|  |  |
| --- | --- |
| 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： |  |

南湖V2项目

2022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 | V1.0 | C | 创建，对应代码版本：3b1a683bf8b2a904a7e4d56372b2cec6ba0ae66c |  |  |
| 2 |  |  |  |  |  |
| 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](#_Toc156295451)

[1.1 文档介绍 1](#_Toc156295452)

[1.2 参考文档 1](#_Toc156295453)

[1.3 术语说明 1](#_Toc156295454)

[2 设计规格 1](#_Toc156295455)

[3 功能描述 2](#_Toc156295456)

[3.1 功能概述 2](#_Toc156295457)

[3.2 功能详述 2](#_Toc156295458)

[3.2.1 缓存指令 2](#_Toc156295459)

[3.2.2 从L2取回数据，软件维护与L1 DCache一致性 3](#_Toc156295460)

[3.2.3 向IFU提供两个连续Cache行 3](#_Toc156295461)

[3.2.4 替换算法 4](#_Toc156295462)

[3.2.5 分支预测指导预取 5](#_Toc156295463)

[3.2.6 地址翻译错误、物理内存保护错误检查 5](#_Toc156295464)

[3.2.7 异常处理基本规范 5](#_Toc156295465)

[4 总体设计 6](#_Toc156295466)

[4.1 整体框图 6](#_Toc156295467)

[4.2 接口列表 6](#_Toc156295468)

[4.3 接口时序 7](#_Toc156295469)

[4.4 时钟复位 10](#_Toc156295470)

[4.5 寄存器配置 10](#_Toc156295471)

[4.6 补充说明 11](#_Toc156295472)

[5 模块设计 11](#_Toc156295473)

[5.1 ICache Memory规格： 11](#_Toc156295474)

[5.2 ICacheMainPipe 12](#_Toc156295475)

[5.2.1 功能概述 12](#_Toc156295476)

[5.2.2 结构框图 14](#_Toc156295477)

[5.2.3 接口列表 14](#_Toc156295478)

[5.2.4 接口时序 15](#_Toc156295479)

[5.3 ICacheMissUnit 17](#_Toc156295480)

[5.3.1 功能概述 17](#_Toc156295481)

[5.3.2 结构框图 18](#_Toc156295482)

[5.3.3 接口列表 18](#_Toc156295483)

[5.3.4 接口时序 18](#_Toc156295484)

[5.3.5 ICacheMissEntry 19](#_Toc156295485)

[5.4 FDIPPrefetch 20](#_Toc156295486)

[5.4.1 功能概述 20](#_Toc156295487)

[5.4.2 结构框图 21](#_Toc156295488)

[5.4.3 接口列表 21](#_Toc156295489)

[5.4.4 接口时序 21](#_Toc156295490)

[5.4.5 IPrefetchPipe 23](#_Toc156295491)

[5.4.6 PrefetchQueue 24](#_Toc156295492)

[5.5 二级模块B 26](#_Toc156295493)

[5.6 关键电路 26](#_Toc156295494)

[5.6.1 时钟切换电路 26](#_Toc156295495)

[6 PPA 27](#_Toc156295496)

[7 验证关注点 27](#_Toc156295497)

[8 Floorplan 建议 27](#_Toc156295498)

[9 遗留问题 27](#_Toc156295499)

[10 附录 28](#_Toc156295500)

[ICache相关Tilelink通道信号详细解释 28](#_Toc156295501)

[Cache miss相关 28](#_Toc156295502)

1. 简介
   1. 文档介绍

本文档是指令缓存（ICache）的文档，描述南湖架构指令缓存设计。

本文档主要用于指导芯片模块的详细设计及验证。

* 1. 参考文档

*列出相关的参考文档。*

1. ITLB文档
2. ICache文档
3. IFU文档
4. FTQ文档
   1. 术语说明

表1.1 术语说明

|  |  |  |
| --- | --- | --- |
| 缩写 | 全称 | 描述 |
| ICache/I$ | Instruction Cache | 指令缓存 |
| L2 Cache | Level Two Cache | 第二级的指令及数据统一缓存 |
| IFU | Instruction Fetch Unit | 取指单元 |
| ITLB | Instruction Translation Lookaside Buffer | 指令地址翻译缓存 |

1. 设计规格
2. 支持缓存指令
3. 支持从L2 Cache取回数据，软件维护与L1 DCache的一致性
4. 支持向IFU提供两个连续的Cache行从而满足一个预测块的数据供应要求
5. 支持检查数据块错误并在出错时触发中断
6. 支持可配置的替换算法
7. 支持使用分支预测指导的指令预取数据到L2 Cache
8. 支持检查地址翻译错误、物理内存保护错误
9. 功能描述
   * 1. *进行功能概述。（从输入、处理、输出几个方面概述设计实现了什么功能）*
     2. *按特性，每条特性分步骤进行详细描述*

*注意:要描述清楚对于错误的输入的处理，比如是支持纠错（怎么纠正），还是上报异常或中断，还是不处理错误但保证不死机，还是直接忽略等。*

*例：*

* 1. 功能概述

ICache模块接收来自FTQ的取指令请求，按照地址生成相应的index寻址至多两个Bank的SRAM，请求同时会发送给ITLB，用于地址翻译。下一拍从中读出Cache中匹配的一个组中所有Cache行的数据及meta信息，ITLB返回地址翻译结果，根据meta信息确定Cache行命中状态，若ITLB缺失，则暂停流水线等待ITLB重填完成。若Cache行命中，则下一拍hit的请求数据在经PMP检查无误后返回给IFU，否则暂停流水线将请求发送给缺失单元MissUnit。ITLB的地址翻译规则、ITLB、PMP对权限检查的具体细节可参考RISC-V特权手册sv39及物理内存保护部分，对应模块文档。

MissUnit收到请求后，将向下游L2 Cache发送Tilelink的Acquire请求，等待MissUnit收到对应数据的Grant请求后同时进行SRAM重填和返回数据给MainPipe，MainPipe将数据返回给IFU。L2 Cache的具体行为可参考对应模块文档。

指令预取器FDIPPrefetch接收来自FTQ的取指packet目标地址，预取器利用ITLB完成地址翻译后进行多重过滤，过滤后入队PrefetchQueue，PrefetchQueue对过滤后的预取请求进行管理，向L2 Cache发送Tilelink Hint请求，把相应的缓存行预取到L2。

* 1. 功能详述
     1. 缓存指令

缓存指令指接受FTQ的取指请求，查找模块内SRAM中记录指令而后返回给IFU的流程。

* + - 1. 接受请求

ICache模块接受一个或两个地址相邻的Cache行读取请求，若处于Miss重填状态则首先等待数据重填，否则直接向Data和Meta SRAM发送请求。无论是否处于Miss重填状态，ICache都会向ITLB发送查找请求。

* + - 1. 读取SRAM及TLB结果

当SRAM、ITLB均可返回数据时，ICache读出相关数据生成TLB异常信号，同时根据Cache替换策略生成待替换Cache行信息。

* + - 1. 读取返回结果

ICache利用ITLB返回的物理地址向PMP发送查询请求，检查物理内存保护权限。若没有错误且在之前ITLB查询也未触发错误，则将命中的缓存行结果返回给IFU。

* + 1. 从L2取回数据，软件维护与L1 DCache一致性

从L2取回数据用于填充及更新缓存，一致性指DCache写入的数据将被及时同步给ICache。

* + - 1. 数据取回

当Cache出现miss时，将会把miss请求记录下来，通过Tilelink总线发送Acquire请求并等待总线回复的grant信号，将grant信号返回的数据暂存并发送grant\_ack请求表明收到了grant请求。

* + - 1. 重填SRAM

数据取回过程结束后，将取得的数据写回所替换的Cache行SRAM。

* + - 1. 返回结果

数据取回过程结束后，同时将数据返回给IFU。

* + - 1. 数据同步

ICache由软件维护一致性，当程序对代码段的数据进行写操作后，紧接着会执行fence.i指令，当ICache收到fencei信号后，将ICache的所有数据置为无效，需要重新向L2请求最新的指令数据。

* + 1. 向IFU提供两个连续Cache行

FTQ所发送的取指请求中包含一个地址，该地址含义为请求ICache返回以该地址起始至多32Byte的指令数据。为提升取指带宽，Cache内SRAM均划分为两个Bank，地址相邻Cache行始终位于不同Bank从而保证相邻两Cache行总是可以被一次读出不产生Bank冲突。

* + - 1. 请求接口

所有与fetch请求相关的端口都有PortNumber（此处为2）个，如IFU需要发送覆盖两个Cache行的请求，则会同时使用两个端口发送请求；如只需发送覆盖一个Cache行的请求，则会使用第一个请求端口而不拉高第二个请求端口valid信号，从而使第二个端口其他信号被忽略。

* + - 1. 内部实现

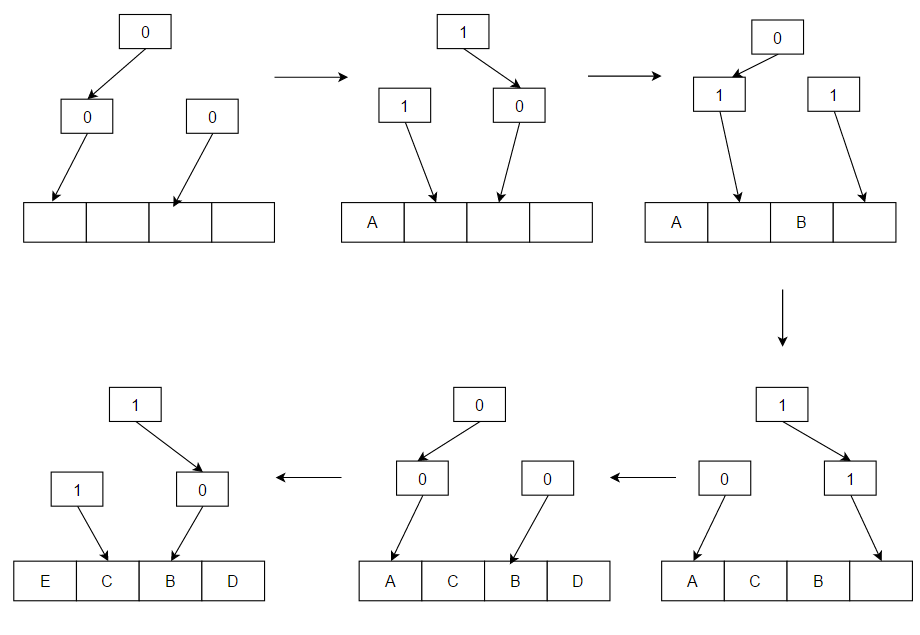
Data和Meta SRAM都划分为两个Bank，存储时将地址相邻的两Cache行分别存储于两不同Bank，则可使用单口SRAM实现两个Cache行数据的同时读取。为保证结果的正确性，在读取逻辑中按照vSetIdx最低位对两Cache行读请求所映射的SRAM Bank进行了MUX选通。

* + 1. 替换算法
       1. 算法集成

在ICache查找TLB结果的同时选出待替换Cache行，对应待替换行也即刻被更新访问时间。

* + - 1. 伪LRU

南湖架构采用伪LRU替换算法，其选取逻辑为：将所有路作为二叉搜索树上的一个叶子结点，每次访问某一路，则从二叉搜索树顶端节点向下遍历到该路所处位置，其路径上的所有中间节点均记录其子结点中上一次访问的结点方向，在遍历时若该方向不指向当前遍历结点方向则将方向反转指向当前遍历方向。在硬件实现上，二叉搜索树存储为近似软件数组的形式，也即最高位存储二叉树根节点，其后分别存储根的两叶子，以此类推。



上图展示了加入A、B、C、D、E过程中伪LRU指针方向的变化。

替换算法主要提供两种接口，在访问某一路时的状态更新及获取当前状态下待替换的路。

* + - * 1. 路访问状态更新

当访问Cache某路时，从二叉树的根节点向下根据该访问的路在二叉树左或右子树上递归生成新的子树部分，访问了左子树则右子树保持不变左子树相同算法更新；访问右子树则左子树保持不变右子树相同算法更新。若访问了右侧子树则该（子树）根节点设置为1否则设置为0。

* + - * 1. 待替换路选取

根据上文所述算法，从子树根节点向下按若根节点值为1遍历左侧否则遍历右侧的方法拼接得到的访问记录即为伪LRU算法所认定的上次访问最早路。

* + 1. 分支预测指导预取
       1. 请求接收

当CSR模块控制该功能开启的信号拉高时，ICache接收来自FTQ的预取请求。

* + - 1. 请求过滤

通过以下的手段对请求进行过滤：

1. 向ITLB发送查询请求，当出现异常或者miss时取消请求。
2. 向Meta发送查询请求，当命中时取消请求。
3. 与上一个请求对比，相同时取消请求。
4. 向PMP发送查询请求，当出现异常时取消请求。
5. 向PrefetchQueue发送查询请求，当命中时取消请求。
6. 检查ICache主流水各级流水线中正在处理的地址，重复时取消请求。
7. 检查MissUnit最近两个周期写回SRAM的地址，重复时取消请求。
8. 检查MissUnit中正在处理的地址，重复时取消请求。
9. 检查ICache主流水各级流水线中正在处理的地址，重复时取消请求。

过滤完成后将请求发送至PrefetchQueue入队。

* + - 1. 请求发送

所有准备发送的预取请求由PrefetchQueue进行管理，PrefetchQueue采用FIFO结构，当TileLink端口空闲时，向L2 Cache发送Hint请求，把相应的缓存行预取到L2。同时会对主流水取指请求进行监听，当取指请求在PrefetchQueue命中时，取消对应的预取请求。

* + 1. 地址翻译错误、物理内存保护错误检查

如前所述，在ICache主流水线中会进行该类检查并通过error通道向后续单元传递错误信号。

* + 1. 异常处理基本规范

检查到的异常，如果为正常功能路径（真正的取指请求）触发，在检查到后将随流水线传递给IFU，随后在处理器后端生成异常信号，软件可进行相应处理；如果为预取路径触发，则在检查到后取消本次预取，相关异常不上报处理。

