(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 113168409 B
(45)授权公告日 2024.10.01

(21)申请号 201980080587.2
(22)申请日 2019.10.16
(65)同一申请的已公布的文献号
申请公布号 CN 113168409 A
(43)申请公布日 2021.07.23
(30)优先权数据
16/212,134 2018.12.06 US
(85)PCT国际申请进入国家阶段日
2021.06.04
(86)PCT国际申请的申请数据
PCT/US2019/056496 2019.10.16
(87)PCT国际申请的公布数据
WO2020/117377 EN 2020.06.11

(73)专利权人 赛灵思公司
地址 美国加利福尼亚州
(72)发明人 H·K·维尔马 B·田
(74)专利代理机构 北京市君合律师事务所 11517
专利代理师 毛健 杜小锋

(51)Int.Cl.
G06F 16/245(2006.01)
G06F 16/2453(2006.01)
(56)对比文件
WO 2016185542 A1,2016.11.24
审查员 曹如水

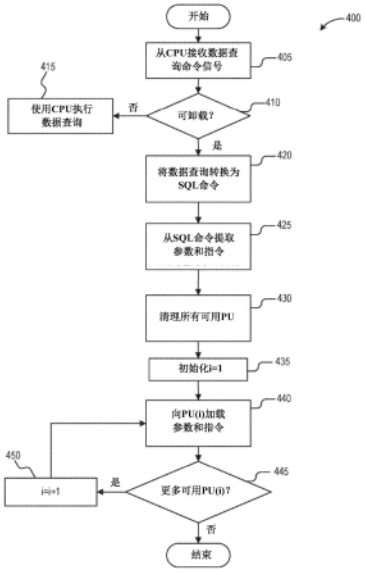
权利要求书2页 说明书12页 附图7页

(54)发明名称

加速数据查询的集成电路和方法

(57)摘要

与硬件加速相关的集成电路和方法包括独立的,可编程的,和并行的处理单元(PU),该处理单元(PU)定制地适应于处理数据流以及聚合结果以响应查询。在示出的例子中,来自数据库的数据流可以被分成数据块并且被分配给相应的PU。每一个数据块可以由其中所述PU中的一个处理以根据预定指令集产生结果。连接单元可以将每个数据块的结果合并和连接在一起,以生成查询的输出结果。在一些实施例中,例如,非常大型的数据库SQL查询可以通过在固定的ASIC或可重新配置的FPGA硬件电路中实现的硬件PU/连接引擎来加速。



CN 113168409 B

1.一种集成电路,所述集成电路包括:

多个处理单元,所述多个处理单元并行排列,所述多个处理单元中的每个处理单元包括第一寄存器,所述第一寄存器加载有指令集组的预定指令集,其中所述指令集组包括从由数据查询转换的预定查询语言命令中提取的多个指令;并且所述多个处理单元中的每个处理单元由硬件电路构成,所述硬件电路被配置为:

通过基于所述预定指令集的指令执行操作来处理数据流的所选的数据块;以及

产生与所述处理单元处理的所述数据流的所选部分相对应的中间输出结果;

以及,

连接电路,所述连接电路被耦接以从所述多个处理单元中的每个处理单元接收所述中间输出结果中的每个输出结果,并生成聚合结果。

2.根据权利要求1所述的集成电路,其特征在于,所述预定指令集的指令为SQL指令。

3.根据权利要求1所述的集成电路,其特征在于,所述预定查询语言命令为SQL命令。

4.根据权利要求1所述的集成电路,其特征在于,所述连接电路被配置为根据与用户定义的查询相关的预定功能生成所述聚合结果。

5.根据权利要求1所述的集成电路,其特征在于,所述多个处理单元中的每个处理单元实现为ASIC中的固定硬件电路。

6.根据权利要求1所述的集成电路,其特征在于,所述多个处理单元中的每个处理单元在FPGA的可编程结构中实现为可重新配置的硬件。

7.根据权利要求1所述的集成电路,其特征在于,所述集成电路还包括调度器电路,所述调度器电路被耦接以接收所述数据流,其中所述调度器电路被配置为选择性地将所述数据块中的每个数据块引导到所述多个处理单元中的一个处理单元。

8.根据权利要求7所述的集成电路,其特征在于,所述调度器电路包括轮循调度器。

9.根据权利要求1所述的集成电路,其特征在于,所述预定指令集中的每个预定指令集还包括从用户定义的查询提取的至少一个参数的功能。

10.根据权利要求9所述的集成电路,其特征在于,所述多个处理单元中的每个处理单元包括算数逻辑单元ALU,所述ALU适用于通过执行所述操作来执行对应的预定指令集,其中所述操作通过使用第一操作数和第二操作数被执行,所述第一操作数包括存储在变量寄存器中的所述数据流的一部分,所述第二操作数包括存储在常数寄存器中的提取的参数中的一个提取的参数。

11.根据权利要求10所述的集成电路,其特征在于,所述多个处理单元中的每个处理单元还包括:

临时寄存器,所述临时寄存器被配置为保持已执行操作的结果;

第一多路复用器,所述第一多路复用器被配置为从所述常数寄存器和所述变量寄存器接收输入;以及

第二多路复用器,所述第二多路复用器被配置为从所述常数寄存器、所述变量寄存器以及所述临时寄存器接收输入。

12.一种配置结构以执行数据查询的方法,所述方法包括:

接收来自用户的数据查询;

将所述数据查询转换为预定查询语言命令;

从所述预定查询语言命令中提取待存储在并行的多个处理单元中的参数,以及,

从所述预定查询语言命令中提取指令以形成待由所述处理单元执行的指令集组,所述指令集组包括多个指令集;以及,

将提取的参数加载到所述多个处理单元中的每个处理单元的第一寄存器中并且将提取的指令加载到所述多个处理单元中的每个处理单元的第二寄存器中,

其中,所述多个处理单元中的每个处理单元被配置为,用其各自的第一寄存器中的所述参数以及基于其各自的第二寄存器中的指令集的指令执行操作,并行地处理数据流的预定数据块。

13.根据权利要求12所述的方法,其特征在于,所述预定查询语言命令为SQL命令,并且所述提取的指令为SQL指令。

14.根据权利要求12所述的方法,其特征在于,所述多个处理单元中的每个处理单元被配置具有相同的指令集。

15.根据权利要求12所述的方法,其特征在于,所述方法进一步包括在加载所述提取的参数和所述提取的指令之前清理所有的处理单元。

加速数据查询的集成电路和方法

技术领域

[0001] 各种实施例大体上涉及具有用于硬件加速的可配置结构(fabric)的集成电路。

背景技术

[0002] 大数据用于指代数据集的研究以及应用,该数据集过于庞大和复杂以至于传统数据处理应用软件不足以处理它们。大数据带来了挑战,而且这些挑战包括但不限于数据捕获、数据存储、数据分析、数据更新、信息隐私以及数据查询。

[0003] 数据查询通常可以指代从数据库表或表组合请求数据或信息。可以将这些数据生成为结构化查询语言(SQL)返回的结果,也可以生成为图形、图表或复杂的结果(例如,来自数据挖掘工具的趋势分析)。SQL是用于在数据库中存储、处理和检索数据的标准语言。

发明内容

[0004] 与硬件加速有关的集成电路和方法包括独立的、可编程的、和并行的处理单元(PU),所述并行处理单元(PU)定制地适应于处理数据流以及聚合结果以响应查询。在示出的例子中,来自数据库的数据流可以被分成数据块并被分配给相应的PU。每一个数据块可以由所述PU中的一个处理以根据预定指令集产生结果。连接单元可以将每个数据块的结果合并和连接在一起,以生成查询的输出结果。在一些实施例中,例如,非常大型的数据库SQL查询可以通过在固定的ASIC或可重新配置的FPGA硬件电路中实现硬件PU/连接引擎来加速。

[0005] 在一些实施例中,现场可编辑门阵列(FPGA)可以提供形成在结构中的电气可重新配置的可编程硬件逻辑电路。FPGA结构可以是可重新配置的,以响应电配置信号,从而提供硬件资源的特定查询布置。如本文所教导,通过为每个查询唯一地定制硬件处理电路,可重新配置的结构可以被布置以产生硬件加速器用于有效地处理大型数据库的查询,从而使单个的FPGA能灵活地加速范围广泛的查询。

[0006] 在一些实施例中,可以制造专用集成电路(ASIC)以提供硬件逻辑电路(例如,数字、模拟)的固定布置。ASIC可以提供硬件加速器,该硬件加速器利用针对一个或多个预定查询的定制的硬件处理电路有效地处理大型数据库的查询,从而一个或多个ASIC(单独地或组合地)可以加速一个或多个预定的查询。

[0007] 各种实施例可以实现一个或多个优点。例如,一些实施例可以在执行例如大型数据库的查询时,显著地减少数据查询响应时间和/或增加处理吞吐量。数据库查询的硬件加速可以利用可定制的(例如,非固定的)硬件块实现,该可定制的硬件块根据每个特定查询的操作和参数被配置。一些实施方式可以在可编程结构设备中被编程以执行客户提供的大型数据库的查询。各种实施例可以将高速数据流处理从固定的中央处理单元(CPU)卸载到定制地编程的硬件处理通道,在该硬件处理通道中,数据块的多个数据流可以根据预定指令集被分别处理。因此,可以实现显著的计算效率,这可以致使对大规模数据库的查询的处理时间显著减少。

