(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106452455 B
(45)授权公告日 2019.10.18

(21)申请号 201610846690.1
(22)申请日 2016.09.23
(65)同一申请的已公布的文献号
申请公布号 CN 106452455 A
(43)申请公布日 2017.02.22
(73)专利权人 华南理工大学
地址 510640 广东省广州市广州天河区五山路381号
(72)发明人 陆以勤 苏炜跃 覃健诚
(74)专利代理机构 广州粤高专利商标代理有限公司 44102
代理人 何淑珍
(51)Int.Cl.
H03M 13/11(2006.01)
H03M 13/00(2006.01)

审查员 杨奕树

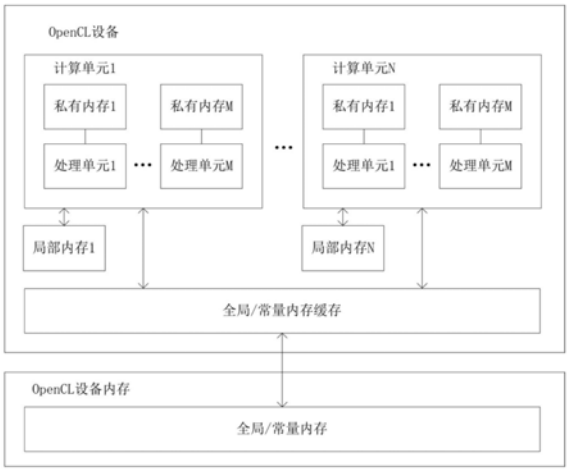
权利要求书3页 说明书5页 附图2页

(54)发明名称

基于OpenCL移动设备QC-LDPC的动态译码方法

(57)摘要

本发明提供基于OpenCL移动设备QC-LDPC的动态译码方法。本发明将QC-LDPC码的校验矩阵的母矩阵信息从宿主机传入OpenCL设备的常量存储器,并分配相应的内存空间;将待处理的码字信息传入OpenCL设备,当码字信息过大时,将码字信息分段,依次分别译码;将待处理的码字信息传入OpenCL设备,根据待译码的数据量来动态地选择最小和算法或者分层译码算法;读取相应的核函数,在OpenCL设备上完成译码算法。本发明针对QC结构LDPC码字的特点,实现了不同码率和码长的QC-LDPC码在不同OpenCL设备上通用的加速译码;并提供了LDPC译码器在OpenCL框架中的并行化实现。能在OpenCL上独立并行地运行很多个LDPC译码器,译码效率大幅度提高。本发明可以节省成本,并且容易软件升级,容易实现多种制式的全网通功能。



CN 106452455 B

1.基于OpenCL移动设备QC-LDPC的动态译码方法，其特征在于包括如下步骤：

步骤1、作为预处理，将QC-LDPC码的校验矩阵的母矩阵信息从宿主机传入OpenCL设备的常量存储器，并分配相应的内存空间；

读取校验矩阵的母矩阵信息，计算码字的参数，包括母矩阵的行数hSeedCols、列数hSeedRows和扩展因子z；

在OpenCL设备上的全局内存中分配相应的内存空间，包括用于存放未译码的码字信息和译码完成后的码字信息；

将待处理的码字信息传入OpenCL设备，由于移动设备上的OpenCL设备的全局内存受限，当码字信息过大时，将码字信息分段，依次分别译码；

步骤2、将待处理的码字信息传入OpenCL设备，根据待译码的数据量来动态地选择最小和算法或者分层译码算法；

步骤3、读取相应的核函数，在OpenCL设备上完成译码算法。

2.根据权利要求1所述的基于OpenCL移动设备QC-LDPC的动态译码方法，其特征在于步骤3具体包括：

(1)最小和算法的核函数在OpenCL设备上并行化实现，包括如下详细步骤：

1)在OpenCL设备上为信息节点后验对数似然比Lp_n和校验节点传递给信息节点的对数似然比Lr_{mn}分配局部内存，局部内存只能在工作组的共享；

2)根据OpenCL设备的参数信息设置内核的工作组数，每个工作组的工作项为z*hSeedRows，这些工作项合作完成译码任务，其中每个工作项负责完成校验矩阵中同一行内的信息的更新；

