b、2a、1a的次序设置(或者可将每对中的第一和第二元素的次序转置)。由于这可使硬件设计更有效率,此方法可更有效率地由硬件实施,因为要被结合在向量指令的给定子计算中的源操作数与结果值可被限制在位于相同向量道内,或仅分散在紧邻的向量道中,而非需要跨两个或更多个向量道的较长跨道信号路径。因此,通过交错每对的第一数据元素与第二数据元素,可减少所需的布线的复杂度与长度。

[0067] 可从进位源向量寄存器的对应数据元素对的第二数据元素的最低有效位获得进位值。响应于向量带进位加法指令,处理电路可利用对应于加法进位输出的值来更新目标向量寄存器的数据元素对的第二数据元素的最低有效位。使用第二数据元素的最低有效位提供进位可以更有效率,因为最低有效位可邻接于对应对的第一数据元素的最高有效位,使得进位与数据值可被写入目标向量寄存器的连续部分。第二数据元素除了最低有效位以外的剩余位,在进位源向量寄存器和累加器向量寄存器中可以是未使用的。尽管留下许多未使用的位看起来浪费向量寄存器文件中的空间,并浪费向量处理单元中的处理资源,但如上文所讨论的,通过此方法可改进执行长整数乘法时的总体性能。

[0068] 由带进位加法指令处理的向量,可具有每者具有2^N个位的数据元素,且第二源数据值也可包括2^N个位,其中N为整数。因此,由于第二源数据值具有与向量道自身相同的大小(对应于两个位数的精确幂次),第二源数据值占用整个向量道,且因此不需对来自输入向量的数据元素(或先前向量指令的结果)进行调整大小或解封才能允许向量带进位加法指令对输入进行运算。相反地,向量带进位加法指令可直接对来自输入向量的选定元素运算。这是可行的,因为进位被保存在与数据结果不同的向量元素中。

[0069] 可从第二数据源向量寄存器的任何元素获得第二源数据值。可提供指令的不同变体,以从第二源数据向量寄存器的不同元素选择源数据值。

[0070] 在一个示例中,第二数据源向量寄存器可提供包括至少一个数据元素对的操作数,可由与目标和进位源向量寄存器中的元素对对应的方式来设置这个数据元素对。例如,每对的第一数据元素和第二数据元素可被交错。可提供向量带进位加法指令的第一变体和第二变体。对于第一变体,第二源数据值可包括第二数据源向量寄存器的对应数据元素对的第一数据元素(指令结果可独立于每对的第二元素)。对于第二变体,第二源数据值可包

括对应对的第二数据元素(指令结果可独立于每对的第一元素)。通过提供两个指令变体以选择不同的元素作为第二源数据值,这使能了在执行两个向量带进位加法指令(第一/第二变体的每者之一)之前不需要执行任何额外的重新排序或解封第二源向量元素,即能够处理输入向量的所有元素,因为两个指令变体的结合可处理第二源向量的每个元素。

[0071] 在其中指令指定与目标向量寄存器不同的第一数据源向量寄存器的实施例中,第一源数据值可类似地被从每对的第一元素(针对第一指令变体)或每对的第二元素(针对第二指令变体)提取出。

[0072] 另一方面,若第一数据源向量寄存器与目标向量寄存器的寄存器相同,则对于每个加法,第一源数据值可被从目标向量寄存器中的对应元素对的第一数据元素提取出(不论所执行的指令变体是第一变体还是第二变体),因为目标向量寄存器的第二数据元素将用于呈现进位信息。

[0073] 再者,可提供带进位加法指令的谓词变体,这与指定至少一个谓词指示的谓词值相关联,每个谓词指示对应于目标向量寄存器的该至少一个数据元素对之一。响应于谓词变体,指令解码电路可控制处理电路,以如前述对目标向量寄存器的数据元素对(对应的谓词指示具有第一值)执行第一/第二数据元素的更新,并抑制数据元素对的更新(对应的谓词指示具有第二值)。谓词值可与指令相关联,通过包括谓词寄存器指定符(识别存储谓词值的谓词寄存器)的指令解码,或可使用默认谓词寄存器以提供谓词值给所有的带进位加法指令谓词变体实例,不论指令的编码为何。对于目标向量寄存器的元素对(对应谓词指示具有第二值),此元素对可将先前值保持存储在目标向量寄存器的此部分中,或可被清除至零或另一预定值。因此,在带进位加法指令为谓词时,相对于在向量的单独数据元素的粒度下作用,谓词可在数据元素对的粒度下作用。

[0074] 可提供向量带进位加法指令的加法与减法变体。对于加法变体,加法包括对第一和第二源数据值以及从进位源向量寄存器的第二数据元素获得的进位值进行相加。对于减法变体,运算包括从第一源数据值减去第二源数据值,且进位值指示减法的借位值。对于减法变体,进位输出代表减法的借位输出。注意到,减法仍可被视为是加法,因为两个操作数的相减等于两个操作数的相加(在一个操作数在执行加法之前为2的补码时)。类似地,减法的借位值可被视为加法的进位值,因为借位值仅对应于-1的进位而非对于加法的+1的进位。

[0075] 在提供了多于一个向量带进位加法指令的变体时,可由多种方式区分变体(第一/第二,或加法/减法)。例如,带进位加法指令的不同变体可具有不同的指令操作码。或者,变体可共享共同操作码,但可在指令编码中具有另一字段以区分变体。在另一示例中,指令的不同变体可具有相同的指令编码,但可由存储在设备的控制寄存器中的模式参数区分,控制寄存器可由前置指令设定以选择在遇到后续的向量带进位加法指令时应使用哪个指令变体。对于第一/第二变体,第一/第二元素选择也可被呈现于由指令读取的谓词或掩码寄存器中。

[0076] 上文讨论的向量带进位加法指令形式(使用进位源向量寄存器和目标向量寄存器的一些元素以传输进位信息用于加法/减法)的优点可为,可相对有效率地以微型架构实施,因为响应于指令仅需写入一个寄存器(目标向量寄存器)。再者,在累加器寄存器被指定为目标和第一源向量寄存器两者的示例中,指令仅需指定一个目标寄存器指定符和两个向

量源寄存器。

[0077] 然而,对于实施大整数值计算的问题的替代性解决方案,可为提供使用谓词寄存器的向量带进位加法指令以传递进位信息。数据处理设备可具有多个谓词寄存器,谓词寄存器包括谓词字段以存储谓词值以控制处理电路所执行的运算的掩码。尽管对于其他类型的向量指令,谓词字段可控制向量处理道的掩码,但对于向量带进位加法指令,可再使用谓词寄存器以呈现加法的进位输出,且也可从谓词寄存器的谓词字段获得加法的进位输入。

[0078] 因此,指令解码电路可响应于向量带进位加法指令(指定目标向量寄存器、第一数据源向量寄存器、第二数据源向量寄存器、输入谓词寄存器与输出谓词寄存器)以控制处理电路,针对目标向量寄存器的给定数据元素,由对应于第一源数据值、第二源数据值与进位值的加法结果的值更新目标向量寄存器的给定数据元素,第一源数据值获得自第一数据源向量寄存器的对应数据元素,第二源数据值获得自第二数据源向量寄存器的对应数据元素,进位值获得自输入谓词寄存器的对应谓词字段。再者,可由对应于该加法的进位输出的值更新输出谓词寄存器的对应谓词字段。

[0079] 尽管此方法可需要更复杂的微型架构(例如,针对谓词寄存器文件的较快速读取/写入路径,以及响应于同一指令而同时写入向量寄存器文件和谓词寄存器文件的能力),此方法的优点为进位未被存储在目标向量寄存器的元素中,因此可使用目标向量寄存器的每个元素以存储加法结果值(并可使用第一/第二数据源向量寄存器的每个元素作为源输入),有效地将响应于单个指令可执行的运算数增加一倍,而因此改进了性能。

[0080] 简言之,上文讨论的两个示例具有可直接对来自输入向量的完整数据元素运算而无需解封或调整运算大小的优点,因为进位信息没有被存储在与数据结果相同的道中,而是被存储在另一向量寄存器道中或谓词寄存器的对应谓词字段中。这对于加速长整数乘法而言是非常有用的。

[0081] 上文讨论的指令形式两者,可由使用控制处理电路的指令解码器的硬件来实施,以响应于向量带进位加法指令而执行所需的运算。例如,指令解码器可包括用于解释向量带进位加法指令的编码的逻辑门,以选择性启用适当的控制信号路径,以如前述控制处理电路而基于输入更新结果寄存器。

[0082] 然而,也可由处理器架构模拟来实施技术,而非以物理硬件来实施。因此,可提供仿真器计算机程序以控制主机处理设备(自身可不支持上文讨论的指令),以提供指令执行环境以执行目标程序代码指令,而在不支持这些指令的目标处理设备上仿真目标程序代码的执行。可由仿真器程序中的指令解码程序逻辑与处理程序逻辑执行指令解码器与处理电路的功能,且寄存器可被实施为存储器中的仿真寄存器数据结构,数据结构存储数据以代表仿真目标处理设备的寄存器。计算机程序可被存储在存储介质上。存储介质可为非暂态存储介质。

[0083] 图1示意性地例示了数据处理设备2的示例,数据处理设备2具有处理电路4以响应于指令解码器6解码的指令而执行数据处理操作。从存储器系统8获取程序指令,存储器系统8可包括一个或更多个缓存和存储器。指令解码器6解码指令以产生控制信号,控制信号控制处理电路4以处理指令,而执行由指令集架构限定的对应运算。例如,解码器6可包括逻辑门以解释操作码以及任何额外的指令控制字段,以产生控制信号,控制信号使处理电路4启用适当的硬件单元以执行运算(诸如算术运算、逻辑运算或加载/存储运算)。寄存器10存

储处理电路4响应于指令而要处理的数据值,以及用于配置处理电路运算的控制信息。响应于算术或逻辑指令,处理电路4从寄存器10读取操作数,并将指令结果写回寄存器10。响应于加载/存储指令,数据值被经由处理逻辑4传输在寄存器10与存储器系统8之间。

[0084] 寄存器10包括标量寄存器文件12,包括多个标量寄存器以存储标量值,标量值包括单个数据元素。标量寄存器可包括整数寄存器以存储整数操作数,以及浮点数寄存器以存储浮点数。或者,可将整数值与浮点数存储在相同的寄存器组中。在指令集架构中由指令解码器6支持的一些指令为标量指令,标量指令控制处理电路4以处理从标量寄存器12读出的标量操作数,以产生要写回标量寄存器12的标量结果。

[0085] 寄存器也包括向量寄存器文件14和谓词寄存器文件16。向量寄存器文件14包括多个向量寄存器,向量寄存器支持存储包括多个数据元素的向量操作数。指令解码器6支持向量指令,向量指令控制处理电路4以对从向量寄存器读出的各自向量操作数元素执行多个向量处理道,以产生要写入标量寄存器12的标量结果,或要写入向量寄存器14的另一向量结果。一些向量指令也可从一个或更多个标量操作数产生向量结果,或可对标量寄存器文件中的标量操作数执行额外标量运算,以及对从向量寄存器文件读出的操作数执行向量处理。因此,一些指令可以为标量向量混合式指令。如同在处理电路内触发算术或逻辑运算的向量算术或逻辑指令,解码器6也可支持向量加载/存储指令,向量加载/存储指令可在向量寄存器14与存储器系统8之间传输数据。

[0086] 图2示出了由谓词寄存器16之一中的谓词信息控制的示例向量运算。在此示例中,向量指令为向量加法指令,向量加法指令控制处理电路4以执行多个独立的加法运算,每个运算将两个输入向量Zm,Zn的对应数据元素对相加,并将加法结果写入目标向量Zd的对应元素。当然,可应用其他的算术或逻辑运算作为核心运算以在每个向量道中执行,而非仅是加法。谓词寄存器P的每个谓词字段控制是否由加法结果更新目标寄存器的对应元素。例如,在此示例中,向量道1、2与5被掩码(由于这些道的对应谓词字段是0),且因此目标寄存器Zd的对应元素不会被设成加法结果。例如,受掩码的道可保持它们先前的值。或者,清除/填零形式的谓词,可将目标寄存器的受掩码元素清除为零或另一预定值,同时非掩码道被由在此道中的加法结果写入。因此,通常可使用谓词寄存器16以控制特定道中的向量运算掩码。例如,条件向量指令可执行比较,以测试给定向量的每个道中的输入元素是否符合一些条件,并基于对应道中的比较结果设定谓词寄存器中的谓词值,且随后取决于谓词来掩码后续的向量指令,使得在每个向量道中此指令的效果被设为条件式并取决于先前条件向量指令执行的比较。

[0087] 尽管图2例示了针对给定向量并行执行所有的加法,但这并非必要的。处理电路4可支持具有各种不同数据元素大小和不同向量长度的向量的处理。例如,256位向量寄存器14可例如被分成32个8位数据元素、16个16位数据元素、8个32位数据元素、4个64位数据元素、或2个128位数据元素,或可被解释成单个256位元素。再者,可支持不同大小的向量寄存器,例如128、256或512位。例如,可由各种方式将向量寄存器库14分割,以提供各种向量寄存器大小(例如,多个具有固定大小的物理寄存器可被由单个向量指令结合并操作,以呈现较大的寄存器大小)。因此,如图1所示,寄存器10也可包括用于控制向量处理的控制信息,诸如指定用于向量运算的数据元素大小的元素大小参数18(向量的一个数据元素中的位数),以及指定向量寄存器长度的向量长度参数20(向量中的总位数,包括所有元素)。在一

些实施方式中,这些值可被硬联机,使得它们对于所有指令都总是相同的。提供指定元素大小18和向量长度20的寄存器仍可为有用的,使得为了在具有不同元素大小或向量长度的各种平台上执行所编写的代码,可读取正执行此代码的系统上实施的特定元素大小或向量长度。然而在其他示例中,元素大小和(或)向量长度可被编程,使得指令可指定针对给定运算要使用哪种元素大小或向量长度。或者,可由向量指令编码而非控制寄存器内的参数,来指定元素大小和(或)向量长度。

[0088] 因此,元素大小或向量长度可改变。取决于所选择的特定大小,向量处理电路可不总是具有足够的处理硬件来并行处理整个向量。若处理逻辑窄于用于给定运算的向量长度,则可由多个循环,多次通过较窄的处理逻辑来处理向量。因此,尽管向量指令可响应于单个指令触发处理电路4以在多个道中执行运算,但这不一定隐含着所有这些道必需被并行处理。在一种极端情况下,一些向量实施方式可仅提供对应于单道的处理逻辑,且随后依次处理所有向量道。在另一种极端情况下,较高性能的实施方式可使用多个并行执行单元并行处理所有向量道。其他实施方式可并行处理多个道,但由多个顺序块整体处理向量。

