|  |  |
| --- | --- |
| 文档编号 |  |
| 文档版本 |  |
| 文档管控 | 内部公开 |
| 存档日期 |  |

XX项目XX模块AS

|  |  |
| --- | --- |
| 编 写： |  |
| 校 对： |  |
| 审 核： |  |
| 批 准： |  |

昆明湖V1项目

2024年XX月XX日

文档修订记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **版本编号** | **变化状态** | **变更说明** | **作者** | **日期** |
| 1 | V0.1 | C |  |  |  |
|  |  |  |  |  |  |
| 3 |  |  |  |  |  |
|  |  |  |  |  |  |

\*变化状态：C—创建，A—增加，M—修改，D—删除

文档审批信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **审核** | **会签** | **批准** | **备注** |
| V1.0 |  |  |  |  |
|  |  |  |  |  |

目 录

[1 简介 1](#_Toc156311333)

[1.1 文档介绍 1](#_Toc156311334)

[1.2 参考文档 1](#_Toc156311335)

[1.3 术语说明 1](#_Toc156311336)

[1.4 技术背景 1](#_Toc156311337)

[2 设计规格 1](#_Toc156311338)

[3 功能描述 2](#_Toc156311339)

[4 总体设计 5](#_Toc156311340)

[4.1 整体框图 5](#_Toc156311341)

[4.2 接口列表 6](#_Toc156311342)

[4.3 接口时序 7](#_Toc156311343)

[4.4 时钟复位 8](#_Toc156311344)

[4.5 寄存器配置 8](#_Toc156311345)

[4.6 补充说明 9](#_Toc156311346)

[5 模块设计 9](#_Toc156311347)

[5.1 二级模块SMSPrefetcher 9](#_Toc156311348)

[5.1.1 功能 9](#_Toc156311349)

[5.1.2 整体框图 9](#_Toc156311350)

[5.1.3 接口列表 9](#_Toc156311351)

[5.1.4 接口时序 9](#_Toc156311352)

[5.1.5 关键电路 10](#_Toc156311353)

[5.1.6 三级模块SMSTrainFilter 10](#_Toc156311354)

[5.1.7 三级模块ActiveGenerationTable 10](#_Toc156311355)

[5.1.8 三级模块PatternHistoryTable 12](#_Toc156311356)

[5.1.9 三级模块PrefetchFilter 12](#_Toc156311357)

[5.2 二级模块L1Prefetcher 13](#_Toc156311358)

[5.2.1 功能 13](#_Toc156311359)

[5.2.2 整体框图 13](#_Toc156311360)

[5.2.3 接口列表 13](#_Toc156311361)

[5.2.4 接口时序 13](#_Toc156311362)

[5.2.5 三级模块TrainFilter 13](#_Toc156311363)

[5.2.6 三级模块StrideMetaArray 15](#_Toc156311364)

[5.2.7 三级模块StreamBitVectorArray 18](#_Toc156311365)

[5.2.8 三级模块MutiLevelPrefetchFilter 23](#_Toc156311366)

[6 PPA 28](#_Toc156311367)

[7 验证关注点 28](#_Toc156311368)

[8 Floorplan 建议 28](#_Toc156311369)

[9 遗留问题 28](#_Toc156311370)

1. 简介
   1. 文档介绍

本文档描述昆明湖数据预取器的功能与设计。

主要用于指导详细设计与验证。

* 1. 参考文档

1. PLRU替换算法
2. [Spatial Memory Streaming论文](https://infoscience.epfl.ch/record/112674/files/spatial_memory.pdf)
3. [Best Offset Prefetch论文](https://inria.hal.science/hal-01254863/document)
4. [Stride预取论文](https://www.computer.org/csdl/api/v1/periodical/mags/co/1978/12/01646791/13rRUx0xPD0/download-article/pdf)
5. Stream预取论文
6. [Feedback Direct Prefetch论文](https://users.ece.cmu.edu/~omutlu/pub/srinath_hpca07.pdf)
   1. 术语说明

表1.1 术语说明

|  |  |  |
| --- | --- | --- |
| **缩写** | **全称** | **描述** |
| MemBlock | Memory Block | 访存单元 |
| ROB | Reorder Buffer | 重排序缓存 |
| Dcache | Data cache | 数据缓存 |
| TLB | Translation Lookaside Buffer | 地址转译后备缓冲器 |
| pc | Program counter | 程序计数器 |
| SMS | Spatial Memory Streaming | 空间内存流 |
| BOP | Best-offset prefetching | 最优偏移预取 |
| FIFO | First In First Out | 先进先出 |
| MSHR | Miss-status Handling Registers | Miss请求处理寄存器 |
| CAM | Content Addressable Memory | 内容寻址存储器 |
| AGT | ActiveGenerationTable | 活动生成表 |
| PHT | PatternHistoryTable | 模式历史表 |
| Vaddr | Virtual Address | 虚拟地址 |
| VIPT | Virtually indexed, physically tagged | 虚地址索引，物理地址tag |

* 1. 技术背景

1. 设计规格
2. 支持虚拟地址空间预取
3. 支持可配置的预取器参数
4. 支持从load/store流水线获取训练数据
5. 支持覆盖跨步型Stride访问模式
6. 支持覆盖连续型Stream访问模式
7. 支持覆盖区域内具有空间局部性的不规则访问模式
8. 支持将数据块预取到Dcache/L2 Cache/L3 Cache
9. 功能描述
10. 功能概述： 数据预取器模块负责提前取回访存指令所需要的数据，以覆盖较高的内存读写延迟，包括从内存读数的load指令和向内存写数的store指令。预取器通过从访存流水线处获取信息进行训练，学习访存模式，预测访存地址，将预取请求发给Dcache或者下级Cache，将数据提前取回。
11. 分特性详细描述：
    1. 特性1：获取训练数据，预处理

为了寻找到一段程序中的访存规律，需要抓取访存历史信息，并对信息进行一定的处理。训练数据从MemBlock的访存流水线处取得，具体包括访存指令的pc、物理地址、虚拟地址、是否命中Dcache、是否命中Dcache中的预取块等。

不同的预取器需要的训练数据不一定相同，如Stream预取器需要所有全局访问历史来训练，不区分是否命中Dcache，也不需要pc，而SMS预取器使用在Dcache中Miss与PrefetchHit的访问历史来训练，需要pc信息。PrefetchHit指命中的Cache块是由预取器预取上来的，将PrefetchHit也算作Miss是因为如果没有预取器的干预，访问这个Cache块应该是Miss。以Stride预取器为例，Stride流下只有将PrefetchHit考虑后预取器看到的才是完整的Stride流，才能达到100%的覆盖率。

对应MemBlock的两条或三条load/store流水线，每一条流水线在任意一个周期都有可能产生一个有效的训练数据。以两条load流水线配置下的MemBlock为例，一个周期下可能同时产生2个load训练数据，2个store训练数据。对于预取器而言，其训练本身有一些条件限制：

1. 为了降低硬件逻辑的复杂性，预取器通常一个周期只处理一个训练数据。
2. 以Cache块地址为基础的预取器（SMS、Stream、BOP）需要对训练数据做块地址过滤，而全地址预取器（Stride）则不需要对训练数据做块地址过滤。
3. 对训练数据顺序敏感的预取器（Stride、BOP）需要训练数据尽可能按照指令顺序排列，而对训练数据顺序不敏感的预取器（SMS、Stream）则没有这个限制。

TrainFilter的设计目的是尽量消除以上3个限制，TrainFilter一个周期接收多个从流水线处获取的训练数据，根据指令顺序（ROB位置的顺序）重新排列，放入到一个固定大小的FIFO队列中。TrainFilter会对Cache块地址进行过滤，如果同一个周期中的多个训练数据的虚拟地址位于同一个Cache块内，则只有指令顺序最老的那一个请求会进入FIFO队列。如果同一个周期中的多个训练请求与在FIFO队列中已有的请求存在块地址相同的情况，则对应的训练请求会被丢弃。FIFO队列按照入队顺序，每一个周期移出一个训练数据给预取器。

* 1. 特性2：训练特定算法下的预取器

接收到经过处理后的训练数据之后，根据不同的预取算法，训练预取器，识别访存模式。

程序拥有空间与时间局部性，每条访存指令的pc不相同，从是否利用pc信息进行训练可以将预取器划分为两类。第一类是不使用pc的全局预取器（BOP，Stream），第二类是使用pc的预取器（SMS，Stride）。

BOP利用程序的全局空间局部性，在发生Dcache miss的地址中寻找覆盖率最高和及时性最好的offset，一轮训练后会得到一个最好的offset，再利用这个offset进行预取（**可能需要详细一些描述，再改改描述**）。

Stream利用程序的全局空间局部性，在所有的load指令访存地址序列中寻找该地址序列是否能构成X、X+1、X+2…的规律（X为Cache块地址），如果可以则将预取器变为激活状态，利用当前访问到的流地址信息进行预取。

SMS利用程序不同pc的空间局部性，将虚拟地址空间分为大小相同的若干region，学习一个region中的Cache块访问模式，在开启新的region访问时复刻已经学习到的模式。（**可能需要详细一些描述，再改改描述**）。

Stride利用程序不同pc的空间局部性，检查同一个pc的load指令访存地址序列是否符合X、X+K、X+2K…的规律（X为Cache块地址，K为步长）。如果可以则按照步长信息进行预取。

* 1. 特性3：产生预取请求

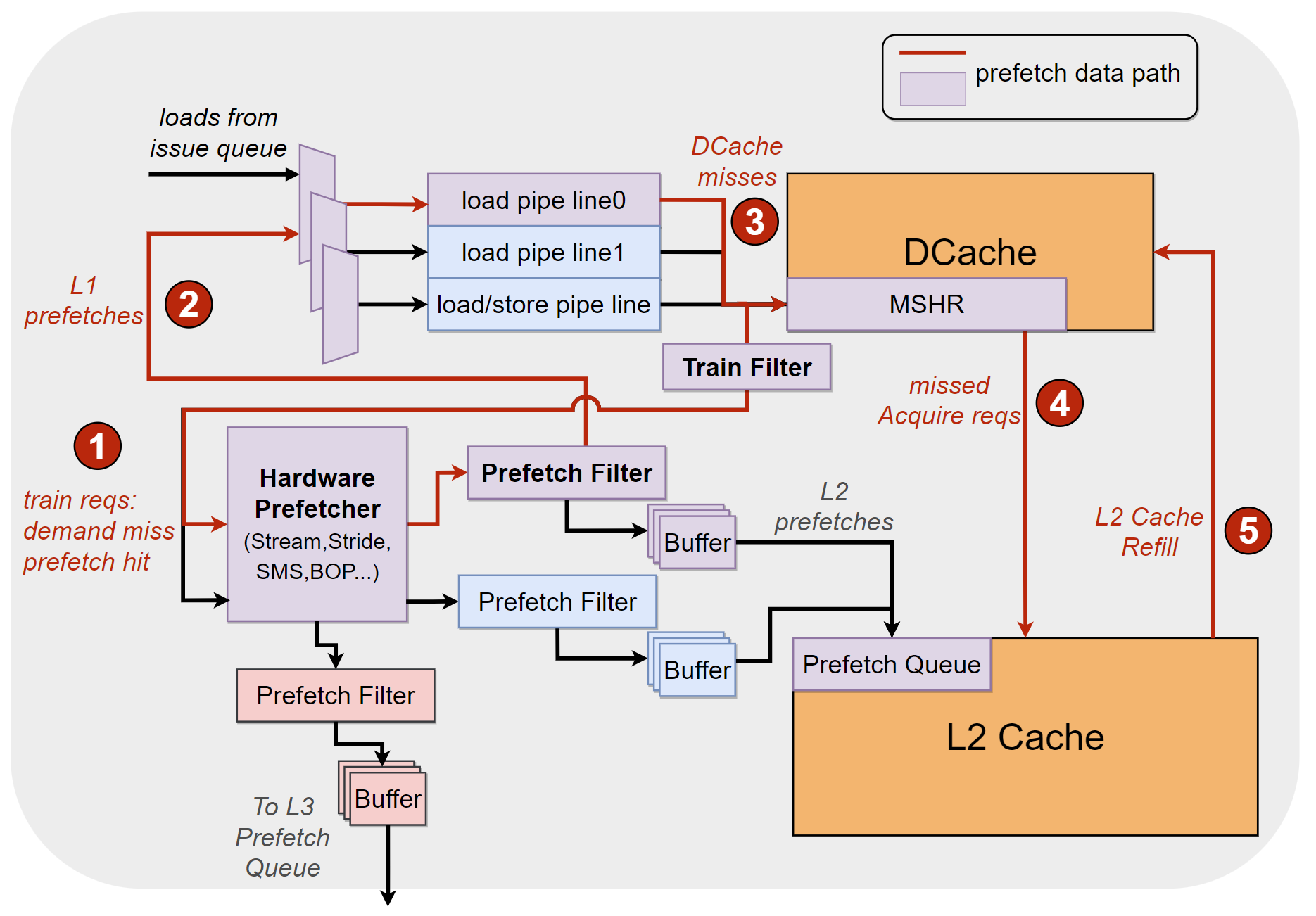
预取请求产生后，需要发给各级Cache以产生效果。例如发送给Dcache的预取请求如果没有命中Dcache则由Dcache负责将数据取回Dcache，可以达到对Dcache进行预取的效果，L2 L3 Cache原理类似。

