(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 117724833 B
(45)授权公告日 2024.05.28

(21)申请号 202311673885.7

(22)申请日 2023.12.06

(65)同一申请的已公布的文献号
申请公布号 CN 117724833 A

(43)申请公布日 2024.03.19

(73)专利权人 无锡众星微系统技术有限公司
地址 214000 江苏省无锡市新吴区菱湖大道111号软件园天鹅座C座6层

(72)发明人 顾大晔 郭二辉 黄小菲 金俊浩 丁诗通 董树林 武卫红 钟世鹏 王烽

(74)专利代理机构 北京动力号知识产权代理有限公司 11775
专利代理师 杨润

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 13/40(2006.01)

(56)对比文件
CN 105681222 A,2016.06.15
CN 112131155 A,2020.12.25
CN 114138502 A,2022.03.04
CN 114553776 A,2022.05.27
CN 114741341 A,2022.07.12
CN 114860785 A,2022.08.05
CN 115002052 A,2022.09.02
CN 116821011 A,2023.09.29
EP 3999972 A1,2022.05.25
US 2021191787 A1,2021.06.24
WO 2021014114 A1,2021.01.28
WO 2021202175 A1,2021.10.07

李少博 .《基于Tag的PCIe总线事物并行处理技术研究》.《中国优秀硕士学位论文全文数据库》.2017,全文.

Dongyang Li 等.《A Parallel and Pipelined Architecture for Accelerating Fingerprint Computation in High Throughput Data Storages》.《IEEE》.2015,全文.

审查员 谢丹

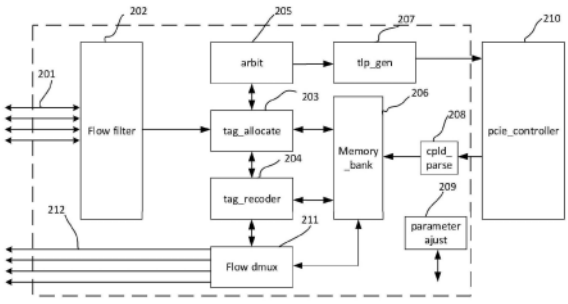
权利要求书2页 说明书8页 附图9页

(54)发明名称

一种基于流属性的PCIe tag缓存自适应资源分配方法和装置

(57)摘要

本发明提供了一种基于流属性的PCIe tag缓存自适应资源分配方法和装置,将读请求分成多种数据流类型。通过流表映射确定读请求的数据流类型,根据数据流类型为读请求从不同tag资源分组分配tag资源,构建对应上下文缓存;在片上缓存中为tag资源分配返回数据缓存,根据分配结果更新上下文缓存,仲裁多个已分配tag资源的读请求,形成读请求包发送到PCIe链路;接收返回包,解析得到对应tag,将返回包的数据负载存储到tag对应返回数据缓存。根据运行实时状态更新tag资源分组、缓存粒度和映射流表。本发明实现了tag自适应分组和共享以及缓存的自适应粒度分配和共享,提高片上缓存利用率,减小芯片面积,满足不同数据流性能需求。



CN 117724833 B

1.一种基于流属性的PCIe tag缓存自适应资源分配方法,其特征在于,包括:

通过流表映射确定来自多个功能实体的PCIe读请求的数据流类型,根据所述数据流类型从对应的tag资源分组中为所述读请求分配tag资源,并构建对应的上下文缓存;

在片上缓存中为所述tag资源分配返回数据缓存,并根据分配结果更新所述上下文缓存,对所述多个已经分配tag资源的读请求进行仲裁,并形成读请求数据包发送到PCIe链路,其中所述返回数据缓存的空间粒度基于各个数据流类型的缓存利用率来实时更新;

从所述PCIe链路接收所述读请求的返回包,解析得到返回包对应的tag,将所述返回包中的数据负载存储到与所述tag对应的返回数据缓存,并返回到对应的功能实体;

所述根据分配结果更新所述上下文缓存,进一步包括:

在上下文缓存的每个表项中记录原始请求的标识、当前tag是否为读请求ID下最早的tag、当前tag对应的返回数据与先前未完成tag的依赖关系、当前tag是否为读请求的最后一个tag、当前tag是否已经分配但并没有发送到PCIe网络、当前tag对应的请求是否已经发出并正在等待返回数据、当前tag对应的响应是否返回超时,以及当前tag的数据是否都已经返回。

2.根据权利要求1所述的基于流属性的PCIe tag缓存自适应资源分配方法,其特征在于,所述通过流表映射确定所述读请求的数据流类型,进一步包括:

将来自不同功能实体或不同流的读请求ID输入流表中,确定读请求的数据流类型,所述数据流类型包括低延时访问、普通访问、高带宽访问和零负载访问,所述流表由软件通过不同读请求ID的流类型来配置,或者由硬件跟踪不同ID读请求的特征信息进行动态更新。

3.根据权利要求1所述的基于流属性的PCIe tag缓存自适应资源分配方法,其特征在于,根据所述数据流类型从对应的tag资源分组中为所述读请求分配tag资源,进一步包括:

针对不同的数据流类型,利用预设配置参数将所有tag资源进行分组,在各个分组之间设置共享区域,在所述数据流类型的对应分组中为所述读请求分配对应的tag资源,当组内的tag资源耗尽时,临时分配所述共享区域的tag资源,并根据tag资源使用情况动态更新分组边界。

4.根据权利要求1所述的基于流属性的PCIe tag缓存自适应资源分配方法,其特征在于,在所述为所述tag资源分配返回数据缓存之前,该方法还包括:

通过将所有处于outstanding状态的tag对应的读长度之和除以outstanding状态的tag数量,乘以当前缓存分组的缓存粒度来计算所述缓存利用率,并根据缓存利用率动态调整缓存粒度。

5.一种基于流属性的PCIe tag缓存自适应资源分配装置,其特征在于,包括:

tag分配单元,用于通过流表映射确定来自多个功能实体的PCIe读请求的数据流类型,根据所述数据流类型从对应的tag资源分组中为所述读请求分配tag资源,并构建对应的上下文缓存;

缓存分配单元,用于在片上缓存中为所述tag资源分配返回数据缓存,并根据分配结果更新所述上下文缓存,对所述多个已经分配tag资源的读请求进行仲裁,并形成读请求数据包发送到PCIe链路,其中所述返回数据缓存的空间粒度基于各个数据流类型的缓存利用率来实时更新;

接收单元,用于从所述PCIe链路接收所述读请求的返回包,解析得到返回包对应的

tag,将所述返回包中的数据负载存储到与所述tag对应的返回数据缓存,并返回到对应的功能实体;

所述缓存分配单元,进一步用于:

在上下文缓存的每个表项中记录原始请求的标识、当前tag是否为读请求ID下最早的tag、当前tag对应的返回数据与先前未完成tag的依赖关系、当前tag是否为读请求的最后一个tag、当前tag是否已经分配但并没有发送到PCIe网络、当前tag对应的请求是否已经发出并正在等待返回数据、当前tag对应的响应是否返回超时,以及当前tag的数据是否都已经返回。

6.根据权利要求5所述的基于流属性的PCIe tag缓存自适应资源分配装置,其特征在于,所述tag分配单元,进一步用于:

将来自不同功能实体或不同流的读请求ID输入流表中,确定读请求的数据流类型,所述数据流类型包括低延时访问、普通访问、高带宽访问和零负载访问,所述流表由软件通过不同读请求ID的流类型来配置,或者由硬件跟踪不同ID读请求的特征信息进行动态更新。

7.根据权利要求5所述的基于流属性的PCIe tag缓存自适应资源分配装置,其特征在于,所述tag分配单元,进一步用于:

针对不同的数据流类型,利用预设配置参数将所有tag资源进行分组,在各个分组之间设置共享区域,在所述数据流类型的对应分组中为所述读请求分配对应的tag资源,当组内的tag资源耗尽时,临时分配所述共享区域的tag资源,并根据tag资源使用情况动态更新分组边界。

8.根据权利要求5所述的基于流属性的PCIe tag缓存自适应资源分配装置,其特征在于,所述缓存分配单元,进一步用于:

在所述为所述tag资源分配返回数据缓存之前,通过将所有处于outstanding状态的tag对应的读长度之和除以outstanding状态的tag数量,乘以当前缓存分组的缓存粒度来计算所述缓存利用率,并根据缓存利用率动态调整缓存粒度。

一种基于流属性的PCIe tag缓存自适应资源分配方法和装置

技术领域

[0001] 本发明属于总线设计领域,特别涉及一种基于流属性的PCIe tag缓存自适应资源分配方法和装置。

背景技术

[0002] PCI-Express(peripheral component interconnect express)是一种高速串行计算机扩展总线标准,广泛用于计算机各组件之间的互连。在PCIe总线事务处理过程中,tag字段和Requester ID字段一起构成transaction ID。其中tag作为传输过程中请求包的唯一标识。PCIe协议规范规定,对于非转发型事务(NP),所有已发出去且未处理完成的事务,都必须有唯一的transaction ID标记。