[0008] 例如,一些实施例可以降低制造成本和减少分散的非同步通讯,例如,通过利用可重新配置的结构设备(例如,FPGA)来用最小的硬件资源以执行高效的并行处理。例如,一些实施例可以在处理一些查询时相对于CPU提高内核级别的性能,和/或提供10-25倍的性能提高。在各种实施例中,最终用户(end user)可以一次或多次灵活地定制具有现场可编程能力的FPGA,以满足动态查询的要求。

[0009] 在一些实施例中,ASIC可以使用专用(例如,固定)硬件电路有利地为一个或多个预定的查询结构提供硬件加速能力。例如,合并了ASIC的一些实施例可以提供具有降低的组件的成本、体积、和/或功率需求的查询硬件加速。

[0010] 在一个示例性的方面,集成电路包括多个处理单元(PU(i)),所述多个处理单元并行布置。所述多个PU(i)中的每一个均由硬件电路构成,所述硬件电路配置为根据使用预定查询语言的指令集组(G)中的预定指令集(S(i))处理数据流的所选的数据块。每个PU(i)产生与该PU(i)处理的数据流的所选部分的相对应的中间输出结果。连接电路被耦接以从多个PU(i)的每一个接收每个中间输出结果,并生成聚合结果。每一个S(i)包括从用户定义的查询中提取的指令的功能。

[0011] 在一些实施例中,所述的预定指令集(S(i))可以包括SQL指令。在一些实施例中,预定查询语言可以包括SQL。在一些实施例中,连接单元可以被配置为根据与用户定义的查询相关的预定功能生成所述聚合结果。在一些实施例中,多个处理单元的每一个可以实现为ASIC中的的固定硬件电路。在一些实施例中,多个处理单元的每一个可以在FPGA的可编程结构中实现为可重新配置的硬件。

[0012] 在一些实施例中,集成电路可以包括调度器电路,所述调度器电路被耦接以接收数据流。所述调度器电路可以被配置为选择性地将每一个所述数据块引导到所述多个PU(i)之一。在一些实施例中,调度器电路可以包括轮循调度器。在一些实施例中,每一个S(i)可以进一步包括从用户定义的查询中提取的至少一个参数的功能。在一些实施例中,所述多个PU(i)的每一个可以包括算数逻辑单元ALU,所述算数逻辑单元适用于通过基于所提取的指令执行操作来执行对应的S(i)。所述操作可以通过使用(i)第一操作数和(ii)第二操作数被执行,所述第一操作数包括存储在变量寄存器中的数据流的一部分,所述第二操作数包括存储在常数寄存器中的提取参数中的一个。

[0013] 在一些实施例中,所述多个处理单元中的每一个可以进一步包括临时寄存器、第一多路复用器和第二多路复用器,所述临时寄存器被配置为保持已执行操作的结果,所述第一多路复用器被配置为从所述常数寄存器和所述变量寄存器接收输入,所述第二多路复用器被配置为从所述常数寄存器、所述变量寄存器以及所述临时寄存器接收输入。

[0014] 在一些实施例中,多个PU(i)中的每一个可以包括指令寄存器,所述指令寄存器被配置为存储将在ALU执行的S(i)。在一些实施例中,每一个PU(i)可以被配置具有相同的S(i)。在一些实施例中,每一个PU(i)可以被配置具有不同的S(i)。预定的查询语言指令集组G中的多个指令可以包括扫描指令和聚合指令。扫描指令可以包括逻辑与操作和逻辑或操作。

[0015] 在一些实施例中,一种配置结构以执行数据查询的方法包括:接收来自用户的数据查询,并将数据查询转换为预定查询语言命令。所述方法还包括从命令中提取待存储在多个并行处理单元PU(i)中的参数以及从命令中提取指令以形成由PU(i)待执行的指令集

组G。所述指令集组G包括多个指令集S(i)。所述方法还包括将提取的参数和提取的指令加载到多个PU(i)中。每一个PU(i)被配置为用其对应的参数和指令集S(i)并行地处理数据流的预定数据块(i)。

[0016] 在一些实施例中,所述预定查询语言可以包括SQL。在一些实施例中,每一个PU(i)可以被配置具有相同的S(i)。所述方法还可以包括在加载提取的参数和提取的指令之前清理所有的PU(i)。

[0017] 在附图和以下描述阐述了各种实施例的细节。根据说明书和附图以及根据权利要求书,其他特征和优点将明显。

附图说明

[0018] 图1描述了可在其中实现所公开的电路和过程的示例性可编程集成电路(IC)。

[0019] 图2(A)描述了具有示例性硬件加速处理引擎的主机计算系统。

[0020] 图2(B)描述了执行示例性数据查询的图2(A)的硬件加速处理系统的框图。

[0021] 图3(A)描述了包括在图2(B)硬件加速处理引擎中的示例性处理单元。

[0022] 图3(B)示出了用于操作图2(B)的示例性处理单元的示例性指令。

[0023] 图4描述了用于被配置以执行数据查询工作的硬件加速处理引擎的示例性设计-时间(design-time)结构重配置方法的流程图。

[0024] 图5描述了硬件加速处理引擎的示例性结构。

[0025] 图6描述了用于利用可编程的硬件加速处理引擎执行数据查询的示例性运行-时间(run-time)方法的流程图。

[0026] 在各个附图中,相似的参考符号指示相似的元件。

具体实施方式

[0027] 为了有助于理解,本文被组织如下。首先,参考图1简要地介绍可以在其上实现所公开的硬件加速处理引擎和过程的示例性可编程集成电路(IC)。其次,参考图2(A)至图3,讨论转到示例性实施例,该实施例说明可配置结构的结构和配置结构的方法。然后,参考图4(A)至图5,示出了处理单元的示例性结构和硬件加速处理引擎的示例性结构。最后,参考图6,示出了用于在运行时间(run-time)执行数据查询的示例性方法。

[0028] 图1描述了可以在其上实现所公开的电路和过程的示例性可编程集成电路(IC)。可编程IC 100包括FPGA逻辑。可编程IC 100利用各种可编程资源实现并且被称为片上系统(SOC)。FPGA逻辑的各种示例包括阵列中几种不同类型的可编程逻辑块。

[0029] 例如,图1示出了可编程IC 100,其包括大量不同的可编程单元块(programmable tile),可编程图单元块包括多千兆位收发器(MGT)101、可配置逻辑块(CLB)102、随机存取存储器块(BRAM)103、输入/输出块(IOB)104、配置和时钟逻辑(CONFIG/CLOCKS)105、数字信号处理块(DSP)106、专用输入/输出块(I/O)107(例如,时钟端口)、以及其他可编程逻辑108(例如,数字时钟管理器、模数转换器、系统监视逻辑)。可编程IC 100包括专用处理器块(PROC)110。可编程IC 100可以包括内部和外部重新配置端口(未示出)。

[0030] 在各种例子中,串行器/解串器(serializer/deserializer)可以通过使用MGT 101实现。MGT 101可以包括各种数据串行器以及解串器。数据串行器可以包括各种多路复

用器的实现。数据解串器可以包括各种解复用器的实现。

[0031] 在FPGA逻辑的一些示例中,每个可编程单元块包括可编程互连元件(INT)111,该可编程互连元件111具有去往和来自每个相邻单元块中的对应互连元件的标准连接124。因此,可编程互连元件一起实现了用于示出的FPGA逻辑的可编程互连结构。可编程互连元件INT 111包括去往和来自相同单元块中可编程逻辑元件的内部连接(intra-connection)120,如包括在图1的示例所示。可编程互连元件INT 111包括去往和来自相同单元块中可编程互连元件INT 111的INT之间的互连(inter-INT-connection)122,如包括在图1的示例所示。

[0032] 例如,CLB 102可以包括可配置逻辑元件(CLE)112,该可配置逻辑元件112可以被编程以连同单个可编程互连元件INT 111实现用户逻辑。BRAM 103可以包括BRAM逻辑元件(BRL)113以及一个或多个可编程互连元件。在一些示例中,单元块中包括的互连元件的数量取决于单元块的高度。在图示的实施方式中,BRAM单元块具有与5个CLB相同的高度,但是也可以使用其他数量(例如,4)。DSP单元块106可以包括DSP逻辑元件(DSPL)114以及一个或多个可编程互连元件。例如,IOB 104可以包括I/O逻辑元件(IOL)115的两个实例以及可编程互连元件INT 111的一个实例。例如,连接到I/O逻辑元件115的实际上的I/O焊盘可以使用金属制造,该金属层叠于各种所示的逻辑块上方,并且可以不限于输入/输出逻辑元件115的区域。

[0033] 图示的实施例中,靠近管芯中心的柱状区域(如图1的阴影所示)被用于配置、时钟、和其他控制逻辑。从列延伸的水平区域109在可编程IC 100的宽度上分配时钟和配置信号。值得注意,对“柱状”和“水平”区域的引用是相对于以纵向方向查看图形的。

[0034] 利用图1示出的架构的一些可编程IC可以包括其他逻辑块,该其他逻辑块破坏构成大部分的可编程IC的规则的柱状结构。其他逻辑块可以是可编程块和/或专用逻辑。例如,图1示出的处理块PROC 110横跨几列的CLB 102和BRAM 103。

[0035] 图1示出了可编程IC架构的示例。列中逻辑块的数量、列的相对宽度、列的数量和顺序、列中包括的逻辑块的种类、逻辑块的相对大小以及互连/逻辑的实施方式仅作为示例提供。例如,在实际的可编程IC中,无论CLB 102出现在何处,都可以包括多于一个相邻列的CLB 102,以促进用户逻辑的有效实现。