预取器可以选择性地将预取请求发送到各级Cache，存在如下几个特性：

1. 昆明湖的数据预取器在虚拟地址空间进行训练，最终发出的预取请求也是虚拟地址空间的请求，而L2 L3 Cache只能处理物理地址空间的请求，需要在发给L2 L3 Cache之前将虚拟地址转换为物理地址。
2. Dcache的MSHR可以对请求按块地址过滤，L2 L3 Cache不会对请求按块地址过滤，如果大量给L2 L3 Cache发送重复的预取请求会造成明显的性能下降。
3. L2 L3 Cache的容量较大，Dcache容量较小，对于准确率较低的预取器，可以选择发给容错率高的下级缓存，对于准确率高的预取器可以选择发给Dcache。

Prefetch Filter可以对预取请求进行过滤，并进行虚实地址转换，最终将请求发给各级Cache。Prefetch Filter每周期接收一个预取请求，预取请求的信息包含region地址，region内需要预取的Cache块。Prefetch Filter内部以region为单位进行组织，内部是一个Buffer，每一个预取请求占用一项。接收到预取请求如果与Buffer内已有的请求重复则进行过滤，任意一个预取请求在没有得到物理地址前不能向Cache发送预取请求，Filter内部的预取请求与MemBlock的TLB进行交互，进行虚实地址转换。在完成转换得到物理地址后，依次将region内需要预取的块地址发出。

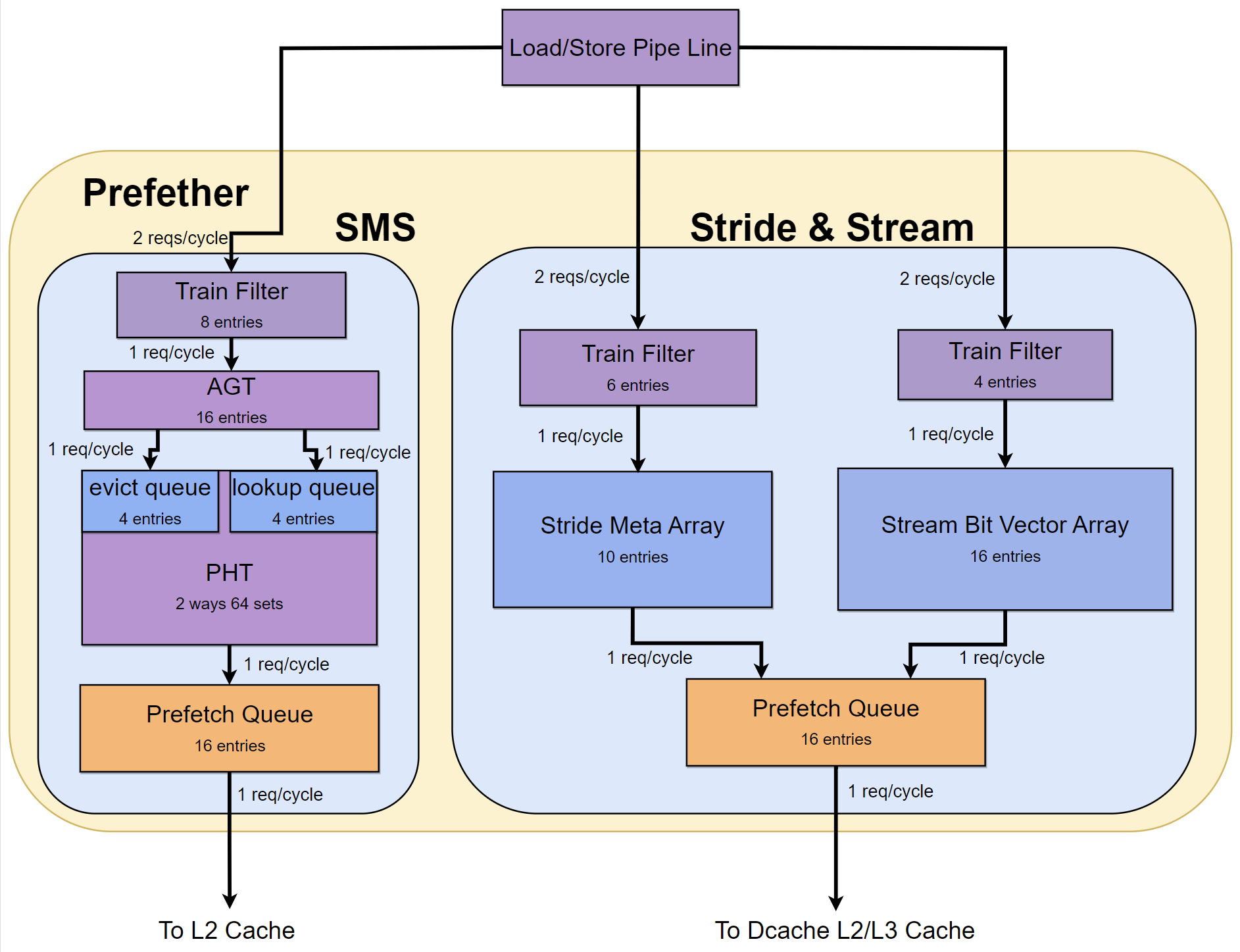
1. 总体设计
   1. 整体框图



上图为预取器与其他相关模块的交互示意图，展示了预取器产生一个L1 Dcache预取请求的流程：

1. 预取器从访存流水线处拿到训练数据。
2. 训练预取器，预取器发出预取请求到访存流水线。
3. 预取请求查询Dcache，结果为miss，分配Dcache MSHR。
4. MSHR向L2 Cache发送Acquire请求。
5. L2 Cache取回数据，回填至Dcache。

对L2 L3 Cache进行预取的流程与以上流程类似，不同的是请求发送给L2 L3 Cache的Prefetch Queue，再由Cache处理，详细内容可以查看对应的验证文档。



上图为内部各模块的关系图。

* 1. 接口列表

见【腾讯文档】Prefetcher模块IO

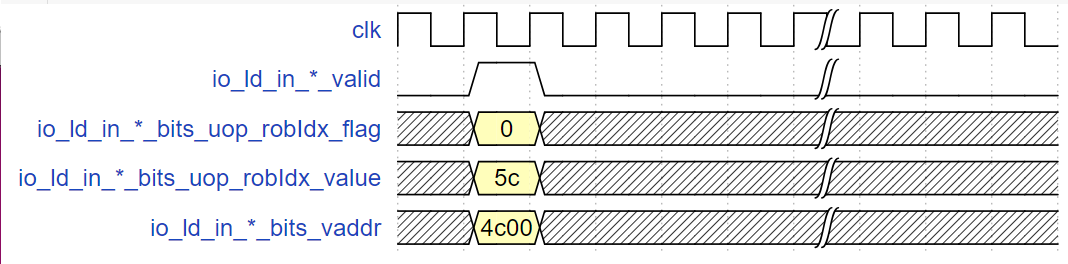
https://docs.qq.com/sheet/DUmJrb3dRRlBjcVlv?tab=2p1yej

表4.1 XXXXIO接口列表

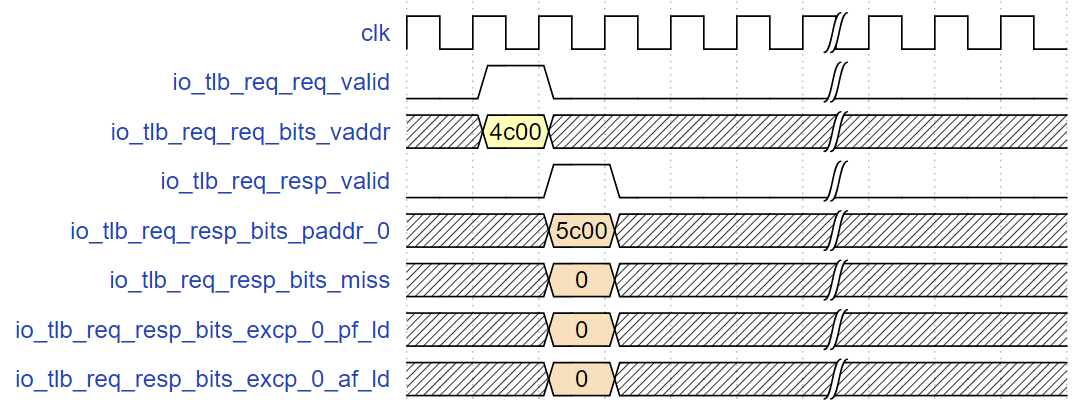
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **信号** | **方向** | **源** | **目的** | **信号组名** | **位宽** | **描述** |
| Clock | input |  |  |  | 1 | 时钟信号 |
| Reset | input |  |  |  | 1 | 异步复位信号 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

* 1. 接口时序

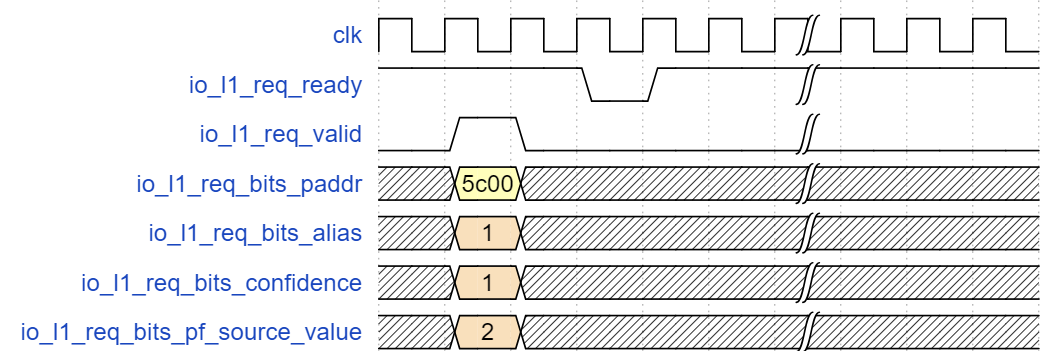
接收Stream训练请求时序，在io\_ld\_in\_\*\_valid为高时收到有效的Stream训练数据



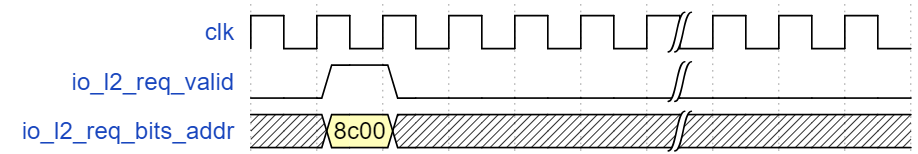
TLB 时序，在io\_tlb\_req\_req\_valid为高时发出有效的TLB转换请求，地址为io\_tlb\_req\_req\_bits\_vaddr，在发出TLB请求下一拍可以得到TLB响应，io\_tlb\_req\_resp\_valid为高代表收到TLB响应。



