(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111124492 B
(45)授权公告日 2022.09.20

(21)申请号 201911300243.6

(22)申请日 2019.12.16

(65)同一申请的已公布的文献号
申请公布号 CN 111124492 A

(43)申请公布日 2020.05.08

(73)专利权人 成都海光微电子技术有限公司
地址 610000 四川省成都市高新区中国(四川)自由贸易试验区成都天府大道中段1366号2栋天府软件园E5座12层23-32号

(72)发明人 蒋宇翔 王晓阳

(74)专利代理机构 北京超凡宏宇专利代理事务所(特殊普通合伙) 11463
专利代理师 蒋姗

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

(56)对比文件
US 2001018735 A1,2001.08.30
US 2017046154 A1,2017.02.16
蔡卫光等.基于提前写回策略的数据转发优化方法.《浙江大学学报(工学版)》.2010,第44卷(第01期),81-86.

审查员 王超

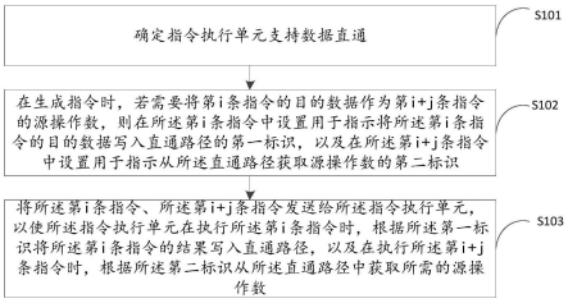
权利要求书3页 说明书15页 附图5页

(54)发明名称

指令生成方法、装置、指令执行方法、处理器及电子设备

(57)摘要

本申请涉及指令生成方法、装置、指令执行方法、处理器及电子设备。该方法包括:确定指令执行单元支持数据直通;在生成指令时,在第i条指令中设置用于指示将第i条指令的目的数据写入直通路径的第一标识,以及在第i+j条指令中设置用于指示从直通路径获取源操作数的第二标识;将第i条指令、第i+j条指令发送给指令执行单元,以使指令执行单元在执行指令时,根据第一标识将第i条指令的结果写入直通路径,以及根据第二标识从直通路径中获取所需的源操作数。通过在指令中设置用于指示将目的数据写入直通路径的第一标识以及用于指示源操作数来源于直通路径的第二标识,依靠软件手段强制使硬件实现显式数据直通,减少了存储器的访问次数。



CN 111124492 B

1.一种指令生成方法,其特征在于,所述方法包括:

确定指令执行单元支持数据直通;

在生成指令时,若需要将第i条指令的目的数据作为第i+j条指令的源操作数,则在所述第i条指令的目的地地址信息中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识,以及在所述第i+j条指令的源操作数地址信息中设置用于指示从所述直通路径获取源操作数的第二标识;其中,i、j均为正整数,且j的最大值不超过硬件所支持的数据直通的最大级数;

将所述第i条指令、所述第i+j条指令发送给所述指令执行单元,以使所述指令执行单元在执行所述第i条指令时,根据所述第一标识将所述第i条指令的结果写入直通路径,以及在执行所述第i+j条指令时,根据所述第二标识从所述直通路径中获取所需的源操作数。

2.根据权利要求1所述的方法,其特征在于,将所述第i条指令、所述第i+j条指令发送给所述指令执行单元,包括:

按照生成顺序将所述第i条指令、所述第i+j条指令进行拼接,得到指令块;

将所述指令块发给解码器,以使所述解码器从所述指令块中依次获取所述第i条指令中的第一关键信息,并将所述第一关键信息发给所述指令执行单元,以及获取所述第i+j条指令中的第二关键信息,并将所述第二关键信息发给所述指令执行单元,所述第一关键信息包括所述第一标识,所述第二关键信息包括所述第二标识。

3.一种指令执行方法,其特征在于,所述方法包括:

获取待执行指令;

获取所述待执行指令中的关键信息,所述关键信息包括:源操作数地址信息和目的地地址信息,所述源操作数地址信息用于指示源操作数的来源,所述目的地地址信息用于指示目的数据的写入路径;

通过判断所述源操作数地址信息是否包含第二标识来判断所述源操作数地址信息指示的源操作数是否来源于直通路径,其中,在所述源操作数地址信息包含所述第二标识时,表征所述源操作数地址信息指示的源操作数来源于直通路径,所述第二标识用于指示从直通路径获取源操作数;

在所述源操作数地址信息指示的源操作数来源于直通路径时,从所述直通路径获取所需的源操作数;

通过判断所述目的地地址信息是否包含第一标识来判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径,其中,在所述目的地地址信息包含所述第一标识时,表征所述目的地地址信息指示的目的数据的写入路径为所述直通路径,所述第一标识用于指示将指令的目的数据写入直通路径;

在所述目的地地址信息指示的目的数据的写入路径为所述直通路径时,将执行所述待执行指令的结果写入所述直通路径。

4.一种处理器,其特征在于,包括:

指令执行单元;

处理器核心,用于确定所述指令执行单元支持数据直通;以及在生成指令时,若需要将第i条指令的目的数据作为第i+j条指令的源操作数,则在所述第i条指令的目的地地址信息中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识,以及在所述第i

+j条指令的源操作数地址信息中设置用于指示从所述直通路径获取源操作数的第二标识；i为正整数；以及还用于将所述第i条指令、所述第i+j条指令发送给所述指令执行单元；其中，i、j均为正整数，且j的最大值不超过硬件所支持的数据直通的最大级数；

所述指令执行单元，用于在执行所述第i条指令时，根据所述第一标识将所述第i条指令的结果写入直通路径，以及，在执行所述第i+j条指令时，根据所述第二标识从所述直通路径中获取所需的源操作数。

5.根据权利要求4所述的处理器，其特征在于，所述处理器还包括解码器，所述处理器核心，用于按照生成顺序将所述第i条指令、所述第i+j条指令进行拼接，得到指令块，并将所述指令块发给所述解码器；

所述解码器，用于从所述指令块中依次获取所述第i条指令中的第一关键信息，并将所述第一关键信息发给所述指令执行单元，以及获取所述第i+j条指令中的第二关键信息，并将所述第二关键信息发给所述指令执行单元，所述第一关键信息包括所述第一标识，所述第二关键信息包括所述第二标识。

6.一种处理器，其特征在于，包括：

解码器，用于获取待执行指令，并获取该待执行指令中的关键信息，所述关键信息包括：源操作数地址信息和目的地地址信息，所述源操作数地址信息用于指示源操作数的来源，所述目的地地址信息用于指示目的数据的写入路径；以及还用于将所述关键信息发给指令执行单元；

所述指令执行单元，用于通过判断所述源操作数地址信息是否包含第二标识来判断所述源操作数地址信息指示的源操作数是否来源于直通路径，其中，在所述源操作数地址信息包含所述第二标识时，表征所述源操作数地址信息指示的源操作数来源于直通路径，所述第二标识用于指示从直通路径获取源操作数；在所述源操作数地址信息指示的源操作数来源于直通路径时，从所述直通路径获取所需的源操作数；以及还用于通过判断所述目的地地址信息是否包含第一标识来判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径，其中，在所述目的地地址信息包含所述第一标识时，表征所述目的地地址信息指示的目的数据的写入路径为所述直通路径，所述第一标识用于指示将指令的目的数据写入直通路径；在所述目的地地址信息指示的目的数据的写入路径为所述直通路径时，将执行所述待执行指令的结果写入所述直通路径。

7.一种电子设备，其特征在于，包括：如权利要求4或5所述的处理器，或者如权利要求6所述的处理器。

8.一种指令生成装置，其特征在于，所述装置包括：

确定模块，用于确定指令执行单元支持数据直通；

生成模块，用于在生成指令时，若需要将第i条指令的目的数据作为第i+j条指令的源操作数，则在所述第i条指令的目的地地址信息中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识，以及在所述第i+j条指令的源操作数地址信息中设置用于指示从所述直通路径获取源操作数的第二标识；i为正整数，j为正整数，且j的最大值不超过硬件所支持的数据直通的最大级数；