[0036] 随着计算机应用在各个领域的不断扩张,各种应用场景对服务器的数据处理能力提出了越来越高的要求。在某些特定的情况下,服务器可以很难平衡资源分配。为了达到所需的处理速度,需要更强大的计算能力。在一些情况下数据处理速度很重要,FPGA可被用于通过硬件加速器分担一些中央处理器(CPU)的工作并进行某些类型的计算。

[0037] 图2(A)描述了包括示例性的硬件加速处理引擎的主机计算系统。主机计算系统200包括硬件加速处理系统205。硬件加速处理系统205包括多个互连的电路子系统,其中之一是与现场可编程门阵列(FPGA)215电气耦接的中央处理单元(CPU)210。FPGA可以包括半导体集成电路衬底。FPGA 215可以通过硬件加速处理引擎分担一些中央处理单元(CPU)的工作,该硬件加速处理引擎可以灵活地重新配置以处理某些复杂的数据库查询指令。FPGA 215提供一个或多个硬件加速处理引擎225以加速数据查询响应速度。在各种实施方式中,可以将计算负担从CPU 210选择性地卸载到提供了特定于查询的硬件电路的硬件加速处理引擎225,以响应于大型数据库查询而有效地产生结果。

[0038] 硬件加速处理引擎225包括一组被并行放置以执行数据处理的处理单元(PU)230，以及配置为连接处理后的结果的连接单元235。每个处理单元230可以被独立地编程为，例如在数据流上，执行预定的过滤和/或聚合操作。在各种示例中，输入到每个处理单元PU 230的数据流可以包括操作和/或记录数据。例如，该操作可以通过用户经由用户接口向硬件加速处理系统205输入期望的查询来发起。记录的数据可以由硬件加速处理系统205从数据库中检索，例如，该数据库可以是与CPU 210进行可操作数据通信(例如，通过通信网络)的远程第三方或政府数据库。在一些实施方式中，CPU 210可以从数据库中检索大量数据记录，以使用用户输入查询的参数进行处理。在一些实施例中，那些操作包括SQL指令。

[0039] 在一些实施方式中，一个或多个处理引擎225可以通过单独或与FPGA 215结合使用定制的ASIC来形成。在这样的实施方式中，具有专用硬件电路的定制ASIC可以被配置为执行在所示的图中表示的一个或多个示例性处理引擎功能。例如，具有定制的固定硬件电路的ASIC可以被配置为硬件电路，该硬件电路被设计为用作能够有效执行查询操作的预定集的至少一部分的DDR读取器、缓冲器、处理单元PU 230和/或连接单元235中的一个或多个。ASIC可以布置有硬件电路，该硬件电路被配置为执行将由硬件加速处理系统205处理的的查询操作预定集。在一些示例中，ASIC中定义的定制的固定硬件配置可能能够执行查询指令，这些指令可以将例如CPU 210和/或一个或多个FPGA(例如，FPGA 215)的计算负担卸载到处理引擎225中。

[0040] 图2(B)描述了图2(A)的硬件加速处理系统执行示例性数据查询的框图。在示出的例子中，硬件加速处理系统205包括CPU 210、FPGA 215、数据库220、以及查询语言处理单元240，查询语言处理单元240包括在FPGA 215的电气可重新配置硬件电路中。在一些实施例中，查询语言处理单元240可以是SQL处理单元引擎。

[0041] CPU 210从用户接收数据查询请求并可以将数据查询命令信号发送到查询语言处理单元240，以对FPGA 215进行编程从而执行数据库查询，该数据库查询可以涉及例如处理数据流。在一些实施例中，数据流可以包括i个不同数据块。在一些实施例中，FPGA 215可以包括，例如，硬件资源、不同种类的寄存器、多路复用器、连接单元、和/或求和器(summers)。

[0042] 查询语言处理单元240提供适合于根据数据查询命令信号处理数据的预定指令集组G。指令集组G包括一个或多个指令集S(i)。如果查询语言处理单元240接受数据查询命令信号，则FPGA 215被重新配置为提供处理单元PU(i)以根据预定的指令集组G并行处理数据流。PU(i)是图2(A)的处理单元230。在一些实施例中，每个处理单元PU(i)可以被配置有相应的硬件资源集，以根据指令集组G中的预定指令集S(i)处理数据流的对应的预定数据块(i)。

[0043] 在示出的例子中，FPGA 215从数据库接收数据流(例如，通过一个或多个缓冲器)。从数据库检索数据流后，数据流的数据块(i)可以通过输入数据调度器分配给i个不同的处理单元PU(i)。然后，这些i个不同的数据块可以被i个并行处理单元PU(i)(例如，PU(1)、PU(2)、PU(3)、PU(4)、……)处理。每一个PU(i)根据其各自的(例如，独立的)指令集S(i)处理它的数据块(i)。在一些实施例中，所有PU(i)可以具有相同的指令集S(i)。在一些实施例中，每一个PU(i)可以具有不同的指令集S(i)。

[0044] 例如，每一个处理单元230可以被独立地编程以执行预定的过滤和聚合操作。在一些实施例中，这些操作可以包括查询指令的预定义组，例如，SQL指令。然后，硬件加速处理

引擎225可以将最终的查询结果传输到CPU 210。

[0045] 在一些实施例中,硬件加速处理引擎225和/或查询语言处理单元240可以部分地由CPU 210执行指令程序来实现,该指令程序在被执行时,至少部分地通过软件驱动的操作而不是完全通过硬件加速电路的操作,使得将被执行的操作产生硬件-加速查询处理的结果。在一些实施例中,硬件加速处理引擎225和/或查询语言处理单元240可以全部或部分地嵌入ASIC的固定电路中。在一些实施方式中,PU(i)可以通过例如嵌入在ASIC中的固定硬件电路和嵌入在FPGA中的可重新编程硬件电路的串联和/或并联组合实现。

[0046] 图3(A)描述了包括在图2(B)的硬件加速处理引擎中的示例性处理单元。在图3(A)的示例中,处理单元230(其可以是图2的处理单元230的实施例)包括指令寄存器305、变量寄存器310以及算数逻辑单元(ALU)325。指令寄存器305被用于存储正在被执行或解码的算数和/或逻辑指令。变量寄存器310接受被分配到处理单元230的数据。算数逻辑单元(ALU)325用于对计算机指令字中的操作数进行算数和逻辑运算。指令数据的类型可以是任何SQL类型。例如,指令数据可以是小数、整数、数据、布尔值(Boolean)。在描述的例子中,指令数据包括整数和布尔值。例如,添加到处理单元230的SQL类型和指令可以使处理单元230作为SQL处理单元操作。

[0047] 在一些实施例中,常数寄存器315可以用于存储常数数据。在操作中,可以在运行时间处理数据流之前输入代表用户提供的查询标准的常数数据,例如,以评估指令的结果(例如,变量数据(大于)常数数据)。在一些实施例中,可以使用临时寄存器320来保存中间结果。在一些实施例中,多路复用器330、335可以被用来选择需要由ALU 325处理的数据。在一些实施例中,可以向变量寄存器310加载数据流的数据块(i)。ALU 325可以通过执行存储在指令寄存器305中的编程指令来对加载的数据块(i)执行操作。

[0048] 在一些实施例中,多路复用器330可以是2:1多路复用器,以及多路复用器335可以是3:1多路复用器。在一些实施例中,多路复用器330可以从常数寄存器315和变量寄存器310接收输入,并且多路复用器335可以从常数寄存器315、变量寄存器310和临时寄存器320接收输入。在一些实施例中,指令寄存器305和常数寄存器315可以被独立地预先编程(例如,在设计时间将各个硬件电路配置为PU(i),该配置是在查询的运行时间执行之前执行的)以实现所需的功能。

[0049] 图3(B)示出了操作图2(B)的示例性处理单元的示例性指令。指令数据的类型可以是任何SQL类型。例如,指令数据可以是小数、整数、数据、布尔值。在所描述的例子中,指令数据包括整数和布尔值。例如,被添加到处理单元230的SQL类型和指令可以使处理单元230作为SQL处理单元操作。在一些实施例中,可以执行集合中的指令以进行SQL过滤和聚合操作。在一些实施例中,指令可以包括过滤操作、加或减运算、和乘法运算。在一些实施例中,过滤操作可以包括与(AND)、或(OR)、非(NOT)、相等(EQ)、不相等(NEQ)。在一些实施例中,这些指令可以是预先可编程的和可重新配置的。在一些实施例中,每一个处理单元230可以执行不同的预先可编程的指令。

[0050] 图4描述了用于硬件加速处理引擎的示例性设计时间结构重新配置的方法的流程图,该硬件加速处理引擎被配置为执行数据查询工作。在该说明性示例中,产生命令信号以执行可编程逻辑电路中可用的硬件资源,以形成硬件加速处理引擎(例如,图2(A)的硬件加速处理引擎225)。在示例性的方法400中,CPU(例如,CPU 210)在405通过CPU 210从用户接

收数据查询命令信号。参考图2A,当CPU 210接收数据查询命令信号时,为了提高数据查询响应速度,可以将查询工作的一部分卸载或分配到硬件加速处理引擎225。CPU 210在410确定是否可以将处理数据流的工作卸载或分配到FPGA 215。如果不适合将该工作卸载到硬件加速处理引擎225,则该过程的控制可以传递给CPU 210来执行指令,以在415执行执行数据查询,而无需使用硬件加速处理引擎225。

