(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 105589814 A
(43)申请公布日 2016.05.18

(21)申请号 201510953863.5

(22)申请日 2015.12.17

(71)申请人 北京大学
地址 100871 北京市海淀区颐和园路 5 号

(72)发明人 孙广宇 张宪 张超 张玮其

(74)专利代理机构 北京万象新悦知识产权代理事务所（普通合伙）11360
代理人 张肖琪

(51)Int.Cl.
G06F 12/0802(2016.01)

权利要求书2页 说明书7页 附图4页

(54)发明名称

针对 Path ORAM 的叉型访问方法

(57)摘要

本发明公布了一种针对 Path ORAM 的叉型访问方法，包括：地址请求队列 ARQ 处理末级缓存请求 LLC 阶段、标签请求队列 LRQ 处理地址请求队列 ARQ 请求的阶段、地址转换逻辑处理标签请求队列 LRQ 请求的阶段，MAC 处理地址转换逻辑请求阶段和内存处理 MAC 请求阶段；MAC 处理地址转换逻辑请求阶段，树状结构的模式不可见存储器 ORAM tree 的被访问方式是叉型的。本发明方法通过隐藏访存模式的硬件结构优化，利用去除相邻两个路径的重叠部分的访问，减小了 ORAM 访存的数量和代价，加快了 ORAM 系统的执行速度，并且还可以降低功耗，从而大大提高系统整体性能。

CN 105589814 A

1.一种针对Path ORAM的叉型访问方法,包括:地址请求队列ARQ处理末级缓存请求LLC阶段、标签请求队列LRQ处理地址请求队列ARQ请求的阶段、地址转换逻辑处理标签请求队列LRQ请求的阶段,MAC处理地址转换逻辑请求阶段和内存处理MAC请求阶段,所述叉型访问方法通过隐藏访存模式的硬件结构优化,达到在很小的硬件代价下降低Path ORAM的访存代价,从而大大提高系统整体性能;具体包括如下步骤:

A.在地址请求队列ARQ处理末级缓存LLC请求阶段,执行如下操作:

A1.设末级缓存(LLC)请求为{op,addr},op表示读操作请求或写操作请求,addr表示末级缓存希望访问的逻辑地址;当末级缓存LLC发生一次miss未命中时,需要对内存进行访问;

A2.当末级缓存LLC对内存进行访问时,对于每个末级缓存LLC请求{op,addr},当地址请求队列ARQ中没有和addr相同的请求时,直接将末级缓存LLC请求插入到地址请求队列ARQ;

A3.将地址请求队列ARQ中的所有请求转换为标签label;

A4.将已被转换为label请求的地址请求队列ARQ请求传输至标签请求队列LRQ,直到标签请求队列LRQ充满数据请求;

B.在标签请求队列LRQ处理地址请求队列ARQ请求的阶段,进入到标签请求队列LRQ中的标签label经过调度后,将最前的标签请求传递给地址转换逻辑;

C.在地址转换逻辑处理标签请求队列LRQ请求的阶段,转换逻辑对标签label请求进行处理,通过叉型访问方式访问树状结构的模式不可见存储器ORAM tree,将标签label请求转化为针对内存逻辑地址的读写请求序列,所述地址转换逻辑得到的读写请求再被传输给MAC;具体执行如下操作:

C1.当前请求不访问上一个请求访问过的逻辑地址:将每个标签label即将转化成的请求中读的部分,对应path中与标签请求队列LRQ中上一个标签label'对应的path不重合的部分;

C2.当前请求不写回上一个请求即将访问的逻辑地址:将每个label即将转化成的请求中写的部分,对应path中与LRQ中下一个label对应的path不重合的部分;

C3.将地址转换逻辑得到的多个读写请求传输给MAC;

D.在MAC处理地址转换逻辑请求的阶段:MAC接受步骤C3所述读写请求后,通过对应的标签位tag判断所述读写请求是命中hit或是未命中miss;当所述读写请求为命中hit时,直接进行读写操作;当所述读写请求为未命中miss时,向内存发送所述读写请求,并通过最近使用策略替换相应的块;

E.在内存处理MAC请求阶段:在内存接收到步骤D所述MAC请求后,将对应的数据返回给ORAM controller或者写入内存。

2.如权利要求1所述针对Path ORAM的叉型访问方法,其特征是,所述步骤A2中,当地址请求队列ARQ中有和addr相同的请求{op',addr}时,分为下列四种情况:

A21)当(op',op)=(read,read)时,将{op,addr}当做普通请求处理即可;

A22)当(op',op)=(read,write)时,{op,addr}被挂起等待{op',addr}被处理完才能发射;

A23)当(op',op)=(write,read)时,直接返回数据给末级缓存;

A24)当(op',op)=(write,write)时,{op',addr}会被取消。

3.如权利要求1所述针对Path ORAM的叉型访问方法,其特征是,所述步骤A3所述将地址请求队列ARQ中的所有请求转换为标签label,具体是当地址表中有对应的label时,通过查询地址表转换为对应的label;当地址表中没有对应的label时,通过层次化的模式不可见存储器hierarchical ORAM转化为一个新的label。

4.如权利要求1所述针对Path ORAM的叉型访问方法,其特征是,所述步骤B所述调度具体执行如下操作:

B1.判断标签请求队列LRQ是否已满,当标签请求队列LRQ不满时,在标签请求队列LRQ中插入无意义标签;当标签请求队列LRQ已满时,执行步骤B2;

B2.对于标签请求队列LRQ中的所有标签,计算得到该标签对应的path与当前正被访存path的重叠部分块数,重叠得最多的块会被提前到LRQ的最前,成为下次被执行的LRQ请求;