发送模块，用于将所述第i条指令、所述第i+j条指令发送给所述指令执行单元，以使所述指令执行单元在执行所述第i条指令时，根据所述第一标识将所述第i条指令的结果写入

直通路径,以及,在执行所述第i+j条指令时,根据所述第二标识从所述直通路径中获取所需的源操作数。

指令生成方法、装置、指令执行方法、处理器及电子设备

技术领域

[0001] 本申请属于计算机技术领域，具体涉及一种指令生成方法、装置、指令执行方法、处理器及电子设备。

背景技术

[0002] 功耗是人们在计算应用中关注的重点，在典型的高强度计算应用中，70～80％的功耗是计算单元使用的，而在计算单元中，50％的功耗是读、写数据所使用的。在典型的科学计算和机器学习应用中，矩阵乘法是最流行的用例之一，在矩阵乘法计算应用中，35-40％的功率是访问向量通用寄存器(Vector General Purpose Register，VGPR)所使用的。

发明内容

[0003] 鉴于此，本申请的目的在于提供一种指令生成方法、装置、指令执行方法、处理器及电子设备，以改善现有在典型的高强度计算应用中需要大量访问存储器，从而导致需要消耗大量功耗的问题。

[0004] 本申请的实施例是这样实现的：

[0005] 第一方面，本申请实施例提供了一种指令生成方法，所述方法包括：确定指令执行单元支持数据直通；在生成指令时，若需要将第i条指令的目的数据作为第i+j条指令的源操作数，在所述第i条指令中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识，以及在所述第i+j条指令中设置用于指示从所述直通路径获取源操作数的第二标识；其中，i、j均为正整数；将所述第i条指令、所述第i+j条指令发送给所述指令执行单元，以使所述指令执行单元在执行所述第i条指令时，根据所述第一标识将所述第i条指令的结果写入直通路径，以及在执行所述第i+j条指令时，根据所述第二标识从所述直通路径中获取所需的源操作数。本申请实施例中，通过在指令中设置用于指示将目的数据写入直通路径的第一标识，以及设置用于指示源操作数来源于直通路径的第二标识，依靠软件手段强制使硬件实现显式数据直通，可以大大减少存储器的访问次数，从而节约大量功耗。

[0006] 结合第一方面实施例的一种可能的实施方式，将所述第i条指令、所述第i+j条指令发送给所述指令执行单元，包括：按照生成顺序将所述第i条指令、所述第i+j条指令进行拼接，得到指令块；将所述指令块发给解码器，以使所述解码器从所述指令块中依次获取所述第i条指令中的第一关键信息，并将所述第一关键信息发给所述指令执行单元，以及获取所述第i+j条指令中的第二关键信息，并将所述第二关键信息发给所述指令执行单元，所述第一关键信息包括所述第一标识，所述第二关键信息包括所述第二标识。本申请实施例中，通过将多条指令按照生成顺序进行拼接，得到指令块，使得当有多个指令块时，本申请实施例生成的指令能够被正确执行，因为硬件在执行当前指令块时须完成整个指令块的执行方能切换到其他指令块，避免在执行指令的过程中途切换到实现其他功能的指令，导致数据直通出现逻辑错误。

[0007] 第二方面，本申请实施例还提供了一种指令执行方法，所述方法包括：获取待执行

指令;获取所述待执行指令中的关键信息,所述关键信息包括:源操作数地址信息和目的地地址信息,所述源操作数地址信息用于指示源操作数的来源,所述目的地地址信息用于指示目的数据的写入路径;判断所述源操作数地址信息指示的源操作数是否来源于直通路径;在所述源操作数地址信息指示的源操作数来源于直通路径时,从所述直通路径获取所需的源操作数;判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径;在所述目的地地址信息指示的目的数据的写入路径为所述直通路径时,将执行所述待执行指令的结果写入所述直通路径。本申请实施例中,通过获取待执行指令中的源操作数地址信息和目的地地址信息等关键信息,并在源操作数地址信息指示的源操作数来源于直通路径时,直接从直通路径中获取所需的源操作数,以及在目的地地址信息指示的目的数据的写入路径为直通路径时,直接将指令执行结果写入直通路径,以减少存储器的访问,从而减少功率消耗。

[0008] 结合第二方面实施例的一种可能的实施方式,判断所述源操作数地址信息指示的源操作数是否来源于直通路径,包括:通过判断所述源操作数地址信息是否包含第二标识来判断所述源操作数地址信息指示的源操作数是否来源于直通路径;在所述源操作数地址信息包含所述第二标识时,表征所述源操作数地址信息指示的源操作数来源于直通路径。本申请实施例中,通过在指令中设置用于指示从直通路径获取源操作数的第二标识,使得可以通过判断源操作数地址信息是否包含第一标识,来快速判断该源操作数地址信息指示的源操作数是否来源于直通路径。

[0009] 结合第二方面实施例的一种可能的实施方式,判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径,包括:通过判断所述目的地地址信息是否包含第一标识来判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径;在所述目的地地址信息包含所述第一标识时,表征所述目的地地址信息指示的目的数据的写入路径为所述直通路径。本申请实施例中,通过在指令中设置用于指示将待执行指令的目的数据写入直通路径的第一标识,使得可以通过判断目的地地址信息是否包含第一标识,来快速判断目的地地址信息指示的目的数据的写入路径是否为直通路径。

[0010] 第三方面,本申请实施例还提供了一种处理器,包括:指令执行单元和处理器核心;处理器核心,用于确定所述指令执行单元支持数据直通;以及在生成指令时,若需要将第i条指令的目的数据作为第i+j条指令的源操作数,则在所述第i条指令中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识,以及在所述第i+j条指令中设置用于指示从所述直通路径获取源操作数的第二标识;i为正整数;以及还用于将所述第i条指令、所述第i+j条指令发送给所述指令执行单元;其中,i、j均为正整数;所述指令执行单元,用于在执行所述第i条指令时,根据所述第一标识将所述第i条指令的结果写入直通路径,以及,在执行所述第i+j条指令时,根据所述第二标识从所述直通路径中获取所需的源操作数。

[0011] 结合第三方面实施例的一种可能的实施方式,所述处理器还包括解码器,所述处理器核心,用于按照生成顺序将所述第i条指令、所述第i+j条指令进行拼接,得到指令块,并将所述指令块发给所述解码器;所述解码器,用于从所述指令块中依次获取所述第i条指令中的第一关键信息,并将所述第一关键信息发给所述指令执行单元,以及获取所述第i+j条指令中的第二关键信息,并将所述第二关键信息发给所述指令执行单元,所述第一关键

信息包括所述第一标识，所述第二关键信息包括所述第二标识。

[0012] 第四方面，本申请实施例还提供了一种处理器，包括：解码器和指令执行单元；解码器，用于获取待执行指令，并获取该待执行指令中的关键信息，所述关键信息包括：源操作数地址信息和目的地地址信息，所述源操作数地址信息用于指示源操作数的来源，所述目的地地址信息用于指示目的数据的写入路径；以及还用于将所述关键信息发给指令执行单元；所述指令执行单元，用于判断所述源操作数地址信息指示的源操作数是否来源于直通路径；在所述源操作数地址信息指示的源操作数来源于直通路径时，从所述直通路径获取所需的源操作数；以及还用于判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径；在所述目的地地址信息指示的目的数据的写入路径为所述直通路径时，将执行所述待执行指令的结果写入所述直通路径。

[0013] 结合第四方面实施例的一种可能的实施方式，所述指令执行单元，用于通过判断所述源操作数地址信息是否包含第二标识来判断所述源操作数地址信息指示的源操作数是否来源于直通路径；在所述源操作数地址信息包含所述第二标识时，表征所述源操作数地址信息指示的源操作数来源于直通路径。

[0014] 结合第四方面实施例的一种可能的实施方式，所述指令执行单元，用于通过判断所述目的地地址信息是否包含第一标识来判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径；在所述目的地地址信息包含所述第一标识时，表征所述目的地地址信息指示的目的数据的写入路径为所述直通路径。

[0015] 第五方面，本申请实施例还提供了一种电子设备，包括：如上述第三方面实施例和/或结合第三方面实施例的一种可能的实施方式提供的处理器，或者如上述第四方面实施例和/或结合第四方面实施例的任一种可能的实施方式提供的处理器。

[0016] 第六方面，本申请实施例还提供了一种指令生成装置，所述装置包括：确定模块、生成模块以及发送模块；确定模块，用于确定指令执行单元支持数据直通；生成模块，用于在生成指令时，若需要将第i条指令的目的数据作为第i+j条指令的源操作数，则在所述第i条指令中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识，以及在所述第i+j条指令中设置用于指示从所述直通路径获取源操作数的第二标识；i为正整数；发送模块，用于将所述第i条指令、所述第i+j条指令发送给所述指令执行单元，以使所述指令执行单元在执行所述第i条指令时，根据所述第一标识将所述第i条指令的结果写入直通路径，以及，在执行所述第i+j条指令时，根据所述第二标识从所述直通路径中获取所需的源操作数。

