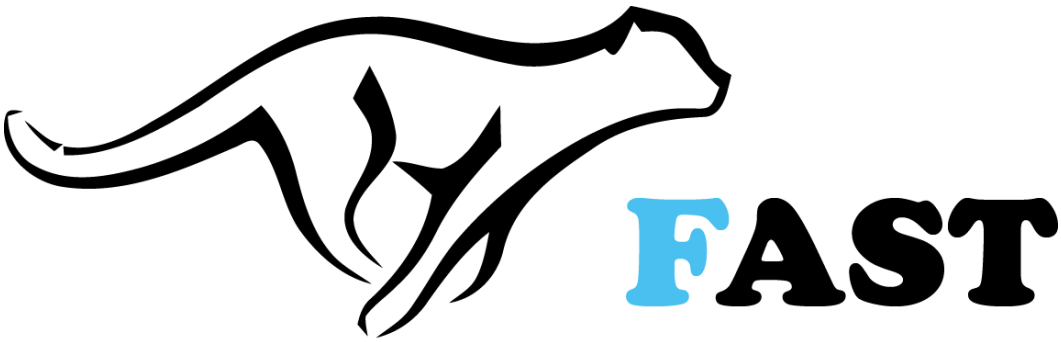
FAST 开源平台原理与应用

主题	FAST 开源平台原理与应用
文档号	
创建时间	2018-08-15
最后修改	2018-08-15
版本号	2.0
文件名	FAST 开源平台原理与应用.pdf
文件格式	Microsoft word

地址：湖南长沙岳麓区中电软件园 6 栋 302 室



FPGA Accelated Switching Platform

FAST 开源平台原理与应用

目录

一、 FAST 概述

1.1 FAST 产生的背景

1.1.1 网络设备平台化趋势与开源技术

随着软件定义网络（SDN）和网络功能虚拟化（NFV）等技术的出现和快速发展，使得网络设备的实现架构逐渐呈现出平台化的发展趋势。由于开源具有鼓励设计重用，避免“重复发明轮子”，摆脱用户对单一开发商的依赖，降低开发成本和缩短开发周期等优点，开源方法也迅速成为推动网络平台发展，提升其技术成熟度的重要手段。

目前网络平台相关的开源项目主要体现在管理控制平面，包括 SDN 控制器软件（ONOS，ODL），NFV 管理软件（OPNFV）等。这些开源项目主要由工业界主导，发展迅速，对 SDN，NFV 的快速发展和落地应用起到了重要的作用。

而 SDN、NFV、5G 等技术的发展，急需网络数据平面的创新。当前网络设备数据平面相关的开源项目相对较少，典型的是 OpenVSwitch（OVS）。OVS 完全由软件实现数据平面 SDN 交换，不但性能较低，难以在千兆以上的网络环境中应用，而且对基于硬件的分组交换技术研究参考价值不大。斯坦福大学在十多年前发起了 NetFPGA 项目，基于 Xilinx FPGA 实现了路由器，交换机和 openflow 交换机等功能，但近年来发展十分缓慢，特别在技术支持和服务方面难以满足科研人员的需求。

1.1.2 对数据平面开源项目的需求

新型交换技术的研究对网络数据平面开源项目具有强烈需求。例如有状态的 SDN 数据平面处理，基于 SDN 交换的安全防护，以及段路由与 SDN 的结合都是当前路由换技术研究的热点，都需要对现有的 SDN 交换机数据平面进行扩展。如果没有相关开源平台进行支撑，研究人员很难从零实现一个全新的，支持上述创新功能的数据平面。下一代网络处理器芯片和交换芯片体系结构研究对数据平面开源项目也有强烈的需求。

下一代网络处理器和交换芯片是 SDN 和 5G 网络设备研制的核心，也是当前学术界最前沿的领域之一。由于缺少支撑研究的开源平台，芯片架构和关

键技术的创新只能依赖网络芯片和网络设备制造商进行，大量科研机构中研究人员的创新思想难以得到验证和应用。

因此，当前需要一个针对网络数据平面的开源项目，该项目的主要特点包括：

（1）基于多核 CPU 和 FPGA 的开源平台

能够同时兼顾分组交换处理的可编程性与性能，以及实现平台易于获得是数据平面开源项目的首要要求。近年来，FPGA 发展迅速，不但可以与 CPU 通过 PCIe 接口高速通信，其自身内部的逻辑容量，内嵌 RAM 容量，以及 DDR 和网络高速接口 IP 都可以满足 40Gbps 以上网络处理的需求。特别是 NetFPGA 的成功经验表明，FPGA 平台要比网络处理器平台更加易于获得，更加易于编程，而且 FPGA 编程的 Verilog 实现代码跨平台移植方便，易于满足不同用户的需求。

（2）实现基本的数据平面交换功能

数据平面的开源项目需要一个可扩展的框架，但不能仅仅停留在框架上。必须有先行者基于开源项目框架首先实现一些基本的网络交换功能，例如 L2 交换，L3 转发和 openflow 交换等。这样可以吸引更多用户关注和使用开源项目代码搭建自己的网络环境。这些用户在使用中的反馈是推进开源项目走向成熟的关键。

（3）支持用户基于开源平台实现自己的创新工作

用户可以在开源平台基础上实现自己的创新工作，而不用重新设计一个完整的数据平面。例如用户可以在硬件中增加定制模块来支持新的协议处理，或是在硬件逻辑中中设置一个“钩子”，将指定流量定向到 CPU，通过编写软件代码实现对特定流量施加特定处理。

（4）用户得到更好的技术支持

由于参与新型交换技术的研究人员在工程实现方面的经验差距很大。例如电子工程专业的可能擅长硬件设计，而计算机专业的可能擅长软件编程；以创新研究和撰写高水平论文为目标的研究人员更加注重原理的快速验证，而一些企业希望利用开源项目能够快速实现产品原型。不论目的如何，用户都希望能够在开源社区获得更加专业，甚至是定制的服务。

此外，开源项目不应涉及太多的商业利益纷争，开源实现的代码不能绑定具体的 FPGA 型号和 CPU 类型。因此开源项目在软硬件设计时需要显式划定平台相关实现与平台无关实现的界限，定义清晰的平台相关实现与平台无关实现间的接口规范。从而支持开源项目适合更多的平台，给用户更多地选择。

1.1.3FAST 项目起源——iRouter

2016 年，在国家 863 项目“新型动态网络关键技术及验证”的研究过程中，国防科技大学等单位基于通用 CPU 加 FPGA 平台研制了可编程网络交换平台（iRouter），iRouter 基于 FPGA 实现了万兆接口上分组的解析，关键字提取，查表转发和输出调度等操作，通过 CPU 与 FPGA 协同实现数据平面功能的动态扩展，包括对 LISP（位置标签分离协议）转发的支持，对 IPv6 路径 MTU 发现的支持以及对 DDOS 攻击检测的支持等。该平台在北京，南京和长沙的多个科研单位部署应用，这些平台通过 CNGI 网络连接，基于 LISP 协议构建了跨广域的 IPv6 新型编址与路由实验网。实验网在支撑 IPv6 新型编址与路由的规模验证方面发挥了重要作用。

iRouter 基于多核 CPU 与 FPGA 的协同处理实现了可扩展的分组转发功能，具有以下特点：

（1）在 FPGA 中实现了包含分组解析，查表关键字生成，匹配查表和输出调度等模块的分组处理流水线，该流水线可以线速处理万兆接口到达的分组。根据预先配置的规则，分组可以直接交换到输出接口发出，也可以定向到 CPU 进行进一步处理；

（2）CPU 上的软件分为数据平面功能扩展软件和控制平面软件。硬件流水线可以根据规则的动作（Action）确定将分组送特定的数据平面扩展软件还是控制平面软件；

（3）CPU 上的数据平面扩展软件或控制平面软件可以将发出的分组通过 DMA 再次送到硬件流水线进行转发；

（4）用户可以通过对硬件流水线的配置，将特定分组送给 CPU 用户空间的指定进程中，而该进程对分组处理后，还能再次将分组送回硬件流水线。因此用户可以在用户空间编写一个进程，实现特定的网络功能，并通过将该功能嵌入到分组处理流水线中，实现数据平面功能的扩展。

此外，FPGA 内部的交换流水线也可以进行扩展。例如，我们在动作执行模块后转接了一个流计数器模块，根据配置，对特定数据流双向到达的分组数目和字节数目进行统计，通过检测流量的不对称性来检测 DDOS 攻击的发生。

因此，iRouter 除了实现基本的 openflow 交换功能外，还提供了一种数据平面功能扩展的机制，支持用户通过对 FPGA 中的交换流水线扩展，或者增加新的数据平面处理进程来扩充交换平台的分组处理功能。

为了进一步完善 iRouter 的设计并支持更多用户使用，我们公开 iRouter

实现架构的细节，FPGA 中交换流水线的 Verilog 实现代码，以及 iRouter 内核程序和 iRouter 库的代码，形成了 FAST 开源项目。

1.2 FAST 的目标与意义

1.2.1 FAST 的目标

FAST 项目一方面基于 iRouter 平台采用的多核 CPU 加 FPGA 的实现架构，能够充分发挥多核 CPU 和 FPGA 两者结合带来的灵活性和高性能的优点。另一方面，FAST 还需进一步清晰的描述其软硬件协同的开放架构，规范其软硬件开发接口，激励第三方研究人员开发和贡献更多的案例，以及吸引更多的用户。

具体的说，FAST 需要在 iRouter 代码基础上，进行三方面的工作。

一是明确 FAST 软硬件协同分组处理的模型，支持在标准的硬件流水线中嵌入新的硬件处理模块，或将用户空间的用户应用相关的软件进程插入流水线，扩充流水线的功能；

二是建立一套完整的 FPGA 流水线设计规范，用户根据规范进行各种自定义模块的设计，易于插入并集成到现有的流水线中，支持最大程度实现模块的重用性，模块内部的逻辑实现不依赖其他外部模块的需求；

三是建立一套用户应用进程的编程 API，提供并不断扩充 FAST 编程库，支持用户按照 socket 编程的方式，在用户空间实现转发平面的处理功能。

1.2.2 FAST 项目的意义

FAST 开源项目对推动当前新型路由交换技术研究具有重要意义，主要表现在以下三方面。

（1）提供新型交换设备的解决方案

基于通用多核 CPU 和 FPGA，FAST 项目已经实现了基本的路由交换功能，并且软硬件代码完全开源。因此用户可以直接基于 FAST 构建自己的交换设备。

FAST 目前提供的解决方案包括：

- L2 交换机
- L3 路由器
- SDN 交换（openflow1.3）
- 精准测量服务

- LISP XTR 路由器

（2）支撑新型交换技术的创新研究与实验

FAST 采用可扩展的软硬件紧耦合分组处理模型，支持用户在平台已有实现基础上，通过对 FPGA 硬件流水线的扩充，或基于 FAST

库编写用户空间程序实现对网络设备数据平面的扩充，从而实现对各种新型的路由交换机制进行实验研究。

- FAST 架构支持的新型交换技术创新研究包括：
- NFV 硬件加速研究
- 智能网卡研究
- 下一代网络处理器研究
- 下一代交换芯片研究
- SDN 交换设备研究

FAST 社区将为新型路由交换技术的研究群体提供交流的平台。2016 年 10 月，第一次 FAST 开源项目工作会议在长沙举行，来自工业界，学术界和教育界的 20 个单位的 33 名研究人员成为 FAST 项目的成员。经过 1 年多的发展，目前已经有超过 30 个单位或研究组使用 FAST 进行科研和教学，主要涉及的方向包括：

- 下一代交换技术研究
- 可重构网络设备实现技术研究
- 低年级本科生路由交换技术实验教学
- 高年级本科生计算机网络综合实验
- 研究生的网络空间安全创新实验
- 新型网络体系结构研究和原理验证

1.3 FAST 平台组成

1.3.1 总体架构

FAST 定义了平台相关逻辑/代码和平台无关逻辑/代码之间的接口规范。其中 FPGA 中的硬件流水线的实现与具体的平台无关，又称为用户模块（UM）。用户可对 UM 进行扩充，在硬件流水线中定制自己的功能，也可利用标准的 FAST UM，通过编写软件程序扩展分组处理功能。用户基于 FAST API 编写的用户空间程序称为用户应用（UA）。

FAST 平台的组成如图 1-1 所示，主要由平台相关软硬件，用户相关软硬

件以及 FAST 支撑软件三部分组成。

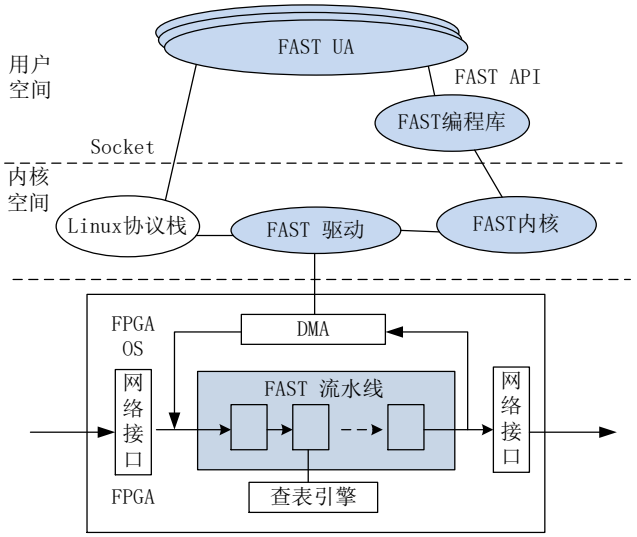


图 1-1 FAST 平台的总体组成

1.3.2 FPGA OS

FAST 平台的 FPGA 中，除了平台无关的 UM 外，还有平台相关的逻辑，称为 FPGA OS。FPGA OS 主要有两个功能，一是屏蔽平台的异构性，使得 UM 的 verilog 代码具有更好的跨平台移植能力；二是为 UM 的实现提供共性的服务。FPGA OS 屏蔽不同 FPGA 平台的异构性主要包括五方面。

