FastRMT: 一种面向微体系结构创新的高速数据平面可编程系统

杨翔瑞[1)] 曾令斌[1)] 刘忠沛[1)] 陈颖文[1)] 吕高锋[1)]
杨 程[2)] 苏金树[1)]

[1)](国防科技大学计算机学院 长沙 410072)
[2)](湖南先进技术研究院 长沙 410006)

摘 要 网络数据平面可编程(Data Plane Programmability)给网络转发设备的数据平面赋予强大的可编程性,在不更换设备的情况下,可以动态部署新型机制与服务,例如路由转发核心机制、网络安全控制功能、网内计算加速服务等.由此,数据平面可编程成为业界和学术界高度关注的新兴技术,并已在主流云服务提供商投入应用.可重构匹配表架构(Reconfigurable Match Table Architecture,RMT)由于出色的处理性能和采用P4语言灵活编程的特性,成为数据平面可编程的热点研究方向.然而,受困于RMT架构复杂的体系结构设计、芯片闭源的服务机制以及门槛较高的FPGA设计开发,使得RMT研究人员难以通过FPGA,对RMT创新设计以及100 Gbps以上真实性能场景进行敏捷验证.本文提出并实现了一种数据平面可编程系统FastRMT,首次开源了FPGA级的系统实现.FastRMT支持RMT架构可编程协议解析、自定义规则匹配、超长指令字的并发动作执行引擎等核心功能,支持P4语言对系统进行编程.FastRMT具备松耦合与模块化的特点,研究人员可以替换模块或者对系统进行动态重构,从而实现新型机制或体系结构的敏捷开发与验证.本工作包含交换机原型与网卡原型两种版本,支持主流FPGA芯片,系统可完成100 Gbps的报文线速处理能力,1500 B报文处理延迟仅为1.22 μs,体现了FastRMT作为基础框架对微体系结构创新和生产线级别验证的优势和可行性.

关键词 数据平面可编程;可重构匹配表;微体系结构;FPGA原型;可编程协议无关报文处理
中图法分类号 TP391 **DOI号** 10.11897/SP.J.1016.2024.00473

FastRMT: A High-Speed Data Plane Programmable System for Micro-Architecture Innovation

YANG Xiang-Rui[1)] ZENG Ling-Bin[1)] LIU Zhong-Pei[1)] CHEN Ying-Wen[1)]
LV Gao-Feng[1)] YANG Cheng[2)] SU Jin-Shu[1)]

[1)](*School of Computer Science*, *National University of Defense Technology*, *Changsha* 410072)
[2)](*Hunan Institute of Advanced Technology*, *Changsha* 410006)

Abstract Network data plane programmability (Data Plane Programmability) gives powerful programmability to the data plane of network forwarding devices, allowing the deployment of new network protocols, updating security functions, and providing in-network computing acceleration services without updating devices. Due to these advantages, data plane programmability has

收稿日期:2023-04-01;在线发布日期:2023-12-26.本课题得到国家自然科学基金(62372462,U22B2005)、湖南省自然科学基金(2023JJ40682)、国防科大青年自主创新科学基金(ZK2023-13)与长沙市杰出创新青年培养计划(KQ2209027)资助.**杨翔瑞**,博士,讲师,主要研究领域为可编程网络、领域定制网络.E-mail:yangxiangrui11@nudt.edu.cn.**曾令斌**,博士研究生,主要研究领域为数据中心网络、智能软件技术.E-mail:zenglingbin16@nudt.edu.cn.**刘忠沛**,博士研究生,主要研究领域为高性能网络与智能网络.**陈颖文**,博士,教授,主要研究领域为无线网络与算力网络.**吕高锋**,博士,副研究员,主要研究领域为数据中心网络与高速互联网络.**杨 程**,博士,副研究员,主要研究领域为分布计算.**苏金树**(通讯作者),博士,研究员,博士生导师,中国计算机学会(CCF)会士,主要研究领域为网络空间安全与因特网体系结构.E-mail:birchsu@139.com.

become an emerging technology that is highly concerned by the industry and academia, and has been put into use by mainstream cloud service providers. Among them, Reconfigurable Match Table architecture (RMT) has become a hot research direction in the programmable data plane due to its excellent processing performance and flexible programming capability with P4 language. However, due to the complex design of the RMT architecture, the closed-source service mechanism of the chip, and the high development threshold of FPGA, it is currently difficult for researchers to innovatively design the micro-architecture of the RMT architecture through FPGA and perform in real performance scenarios (above 100 Gbps). Motivated by the need of research on data plane programmable micro-architecture, this paper proposes an open-source design of a flexible and high-speed programmable data plane prototype system for the first time. The system supports core functions such as RMT architecture programmable protocol parsing, custom rule matching, and action engines based on very-long instruction words, and supports programming of the system via P4 language. In addition, FastRMT also has the characteristics of loose coupling and modularity, which facilitates researchers to replace or reconstruct modules, thereby enabling agile development and verification of new mechanisms or architectures. This work includes two versions of switch prototype and network interface card prototype, supporting mainstream FPGA chips. The system can complete 100 Gbps line-speed packet processing capability, and the 1500 B packet processing delay is only 1.22 μs, which reflects the micro-architecture innovation of FastRMT as a basic framework. and the advantages and feasibility of production line-level verification.

Keywords data plane programmability; Reconfigurable Match Table (RMT); micro-architecture, FPGA Prototype; Programming Protocol-Independent Packet Processers (P4)

1 引 言

当前新型网络协议与分布式应用层出不穷,按需定制转发设备报文成为日益迫切的需求[1]. 传统交换架构仅可支持固化的协议解析与简单的转发等操作,并不具备可重构或可编程的特性,因此无法支持复杂多变的转发业务逻辑. 数据平面可编程技术则通过可重构的匹配转发模型与定制指令的专用网络处理器实现了在不替换物理设备与网络处理芯片的情况下支持新型网络协议与网络应用. 其中,Bosshart等人[2]提出的数据平面可编程流水线架构RMT由于其较高的灵活性与极佳的处理性能受到了热点关注. RMT在OpenFlow[3]基础上,通过协议无关的报文头解析器,支持超长指令字的动作执行引擎等功能模块实现了可编程性和处理性能的较好统一. 该架构原生支持P4语言编程,并能够快速部署包括VxLAN、RCP等新型协议,负载均衡[4]、防火墙[5]等网络功能,以及简单的网内计算[6]等服务. 由于以上优势,RMT架构得到了国内外主流云服务提供商的广泛关注并逐步投入运营网络. 例如,阿里云利用基于RMT架构的Tofino交换机部署了其新一代确定性网络uFAB架构[7],并实现了实时随流检测与微秒级的TCP拥塞响应. 基于可编程数据平面的新一代网络转发设备极大提升了网络灵活性与智能化水平.

目前,RMT架构与用于在RMT架构下描述网络转发行为的P4语言[8]仍然处在早期发展阶段,在有状态报文处理、复杂计算以及多租户隔离与资源共享等方面存在着明显不足. 例如,Gebara等人[9]指出,RMT架构仅可利用有限的片上内存管理简单状态,目前难以支持有状态网络报文处理,这导致涉及大量状态信息的网内计算难以部署. 另外,现有RMT架构受限于仅可从报文头部字段提取的头部字段以及架构自定义的metadata字段用于匹配-动作,无法对节点内突发事件进行响应[10-11]. 而随着更多网络服务(如负载均衡器、防火墙、网内遥测等)和网内计算应用(如键值存取、分布式机器学习等)被部署到云数据中心交换机与智能网卡侧,基于RMT架构的转发设备就应当支持不同网络业务在单节点的逻辑隔离与资源共享[12-13],从而满足安全与性能等多方面要求. 而从网内计算的角度来说,浮点数运算指令的缺失一定

程度阻碍了需要该类指令的网内计算应用部署[14]. 因此，虽然RMT架构得到了学术界与业界的广泛认可，但距离满足当前和未来的很多应用场景仍有较多不足. 在这种背景下，国内外研究人员近年来倾向于对RMT架构进行优化与重构[15-16]，期望可以解决上述问题.

然而，与针对RMT架构的研究迅速增长形成对比的是，当前绝大多数研究难以对所提出方法的功能、性能以及安全性进行敏捷验证，这在很大程度上限制了数据平面可编程技术的快速发展. 导致该问题的核心原因是当前研究人员缺少一种能够实现寄存器级仿真(Register Transfer Level，RTL)，且便于灵活重构的RMT架构实验平台[17]. 目前，基于软件的P4仿真工具如BMv2[18]等仅能作为网络行为级仿真工具，无法在真实硬件(如FPGA)实现RMT数据平面寄存器级仿真. 这种软硬件间鸿沟导致绝大多数软件仿真器无法用于支持数据平面可编程微体系结构研究. 而支持RMT架构的唯一商用级交换芯片Tofino[19]则未向用户开放其内部微体系结构设计，仅允许用户通过P4对其数据平面行为进行描述，不可观察或修改芯片内部逻辑. FPGA作为一种能够在寄存器级对真实微体系结构进行实际验证的平台，其虽然能够在性能、资源开销、逻辑可行性等方面进行更充分验证[20]，但其学习门槛高，从头搭建验证系统难度较大，一般采用已具备PHY与MAC层逻辑、PCIe/DMA引擎以及基础转发模块的FPGA网络转发为平台进行敏捷系统验证[20-21]. 然而，RMT作为一种复杂的可编程网络分组处理流水线，仅基于上述FPGA网络原型仍需要巨大工作量才能部署基础的RMT转发逻辑开展各类验证实验. 从目前来看，业界仍缺少一种包含RMT基础逻辑底座的FPGA开源设计与原型系统用于协助研究人员开展相关研究. 例如，目前业界可使用基于NetFPGA的P4NetFPGA[22]与P4FPGA[23]等项目，但是这些工作仅支持将P4语言描述的网络功能映射为RTL逻辑，并未将RMT的微体系结构作为具体的硬件逻辑部署在FPGA芯片上，因此难以支持RMT架构下微体系结构的创新与敏捷验证. 本文认为，在FPGA平台上为研究人员提供一种灵活可扩展的RMT数据平面底座对相关领域的研究至关重要.

因此，本文提出一种灵活可扩展的RMT数据平面设计——FastRMT，以及对应的verilog实现，用于支持研究人员面向可编程数据平面微体系结构进行创新研究与敏捷验证. 与已有工作相比，本文的贡献如下：

(1)本文在当前一代FPGA片上资源约束下，完整提供了RMT架构的所有关键功能模板，包括：协议无关解析单元、动作指令匹配单元、基于超长指令字的动作执行引擎等关键功能模块. 而为了支持更加丰富的网内计算功能，FastRMT在经典RMT架构的基础上扩展了多类型计算、存储以及安全指令.