[0003] 随着PCIe链路速率越来越快,为了充分地利用链路带宽,要求设备能够支持的outstanding请求(描述链路中请求包已经发出,但应答数据包未返回的一种状态)个数越来越多,对应的缓存空间的需求也越来越大。目前的PCIe设备往往都是多功能实体(function),具有独立的配置空间,每个function实现不同类型的主机访问。为了适应虚拟化、多类型访问下的带宽利用率以及合理的芯片性能(访问延时、面积…)的需求,在PCIe类芯片的电路设计方面,对tag的分配、缓存配置等技术提出了更高的要求。

[0004] 然而传统的专利和方案大多使用静态分配方式,或按照不同的流类型、或按照function将tag资源按组分配。对于为CPLD预留的缓存,每个tag对应的缓存按固定粒度(或者按照读完成边界RCB为粒度,或者按照最大载荷大小MPS为粒度)进行分配,无法区分不同读请求的特征,以及对缓存进行更细粒度管理,在链路带宽的不断提高,并且流量类型越来越复杂的情况下,这种分配方式是非常低效的,同时会造成缓存资源的浪费。

发明内容

[0005] 本发明的目的在于提供一种基于流属性的PCIe tag缓存自适应资源分配方法和装置,旨在提高tag资源和片上缓存的利用率。

[0006] 根据本发明的第一方面,提供了一种基于流属性的PCIe tag缓存自适应资源分配方法,包括:

[0007] 通过流表映射确定来自多个功能实体的PCIe读请求的数据流类型,根据所述数据流类型从对应的tag资源分组中为所述读请求分配tag资源,并构建对应的上下文缓存;

[0008] 在片上缓存中为所述tag资源分配返回数据缓存,并根据分配结果更新所述上下文缓存,对所述多个已经分配tag资源的读请求进行仲裁,并形成读请求数据包发送到PCIe链路,其中所述返回数据缓存的空间粒度基于各个数据流类型的缓存利用率来实时更新;

[0009] 从所述PCIe链路接收所述读请求的返回包,解析得到返回包对应的tag,将所述返回包中的数据负载存储到与所述tag对应的返回数据缓存,并返回到对应的功能实体。

[0010] 优选地,所述通过流表映射确定所述读请求的数据流类型,进一步包括:

[0011] 将来自不同功能实体或不同流的读请求ID输入流表中,确定读请求的数据流类

型,所述数据流类型包括低延时访问、普通访问、高带宽访问和零负载访问,所述流表由软件通过不同读请求ID的流类型来配置,或者由硬件跟踪不同ID读请求的特征信息进行动态更新。

[0012] 优选地,所述根据所述数据流类型从对应的tag资源分组中为所述读请求分配tag资源,进一步包括:

[0013] 针对不同的数据流类型,利用预设配置参数将所有tag资源进行分组,在各个分组之间设置共享区域,在所述数据流类型的对应分组中为所述读请求分配对应的tag资源,当组内的tag资源耗尽时,临时分配所述共享区域的tag资源,并根据tag资源使用情况动态更新分组边界。

[0014] 优选地,所述根据分配结果更新所述上下文缓存,进一步包括:

[0015] 在上下文缓存的每个表项中记录原始请求的标识、当前tag是否为该ID下最早的tag、当前tag对应的返回数据与先前未完成tag的依赖关系、当前tag是否为读请求的最后一个tag、当前tag是否已经分配但并没有发送到PCIe网络、当前tag对应的请求是否已经发出并正在等待返回数据、当前tag对应的响应是否返回超时,以及当前tag的数据是否都已经返回。

[0016] 优选地,在所述为所述tag资源分配返回数据缓存之前,该方法还包括:

[0017] 通过将所有处于outstanding状态的tag对应的读长度之和除以outstanding状态的tag数量,乘以当前缓存分组的缓存粒度来计算所述缓存利用率,并根据缓存利用率动态调整缓存粒度。

[0018] 根据本发明的第二方面,提供了一种基于流属性的PCIe tag资源分配装置,包括:

[0019] tag分配单元,用于获取来自多个功能实体的PCIe读请求,通过流表映射确定所述读请求的数据流类型,根据所述数据流类型从对应的tag资源分组中为所述读请求分配tag资源,并构建对应的上下文缓存;

[0020] 缓存分配单元,用于在片上缓存中为所述tag资源分配返回数据缓存,并根据分配结果更新所述上下文缓存,对所述已经分配tag资源的读请求进行仲裁,并形成读请求数据包发送到PCIe链路,其中所述返回数据缓存的空间粒度基于各个数据流类型的缓存利用率来确定;

[0021] 接收单元,用于从所述PCIe链路接收所述读请求的返回包,解析得到返回包对应的tag,将所述返回包中的数据负载存储到与所述tag对应的返回数据缓存,并返回到对应的功能实体。

[0022] 相比于现有技术,本发明的技术方案具备以下优点:

[0023] 通过读数据访问请求的分类管理更好地满足不同流的性能需求。将tag资源按照不同的数据流类型分组管理,使得不同流的tag分配可以并行处理,增加吞吐量。tag资源在不同类型数据流之间可以实时共享,防止不同数据流间由于访问突发导致的阻塞,提高不同数据流的并发性能。在对数据流没有先验知识的情况下,通过自适应更新迭代满足不同类型访问的性能需求,既可以满足低延时数据流对链路延时较高的需求,又可以满足高带宽的访问突发需求,为主机侧的IO操作提供更大的优化空间,流分类方法便于tag分组和缓存管理。依据不同的数据流类型和tag分组对缓存进行分组,不同的分组按照不同的粒度管理,同时不同分组的缓存粒度可以独立实时更新,极大的提高了片上缓存的利用率。按照

tag号可以唯一索引内部的tag上下文,缓存地址等信息,使得对tag上下文以及对应缓存地址的索引效率非常高。该方法不限于硬件实现,也可以配合软件实现,实现方式灵活,既可以作为芯片设计的实现,同样也可以用于芯片建模下的性能收敛仿真。

[0024] 本发明的其它特征和优点将在随后的说明书中阐述,并且部分地从说明书中变得显而易见,或者通过实施本发明而了解。本发明的目的和其他优点可以通过在说明书、权利要求书以及附图中所指出的结构和流程来实现和获取。

附图说明

[0025] 为了更清楚地说明本发明实施例或现有技术中的技术方案,下面将对实施例或现有技术描述中所需要使用的附图作简单介绍,显而易见的是,下面描述中的附图是本发明的某些实施例,对于本领域普通技术人员而言,在不付出创造性劳动的前提下,还可以根据这些附图获取其他的附图。

[0026] 图1是根据本发明的典型的PCIe多function终端设备的结构图。

[0027] 图2是根据本发明的基于流属性的PCIe tag资源分配方法的总体流程图。

[0028] 图3是实现根据本发明方法的读通道的电路结构图。

[0029] 图4是根据本发明的基于function ID/flow ID索引数据流类型的流表结构图。

[0030] 图5是根据本发明的基于数据流类型的tag分组示意图。

[0031] 图6是根据本发明的不同分组tag和片上缓存之间的映射关系示意图。

[0032] 图7是根据本发明的tag上下文的数据结构示意图。

[0033] 图8是根据本发明的tag分组和缓存细粒度映射示意图。

[0034] 图9是根据本发明的基于tag分组边界的实时更新流程图。

[0035] 图10是根据本发明的基于缓存使用率的缓存管理粒度大小的实时更新流程。

[0036] 图11是根据本发明的基于function ID/flow ID数据流类型的实时更新流程。

具体实施方式

[0037] 为了使本发明实施例的目的、技术方案和优点更加清楚,下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清楚、完整地说明,显然,所描述的实施例是本发明一部分实施例,而不是全部的实施例。基于本发明中的实施例,本领域普通技术人员在没有做出创造性劳动前提下所获取的所有其他实施例,都属于本发明保护的范围。

[0038] 基于以上分析,本发明提出一种基于流属性的PCIe tag缓存自适应资源分配方法和装置,主要适用于PCIe协议接口设备,根据流属性将tag按组动态分配,确定PCIe功能节点的读通道的流的分类方式,将tag资源按照不同的数据流类型进行分组,然后根据不同的tag分组,分别维护tag上下文,为不同类型的tag分配不同的数据缓存大小,不同分组对应的缓存按不同粒度管理,并且这些粒度是动态可调整的,以实现tag和缓存更高的利用率。

[0039] 一个PCIe设备往往包含多个功能实体(function),每个function都有对主机(主机内存/cache)或其他节点的读数据请求,这些读请求通常包括对大批量数据的访问、对控制数据的访问、为了写确认的0长度数据读请求和对一般数据结构的访问。不同的访问类型对访问延时和数据突发要求不同,因此在本发明具体的实施例中,将读请求细分为低延时访问、普通访问、高带宽访问、零负载访问四种类型。这四种类型的特点各有不同。低延时访

问的特点是要求低延时,每次请求数据长度小。高带宽访问往往包含对一段地址空间的连续访问,一个请求往往要跨多个tag,消耗多块缓存空间。零负载访问的特点是写后读,保证之前的写请求成功写入到主存,不需要返回数据。普通访问的特点介于低延时访问和高带宽访问之间,消耗的缓存空间比低延时访问大,但一般不会跨tag。