一是不同的 FPGA 型号，如来自不同厂商，具有不同的开发工具，FPGA 具有不同的容量和速度等级等；

二是不同的网络接口，如千兆以太网接口和万兆以太网接口，以及不同的接口数目等；

三是不同的外部接口类型，如有的连接 TCAM 协处理器，有的连接 DDR 存储器，有的连接 SRAM 存储器等；

四是不同的接口逻辑实现方式，例如有的以太网 MAC 逻辑在 FPGA 内嵌的 IP 核实现，有的 MAC 逻辑由 FPGA 外部的 PHY/MAC 一体的芯片实现。不同的接口逻辑实现方式获取接口状态，如 up/down，的方式也不同；

五是 FPGA 与 CPU 不同的通信方式，如果 FPGA 与 CPU 在同一个板卡上或机箱内，FPGA 可能通过 PCIe 总线与 CPU 进行通信，否则， FPGA 可能通过以太网与物理位置相分离的 CPU 进行通信。

FPGA OS 为简化 UM 设计提供各种通用服务，如分组的接收和发送，外部存储器的访问，与 Cpu 通信的高速 DMA，以及分组处理中需要的规则匹配等。

1.3.3 FAST 支撑软件

FAST 支撑软件包含 Linux 用户空间实现的 FAST 库，在 Linux 内核中实现的 FAST 内核路径控制，以及内核中的 UM 仿真软件三部分。

FAST 库又分为 FAST 标准库和 FAST 定制库。FAST 标准库对应标准的 FAST UM 实现，为 UA 提供分组收发，规则配置，平台信息获取（如接口计数器）等功能；FAST 定制库实现对 UM 中用户扩充模块的管理和访问。FAST 库通过 Netlink 机制与内核中的 FAST 路径控制软件通信。

FAST 内核路径控制软件主要在内核中实现多个 UA 与 UM 通信的 MUX/DMU 功能。内核 UM 仿真软件主要实现对 FPGA 中 UM 功能的仿真，在不包含 FPGA 的标准终端和服务器上支持 UA 的运行。

1.3.4 用户相关软硬件

用户相关软硬件是用户根据自己实验需求开发的，与平台无关的软件 UA 和硬件 UM。用户功能可根据开发时间和性能要求等限制条件在 UA 和 UM 中划分。通常功能的主体在 UA 中实现，UM 实现对特定功能，如规则匹配，的加速。

FAST 编程库为 UA 开发提供调用平台资源的接口。用户的 UA 设计也完全不用考虑底层 FPGA 的实现方式，只是调用 FAST API 提供的函数接收分组，发送分组，对 FPGA 中抽象出来的 UM 进行编程控制等。除了 FAST API，UA 也可调用 Linux 平台提供的标准系统调用，如 socket，进行编程。

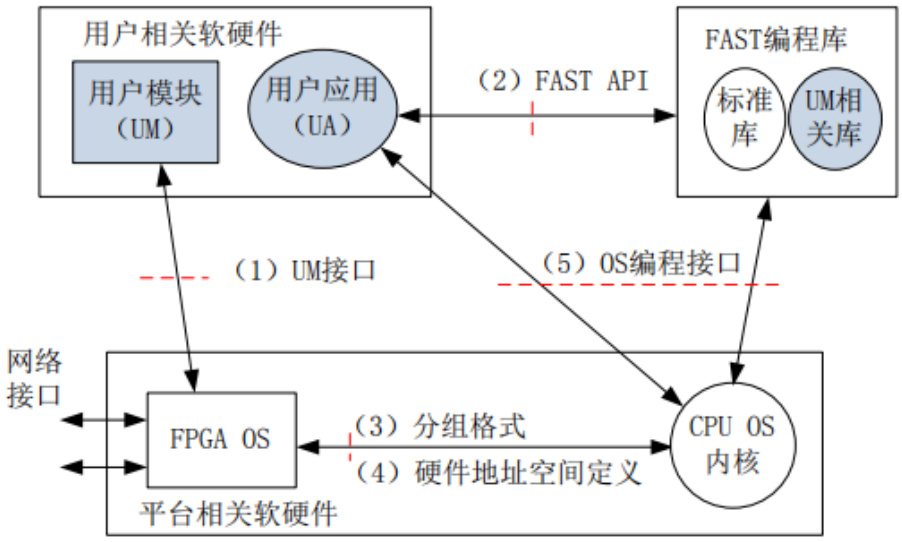


图 1-2 FAST 用户相关软硬件开发接口

1.4 FAST 规范

用户基于 FAST 平台进行实验开发需要遵循相关的规范，主要包括 UM 接口规范，FAST API，FAST 分组穿越软硬件界面时的格式，以及 FAST 平台硬件空间地址的定义等。

1.4.1 UM 接口规范

UM 接口规范定义了 UM 逻辑与 FPGA OS 的接口信号和工作时序，通常 UM 与 FPGA OS 的接口包括接收和发送分组的接口，接收 UA 管理的接口，调用 FPGA OS 提供其他各类服务的接口等。

1.4.2 FAST API 规范

FAST API 规范定义了 UA 访问底层资源使用的关键数据结构和相关函数的定义等。FAST 平台为 UA 开发提供了分组收发，规则管理等标准的 API，实现这些标准 API 的库称为标准库。如果用户在 UM 中需要增加一些特殊的处理功能，如深度内容检测（DPI），对这些特殊资源的管理使用就需要开发相应的 API 实现这些 API 的库称为 UM 相关库。

1.4.4 硬件地址空间划分

硬件地址空间规范定义了 FPGA OS 和 UM 硬件中软件可访问资源，如寄存器，计数器和存储器等，在独立的硬件地址空间中的定义。唯一的硬件地址空间定义是 UA 和 FAST 库跨硬件平台可移植的前提。

二、 FAST 分组处理模型

FAST 面向通用 CPU 加 FPGA 的异构处理平台实现，采用了软硬件协同的处理模型。对处理性能要求较高的通用处理功能由 FPGA 硬件流水线实现，用户特定需求相关的处理功能由用户空间的软件编程实现。

软硬件协同的分组处理是 FAST 区别其他数据平面开源项目的最重要的特点。模块（Module）是 FAST 平台中交换功能定义、资源分配以及代码重用的基本单元，特定分组处理功能到软硬件模块的映射也是 FAST 软硬件协同分组处理的关键。

2.1 功能模块

2.1.1 功能模块定义

FAST 中的模块是能够实现特定网络处理功能的，具有唯一标识的，相对独立的一个软件程序（UA）或 FAST 流水线中的硬件逻辑块。FAST 模块能够按需连接，组成上下游关系，实现分组处理的流水线。FAST 模块具有一些共有的属性：

（1）用户可见

模块的用户可见是在交换功能设计时，模块实现的功能是用户可以感知到并且关注的功能。如 FPGA 实现的分组协议解析逻辑块、输出调度逻辑块、IEEE 1588 端节点的时钟同步功能块以及操作系统内核中的 TCP/IP 协议栈等。这些功能都与用户实现的分组交换目标密切关联。FPGA OS 和操作系统内核中用户不可见且不关心的功能实现不能称为模块，例如 FPGA 中的 DMA 引擎、DDR 接口控制器以及操作系统中的 FAST 驱动程序等。

（2）独立可重用

模块间是相互独立的，在功能实现上没有明确的功能依赖或调用关系。模块之间只存在上下游关系，不存在父模块和子模块关系。不同的模块可能来自不同的开发人员，这些模块在不同的设计中可以重用，用户通过不同模块的组合实现不同的分组交换功能。因此硬件模块的设计必须遵循 FAST 流水线规范中对模块接口信号语法语义的要求，FAST 软件 UA 模块设计必须基于 FAST API，不能对其上游或下游模块的实现方法提出任何假设。

（3）唯一标识

FAST 平台中每个模块都用被称为 MID 的 8 比特 ID 唯一标识，因此平台中最多支持 256 个模块。FAST 规定 0-127 为 FAST 流水线中硬件模块的 MID 标识，128-255 为 FAST 软件模块的标识。硬件模块在实例化时通过外部连线获取 MID，软件 UA 在初始化注册时获取自己的 MID。在软硬件协同分组处理中，对分组目的模块号（DMID）的设置是实现分组处理路经控制的重要手段。

2.1.2 功能模块

FAST 开源项目为用户提供了经过测试可直接使用的通用软硬件模块。其中 FAST 流水线中的通用硬件模块包括：

- 通用分组解析模块（GPP），实现以太网、ARP、IPv4、IPv6、ICMP/ICMPv6，TCP 和 UDP 等协议的解析功能；
- 通用关键字提取模块（GKE），实现 IPv4/IPv6 分组的多元组（五元组+输入端口号等）信息的提取和查表关键字的生成；
- 通用查表模块（GME），实现 TCAM 查表功能，将 GKE 提取的五元组信息映射为 14 位的 FlowID。
- 通用动作模块（GAC），根据 FlowID 查找动作表，获取分组的目的输出接口或 UA 的 MID（如果需要将分组定向到软件处理）。
- 通用输出引擎（GOE），实现基于令牌桶的输出分组 Meter，丢弃分组计数等功能；

通用的软件模块包括：

OpenFlow 通道 UA（OXFP），在 FAST 硬件流水线与源端 SDN 控制器之间建立 OpenFlow 1.3 协议通道的连接，实现 packet-in/packet-out 分组交换以及 FlowMOD 等消息的解析执行操作。

不同用户根据自身需求，已开发的 FAST 流水线模块还包括 IEEE 1588 PTP 协议解析模块、DDOS 前端检测模块、TCP 代理模块、Lisp 协议封装/解封装模块、假冒源 IPv6 地址检测模块、测量分组定时发送模块及传输协议跳变模块等。实现的软件 UA 包括 IPv6 路经 MTU 发现模块、DDOS 检测控制模块、状态防火墙模块及 LISP 映射管理代理模块等。

上述用户自定义模块与 FAST 提供的基本软硬件模块组合，即可在标准 OpenFlow 交换基础上，扩充更多的用户定制功能，实现有状态的数据平面处理。

2.2 FAST 软硬件协同处理模型

2.2.1 模型的基本原理

FAST 平台软硬件协同的分组处理模型如图 2-1 所示，包含 n 个由 FPGA 实现的硬件处理模块，以及 m 个运行在 CPU 上的软件处理模块。其中 FPGA 硬件中的 n 个处理模块组成 FAST 硬件流水线，到达 FPGA 硬件的分组依次由硬件流水线的第 1 级（模块 X1），第 2 级（模块 X2）…第 n 级（模块 Xn）进行处理。其中 P 代表完整的分组，MD 代表元数据，PHV 代表分组头向量。

每个分组在硬件流水线中传递时，会携带元数据（MD）和分组头向量

（PHV）。元数据包含分组处理的中间结果，用于上下游模块之间交换分组处理的控制信息；分组特征向量主要描述分组自身的特性，由分组头部 128 字节以及分组的协议类型

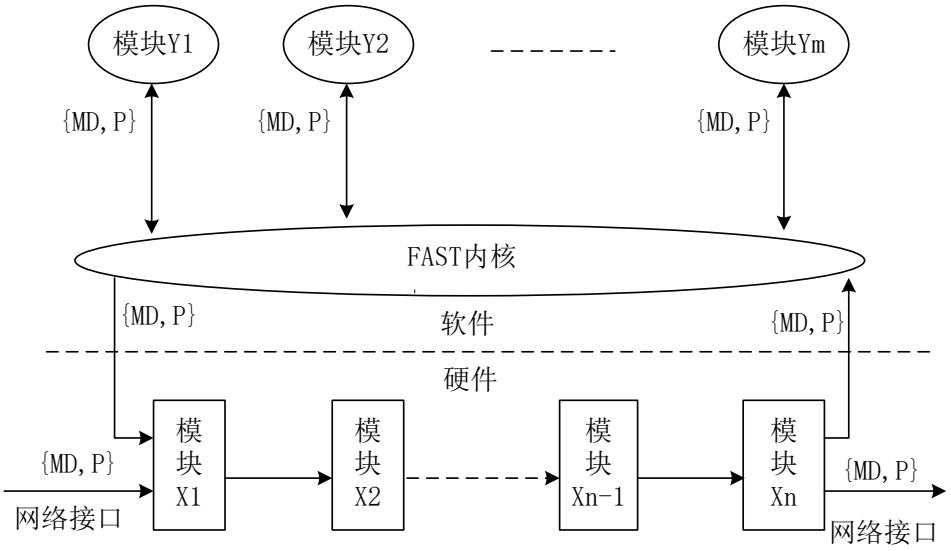


图 2-1 FAST 的软硬件协同处理模型

FAST 流水线的第一个模块 X1 将完整的分组送缓冲区缓存，同时生成分组的 PFV，后续模块根据 PFV 对分组进行处理，如提取查表关键字等；FAST 流水线的倒数第二级 Xn-1 模块将完成的分组读出，根据需要修改分组内容，如更换目的 MAC 地址等，然后送最后一级模块 Xn 进行输出调度。

软硬件接头实现 FAST 硬件流水线与各软件模块 Y1…Ym 之间的分组及其元数据的数据交换，主要由 FPGA OS 和 CPU OS 内核中相应的软硬件实现。

每个软件模块一般为用户空间独立运行的进程，又称为用户应用（UA），UA 实现用户定义的处理功能。FAST 流水线与 UA 之间交换的信息包括完整的分组以及分组的元数据信息。

在 FAST 软硬件协同处理模型中，软硬件模块基于目的模块 ID（DMID）进行通信，如图 2-2 所示。其中硬件流水线中的模块 M1，M2.。。Mn 位置相对固定，软件模块 Mx 和 My 可以根据需要动态加载。

网络接口接收的分组首先进入硬件流水线处理，硬件流水线中最后一个模块 Mn 处理完成后，根据处理结果，通过设置分组元数据中的 DMID 值，指示 FPGA OS 将分组直接从网络接口输出，或者送软件模块 Mx（DMID=x）或 My（DMID=y）进行处理。