(2)本文基于ARM-AMBA通用总线机制的松耦合接口对各流水线与各功能模块间的接口与数据位宽等参数进行标准化，支持用户通过顶层宏定义等方式对数据通路位宽、流水级数量、各匹配表规格进行敏捷重构，便于用户按需定制流水线设计，从而根据FPGA平台需求适配不同的资源、频率以及时序约束.

(3)本文提出并部署了一种安全灵活的带内控制通路，支持用户采用UDP封装的控制报文灵活对各流水级的各功能模块的控制寄存器与各类表项进行灵活的批量读写操作. 因此用户可根据应用场景(如网卡、交换机、网关等)选择以本地或远程方式对系统控制寄存器与各类表项进行读写操作.

(4)本文采用verilog语言对所提出设计进行了原型实现与系统验证，能够满足多种类型的创新设计与实验需求. 本文同时基于两款学术界主流FPGA交换机[24]与网卡[25]原型开源了适配两类场景的RMT数据平面原型，便于使用者按需选用. 通过真实系统实验验证，FastRMT在当前主流FPGA芯片上支持最小256 B报文的100 Gbps线速处理能力，1500 B大报文的单向处理延迟仅为1.22 μs，因而能够支持研究人员利用FastRMT对所提出机制在当前主流云数据中心进行生产线级别验证. FastRMT近两年来成功支持了多种可编程数据平面创新机制设计与验证，充分证明了其可用性与易用性.

本文的主要结构如下：第二节介绍可编程数据平面的相关研究和本文的研究动机，第三节从体系结构角度对本工作进行整体介绍，第四节详细介绍本工作的系统实现和优化设计，第五节对原型系统的资源、性能与功能方面进行详细实验验证，第六节对本文进行总结与展望.

2 相关研究与动机

2.1 基于软件仿真的可编程交换

基于软件仿真的可编程交换平台支持P4等高层次网络描述语言进行编程，并通过CPU执行用户描述的网络功能. BMv2[18]是一种基于C++的面向P4编程的软件交换机，直接读取开源P4编译器输出的JSON文件作为网络分组处理的配置参数，并根据所编译的P4程序进行报文处理. 虽然BMv2可支持如带内遥测[26]、基于P4的网络应用验证[27-28]等一系列功能. 然而，其作为基于CPU实现的软件交换机，与其它可编程软件交换机如PiP4[29]等类似，仅可支持对网络行为的仿真，并未实现可编程数据平面的架构与报文处理过程，无法支持对可编程数据平面相关机制的研究.

2.2 基于ASIC的可编程交换

目前基于ASIC的商用可编程交换芯片可分为两种技术路线：一是基于流水线架构的PISA可编程网络处理芯片，代表性工作有Intel公司用于交换机的Tofino系列可编程交换芯片[19]与用于终端、服务器通用数据处理加速的Mount Evans IPU[30]. 基于流水线架构的PISA可编程网络处理ASIC原生支持RMT数据平面可编程架构，并支持基于P4的一系列软件定义网络功能与应用(如网络遥测[31]，负载均衡功能[32]以及分布式机器学习参数聚合[33]等). 然而，目前PISA架构的ASIC芯片设计细节均未开放，并且ASIC逻辑固定，无法支持研究人员根据需求进行修改，仅能够通过芯片所提供的编译器以及运行时工具配置固化的数据平面硬件架构，限制了研究人员对数据平面架构细节的研究广度.

二是基于多核架构的RoC(Run-to-Complete)可编程网络处理芯片，代表性工作有Juniper用于交换机的Trio系列[34]可编程交换芯片以及AMD的Pensando DSC-200系列DPU[35]. 基于多核架构的可编程交换芯片多采用通用或专用处理器核实现报文处理功能，一般支持使用P4或其它软件编程语言(例如C)对网络报文处理逻辑进行编程. 虽然其借助处理器核因而灵活性相较PISA架构较好，但是一般也拥有较大的性能开销，导致其难以达到较高处理性能. 除此以外，其与基于PISA架构的Tofino等芯片类似，采用逻辑固定的ASIC流片，无法支持研究人员根据需要修改数据平面硬件架构，因而难以支持可编程高速数据平面的相关研究.

2.3 基于FPGA的可编程交换

由于基于软件与ASIC可编程交换技术的上述特点，难以支持研究人员对可编程高速数据平面技术的研究验证，部分工作提出了基于FPGA的可编程数据平面. 与基于CPU的软件交换机相比，FPGA能在RTL实现对数据平面功能的硬件逻辑描述，从而利于研究人员对数据平面微体系架构与机制进行研究验证. 而与逻辑无法被更新的ASIC交换结构相比，FPGA支持对硬件逻辑根据需求随时进行重构，从而支持可编程交换数据平面的敏捷验证. 目前，领域内较知名的基于FPGA的可编程交换平台有NetFPGA[21,24]，FAST[36]等. 其中NetFPGA由斯坦福大学开发，采用Xilinx Virtex-7 690T并支持最多一组PCIe与四组网络端口进行网卡或交换原型验证.

为了支持对可编程高速数据平面的实验验证，Ibanez与Wang等人分别提出了P4NetFPGA[22]与P4FPGA[23]. 其中，P4NetFPGA基于NetFPGA SUME[21]项目，允许用户通过P4对交换逻辑进行描述，并通过Xilinx的P4-SDNet[37]工具链将P4逻辑转为verilog逻辑，从而烧写到NetFPGA SUME并实现P4描述的网络功能. 而P4FPGA则更进一步，其主要贡献是提出了一种高效的编译器设计，支持将P4语言转为FPGA平台无关的verilog逻辑，并且支持在不同FPGA型号上实现接近ASIC的报文转发性能.

然而，目前公开的基于FPGA的可编程交换实验平台均采用从P4到verilog/VHDL语言映射的技术路线，通过每次在FPGA上烧写不同的verilog/VHDL逻辑实现可编程特性，并未在FPGA上真正部署可编程数据平面转发模型. 从这个角度看，这些已有工作对支持基于P4的网络功能研究具有较好的支持，但由于并未实现RMT转发的基本架构，使用者可以对可编程数据平面转发模型进行粗粒度修改. 但粗粒度修改难以支撑对可编程数据平面开展微体系结构创新的研究.

2.4 小 结

可编程数据平面因其灵活性、适用性受到各方广泛关注. 前期相关领域研究人员通过软件模拟器或商用闭源可编程交换芯片进行应用开发或验证. 然而，基于软件的可编程数据平面无法仿真RMT微体系结构；基于ASIC芯片的可编程数据平面无法支持用户修改芯片提供接口的内部逻辑，无法充分支撑相关人员进行细粒度的微体系结构创新. 因

此，基于FPGA构建代码开源的RMT系统，即本文所提的FastRMT，用于支撑相关人员进行RTL的可编程交换微体系结构创新是必要的，能够支持数据平面可编程领域快速发展.

3 架构设计

3.1 灵活可扩展可编程数据平面设计与实现挑战

FastRMT的目标是构建满足工业界与学术界可编程数据平面微体系结构创新实验的开源平台，整体来说需要满足完备功能、模块低耦合特性、灵活可扩展控制通路、跨平台敏捷移植以及高性能等多方面要求.具体来说，FastRMT的体系结构设计应当重点解决以下五个方面的挑战：

(1)挑战一：完备精细的功能给平台线速运行带来了挑战

FastRMT旨在为用户提供一种基于RMT架构的数据平面可编程底座，具备RMT架构的核心功能，具体包括：协议无关解析与逆解析功能、通用关键字查表功能、超长指令字动作指令执行功能以及其它用于实现报文交换、转发的基本功能.同时，为了能够支持用户在生产环境进行实验测试，系统必须具有100GE场景下针对典型报文大小的线速处理能力.但完备精细的可编程报文处理功能必然需要构建复杂的处理逻辑，这对线速运行目标造成了挑战.

(2)挑战二：功能的强关联性给平台创新实践带来了挑战

允许用户便捷地对FastRMT平台进行按需调整、定制开发，进而支持可编程数据平面某些或全部功能的快速发展，是平台设计的关键所在.为实现这个目标，必然要求FastRMT各功能进行解耦，并进行模块化构建.然而为了保证平台性能，各功能间往往联系密切、耦合程度较高.如何在不降低性能的前提下，对平台功能进行细粒度解构，实现松耦合、模块化设计，无疑是个现实挑战.

(3)挑战三：包括网卡、交换机与网关在内的多样化部署场景给控制通路设计带来了挑战

FastRMT作为一种可编程数据平面底座，在支持用户实验中需要根据用户需求以网卡、交换机或多功能网关的形式进行部署，不同的部署场景对FastRMT面向控制平面所暴露的控制通路兼容性提出了极大的挑战.具体来说，FastRMT应当支持用户通过本地与远程两种方式对数据平面的逻辑表项以及各类控制寄存器进行敏捷读写操作，并且应当使得控制通路尽可能与具体系统解耦.另外，由于用户可能复用FastRMT控制通路对重构或新增模块进行表项与寄存器读写，因此控制通路应当具有良好的可扩展性.

(4)挑战四：多样安全威胁与有限资源条件给平台安全防护带来了挑战

良好的平台系统需有配套的安全体系进行防护，然而过复杂的安全机制设计又会占用过多资源，进而影响系统性能.基于这类情况，如何在保障平台性能的前提下，尽可能地对主流安全威胁进行识别、防御，构建简洁有效的安全防护机制，实现安全和性能的有效平衡，是个亟需解决的挑战之一.

(5)挑战五：差异化的FPGA给跨平台移植带来了挑战

面对当前架构差异明显的各类商用FPGA芯片与开发平台，FastRMT需具有良好的可移植性和兼容性，才能满足广大使用者的需求.然而，为了保证FastRMT的性能和功能，系统实现工作必须充分考虑硬件平台的具体情况；但过多的与具体FPGA型号结合又会导致FastRMT难以实现跨平台移植.FastRMT设计实现时，如何既充分立足底层硬件平台，同时又进行有效地抽象透明，实现系统与硬件的解耦，支持FastRMT跨平台部署，这无疑是个巨大的挑战.

