计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202111205
60(6): 1322−1336, 2023

nPSA: 一种面向 TSN 芯片的低延时确定性交换架构

付文文[1] 刘汝霖[1] 全 巍[1] 姜旭艳[1] 孙志刚[1,2]
[1](国防科技大学计算机学院 长沙 410073)
[2](并行与分布处理国防科技重点实验室(国防科技大学) 长沙 410073)
(fuwenwen94@nudt.edu.cn)

nPSA: A Low-Latency, Deterministic Switching Architecture for TSN Chips

Fu Wenwen[1], Liu Rulin[1], Quan Wei[1], Jiang Xuyan[1], and Sun Zhigang[1,2]
[1] (*College of Computer Science and Technology*, *National University of Defense Technology*, *Changsha* 410073)
[2] (*Science and Technology on Parallel and Distributed Processing Laboratory* (*National University of Defense Technology*), *Changsha* 410073)

Abstract Time-sensitive networking (TSN) guarantees the real-time and determinism for critical traffic through spatio-temporal resource planning. The planning tool uses the maximum switching delay of each chip under heavy load as input parameters when allocating temporal resources. In order to satisfy the low-delay requirement of TSN applications, the TSN chip designers are supposed to minimize the maximum switching delay as an important goal. Current commercial TSN chips generally adopt a single-pipeline switching architecture, which is prone to "complete frame blocking" at the entrance of the pipeline, resulting that it is hard to reduce the maximum switching delay. Therefore, we propose a multi-pipeline switching architecture named nPSA based on time division multiplexing mechanism, which optimizes the "complete frame blocking" into a "slice blocking" problem. Moreover, the weighted round-robin slot allocation algorithm (WRRSA) is proposed for the time division multiplexing mechanism to calculate the slot allocation scheme under different port types. At present, the nPSA architecture and WRRSA algorithm have been applied in the OpenTSN open-source chip and the "HX-DS09" ASIC chip. The actual test results show that the maximum switching delay time experienced by the 64B critical frame in the OpenTSN chip and the "HX-DS09" chip are 1648ns and 698ns, respectively. Compared with the theoretical value are the TSN switching chip designed based on the single-pipeline architecture, the delay value are reduced by about 88% and 95% respectively.

Key words time-sensitive networking; switching architecture; maximum switching delay; time division multiplexing; multi-pipelines; slot allocation algorithm

摘 要 时间敏感网络(time-sensitive networking, TSN)通过时空资源规划保证关键流量传输的实时性和确定性,规划工具在分配时间资源时使用关键帧,在重负载情况下进出芯片的最大交换延时时作为输入参数.为了满足 TSN 应用的低传输延时要求,TSN 芯片设计时需要以最小化最大交换延时为重要目标.当前商用 TSN 芯片一般采用单流水线交换架构,容易在流水线的入口处发生"完整帧阻塞"问题,导致芯片的最大交换延时难以降低.针对此问题,提出了一种基于时分复用的多流水线交换架构(n-pipeline switching architecture, nPSA)该架构将"完整帧阻塞"问题优化成"切片阻塞"问题.同时,提出了面向时分复用机制的加权轮询式时隙分配算法(WRRSA)以求解不同端口类型组合下的时隙分配方案.目前 nPSA 架构和 WRRSA 算法已经在 OpenTSN 开源芯片和"枫林一号"ASIC 芯片(HX-DS09)中得到应用.实际测试结果显示,长度为 64 B 的关键帧在 OpenTSN 芯片和"枫林一号"芯片中经历的最大交换延时分

收稿日期: 2021-12-07; **修回日期**: 2022-06-27
通信作者: 刘汝霖(rl_liu@nudt.edu.cn)

别为 1648 ns 和 698 ns，与基于单流水线架构的 TSN 交换芯片的理论值相比，延时数值分别降低约 88% 和 95%.

关键词 时间敏感网络；交换架构；最大交换延时；时分复用；多流水线；时隙分配算法

中图法分类号 TP391

在工业自动化、自动驾驶等实时领域，总线技术已逐渐达到带宽瓶颈，无法承载日益增长的流量规模，并且各总线技术互不兼容，制约了其发展前景[1-2]. 由于以太网具有高带宽与良好兼容性的特点，工业界和学术界期望采用以太网技术弥补总线技术的不足. 然而，以太网“尽力而为”的转发模式容易导致帧丢失或经历不确定的传输延时，这使其难以满足实时应用的实时性和确定性要求. 因此相关工作组在传统以太网标准基础上扩展时间同步和时间感知整形内容，制定了时间敏感网络（time-sensitive networking, TSN）标准[3].

在 TSN 系统中，为了保证关键流量传输延时的确定性和实时性，TSN 规划工具需要预先为关键流量规划其在每个节点的发送时间. 其中关键流量是被调度的对象，因此也称被调度流量（scheduled traffic, ST）. 规划发送时间时，规划工具将重负载情况下 ST 帧在各节点中可能经历的最大交换延时作为规划算法的输入参数[4]. 节点的最大交换延时与相邻节点发送时间的间隔呈正相关关系（具体分析见 1.1 节）. 为了缩短相邻节点的发送时间间隔以实现更低的传输延时，进而满足 TSN 应用对传输延时的苛刻要求（如汽车自动化中的动力系统和底盘控制要求传输延时小于 10μs[5]），TSN 交换芯片设计时需要以最小化最大交换延时为重要目标.

当前的商用 TSN 交换芯片（如 BROADCOM 公司的 BCM53154[6] 芯片）一般沿用以太网中主流的单流水线交换架构（1-pipeline switching architecture, 1PSA）[7-8]. 在 1PSA 架构中，所有端口的帧共享一条处理流水线，这使得不同端口的帧因竞争流水线的处理资源而在流水线入口处发生完整帧阻塞问题，产生阻塞延时. 最大阻塞延时与端口类型、流水线工作频率和处理位宽等因素相关. 以 2×10 Gbps+8×1 Gbps 端口组合的 TSN 芯片为例，假设流水线的工作频率为 125 MHz，处理位宽为 256 b，帧在该芯片中可能经历的最大阻塞延时超过 24 μs（具体分析见 1.2 节）. 这导致基于 1PSA 架构的 TSN 交换芯片的最大交换延时难以降低，并且需要消耗大量的存储资源缓存阻塞延时内到达的帧数据（下文简称该缓存为阻塞缓存）.

针对已存在的问题，本文面向 TSN 芯片提出了一种超低阻塞的多流水线交换架构（n-pipeline switching architecture, nPSA）. 根据 TSN 标准描述[3]，TSN 的许多功能（如整形调度、帧抢占、帧复制与消除）适合在各端口逻辑中部署，在单流水线中实现和管理复杂. 因此 nPSA 架构为每个端口都实例化一条独享的处理流水线，可消除 1PSA 架构中在流水线入口处的阻塞问题. 而且为了提升存储资源的利用率和降低功耗，nPSA 采用集中式共享存储结构，即所有端口的帧共享集中缓冲区（具体分析见 1.3 节）. 这使得 nPSA 架构的多条流水线需要竞争集中缓冲区，进而引入新的 nPSA 阻塞问题.

为了最小化因竞争集中缓冲区而产生的阻塞延时和阻塞缓存，nPSA 架构通过基于时分复用的访存机制将 1PSA 架构中的完整帧阻塞问题优化成切片阻塞问题. 该访存机制的基本思想包括：1）各流水线累积接收的数据量达到切片寄存器的容量时（切片寄存器容纳的数据量远小于完整帧的数据量）便可写入集中缓冲区；2）各流水线写入/读取集中缓冲区的时隙由算法预先规划好，可保证访存操作有序无冲突地进行. 此外，为了能使该访存机制运行，本文提出了加权轮询式时隙分配算法（weighted round-robin slot allocation algorithm, WRRSA）. 该算法可灵活地求解不同端口组合下各流水线访问集中缓冲区的时隙分配方案.

然后，本文从理论维度比较 nPSA 和 1PSA 架构在多种典型的芯片配置下产生的最大阻塞延时和阻塞缓存所需的存储资源. 理论评估结果表示 nPSA 架构产生的最大阻塞延时和所需的阻塞缓存容量比 1PSA 架构低 2 个数量级. nPSA 架构和 WRRSA 算法在 OpenTSN 开源芯片[9-10] 和“枫林一号”ASIC 芯片[11] 中得到应用. 为了证实理论评估结果，本文对 OpenTSN 芯片（基于 FPGA 实现）和“枫林一号”芯片的最大交换延时进行实测. 测试结果显示在不同流量负载下，长度为 64 B 的 ST 帧在 OpenTSN 芯片和“枫林一号”芯片的最大交换延时（头进头出）分别为 1648 ns 和 698 ns（其中最大阻塞延时都为 72 ns）. 与基于

1PSA 架构的 TSN 交换芯片的理论值相比，最大交换延时分别降低约 88% 和 95%. 而且 OpenTSN 芯片和“枫林一号”芯片的阻塞缓存容量都为 2.25 Kb，与基于 1PSA 架构的 TSN 芯片的理论值相比都降低约 97%. 此外，本文分别基于 OpenTSN 芯片和“枫林一号”芯片(ASIC 芯片验证板)搭建了支持混合优先级流量传输的真实环境并进行实验. 实验结果显示 OpenTSN 芯片和“枫林一号”芯片可为 ST 流量提供亚微秒级确定性和微秒级实时性传输服务，满足现有 TSN 场景对确定性和实时性的需求.