[0089] 将理解到,元素大小与向量长度指示18、20仅为可存储在寄存器10中的控制信息的一些示例。其他示例可包括程序计数器寄存器以指示代表当前执行点的指令地址、堆栈指针寄存器以指示堆栈数据结构的存储器8中的位置的地址(以在处理例外(exception)时存储或恢复状态)、以及链结寄存器以存储在功能执行之后处理要分支前往的功能返回地址。

[0090] 图3与图4示出了根据一个示例的向量带进位加法指令的加法和减法变体。对于加法与减法变体两者,示出了进一步的指令的第一变体和第二变体:

[0091] ADCLB Zda,Zn,Zm//带进位长加法(底)-第一变体、加法变体

[0092] ADCLT Zda,Zn,Zm//带进位长加法(顶)-第二变体、加法变体

[0093] SBCLB Zda,Zn,Zm//带进位长减法(底)-第一变体、减法变体

[0094] SBCLT Zda,Zn,Zm//带进位长减法(顶)-第二变体、减法变体对于每个变体,Zda代表目标向量寄存器(也作为第一源数据向量寄存器,并可称为累加器向量寄存器),Zn代表第二源数据向量寄存器,且Zm代表进位源向量数据寄存器。这些指令可用于加速大整数的处理,通过执行加法或减法使得进位或借位输入信息被与实际数据值交错在累加器向量寄存器与进位源向量寄存器中。

[0095] 因此,进位信息为源与目标向量寄存器的组成部分。有用的是将源向量与目标向量内的“进位”(或“借位”)信息的位置,限制在邻接于执行将“进位”作为输入(或产生“进位”)的数学运算的道。见图3与图4,其中寄存器Zda、Zn、Zm被分成元素对。每对包括对应于偶数元素的“底”元素(第一元素),以及对应于奇数元素的“顶”元素(第二元素)。对于寄存器Zda、Zm,进位信息位于每对的顶元素中,而底元素提供对应的数据。尽管其他示例可由不同方式设置每对的数据/进位元素(例如将所有数据元素放在寄存器的一半中并将对应的进位元素放在另一半中),但使数据与进位元素交错可为较有效率的,因为用于在Zda中产生邻接对元素的输入被从对应的Zn、Zm对元素中选择,而避免需要使用任何较长的跨元素路径(延展跨越向量的两个或更多个元素),这可使得处理电路的微架构实施方式更有效率。因此,这些运算有效地将每个向量分割成“粒度”或分段,大小为向量元素的两倍。

[0096] 图3示出了ADCLB与ADCLT指令的运算。为了清楚说明,仅图示说明一个“粒度”(邻

接顶元素与底元素对)的详细运算。将理解,对每个其他的粒度(元素对)执行相应的运算。ADCLB指令将一个源Zn的偶数元素B(在ADCLT变体的情况中为奇数元素T)相加至目标向量寄存器的偶数元素,使用第二源Zm中的每个粒度(奇数元素)的上半部的最低有效位作为进位输入。所产生的总和被写入目标向量中的每个粒度的下半部,而所产生的进位信息被写入目标中的每个粒度的上半部的最低有效位。Zda中每个粒度的上半部的剩余位保持未使用。结果独立于进位源寄存器Zm中每个粒度的底(偶数)元素的位。再者,对于ADCLB,结果独立于数据源寄存器Zn的未选择(顶/奇数)元素,且对于ADCLT,结果独立于Zn的未选择(底/偶数)元素。

[0097] 通过以此方式交错进位信息与输入或结果数据,可确保进位链为通常数据流的部分。这使能了数级并行性。首先,可存在一个以上的同时进行中的进位链,因为在向量寄存器的粒度处解析了相依性。第二,指令的B/T变体可被独立计算(在一些微架构中可同时计算,若B/T变体可被融合成单个微运算,或两个变体可被管线并行处理)。由于ADCL指令通常消费由先前乘法步骤产生的部分乘积,这些指令使得可能设计程序序列,使得乘法和加法步骤为“精简型”,且没有用于处理进位的额外运算,因为进位已经实质上被结合至累加器。这减少了管理进位信息的开销。

[0098] 尽管在图3与图4中每对元素的“第一元素”位于比第二元素更低的元素位置,使得结果被写入每对的底元素,但在其他实施方式中,在寄存器内第一元素和第二元素可被转置,使得结果更新每对的顶元素且进位被写入底元素。然而,在微架构中将进位提供在每对的较高有效元素是较有效率的,因为这可对应于其中位可由加法电路输出的排序,由于进位代表比加法数据结果的最高有效位更大的一个位位置。

[0099] 图4示出了向量带进位加法指令的减法变体的第一/第二变体的对应处理。SBCLB与SBCLT指令的运算或数据流分别类似于ADCLB与ADCLT,除了核心运算为“减法”以外(即同一加法,但在加法之前从Zn选择的元素被2补码化),且“借位”信息作为输入并在输出处产生而非“进位”信息(借位代表-1的进位)。Zn数据输入的2补码化以及借位反转两者,可简单地由将Zn数据源输入与借位输入值两者反转输入加法器来实施。不需要分别将1加至Zn数据源输入,因为借位输入值的反转有效地提供了此效果。在借位值为0的情况下,反转的借位输入值1对2补码提供1的加法,且不需要借位。在借位值为1的情况下,反转的借位输入值0反映了对于借位的1的减法抵销了对于2补码的1的加法。因此,一旦数据输入与借位输入值都已被反转,则可由与加法变体相同的方式来执行加法。

[0100] 图4A示出了向量带进位加法指令的谓词变体的示例。在此示例中,谓词被施加至图3示出的ADCLB变体,但可提供类似的ADCLT、SBCLB和SBCLT的谓词版本。对目标寄存器Zda的每对邻接顶/底元素执行的核心带进位加法运算,与图3相同。然而,指令也与谓词值P相关联,谓词值P例如存储在谓词寄存器16之一中。谓词值包括类似于图2的谓词字段的多个谓词字段,但针对图3和图4所示的类型的向量带进位加法指令,仅有一半的谓词字段为有效的,由于谓词被施加于邻接元素对的粒度处,而不是针对每个元素施加。在此示例中,有效谓词字段为对应于每对的底(偶数)元素的字段(且因此结果独立于对应于每对的顶元素的谓词字段,这些字段由X标记以指示这些字段的值不会影响结果)。然而,其他实施方式可选择使用针对每对元素的顶(奇数)谓词字段作为有效谓词字段。

[0101] 对于其中对应谓词指示为1的目标向量的元素对,由与图3相同的方式执行带进位

加法运算,以由加法产生的数据D更新此对的第一元素,并由带进位加法输出C更新此对的第二元素。然而,对于其中对应谓词指示为0的元素对(如在此示例中示出的元素4与5),目标向量Zda'的这些元素的更新被抑制。相对地,目标向量Zda'的这些元素可保持它们的先前值(合并谓词),或可被清除至零或另一预定值(填零谓词)。因此,谓词施加至每个粒度(元素对),而非施加至每个单独元素。将理解,图4A示出谓词寄存器16的一个示例格式,但可使用提供谓词指示(指示是否应掩码目标寄存器Zda'的每对元素)的任何其他技术。

[0102] 图5是示出向量带进位加法指令的处理的流程图,向量带进位加法指令指定目标(累加器)向量寄存器Zda、第二数据源向量寄存器Zn以及进位源向量寄存器Zm(在此示例中目标向量寄存器Zda也作为第一数据源向量寄存器)。在步骤50,指令解码器判定是否已检测到这种向量带进位加法指令,且在未检测到时根据当前指令所代表的运算来处理当前指令。在遇到向量带进位加法指令时,在步骤52指令解码器判定指令是"顶"(第二)还是"底"(第一)变体。若指令为顶变体,则在步骤54,从第二数据源向量寄存器Zn中的每对元素选择的元素将为此对的顶元素(换言之,为奇数元素(第二元素)),而若指令为底(第一)变体,则在步骤56,从第二数据源向量寄存器Zn选择每对的底(第一或偶数)元素。

[0103] 在步骤58,指令解码器判定指令是加法变体还是减法变体。若指令为加法变体,则在步骤60,针对目标向量寄存器Zda的每对元素,将此对的底(第一)元素的新值Zda(B)'设定为:(1)目标向量寄存器中此元素对的底元素的先前值Zda(B);(2)在步骤54或56选择的第二数据源寄存器Zn中对应元素对的选定元素(顶或底);以及(3)从进位源向量寄存器Zm中对应元素对的顶元素Zm(T)提取出的进位输入。再者,目标向量寄存器Zda中对应元素对的顶元素Zda(T)'的最低有效位被设为加法的进位输出。

[0104] 若指令为减法变体,则在步骤62,目标向量寄存器的每对元素的底元素Zda(B)'被设定为:此底元素的先前值Zda(B)减去在步骤54或56选择的数据源向量Zn中对应元素对的值,减去从进位源向量寄存器源Zm中对应元素对的顶元素Zm(T)提取出的进位值所代表的借位值。再者,目标向量寄存器对应元素对的顶元素Zda(T)'的最低有效位,被设为减法的借位输出(即,Zda(T)的lsb被设为执行减法的加法器电路的进位输出上的输出值)。

[0105] 尽管图3至图5的示例示出了破坏性指令格式(其中同一寄存器Zda作为目标寄存器和第一数据源寄存器两者),可提供将不同于目标寄存器的另一向量寄存器指定为第一数据源寄存器的非破坏性指令格式,以避免在写入结果至目标寄存器时覆写第一源向量。

[0106] 下面讨论这些指令的示例使用情况。

[0107] 数种重要的密码学工作负载的最耗时的部分,涉及对大整数值执行数学运算的例程。除了密码学方案以外,涉及大数的数学运算支持资源库,诸如用于财务软件和一些科学应用的GNU Multiprecision Arithmetic(GMP)资源库。这种数字通常被使用在由于重现性或数学严谨性的考虑而不适合使用浮点运算(不考虑其范围)的实例中。RSA算法为最常被使用的公钥密码学算法。RSA算法的安全性,依赖将大数因子分解的所感知到的难度。为了加密或解密信息,处理器需要在称为模幂运算的技术中尽快地将大数相乘。例如在RSA2048中,有许多2048位数字的连续乘法,给出4096位乘积,随后被减少。

[0108] 整数相乘的最直接方法(常称为"教科书"方法),对于n位源需要$O(n^2)$个步骤,因为一个源中的每一位需要乘以另一源中的每一位。存在所需要的"加法"和"数据再对齐"(移位)运算,如下列示例所示:

[0109] 111*543＝333+(444<<1位)+(555<<2位)＝60273

[0110] 然而,Karatsuba算法可通过使用分治法,将两个n位数字的乘法减少至最多$O(n^{log_2 3})$个位乘法。此方法在“乘法”与“加法”、“减法”等之间进行权衡,并在大的“n”之下得出良好的性能优点。每个具有大小“n”输入操作数的乘法,被分成多个具有大小“n/2”的输入操作数的较小乘法,以及一些辅助运算:“加法”、“减法”等。可正式地编写将两个n位整数A与B相乘而产生2n位结果C的表示式,如图6顶部的方程式所示。注意到在此示例中,使用Karatsuba算法的减法变体,以确保部分乘积不会具有难以处理的大小。因此,两个2048位整数的乘法,可被分解成三次1024位整数的子乘法,如图6所示。每个1024位乘法自身可被由相同方式分解成三次512位整数乘法。每个512位乘法随后可进一步被分解成三次256位整数乘法。最后会到达一点,此时由于再结合总体结果所需的额外加法、减法或移位,再进一步分解乘法会产生较大的总和开销。

[0111] 因为子乘法为独立的,在适当的数据布局与组织下,可能将Karatsuba算法向量化,如图6所示。例如,若Karatsuba树的叶节点提供256位子乘法,则Karatsuba树的给定分支的三个256位子乘法的每一者可被分配至向量的单独道A、B、C,如图6中上方示例所示。再者,在较大的向量之下,Karatsuba树的较高位准处的不同的512x512位乘法的6个256位子乘法,可被分配至6道8元素向量(每组三个道A、B、C代表对应512位节点的三个256x256位乘法),如图6中下方示例所示。

[0112] 例如,若选择64位元素大小,则可将两个256位整数A与B相乘,如下:A x B＝{a_3,a_2,a_1,a_0}x{b_3,b_2,b_1,b_0}。图7绘制了属于此乘法的部分乘积。注意到,乘积是从右到左交错的,因为每个部分乘积需要被“再对齐”以在适当的列中累加。

[0113] 图8示出了代码序列,呈现为了执行图7中第二“支”的计算以计算部分乘积a_1 x{b_3,b_2,b_1,b_0}所需要的一个可能的指令序列实例。代码序列开始于一些乘法,以产生各个部分乘积的低与高64位部分。注意到,每个256位x256位乘法实例占用向量处理中的单道,且每个256位操作数的64位分量垂直分布(在由单独指令作用于的单独寄存器中),而非跨越每个向量。这隐含着为了产生对于a_1 x{b_3,b_2,b_1,b_0}计算的部分乘积,寄存器z_1(包括来自不同源操作数的a_1)需要乘以z_4(从多个不同的源操作数提供b_0)、z5(b_1)、z6(b_2)与z7(b_3)(图8中第1行至第8行)的每一者。

[0114] 在图8中,adclb与adclt指令用于将部分乘积加入对于每道的累加器:zacc1LB、zacc1LT、zacc0HB等。执行对应的adclb与adclt指令对,以分别处理来自乘法的部分乘积的偶数与奇数元素。由暴露4路并行处理的方式来使用指令:B&T变体为独立的,并因此可被有益地排程。此外,存在两个进位链,进位链能够被同时处理。一个进位链开始于第10行,且另一个进位链开始于第12行。随后可通过第14、18、26行上的指令追踪“B”变体的第一进位链。可通过第16、20、29行的指令追踪第二进位链。存在对于“T”变体的类似链。第32、33行的指令被用于合并先前的独立进位链,同时将最终部分乘积(a1×b_3(H))加入累加器。

[0115] 图9和图10A至图10I示出了使用向量带进位加法指令的另一示例。图9示出了用于执行乘法{a_3,a_2,a_1,a_0}x b_i的部分乘积的加法的ADCLB指令的示例指令序列,这代表较大乘法的单支,此支可为用于将大整数相乘的Karatsuba树的部分。图9中假定部分乘积已被产生并存储在向量寄存器P_{0i}lo,P_{1i}lo…,P_{3i}hi中(其中P_{ji}lo和P_{ji}hi代表a_j与b_i乘积的下半部与上半部)。AccB[i]至AccB[i+4]代表5个累加器寄存器,分别用于存储第0列至第4列中示

出的乘积{a_3,a_2,a_1,a_0}x b_i的5个部分。每对向量道代表施加至不同输入值的完全独立的计算(相较于其他向量道),因此对于图7示出的给定计算类型所需的每个部分乘积加法,被通过图9的指令序列执行在单对道内(例如,每对道中的计算对应于Karatsuba树的不同节点)。为了简洁说明,图10A至图10I仅示出了对偶数向量道执行的ADCLB指令,可对奇数道执行未示出的对应ADCLT指令。为了清楚说明,将第一进位链示为不具有星号(例如C_{AccBi})并将第二进位链示为具有星号(例如C_{AccBi*}),来区分两个独立的进位链。