1. 总体设计

*总体设计的标准：对设计进行分解，完成子模块划分、顶层接口定义、关键控制电路/时序/数据流/控制流的设计。*

*总体设计面向的对象：顶层集成人员、验证人员、软件人员、设计人员*

* 1. 整体框图

*整体框图及文字说明。*

*图中不同属性的信号要注意区分开，比如控制信号与数据信号、一般信号与时钟复位信号等。控制流最好用数字标明先后关系。简洁起见，图中可以使用代号标记一组信号，然后在接口列表一节中进行解释，比如使用axi代表AXI接口的所有信号。*

*文字描述要简洁清晰，重点是描述模块外部接口，以及内部各子模块功能、模块之间的连接关系。*

*例如：*

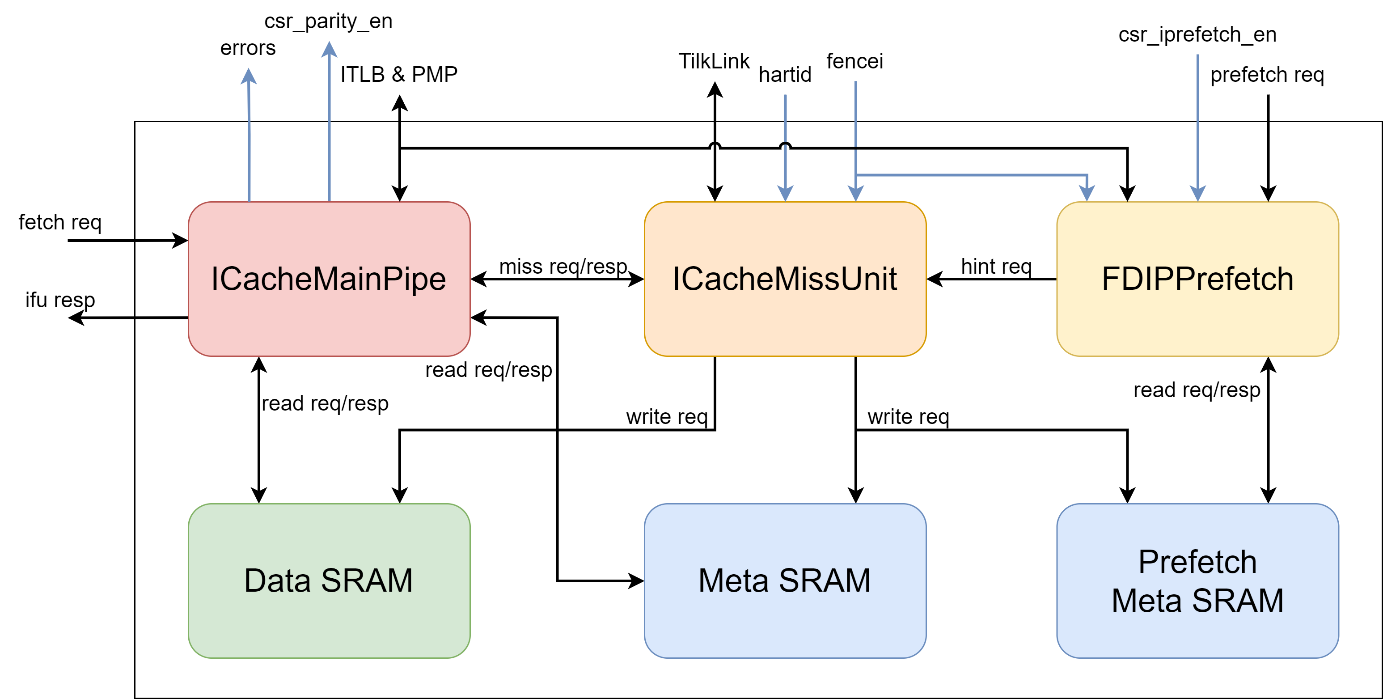
**

图3.1 ICache模块整体框图

* 1. 接口列表

*描述本模块所有的输入输出接口。接口列表是顶层集成的重要依据，需要准确清晰，并说明注意事项。*

*其中：*

1. *源/目的一栏描述源头模块和目的地模块*
2. *描述一栏需要列举信号每bit含义描述、取值范围、跟其他信号之间的约束（比如Xdata在Xvalid为高时有效、Xmode为2时signalA取值只能为1、2等等））*

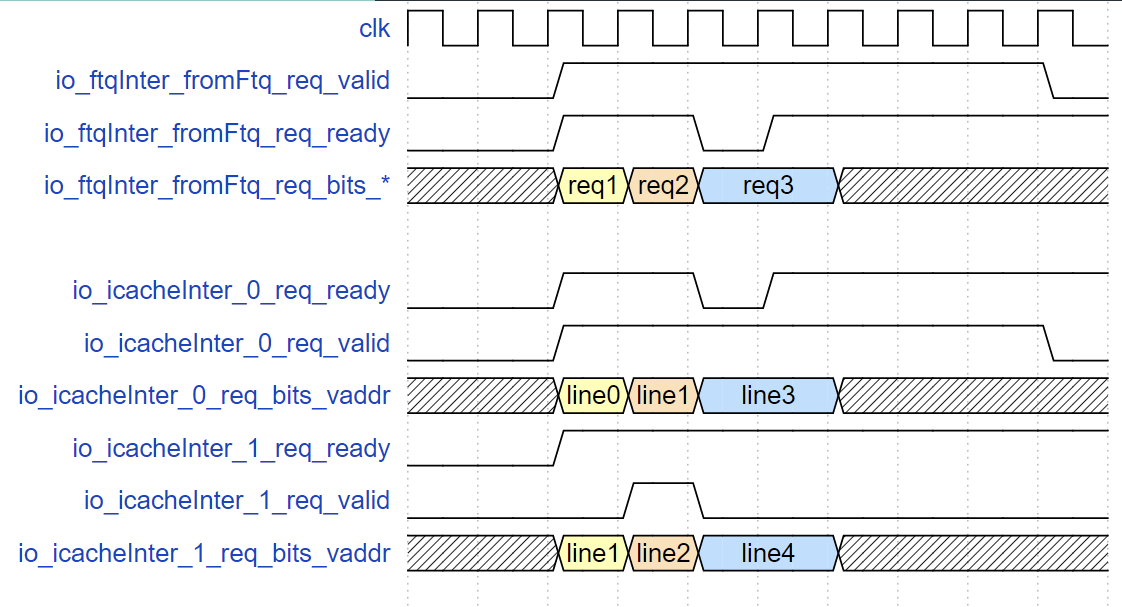
参照ICache接口文档

* 1. 接口时序

*顶层接口时序图*

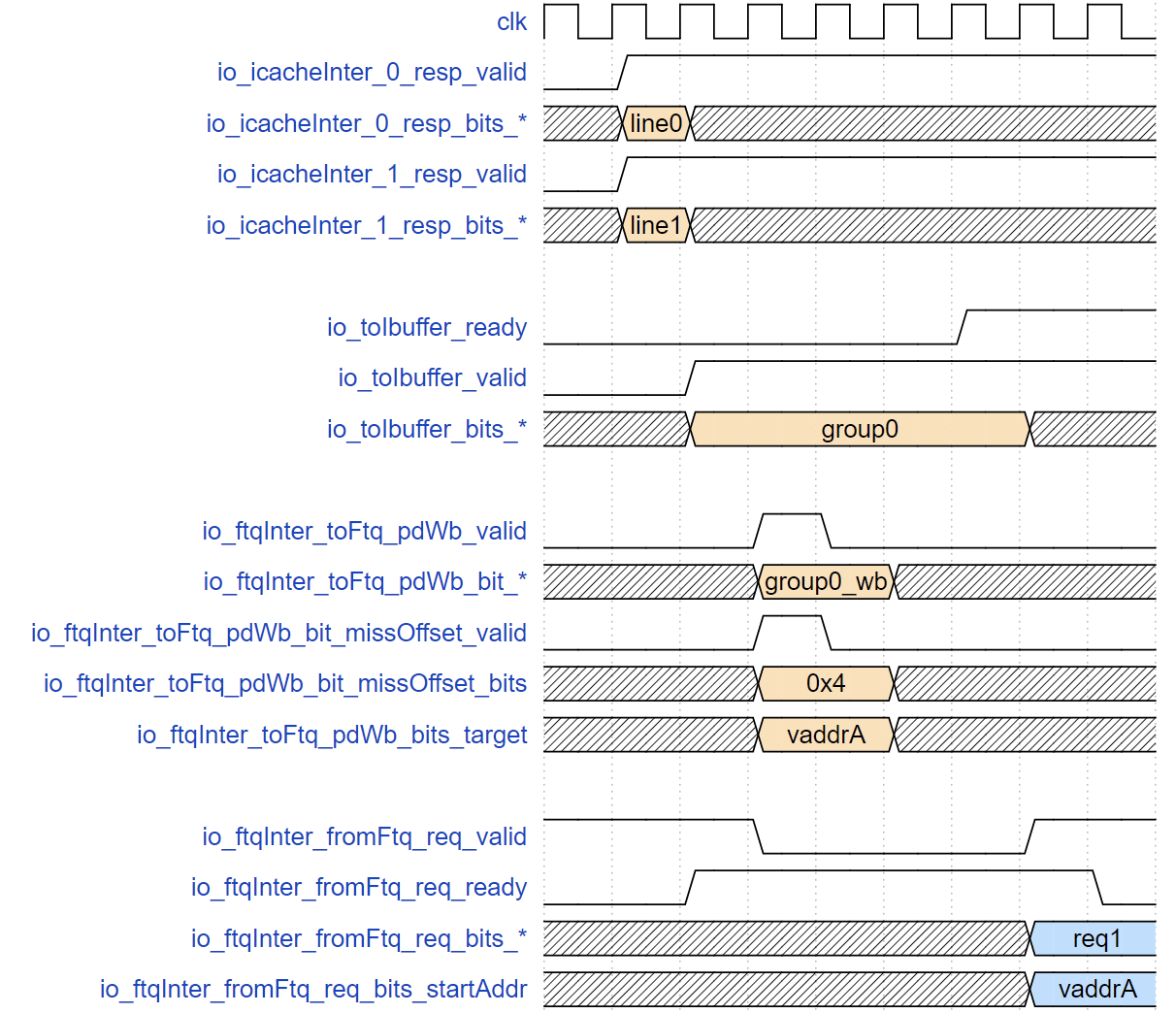
1. *非标准接口需要有关键信号的时序示意图*
2. *标准接口说明使用的接口协议即可（如AXI、APB等）*

*例:*

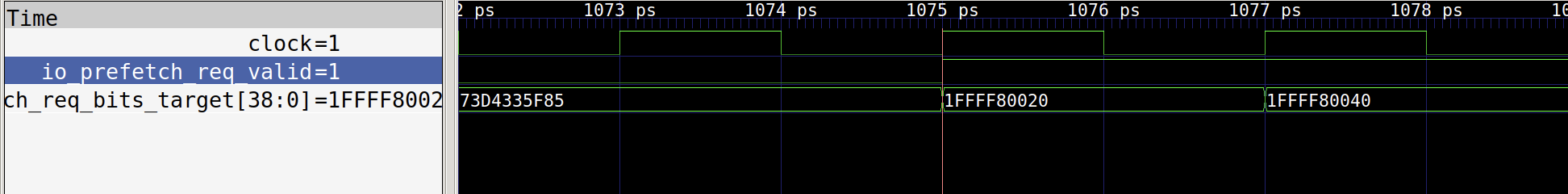
1. *FTQ请求接口时序示例*

*上图示意了三个FTQ请求的示例，req1只请求缓存行line0，紧接着req2请求line1和line2，当到req3时，由于指令缓存SRAM写优先，此时指令缓存的读请求ready被指低，req3请求的valid和地址保持直到请求被接收。*

1. *ICache返回接口以及到Ibuffer和写回FTQ接口时序示例*

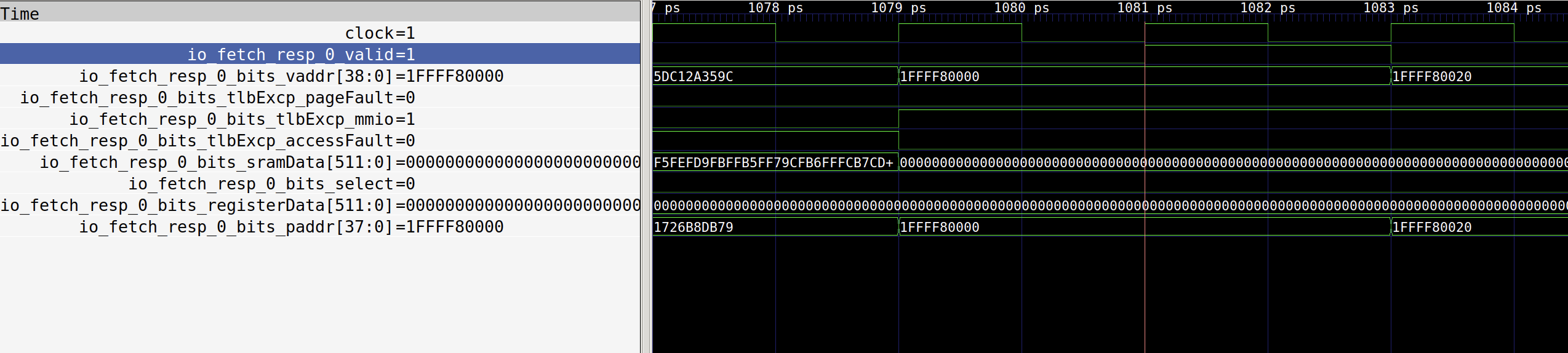
*上图展示了指令缓存返回数据到IFU发现误预测直到FTQ发送正确地址的。。。。。*。

