**一、引言与背景**  
网络接口控制器（NIC）是计算机与网络进行交互的网关。NIC构成了软件协议栈和网络之间的桥梁，该桥梁的功能定义了网络接口。网络接口的功能以及这些功能的实现都在迅速发展。这些变化是由提高线速和支持高性能分布式计算和虚拟化的NIC功能的双重要求所驱动的。不断提高的线速导致许多NIC功能必须在硬件而非软件中实现。同时，需要新的网络功能，例如对多个队列的精确传输控制，以实现高级协议和网络体系结构。

为了以实际的线速满足对新的网络协议和体系结构的开放式开发平台的需求，我们正在开发一种基于FPGA的开源高性能NIC原型平台。这个称为Corundum的平台能够以至少94Gbps的速度运行，是完全开源的，并且连同其驱动程序一起，可以在整个网络协议栈中使用。该设计既便携式又紧凑，支持许多不同的设备，同时即使在较小的设备上也留有足够的资源用于进一步的自定义。Corundum的模块化设计和可扩展性允许共同优化的硬件/软件解决方案在现实的环境中开发和测试高级网络应用程序。

**A.动机和以前的工作**  
通过介绍当前如何在硬件和软件之间划分现有NIC设计中的网络接口功能，可以了解开发Corundum的动机。硬件NIC功能分为两个主要类别。第一类包括简单的卸载功能，这些功能可以从CPU中删除某些按数据包进行的处理，例如校验和/哈希计算和分段卸载，这些功能使网络协议栈可以批量处理数据包。第二类包括必须在NIC上的硬件中实现才能实现高性能和公平性的功能。这些功能包括流量控制，速率限制，负载平衡和时间戳。

一般来说，NIC的硬件功能内置于专有的专用集成电路（ASIC）中。结合规模经济，可以低成本实现高性能。但是，这些ASIC的可扩展性受到限制，添加新硬件功能的开发周期可能既昂贵又耗时[1]。为了克服这些限制，已经开发了各种智能NIC和软件NIC。智能NIC通常通过提供许多可编程处理核心和硬件原语，在NIC上提供强大的可编程性。这些资源可用于从主机上卸载各种应用程序，网络和虚拟化操作。但是，智能NIC不一定能够很好地适应高线路速率，并且硬件功能可能会受到限制[1]。

软件NIC通过绕过大多数硬件卸载功能，在软件中实现网络功能来提供最大的灵活性。因此，可以快速开发和测试新功能，但是要进行各种折衷，包括消耗主机CPU周期，并不一定要支持全线速运行。另外，由于软件固有的随机中断驱动特性，要求精确传输控制的网络应用程序的开发是不可行的[2]。尽管如此，许多研究项目[3]–[6]通过修改网络协议栈或使用诸如Data Plane Development Kit（DPDK）[7]之类的内核绕过框架，在软件中实现了新颖的NIC功能。

基于FPGA的NIC结合了基于ASIC的NIC和软件NIC的功能：它们能够以线速运行并提供低延迟和精确定时，同时新功能的开发周期相对较短。还开发了高性能，基于FPGA的专有NIC。例如，阿里巴巴开发了一个完全定制的基于FPGA的RDMA-onlyNIC，他们用它来运行精密拥塞控制协议（HPCC）的硬件实现[8]。商业产品也存在，包括Exablaze [9]和Netcope [10]提供的产品。

不幸的是，类似于基于ASIC的NIC，可商用的基于FPGA的NIC往往具有无法修改的基本“黑匣子”功能。基本NIC功能的封闭性严重限制了它们在开发新的网络应用程序时的效用和灵活性。

商业上可用的高性能DMA组件，例如Xilinx XDMA内核和QDMA内核，以及Atomic Rules ArkvilleDPDK加速内核[11]都没有提供完全可配置的硬件来控制传输数据流。Xilinx XDMA内核是为计算卸载应用程序而设计的，因此提供了非常有限的排队功能，并且没有简单的方法来控制传输调度。Xilinx QDMA内核和Atomic Rules ArkvilleDPDK加速内核通过支持少量队列并提供DPDK驱动程序而面向网络应用程序。但是，支持的队列数量很少（XDMA内核为2K队列，而Arkville内核为128个队列）而且两个内核都不提供用于精确控制数据包传输的简单方法。

另外，还有诸如NetFPGA [12]之类的开源项目，但是NetFPGA项目仅提供用于基于FPGA的常规数据包处理的工具箱，而并非专门为NIC开发而设计的[P1] 。此外，NetFPGA NIC参考设计利用了Xilinx XDMA内核，该内核并非为网络应用而设计。用Corundum替换NetFPGA板的参考NIC设计中的Xilinx XDMA内核，将提供一个功能更强大，更灵活的原型开发平台。

基于FPGA的分组处理解决方案包括实现网络应用程序卸载的Catapult [13]和实现FPGA上可重构匹配引擎的FlowBlaze [14]。但是，这些平台将标准的NIC功能留给了单独的基于ASIC的NIC，并且完全作为“线下突击”进行操作，没有提供对NIC调度程序或队列的明确控制。

其他项目使用软件实现或部分硬件实现。Shoal [15]描述了一种网络架构，该网络架构使用自定义NIC和快速的第1层电交叉点交换机执行小规模路由。Shoal是用硬件构建的，但仅在没有主机连接的情况下通过综合流量进行评估。SENIC [3]描述了基于NIC的可扩展速率限制。单独评估了调度程序的硬件实现，但是系统级评估是在具有自定义排队规则（qdisc）模块的软件中进行的。PIEO [16]描述了一种灵活的NIC调度程序，它是在硬件中单独进行评估的。NDP [5]是用于数据中心应用程序的拉模式传输协议。NDP已通过DPDK软件NIC和基于FPGA的交换机进行了评估。Loom [6]描述了一种有效的NIC设计，可以使用BESS在软件中对其进行评估。