[0051] 在410,如果工作适合被卸载到硬件加速处理引擎225,则CPU 210在420将数据查询命令信号转换成预定查询语言命令,例如,SQL命令。例如,通过使用SQL命令,CPU210在425提取参数和指令。在所描述的例子中,在430,CPU 210通过清理先前已编程到FPGA 215的可编程逻辑中的任何预先存在的配置参数或指令,清理所有可用PU。

[0052] 为了准备将FPGA 215配置为卸载来自用户的数据查询命令信号,CPU 210在435开始变量i＝1。在440,CPU加载与处理单元PU(i)相对应的提取的参数和指令集。如果,在445,更多的PU(i)可用于处理任何其他的提取的参数和指令集,则在450,CPU对变量i递增并循环回到440。如果,在445,没有更多的PU(i)可用于处理任何其他的提取的参数和指令集,则方法400结束。

[0053] 在示出的例子中,FPGA 215可以被配置为选择是否从CPU卸载工作。例如,FPGA卸载开关可以被用作接受或拒绝卸载。是否接受或拒绝工作可以取决于工作的类型。在所描述的例子中,如果工作与数据扫描和聚合有关,FPGA 215可以被配置为接受工作。如果FPGA拒绝卸载,则CPU 210可以处理查询。响应于数据查询命令信号,查询语言处理单元240可以被配置为产生指令集组G。指令集组G可以包括一个或多个指令集S(i)。每一个并行的处理单元(例如,图2B的PU(i))可以被编程为执行预定的指令集S(i)。更具体地,查询语言处理单元240可以被配置为将查询转换成SQL命令,从SQL命令提取参数,以及产生将由FPGA 215中的每个对应的处理单元实现的指令集。在一些实施例中,每一个处理单元可以具有相同的指令集。在一些实施例中,每一个处理单元可以执行不同的指令集。

[0054] 接下来,指令集组G和提取的参数可以被加载到FPGA中并行的处理单元以处理数据流。例如,PU(1)可以被第一指令集S(1)加载,PU(2)和PU(3)可以被第二指令集S(2)加载。更具体地,对于每一个PU(i),常数寄存器(例如,图3(A)的常数寄存器315)可以被加载具有一个或多个常数。指令寄存器(例如,图3(A)的指令寄存器305)可以被加载具有预定指令集S(i)。在将新参数加载到每一个处理单元前,可以清理每一个处理单元。例如,可以清理常数寄存器和指令寄存器。在一些实施例中,还可以清理变量寄存器(例如,变量寄存器310)以及临时寄存器(例如,临时寄存器320)。

[0055] 然后,硬件加速处理引擎可以准备好以处理数据流。

[0056] 进一步描述了说明性示例以解释设计时间过程。例如,用户想知道在线零售商发生在1994年的累计收入,商品折扣在5％和7％之间,商品数量少于24个。然后,用户可以向CPU 210发送查询。查询与数据扫描和聚合有关。因此,FPGA 215具有处理该查询的能力。然后,CPU 210将查询工作卸载到FPGA。查询语言处理单元240可以接受查询并将查询转换成SQL命令。

[0057] 示例查询可以是:

[0058] SELECT SUM(L_EXTENDEDPRICE*L_DISCOUNT)AS REVENUE

[0059] FROM LINEITEM

[0060] WHERE L_SHIPDATE>＝’1994-01-01’AND L_SHIPDATE<’1995-01-01’

[0061] AND L_DISCOUNT BETWEEN.06-0.01AND.06+0.01AND L_QUANTITY<24

[0062] 可以使用软件将查询转换为SQL命令。软件可以从SQL命令中提取参数和指令。处理单元前的示例性的SQL命令如下所示：

[0063] 定义常数如下-

[0064] def l_quantity 0

[0065] def l_extendedprice 1

[0066] def l_discount 2

[0067] def l_shipdate 3

[0068] def d1994_01_01 1994/01/01

[0069] def c1 1

[0070] def d1995_01_01 1995/01/01

[0071] def c5 500

[0072] LD CONST[0]d1994_01_01

[0073] GE VAR[l_shipdate]CONST[0]TEMP[C0]

[0074] #save the result of"l_shipdate<date’1994-01-01’+interval’1’year"to TEMP1

[0075] LT VAR[l_shipdate]CONST[1]TEMP[1]

[0076] #save the result of"l_shipdate>＝date’1994-01-01’

[0077] #and l_shipdate<date’1994-01-1’+interval’1’year"to TEMP0

[0078] AND,TEMP,TEMP0,BOOL,TEMP,TEMP1,BOOL,TEMP,TEMP0,BOOL</V>

[0079] #Decompose"l_discount between.06-0.01and.06+0.01"as

[0080] #a."l_discount>＝.05"and

[0081] #b."l_discount<＝.07"

[0082] #save the result of"l_discount<.05"to TEMP1

[0083] <V>GE,VAR,l_discount,DECIMAL,CONST,exp3_1_con,DECIMAL,TEMP,TEMP1,BOOL</V>

[0084] <V>AND,TEMP,TEMP0,BOOL,TEMP,TEMP1,BOOL,TEMP,TEMP0,BOOL</V>

[0085] <V>LE,VAR,l_discount,DECIMAL,CONST,exp3_2_con,DECIMAL,TEMP,TEMP1,BOOL</V>

[0086] <V>AND,TEMP,TEMP0,BOOL,TEMP,TEMP1,BOOL,TEMP,TEMP0,BOOL</V>

[0087] #save the result of"l_quantity<24"to TEMP1

[0088] <V>LT,VAR,l_quantity,DECIMAL,CONST,exp4_con,DECIMAL,TEMP,TEMP1,BOOL</V>

[0089] #save the result of the whole qualifier to TEMP0

[0090] <V>AND,TEMP,TEMP0,BOOL,TEMP,TEMP1,BOOL,TEMP,TEMP0,

[0091] BOOL</V>

[0092] </E>

[0093] 如图4示出,可以将提取的参数和指令集S分别加载到在FPGA 215的可用处理单元

中以执行查询操作。当数据在运行时间流式传输,这些编程的处理单元可以开始执行查询操作,参考图6进一步详细描述其示例。

[0094] 图5描述了硬件加速处理引擎的示例性结构。参考图2(A-B),图5描述了硬件加速处理引擎225的实施例,其包括输入数据调度器505和连接单元235,数据调度器505协调对PU(i)230的输入,而连接单元235处理PU(i)230的输出。

[0095] 输入数据调度器505可以被配置为将输入数据流分成几个预定数据块。然后这些数据块可以被分配给适当配置的PU(i)230,以执行适合于该数据块的预定指令集S(i)。在一些实施例中,至少两个或多个PU(i)对数据块并行操作。在所描述的示例中,例如,数据流可以经由与数据流源(例如,数据库)可操作地通信的高速数据通信通道(例如,PCIe,DMA)被传递到硬件加速处理引擎225。由输入数据调度器505处理的流数据的每个块可以在传送到PU(i)230之前通过BRAM块被缓冲。

[0096] 连接单元235可以被配置为合并由每一个PU(i)230处理的结果。在一些实施例中,连接单元235可以是硬件单元。在一些实施中,每一个处理单元230可以被配置为执行预定数据过滤和聚合操作。

[0097] 硬件加速处理引擎225还包括调度器505,以调度在硬件加速处理引擎中的工作。调度器505将数据流中的数据块分配到相应的处理单元。在一些实施例中,调度器505可以是轮循(round-robin)调度器。在一些实施例中,每个处理单元230可以被注入其自己的具有不同的读取和写入数据宽度的HTTP实时串流(HLS)流。在一些实施例中,可以使用条带读取缓冲器(striped read buffer)来最小化使用块RAM读取和写入缓冲器。在一些实施例中,可以使用宽DDR(双倍数据速率)宽度来最大化使用的光学PU资源的效率和处理周期数量。在一些实施例中,DDR突发长度可以足够长以减少DDR的低效率。

[0098] 图6描述了用于使用可编程硬件加速处理引擎执行数据查询的示例性运行时间方法的流程图。在方法600中,可编程硬件加速处理引擎(例如,图5的硬件加速处理引擎)在605从数据库检索数据流。在610,调度器(例如,图5的调度器505)响应于编程的指令,将数据流调度和划分为不同的数据块,以由可编程硬件加速处理引擎中的不同、独立且可编程的处理单元(例如,图5中的处理单元230)处理。然后,在615,处理引擎中的每一个独立的且可编程的处理单元执行预定的数据过滤以及聚合操作以产生处理的结果。在一些实施例中,SQL查询可以被用于实现数据过滤和聚合操作。在620,这些处理的结果通过连接单元(例如,图5的连接单元235)被连接以形成输出结果。在625,确定是否终止数据查询过程。如果数据流有更多的记录要处理,则方法600返回到605。如果数据流没有更多的记录要处理,则该方法已确定查询的结果并完成。在一些实施例中,例如,然后可以在用户界面上显示该查询结果输出。

[0099] 在示出的例子中,查询的算法可以是:

```
LOAD INSTRUCTION REGISTERS

DO FOR EACH DATA BLOCK {
```