[0017] 本申请的其他特征和优点将在随后的说明书阐述，并且，部分地从说明书中变得显而易见，或者通过实施本申请实施例而了解。本申请的目的和其他优点可通过在所写的说明书以及附图中所特别指出的结构来实现和获得。

附图说明

[0018] 为了更清楚地说明本申请实施例或现有技术中的技术方案，下面将对实施例中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。通过附图所示，本申请的上述及其它目的、特征和优势将更加清晰。在全部

附图中相同的附图标记指示相同的部分。并未刻意按实际尺寸等比例缩放绘制附图，重点在于示出本申请的主旨。

[0019] 图1示出了本申请实施例提供的一种指令生成方法的流程示意图。

[0020] 图2示出了本申请实施例提供的一种VOP3指令中各字段的示意图。

[0021] 图3示出了本申请实施例提供的执行完整指令块的逻辑示意图。

[0022] 图4示出了本申请实施例提供的一种指令执行方法的流程示意图。

[0023] 图5示出了本申请实施例提供的一种矩阵乘法应用的硬件示意图。

[0024] 图6示出了本申请实施例提供的一种指令生成装置的功能模块框图。

[0025] 图7示出了本申请实施例提供的一种处理器的结构框图。

具体实施方式

[0026] 下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行描述。

[0027] 应注意到：相似的标号和字母在下面的附图中表示类似项，因此，一旦某一项在一个附图中被定义，则在随后的附图中不需要对其进行进一步定义和解释。同时，在本申请的描述中诸如“第一”、“第二”等之类的关系术语仅仅用来将一个实体或者操作与另一个实体或操作区分开来，而不一定要求或者暗示这些实体或操作之间存在任何这种实际的关系或者顺序。而且，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、物品或者设备不仅包括那些要素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、物品或者设备所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括所述要素的过程、方法、物品或者设备中还存在另外的相同要素。

[0028] 再者，本申请中术语“和/或”，仅仅是一种描述关联对象的关联关系，表示可以存在三种关系，例如，A和/或B，可以表示：单独存在A，同时存在A和B，单独存在B这三种情况。

[0029] 鉴于当前在典型的高强度计算应用中需要大量访问存储器，从而导致需要消耗大量功耗的问题，例如，在矩阵乘法应用中，需要反复不断的从向量通用寄存器VGPR中获取源操作数A、源操作数B和源操作数C，进行A*B+C计算，然后将相乘的结果写入VGPR中。例如，在执行instruction0(指令0)时，从VGPR中获取源操作数A、源操作数B和源操作数C(此时为0)，在计算完成时，需要将计算结果写入VGPR中；在执行instruction1(指令1)时，又从VGPR中获取源操作数A、源操作数B和源操作数C(此时为前一次计算的结果，即C1＝A0+B0+C0)，在计算完成时，需要将计算结果写入VGPR中，如此反复，直至完成矩阵计算。因此，本申请实施例中提供了一种新指令，使得可以将VGPR的访问转换为数据直通，也即可以将第i条指令的目的数据(计算结果)作为第i+j条指令的源数据，从而硬件直接跳过第i条指令的目的数据(计算结果)的VGPR写和第i+j条指令的源操作数的VGPR读。其中，i、j均为正整数，且j的最大值不超过硬件所支持的数据直通的最大级数，例如假设硬件所支持的数据直通的最大级数为2，则j的最大值为2。

[0030] 请参阅图1，为本申请实施例提供的一种指令生成方法，下面将结合图1对其所包含的步骤进行说明。

[0031] 步骤S101：确定指令执行单元支持数据直通。

[0032] 本申请实施例中，在生成指令之前，由编译器(程序软件)来检测指令执行单元

(Instruction Execution)是否支持数据直通,在确定指令执行单元(硬件)支持数据直通时,执行步骤S102。目前,大多数硬件支持1/2级隐式数据直通:当硬件执行instruction1或instruction2时,若检测到可以从直通路径(forwarding path)获得源数据,则可以跳过VGPR的读取,直接从直通路径获取源数据,但是该方式依然存在大量instruction0(前一条指令)的存储器写操作,并且还存在forwarding(直通)不成功的情况,也即,即便硬件支持隐式数据直通,但是并不一定能从直通路径获得源数据。

[0033] 步骤S102:在生成指令时,若需要将第i条指令的目的数据作为第i+j条指令的源操作数,在所述第i条指令中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识,以及在所述第i+j条指令中设置用于指示从所述直通路径获取源操作数的第二标识。

[0034] 在确定硬件支持数据直通,在生成指令时,若需要将第i条指令的目的数据作为第i+j条指令的源操作数,则在第i条指令中设置用于指示将第i条指令的目的数据写入直通路径的第一标识。

[0035] 其中,i、j均为正整数,且j的最大值不超过硬件所支持的数据直通的最大级数,例如假设硬件所支持的数据直通的最大级数为2,则j的最大值为2。若需要将前一条指令的目的数据作为后一条指令的源操作数时,则在生成第i条指令时,在第i条指令中设置第一标识,在生成第i+1条指令时,在第i+1条指令中设置第二标识。若需要将上一条指令的目的数据作为下下条指令的源操作数时,则在生成第i条指令时,在第i条指令中设置第一标识,在生成第i+2条指令时,在第i+2条指令中设置第二标识。

[0036] 为了便于理解,本申请以生成具有3个操作数的向量运算指令VOP3(Vector Operation with 3 Operand)指令为例进行说明,如图2所示。设定类型为“110010”,即110010表示指令是VOP3指令。其中,VOP3指令中的各个字段的含义,见表1。

[0037] 表1

[0038]

字段	位数	内容描述
Result_ID	[7：0]	用于指示结果的写入位置
/	[13：8]	
BS（Block Start）	[14：14]	指令块开始
BE（Block End）	[15：15]	指令块结束
Operantion_code	[25：16]	即操作码，硬件支持各种常见操作包括加，减，乘，除，乘加等，通过定义指令使用哪种操作。
Instruction_group	[31：26]	定义指令类型（110010）
Operand0_ID0	[40：32]	如果 VF(Vector Forwarding) = 125，操作数 Operand0 来源于 forword path（直通路径）
Operand1_ID1	[49：41]	如果 VF(Vector Forwarding) = 125，操作数 Operand1 来源于 forword path（直通路径）
Operand2_ID2	[58：50]	如果 VF(Vector Forwarding) = 125，操作数 Operand2 来源于 forword path（直通路径）
/	[62：59]	
DF（Destination Forwarding）	[63：63]	目的地直通 DF==1，则忽略 Result_ID，仅将结果写入直通路径

[0039] 需要说明的是，表1中的各个字段的位数(位宽)是相对固定的，其位置是可以变化的，例如，Operand0_ID0可以不再是[40:32]这一位数，其可以是在[8:0]这一位数，其余字段的情况与之类似。

[0040] 其中，Operand0_ID0字段、Operand1_ID1字段以及Operand2_ID2字段用于指示操作数的来源，即从何处去获取源操作数，也即对于操作数Operand0，如果该字段中的VF＝125(第二标识)时，则表明操作数Operand0来源于直通路径，否则从Operand0_ID0指向的位置获取操作数Operand0；对于操作数Operand1，如果该字段中的VF＝125(第二标识)时，则表明操作数Operand1来源于直通路径，否则从Operand0_ID1指向的位置获取操作数Operand1；对于操作数Operand2，如果该字段中的VF＝125(第二标识)时，则表明操作数Operand2来源于直通路径，否则从Operand0_ID2指向的位置获取操作数Operand2。Result_ID字段以及DF字段用于指示目的数据的写入路径，如果DF＝1(第一标识)，则将目的数据直接写到直通路径，否则将其写到Result_ID指向的位置。其中，需要说明的是，用于指示操作数来源于直通路径的VF值并不限于125，同理，用于指示将目的数据写入直通路径的值并不限于1。

[0041] 结合表1进行说明，若需要将第i条指令的目的数据作为第i+j条指令的源操作数，则将第i条指令中的目的地直通DF字段中的数值设置为1，以及将第i+j条指令中用于指示源操作数来源的地址字段中的VF的数值设置为125，也即，若将Operand0_ID0字段中的向量直通VF值设置为125，则表示操作数Operand0来源于直通路径；若将Operand0_ID1字段中的向量直通VF值设置为125，则表示操作数Operand1来源于直通路径；若将Operand0_ID2字段