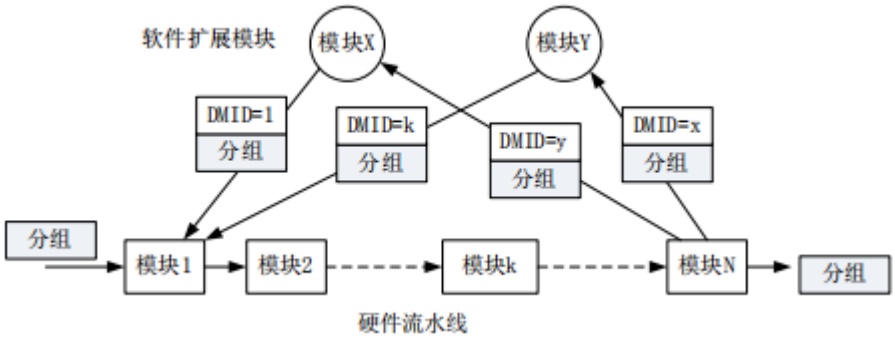


图 2-2 FAST 软硬件模块的通信

每个软件模块在处理完成分组后，将分组发送回硬件流水线，分组首先送到流水线入口模块，然后再依次经过每个硬件模块进行处理。根据模块实现模型，每个模块首先比较分组的 DMID 是否为本地 MID，若不等于，该模块不会处理分组，会将分组直接送流水线的下游模块进行处理。

例如图 2-2 中，软件模块 Y 将发送给硬件流水线分组的元数据中将 DMID 标记为 k，1<k<n，则硬件流水线中模块 k 之前的其他模块不会对分组进行任何处理，直接将分组送到下游模块，直到模块 K。典型的，软件模块 X 为内核协议栈，而软件模块 Y 为数据平面的功能扩展模块。硬件流水线如果需要将分组送控制平面处理，如目的 IP 地址为本地 IP 的分组，则将分组的 DMID 设置为 X，该分组会送协议栈和控制平面处理；如果需要将分组送数据平面的扩展功能模块 Y 处理，则将分组的 DMID 设为 Y。同样的，如果控制平面需要发送一个分组而无需硬件流水线处理，则将该分组的 DMID 设置为 n。该分组进入硬件流水线后，会旁路掉所有的硬件处理模块，最后经模块 n 发给 FPGA OS 输出。

2.2.2 分组处理流水线的实现

由于网络接收的不同分组可能具有不同的属性，属于不同的流水线等价类，因此数据平面需要支持不同的分组处理流水线。在 FAST 架构中，根据模块的处理模型以及对 DMID 的设置，可以精确控制分组处理的模块，以此多个流水线可以方便的映射到 FAST 分组处理架构中。

2.2.3 分组处理流水线的动态扩展

分组处理流水线是由多个功能独立的功能模块组成，分组处理路径也可根据用户需求进行改变，不但用户可根据需求任意的增加或删除软件功能模块，还可以通过离线加载的方式重新配置硬件功能模块，满足了系统设计者对于网

络功能可扩展的需求。根据模块定义可知，每个模块中都有 NMI 表项，基于 NMI 表项的流水线可通过增删改 NMI 表项内容扩展和替换原有流水线。FAST 中每个模块具有自己的 MID，称为本地 MID（LMID），分组元数据 M 中携带接收分组的模块 ID，即 DMID。按照 FAST 模块的定义，每个模块接收到分组后，首先判断 M 中的 DMID 是否等于 LMID，若等于则表示该模块需要处理该分组，处理完成后需查找模块内部 NMI 表项，通过规则匹配定向到下行分组处理模块中；若不等于则表示该模块不需要处理该分组，直接查找模块内部 NMI 表项，通过规则匹配定向到下行分组处理模块中。

基于 NMI 表的流水线扩展技术正是利用了 FAST 模块中 NMI 表项可灵活配置的特点。用户若需要增加新功能，只需按照 FAST 模块定义生成模块，并选定需要插入流水线的位置，修改与新增模块相连的上行模块的 NMI 表项规则即可实现对新功能的增加。用户若需要替换和删除功能，首先定位需要删除或替换的功能模块，然后修改其连接的上行模块的 NMI 表项，即可实现对功能模块的删除和替换。

FAST 流水线是由多个功能独立的功能模块组成，分组处理的路径也可根据用户需求进行改变，用户不但可以根据需求任意的增加或删除软件功能模块，还可以通过离线加载的方式重新配置硬件功能模块，从而满足系统设计者对于网络功能可扩展的需求。

FAST 流水线的分组处理主要通过每个模块内部的 NMI 表项进行控制信息的传输，并通过 Metadata 进行数据信息的传输。Metadata 和表项相结合，可表示各模块之间数据和控制转移的关系，以及各模块之间的依赖关系。流水线动态增加的模块有两种方式获取自己的模块号。一种是向平台的流水线管理逻辑注册，由管理逻辑动态产生并分配一个全局唯一的模块号给新增模块，如图 2-3 所示。还有一种是每个模块根据实现的功能，离线分配一个固有的模块号。一般对于集成多方模块的平台采用前一种分配方法，对集成单一开发者的平台可采用第二种相对简单的静态分配方法。

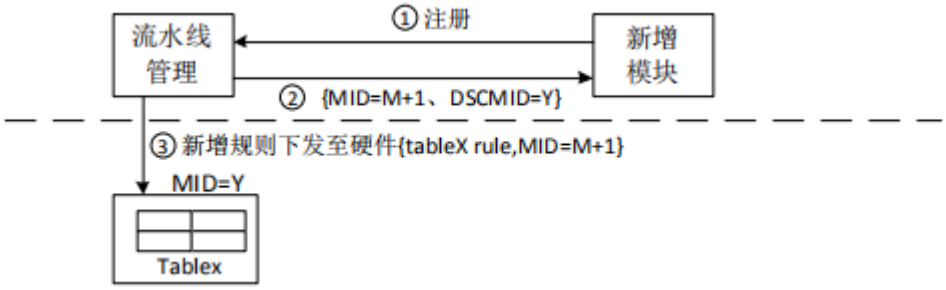


图 2-3 扩展功能模块硬件注册流程

基于 NMI 的流水线扩展原理如图 2-4 所示，流水线 P（i）=<IE、x、

y、z、OE>新增模块 s 后变为=<IE、x、y、z、s、OE>。若 R 代表规则全集，即所有的分组均满足规则 R，每个输入的分组都使用相同的处理流水 IE-x-y-z-OE。需要注意的是，虽然有的分组会单播，有的组播，有些会丢弃，还有的会送到控制平面，但每个分组经过的流水线模块是一致的，因此他们属于相同的流水线等价类。

若交换机需要扩充新的处理功能，例如支持隧道的发起和终结功能（支持 LISP 和 VxLAN 等隧道的封装和解封装）。由于原有的硬件流水线不支持隧道的封头和去头功能，在隧道起点由于封装隧道头长度超过 MTU 而带来的分片问题，以及隧道端点的重组问题，因此需要增加相应的软件模块进行相应的处理。处理流水线之间传递的分组 P 的处理行为由二元组<Rule，DMID>控制，Rule=R1 表示分组 P 的属性满足规则 R1，DMID 为流水线中下一个处理分组的模块号。

新增模块 s 在流水线的处理流程为：1、分组进入 LMIDz 模块；2、LMIDz 对分组信息进行查表，通过匹配 NMI 表项规则将分组转发至 LMIDs 中；3、LMIDs 模块对分组进行处理；4、LMIDs 将处理完成的分组通过查 NMI 表转发至后续分组处理模块（LMIDx）；5、LMIDx 模块对分组进行处理。

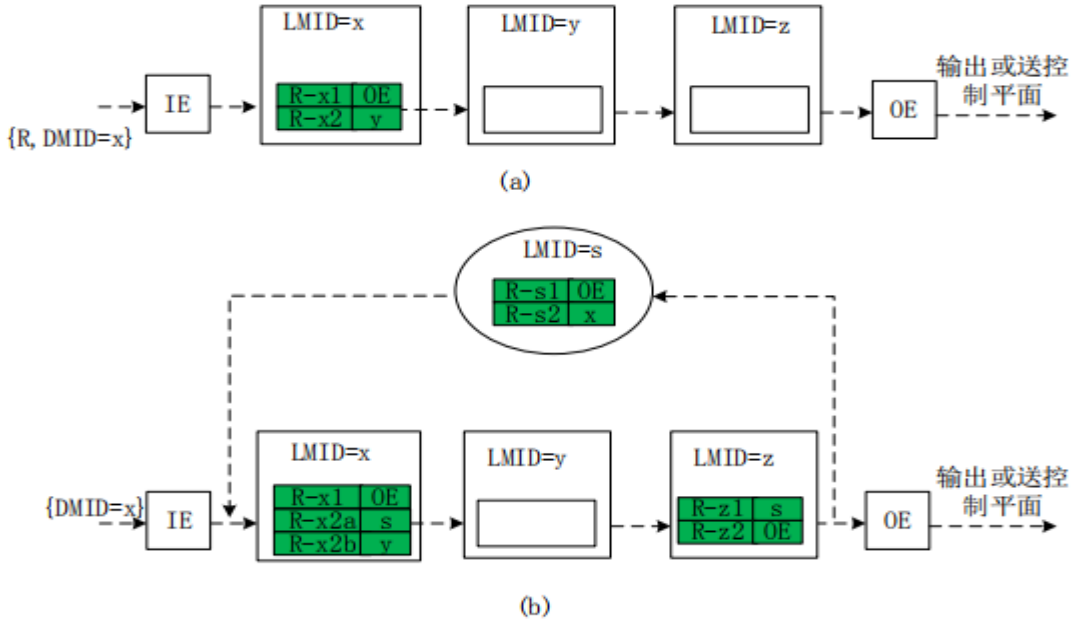


图 2-4 基于 NMI 的 FAST 流水线扩展原理

FAST 通过流水线管理模块对硬件模块的转发规则进行配置，硬件对分组进行查表匹配后根据匹配的规则进行转发。软硬件模块之间交换使用 metadata 数据结构，该结构定义了分组传输的目的 MID 以及源 MID，根据这个信息，分组便可以在任意的模块之间进行数据传递。

三、 元数据

模块是 FAST 平台实现分组处理的基本单元，软硬件模块间高效的信息交互对实现软硬件协同分组处理具有重要意义。元数据（Metadata）是 FAST 平台中模块间信息交换的核心数据结构，是控制分组在软硬件模块间的处理路径以及信息交换的关键。

3.1 元数据的定义

元数据（Metadata）是随分组一起在 FAST 平台软硬件协同的分组处理流水线中传递的，包含分组当前处理状态的数据结构。元数据在 FAST 报文的前 32 字节携带，每个分组进出 UM 的第 1 拍 16 字节为 Metadata0，第二拍数据为 Metadata1，其数据格式如图 3-1 所示。

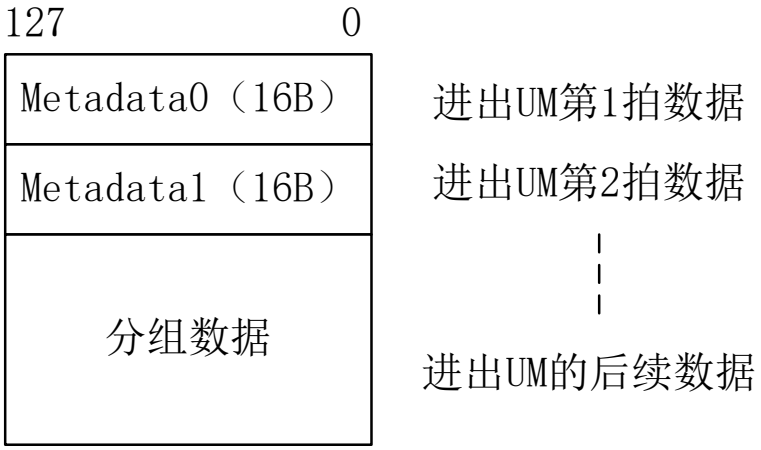


图 3-1 32 字节元数据的位置

3.2 元数据中包含的内容

[127]	1	pktsrc	分组的来源，0 为网络接口输入，1 为 CPU 输入
[126]	1	pktdst	分组目的，0 为网络接口输出，1 为送 CPU
[125:120]	6	inport	分组的输入端口号

[119:118]	2	outtype	00:单播；01：组播；10：泛洪；11：从输入接口输出
[117:112]	6	outport	单播：分组输出端口 ID，组播/泛洪：组播/泛洪表地址索引
[111:109]	3	priority	分组优先级
[108]	1	discard	丢弃位
[107:96]	12	len	包含 MetaData 字段的分组长度
[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，如图 3-2 所示
[71:64]	8	seq	分组接收序列号
[63:50]	14	flowid	流 ID
[49:32]	18	reserve	保留
[31:0]	32	ts	时间戳

Metadata1:为用户预留的 16B 的自定义空间，用户可以根据自己需求，自定义内容及使用。

分组类型	封装	PST 编码
IPv4 TCP	Eth/IPv4/TCP	0000 0001
IPv4 UDP	Eth/IPv4/UDP	0000 0010
ARP	Eth/ARP	0000 0011
ICMP	Eth/IP/ICMP	0000 0100
IPv6 TCP	Eth/IPv6/TCP	1000 0001
IPv6 UDP	Eth/IPv6/UDP	1000 0010
ICMPv6	Eth/IPv6/ICMPv6	1000 0011
未识别		0000 0000

图 3-2 PST 编码

3.3 元数据在软硬件协同的作用

FAST 平台中传输的每个分组都携带一个元数据块，用于存放分组的接收信息（如端口号，接收时刻等）、路径控制信息（如下一个模块号 DMID）、处理的中间状态（如流分类的标识 FlowID）以及用户自定义状态信息等。

