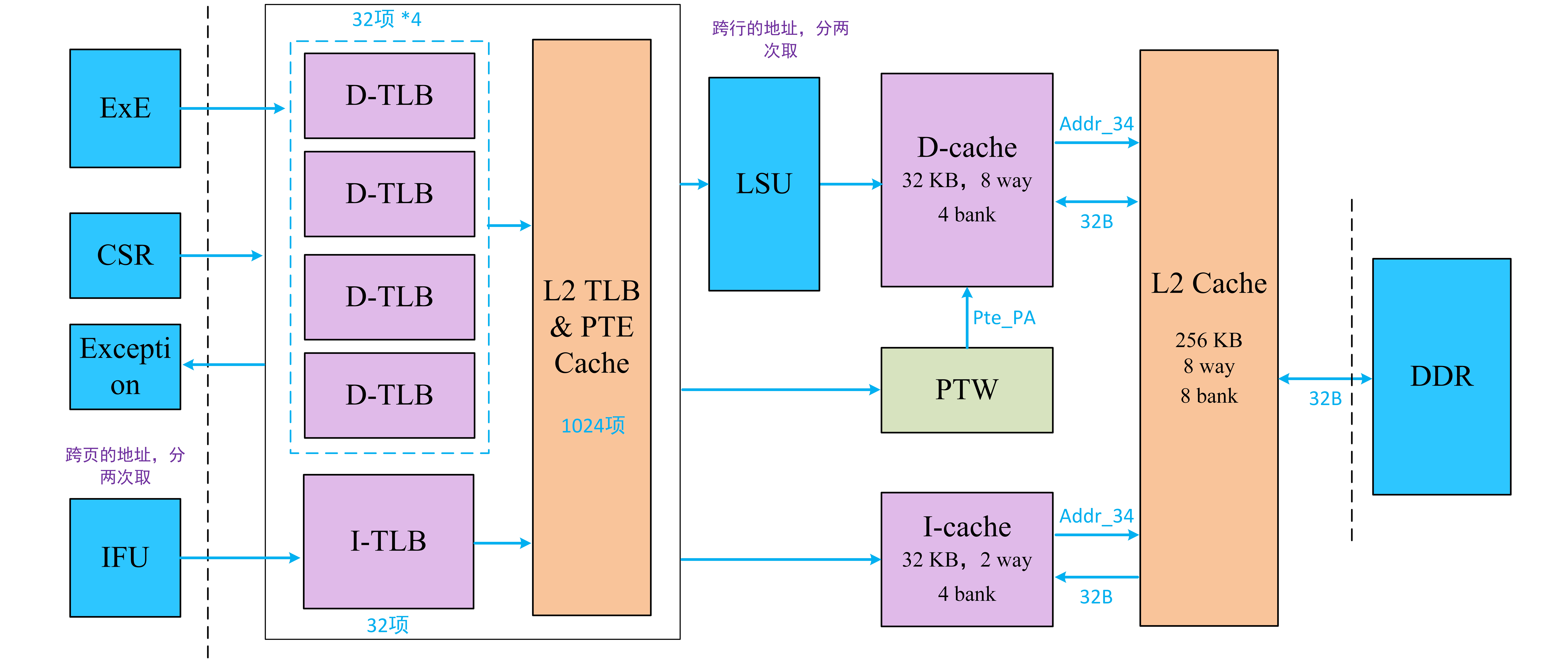
# 存储系统

## 存储系统概述

### 整体架构

本次设计中，存储系统主要由存储管理单元（Memory Management Unit, MMU）和Cache两部分组成。存储管理单元主要通过对页表的查询和检查，对主存以及Cache的空间进行权限划分，Cache基于局部性原理，支持CPU核对主存的访存操作。存储系统架构如图 1.1.1 所示。

图 1.1.1 存储系统整体架构图

其中，MMU包括TLB（快表）与PTW（页表遍历）两个模块，Cache则由Dcache、Icache和L2 Cache三部分组成。

存储系统中，MMU主要与CPU核中的EX（执行）、IFU（取指）、LSU（访存）、CSR（状态控制寄存器）和Retire（出局）交互，主要进行虚实地址转换，以及page fault和access fault的检测。

Cache部分与LSU、MMU以及DDR进行交互，负责支撑处理器从存储器中取指令、以及load和store指令的执行。对于取指来说，本系统最高支持同时读取32字的指令。对于访存指令，存储系统提供了并发访问的功能，能同时支持四条load或store指令的同时访问。