3)使用待译码字对信息节点后验对数似然比Lp_n^i,初始化，并将校验节点传递给信息节点的对数似然比Lr_{nm}^i置零，其中i代表当前迭代次数，完成初始化，

$$Lp_n^0 = \ln\frac{p_n^0(0)}{p_n^0(1)} = 2\frac{y_n}{\sigma^2} \qquad Lr_{nm}^0 = 0;$$

4)循环执行以下步骤：其中同一工作组内z*hSeedRows个工作项分别负责校验矩阵z*hSeedRows个行的相关信息的更新：

4.1)信息节点传递给校验节点的对数似然比Lq_{nm}^i的更新：

$$Lq_{nm}^i = Lp_n^0 + \sum_{m\epsilon M(n)\backslash m} Lr_{mn}^{i-1} \quad ;$$

4.2)校验节点传递给信息节点的对数似然比Lr_{nm}^i的更新：

$$Lr_{mn}^i = \prod_{n'\epsilon N(m)\backslash n} sign(Lq_{n'm}^i)\min_{n'\epsilon N(m)\backslash n}\left|Lq_{n'm}^i\right| \quad ;$$

4.3)信息节点后验对数似然比Lp_n^i的更新：

$$Lp_n^i = Lp_n^0 + \sum_{m\epsilon M(n)} Lr_{mn}^{i-1} ;$$

4.3)信息节点的硬判决和校验：

$$c_n^i = \begin{cases} 1 & if \ \ Lp_n^i < 0 \\ 0 & if \ \ Lp_n^i > 0 \end{cases} ;$$

5)循环以上1)～4)过程,直到经校验所译码为正确的,或者达到最大译码次数,则跳出循环,将译码结果传回宿主机内存;

(2)分层译码算法的核函数在OpenCL设备上并行化实现,包括如下详细步骤:

1)在OpenCL设备上为信息节点后验对数似然比$Lp_n^{i,k}$,局部内存只能在工作组的共享;校验节点传递给信息节点的对数似然比Lr_{nm}^i将保存在各个工作项的私有内存中;

2)根据OpenCL设备的参数信息设置内核的工作组数,每个工作组的工作项为z,这些工作项合作完成译码任务其中,每个工作项负责完成校验矩阵中同一层内的信息的更新;

3)使用待译码字对信息节点后验对数似然比Lp_n^i,初始化,并将校验节点传递给信息节点的对数似然比Lr_{nm}^i置零,其中i代表当前迭代次数,k代表当前所在层数,完成初始化,

$$Lp_n^{1,0} = \ln\frac{p_n^0(0)}{p_n^0(1)} = 2\frac{y_n}{\sigma^2} \qquad Lr_{nm}^{0,k} = 0 \ ;$$

4)循环执行以下步骤:其中同一工作组内z个工作项分别负责校验矩阵一个层内的z个行的相关信息的更新:

4.1)信息节点传递给校验节点的对数似然比Lq_{nm}^i的更新:

$$Lq_{nm}^{i,k} = Lp_n^{i,k-1} - Lr_{mn}^{i-1,k} \ ;$$

4.2)校验节点传递给信息节点的对数似然比Lr_{nm}^i的更新:

$$Lr_{mn}^{i,k} = \prod_{n' \epsilon N(m)\backslash n} sign(Lq_{n'm}^{i,k}) \min_{n' \epsilon N(m)\backslash n} \left| Lq_{n'm}^{i,l} \right| \ ;$$

4.2)信息节点后验对数似然比Lp_n^i的更新:

$$Lp_n^{i,k} = Lq_{nm}^{i,k} + Lr_{mn}^{i,k} \ ;$$

4.3)仅仅在完成最后一层的Lp_n^i的更新后,才进行信息节点的硬判决和校验:

$$c_n^i = \begin{cases} 1 & if \ \ Lp_n^i < 0 \\ 0 & if \ \ Lp_n^i > 0 \end{cases} ;$$

5)循环以上过程,直到经校验所译码为正确的,或者达到最大译码次数,则跳出循环,将译码结果传回宿主机内存。

3.根据权利要求1所述的基于OpenCL移动设备QC-LDPC的动态译码方法,其特征在于步骤2中,预先确定一个阈值,对于目前的移动设备通常阈值为几百KB,根据待译码的传输速率,当传输速率低于阈值时,动态地选择高度并行的最小和算法,能够在数据量较小时,获