Corundum的开发与所有这些项目都不同，因为它是完全开源的，并且可以以实际的线速与标准主机网络协议栈一起运行。它提供了数千个传输队列，并带有可扩展的传输调度程序，以实现对流的细粒度控制。最后我们建立了一个强大而灵活的开源平台，用于开发结合了硬件和软件功能的网络应用程序。

**二、实施方式**  
Corundum具有几种独特的体系结构特点。首先，将硬件队列状态有效地存储在FPGA块RAM中，从而支持数千个可单独控制的队列。这些队列与接口相关联，每个接口可以具有多个端口，每个端口都有自己的独立传输调度程序。这样就可以对数据包传输进行极其精细的控制。调度器模块的设计是为了修改或交换的。完全可以实现不同的传输调度方案，包括实验调度器。再加上PTP时间同步，这样可以实现基于时间的调度，包括高精度的TDMA。

Corundum的设计是模块化且高度参数化的。可以在综合时通过Verilog参数设置许多配置和结构选项，包括接口和端口计数，队列计数，内存大小，调度程序类型等。这些设计参数公开在驱动程序读取以确定NIC配置的配置寄存器中，使同一驱动程序无需修改即可支持许多不同的板卡和配置。

当前的设计支持Xilinx Ultrascale PCIe硬IP内核接口的PCIe DMA组件。目前尚未实现对其他FPGA中常用的PCIe TLP接口的支持，这是未来的工作。这种支持应该可以在更多的FPGA上进行操作。

Corundum的占用空间相当小，即使在相对较小的FPGA上，也有足够的空间用于附加逻辑。例如，ExaNIC X10[9]的Corundum设计，是一个双端口10G设计，具有PCIe gen 3 x8接口和512位内部数据路径，其消耗的逻辑资源不到第二小的Kintex Ultrascale FPGA(KU035)上可用逻辑资源的四分之一。表一列举了几个目标平台的资源，放在本文的最后。

本节的其余部分描述了FPGA上Corundum的实现。首先，给出了主要功能块的高层概述。然后，讨论了几个独特的体系结构特征和功能块的细节。

**A.高层概述**  
CorundumNIC的框图如图1所示。从较高的层次来看，NIC由3个主要的嵌套模块组成。顶层模块主要包含支持和接口组件。这些组件包括PCI Express硬IP内核和DMA接口，PTP硬件时钟以及包括MAC，PHY和相关串行器的以太网接口组件。顶层模块还包括一个或多个接口模块实例。每个接口模块都对应于操作系统级别的网络接口（例如eth0）。每个接口模块都包含队列管理逻辑以及描述符和完成处理逻辑。队列管理逻辑维护所有NIC队列的队列状态-传输，传输完成，接收，接收完成和事件队列。每个接口模块还包含一个或多个端口模块实例。每个端口模块都提供一个到MAC的AXI流接口，并包含一个传输调度程序，传输和接收引擎，传输和接收数据路径以及一个暂存RAM，用于在DMA操作期间临时存储传入和传出的数据包。

对于每个端口，端口模块中的传输调度程序决定将哪些队列指定用于传输。传输调度程序为发送引擎生成命令，这些命令协调传输数据路径上的操作。调度程序模块是一个灵活的功能块，可以对其进行修改或替换，以支持任意调度，这些调度可以是事件驱动的。调度程序的默认实现是简单循环。与同一接口模块关联的所有端口共享同一组传输队列，并显示为操作系统的单个统一接口。通过仅更改传输调度程序设置，而不会影响网络协议栈的其余部分，这可以使流在端口之间迁移或在多个端口之间实现负载平衡。这种动态的，调度程序定义的队列到端口的映射是Corundum的独特功能，可以使人们能够研究新的协议和网络体系结构，包括并行网络（例如P-FatTree [17]和光交换网络，例如RotorNet [18]）。和Opera [19]。

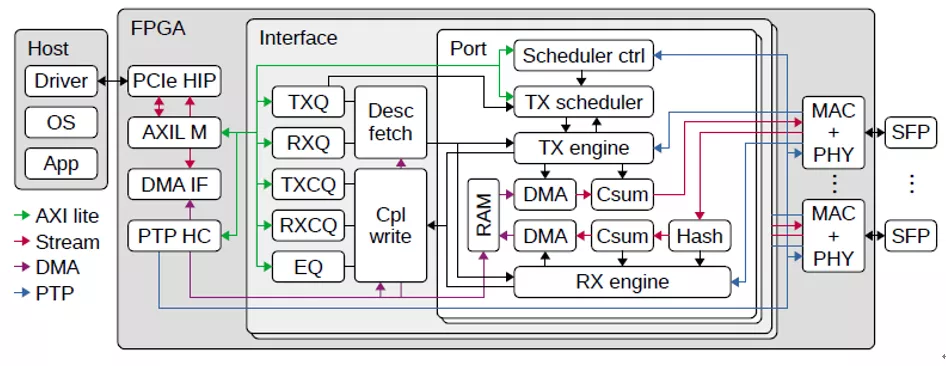


图1.CorundumNIC的框图。PCIe HIP：PCIe硬IP内核； AXIL M：AXI lite Master； DMA IF：DMA接口； PTP HC：PTP硬件时钟； TXQ：传输队列管理器； TXCQ：传输完成队列管理器； RXQ：接收队列管理器； RXCQ：接收完成队列管理器； EQ：事件队列管理器； MAC + PHY：以太网媒体访问控制器（MAC）和物理接口层（PHY）。