3.2 FastRMT整体框架

FastRMT整体架构如图1所示.该架构采用可灵活扩展的多级流水线结构，包含安全预处理过滤器、协议无关解析单元、协议无关逆解析单元三个整体模块，而在流水线中的每个流水级中则包含关键字提取单元、动作指令匹配单元、动作执行引擎等三个流水级子模块.当报文进入协议无关解析单元后，其报文头部以及负载部分将会被放置于数据缓存中进行缓存，而报文的前N个字节将被提取进入多级流水线中进行匹配动作操作.每个模块间数据流与控制流均采用通用流数据传输协议AXI-Stream[38]依次由各模块进行处理，下面按照流水线先后顺序阐述各模块功能与设计.

3.2.1 安全预处理过滤器

区别于经典RMT架构，FastRMT在流水线入口处增加了安全预处理过滤器，为研究人员提供灵活可控的流量安全管理与过滤，支持对非场景相关报文在进入流水线之前进行过滤.该功能模块位于整个流水线的入口端，主要实现三点功能：一是在用

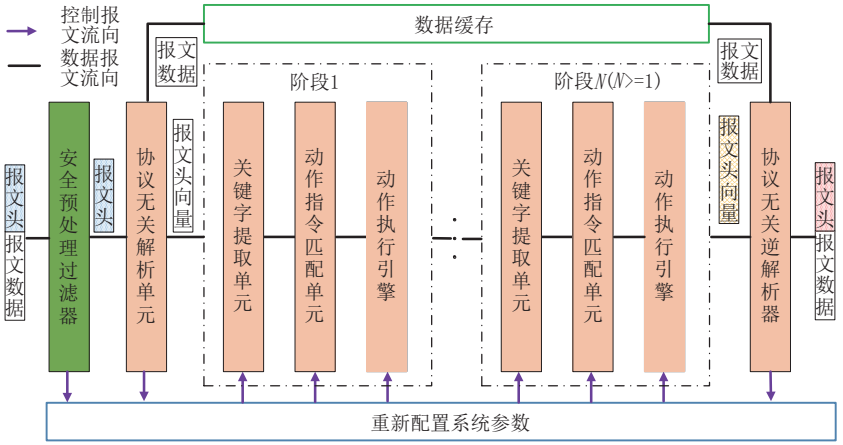


图1 FastRMT系统架构与数据流图

户针对不同数据报文进行独立测试时，将用户不关心或非法的数据报文进行过滤（如在进行UDP流量行为测试时，过滤所有TCP以及其它传输层协议的背景流量，保证测试结果的可信度）. 作为支持可编程数据平面微体系结构创新的实验平台，FastRMT在流水线前端部署过滤器的方式，使用户可以对进入流水线的报文进行自定义过滤，并在实际组网等场景中发挥着减小误差等关键作用；二是在用户通过带内控制报文对流水线功能进行重构时，安全预处理过滤器采用cookie机制对控制报文的合法性进行检查，避免来自第三方基于重放攻击等方式构造的恶意报文进入流水线，从而影响系统行为；三是数据记录及状态更新，安全预处理过滤器会记录合法报文数、整体处理字节数以及其它流水线状态数据，从而帮助研究人员通过寄存器读写等方式实时获取流水线整体状态数据.

3.2.2 协议无关解析单元

协议无关解析单元是经典RMT架构中支持协议无关报文处理的核心功能模块之一. 在FastRMT中，该模块主要功能是提取报文头部内的特定字段（即需要在流水线中进行匹配和修改操作的字段），与报文传输过程中的元数据（metadata）结合，构成报文头部向量（Packet Header Vector，PHV）. 为了实现高性能协议解析，不同于经典RMT架构中基于有限状态机（Finite State Machine，FSM）的解析模式，FastRMT采用VLAN或VxLAN标签进行匹配，并在匹配到特定表项后进行静态并行解析，从而避免解析环路，使协议无关解析单元具备线速解析能力. 具体来说，FastRMT的协议无关解析单元设计如图2所示. PHV包含报文头部关键字段、元数据等信息. 当报文进入协议无关解析单元时，报文中的流标识字段（如VLAN ID、VNI或五元组等）将被提取出来. 随后，通过ID字段将会在解析动作表中查询得到解析动作指令，从而在报文的前N字节内提取后续所需的字段，并按照解析动作指令的要求将所提取到的字段放置在PHV中的指定位置. 该过程完成后将送至后续流水级对PHV进行匹配动作操作.

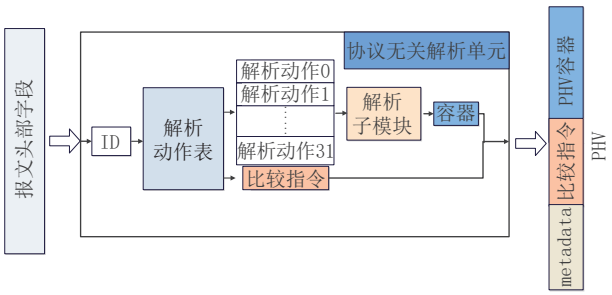


图2 协议无关解析单元图

3.2.3 关键字提取单元

由于PHV长度较长，直接由T-CAM等查表引擎进行匹配会造成较大的存储资源开销. 因此，关键字提取单元从PHV中提取仅用于匹配的字段，构造更加精简的查表关键字. 如图3所示，首先，关键字提取单元提取PHV中的ID字段，并根据ID字段匹配表中对应的提取指令. 关键字提取单元使用提取指令对PHV的对应关键字进行提取，并写入到关键字的特定位置. 需要指明的是，FastRMT中的查表关键字是定长字段，而为了解决不同网络处理逻辑所需要关键字长短不一的问题，关键字提取单元在利用PHV ID获取关键字提取指令时会同步获取对应的掩码项，通过关键字和掩码项共同作用，指明后续查表所需要使用的具体字段从而支持灵活变长的关键字匹配操作.

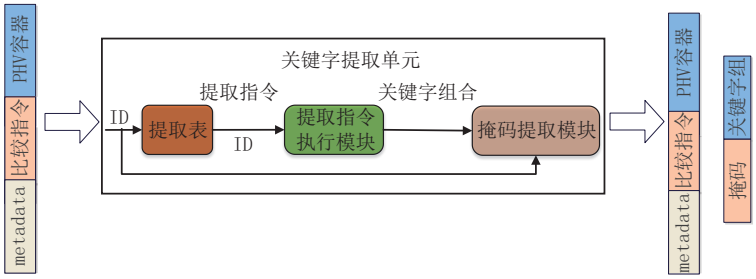


图3　关键字提取单元图

3.2.4　动作指令匹配单元

动作指令匹配单元的功能是通过带掩码的关键字匹配方式获取动作执行指令．如图4所示，在FastRMT中，动作指令匹配单元的输入为PHV、关键字组、掩码项．首先，关键字组与掩码项会配合在CAM模块中进行匹配操作，若匹配命中，则将匹配获取的表项序号（即index值）送至动作指令表．然后，动作指令表通过以该表项序号为地址进行访存操作，可获取一条包含了多条子指令的超长指令字指令，用于对PHV字段进行操作．与经典RMT架构相同，FastRMT采用超长指令字来作为动作执行指令．超长指令字可以携带多条指令，从而支持FastRMT流水线实现多指令并行并提高单位时间逻辑处理效率．

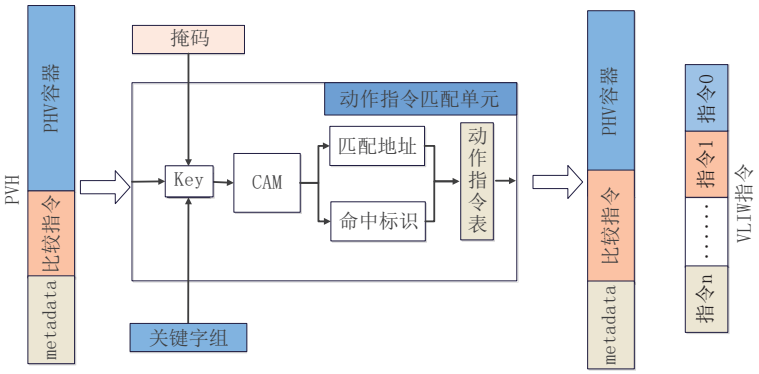


图4　动作指令匹配单元图

3.2.5　动作执行引擎

动作执行引擎采用动作指令匹配单元中获取的超长指令字对PHV字段进行修改．如图5所示，当上一级（即动作指令匹配单元）执行匹配操作完成后，将会把PHV与超长指令字指令同步输出到动作执行引擎中．该引擎中的算术逻辑单元（Arithmetic

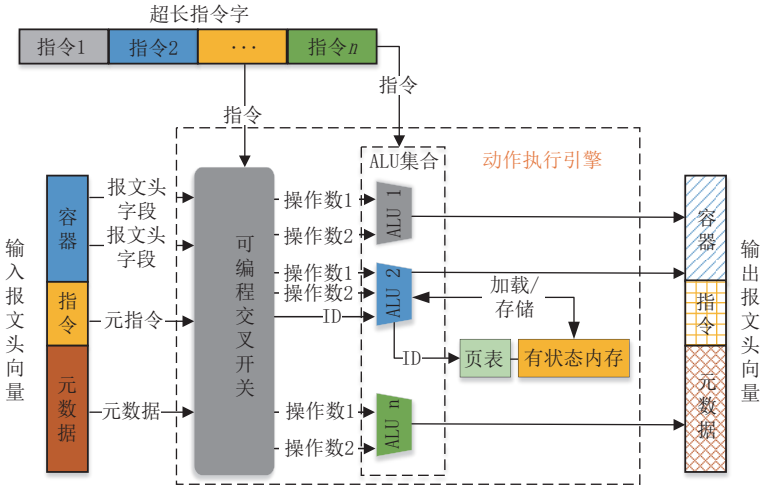


图5　动作执行引擎图

and Logic Unit，ALU)集合会将超长指令字的子指令分解到各ALU中，同时可编程交叉开关模块会读取超长指令字中的操作数提取规则，并以并发方式将各ALU所需要的操作数送至对应ALU用于进行计算操作．ALU集合中的ALU会根据超长指令字中的操作类型，对操作数进行计算，并将计算结果写回PHV中．当所有子指令执行完成后，动作执行引擎会将PHV输出至下一流水级中进行匹配动作操作．动作执行引擎中的部分ALU附有内存模块，用于存储处理过程中的状态信息，通过加载/存储等操作，实现对带状态的业务进行处理，具体为有状态内存与相应的ALU模块通过硬连线互连，当指令作用时，有状态内存和相应ALU模块配合完成计算与状态更新或读取等操作．