1. auto\_client\_out\_\*信号为标准Tilelink接口
2. prefetch请求接口



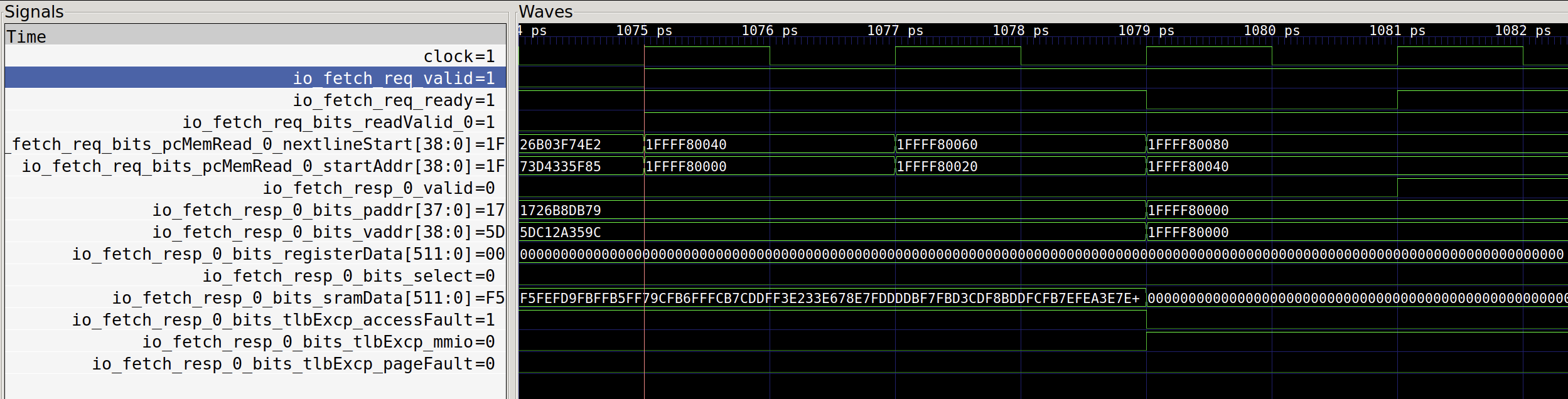
上图展示了连续两个prefetch请求，分别发送了0x1FFFF8020和1FFFF80040地址的预取请求。

1. fetch请求回复接口



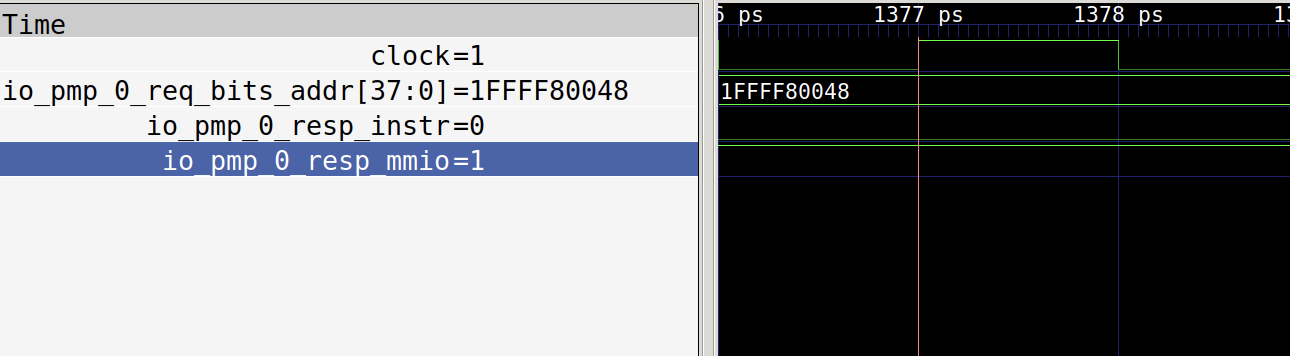
上图展示了一个fetch回复请求，为虚地址和物理地址都为0x1FFFF80000的请求回复了一个经查询为MMIO地址的结果，不包含有效的缓存行数据。

1. fetch请求接口



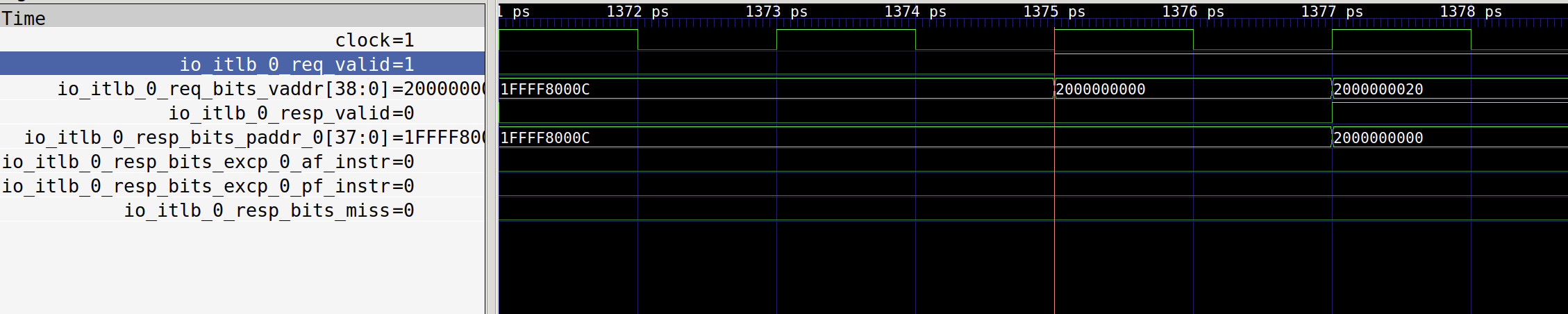
上图展示了一个fetch请求，虚地址为1FFF80000。

1. pmp请求与回复



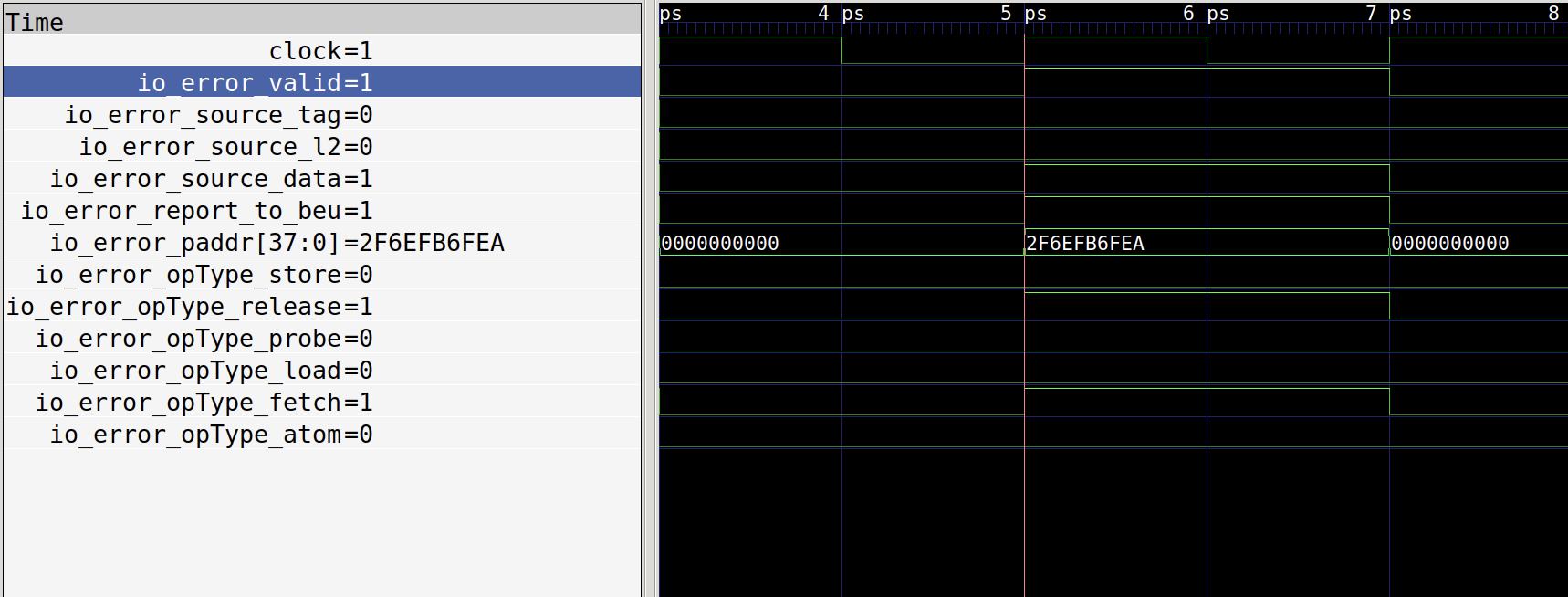
上图展示了一个PMP请求在当拍得到回复表明该区域为MMIO。

1. ITLB查询与回复接口



上图展示了一个ITLB查询0x2000000000地址翻译结果并在下一拍得到相应地址回复的时序。

1. error接口



上图展示了一个error通道有效信号，该信号展示了一个触发总线错误的release请求。

* 1. 时钟复位

*描述时钟相关设计：时钟域划分、时钟信号来源、时钟频率、时钟门控等。*

*描述复位相关设计：复位信号来源、异步复位还是同步复位、复位过程、解复位过程。*

*描述时钟信号与复位信号的对应关系，例如：*

|  |  |  |
| --- | --- | --- |
| Module | Clock | Reset |
| Core | Clk\_core | rst\_core\_logic\_n  rst\_core\_cfg\_n |
|  | Clk\_iref (apb\_slave) | rst\_iref\_n |
| DDRC/P | Clk\_ddr\_c\_p | Rst\_ddrc\_n  Rst\_ddrp\_n |
|  | Clk\_ddr\_axi | Rst\_ddri\_n |

* 1. 寄存器配置

*如果模块涉及寄存器配置（包括状态信息、统计信息通过寄存器上报软件）需要简要描述相关的寄存器，描述可使用表格，格式如下。*

*总体AS可忽略此节*

表4.2 ICache寄存器说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **寄存器** | **地址** | **复位值** | **属性** | **描述** |
| spfctl | 0x5C1 | 64’d3 | RW | bit63-2: 保留  bit1: L2缓存预取配置信号  bit0: 指令缓存预取配置信号 |

*注：RO——只读寄存器；RW——可读可写寄存器*

* 1. 补充说明

*可选项。按照模块特点，根据4.1整体框图的划分，补充部分核心模块、关键电路、关键信号信号的说明。*

1. 模块设计

*二级模块及往下一般为详细设计范畴，可以放在这个文档中，也可以单独成文，如果是总体AS可以不涉及此章节。*

*模块设计的标准：能用于指导RTL代码的编写。理想情况下，RTL代码是对设计方案的翻译。*

*模块设计面向的对象：模块设计人员、模块验证人员*

*本模块下面各级子模块的详细设计说明。包括模块功能概述、模块IO、模块的设计框图、控制和数据流向、关键设计（流水线、memory（ram、fifo、寄存器组等）、主控制电路（包含不限于状态机、仲裁、关键握手时序等））。*

*对于关键设计描述要求：*

1. *Memory（ram、寄存器组、fifo等）：宽度、深度、接口含义、读写时序、data的详细描述、data在memory中存放的格式等信息*
2. *流水：有流水线框图、每一级流水线描述*
3. *仲裁：仲裁策略、优先级处理等*
4. *状态机：有状态机设计图，需要有每个状态描述、状态之间的跳转条件、复位状态等。*

*例如：*

* 1. ICache Memory规格：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 名称 | bank数 | set | way | width | 含义 | 端口情况 | 特性 |
| Meta SRAM | 2 | 128 | 4 | 24bit | 24bit全部为tag数据 | 单口 | 上电复位，读保持 |
| Prefetch Meta SRAM | 1 | 256 | 4 | 24bit | 24bit全部为tag数据 | 单口 | 上电复位，读保持 |
| Data SRAM | 2 | 128 | 4 | 512bit | 512 bit全部为cache行数据 | 单口 | 上电复位，读保持 |

* 1. ICacheMainPipe
     1. 功能概述
        1. 接受请求

ICache模块顶层的fetch信号直接通向ICacheMainPipe模块，实际动作也在该模块内完成。fetch信号中两个req.valid信号一旦拉高，则视为请求有效，否则始终忽略请求中其他信号；其中第二个req.valid拉高时隐含了第一个req.valid也拉高的要求。当请求有效时，在本拍内，若当前不存在miss的请求，则SRAM的有效读请求会被从metaArray.toIMeta和dataArray.toIData分别发送给Meta和Data SRAM，若当前存在miss请求，则需等待miss的请求重填结束才可发送该请求。其中，发送给SRAM的请求所使用的set index来自请求的虚地址截取，isDoubleLine为传入两req.valid信号均有效。同时，无论当前是否有Miss的请求，只要当前存在有效的请求，就会利用itlb.req向ITLB发送查找请求，其中请求有效信号为本阶段传入的valid信号，size固定为3，虚地址及调试用的pc值为传入请求的虚地址。该接口在valid无效时其他信号同样无意义。

* + - 1. 读取SRAM及TLB结果

当ICache、ITLB和流水线都可进入下一阶段时请求进入到结果读取阶段。具体地，ICache可进入下一阶段的条件是两个fetch请求的ready信号都拉高，也即Meta和Data两块SRAM现在都可接受读请求。ITLB可进入下一阶段的条件是两个TLB查询端口的ready信号都拉高，也即两ITLB查询端口都可正常接受请求。流水线可进入下一阶段的条件是当前流水线内没有Miss正在处理的请求。

进入该流水级后，前一流水级传入的虚地址以及由输入通过组合逻辑生成的set index、仅第一个查询请求有效、双端口查询均有效等信号均被存储到本级的寄存器中。

从本流水级进入下一流水级的条件是收到了全部ITLB返回结果，如ITLB返回的miss信号为高，则在该流水级继续发送请求，直到ITLB完成重填miss信号为低。

得到ITLB返回的物理地址后利用其作为tag与SRAM读出的tag比较，选出命中的Cache行，当两个端口都发送了查询请求时，hit的条件是两个端口都有命中。此外，在本流水级，还会根据配置的Cache替换策略调用对应逻辑选出被替换的Cache行。