[0116] 图10A:AccB[i]代表第0列中的累加。对于AccB[i]中的每对元素,此对的下(偶数)元素被设为以下的总和:AccB[i]中下元素的先前值;P_{0i}lo中相应元素对的底元素;以及作为进位输入的0(由于这是结果的最低有效部分,且因此不需要来自更低加法的进位)。加法结果已呈现对于此部分结果的末端结果值Res[i],因为没有进一步的进位能够被注入结果的最低有效部分。来自加法的进位输出C_{accBi}被写入AccB[i]中对的顶元素的最低有效位,并代表对第1列最低有效位均等有效的进位位。

[0117] 图10B:AccB[i+1]代表第1列中的累加。对于AccB[i+1]中的每对元素,此对的下(偶数)元素被设为以下的总和:AccB[i+1]中下元素的先前值;P_{1i}lo中相应元素对的底元素;以及作为进位输入的C_{accBi}。换言之,先前ADCLB指令的目标寄存器AccB[i]被指定为下一ADCLB指令的进位源向量寄存器,使得在第0列中先前加法产生的进位C_{accBi}被输入作为对于第1列中加法的进位输入。来自第1列累加的进位输出$C_{accBi+1}$被存储在AccB[i+1]中每对的顶元素的最低有效位,以代表要施加至第2列中累加的最低有效位的进位。

[0118] 图10C:AccB[i+2]代表第2列中的累加。再次说明,先前指令的结果向量AccB[i+1]变为下一指令的进位源向量寄存器,使得对于每对元素,第1列中先前加法产生的进位$C_{accBi+1}$被从AccB[i+1]中相应对的顶元素提取出,并被加入AccB[i+2]与P_{2i}lo的累加,这被写入AccB[i+2]中元素对的底部,且产生的进位输出$C_{accBi+2}$被存储在AccB[I+2]中每对元素的顶元素中。

[0119] 图10D:此指令标记第二进位链的开始,独立于通过图10A至图10C所采取的进位链。此指令代表部分乘积P_{0i}hi的第1列中对来自图10B的AccB[i+1]的加法。注意到,AccB[I+1]的奇数元素中的进位值已被在图10C中消耗,因此可被由新进位值$C_{accBi+1*}$在奇数元素中覆写,新进位值$C_{accBi+1*}$来自AccB[i+1]的偶数元素与P_{0i}hi的偶数元素的每个加法。

[0120] 图10E:此指令为第一进位链的部分,并使用来自图10C的进位输出$C_{accBi+2}$(存储在AccB[i+2]中每对元素的高元素)作为对AccB[i+3]与P_{3i}lo中每对元素的低元素的加法的进位输入(代表第3列中的加法)。所产生的进位输出$C_{accBi+2}$被存储在AccB[i+3]中每对元素的高元素。

[0121] 图10F:此指令在第二进位链中使用来自图10D的进位输出$C_{accBi+1*}$(对第2列最低有效位均等有效),作为对第2列中AccB[i+2]与P_{1i}hi加法的进位输入。

[0122] 图10G:此指令在第一进位链中对AccB[i+4]的偶数道与P_{3i}hi执行向量加法,且每个加法的进位输入来自AccB[i+3]的奇数道(来自图10E)。尽管指令仍可将各自加法的进位输出写入AccB[i+4]的奇数元素的最低有效位,实际上任何随后的指令不需要这些进位,因为此累加代表结果的最高有效位。

[0123] 图10H:此指令在第二进位链中对AccB[i+3]的偶数道与P_{2i}hi执行向量加法,且每个加法的进位输入$C_{accBi+2*}$来自AccB[i+2]的奇数道(来自图10F)。

[0124] 图10I:最后,此指令在第二进位链中对AccB[i+4]的偶数道与填零(zeroed)数据源寄存器执行向量加法(第二数据输入为零,因为仅有一个部分乘积要被加入最高有效列,这已由图10G中的指令加入)。每个加法的进位输入$C_{accBi+2*}$来自AccB[i+3]的奇数道(来自图10G)。每个加法的结果被写入累加器寄存器AccB[i+4]的偶数道。再次说明,尽管可由每个加法的进位输出写入奇数道,作为指令的正常行为部分,但这些进位对于任何进一步的指令是不需要的。

[0125] 因此,此示例示出了ADCLB指令如何有效率地允许乘法部分乘积的加法的进位信息被与数据结果自身一同传输,以使能施加至两位数幂的元素的输入向量(例如,P_{0i}lo to P_{3i}hi)的这种乘法的有效率的计算,使得在执行加法之前不需要将向量乘法指令的结果解封。这改进了涉及长整数乘法的计算性能,例如使用Karatsuba算法。

[0126] 图11示出了实施带进位加法指令的替代方式。相对于采取来自向量寄存器交替元素的进位输入,从谓词寄存器文件16获得来自输入谓词寄存器Pm的对应谓词字段的进位输入。类似地,进位输出被写入目标谓词寄存器Pd2中的相应谓词字段。通过此方法,可由真实数据值完全占用向量,因为不需要在每个其他元素中呈现进位。因此,对于八元素向量,这将允许响应于同一指令执行八个单独累加。结果向量Zda1的每个元素被设为此元素的先前值与数据源向量Zn的对应元素的和或差。从输入谓词寄存器Pm的对应谓词字段提取出加法的进位输入,且来自加法的进位输出被写入输出谓词寄存器Pd2的对应谓词字段。再次说明,可类似于图3与图4所示的来提供加法和减法变体,使得进位输入/输出可代表进位或借位信息(借位代表要从下一最高有效位进行的扣除,而进位代表要对下一最高有效位进行的相加,即借位为进位-1)。

[0127] 尽管图11中所示的方法可产生较紧密封装的向量寄存器文件并因此使能在半数的指令中执行累加(例如,不需要单独的ADCLB与ADCLT指令,因为元数据向量的所有元素可被在一个指令中处理),但这可需要较复杂的微架构修改,因为除了写入向量寄存器文件以外指令需要额外写入谓词寄存器文件,这对于许多向量微架构而言是不常见的。因此,要使用图3、4或图11中的方法,可取决于软件效率与硬件微架构效率之间的权衡。

[0128] 图12示出了带进位加法指令的谓词形式的处理的流程图。在步骤100,检测是否遇到这种向量带进位加法指令,若未检测到,则指令解码器6控制处理电路4执行由所检测到的指令类型代表的其他运算。若指令为向量带进位加法指令,则在步骤102,检测指令是加法变体还是减法变体。注意到,对于此指令而言没有第一和第二(顶或底)变体,因为通过移动进位信息以分离谓词寄存器,不需要保存一半的元素给进位,且因此可能在单一指令中对数据源寄存器的所有元素进行处理。

[0129] 若向量带进位加法指令为加法变体,则在步骤104,目标寄存器Zda1(i)'的每个元素i的新值被设定为等于此数据元素先前值Zda1(i)、数据源向量寄存器的对应数据元素Zn(i)以及从输入谓词寄存器Pm(i)的对应谓词字段i取得的进位输入的总和。再者,输出谓词寄存器的对应谓词字段Pd2(i)被设为等于加法的进位输出。

[0130] 若指令为减法变体,则在步骤106,目标寄存器的每个元素Zda1(i)'被设为新值,新值对应于此元素的先前值Zda1(i)减去数据源寄存器的对应元素Zn(i),减去输入谓词寄存器Pm(i)的对应谓词字段所指示的借位值。再次说明,输出谓词寄存器的对应谓词字段Pd2(i)被设为等于减法的借位输出。将理解到,对每个各自的元素位置(即i=0...N-1,其

中N为向量中元素的总数)单独执行步骤104与106。

[0131] 尽管在图11和图12示出了示例,其中相同寄存器Zda1充当目标向量寄存器和第一源向量寄存器两者,但同样可以提供非破坏性变体,其中将单独的寄存器指定为第一源向量寄存器。

[0132] 也可提供图11示出的带进位加法指令的谓词变体,其中以与图2所示类似的方式逐元素地施加谓词。

[0133] 图13例示了可使用的仿真器实施例。尽管前述实施例在用于操作支持有关技术的特定处理硬件的设备与方法来实施,但也可提供根据本文所述实施例的通过使用计算机程序来实施的指令执行环境。这种计算机程序通常被称为仿真器,只要它们提供硬件架构的基于软件式实施方式。各种仿真器计算机程序包括仿真器、虚拟机、模型、与二进制转译器(包括动态二进制转译器)。通常来说,仿真器实施方式可运行在主机处理器230上,可选地运行支持仿真程序210的主机操作系统220。在一些设置中,硬件与所提供的指令执行环境和/或提供在同一主机处理器上的多个不同的指令执行环境之间可存在多个模拟层。在过去历史上,需要强大的处理器以提供由合理速度执行的仿真器实施方式,但这种方法在一些情况中可被解决,诸如在由于兼容性或再使用理由而需要运行另一处理器的本机代码时。例如,仿真器实施方式可提供具有主机处理器硬件不支持的额外功能的指令执行环境,或提供通常与不同硬件架构相关联的指令执行环境。Robert Bedichek所著的“Some Efficient Architecture Simulation Techniques”(Winter 1990USENIX Conference)第53至63页说明了模拟的概述。

[0134] 就先前已参考特定硬件构造或特征说明了实施例而言,在模拟的实施例中,可通过合适的软件构造或特征来提供等效的功能。例如,可在模拟的实施例中实施特定电路为计算机程序逻辑。类似地,诸如寄存器或缓存的存储器硬件,可被实施在模拟的实施例中作为软件数据结构。在其中前述实施例已提及的一种或多种硬件组件存在于主机硬件(例如主机处理器230)上的设置中,一些模拟的实施例可在适合时使用主机硬件。

[0135] 仿真器程序210可被存储在计算机可读取存储介质(可为非暂态介质)上,并对目标代码200(可包括应用、操作系统与管理器)提供程序接口(指令执行环境),程序接口与仿真器程序210所建模的硬件架构的应用程序接口相同。因此,可从使用仿真器程序210的指令执行环境内执行目标代码200的程序指令,使得实际上不具有前述设备2硬件特征的主机计算机230可仿真这些特征。例如,仿真器程序210可分别包括指令解码程序逻辑212、处理程序逻辑214、以及寄存器数据结构216(功能上对应于指令解码器6)、处理电路4以及寄存器10。例如,解码程序逻辑212可包括仿真器程序210的一系列的“若(if)”语句,以检查目标代码200的指令的指令编码以判定要执行的操作,且处理程序逻辑214可对应于要对特定指令启用的“则(then)”例程,以将它们映射至要由主机操作系统220执行的对应指令。寄存器数据结构216可包括存储器区域,存储器区域经分配以仿真正由仿真器程序210仿真的仿真式设备2的寄存器。

[0136] 在本申请中,词语“被配置为.....”用于表示设备的组件具有能够执行所定义的操作的配置。在此背景内容中,“配置”表示硬件或软件的互连的布置或方式。例如,设备可具有提供所定义的操作的专用硬件,或者可以对处理器或其他处理设备进行编程以执行该功能。“配置为”并不意味着需要以任何方式改变设备组件以便提供定义的操作。

[0137] 尽管本文已参照附图详细说明了本发明的说示例性实施例,但应理解,本发明并不限于这些精确的实施例,且本领域技术人员可进行各种改变和修改,而不脱离如所附权利要求限定的本发明的精神与范围。