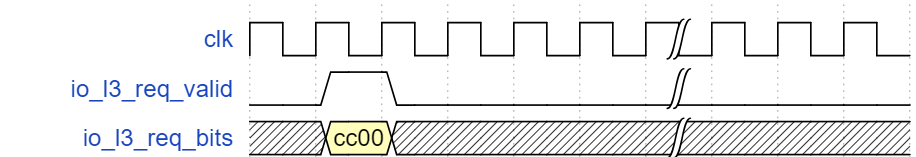
L1 预取时序，在io\_l1\_req\_ready & io\_l1\_req\_valid同时为高时，发给L1 Cache的预取请求握手成功。



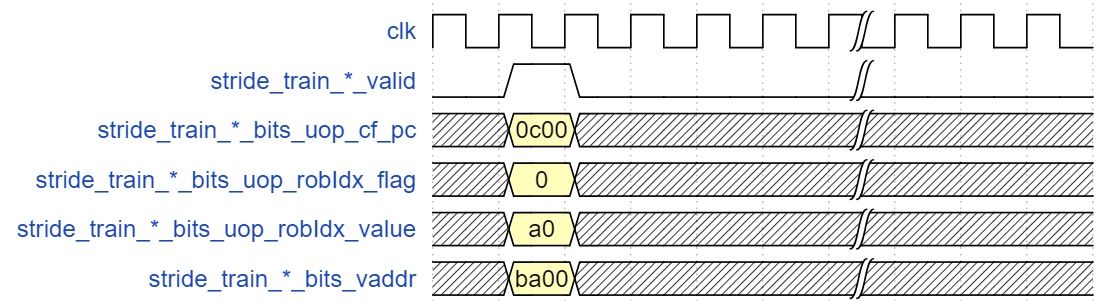
L2 预取时序，在io\_l2\_req\_valid为高时，发给L2 Cache的预取请求握手成功



L3 预取时序，在io\_l3\_req\_valid为高时，发给L3 Cache的预取请求握手成功



接收Stride训练请求时序，在stride\_train\_\*\_valid为高时收到有效的Stride训练数据



* 1. 时钟复位

|  |  |  |
| --- | --- | --- |
| Module | Clock | Reset |
| SMSPrefetcher | Clock | Reset |
| SMSTrainFilter | Clock | Reset |
| ActiveGenerationTable | Clock | Reset |
| PatternHistoryTable | Clock | Reset |
| PrefetchFilter | Clock | Reset |
| L1Prefetcher | Clock | Reset |
| TrainFilter | Clock | Reset |
| StrideMetaArray | Clock | Reset |
| StreamBitVectorArray | Clock | Reset |
| MutiLevelPrefetchFilter | Clock | Reset |

* 1. 寄存器配置

表 CSR配置寄存器说明

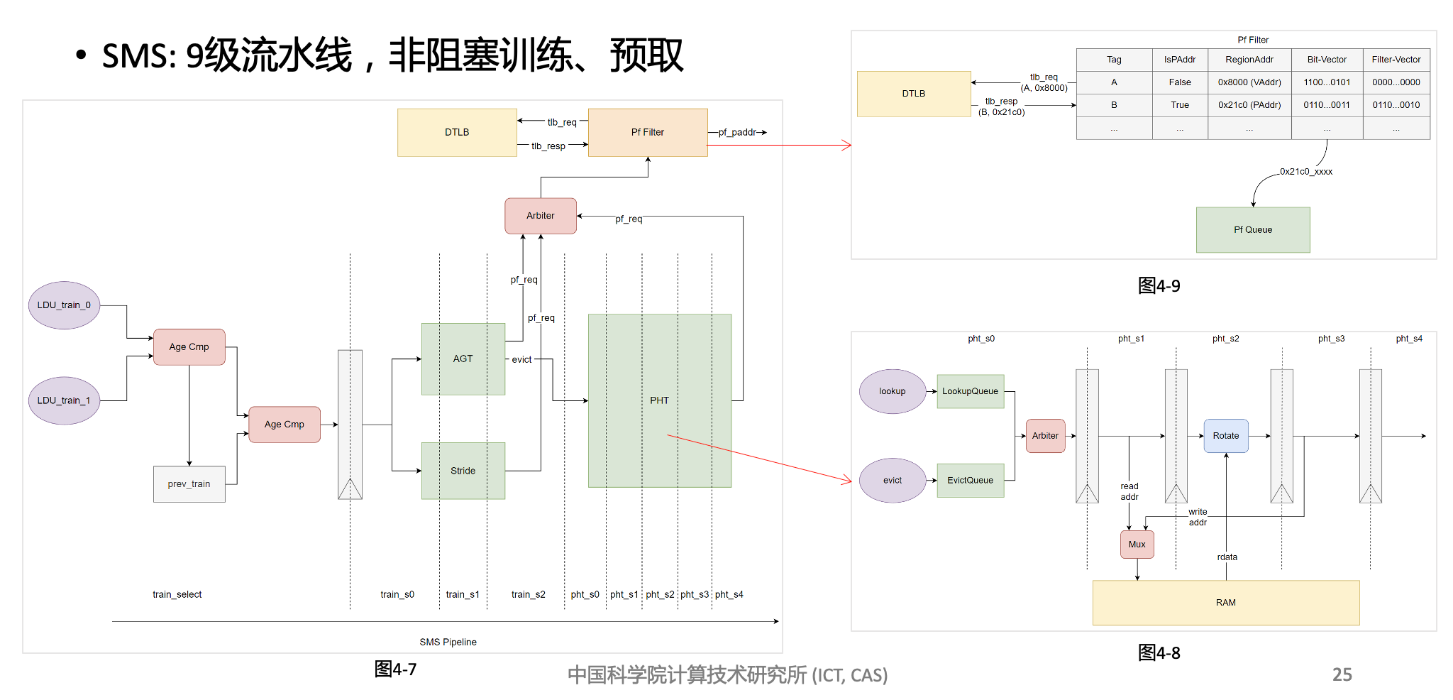
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **寄存器** | **地址** | **复位值** | **属性** | **描述** |
| spfctl | 0X5C1 | 32’d97078 | RW | bit0: ICache预取器开关  bit1: L2Cache预取器开关  bit2: L1 DCache预取器开关  bit3: L1 DCache预取器在Prefetch Train下训练  bit4: SMS AGT开关  bit5: SMS PHT开关  bit9-6: SMS AGT active阈值  bit10-15: SMS AGT active page stride  bit63-16: 保留 |

* 1. 补充说明

1. 模块设计
   1. 二级模块SMSPrefetcher
      1. 功能

实现SMS

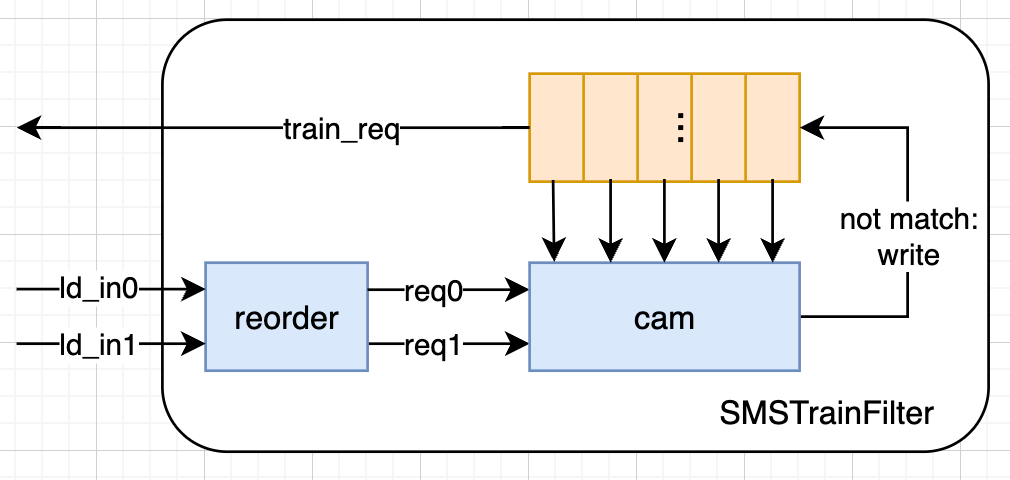
* + 1. 整体框图



* + 1. 接口列表
    2. 接口时序
    3. 关键电路
    4. 三级模块SMSTrainFilter
       1. 功能

对预取请求进行合并，并存在队列中。合并过程中，新来的预取请求按照robIdx升序排序，顺序写入队列。若新来的预取请求有效且在队列中存在相同项，则丢弃该请求，否则写入队列。其中“相同”判断标准为两者 block\_hash\_tag相同。

* + - 1. 整体框图



* + - 1. 接口列表

#TODO：先更新在excel中

* + - 1. 接口时序

简述：接收请求、重排序、CAM并写回均在一周期内完成。

详述：#TODO

* + - 1. 关键电路
    1. 三级模块ActiveGenerationTable
       1. 功能
       2. 整体框图
       3. 接口列表
       4. 接口时序
       5. 关键电路

ActiveGenerationTable 用于记录当前存在于Cache中的块的访问记录，其中有16项AGTEntry全相联。

其中各项含义如下：

●pht\_index：PC[6:2]用于索引PHT项

●pht\_tag：PC[19:7]，用于标识PC位置，对应不同访存指令，区分不同PHT项

●region\_bits：region内的16个block独热码表示，其中对应的block为1，其他为0。

●region\_tag：表示是哪一个region

●region\_offset：标识region内的哪一个 block

●access\_cnt：该region访问次数，其为饱和计数器，4bit，最多记录15次。

●decr\_mode：用于标识这个Entry是否为负向，判断方式为：Entry被分配时如Array中仅存在当前Region相邻的下一个Region，不存在当前region和其上一个Region，则认为该Entry为负向，否则为正向。负向的理解为：存在于AGT中的一个region的上一个Region正在被访问中。

1.查找请求

若当前要查找的region为X，通过region\_tag 查找region X、X-1、X+1是否在AGT中。其中在查找X-1、X+1的时候需要注意当前请求是否允许跨region的访问。

如果replace和evict的是同一项，或当前需要读 AGT，则将 s0\_dcache\_evict拉低，延迟处理该请求（即ready位当拍拉低）。

访问时，如果当前region没有在AGT中找到，则新分配一项，分配位置由replacement提供的项决定；若当前region命中，则更新该项。

若找到了region X或X-1或X+1，则在 pf\_gen中记录该项。

2.访问更新：

如果当前region访问在AGT中命中，则在原项基础上region\_bits中添加当前region访问的region\_bits（即二者相与），若当前region访问的block，没有出现在原项的region\_bits中，则原项access\_cnt加1.

3.预取请求发送：

1.发送条件：

●prev\_lookup\_valid 即 s1\_lookup\_valid

●分配项，需要 X-1/X+1存在AGT中；更新项，X本身就在AGT中

●不允许超过地址限制

●访问次数需要超过一定阈值

●AGT全局使能

2.发送地址：如果对应的AGT项的decr\_mode为真，则当前虚拟地址的{region\_tag,region\_offset} 减去传入的act\_stride，否则加上act\_stride。得到新的{region\_tag, region\_offset}，再结合访问地址的高位，得到新的地址。  
为什么这么做呢？为什么不一次性到位？

3.