[0100]

```
    LOAD VARIABLE REGISTERS

    EXECUTE INSTRUCTIONS
```

MOVE RESULT BACK TO HOST CPU

[0101] }

[0102] 进一步描述了说明性示例以解释设计时间的过程。例如,FPGA 215可以包括可用于执行查询操作的5个处理单元。提取的参数和指令可能已经被加载到FPGA 215中以执行查询操作。然后,FPGA可以从数据库中检索与在线零售商的销售记录相关的数据流。销售记录可能包括200页数据。FPGA可以从数据中检索20页的记录数据。调度器505可以将在页面1中检索的数据分配给第一处理单元,并且将在页面2中的数据分配给第二处理单元,等等。分配结果的示例在下表显示。

[0103]

	分配数据到正确的 PU			
PU1	页面 1 中的数据	页面 6 中的数据	页面 11 中的数据	页面 16 中的数据
PU2	页面 2 中的数据	页面 7 中的数据	页面 12 中的数据	页面 17 中的数据
PU3	页面 3 中的数据	页面 8 中的数据	页面 13 中的数据	页面 18 中的数据
PU4	页面 4 中的数据	页面 9 中的数据	页面 14 中的数据	页面 19 中的数据
PU5	页面 5 中的数据	页面 10 中的数据	页面 15 中的数据	页面 20 中的数据

[0104] 当数据被加载到变量寄存器时,每个处理单元然后可以执行其指令。临时结果可以存储在临时寄存器320中。在处理单元PU1查询了所有页面1、6、11、16之后,PU1可以在指令下输出页面1、6、11、16的第一累积收入总和1。PU2将输出第二个累积收入总和2等等。输出连接单元235执行最终的计算并将最终的结果返回给用户。

[0105] 尽管已经参考附图描述了各种实施例,但其他实施例是可能的。例如,在一些实施例中,SQL查询可以被翻译成指令并且被加载以在FPGA上执行。在一些实施例中,可以执行不同的SQL查询而无需在FPGA上重新编译。在一些实施例中,这些执行的SQL查询可能适合于OLAP(在线分析处理)。在一些实施例中,PostgreSQL及其各种扩展可以用于数据分析和GIS应用。

[0106] 在一些实施例中,用户可以在加速的FPGA平台上执行他们现存的Postgres SQL查询。在一些实施例中,硬件加速处理引擎可以是大规模并行SQL处理单元,并且可以针对每个连续的用户查询即时生成用于大规模并行SQL处理单元的指令代码。在一些实施例中,可以在FPGA中对Postgres存储页面进行本地解析,以扫描关系行从而选择由“where”子句指定的行,并且用户可以使用所有现有的Postgres功能来执行远程或本地查询。

[0107] 在一些实施例中,例如,用户可以在VU9P设备上使用32SQL PU用于F1实现。在一些实施例中,每一个PU可以扩展到散列、排序或客户特定的指令。在一些实施例中,可以从使用多个缓冲器的FPGA对正在处理的数据进行块流式传输(block-streamed),以保持输入和输出数据,从而确保FPGA加速内核不会数据匮乏(starved)。

[0108] 实施例的一些方面可以被实现为计算机系统。例如,各种实施例可以包括数字和/或模拟电路、计算机硬件、固件、软件或其组合。设备元件可以在有形地体现在信息载体中(例如,在机器可读存储设备中)的计算机程序产品中实现,由可编程处理器执行;并且方法可以通过执行指令程序的可编程处理器来执行,以通过对输入数据进行操作并生成输出来

执行各种实施例的功能。一些实施例可以有利地在能够在可编程系统上执行的一个或多个计算机程序中实现,该可编程系统包括至少一个可编程处理器,该至少一个可编程处理器被耦合从数据存储系统、至少一个输入设备和/或至少一个输出设备接收数据和指令,以及向数据存储系统、至少一个输入设备和/或至少一个输出设备传输数据和指令。计算机程序是一组指令,可以在计算机中直接或间接使用它们来执行特定的活动或带来特定的结果。计算机程序可以以任何编程语言(包括编译或解释语言)的形式撰写,并且其可以以任何形式(包括作为独立程序或作为模块、组件、子例程或其他适合在计算环境中使用的单元)被部署。

[0109] 作为示例而不被限制,用于执行指令程序的合适处理器包括通用微处理器和专用微处理器,其可以包括任何种类计算机的单个处理器或多个处理器中的一个。通常,处理器可以从只读存储器或随机存取存储器或两者中接收指令和数据。计算机的基本元件是用于执行指令的处理器和用于存储指令和数据的一个或多个存储器。适用于有形地体现计算机程序指令和数据的存储设备包括所有形式的非易失性的存储器,包括,例如,半导体存储器设备(例如,EPROM、EEPROM以及闪存设备)。处理器和存储器可以由ASIC(专用集成电路)补充或并入ASIC。在一些实施例中,处理器和存储器可以由硬件可编程设备(例如FPGA)补充或并入硬件可编程设备。

[0110] 在一些实施方式中,每一个系统可以用相同或相似信息编程和/或用存储在易失性和/或非易失性存储器中基本相同的信息进行初始化。例如,一个数据接口可以被配置为在与适当主机设备(例如,台式计算机或服务器)连接时执行自动配置,自动下载和/或自动更新功能。

[0111] 在各种实施例中,计算机系统可以包括非暂时性存储器。存储器可以连接一个或多个处理器,该一个或多个处理器可以被配置为存储数据和计算机可读指令,包括处理器可执行程序指令。数据和计算机可读指令可以由一个或多个处理器访问。当由一个或多个处理器执行时,处理器可执行程序指令可以使一个或多个处理器执行各种操作。

[0112] 在各种实施例中,计算机系统可以包括物联网(IoT)设备。IoT设备可以包括嵌入有电子器件的对象、软件、传感器、执行器和使这些对象能够收集和交换数据的网络连接。IoT设备通过经由接口将数据发送到另一个设备,可以与有线或无线设备一起使用。IoT设备可以收集有用的数据,然后自主地在其他设备之间流动该数据。

[0113] 可以通过使用包括各种电子硬件的电路实现模块的各种示例。作为示例而非限制,硬件可以包括晶体管、电阻器、电容器、开关、集成电路和/或其他模块。在各种示例中,模块可以包括模拟和/或数字逻辑、分立组件、迹线和/或在包括各种集成电路的硅基板上制造的存储电路。在一些实施例中,模块可以包括执行由处理器执行的预编程指令和/或软件。例如,各种模块可以涉及硬件和软件两者。

[0114] 在示例中,集成电路包括:并行布置的多个处理单元(PU(i))和连接电路,多个PU(i)中的每一个由硬件电路形成,该硬件电路被配置为根据使用预定查询语言的指令集组(G)中的预定指令集(S(i))处理数据流的所选的数据块,多个PU(i)中的每一个生成与由PU(i)处理的数据流的所选部分对应的中间输出结果,连接电路被耦接以从多个PU(i)的每一个接收每一个中间输出结果以及生成聚合结果,其中,每一个S(i)包括从用户定义的查询中提取的指令的功能。

[0115] 在示例中,预定指令集(S(i))包括SQL指令。在示例中,预定查询语言包括SQL。在示例中,连接单元被配置为根据与用户定义的查询相关联的预定功能生成聚合结果。在示例中,多个处理单元的每一个被实现为ASIC中的固定硬件电路。在示例中,多个处理单元的每一个在FPGA的可编程结构中被实现为可重新配置硬件。在示例中,集成电路包括被耦接以接收数据流的调度器电路,其中调度器电路被配置为选择性地将每一个数据块引导到多个PU(i)中的一个PU(i)。在示例中,调度器电路包括轮循调度器。在示例中,每一个S(i)还包括从用户定义的查询中提取的至少一个参数的功能。

[0116] 在示例中,多个PU(i)的每一个包括算数逻辑单元ALU,其适于通过基于所提取的指令执行操作来执行相应的S(i),其中,所述操作通过使用(i)第一操作数和(ii)第二操作数被执行,第一操作数包括存储在变量寄存器中的数据流的一部分,第二操作数包括存储在常数寄存器中的提取的参数之一。在示例中,多个处理单元的每一个还包括:临时寄存器、第一多路复用器和第二多路复用器,临时寄存器被配置为保持已执行的操作的结果,第一多路复用器被配置为从常数寄存器和变量寄存器接收输入,第二多路复用器被配置为从常数寄存器、变量寄存器和临时寄存器接收输入。

[0117] 在示例中,一种用于配置结构以执行数据查询的方法,所述方法包括:接收来自用户的数据查询;将数据查询转换为预定查询语言命令;从命令中提取待存储在多个并行处理单元PU(i)中的参数,以及从命令中提取指令以形成由PU(i)待执行的指令集组G,指令集组G包括多个指令集S(i),以及将提取的参数和提取的指令加载到多个PU(i)中,其中每一个PU(i)被配置为用其对应的参数和指令集S(i)并行地处理数据流的预定数据块(i)。

[0118] 在示例中,预定查询语言包括SQL。在示例中,每一个PU(i)被配置具有相同S(i)。在示例中,该方法进一步包括在加载提取参数和提取指令前清理所有的PU(i)。

[0119] 已经描述了大量的实施例。尽管如此,可以理解的是,可以进行各种修改。例如,如果以不同的顺序执行所公开的技术的步骤,或者如果以不同的方式组合所公开的系统的组件,或者如果该组件补充有其他组件,则可能实现有利的结果。因此,其他实施方式在所附权利要求的范围内。