3.2.6 协议无关逆解析单元

协议无关逆解析单元执行与协议无关解析单元相反的操作，即将处理好的PHV进行逆解析，然后合并至原始报文中．首先，协议无关逆解析器根据PHV ID信息，在逆解析表中进行匹配，获得对应的逆解析指令．逆解析指令对PHV中的报文头关键字段进行提取重组，然后写至报文体．需要注意的是，逆解析指令并不提取PHV中的元数据．元数据将随流水线流出，由具体实现的平台决定如何处理这些元数据．

3.2.7 安全可扩展数据通路

作为可编程数据平面微体系结构实验平台，FastRMT需要为用户提供一种支持按需扩展的数据平面表项与规则配置接口，支持用户在控制平面通过包括PCIe或以太网等通路在内的方式对数据平面控制寄存器与各类表项进行灵活配置管理．因此，FastRMT设计采用菊链方式(Daisy Chain)方式以带内控制报文对流水线各模块寄存器与表项进行远程或本地读写操作．具体来说，读写指令被封装在带有特定标识的报文负载中，支持单独读写、批量读写、安全认证等功能，并支持用户动态扩展读写指令类别与读写范围．FastRMT流水线中每个模块与模块内各类表资源采用二级编址方式统一编号，包含读写指令的控制报文按顺序逐个通过各功能模块，当读写指令的编号索引与所处的模块或表资源编号匹配时，由该模块对读写指令进行译码并执行．其具体设计在第四节进行详细阐述．

3.3 开放性设计

作为面向学术与工业界研究人员提供的微体系结构创新平台，FastRMT不仅需要性能功能上高效完备，还需具有良好的开放性，促进使用者对平台的快速理解和定制修改．FastRMT的开放性包含可用性和易用性两部分．

3.3.1 可用性

为保证FastRMT平台的可用性，本文在FastRMT开源获取和风格规范层面进行针对性设计．

(1)开源共享

FastRMT平台的核心代码发布在GitHub平台，面向可编程数据平面研究领域用户进行开源共享．用户可通过Issue和Pull-Request机制基于FastRMT平台进行实验平台的迭代优化，完善配套生态，为学术界、工业界相关人员提供良好交互机制．

(2)风格规范

FastRMT平台严格遵守工程实现规范，代码风格良好，设计、说明等配套文档齐备．使用者可借助文档，实现快速深度理解FastRMT体系架构和技术细节．

3.3.2 易用性

为便于使用者对FastRMT进行定制修改，加快实现微体系结构创新工作．FastRMT平台将系统通用参数进行抽象凝练，通过宏定义方法进行统一管理；并利用标准化接口协议(AXI4-Stream)，对模块间通信进行规范，具体如下．

(1)宏定义

FastRMT平台对需要根据使用者需求进行动态变更的硬件参数、软件参数，通过宏定义机制编排设定．当使用者需要对FastRMT部署环境、性能参数进行调整时，通过宏定义变更实现，有效减少修改底层代码导致的隐患．

(2)标准化接口

AXI4-Stream是业界主流的标准协议接口，常用于芯片内模块间的高速数据流传输．FastRMT严格采用AXI4-Stream进行模块间通信，不再自行定义接口，使用者无需另外学习接口规则便可对模块进行调整修改．

4 系统实现与优化

本节对FastRMT各模块的功能性能以及跨平台部署机制等方面进行详细介绍．FastRMT流水线包含共约11 000行verilog逻辑代码，用于支持RMT架构的核心功能，并提供高可扩展与敏捷重构的特性．系统实现与数据流转如图6所示．

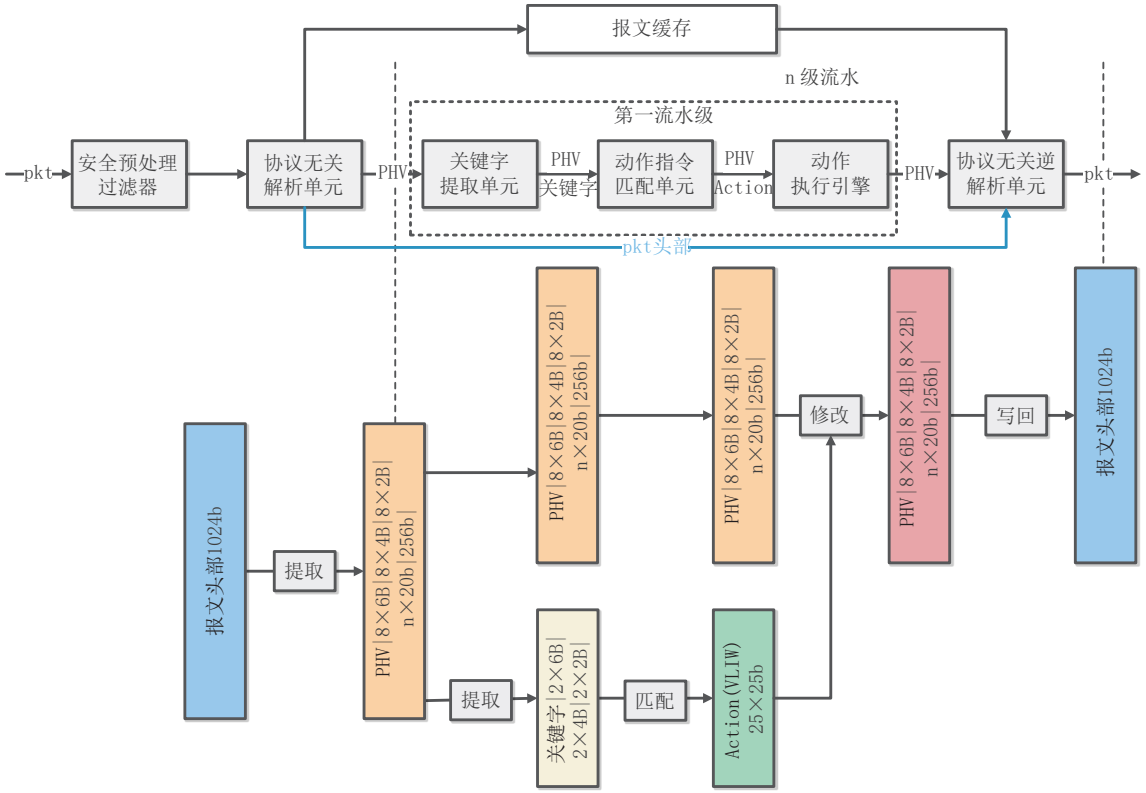


图6　FastRMT 系统实现与数据流图

4.1　功能模块

4.1.1　过滤器

该模块位于流水线的入口，负责控制报文的安全认证与非法报文的过滤. 过滤器内设置两组由控制平面读写且可以通过 AXI-L 协议[38]的寄存器 *ctrl_token* 与 *drop_flag*. 其中，*ctrl_token* 由当前硬件时间戳进行初始化，其值受到本地计数器和控制报文双重影响：若预配置时间 T 内未接收到控制报文，则 *ctrl_token* 值进行自增；而若时间 T 内接收到控制报文，则 *ctrl_token* 值根据当前时间戳采用哈希函数进行更新. FastRMT 通过该方法实现对控制报文重放攻击的防御. 而 *drop_flag* 则由过滤器内部协议相关的有限状态机进行控制. 该寄存器允许用户通过修改过滤器内部的有限状态机实现在特定时间对特定报文的过滤. 该模块在 Menshen[17]中用于支持多租户运行时重编程阶段的流量隔离机制，也可用于过滤或提取特定协议类型的报文.

4.1.2　协议无关解析单元与逆解析单元

FastRMT 的解析单元提供协议无关的 PHV 提取与解析操作，支持根据 VLAN ID 或 VNI 面向不同租户虚拟网络进行提取和解析. 对于任何报文，可编程解析器通过 VLAN ID 或 VNI 作为索引，查询解析器内部基于 BRAM 的协议解析表，然后得到一条解析表项. 解析表的表项格式如图 7 所示，实现并行提取 PHV 的填充字段. 每条解析表项由 11 条子指令构成，前 10 个子指令各包含 16 位，即 2 字节. 其中，[12:6]共 7 位表示提取字段相较于报文头起始偏移，[5:4]共 2 位表示字段长度类型，2'b00 代表 2 字节字段，2'b01 代表 4 字节字段，2'b10 代表 6 字节字段，2'b11 代表 8 字节字段. [3:1]共 3 位表示字段放置于 PHV 中的索引，这 3 位与 2 位的字段长度类型可以确定一个字段在 PHV 的具体位置，如 3'b001 与 2'b01 可确定当前提取 4 字节字段，并将其放置于 PHV 的第 1 个 4 字节位置(slot). 16 位中的最

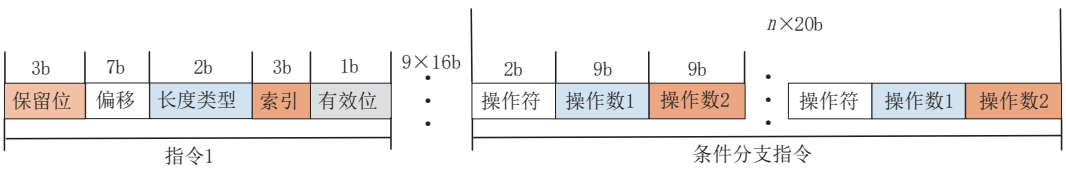


图7　解析表的表项格式图

低位为有效位，用于标识当前子指令是否有效．第11个子指令为$n\times20$b的选择指令，其中n为流水线中匹配动作流水级的级数，每级指令长度为20b，其中高2位为操作符，2b'00为＞，2b'01为＞＝，2b'10为＝＝，紧接的18位分别为操作符的两个操作数在PHV容器中的索引或立即数的值，其索引方式详见文献[39]．解析单元从报文头部的1024比特中提取需要的字段，填充至PHV．PHV与同步产生的metadata（含时间戳、输入端口号、转发端口等信息）一起进入到流水线中进行处理．另外，为了支持P4语言中的if-else条件语句，PHV同时插入了来自解析表项中的if-else条件语句指令．PHV格式如图8所示，共包含868位提取字段、$n\times20$位与额外256位的metadata字段．其中，metadata字段的低128位复用NetFPGA项目的metadata定义，而高128位则支持用户对其进行自定义．而逆解析单元操作过程与解析单元相反，用于在报文输出流水线时将PHV与报文进行拼装，因此这里不再赘述．