[0040] 为了更好地使用tag资源,本发明根据上述流的属性,将整个tag资源分成4组。不同分组之间的边界不是固定的,根据运行时状态自动调整各组窗口大小。同时窗口之间存在共享区域,在tag分配时作为共享资源分配。

[0041] 根据上述tag分组的属性,将数据缓存资源分成了三组,其中零负载访问对应的tag不需要缓存。为了细粒度地利用缓存资源,对于不同分组,tag对应的缓存块大小是不同的,同时基于tag-ram利用率,实时更新tag对应缓存块的大小,同时在突发情况下,低延时访问和高带宽访问可以共享普通流对应的缓存。

[0042] 将来自不同function的读访问,按照function ID和流ID(flow ID)与4种数据流访问类型进行映射,形成流分类表。该流分类表可以根据访问状态实时的更新。根据上述流的属性,按照一定的规则将不同的请求通过调度发送到PCIe链路。调度规则是保证高带宽流的多个请求连续、低延时流尽量不阻塞。利用tag上下文记录当前tag的状态,根据tag状态,将返回的数据按照目的function ID和flow ID返回给对应的function。

[0043] 本发明提出的方案通过合理的tag分配,有效解决不同数据流之间的阻塞,在不同数据流之间对低延时和高带宽进行优化平衡,同时通过细粒度的缓存管理,提升了缓存利用率,减少了实际的缓存大小,从而最终减小了芯片面积。

[0044] 在图1所示的PCIe设备中,相互独立的功能实体101～104称作function。数据通路上包含了三个独立的数据通道,其中105表示为function读Host的数据通道,称作读通道,106表示为function写host的数据通道,称作写通道,以及107表示为host访问function的访问通道。108表示实现PCIe传输层到高速串行接口的控制器,用于将应用层的读请求协议转换成传输层报文tlp,发送到PCIe互联网络109。功能实体101～104向PCIe链路发起不同的读请求,而每一个读请求需要来自链路的应答,因此需要对这些请求进行合理的tag标记、缓存分配以及优先级仲裁等操作,以适应多请求的负载均衡、低延时以及缓存资源的合理分配。本发明描述的功能电路是在图1的读通道105中实现的。

[0045] 参见图2的流程图,本发明提供的所述基于流属性的PCIe tag缓存自适应资源分配方法包括:

[0046] 步骤101:通过流表映射确定来自多个功能实体的PCIe读请求的数据流类型,根据所述数据流类型从对应的tag资源分组中为所述读请求分配tag资源,并构建对应的上下文缓存。

[0047] 图3示出了图1中所示的读通道105的一种实现结构。来自各个功能实体function的不同类型的读请求201经过流分类器202的流表映射后,映射成不同的数据流类型。经过流表映射后的读请求根据数据流类型,由tag分配器203进行tag分配,将分配的结果形成上下文,更新到tag上下文缓存204中。

[0048] 图4描述了一种基于function ID/flow ID索引的查表结构。该表在图3中的流分类器202中维护。来自不同function、不同flow的请求首先通过该表得到当前请求属于哪一种数据流类型。数据流类型可以是上文提到的4种流类型。该表可以静态配置也可以动态配

置。在静态配置下,由软件不同读请求ID的流类型,静态配置该表。在动态配置下,可以由硬件跟踪不同ID读请求的特征信息进行动态更新。

[0049] 图5描述了按照数据流类型进行的tag分组。tag分组的配置参数401使用4组整数参数将整个tag资源分成4组:低延迟访问tag、普通访问tag、高带宽访问tag、零负载tag,形成tag分组的线性排布结果402,不同tag分组之间没有交叠。组间的边界区域即为共享区域,当组内的tag资源耗尽,可以临时分配共享区域的tag。在工作过程中,可实时跟踪不同数据流类型的tag消耗情况,根据消耗情况,实时更新不同分组的边界。

[0050] 步骤102:在片上缓存中为所述tag资源分配返回数据缓存,并根据分配结果更新所述上下文缓存,对所述多个已经分配tag资源的读请求进行仲裁,并形成读请求数据包发送到PCIe链路,其中所述返回数据缓存的空间粒度基于各个数据流类型的缓存利用率来实时更新。

[0051] 参见图3,在完成tag分配后,通过缓存管理模块206为所分配的tag分配对应的返回数据缓存,分配结果形成上下文,同步更新到所述上下文缓存204。tag分配器203根据上下文缓存204维护的不同类型流的上下文,对多个已经完成tag分配的读请求发送到仲裁模块205进行仲裁,仲裁模块205根据预设规则仲裁多路请求,并将仲裁结果形成上下文以更新上下文缓存204,并将组包的相关信息(包含tag、读地址、读数据长度等信息,读地址和读数据长度等信息同样在上下文缓存中缓存,在此不做详细描述),发送到包发生器模块207。包发生器模块207根据接收到的信息形成读请求数据包,通过PCIe协议控制器210发送到PCIe链路。

[0052] 图6描述了tag和返回数据缓存的映射关系。其中501表示低延迟访问tag分组,502表示普通访问tag分组,503表示高带宽访问tag分组,504表示零负载tag分组。505标识低延迟访问tag分组对应的数据缓存,506表示普通访问tag分组对应的数据缓存,507表示高带宽访问tag分组对应的数据缓存,而零负载tag分组不需要缓存。508描述了对应数据缓存块的最后一个有效字节在缓存块中的偏移。每个tag和数据缓存的独立区域相对应。不同分组的缓存块大小不同,其中缓存块505粒度最小,缓存粒度506大小适中,缓存块507粒度最大。在上述tag-buffer的映射结构中,使用tag即可以唯一标记一块内存区域。

[0053] 如图7所示描述了tag对应的上下文数据结构。其中低延迟访问tag分组601、普通访问tag分组602、高带宽访问tag分组603和零负载tag分组604中的每个分组中的每个tag对应一个上下文表项。其中605为一个上下文表项的描述,表项中的function ID/flow ID即原始请求的标识。由于同一function ID/flow ID内的读返回数据需要保序,使用pre tag和next tag跟踪当前tag是否为该ID下最早的tag。order字段指示当前tag对应的返回数据与先前未完成tag具有的依赖关系,当order为0时,该tag对应的读请求不需要保序,当该tag对应的数据返回后,可以立即将数据返回到对应的function接口。由于在highband流中,可能出现一个读请求跨多个tag的情况出现,last字段标识当前tag是否为读请求的最后一个tag。req字段标识当前tag已经分配,但并没有发送到PCIe网络;wait字段标识当前tag对应的请求已经发出并正在等待返回数据:timeout字段标识该tag对应的响应返回超时,会触发超时回收等操作。complete字段标识当前tag的数据是否都已经返回。

[0054] 对于4种数据流类型发出的访问请求,其仲裁策略可以采用基于优先级加权的仲裁策略,优先级从高到低依次为低延迟访问low_latency、零负载访问zero load、普通访问

normal、高带宽访问high band。在高带宽类型的请求中,其读取长度可能跨多个tag,当这种请求仲裁成功时,必须保证最后一个tag对应的读请求发送完成后,仲裁模块才能接收新的请求。

[0055] 步骤103:从所述PCIe链路接收所述读请求的返回包,解析得到返回包对应的tag,将所述返回包中的数据负载存储到与所述tag对应的返回数据缓存,并返回到对应的功能实体。

[0056] PCIe协议控制器210接收到返回的CPL/CPLD(PCIe总线协议中定义的一种返回包类型,包含读请求返回的读状态和读数据),返回到包解析模块208进行包的合法性检查和tag匹配。针对经过匹配的CPL/CPLD,提取所包含的数据和返回信息写入缓存管理模块206对应的tag缓存,并更新上下文缓存204的tag上下文信息,当对应读请求的所有数据都返回后,通过流分选模块211按照原始的流ID分选到不同的function数据返回接口212,向function返回读数据。

[0057] 图8描述了在工作过程中,各数据结构的一种运行时状态。其中704表示低延迟访问tag分组,701表示704对应的缓存分组,其中一个tag对应的数据缓存块的大小是32B。705表示普通访问tag分组,702表示705对应的数据缓存分组,其中一个tag对应的数据缓存块的大小是64B。706表示高带宽访问tag分组,703表示706对应的数据缓存分组,其中一个tag对应的数据缓存块的大小是128B。704、705、706、707这四种tag分组之间存在共享区域,可以在运行时跟踪统计各分组的tag空置率,实时更新tag分组表。三个数据缓存分组701、702、703之间存在类似tag分组的线性排布,根据tag分组的实时变化,对数据缓存分组701、702、703进行实时调整。根据tag-ram利用率,调整各分组中的缓存粒度。708描述了已经分配的tag对应的上下文信息。图8给出的所有数据结构都可以使用tag作为索引,快速查找。

[0058] 图9描述了tag分组的边界更新的一种方式。如图9所示,对于当前的tag分组,维护两个计数器low_cnt和high_cnt。当有新的读请求需要分配tag时,如果加上了当前请求后使用的tag数量已经超过了tag分组内的tag总数,则low_cnt减1,high_cnt加1;当有新的读请求需要分配tag时,加上当前请求后使用的tag数量没有超过tag分组内的tag数量,则low_cnt加1,high_cnt减1。当high_cnt累积到一定阈值后,tag边界向共享区域扩展1个,并清除high_cnt。当low_cnt累积到一定阈值后,tag边界向自身区域方向回退1个。图9根据tag分组动态利用率简单地调整边界。关于计数和阈值的选取,可以根据具体需求设定。