若ITLB返回page fault信息，则会生成相关判断信号，等待下一流水级和PMP产生的fault一起处理。page fault触发条件为当前虚地址的翻译关系在页表中无效或所记录权限不允许执行操作。

* + - 1. 返回结果

如前所述的条件下，若本流水级下之前的重填操作已经结束或当前阶段空闲，上一流水级的取指请求可进入本流水级。上一流水级从最早流水级接收的虚地址等数据以及上一流水级生成的端口命中、bank miss、waymask、待替换Cache行、tag匹配信息、读出的Cache行也将被存入本流水级寄存器中。

若没有发生任何miss，本流水级会将暂存下来来的物理地址作为请求地址，地址大小恒定取3发送给PMP单元做物理内存保护检查，结果当拍返回。其查询valid条件为本流水级有效且不处于miss重填状态，valid无效时其他数据均无意义。若返回结果触发了PMP错误或曾在上一级触发了TLB错误，则生成本流水级包含异常的信号，这两种信号都仅在给IFU的输出valid拉高时有意义。

具体地，若PMP寄存器内配置禁止了当前特权级的访问或为MMIO地址，则会触发对应错误信号（access fault/mmio）拉高。

若没有触发任何错误和miss，则本流水级将向IFU返回从Data SRAM中读取的Cache行、物理地址、虚地址信息，此时异常相关信号均应已经拉低。

* + - 1. 一致性

ICache的一致性由软件维护，当收到一致性维护信号时需要同时对ICacheMainPipe冲刷，冲刷每一级流水并且将模块内部暂存的数据置为无效。具体行为如下：

当io\_fencei信号为高时，刷新s1流水线，并且不允许s0和s1 fire。

当io\_fencei信号为高并且s2\_valid为高时，寄存io\_fencei信号，等到s2 fire也即s2处理完成后，冲刷missSlot中的数据，并且不向IFU发送响应。

当io\_fencei信号为高并且s2\_valid为低时，冲刷missSlot中的数据。

其中，s1、s2分别指ICacheMainPipe中的第一级和第二级流水；s0指与第一级流水相连的输入信号；missSlot是在s2 miss处理时的一个部件，由寄存器实现，里面存放从MissUnit返回的缓存行数据。

* + 1. 结构框图

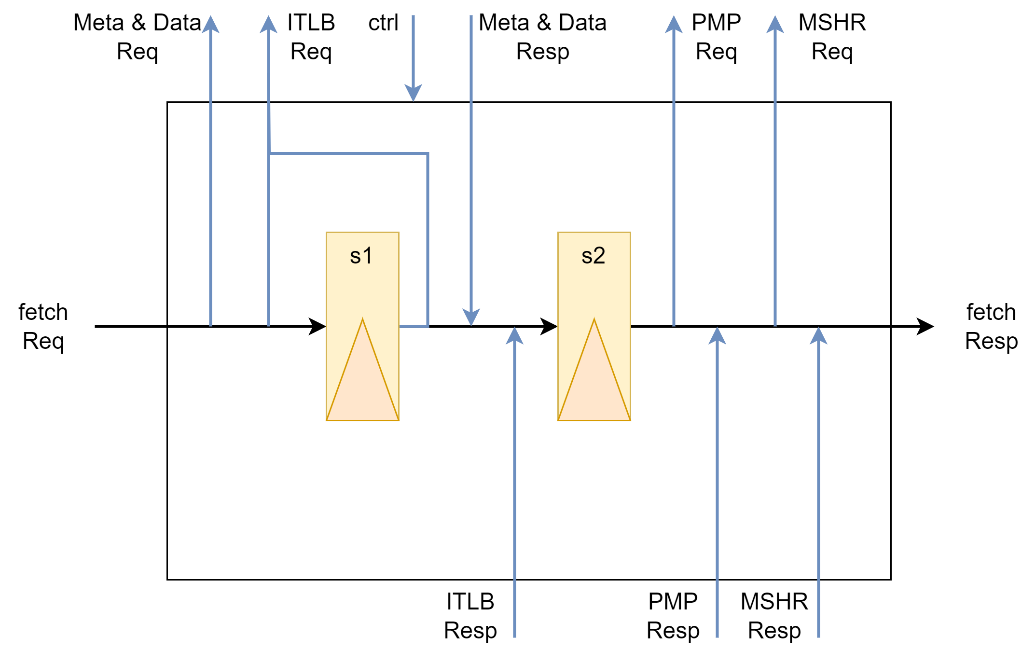


图5.1 ICacheMainPipe结构框图

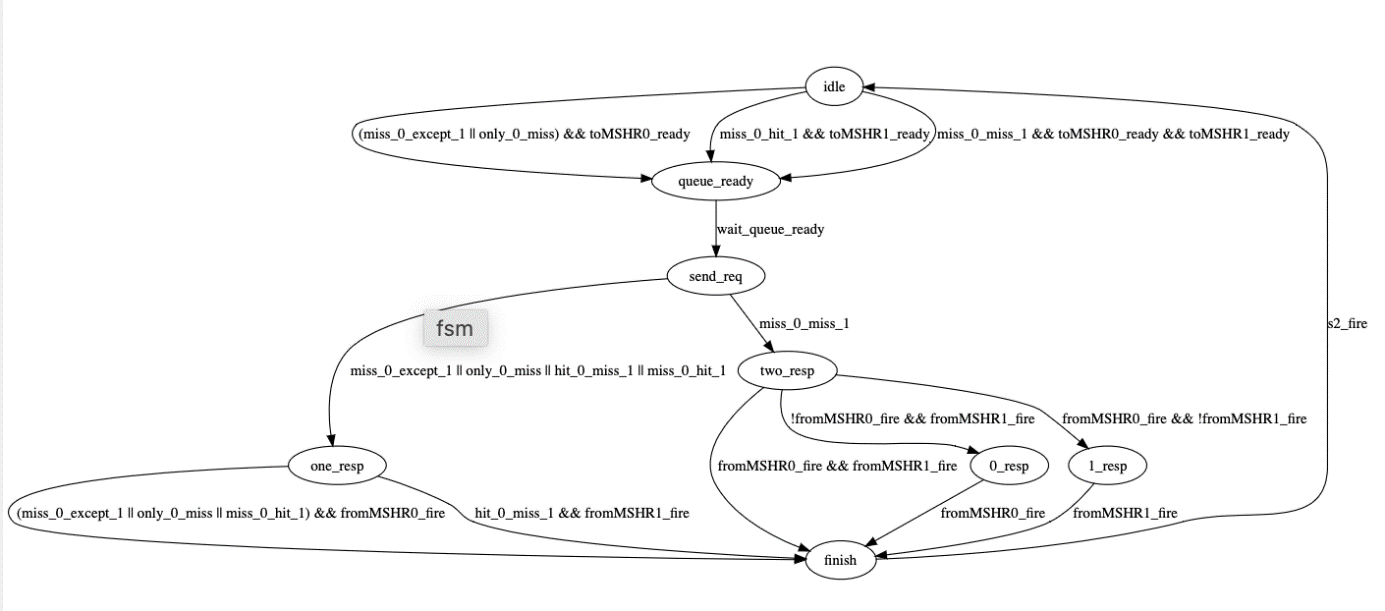
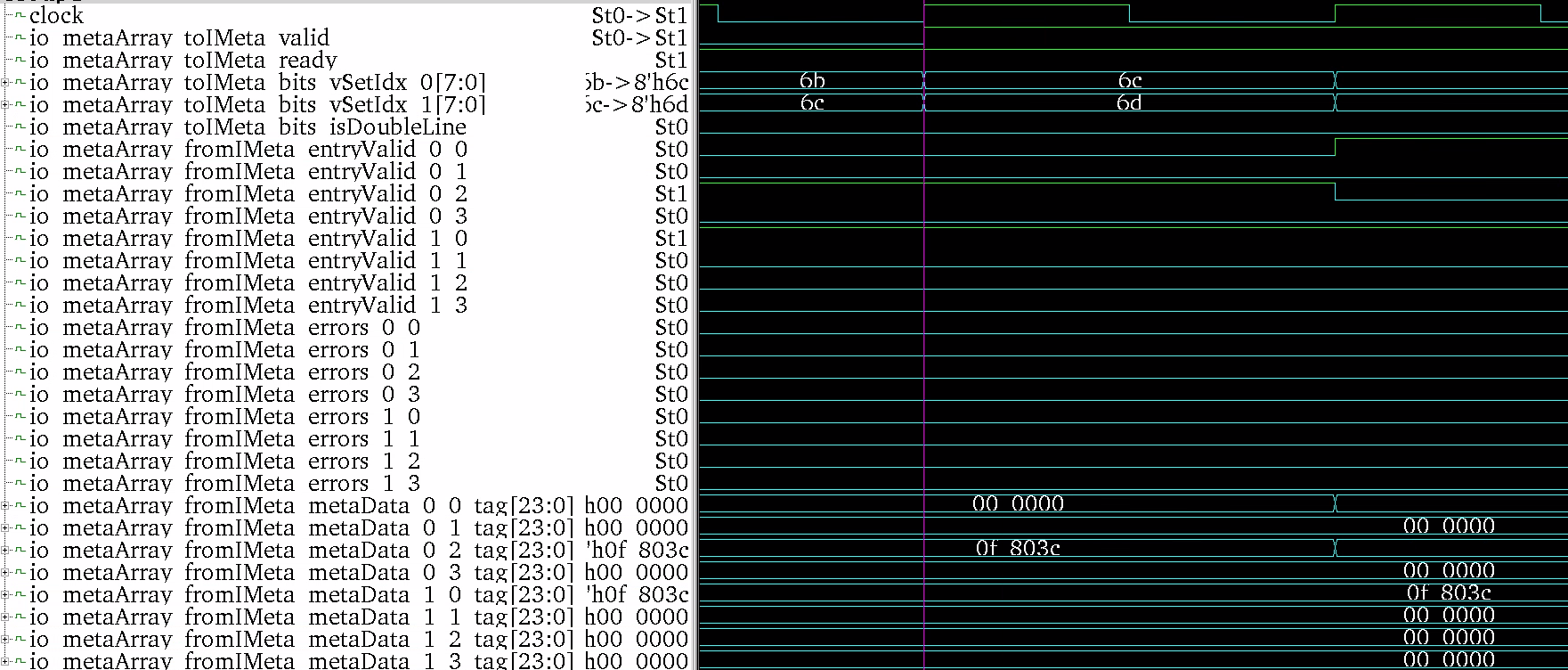


图5.2 ICacheMainPipe Miss状态机

* + 1. 接口列表

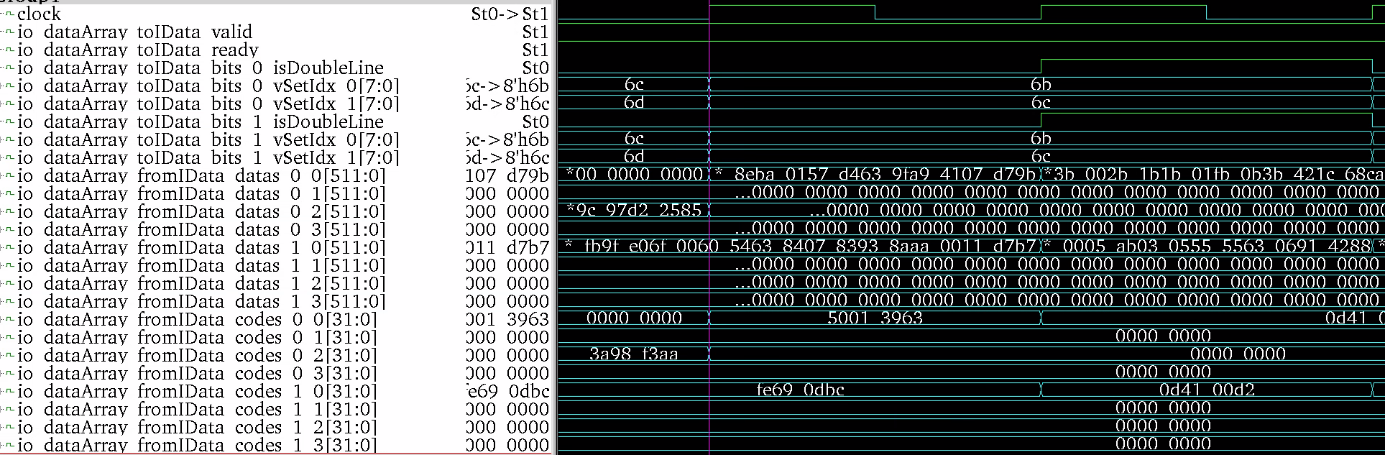
参照接口文档。

* + 1. 接口时序
       1. metaArray接口



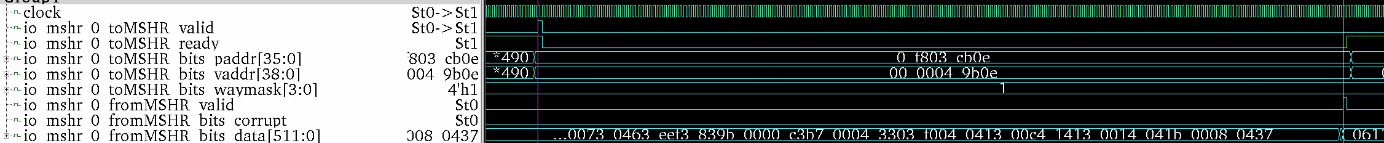
上图展示了metaArray接口时序，在请求有效一拍后收到来自meta SRAM的回复数据。