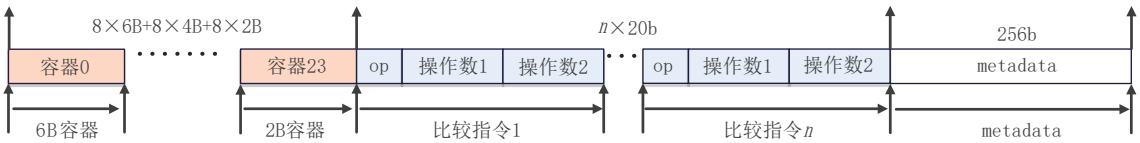


图8 PHV字段格式示意图

4.1.3 关键字提取单元

如第三节所述，关键字提取单元用于从PHV中提取匹配关键字，支持根据VLAN ID或VNI面向不同租户虚拟网络的关键字提取．该模块包含一个基于BRAM单元的关键字提取规则查找表．查找表的表项索引为随PHV同步传输的VLAN ID或VNI字段，而匹配到的表项位宽18位，可以分为6项，每项3位的提取子指令．从高至低位分别为面向2字节、4字节与6字节的提取规则．因此，3位子指令唯一地确定了用于构造匹配关键字的PHV字段．而为了支持带掩码匹配，在查找关键字提取规则时也会同步在掩码表中通过同样方式获取掩码表项，掩码表项位宽为2B×2+4B×2+6B×2共计24B，根据用户需要对生成的关键字进行掩码操作．该模块最后向动作指令匹配单元输出PHV、所匹配关键字以及其对应的掩码用于进行匹配动作．

4.1.4 动作指令匹配单元

该单元是FastRMT中用于实现自定义匹配的核心模块，其核心由一个基于CAM的匹配引擎与一个基于BRAM的包含超长指令字指令在内的动作表构成．由关键字提取单元生成的查表关键字以及掩码到达后，首先由基于FPGA片上SRAM搭建的16×24B项CAM单元进行匹配查表操作①．当关键字匹配后，将输出当前规则所对应动作指令在BRAM表项（存储有超长指令字指令）索引．该索引将会从BRAM中读取一条PHV的处理指令，该指令的位宽为625b，由25条位宽为25b的子指令构成．为了支持转发操作、网内计算、网内测量等多种类型网内应用，动作指令匹配单元的超长指令字支持特定端口转发、丢弃报文、立即数/寄存器整型算数运算操作、立即数/寄存器逻辑运算操作以及面向片上BRAM的访存操作．

具体而言，超长指令字包含的子指令模式共有如图10所示的5种．其中25b子指令的高4位为指令操作码，目前共支持三类共12条指令，其对应功能与描述详见表1，三类子指令的指令格式如图9所示．其中，第一类为双容器操作数指令，该类指令从PHV中提取两个容器的操作数并进行二元运算操作．该类指令的指令操作码后紧邻10位分别为第1与第2个操作数在PHV中的地址，其中高2位为容器类型（00、01、10分别代表2B、4B与6B），低3位为

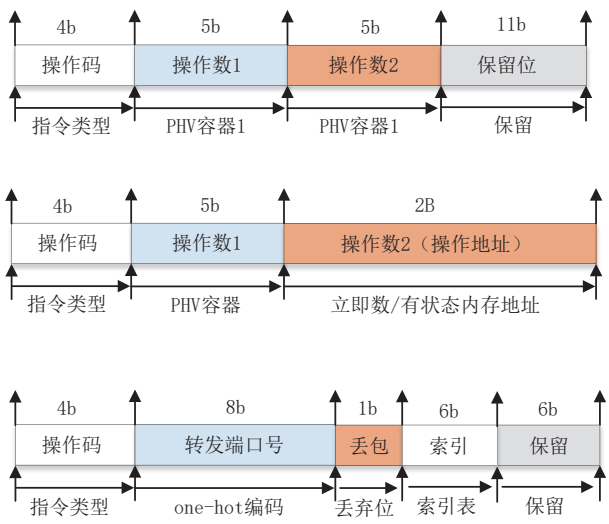


图9 FastRMT三类子指令格式

① 需注意当前FastRMT源码版本暂不支持哈希查表与最长前缀匹配查表，后续版本将进行增量设计与部署．

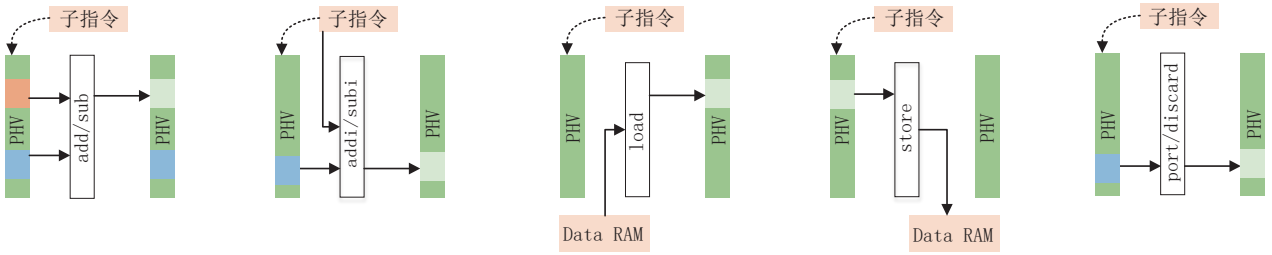


图10　FastRMT子指令操作模式

表1　RMT流水线指令功能表

指令码	指令	功能描述
4b'0001/0010	add/sub	将PHV容器中两个操作数相加/减，并将结果覆盖第一个操作.
4b'1001/1010	addi/subi	PHV容器值与立即数相加/减，并将结果覆盖第一个操作数.
4b'1110/1111	and/or	PHV容器值与立即数相与(或)，并将结果覆盖第一个操作数.
4b'1011	load	将立即数对应地址的值从内存读出，并覆盖第一个操作数.
4b'1000	store	将指令携带的PHV容器对应值写入立即数对应内存地址.
4b'1100	port	修改发送端口到操作数对应值(bitmap).
4b'1101	discard	丢弃当前报文.
4b'1110	set	将PHV对应容器的值修改为立即数值.

操作数所在的特定容器索引号. 而指令的低11位则为保留位，用于后续对不同功能选项进行扩展. 第二类为单容器操作数指令，该类指令依据紧邻操作码的5位地址从PHV中提取一个特定容器作为第一个操作数，将指令低16位作为立即数或存取行为的地址字段. 第三类网络专用指令，其紧邻操作码由高到低分别为8位基于one-hot的转发端口标识，1位报文丢弃标识，6位指定特定流水级匹配表标识以及6位保留位，用于其它网络处理相关功能的扩展.

4.1.5　动作执行引擎

如图11所示，该引擎在FastRMT中的主要功能为采用单周期并发方式执行所匹配到的指令. 该引擎同步接收来自动作指令匹配单元的PHV(附带metadata信息)与超长指字指令，并根据指令完成对PHV的修改，然后将新的PHV字段进行输出. 具体来看，FastRMT的指令执行引擎包含一组可编程交叉开关(Crossbar)结构与25组用于指令执行的综合运算单元(ALU). 其中PHV通过Crossbar与不同的ALU的输入端进行连接，并在25组子指令的控制下，解析出用于ALU计算的操作数. 由于PHV中从高到低分别存放6B、4B与2B的报文头字段，因此Crossbar按照相同排列从输入端的PHV中根据每条子指令的操作数索引提取出相应的字段作为25组指令的操作数，并将其输出到ALU部分. 另一端，25组ALU通过硬连线(hard-wire)与Crossbar的

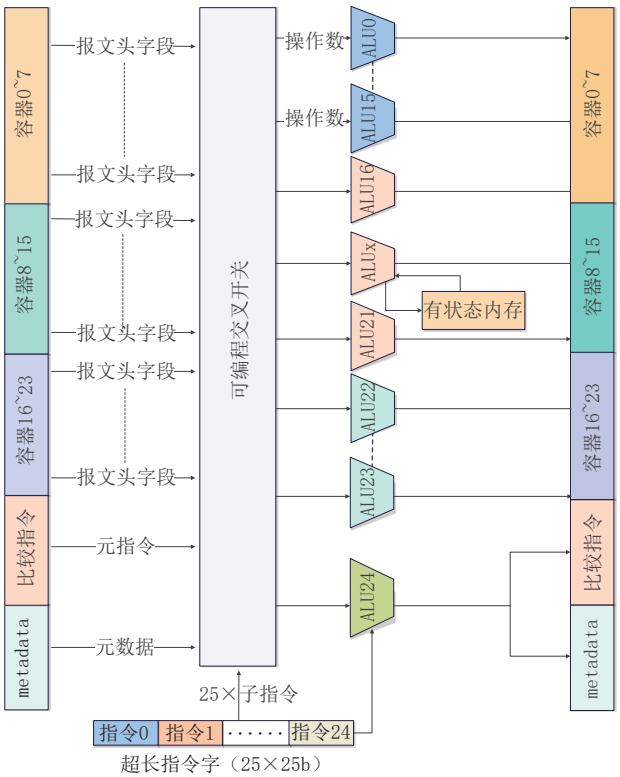


图11　动作执行引擎系统实现图

输出端相连，该机制恰好支持从Crossbar的输出端读出每组ALU对应的操作数，并在子指令的指引下进行译码和执行操作. 动作执行引擎从高位到低位的前24组ALU支持赋值、整型加法、减法、乘法、逻辑运算以及load与store指令，用于支持常见网内计算操作. 而第25组ALU用于对PHV所携带的

metadata进行操作，从而支持丢包、广播以及向特定端口转发等常见网络转发行为. 所有ALU接收操作数后进行指令执行的延迟时间相同，保证了ALU并行执行指令后同步在ALU输出端给出执行结果，并将其组合形成更新后的PHV.

4.2 安全可扩展控制通路

为了灵活、安全地修改流水线中各类表项，FastRMT采用带内控制方式，通过UDP报文对流水线表项进行重配置. 为了有效区分数据报文与控制报文，控制报文目的端口号统一采用0xf1f2，并采用cookie验证机制进行安全访问控制，以防御控制报文重放攻击等恶意行为.

(1)重配置机制

具体而言，FastRMT的带内控制报文格式如图12所示. 其使用8b标记负载所携带表项的目的流水级以及对应功能模块的ID号，采用4b标记命令类型(如读操作或写操作，以及批处理等)，另外还有4b作为保留字段用于后续扩展. 控制报文携带15B的填充字段用于在256b或512b数据位宽情况下进行数据对齐. 最后，负载字段用于携带配置表项的内容，该部分为变长字段，从而支持对FastRMT流水线中不同位宽表项进行灵活配置.