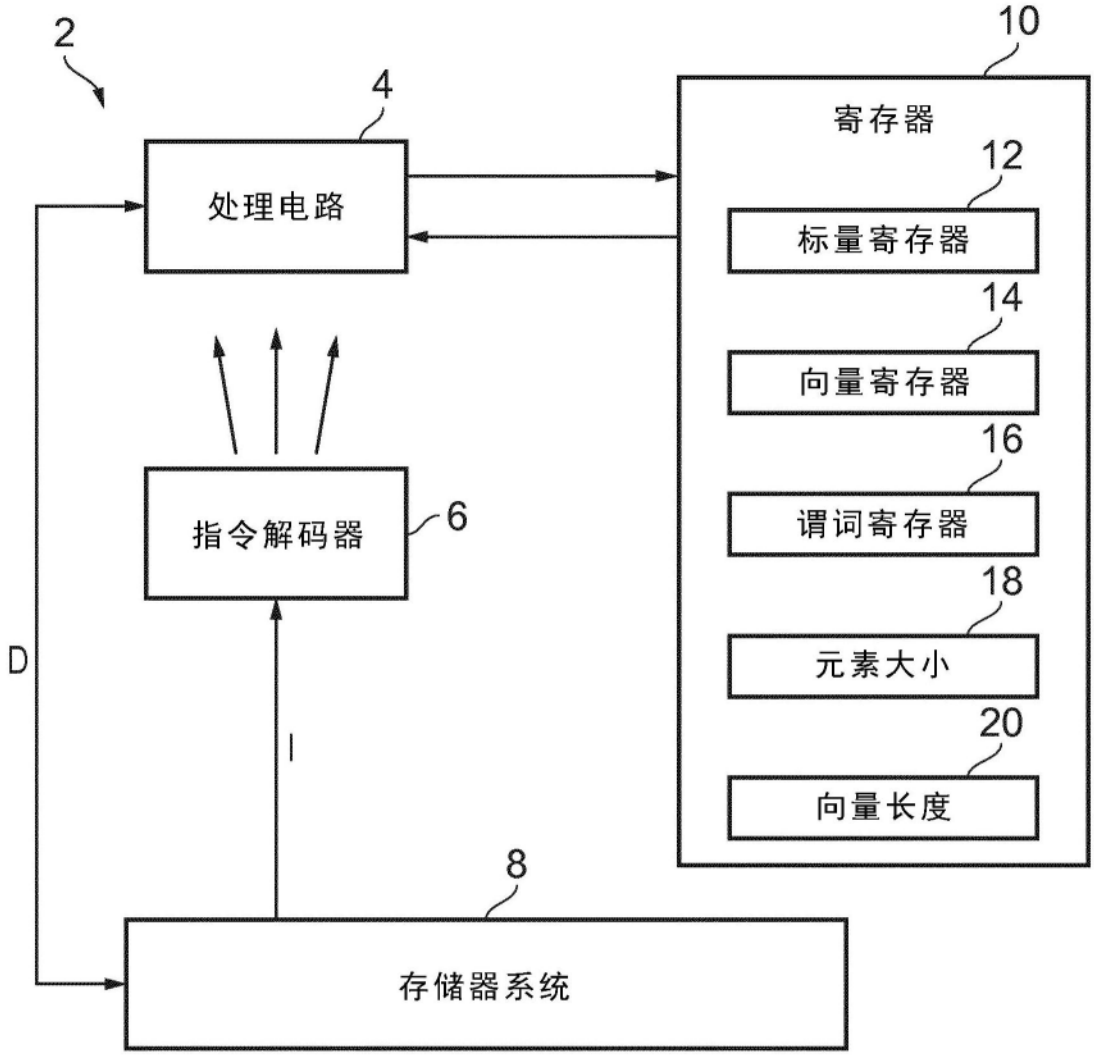

2
4
处理电路
10
寄存器
12
标量寄存器
14
向量寄存器
16
谓词寄存器
18
元素大小
20
向量长度
指令解码器
6
D
I
8
存储器系统

图1

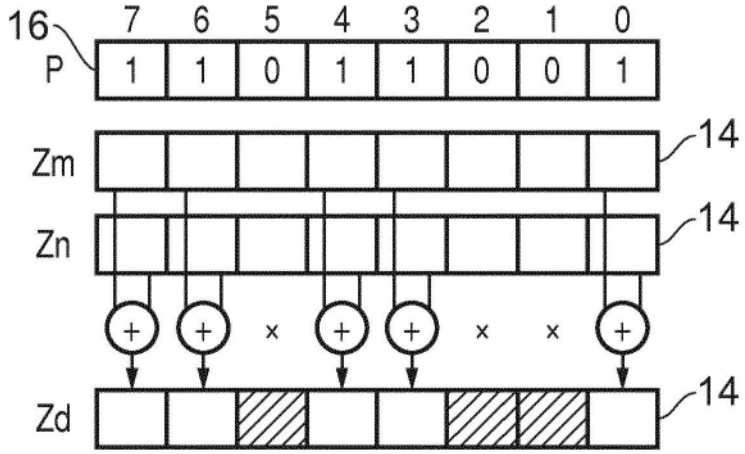

16
7 6 5 4 3 2 1 0
P
1 1 0 1 1 0 0 1
Zm
14
Zn
14
Zd
14

图2

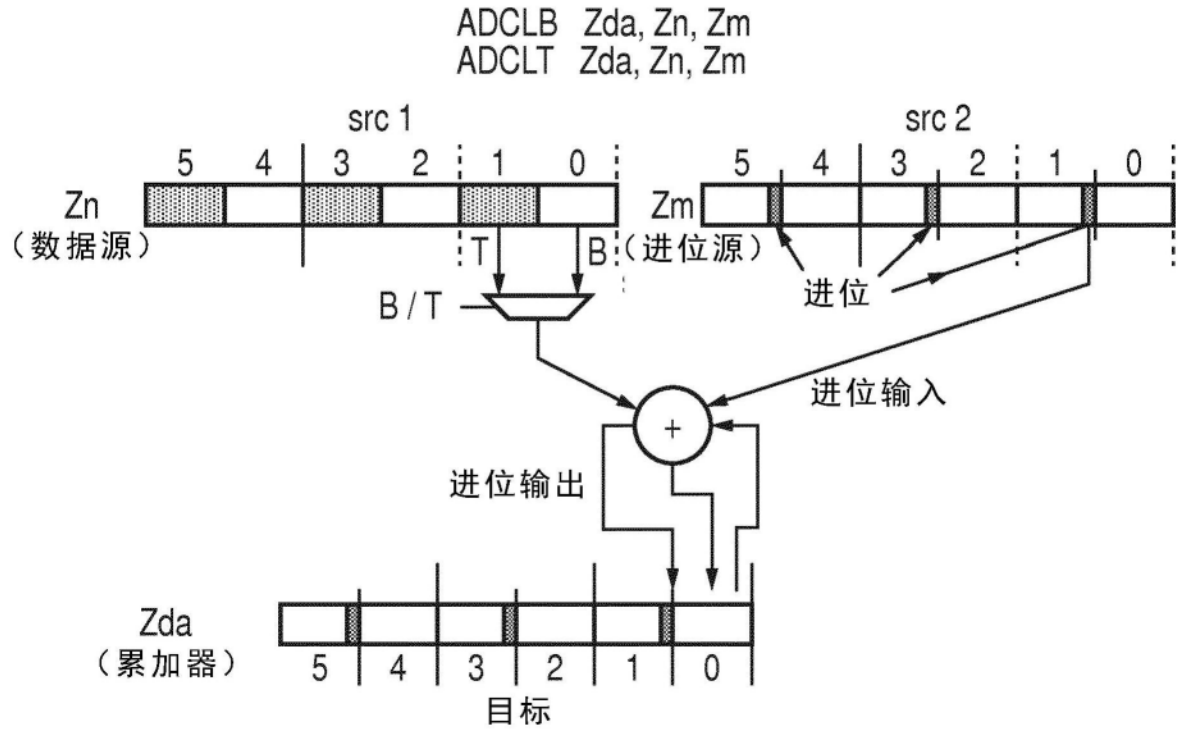

ADCLB Zda, Zn, Zm
ADCLT Zda, Zn, Zm
src 1
src 2
5 4 3 2 1 0
Zn
（数据源）
Zm
（进位源）
T
B
B / T
进位
进位输入
+
进位输出
Zda
（累加器）
目标

图3

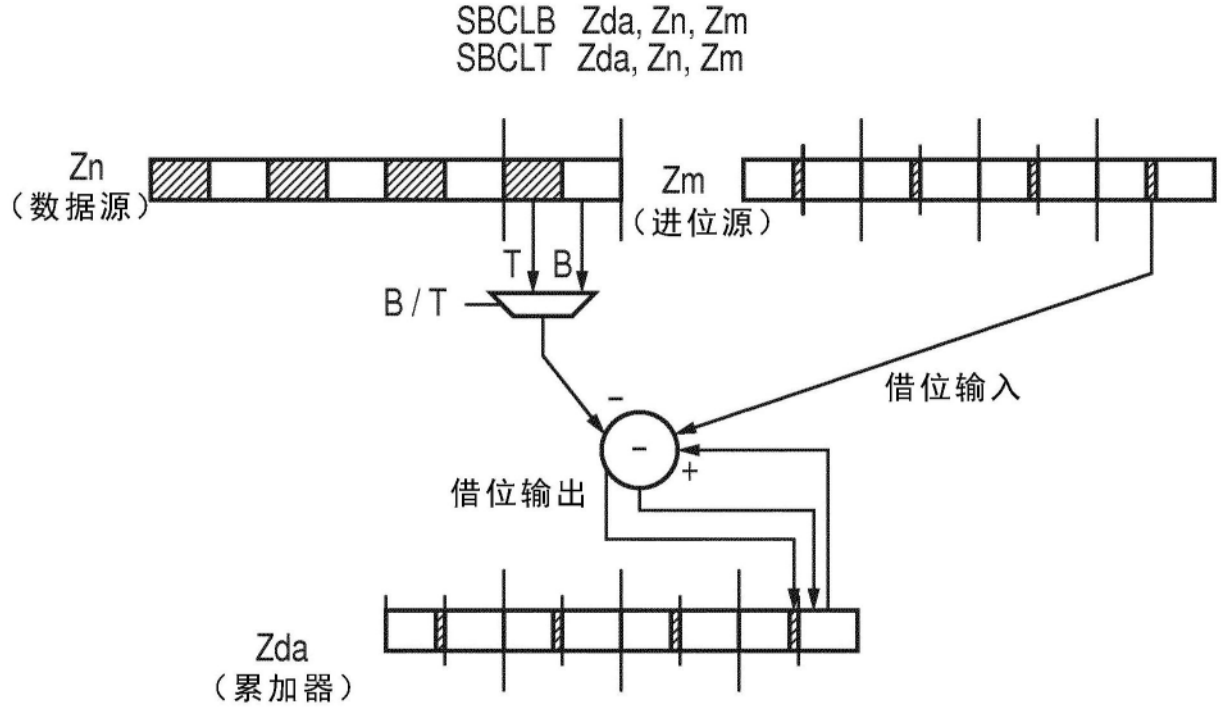

SBCLB Zda, Zn, Zm
SBCLT Zda, Zn, Zm
Zn
（数据源）
Zm
（进位源）
T
B
B / T
借位输入
−
−
+
借位输出
Zda
（累加器）

图4

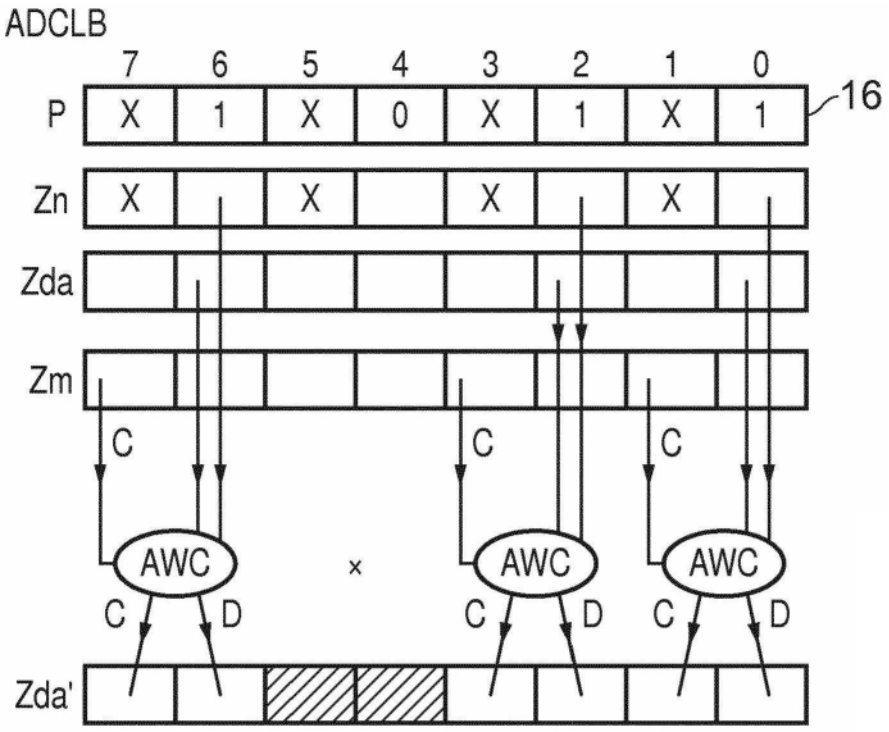

ADCLB
7 6 5 4 3 2 1 0
P X 1 X 0 X 1 X 1
16
Zn X X X X
Zda
Zm
C C C
AWC AWC AWC
C D C D C D
Zda'

图4A

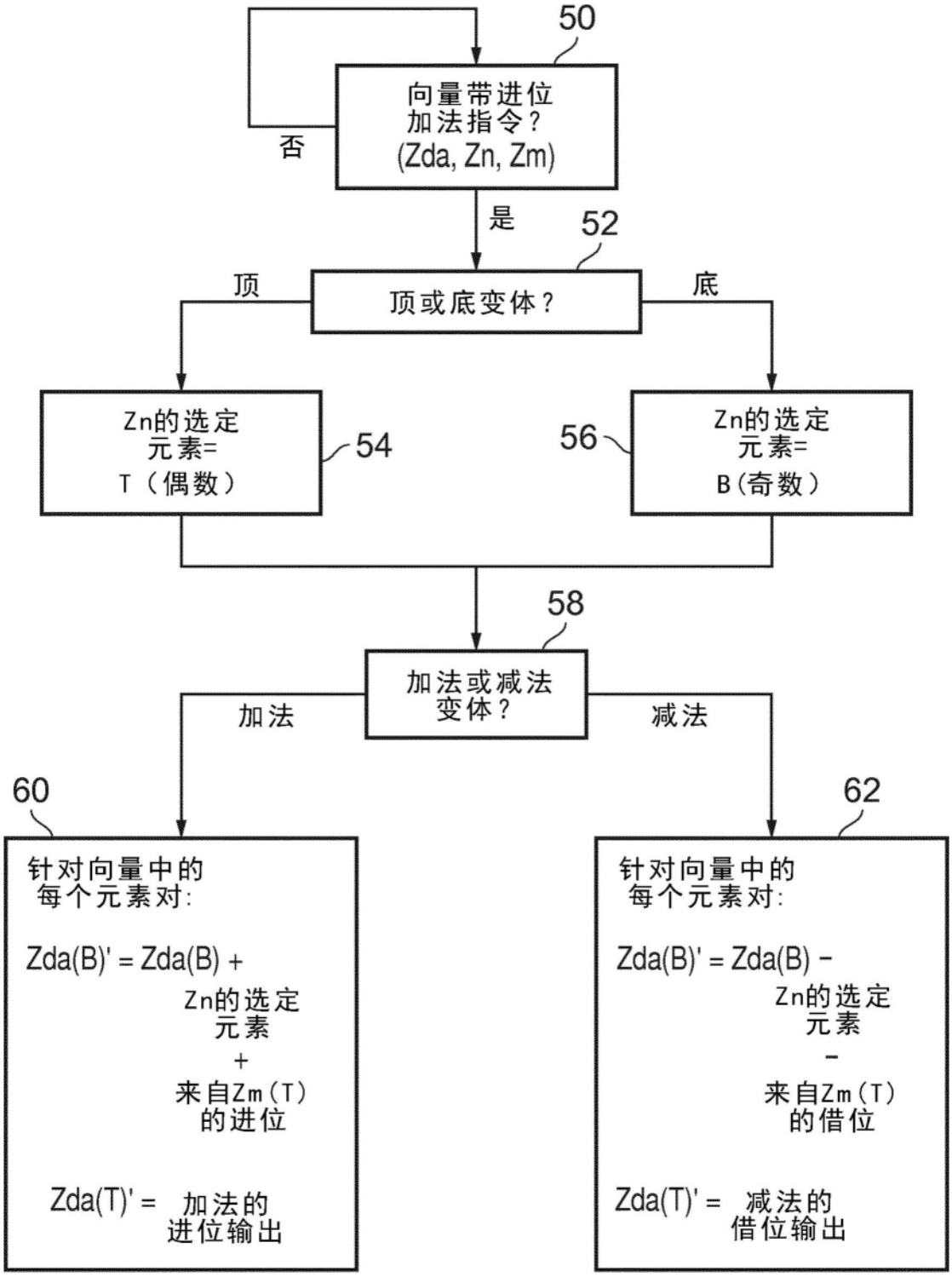

50
向量带进位
加法指令？
(Zda, Zn, Zm)
否
是
52
顶或底变体？
顶
底
Zn的选定
元素=
T（偶数）
54
56
Zn的选定
元素=
B（奇数）
58
加法或减法
变体？
加法
减法
60
针对向量中的
每个元素对:
Zda(B)' = Zda(B) +
Zn的选定
元素
+
来自Zm(T)
的进位
Zda(T)' = 加法的
进位输出
62
针对向量中的
每个元素对:
Zda(B)' = Zda(B) -
Zn的选定
元素
-
来自Zm(T)
的借位
Zda(T)' = 减法的
借位输出