本文主要贡献有 4 点：

1)面向 TSN 芯片提出了一种多流水线交换架构 nPSA，有效降低最大交换延时和存储资源消耗；

2)提出了基于时分复用的访存机制，以保证各流水线能够超低阻塞地访问集中缓冲区；

3)提出了加权轮询式时隙分配算法，以求解不同端口组合下的时隙分配方案；

4)在真实的 TSN 芯片中对 nPSA 架构进行实践，并通过实验证明了其优点.

1 背景与挑战

1.1 最大交换延时作用分析

与软件定义网络(software defined network, SDN)[12]类似，TSN 系统一般采用控制平面与数据平面解耦的网络架构. 控制平面为所有 ST 帧规划其在各节点的发送时间；数据平面根据规划结果执行相应的按时转发操作.

发送时间的规划粒度是规划算法的重要参数，直接影响算法的计算复杂性和可规划的 ST 流量规模. 根据现有的规划算法，发送时间的规划粒度可分为时间颗粒(time granularity, TG)和时间槽(time slot, TS)2 类. 基于时间颗粒的规划算法一般以数十纳秒至 1 μs 为时间单元[3,13]，允许 ST 帧在相邻节点传输时占用多个时间单元. 而基于时间槽的规划算法将时间跨度增大，保证在任意相邻节点中 1 个时间槽最多传输 1 个 ST 帧，且保证在相邻节点中当前时间槽在上游节点发送的 ST 帧能够在下一时间槽从下游节点中发出. 时间槽的设置一般为数十微秒[4].

为了对基于 2 种粒度的规划算法性能进行深入比较，本文设置如图 1(a)所示的实验场景，其中 f_1和 f_2为 ST 流. 面向该场景，基于 2 种粒度的规划结果分别如图 1(b)(以时间颗粒为规划粒度)和图 1(c)(以时间槽为规划粒度)的甘特图所示，通过比较可直接得出 2 个结论.

1) 在基于时间颗粒的规划算法中，发送时间的可选解远多于基于时间槽的规划算法，即图 1(b)中横坐标的值域远大于图 1(c)，进而可知图 1(b)的计算复杂性远高于图 1(c). 此外，时间颗粒比时间槽的粒度更细. 在相同的规划周期内，基于时间颗粒的规划算法会生成规模更大的规划表，进而需要更多的片上存储资源.

2) 基于时间槽的规划算法可调度的 ST 流规模更小. 由于 1 个时间槽只能传输 1 个 ST 帧，且时间槽的设置面向最长的 ST 帧，因此传输 ST 短帧时，ST 流量对链路带宽的利用率较低. 例如，当 f_1的报文长度是 f_2的 2 倍时，图 1(c)中 ST 帧在第 1 个时间槽对链路 L_2 的带宽利用率仅为 50%.

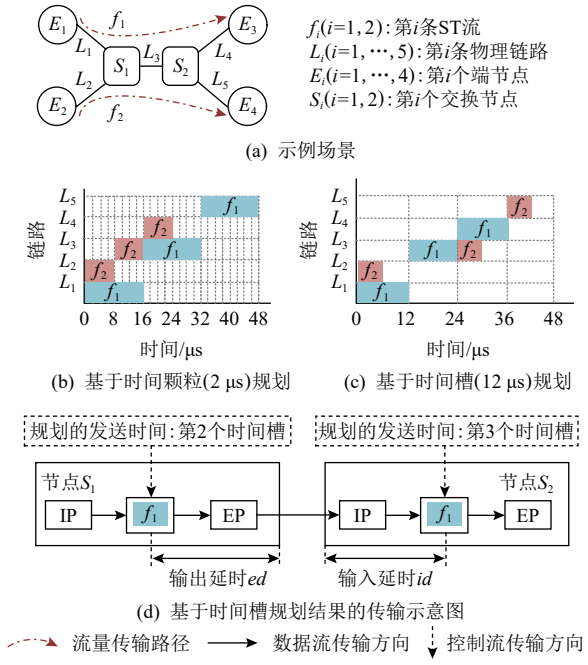


Fig. 1 TSN planning results and transmission scenario

图 1 TSN 规划结果与传输场景

由于 TSN 技术继承以太网技术的高带宽特性，且替代目标为低带宽的总线技术，因此大多数 TSN 场景中，ST 流量带宽占链路带宽的比例很小[1,5]. 这使得基于时间槽的规划算法的理论缺点在众多实际 TSN 场景中可忽视. 因此，基于时间槽的规划算法成为 TSN 规划算法的主要趋势.

根据基于时间槽的规划算法性质(在相邻节点中，当前时间槽在上游节点发送的 ST 帧，能够在下一时间槽从下游节点中发出)，当 ST 帧在下游节点的发送时间槽始终比上游节点的发送时间槽大 1 时，相邻节点的发送时间间隔最小，且端到端传输

延时最小. 最小端到端传输延时的取值范围为$((h-1)\times V_{TS}, (h+1)\times V_{TS})$, 其中$h$为传输路径中交换节点的数量, V_{TS}为时间槽的取值, 进而可知 ST 帧的最小端到端传输延时与时间槽的取值呈线性正相关关系. 因此为了满足汽车动力系统和底盘控制等 TSN 应用对传输延时的苛刻要求, 时间槽的取值需要尽可能小.

由于当前时间槽在上游节点发送的 ST 帧能够在下一时间槽从下游节点中发出(如图 1(d)中 ST 帧在第 2 个时间槽从上游节点 S_1 发出, 在第 3 个时间槽从下游节点 S_2 发出), 时间槽的取值需要满足式(1)中所示的约束.

$$V_{TS} \geqslant S_1 \cdot ed_{max} + S_2 \cdot id_{max} + \delta + Gb + \frac{L}{r}. \quad (1)$$

其中, $S_1.ed_{max}$为节点 S_1 的最大输出延时(帧开始被调度至该帧开始输出到链路中可能经历的最大延时), $S_2.id_{max}$为节点 S_2 的最大输入延时(帧开始输入到芯片至该帧开始写入集中缓冲区中可能经历的最大延时). δ为相邻节点的时钟误差. Gb为用于消除非 ST 帧对 ST 帧确定性干扰的保护带[14]. L为报文长度, r为链路速率. 式(1)表示时间槽的取值需要大于上游节点的最大输出延时、下游节点的最大输入延时、相邻节点时钟误差、保护带和发送延时(L/r)之和.

式(1)中芯片相关的参数包括id_{max}, ed_{max}, δ. 由于时钟同步的相关研究众多, 且同步精度δ一般维持在数十纳秒级[15-16], 对时间槽的取值影响比较小, 本文面向 TSN 芯片设计聚焦于降低id_{max}和ed_{max}取值. 由于id_{max}和ed_{max}是芯片最大交换延时的组成部分, 因此 TSN 芯片需要以最小化最大交换延时为重要设计目标.

1.2 1PSA 交换架构的传输延时分析

目前 TSN 芯片沿用以太网芯片中主流的 1PSA 交换架构, 如图 2 所示. 1PSA 架构使用单流水线处理所有端口输入的帧, 这使得从不同端口同时输入的帧在流水线入口处相互阻塞.

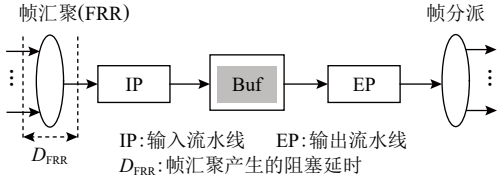


Fig. 2 1PSA switching architecture

图 2 1PSA 交换架构

例如, 帧 A 和帧 B 同时分别从不同端口输入. 2 个帧竞争共享流水线的方式理论上可分为 2 种: 1)切片轮询(slice round-robin, SRR). 该方式需要将帧 A 和帧 B 都切分成若干切片. 流水线轮询调度帧 A 和帧 B 的切片. 2)完整帧轮询(frame round-robin, FRR), 即流水线调度完整帧 A(或帧 B)的所有数据后才会调度另一个帧. 由于单流水线无法识别未携带帧首部的切片, 而且给每个切片增加首部信息会使额外增加的资源过多(增加的资源包括首部添加逻辑、首部消除逻辑、存储首部信息的寄存器等), 因此传统以太网交换芯片采用 FRR 的方式分配流水线资源[8]. 然而该方式必然使帧 B(或帧 A)经历另一个帧的阻塞延时(设为D_{FRR}), 即等待另一个帧传输完所需的时间.