* + - 1. dataArray接口



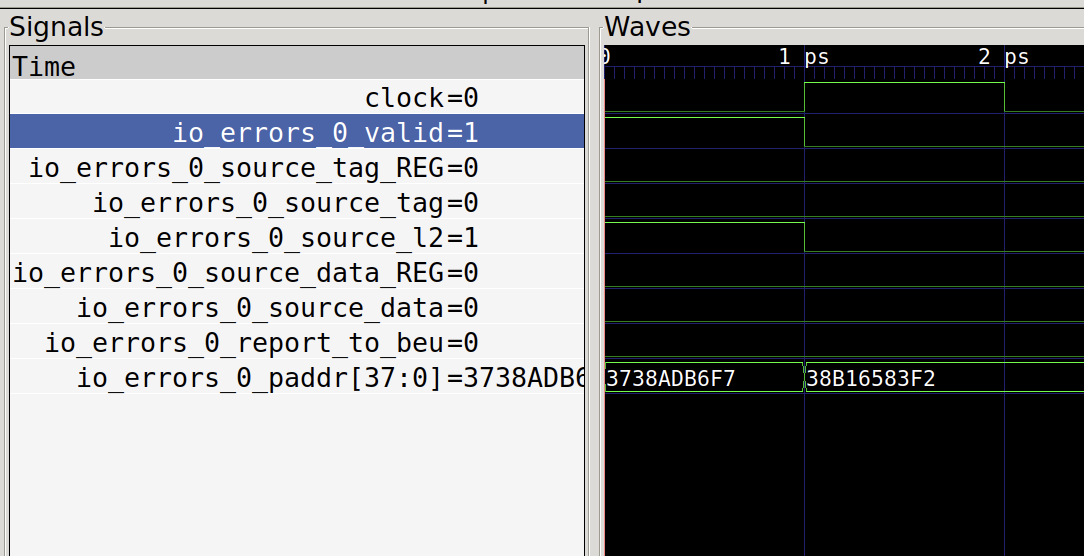
上图展示了dataArray接口时序，在请求有效一拍后收到来自data SRAM的回复数据。

* + - 1. MSHR接口



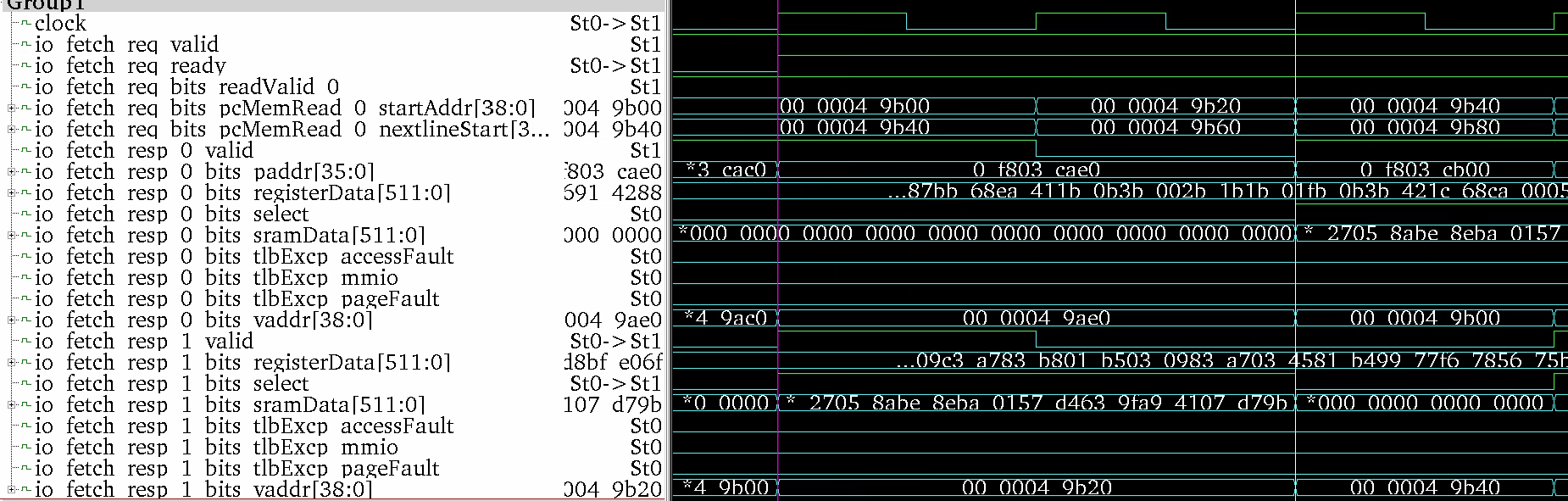
上图展示了MSHR接口时序，在请求被接收后经一段时间fromMSHR通道发回miss重填的数据。

* + - 1. error接口



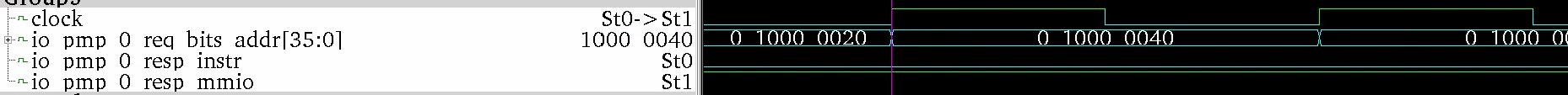
上图展示了error接口时序，当valid有效时其他信号有效。

* + - 1. fetch接口



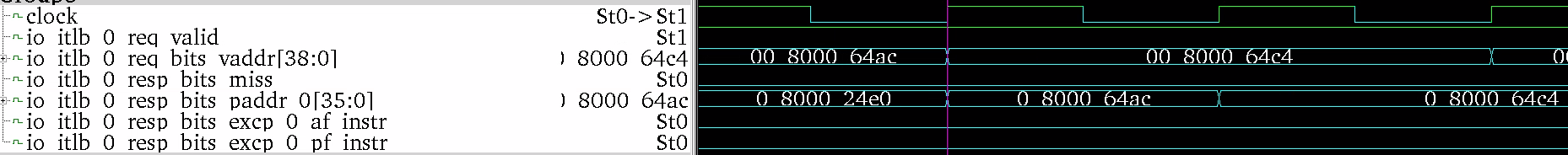
上图展示了fetch接口时序，当fetch请求信号握手成功一段时间后，回复端口信号valid有效并返回数据。

* + - 1. pmp接口



上图展示了PMP接口时序，在收到请求当拍，PMP返回对应地址的状态信号。

* + - 1. ITLB接口



上图展示了ITLB接口时序，在发送请求后的下一拍，返回数据。

* 1. ICacheMissUnit
     1. 功能概述

从L2取回数据用于填充及更新缓存。

如前所述，从L2取回数据主要动作在MissUnit完成，其发往MissUnit的请求生成在主流水线的最后一级。本流水级会根据两个Cache行的情况生成其分别的hit、miss信号及两Cache行hit/miss关系整合信号，当wait\_queue状态机位于ready状态时，根据miss状态拉高对应toMSHR端口的valid信号，表明发出了请求。请求内包含的物理地址、虚地址、waymask，仅在valid信号拉高时有意义。而一旦该请求拉高，missSwitchBit状态也将被拉高，表明当前存在正在执行的miss重填请求，后续请求将被阻塞在更早流水级避免覆盖请求。missStateQueue也在该请求时维护自身状态以便检查未来的请求是否和已有请求发生重合从而无需重复请求。

* + - 1. Acquire请求

MissUnit接收到来自Cache主流水线的请求，将miss请求存储到本模块内寄存器中，而后准备利用Tilelink总线在下一拍发送acquire block请求。Acquire请求的id为本模块的id，地址为请求的物理地址，宽度为一个Cache行宽度，opcode为4’b0100，表明请求类型；souce为2’b01或2’b10，分别代表MissUnit中的两个表项；user\_reqSource为4’b0001，表明类型是取指。

Tilelink Acquire信号fire后，本模块将等待tilelink总线的grant回复，mem\_grant将按beat返回数据及可能的数据损坏信号，本模块将返回数据全部暂存在respDataReg内，当refill结束后进入下一阶段。

* + - 1. 替换Cache行

当refill结束后，进入wait\_resp阶段，同时响应MainPipe和进行Cache行的替换和。拉高Meta和Data SRAM的写入valid信号等待握手，所写入数据为其Acquire阶段读到暂存的数据。同时拉高MainPipe resp输出valid信号并在该通道发送acquire取回的数据和是否损坏的corrupt信号给主流水线。

* + - 1. TileLink总线仲裁

MissUnit中有两个ICacheMissEntry，每个entry可以处理一个请求，另外发往L2的预取请求也通过MissUnit进行发送，通过TLArbiter进行三选一的仲裁，其中ICacheMissEntry的优先级高于预取请求。

当TileLink总线返回数据时，再根据数据的id将数据送往对应的请求来源。

* + - 1. 一致性

MissUnit通过软件维护一致性，当miss处理过程中fencei信号为高时，则在该miss处理完成后不将该请求写入SRAM。

* + 1. 结构框图

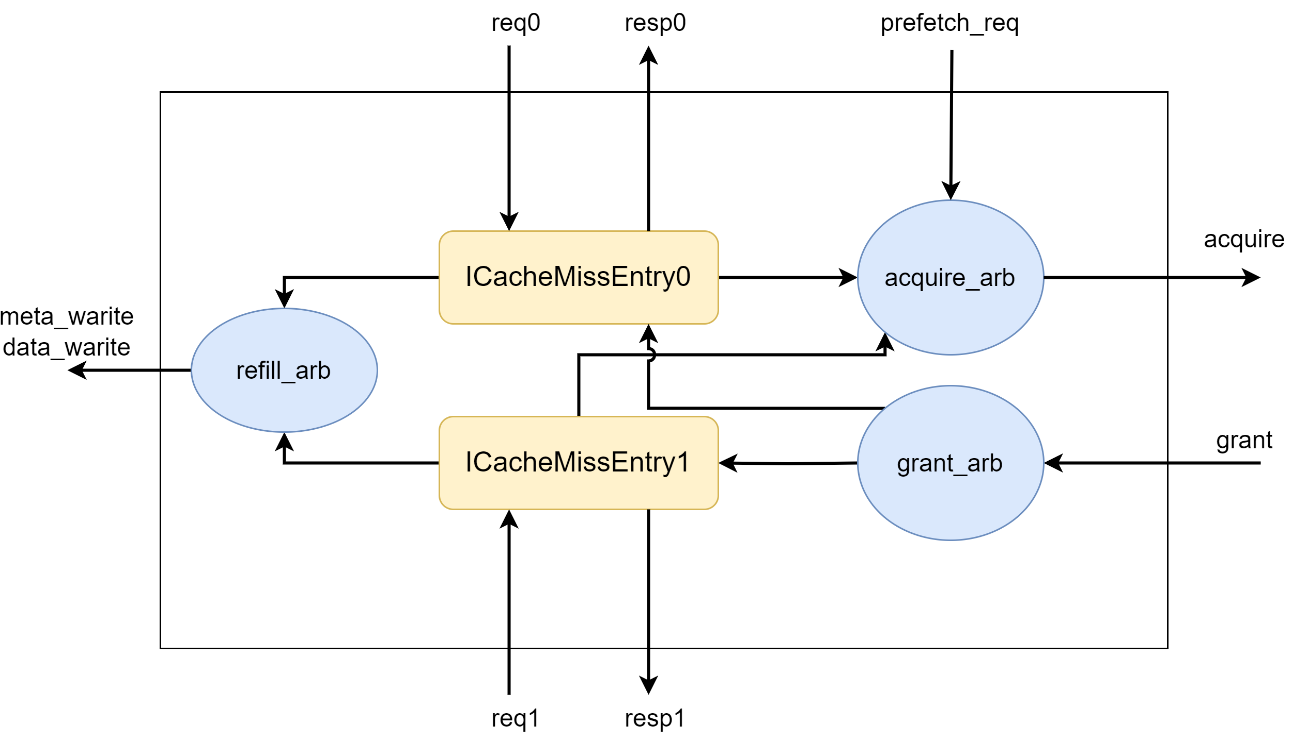
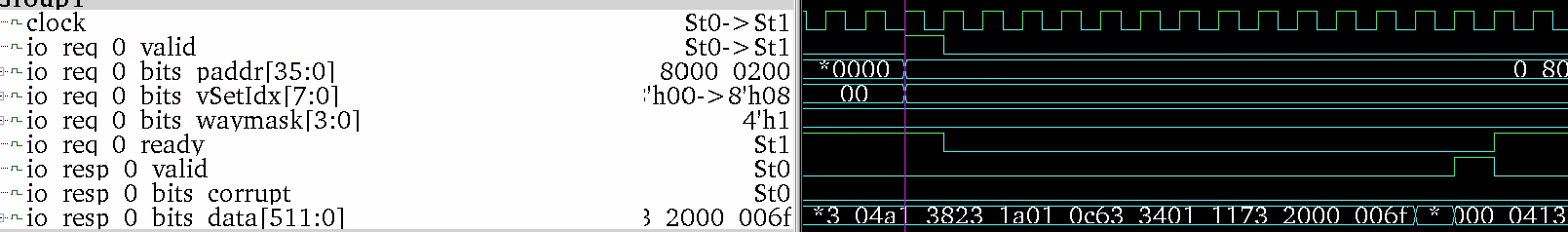