FPGA OS 必须为每个从端口接收的分组创建并初始化一个元数据块，每个软件 UA 也必须为其产生的分组构建并初始化一个元数据块。当硬件模块把分组 DMID 设置为某个 UA 的 MID，或者 UA 把分组的 DMID 设置为某个硬件模块的 MID 时，分组需要携带元数据穿越软硬件界面，如图 3-3 所示。假设硬件流水线由 X、Y、Z 三个模块组成，作为流水线最后一级的模块 Z 通常为输出引擎，FPGA OS 将 DMID 为 Z 的分组送输出接口发出。

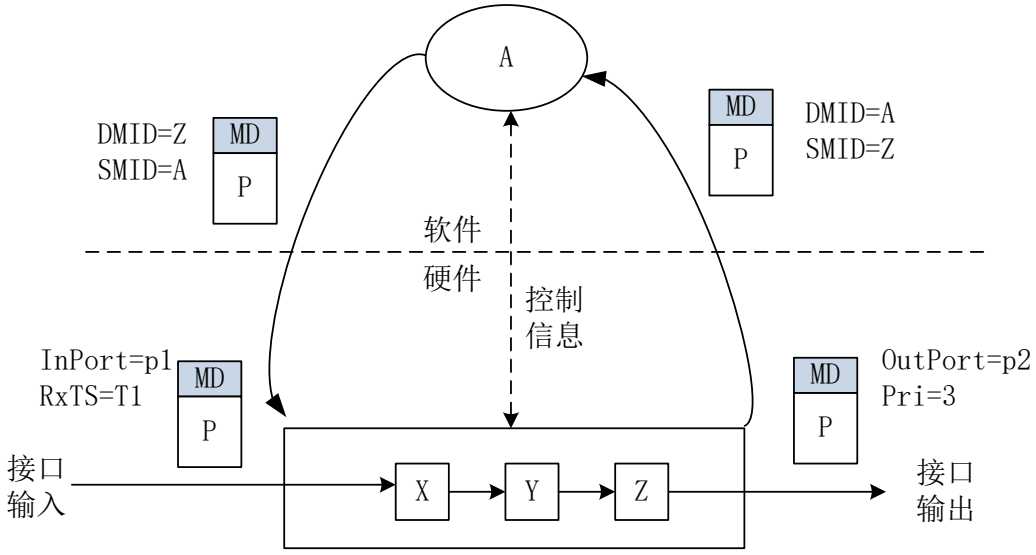


图 3-3 分组携带元数据穿越软硬件界面

以网络接收分组的处理流程为例，当接口 p1 接收到分组 p 时，FPGA OS 会在 p 得头部之前添加一个长度为 32 字节的元数据块 MD。FAST 流水线模块 X、Y 和 Z 会根据 MD 和 p 的内容进行转发决策。例如当模块 Y 决策将分组转发到端口 P2 输出，优先级为 3 时， Y 会将分组 MD 中的 DMID 设置为 Z（网络接口输出），outport 设置为 p2，pri 设置为 3，模块 Z 将根据上述设置将 p 送 FPGA OS，经接口 p2 优先级位 3 的输出队列发出。

如果平台加载了一个软件程序 A，功能是对 p1 接口输入，p2 接口输出的所有分组进行安全检查，那么 A 可向模块 Z 配置规则，将所有输入接口为 p1，输出接口为 p2 的分组的 DMID 设置为 A。因此模块 Z 会将分组 p 的 DMID 设置为 A，FPGA OS 会将 p 送到软件模块 A 进行处理，同时将 p 的 SMID 设置为 Z，表示这是硬件模块 Z 发出的分组。

当 A 完成对 p 的安全检查需要继续从 p2 发出分组 p 时，可将 p 的 DMID 设置为 Z，将 p 再次发送回硬件流水线。虽然 p 首先经过了硬件流水线中的模块 X 和 Y，但由于 DMID 为 Z，所以 X 和 Y 不会处理分组 p，而是将其送到 Z 处理。Z 发现 p 的 SMID 为 A，表示已经过安全检查，因此会将分组 DMID 修改为 Z，由 FPGA OS 从端口发出。

以上例子表明，DMID 和 SMID 在软硬件协同处理分组转发路径的控制中发挥了关键作用。硬件模块可通过设置 DMID 将分组（经 FPGA OS 的 DMA，FAST 驱动，FAST 内核，FAST 库）送指定的 UA 处理，UA 也可以通过设置 DMID 将分组按照相反的顺序送到指定的硬件模块进行处理。

3.4 协议标准头（PSH）

协议标准头包含分组前 128 字节，如图 3-4 所示，

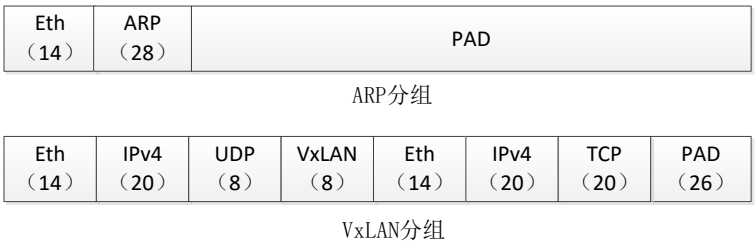


图 3-4 PSH 示例图

3.5 分组处理流程

FAST 流水线的处理流程如图 3-5 所示，其中 PKT 代表分组内容，M 代表 Metadata，PSH 为分组标准头。每个模块在 UM 实例化时带有本地模块 ID（LMID）和下一模块 ID（NMID）两个参数。报文输入时，UM 各模块将 LMID 与 M 中 DMID 值进行对比，若相等，则处理此分组，并将 M 中的 DMID 修改为 NMID 值；若不相等，则直接转发输出。

当用户插入自定义模块（UDM）时，用户需修改其与上下级模块的数据接口信号，并修改 UDM 与其上级模块对应的 LMID 及 NMID 即可。

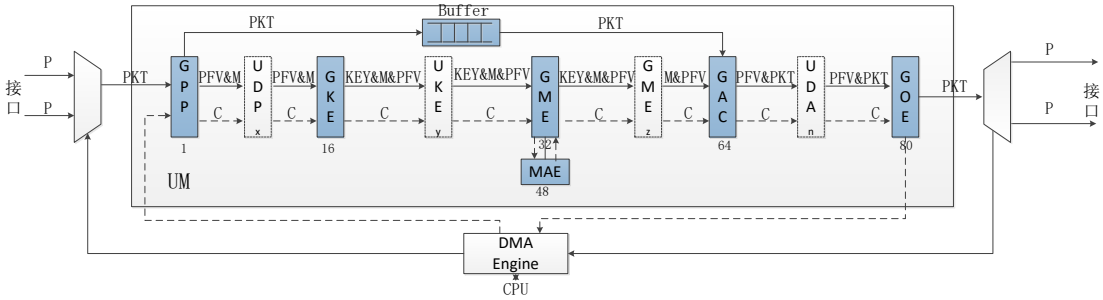


图 3-5 分组处理流程图

四、 FAST 硬件流水线

在 FPGA 中实现的硬件流水线是 FAST 平台实现分组硬件处理的基础。FPGA OS 为 FAST 流水线提供了基本的分组收发，高速 DMA 和查表等服务。FAST 流水线与 FPGA OS 的接口定义直接反映了分组硬件处理功能在 FPGA OS 和 FAST 流水线中的划分，也是 FAST 平台提供商必须考虑的关键问题。

4.1 FAST 与 OS 的接口

FAST 定义了 UM 规范。UM 规范是 FAST 流水线和 FPGA OS 的接口。

4.1.1 FAST 流水线接口定义

FAST 流水线与 FPGA OS 之间定义了 7 个接口，分别是 Pktin/Pktout、Cin/Cout、Ctrl、MEI 和 AUX。如图 4-1 所示。

Pktin/Pktout 是流水线接收和发送分组的接口，Cin/Cout 分别是流水线接收和发送流水线控制信息的接口，Ctrl 是 CPU 根据虚拟地址空间的定义，通过该接口提供的读写操作原语，对 FAST UM 内部的寄存器、各类控制表进行维护管理的接口，MEI（Match Engine Interface）是流水线访问 FPGA OS 提供的匹配引擎的接口。AUX 是 FPGA OS 为 FAST 流水线提供时钟、复位和时间戳等信息的接口。

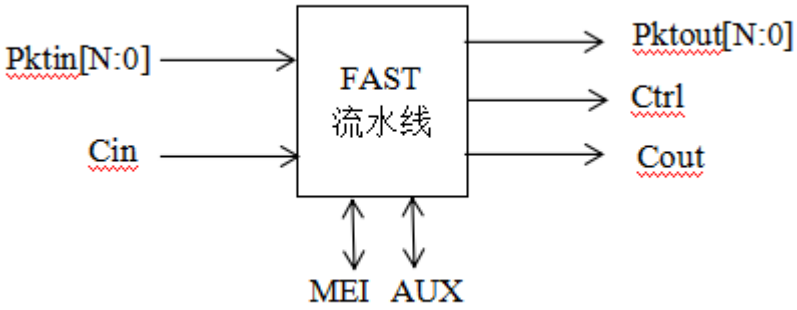


图 4-1 FAST 与 OS 的接口

各接口信号的详细定义如表 4-1 所示。其中 MEI 接口中的信号宽度 N1/N2 与平台相关。Clk 时钟频率与平台相关，一般不小于 125MHz。所有 I/O 信号的变化都与 CLK 同步。

表 4-1 各接口的详细定义

接口类型	信号名称	方向	信号描述
Pktin 接口	pktin_data_valid	In	接收分组数据有效
	pktin_data[133:0]	In	接收的分组数据
	pktin_ready	Out	可以接收新的分组指示
Pktout 接口	pktout_data_valid	Out	发送分组数据有效
	pktout_data[133:0]	Out	发送的分组数据
	pktout_ready	In	可以发送分组指示
Cin 接口	cin_data_valid	In	写入控制消息有效
	cin_data[127:0]	In	写入的控制消息
	cin_ready	Out	可接收控制消息
Cout 接口	cout_data_valid	Out	写出控制消息有效
	cout_data[127:0]	Out	写出的控制消息
	cout_ready	In	可接收控制消息
MEI 接口	key_valid	out	查表关键字有效
	key[N1-1,0]	out	输出的查表关键字
	Me_ready	In	匹配引擎准备好
	Flowid_valid	In	匹配返回的结果有效
	Flowid[N2-1,0]	In	匹配返回的结果
	Match_flag	In	是否匹配标识
AUX 接口	Clk	In	时钟信号
	reset_n	in	低有效的复位信号
	TimeStamp[31:0]	in	本地时间戳
Ctrl 接口	ctrl_valid	in	总线有效
	ctrl_cmd	in	总线读写
	ctrl_datain[31:0]	in	总线输入数据

	ctrl_dataout[31:0]	out	总线输出数据
	ctrl_addr[31:0]	in	总线地址

FAST 流水线通过 Key 接口向查表协处理器提交查表关键字，从 FlowID 接口接收返回的查表结果（匹配的地址）。包括是否匹配（Match_flag 为 1 表示匹配，0 表示未匹配），以及匹配的规则序号 FlowID 等。

只要 Me_ready 信号有效，FAST 流水线可连续地向查表协处理器提交查表请求，查表协处理器必须保证这些查表结果按照查表请求提交的顺序返回查表结果。

查表协处理器中规则的配置管理方法与具体的平台相关。FAST 库在实现时必须对不同的平台进行适配，提供管理这些规则的 API 接口。

4.1.2 数据通路 Pktin/Pktout 数据格式

Pktin 和 Pktout 两个接口采用相同的分组格式，数据宽度为 134 位，其中 128 位为报文数据，包括 32 字节的 Meatdata 和分组数据，最高 6 位为控制信息。如图 4-2 所示。

报文数据的格式为接口收发的以太网报文格式。其中以太网报文格式中不包含最后的 4 字节 CRC 字段，接收时，FPGA OS 负责接收时进行 CRC 校验和剥离，发送时，FPGA OS 会计算分组的 CRC 字段并附加在报文最后。

数据通路的[133:132]位为报文数据的头尾标识。01 标识报文头部，11 标识报文中间数据，10 标识报文尾部。由于不同报文具有不同长度，因此在报文数据最后一排可能存在一些无效的字节。数据报文的最后一拍的[131:128]位用来标识无用字节的个数。其中 0000 表示 16 个字节全部有效；0001 标识最低 1 个字节无效，最高 15 个字节有效；以此类推，1111 表示最低 15 个字节无效，最高 1 个字节有效。

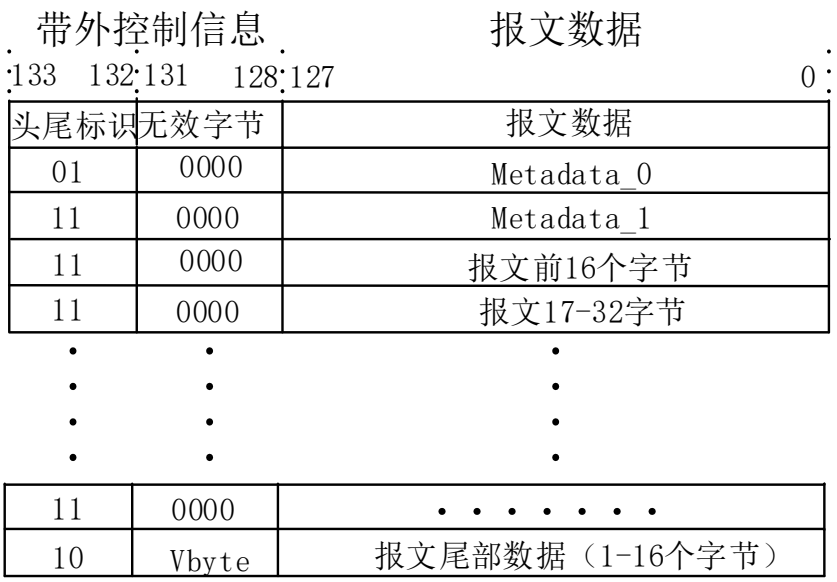


图 4-2 数据通路 Pktin/Pktout 数据格式

设 FAST 流水线的时钟频率为 xMhz，因此数据通路理论传输带宽为 128bit*xMHz。例如当 x=125 时，通路带宽为 16Gbps。

4.1.3 控制通路 Cin/Cout 数据格式

FAST 流水线的控制通路 Cin 和 Cout 采用相同的消息格式。对 FAST 流水线模块的每次读写操作都转换成一个 128 位宽的命令字写入 FAST 流水线。命令字的格式如表 4-2 所示。与流水线处理分组信息一样，每个模块只处理 DMID 等于本地模块号的命令字。相应的模块完成读写操作后，修改命令字，并将命令字发送给下游模块，最后命令字从 Cout 接口输出。FPGA OS 负责将软件读写请求转换成 FAST 流水线的命令字，并将 Cout 输出的命令字返回给由 SMID 标识的软件模块。