得较低的时间延迟;当传输速率阈值时,选择部分并行的分层译码算法,单组译码更节省计算资源,能够在数据量较大时,获得较高的吞吐率;将对应算法的核函数编译,设置相应的参数,并在OpenCL设备上运行。

基于OpenCL移动设备QC-LDPC的动态译码方法

技术领域

[0001] 本发明涉及一种数字信号处理技术领域的译码系统，具体涉及基于OpenCL移动设备QC-LDPC动态译码方法。

背景技术

[0002] 在1962年，低密度奇偶检查码(LDPC code)即被Gallager提出[1]，并被证明其错误校正能力非常接近理论最大值，香农极限(Shannon Limit)；不过受限于当时技术，低密度奇偶检查码并无法实作。最近几年，低密度奇偶检查码被重新发现[2]，并随着集成电路的技术演进，低密度奇偶检查码的实作逐渐可行，而成为各种先进通讯系统的频道编码标准。目前，用于LDPC码译码的算法包括和积算法、最小和算法和分层译码算法等。

[0003] 准循环低密度奇偶校验码(Quasi-Cyclic LDPC，QC LDPC)是一类由小的零方阵和小的循环矩阵构成校验矩阵的LDPC码，是LDPC码的一个重要分支，也是目前LDPC码构造中一种最具实用性的结构。其结构是的其编码可以通过简单的移位寄存器实现，具有实现复杂度低的优点。除此之外，QC结构还非常适合实现部分并行的译码器结构，从而获得吞吐率和硬件复杂度方面的折中。在此之上，编码器和译码器的层内并行运算得以实现，编译码的吞吐率都得以提高。

[0004] 可编程GPU已经发展成为一种高度并行化、多线程、多核心的处理器，并且具有杰出的计算功率和极高的存储器带宽。在我们熟知的桌面平台，GPU得到了极为广泛的应用。近几年，随着并行化的发展，越来越多的移动设备硬件厂商重视对并行化标准OpenCL的支持和应用，该标准以异构平台为目标。并行计算已经在移动平台具备硬件条件和变成标准的支持，而并行化又可以带来提升设备硬件利用效率，同时GPU的低主频特性又可以在一定程度上降低功耗，因此在智能手机等移动平台实现并行计算具有巨大的潜在价值，特别在当前手机续航时间不能满足用户要求的背景下，并行化的特性显得尤为重要。目前，移动设备市面上支持OpenCL框架的GPU主要为四个公司产品：Qualcomm，Imagination Technologies，ARM，Vivante。

[0005] 经过对现有技术的文献检索发现，专利申请号201210330765.2的中国专利，专利名称为“基于GPU架构的QC-LDPC码的加速译码方法”，公开了一种基于GPU架构的QC-LDPC码的加速译码方法，主要适用于桌面端的GPU，译码的算法也较为单一，未能充分发挥软件实现译码易于动态切换的优点。

发明内容

[0006] 本发明的目的在于克服现有技术存在的上述不足，提供基于OpenCL架构的移动设备如手机上QC-LDPC码的动态译码方法，是一种准循环低密度奇偶校验码的在移动设备上的动态译码方法，具体技术方案如下。

[0007] 基于OpenCL移动设备QC-LDPC的动态译码方法，包括如下步骤：

[0008] 步骤1、作为预处理，将QC-LDPC码的校验矩阵的母矩阵信息从宿主机传入OpenCL

设备的常量存储器，并分配相应的内存空间；

[0009] 读取校验矩阵的母矩阵信息，计算码字的参数，包括母矩阵的行数hSeedCols、列数hSeedRows和扩展因子z；

[0010] 在OpenCL设备上的全局内存中分配相应的内存空间，包括用于存放未译码的码字信息和译码完成后的码字信息；

[0011] 将待处理的码字信息传入OpenCL设备，由于移动设备上的OpenCL设备的全局内存受限，当码字信息过大时，将码字信息分段，依次分别译码；

[0012] 步骤2、将待处理的码字信息传入OpenCL设备，根据待译码的数据量来动态地选择最小和算法或者分层译码算法；

[0013] 步骤3、读取相应的核函数，在OpenCL设备上完成译码算法。

[0014] 进一步地，步骤3具体包括：

[0015] (1)最小和算法的核函数在OpenCL设备上并行化实现，包括如下详细步骤：

[0016] 1)在OpenCL设备上为信息节点n后验对数似然比Lp_n和校验节点m传递给信息节点n的对数似然比Lr_{mn}分配局部内存，局部内存只能在工作组的共享；