图5.3 ICacheMissUnit结构框图

* + 1. 接口列表

参照接口文档。

* + 1. 接口时序
       1. Miss请求接口



上图展示了miss请求握手后一段时间收到对应回复的时序。

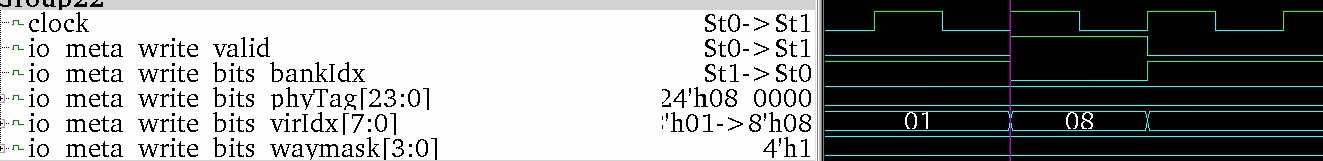
* + - 1. mem\_acquire接口

标准Tilelink A通道。

* + - 1. mem\_grant接口

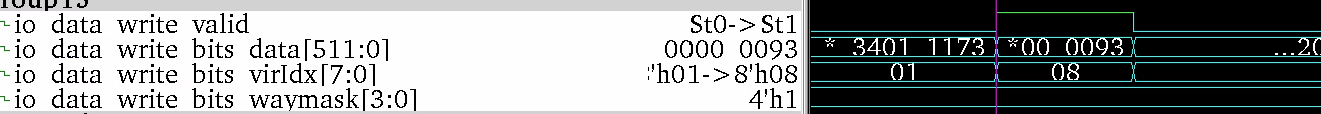
标准Tilelink D通道。

* + - 1. meta write接口



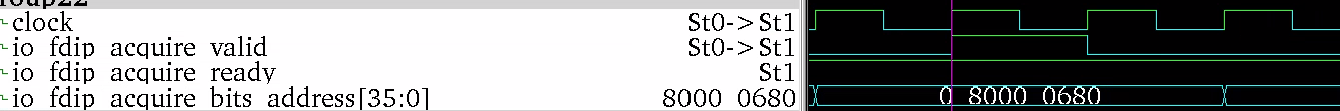
上图展示了meta写操作接口，写操作握手后一拍完成实际写入。

* + - 1. data write接口



上图展示了data写操作接口，写操作握手后一拍完成实际写入。

* + - 1. prefetch接口



上图展示了一次预取请求的时序。

* + 1. ICacheMissEntry
       1. 功能概述

ICacheMissEntry是ICacheMissUnit内实际完成各项操作的功能单元，内部由图5.4所示的状态机进行控制，功能与前文所述一致，共在ICacheMissUnit内实例化2个，可并发处理2个请求。

* + - 1. 结构框图

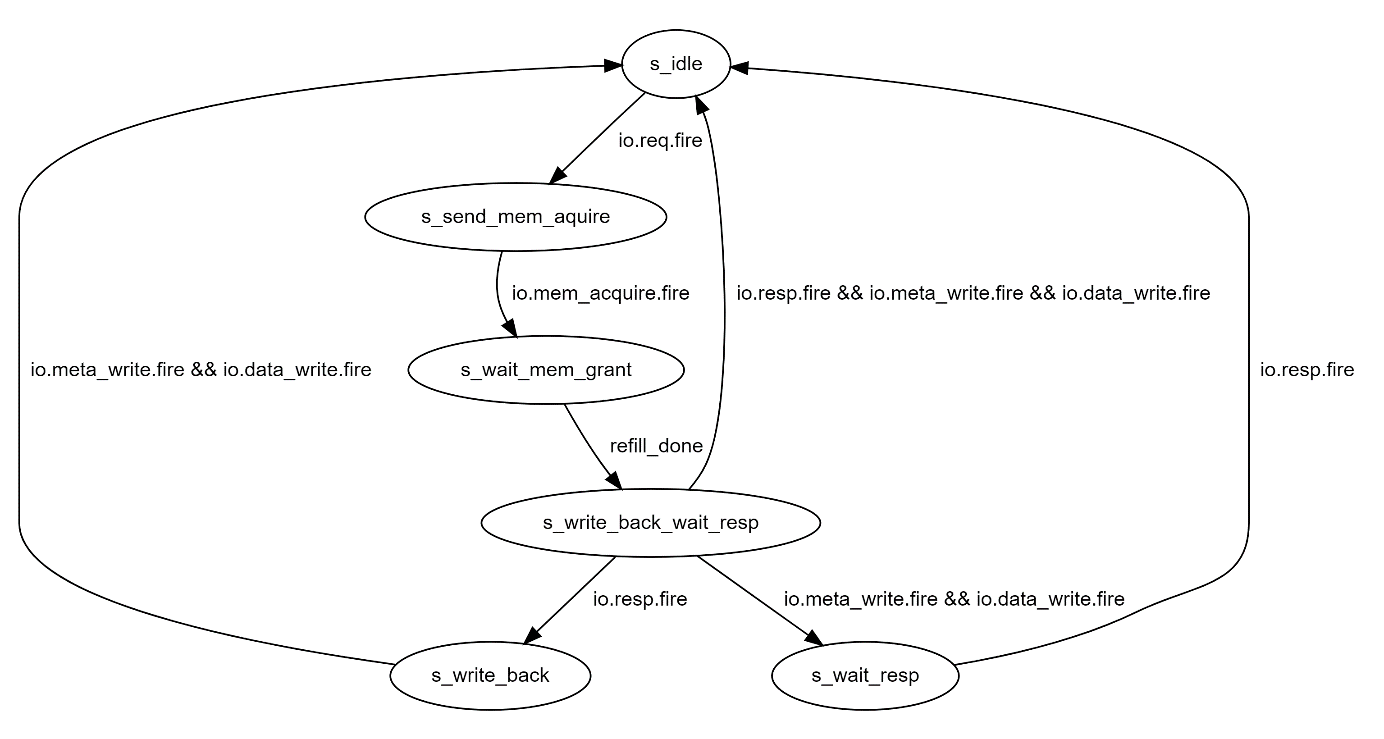
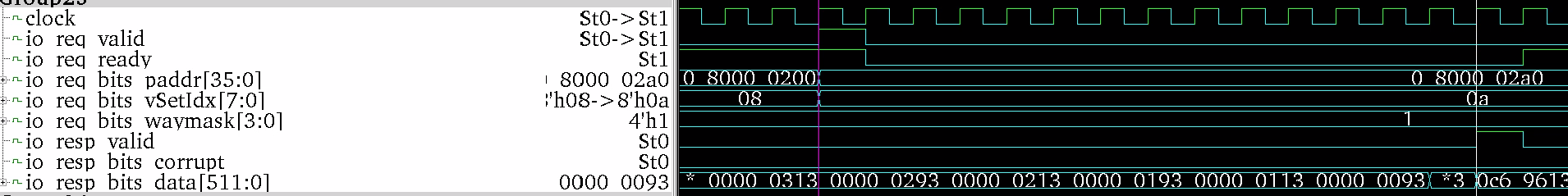


图5.4 ICacheMissEntry状态机

* + - 1. 接口时序
         1. miss请求接口



上图展示了miss请求握手后一段时间收到对应回复的时序。

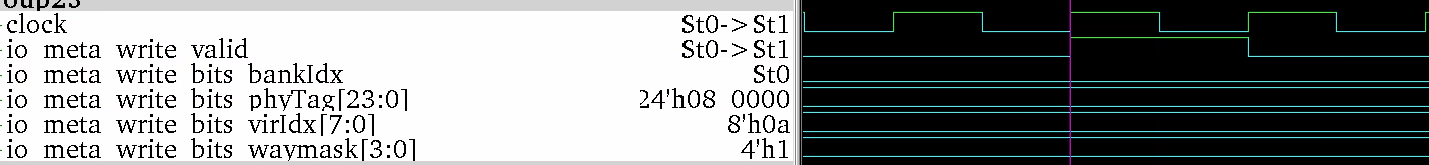
* + - * 1. mem\_acquire接口

标准Tilelink A通道。

* + - * 1. mem\_grant接口

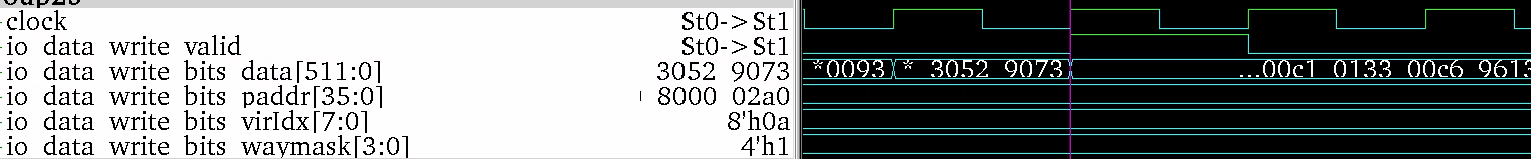
标准Tilelink D通道。

* + - * 1. meta write接口



上图展示了meta写操作接口，写操作握手后一拍完成实际写入。

* + - * 1. data write接口



上图展示了data写操作接口，写操作握手后一拍完成实际写入。

* 1. FDIPPrefetch
     1. 功能概述

ICache的预取策略为FDIP（Fetch Direct Instruction Prefetch），根据分支预测器产生的预测地址，提前将对应缓存行的请求发送到L2Cache，若L2Cahce中没有该缓存行，就向下级存储发送对应的请求，将该缓存行放到L2Cache中。该功能由 FDIPPrefetch模块实现，其中子模块IPrefetchPipe负责接收FTQ发送的预取请求，并且进行过滤，过滤完成后将请求发向PrefetchQueue；子模块PrefetchQueue负责管理过滤后的请求，等到TileLink端口空闲时发送预取请求；另外有模块PrefetchMetaData在ICache中例化，由SRAM实现，是Meta SRAM的副本，存储数据与Meta保持一致。

* + 1. 结构框图

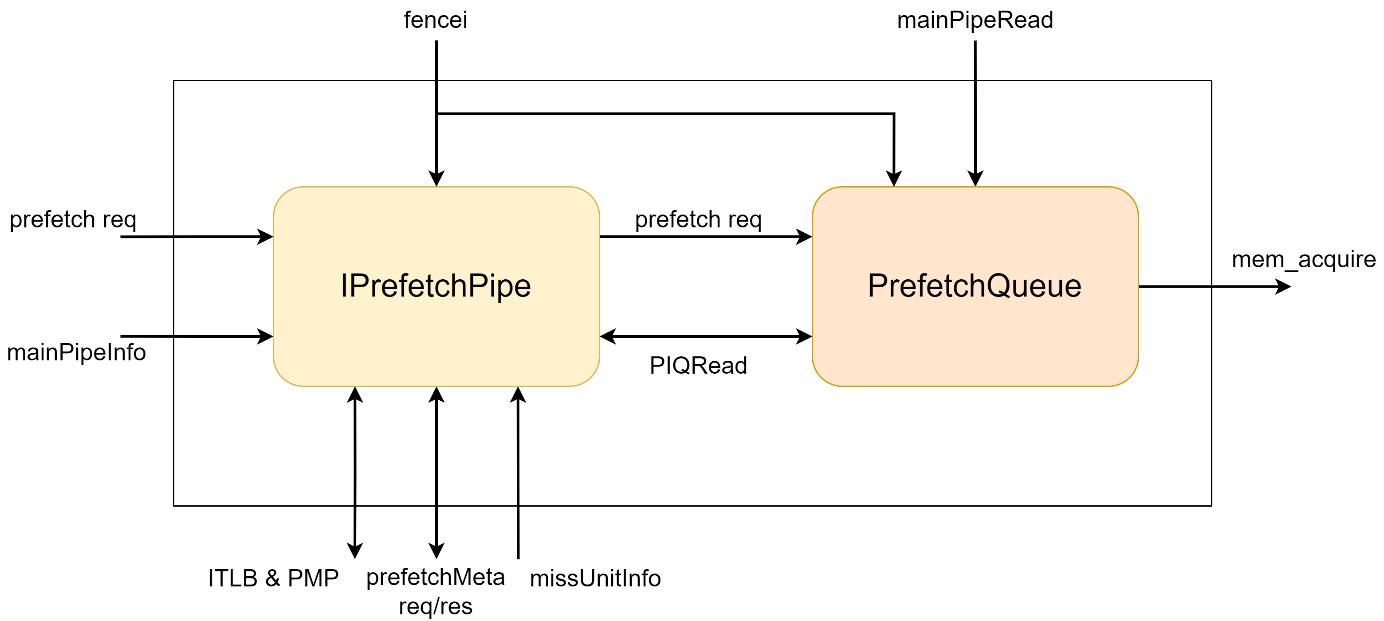
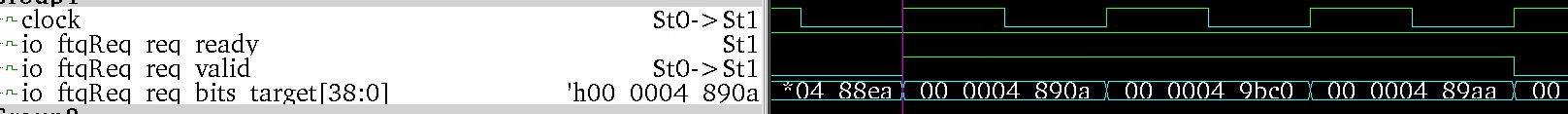


