# 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。

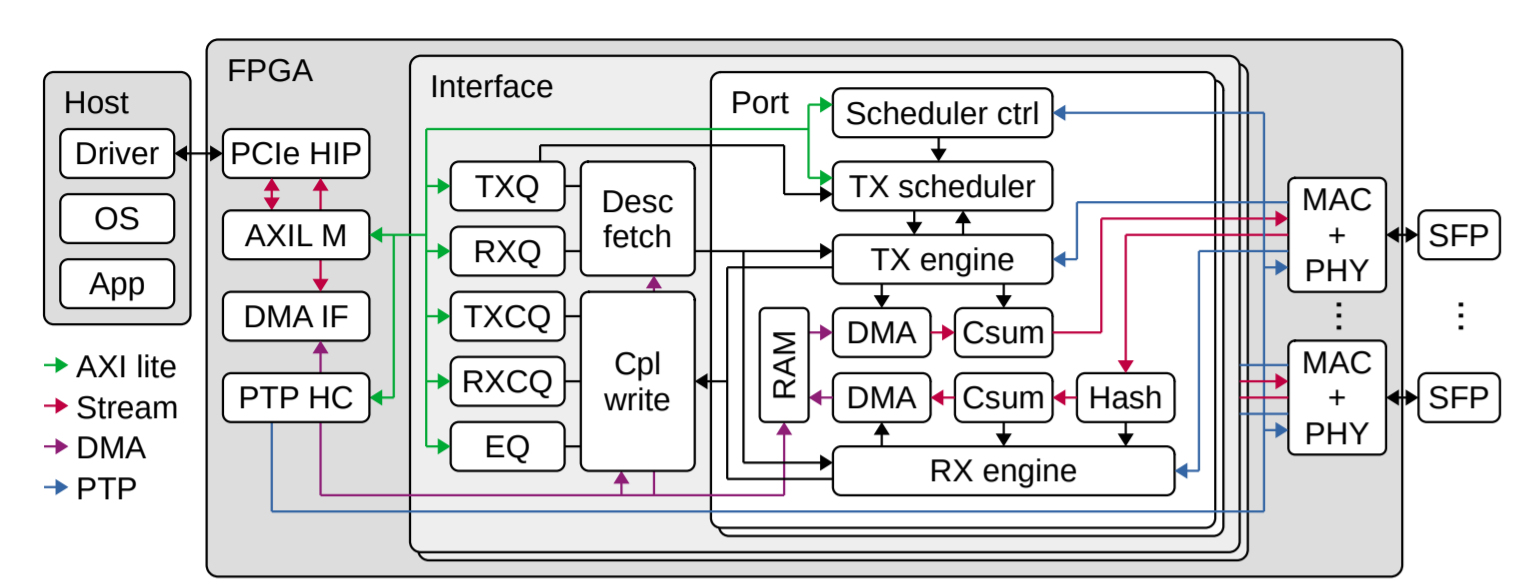


图1 Corundum网卡架构图

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。

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）。

## **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128b，每个数据帧携带两拍共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），并将tuser信号量从128b扩展为130b，将[129:128]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，将1b的tuser以FAST 2.0的metadata格式扩展为130b（高2位用于标记帧头帧尾等）；
2. 在TX通路中，将130b的tuser恢复为1b，从而保证MAC core的正确运行；
3. 在RX通路中，对共计2拍，256b的metadata各字段以FAST所定义格式进行填充。

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**
  2. **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**
        2. **模块接口**
        3. **状态机设计**
        4. **其它**
     3. **其它模块设计**
  3. **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>