[0017] 2)根据OpenCL设备的参数信息设置内核的工作组数，每个工作组的工作项为z*hSeedRows，这些工作项合作完成译码任务，其中每个工作项负责完成校验矩阵中同一行内的信息的更新；

[0018] 3)使用待译码字对信息节点后验对数似然比Lp_n^i，初始化，并将校验节点传递给信息节点的对数似然比Lr_{nm}^i置零，其中i代表当前迭代次数，完成初始化，

[0019] $$Lp_n^0 = \ln\frac{p_n^0(0)}{p_n^0(1)} = 2\frac{y_n}{\sigma^2} \qquad Lr_{mn}^0 = 0;$$

[0020] 其中，$p_n^0(0)$表示在译码之前，即第0次迭代时，信息节点n值为0的概率，同理，$p_n^0(1)$为此时信息节点n值为1的概率，$\frac{y_n}{\sigma^2}$为信息节点n接收信号与信道噪声之比；

[0021] 4)循环执行以下步骤：其中同一工作组内z*hSeedRows个工作项分别负责校验矩阵z*hSeedRows个行的相关信息的更新：

[0022] 4.1)信息节点传递给校验节点的对数似然比Lq_{nm}^i的更新：

[0023] $$Lq_{nm}^i = Lp_n^0 + \sum_{m'\in M(n)\backslash m} Lr_{m'n}^{i-1} \quad ;$$

[0024] 其中，Lq_{nm}^i表示第i次迭代时信息节点n传递给校验节点m的信息的对数似然比，m′∈M(n)\m表示除了校验节点m之外所有与信息节点n有连接的校验节点m′；

[0025] 4.2)校验节点传递给信息节点的对数似然比Lr_{nm}^i的更新：

[0026] $$Lr_{mn}^i = \prod_{n'\in N(m)\backslash n} sign(Lq_{n'm}^i) \min_{n'\in N(m)\backslash n}\left|Lq_{n'm}^i\right| \quad ;$$

[0027] 其中，sign函数表示符号函数，即当输入为负数时输出-1，输入为正数时输出1，输入为0时输出0；n′∈N(m)\n表示除了信息节点n之外所有与校验节点m有连接的信息节点n′；$\min_{n'\in N(m)\backslash n}\left|Lq_{n'm}^i\right|$表示在n′∈N(m)\n范围内$\left|Lq_{n'm}^i\right|$所能取得的最小值；

[0028] 4.3)信息节点后验对数似然比Lp_n^i的更新：

[0029] $$Lp_n^i = Lp_n^0 + \sum_{m\epsilon M(n)} Lr_{mn}^{i-1} ;$$

[0030] 4.4)信息节点的硬判决和校验：

[0031] $$c_n^i = \begin{cases} 1 & if \ \ Lp_n^i < 0 \\ 0 & if \ \ Lp_n^i > 0 \end{cases} ;$$

[0032] 其中，c_n^i表示第i次迭代后信息节点n经过硬判决后的值；

[0033] 5)循环以上过程，直到经校验所译码为正确的，或者达到最大译码次数，则跳出循环，将译码结果传回宿主机内存；

[0034] (2)分层译码算法的核函数在OpenCL设备上并行化实现，包括如下详细步骤：

[0035] 1)在OpenCL设备上为信息节点后验对数似然比$Lp_n^{i,k}$，局部内存只能在工作组的共享；校验节点传递给信息节点的对数似然比Lr_{nm}^i将保存在各个工作项的私有内存中；

[0036] 2)根据OpenCL设备的参数信息设置内核的工作组数，每个工作组的工作项为z，这些工作项合作完成译码任务其中，每个工作项负责完成校验矩阵中同一层内的信息的更新；

[0037] 3)使用待译码字对信息节点后验对数似然比Lp_n^i初始化，并将校验节点传递给信息节点的对数似然比Lr_{nm}^i置零，其中i代表当前迭代次数，k代表当前所在层数，完成初始化，

[0038] $$Lp_n^{1,0} = \ln \frac{p_n^0(0)}{p_n^0(1)} = 2\frac{y_n}{\sigma^2} \qquad Lr_{nm}^{0,k} = 0 ;$$