图5

$$C = A_H \cdot B_H 2^n + ((A_L + A_H) \cdot (B_L + B_H) - A_L \cdot B_L - A_H \cdot B_H) 2^{n/2} + A_L \cdot B_L$$

$$(A_L + A_H) \cdot (B_L + B_H) - A_L \cdot B_L - A_H \cdot B_H = A_L \cdot B_L + A_H \cdot B_H - |A_H - A_L| \cdot |B_H - B_L|$$

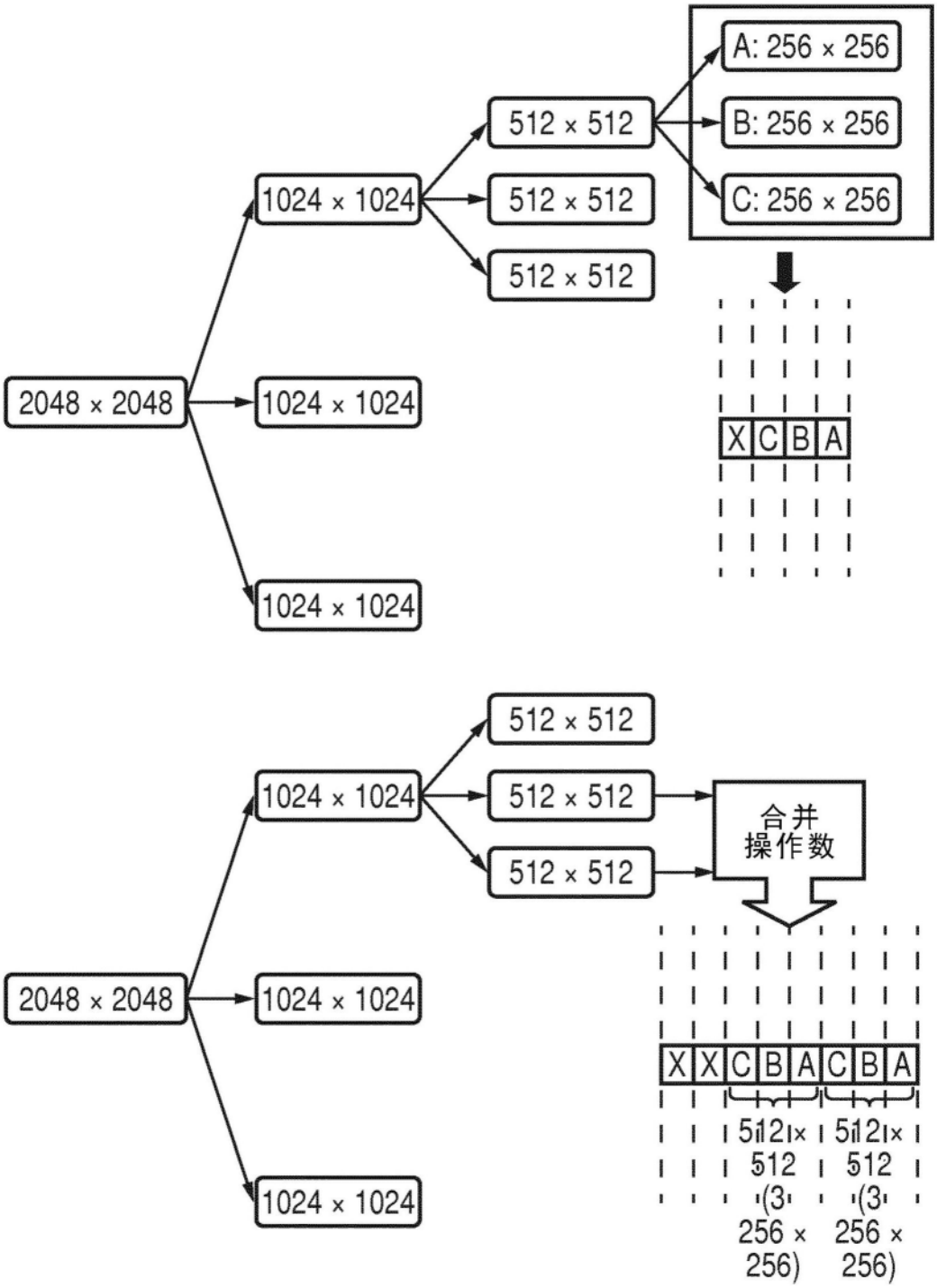


图6

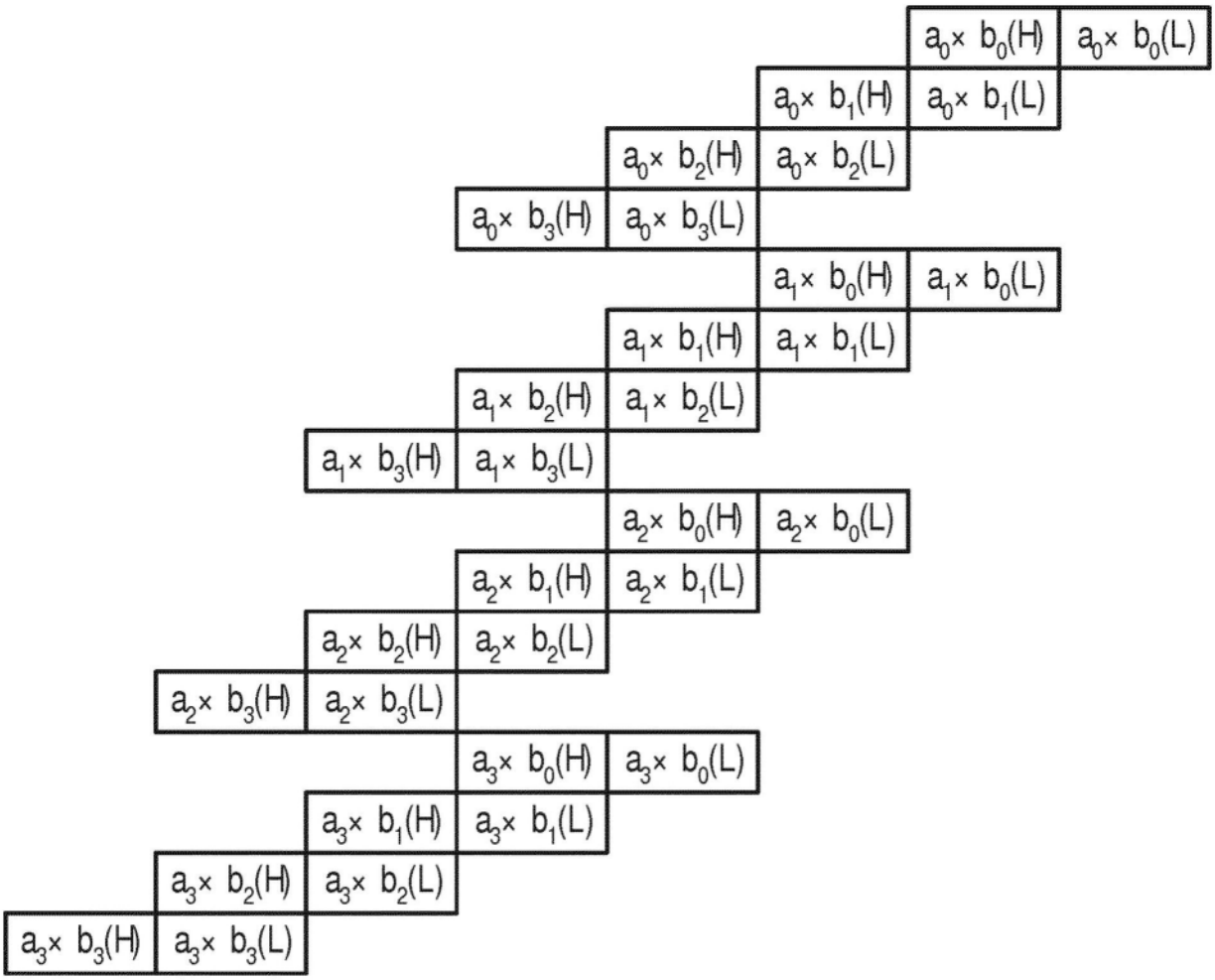

a0× b0(H)
a0× b0(L)
a0× b1(H)
a0× b1(L)
a0× b2(H)
a0× b2(L)
a0× b3(H)
a0× b3(L)
a1× b0(H)
a1× b0(L)
a1× b1(H)
a1× b1(L)
a1× b2(H)
a1× b2(L)
a1× b3(H)
a1× b3(L)
a2× b0(H)
a2× b0(L)
a2× b1(H)
a2× b1(L)
a2× b2(H)
a2× b2(L)
a2× b3(H)
a2× b3(L)
a3× b0(H)
a3× b0(L)
a3× b1(H)
a3× b1(L)
a3× b2(H)
a3× b2(L)
a3× b3(H)
a3× b3(L)

图7

```
1   mul     za1b0L, z1.d, z4.d
2   mulh    za1b0H, z1.d, z4.d
3   mul     za1b1L, z1.d, z5.d
4   mulh    za1b1H, z1.d, z5.d
5   mul     za1b2L, z1.d, z6.d
6   mulh    za1b2H, z1.d, z6.d
7   mul     za1b3L, z1.d, z7.d
8   mulh    za1b3H, z1.d, z7.d
9
10  adclb   zacc1LB, za1b0H, z_zero
11  adclt   zacc1LT, za1b0H, z_zero
12  adclb   zacc0HB, za1b0L, z_zero
13  adclt   zacc0HT, za1b0L, z_zero
14  adclb   zacc1HB, za1b1H, zacc1LB
15  adclt   zacc1HT, za1b1H, zacc1LT
16  adclb   zacc1LB, za1b1L, zacc0HB
17  adclt   zacc1LT, za1b1L, zacc0HT
18  adclb   zacc2LB, za1b2H, zacc1HB
19  adclt   zacc2LT, za1b2H, zacc1HT
20  adclb   zacc1HB, za1b2L, zacc1LB
21  adclt   zacc1HT, za1b2L, zacc1LT
22  //store
23  trn1    ztemp, zacc0HB, zacc0HT
24  st1d    {ztemp}, p1, [zptr, #8]
25  //**
26  adclb   zacc2HB, z_zero, zacc2LB
27  adclt   zacc2HT, z_zero, zacc2LT // 无法进行进位输出
28
29  adclb   zacc2LB, za1b3L, zacc1HB
30  adclt   zacc2LT, za1b3L, zacc1HT
31  //fixup carry
32  adclb   zacc2HB, za1b3H, zacc2LB
33  adclt   zacc2HT, za1b3H, zacc2LT
34  trn2    zacc3LB, zacc2HB, z_zero
35  trn2    zacc3LT, zacc2HT, z_zero
```

图8

$$\mathrm{ADCLB}(\mathrm{AccB}[i], P_{0i}\mathrm{lo}, 0) \rightarrow \mathrm{AccB}[i], c_{\mathrm{AccBi}}$$

$$\mathrm{ADCLB}(\mathrm{AccB}[i+1], P_{1i}\mathrm{lo}, c_{\mathrm{AccBi}}) \rightarrow \mathrm{AccB}[i+1], c_{\mathrm{AccBi+1}}$$

$$\mathrm{ADCLB}(\mathrm{AccB}[i+2], P_{2i}\mathrm{lo}, c_{\mathrm{AccBi+1}}) \rightarrow \mathrm{AccB}[i+2], c_{\mathrm{AccBi+2}}$$

$$\mathrm{ADCLB}(\mathrm{AccB}[i+1], P_{0i}\mathrm{hi}, 0) \rightarrow \mathrm{AccB}[i+1], c_{\mathrm{AccBi+1}}$$

$$\mathrm{ADCLB}(\mathrm{AccB}[i+3], P_{3i}\mathrm{lo}, c_{\mathrm{AccBi+2}}) \rightarrow \mathrm{AccB}[i+3], c_{\mathrm{AccBi+3}}$$

$$\mathrm{ADCLB}(\mathrm{AccB}[i+2], P_{1i}\mathrm{hi}, c_{\mathrm{AccBi+1}}) \rightarrow \mathrm{AccB}[i+2], c_{\mathrm{AccBi+2}}$$

$$\mathrm{ADCLB}(0, P_{3i}\mathrm{hi}, c_{\mathrm{AccBi+3}}) \rightarrow \mathrm{AccB}[i+4], 0$$

$$\mathrm{ADCLB}(\mathrm{AccB}[i+3], P_{2i}\mathrm{hi}, c_{\mathrm{AccBi+2}}) \rightarrow \mathrm{AccB}[i+3], c_{\mathrm{AccBi+3}}$$

$$\mathrm{ADCLB}(\mathrm{AccB}[i+4], 0, c_{\mathrm{AccBi+3}}) \rightarrow \mathrm{AccB}[i+4], 0$$

图9

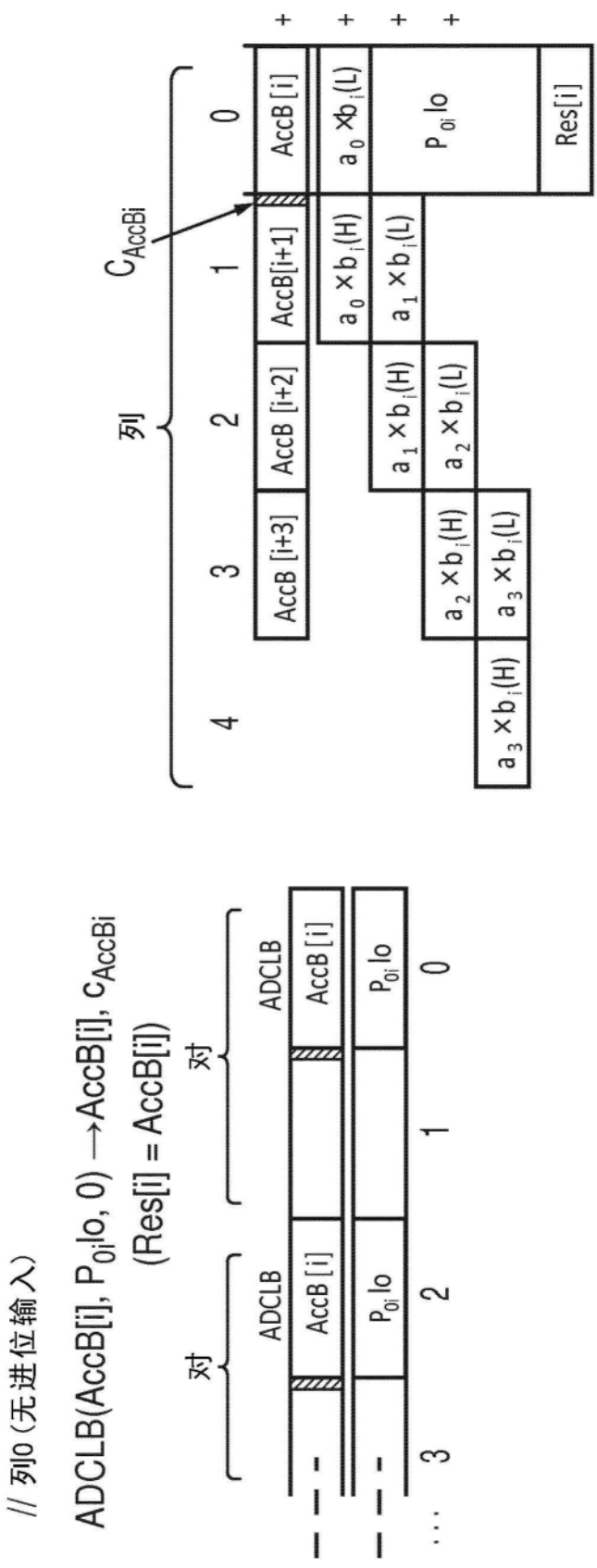

// 列0（无进位输入）
ADCLB(AccB[i], P0i:lo, 0) →AccB[i], cAccBi
(Res[i] = AccB[i])
对
ADCLB
AccB [i]
P0i lo
对
ADCLB
AccB [i]
P0i lo
3
2
1
0
列
0
1
2
3
4
cAccBi
AccB [i]
AccB[i+1]
AccB [i+2]
AccB [i+3]
a0 ×bi(H)
a0 ×bi(L)
a1 ×bi(H)
a1 ×bi(L)
a2 ×bi(H)
a2 ×bi(L)
a3 ×bi(H)
a3 ×bi(L)
P0i lo
Res[i]
+
+
+
+

