# FECON设计文档

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本号 | 修改人 | 修改时间 | 备注 |
| 1.0 | 杨翔瑞 | 2020.06.17 | 第一版 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

FECON将FAST的硬件流水线架构移植到基于FPGA的开源架构corundum中，从而使得FAST的流水线模型能够在corundum所支持的多种平台上运行，方便所有使用corundum作为基础平台的用户的开发工作。

FECON将支持的基本特征有：

1. 支持FAST的UM编程模型（包括metadata, PHV以及MID抽象）；
2. 支持FAST 的UA 接口；
3. 兼容目前corundum提供的基本功能（checksum, hash等）。

FECON将基于ExaNIC10G作为开发与调试的硬件平台，基于github进行协同开发。预计开发周期为1个月，在7月下旬完成前期开发任务（硬件部分），并在github发布。用户可下载并借助ExaNIC10G作为平台进行使用。9月前完成软件部分开发一起其它平台的兼容支持，从而能够使用FAST的软硬件协同所提供的全部功能。

## 引言

* 1. **背景与思路**

背景略。

Corundum的架构如图1所示。根据数据帧的流向corundum的数据通路可被分为TX与RX。从RX路径看，数据帧从MAC核进入FPGA的核心逻辑后采用了AXIS Stream协议进行传输。在Port模块中，数据流首先经过Hash模块、Checksum模块，然后交由DMA模块进行处理。而TX路径与之相反。

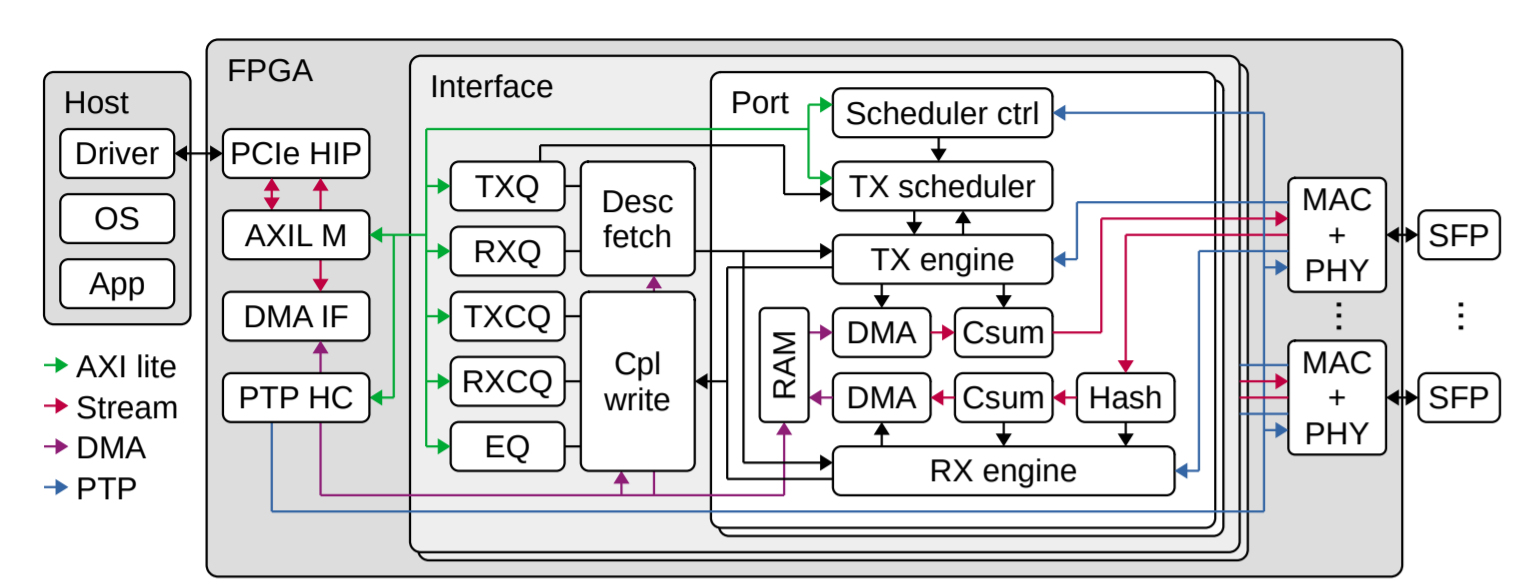


图1 Corundum网卡架构图

因此，FECON计划采用的方式是在RX路径的数据帧被交由DMA模块处理前（或TX路径的数据帧由DMA模块读出并发送前），交由插入的FAST的UM模块进行处理。其原理如图2所示。其中蓝色部分为FECON需要插入的模块。注意，由于目前corundum的设计未提供metadata的抽象，因此FECON需要利用AXIS Stream的接口进行扩展，从而为每个数据帧提供metadata。