[0059] 图10描述了tag对应的缓存分组的缓存粒度的大小调整流程。如图10所示,每次读请求的分配缓存空间都会触发当前缓存分组内的缓存利用率的计算。在一个实施例中,缓存利用率的计算方式是所有处于outstanding状态的tag对应的读长度之和除以outstanding状态的tag数量乘以当前缓存分组的缓存粒度。当缓存利用率高于上限时,hig_cnt累加,当high_cnt达到预定义阈值,将片上缓存的缓存粒度向上按预设步长调整;当片上缓存利用率低于下限时,low_cnt累加,当low_cnt低于预定义阈值,将片上缓存的缓存粒度向下按预设步长调整。关于计数和预定义阈值的选取,可以根据具体需求设定。

[0060] 图11描述了function ID/flow ID对应的数据流类型的更新流程。当对应的function ID/flow ID出现新的读请求,如果请求的size大小大于当前flow type对应的mem_size大小,则low_cnt减1,high_cnt加1;如果请求的size大小小于当前flow type对应的mem_size大小,则low_cnt加1,high_cnt减1。如果low_cnt达到预定义阈值,则将当前

function ID/flow ID对应的数据流类型按图示方向迁移。具体可以将普通访问normal迁移到低延迟访问low_latency,或者将高带宽访问high band迁移到普通访问normal。当high_cnt达到预定义阈值时,则将当前function ID/flow ID对应的数据流类型按图示方向迁移,具体可以将低延迟访问low_latency迁移到普通访问normal,或者将普通访问normal迁移到高带宽访问high band。

[0061] 需要说明的是,图9-图11的更新流程可以通过各自增加开关的方式,独立控制。

[0062] 可见,本发明提出的基于流属性的PCIe tag缓存资源分配方法,相比于现有技术具备以下优点:

[0063] 将设备对PCIe节点的读数据访问请求按属性分成不同的流,通过分类管理更好地满足不同流的性能需求。将tag资源按照不同的数据流类型分组管理,使得不同流的tag分配可以并行处理,增加吞吐量。不同tag分组边界之间设置缓冲区,分组边界实时更新,tag资源在不同类型数据流之间可以实时共享,防止不同数据流间由于访问突发导致的阻塞。根据不同的数据流类型和tag分组对缓存进行分组,不同的分组按照不同的粒度管理,这样可以提高缓存的利用率、不同缓存之间设置缓冲区,根据运行时情况动态共享和边界更新,这样可以提高整体缓存的利用率。为不同的数据流建立流映射表,并根据运行状态实时更新,可以在对数据流没有先验知识的情况下,通过自适应更新迭代满足不同类型访问的性能需求。4种数据流类型发出的访问请求,其仲裁策略可以采用基于优先级加权的仲裁策略,优先级从高到低依次为低延迟访问low_latency、零负载访问zero load、普通访问normal、高带宽访问high band,这样既可以满足低延时数据流对链路延时较高的需求,又可以满足高带宽的访问突发需求,为主机侧的IO操作提供更大的优化空间。在高带宽类型的请求,必须保证最后一个tag对应的读请求发送完成后,仲裁电路才能接收新的请求,这样可以避免大批量数据的断流传输。按照tag号可以唯一索引内部的tag上下文,缓存地址等信息,使得对tag上下文以及对应缓存地址的索引效率非常高。提供的流分类方法便于tag分组和缓存管理,包括不限于本文提到的4种分组方式。本发明提供的方法不限于硬件实现,也可以配合软件实现,实现方式灵活。由于具有自适应的特性,既可以作为芯片设计的实现,同样也可以用于芯片建模下的性能收敛仿真。

[0064] 相应地,本发明在第二方面提供了一种基于流属性的PCIe tag资源分配装置,包括:

[0065] tag分配单元,用于通过流表映射确定来自多个功能实体的PCIe读请求的数据流类型,根据所述数据流类型从对应的tag资源分组中为所述读请求分配tag资源,并构建对应的上下文缓存;

[0066] 缓存分配单元,用于在片上缓存中为所述tag资源分配返回数据缓存,并根据分配结果更新所述上下文缓存,对所述多个已经分配tag资源的读请求进行仲裁,并形成读请求数据包发送到PCIe链路,其中所述返回数据缓存的空间粒度基于各个数据流类型的缓存利用率来实时更新;

[0067] 接收单元,用于从所述PCIe链路接收所述读请求的返回包,解析得到返回包对应的tag,将所述返回包中的数据负载存储到与所述tag对应的返回数据缓存,并返回到对应的功能实体。

[0068] 上述装置可通过上述第一方面的实施例提供的基于流属性的PCIe tag缓存自适

应资源分配方法实现,具体的实现方式可以参见第一方面的实施例中的描述,在此不再赘述。

[0069] 可以理解,上述实施例中描述的电路结构、字段结构和参数仅为举例。本领域技术人员还可以根据使用需要,对以上多个实施例的结构特征进行容易想到的组合和调整,而不应将本发明的构思限制于上述示例的具体细节。例如本文提到的4种分组方式,仅仅是用于实现tag分配和缓存分配的具体示例。在应用中并不局限于这4种分组方式。

[0070] 尽管参照前述实施例对本发明进行了详细的说明,本领域的普通技术人员应当理解,其依然可以对前述各实施例所记载的技术方案进行修改,或者对其中部分技术特征进行等同替换;而这些修改或者替换,并不使相应技术方案的本质脱离本发明各实施例技术方案的精神和范围。

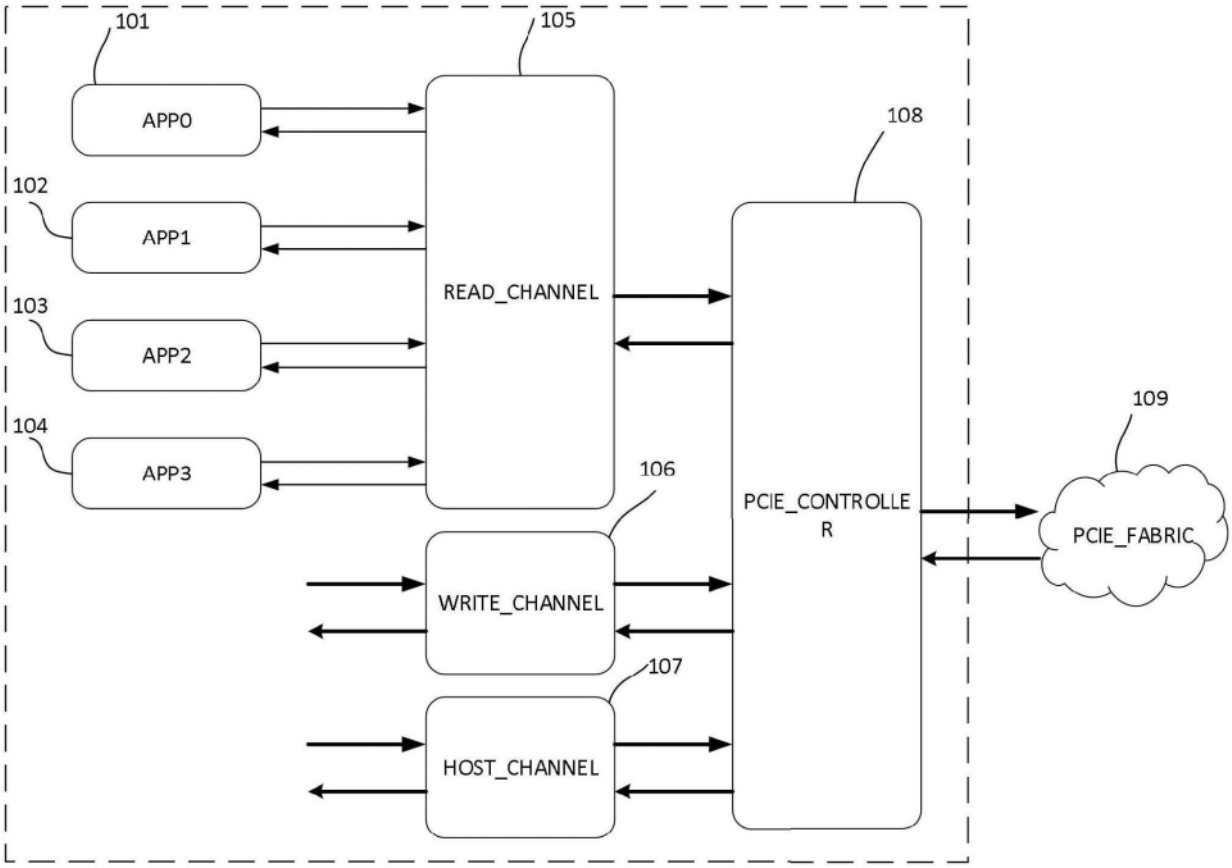

101
105
APP0
108
102
APP1
READ_CHANNEL
103
APP2
109
104
APP3
106
PCIE_CONTROLLE
R
PCIE_FABRIC
WRITE_CHANNEL
107
HOST_CHANNEL