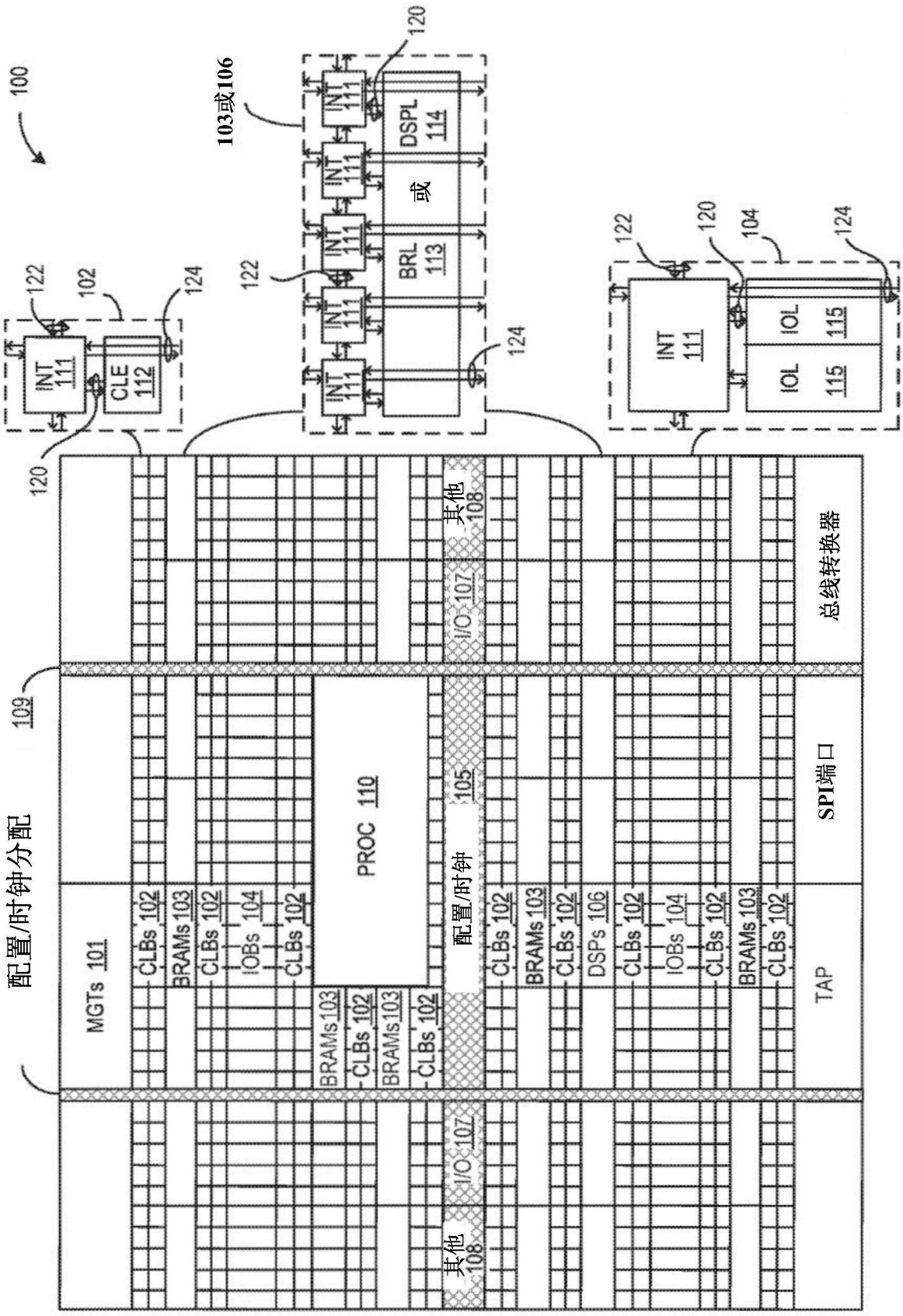

100
103或106
120
DSPL 114
或
BRL 113
INT 111
122
124
102
CLE 112
104
IOL 115
配置/时钟分配
109
MGTs 101
CLBs 102
BRAMs 103
IOBs 104
DSPs 106
PROC 110
其他 108
I/O 107
配置/时钟 105
总线转换器
SPI端口
TAP

图1

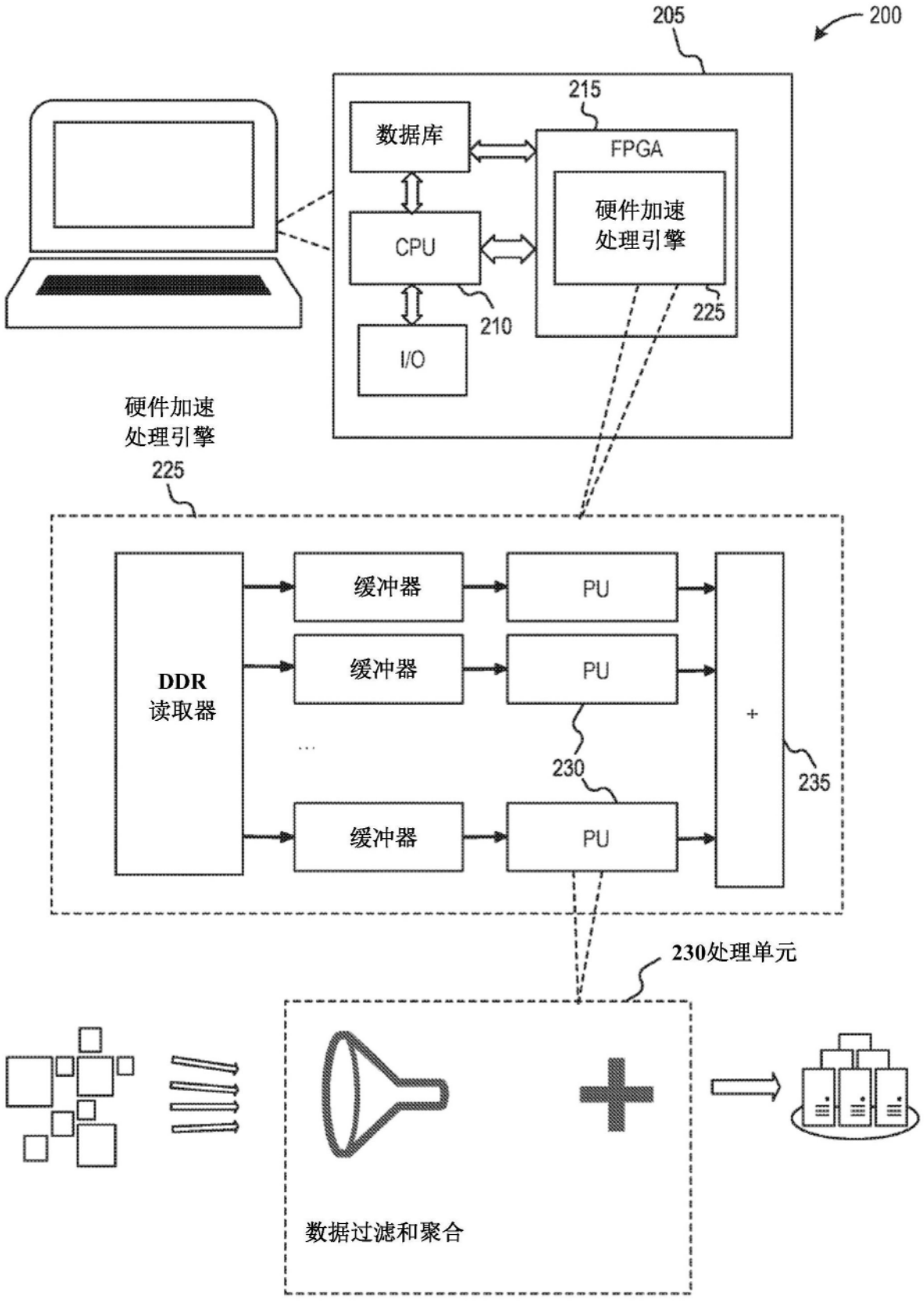

205
200
215
数据库
FPGA
硬件加速
处理引擎
CPU
210
225
I/O
硬件加速
处理引擎
225
DDR
读取器
缓冲器
PU
缓冲器
PU
...
230
缓冲器
PU
+
235
230处理单元
数据过滤和聚合

图2(A)