### 接口定义

1. MMU输入输出信号
2. Cache输入输出信号

表 1.1.1 Cache输入信号

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 长度 | 来源 | 含义 |
| i\_lsu\_drive0\_cache | 1 | LSU | 来自LSU的请求信号 |
| i\_freeNext0\_lsu | 1 | LSU | 驱动LSU后得到的应答信号 |
| i\_lsu\_PA0\_34 | 34 | LSU | 来自LSU的物理地址 |
| i\_lsu\_storeData0\_32 | 32 | LSU | 来自LSU的写入数据 |
| i\_lsu\_rd0\_5 | 5 | LSU | 当前指令对应的寄存器编号 |
| i\_lsu\_index0\_6 | 6 | LSU | 当前指令对应的指令标号 |
| i\_lsu\_flag0\_10 | 10 | LSU | 来自LSU的标记位，其中第10位指示当前指令是load还是store |
| i\_lsu\_bypass0 | 1 | LSU | 当前指令是否需要经过旁路 |
| i\_ptw0\_drive | 1 | MMU | MMU页表遍历的请求信号 |
| i\_freeNext\_ptw | 1 | MMU | 驱动MMU后得到的应答信号 |
| i\_ptw\_ptePA\_34 | 34 | MMU | MMU页表遍历时发来的物理地址 |
| i\_ifu\_drive0 | 1 | IFU | 来自IFU的请求信号 |
| i\_freeNext0\_ifu | 1 | IFU | 驱动IFU后得到的应答信号 |
| i\_ifu\_PA0\_34 | 34 | IFU | 来自IFU的物理地址 |
| i\_freeNext\_retire\_load | 1 | 出局 | 驱动出局后得到的应答信号 |
| i\_DDR\_drive\_cache | 1 | 主存 | 来自主存的请求信号 |
| i\_read\_freeNext\_DDR | 1 | 主存 | 驱动主存读Cache line后得到的应答信号 |
| i\_write\_freeNext\_DDR | 1 | 主存 | 驱动主存写Cache line后得到的应答信号 |
| i\_DDR\_refill\_data\_256 | 256 | 主存 | 主存回填的Cache line |

表 1.1.2 Cache输出信号

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 长度 | 去向 | 含义 |
| o\_lsu\_free0\_cache | 1 | LSU | 给LSU的应答信号 |
| o\_driveNext0\_lsu | 1 | LSU | Store执行完毕驱动LSU的请求信号 |
| o\_lsu\_rd0\_5 | 5 | LSU | 指令对应的寄存器编号 |
| o\_lsu\_index0\_6 | 6 | LSU | 指令对应的指令标号 |
| o\_lsu\_flag0\_10 | 10 | LSU | 给LSU的标记位，其中第10位指示当前指令是load还是store |
| o\_lsu\_bypass0 | 1 | LSU | 指令是否需要经过旁路 |
| o\_ptw0\_free | 1 | MMU | 给MMU的应答信号 |
| o\_pte\_32 | 32 | MMU | 页表项 |
| o\_ifu\_free0 | 1 | IFU | 给IFU的应答信号 |
| o\_driveNext0\_ifu | 1 | IFU | 取指令完毕驱动IFU的请求信号 |
| o\_instr0\_to\_ifu\_256 | 256 | IFU | 取指令完毕给IFU的指令 |
| o\_driveNext\_retire\_load | 1 | 出局 | Load执行完毕驱动出局的请求信号 |
| o\_loadData0\_to\_retire\_38 | 38 | 出局 | Load执行完毕输出的指令标号和数据 |
| o\_DDR\_free\_cache | 1 | 主存 | 给主存的应答信号 |
| o\_read\_driveNext\_DDR | 1 | 主存 | 读主存的请求信号 |
| o\_write\_driveNext\_DDR | 1 | 主存 | 写主存的请求信号 |
| o\_write\_data\_to\_DDR\_256 | 256 | 主存 | 写主存的数据 |

LSU的存储访问、IFU的取指令，都是支持四路并发的，其中LSU可以实现三路store一路load的并行处理，在此只列出部分端口。

### 工作机制

在IFU进行取指令或者执行模块遇到load/store指令时，存储系统将被启动。在IFU需要取指令时，首先会将PC作为虚拟地址发往ITLB，如果命中ITLB，将会直接给出一个物理地址。IFU将会在该物理地址的基础上，同时产生三个缓存行的物理地址访问Icache。由于Icache采用了4个bank并行访问的架构，这三个物理地址可以同时访问三个相邻缓存行。