表 4-2 命令字的格式

位置	字段	含义
[127]	Path	0:数据；1:控制
[126:124]	TYPE	001：读帧；010 写帧；011 读响应帧；100：硬件主动触发帧
[123:112]	Seq	访问序号
[111:104]	SMID	发出 C 消息的模块 ID
[103:96]	DMID	接收 C 消息的模块 ID

[95:64]	Addr	操作地址
[63:32]	Mask	写数据掩码
[31:0]	Data	写的数据/读返回数据

4.2 流水线的组成

如图 4-2 所示，FAST 基本的流水线由 5 个通用功能模块组成，可支持基本的 openflow 转发功能。通过对基本流水线扩展，可以实现更加复杂的交换功能。

4.2.1 基本硬件流水线结构

FAST 基本流水线包含通用分组解析（GPP）、通用关键字提取（GKE）、通用匹配引擎（GME）、通用转发动作（GAC）和通用输出引擎（GOE）五个模块，以及一个分组缓冲区（buffer），如图 4-3 所示。由于 Buffer 对软件是不可见的，与具体的分组处理功能无关，因此不是 FAST 的硬件模块，也没有 MID 编号。

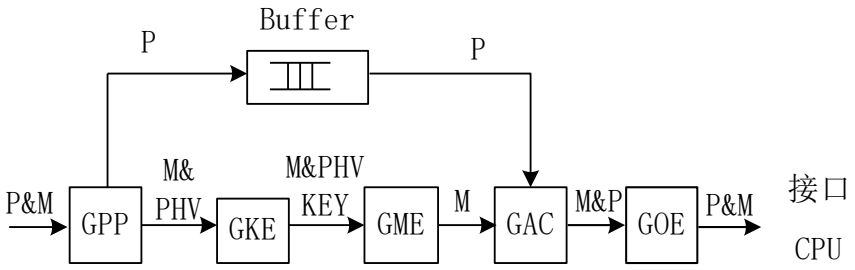


图 4-3 流水线的组成

GPP 将到达的分组按照到达的先后顺序将分组写入 buffer 缓存，同时提取分组的前 128 字节作为分组头向量（PHV）随分组元数据（M）在模块间传递，模块 GAC 以此将分组从 buffer 读出，重新与元数据组合，发往 GOE 模块。在 GPP 将分组缓存，只在流水线中传递 PHV 的优点是简化流水线的设计，避免变长报文对流水线处理性能的影响。

M 和 PHV 在 verilog 中分别定义为 256 位和 1024 位的向量，KEY 定义为 296 位（IPv6 五元组）的向量，因此上游模块使用 1 个时钟周期即可将 M、PHV 以及 KEY 信息传递到下游模块。

4.2.2 流水线的处理流程

（1）GPP 模块

GPP 模块解析到达分组的 L2-L4 层协议，将解析结果写入分组元数据中 8 比特的 PST 字段。PST 编码中 0XXX XXXX 编码对应 IPv4 相关协议，1XXX XXXX 编码对应 IPv6 相关协议。0000 0000 表示未识别的协议。

目前 GPP 支持的 PST 编码类型如表 4-3 所示：

表 4-3PST 编码类型

PST 编码	协议类型（封装格式）
0000 0000	未识别
0000 0001	Eth/IPv4/TCP
0000 0010	Eth/IPv4/UDP
0000 0011	Eth/ARP
0000 0100	ETH/IPv4/ICMP
0000 0100-0111 1111	保留
1000 0001	Eth/IPv6/TCP
1000 0010	Eth/IPv6/UDP
1000 0011	Eth/IPv6/ICMPv6
1101 0000-1111 1111	保留

GPP 模块同时负责把分组送报文缓冲区按照先进先出的方式缓存，同时生成每个分组的 PHV，与分组元数据一起向流水线下游传送。

（2）GKE 模块

GKE 负责根据元数据中的 PST 值，从 PHV 中提取查表关键字。当分组 PST 确定时，关键字在 PHV 中具有确定的位置。例如，对于提取 IPv4/TCP/UDP 报文的五元组，可离线计算得：IPv4 源 IP 地址到以太网帧起始的偏移量为 26（208bit）字节，目的 IP 偏移量为 30 字节（240bit）。协议域偏移为 23 字节（184bit），TCP/UDP 源和目的端口号分别为 34（272bit）和 36（288bit）字节。显然，如果在关键字提取时需要 TCP 的 SYN 等标志位，可以计算这些标志位的偏移量，直接赋值即可。用 Verilog 描述的 IPv4 五元组关键字提取代码如图 4-4 所示。

```
        If （ PST=8’h00000001 || PST=8’h00000010 ） //IPv4 TCP/UDP
        Begin
```

```
        Src_IP<=PHV[239:208]；//26*8=208
        Des_IP<= PHV[271:240]；//30*8=240
        Pro_type<=PHV[191:184];//23*8=184
        Src_port<=PHV[287:272];//34*8=272
        Des_Port<=PHV[303:288];//36*8=288
End
```

图 4-4Verilog 描述的 IPv4 五元组关键字提取代码

GKE 支持对 IPv4/IPv6 的 TCP/UDP/ICMP 五元组提取（ICMP 没有端口号），其中 IPv4 和 IPv6 的关键字具有不同的格式。

（3）GME 模块

GME 模块实现类似 TCAM 的功能，将包含五元组的 key 与 TCAM 中的带掩码的五元组规则进行匹配，返回匹配的 FlowID，如果匹配不命中，FlowID 为全 3F。GME 将返回的 flowID 信息填写到元数据的 FlowID 字段中。

不同的 FPGA 平台上，GME 的匹配有不同的实现方法，以及不同的规则数目和规则宽度等。实现方式也可能是使用 FPGA 片外的 TCAM 芯片，或者 FPGA 片内的 TCAM 逻辑。

由于 GME 输入的 KEY 可能有多种格式，因此 GME 的 KEY 与元数据中协议类型（PST）字段合并组成查表关键字，软件在配置查表规则时，不同格式的规则前面要带上不同的 PST 编码。

（4）GAC 模块

GAC 模块包含 Action 表，通常表项的大小与 FlowID 的宽度有关。例如系统支持 4K 条五元组标识的流，那么 FlowID 的宽度为 12，在 GAC 中的 Action 表也有 4K 项。每个 Action 表包含对分组的转发操作，包括丢失，转发到特定输出端口，或送到特定的软件 UA 处理等。GAC 根据转发操作相关更新分组元数据中的字段，信息同时将分组从 buffer 中读出，与元数据一起发给下游模块。

GAC 实现对分组元数据中 OutPort、discard、DMID 等域的修改，决定分组的转发交换行为。

（5）GOE 模块

GOE 模块负责 FAST 流水线输出分组的处理，主要包含以下 2 个功能。一是根据配置对 FlowID/DMID 标识流的令牌桶限速，例如作为 Openflow 交换机实现时，GOE 可以控制 Packet-in 分组（DMID 为 openflow 通道控制器）的流量，二是对丢弃分组的计数。由于分组在 Buffer 中是顺序存储的，因此即使

GAC 之前的模块不能随意丢弃分组或者分组元数据。GPP 等模块如果决策要丢弃分组时，需将分组元数据中的 discard 位置 1，将 DMID 设置为 GOE 的 MID，这样分组就会旁路掉 GOE 模块前其他模块的处理，直到 GOE 模块。GOE 模块实现对分组的丢弃，并进行统计计数。

4.3 硬件流水线的扩展及其示例

针对不同应用场景在不同时间内对交换功能的需求不一致的特点，FAST 流水线支持利用 FPGA 现场可编程特性，动态增加和卸载流水线模块以满足不断变化需求的特性。

4.3.1 硬件流水线的扩展

FAST 基本流水线包含通用分组解析（GPP）、通用关键字提取（GKE）、通用匹配引擎（GME）、通用转发动作（GAC）和通用输出引擎（GOE）五个模块。基于这五个模块搭建的基本架构，FAST 流水线支持动态模块的插入和删除，如图 4-5 所示。

在 GAC 之前的流水线称为 Ingress 流水线，主要在输入端口的上下文中对分组进行转发决策处理；由于分组元数据到达 GAC 模块后，可根据 GME 的匹配结果查表的到分组输出的接口号（或软件 UA 编号），因此在 GAC 之后分组在输出接口的上下文中处理，因此 GAC 模块之后的流水线称为 Egress 流水线。

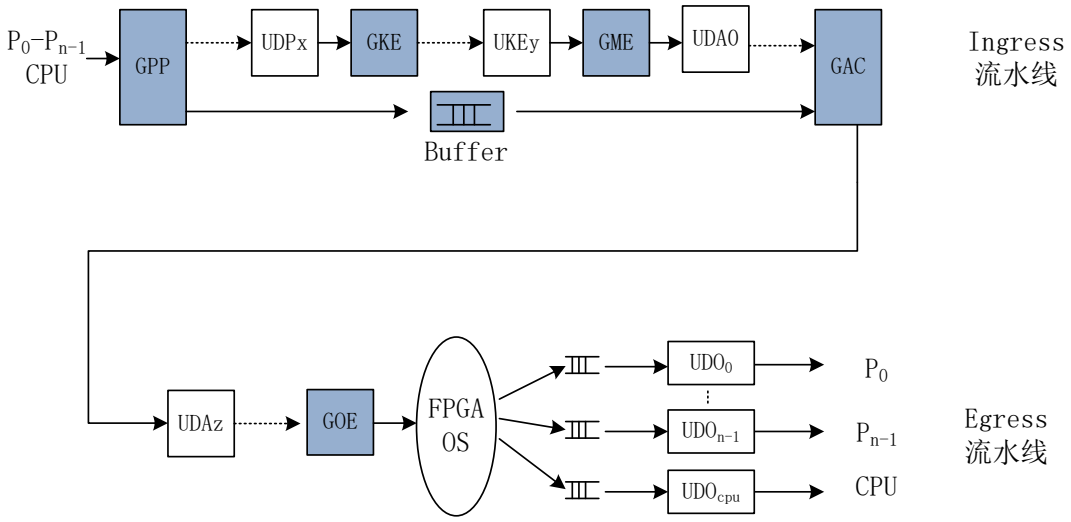


图 4-5 硬件流水线的扩展

FAST 流水线的功能扩展主要通过增加以下几类硬件模块。

（1）用户定义的解析（UDP）模块

在 GPP 模块之后增加一个或多个 UDP 模块，可实现对 GPP 不支持的其他协议进行解析，并根据解析结果修改元数据中的协议类型字段 PST。一个 UDP 模块在修改完 PST 后，将 DMID 设置为 GKE，旁路掉后续的 UDP，即每个分组最多支持一个 UDP 模块的协议分析。

（2）UKE 模块

在 GKE 模块之后增加用户定义的关键字提取模块 UKE，扩展支持 GKE 不支持的关键字提取能力，UKE 可以修改 KEY 字段。通常如果一个 UKE 模块完成 KEY 的提取和修改后，直接将 DMID 设置为 GME，旁路掉后续的 UKE（如果有的话），即每个分组最多支持一个 UKE 模块的处理。

（3）UDA 模块

UDA 模块主要实现用户定义的 action 处理。UDA 模块插在 GME 和 GAC 之间可以扩充 Ingress 流水线的处理功能，插入在 GAC 和 GOE 之间可以扩充 Egress 流水线的处理功能。位于 GAC 之前和之后的 UDA 模块有两点不同。一是 UDA 之前的只能对分组的元数据进行操作，而 GAC 之后的 UDA 可以直接修改整个分组；二是 GAC 之前的 UDA 是在分组输入的上下文处理分组，而 GAC 之后的 UDA 是在分组输出端口的上下文中处理分组。

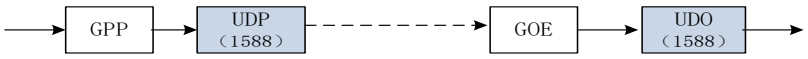
（4）UDO 模块

UDO 模块物理位置并不在 FAST 流水线之内，而是位于 FPGA OS 中分组最终从 FPGA 引脚输出之前。由于 UDO 的位置位于分组输出调度之后，因此 UDO 处理的分组在输出时不存在任何阻塞，能够保证分组离开后到最终输出到链路上有一个确定的延时。其主要作用是获取精确的分组输出时间戳，可以支持 IEEE 1588 透明时钟的计算或者网络测量中精确发送时间的获取。

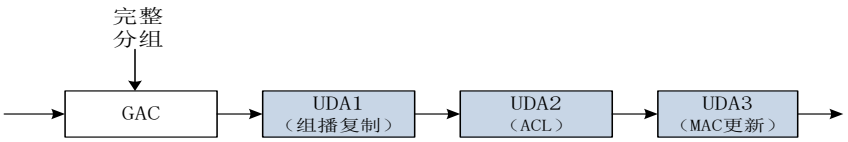
FAST 流水线扩展模块的设计必须遵循标准的模块接口定义，而且扩展插入的模块对上下游模块都是透明的，即流水线中原有的上下游模块都应该感知不到新模块的插入。