在接收方向，传入的数据包通过流哈希模块确定目标接收队列，并为接收引擎生成命令，这些命令协调对接收数据路径的操作。由于同一接口模块中的所有端口共享同一组接收队列，因此不同端口上的传入流将合并到同一组队列中。还可以向NIC添加自定义模块，以在传入数据包通过PCIe总线之前对其进行预处理和过滤。

NIC上的组件与多个不同的接口互连，包括AXI lite，AXI流和用于DMA操作的自定义分段存储器接口，这将在后面讨论。AXI lite用于从驱动程序到NIC的控制路径。它用于初始化和配置NIC组件，并在发送和接收操作期间控制队列指针。AXI stream接口用于在NIC内传输打包数据，包括PCIe传输层数据包（TLP）和以太网帧。分段存储器接口用于将PCIe DMA接口连接到NIC数据路径以及描述符和完成处理逻辑。

大部分NIC逻辑都运行在PCIe用户时钟域中，对于所有当前的设计变体，名义上为250 MHz。异步FIFO用于与MAC接口，这些MAC在串行器中运行，以适当地发送和接收时钟域-10G为156.25 MHz，25G为390.625 MHz，100G为322.266 MHz。

以下各节描述了NIC中的几个关键功能块。

**B.流水线队列管理**  
Corundum NIC和驱动程序之间的数据包数据通信通过描述符和完成队列进行调解。描述符队列形成主机到NIC的通信通道，承载有关各个数据包在系统内存中存储位置的信息。完成队列构成了NIC到主机的通信通道，其中包含有关已完成的操作和关联的元数据的信息。描述符和完成队列被实现为驻留在DMA可访问的系统内存中的环形缓冲区，而NIC硬件则维护必要的队列状态信息。此状态信息包括指向环形缓冲区DMA地址的指针，环形缓冲区的大小，生产者和使用者指针以及对关联的完成队列的引用。每个队列所需的描述符状态适合128位。

Corundum NIC的队列管理逻辑必须能够有效地存储和管理数千个队列的状态。这意味着队列状态必须完全存储在FPGA的Block RAM（BRAM）或Ultra RAM（URAM）中。由于需要128位RAM，并且URAM块为72x4096，因此存储4096个队列的状态仅需要2个URAM实例。利用 URAM 实例可以将队列管理逻辑扩展到每个接口至少处理 32,768 个队列。

为了支持高吞吐量，NIC必须能够并行处理多个描述符。因此，队列管理逻辑必须跟踪多个正在进行的操作，并在操作完成时向驱动程序报告更新的队列指针。跟踪进程中操作所需的状态比描述符状态小得多，因此可以将其存储在触发器和分布式RAM中。

NIC设计使用两个队列管理器模块：queue\_manager用于管理主机到NIC的描述符队列，而cpl\_queue\_manager用于管理NIC到主机的完成队列。除了指针处理，填充处理和门铃/事件生成方面的一些细微差别外，这些模块相似。由于相似之处，本节将仅讨论queue\_manager模块的操作。

用于存储队列状态信息的BRAM或URAM阵列对于每个读取操作都需要几个延迟周期，因此queue\_manager是使用流水线结构构建的，以促进多个并发操作。流水线支持四种不同的操作：寄存器读取，寄存器写入，出队/入队请求和出队/入队提交。通过AXI lite接口进行的寄存器访问操作使驱动程序可以初始化队列状态，并提供指向已分配的主机内存的指针，以及在正常操作期间访问生产者和使用者指针。

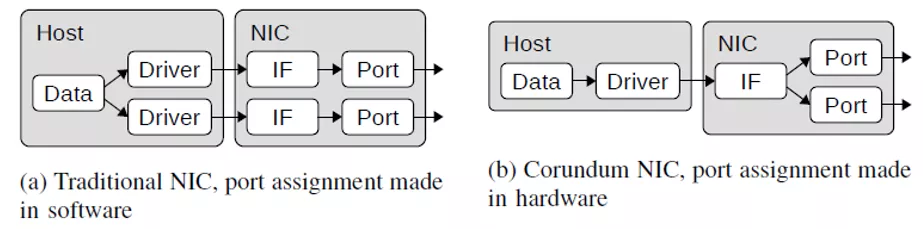
**C.发送调度程序**  
Corundum NIC中使用的默认传输调度程序是在tx\_scheduler\_rr模块中实现的简单循环调度程序。调度器向发送引擎发送命令，从NIC传输队列中启动传输操作。循环调度器包含所有队列的基本队列状态，一个FIFO用于存储当前活动队列并执行循环调度，一个操作表用于跟踪进程中的传输操作。

与队列管理逻辑类似，循环传输调度程序还将队列状态信息存储在FPGA上的BRAM或URAM中，以便可以扩展以支持大量队列。传输调度程序还使用处理流水线来隐藏内存访问延迟。

传输调度器模块具有四个主要接口：AXI lite寄存器接口和三个stream接口。AXI lite接口允许驱动程序更改调度程序参数并启用/禁用队列。当驱动程序将数据包排队发送时，第一个流接口从队列管理逻辑提供门铃事件。第二个流接口将由调度器生成的传输命令携带到发送引擎。每个命令都包含要发送的队列索引以及用于跟踪进程中操作的标签。最终的流接口将传输操作状态信息返回给调度程序。状态信息会通知调度程序已传输数据包的长度，或者是否由于队列为空或禁用而导致传输操作失败。

传输调度程序模块可以扩展或替换以实现任意调度算法。这使Corundum可用作评估实验调度算法的平台，包括SENIC [3]，Carousel [4]，PIEO [16]和Loom [6]中提出的算法。还可能向发射调度器模块提供其他输入，包括来自接收路径的反馈，这些输入可用于实现新协议和拥塞控制技术，例如NDP [5]和HPCC [8]。将调度程序连接到PTP硬件时钟可用于支持TDMA，TDMA可用于实现RotorNet [18]，Opera [19]和其他电路交换体系结构。