对于load/store指令来说，可以同时进行4条指令的并行访问。当执行模块产生访存指令的虚拟地址后，首先进入DTLB进行虚实地址转换，转换完成后，将对应的物理地址发往LSU，由LSU检测地址相关性，以及调度访问Dcache的顺序。

在ITLB和DTLB的工作中，如果发生TLB缺失，则会访问L2级别的TLB；在L2级别再缺失，则会启动PTW模块对Dcache进行访问。如果在访问TLB或者在进行页表遍历的途中出现了缺页异常，则会启动中断异常处理模块；页表遍历途中或TLB命中产生的物理地址，会经过PMP（物理存储保护）检测是否有Access Fault。其中，取指令时出现的异常，会直接访问中断异常模块进行异常处理，load/store指令产生的异常则会发往出局模块。

在本次设计中Icache、Dcache都为32KB大小，缓存行的大小为32B，Icache两路组相联，使用fifo替换策略，Dcache采用八路组相联，采用PLRU替换策略。对于LSU来说，Dcache有4个独立的存储访问端口，LSU会根据需要分别发出对应的事件和地址。其中，Dcache根据物理地址的[ 6 : 5 ]位来区分地址对应的是哪一个bank。IFU和Icache的交互也是类似的。

当L1级的Cache缺失时，则会访问L2级的Cache。L2 Cache根据物理地址的[ 7 : 5 ]位区分出了8个bank，每两个bank对应Icache、Dcache中的一个。L2级采用八路组相联，使用的是PLRU替换策略。对于同时访问同级别Cache冲突的情况，都采用轮询仲裁的方式进行处理。

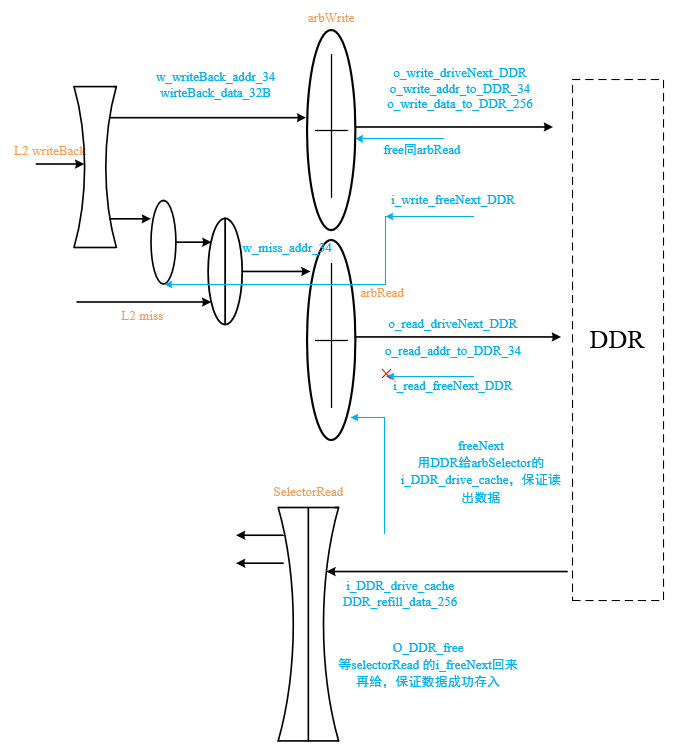


图 1.1.2 与主存的交互

图 1.1.2 展示了L2级Cache与主存的交互图。L2级Cache有两种访问主存的情况：①当L2 Cache缺失了某一缓存行时；②当L2 Cache需要写回某一缓存行时。当缺失缓存行时，发现需要被驱逐的行Dirty位为1，说明被驱逐的行被修改过，需要写回到主存，此时就是情况②出现。

对于情况①，会直接激活读取仲裁的arbRead，直接向DDR发出读请求，等将数据读出后，DDR会通过SelectorRead将事件和数据发送给L2 Cache对应的Bank。对于情况②，默认写回后还需要从DDR中找到缺失行，所以会先激活arbWrite对DDR发出写事件，然后再激活读取仲裁进行读操作。

## MMU

## Cache

### L1 指令Cache

Icache采用两路组相联，使用fifo替换策略，数据部分为32KB，标记和有效位空间为2.625KB，使用SRAM IP核生成。本子模块采用了异步微流水线进行时序控制，具体结构如图 1.3.1 所示。