连续工作模式公平调度算法下[17], 帧在汇聚时的阻塞延时D_{FRR}(帧在输入队列中头进头出的延时)满足式(2)所示的关系式, 关系式中使用的参数含义如表 1 所示, 详细证明见附录 A 的定理 A1.

$$D_{FRR} < \frac{N \times L}{f \times B - (V_{total} - V_{min})}. \quad (2)$$

Table 1 Related Parameters of D_{FRR} in 1PSA

表 1 1PSA 中的 D_{FRR} 相关参数

参数	含义
N	端口数量
V_{total}	各端口的传输速率之和
V_{min}	最小的端口传输速率
B	流水线的处理位宽
f	流水线的处理频率
L	以太网的最大帧长度

每个端口需要在帧汇聚处设置一个输入队列作为阻塞缓存. 为了保证在连续工作模式公平调度算法下, 当帧在端口i的输入队列中无溢出时, 该输入队列的容量C_i满足式(3)所示关系式, 详细证明见附录 A 的定理 A2.

$$C_i \geqslant \frac{N \times L}{f \times B - (V_{total} - V_i)} \times V_i. \quad (3)$$

以 2×10 Gbps+8×1 Gbps 端口的 TSN 芯片为例, 为了保证所有端口能够线速传输, 1PSA 架构中单流水线的处理速率($f \times B$)需要大于或等于所有端口的速率之和(28 Gbps). 根据式(2)可知, 流水线的处理速率越快, 即f和B越大, 阻塞延时D_{FRR}越小. 然而, 功耗与f呈正相关关系[18], 因此f不能设置过大. 而且B越大, 消耗的资源也相应地增加. 假设该芯片 f= 125 MHz, B= 256 b, 处理速率为32 Gbps. 此外, TSN 芯片支持的最大帧长度为 1518 B. 根据式(2)可计算出D_{FRR}的理论最大值, 约 24.18 μs. 这使得D_{FRR}成为芯片

最大交换延时的主要部分, 进而导致 1PSA 交换架构的最大交换延时难以降低. 而且根据式(3), 该 TSN 芯片中每个万兆端口对应的阻塞缓存容量应设置为 10.8 KB, 每个千兆端口对应的阻塞缓存容量应设置为 3.1 KB. 这使得基于 1PSA 交换架构的 TSN 芯片需要消耗大量宝贵的片上存储资源.

1.3 TSN 交换架构设计思路与挑战

根据 TSN 标准描述, TSN 功能(如帧复制与消除、帧剥夺、整形调度等)适合在 TSN 芯片的端口逻辑中部署. 因此 TSN 芯片适合为每个端口部署一条独享的流水线, 以降低上述 TSN 功能实现和管理的复杂性, 而且实例化多条流水线可消除 1PSA 架构中的阻塞延时.

此外为了提升存储资源利用率, TSN 交换架构一般采用共享存储结构[19], 即所有端口的帧共享缓冲区, 而且 TSN 芯片常用于原味替代现有的以太网或总线芯片. 在这些场景中 TSN 芯片继续使用被替代芯片的电源, 使功耗高于被替代芯片的 TSN 芯片难以适用于此类场景. 由于分布式共享存储结构允许多个读写操作同时进行, 这使其峰值功耗远大于集中式共享存储结构. 并且集中式共享存储结构的实现和管理复杂性远低于分布式共享存储结构, 这些使集中式共享存储结构在 TSN 交换设备中更适用.

在集中式共享存储结构中, 所有端口的帧共享集中缓冲区, 这使得实例化多条流水线需要竞争共享缓冲区, 引入新的阻塞延时与阻塞缓存. 为了降低因竞争缓冲区而产生的阻塞延时并降低阻塞缓存容量, 基于集中式共享存储的多流水线交换架构需要解决 2 个挑战.

挑战 1: 如何保证各流水线超低阻塞地访问集中缓冲区. 从不同端口同时输入的帧都需要写入集中缓冲区, 然而缓冲区仅有单或双访问接口. 若不对帧的访存操作进行控制, 则不同端口输入的帧可能相互阻塞, 产生类似于 1PSA 交换架构中的阻塞延时, 进而无法有效地降低芯片内的最大交换延时.

挑战 2: 如何适配不同场景下芯片端口(或流水线)数量和处理速率的多样性. 为了适配不同 TSN 芯片中端口(或流水线)数量和处理速率的多样性, TSN 交换架构急需一种灵活的算法为各端口(或流水线)合理地分配集中缓冲区的访问带宽.

为了解决这 2 个挑战, 本文面向 TSN 交换芯片提出一种基于时分复用访存的多流水线交换架构(nPSA), 并为该架构设计了一种加权轮询式时隙分配算法(WRRSA).

2 nPSA 架构设计

如图 3 所示, nPSA 架构的基本组成包括输入流水线(ingress pipeline, IP)、基于时分复用的切片汇聚(slice-based time division multiplexing, STDM)集中缓冲区(Buf)、切片分派和输出流水线(egress pipeline, EP). 其中, IP 完成帧接收与解析等操作; STDM 缓存从 IP 输入的帧切片, 并基于时分复用的方式往集中缓冲区中写入缓存的切片; 集中缓冲区 Buf 存储所有端口输入的帧数据; 切片分派通过时分复用的方式从集中缓冲区中读取帧数据, 并将帧数据传输给各 EP(由于切片分派原理与 STDM 类似, 本文不对其展开描述); EP 完成队列门控和调度输出等功能.

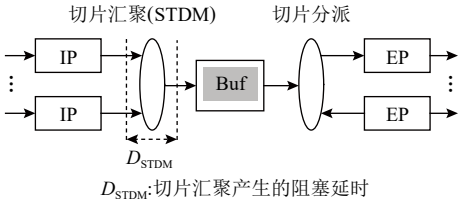


Fig. 3 nPSA switching architecture

图 3 nPSA 交换架构

通过与 1PSA 架构进行比较, nPSA 架构的主要包括 2 点特点:

1)nPSA 架构为每个端口实例化一条独享的流水线. 虽然实例化多条流水线理论上使流水线中逻辑资源的消耗增加, 但是 nPSA 架构消除了 1PSA 架构中各端口的输入队列, 可有效节省存储资源. 而且, 由于 TSN 功能(队列整形调度、帧剥夺), nPSA 大多适合在各端口逻辑中部署, 通过逐端口的流水线实现 TSN 功能可有效降低实现和管理复杂性. 此外, TSN 流水线的处理逻辑简单, 无须集成以太网中复杂的 IP 层查表功能. 这些使得与 1PSA 架构相比, nPSA 架构虽然增加逻辑资源, 但是大幅降低存储资源, 整体上有利于减小芯片面积.

2)nPSA 架构使用 STDM 逻辑处理各流水线的数据汇聚, 将帧阻塞问题优化成切片阻塞问题, 即将阻塞延时由 1PSA 架构中的 D_{FRR} 优化为 D_{STDM}. 在 1PSA 架构中, 为了单流水线能够识别各端口输入的帧数据, 1PSA 架构中的汇聚逻辑必须缓存完整的帧后才能往后传输. 而 nPSA 架构中的 STDM 逻辑只需缓存帧切片, 无须缓存整个帧. 这是因为 nPSA 架构的缓冲区管理逻辑可通过帧数据的载体识别帧切片的来源. 而 1PSA 架构中各端口输入的帧在单流水线

中混合后，其缓冲区管理逻辑无法通过载体识别未携带首部信息的帧切片来源.

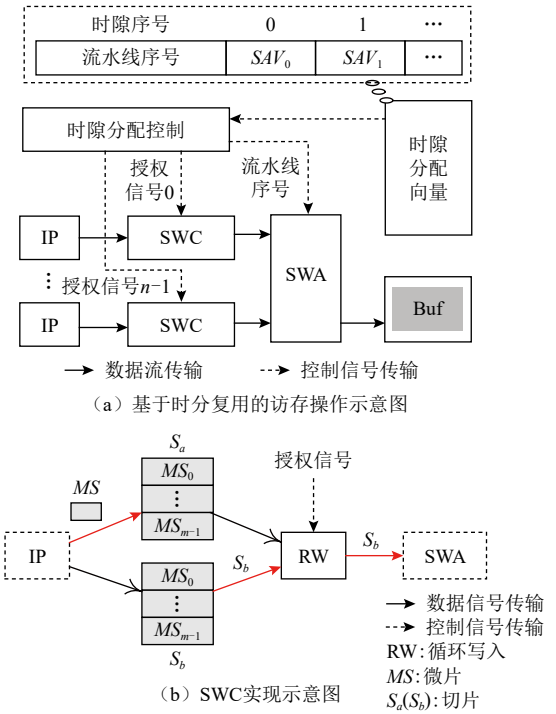
由于 nPSA 架构采用集中存储结构缓存所有端口的帧数据，这使得多条流水线需要竞争集中存储资源. 为了在竞争存储资源时保证各流水线进行超低阻塞的访存操作(挑战 1)，nPSA 架构集成了基于时分复用的访存机制. 而且，为了灵活适配流水线数量和速率的多样性(挑战 2)，本文为访存机制搭配了 WRRSA 算法.

3 关键技术

3.1 基于时分复用的访存机制

nPSA 交换架构通过基于时分复用的访存机制实现超低的阻塞延时与阻塞缓存容量. 该访存机制的主要思想包括：1)通过为每条流水线合理地分配访问集中缓冲区的时隙，保证每个时隙仅被 1 条流水线独享，进而消除访存冲突；2)通过切片轮询替换完整帧轮询，大幅降低阻塞延时与阻塞缓存的容量；3)通过流水线的序号识别切片来源，消除为识别切片来源而消耗的额外的资源.

基于时分复用的访存机制涉及的核心组件如图 4(a)所示，包括时隙分配向量、时隙分配控制、切片写入控制(slice write control, SWC)和切片写入仲裁(slice write arbitrate, SWA).