[0039] 4)循环执行以下步骤：其中同一工作组内z个工作项分别负责校验矩阵一个层内的z个行的相关信息的更新：

[0040] 4.4)信息节点传递给校验节点的对数似然比Lq_{nm}^i的更新：

[0041] $$Lq_{nm}^{i,k} = Lp_n^{i,k-1} - Lr_{mn}^{i-1,k} ;$$

[0042] 其中，k表示当前的层数，$Lq_{nm}^{i,k}$表示在第i次迭代中，第k层中信息节点n传递给校验节点m的信息的对数似然比；$Lr_{mn}^{i-1,k}$表示在第i次迭代中，第k层中校验节点m传递给信息节点n的信息的对数似然比；

[0043] 4.2)校验节点传递给信息节点的对数似然比Lr_{nm}^i的更新：

[0044] $$Lr_{mn}^{i,k} = \prod_{n\prime\epsilon N(m)\backslash n} sign(Lq_{n\prime m}^{i,k}) \min_{n\prime\epsilon N(m)\backslash n} \left|Lq_{n\prime m}^{i,k}\right| ;$$

[0045] 4.5)信息节点后验对数似然比Lp_n^i的更新：

[0046] $$Lp_n^{i,k} = Lq_{nm}^{i,k} + Lr_{mn}^{i,k} ;$$

[0047] 4.6)仅仅在完成最后一层的Lp_n^i的更新后，才进行信息节点的硬判决和校验：

[0048] $$c_n^i = \begin{cases} 1 & if\ Lp_n^i < 0 \\ 0 & if\ Lp_n^i > 0 \end{cases};$$

[0049] 5)循环以上过程，直到经校验所译码为正确的，或者达到最大译码次数，则跳出循环，将译码结果传回宿主机内存。

[0050] 进一步地，步骤2中，预先确定一个阈值，对于目前的移动设备通常阈值为几百KB，根据待译码的传输速率，当传输速率低于阈值时，动态地选择高度并行的最小和算法，能够在数据量较小时，获得较低的时间延迟；当传输速率阈值时，选择部分并行的分层译码算法，单组译码更节省计算资源，能够在数据量较大时，获得较高的吞吐率。将对应算法的核函数编译，设置相应的参数，并在OpenCL设备上运行。

[0051] 与现有技术相比，本发明具有如下优点和技术效果：

[0052] 本发明能将CPU作为控制器，将码字的接收信息放入到全局存储器中，并在完成所有初始化过程后，发起运行核函数的命令；合理地配置OpenCL的各项参数，在每个并行工作组中实现一组译码。通过同一工作组中不同工作项的合作完成最小和算法和分层译码算法。与硬件实现的译码方法不同，基于OpenCL的软件实现可以根据当前的传输速率动态选择译码算法，在传输速率低的时候选择具有低延迟的最小和算法，在传输速率高的时候选择具有高吞吐率的分层译码算法，算法切换可以方便地通过OpenCL核函数的切换实现。本发明针对QC结构LDPC码字的特点，实现了不同码率和码长的QC-LDPC码在不同OpenCL设备上通用的加速译码；并提供了LDPC译码器在OpenCL框架中的并行化实现。能在OpenCL上独立并行地运行很多个LDPC译码器，译码效率大幅度提高。LDPC本身用于4G信号传输，手机移动端实现高效的LDPC译码算法，可以取消手机用来通信的基带芯片中解码的功能，可以节省成本，并且容易软件升级，容易实现多种制式的全网通功能。

附图说明

[0053] 图1是本发明中将准循环LDPC码的母矩阵进行扩展的结构示意图。

[0054] 图2是本发明中所使用到的OpenCL平台模型示意图。

[0055] 图3是本发明中所使用到的OpenCL内存结构示意图。

[0056] 图4是本发明中所使用到的OpenCL编程模型示意图。

具体实施方式

[0057] 下面结合具体实施例对本发明进行详细说明。以下实施例将有助于本领域的技术人员进一步理解本发明，但不以任何形式限制本发明。应当指出的是，对本领域的普通技术人员来说，在不脱离本发明构思的前提下，还可以做出若干变形和改进，这些都属于本发明的保护范围。

[0058] 如图1所示，是使用基于IEEE 802.16e标准的LDPC校验矩阵的母矩阵，以码率为5/6的码字为例，选取扩展因子z为24，即该校验矩阵为96×576。