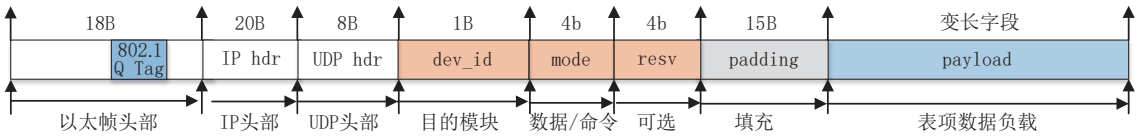


图12 FastRMT安全可扩展控制通路报文

(2)安全防御机制

为有效防止重放攻击，带内控制报文填充字段的前4个字节，用于携带防止重放攻击的cookie字段. 该字段由记录控制通路配置次数的序列号经哈希函数生成，仅当发送的控制报文中的cookie字段与数据平面本身维护的cookie字段匹配时才能正常对数据平面进行配置. 考虑控制报文丢失或者被恶意节点截取等特殊情况，本文在流水线的末端，增加了控制报文反馈器，构建控制报文信息流转的闭环系统. 当控制报文反馈器收到控制报文时，会向报文发送方反馈确认信号，实现对报文信息的确认，有效应对控制报文丢失、截取等特殊情况.

4.3 跨平台部署

为了支持用户基于不同网络平台开展RMT架构的研究与实践，本工作分别采用两种数据位宽以及时序约束将FastRMT整合进Corundum与NetFPGA平台，从而支持在网卡和交换机两种平台进行实验. 具体来看. FastRMT具有512b与256b两种位宽. 其中，512b位宽与Corundum核心代码整合，支持最高在100 Gbps端口速率下的网卡平台进行基于P4的可编程报文处理，其核心逻辑工作频率可达250 MHz；而256b版本则与NetFPGA SUME平台进行整合，支持4路10 Gbps速率的可编程交换功能，其核心逻辑工作频率最高为200 MHz. 目前，FastRMT的开源代码支持用户直接综合与布局布线，并部署到基于Corundum的FPGA网卡或NetFPGA SUME交换平台.

4.4 语言与编译器适配

作为高度可编程的高速数据平面流水线，FastRMT包含的各类异构CAM表与RAM表的规模较大、种类较多. 因此，用户通过手动编码操作控制报文从而对FastRMT进行编程不仅易错，而且复杂度较高. 为了便于用户使用FastRMT进行各类网络架构或应用的敏捷实验，本工作同时在开源P4编译器P4C[40]基础上对编译后端逻辑进行修改，从而兼容FastRMT流水线中的各类异构表项. 用户可通过高层次网络行为描述语言P4对FastRMT的报文处理逻辑进行细粒度控制. 具体来说，FastRMT编译器保留了P4C中P4基本语法前端静态解析逻辑并生成中间表达形式(Intermediate Representation，IR)，并将IR转义为与FastRMT相适配的各类异构表的静态表项并写入文件. 最后，当需要对FastRMT进行配置时，则通过FastRMT配置代理读取静态表项内容，并构造控制报文读取配置文件中静态表项，从而将对应配置下发至硬件流水线①.

5 实验验证

本节对FastRMT原型系统的资源、性能表现与功能进行实验验证. 本文使用Verilog HDL实现了RTL级FastRMT，并基于Xilinx Ultrascale+ VCU118与VX690T FPGA分别构建了网卡与交换机原型系

① FastRMT编译器：git@github. com：Winters123/rmt-compiler. git

统，从而支持功能与性能验证.

5.1　资源评估

首先对FastRMT的资源占用情况进行评估. 本文将FastRMT基于Ultrascale+ VCU118与开源网卡平台Corundum[25]进行整合，并基于VX690T与开源交换平台NetFPGA SUME[21]进行整合，具体如图13(a)(b)所示. 其中，网卡版本支持QSFP-28双端口，每个端口速率最高为100 Gbps，交换版本支持SFP+四端口，每个端口速率最高为10 Gbps. 两版本的FPGA总体资源开销表如表2所示. 整体来看，采用五级Match-Action结构的FastRMT无论是网卡版本还是交换版本所消耗的LUT与BRAM资源均有限，并预留了充分片上资源用于使用者增加其它功能逻辑. 网卡版本数据位宽为512位，时钟频率为250 MHz，最大数据处理速率可达128 Gbps；而交换版本数据位宽为256位，时钟频率为200 MHz，最大数据处理速率可达51.2 Gbps.

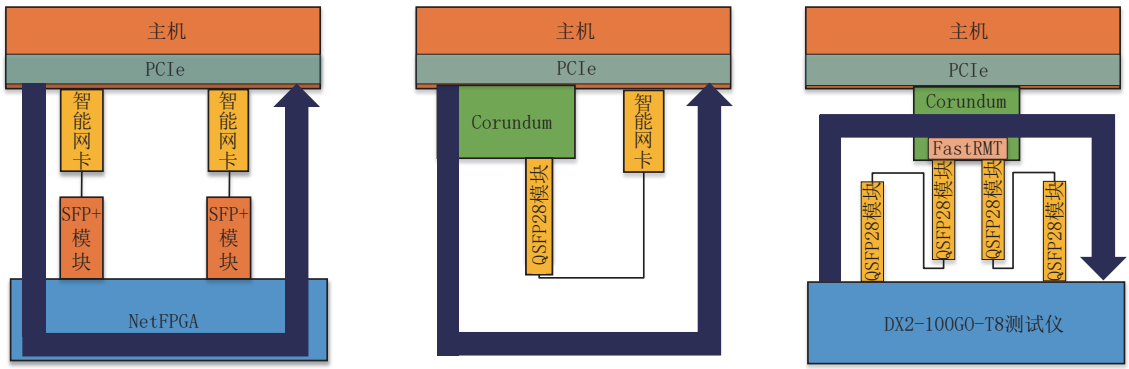


图13　FastRMT性能评估实验设置

表2　FastRMT原型系统FPGA资源消耗表

	LUT	FF	BRAM	DSP
Corundum网卡	61902 (5.24%)	77688 (3.29%)	349 (16.16%)	0
FastRMT网卡	235509 (13.63%)	230031 (6.66%)	315.5 (11.74%)	0
NetFPGA交换	42430 (9.79%)	69619 (8.04%)	245.50 (16.70%)	0
FastRMT交换	203956 (47.08%)	215706 (24.90%)	538 (36.60%)	0

计量单位：BRAM块数、LUT个数

5.2　性能评估

本节对FastRMT的性能表现进行评估，具体包含吞吐量性能和延迟性能两个方面. 实验设置如图13(c)所示，本文将FastRMT嵌入至Corundum开源网卡的数据通路中，并通过思博伦网络测试仪DX2-100GO-T8由QSFP28 100GE端口进行直连构成闭环回路，对不同网络报文的处理性能进行实时记录. 网络测试仪可以按需产生不同速率、大小的测试流量，用以支撑FastRMT性能评估. 考虑到吞吐量性能、延迟性能与FastRMT内部的报文处理逻辑无关，为便于测试，本文设置FastRMT内部报文处理逻辑为默认转发规则. 经20轮独立性能测试，其吞吐率结果如图14所示.

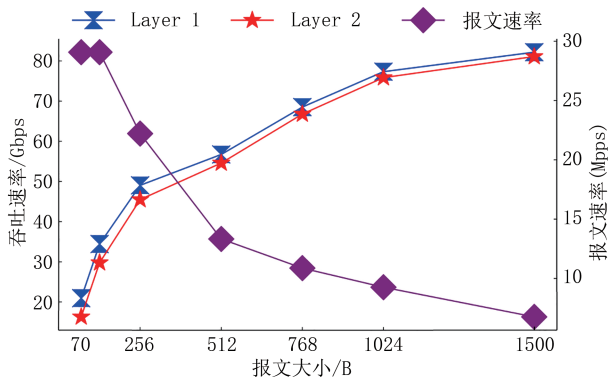


图14　FastRMT吞吐性能评估

测试结果显示，随着报文逐渐增大，流水线的处理性能逐步增强，当报文大小达到1500 B时，流水线的处理性能接近线速. 需要指出的是，这是在未对流水线未做任何深度优化的情况下的表现，理论上FastRMT可以通过增加解析器/逆解析器数量或优化其内部微体系结构设计，从而实现并行处理，达到针对64 B～256 B小报文的线速处理. 而从报文处理延迟角度看，基于Corundum的FastRMT整体延迟性能测试结果如图15所示，对于70 B至1500 B的报文，其延迟仅为1 μs至1.25 μs之间，对Corundum

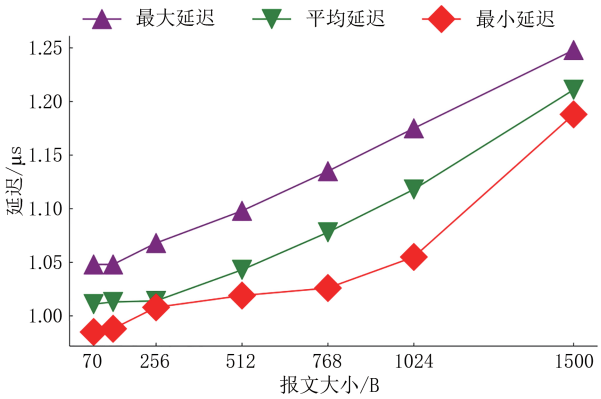


图15 FastRMT延迟性能评估

的报文处理延迟并未产生显著影响.

为了验证FastRMT在部署新型在网计算等功能时通过控制通路配置流水线的性能表现,本文挑选四种典型P4示例应用(P4 Calc[41], NetChain[42], Firewall[41], NetCache[43])并通过FastRMT的安全可扩展控制通路对流水线进行重配置并使其生效.其总体配置表项数量与的对应的平均延时如表3所示.从性能测试结果看,FastRMT能够较好支撑在100GE真实网络系统中,进行可编程数据平面相关技术测试.

表3 FastRMT流水线重配置延迟表

	表项数量	配置延时(ms)
P4 Calc	9	0.55
NetChain	11	0.57
Firewall	32	0.61
NetCache	141	0.70

5.3 应用用例评估

为进一步验证FastRMT系统的可用性与支持可编程数据平面微体系结构创新的效率优势,本文选取基于P4语言编写的多种典型应用进行测试,具体包含防火墙、计算器、负载均衡、源路由、NetCache、NetChain共计六种应用.经测试表明,所有应用用例在功能正确运行的基础上均可与基础FastRMT转发测试达到相同性能,即接近100 Gbps的线速处理能力.这验证了FastRMT将有能力在数据中心网络、边缘计算网络等高速网络场景下支撑可编程数据平面相关在网计算等技术的实网测试.