图2 FECON原理示意图

另一个挑战是：如何保证metadata在数据帧经过DMA到达主存的过程中不会丢失？根据corundum的设计，数据帧在经过DMA处理通过PCIe到达主存前会在FPGA的片上RAM进行一次缓存。为了保证metadata与数据帧的同步，metadata也应当在此时进行缓存。比较直接的方法是在RAM模块缓存数据帧时采用一块较小RAM同时用来存储metadata，其写入/读出的使能信号等与数据帧同步，并且采用axis\_tuser信号量进行传输，从而采用一组AXI便可进行数据帧与metadata的同步传输。

**更正**：根据FAST标准，metadata在进入FAST Pipeline之前其256位应当被插入到pkt之前，并不需要单独的通路用于传输metadata。所以，我们在md\_attach模块中将tuser移作他用，并将metadata直接插入到tdata中进行传输。

在FAST的设计中，metadata位于报文的头部，随报文一起进行传输。而在corundum中，若我们使用tuser信号用于传输metadata，则可能导致metadata无法被主存接收。此时我们将采用在数据帧前增加一拍的方式将tuser中的metadata与数据帧合并进行传输（同时需要修改descriptor中数据帧长度的寄存器值）。

**更正**：这里需要检查descriptor中是否携带报文长度信息，若携带，则需要将其修改为原报文长度另外增加32字节。这里比较奇怪的是目前发现descriptor信息并非通过MAC核生成，所以需要找到desc的位置并且在该位置修改descriptor中报文的长度信息。

## **功能**

* 1. **FAST编程模型**

FECON将能够支持用户基于corundum使用FAST编程模型进行NIC开发。除少数区别外（如数据位宽从128bit变为256bit），用户能够使用FAST所提供的基于UM的硬件编程模型以及基于UA的软件编程接口。

* 1. **FAST流水线**

用户将能够使用FAST2.0标准的FAST五级流水线进行定制的数据帧的处理。（具体详见FAST2.0白皮书）

## **FECON总体设计**

FECON基于Corundum代码库进行重构，分别在软硬件层面进行FAST模块的“插入”，用于在基于corundum的NIC内提供基于FAST的编程模型与接口。本章从总体设计的角度阐述FECON的软硬件设计。在3.1节与3.2节我们分别阐释FECON硬件部分与软件部分的设计框架。在3.3节，我们详细讨论在开发过程中面临的挑战以及可能的解决方案。

* 1. **FECON硬件架构**

Corundum在硬件部分提供了interface-->port的框架。每个interface在OS中对应单独的以太网接口（如eth0等），而单interface包含一个或多个port模块。在corundum中，针对数据帧内容的操作（checksum的计算，hash值的计算等）均在Port模块中进行，而数据帧所对应的描述符（descriptor）的队列管理则在interface模块中进行。

因此，为了在插入FAST模块的同时不破坏corundum目前的逻辑结构，FAST的模块将被插入Port模块内部：当数据帧以AXI Stream数据的形式从MAC核进入Port模块时，FECON将接管数据帧，为数据帧增加metadata并在DMA模块处理前交由插入FAST模块进行处理。其逻辑流程如图3所示，其中蓝色部分为需要插入的主要模块。



图3 增加FAST模块的数据帧处理流程图

其中metadata attach模块用于：1) 在RX路径上为接收自MAC核的数据帧增加metadata；2) 在TX路径上为来自FAST流水线的数据帧剥离metadata。由于数据帧以AXI Stream形式进行传输，根据AXI Stream Spec定义，tuser信号可用于跟随所传输数据进行用户定制的数据传输，因此metadata借助tuser与数据帧同步进行传输。tuser位宽定义为256b，每个数据帧携带1拍共256b的metadata，其格式定义参见“FAST 2.0白皮书”20页。注意metadata将于数据帧的前两拍（前64B）同步传输，并且与数据帧共用AXIS的valid信号。