[0059] 如图2所示，本实验中所使用的OpenCL的平台模型，本实验使用型号为PowerVR G6200的GPU作为OpenCL设备，MTK的Helio X10CPU作为宿主机进行测试。

[0060] 如图3所示，一个工作组占用OpenCL设备上一个计算单元，一个工作项占用计算单

元内的一个处理单元。初始化时，主要在全局内存分配相应空间。CPU作为宿主机，首先对信源数据进行LDPC编码，并对编码后的比特加上AWGN噪声，之后作为待译码数据放入OpenCL设备内存的全局内存里。并根据传输速率执行相应的核函数。

[0061] 如图4所示，具体的，将并行运行1000个工作组，同一工作组中各个工作项合作完成译码。

[0062] 当使用最小和算法译码时，过程如下：设置每个工作组包含96个工作项，每个工作项分别完成校验矩阵中的一行内的信息节点传递给校验节点的对数似然比Lq_{nm}的更新，此时并不需要保存完整的Lq_{nm}，仅保存Lq_{nm}的最小值、次小值以及最小值位置的索引，并用于校验节点传递给信息节点的对数似然比Lr_{mn}的更新，之后是信息节点后验对数似然比Lp_n的更新，以此视为完成一次迭代过程。循环执行以上迭代过程，直到经过校验译码正确，或者达到最大译码次数。

[0063] 当使用分层译码算法译码时，过程如下：设置每个工作组包含24个工作项，此时总共有4个层，从第一层开始，每个工作项分别完成校验矩阵中同一层内的各行的信息节点传递给校验节点的对数似然比Lq_{nm}的更新，此时并不需要保存完整的Lq_{nm}，仅保存Lq_{nm}的最小值、次小值以及最小值位置的索引，并用于校验节点传递给信息节点的对数似然比Lr_{mn}的更新，以上视为完成一层的更新，之后从第二层开始往下更新，直到最后一层更新完成。在对全部层完成一次更新后，之后是信息节点后验对数似然比Lp_n^i的更新，以此视为完成一次完整的迭代过程。循环执行以上迭代过程，直到经过校验译码正确，或者达到最大译码次数。

[0064] 最后各个工作项负责将对应的数据组写回到全局内存，完成译码。

0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0

1	25	55	-1	47	4	-1	91	84	8	86	52	82	33	5	0	36	20	4	77	80	0	-1	-1
-1	6	-1	36	40	47	12	79	47	-1	41	21	12	71	14	72	0	44	49	0	0	0	0	-1
51	81	83	4	67	-1	21	-1	31	24	91	61	81	9	86	78	60	88	67	15	-1	-1	0	0
50	-1	50	15	-1	36	13	10	11	20	53	90	29	92	57	30	84	92	11	66	80	-1	-1	0