图5.10 IPrefetchPipe结构框图

* + 1. 接口列表

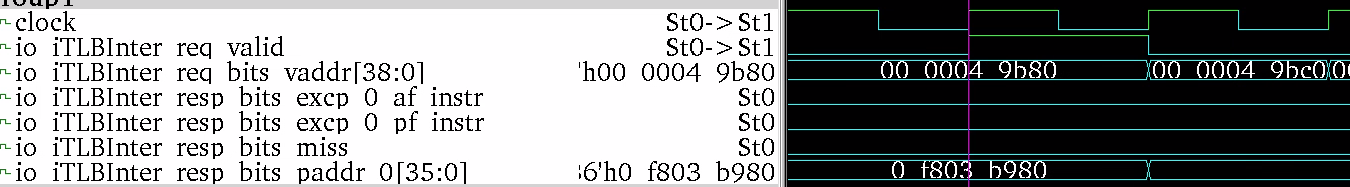
参照接口文档。

* + 1. 接口时序
       1. FTQ预取请求接口



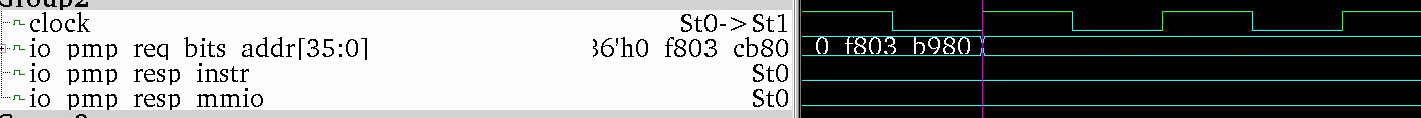
上图为连续三次FTQ预取请求握手，其中当前请求地址为0x4890a。

* + - 1. ITLB查询接口



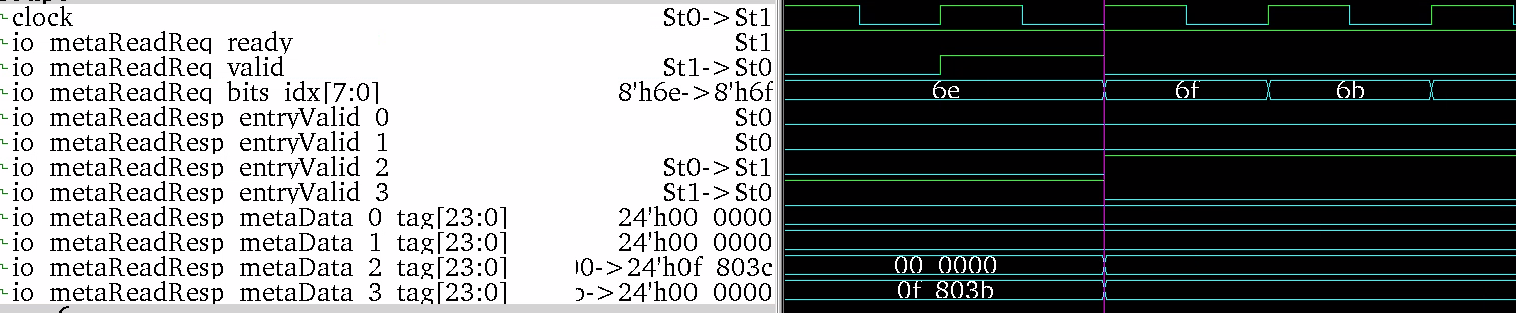
上图为一次命中的ITLB请求，其中当前请求虚地址为0x49b80。

* + - 1. pmp查询接口



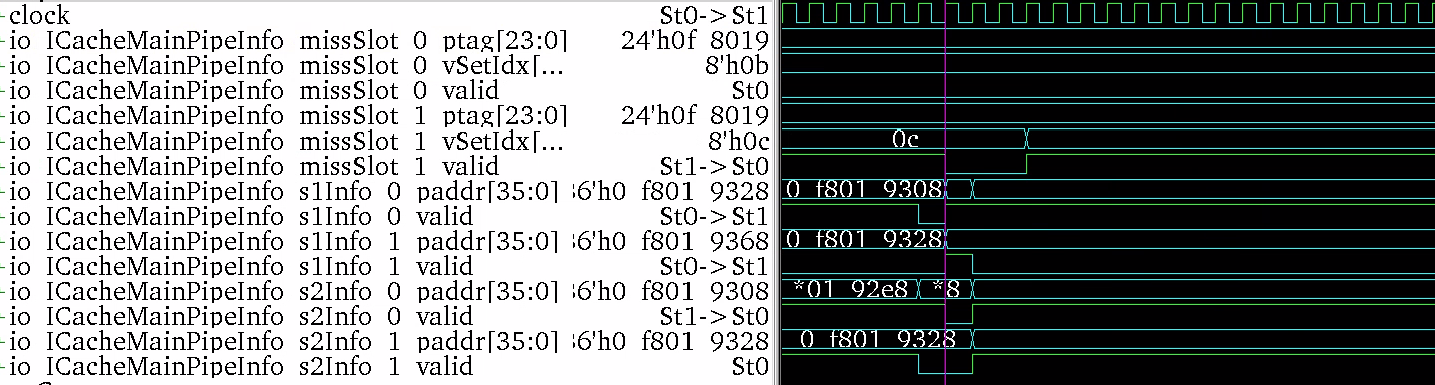
上图为一次PMP查询请求，在当拍返回检查结果。

* + - 1. PrefetchMeta读接口



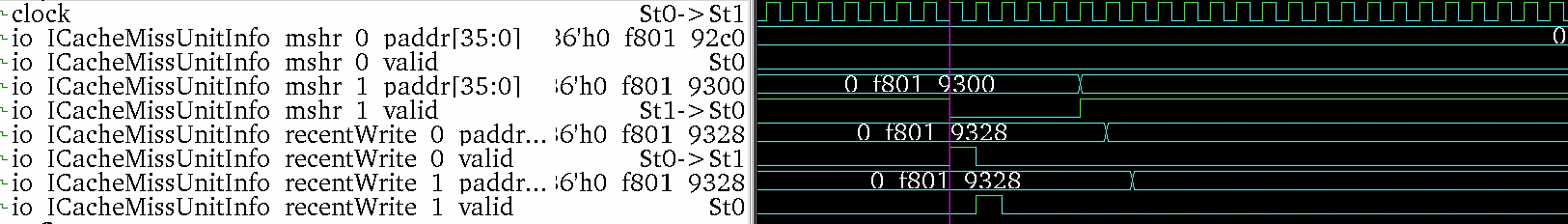
上图为一次PrefetchMeta读请求，在握手后一拍获得读回meta数据。

* + - 1. mainPipeInfo接口



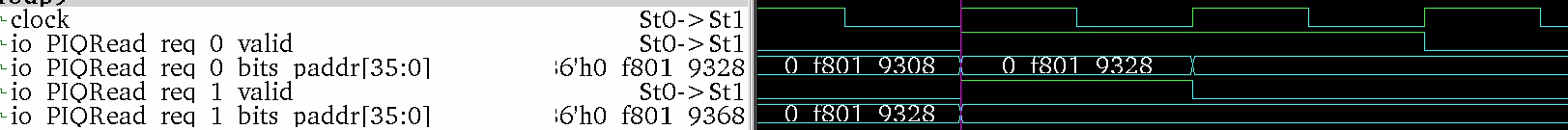
上图为ICacheMainPipe中missSlot和各级流水线的状态信息，通过mainPipeInfo接口传递。

* + - 1. missUnitInfo接口



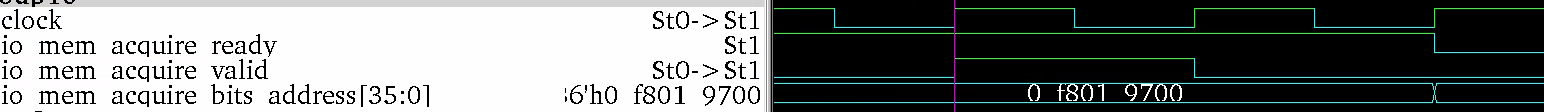
上图为ICacheMissUnit中MSHR和最近写入SRAM的地址信息，通过missUnitInfo接口传递。

* + - 1. PIQ 读接口



上图为一次来源于ICacheMainPipe的读请求，该请求只是为了检查PrefetchQueue中是否又相同的请求，不需要响应。

* + - 1. mem\_acquire接口



上图为一次mem\_acquire握手，将请求通过ICacheMissUnit转发到L2Cache。

* + 1. IPrefetchPipe
       1. 功能概述

在IPrefetchPipe一共有两级流水线，在每一级流水线对预取请求进行过滤。预取请求由FTQ直接发出，该请求对应的缓存行可能已经在data SRAM中存在，也可能正在ICacheMainPipe和ICacheMissUnit中处理，所以需要对这些情况进行过滤。同时还需要对请求进行ITLB地址翻译和PMP权限检查，如果出现异常则取消该预取请求。

在请求进入IPrefetchPipe的当拍，执行以下操作：

1. 将请求发送至prefetchMeta，读取meta中的数据。
2. 将请求发送至ITLB进行地址翻译。
3. 检查请求与当前流水线中的请求是否重复，重复时取消该预取请求。
4. 若fencei信号为高，则取消该预取请求。

当prefetchMeta的ready信号为高并且该预取请求不需要取消时进入下一级流水线。

在第一级流水线，执行以下操作：

1. 接收ITLB返回的数据，若发生miss或者产生异常时取消该预取请求。
2. 接收prefetchMeta返回的数据，直接寄存到下一流水级处理。
3. 若fencei信号为高，则取消该预取请求。

当下一级流水线空闲时并且预取请求不需要取消进入到下一级流水。

在第二级流水线，执行以下操作：

1. 将请求与上一级流水线中寄存的meta数据对比，如果命中则取消该请求。
2. 进行PMP权限检查，在当拍返回结果，产生异常时取消该请求。
3. PrefetchQueue检查，向PrefetchQueue发送读请求，当请求在PrefetchQueue的PIQ中命中时取消该请求。
4. MainPipeInfo检查，检查该请求所在的缓存行是否正在MainPipe中处理，若存在则取消该请求。
5. MissUnitInfo检查，检查该请求所在的缓存行是否正在MissUnit中处理或者与最近写入SRAM的缓存行重复，若存在则取消该请求。
6. 若fencei信号为高，则取消该请求。

如果预取请求不需要取消时向PrefetchQueue发送入队请求。

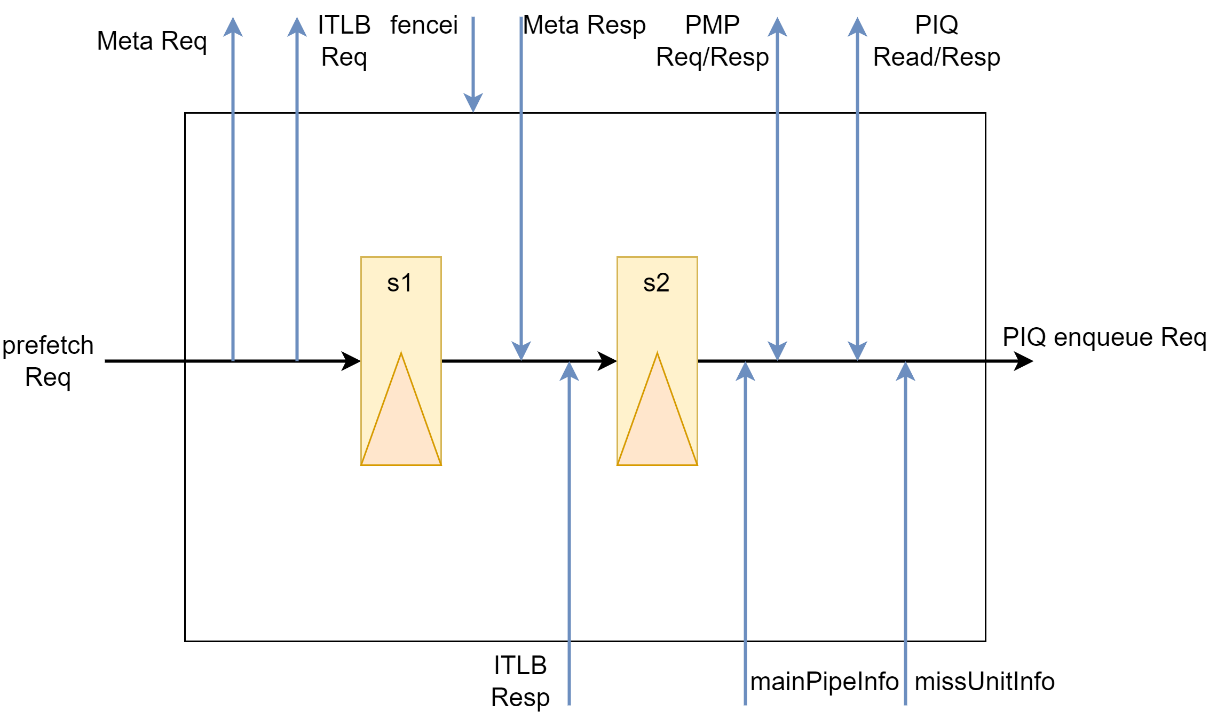
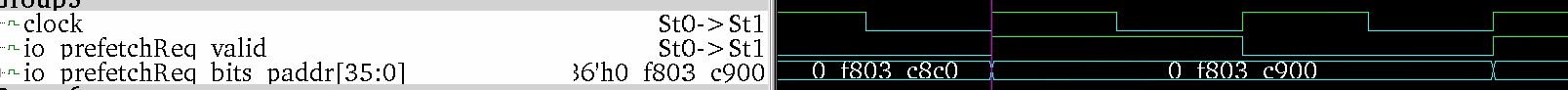


图5.11 IPrefetchPipe结构框图

* + - 1. 接口列表

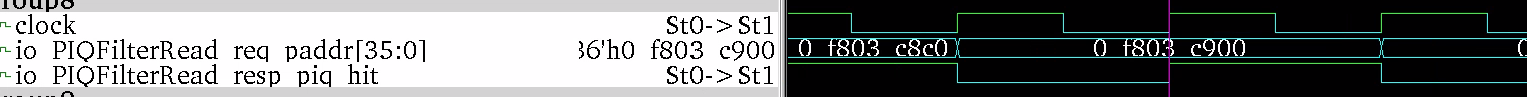
参照接口文档。

* + - 1. 接口时序
         1. PrefetchQueue入队请求接口



上图为一次入队请求，其中当前请求地址为0xf803c900。

* + - * 1. PrefetchQueue查询接口



上图为一次PrefetchQueue查询请求，其中当前请求地址为0xf803c900。

其它接口与FDIPPrefetch直接相连，参照5.4.4节。

* + 1. PrefetchQueue
       1. 功能概述

PrefetchQueue对过滤后的预取请求进行管理，将入队的请求放到PIQ中，PIQ采用FIFO实现，深度为12。负责管理请求的入队、出队，以及处理IPrefetchPipe和MainPipe的相关请求。

PIQ采用非阻塞的方式，当PIQ满时，会将最前面的请求丢弃。

当出队时，将最前面的表项放入handleEntry中，handleEntry中存放正在处理的预取请求，handleEntry通过一个简单的状态机与ICacheMissUnit进行交互，当TileLink端口空闲时将请求发往L2Cache。

当收到IPrefetchPipe发送的读请求时，将请求的地址与PIQ和handleEntry中的请求地址进行对比，若命中则拉高在当拍返回的hit信号。

MainPipe在第一级流水中发送过滤请求，当收到MainPipe发送的过滤请求时，将请求的地址与PIQ和handleEntry中的请求地址进行对比，将命中的请求置位无效，不需要进行预取。