中的向量直通VF值设置为125，则表示操作数Operand2来源于直通路径。本申请实施例中，通过在指令中设置用于指示将目的数据写入直通路径的第一标识(DF＝1)，以及设置用于指示源操作数来源于直通路径的第二标识(VF＝125)，依靠软件手段强制使硬件实现显式数据直通，以保证一定可以执行forwarding。

[0042] 步骤S103：将所述第i条指令、所述第i+j条指令发送给所述指令执行单元，以使所述指令执行单元在执行所述第i条指令时，根据所述第一标识将所述第i条指令的结果写入直通路径，以及在执行所述第i+j条指令时，根据所述第二标识从所述直通路径中获取所需的源操作数。

[0043] 在生成第i条指令和第i+j条指令后，将第i条指令、第i+j条指令发送给指令执行单元去执行。其中，指令执行单元在执行第i条指令时，根据第一标识将第i条指令的结果写入直通路径，以及在执行第i+j条指令时，根据第二标识从直通路径中获取所需的源操作数。例如，若第i条指令中的DF＝1，则直接将第i条指令的结果写入直通路径，若第i+j条指令中的Operand0_ID0字段中的VF＝125，则直接从直通路径中获取源操作数Operand0。

[0044] 考虑到在执行实现同一功能的指令时，不允许中途插入实现其他功能的指令，因此，为了保证当有多个实现不同功能的指令时，本申请提供的这种直通方案能够正确实施，本申请实施例中，还可以在将第i条指令、第i+j条指令发送给指令执行单元的时候，先按照生成顺序将第i条指令、第i+j条指令进行拼接，得到指令块(也即指令组或指令集)，然后将指令块发给解码器(硬件)，以使解码器从指令块中依次获取第i条指令中的第一关键信息，并将第一关键信息发给指令执行单元，以使指令执行单元根据第一关键信息中的第一标识将第i条指令的执行结果写入直通路径；以及获取第i+j条指令中的第二关键信息，并将第二关键信息发给指令执行单元，以使指令执行单元根据第二关键信息中的第二标识从直通路径中获取所需的源操作数。其中，第一关键信息包括第一标识，第二关键信息包括第二标识。这样在执行指令时，只有执行完当前指令块中的所有指令后，才会切换到其他指令块。其中，该指令组包括组头(group header)和组主体(group body)。组头定义了该指令组使用多少资源，指令组包括多少条指令；组主体包含指令组的所有指令。

[0045] 为了避免在执行当前指令块的过程中，又切换到其他指令块，可以通过增加一个阻止锁(lock)，用于在硬件运行一个指令块时锁定仲裁逻辑(Arbitration)。在每个周期，解码器从指令块中读取一条指令来执行，当遇到“BS＝1”(表示指令块开始)的指令时，该“锁”逻辑被使能(enable)，仲裁逻辑记住当前的“wave_id”，也即仲裁逻辑只能从当前指令块中选择指令。当遇到“BE＝＝1”(表示指令块结束)的指令时，“锁”逻辑将被去使能(disable)，使得仲裁逻辑解锁，进入正常模式。换句话说，硬件必须完成整个指令块的执行方能切换到其他指令块，其逻辑示意图如图3所示。

[0046] 为了支持新指令，本申请实施例还提供了一种指令执行方法，如图4所示，下面将结合图4对其所包含的步骤进行说明。

[0047] 步骤S201：获取待执行指令。

[0048] 步骤S202：获取所述待执行指令中的关键信息，所述关键信息包括：源操作数地址信息和目的地地址信息。

[0049] 在获取待执行指令后，获取待执行指令中的关键信息，关键信息包括：源操作数地址信息和目的地地址信息。例如，获取图1所示的指令中的Operand0_ID0字段、Operand1_

ID1字段以及Operand2_ID2字段对应的源操作数地址信息，以及获取图1所示的指令中的Result_ID字段以及DF字段对应的目的地地址信息。其中，所述源操作数地址信息用于指示源操作数的来源，所述目的地地址信息用于指示目的数据的写入路径。

[0050] 步骤S203：判断所述源操作数地址信息指示的源操作数是否来源于直通路径。

[0051] 在获取到源操作数地址信息后，判断该源操作数地址信息指示的源操作数是否来源于直通路径，在为是时，执行步骤S204，在为否时，从源操作数地址信息指向的地址获取源操作数。

[0052] 其中，判断指示的源操作数是否来源于直通路径的过程可以是：通过判断所述源操作数地址信息是否包含第二标识来判断所述源操作数地址信息指示的源操作数是否来源于直通路径；在所述源操作数地址信息是否包含第二标识(VF＝125)时，表征所述源操作数地址信息指示的源操作数来源于直通路径。需要说明的是，用于指示操作数来源于直通路径的VF值并不限于125。

[0053] 步骤S204：从所述直通路径获取所需的源操作数。

[0054] 在源操作数地址信息指示的源操作数来源于直通路径时，从直通路径获取所需的源操作数。

[0055] 步骤S205：判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径。

[0056] 在获取到目的地地址信息后，判断该目的地地址信息指示的目的数据的写入路径是否为直通路径，在为是时，执行步骤S206，在为否时，将目的数据写入目的地地址信息指向的地址。

[0057] 可选地，判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径的过程可以是：通过判断所述目的地地址信息是否包含第一标识(DF＝1)来判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径；在所述目的地地址信息是否包含第一标识时，表征所述目的地地址信息指示的目的数据的写入路径为所述直通路径。需要说明的是，用于指示将目的数据写入直通路径的值并不限于1。

[0058] 步骤S206：将执行所述待执行指令的结果写入所述直通路径。

[0059] 在目的地地址信息指示的目的数据的写入路径为直通路径时，将执行所述待执行指令的结果写入直通路径。

[0060] 本申请实施例中，当编译器(软件程序)检测到硬件可以使用直通数据作为源数据时，其显式地将instruction0的目的数据直通到instruction1或instruction2的源，实现数据直通，从而硬件跳过了instruction0的VGPR写和instruction1或instruction2的VGPR读，节省了大量功耗。为了便于理解，下面将结合图5所示，对将本申请提供的方法应用在矩阵乘法中的示例进行说明。

[0061] 需要说明的是，图5中的3个A的临时寄存器(Temp Register For A)在物理意义上均为同一个临时寄存器，只不过是因为要延迟3个单位时刻，所以才体现出有3个；同理，图中的2个B的临时寄存器(Temp Register For B)在物理意义上均为同一个临时寄存器，只不过是因为要延迟2个单位时刻，所以才体现出有2个。由于VGPR只有一个读口，因此需要将三个输入操作数(A,B,C)在时间上错开，可以通过临时寄存器进行延迟，最后在算术运算单元(Arithmetic Logic Unit,ALU)的入口处对齐，也即在第一时刻将从VGPR中获取到操作

数A临时放在A的临时寄存器中，在第二时刻将从VGPR中获取到操作数B放在B的临时寄存器中，在第三时刻将从VGPR中获取到操作数B放在C的临时寄存器中，这样三个输入操作数(A，B，C)便可同时输入到算术运算单元ALU中进行计算。图中的虚线所示的流程逻辑为现有指令的执行逻辑，也即在执行instruction0(指令0)时，从VGPR中获取源操作数A、源操作数B和源操作数C(此时为0)，在计算完成时，需要将计算结果写入VGPR中；在执行instruction1(指令1)时，又从VGPR中获取源操作数A、源操作数B和源操作数C(此时为前一次计算的结果，即C1＝A0+B0+C0)，在计算完成时，需要将计算结果写入VGPR中，如此反复，直至完成矩阵计算。而采用本申请提供的新指令后的执行逻辑为图中的实线所示，也即图中标有①和②的实线。从中可以很清楚的看出，采用本申请提供的新指令后，ALU的输出被直接旁路到ALU的输入。其中，图中标有①的实线表示的是操作数相同的情况的数据直通，此时，直接将ALU的输出作为ALU的三个操作数的输入，也即适用于A＝B＝C的情形。图中标有②的实线表示的是直接将ALU的输出作为ALU的三个操作数其中一个的输入。其中，图中的三个结果的临时寄存器(Temp Register For Result)表示的三条路径，即ALU的输出要么作为操作数A的输入，要么作为操作数B的输入，要么作为操作C的输入。