图1

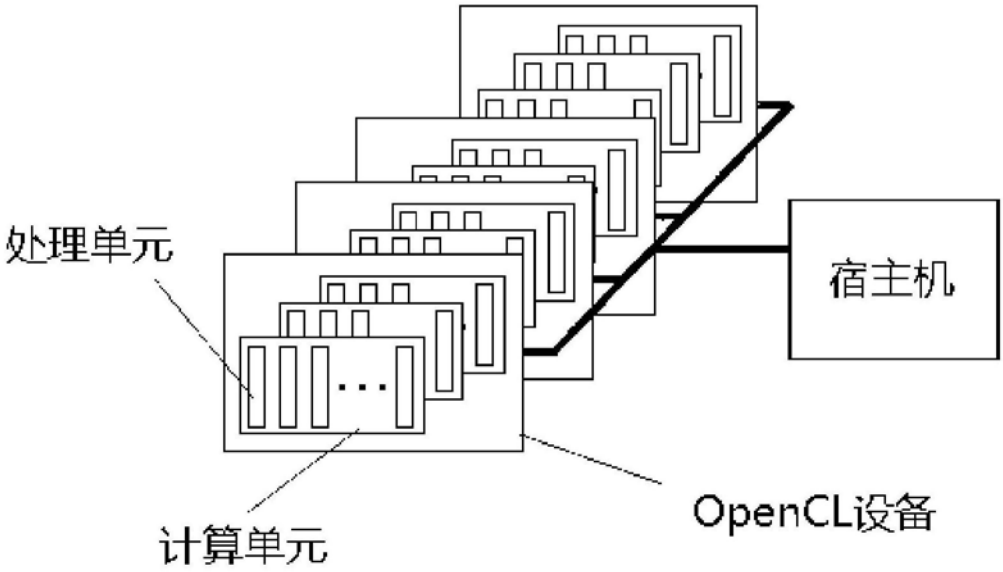


图2

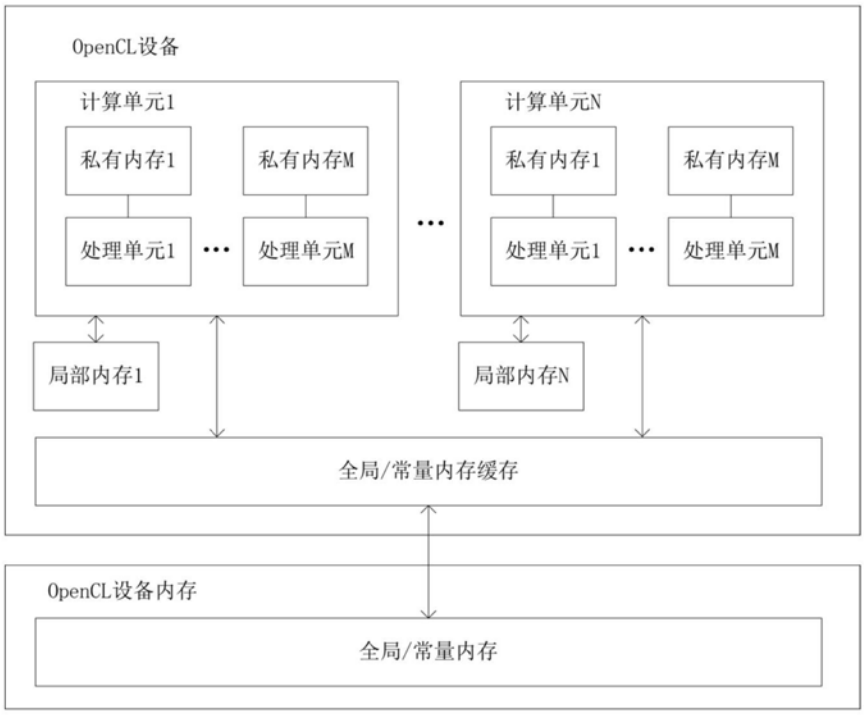

OpenCL设备
计算单元1
私有内存1
私有内存M
处理单元1
处理单元M
计算单元N
私有内存1
私有内存M
处理单元1
处理单元M
局部内存1
局部内存N
全局/常量内存缓存
OpenCL设备内存
全局/常量内存

图3

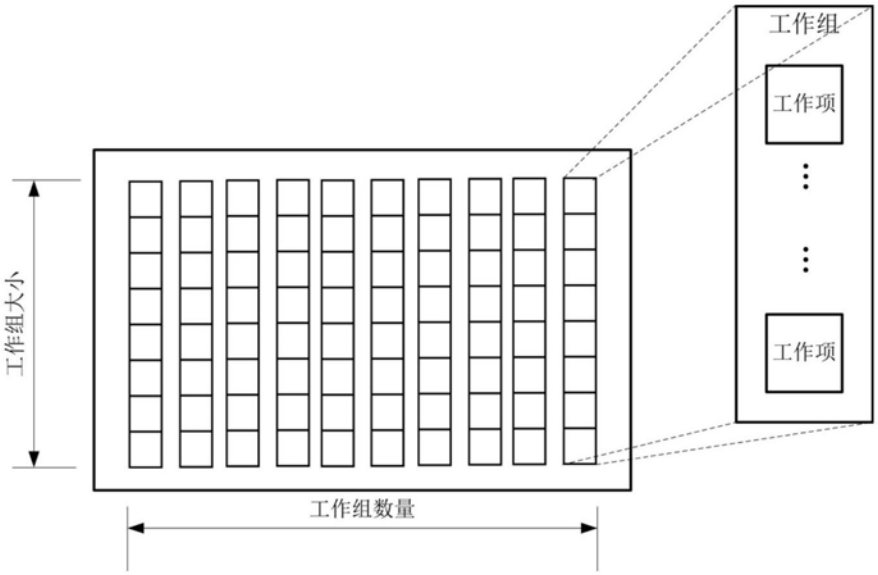

工作组
工作项
工作项
工作组大小
工作组数量

图4