图10A

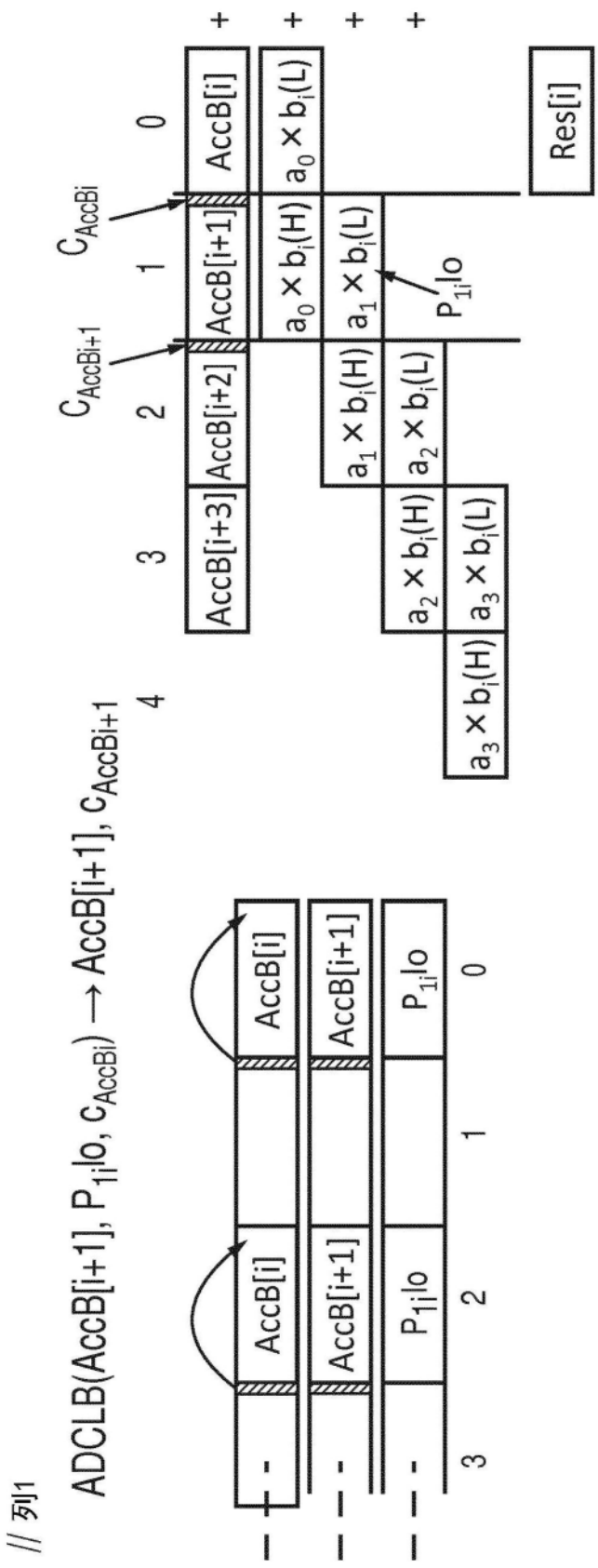

// 列1
ADCLB(AccB[i+1], P1i|o, cAccBi) → AccB[i+1], cAccBi+1
AccB[i]
AccB[i+1]
P1i|o
0
1
2
3
4
cAccBi
cAccBi+1
AccB[i]
AccB[i+1]
AccB[i+2]
AccB[i+3]
a0 × bi(L)
a0 × bi(H)
a1 × bi(L)
a1 × bi(H)
a2 × bi(L)
a2 × bi(H)
a3 × bi(L)
a3 × bi(H)
P1i|o
Res[i]
+
+
+
+

图10B

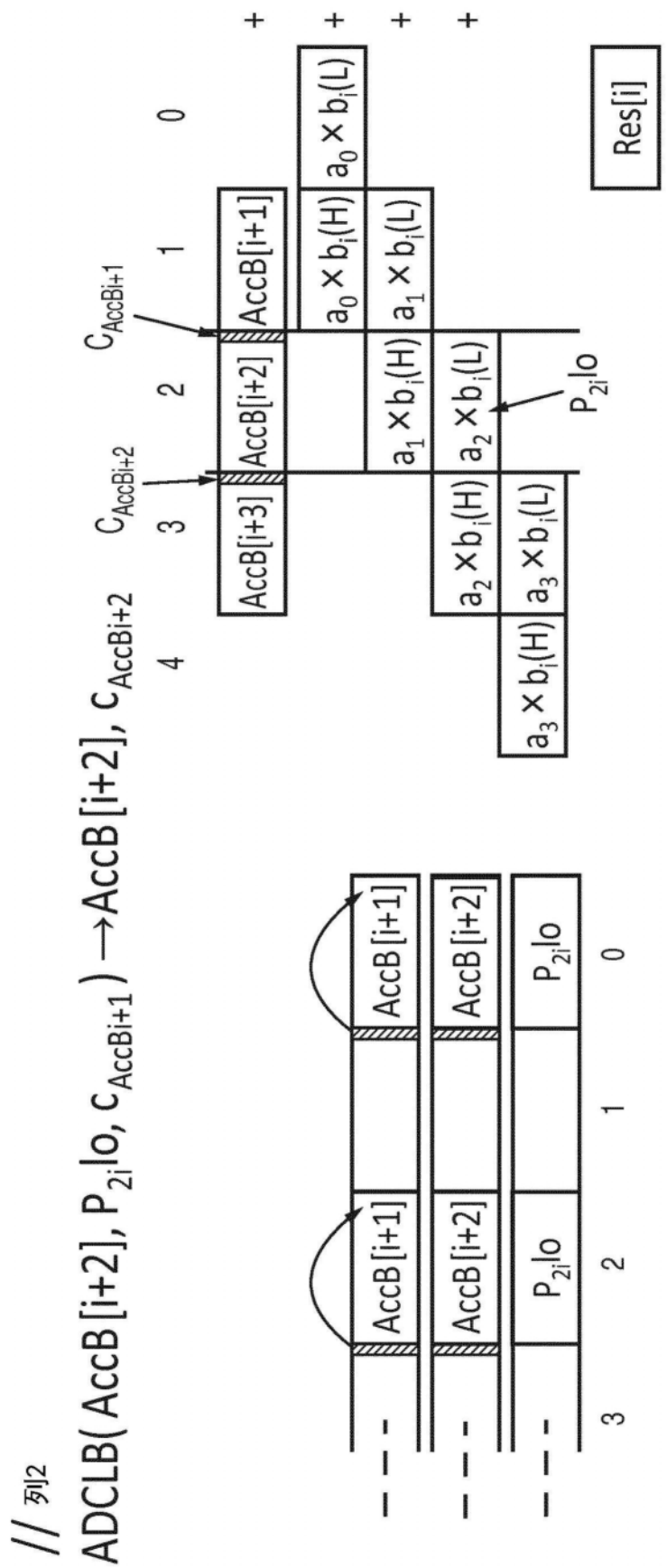

// 列2
ADCLB(AccB[i+2], P2i:0, cAccBi+1) →AccB[i+2], cAccBi+2
cAccBi+1
cAccBi+2
AccB[i+3]
AccB[i+2]
AccB[i+1]
a0 × bi(H)
a0 × bi(L)
a1 × bi(H)
a1 × bi(L)
a2 × bi(H)
a2 × bi(L)
a3 × bi(H)
a3 × bi(L)
P2i:0
Res[i]
AccB[i+1]
AccB[i+2]
P2i:0
0
1
2
3
4

图10C

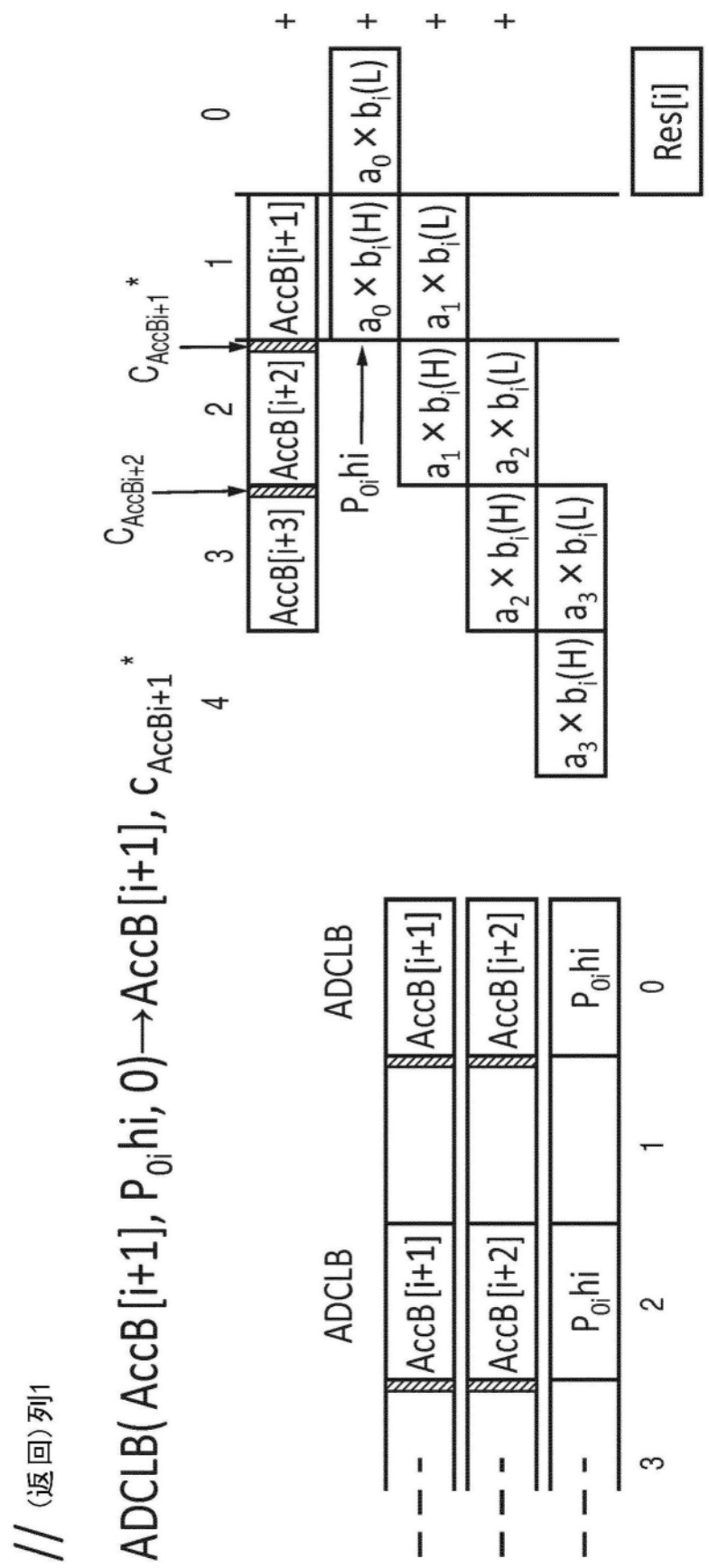

// (返回)列1
ADCLB(AccB[i+1], P0i hi, 0)→AccB[i+1], cAccBi+1*
ADCLB
ADCLB
AccB[i+1]
AccB[i+2]
P0i hi
AccB[i+1]
AccB[i+2]
P0i hi
3 2 1 0
cAccBi+2
cAccBi+1*
AccB[i+3]
AccB[i+2]
AccB[i+1]
4 3 2 1 0
P0i hi
a3 × bi(H)
a3 × bi(L)
a2 × bi(H)
a2 × bi(L)
a1 × bi(H)
a1 × bi(L)
a0 × bi(H)
a0 × bi(L)
+ + + +
Res[i]