FastRMT通过标准化可编程数据平面流水线各模块间接口,提供统一的可编程解析器、动作执行引擎等模块模板与支持安全可扩展的控制通路等方式极大简化了基于可编程数据平面的微体系结构创新设计与FPGA验证.本节分别通过基于RMT的在网推理引擎与网络的可编程解析器设计两个应用用例验证FastRMT对基于可编程数据平面微体系结构创新的支持效果.

5.3.1 在网推理引擎用例

从在网AI推理需求来看,随着网络设备的日益增加,核心网的路由表日益膨胀,路由项数量迅速增加到无法有效处理的程度,从而导致网络性能下降、资源浪费、路由不稳定等问题,即“路由爆炸”.传统的基于查表的路由方法需要大量的路由表项,而采用机器学习推理替代路由查表将节省大量的存储空间,原因是一个机器学习模型中的权重参数远远小于路由表的大小.因此可在数据平面中设计实现机器学习推理代替路由查表过程.

本节通过在FastRMT基础上扩展支持在网AI推理引擎验证FastRMT对基于可编程数据平面微体系结构创新的支持效果.该用例在支持所有P4可编程数据平面能力的基础上,融合了单指令多数据流(Single Instruction Multiple Data, SIMD)与多指令多数据流(Multiple Instruction Multiple Data, MIMD)两种计算模式,形成匹配动作单元与AI计算单元并行计算架构,使数据平面不仅能够支持协议无关网络分组处理,还能对承载AI推理与训练的分组数据实现在网计算.为包括在网AI推理与训练提供高性能处理平台,本节在该架构中部署多层感知机(Multilayer Perceptron, MLP)模型实现基于AI的报文分类,替代传统的基于TCAM查表的路由方法,解决核心路由表路由爆炸问题.如图16所示,其关键增量为:在FastRMT每级动作执行引擎旁增加一级并行的AI计算单元,在该单元内部包括一层矩阵向量乘法计算核、一组模型偏置值计算核与一组激活函数计算核.

基于FastRMT架构进行IPv6路由推理的具体设计如下:(1)PHV预留前4个8 B容器进行矩阵向量运算,解析器决定是否需要矩阵运算,若需要则将PHV标识位flag字段置1;(2)AI计算单元对传入的PHV判断标志位,若需要进行矩阵运算,则提取前4个8 B容器的数据,构成1×32的向量;(3)对于数据长度达不到32的向量,后面补0即可,不影响矩阵向量乘法结果;(4)每级32个乘加器,每个乘加器中放置1×32的权重列向量,用不到的全部置0,不足32的也在后面补0.即每个AI运算单元实际执行[1,32]×[32,32]的矩阵向量乘,通过置0可以实现不同大小的矩阵向量乘,以此进行升维降维;(5)针对该

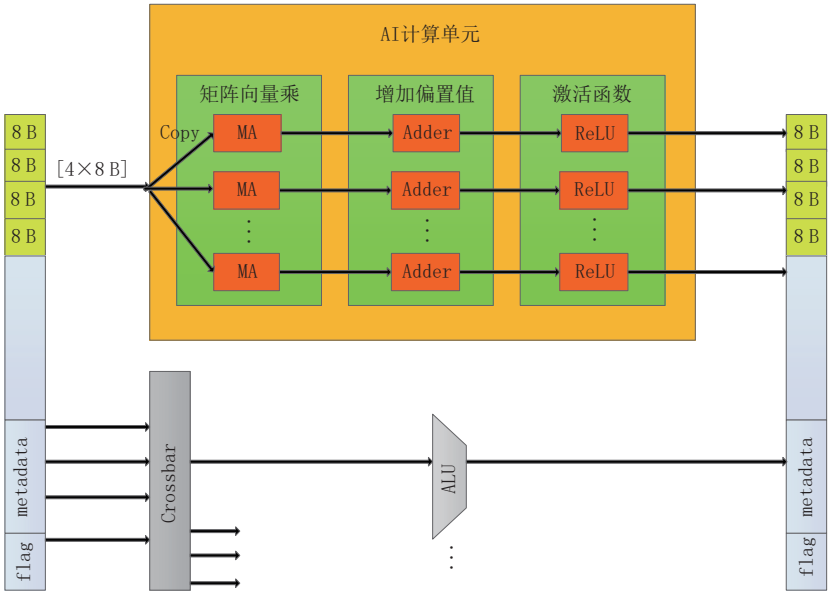


图16　基于FastRMT的在网AI推理引擎结构示意图

场景实现四层神经元，权重矩阵大小分别为[16，32]、[32，16]、[16，8]、[8，1]；(6)每个阶段输出一个32元素的向量，与输入的PHV位置相对应写回．最后一层神经元得到1 B的结果．逆解析器将输出接口号写入metadata；(7)AI计算单元参数由控制报文配置．一个阶段共32×32＋32＝1056 B参数．

在NetFPGA SUME搭建上述架构FPGA原型系统，采用FastRMT在10天左右即可完成系统开发调试，显著缩短原型系统搭建与验证时间．通过复用FastRMT的相关模块，所需增加verilog代码行数仅为1100行．经测试，该架构的资源消耗如表4所示．在10 Gbps速率下，每个报文处理延迟为1.5 μs，在实验室受控组网场景下，经过5K数据集训练后，随机构造IPv6路由表的路由推理正确率在98.3%以上．随着路由表的扩大，存储开销也不断增加．而神经网络中的权重参数所占用的存储空间固定，并不会随着路由表项的增加而增加，相比于不同大小的路由表其节省的存储空间比例会随之增加．基于FastRMT进行AI推理的路由准确率与存储优化情况如表5所示．该用例证明FastRMT对基于可编程数据平面微体系结构在面向AI在网计算方面创新场景的支持效果较好．

表4　基于FastRMT的在网AI推理引擎资源消耗表

	LUT	BRAM
NetFPGA交换	61902 (5.24%)	77688 (3.29%)
FastRMT交换	235509 (13.63%)	235509 (13.63%)
AI推理引擎	42430 (9.79%)	69619 (8.04%)

计量单位：BRAM块数、LUT个数

表5　路由准确率与存储优化情况表

训练集大小	正确报文数量	路由准确率	存储空间优化比例
5K	4916	98.3%	98.7%
50K	49 357	98.7%	99.87%
500K	496 488	99.3%	99.99%

5.3.2　解析单元/逆解析单元性能优化用例

RMT架构在处理性能方面面临的主要瓶颈在于协议无关解析单元与逆解析单元．近年来部分研究尝试通过优化解析单元与逆解析单元的微体系结构设计提升可编程数据平面的整体处理性能．本节采用深度流水线(deep pipeline)与交叉开关稀疏化技术对协议无关解析单元与逆解析单元进行性能优化，并通过FastRMT对优化后效果进行验证，从而证明FastRMT在支持可编程数据平面微体系结构创新的敏捷实验验证的优势．

经典RMT架构中的协议无关解析单元由于采用了基于RAM与CAM单元组合的FSM实现协议字段提取与PHV填充，因此其性能受限于FSM的迭代轮次．而对于逆解析单元来说，其性能瓶颈在于将PHV中各协议字段随机写回报文的固定位宽的首部字段．由于需写回字段可能来自报文首部任意字节，最高效方法即为采用Crossbar或CLOS网络进行随机字段写回．然而，上述方法难以在较高时

钟频率实现FPGA片上布局布线，因此一般也采用FSM进行串行的PHV字段写回，其代价就是较低的逆解析处理性能．本节对协议无关解析单元与逆解析单元优化中对逆解析单元的报文首部字段写回功能微体系结构进行优化，从而RMT实现更高的报文处理性能．具体而言，该优化方法增加多级解析或逆解析流水线替换解析单元与逆解析单元的FSM，通过较小的逻辑资源开销换取更高的吞吐率，从而提升解析与逆解析性能．另一方面，为了降低布局布线的难度同时限制流水级的数量，在每个流水级中采用一组非全连接的稀疏化Crossbar将6组PHV字段进行并发写回，从而进一步提升其性能．

由于采用了标准AXI4-Stream总线接口格式与基于宏定义的位宽参数选择，优化的解析单元与逆解析单元开发仅涉及更换原解析单元与逆解析单元，并仅需增加verilog代码800行．本节将优化后的解析单元与逆解析单元与FastRMT进行整合，并通过Xilinx AlveoU250经DX2-100GO-T8测试仪进行吞吐测试，其测试结果如图17所示．由测试结果发现，优化后的FastRMT在报文大小≥256 B即可达到100 Gbps线速处理能力．该用例验证了FastRMT在可编程微体系结构功能优化实验验证方面的支持效果．

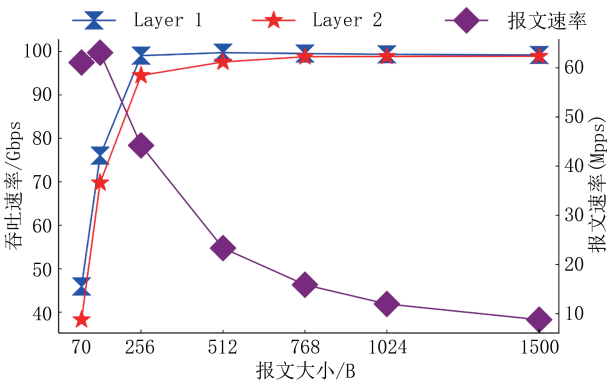


图17 FastRMT优化后吞吐性能评估

6 总结与展望

本文介绍了FastRMT，一个面向网络微体系结构创新的可编程高速数据平面原型系统，利用verilog语言进行实现，并基于Corundum、NetFPGA平台对FastRMT的功能和性能进行验证，实验发现该系统功能完备，能够支持面向数据平面新型微体系结构的创新设计，并且在能够满足100 Gbps线速处理需求．最后本文对FastRMT进行开源，从而支持学术界和工业界开展可编程数据平面微体系结构、在网计算等方面的相关研究与创新．

下一阶段，将在现有FastRMT的基础上，结合特定领域场景需要，对FastRMT的模块进行定制优化和深度修改，以期更好地满足学术界和工业界的需要．具体包含以下三个方面：(1)将引入基于PTP的精确时间同步机制，并结合PIFO[44]或PIEO[45]等可编程调度器，从而支持对时间敏感网络等新型领域网络技术开展敏捷试验；(2)将扩展向量或矩阵运算指令，从而支持对可编程数据平面进行机器学习等新型领域网络技术开展敏捷试验；(3)除目前带通配匹配引擎以外，引入哈希查表与最长前缀匹配功能模块从而支持更加丰富的试验场景．