**D.端口和接口**  
Corundum的独特体系结构特征是端口和网络接口之间的分隔，因此多个端口可以与同一接口关联。当前的大多数NIC每个接口支持一个端口，如图2a所示。当网络协议栈将数据包排队以便在网络接口上传输时，数据包将通过与该接口关联的网络端口注入网络。但是，在Corundum中，每个接口都可以关联多个端口，因此可以在出队时由硬件决定将数据包注入到网络中的哪个端口，如图2b所示。

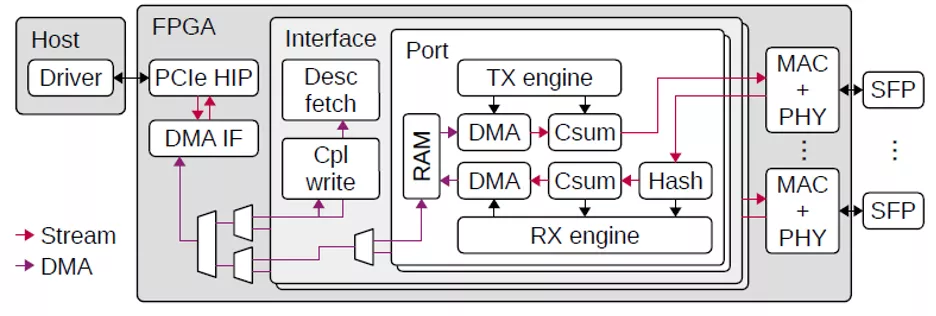


**图2. NIC端口和接口架构比较**

与同一网络接口模块关联的所有端口共享同一组传输队列，并显示为操作系统的单个统一接口。这样，通过仅更改传输调度程序设置，就可以在端口之间迁移流或在多个端口之间实现负载平衡，而不会影响其余的网络协议栈。动态的，由调度程序定义的队列到端口的映射使人们能够研究新的协议和网络体系结构，包括诸如P-FatTree [17]的并行网络以及诸如RotorNet [18]和Opera [19]的光交换网络。

**E.数据路径以及发送和接收引擎**  
Corundum在数据路径中同时使用了内存映射接口和流接口。AXI stream用于在端口DMA模块，以太网MAC，校验和与哈希计算模块之间传输以太网数据包数据。AXI stream还用于将PCIe硬IP内核连接到PCIe AXI lite主模块和PCIe DMA接口模块。定制的分段存储器接口用于将PCIe DMA接口模块，端口DMA模块以及描述符和完成处理逻辑连接到内部暂存器RAM。

AXI stream接口的宽度由所需带宽确定。除以太网MAC外，核心数据路径逻辑完全在250 MHz PCIe用户时钟域中运行。因此，到PCIe硬IP内核的AXI流接口必须与硬核接口宽度匹配-PCIe Gen 3 x8为256位，PCIe Gen 3 x16为512位。在以太网端，接口宽度与MAC接口宽度匹配，除非250 MHz时钟太慢而无法提供足够的带宽。对于10G以太网，MAC接口是156.25 MHz的64位，可以以相同的宽度连接到250 MHz的时钟域。对于25G以太网，MAC接口在390.625 MHz时为64位，因此必须转换为128位才能在250 MHz时提供足够的带宽。对于100G以太网，Corundum在Ultrascale Plus FPGA上使用Xilinx 100G硬CMAC内核。MAC接口在322.266 MHz时为512位，它以512位在250 MHz时钟域上连接，因为它需要以大约195 MHz的频率运行才能提供100 Gbps。



**图3. 图1的简化版本，显示了NIC数据路径**

NIC数据路径的框图如图3所示，它是图1的简化版本。PCIe硬IP内核（PCIe HIP）将NIC连接到主机。两个AXI stream接口将PCIe DMA接口模块连接到PCIe硬IP内核。一个接口用于读写请求，一个接口用于读取数据。然后，PCIe DMA接口模块通过一组DMA接口多路复用器连接到描述符获取模块，完成写入模块，端口暂存RAM模块以及RX和TX引擎。在朝向DMA接口的方向上，多路复用器组合了来自多个源的DMA传输命令。在相反的方向上，它们路由传输状态响应。它们还管理分段存储器接口以进行读取和写入。顶层多路复用器将描述符流量与分组数据流量结合在一起，为描述符流量提供更高的优先级。接下来，一对多路复用器组合来自多个接口模块的流量。最后，每个接口模块内的一个附加多路复用器将来自多个端口实例的分组数据流量组合在一起。

发送引擎和接收引擎负责协调传输和接收数据包所需的操作。发送和接收引擎可以处理多个正在进行的数据包，以实现高吞吐量。如图1所示，发送和接收引擎连接到发送和接收数据路径中的几个模块，包括端口DMA模块以及哈希和校验和卸载模块，以及描述符和完成处理逻辑以及时间戳接口模块、以太网MAC模块。

发送引擎负责协调数据包的传输操作。发送引擎处理来自传输调度程序的特定队列的传输请求。使用PCIe DMA引擎进行低级处理后，数据包将通过传输校验和模块，MAC和PHY。一旦发送了数据包，发送引擎将从MAC接收PTP时间戳，建立完成记录，并将其传递给完成写入模块。

与发送引擎类似，接收引擎负责协调数据包的接收操作。传入的数据包通过PHY和MAC。在包括哈希和时间戳的底层处理之后，接收引擎将向PCIe DMA引擎发出一个或多个写请求，以将数据包数据写出到主机内存中。写操作完成后，接收引擎将构建一个完成记录，并将其传递给完成写模块。