* + 1. 三级模块PatternHistoryTable
       1. 功能
       2. 整体框图
       3. 接口列表
       4. 接口时序
       5. 关键电路
    2. 三级模块PrefetchFilter
       1. 功能
       2. 整体框图
       3. 接口列表
       4. 接口时序
       5. 关键电路
  1. 二级模块L1Prefetcher
     1. 功能

L1Prefetcher是对Stride，Stream预取器各个组件的包装层。

* + 1. 整体框图

见4.1

* + 1. 接口列表

见4.2

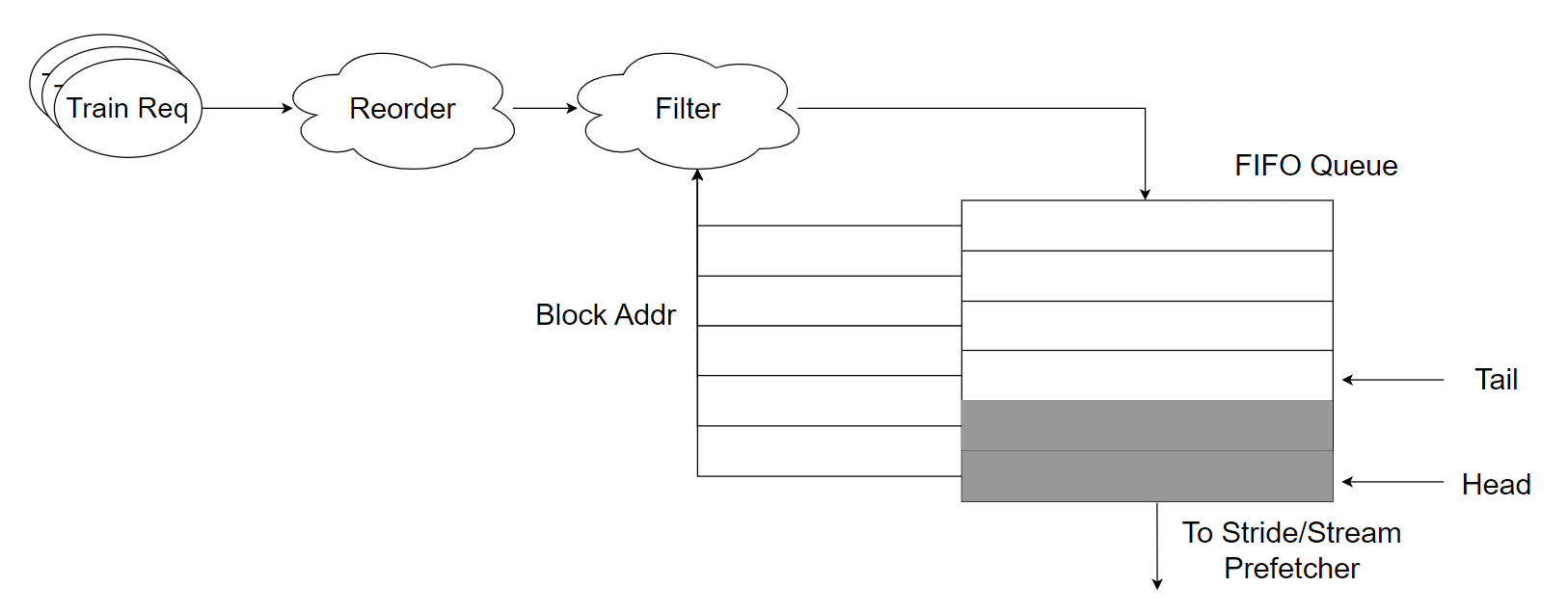
* + 1. 接口时序

见4.3

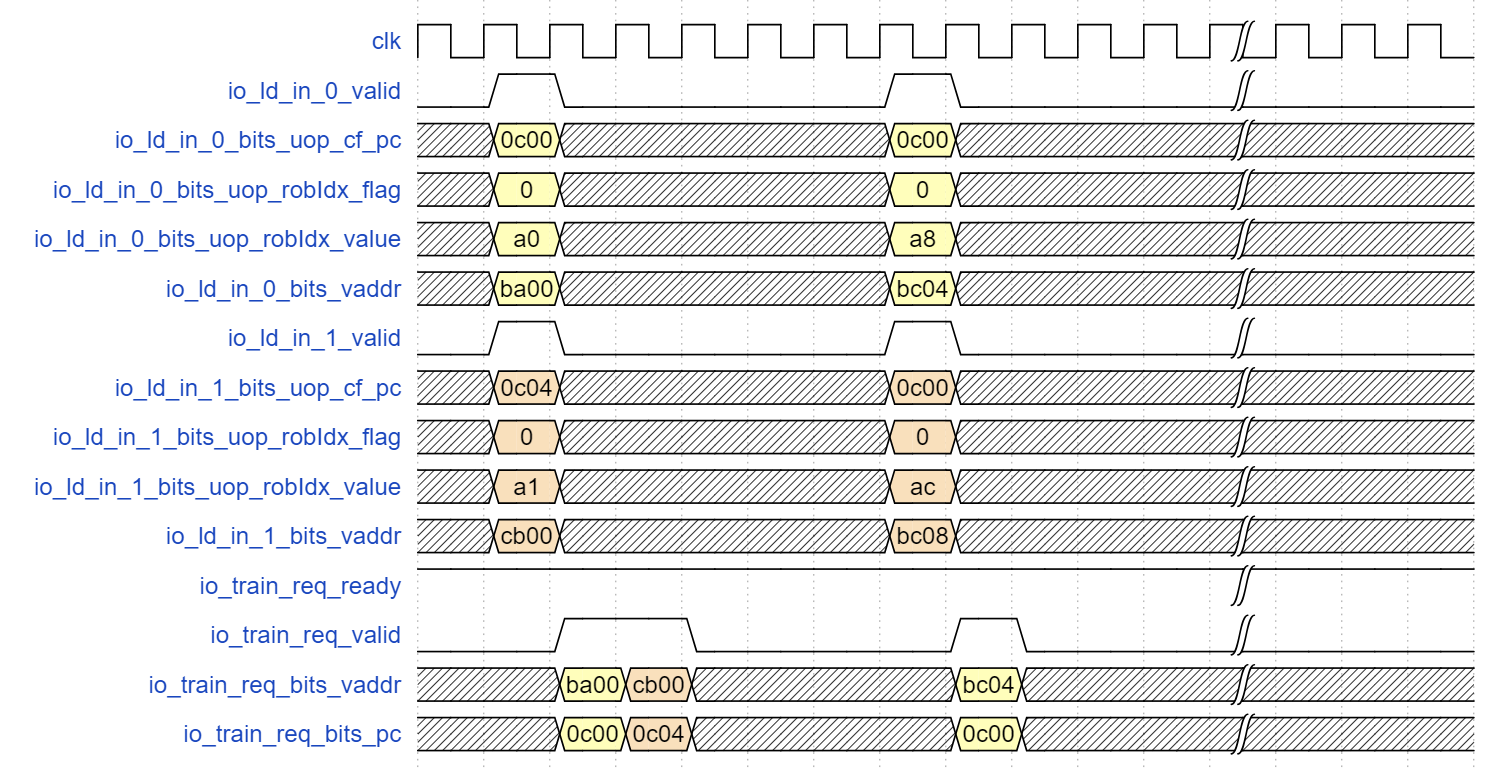
* + 1. 三级模块TrainFilter
       1. 功能

接收外部访存流水线发来的预取训练请求，一个周期接收多个训练请求，例如Stream预取器一个周期接收N条Load流水线的预取训练请求，对这些请求进行预处理如块地址过滤，每一个周期向外发出一个预取训练请求给预取器训练。

* + - 1. 整体框图



* + - 1. 接口列表
      2. 接口时序



上图中第2个上升沿接收到2条load流水线的训练请求，pc分别为0xc00和0xc04，虚拟地址分别为0xba00和0xcb00，ROB顺序为第0个训练请求老于第1个训练请求。由于两个地址不在同一个Region，不需要过滤，按照ROB顺序入队，分别在第3、4个上升沿将训练请求发出给预取器。

第8个上升沿接收到2条load流水线的训练请求，pc都为0xc00，虚拟地址分别为0xbc04和0xbc08，ROB顺序为第0个训练请求老于第1个训练请求。由于两个地址在同一个Region，第1个训练请求被过滤掉，只有第0个训练请求入队，在第9个上升沿将训练请求发出给预取器。

* + - 1. 关键电路

TrainFilter是一个循环的FIFO队列，主要有两部分关键电路：

1. **入队：**

根据Load访存流水线数量确定入队数量，昆明湖最多拥有3条Load流水线，因此最多一个周期入队3条训练请求，3条训练请求按照ROB Index顺序从老到新重新排列。重排后的X、Y、Z三条请求依次判断是否能够入队：X查询TrainFilter中已有的请求，存在块地址相同则不入队，Y查询X和TrainFilter中的请求，存在块地址相同则不入队，Z查询X、Y和TrainFilter中的请求，存在块地址相同则不入队。如果判断后只有X、Z可以入队，则按照X、Z的顺序插入到FIFO队列的尾部，如果FIFO队列已满，则不入队。