4.3.2 硬件流水线扩展的示例

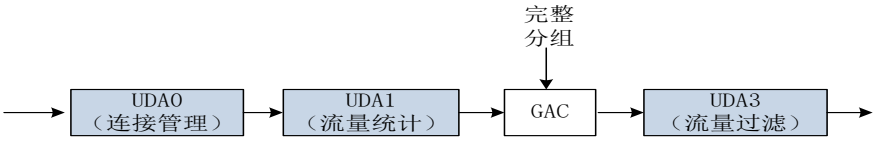
通过对 FAST 硬件流水线进行扩展，可以支持在原有的交换平台中扩充支持新的协议以及新的分组处理机制。例如下图分别介绍了通过扩展 UDP 和 UDA 模块支持 IEEE 1588，L3 转发控制以及安全网关功能的例子。



（a）对1588 PTP协议的支持



（b）路由器转发动作扩展



（c）安全网关动作扩展

在（a）中，由于 GPP 协议无法识别封装在特定 UDP 端口号中传输的 IEEE 1588 的 PTP 协议，因此 GPP 处理后 PST 字段表示的分组类型为 ETH/IP/UDP，如果交换平台需要支持 IEEE 1588 的 PTP 协议，需要插入一个 UDP 模块，对使用 319 和 320 端口的 UDP 分组进行解析。如果 UDP 目地端口号为 319，即 UDP 内部包含需要交换平台计算透明时钟的事件消息（sync，delay_req），这些消息可在 PST 中标记，UDO 会根据 PST 的标记跟新分组的透明时钟；如果目的端口号为 320，则其中包含不需要透明时钟计算的 PTP 通用消息，UDP 会进一步分析消息类型，将分组转发（follow_up 消息或 delay_resp 消息），或送特定的软件 UA 处理（BMC 协议消息）。

在（b）中，增加 UDA1 实现输出组播复制功能，根据组播分组元数据中 Outport 携带的组播组信息查表获得输出接口集合，将分组一次复制发送到每个输出接口；而 UDA1 实现绑定到特定输出接口的 ACL 过滤功能；UDA3 实现对输出分组的修改，例如修改目的 MAC 等。因此通过扩展 Egress 流水线中的 UDA 模块，可以使 FAST 流水线支持三层的单播和组播转发功能。显然，每个新增的 UDA 模块内部也包含相应的控制表格，这些表格会通过相应的软件配置。

在（c）中，可在 Ingress 流水线中增加连接管理和复杂的流量统计功能，实现与特定输入端口绑定的安全控制。利用连接管理功能，可以维护五元组标识的 TCP/UDP/ICMP 的连接状态，实现状态防火墙，并为其他基于流的 middlebox 功能提供流的状态信息；而利用复杂的流量统计功能可以检测到特定输出接口甚至是到特定服务器流量的非对称性，用于早期的发现 DDOS 攻击；而在 Egress 流水线中增加流量过滤功能，可对具有特定属性分组进行过滤，例如在“勒索病毒”爆发期间，可以针对该病毒的特点快速部署硬件过滤模块，对传播病毒的恶意分组进行过滤，而过了“勒索病毒”爆发期以后，该模块可

以删除以节约资源用于其他功能的实现。

五、 FAST 平台软件实现原理

软硬件协同是 FAST 平台分组处理的特色。FAST 平台软件主要实现 FAST 硬件流水线与 FAST 用户应用（UA）间数据通路和控制通路的信息交互，为 UA 编程提供函数库，是支撑 UA 运行的基础。FAST 平台软件主要由 Linux 内核中的 FAST 驱动和 FAST 内核，以及用户空间的 FAST 编程库组成。

5.1 FAST UA 的通信模式

FAST UA 运行时与 FPGA 以及网络接口有多种通信需求，包括 FAST 分组收发、对 FPGA 的配置管理，以及与远端主机进行标准的 socket 通信等。

（1）FAST 分组收发

UA 从 FAST 流水线接收 FAST 分组，以及向 FAST 流水线发出自己产生或者转发的 FAST 分组。当 UA 实现 middlebox 处理功能时，如防火墙功能，使用 FAST 分组收发的方式与底层 FPGA 进行通信。这些分组遵循元数据加以太网帧的 FAST 分组格式定义。FAST 分组格式定义将在 FAST 编程 API 介绍时给出。

（2）对 FPGA 的配置管理

UA 对 FPGA 的配置管理包括对 FAST 流水线的配置管理以及对 FPGA OS 的配置管理。典型的对 FAST 流水线配置管理操作包括配置 GAC 模块中的 action 表，读取 GOE 模块中的计数器等；典型的对 FPGA OS 的配置管理包括读取端口接收发送计数器，配置 FPGA OS 提供的匹配协处理器得流表项等。

（3）与远程设备的 socket 通信

FAST UA 还可使用标准 socket()机制通过内核协议栈与远程的网络设备进行通信。根据目的 IP 地址，内核协议栈确定该通信是通过挂接到 CPU 上标准以太网接口进行，还是通过挂接到 FPGA 上的网络接口进行。如果是前者，协议栈将会以 sk-buf 的格式与标准以太网驱动交换收发的分组；如果是后者，协议栈将会把分组送 FAST 内核，由 FAST 内核将分组封装成 FAST 格式，通过 FAST 驱动向 FPGA 中的流水线发出。

需要注意的是，FAST UA 可能同时使用多种通信模式。例如 openflow 通道代理（OXFP）UA 既要通过 socket 机制与远端的 SDN 控制器通信，实现

packet-in/packet-out/flowmod 等 openflow 消息的交互，又要与 FAST 流水线进行 FAST 分组的交互。由于上述通过 Linux 内核的通信都是阻塞式的，因此需要为不同的通信创建不同的线程。

5.2 FAST 平台软件的实现

FAST 平台软件实现示意如图 5-1 所示，其中包含 FAST 驱动，FAST 内核，FAST 编程库以及两个 UA，UAx 和 UAy。FAST 内核维护的 Netlink 端口映射表 NPMT（Netlink Port Mapping Table）是 FAST 平台软件中的核心数据结构，主要存放 UA 的 MID 与 NetLink 端口号之间的映射关系。

FAST 平台软件实现时在用户空间并不存在一个集中的 UA 管理程序。每个 UA 在启动时都独立地通过 FAST 库函数向 FAST 内核注册，获取 MID 以及与 NetLink 端口号的映射关系。NPMT 中也保存了各种用于管理维护的各种计数器信息，例如某个 UA 接收和发送分组的个数等。

FAST 编程库为 UA 设计提供 FAST 分组收发的 API 函数，这些函数通过 Netlink 机制与 Linux 内核中的 FAST 内核通信。Netlink 是一种基于 socket 缓存队列的，内核与用户态应用之间传递的消息的异步通信机制。FAST 编程库会在 UA 实现时编译到 UA 地址空间中，为防止不同 UA 访问 FAST 内核中共享数据结构存在竞争，FAST 编程库在设计时考虑了对 NetLink 映射表等共享资源访问的互斥机制。

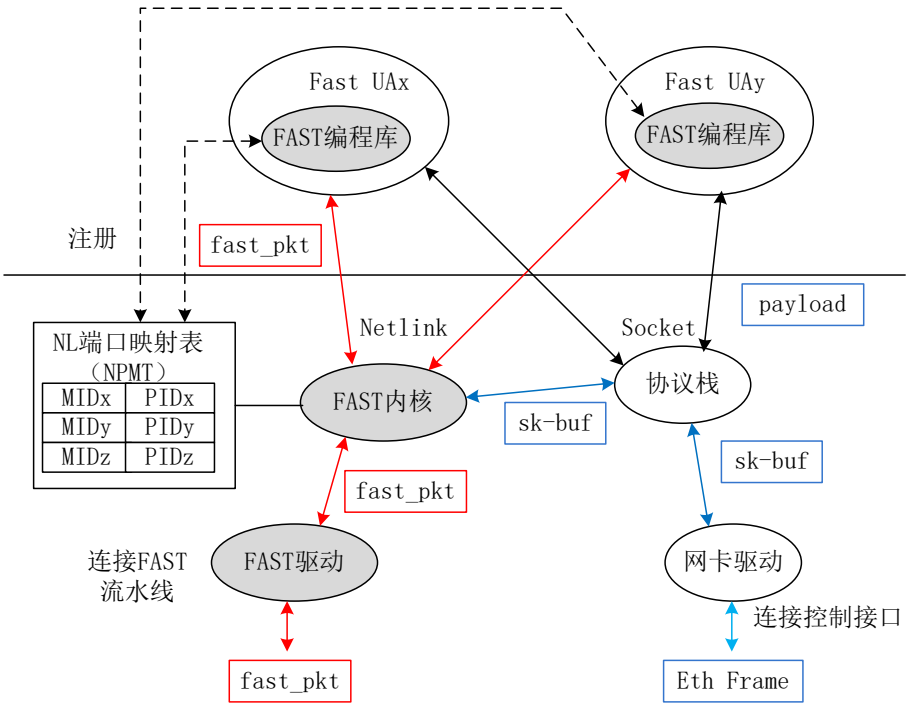


图 5-1FAST 平台软件实现示意图

FAST 内核维护 NPMT 数据结构。每个 UA 启动时，会通过 FAST 编程库提供的初始化函数向 FAST 内核注册，由用户显式地为 UA 分配一个唯一的 MID 编号。FAST 内核对分组处理的操作如表 5-1 所示。

表 5-1FAST 内核对分组处理的操作

分组来源	分组格式	处理方式
FAST 驱动	FAST 分组	（1）DMID= 128，将 FAST 分组转换成 sk-buf 的格式，送内核协议栈处理； （2）DMID>128，查找 NMPT 表，获取 DMID 对应 UA 的 NetLink 端口号，由内核主动向 UA 发起会话，将分组传递到 UA。若查表不命中，则丢弃分组； （3）DMID<128，丢弃
协议栈	Sk-buf	将分组格式转换成 FAST 分组格式，向 FAST 驱动发出
UA	FAST 分组	（1）DMID<128，直接送 FAST 驱动发出； （2）DMID=128，将 FAST 分组转换成 sk-buf 的格式，送内核协议栈处理； （3）DMID>128，查找 NMPT 表，获取 DMID 对应 UA 的 NetLink 端口号，由内核主动向 UA 发起会话，将分组传递到 UA；若查表不命中，则丢弃分组；

因此 FAST 内核根据 FAST 分组的 DMID 实现了 FAST 流水线，多个 FAST UA 以及协议栈之间的分组交换。从协议栈角度看，FAST 内核是一个特殊的网络接口，在内核中的地位与标准的网络接口驱动基本一致。由于协议栈发出的分组无分组的输出接口信息，该分组进入 FAST 流水线后需要通过正常的转发获取输出接口的信息。

FAST 驱动负责控制与 FPGA 的 DMA 通信，FAST 驱动与硬件平台密切相关，针对不同实现平台，DMA 可以是轮询方式，也可以是中断方式。不同平台中硬件 DMA 寄存器的地址也有所差异，因此 FAST 驱动移植是 FAST 平台软件移植的关键。由于 FAST UA 编程基于 FAST 编程 API 和 Linux 标准系统调用，因此不同的硬件平台对 UA 编程是完全透明的。

六、 用户应用（UA）编程

6.1 硬件抽象

6.1.1 虚拟地址空间

FAST 硬件平台中 FPGA OS 和 UM 中都存在需要软件参与管理的寄存器、计数器和 RAM 等资源。虚拟地址空间（Virtual Address Space，简称 VAS）实现了对这些硬件资源的抽象，是 FPGA 硬件与 CPU 软件间的重要界面。

FPGA OS 是平台相关的，不同平台中 FPGA OS 实现的方式可能存在较大差异。但这些平台需要为软件的管理操作提供相同的抽象，例如接口状态寄存器、接口控制寄存器，接口计数器等。

每个 FPGA OS 都需要提供一个物理接口状态寄存器，每个物理端口的状态对应其中的一个比特位。比特位为 0 表示该接口处于 down 状态，为 1 表示接口处于 up 状态。CPU 相关软件通过定时轮询该寄存器获取端口状态的变化，当发现某个物理端口对应的状态位由 0 变成 1 时，表示该接口状态由 down 变成 up。对于标准的 openflow 交换机，此时会触发向控制器发送一个异步消息。控制器获取交换机特定接口由 down 变成 up 后，会立刻触发 LLDP 协议，启动通过交换机的接口进行拓扑发现过程。

FPGA OS 中还需要提供其他软件可管理的资源，例如每个接口对应的分组收发计数器，接口的配置寄存器等。如果平台还有一些软件通过 FPGA 进行控制的外部设备（如 PHY 芯片，面板指示灯等），也需要在 FPGA 中提供相关寄存器，由软件配置这些寄存器进行管理。

如果平台提供了查表协处理器，支持 FAST UM 通过 ToMatch 和 FromMatch 接口实现关键字的查表，那么 FPGA OS 必须向软件提供管理查表协处理器的接口，即软件通过 FPGA OS 实现对查表规则的管理，而不需要 FAST UM 的干预。

以 FPGA OS 中的接口状态寄存器为例，一些平台的 FPGA OS 可能通过芯片管理引脚（MDC/MDIO）从物理层芯片中读取对应端口的 up/down 状态，而另外一些平台可能从内嵌的 MAC 核中获取接口的 up/down 状态。如何获取寄存器信息对软件是透明的，软件只需使用 VAS 空间定义的接口状态寄存器地址就可以获取该寄存器的值，因此是可以跨平台迁移的。

FAST UM 内部也有大量的状态和配置信息需要软件进行管理，典型包括以下几类。

（1）UM 内部控制分组转发的数据结构。

（2）UM 内部的各种计数器

（3）UM 逻辑工作的参数表，如令牌桶、输出调度参数表等；

（4）UM 逻辑的工作状态，如存储分组的 FIFO 队列长度等。

与 FPGA OS 中可管理的硬件资源的数量和种类相对固定不同，FAST UM 中需要软件管理的资源在类型和数量上随着 UM 功能的不同而不同。例如，同样是 UM 内部的 RAM，可能包含的是 openflow 的流表、标准二层交换机的转发表，路由器的 FIB 表，网络接口的邻接表等，这些表具有不同的语法和语义，这对软件的管理提出了较大的挑战。