描述符读取和完成写入模块的操作类似于发送和接收引擎。这些模块处理来自发送和接收引擎的描述符/完成读/写请求，向队列管理器发出入队/出队请求，以获取主机内存中的队列元素地址，然后向PCIe DMA接口发出请求以传输数据。完成写入模块还负责通过将发送和接收完成队列排队在适当的事件队列中并写出事件记录来处理事件。

**F.分段内存接口**  
对于PCIe上的高性能DMA，Corundum使用自定义分段存储器接口。该接口被分成最大128位的段，并且整体宽度是PCIe硬IP内核的AXI流接口宽度的两倍。例如，将PCIe Gen 3 x16与PCIe硬核中的512位AXI流接口一起使用的设计将使用1024位分段接口，该接口分成8个段，每个段128位。与使用单个AXI接口相比，该接口提供了改进的“阻抗匹配”，从而消除了DMA引擎中的对齐和互连逻辑中的仲裁，从而消除了背压，从而提高了PCIe链路利用率。具体地说，该接口保证DMA接口可以在每个时钟周期执行全宽度，未对齐的读取或写入。此外，使用简单的双端口RAM（专用于在单个方向上移动的流量）消除了读写路径之间的争用。

除了使用三个接口（而不是五个）之外，每个网段的运行方式均与AXI lite类似。一个通道提供写地址和数据，一个通道提供读地址，一个通道提供读数据。与AXI不同，不支持突发和重新排序，从而简化了接口逻辑。互连组件（多路复用器）负责维护操作的顺序，即使在访问多个RAM时也是如此。这些段通过单独的流控制连接和互连排序逻辑的单独实例彼此完全独立地运行。另外，操作是基于单独的选择信号而不是通过地址解码进行路由的。此功能消除了分配地址的需要，并允许使用可参数化的互连组件，这些组件以最少的配置适当地路由操作。

字节地址被映射到分段的接口地址上，最低的地址位确定段中的字节通道，接下来的位选择段，最高的位确定该段的字地址。例如，在一个1024位分段接口中，分成8个128位段，最低的4个地址位将确定段中的字节通道，接下来的3位将确定该段。其余位确定该段的地址总线。

**G.设备驱动程序**  
Corundum NIC通过内核模块连接到Linux内核网络协议栈。该模块负责初始化NIC，注册内核接口，为描述符和完成队列分配DMA可访问的缓冲区，处理设备中断以及在内核和NIC之间传递网络流量。

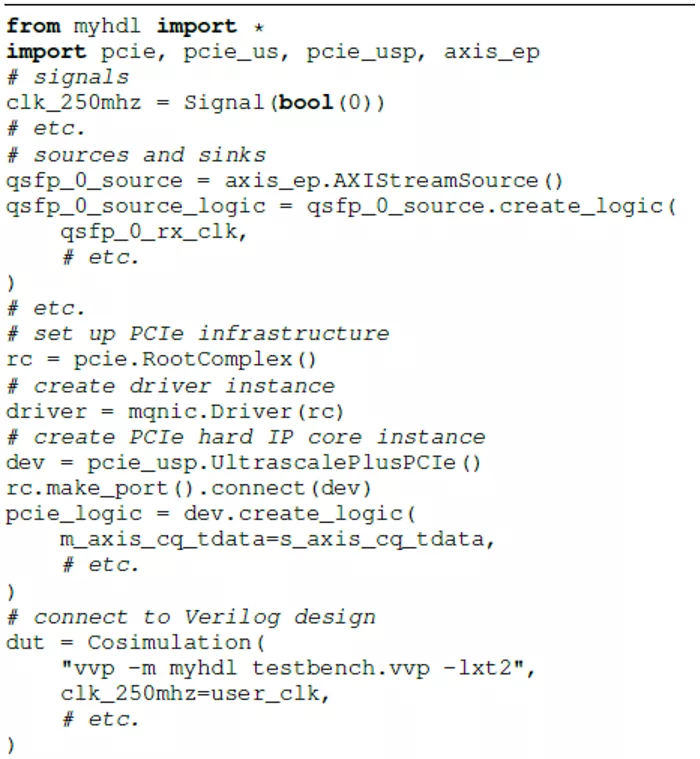
NIC使用寄存器空间公开参数，包括接口数量，端口数量，队列数量，调度程序数量，最大传输单元（MTU）大小以及PTP时间戳和卸载支持。驱动程序在初始化期间读取这些寄存器，因此它可以配置自身并注册内核接口以匹配NIC设计配置。这种自动检测功能意味着驱动程序和NIC松耦合。当在不同的FPGA板，不同的Corundum设计变体和不同的参数设置中使用驱动程序时，通常不需要针对核心数据路径进行修改。

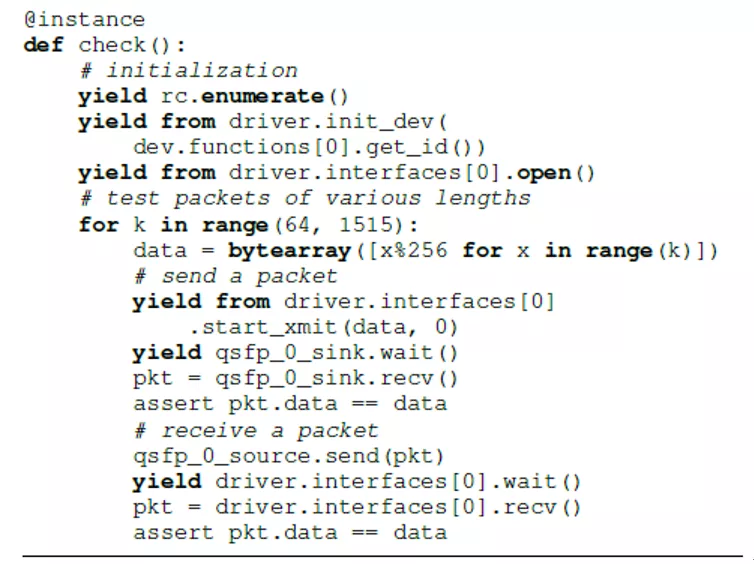
**H.仿真框架**  
包含了一个广泛的基于Python的开源仿真框架，以评估完整设计。该框架使用Python库MyHDL构建，并包括PCI Express系统基础架构，PCI Express硬IP内核，NIC驱动程序和以太网接口的仿真模型。该仿真框架通过提供整个系统状态的可视性，有助于调试完整的NIC设计。