B3.在B2步骤执行过程中,如果有新的A步骤(A4步骤和B步骤同时进行)产生的label请求进入LRQ,新的label会替换无意义label,并且这个label会和LRQ最前的label进行比较,看谁与当前path的重合度更高,更高的会在LRQ最前;

B4.标签请求队列LRQ中最前的请求会被传输给地址转换逻辑,等待地址转换逻辑将其转化为内存能理解的读写操作。

5.如权利要求4所述针对Path ORAM的叉型访问方法,其特征是,步骤B2所述计算得到该标签对应的path与当前正被访存path的重叠部分块数,具体通过异或该标签对应的path与当前正被访存path的label值,根据异或结果中0出现的最高位得到重叠块数。

6.如权利要求1所述针对Path ORAM的叉型访问方法,其特征是,步骤D所述MAC处理地址转换逻辑请求的阶段,具体执行如下操作:

D1.MAC只缓存Path ORAM从相邻路径平均重叠长度的层数和向上的层数的块地址,缓存的块数等于MAC大小除以数据块大小;所述相邻路径平均重叠长度根据标签请求队列LRQ的长度计算得到;

D2.针对每个请求,MAC判断该请求是否属于自己缓存的层;当该请求不属于自己缓存的层时,直接将该请求传输给内存;当请求对应的块处于相邻路径平均重叠长度层数和向上层数的块地址时,将该请求缓存;

D3.当该请求属于自己缓存的层时,首先计算出在对应的组set,通过对应的标签位tag计算得到是命中hit还是未命中miss;当命中hit时,直接进行读写操作;当未命中miss时,向内存发送该请求,并通过最近使用策略替换相应的块。

7.如权利要求6所述针对Path ORAM的叉型访问方法,其特征是,步骤D3所述通过对应的组set和对应的标签位tag计算得到是命中hit还是未命中miss,具体计算方法如下:首先将请求的地址转化为对应树状结构中的x层的第y个块;然后x除以每个set大小得到值设为w;在第w+1组中检验标签位tag,根据是否有tag值等于请求的逻辑地址得到是命中hit还是未命中miss。

8.如权利要求1所述针对Path ORAM的叉型访问方法,其特征是,步骤E所述ORAM controller接收返回数据后,把数据全部放在stash中,并将LLC的请求{op,addr}所需的数据返回给CPU;再随机分配一个新的值赋给position map中LLC请求对应的块的标签label。

针对Path ORAM的叉型访问方法

技术领域

[0001] 本发明属于信息安全领域,涉及一种隐藏访存模式的硬件结构优化方法,具体涉及一种针对Path ORAM的叉型访问方法。

背景技术

[0002] 基于路径的模式不可见存储器(Path Oblivious RAM,Path ORAM)是一种可以隐藏访存模式的硬件结构。Path ORAM拥有高访存效率,算法简单等优点。由于其可行性,Path ORAM被认为是最有可能用于安全处理器(Secure Processor)上的ORAM结构之一,其原型被实现在了FPGA以及ASIC代码上。但是Path ORAM还是存在一些问题,最大的问题就是访存代价过高。

[0003] 在文献“Maas M,Love E,Stefanov E,et al.Phantom:Practical oblivious computation in a secure processor.Acm Ccs,2013:311-324”中,美国Martin Maas等人用FPGA实现了第一个基于Path ORAM的安全处理器Phantom,通过使用顶部缓存法(treetop caching)以及一种树状结构来组织stash,从而降低了访存代价;但是,Phantom的每个数据块大小为4KB,和通常基于64B的块大小的ORAM设计相比,大大增加了其访存方面的代价。在文献“Fletcher C W,Ren L,Kwon A,et al.Freecursive oram:[nearly]free recursion and integrity verification for position-based oblivious ram.2015.”中,美国的Christopher W.Fletcher等人提出一种优化Hierarchical Path ORAM中对Position Map访问的方法,该方法引入了PosMap Lookaside Buffer的结构来缓存PosMap的信息,同时,PosMap压缩技术也有效降低了访存代价;但是,该方法针对Path ORAM的访存代价还是很高。

发明内容

[0004] 为了克服上述现有技术的不足,本发明提供一种针对Path ORAM的叉型访问方法,该方法是一种隐藏访存模式的硬件结构优化方法,能够达到在很小的硬件代价下降低Path ORAM的访存代价,从而大大提高系统整体性能。

[0005] 为了行文方便,本文所有术语定义如下:

[0006] ORAM Tree(树状结构的模式不可见存储器):指外部存储memory的一种树形组织形式。外部memory被逻辑上组织成一个二叉树结构。每个节点叫做一个bucket(篮子),其中包含Z个加密后的memory块(Z为一个大于3的整数)。每个叶节点都有一个标号,叫做label。每次memory访问都是对ORAM tree的一条从某个叶子节点(label-k)到根节点的路径进行访问(简称为path-k),会将这条path上的所有memory块取入到处理器中(读阶段),处理器读取出相应块后,会把数据写回到这条path(写阶段)。

[0007] ORAM Controller(模式不可见存储器的控制器):指processor(处理器)内部的控制ORAM工作的相关电路;包含三个部分:地址表(Position Map),隐蔽缓存(Stash)以及地址转换逻辑(Address Logic)。地址表是一个包含每个program地址的对应label的查询表,

隐蔽缓存是一个缓存结构，用来存放每次访存上来path中的数据块。地址转换逻辑用来计算每个program地址转换后对应path上每个块的地址。

[0008] Label Request Queue(LRQ，标签请求队列)：表示在处理器中，大量program地址经过地址表转换为大量的label后，存放label的一个队列结构。Label Request Queue里的每个label请求都有一个counter，用来记录label进入queue的时间。

[0009] Address Request Queue(ARQ，地址请求队列)：表示在处理器中，大量的program地址访存请求存放的一个队列结构。每个访存请求都有一个Ready标志位，用来解决数据冒险问题。