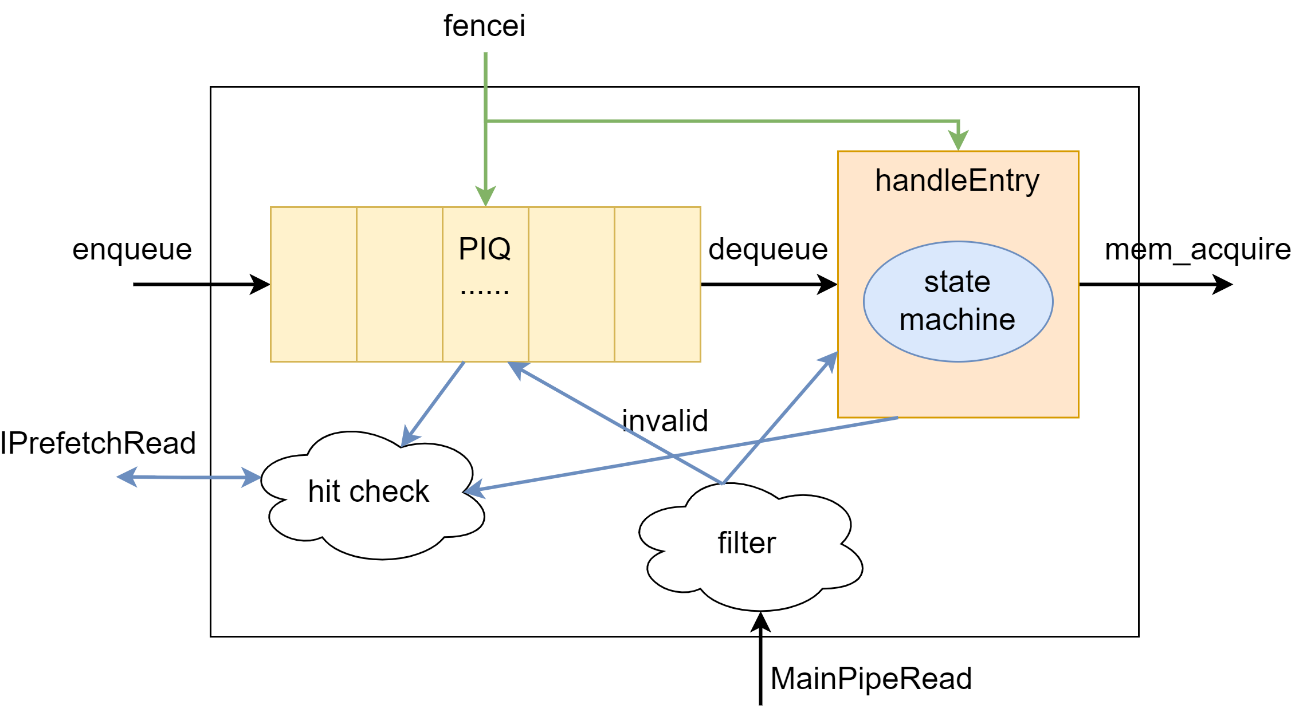
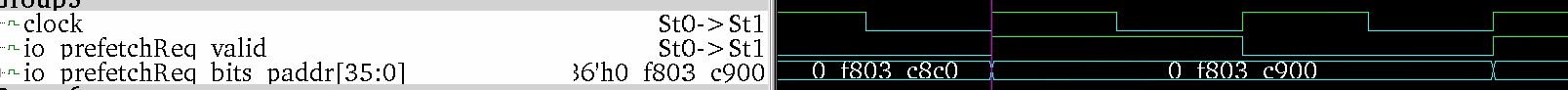


图5.12 IPrefetchPipe结构框图

* + - 1. 接口列表

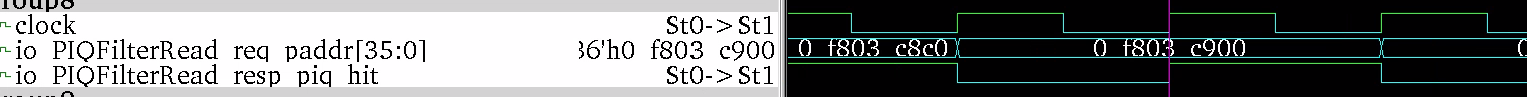
参照接口文档。

* + - 1. 接口时序
         1. 预取请求入队接口



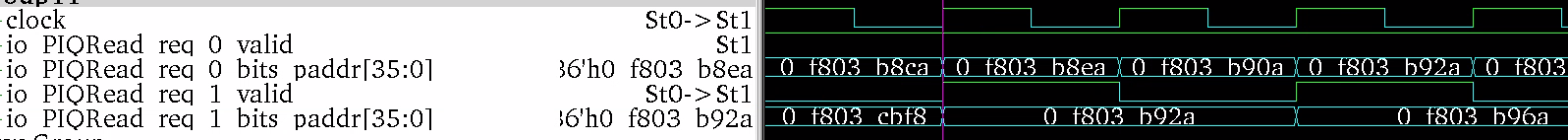
上图为一次入队请求，其中当前请求地址为0xf803c900。

* + - * 1. IPrefetchPipe查询接口

、

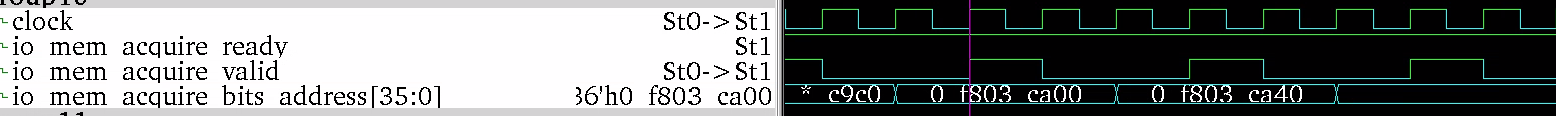
上图为一次PrefetchQueue查询请求，其中当前请求地址为0xf803c900。

* + - * 1. MainPipe过滤接口



上图为一次MainPipe过滤请求，不需要响应。

* + - * 1. mem\_acquire接口



上图为一次mem\_acquire握手，将请求通过ICacheMissUnit转发到L2Cache。

* 1. 二级模块B
  2. 关键电路

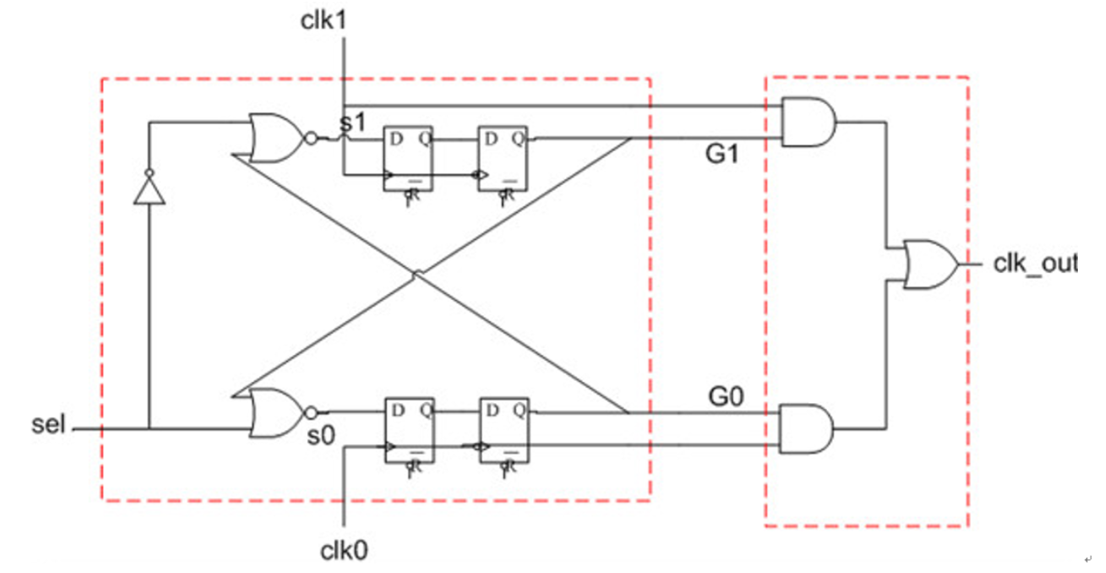
*例如：*

* + 1. 时钟切换电路

上述CRG框图中，紫色粗框内的clock\_mux是动态时钟切换模块，在用户程序配置PLL之后，通过配置寄存器cfg\_clk\_sel=1将系统时钟从晶振动态切换到PLL时钟。

为了保证时钟动态切换不会导致系统出错，需要使用无毛刺时钟切换电路，电路图如下：

图 2 无毛刺时钟切换电路图



上图中，下面两个寄存器的复位值为1，上面两个寄存器的复位值为0。复位时，clk\_out默认选择clk0时钟。

对于两级同步寄存器，同步器的第一级采用时钟上升沿触发，第二级采用时钟下降沿触发。

1. PPA

*内容包含:*

1. *Power*

*描述功耗设计目标*

*详细描述设计的功耗预估情况*

1. *Performance*

*详细描述设计的性能目标数据*

*详细描述性能的预估情况*

1. *Area*

*详细描述设计的面积目标数据*

*详细描述面积的预估情况*

*4.为优化PPA做的一些关键设计点（例如为了时序收敛做的一些面积/功耗/性能上的折中）*

1. 验证关注点

*从设计角度列举需要验证人员特别关注的测试点。*

对于ICahceMainPipe子模块，验证人员需要关注当io\_fencei信号为高时，模块的行为是否正确，预期行为参照5.2.1.4中描述。

1. Floorplan 建议

*芯片的floorplan考虑，依据数据流向，IO排布，模块大小进行芯片布局摆放设计*

*不涉及填“NA”*

1. 遗留问题

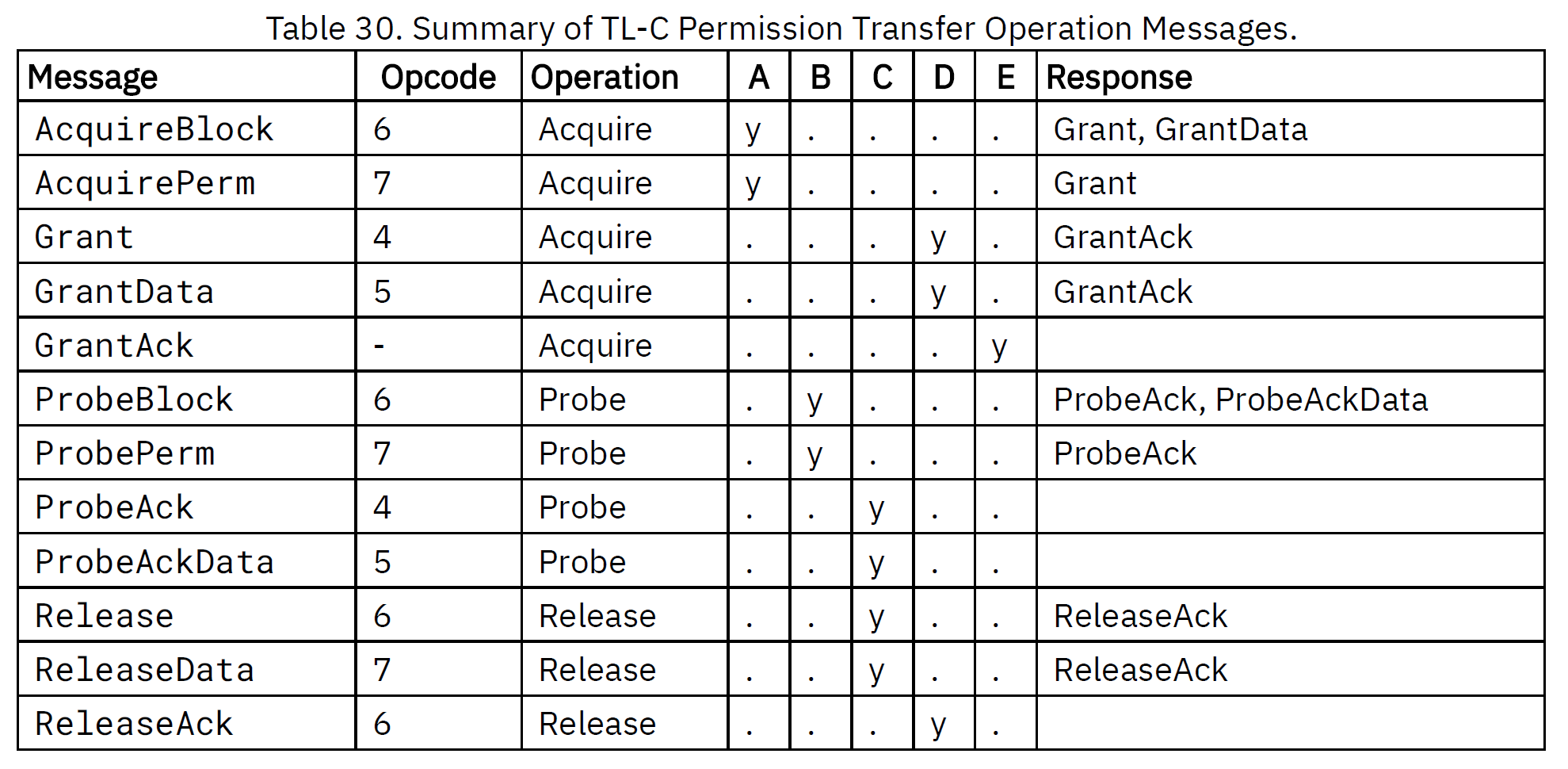
*需要跟踪的遗留问题*

*不涉及填“NA”*

1. 附录

# ICache相关Tilelink通道信号详细解释

* TL-C相关Message信息（Opcode，Operation以及通道、对应的响应message）



## Cache miss相关

* 涉及模块：ICacheMissUnit
* 涉及Tilelink message：AquireBlock/GrantData
* 注：以下表格从Tilelink手册中复制过来，前缀“a\_”表示A通道，具体信号可能是另外形式，比如“a\_opcode”在实际verilog模块里信号名为” auto\_client\_out\_a\_bits\_opcode”