PCIe仿真框架的核心由大约4,500行Python组成，并包括PCIe基础结构组件的事务层模型，其中包括根联合体，功能，端点和交换机以及高级功能，包括配置空间，功能，总线枚举，根联合体内存分配，中断和其他功能。其他模块由另外4,000行Python组成，提供FPGA PCIe硬IP核的模型，与模拟的PCIe基础设施交换事务层流量，并驱动可连接至共同仿真的Verilog设计的信号。

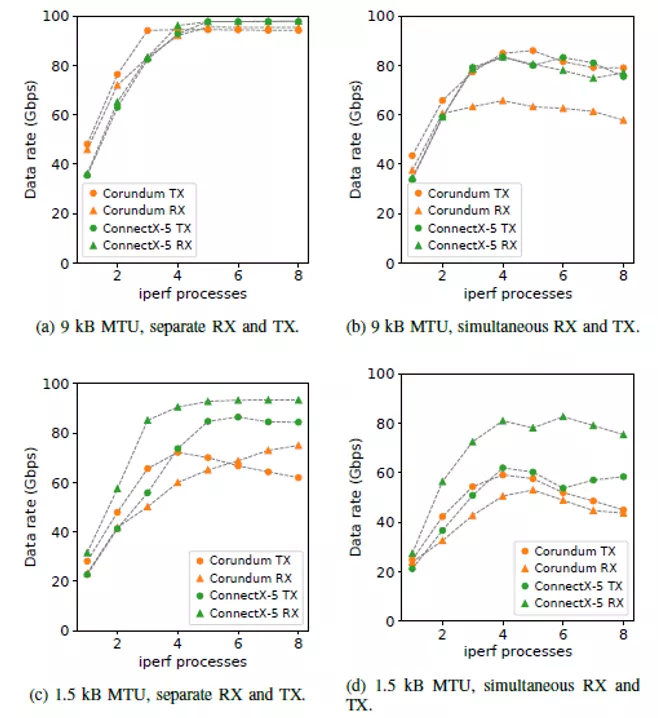
模拟Corundum需要几行代码来实例化和连接所有组件。清单1显示了使用模拟框架发送和接收各种大小的数据包的简化测试台，在Icarus Verilog中共同模拟了Verilog设计。该测试平台实例化了以太网接口端点，PCIe根联合体和驱动程序的仿真模型，并将它们连接到协同仿真的设计。然后，它初始化PCIe基础结构，初始化驱动程序模型，并发送，接收和验证几个不同长度的测试数据包。

**三、结果**  
在安装在Dell R540服务器（双Xeon 6138）中的Alpha Data ADM-PCIE-9V3板上评估了CorundumNIC的100G变体，该主板连接到同一服务器上的最新商业级NIC（Mellanox ConnectX-5）用QSFP28直接连接铜缆。还对在同一台计算机上安装的另外两个Mellanox ConnectX-5 NIC进行了评估，以进行比较。最多使用八个iperf3实例来饱和链接。





清单1. NIC测试台的缩写。包括设置PCIe，以太网接口和驱动程序模型，初始化模拟的PCIe总线和驱动程序以及发送和接收测试数据包。为简洁起见，大多数信号已删除。



**图4.Corundum和MellanoxConnectX-5的TCP吞吐量比较**

为了将Corundum与Mellanox ConnectX-5的性能进行比较，最初将两个NIC的最大传输单位（MTU）配置为9000字节。对于此配置，Corundum可以分别实现95.5 Gbps RX和94.4 Gbps TX（图4a）。在相同条件下，Mellanox ConnectX-5 NIC的RX和TX均达到97.8 Gbps。当运行iperf的其他实例同时在两个方向上使链接饱和时，Corundum的性能将下降到65.7 Gpbs RX和85.9 Gbps TX（图4b）。对于现有的测试台，对于RX和TX，Mellanox NIC的性能也下降到83.4 Gbps。在全双工模式下，Corundum和ConnectX-5的性能下降都表明软件驱动程序可能是导致性能下降的重要原因。具体来说，当前版本的驱动程序仅支持Linux内核网络协议栈。支持诸如DPDK之类的内核绕过框架的参考设计是未来工作的目标。此设计应提高全双工模式的性能，并可针对特定应用进行定制。

图4c和4d比较了1500字节MTU的性能。对于这种情况，Corundum可以分别实现75.0 Gbps RX和72.2 Gbps TX（图4c），同时实现53.0 Gbps RX和57.6 Gbps TX（图4d）。随着iperf实例数量的增加，图4c中TX和RX之间Corundum的性能差异是由进程内传输数据包数量的限制以及PCIe往返延迟引起的。通过将进程内传输操作的数量从8个增加到16个，可以验证这一点。这将速率从65.6 Gbps RX和45.7 Gbps TX提高到了图4c中所示的75.0 Gbps RX和72.2 Gbps TX速率。为了进行比较，在相同条件下，Mellanox ConnectX-5 NIC可以分别实现93.4 Gbps的RX和86.5 Gbps的TX，同时实现82.7 Gbps的RX和62.0 Gbps的TX。