虚拟地址空间就是将 FPGA OS 和 FAST UM 中需要软件访问的资源（寄存器、计数器、控制表等）统一编址到一个 64 位的地址空间，为 FAST 平台软件（如表管理软件、设备管理软件）和各类用户定义软件（UA，如 FAST 白皮书中提到的精准测量服务 AMS 软件）访问这些硬件资源提供统一的接口和方法。

FPGA OS 中的端口状态寄存器、端口计数器和外设控制寄存器分别映射到 OAddrx，OAddry 和 OAddrz 地址。而 FAST UM 中流表项分别映射到 UAddr0，UAddr1 和 UAddrN-1 等。

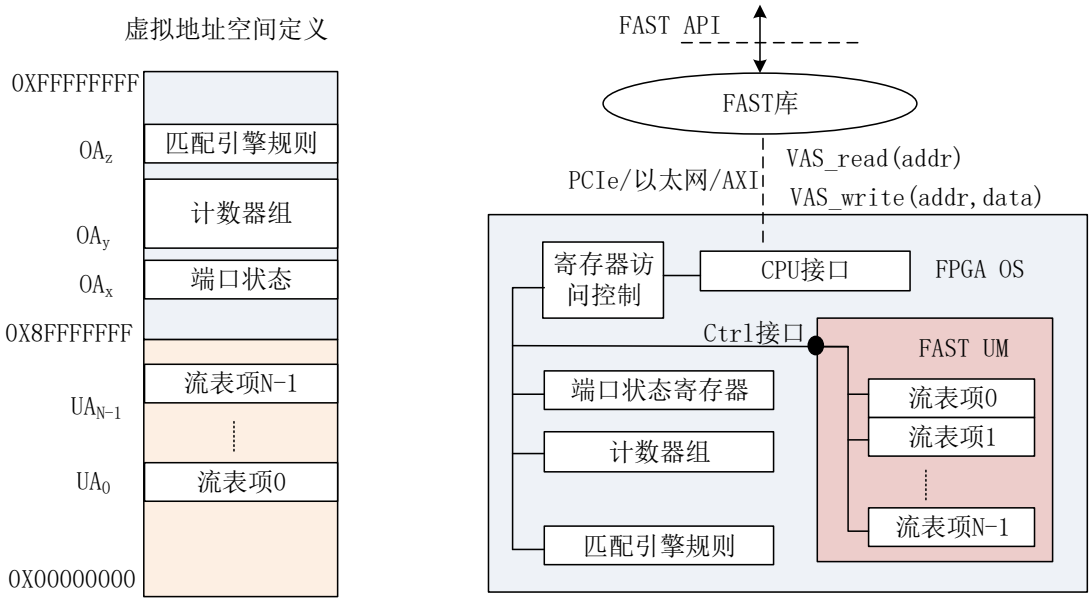


图 6-1 虚拟地址空间原理

UM 定义空间的划分如表 6-1 所示。

表 6-1 FAST 流水线的虚拟地址空间范围

空间范围	名称
0x0008 0000-0x0008 1FFF	DataCahe 空间
0x0008 2000-0x0008 3FFF	GPP 空间
0x0008 4000-0x0008 5FFF	GKE 空间

0x0008 6000-0x0008 83FF	GME 空间
0x0008 8400-0x0008 9FFF	GAC 空间
0x0008 A000-0x0008 BFFF	GOE 空间

例如 FAST 规定 FPGA OS 中接口状态寄存器，接口控制寄存器一级时间戳寄存器的定义如表 6-2 所示。

表 6-2 FPGA OS 中固定的寄存器定义

地址	含义
0x0018 0000-0x0018 FFFF	Port1 寄存器空间
0x0019 0000-0x0019 FFFF	Port2 寄存器空间
0x000A 0000-0x000A FFFF	Port3 寄存器空间
0x000B 0000-0x000B FFFF	Port4 寄存器空间

详细的 FPGA OS 中虚拟地址空间的定义见具体的平台说明文档。

6.1.2FAST 分组

fast_packet 定义了 FAST 平台内部数据报文规范，格式如图 6-1 所示。

```
structfast_packet{
    structum_metadata um; /*UM 模块数据格式定义*/
    u8 data[0];              /*完整以太网报文数据*/
}__attribute__((packed));
```

图 6-1 FAST Packet 的定义

6.1.3 重定向表规则

fast_rule 是定匹配表规则的数据结构，实现硬件对于报文的转发功能，定义如图 6-2 所示。

```
structfast_rule{
    structflow rule;          /*规则内容存储结构*/
    structflow mask;          /*规则掩码设置结构,与上面一一对应*/
    u32 priority;        /*规则的优先级设置*/
    u32 valid;/*规则的有效标志设置,要删除一条规则只需要将此标记置为0,并更新到硬件*/
    u32 action;               /*规则所对应的执行动作*/
    u32pad[29];        /*目前,根据硬件规则空间做的保留*/
```

```
}__attribute__((packed));
```

图 6-2 FAST Rule 的定义

6.2 编程模型

6.2.1ODP 编程模型的特点

FAST 编程模型称为 ODP(Open DataPath)模型。ODP 编程 API 主要实现 UA 向平台注册，配置硬件流水线，从硬件流水线接受分组以及向硬件流水线发送分组等功能，如表 6-4 所示。

表 6-3 ODP 主要的编程 API

```
int fast_ua_init(int mid,fast_ua_recv_callback callback) //UA 注册
Int fast_add_rule(struct fast_rule *rule)//配置规则
void fast_ua_recv()//启动接受分组
void fast_ua_hw_wr(u8 dmid,u32 addr,u32 value,u32 mask);//通过环形控制通路写入数据
int fast_ua_send(struct fast_packet *pkt,int pkt_len)//发送分组
```

ODP 有两个重要特点。第一个特点是支持用户直接对网络设备的硬件流水线进行编程。通过 fast_add_Rule() API，UA 可以直接配置硬件流水线中的流表规则。规则的内容包含匹配的关键字值，匹配的掩码，匹配后执行的动作等。其中匹配执行的动作中包含 DMID，PktDes，OutPortBM 等字段，根据这些字段的组合，可以控制分组在硬件流水线输出后的流向。

例如，某个 UA 可进行所有 ARP 分组的处理，对特定视频流的传输质量进行监测。针对 IPTV 传输质量监测需求，RFC4445 定义了两个关键指标，一是延时抖动，二是丢包率。媒体传送 指数 MDI（Media Delivery Index）指标，对流媒体在网络中传输的延时特性 DF（Delay Factor）和丢失率指标 MLR（Media Loss Rate）进行测量。

IPTV 提供商进行 MDI 指标测量在网络中一方面可以评估 ISP 提供的传输服务质量，另一方面可以对用户满意度体验 QOE 进行评估。

第二个特点是支持软硬件协同处理，即用户 UA 可以通过 Metadata 直接访问到 FPGA 的处理结果，因此可以将一些适合硬件处理的复杂操作，如分类，

卸载到硬件中处理。

6.2.2 与现有编程模型比较

FAST ODP 编程 API 与其他用户空间网络编程 API 的比较见表 6-4。

表 6-4 ODP 与其他编程 API 的比较

编程模型	面向对象	主要应用
Socket	通信对端	实现网络中应用程序之间的通信，通常基于 TCP 或 UDP 协议
NetLink	内核模块	用户空间程序与内核中的模块进行通信，例如 OVS 用户控件程序通过 NetLink 与内核中的 datapath 模块通信
libpcap/Libnet	网络接口	用户控件模块直接从网络接口驱动接收以太网分组或发送分组，旁路 TCP/IP 协议栈
DPDK	网络接口	用户空间程序直接从网络接口收发分组，旁路掉接口驱动，内核的缓冲区管理等机制
ODP	数据平面	对数据平面进行编程

FAST_UA 是 FAST 构架下软件部分的用户应用（User Application），可以实现平台无关的核心管理软件，转发面扩展软件以及配置管理关键等功能，由用户在 FAST 构架下根据编程手册自行开发。本文将详细指导用户在 FAST 构架下，开发用户应用模块（FAST_UA）。

6.3 编程 API

FAST 提供三类通用编程 API，包括 UA 管理 API，规则管理 API 和硬件管理 API。

6.3.1UA 管理 API

UA 管理 API 用于 UA 向平台注册，送平台接收 FAST 分组，向平台发送 FAST 分组，以及打印接收的 FAST 分组等。

fast_ua_init()：UA 模块初始化函数

fast_ua_destory()：UA 模块注销函数

fast_ua_send()：UA 发送报文功能函数

fast_ua_recv()：UA 启动报文接收线程

print_pkt()：打印 fast 结构报文数据

6.3.2 规则管理 API 管理

规则管理 API 负责配置 FPGA OS 中嵌入匹配引擎中德规则，包括规则的增加，删除，修改和打印等。

print_hw_rule()：打印配置到硬件的规则；

print_sw_rule()：打印软件缓存的规则

init_rule()：初始化规则模块

fast_add_rule()：添加一条规则

fast_modify_rule()：修改指定位置的规则

fast_del_rule() ：删除一条规则

read_hw_rule()：从硬件读取一条指定的规则

6.3.3 硬件管理 API

硬件管理 API 提供了硬件资源初始化，直接读写虚拟地址空间中的数据等。

int fast_init_hw()：初始化硬件

fast_distory_hw()：释放硬件资源

fast_reg_rd()：读取指定寄存器

fast_reg_wr()：往硬件寄存器写入一个指定的值

fast_ua_hw_wr()：通过环形控制通路向硬件写入一个特定的值

fast_ua_hw_rd()：通过环形控制通路读取硬件中的数据

6.3.4 订制功能 API

定制功能 API 实现 UA 对 UM 中用户定制功能管理的 API。

例如，当用户在 UM 用户定义输出引擎（UDO）中实现对 Netflow 支持时，需根据管理员的配置，基于分组的 MD 和 PFV 等信息生成 netflow 分组，向外部发送相关的统计信息。因此 UDO 中需要保存外部 Netflow 数据包采集软件所在的服务器地址 Netflow_Server_Addr。虽然 UA 可通过硬件管理 API 通过写寄存器的方式直接配置服务器地址（Netflow_Server_Addr 的地址寄存器已经

在虚拟地址空间中定义），但不直观，易于出错。

因此，可设计专门的定制功能库，其中包含用于读写 Netflow_Server_Addr 寄存器的 API，UA 直接调用这些 API 对 Netflow_Server_Addr 进行管理，不但可读性好，而且不易出错。

需要注意的是，由于 Netflow_Server_Addr 寄存器映射到虚拟地址空间中的位置是确定的，因此对应的订制库中相应的 API 实现是完全与具体的硬件平台无关的，具有很好的可移植性。

6.4 其他应该考虑的问题

6.4.1UA 在设备中的角色

UA 在网络设备中有两种角色。一是实现分组处理流水线中特定的分组处理功能，作为嵌入硬件流水线的“协处理器”；二是实现相对独立的，动态加载的数据平面服务。例如 UA 可以在 FAST 平台上实现相对独立的服务或 Middlebox 功能，例如实现异常流量监测功能，web 防火墙，以及可编程的流量产生等功能。

6.4.2UA 的地址

每个 UA 可以使用 socket 机制，独立的与远端的实体进行通信。如果 FAST 平台工作在交换机模式，UA 使用交换机控制软件的 IP 地址。

七、 支持 FAST 的 iRouter 平台

我们基于大容量 FPGA 和多核 CPU 实现了功能可扩展的可编程网络平台 iRouter。iRouter 参照 FAST 交换模型，将 FPGA 分组硬件处理流程划分为 5 级流水线，分别由 5 个独立的模块实现分组解析、关键字提取、查表、动作执行和输出调度功能，同时支持在 CPU 上的用户空间以进程模式编写各种扩展的软件模块，通过软硬件协同支持各种网络功能的扩展。

7.1 iRouter 实现

iRouter 平台基于 Altera 公司的大容量 FPGA 和 Intel 多核 CPU 实现。内部

采用 FAST 交换模型，支持软硬件协同的分组交换处理。用户可通过定制并增加硬件模块和软件模块来实现分组处理功能的扩展。

7.1.1iRouter 系统组成

（1）FPGA OS

FPGA OS 是 FPGA 中的逻辑，为用户逻辑 UM 提供运行环境。与操作系统类似，FPGA OS 一方面向 UM 屏蔽不同板卡和 FPGA 的差异，另一方面为简化 UM 的设计提供各种通用服务，如分组的接收和发送，外部存储器的访问，与 CPU 通信的高速 DMA，以及分组处理中需要的规则匹配等。

FPGA OS 还可以为简化 UM 的设计提供各种通用服务，如分组的接收和发送，外部存储器的访问，与 CPU 通信的高速 DMA，以及分组处理中需要的规则匹配等。

（2）iRouter 支撑软件

iRouter 支撑软件包含 Linux 用户空间实现的 iRouter 库，在 Linux 内核中实现的 iRouter 内核路径控制，以及内核中的 UM 仿真软件三部分。

iRouter 库又分为 iRouter 标准库和 iRouter 定制库。iRouter 标准库对应标准的 iRouter UM 实现，为 UA 提供分组收发，规则配置，平台信息获取（如接口计数器）等功能；iRouter 定制库实现对 UM 中用户扩充模块的管理和访问。通过 Netlink 机制与内核中的 iRouter 路径控制软件通信。iRouter 内核路径控制软件主要在内核中实现多个 UA 与 UM 通信的 MUX/DMUX 功能。内核 UM 仿真软件主要实现对 FPGA 中 UM 功能的仿真，在不包含 FPGA 的标准终端和服务器上支持 UA 的运行。