(a) 基于时分复用的访存操作示意图

(b) SWC实现示意图

Fig. 4 STDM-based memory access mechanism

图 4 基于 STDM 的访存机制

时隙分配向量表示在每个时隙中，SWA 应该调度哪条流水线输入的帧数据. 如图 4(a)所示，向量内容为流水线序号，地址信息为时隙序号. 向量中的地址数量为时隙分配操作循环 1 次所需的时隙数量，设为轮询周期 sn_{period}. sn_{period} 通过式(4)获取.

$$sn_{\text{period}} = \frac{V_{\text{total}}}{gcd(V[0], V[1], \cdots, V[N-1])}. \tag{4}$$

V_{total} 为各流水线(或端口)的速率之和，$gcd(V[0], V[1], \cdots, V[N-1])$ 为所有流水线速率的最大公约数. 例如，在 8×1 Gbps+2×10 Gbps 的端口组合中，端口速率的最大公约数为 1024Mbps(由于目前端口速率单位一般是百兆、千兆或万兆，因此最大公约数使用 Mbps 作为单位)，sn_{period}=28. 而且为了提升时隙分配的粒度，时隙设置为缓冲区管理逻辑的硬件时钟周期. 例如，缓冲区管理逻辑工作在 125 MHz 频率下，其硬件时钟周期为 8 ns，即时隙为 8 ns.

时隙分配控制逻辑用以接收时隙分配向量，并根据时隙分配向量对 SWA 和 SWC 进行控制. 控制的具体方式为在每个时隙中，时隙分配控制逻辑从向量中获取下一时隙应该调度的流水线序号. 然后，时隙分配控制逻辑给流水线序号对应的 SWC 发送有效的授权信号，并且通知 SWA 在下一时隙应该调度的流水线序号.

SWC 主要包括循环输入寄存器组(如 S_a 和 S_b)和循环写入(recurrent writing, RW)逻辑. 如图 4(b)所示，SWC 使用循环输入寄存器组存储对应流水线输入的微片(micro-slice, MS). MS 的位宽为 B_{MS}，即对应流水线 1 个时钟周期处理的数据量. MS 用于组成切片，Sa(sb) 的位宽为 B_{S}，即缓冲区管理和 RW1 个时钟周期处理的数据量. B_{S} 与 B_{MS} 的关系满足 $B_{\text{S}} = m \times B_{\text{MS}}$($m$ 为正整数). 例如，在 8×1 Gbps+2×10 Gbps 的端口组合中，千兆端口对应的 IP 和 EP 工作频率为 125 MHz，万兆端口对应的 IP 和 EP 工作频率为 156.25 MHz，缓冲区管理逻辑的工作频率为 256 MHz；则千兆端口对应的 MS 位宽为 8 b，万兆端口对应的 MS 位宽为 64 b，缓冲区管理逻辑的处理位宽为 128 b，即切片位宽 B_{S} 为 128 b.

当 RW 逻辑从时隙分配控制逻辑接收到有效的授权信号时，才可输出循环输入寄存器中的数据. 输出方式采用循环输出，即本次输出了 S_b 中的数据，下次只能输出 S_a 中的数据，并不断循环. 并且只有当目标寄存器的数据已满时，才允许输出该寄存器内的数据. 输出时，RW 逻辑将寄存器中的数据转发给 SWA，并等待其接收.

SWA 接收时隙分配控制逻辑传输的流水线序号，在下一时隙从该流水线序号对应的 SWC 逻辑中接收帧数据. 帧数据接收成功后，SWA 将该数据直接写入集中缓冲区中，并返回成功信号给 SWC 逻辑.

3.2 加权轮询式时隙分配算法（WRRSA）

基于芯片端口配置的多样性，nPSA 架构的访存机制需要搭配合适的时隙分配算法以生成时隙分配向量. 该时隙分配算法的设计前提是保证各流水线无中断地输出任意帧，并且保证帧数据不会在循环寄存器中溢出. 基于此，WRRSA 算法的设计思包括：

1)各流水线访问集中缓冲区的时隙数量比例等于相应流水线速率占流水线总速率的比例. 例如，10 Gbps 流水线分配的时隙数量是 1 Gbps 流水线的 10 倍.

2)各流水线分配的时隙均匀分布. 为了减少循环寄存器组的存储容量，各流水线分配的时隙需要均匀分布以降低时隙间隔的最大值.

WRRSA 算法的详细描述见算法 1，其相关参数见表 2.

Table 2 Related Parameters of WRRSA Algorithm

表 2 WRRSA 算法相关参数

参数	含义	参数	含义
N	流水线（或端口）数量	$V[i]$	流水线i的传输速率.流水线按照速率由高至低排序，即$V[i]\geqslant V[i+1](i\leqslant N-1)$
V_b	缓冲区管理逻辑的处理速率		
sn_{period}	轮询周期包含的时隙数量	$sn[i]$	流水线i在轮询周期中分配的时隙数量
$interval[i]$	流水线i的相邻时隙间隔	V_{total}	所有流水线速率之和
$reInitial$	重复初始化标识为 0 表示首次初始化；为 1 则表示重复初始化	$tag[i]$	时隙已分配标识为 0 时，表示轮询周期中第i个时隙未被分配
$SV[i][j]$	表示第i条流水线分配的第j个时隙的序号	$SAV[i]$	表示分配到第i个时隙的流水线序号

算法 1. 加权轮询式时隙分配算法(WRRSA).

输入：N, $V[N]$, sn_{period}, V_{total};

输出：$SAV\left[sn_{\text{period}}\right]$.

① for $0\leqslant i\leqslant N-1$

② $sn[i]=sn_{\text{period}}\times\dfrac{V[i]}{V_{\text{total}}}$; /*计算各流水线分配的时隙数量*/

③ $interval[i]=\left\lceil\dfrac{sn_{\text{period}}}{sn[i]}\right\rceil$; /*计算各流水线的时隙间隔*/

④ end for

⑤ for $0\leqslant i\leqslant N-1$

⑥ for $0\leqslant j\leqslant sn[j]-1$

⑦ if($j==0$)&&($reInitial==0$)/*首次为各流水线分配的第 1 个时隙的序号进行赋值*/

⑧ $SV[i][j]=i$;

⑨ else

if ($j==0$)&&($reInitial==1$) /*非首次为各流水线分配的第 1 个时隙的序号进行赋值*/

⑩ $SV[i][j]=SV[i][j]+1$;

⑪ else /*为各流水线分配的后续(非第 1 个)时隙的序号进行赋值*/

⑫ $SV[i][j]=SV[i][j-1]+interval[i]$;

⑬ end if

⑭ $cur\_slot\_seq=SV[i][j]$;

⑮ while ($tag[cur\_slot\_seq]==1$) /*当期望分配的时隙已分配给之前的流水线时，则将期望的时隙序号进行循环减 1(往前移)，直到找到最近的不冲突时隙序号*/

⑯ $SV[i][j]=SV[i][j]-1$;

⑰ $cur\_slot\_seq=SV[i][j]$;

⑱ end while

⑲ $tag[cur\_slot\_seq]==1$; /*将已分配时隙序号进行标记*/

⑳ if ($j==sn[j]-1$) &&

($sn_{\text{period}}-SV[i][j]+SV[i][0])>\left\lceil\dfrac{V_b}{V[i]}\right\rceil$)/*任一流水线的时隙序号赋值结束后，当最后一个时隙序号与下轮第 1 个时隙序号的间隔大于$V_b/V[i]$时，完整帧输出可能中断(具体分析见附录 A 中特性 3 的证明). 此时，重新分配该流水线的时隙序号*/

㉑ $reInitial=1$;

㉒ $j=0$;

㉓ end if

㉔ end for

㉕ end for

㉖ for $0 \leqslant i \leqslant N-1$ /*将 2 维数组SV转化成时隙分配向量SAV*/

㉗ $cur\_slot\_seq = SV[i][j]$;

㉘ $SAV[cur\_slot\_seq] = i$;

㉙ end for

基于算法 1 描述，在 8×1 Gbps+2×10 Gbps 的端口组合中，且缓冲区管理逻辑的处理速率为 32 Gbps 时，WRRSA 算法为该 TSN 芯片计算出的时隙分配向量如式(5)所示，其中序号 1 和 2 表示 2 个万兆端口对应的流水线序号，3~10 表示 8 个千兆端口对应的流水线序号. 轮询周期(sn_{period})为 28，千兆端口的时隙间隔为 28，万兆端口分配的时隙间隔为 3.

$$SAV[28] = \{1, 2, 3, 1, 2, 4, 1, 2, 5, 1, 2, 6, 1, 2, 7, 1, 2, 8, 1, 2, 9, 1, 2, 10, 1, 2, 1, 2\}. \quad (5)$$

WRRSA 算法具有 3 个特性，特性证明见附录 A.

1)nPSA 架构的最大阻塞延时(即D_{STDM}的最大值)为sn_{period}个时隙. 在 8×1 Gbps+2×10 Gbps 端口组合的 TSN 芯片中，D_{STDM}的最大值为 28 个时隙.

2)帧数据在循环输入寄存器组中无溢出的充分必要条件是循环输入寄存器组中寄存器的数量不小于 2.

3)帧传输不中断的充分必要条件是 SWC 中循环输出寄存器组中寄存器的数量不小于 2.

4 实　　现

为了验证 nPSA 架构的优点，OpenTSN 开源芯片(基于 FPGA 实现)和“枫林一号”ASIC 芯片对其进行了实践.

1)OpenTSN 芯片. 该芯片实现架构如图 5(a)所示，其核心组成包括 IP(输入流水线)、配置管理、时分复用访存、时间同步和 EP(输出流水线). 其开源代码和设计文档见文献 [9].