图1

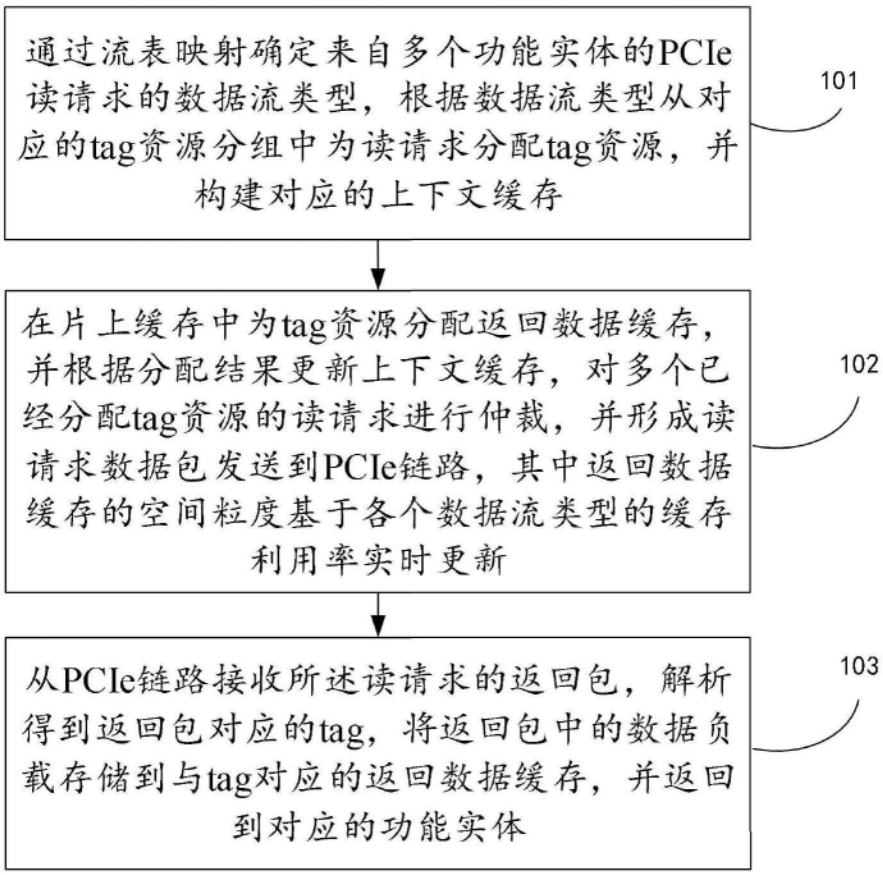

通过流表映射确定来自多个功能实体的PCIe读请求的数据流类型，根据数据流类型从对应的tag资源分组中为读请求分配tag资源，并构建对应的上下文缓存
101
在片上缓存中为tag资源分配返回数据缓存，并根据分配结果更新上下文缓存，对多个已经分配tag资源的读请求进行仲裁，并形成读请求数据包发送到PCIe链路，其中返回数据缓存的空间粒度基于各个数据流类型的缓存利用率实时更新
102
从PCIe链路接收所述读请求的返回包，解析得到返回包对应的tag，将返回包中的数据负载存储到与tag对应的返回数据缓存，并返回到对应的功能实体
103

图2

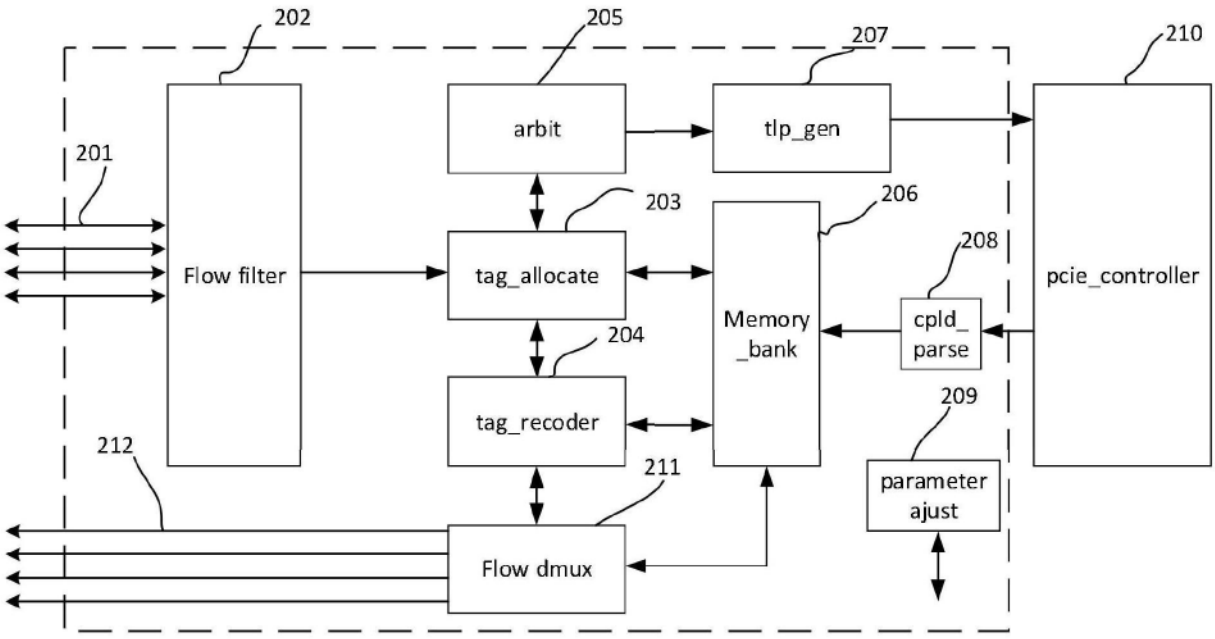

202
205
207
210
201
arbit
tlp_gen
203
206
208
Flow filter
tag_allocate
Memory _bank
cpld_ parse
pcie_controller
204
tag_recoder
209
212
211
parameter ajust
Flow dmux

图3

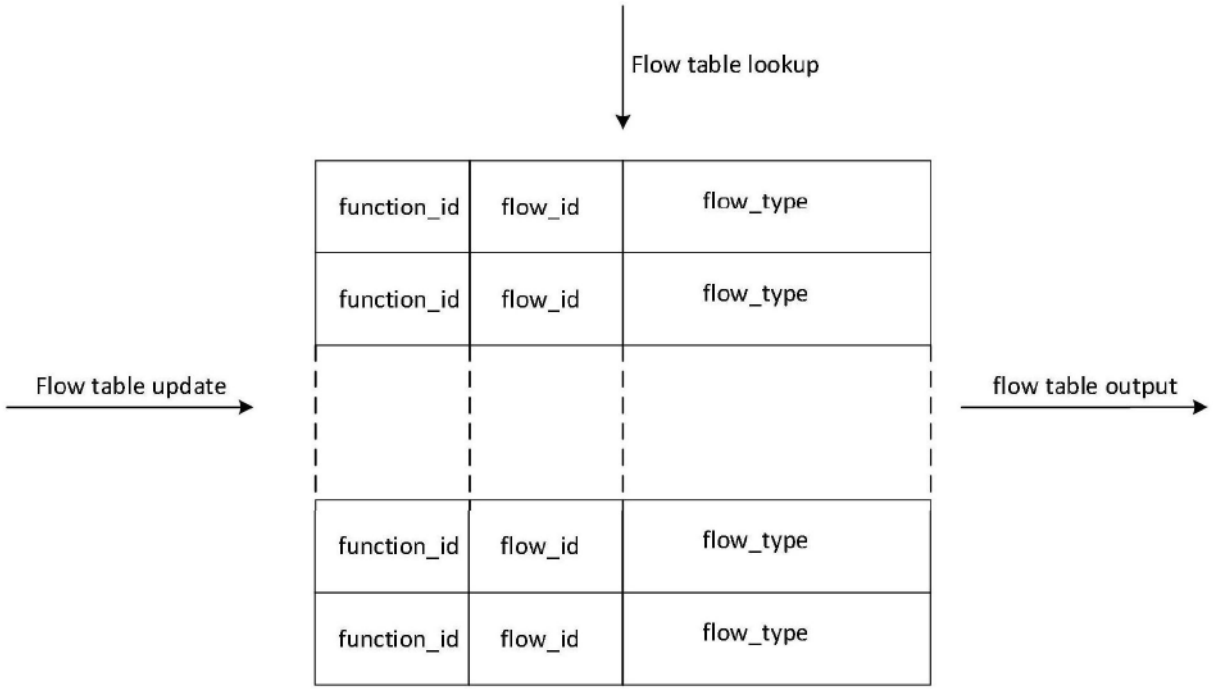

Flow table lookup
function_id
flow_id
flow_type
function_id
flow_id
flow_type
Flow table update
flow table output
function_id
flow_id
flow_type
function_id
flow_id
flow_type