[0010] Merging Aware Cache(MAC，考虑路径合并策略的高速缓存)：表示一个在地址转换逻辑之后的缓存结构，只用来缓存ORAM tree中特定层的数据。实现时，可以用传统cache结构，每个Oram tree中的memory块都可以根据块地址转换为特定的组数(set number)，每个set内部采用LRU替换策略。

[0011] 本发明的原理是，首先对于任意一个末级缓存(LLC)发来的访存请求，将其插入到ARQ中，通过dependency(数据依赖性)判断以及在地址表的辅助下，将其转化为label值并插入到LRQ中，然后进行调度(scheduling)算法，将与前一个访存path重叠度最高的label提到queue的最前等待执行。每次LRQ最前的一个label会被传递给地址转换逻辑，转换为一系列逻辑地址(即在ORAM tree中与label对应的path上所有块地址)后，首先判断每个地址是否在MAC中，如果在，将其对应数据从MAC中读取到隐蔽缓存中，如果不在就访问memory的相应位置。在隐蔽缓存处理完数据、末级缓存的请求被完成后，隐蔽缓存会重新写回数据到label对应的path，特定层的数据会被写回到MAC中，其余层的数据会被写回到memory中。由此实现在很小的硬件代价下降低Path ORAM的访存代价，能够大大提高系统整体性能。

[0012] 本发明提供的技术方案是：

[0013] 一种针对Path ORAM的叉型访问方法，包括：地址请求队列ARQ处理末级缓存请求LLC阶段、标签请求队列LRQ处理地址请求队列ARQ请求的阶段、地址转换逻辑处理标签请求队列LRQ请求的阶段，MAC处理地址转换逻辑请求阶段和内存处理MAC请求阶段，所述叉型访问方法通过隐藏访存模式的硬件结构优化，达到在很小的硬件代价下降低Path ORAM的访存代价，从而大大提高系统整体性能；具体包括如下步骤：

[0014] A.在地址请求队列ARQ处理末级缓存LLC请求阶段，执行如下操作：

[0015] A1.末级缓存(LLC)请求设为{op,addr}，op表示读操作请求或写操作请求，addr表示末级缓存希望访问的逻辑地址；当末级缓存(LLC)发生一次miss(未命中)时，需要对内存进行访问；

[0016] A2.来自末级缓存LLC的请求需要对内存进行访问时，对于每个末级缓存LLC请求{op,addr}，根据地址请求队列ARQ中是否有和addr相同的请求{op',addr}分别进行处理，当地址请求队列ARQ中没有和addr相同的请求时(addr不冲突)，直接将末级缓存LLC请求插入到地址请求队列ARQ；

[0017] 当地址请求队列ARQ中有和addr相同的请求{op',addr}时，分为下列四种情况：

[0018] A21)当(op',op)＝(read,read)时，将{op,addr}当做普通请求处理即可；

[0019] A22)当(op',op)＝(read,write)时，{op,addr}被挂起等待{op',addr}被处理完才能发射；

[0020] A23)当(op',op)＝(write,read)时,直接返回数据给末级缓存;

[0021] A24)当(op',op)＝(write,write)时,{op',addr}会被取消;

[0022] A3.对于ARQ中的所有请求,通过查询地址表转换为对应的label,如果地址表中没有对应的label,通过层次化模式不可见存储器(hierarchy ORAM)中的相应机制转化为一个新的label;

[0023] A4.将已被转换为label请求的ARQ请求传输至标签请求队列LRQ,直到标签请求队列LRQ已经充满数据请求,这时ARQ会等待LRQ有空位再传输,直到LRQ中没有无意义请求时停止传输;

[0024] B.在标签请求队列LRQ处理地址请求队列ARQ请求的阶段,label进入到标签请求队列LRQ,标签请求队列中的label经过调度后,最前的label传递给地址转换逻辑;调度具体执行如下操作:

[0025] B1.首先判断LRQ是否已满(即充满数据请求),如果不满,在LRQ中插入无意义label(dummy label);

[0026] B2.对于LRQ中的所有label,判断label对应的path与当前正被访存path的重叠部分块数。判断具体可以通过异或两个path的label值,看异或结果中0出现的最高位,例如是label中的第k个比特,那么重叠块数即为Z(篮子bucket里内存块的数量)*k。重叠得最多的块会被提前到LRQ的最前,成为下次被执行的LRQ请求;

[0027] B3.在B2步骤执行过程中,如果有新的A步骤(A4步骤和B步骤同时进行)产生的label请求进入LRQ,新的label会替换无意义label,并且这个label会和LRQ最前的label进行比较,看谁与当前path的重合度更高,更高的会在LRQ最前;

[0028] B4.最前的LRQ请求会被传输给地址转换逻辑,等待地址转换逻辑将其转化为内存能理解的读写操作;

[0029] C.在地址转换逻辑处理LRQ请求的阶段,转换逻辑对label请求进行处理,将其转化为针对内存逻辑地址的读写序列;执行如下操作:

[0030] C1.对于每个label即将转化成的请求中读的部分,对应path中与LRQ中上一个label(即label')对应的path不重合的部分,当前请求不访问上一个请求访问过的逻辑地址;即上一个请求访问过的逻辑地址,本请求不会访问;

[0031] C2.对于每个label即将转化成的请求中写的部分,对应path中与LRQ中下一个label对应的path不重合的部分;当前请求不写回上一个请求即将访问的逻辑地址,即下一个请求即将访问的逻辑地址,本请求不会将其写回。

[0032] C3.地址转换逻辑得到的大量读写请求将被传输给MAC;

[0033] 在C2和C3步骤的控制下,树状结构的模式不可见存储器(ORAM tree)的被访问的方式恰好是叉型的,因此,本发明提供方法称为叉形访问方法。而现有的Path ORAM的访问方式下,把柄部分会被多读写1次;因此,本发明提供方法提高了效率。