IP 的组成如图 5(b)所示，包括接收控制、帧解析、转发查表、描述符生成、缓冲区申请、缓冲区写模块. IP 负责将配置帧、同步帧和数据帧解析后分别转发给配置管理、时间同步和时分复用访存模块，并且提取数据帧的描述符(包含对应帧的关键信息组)转发给 EP；配置管理通过解析配置帧获取配置内容，并将配置内容写入对应寄存器或表；时分复用访存完成多条流水线的访存操作；时间同步为 IP，EP 和时分复用访存逻辑提供全局统一的时间；EP 的组成如图 5(c)所示，包括描述符接收、入队控制、描述符队列、输出调度、缓冲区读、发送控制模块. 其主要负责根据接收的帧描述符提取对应的数据帧并按照规划的时间输出.

为了测试 nPSA 架构中每条 TSN 流水线所需的逻辑资源，本文基于 FPGA 实现了 OpenTSN 开源芯片逻辑. OpenTSN 芯片具有 9 个千兆接口，资源消耗如表 3 所示. 其中每条输入与输出流水线消耗的自适应查找表(adaptive look-up table)ALUT、寄存器、自适应逻辑模块(adaptive logic module, ALM)资源都小于对应总资源的 8%，且存储块数量仅消耗 13 584，仅占芯片总存储块的 1.6%.

Fig. 5　OpenTSN chip implementation

图 5　OpenTSN 芯片实现

Table 3　Resource Consumption of OpenTSN Chip

表 3　OpenTSN 芯片资源消耗

芯片区域	存储块	ALUT	寄存器	ALM
每条输入流水线	144	510	782	774
每条输出流水线	13 440	1 194	948	461
芯片总资源	849 584	22 936	22 225	16 018
资源比例/%	1.6	7.4	7.8	7.7

注：资源比例=(每条输入+输出流水线)/总资源.

2)“枫林一号”芯片. “枫林一号”的芯片架构与设计思路见文献 [11]. “枫林一号”芯片具有 9 个千兆

端口. 其采用国产 130 nm 工艺流片，芯片封装为 256 引脚 QFP 封装，面积仅为 9 nm×9 nm，峰值功耗仅为 0.46 W.

5 评 估

5.1 理论评估

通过对 1PSA 架构和 nPSA 架构进行比较可知，1PSA 架构的阻塞延时D_{FRR}和 nPSA 架构的阻塞延时D_{STDM}是 2 种架构交换延时的主要变量因素. 为了证明 nPSA 架构可有效降低 1PSA 架构的最大交换延时，本文基于 4 种典型的 TSN 芯片设置对 2 种架构的最大阻塞延时进行量化比较. 量化结果如表 4 所示. 在 4 种典型的 TSN 芯片端口设置中，与 1PSA 架构的D_{FRR}最大值相比，nPSA 架构的D_{STDM}最大值可降低 2 个数量级.

Table 4 Comparison of Theoretical Blocking Delay Between 1PSA Architecture and nPSA Architecture

表 4 1PSA 架构和 nPSA 架构的理论阻塞延时比较

端口组合	缓冲区管理逻辑处理频率 f/MHz	缓冲区管理逻辑处理位宽 B/b	nPSA 架构D_{FRR}理论最大值/μs	nPSA 架构D_{STDM}理论最大值/μs
6×1 Gbps+12×100 Mbps	128	64	242.3	0.576
9×1 Gbps	128	128	13.6	0.072
8× Gbps+2×10 Gbps	256	128	24.2	0.112
16×10Gbps	512	320	19.4	0.032

注：1PSA 架构D_{FRR}求值公式 $D_{\mathrm{FRR}} \leqslant \dfrac{N \times L}{f \times B - (V_{\mathrm{total}} - V_{\mathrm{min}})}$；nPSA 架构$D_{\mathrm{STDM}}$求值公式 $D_{\mathrm{STDM}} \leqslant sn_{\mathrm{period}} \times 1/f$.

本文基于上述 4 种典型的 TSN 芯片设置，对 2 种架构所需的阻塞缓存容量进行量化比较. 量化结果如表 5 所示，与 1PSA 架构阻塞缓存需要消耗的存储容量相比，nPSA 架构也可降低 2 个数量级.

Table 5 Comparison of Theoretical Blocking Cache Capacity Between 1PSA Architecture and nPSA Architecture

表 5 1PSA 架构和 nPSA 架构的理论阻塞缓存容量比较

端口组合	缓冲区管理逻辑处理频率 f/MHz	缓冲区管理逻辑处理位宽 B/b	1PSA 架构阻塞缓存容量理论值/KB	nPSA 架构阻塞缓存容量理论值/KB
6×1 Gbps+12×100 Mbps	128	64	127.2	0.288
9×1 Gbps	128	128	15.3	0.288
8× Gbps+2×10 Gbps	256	128	45.8	0.32
16×10 Gbps	512	320	387.6	1.28

注：1PSA 架构阻塞缓存容量求值公式 $C \geqslant \sum_{i=0}^{N-1} \dfrac{N \times L \times V_i}{f \times B - (V_{\mathrm{total}} - V_i)}$；nPSA 架构阻塞缓存容量求值公式 $C \geqslant \sum_{i=0}^{N-1} \left(\left\lceil \dfrac{V_i \times (i_{\max} - 1)}{V_{\mathrm{b}}} \right\rceil + 1 \right) \times B_{\mathrm{b}}$.

5.2 实验评估

OpenTSN 芯片和“枫林一号”芯片 HX-DS09 都具有 9 个千兆端口. OpenTSN 芯片和中“枫林一号”芯片每条流水线的工作频率和位宽分别为 125 MHz 和 8 b，且缓冲区管理逻辑的工作频率和位宽分别为 125 MHz 和 128 b.

1)最大交换延时. 为了验证 nPSA 架构的优点，本文首先对 OpenTSN 芯片和“枫林一号”芯片的最大交换延时进行实测. 最大交换延时是芯片能力参数，不是组网时的网络延时参数，因此测量拓扑和被测芯片的实际部署拓扑可不同. 测试拓扑为被测芯片与测试仪直连. 具体测试步骤为在 1 个测试端口中输入一条恒定的 ST 流量，帧长度为 1 500 B. 同时，向其他所有个端口都输入动态的背景流量，且背景流量帧长度也为 1 500 B. 所有流量的输出端口号与其输入端口号相同，且通过设置保证帧能够即到即走.

测试时长 1 h，其实验结果如表 6 所示，ST 帧在芯片内经历的最大交换延时与其他端口传输的流量无关(即最大交换延时具有确定性)，在 OpenTSN 芯片和“枫林一号”芯片中的最大交换延时(头进头出)分别为 1648 ns 和 698 ns，其中最大阻塞延时都为 72 ns. 根据式(2)、基于 1PSA 交换架构的相同 TSN 芯片(具有相同的端口、处理频率和位宽)的阻塞延时D_{FRR}的理论最大值约为 13.6 μs. 因此 OpenTSN 芯片和“枫林一号”芯片的最大交换延时分别降低约 88% 和 95%.

Table 6 Comparison of Maximum Switching Delay of Two Chips

表 6 2 块芯片的最大交换延时比较

测试流量/Mbps	背景流量/Mbps	最大交换延时/ns	
		OpenTSN 芯片	“枫林一号”
100		1 608	674
500	100	1 640	688
800	800	1 648	698

由表 6 可证明，与 1PSA 架构相比，nPSA 架构可有效降低 TSN 芯片的最大交换延时.

2)端到端传输延时. 本文通过验证 OpenTSN 芯片和“枫林一号”的实时性和确定性以证明 nPSA 架构的可行性. 本文基于 OpenTSN 芯片构建了如图 6 所示的真实测试环境. 该环境包括 2 块 OpenTSN 芯片(基于 FPGA 实现)、2 台测试仪和 3 台 PC 终端. 其中测试仪 1 发出 1 条 ST 流(ST_1)和 1 条 BE 流(BE_1, 传统以太网中尽力而为的流量, 作为背景流), 测试仪 2 也发出 1 条 ST 流(ST_2)和一条 BE 流(BE_2). 4 条流量的传输路径如图 6 所示.

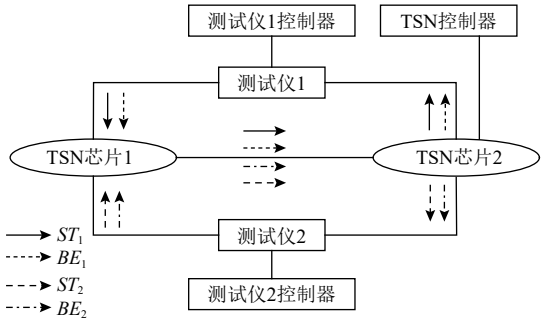


Fig. 6 OpenTSN experiment topology

图 6 OpenTSN 实验拓扑

在 OpenTSN 实验 1 中, 流特征如表 7 所示. 最长的 ST 报文为 128 B, 在 1 Gbps 链路的传输时间为(报文长度/链路速率)为 1 μs. 背景流(BE_1, BE_2)的报文长度为 128 B, 因此背景流对 ST 帧确定性的干扰最大为 1 μs(=128 B/1 Gbps), 进而需要设置 1 μs 的保护带. 此外 2 块 TSN 芯片的时钟同步精度时钟维持在 100 ns 以内, 且 OpenTSN 芯片的最大交换延时为 1648 ns. 根据式(1), 本实验将时间槽设置为 4 μs.