图 1.3.1 指令Cache子模块

其中，Icache使用 [13: 5] 作为访问SRAM的索引位，而 [33:14] 作为指令Cache的标记位。

指令Cache主要支持IFU取指令和L2级Cache回填两种操作。当IFU发出取指令的事件时，则根据IFU提供的物理地址，使用其中的索引部分对SRAM进行读操作，此时写使能置为无效，仅从中读出数据部分和标记部分，再通过Tag Compare部分判断当前地址是否命中。如果命中则直接把32字节的指令发给IFU；如果比较后发现缺失，则会请求L2级Cache回填。

L2级Cache在得到对应缓存行后，对指令Cache进行回填，此时还是经过SRAM，但会将写使能置为有效，缓存行填入SRAM，然后从“回填完成”直接给出指令给IFU。以下是指令Cache输入输出信号表。

表 1.3.1 指令Cache输入信号

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 长度 | 来源 | 含义 |
| rst | 1 | 顶层 | 复位信号 |
| i\_Itlb\_drive | 1 | IFU | 取指令请求 |
| i\_Itlb\_PA\_34 | 34 | IFU | 取指令物理地址 |
| i\_L2Cache\_drive | 1 | L2 Cache | L2级Cache发送的请求信号 |
| i\_freeNext\_L2Cache | 1 | L2 Cache | 驱动L2级Cache得到的应答信号 |
| i\_L2Cache\_refillLine\_32B | 256 | L2 Cache | L2级Cache填充的缓存行 |
| i\_freeNext\_ifu | 1 | IFU | 驱动IFU得到的应答信号 |

表 1.3.2 指令Cache输出信号

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 长度 | 去向 | 含义 |
| o\_Itlb\_free | 1 | ITLB | 给取指令操作的应答信号 |
| o\_L2Cache\_free | 1 | L2 Cache | 给L2级Cache的应答信号 |
| o\_driveNext\_L2Cache | 1 | L2 Cache | 驱动L2级Cache的请求信号 |
| o\_miss\_Addr\_to\_L2cache\_34 | 34 | L2 Cache | 指令Cache对应缺失行的地址 |
| o\_driveNext\_ifu | 1 | IFU | 驱动IFU的请求信号 |
| o\_hit\_data\_to\_ifu\_32B | 256 | IFU | 给IFU输出的指令 |

### L1 数据Cache

L1级数据Cache采用八路组相联，使用PLRU替换策略，PLRU如图 1.3.2示意，通过二叉树的形式，每一级用0表示左子树最近未被访问，用1表示右子树最近未被访问，逐级的向下直到叶子节点。选出树中最近最少用的一路，进行驱逐，并更新父亲节点直到 [0]节点。对应的策略真值表如表 1.3.1 所示。

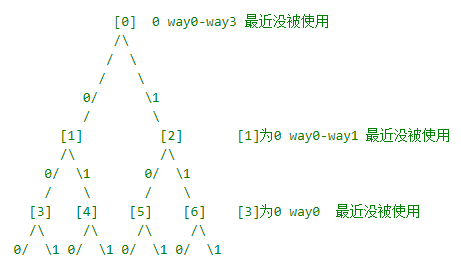
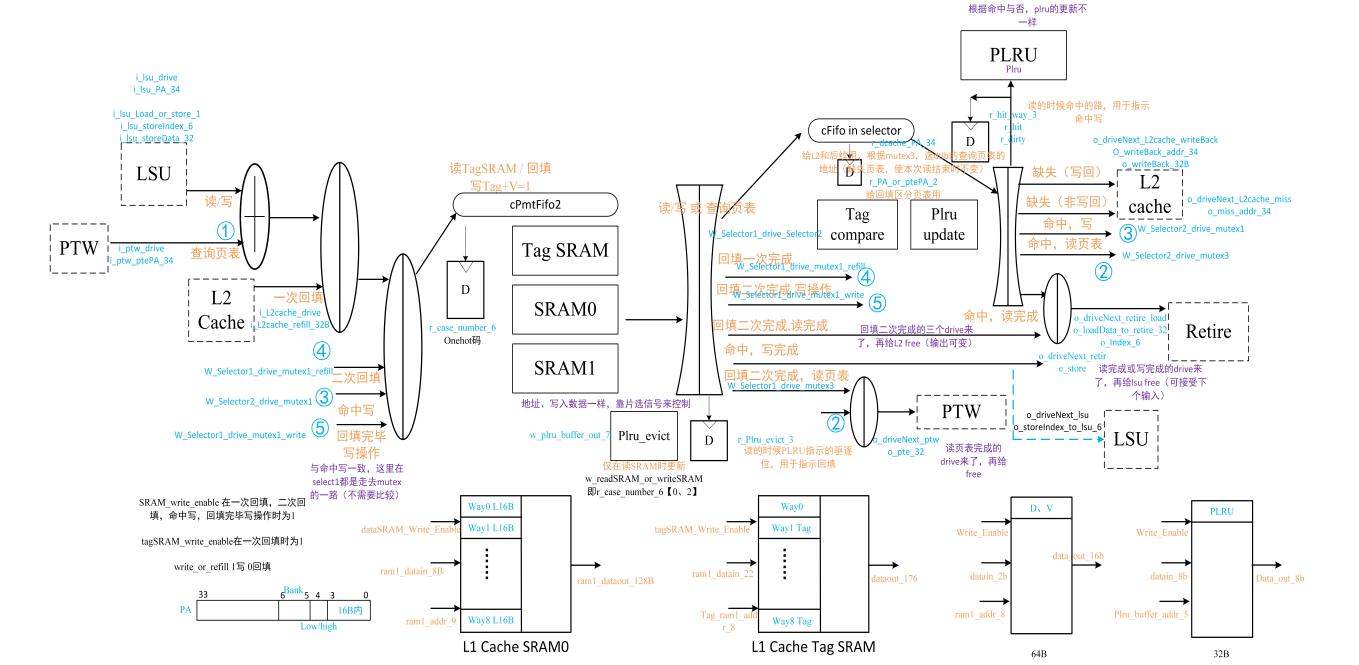