作者贡献说明 杨翔瑞、曾令斌为共同第一作者．

参 考 文 献

[1] Hancock D, Van der Merwe J. Hyper4: Using p4 to virtualize the programmable data plane//Proceedings of the 12th International on Conference on Emerging Networking Experiments and Technologies. New York, USA, 2016: 35-49

[2] Bosshart P, Gibb G, Kim H S, et al. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 99-110

[3] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74

[4] Ye J L, Chen C, Chu Y H. A weighted ECMP load balancing scheme for data centers using P4 switches//Proceedings of the IEEE 7th International Conference on Cloud Networking (CloudNet). Tokyo, Japan, 2018: 1-9

[5] Datta R, Choi S, Chowdhary A, et al. P4guard: Designing p4 based firewall//Proceedings of the MILCOM 2018 IEEE Military Communications Conference (MILCOM). New York, USA, 2018: 1-6

[6] Fei J, Ho C Y, Sahu A N, et al. Efficient sparse collective communication and its application to accelerate distributed deep learning//Proceedings of the 2021 ACM SIGCOMM Conference. Boston, USA, 2021: 676-691

[7] Wang S, Gao K, Qian K, et al. Predictable vFabric on informative data plane//Proceedings of the ACM SIGCOMM 2022 Conference. Amsterdam, Netherlands, 2022: 615-632

[8] Bosshart P, Daly D, Gibb G, et al. P4: Programming protocol-independent packet processors. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87-95

[9] Gebara N, Lerner A, Yang M, et al. Challenging the stateless quo of programmable switches//Proceedings of the 19th ACM

Workshop on Hot Topics in Networks. New York, USA, 2020: 153-159

[10] Yu L, Sonchack J, Liu V. Mantis: Reactive programmable switches//Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. Salt Lake City, USA, 2020: 296-309

[11] Ibanez S, Antichi G, Brebner G, et al. Event-driven packet processing//Proceedings of the 18th ACM Workshop on Hot Topics in Networks. New Jersey, USA, 2019: 133-140

[12] Krude J, Hofmann J, Eichholz M, et al. Online reprogrammable multi tenant switches//Proceedings of the 1st ACM CoNEXT Workshop on Emerging in-Network Computing Paradigms. Orlando, USA, 2019: 1-8

[13] Zheng P, Benson T, Hu C. P4visor: Lightweight virtualization and composition primitives for building and testing modular programs//Proceedings of the 14th International Conference on Emerging Networking Experiments and Technologies. Heraklion, Greece, 2018: 98-111

[14] Yuan Y, Alama O, Fei J, et al. Unlocking the power of inline Floating-Point operations on programmable switches//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). Washington, USA, 2022: 683-700

[15] Feng Y, Chen Z, Song H, et al. Enabling in-situ programmability in network data plane: From architecture to language//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). Washington, USA, 2022: 635-649

[16] Chole S, Fingerhut A, Ma S, et al. dRMT: Disaggregated programmable switching//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Los Angeles, USA, 2017: 1-14

[17] Wang T, Yang X, Antichi G, et al. Isolation mechanisms for high-speed packet-processing pipelines//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). Washington, USA, 2022: 1289-1305

[18] Github Repository of BMv2. https://github.com/p4lang/behavioral-model

[19] Intel Tofino Series Switches. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series.html

[20] Yang X, Sun Z, Li J, et al. FAST: Enabling fast software/hardware prototype for network experimentation//Proceedings of the International Symposium on Quality of Service. Salt Lake City, USA, 2019: 1-10

[21] Zilberman N, Audzevich Y, Covington G A, et al. NetFPGA SUME: Toward 100 Gbps as research commodity. IEEE Micro, 2014, 34(5): 32-41

[22] Ibanez S, Brebner G, McKeown N, et al. The p4-> netfpga workflow for line-rate packet processing//Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. New York, USA, 2019: 1-9

[23] Wang H, Soulé R, Dang H T, et al. P4fpga: A rapid prototype framework for p4//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 122-135

[24] Lockwood J W, McKeown N, Watson G, et al. NetFPGA—an open platform for gigabit-rate network switching and routing//Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education. San Diego, USA, 2007: 160-161

[25] Forencich A, Snoeren A C, Porter G, et al. Corundum: An open-source 100-gbps nic//Proceedings of the 28th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). Fayetteville, Arkansas, 2020: 38-46

[26] Kumazoe K, Shibata M, Tsuru M. A P4 BMv2-based feasibility study on a dynamic in-band control channel for SDN//Proceedings of the International Conference on Intelligent Networking and Collaborative Systems. Cham: Springer International Publishing, Tokyo, Japan, 2022: 442-451

[27] Shukla A, Hudemann K, Vági Z, et al. Fix with p6: Verifying programmable switches at runtime//Proceedings of the 2021 IEEE Conference on Computer Communications. Vancouver, Canada, 2021: 1-10

[28] Abdelsalam A, Tulumello A, Bonola M, et al. Pushing network programmability to the limits with SRv6 uSIDs and P4//Proceedings of the 3rd P4 Workshop in Europe. Barcelona, Spain, 2020: 62-64

[29] Laki S, Stoyanov R, Kis D, et al. P4Pi: P4 on Raspberry Pi for networking education. ACM SIGCOMM Computer Communication Review, 2021, 51(3): 17-21

[30] Intel Mount Evans DPU IPU Arm Accelerator at Hot Chips 33. https://www.servethehome.com/intel-mount-evans-dpu-ipu-arm-accelerator-at-hot-chips-33/

[31] P4 INT implementation for Tofino switches. https://github.com/GEANT-DataPlaneProgramming/int-platforms

[32] Ye J L, Chen C, Chu Y H. A weighted ECMP load balancing scheme for data centers using P4 switches//Proceedings of the IEEE 7th International Conference on Cloud Networking (CloudNet). Tokyo, Japan, 2018: 1-4

[33] P4 INT implementation for Tofino switches. https://github.com/microsoft/SwitchML

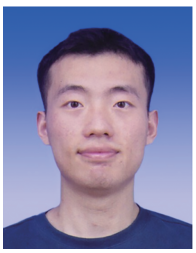
[34] Yang M, Baban A, Kugel V, et al. Using trio: juniper networks' programmable chipset-for emerging in-network applications//Proceedings of the ACM SIGCOMM 2022 Conference. Amsterdam, Netherlands, 2022: 633-648

[35] Pensando DSC-200 Distributed Services Card. https://www.amd.com/system/files/documents/pensando-dsc-200-product-brief.pdf

[36] Yang X, Sun Z, Li J, et al. FAST: Enabling fast software/hardware prototype for network experimentation//Proceedings of the International Symposium on Quality of Service. Phoenix, USA, 2019: 1-10

[37] Yazdinejad A, Bohlooli A, Jamshidi K. P4 to SDNet: Automatic generation of an efficient protocol-independent packet parser on reconfigurable hardware//Proceedings of the 8th International Conference on Computer and Knowledge Engineering. Mashhad, Iran, 2018: 159-164

[38] AMBA AXI4 Interface Protocol. https://www.xilinx.com/products/intellectual-property/axi.html
[39] FastRMT project github repository. https://github.com/Winters123/FastRMT.git (in Chinese)
(FastRMT开源项目流水线源码. https://github.com/Winters123//FastRMT.git)
[40] Opensource P4 Compiler. https://github.com/p4lang/p4c.git
[41] P4 Tutorial. https://github.com/p4lang/tutorials
[42] Jin X, Li X, Zhang H, et al. NetChain: Scale-free sub-RTT coordination//Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation. Washington, USA, 2018: 35-49
[43] Jin X, Li X, Zhang H, et al. Netcache: Balancing key-value stores with fast in-network caching//Proceedings of the 26th Symposium on Operating Systems Principles. Shanghai, China, 2017: 121-136
[44] Sivaraman A, Subramanian S, Alizadeh M, et al. Programmable packet scheduling at line rate//Proceedings of the 2016 ACM SIGCOMM Conference. Florianopolis, Brazil, 2016: 44-57
[45] FastShrivastav V., scalable, and programmable packet scheduler in hardware//Proceedings of the ACM Special Interest Group on Data Communication. Beijing, China, 2019: 367-379



YANG Xiang-Rui, Ph. D., lecturer. His research interests include programmable network and domain-specific network.

ZENG Ling-Bin, Ph.D. candidate. His research interests include data center network and intelligent software technology.

LIU Zhong-Pei, Ph. D. candidate. His research interests include high-performance network and intelligent network.

CHEN Ying-Wen, Ph. D. His research interests include wireless networks and arithmetic networks.

LV Gao-Feng, Ph. D. His main research interests include data center networks and high-speed interconnection networks.

YANG Cheng, Ph. D. His research interest includes distribution computation.

SU Jin-Shu, Ph. D., professor, Ph. D. supervisor. CCF Fellow. His current research interests include cyberspace security and Internet architecture.

Background

This paper aims to provide a flexible and fully-functional FPGA-based experimentation platform for network data plane programmability research, as far as we concerned, for the first time. Compared with existing works, which leverage software-based P4 simulator or ASIC-based programmable switch chips for network behavior simulation, this paper implements a fully-functional RMT (reconfigurable match table architecture) pipeline and flexible interface design for agile experiments on the level of micro-architecture of data plane programmability. Network data plane programmability (Data Plane Programmability) gives powerful programmability to the data plane of network forwarding devices, allowing the deployment of new network protocols, updating security functions, and providing in-network computing acceleration services without updating devices. Due to these advantages, data plane programmability has become an emerging technology that is highly concerned by the industry and academia, and has been put into use by mainstream cloud service providers. Among them, reconfigurable match table architecture (RMT) has become a hot research direction in the programmable data plane due to its excellent processing performance and flexible programming capability with P4 language. However, due to the complex design of the RMT architecture, the closed-source service mechanism of the chip, and the high development threshold of FPGA, it is currently difficult for researchers to innovatively design the micro-architecture of the RMT architecture through FPGA and perform in real performance scenarios (>=100 Gbps). Motivated by the need of research on data plane programmable micro-architecture, this paper proposes an open-source design of a flexible and high-speed programmable data plane prototype system for the first time. The system supports core functions such as RMT architecture programmable protocol parsing, custom rule matching, and action engines based on very-long instruction words, and supports programming of the system via P4 language. In addition, FastRMT also has the characteristics of loose coupling and modularity, which facilitates researchers to replace or reconstruct modules, thereby enabling agile development and verification of new mechanisms or architectures. This paper is supported by the National Natural Science Foundation of China under Grant 62372462 and is part of the research of the key methods and technologies in Domain-Specific Network (DSN).