[0062] 从图5可以清楚的看出，应用本申请实施例提供的方法后，仅涉及第一条指令的VGPR的读和最后一条指令的VGPR的写，省略了大量中间指令的VGPR的读和写，可以减少大量的功耗。接下来以具体的矩阵A乘以矩阵B的示例进行说明，在进行矩阵乘法时，instruction0的结果被用作instruction1的源操作数，instruction1的结果被用作instruction2的源操作数。在常规模式下，instruction0将结果被写入到VGPR，instuction1从VGPR读取其源操作数，instruction1将结果写入到VGPR，instuction2从VGPR读取其源操作数。以下以$C_{64x64}＝A_{64x64}*B_{64x64}$作为示例，需要说明的是，这里用64X64的矩阵大小仅是示例，不限于此。并且假设有64个算术运算单元(ALU)，每一个算术运算单元具有200x64 bit的VGPR空间。

[0063] 计算过程大致如下：

[0064] 1)矩阵A以线性模式加载到LDS(Local Data Share，本地共享单元)：

[0065] A(0,0)→LDS(Address0)；//A(0,0)存储在LDS的Address0的位置；

[0066] A(0,1)→LDS(Address1)；//A(0,1)存储在LDS的Address1的位置；

[0067] A(0,2)→LDS(Address2)；//A(0,2)存储在LDS的Address2的位置；

[0068] ……

[0069] 2)矩阵B加载到VGPR空间，如表2所示。

[0070] 表2

[0071]

ALU0	ALU1	ALU2	……	ALU62	ALU63
B0,0	B0,1	B0,2	……	B0,62	B0,63
B1,0	B1,1	B1,2	……	B1,62	B1,63
……	……	……	……	……	……
B63,0	B63,1	B63,2	……	B63,62	B63,63

[0072] 其中，不同的VGPR存储不同的行，在计算时，矩阵A中的元素逐个并行地被加载到64个ALU中，与64个向量通用寄存器中各自存储的列对应的元素进行相乘，64个ALU并行地将矩阵A中同一行中的元素逐个与矩阵B的对应元素产生的相乘结果依次累加，得到矩阵C

同一行的所有元素，从而完成矩阵A和第二矩阵B的乘法运算。

[0073] 3)计算矩阵C：

[0074] 在常规模式下计算矩阵C的指令如下：

[0075] M0_register＝start_address;//M0寄存器的初始地址，其中，M0寄存器用于存储读取矩阵A中每个元素的地址，并且在64个ALU并行地根据M0寄存器当前的地址从所LDS读取矩阵A中对应的元素后自动更新到下一个元素对应的地址；

[0076] //----------

[0077] //Calculate the first row of Matrix C(计算矩阵C的第一行):

[0078] //C(0,0)is calculated on ALU_Index0:ALU_Index＝0(ALU0计算C(0,0)).

[0079] //C(0,1)is calculated on ALU_Index1:ALU_Index＝1(ALU1计算C(0,0)).

[0080] //.....

[0081] //----------每个ALU分别计算矩阵C的第一行中对应的一个元素的执行指令如下：

[0082] Block_Start::C(0,ALU_Index)＝LDS_Direct(M0_register)*B(0,ALU_Index)；

[0083] C(0,ALU_Index)＝LDS_Direct(M0_register)*B(1,ALU_Index)+C(0,ALU_Index)；

[0084] C(0,ALU_Index)＝LDS_Direct(M0_register)*B(2,ALU_Index)+C(0,ALU_Index)；

[0085] C(0,ALU_Index)＝LDS_Direct(M0_register)*B(3,ALU_Index)+C(0,ALU_Index)；

[0086] C(0,ALU_Index)＝LDS_Direct(M0_register)*B(4,ALU_Index)+C(0,ALU_Index)；

[0087] .....

[0088] Block_End::C(0,ALU_Index)＝LDS_Direct(M0_register)*B(63,ALU_Index)+

[0089] C(0,ALU_Index)；

[0090] //----------

[0091] //Calculate the second row of Matrix C(计算矩阵C的第二行):

[0092] //C(1,0)is calculated on ALU_Index0(ALU0计算C(1,0)).

[0093] //C(1,1)is calculated on ALU_Index1(ALU1计算C(1,1)).

[0094] //.....

[0095] //----------每个ALU分别计算矩阵C的第二行中对应的一个元素的执行指令如下：

[0096] Block_Start::C(1,ALU_Index)＝LDS_Direct(M0_register)*B(0,ALU_Index)；

[0097] C(1,ALU_Index)＝LDS_Direct(M0_register)*B(1,ALU_Index)+C(1,ALU_Index)；

[0098] C(1,ALU_Index)＝LDS_Direct(M0_register)*B(2,ALU_Index)+C(1,ALU_Index)；

[0099] C(1,ALU_Index)＝LDS_Direct(M0_register)*B(3,ALU_Index)+C(1,ALU_Index)；

[0100] C(1,ALU_Index)＝LDS_Direct(M0_register)*B(4,ALU_Index)+C(1,ALU_Index)；

[0101] ......

[0102] Block_End::C(1,ALU_Index)＝LDS_Direct(M0_register)*B(63,ALU_Index)+

[0103] C(1,ALU_Index)；

[0104] ......

[0105] //----------

[0106] //Calculate the last row of Matrix C(计算矩阵C的最后一行)：

[0107] //C(63,0)is calculated on ALU_Index0(ALU0计算C(63,0))．

[0108] //C(63,1)is calculated on ALU_Index1(ALU1计算C(63,1))．

[0109] //......

[0110] //----------每个ALU分别计算矩阵C的最后一行中对应的一个元素的执行指令如下：

[0111] Block_Start::C(63,ALU_Index)＝LDS_Direct(M0_register)*B(0,ALU_Index)；

[0112] C(63,ALU_Index)＝LDS_Direct(M0_register)*B(1,ALU_Index)+C(63,ALU_Index)；

[0113] C(63,ALU_Index)＝LDS_Direct(M0_register)*B(2,ALU_Index)+C(63,ALU_Index)；

[0114] C(63,ALU_Index)＝LDS_Direct(M0_register)*B(3,ALU_Index)+C(63,ALU_Index)；

[0115] ......

[0116] Block_End::C(63,ALU_Index)＝LDS_Direct(M0_register)*B(63,ALU_Index)+

[0117] C(63,ALU_Index)；

[0118] 参照上述指令表，可以看出：计算矩阵C的每一行需要64条指令，因此，总指令数是64x64＝4096。每条指令都在每一个线程上执行一次，因此，总执行次数是64x64x64。用于计算矩阵C每一行的指令块的第一条指令，例如如下：

[0119] C(63,ALU_Index)＝LDS_Direct(M0_register)*B(0,ALU_Index)；

[0120] 这里存在一次VGPR读，一次VGPR写，这种指令一共出现64次，因此一共有64x64次读和64x64次写。指令块的其他行，例如如下：

[0121] C(63,ALU_Index)＝LDS_Direct(M0_register)*B(1,ALU_Index)+C(63,ALU_Index)；

[0122] 这里存在两次VGPR读，一次VGPR写，这种指令一共出现63x64次，因此一共有2x63x64x64次读和63x64x64次写。有关VPGR的读写总结如表3所示。

[0123] 表3

[0124]

	矩阵B：源矩阵	矩阵C：结果矩阵	总数
读	64x64x64=218	$63x64x64=63x2^{12}\approx2^{18}$	2^{19}
写	0	$64x64x64=2^{18}$	2^{18}

[0125]

	$2^{18}+2^{19}$

[0126] 利用本申请实施例提供的显式的向量直通技术,计算矩阵C的指令如下:

[0127] M0_register＝start_address;

[0128] //--

[0129] //Calculate the first row of Matrix C(计算矩阵C的第一行):

[0130] //C(0,0)is calculated on ALU_Index0:ALU_Index＝0.

[0131] //C(0,1)is calculated on ALU_Index1:ALU_Index＝1.

[0132] //......

[0133] //--

[0134] Block_Start::Forwarding＝LDS_Direct(M0_register)*B(0,ALU_Index);

[0135] Forwarding＝LDS_Direct(M0_register)*B(1,ALU_Index)+Forwarding;

[0136] Forwarding＝LDS_Direct(M0_register)*B(2,ALU_Index)+Forwarding;

[0137] Forwarding＝LDS_Direct(M0_register)*B(3,ALU_Index)+Forwarding;

[0138] Forwarding＝LDS_Direct(M0_register)*B(4,ALU_Index)+Forwarding;

[0139] ......

[0140] Block_End::C(0,ALU_Index)＝LDS_Direct(M0_register)*B(63,ALU_Index)+Forwarding;

[0141] //--

[0142] //Calculate the second row of Matrix C(计算矩阵C的第二行):

[0143] //C(1,0)is calculated on ALU_Index0.