FAST流水线模块将实现FAST架构中UM的主要功能，其默认主要包含五级流水线，完成：通用分组解析，通用关键字提取，通用查表，通用动作，通用输出引擎五项功能。由于corundum中使用了位宽为256b的AXIS数据通路，因此FAST的数据通路位宽也将由128b（134b）调整为256b。另外，与最初FAST针对的网络转发平台不同，corundum作为开源NIC在内部定义了TX/RX两条数据通路，因此FAST流水线将需要两条流水线分别处理TX与RX的数据。

在FAST模型中，各模块使用134b中的[133:132]标记数据帧头（01），帧尾（10）以及帧体（11），并且使用[131:128]标记本拍数据中的有效字节数。为了在corundum中支持相似功能，我们将使用AXIS中的tkeep信号标记本拍数据中的有效字节数（目前corundum已支持），并利用tuser用于标记帧头，帧尾以及帧体。

需要注意的是，由于目前未知corundum现使用的位宽为1b的tuser信号是否在DMA传输和队列处理时有其它作用，我们在经由FAST流水线处理完毕后还应当恢复tuser的格式，同时将tuser中256位的metadata信号插入tdata（需要在数据帧的头部增加一拍）。

根据上述设计，MD\_ATTACH模块的连接关系如图4所示：



图4 MD\_ATTACH模块连接关系图

MD\_ATTACH需要完成的工作包括：

1. 在RX通路中，在每个数据帧前填充一拍的metadata；
2. 在TX通路中，从数据帧头中将首拍的metadata剔除，保证MAC core的正确运行；
3. 在RX通路中，对tuser修改为2b，支持帧头、帧尾标记。

而对于FAST\_Pipeline模块，其连接关系如图5所示：



图5 FAST\_Pipeline模块连接关系图

FAST\_Pipeline需要完成的工作包括：

1. 在TX与RX通路分别实现FAST架构UM五级流水的基本功能；
2. 将FAST架构中128b数据通路调整为256b；
3. 在模块内部独立查表模块。
   1. **FECON软件架构**

略

## **FECON详细设计**

本章将对FECON的软硬件接口设计、基本数据结构及各部分功能实现进行详细设计。

* 1. **FECON metadata（AXIS tuser）格式定义**

根据上述所述，FECON利用AXIS tuser用于传输metadata。对于每个数据帧，共包含2拍tuser信号共计传输256b，其首拍(128b)格式定义如表一所示：

表1 metadata格式定义

|  |  |  |  |
| --- | --- | --- | --- |
| [127] | 1 | pktsrc | 分组来源，0为网络，1为CPU |
| [126] | 1 | pkt | 分组目的，0为网络，1为CPU |
| [125:124] | 2 | flag | 01帧头；11帧体；10帧尾 |
| [123:120] | 4 | inport | 分组输入端口号 |
| [119:118] | 2 | outtype | 00:单播；01:组播；10泛洪；11：从输入端口输出 |
| [117:112] | 6 | outport | 单播:分组输出端口ID， 组播/泛洪：组播或泛洪地址索引表 |
| [111:109] | 3 | priority | 分组优先级 |
| [108] | 1 | discard | 丢弃位 |
| [107:96] | 12 | len | 分组长度 |
| [95:88] | 8 | smid | 最近一次处理分组的模块ID |
| [87:80] | 8 | dmid | 下一个处理分组的模块ID |
| [79:72] | 8 | pst | 标准协议类型，详见 |
| [71:64] | 8 | seq | 分组接收序列号 |
| [63:50] | 14 | flow\_id | 流ID |
| [49:32] | 18 | reserve | 保留 |
| [31:0] | 32 | ts | 时间戳 |

根据“FAST 2.0白皮书”，PST字段的当前定义如下图所示：



* 1. **FECON硬件模块设计**
     1. **MD\_ATTACH模块设计**
        1. **整体框架**
        2. **模块接口**
        3. **状态机设计**
        4. **其它**
     2. **FAST\_Pipeline模块设计**
        1. **整体框架**