图 1.3.2 PLRU替换策略

表 1.3.3 PLRU策略真值表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 第几行 | [6] | [5] | [4] | [3] | [2] | [1] | [0] |
| 0 | x | x | x | 0 | x | 0 | 0 |
| 1 | x | x | x | 1 | x | 0 | 0 |
| 2 | x | x | 0 | x | x | 1 | 0 |
| 3 | x | x | 1 | x | x | 1 | 0 |
| 4 | x | 0 | x | x | 0 | x | 1 |
| 5 | x | 1 | x | x | 0 | x | 1 |
| 6 | 0 | x | x | x | 1 | x | 1 |
| 7 | 1 | x | x | x | 1 | x | 1 |

L1数据Cache中，数据部分为32KB，标记和有效位空间为2.75KB。L1级数据Cache使用物理地址的 [ 11: 5 ] 位作为索引，[ 33: 12 ] 位作为标记。Dcache主要包括LSU读/写操作、页表遍历读操作、L2级Cache回填操作以及回写L2级Cache。L1数据Cache的具体结构如图 1.3.3 所示。

图 1.3.3 数据Cache子模块

在数据Cache中，SRAM分为标记使用的Tag SRAM和数据使用的Data SRAM，D、V以及替换策略则使用buffer进行存储。对于Tag SRAM，支持一路写入，八路读出的操作。

对于Data SRAM，本次使用的SRAM仅为单端口读写，考虑到回填操作需要向Data SRAM写入256位数据，但写操作仅需要写入32位数据，我们将Data SRAM从逻辑上拆成了两部分，每一部分仅存储缓存行的一半。每一部分支持128字节的读出和8字节的写入。当处理store指令时，则直接将空出来的4字节用Data SRAM给出的数据进行填充，一次写Data SRAM完成；处理回填时，则需要填两次，每一次两个部分各填8字节。

对于LSU，当需要执行load/store指令时，首先会根据物理地址的 [ 6: 5 ] 位去寻找四个Dcache bank中的其中一个，并发出请求信号，同时将物理地址和store所需要的数据发送。通过前面的汇聚模块，LSU的请求会触发fifo，从而更新保存状态的寄存器，使得SRAM到达对应的读写状态。无论是load指令还是store指令，第一次进入都会从Tag SRAM和Data SRAM中读出标记和数据，并通过择路去往Tag Compare模块。在比较之后，得出是否命中，如果命中，则根据当前是load还是store前往命中的两条路径；如果未命中，则前往L2级Cache，激活写回或者回填操作。

对于未命中但是组内八路缓存行都有效的情况下，需要驱逐出其中的一行，以便L2级Cache将需要的缓存行回填进来。这里根据该组PLRU的状态选中驱逐的行，如果该行的Dirty位有效，则需要进行写回后再回填；Dirty位无效，则直接进行回填操作。