Table 7 Flow Characteristics of OpenTSN Experiment 1

表 7 OpenTSN 实验 1 的流量特征

流	报文长度/B	发送周期/μs	流量带宽/Mbps
ST_1	128	128	16
ST_2	128	128	8
BE_1	128	随机发送	500
BE_2	128	随机发送	500

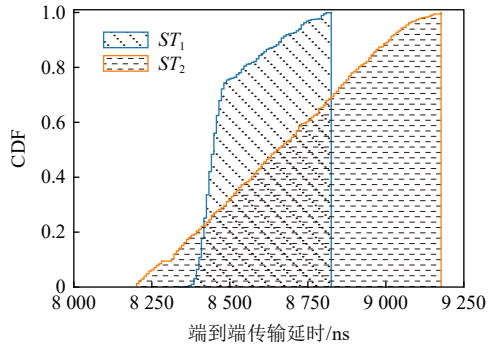
在 OpenTSN 实验 1 中, ST 报文在门控周期的第 0 个时间槽(0~4 μs)从测试仪中发出. 2 块 TSN 芯片的门控表如表 8 所示, 即 2 条 ST 流期望在第 1 个时间槽(4~8 μs)从 TSN 芯片 1 发出, 在第 2 个时间槽(8~12 μs)从 TSN 芯片 2 发出.

OpenTSN 实验 1 的结果如图 7 所示, ST_1 的端到端传输延时在 8 337~8 825 ns 的范围内波动, 其传输延时抖动小于 600 ns. ST_2 的端到端传输延时在 8 201~9 177 ns 的范围内波动, 其传输延时抖动小于 1 000 ns. 该实验结果表明 ST 报文在测试仪至 TSN 芯片 1 链路中的传输时间(包括在下游节点的缓存时间)为 0~4 μs, 在 TSN 芯片 1 至 TSN 芯片 2 链路的传输时间为 4~8 μs, 在 TSN 芯片 2 至测试仪链路的传输时间为 8 μs 以后, 与预期结果一致.

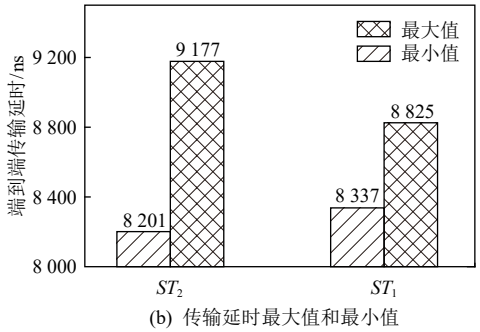
Table 8 Gate Control List of OpenTSN Experiment 1

表 8 OpenTSN 实验 1 的芯片门控列表

芯片	队列	门控状态/时间槽(4 μs)序号						
TSN 芯片 1	ST_1	0/0	1/1	0/2	0/3	0/4	…	0/31
	ST_2	0/0	1/1	0/2	0/3	0/4	…	0/31
TSN 芯片 2	ST_1	0/0	0/1	1/2	0/3	0/4	…	0/31
	ST_2	0/0	0/1	1/2	0/3	0/4	…	0/31



(a) 传输延时分布



(b) 传输延时最大值和最小值

Fig. 7 End-to-end transmission delay of OpenTSN experiment 1

图 7 OpenTSN 实验 1 的端到端传输延时

OpenTSN 实验 2 中测试流量的流量特征如表 9 所示. 按照式(1), 该实验通过调整报文长度修改时间槽的取值. 在该实验中, 最长的 ST 报文为 512 B, 在 1 Gbps 链路的传输时间(报文长度/链路速率)为 4 μs. 背景流(BE_1, BE_2)的报文长度为 1280 B, 因此背景流对 ST 帧确定性的干扰最大为 10 μs(=1280 B/1 Gbps), 进而需要设置 10 μs 的保护带. 此外 OpenTSN 芯片的同步精度为 100 ns. 根据式(1), 本实验将时间槽设置为 16 μs.

Table 9 Flow Characteristics of OpenTSN Experiment 2

表 9 OpenTSN 实验 2 的流量特征

流	报文长度/B	发送周期/μs	流量带宽/Mbps
ST_1	512	512	8
ST_2	256	512	4
BE_1	1 280	随机发送	500
BE_2	1 280	随机发送	500

ST 报文在测试仪的发送时间和 2 块 TSN 芯片的门控表的设置与 OpenTSN 实验 1 相同. OpenTSN 实验 2 的测试结果如图 8 所示. 与预期一致, ST_1 的端到端传输延时围为 33 885～34 005 ns, ST_2 的端到端传输延时围为 33 881～33 972 ns. 2 条 ST 流的传输延时抖动都维持在 0.2 μs 以内.

本文构建了包含 2 块“枫林一号”芯片 HX-DS09

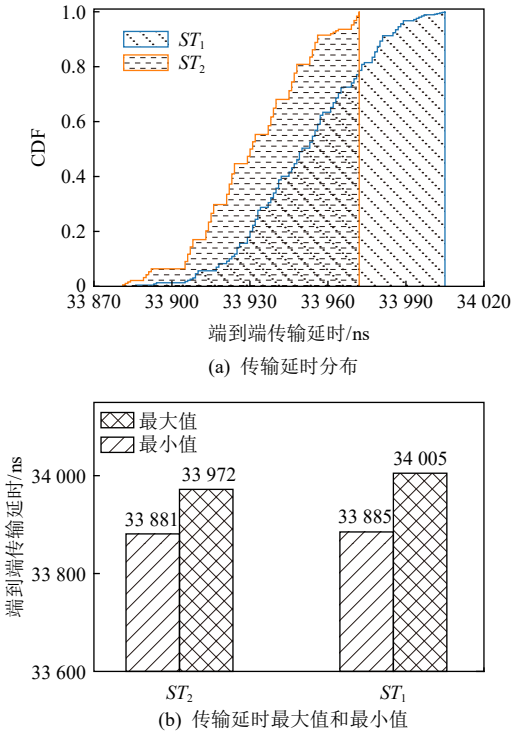


Fig. 8 End-to-end transmission delay of OpenTSN experiment 2

图 8 OpenTSN 实验 2 的端到端传输延时

验证板、4 台 PC 机、1 台示波器、2 个网络摄像头和 1 台测试仪的真实环境, 如图 9 所示. 网络摄像头 1 向对应的视频接收端发送视频流(BE_1 流量), 带宽为 5 Mbps; 网络摄像头 2 向相应接收端发送的视频流(BE_2 流量), 带宽也为 5 Mbps; 测试仪往验证板 1 注入 3 条 ST 流(ST_1, ST_2, ST_3), 剩余带宽发送 BE_3 流量. 3 条 ST 流的流向相同, 都先后经过 TSN 芯片 1、TSN 芯片 2, 最后返回测试仪. 实验场景中所有链路带宽为 1 Gbps.

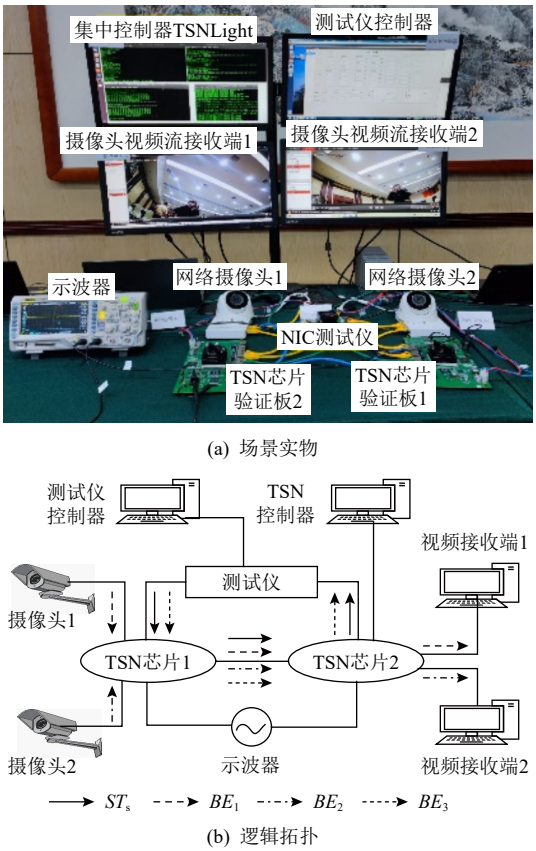


Fig. 9 HX-DS09 experiment topology

图 9 HX-DS09 实验拓扑

每条 ST 流的报文长度为 128 B, 在 1 Gbps 链路的传输时间(报文长度/链路速率)为 1 μs. 而且“枫林一号”将所有长报文切成多个 64~128 B 的切片, 因此切片对 ST 帧确定性的干扰最大为 1 μs(=128 B/1 Gbps), 进而需要设置 1 μs 的保护带. 此外, 2 块 TSN 芯片的时钟同步精度时钟维持在 100 ns 以内. 根据式(1), 本实验将时间槽设置为 4 μs.