[0034] 在C2和C3步骤的控制下,树状结构的模式不可见存储器(ORAM tree)的被访问的方式恰好是叉型的,因此,本发明提供方法称为叉形访问方法。而现有的Path ORAM的访问方式下,把柄部分会被多读写1次;因此,本发明提供方法提高了效率。

[0035] D.在merge的缓存结构(Merging Aware Cache(MAC)处理地址转换逻辑请求的阶段,针对MAC接受读写序列后,通过对应的标签(tag)位判断是否hit以及miss,操作方式和

普通cache一致;执行如下操作:

[0036] D1.对于MAC,只缓存Path ORAM从“相邻路径平均重叠长度”层数以及向上一定层数的块地址,缓存的块数和MAC的大小有关;缓存的块数等于MAC大小除以数据块大小;相邻路径平均长度取决于标签请求队列LRQ的长度,经验公式是2+log2(LRQ长度)这个数的像下取整。

[0037] D2.对于每个请求,MAC首先判断是否属于那些自己缓存的层,如果不属于,直接将此请求传输给内存;即:只要请求对应的块处于“相邻路径平均重叠长度”层数以及向上一定层数的块地址,这个块会被缓存,否则不会。

[0038] D3.如果属于相应的层,那么首先计算出在对应的set(组),通过对应的标签(tag)位判断是否hit(命中),如果hit,直接进行读写操作。计算方法如下:首先将请求的地址转化为对应树状结构中的x层的第y个块。然后x除以每个set大小,假设得到w。那么在第w+1那个组中去检验tag,看是否有tag值等于请求的逻辑地址,若等于为hit、若不等于为miss。如果miss(未命中),那么向内存发送相应的请求,并采用LRU(最近使用策略)替换此set中相应的块;

[0039] E.内存处理MAC请求阶段:在内存接收到MAC请求后,返回或者写入对应的数据即可:数据返回给ORAM controller或者数据写入内存:

[0040] E1.ORAM controller接收返回数据后,把数据全部放在stash中,并且把LLC的{op,addr}请求所需数据返回给CPU;

[0041] E2.position map中LLC请求对应的块的label会随机分配一个新的值。

[0042] 与现有技术相比,本发明的有益效果是:

[0043] 本发明提供一种针对Path ORAM的叉型访问方法,该方法是一种隐藏访存模式的硬件结构优化方法。通过本发明所提供的Path ORAM的访存模式,利用去除相邻两个路径的重叠部分的访问,减小了ORAM访存的数量,加快了ORAM系统的执行速度,并且还可以降低功耗。具体地,本发明具有如下优点:

[0044] (一)通过叉形访问来避免对两个相邻ORAM请求重叠部分的访问;

[0045] (二)通过对LRQ中的ORAM请求的顺序进行调度来增加相邻ORAM请求重叠程度;

[0046] (三)通过在地址转换逻辑后增加一个只缓存从“相邻路径平均重叠长度”层数以及向上一定层数的块地址数据的缓存进一步减小访存数量。

附图说明

[0047] 图1是本发明提供的针对Path ORAM的叉型访问方法的流程框图。

[0048] 图2是本发明提供方法中ARQ处理LLC请求的流程框图。

[0049] 图3是本发明提供方法中LRQ处理ARQ请求的流程框图。

[0050] 图4是本发明提供方法中地址转换逻辑处理LRQ请求流程框图。

[0051] 图5是本发明提供方法中MAC处理地址转换逻辑请求的流程框图。

[0052] 图6是本发明提供方法中内存处理MAC请求阶段的流程框图。

具体实施方式

[0053] 下面结合附图,通过实施例进一步描述本发明,但不以任何方式限制本发明的范

围。

[0054] 本发明提供一种针对Path ORAM的叉型访问方法，该方法是一种隐藏访存模式的硬件结构优化方法，能够达到在很小的硬件代价下降低Path ORAM的访存代价，从而大大提高系统整体性能。

[0055] 本发明提供的针对Path ORAM的叉型访问方法具体包括五个阶段——ARQ处理LLC请求阶段，LRQ处理ARQ请求阶段，地址转换逻辑处理LRQ请求阶段，MAC处理地址转换逻辑请求阶段，内存处理MAC请求阶段。

[0056] 对于1个Path ORAM(例如三层的DDR3内存)，Path ORAM的层数取决于内存、数据块的大小以及bucket的大小；每个bucket中包含Z个(Z是一个大于等于4的整数)加密后的memory块(数据块)；新来的末级缓存LLC请求序列是{op1,addr1},{op2,addr2}(其中op1，op2＝read或者write)。{op1,addr1}表示对addr1的数据进行op1操作(读或者写)的一个请求，{op2,addr2}表示对addr2的数据进行op2操作(读或者写)的一个请求。现在正在处理的请求是{op0,addr0}。ARQ能存储128个请求(ARQ长度可以是任意正整数，本例中ARQ长度128)，LRQ的长度是64(LRQ长度可以是任意正整数，本例中为64)，MAC的大小可以是任意整数个的数据块大小，假设processor内部控制ORAM工作的相关电路包含的地址表(Position Map)中，{op1,addr1},{op2,addr2},{op0,addr0}这3个请求对应的标签分别叫做label1、label2、label0；针对Path ORAM的叉型访问方法包括如下步骤：

[0057] A.在地址请求队列ARQ处理末级缓存请求LLC阶段，执行如下操作：

[0058] A1.当末级缓存(LLC)发生一次miss(未命中)需要对memory进行访问，对于每个末级缓存请求{op,addr}(op表示请求是读还是写操作，addr表示末级缓存希望访问的逻辑地址)，首先判断ARQ中是否有和addr相同的请求{op’,addr}，如果有分为下列四种情况：