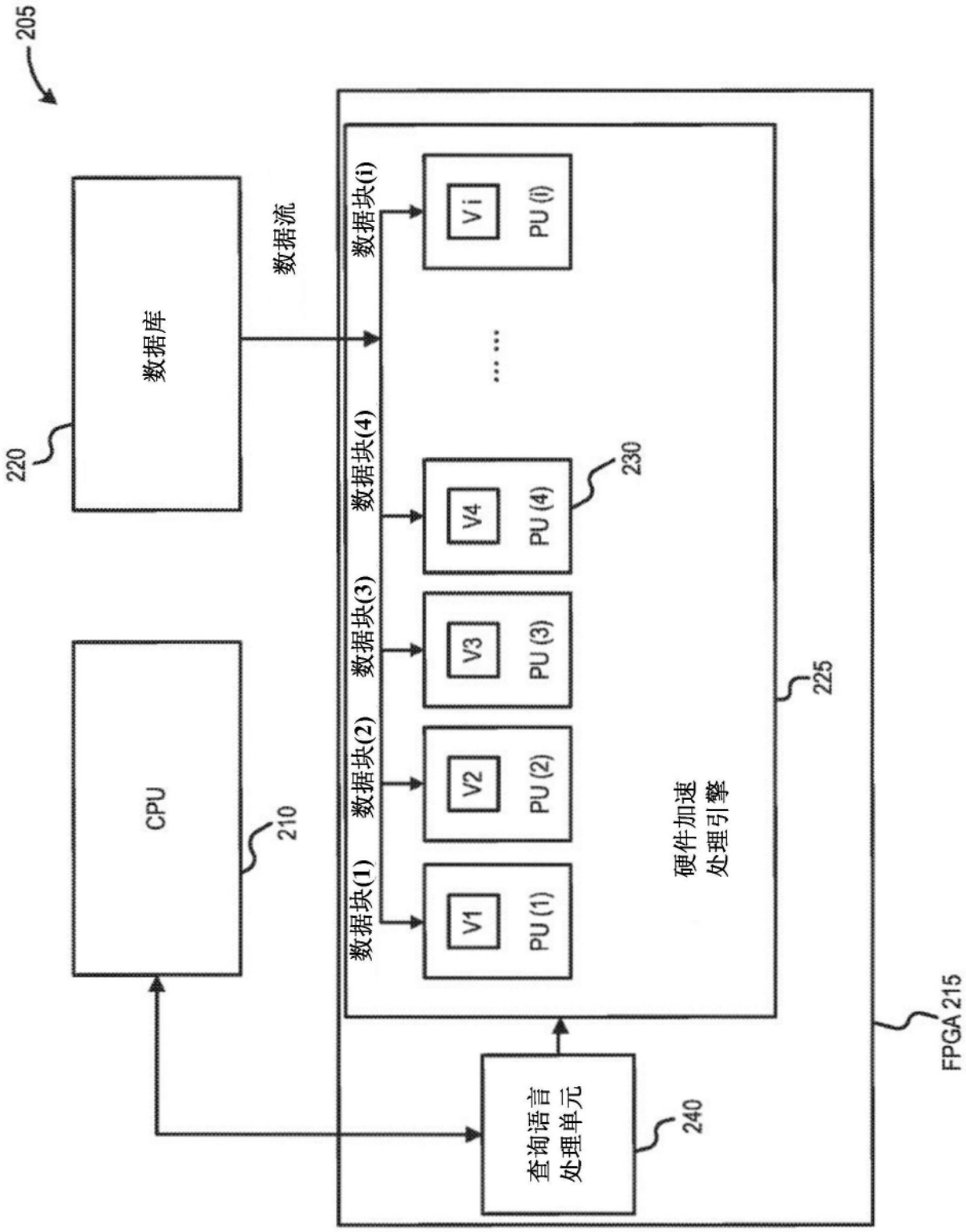

205
数据库
220
数据流
CPU
210
数据块(1)
数据块(2)
数据块(3)
数据块(4)
数据块(i)
V1
PU (1)
V2
PU (2)
V3
PU (3)
V4
PU (4)
Vi
PU (i)
230
硬件加速
处理引擎
225
查询语言
处理单元
240
FPGA 215

图2(B)

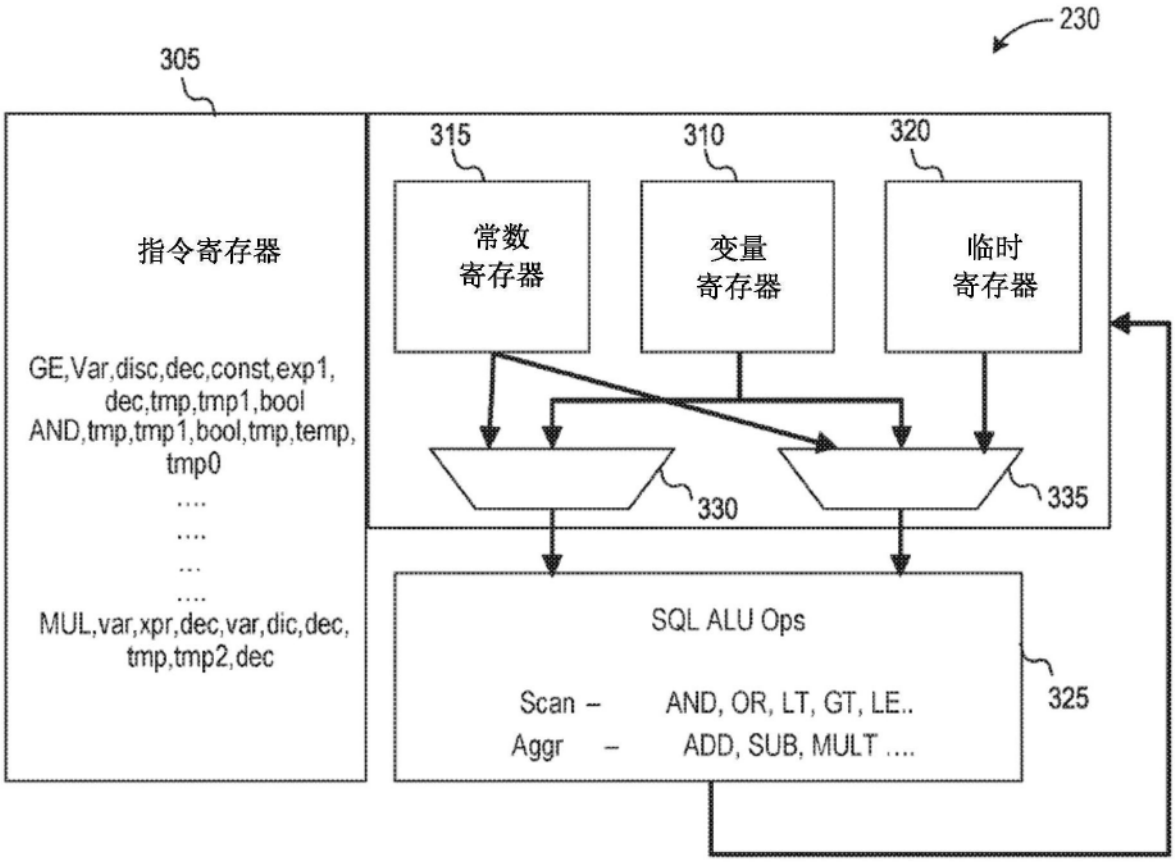


图3(A)

指令	左&右操作数	结果	说明
AND, OR	64 bit reg	1-bit bool	过滤操作
NOT, GT, GE, LT, LE, EQ, NEQ	64 bit reg	1-bit bool	过滤操作
PLUS, SUB	64 bit reg	64 bit reg	加/减
MULT	64 bit reg	64 bit reg	乘

图3(B)

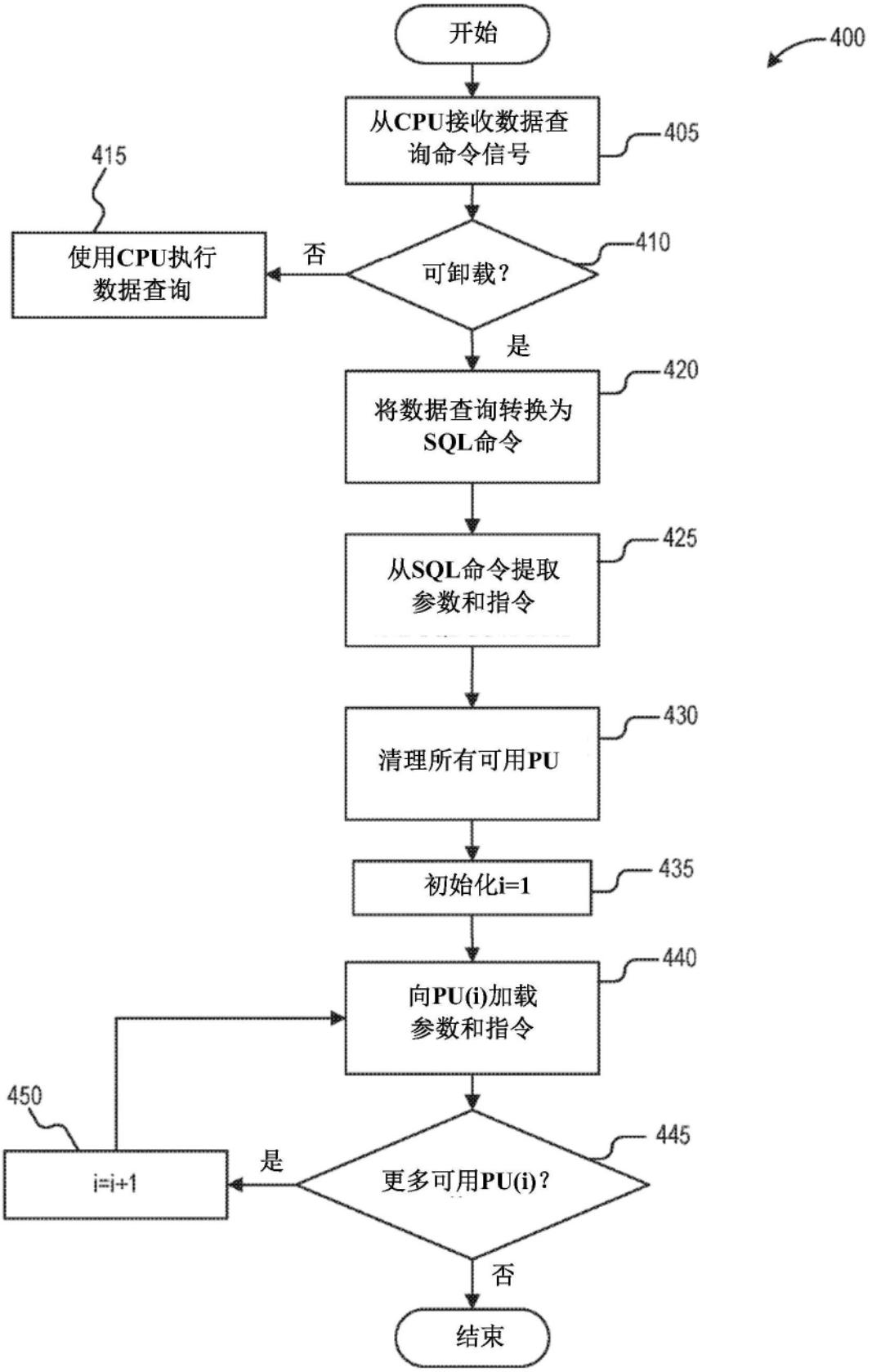

开始
400
从CPU接收数据查询命令信号
405
415
使用CPU执行数据查询
否
可卸载?
410
是
420
将数据查询转换为SQL命令
425
从SQL命令提取参数和指令
430
清理所有可用PU
435
初始化i=1
440
向PU(i)加载参数和指令
450
i=i+1
是
445
更多可用PU(i)?
否
结束