图4

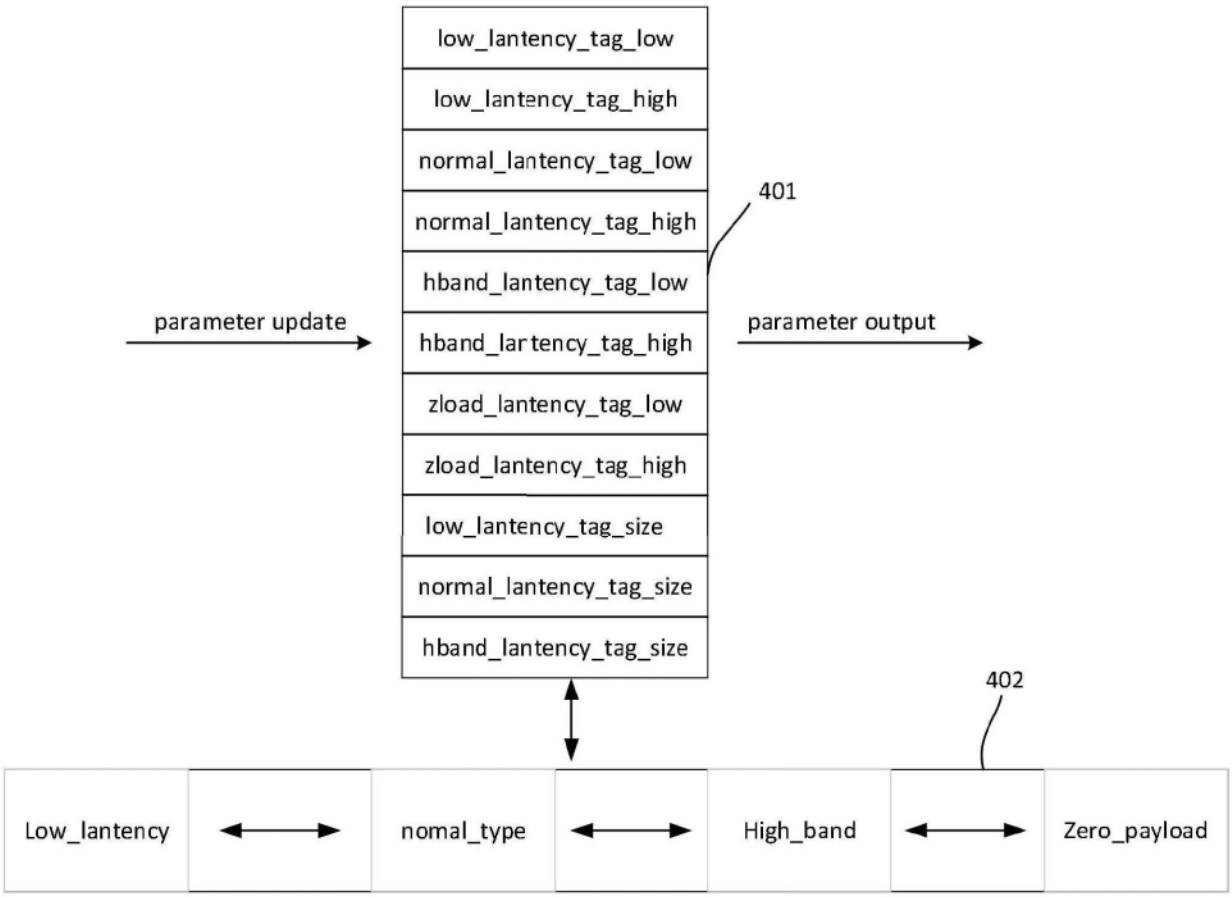

low_lantency_tag_low
low_lantency_tag_high
normal_lantency_tag_low
normal_lantency_tag_high
hband_lantency_tag_low
hband_lartency_tag_high
zload_lantency_tag_low
zload_lantency_tag_high
low_lantency_tag_size
normal_lantency_tag_size
hband_lantency_tag_size
401
parameter update
parameter output
402
Low_lantency
nomal_type
High_band
Zero_payload