为了测试PTP时间戳的性能，将两个10G模式的CorundumNIC连接到用作PTP边界时钟的Arista 40G数据包交换机。NIC配置为输出源自PTP时间的固定频率信号，该信号由示波器捕获。在启用PTP时间戳的情况下实施Corundum时，可以将硬件时钟与linux ptp同步到50 ns以上。链路饱和时，时间同步性能不变。

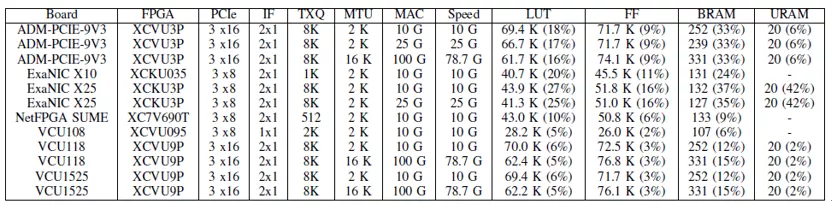
**四、案例研究：时分多址（TDMA）**  
精确的网络准入控制是高线路速率下的关键网络功能。Corundum提供了数千个传输队列，可用于在多个终端主机之间同步的精细时间范围内分离和控制传输数据。此功能提供了一个独特的工具箱，可用于开发新的强大的NIC功能。确定要实现的网络功能以及这些功能对网络性能的影响是一个活跃的研究领域[3]–[5]，[16]。

为了演示如何将Corundum用于精确的传输控制，我们为TDMA实现了具有固定时间表的简单参考设计。从此基本设计和Corundum的模块化结构开始，可以实现针对新型网络协议的自定义调度程序，这些调度程序对整个体系结构的影响最小。

固定的TDMA时间表可以通过IEEE 1588 PTP在多个主机之间同步。在TDMA调度程序控制模块的控制下，通过根据PTP时间启用和禁用传输调度程序中的队列来实现TDMA。队列启用和禁用命令在TDMA调度程序控制模块中生成，并在TDMA调度的每个时隙的开始和结束时发送到发送调度器。TDMA调度器在时隙足够长的假设下操作，使得TDMA调度器控制模块可以在当前时隙期间为下一个时隙做准备。另外，在每个时隙中必须有相对少量的队列处于活动状态，因此启用或禁用的第一个和最后一个队列之间的时滞较小。

TDMA调度程序控制模块在250 MHz PCIe用户时钟域中运行。结果，每个队列花费4 ns遍历每个传输队列以准备下一个时隙（8,192个传输队列总计约32.8 us）。类似地，它需要4 ns的时间来生成每个启用或禁用命令，以发送到传输调度程序模块。

**A. TDMA性能**  
在Alpha DataADM-PCIE-9V3板上评估了具有256个传输队列的100G TDMA变体的Corundum NIC，该板安装在Dell R540服务器（双Xeon 6138）上，连接到Mellanox ConnectX-5 NIC。使用了八个实例的iperf3来饱和链路，两个网卡的MTU配置为9 kB。在禁用TDMA的情况下，网卡以94.0 Gbps的速度运行。TDMA调度器被配置为运行一个周期为200 µs的调度，包含两个100 µs的时隙，在第一个时隙中启用所有传输队列，在第二个时隙中禁用。在100 Gbps的传输数据路径中，11个数据包的间隔为8 µs（每个数据包11个0.72 µs），再加上1 µs来禁用所有256个队列，Corundum可以以两个数据包长度或1.4 µs的精度控制离开网卡的数据。



**表I 资源利用**

使用200us的时间表，以10 Gbps的线速和1500字节的MTU运行附加测试。该时间段被划分为两个100 us的时隙。考虑到10 Gbps的传输数据路径中的32个数据包的间隔为38 us（每个数据包32 1.2 us）加上1 us以禁用所有256个队列，Corundum可以以两个数据包长度或两个数据包的精度控制离开NIC的数据 2.4us。

**五、结论**  
在本文中，我们介绍了Corundum，这是一种基于FPGA的开源高性能NIC。初始设计的测量性能提供了现实的线速，足以开发和测试新的网络应用程序。现有的和计划中的开源参考设计可实现自定义和进一步的性能改进。这些功能为网络研究和开发提供了强大的原型平台，包括基于NIC的调度程序，例如SENIC [3]，Carousel [4]，PIEO [16]和Loom [6]，新协议和拥塞控制技术，例如 NDP [5]和HPCC [8]。Corundum还启用了新的并行网络体系结构，例如P-FatTree [17]，RotorNet [18]和Opera [19]。优化设计以提高较小数据包大小的性能，以及基于精确数据包传输为新的网络协议定制设计是当前工作的目标。

**参考资料**  
[1]D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H.Angepat, V. Bhanu, A. Caulfield, E. Chung, H. K. Chandrappa, S. Chaturmohta, M.Humphrey, J. Lavier, N. Lam, F. Liu, K. Ovtcharov, J. Padhye, G. Popuri, S.Raindel, T. Sapre, M. Shaw, G. Silva, M. Sivakumar, N. Srivastava, A. Verma, Q.Zuhair, D. Bansal, D. Burger, K. Vaid, D. A. Maltz, and A. Greenberg, “Azureaccelerated networking: SmartNICs in the public cloud,” in 15th USENIXSymposium on Networked Systems Design and Implementation (NSDI 18). Renton, WA:USENIX Association, Apr. 2018, pp. 51–66. [Online]. Available: <https://www.usenix.org/conference/nsdi18/presentation/firestone>  
[2]B. Stephens, A. Akella, and M. M. Swift, “Your programmable NIC should be aprogrammable switch,” in Proceedings of the 17th ACM Workshop on Hot Topics inNetworks, ser. HotNets ’18. New York, NY, USA: Association for ComputingMachinery, 2018, p. 36–42. [Online]. Available:<https://doi.org/10.1145/3286062.3286068>