图4

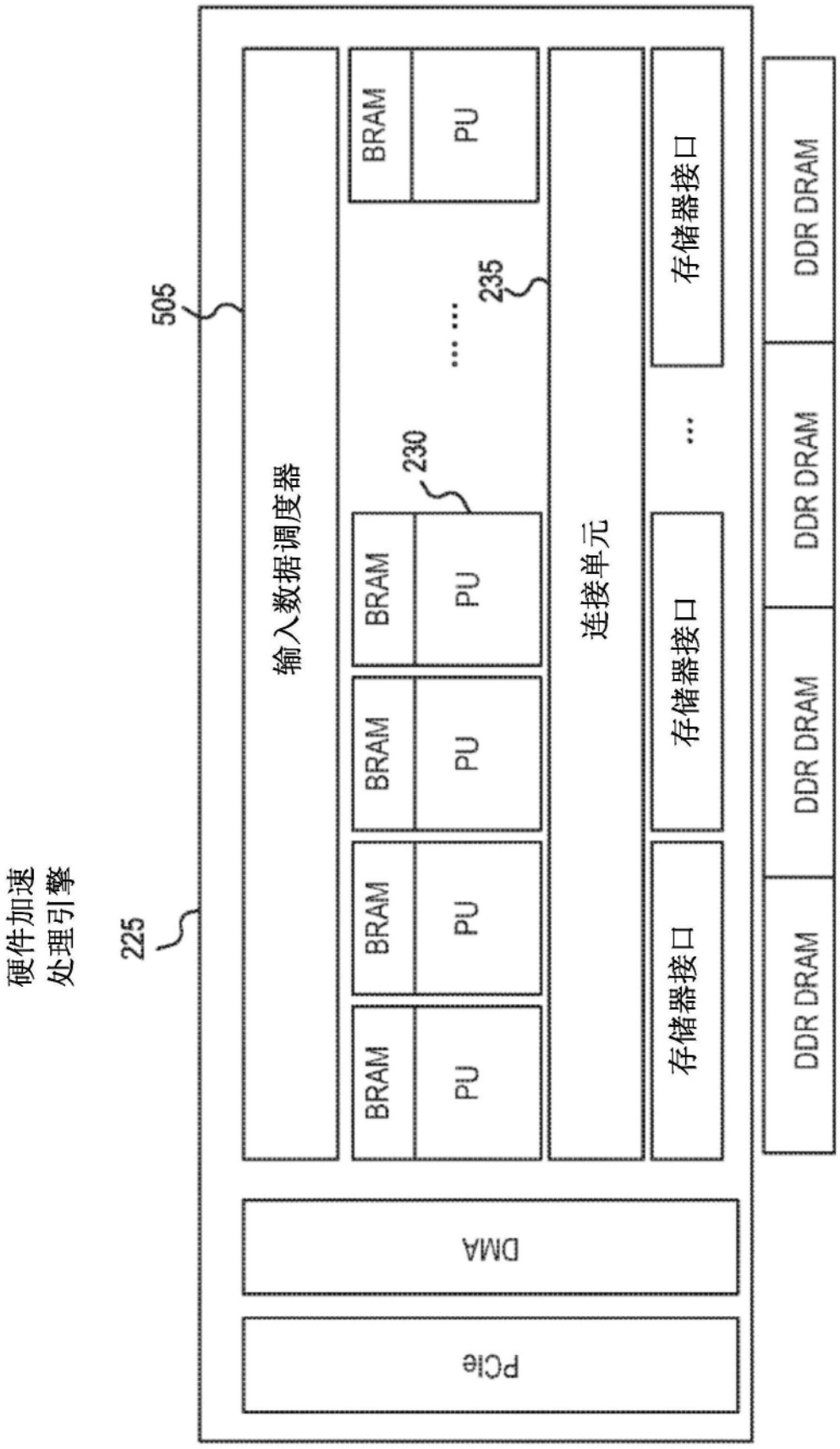

硬件加速
处理引擎
225
505
输入数据调度器
BRAM
PU
230
235
连接单元
存储器接口
DDR DRAM
PCIe
DMA

图5

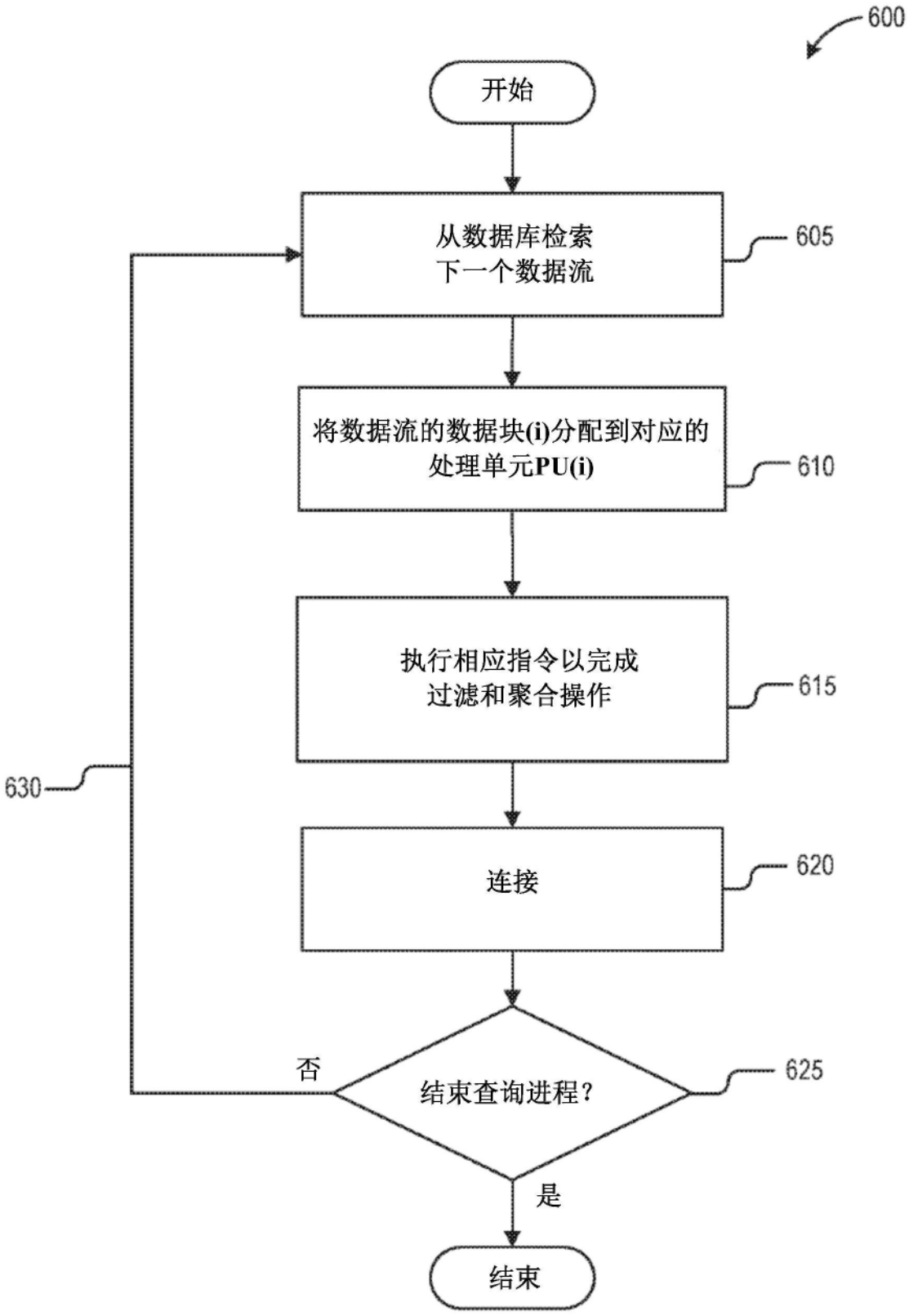

600
开始
从数据库检索
下一个数据流
605
将数据流的数据块(i)分配到对应的
处理单元PU(i)
610
执行相应指令以完成
过滤和聚合操作
615
630
连接
620
否
结束查询进程?
625
是
结束

图6