[0144] //C(1,1)is calculated on ALU_Index1.

[0145] //..........

[0146] //--

[0147] Block_Start::Forwarding＝LDS_Direct(M0_register)*B(0,ALU_Index);

[0148] Forwarding＝LDS_Direct(M0_register)*B(1,ALU_Index)+Forwarding;

[0149] Forwarding＝LDS_Direct(M0_register)*B(2,ALU_Index)+Forwarding;

[0150] Forwarding＝LDS_Direct(M0_register)*B(3,ALU_Index)+Forwarding;

[0151] Forwarding＝LDS_Direct(M0_register)*B(4,ALU_Index)+Forwarding;

[0152] ......

[0153] Block_End::C(1,ALU_Index)＝LDS_Direct(M0_register)*B(63,ALU_Index)+Forwarding;

[0154] ......

[0155] //--

[0156] //Calculate the last row of Matrix C(计算矩阵C的最后一行):

[0157] //C(63,0)is calculated on ALU_Index0.

[0158] //C(63,1)is calculated on ALU_Index1.

[0159] //..........

[0160] //--

[0161] Block_Start::Forwarding＝LDS_Direct(M0_register)*B(0,ALU_Index);

[0162] Forwarding＝LDS_Direct(M0_register)*B(1,ALU_Index)+Forwarding;

[0163] Forwarding＝LDS_Direct(M0_register)*B(2,ALU_Index)+Forwarding;

[0164] Forwarding＝LDS_Direct(M0_register)*B(3,ALU_Index)+Forwarding;

[0165] Forwarding＝LDS_Direct(M0_register)*B(4,ALU_Index)+Forwarding;

[0166] ......

[0167] Block_End::C(63,ALU_Index)＝LDS_Direct(M0_register)*B(63,ALU_Index)+Forwarding;

[0168] 参照上述指令表,可以看出:计算矩阵C的每一行需要64条指令,因此,总指令数是64x64＝4096。每条指令都在每一个线程上执行一次,因此,总执行次数是64x64x64。用于计算矩阵C每一行的指令块的最后一条指令,例如如下:

[0169] C(63,ALU_Index)＝LDS_Direct(M0_register)*B(63,ALU_Index)+Forwarding;

[0170] 这里,存在一次VGPR读,一次VGPR写,这种指令一共出现64次,因此一共有64x64次读和64x64次写。指令块的其他行,例如如下:

[0171] Forwarding＝LDS_Direct(M0_register)*B(4,ALU_Index)+Forwarding;

[0172] 这里,存在一次VGPR读,这种指令一共出现64次,因此一共有63x64x64次读。有关VPGR的读写总结如表4所示。

[0173] 表4

[0174]

	矩阵B:源矩阵	矩阵C:结果矩阵	总数
读	$64x64x64=2^{18}$	0	2^{18}
写	0	$64x64=2^{12}$	2^{12}
			$2^{12}+2^{18}\approx 2^{18}$

[0175] 综上所述,在上述典型的矩阵乘法示例中,利用本申请的显式向量直通技术,VGPR读写次数从$3x2^{18}$优化为2^{18},减少为大约1/3,因此可以节省大量能耗。

[0176] 如图6所示,本申请实施例还提供了一种指令生成装置100,包括:确定模块110、生成模块120和发送模块130。

[0177] 确定模块110,用于确定指令执行单元支持数据直通。

[0178] 生成模块120,用于在生成指令时,若需要将第i条指令的目的数据作为第i+j条指令的源操作数,在所述第i条指令中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识,以及在所述第i+j条指令中设置用于指示从所述直通路径获取源操作数的第二标识;i为正整数。

[0179] 发送模块130,用于将所述第i条指令、所述第i+j条指令发送给所述指令执行单元,以使所述指令执行单元在执行所述第i条指令时,根据所述第一标识将所述第i条指令的结果写入直通路径,以及,在执行所述第i+j条指令时,根据所述第二标识从所述直通路径中获取所需的源操作数。

[0180] 可选地,所述发送模块130,用于按照生成顺序将所述第i条指令、所述第i+j条指令进行拼接,得到指令块;将所述指令块发给解码器,以使所述解码器从所述指令块中依次获取所述第i条指令中的第一关键信息,并将所述第一关键信息发给所述指令执行单元,以及获取所述第i+j条指令中的第二关键信息,并将所述第二关键信息发给所述指令执行单元,所述第一关键信息包括所述第一标识,所述第二关键信息包括所述第二标识。

[0181] 本申请实施例所提供的指令生成装置100，其实现原理及产生的技术效果和前述方法实施例相同，为简要描述，装置实施例部分未提及之处，可参考前述方法实施例中相应内容。

[0182] 如图7所示，图7示出了本申请实施例提供的一种处理器200的结构框图。所述处理器200包括：处理器核心210(内核)、解码器220以及指令执行单元230。处理器核心210、解码器220和指令执行单元230之间通过总线互连连接。

[0183] 处理器核心210中固化了程序代码，当这些程序代码被执行时，用于生成指令，对应到本申请中，该处理器核心210用于确定指令执行单元230支持数据直通；以及在生成指令时，若需要将第i条指令的目的数据作为第i+j条指令的源操作数，则在所述第i条指令中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识，以及在所述第i+j条指令中设置用于指示从所述直通路径获取源操作数的第二标识；i为正整数；以及还用于将所述第i条指令、所述第i+j条指令发送给所述指令执行单元230；其中，i、j均为正整数。

[0184] 所述指令执行单元230，用于在执行所述第i条指令时，根据所述第一标识将所述第i条指令的结果写入直通路径，以及，在执行所述第i+j条指令时，根据所述第二标识从所述直通路径中获取所需的源操作数。

[0185] 可选地，所述处理器核心，还用于按照生成顺序将所述第i条指令、所述第i+j条指令进行拼接，得到指令块；将所述指令块发给所述解码器220。相应地，解码器220，用于从所述指令块中依次获取所述第i条指令中的第一关键信息，并将所述第一关键信息发给所述指令执行单元230，以及获取所述第i+j条指令中的第二关键信息，并将所述第二关键信息发给所述指令执行单元230。其中，所述第一关键信息包括所述第一标识，所述第二关键信息包括所述第二标识。

[0186] 此外，所述解码器220，还用于获取待执行指令，并获取该待执行指令中的关键信息，所述关键信息包括：源操作数地址信息和目的地地址信息，所述源操作数地址信息用于指示源操作数的来源，所述目的地地址信息用于指示目的数据的写入路径；以及还用于将所述关键信息发给所述指令执行单元230。相应地，所述指令执行单元230，用于：判断所述源操作数地址信息指示的源操作数是否来源于直通路径；在所述源操作数地址信息指示的源操作数来源于直通路径时，从所述直通路径获取所需的源操作数；判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径；在所述目的地地址信息指示的目的数据的写入路径为所述直通路径时，将执行所述待执行指令的结果写入所述直通路径。

[0187] 可选地，所述指令执行单元230，用于通过判断所述源操作数地址信息是否包含第二标识来判断所述源操作数地址信息指示的源操作数是否来源于直通路径；在所述源操作数地址信息包含所述第二标识时，表征所述源操作数地址信息指示的源操作数来源于直通路径。可选地，所述指令执行单元230，用于通过判断所述目的地地址信息是否包含第一标识来判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径；在所述目的地地址信息包含所述第一标识时，表征所述目的地地址信息指示的目的数据的写入路径为所述直通路径。

[0188] 其中，上述的处理器200可能是一种集成电路芯片，具有信号的处理能力。上述的处理器可以是通用处理器，包括中央处理器(Central Processing Unit，CPU)、网络处理器(Network Processor，NP)等；还可以是数字信号处理器(Digital Signal Processor，

DSP)、专用集成电路(Application Specific Integrated Circuit,ASIC)、现场可编程门阵列(Field Programmable Gate Array,FPGA)或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件。可以实现或者执行本申请实施例中的公开的各方法、步骤及逻辑框图。通用处理器可以是微处理器或者该处理器200也可以是任何常规的处理器等。

[0189] 本申请实施例还提供了一种包括上述处理器器的电子设备,该电子设备可以是电脑、服务器等设备。

[0190] 本申请实施例还提供了一种非易失性计算机可读取存储介质(以下简称存储介质),该存储介质上存储有计算机程序,该计算机程序被上述的处理器200运行时,执行上方法实施例所示的指令生成方法以及指令执行方法所包含的步骤。