[0059] 1)(op’,op)＝(read,read)，将{op,addr}当做普通请求处理即可；

[0060] 2)(op’,op)＝(read,write)，{op,addr}被挂起等待{op’,addr}被处理完才能发射；

[0061] 3)(op’,op)＝(write,read)，直接返回数据给末级缓存；

[0062] 4)(op’,op)＝(write,write)，{op’,addr}会被取消。

[0063] A2.如果addr不冲突，ARQ中没有和addr相同的请求{op’,addr}，直接将LLC请求插入到ARQ即可；

[0064] A3.对于ARQ中的所有请求，通过查询地址表，被转换为对应的label，如果地址表中没有对应的label，会通过h层次化的模式不可见存储器(hierarchy ORAM)中的相应机制转化为一个新的label；

[0065] A4.ARQ请求(已被转换为对应的label请求)会不停传输至标签请求队列LRQ，直到标签请求队列LRQ已经充满数据请求，这时ARQ会等待LRQ有空位再传输(直到LRQ中没有无意义请求时停止传输)；

[0066] B.在标签请求队列LRQ处理地址请求队列ARQ请求的阶段，label进入到LRQ，队列中的label经历调度算法后，最前的label传递给地址转换逻辑；调度具体执行如下操作：

[0067] B1.首先判断LRQ是否已满(即充满数据请求)，如果不满，在LRQ中插入无意义label(dummy label)；

[0068] B2.对于LRQ中的所有label，判断label对应的path与当前正被访存path的重叠部

分块数。判断具体可以通过异或两个path的label值，看异或结果中0出现的最高位，例如是label中的第k个比特，那么重叠块数即为Z*k。重叠得最多的块会被提前到LRQ的最前，成为下次被执行的LRQ请求；

[0069] B3.在B2步骤执行过程中，如果有新的A步骤(A4步骤和B步骤同时进行)产生的label请求进入LRQ，新的label会替换无意义label，并且这个label会和LRQ最前的label进行比较，看谁与当前path的重合度更高，更高的会在LRQ最前；

[0070] B4.最前的LRQ请求会被传输给地址转换逻辑，等待地址转换逻辑将其转化为内存能理解的读写操作；

[0071] C.在地址转换逻辑处理LRQ请求的阶段，转换逻辑对label请求进行处理，将其转化为针对内存逻辑地址的读写序列；执行如下操作：

[0072] C1.对于每个label即将转化成的请求中读的部分，对应path中与LRQ中上一个label(即label')对应的path不重合的部分；即上一个请求访问过的逻辑地址，本请求不会访问；

[0073] C2.对于每个label即将转化成的请求中写的部分，对应path中与LRQ中下一个label对应的path不重合的部分；即下一个请求即将访问的逻辑地址，本请求不会将其写回。

[0074] C3.地址转换逻辑得到的大量读写请求将被传输给MAC；

[0075] 在C2和C3步骤的控制下，树状结构的模式不可见存储器(ORAM tree)的被访问的方式恰好是叉型的，因此，本发明提供方法称为叉形访问方法。而现有的Path ORAM的访问方式下，把柄部分会被多读写1次；因此，本发明提供方法提高了效率。

[0076] D.在merge的缓存结构(Merging Aware Cache(MAC)处理地址转换逻辑请求的阶段，针对MAC接受读写序列后，通过对应的标签(tag)位判断是否hit以及miss，操作方式和普通cache一致；执行如下操作：

[0077] D1.对于MAC，只缓存Path ORAM从“相邻路径平均重叠长度”层数以及向上一定层数的块地址，缓存的块数和MAC的大小有关；缓存的块数等于MAC大小除以数据块大小；相邻路径平均长度取决于标签请求队列LRQ的长度，经验公式是2+log2(LRQ长度)这个数的像下取整。

[0078] D2.对于每个请求，MAC首先判断是否属于那些自己缓存的层，如果不属于，直接将此请求传输给内存；即：只要请求对应的块处于“相邻路径平均重叠长度”层数以及向上一定层数的块地址，这个块会被缓存，否则不会。

[0079] D3.如果属于相应的层，那么首先计算出在对应的set(组)，通过对应的标签(tag)位判断是否hit(命中)，如果hit，直接进行读写操作。计算方法如下：首先将请求的地址转化为对应树状结构中的x层的第y个块。然后x除以每个set大小，假设得到w。那么在第w+1那个组中去检验tag，看是否有tag值等于请求的逻辑地址。如果miss(未命中)，那么向内存发送相应的请求，并采用LRU(最近使用策略)替换此set中相应的块；

[0080] E.内存处理MAC请求阶段：在内存接收到MAC请求后，返回或者写入对应的数据即可：数据返回给ORAM controller或者数据写入内存：

[0081] E1.ORAM controller接收返回数据后，把数据全部放在stash中，并且把LLC的{op,addr}请求所需数据返回给CPU；

[0082] E2.position map中LLC请求对应的块的label会随机分配一个新的值。

[0083] 下面通过实例对本发明做进一步说明：

[0084] 对于1个三层的DDR3内存，每个bucket中包含Z个加密后的memory块；假设Z的值是4，并且假设新来的LLC请求序列是(read,0)(write,1)(read,2)(write,3)，现在正在处理的请求是(write,0)。ARQ,LRQ的长度都是4，MAC的大小是一个块大小，假设processor内部控制ORAM工作的相关电路包含的地址表(Position Map)中，0,1,2,3四个逻辑地址对应的label分别是2,0,1,3。本发明的工作分为五个阶段，如图1所示，包括ARQ处理LLC请求阶段，LRQ处理ARQ请求阶段，地址转换逻辑处理LRQ请求阶段，MAC处理地址转换逻辑请求阶段，内存处理MAC请求阶段。