图5

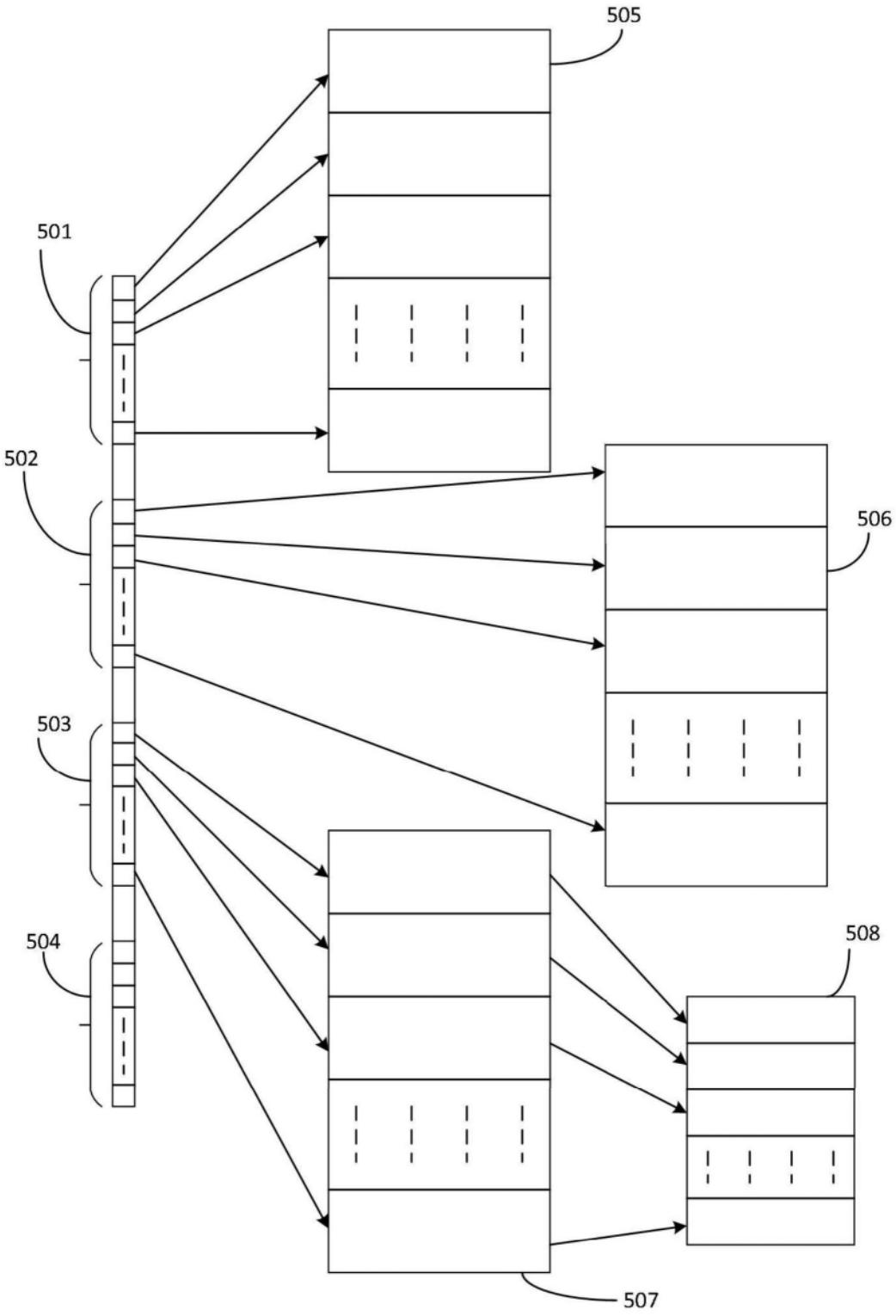


图6

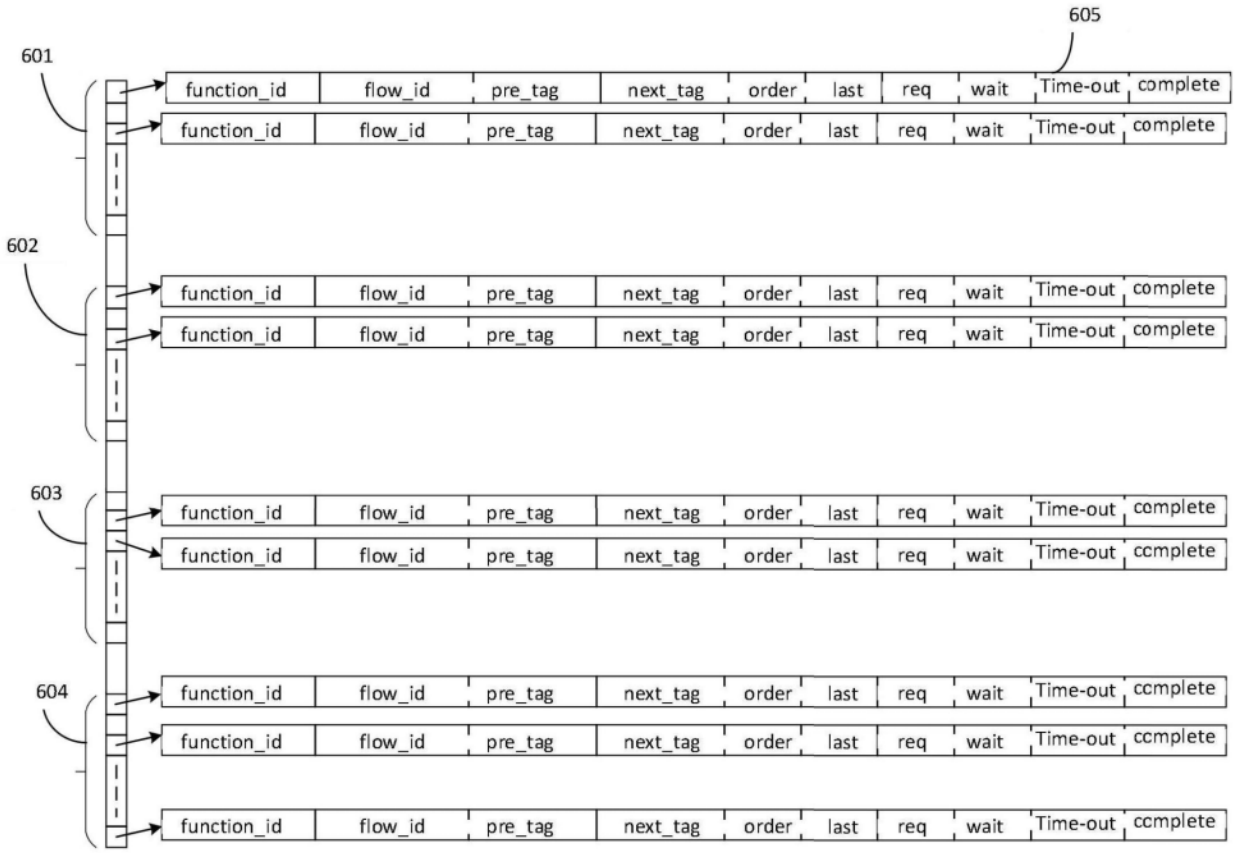

605
601
602
603
604
function_id
flow_id
pre_tag
next_tag
order
last
req
wait
Time-out
complete
ccmplete

图7

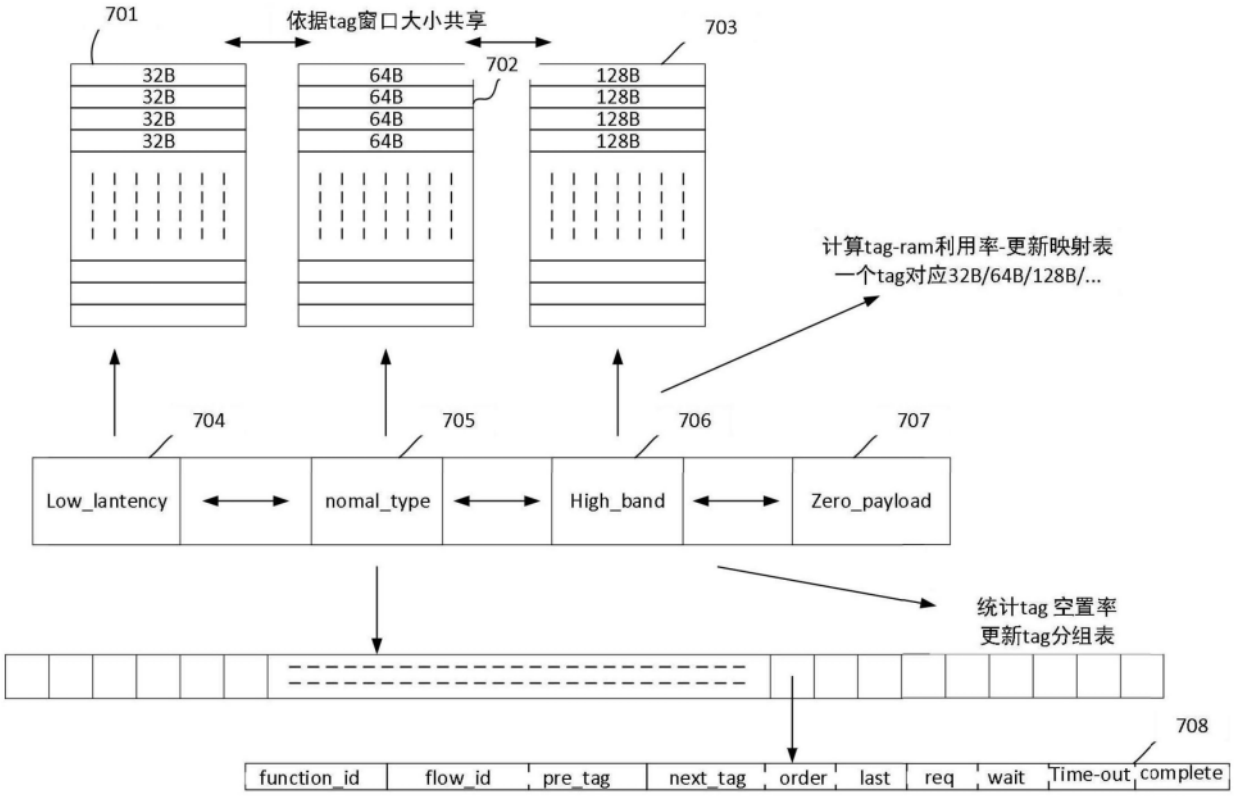

701
依据tag窗口大小共享
703
702
32B
32B
32B
32B
64B
64B
64B
64B
128B
128B
128B
128B
计算tag-ram利用率-更新映射表
一个tag对应32B/64B/128B/...
704
705
706
707
Low_lantency
nomal_type
High_band
Zero_payload
统计tag 空置率
更新tag分组表
708
function_id
flow_id
pre_tag
next_tag
order
last
req
wait
Time-out
complete

图8

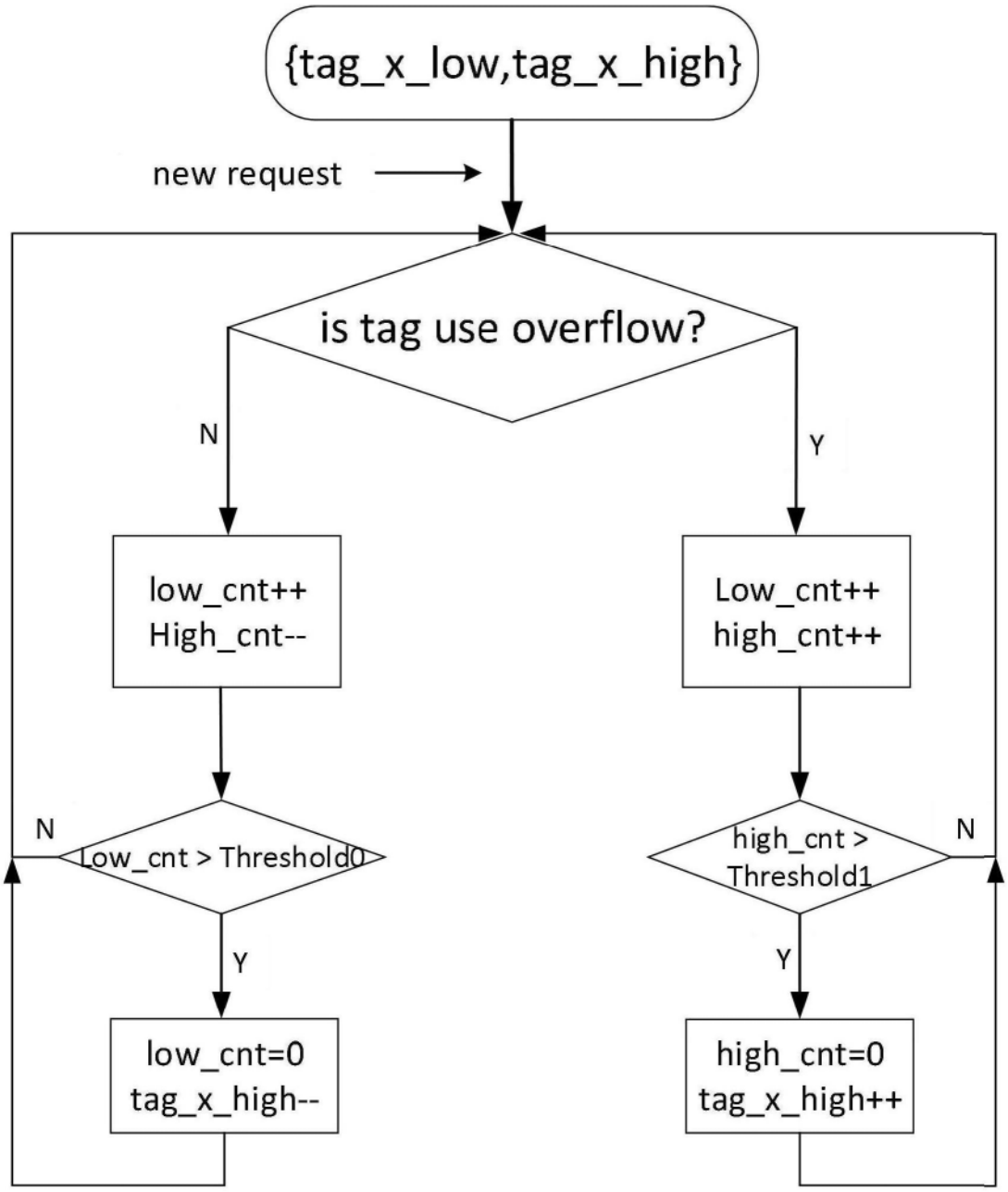

{tag_x_low,tag_x_high}
new request
is tag use overflow?
N
Y
low_cnt++
High_cnt--
Low_cnt++
high_cnt++
N
Low_cnt > Threshold0
high_cnt > Threshold1
N
Y
Y
low_cnt=0
tag_x_high--
high_cnt=0
tag_x_high++