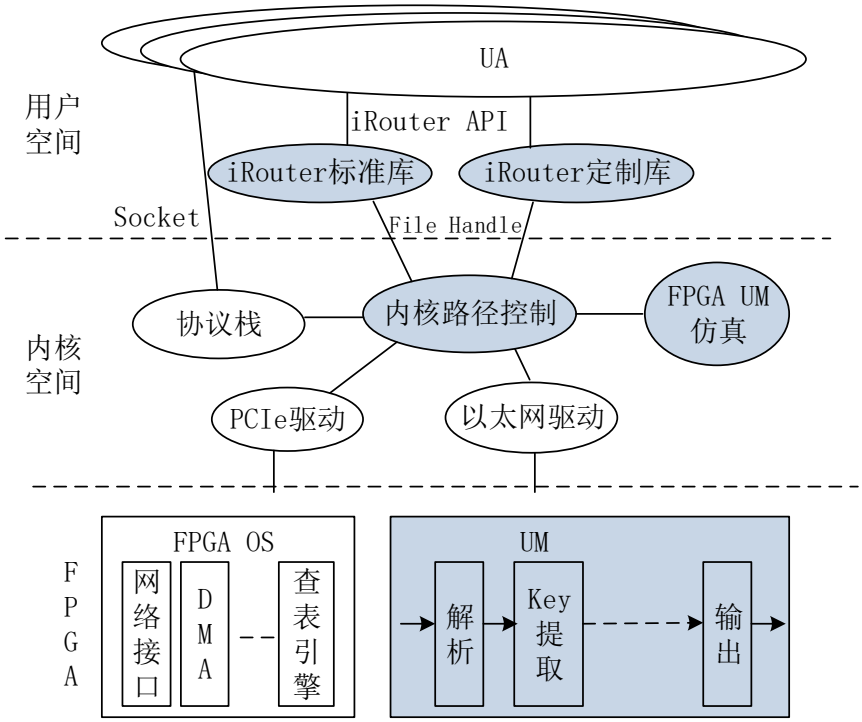


图 7-1 iRouter 系统的组成

标准 iRouter 库主要实现以下功能。一是表管理功能，负责所有控制器配置表格的维护，控制器 flowmod 命令对表操作的命令主要由核心管理软件响应和实现，并实时把新添加的流规则或者更新的流规则经过流表映射的算法，I/O 分组驱动写入到硬件 RAM 中，硬件根据表项转发，处理分组；另外该软件还提供删除流的接口，用来支持删除软件中的流规则信息以及调用接口删除硬件中对应的流规则信息；二是硬件算法实现管理功能。例如有些硬件采用 TCAM 实现带掩码的查表，而有些则采用 SBV 算法实现 288 位带掩码的查表。算法相关软件需要根据表管理软件要求，根据平台相关的算法实现特点，实现对硬件中流表的管理；三是统计管理功能，实现平台的各类计数器的管理，将硬件实现的计数器（如流表项匹配分组数目计数器、端口接收计数器）和软件实现的计数器（流表项存活时间）进行统一的管理。例如：数据平面实现的令牌桶参数、输出调度器配置的参数计算等；四是内嵌的 Openflow 协议通道，支持平台与外部的 SDN 控制器建立连接，传递流表，流量统计信息和未命中报文通过 Openflow 通道上传；五是链路状态探测功能，对各端口链路状态进行实时探测，发现链路状态改变后将探测到的链路信息通过 OFP_PORT_STAUS 消息上报至控制器。

（3）用户应用 UA

用户在 Iinux 用户空间编写的分组处理进程。每个处理进程对应 FAST 模型中的一个软件模块，硬件流水线通过设置分组 Metadata 中的目的模块号来指定

接收该分组的软件模块。

每个 UA 在启动时向平台进行注册，获取自己的 MID 值，通过 iRouter 标准库提供的分组接收函数接收分组和发送分组，并向流水线配置相应的转发规则，获取 FPGA OS 中相应的状态，如接口的计数器信息，对 UM 中的硬件流水线进行管理。根据 FAST 模型，当 UA 需要将分组发回硬件流水线时，需要在分组的 Metadata 中指明接收该分组的硬件流水线模块号。

由于 UA 是用户态进程，因此也可以通过 socket 机制与远程的终端进行通信，甚至可以通过 Libpcap 从标准的网络接口抓包，通过 libnet 向标准网络接口发送分组。UA 也可以调用标准 C 库中的函数实现特定的处理功能，如文件处理等。

（4）用户模块 UM

FPGA OS 为用户模块 UM 提供标准的运行环境。UM 主要实现 FAST 模型中的硬件流水线功能。

7.1.2 iRouter 的实现

我们实现的 iRouter 实物如图 7-2 所示。支持 2 个万兆以太网接口和 8 个千兆以太网接口。其中 FPGA 为 Altera 公司 Arria V GT 系列，型号为 5AGTMC3D3F31I5N，拥有 58,900 个 ALM，156,000 个逻辑单元（LE）。FPGA 外部包含 1 片 18Megabit 高速同步 SRAM，以及 4 片 2GbDDR3 SDRAM，因此具有较强的片外存储能力。

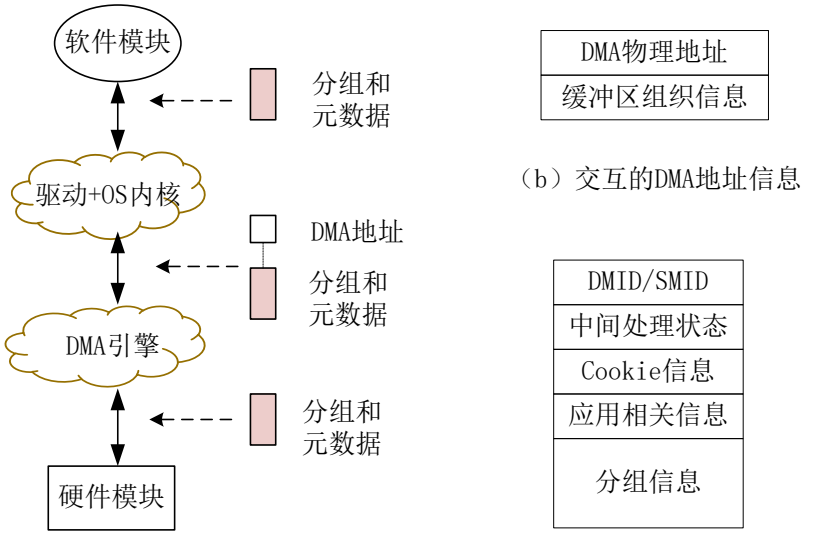
图 7-2 iRouter 实物图

设备采用 Intel 双核 Atom CPU，提供 1 路 RJ45 管理串口，1 路 10/100/1000Mbps 以太网口，1 路 USB2.0 接口。32G 板载电子硬盘，用于存储操作系统。

iRouter 中的 FPGA 与 CPU 通过 PCIE2.0 总线连接，采用 SDB-DMA 机制，由 FPGA OS 实现 DMA 缓冲区的分配与回收，减轻了 CPU 的处理负担。因此具有较高性能。

7.1.3 软硬件交互的实现

iRouter 采用 SDB-DMA 和 FAST 交换架构，其软硬件交互的原理如图 7-3 所示。以分组的上行传输为例，当 UM 中的硬件流水线需要将分组送给用户空间的 UA 时，根据 FAST 模型，该分组和元数据将会通过 UM 的 toPort 接口发出，FPGA OS 根据元数据中的标志位判断该分组需要送 CPU 处理，然后该分组将送给 FPGA OS 中的 SDB-DMA 引擎处理。



（b）交互的DMA地址信息

（a）交互的流程

（c）交互的分组和元数据信息

图 7-3 FAST 的软硬件信息交互原理

图 7-3（b）包含了为支持 SDB-DMA，在分组缓冲区中需要预先设置的信息，包括缓冲区的物理地址以及缓冲区的组织信息。缓冲区的物理地址供 FPGA 中的 SDB-DMA 使用，即使用该地址发起 PCIe 读写操作，CPU 硬件会将该地址映射到内存空间。缓冲区组织信息包含接收链表的下一个缓冲区的地址信息，该信息由 CPU 的 DMA 驱动使用，因此地址是虚拟地址，必须由 CPU 核进行虚实地址转换后使用。

图 7-3（c）包含了 UM 流水线与 UA 进行分组交换时的元数据信息。元数据信息直接在分组头部，主要包含通信的目的 MID 和源 MID 号，各种中间处理状态信息，如 FPGA OS 标记的分组接收时间戳，分组接收端口号，硬件流水线通过查表的到的分组的流 ID 号等。UA 还可以向 GME 的表中设置 cookie 信息，用于硬件流水线告知 UA，该分组上报 UA 的原因，如匹配的规则 ID 等。应用相关信息是与 iRouter 扩充的特定功能相关的，是对元数据的扩展，例如本文 7.3 节中，为了实现 UA 与流水线交换的分组精准发送的时间信息等。

iRouter 的可扩展分组处理流水线如图 7-4 所示，主要分为数据平面硬件模块、数据平面软件 UA 和控制平面软件三部分组成。模块是 iRouter 平台中网络功能实现的基本单元，每个模块由 8 位的模块 ID（MID）标识，其中 0-127 代表数据平面硬件中的模块，128 代表控制平面模块（TCP/IP 协议栈），129-255 代表数据平面软件扩展的模块。通过定义控制不同分组处理通过的模块序列，可实现网络设备多样化的分组处理流程，通过扩充新的软硬件模块，可以支持新的分组处理功能。

每个输入接口接收的分组（简称 P）首先被加上 32 字节的 Metadata 信息（简称 M），M 包括分组接收的时间戳，输入接口号，输入接口的接收序列号，以及分组在流水线处理中保存的中间状态，如分组的下一处理模块号（DMID），查表匹配的流 ID 等；

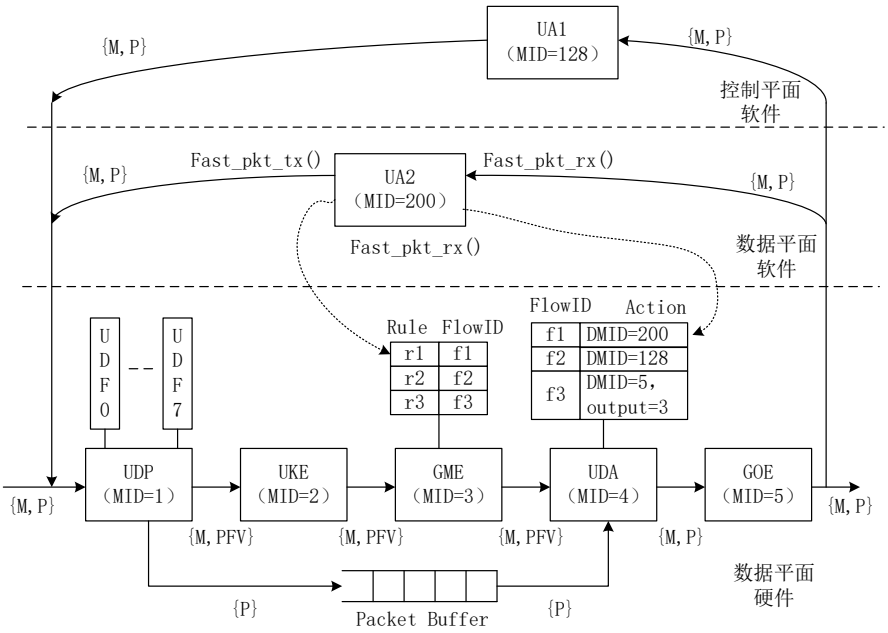


图 7-4 iRouter 的可扩展分组处理流水线

数据平面硬件由 FPGA 实现，分为 FPGA OS 和用户模块（UM）。FPGA OS 实现了网络设备平台通用的处理功能，如分组的接收发送、CRC 校验、外部存储器接口以及与 CPU 通信的 DMA 引擎等。不论设备实现的是交换机还是网关等其他功能，FPGA OS 都保持不变，而 UM 逻辑存在差异。

数据平面 UA 软件是位于 CPU 用户空间、基于 FAST 开放数据平面接口（ODP API）开发的与用户功能相关的软件进程。该进程通过 add_rule（）函数直接配置硬件 UM 中相关表格，使数据平面硬件将满足特定规则的数据平面分组定向到本进程处理。例如，图 1 中 MID 编号为 200 的用户进程通过向 GME 流表和 UDA 的动作表配置（r1-f1）以及（f1-MID=200）的规则，将满足规则 r1 的所有分组定向到 UA 处理。UA 通过 UA_recv()函数接收分组 P 和相

应的 metadata，对分组处理后，通过 UA_send()函数将分组发出。UA 在发送分组时，通过设置 metadata 中的 DMID 值，可以指定接收处理该分组的硬件流水线模块。例如将分组直接从输出接口发出时，可在 Metadata 中设置 DMID=5 以及相应的输出接口；若要硬件流水线重新解析处理该分组，只需将 DMID 设置成 1。

iRouter 数据平面模块通过将分组 metadata 中 DMID 设置为 128，即可将分组送给控制平面协议栈，由控制平面协议栈对分组进行分析后，送不同的协议模块处理（如 ARP 协议、各种路由协议），或者通过 socket 接口送本地的应用程序。

八、 结束语

FAST 开源项目希望能够为国内高等院校，相关研究所和公司中从事网络技术研究人员提供开源的可编程开发平台，避免研究人员将大量精力花费在网络平台的研制上而不断的“重新发明轮子”。

FAST 项目中采用的软硬件协同分组处理的架构能够有效支持在网络设备上部署新的协议，新的分组处理机制和新的服务，因此能够快速实现对网络新技术的验证。

FAST 项目目前仅实现了路由和交换的基本平台，仍有大量的软硬件开发工作需要进行，希望更多的研究人员关注 FAST 开源项目，参与项目的开发，或为 FAST 项目的发展提出意见和建议。