[0191] 需要说明的是,本说明书中的各个实施例均采用递进的方式描述,每个实施例重点说明的都是与其他实施例的不同之处,各个实施例之间相同相似的部分互相参见即可。

[0192] 在本申请所提供的几个实施例中,应该理解到,所揭露的装置和方法,也可以通过其它的方式实现。以上所描述的装置实施例仅仅是示意性的,例如,附图中的流程图和框图显示了根据本申请的多个实施例的装置、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上,流程图或框图中的每个方框可以代表一个模块、程序段或代码的一部分,所述模块、程序段或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意,在有些作为替换的实现方式中,方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如,两个连续的方框实际上可以基本并行地执行,它们有时也可以按相反的顺序执行,这依所涉及的功能而定。也要注意的是,框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合,可以用执行规定的功能或动作的专用的基于硬件的系统来实现,或者可以用专用硬件与计算机指令的组合来实现。

[0193] 另外,在本申请各个实施例中的各功能模块可以集成在一起形成一个独立的部分,也可以是各个模块单独存在,也可以两个或两个以上模块集成形成一个独立的部分。

[0194] 所述功能如果以软件功能模块的形式实现并作为独立的产品销售或使用时,可以存储在一个计算机可读取存储介质中。基于这样的理解,本申请的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的部分可以以软件产品的形式体现出来,该计算机软件产品存储在一个存储介质中,包括若干指令用以使得一台计算机设备(可以是个人计算机,笔记本电脑,服务器,或者网络设备等)执行本申请各个实施例所述方法的全部或部分步骤。而前述的存储介质包括:U盘、移动硬盘、只读存储器(Read-Only Memory,ROM)、随机存取存储器(Random Access Memory,RAM)、磁碟或者光盘等各种可以存储程序代码的介质。

[0195] 以上所述,仅为本申请的具体实施方式,但本申请的保护范围并不局限于此,任何熟悉本技术领域的技术人员在本申请揭露的技术范围内,可轻易想到变化或替换,都应涵盖在本申请的保护范围之内。因此,本申请的保护范围应所述以权利要求的保护范围为准。

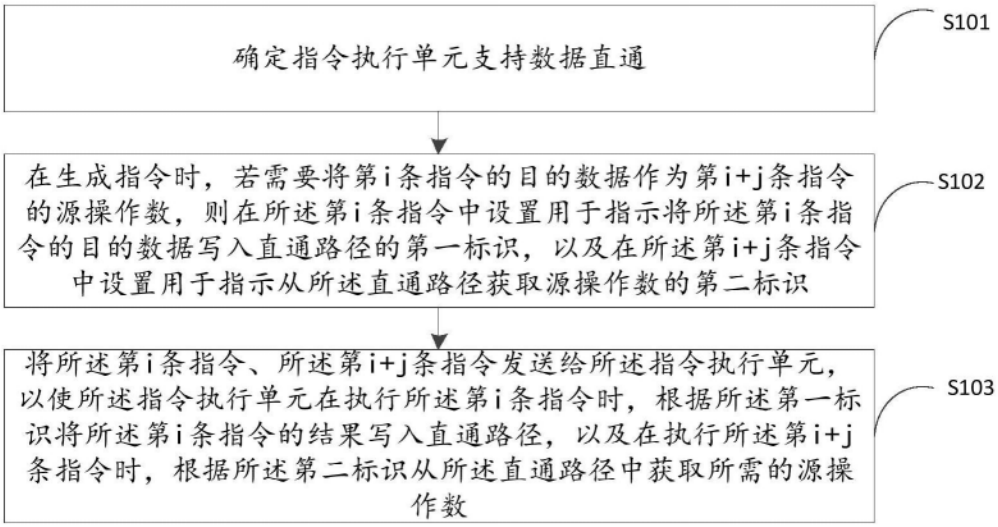

确定指令执行单元支持数据直通
S101
在生成指令时，若需要将第i条指令的目的数据作为第i+j条指令的源操作数，则在所述第i条指令中设置用于指示将所述第i条指令的目的数据写入直通路径的第一标识，以及在所述第i+j条指令中设置用于指示从所述直通路径获取源操作数的第二标识
S102
将所述第i条指令、所述第i+j条指令发送给所述指令执行单元，以使所述指令执行单元在执行所述第i条指令时，根据所述第一标识将所述第i条指令的结果写入直通路径，以及在执行所述第i+j条指令时，根据所述第二标识从所述直通路径中获取所需的源操作数
S103

图1

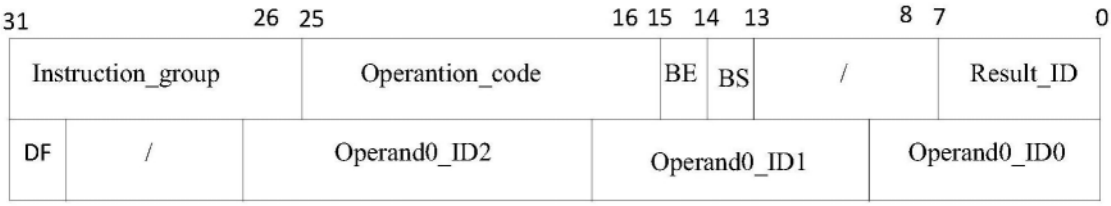

31
26
25
16
15
14
13
8
7
0
Instruction_group
Operantion_code
BE
BS
/
Result_ID
DF
/
Operand0_ID2
Operand0_ID1
Operand0_ID0

图2

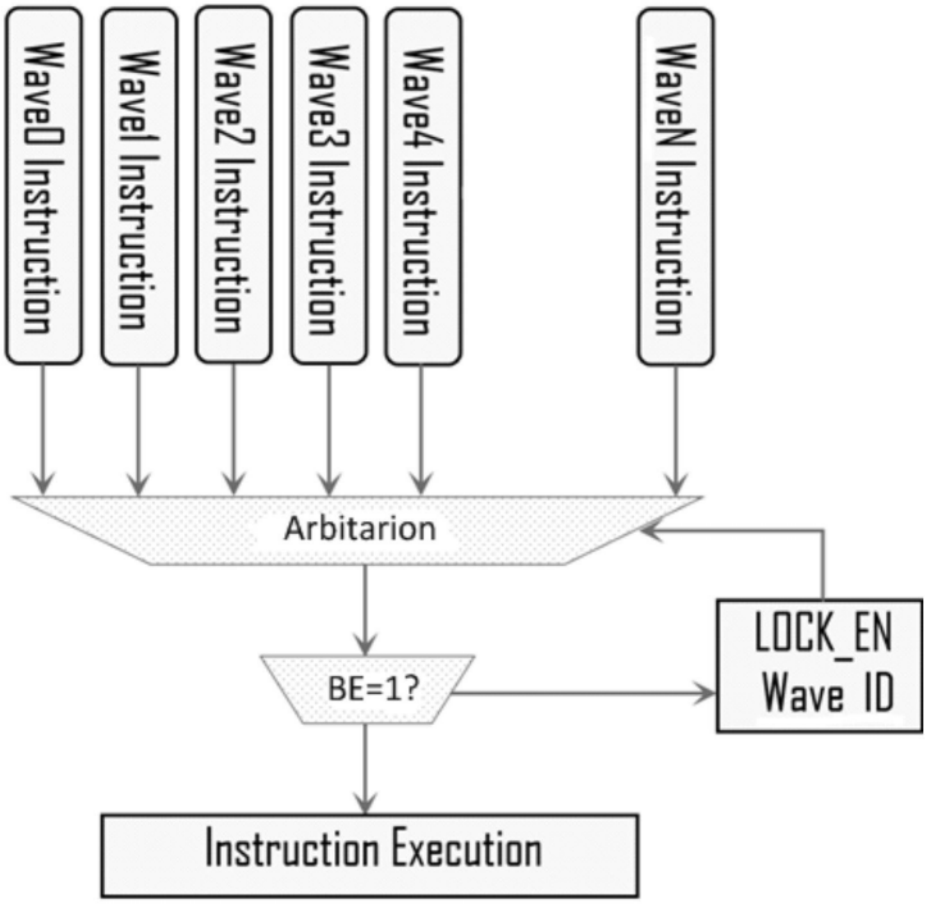

Wave0 Instruction
Wave1 Instruction
Wave2 Instruction
Wave3 Instruction
Wave4 Instruction
WaveN Instruction
Arbitarion
BE=1?
LOCK_EN
Wave ID
Instruction Execution