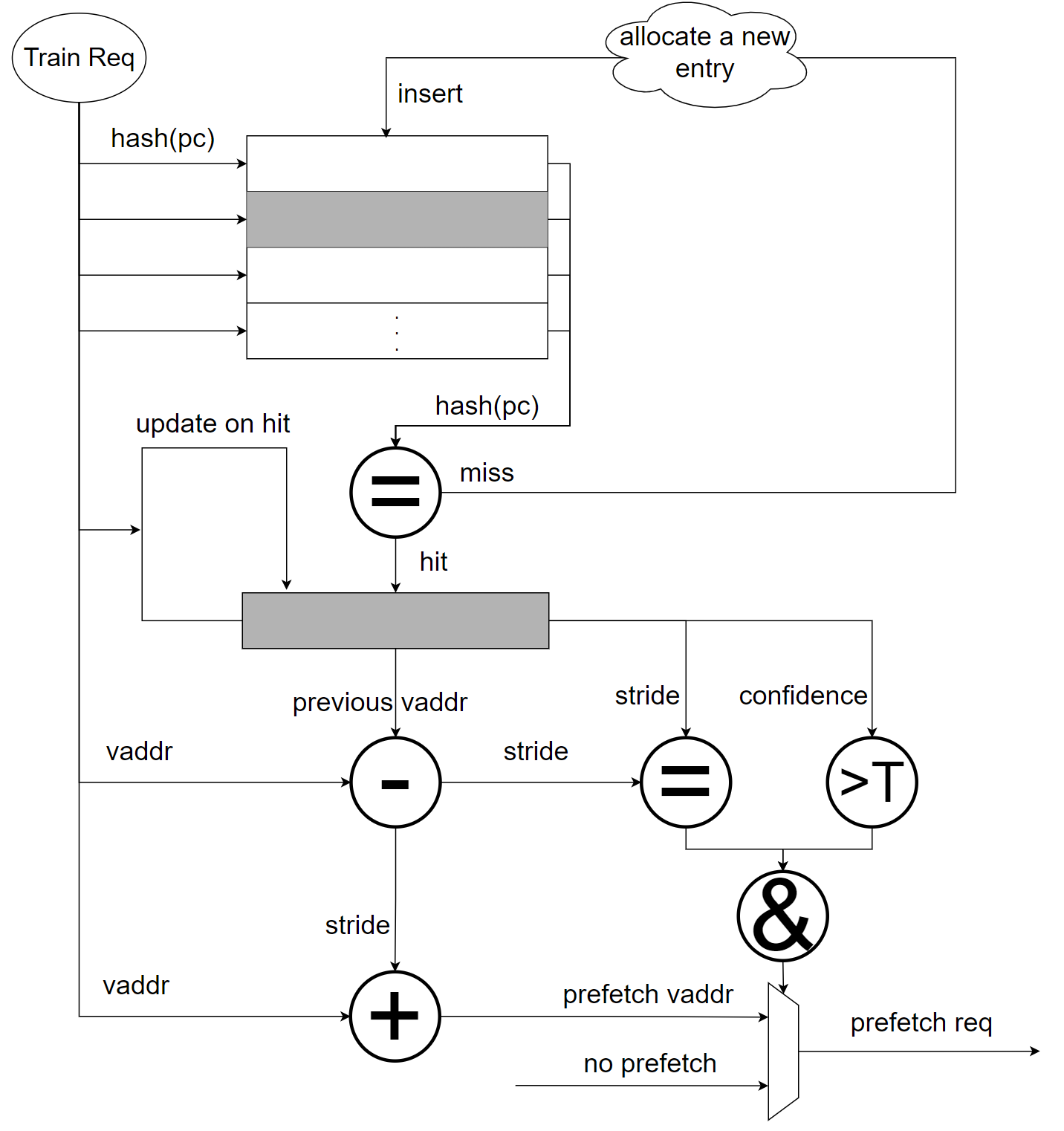
1. **出队：**

利用FIFO队列的头指针，当FIFO队列非空时，从队列头部每周期依次出队一条训练请求。

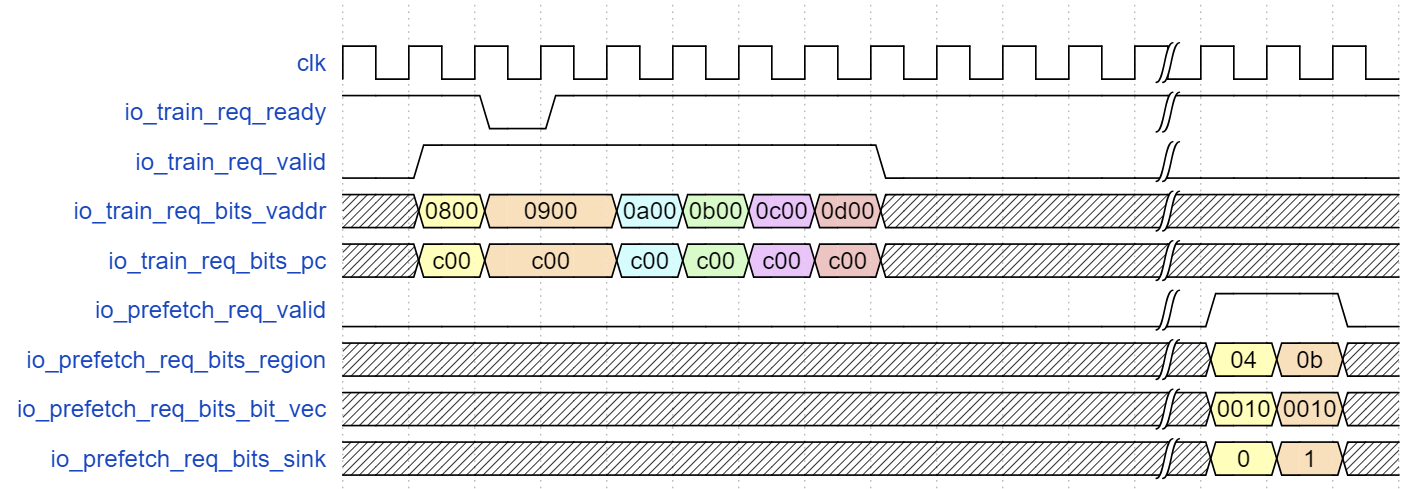
* + 1. 三级模块StrideMetaArray
       1. 功能

Stride预取器主体部分，负责检测X、X+K、X+2K…的访存规律，提前发出预取请求覆盖Stride访存模式。

* + - 1. 整体框图



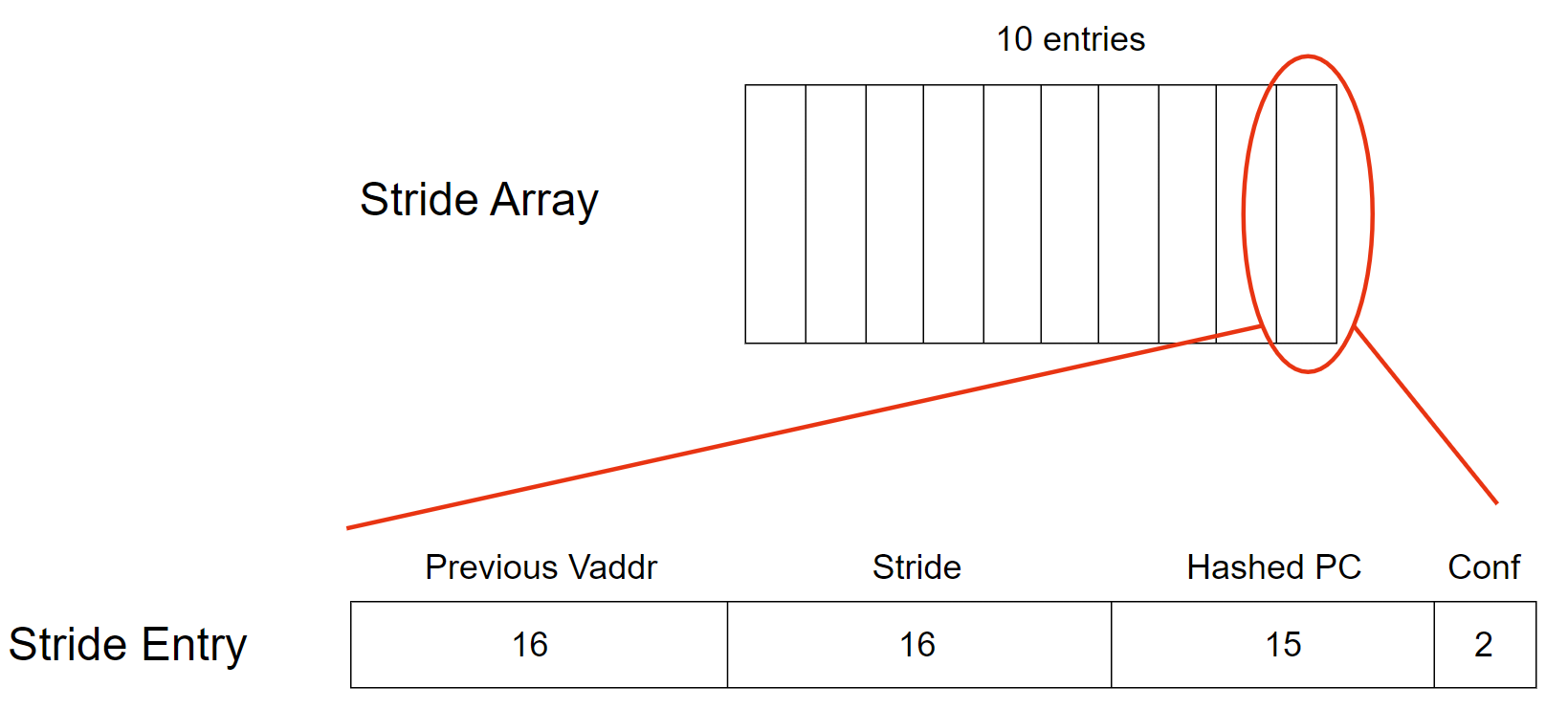
* + - 1. 接口列表
      2. 接口时序



上图从第2个上升沿连续到来pc为0xc00的6的训练请求，虚拟地址分别为0x800、0x900、0xa00、0xb00、0xc00、0xd00。Stride预取器学习到pc为0xc00的指令访存地址满足以0x100为步长的Stride序列模式。因此在地址为0xd00到来时触发两个预取请求，其中发给L1的预取请求为0xd00 + 4 \* 0x100，发给L2的预取请求为0xd00 + 32 \* 0x100。

* + - 1. 关键电路

Stride预取器中的主要存储结构为Stride Meta Array，其中一共有10项Stride Entry。



Stride Entry结构说明：

* Previous Vaddr：前一次访问的虚拟地址，存储全地址，包括块内偏移部分，即Vaddr[15:0]。
* Stride：学习到的Stride值，当前访问与前一次访问的全地址差值
* Hashed PC：经过哈希的PC值，用来标识不同的Load指令，理想情况下一个Hashed PC对应一条Load指令
* Conf：置信度，用来指示当前PC训练出的Stride的可信程度

Stride预取器主要有两部分关键电路：

1. **接收训练请求：**

Stride预取器的训练分为两级流水线s0和s1

S0：接收训练请求，对PC做哈希操作，得到一个16位的哈希后的PC值，与Stride Meta Array中的10项Stride Entry的Hashed PC匹配。如果有Stride Entry匹配成功则为命中，否则为miss。miss的情况下利用PLRU替换算法选出一个Entry。

S1：根据S0是否命中执行不同的操作：

* 命中：读出命中的Stride Entry中的Previous Vaddr，将当前虚拟地址减去Previous Vaddr得到新的Stride。新的Stride有效的条件为：Stride大于1个Cache块（小于或等于1的情况应该被Stream覆盖），Stride不是负数。当新的Stride与Entry中的Stride相同时，增加置信度；两者不相同时，如果Entry中的Stride置信度较低（小于等于1），则用新的Stride替换掉Entry中的Stride；如果Entry的置信度较高，则不替换Stride，只降低置信度。
* Miss：设置新的Entry的信息：Previous Vaddr为当前训练请求的虚拟地址低16位，Stride为0，Hashed PC为当前训练请求经过哈希操作后的PC，Conf为0。将新的Entry信息更新到s0阶段替换算法选出来的Entry中。

1. **发出预取请求：**

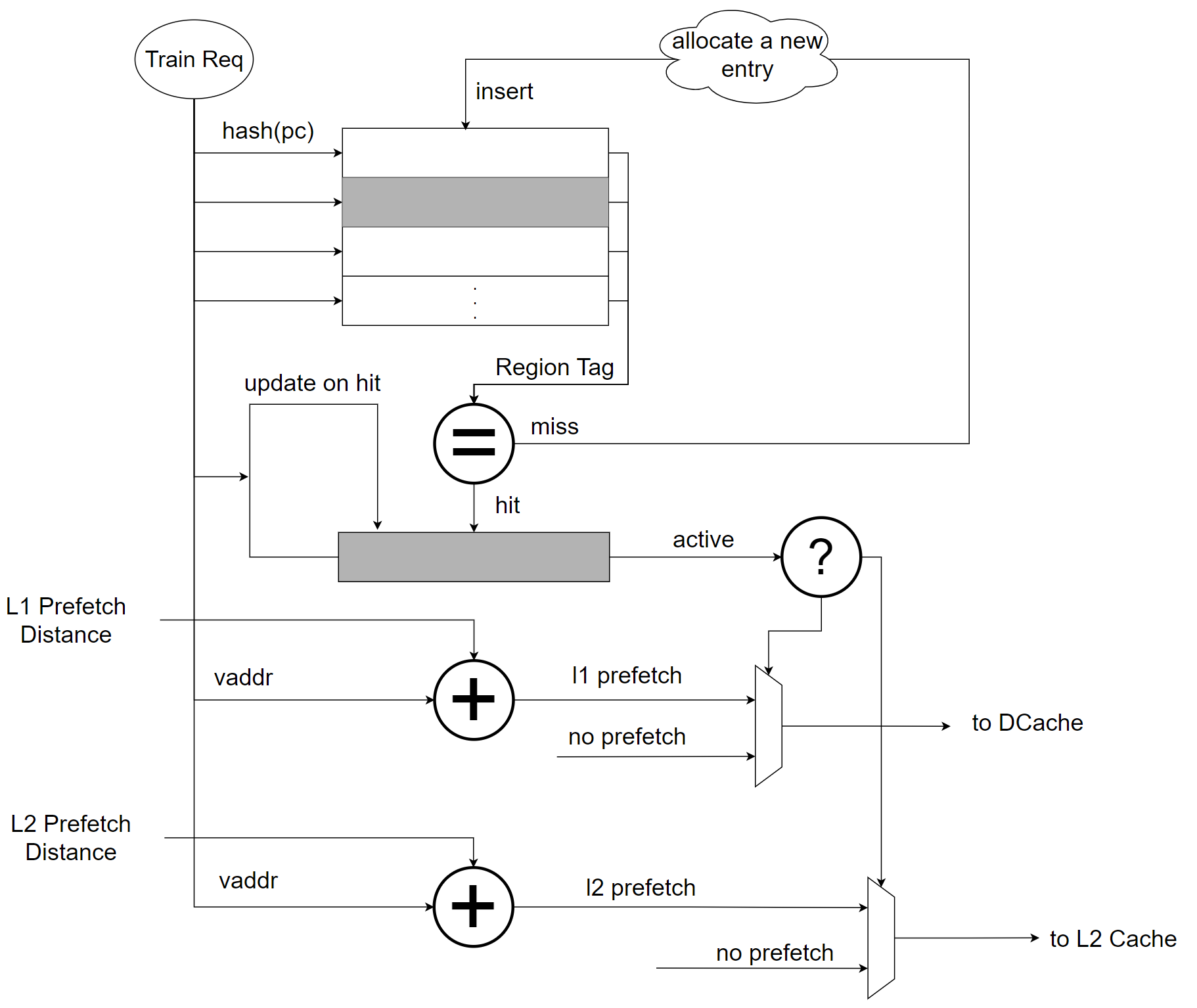
s2、s3和s4阶段用于产生和发送预取请求。s1阶段训练时，如果发现训练请求命中已有的Entry，且新的Stride与Entry中已有的Stride相同，且Entry的Stride置信度达到了最大值，就可以产生预取请求。s2阶段产生给L1 Dcache与L2 Cache的预取请求信息，分别在s3，s4阶段向外发出。