图10D

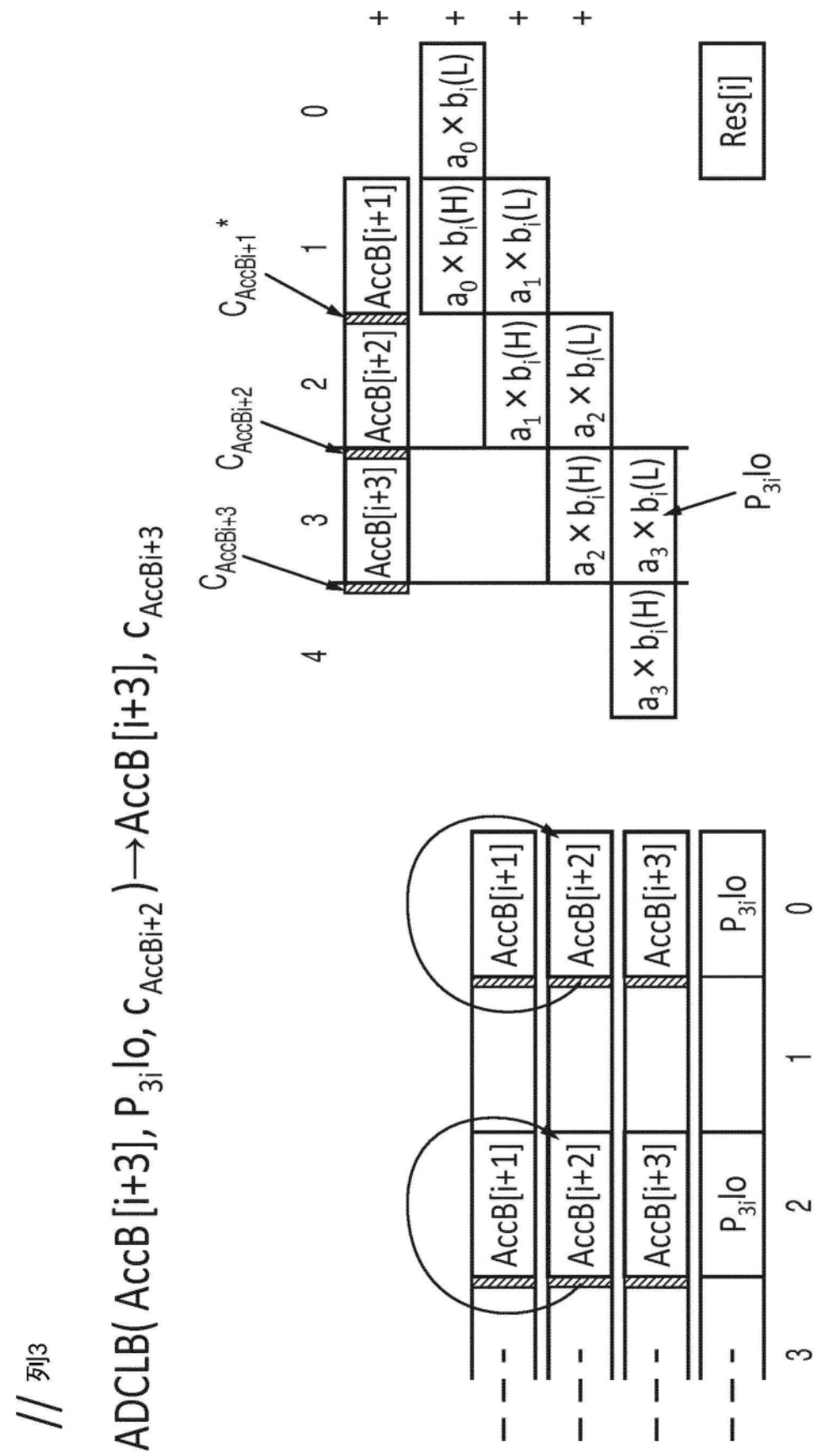

// 列3
ADCLB(AccB [i+3], P3i lo, cAccBi+2)→AccB [i+3], cAccBi+3
cAccBi+3
cAccBi+2
cAccBi+1*
AccB[i+3]
AccB[i+2]
AccB[i+1]
a3 × bi(H)
a3 × bi(L)
a2 × bi(H)
a2 × bi(L)
a1 × bi(H)
a1 × bi(L)
a0 × bi(H)
a0 × bi(L)
P3i lo
Res[i]
0
1
2
3
4
+
+
+
+
AccB[i+1]
AccB[i+2]
AccB[i+3]
P3i lo
AccB[i+1]
AccB[i+2]
AccB[i+3]
P3i lo
0
1
2
3

图10E

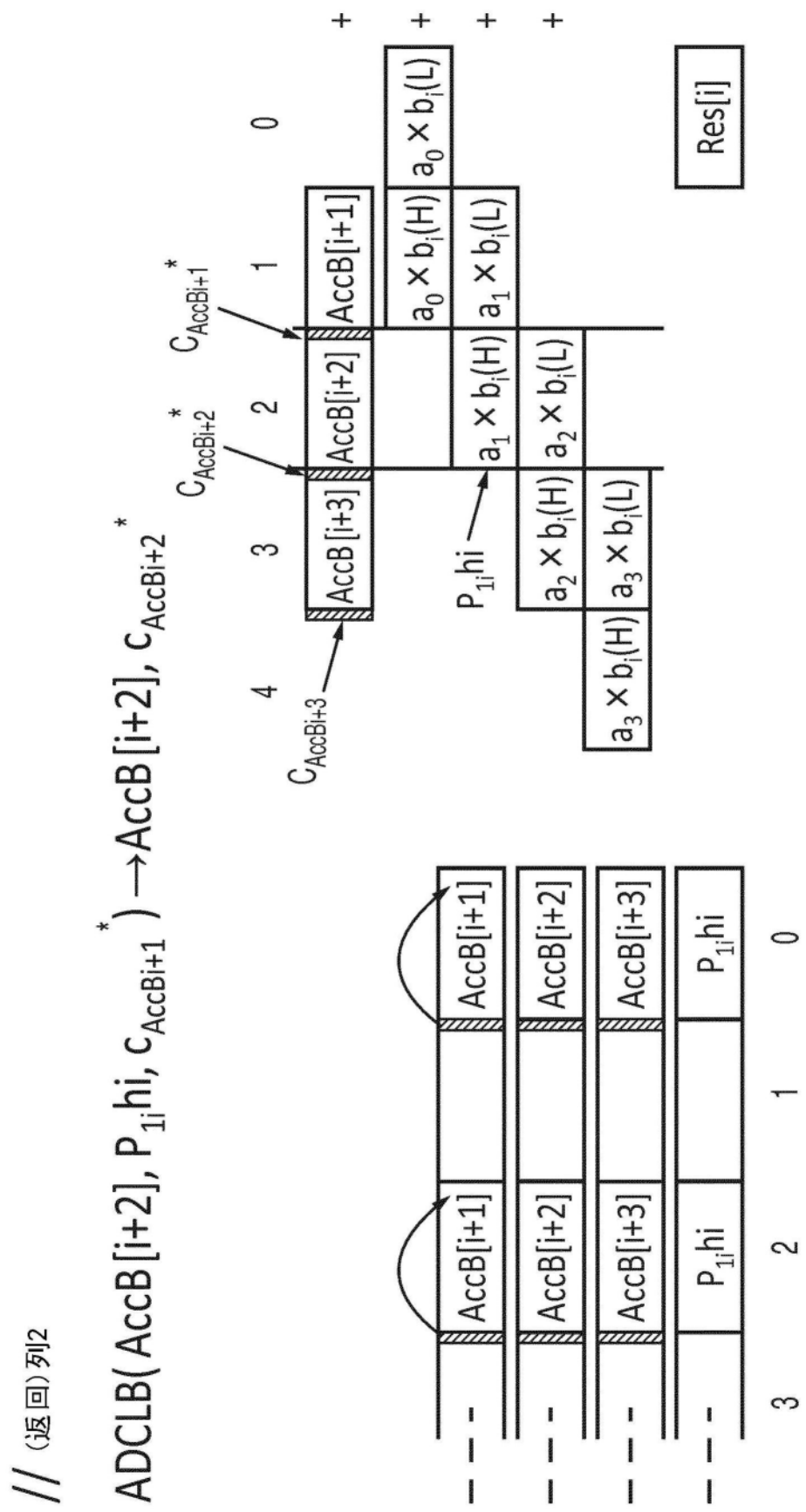

// (返回)列2
ADCLB(AccB[i+2], P1i hi, cAccBi+1*) →AccB[i+2], cAccBi+2*
cAccBi+3*
cAccBi+2*
cAccBi+1*
AccB[i+3]
AccB[i+2]
AccB[i+1]
a0 × bi(H)
a0 × bi(L)
a1 × bi(H)
a1 × bi(L)
a2 × bi(H)
a2 × bi(L)
a3 × bi(H)
a3 × bi(L)
P1i hi
Res[i]
0
1
2
3
4
+
+
+
+
AccB[i+1]
AccB[i+2]
AccB[i+3]
P1i hi
0
1
2
3

图10F

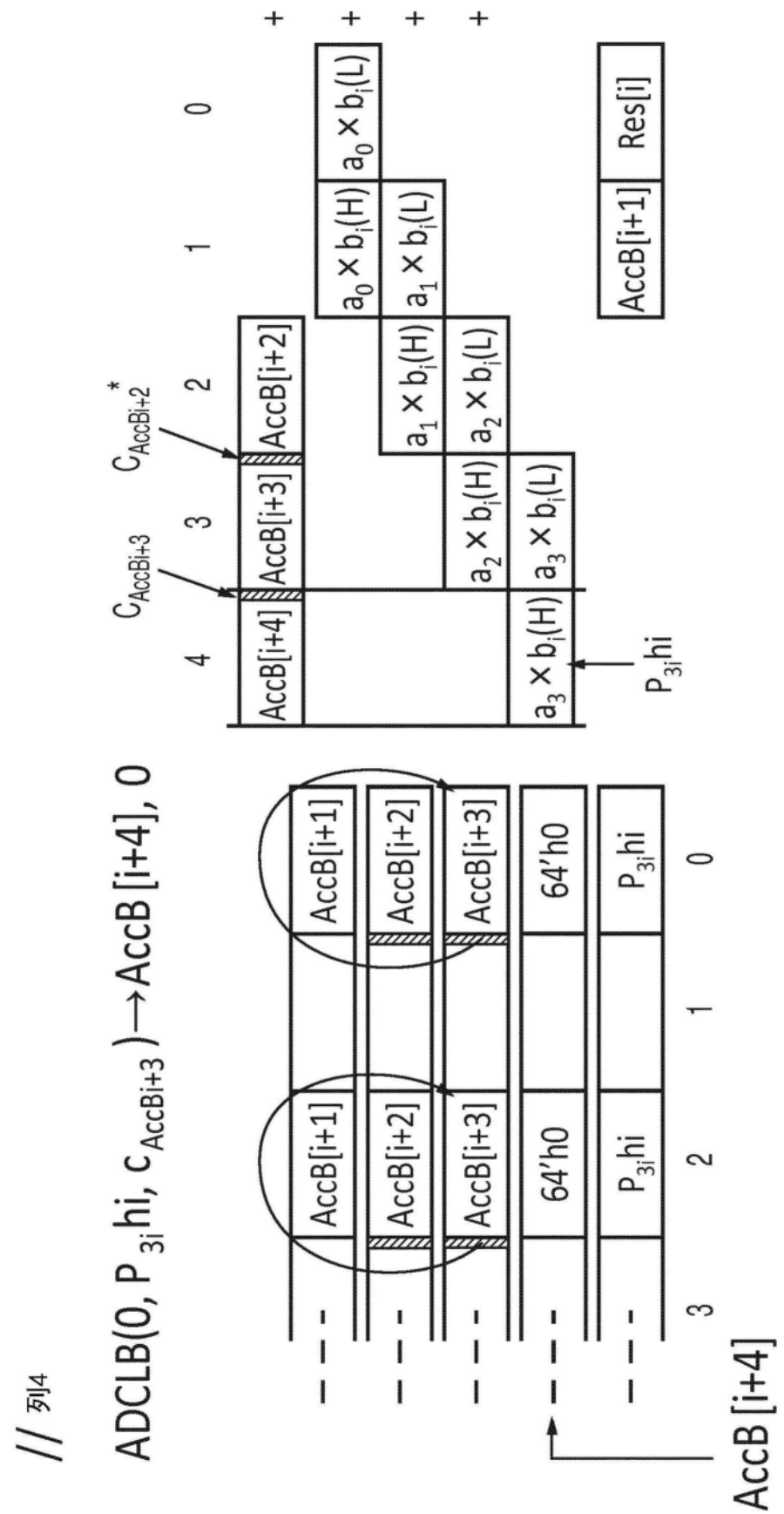

// 列4
ADCLB(0, P 3i hi, c AccBi+3)→AccB [i+4], 0
AccB [i+4]
AccB[i+1]
AccB[i+2]
AccB[i+3]
64'h0
P3i hi
Res[i]
C AccBi+3
C AccBi+2 *
a0 × bi(H)
a0 × bi(L)
a1 × bi(H)
a1 × bi(L)
a2 × bi(H)
a2 × bi(L)
a3 × bi(H)
a3 × bi(L)
P3i hi
0
1
2
3
4
+

图10G

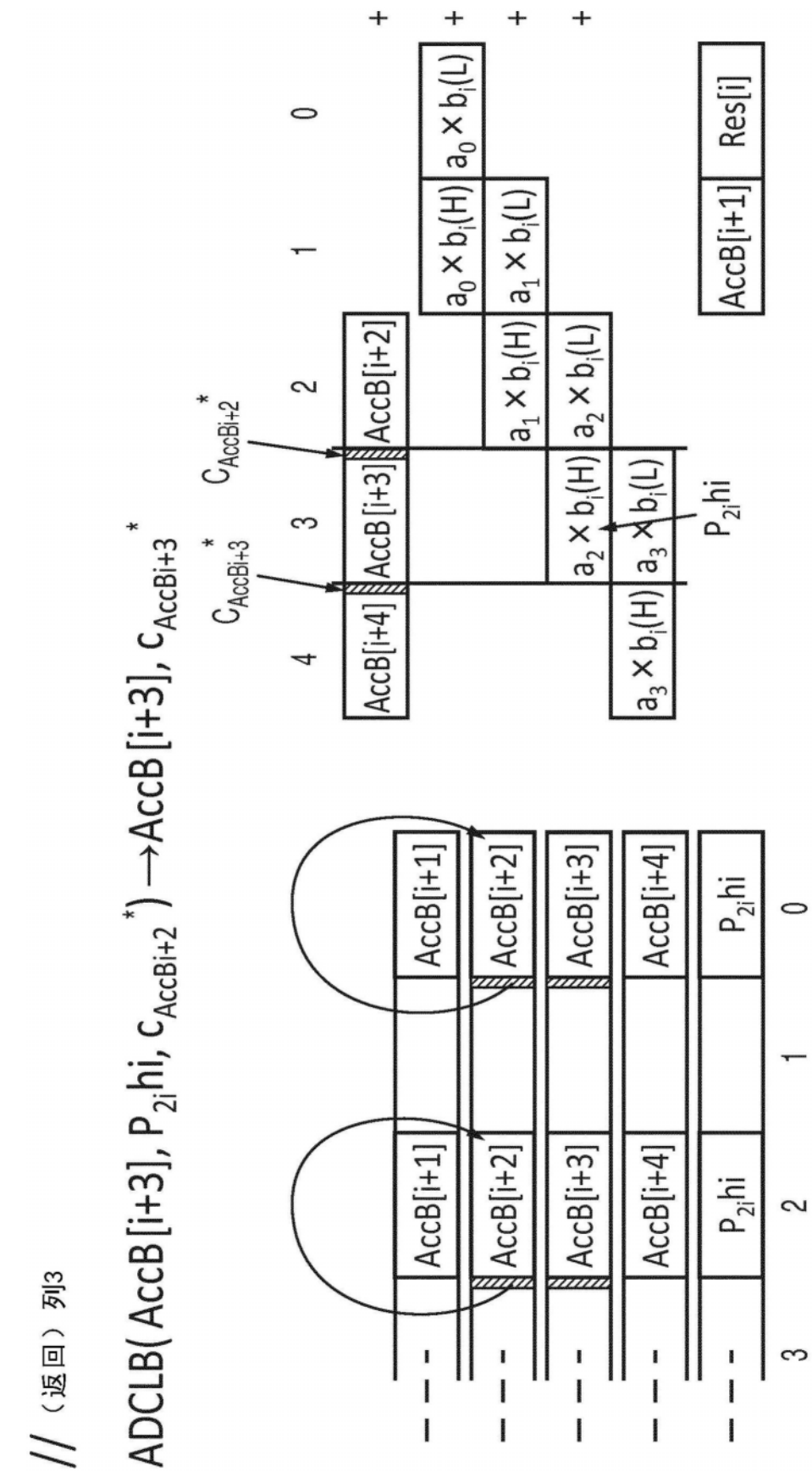

// （返回）列3
ADCLB(AccB[i+3], P2i:hi, cAccBi+2*) →AccB[i+3], cAccBi+3*
cAccBi+2*
cAccBi+3*
AccB[i+4]
AccB[i+3]
AccB[i+2]
AccB[i+1]
a0 × bi(H)
a0 × bi(L)
a1 × bi(H)
a1 × bi(L)
a2 × bi(H)
a2 × bi(L)
a3 × bi(H)
a3 × bi(L)
P2i:hi
Res[i]
4
3
2
1
0
+
AccB[i+1]
AccB[i+2]
AccB[i+3]
AccB[i+4]
P2i:hi
3
2
1
0