图3

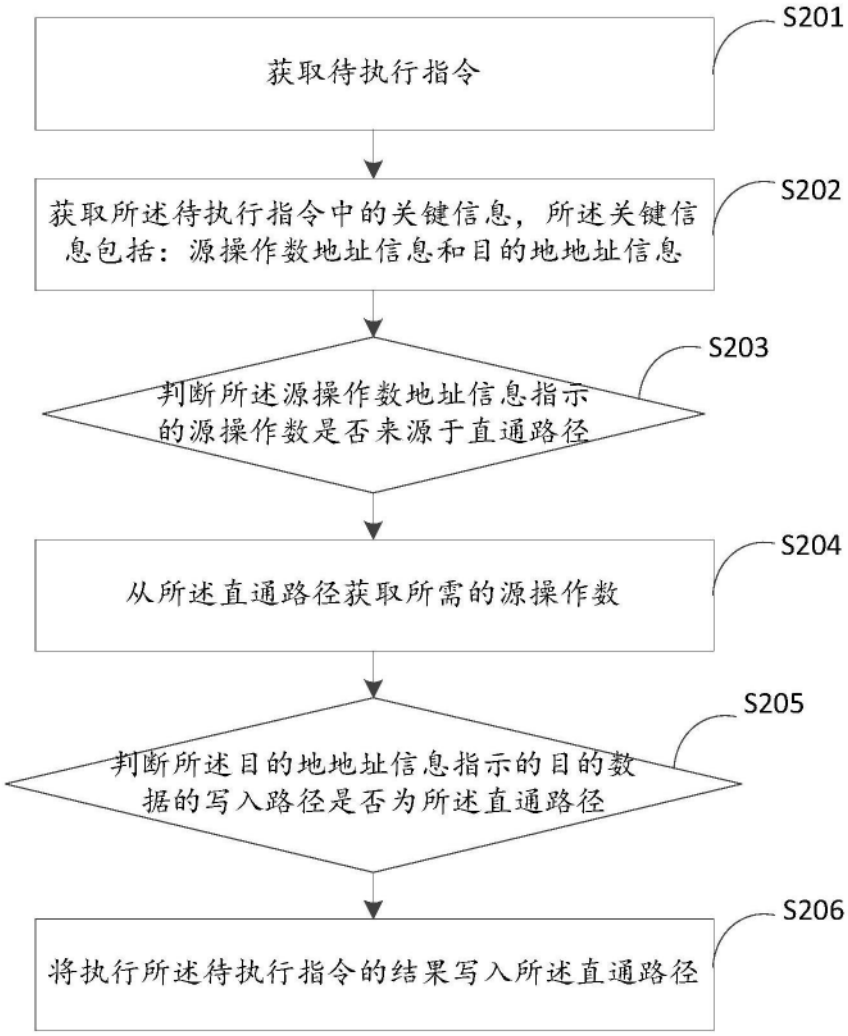

S201
获取待执行指令
S202
获取所述待执行指令中的关键信息，所述关键信息包括：源操作数地址信息和目的地地址信息
S203
判断所述源操作数地址信息指示的源操作数是否来源于直通路径
S204
从所述直通路径获取所需的源操作数
S205
判断所述目的地地址信息指示的目的数据的写入路径是否为所述直通路径
S206
将执行所述待执行指令的结果写入所述直通路径

图4

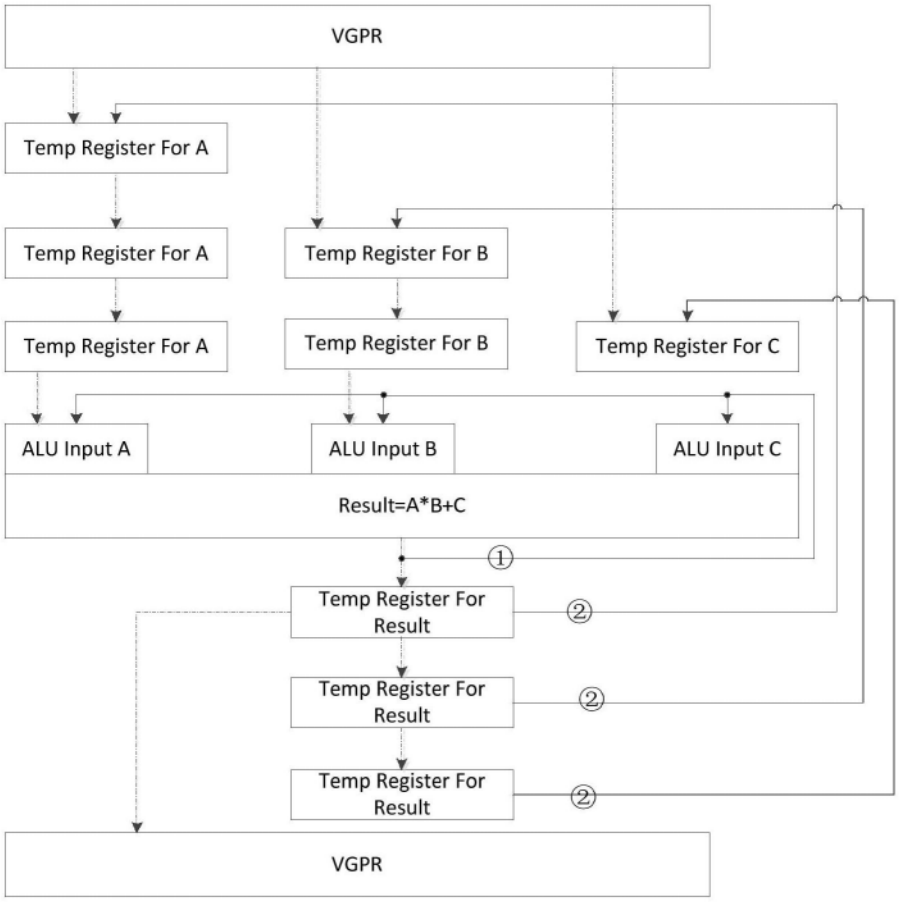

VGPR
Temp Register For A
Temp Register For A
Temp Register For B
Temp Register For A
Temp Register For B
Temp Register For C
ALU Input A
ALU Input B
ALU Input C
Result=A*B+C
①
Temp Register For Result
②
Temp Register For Result
②
Temp Register For Result
②
VGPR

图5

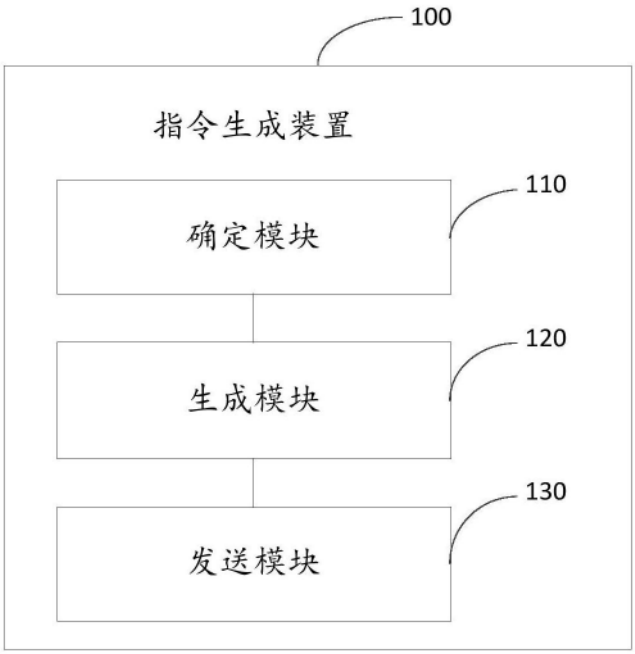

100
指令生成装置
确定模块
110
生成模块
120
发送模块
130

图6

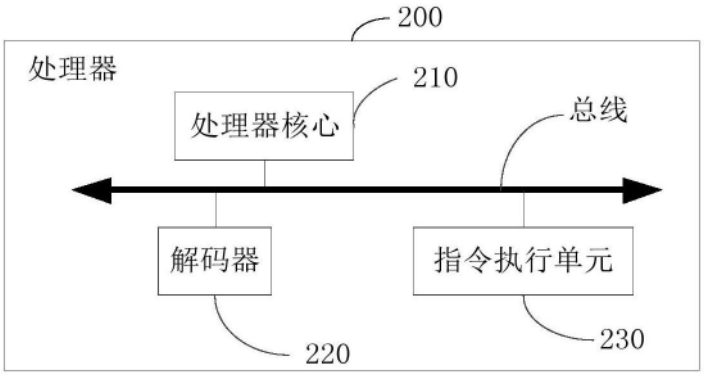


图7