如果某一个PC对应的Stride Entry已经学习到这个PC对应的Load指令的访存模式符合X、X+K、X+2K…的规律。s2阶段的训练请求地址为Y，则产生地址为Y+4K的预取请求发送给L1 Dcache，产生地址为Y+32K的预取请求发送给L2 Cache。使用Y+4K而不是Y+K发给Dcache是为了提升预取请求的及时性。

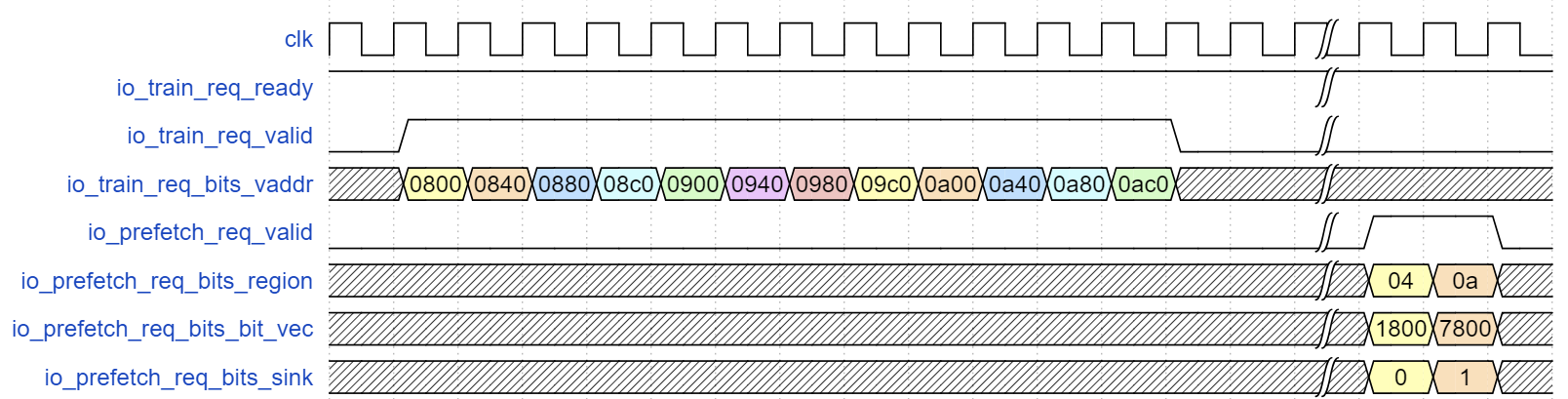
* + 1. 三级模块StreamBitVectorArray
       1. 功能

Stream预取器主体部分，负责检测X、X+1、X+2…的访存规律，提前发出预取请求覆盖Stream访存模式。

* + - 1. 整体框图



* + - 1. 接口列表
      2. 接口时序

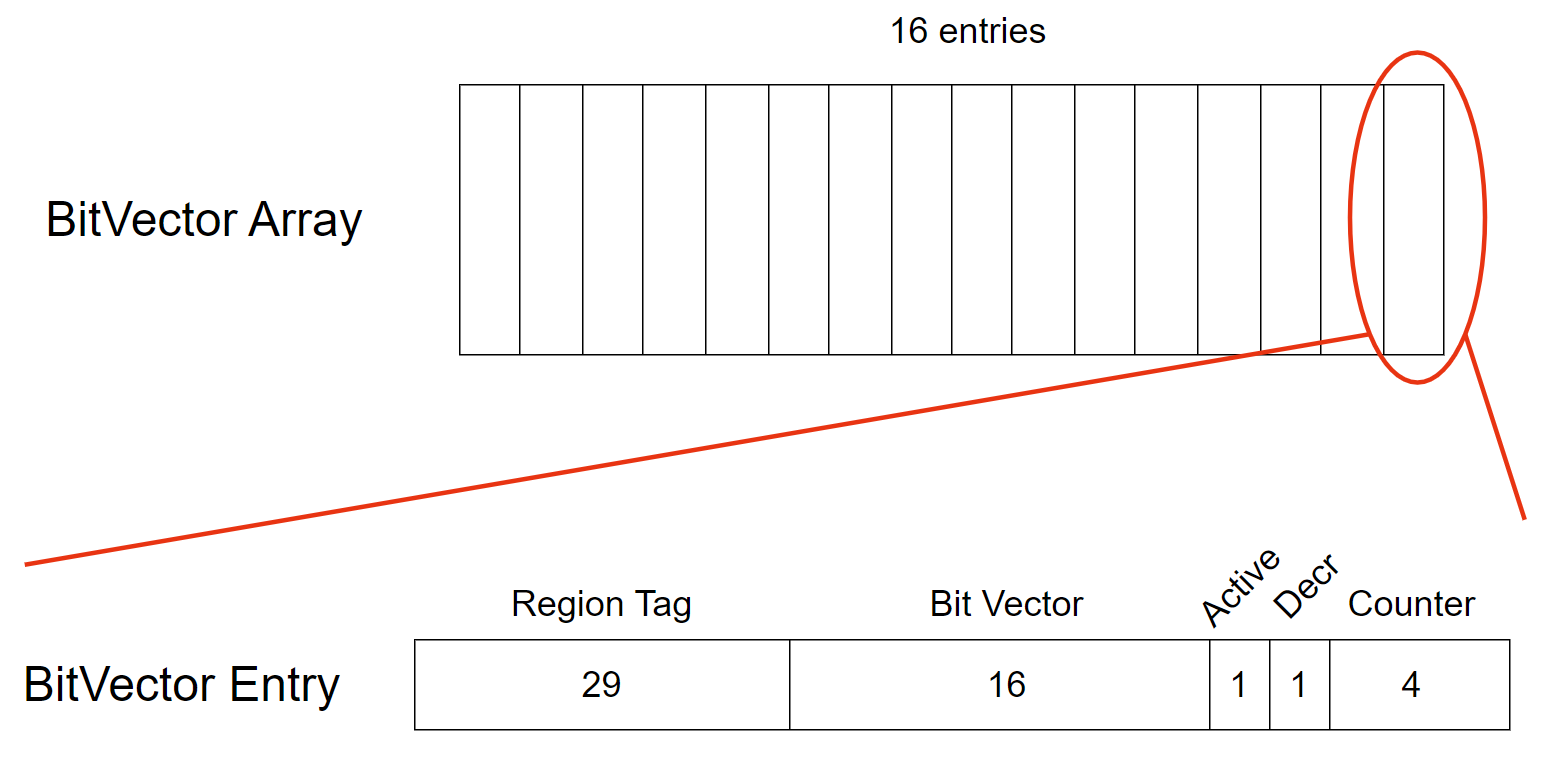


上图从第2个上升沿到来连续收到12个训练请求，从0x800到0xac0，地址依次递增0x40，Stream预取器训练后学习到此地址规律满足地址依次递增的Stream访存模式，在收到第12个训练请求时Stream预取器被激活，向L1发出基地址为0xac0 + 32 \* 0x40，长度为2个块的预取请求，向L2发出基地址为0xac0 + 128 \* 0x40，长度为4个块的预取请求

* + - 1. 关键电路

Stream预取器将虚拟内存空间均分为若干个大小为1KB的Region，类似于SMS预取器。

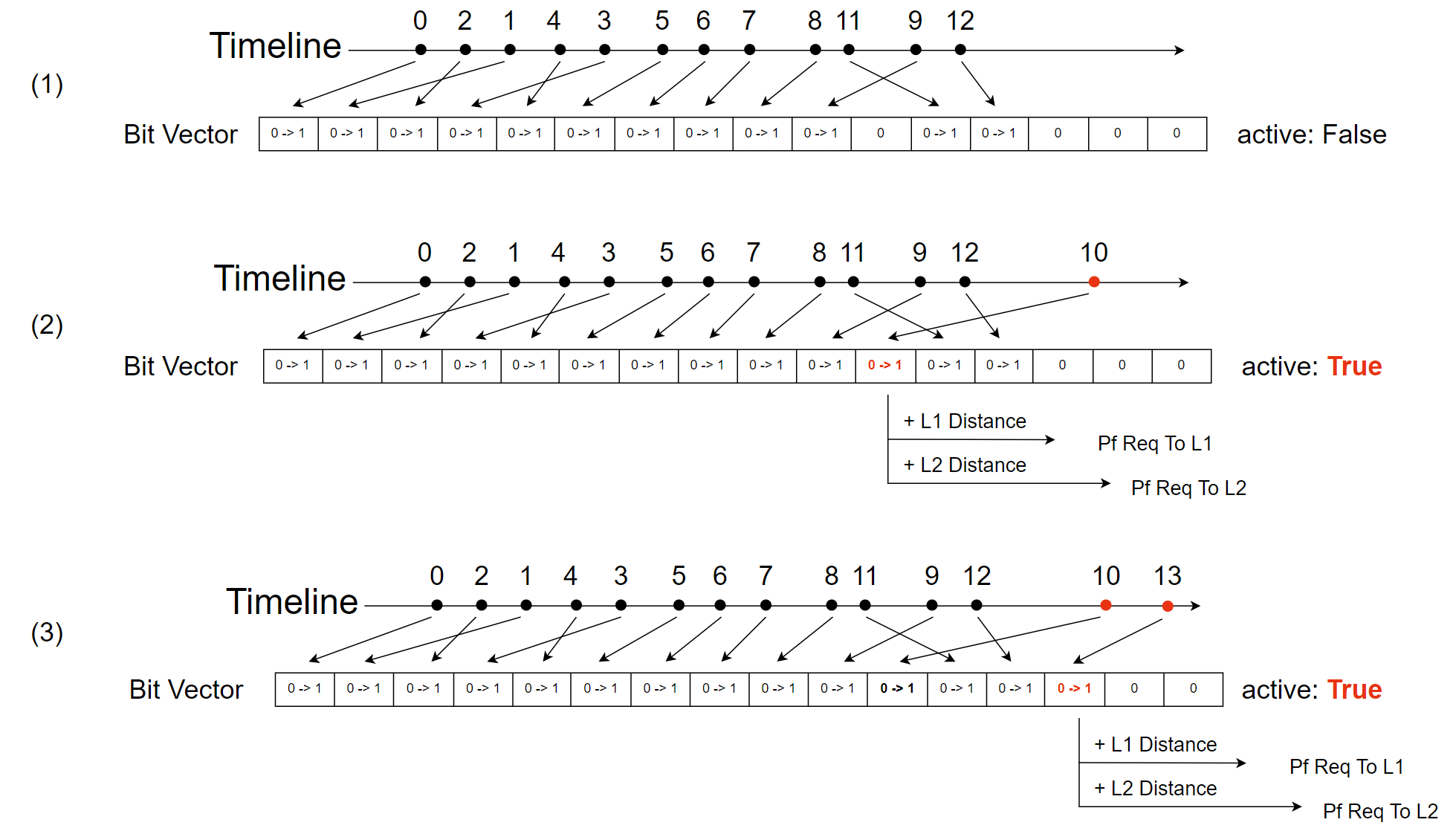
Stream预取器中的主要存储结构为Bit Vector Array，其中一共有16项Bit Vector Entry。



Bit Vector Entry结构说明：

* Region Tag：用于标识当前Entry属于哪一个Region，每一个Region有唯一的地址标识，具体为Vaddr[38:10]，Array中的不同Entry的Region Tag一定不相同。
* Bit Vector：用于标识当前1KB的Region内16个Cache块中哪些被访问过。被访问过标为1，否则为0。
* Active：用于标识这个Entry是否被激活，被激活状态下可以触发预取。两种情况下Entry会被标记为激活状态：
  + 这个Region中16个Cache块被访问的数量达到12个及以上
  + 这个Region存在相邻的Region，且相邻的Region处于激活状态
* Decr：用于标识这个Entry是否为负向Stream，判断方式为：Entry被分配时如果Array中存在当前Region相邻的下一个Region则认为该Entry为负向Stream，否则为正向Stream。
* Counter：用于记录Region中被访问过的Cache块数量，超过11个之后将Entry设置为激活状态。