FAST\_Pipeline模块为FAST UM五级流水线在corundum中的映射，位于md\_attach模块与dma\_client\_axis\_sink & dma\_client\_axis\_source模块间。FAST\_Pipeline模块的接口以及前后模块的连接关系如图6所示。FAST的控制通路由AXI Light协议管理，具体设计见4.2.3节。



图6 FAST\_Pipeline模块连接关系图

* + - 1. **模块接口**

详细来看，FAST\_Pipeline模块顶层接口定义如表2所示，主要由两组（TX与RX）AXI Stream接口构成。

表2 FAST\_Pipeline模块输入输出信号定义表

|  |  |  |  |
| --- | --- | --- | --- |
| **信号名称** | **方向** | **宽度** | **信号描述** |
| clk | input | 1 | 时钟信号 |
| rst\_n | input | 1 | 复位信号 |
| tx\_axis\_tdata\_int | input | 256 | 写数据 |
| tx\_axis\_tkeep\_int | input | 32 | TX写数据有效字节数 |
| tx\_axis\_tvalid\_int | input | 1 | TX写有效信号 |
| tx\_axis\_tready\_int | output | 1 | TX写反压信号 |
| tx\_axis\_tuser\_int | input | 1 | TX写tuser信号 |
| rx\_axis\_tdata\_int | output | 256 | RX读数据 |
| rx\_axis\_tkeep\_int | output | 32 | RX读数据有效字节数 |
| rx\_axis\_tvalid\_int | output | 1 | RX读有效信号 |
| rx\_axis\_tready\_int | input | 1 | RX读反压信号 |
| rx\_axis\_tuser\_int | output | 2 | RX读帧头/帧尾信号 |
| tx\_axis\_tdata\_fast | input | 256 | TX读数据 |
| tx\_axis\_tkeep\_fast | input | 32 | TX读数据有效字节数 |
| tx\_axis\_tvalid\_fast | input | 1 | TX读有效信号 |
| tx\_axis\_tready\_fast | output | 1 | TX读反压信号 |
| tx\_axis\_tuser\_fast | output | 2 | TX读帧头/帧尾信号 |
| rx\_axis\_tdata\_fast | output | 256 | RX写数据 |
| rx\_axis\_tkeep\_fast | output | 32 | RX写数据有效字节数 |
| rx\_axis\_tvalid\_fast | output | 1 | RX写有效信号 |
| rx\_axis\_tready\_fast | input | 1 | RX写反压信号 |
| rx\_axis\_tuser\_fast | output | 1 | RX写tuser信号 |
|  |  |  |  |

* + - 1. **模块设计**

由于corundum将数据通路分为了TX与RX两条通路，而FAST本身五级流水线适用的为单流水线，因此在FAST\_Pipeline中将使用两条独立的流水线FAST\_TX与FAST\_RX分别处理数据帧的发送和接收。此部分主要实现FAST流水线的迁移，因此不存在流水线的设计。

以FAST\_Pipeline模块为顶层的模块结构图如图7所示。其中，DS\_wrapper与MD\_wrapper分别为负责将corundum目前使用的AXI接口转为FAST标准接口。而Pipeline\_TX与Pipeline\_RX则为FAST UM的核心逻辑。



图7 FAST\_Pipeline模块结构图（详细）

* + - 1. **其它**

**注1**：图7未包含控制通路与查表模块设计，但是具体考虑如下：

1. FAST控制通路与AXIL在corundum中功能相同，为兼容性考虑，FAST\_Pipeline将使用AXIL协议进行内部模块用户自定义寄存器的读写。
2. 查表模块在FAST中为独立的IP模块，因此表项的查找与corundum设计无关，而表项的读写以及修改则需要a) 软件层面调用UA接口；b) 硬件层面使用AXIL协议进行BRAM的读写操作。相比较其它部分逻辑较为复杂，将在后续设计进行支持。
   * 1. **其它模块设计**
   1. **FECON软件详细设计**

略

## 参考资料

1. Corundum: <http://cseweb.ucsd.edu/~snoeren/papers/corundum-fccm20.pdf>
2. AXI4-Stream: <https://static.docs.arm.com/ihi0051/a/IHI0051A_amba4_axi4_stream_v1_0_protocol_spec.pdf>
3. FAST-ANT: <https://github.com/fast-codesign/FAST-ANT>