图10H

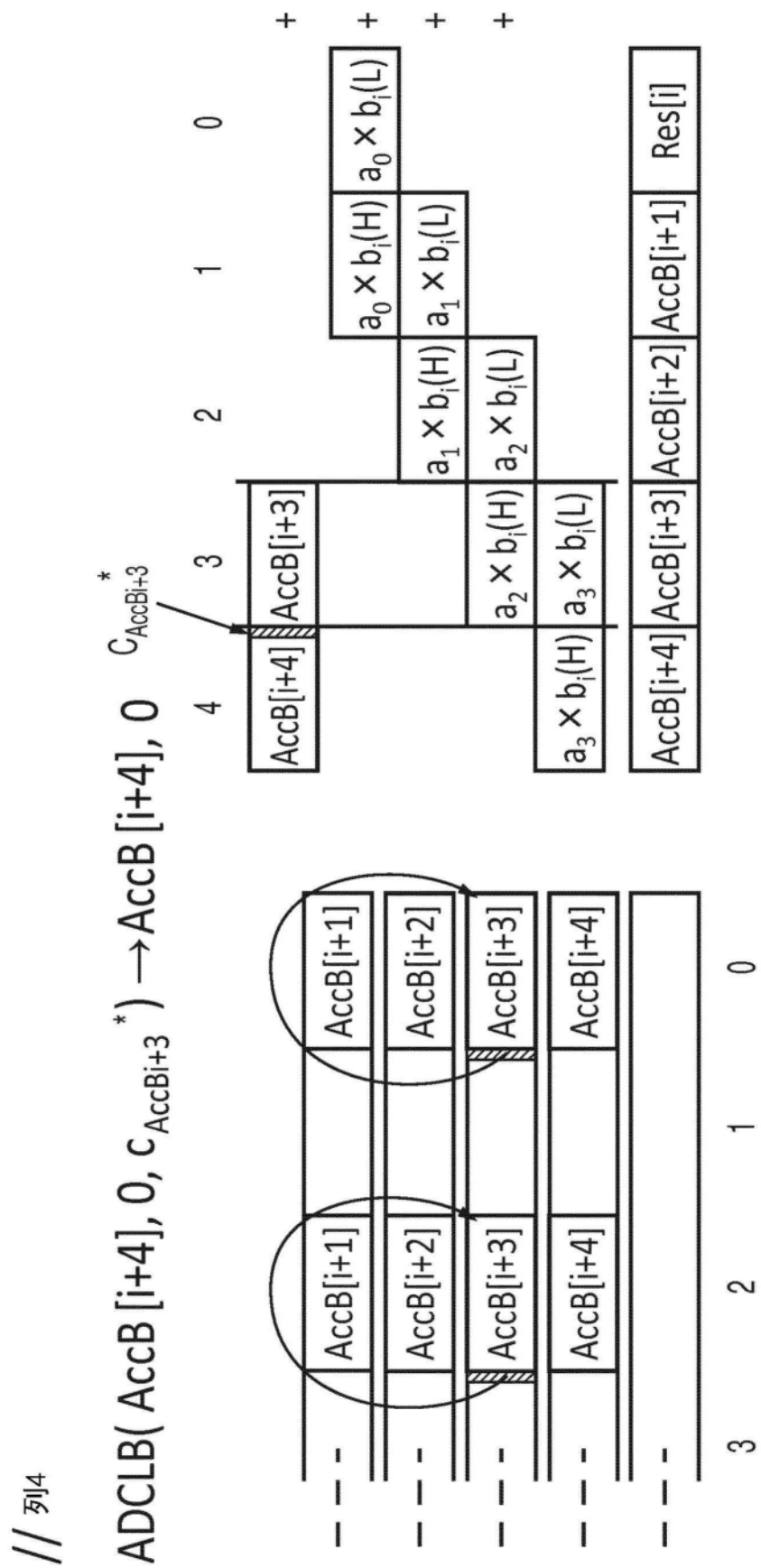

// 列4
ADCLB(AccB [i+4], 0, $c_{AccBi+3}^{*}$) →AccB [i+4], 0
$C_{AccBi+3}^{*}$
AccB[i+4]
AccB[i+3]
$a_3 \times b_i(H)$
$a_3 \times b_i(L)$
$a_2 \times b_i(H)$
$a_2 \times b_i(L)$
$a_1 \times b_i(H)$
$a_1 \times b_i(L)$
$a_0 \times b_i(H)$
$a_0 \times b_i(L)$
Res[i]
AccB[i+1]
AccB[i+2]
AccB[i+3]
AccB[i+4]
4 3 2 1 0
+ + + +

图10I

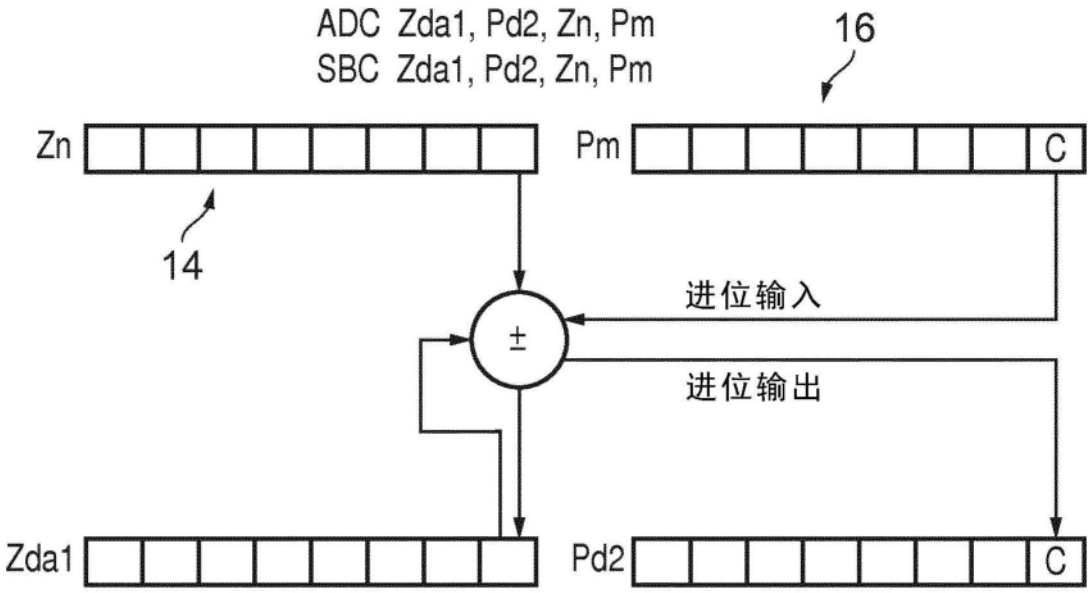

ADC Zda1, Pd2, Zn, Pm
SBC Zda1, Pd2, Zn, Pm
16
Zn
Pm
C
14
进位输入
±
进位输出
Zda1
Pd2
C

图11

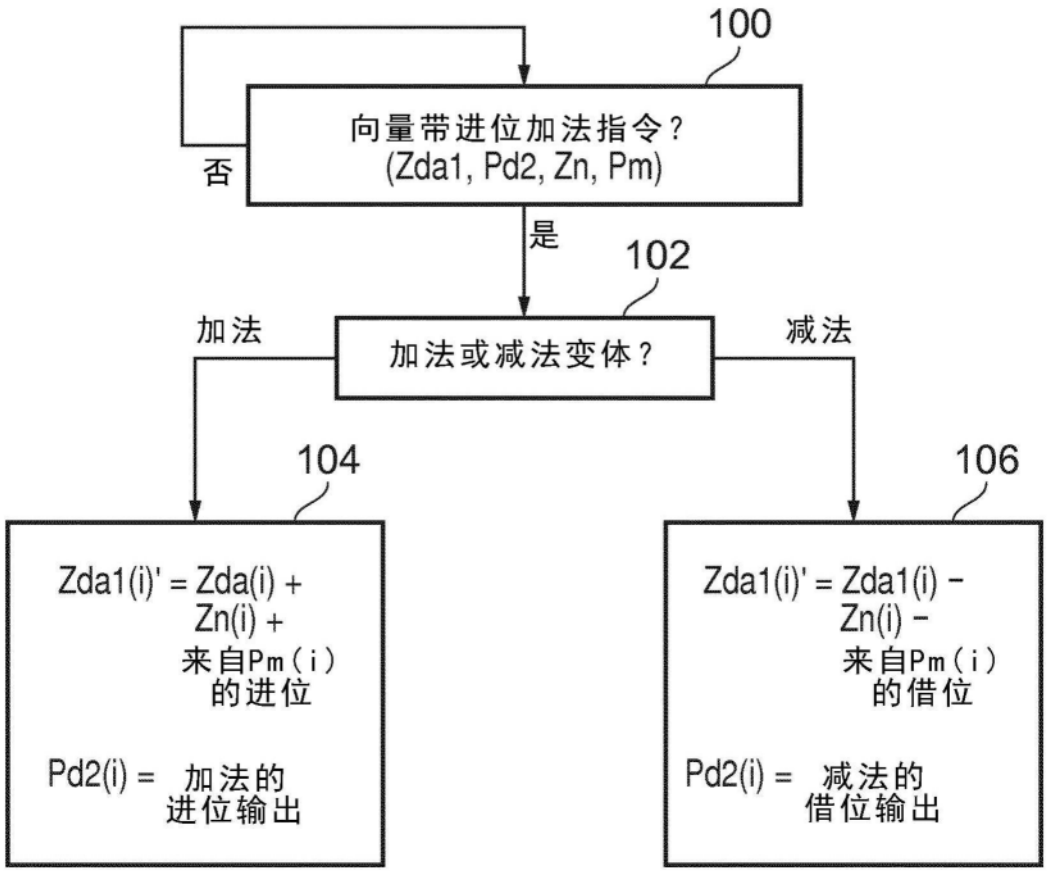

100
向量带进位加法指令？
(Zda1, Pd2, Zn, Pm)
否
是
102
加法
加法或减法变体？
减法
104
106
Zda1(i)' = Zda(i) +
Zn(i) +
来自Pm(i)
的进位
Pd2(i) = 加法的
进位输出
Zda1(i)' = Zda1(i) -
Zn(i) -
来自Pm(i)
的借位
Pd2(i) = 减法的
借位输出

图12

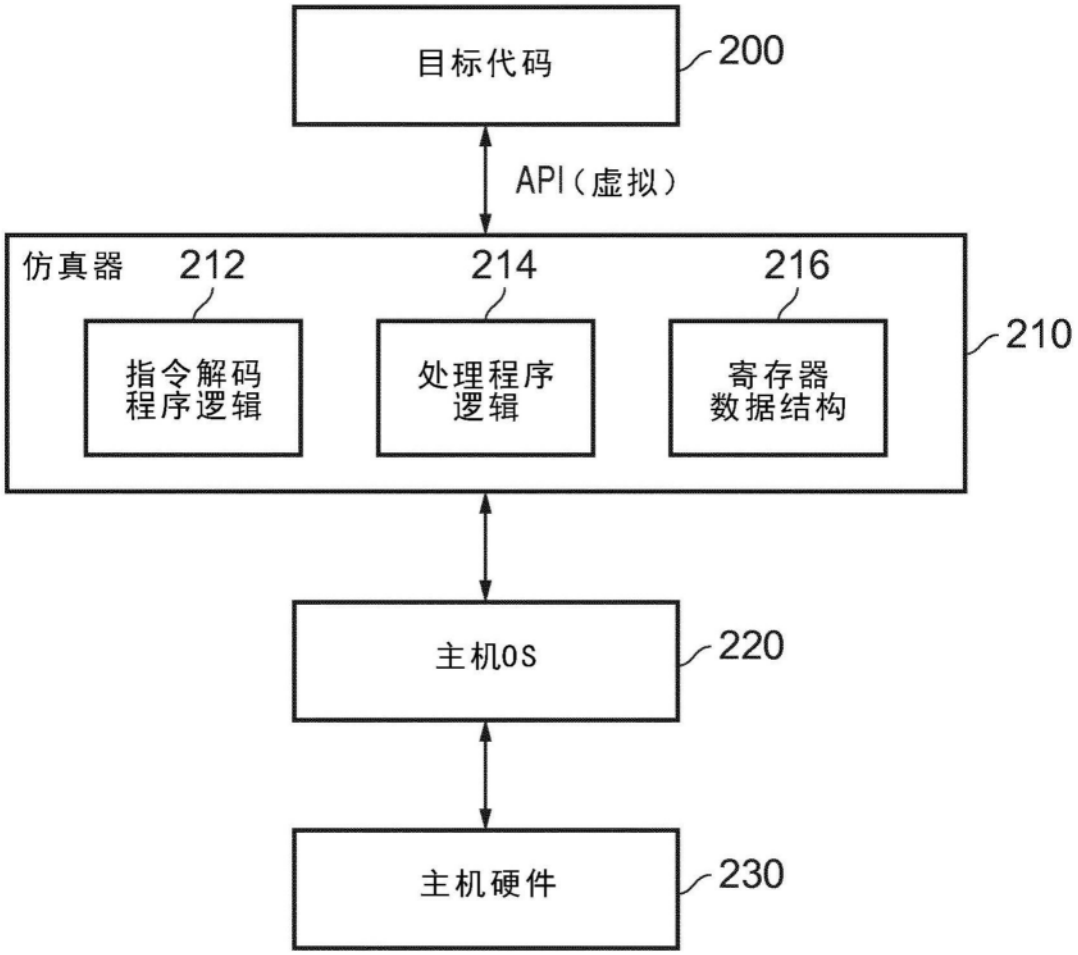

目标代码
200
API（虚拟）
仿真器
212
214
216
指令解码
程序逻辑
处理程序
逻辑
寄存器
数据结构
210
主机OS
220
主机硬件
230

图13