为了降低端到端传输延时, 这 3 条 ST 流在 TSN 芯片 1 中规划在第 1 个时间槽(0~4 μs)发送, 在 TSN 芯片 2 中规划的发送时间为第 2 个时间槽(4~8 μs). 本实验在每条 ST 流中随机采样 100 个报文获取其端到端传输延时. 如图 10 所示, 所有报文的最大端到端

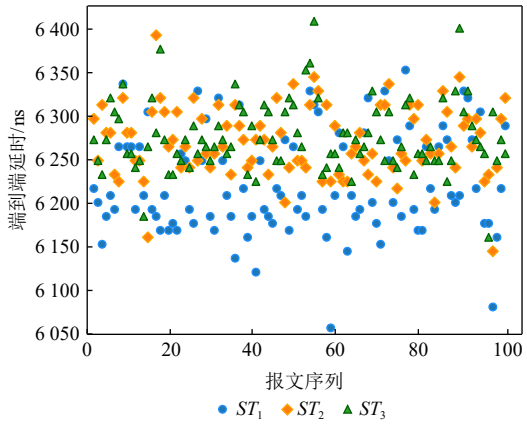


Fig. 10 End-to-end transmission delay of HX-DS09 experiment

图 10 HX-DS09 实验的端到端传输延时

传输延时为 6.42 μs, 其抖动为 0.35 μs. 这与端到端传输延时的理论值保持一致.

上述关于端到端传输延时的测试结果可证明, nPSA 架构通过降低 TSN 芯片的最大交换延时, 以允许更小的时间槽, 进而实现微秒级或数十微秒级的端到端传输延时.

6 相关工作

由于 TSN 标准仅在逻辑层面描述 TSN 技术, 未对实现相关的交换架构进行规定. 因此 TSN 芯片采用的交换架构是可以自定义的. 目前商用 TSN 交换芯片(如 BROADCOM 公司的 BCM53154, BCM53156, BCM53158 芯片[6])沿用以太网交换芯片中的 1PSA 架构, 其最大交换延时高的特点使其难以适用于对实时性具有苛刻要求的 TSN 应用.

优化 1PSA 交换架构的目标和以太网芯片中使用直通式交换架构替代存储转发式交换架构的目标不同. 直通式交换架构通过消除完整帧写入与读取的过程, 降低芯片的最小交换延时[20-21]. 然而, 本文优化 1PSA 交换架构的目标是最小化最大交换延时.

文献 [22] 提出一种基于交叉开关矩阵(crossbar)的分布式共享缓冲区架构(DSB), 其通过物理隔离的方式避免多端口帧同时访问同一共享缓冲区, 具体为将同时输入的帧映射至不同的缓冲区中进行写入操作, 进而消除访存冲突. 与 nPSA 架构相比, DSB 架构可消除阻塞延时和阻塞缓存. 但是多块缓冲区同时写入数据会使芯片的峰值功耗急剧增加. TSN 芯片在许多场景中用于原味替代现有的以太网或总线芯片. 在这些场景中 TSN 芯片继续使用被替代芯片的电源, 这使得高功耗的 TSN 芯片难以适用于此类场景. 此外由于 ST 帧需要按时发送, 为了支持较长时间的 ST 帧缓存, DSB 的共享缓冲区需要消耗大量珍贵的片上存储资源. 因此基于交叉电路的 DSB 架构不适用于 TSN 芯片.

文献 [23] 对 DSB 架构进行优化, 提出一种面向 TSN 交换芯片的时间触发交换架构(SMS). 该架构对 ST 流量和非 ST 流量使用单独的共享缓冲区, 并且从降低存储资源消耗和提升 ST 帧的错误容忍角度设置访存机制. 这与本文降低最大化交换延时的初衷截然不同, 因此文献 [23] 的访存机制并不能直接用以指导本文访存机制的设计.

7 总 结

本文面向 TSN 芯片针对 1PSA 架构中存在的阻塞延时过大问题, 提出了多流水线交换架构 nPSA. nPSA 架构通过基于时分复用的访存机制保证各流水线超低阻塞地访问集中缓冲区. 另外, 针对不同应用场景对端口多样性的需求, 本文提出了一种加权轮询式时隙分配算法 WRRSA, 能够快速地为各流水线分配访问集中缓冲区的时隙. 最后, 通过理论分析和芯片实测, 本文对 nPSA 架构的性能优点进行了验证, 证明该架构及 WRRSA 算法在降低最大交换延时和减少资源消耗方面具有显著优势.

作者贡献声明: 付文文负责全文框架设计; 刘汝霖负责论文撰写; 全巍负责实验设计; 姜旭艳负责实验实现和分析; 孙志刚提出论文构思.

参 考 文 献

[1] Sanchez-Garrido J, Aparicio B, Ramirez J G, et al. Implementation of a time-sensitive networking (TSN) Ethernet bus for microlaunchers[J]. IEEE Transactions on Aerospace and Electronic Systems, 2021, 57(5): 2743−2758

[2] Yan Jinli, Quan Wei, Jiang Xuyan, et al, Injection time planning: Making CQF practical in time-sensitive networking [C] //Proc of the IEEE Conf on Computer Communications, Piscataway, NJ: IEEE, 2020: 616−625

[3] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Local and Metropolitan Area Networks[S]. Piscataway, NJ: IEEE, 2018

[4] Steiner W. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks [C] //Proc of the 31st IEEE Real-Time Systems Symp. Piscataway, NJ: IEEE, 2010: 375−384

[5] Nasrallah A, Thyagaturu A S, Alharbi Z, et al. Ultra-low latency

(ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research[J]. IEEE Communications Surveys & Tutorials, 2019, 21(1): 88−145

[6] BROADCOM Company. BCM53154/BCM53156/BCM53158 ultra-low power layer2 Gbps switch with 10G uplinks [EB/OL]. (2020-05-06)[2021-11-21]. https://media.digikey.com/pdf/Data%20Sheets/Avago%20PDFs/BCM53154,BCM53156,BCM53158.pdf

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

[8] Sivaraman A, Subramanian S, Alizadeh M, et al. Programmable packet scheduling at line rate [C] //Proc of the 32nd ACM SIGCOMM Conf. New York: ACM, 2016: 44−57

[9] Sun Zhigang. openTSN [CP/OL]. [2021-11-22]. https://gitee.com/opentsn/

[10] Quan Wei, Fu Wenwen, Yan Jinli, et al. OpenTSN: An open-source project for time-sensitive networking system development[J]. CCF Transactions on Networking, 2020, 3(9): 51−65

[11] Quan Wei, Fu Wenwen, Sun Zhigang, et al. HX-DS09: A low-power time-sensitive network chip customized for high-end equipment[J]. Journal of Computer Research and Development, 2021, 58(6): 1242−1245 (in Chinese)
(全巍, 付文文, 孙志刚等. 枫林一号: 一款面向高端装备定制的低功耗时间敏感网络芯片[J]. 计算机研究与发展, 2021, 58(6): 1242−1245)

[12] Mckeown N, Anderson T, Balakrishnan H. OpenFlow: Enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69−74

[13] Craciunas S S, Oliver R S, Chmelik M, et al. Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks [C] //Proc of the 24th Int Conf on Real-time Networks & Systems. New York: ACM, 2016: 183−192

[14] Hanphil L, Juho L, Chulsun P, et al. Time-aware preemption to enhance the performance of audio/video bridging(AVB) in IEEE 802.1 TSN [C] //Proc of the 1st IEEE Int Conf on Computer Communication and the Internet. Piscataway, NJ: IEEE, 2016: 99−103

[15] Geng Yilong, Liu Shiyu, Yin Zi. Exploiting a natural network effect for scalable, fine-grained clock synchronization [C] //Proc of the 15th USENIX Conf on Networked Systems Design and Implementation. New York: ACM, 2018: 81−94

[16] Shrivastav V, Lee K S, Wang Han, et al. Globally synchronized time via datacenter networks [C] //Proc of the 32nd ACM SIGCOMM Conf. New York: ACM, 2016: 454–467

[17] Li Tao, Sun Zhigang, Chen Yijiao, et al. A new message processing model for the next-generation internet experimental platform—EasySwitch[J]. Chinese Journal of Computers, 2011, 34(11): 2187−2196 (in Chinese)
(李韬, 孙志刚, 陈一骄, 等. 面向下一代互联网实验平台的新型报文处理模型——EasySwitch[J]. 计算机学报, 2011, 34(11): 2187−2196)

[18] Zhang Yingyue. Research on low power consumption technology based on microprocessor chip [D]. Xiangtan: Xiangtan University, 2020 (in Chinese)
(张莹月. 基于微处理器芯片的低功耗技术研究[D]. 湘潭: 湘潭大学, 2020)

[19] Wang Yang. Research on high-speed exchange structure based on shared storage [D]. Wuhan: Huazhong University of Science and Technology, 2008 (in chinese)
(汪洋. 基于共享存储的高速交换结构研究[D]. 武汉: 华中科技大学, 2008)

[20] Shin K, Choi S, Kim H. Flit scheduling for cut-through switching: Towards near-zero end-to-end latency[J], IEEE Access, 2019, 7: 66369−66383

[21] Dolter J W, Ramanathan P, Shin K G. Performance analysis of virtual cut-through switching in HARTS: A hexagonal mesh multicomputer[J]. IEEE Transactions on Computers, 1991, 40(6): 669−680

[22] Soteriou V, Ramanujam R S, Peh L S, et al. Design of a high-throughput distributed shared-buffer NoC router [C] //Proc of the 4th ACM/IEEE Int Symp on Networks-on-Chip. New York: ACM, 2010: 69−78

[23] Li Zonghui, Wan Hai, Zhao Xibin, et al. Time-triggered switch-memory-switch architecture for time-sensitive networking switches[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(1): 185−198

Fu Wenwen, born in 1994. PhD, assistant professor. His main research interests include programmable network and time-sensitive network.
付文文, 1994 年生. 博士, 助理研究员. 主要研究方向为可编程网络和时间敏感网络.