[0085] 图2是ARQ处理LLC请求的流程图。如图2所示，在ARQ处理LLC请求阶段，由于(read,0)与(write,0)是读写同一个地址，(read,0)请求直接被返回数据，因此ARQ中会加入3个请求(write,1)、(read,2)、(write,3)，查询Position Map注册表经过posmap的转换后，0,1,3这三个label请求会被发送给LRQ。

[0086] 图3是LRQ处理ARQ请求的流程图。如图3所示，LRQ处理ARQ请求阶段，由于path-3与path-2重合度最高(path-3和path-2重合路径长度为2个bucket，另外两个的重合路径长度只有1)，所以4这个请求会被提到最前，作为下一个被LRQ传输给下一阶段的请求。因此LRQ中的请求顺序是3,0,1。

[0087] 图4是地址转换逻辑处理LRQ请求流程图。参考图4，在地址转换逻辑处理LRQ请求阶段，由于path-3和path-2重合的地址是(0,1,2,3)(8,9,10,11)，和path-0重合的地址是(0,1,2,3)。转换逻辑对label请求进行处理，将其转化为针对内存逻辑地址的读写序列，所以这个label请求会转化为对(20,21,22,23)这些地址的读请求，以及对(4,5,6,7)(12,13,14,15)这些地址的写请求(括号中四个数表示一个bucket)。

[0088] 图5是MAC处理地址转换逻辑请求的流程图。参考图5，在MAC处理地址转换逻辑请求，假设(20,21,22,23)这个地址恰好在MAC中，那么这四个地址会直接返回给LLC；若不然，对(20,21,22,23)这四个地方的读请求会发送给内存，并且在数据进入MAC后会替换原有的数据。

[0089] 图6是内存处理MAC请求阶段的流程图。参考图6，在内存接收了MAC的读请求后，会返回(20,21,22,23)这四个地址的数据给ORAM controller，数据直接进入ORAM的stash；对于(4,5,6,7)(12,13,14,15)这些地址的写请求，内存直接写入对应地址即可。并且3这个逻辑对应的label在position map中会随机分配一个新的数(可能就不是3了)。

[0090] 需要注意的是，公布实施例的目的在于帮助进一步理解本发明，但是本领域的技术人员可以理解：在不脱离本发明及所附权利要求的精神和范围内，各种替换和修改都是可能的。因此，本发明不应局限于实施例所公开的内容，本发明要求保护的范围以权利要求书界定的范围为准。

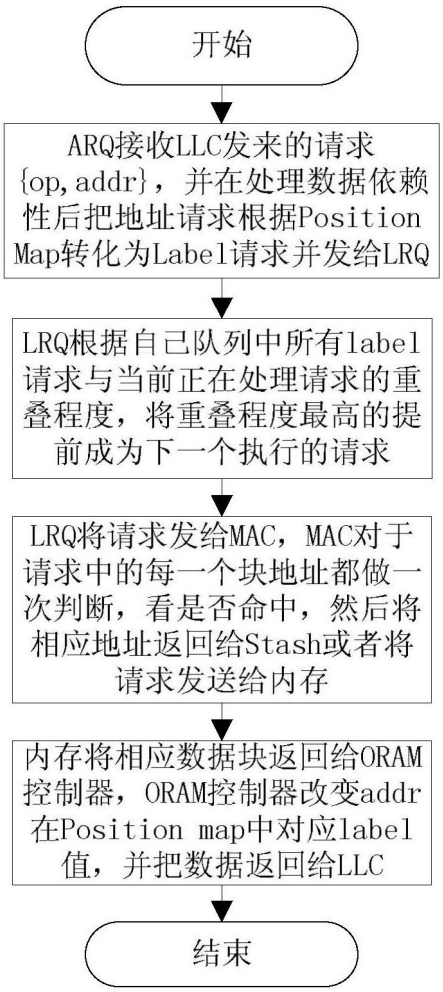
开始
ARQ接收LLC发来的请求{op,addr}，并在处理数据依赖性后把地址请求根据Position Map转化为Label请求并发给LRQ
LRQ根据自己队列中所有label请求与当前正在处理请求的重叠程度，将重叠程度最高的提前成为下一个执行的请求
LRQ将请求发给MAC，MAC对于请求中的每一个块地址都做一次判断，看是否命中，然后将相应地址返回给Stash或者将请求发送给内存
内存将相应数据块返回给ORAM控制器，ORAM控制器改变addr在Position map中对应label值，并把数据返回给LLC
结束