Bit Vector Entry工作示意图如下：



时间轴上不同时间访问不同的Cache块，访问Region的第0个块记为0，第1个块记为1。上图（1）中一共包含12次访问，访问Region的第0个块时，分配一个新的Bit Vector Entry，并将Bit Vector中第0位从0标记为1，激活状态目前为false。访问Region的第2个块时，更新这一项Bit Vector，将第2位从0标记为1，激活状态目前为false。后续的1、4、3…等访问流程类似。

上图（2）时间轴标红处表明访问到Region的第10个块，将这个Bit Vector的第10位从0标记为1，此时Bit Vector为1的数量超过了阈值12，将Bit Vector的激活状态设置为True。以该Region的第10个块位置为基础，加上L1 Distance，预取2个块到Dcache，加上L2 Distance，预取4个块到L2 Cache。目前设置的L1 Distance为32个块，L2 Distance为128个块。

上图（3）新到来一个对Region的第13个块的访问，处理流程与以上描述类似。如果后续的连续访问超过了这个Region，到达下一个Region，则会新分配一项Bit Vector Entry，新分配的Entry自动变为激活状态。

Stream预取器主要有三部分关键电路：

1. **接收训练请求：**

Stream预取器的训练分为两级流水线s0和s1：

S0：从训练请求中获取虚拟地址，将虚拟地址转换为Region地址，并生成当前Region相邻的前后两个Region地址。三个Region地址并行查询BitVectorArray，比较每一个Entry的Region Tag，得到三个命中向量。如果BitVectorArray中已经存在相同Region的Entry，则为命中，否则为miss。如果miss则需要申请一项新的Entry，使用PLRU替换算法得到最近最不常被使用到的Entry。

S1：根据S0是否命中执行不同的操作

* 命中：计算本次训练请求位于Region中的哪一个Cache Block，并将Entry的Bit Vector中该位置标为1，表示被访问过。更新Entry中的Counter，如果这个Cache Block是第一次被访问，将Counter加1，当Counter达到12时，将Entry标记为激活状态。
* Miss：设置新Entry的信息，Region Tag为训练请求虚拟地址所在的Region；Bit Vector为One-Hot向量，1的位置为训练请求虚拟地址所在该Region的Cache块编号；Counter记为1；根据前文提到的正负向Stream检测方式设置Decr；根据相邻Regon是否处于激活状态来设置Active。将新的Entry信息更新到s0阶段替换算法选出来的Entry中。

1. **发出预取请求：**

s2、s3、s4、s5级流水线负责触发预取请求

s2阶段计算发出到L1/L2/L3 Cache的预取请求地址和相关的控制信息，具体为当前虚拟地址加上一个固定值D，D为Distance，L1 Distance为32个Cache块，L2 Distance为 128个Cache块，L3 Distance为256个Cache块。从预取地址开始连续预取N个块，N为Degree，L1 Degree为2，L2 Degree为4，L3 Degree为8。

举例来说，如果一个Region Entry学习到当前地址符合X、X+1、X+2…的规律，如果训练请求访问的地址为Y，则将地址为Y+32，Y+33的预取请求发送给Dcache，Y+128、Y+129、Y+130和Y+131的预取请求发送给L2 Cache，Y+256、Y+257、Y+258、Y+259、Y+260、Y+261、Y+262和Y+263的预取请求发送给L3 Cache。

一旦满足触发预取的要求（当前Region处于激活状态），s3，s4，s5级流水线分别发出L1/L2/L3级的预取请求（目前不启用L3预取）。

1. **Stride冲突避免：（昆明湖目前没有启用）**

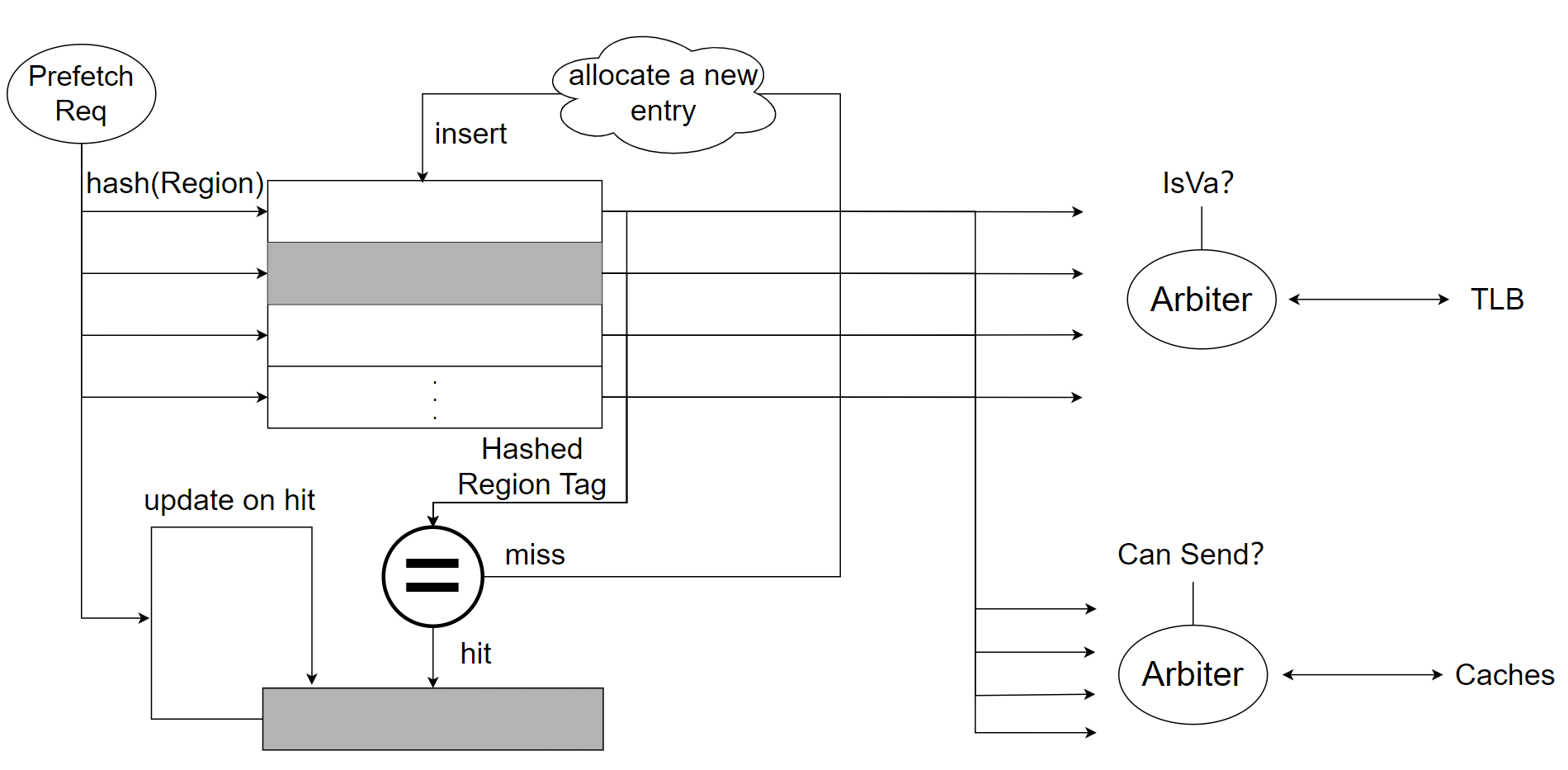
由于Stream是特殊的步长为1的Stride访问模式，在Stream访问模式下，Stride预取器也可以学习到相应的访问模式，即步长为1。但Stride预取器在Stream流下表现不佳，因为Stride是一个degree-1的预取器，在大量连续的Stream流访问下，覆盖率和及时性不够。因此在Stream预取器已经检测到Stream模式后，Stride预取器可以不用发出预取请求。

Stride预取器会在训练阶段s0查询Stream预取器，s1阶段进行region tag比较，s2阶段得到比较结果，并判断此region是否为处于激活状态，如果处于激活状态则不触发Stride预取。

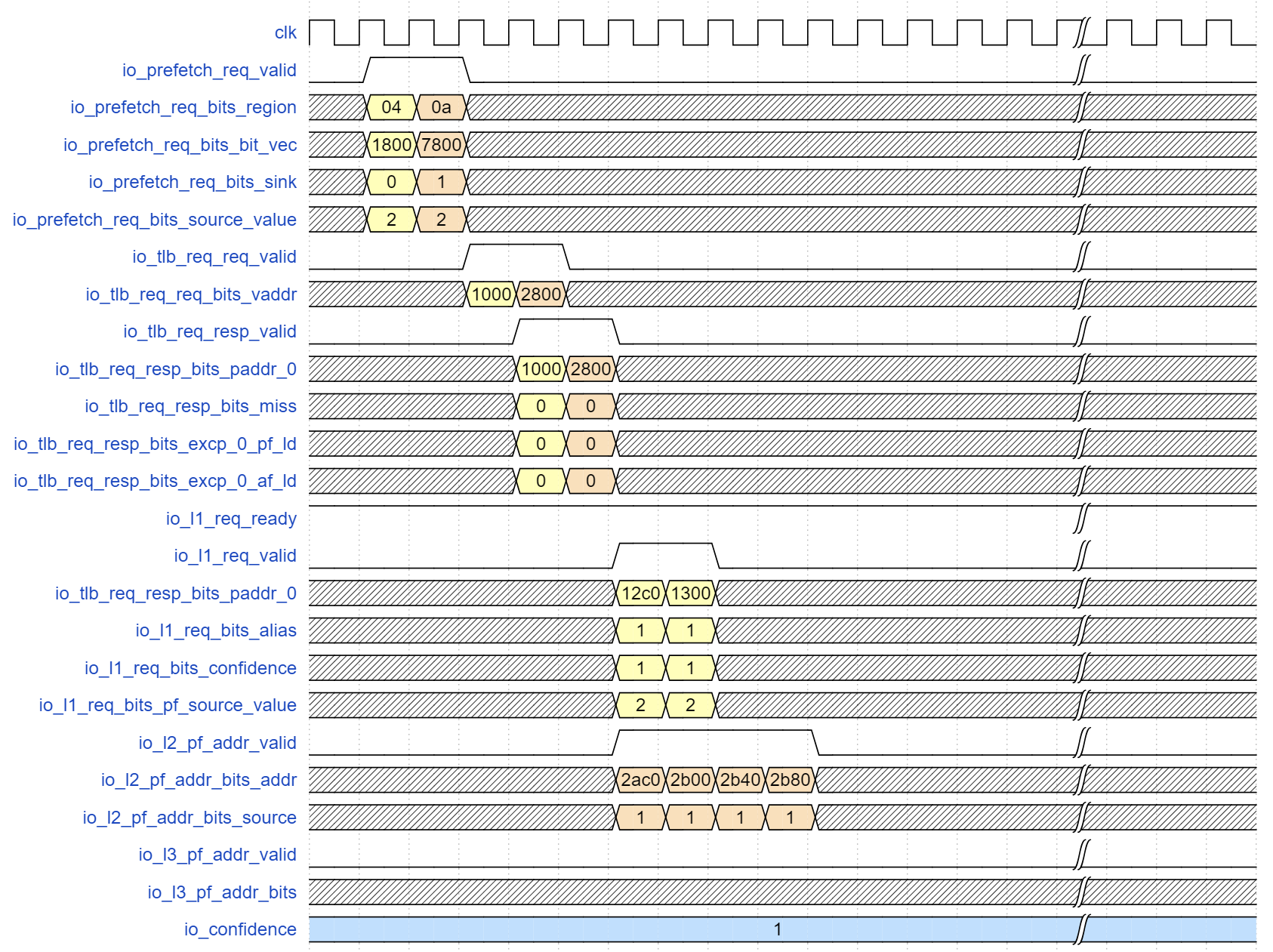
* + 1. 三级模块MutiLevelPrefetchFilter
       1. 功能