图9

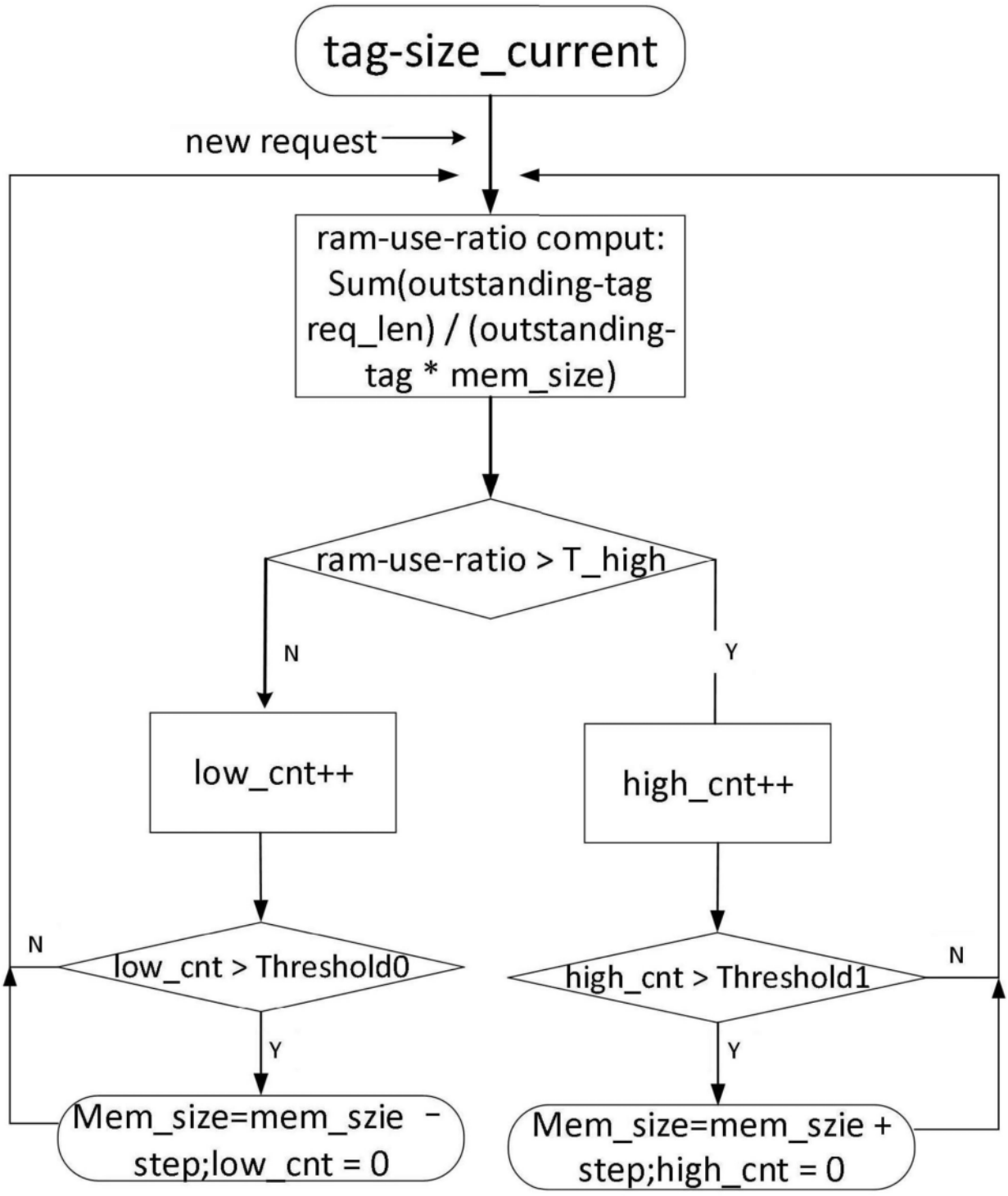

tag-size_current
new request
ram-use-ratio comput: Sum(outstanding-tag req_len) / (outstanding-tag * mem_size)
ram-use-ratio > T_high
N
Y
low_cnt++
high_cnt++
N
low_cnt > Threshold0
high_cnt > Threshold1
N
Y
Y
Mem_size=mem_szie - step;low_cnt = 0
Mem_size=mem_szie + step;high_cnt = 0

图10

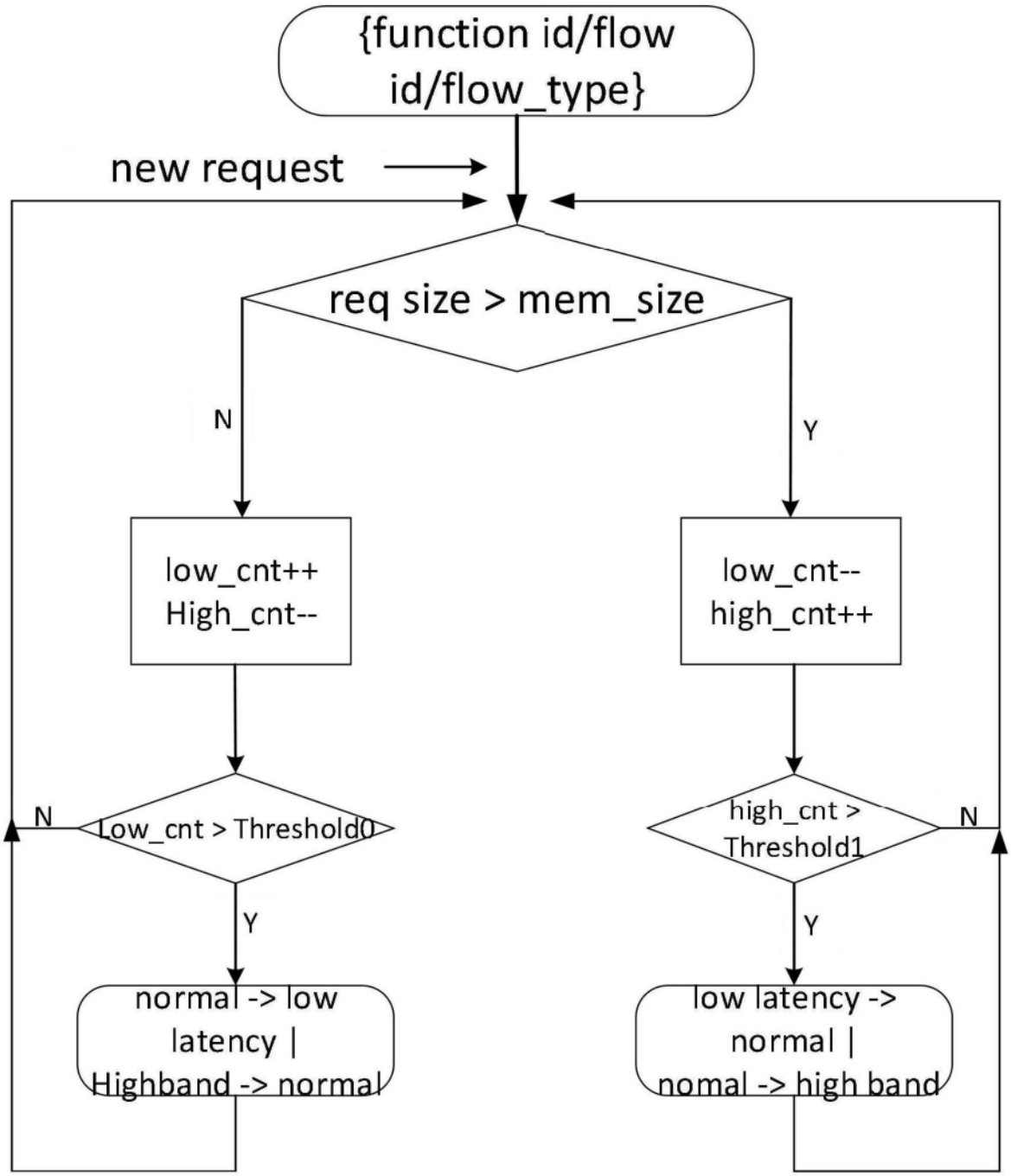

{function id/flow id/flow_type}
new request
req size > mem_size
N
Y
low_cnt++
High_cnt--
low_cnt--
high_cnt++
N
Low_cnt > Threshold0
high_cnt > Threshold1
N
Y
Y
normal -> low latency | Highband -> normal
low latency -> normal | nomal -> high band

图11