图1

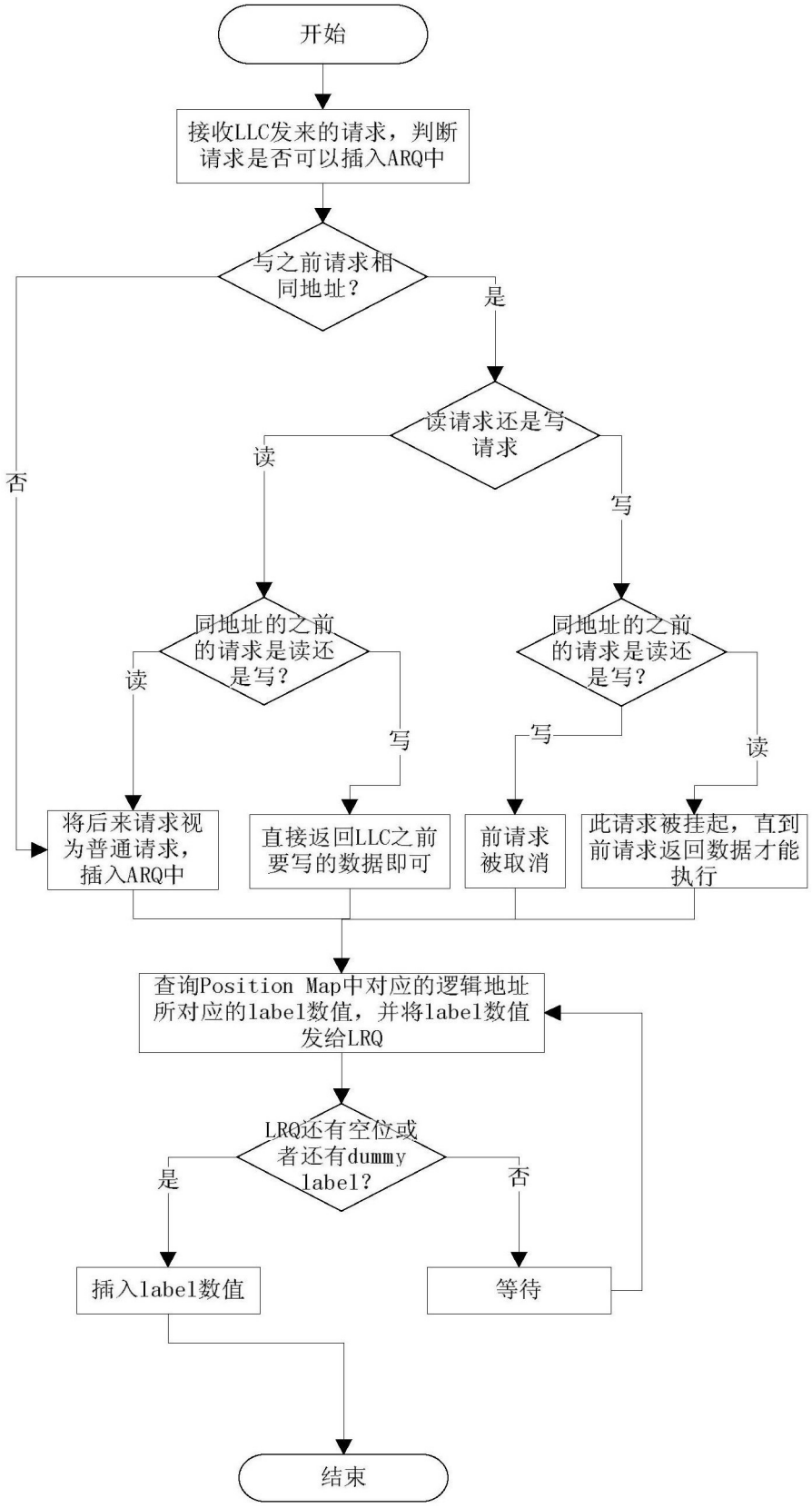
开始
接收LLC发来的请求，判断请求是否可以插入ARQ中
与之前请求相同地址？
是
否
读请求还是写请求
读
写
同地址的之前的请求是读还是写？
同地址的之前的请求是读还是写？
读
写
写
读
将后来请求视为普通请求，插入ARQ中
直接返回LLC之前要写的数据即可
前请求被取消
此请求被挂起，直到前请求返回数据才能执行
查询Position Map中对应的逻辑地址所对应的label数值，并将label数值发给LRQ
LRQ还有空位或者还有dummy label？
是
否
插入label数值
等待
结束

图2

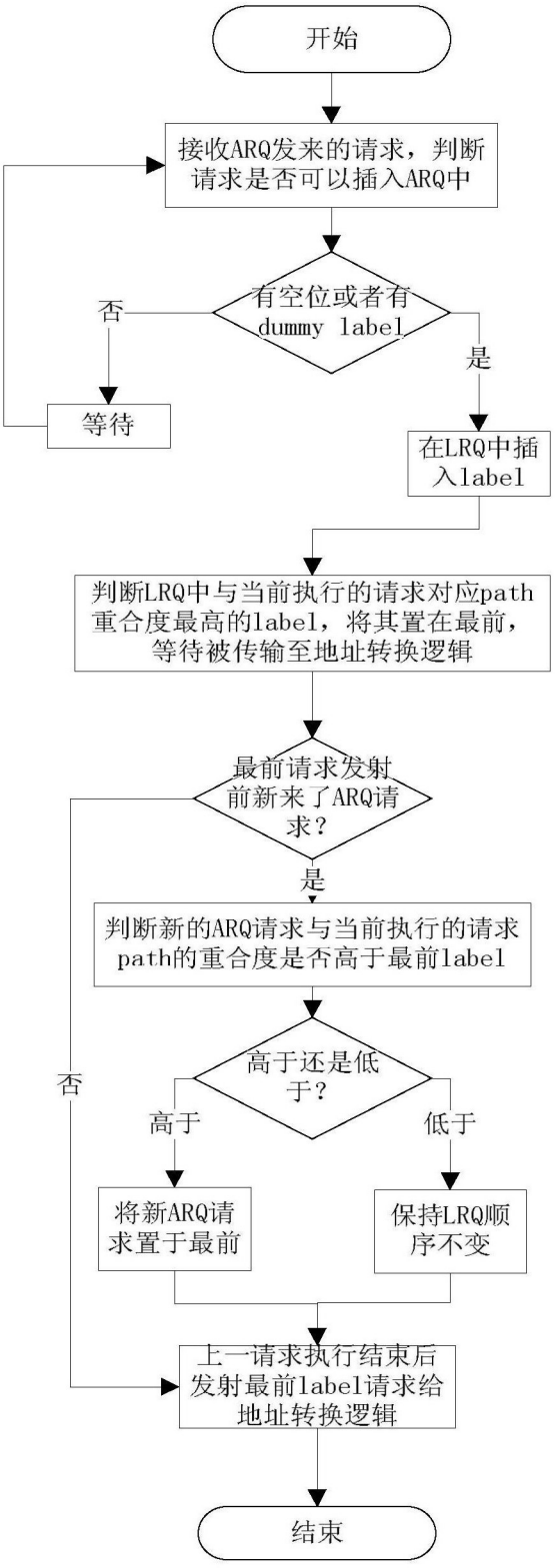
开始
接收ARQ发来的请求，判断请求是否可以插入ARQ中
有空位或者有dummy label
否
是
等待
在LRQ中插入label
判断LRQ中与当前执行的请求对应path重合度最高的label，将其置在最前，等待被传输至地址转换逻辑
最前请求发射前新来了ARQ请求？
是
否
判断新的ARQ请求与当前执行的请求path的重合度是否高于最前label
高于还是低于？
高于
低于
将新ARQ请求置于最前
保持LRQ顺序不变
上一请求执行结束后发射最前label请求给地址转换逻辑
结束