Stream与Stride的Prefetch Filter主体部分。用于接收Stream与Stride预取器发来的各级预取请求，对重复的预取请求进行过滤，将预取请求从虚拟地址空间转换为物理地址空间，并根据预取请求的期望目的层级，向各层Cache转发物理地址空间预取请求。

* + - 1. 整体框图



* + - 1. 接口列表
      2. 接口时序



上图表示从第2个上升沿开始连续接收到两个由Stream预取器发来的预取请求，第1个的sink为0，表示发给L1 Cache，Region为0x4，Bit Vector显示这个请求希望预取Region为0x4的第12、13个块（下标从1开始）。第2个sink为1，表示发给L2 Cache，Region为0xa，Bit Vector显示这个请求希望预取Region为0xa的第12、13、14、15个块（下标从1开始）。

接收到两个预取请求后分别分配各一项Entry，首先将虚地址Region转换为物理地址Region。在第4、5个上升沿，分别将两个预取请求所在页的虚拟地址发送TLB转换请求给TLB，在第5、6个上升沿，收到TLB转换的响应结果，两个预取请求的Entry得到物理地址，分别为0x1000、0x2800。

得到Region的物理地址后，通过Bit Vector中指示的要预取的块的index，可以计算出具体的预取地址，L1通过io\_l1\_req接口发出，由于Dcache为VIPT，每一路大小为16KB，需要附带两位的alias位，以让预取请求能索引到正确的Set，alias为Vaddr[13:12]，也是Region[3:2]。由于io\_confidence一直为1，io\_l1\_req\_bits\_confidence也为1。

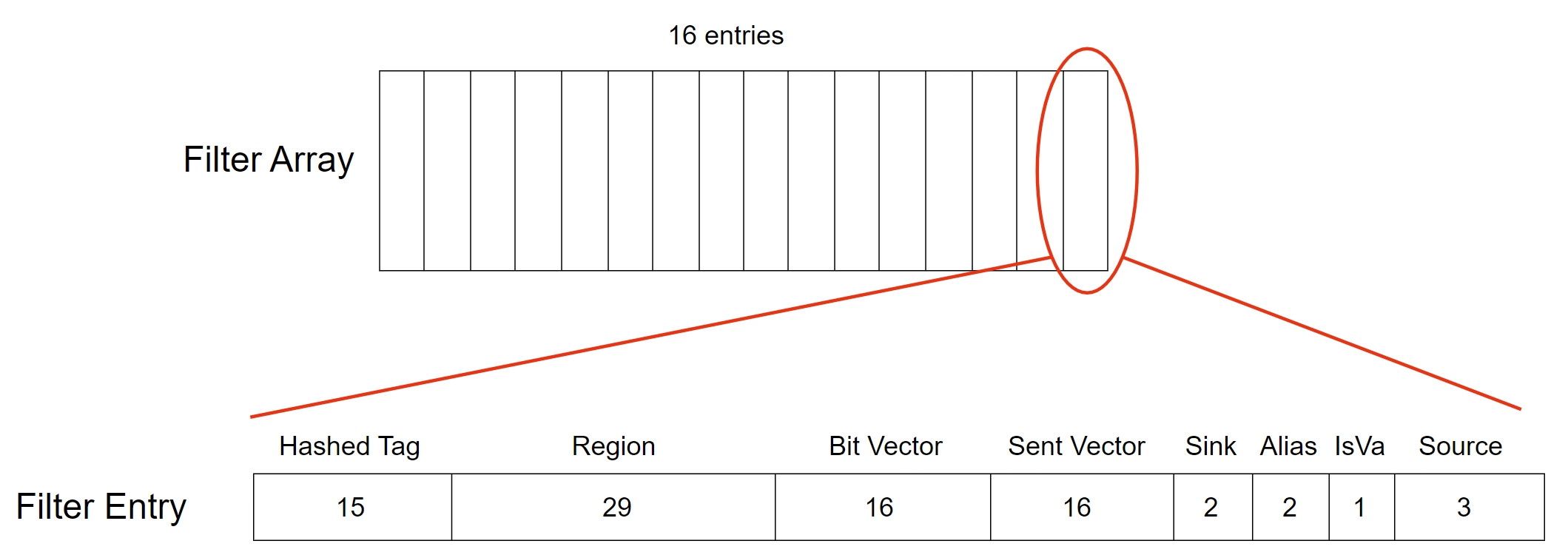
L2通过io\_l2\_pf\_addr接口发出，连续发出四个。

* + - 1. 关键电路

MutiLevelPrefetchFilter内部有5条并行的流水线，分别为：

* 入队流水线
* TLB交互流水线
* L1预取流水线
* L2预取流水线
* L3预取流水线

MutiLevelPrefetchFilter中的主要存储结构为Filter Array，其中一共有16项Filter Entry。MutiLevelPrefetchFilter与Stream预取器类似，将内存以Region为粒度划分，一个Region的大小为1KB。



Filter Entry结构说明：

* Hashed Tag：当前Entry所在Region地址的哈希值（虚拟地址空间），用来作为Region标识，区分不同的Region
* Region：当前Region的完整Region地址，在TLB翻译前为Vaddr[38:10]，在TLB翻译后为Paddr[35:10]
* Bit Vector：用于标识当前1KB的Region内16个Cache块中哪些需要被预取。需要被预取标为1，否则为0。
* Sent Vector：用于标识当前1KB的Region内16个Cache块中的预取请求哪些已经发出。已经发出标为1，否则为0。
* Sink：预取请求发送的目的端，0表示发送到L1 Dcache，1表示发送到L2 Cache，2表示发送到L3 Cache
* Alias：别名位，即Vaddr[13:12]，由于Dcache按照VIPT的方式组织，每一路的大小（16KB）超过页面大小（4KB），因此需要两位的页别名，发送给L1 Dcache的预取请求需要Alias位才能确定访问Dcache的哪一个Set。
* IsVa：标识Entry中的Region字段当前是否为虚拟地址，在TLB翻译前IsVa为1，在TLB翻译后IsVa为0
* Source：标识预取请求是由哪个预取器发出，0为无效，1为Stride，2为Stream，3为Store预取。目前只使用Stride与Stream

MutiLevelPrefetchFilter主要有四部分关键电路：

1. **接收预取请求（入队流水线）**

接收预取请求分为两级流水s0与s1：

S0：接收预取请求，根据预取请求的Region地址计算哈希，同时读取Filter Array中的16项Filter Entry，并行匹配，如果预取请求的Hashed Region Tag与已有Filter Entry的Hashed Tag相同则为命中，否则为Miss。Miss的情况下利用PLRU替换算法选出一项进行替换

S1：根据S0是否命中执行不同的操作

* 命中：更新命中项的Bit Vector，例如收到预取要求取回当前Region中的第0,1,2,3项，则将Bit Vector中的0,1,2,3位标为1。如果收到的预取请求目的层级更靠近核内，则将Sink升级。例如当前Filter Entry的Sink为1表示Filter Entry要向L2 Cache发送预取请求，收到Sink为0的预取请求要求取回此Region的第0,1项，则将Filter Entry的Sink改为0，并将Bit Vector的0,1位标为1，之后此项Filter Entry不再向L2 Cache发送预取请求，而是向L1 Dcache发送预取请求。
* Miss：设置新的Filter Entry信息，Hashed Tag为预取请求Region地址的哈希值，Region为预取请求的Region地址，Bit Vector为这个预取请求想要在Region中预取哪些块，Sent Vector为全0，Sink为预取请求的目标层级，Alias为Vaddr[13:12]，isVa为1，Source为发出预取请求的预取器标号。将新的Filter Entry更新到S0阶段替换算法选出的替换项中。

1. **发出TLB请求**

发出TLB请求分为两级流水线s0与s1：

S0：对16项Filter Entry进行仲裁，Filter Entry中没有完成TLB转换的Entry参与仲裁，即IsVa为1的项，仲裁后得到一项Entry，从Region中取出虚拟页号，产生一个TLB翻译请求，进行锁存

S1：将S0选出的TLB翻译请求发送给TLB。如果S1阶段发现此时正好有新的预取请求需要进入Filter Array，且需要替换正在发送TLB翻译请求的Filter Entry，发生冲突，S1阶段停止发送TLB翻译请求。

1. **接收TLB响应**

由于TLB不会被阻塞，在发出请求的下一级流水级就可以得到响应，即在TLB请求发出的s1阶段的下一级流水线s2收到TLB响应。根据TLB响应进行不同操作

* TLB命中
* TLB触发异常，如缺页，将这一项Filter Entry设置为invalid状态，即IsVa改为0，Sent Vector改为全1，Bit Vector改为全0
* TLB正常返回物理地址，将这一项Filter Entry的Region改为翻译后的Region地址，IsVa改为0
* TLB Miss，不修改Filter Entry的任何状态，等待Filter Entry重新参与仲裁和发送TLB翻译请求

1. **发出L1/L2/L3预取请求**

由于L1 Dcache与L2/L3 Cache有特性上的区别，对应的预取请求发送操作也有所不同。L1 Dcache大小远小于L2/L3 Cache，MSHR数量较少，同时发给L1 Dcache的预取请求并不一定会最终产生作用（bank冲突被流水线刷掉，MSHR满被丢弃等），为了尽可能使发给L1 Dcache的请求发挥作用，同一个地址的预取请求可以发送多次，由于L1 Dcache的MSHR有同地址过滤功能，发重复地址的预取请求并不会造成大量的性能损失，因此L1预取请求发送时不考虑Sent Vector过滤，只看Bit Vector。

L2/L3 Cache的MSHR数量较多，但不支持过滤，如果收到大量的重复地址预取请求容易造成阻塞影响性能，因此L2/L3预取请求发送时需要考虑Sent Vector过滤，已经发过的预取请求不能再次发送。

* L1 Dcache：发给L1 Dcache分为两级流水线
* s0阶段对16项Filter Entry进行仲裁，Filter Array中Sink为L1 Cache的Entry可以参与仲裁，参与仲裁的Entry根据Bit Vector的中的信息选取一个请求参与仲裁，例如某一个Filter Entry的Bit Vector为1000\_1000\_1000\_100**1**，选出位置0即这个Region的第0个Cache块发出预取。同理如果s0阶段的Filter Entry正好要被替换，就不能参与仲裁。最终s0阶段将仲裁出一个Filter Entry的预取请求并锁存。
* s1阶段将s0阶段锁存的预取请求发给访存流水线，s1阶段检查请求的合法性，地址必须位于DRAM空间，否则丢弃。s1阶段请求发出后，将对应Filter Entry的Bit Vector信息修改，如1000\_1000\_1000\_100**1**修改为1000\_1000\_1000\_100**0**。s1阶段需要修改Bit Vector，入队流水线也有可能更新Bit Vector，如果同一周期两条流水线都需要更新同一个Filter Entry的Bit Vector，则让L1预取请求发送流水线暂停发送，等待入队流水线完成操作再执行。
* L2/L3 Cache：MutiLevelPrefetchFilter内部只有一级流水线，外部由于物理距离过长，线延迟高，需要多级流水线打拍。
* 内部的一级流水线对16项Filter Entry进行仲裁，Filter Array中Sink为L2/L3 Cache的Entry可以参与仲裁，参与仲裁的Entry根据Bit Vector和Sent Vector的中的信息选取一个请求参与仲裁，例如某一个Filter Entry的Bit Vector为1000\_1000\_1000\_**1**00**1**，Sent Vector为0000\_0000\_0000\_000**1**，选取还未发送过的请求进行发送，即选Sent Vector对应位置为0，Bit Vector对应位置为1的请求发送，这个例子中选择Region中的第3个块。与L1类似如果这个Filter Entry正要被替换，就不能参与仲裁。最终s0阶段将仲裁出一个Filter Entry的预取请求并向外发送，将对应预取请求Filter Entry的Sent Vector位置标为1，在这个例子中Sent Vector由0000\_0000\_0000\_000**1**变为0000\_0000\_0000\_**1**00**1**。

1. PPA
2. 验证关注点
3. Floorplan 建议

预取器模块不在时序关键路径上，对训练或者发出预取请求进行加拍不会对性能造成显著影响，因此可以将预取器模块摆放在较远的位置。

1. 遗留问题