Liu Rulin, born in 1989. PhD, assistant professor. His main research interests include network architecture and related IC design.
刘汝霖, 1989 年生. 博士, 助理研究员. 主要研究方向为网络体系结构和相关集成电路设计.

Quan Wei, born in 1987. PhD, associate professor. His main research interests include time-sensitive network, software defined network and FPGA design.
全　巍, 1987 年生. 博士, 副研究员. 主要研究方向为时间敏感网络、软件定义网络和 FPGA 设计.

Jiang Xuyan, born in 1998.PhD candidate. Her main research interests include traffic planning and task scheduling in time-sensitive network.
姜旭艳, 1998 年生. 博士研究生. 主要研究方向为时间敏感网络中的流量规划和任务调度.

Sun Zhigang, born in 1974. PhD, professor. His main research interests include software defined network, time-sensitive network, network architecture, FPGA design and network security.
孙志刚, 1974 年. 博士, 研究员. 主要研究方向为软件定义网络、时间敏感网络、网络体系架构、FPGA 设计以及网络安全.

附录 A.

定理 A1. 在连续工作模式公平调度算法下，帧在输入队列中的阻塞延时(头进头出的延时) $D_{\mathrm{FRR}} < (N\times L)/(f\times B-(V_{\mathrm{total}}-V_{\mathrm{min}}))$.

证明. 在任何 $[T_0, T_1]$ 连续调度周期中，调度输出的速率不小于各队列的输入速率之和，因此在时间区间 $[T_0, T_1]$ 内，T_0时各队列存储的数据量最大，最大不超过 $N\times L$. 否则，在T_0之前的某个时刻必有某个输入队列的长度大于L，即至少积累一个完整报文，这与T_0才开始调度是矛盾的.

在调度周期的任何时刻，$N-1$ 个输入队列中原有的数据量加上到达的数据量大于调度输出的数据量，因此，在时间段 $[T_0, T_1]$ 连续调度中，报文 p在时刻t_1开始进入端口i的输入队列，其头进头出的延时 D_{FRR}满足

$$D_{\mathrm{FRR}}\times f\times B < TD(t_1)+D_{\mathrm{FRR}}\times(V_{\mathrm{total}}-V_i) < TD(t_1)+D_{\mathrm{FRR}}\times(V_{\mathrm{total}}-V_{\mathrm{min}}),$$

$TD(t_1)$为时刻 t_1各队列累计的数据量，即 $TD(t_1)\leqslant N\times L$, 因此可得$D_{\mathrm{FRR}} < (N\times L)/(f\times B-(V_{\mathrm{total}}-V_{\mathrm{min}}))$.

证毕.

定理 A2. 在连续工作模式公平调度算法下，当端口i输入队列容量$C_i \geqslant (N\times L\times V_i)/f\times B-(V_{\mathrm{total}}-V_i)$时，帧在端口$i$输入队列中不溢出.

证明. 采用反证法. 假设在时刻T_1调度器开始调度端口i输入队列，在此前调度器最后一次结束调度时该输入队列的时刻为T_0, 即在区间 $[T_0, T_1]$ 区间，端口i输入队列无调度输出. 又假设时刻t_1帧在端口i输入队列中溢出. 由于调度器的调度输出速率大于端口i的输入速率，因此端口i输入队列在调度输出期间，队列中的数据持续减少，不会溢出，进而可知 $t_1\in[T_0, T_1]$. 帧在时刻 t_1溢出，可知 $C_i < TD(T_0)+(t_1-T_0)\times V_i$. 由于时刻$T_0$调度器对端口$i$输入队列的调度结束，时刻$T_0$端口$i$输入队列中最多存储一个不完整的帧$p$. 设帧 p的报文长度为 $l(p)$, 可知 $TD(T_0) < l(p)$, 即$C_i < l(p)+(t_1-T_0)\times V_i$.

此时帧p的最大阻塞延时为$l(p)/V_i+(T_1-T_0)$. 根据定理 A1, 可知 $l(p)/V_i+(T_1-T_0) < (N\times L)/(f\times B-(V_{\mathrm{total}}-V_i))$, 进而可知$C_i < l(p)+(T_1-T_0)\times V_i < (N\times L\times V_i)/f\times B-(V_{\mathrm{total}}-V_i)$, 与假设矛盾，定理 A2 得证.

证毕.

特性 1 证明. 为了使能各流水线线速访存，缓冲区管理逻辑的的处理速率V_{b}需要满足式(A1)：

$$V_{\mathrm{b}} \geqslant \sum_{i=0}^{N-1} V[i]. \tag{A1}$$

由于轮询周期为sn_{period}个时隙，流水线i在轮询周期内分配的时隙数量为$sn[i]$, 可根据式(A2)计算流水线i分配的访存带宽$W[i]$:

$$W[i] = V_{\mathrm{b}}\times\frac{sn[i]}{sn_{\mathrm{period}}}. \tag{A2}$$

根据加权轮询式时隙分配算法对$sn[i]$的赋值(算法 1 行②)可知式(A3)：

$$\frac{sn[i]}{sn_{\mathrm{period}}} = \frac{V[i]}{\sum_{i=0}^{N-1} V[i]}. \tag{A3}$$

根据式(A2)和式(A3)可得出式(A4)：

$$W[i] = V_{\mathrm{b}}\times\frac{V[i]}{\sum_{i=0}^{N-1} V[i]}. \tag{A4}$$

根据式(A1)和(A4)，可得关系式(A5)：

$$W[i] \geqslant V[i]. \tag{A5}$$

在轮询周期内，流水线i分配的访存带宽大于或等于流水线i的数据输入速率，即帧开始从循环输入寄存器中往集中缓冲区写入后，帧数据以线速写入集中缓冲区中. 因此，帧在输入寄存器中的阻塞延时为第 1 个切片到达寄存器至该切片被调度的延时.

在每个轮询周期内，每条流水线至少都会分配 1 个时隙. 当流水线i在轮询周期num_{period}内只分配到 1 个时隙j, 且流水线i在时隙j第 1 次写满一个循环输入寄存器时，该切片在下一个轮询周期的时隙j被调度，此时帧的阻塞延时最长，为num_{period}.

证毕.

特性 2 证明. 采用反证法. 假设帧数据在流水线i对应的循环输入寄存器中在时刻T_1有溢出，此前最后一次调度该循环输入寄存器的时刻为T_0. 由于寄存器组中寄存器的数量不小于 2, 则时刻T_0循环输入寄存器组中空闲的容量最小为B_{b}(一个寄存器为空). 由于时刻T_1报文在循环输入寄存器中溢出，则时刻T_0与T_1满足关系式(A6)(其中B_i与 f_i分别为流水线i的处理位宽和处理速率，B_{b}与 f_{b}分别为缓冲区管理逻辑的处理位宽和处理速率)，即T_1-T_0内写入的数据量超过寄存器组空闲的容量B_{b}, 且T_1-T_0的时间间隔小于等于$\frac{i_{\max}-1}{f_{\mathrm{b}}}$(对寄存器进行同时读写时，可通过编码控制读后写或写后读，因此寄存器中存储的数据已满时，同时读写不会造成帧数据溢出).

$$(T_1-T_0)\times f_i\times B_i > B_{\mathrm{b}},$$

$$T_1 - T_0 \leqslant \frac{i_{\max} - 1}{f_b}. \tag{A6}$$

关系式(A6)可转化成$\frac{B_b}{V[i]} < T_1 - T_0 \leqslant \frac{i_{\max} - 1}{f_b}$. 根据时隙分配算法对时隙序号间隔的约束(算法1行③⑳)且$\left\lceil \frac{V_b}{V[i]} \right\rceil \geqslant \left\lceil \frac{sn_{\text{period}}}{sn[i]} \right\rceil$(根据式(A1)和式(A3)得出),可知调度器从同一循环输入寄存器调度数据的时隙间隔最大值$i_{\max} \leqslant \left\lceil \frac{V_b}{V[i]} \right\rceil$.$i_{\max} \leqslant \left\lceil \frac{V_b}{V[i]} \right\rceil$, 可得出$i_{\max} - 1 < \frac{V_b}{V[i]}$, 进而可得出$\frac{i_{\max} - 1}{f_b} < \frac{B_b}{V[i]}$与式(A6)矛盾,因此假设不成立.

证毕.

特性3证明. 采用反证法. 假设帧数据在流水线i对应的循环输出寄存器中在时刻T_1传输中断,在此前缓冲区数据最后一次输入该循环输出寄存器中的时刻为T_0,则时刻T_0该循环输出寄存器组中的数据至少为B_b.

由于时刻T_1传输中断,则时刻T_0与T_1满足关系式(A6). 即$T_1 - T_0$内输出的数据量超过寄存器组中的数据量B_b,且$T_1 - T_0$的时间间隔小于等于$\frac{i_{\max} - 1}{f_b}$,与关系式(A6)矛盾(具体证明过程同特性2证明),因此假设不成立.

证毕.