启动回填操作后，L2级Cache回将回填行连带请求信号发送回来，再次触发模块开始的汇聚，并更新触发器保存的状态，使得Tag SRAM和Data SRAM的写使能有效。在一次回填时，会将标记填入，并写入部分的缓存行，随即触发二次回填。二次回填时，Tag SRAM的写使能置为无效，Data SRAM写使能保持有效，写入剩余的缓存行。，回填完成后，根据是load、store还是页表遍历，通过择路去往对应的输出端口。

写操作需要在命中或者回填完成后进行一次额外的写入操作，此时会将需要写入的4字节和已经在Data SRAM中存在的4字节拼凑成8字节，再写入SRAM

页表遍历也是访问数据Cache的一种可能，该情况的操作与LSU的load操作类似，区别在于命中后或者回填完成后，会将读取到的页表项发送给MMU的PTW模块。页表遍历和LSU的访问不能同时运行，所以使用仲裁模块进行处理。

在数据Cache运行的过程中，主要由PLRU\_evict和PLRU\_update处理替换策略的更新以及输出，并通过PLRU buffer进行存储。PLRU buffer仅在load/store指令或者查询页表的开始会被更新，在回填以及命中后写操作是不被更新的。

表 1.3.4 数据Cache输入信号

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 长度 | 来源 | 含义 |
| rst | 1 | 顶层 | 复位信号 |
| i\_lsu\_drive | 1 | LSU | LSU访问的请求信号 |
| i\_freeNext\_lsu | 1 | LSU | 驱动LSU得到的应答信号 |
| i\_lsu\_PA\_34 | 34 | LSU | LSU访问的物理地址 |
| i\_lsu\_storeData\_32 | 32 | LSU | LSU的store操作所需要写的数据 |
| i\_lsu\_storeIndex\_6 | 6 | LSU | LSU访问时指令的标号 |
| i\_lsu\_load\_or\_store | 1 | LSU | LSU访问是load还是store操作 |
| i\_freeNext\_retire\_store | 1 | 出局 | Store完成驱动出局的应答信号 |
| i\_freeNext\_retire\_load | 1 | 出局 | Load完成驱动出局的应答信号 |
| i\_ptw\_drive | 1 | MMU | 查找页表时给的请求信号 |
| i\_freeNext\_ptw | 1 | MMU | 驱动MMU给的应答信号 |
| i\_ptw\_ptePA\_34 | 34 | MMU | 查找页表时给的物理地址 |
| i\_L2cache\_drive | 1 | L2 Cache | L2级Cache的请求信号 |
| i\_freeNext\_L2cache\_writeBack | 1 | L2 Cache | 写回操作时驱动L2级Cache的应答信号 |
| i\_freeNext\_L2cache\_miss | 1 | L2 Cache | 缺失时驱动L2级Cache的应答信号 |
| i\_L2cache\_refill\_32B | 256 | L2 Cache | L2级Cache回填的缓存行 |

表 1.3.5 数据Cache输出信号

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 长度 | 去向 | 含义 |
| o\_lsu\_free | 1 | LSU | 给LSU的应答信号 |
| o\_driveNext\_lsu | 1 | LSU | 驱动LSU的请求信号 |
| o\_storeIndex\_to\_lsu\_6 | 6 | LSU | 完成的指令的标号 |
| o\_driveNext\_retire\_store | 1 | 出局 | 写完成 |
| o\_driveNext\_retire\_load | 1 | 出局 | 读完成 |
| o\_loadData\_to\_retire\_32 | 32 | 出局 | 读完成输出的数据 |
| o\_ptw\_free | 1 | MMU | 页表遍历时给MMU的应答信号 |
| o\_driveNext\_ptw | 1 | MMU | 页表遍历结束驱动MMU的请求信号 |
| o\_pte\_32 | 32 | MMU | 页表遍历得到的页表项 |
| o\_L2cache\_free | 1 | L2 Cache | 给L2级Cache的应答信号 |
| o\_driveNext\_L2cache\_writeBack | 1 | L2 Cache | 写回时驱动L2级Cache的请求信号 |
| o\_driveNext\_L2cache\_miss | 1 | L2 Cache | 缺失时驱动L2级Cache的请求信号 |
| o\_miss\_addr\_34 | 34 | L2 Cache | 缺失对应缓存行的物理地址 |
| o\_writeBack\_addr\_34 | 34 | L2 Cache | 写回对应缓存行物理地址 |
| o\_writeBack\_32B | 256 | L2 Cache | 写回的缓存行数据 |