[3]S. Radhakrishnan, Y. Geng, V. Jeyakumar, A. Kabbani, G. Porter, and A. Vahdat,“SENIC: Scalable NIC for end-host rate limiting,” in 11th USENIX Symposium onNetworked Systems Design and Implementation (NSDI 14). Seattle, WA: USENIXAssociation, 2014, pp. 475–488. [Online]. Available:[https://www.usenix.org/conference/nsdi14/technical-sessions/presentation...](https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/radhakrishnan)

[4]A. Saeed, N. Dukkipati, V. Valancius, V. The Lam, C. Contavalli, and A. Vahdat,“Carousel: Scalable traffic shaping at end hosts,” in Proceedings of theConference of the ACM Special Interest Group on Data Communication, ser.SIGCOMM ’17. New York, NY, USA: Association for Computing Machinery, 2017, p.404–417. [Online]. Available: <https://doi.org/10.1145/3098822.3098852>

[5]M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M.W´ojcik, “Re-architecting datacenter networks and stacks for low latency andhigh performance,” in Proceedings of the Conference of the ACM Special InterestGroup on Data Communication, ser. SIGCOMM ’17. New York, NY, USA: Associationfor Computing Machinery, 2017, p. 29–42. [Online]. Available:<https://doi.org/10.1145/3098822.3098825>

[6]B. Stephens, A. Akella, and M. Swift, “Loom: Flexible and efficient NIC packetscheduling,” in 16th USENIX Symposium on Networked Systems Design andImplementation (NSDI 19). Boston, MA: USENIX Association, Feb. 2019, pp. 33–46.[Online]. Available: <https://www.usenix.org/conference/nsdi19/presentation/stephens>

[7]“Data plane development kit,” <https://www.dpdk.org/>.

[8]Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F.Kelly, M. Alizadeh, and et al., “HPCC: High precision congestion control,” inProceedings of the ACM Special Interest Group on Data Communication, ser.SIGCOMM ’19. New York, NY, USA: Association for Computing Machinery, 2019, p.44–58. [Online]. Available: <https://doi.org/10.1145/3341302.3342085>

[9]“Exablaze,” <https://exablaze.com/>.

[10]“Netcope technologies,” <https://www.netcope.com/en>.

[11]“Atomic rules,” <http://www.atomicrules.com/>.

[12]N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, “NetFPGA SUME:Toward 100 Gbps as research commodity,” IEEE Micro, vol. 34, no. 5, pp. 32–41,Sep. 2014.

[13]A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S.Heil, M. Humphrey, P. Kaur, J.-Y. Kim, and et al., “A cloud-scale accelerationarchitecture,” in The 49th Annual IEEE/ACM International Symposium onMicroarchitecture, ser. MICRO-49. IEEE Press, 2016.

[14]S. Pontarelli, R. Bifulco, M. Bonola, C. Cascone, M. Spaziani, V. Bruschi, D.Sanvito, G. Siracusano, A. Capone, M. Honda, F. Huici, and G. Siracusano,“FlowBlaze: Stateful packet processing in hardware,” in 16th USENIX Symposiumon Networked Systems Design and Implementation (NSDI 19). Boston, MA: USENIX Association,Feb. 2019, pp. 531–548. [Online]. Available:<https://www.usenix.org/conference/nsdi19/presentation/pontarelli>

[15]V. Shrivastav, A. Valadarsky, H. Ballani, P. Costa, K. S. Lee, H. Wang, R.Agarwal, and H. Weatherspoon, “Shoal: A network architecture for disaggregatedracks,” in 16th USENIX Symposium on Networked Systems Design and Implementation(NSDI 19). Boston, MA: USENIX Association, Feb. 2019, pp. 255–270. [Online].Available: <https://www.usenix.org/conference/nsdi19/presentation/shrivastav>

[16]V. Shrivastav, “Fast, scalable, and programmable packet scheduler in hardware,”in Proceedings of the ACM Special Interest Group on Data Communication, ser.SIGCOMM ’19. New York, NY, USA: Association for Computing Machinery, 2019, p.367–379. [Online]. Available: <https://doi.org/10.1145/3341302.3342090>

[17]W. M. Mellette, A. C. Snoeren, and G. Porter, “P-FatTree: A multi-channeldatacenter network topology,” in Proceedings of the 15th ACM Workshop on HotTopics in Networks, ser. HotNets ’16. New York, NY, USA: Association forComputing Machinery, 2016, p. 78–84. [Online]. Available: <https://doi.org/10.1145/3005745.3005746>

[18]W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren,and G. Porter, “RotorNet: A scalable, low-complexity, optical datacenternetwork,” in Proceedings of the Conference of the ACM Special Interest Group onData Communication, ser. SIGCOMM ’17. New York, NY, USA: Association forComputing Machinery, 2017, p. 267–280. [Online]. Available: <https://doi.org/10.1145/3098822.3098838>

[19]W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter,“Expanding across time to deliver bandwidth efficiency and low latency,” in17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20).Santa Clara, CA: USENIX Association, Feb. 2020. [Online].Available:<https://www.usenix.org/conference/nsdi20/presentation/mellette>.

文章原文链接地址：A. Forencich, A. C. Snoeren, G. Porter, G. Papen, \*Corundum: An Open-Source 100-Gbps NIC,\* in FCCM'20, [Paper](<https://www.cse.ucsd.edu/~snoeren/papers/corundum-fccm20.pdf>)。

源代码工程地址：<https://github.com/ucsdsysnet/corundum>。