图3

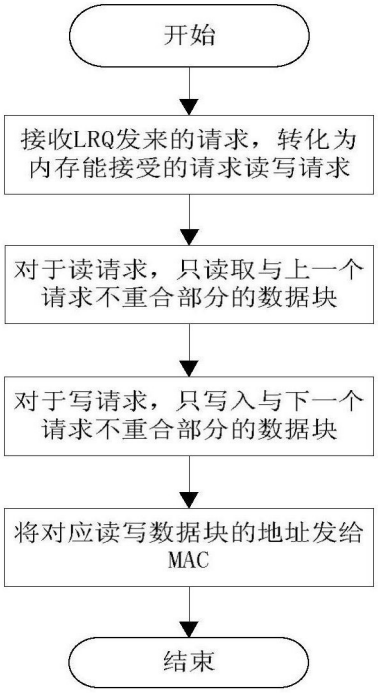
开始
接收LRQ发来的请求，转化为内存能接受的请求读写请求
对于读请求，只读取与上一个请求不重合部分的数据块
对于写请求，只写入与下一个请求不重合部分的数据块
将对应读写数据块的地址发给MAC
结束

图4

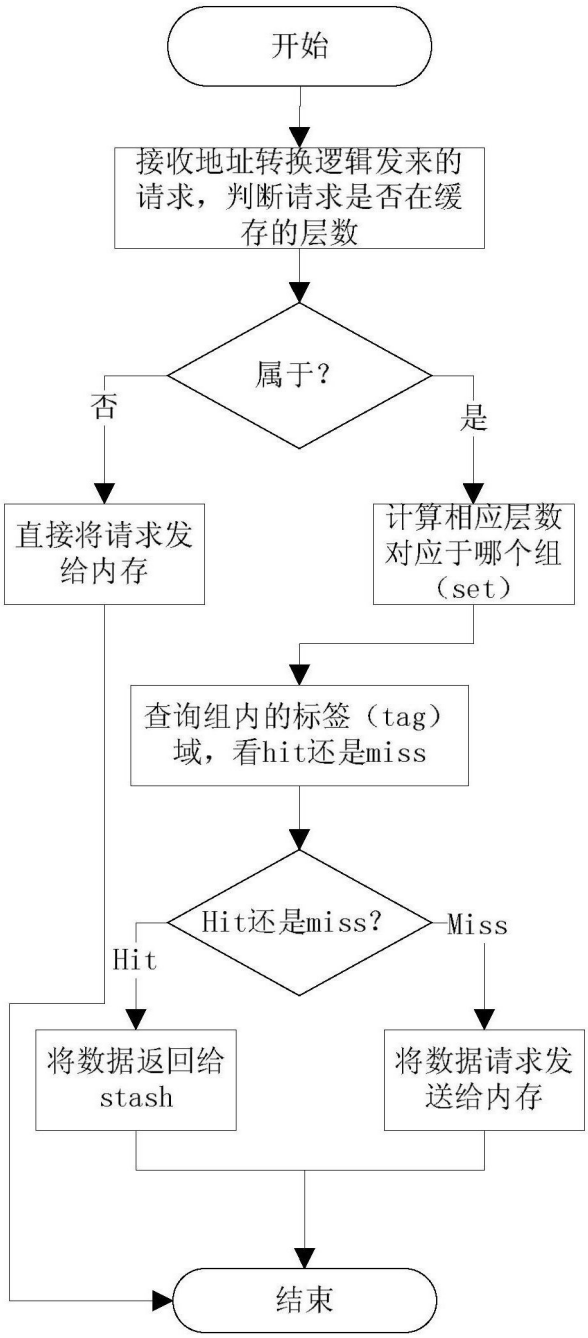
开始
接收地址转换逻辑发来的请求，判断请求是否在缓存的层数
属于?
否
是
直接将请求发给内存
计算相应层数对应于哪个组（set）
查询组内的标签（tag）域，看hit还是miss
Hit还是miss?
Hit
Miss
将数据返回给stash
将数据请求发送给内存
结束

图5

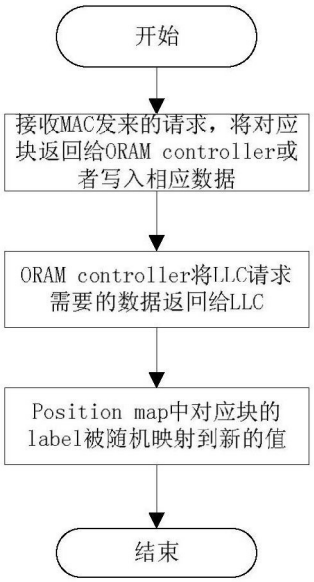
开始
接收MAC发来的请求，将对应块返回给ORAM controller或者写入相应数据
ORAM controller将LLC请求需要的数据返回给LLC
Position map中对应块的label被随机映射到新的值
结束

图6