CHINA LINUX KERNEL
中国Linux内核开发者大会

華中科技大學
网络空间安全学院
School of Cyber Science and Engineering, HUST

第19届中国 Linux内核开发者大会

赞助单位

- 華中科技大學
- HUAWEI
- 阿里云
- FUJITSU
- 腾讯云
- oppo
- 火山引擎
- vivo
- 龙芯中科 LOONGSON TECHNOLOGY
- 蚂蚁集团 ANT GROUP

支持单位

- 清華大學 Tsinghua University
- 迪捷软件 DIGIPROTO
- OpenEuler
- OpenAnolis 龙蜥社区
- OpenCloudOS

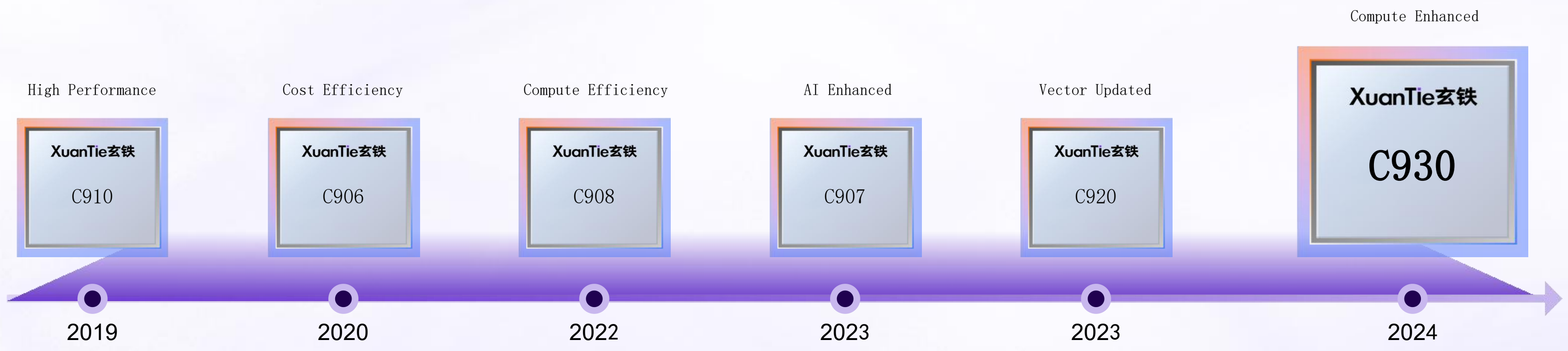
支持社区&媒体

- csdn
- 云巅论剑
- sf 思否
- 阅码场 yomocode
- InfoQ 极客传媒
- 51CTO
- 开源江湖

2024年10月 湖北 • 武汉

玄铁是阿里巴巴达摩院旗下品牌，致力于推动RISC-V架构的前沿研究和建立在该架构基础上的开源生态的建立和发展，是国际 RISC-V 生态引领者，为数字化时代提供强大、智能、安全、开放的新型计算架构。

自成立以来，团队持续深耕 RISC-V 技术研发及生态建设，并陆续推出了一系列玄铁处理器，可满足高中低全系列性能需求。玄铁积极拥抱开源，坚持开放创新，已逐渐构建起以 RISC-V 为核心的生态体系，与生态伙伴协同推动 RISC-V 芯片、开发工具、操作系统、应用解决方案等不同层面的软硬一体化发展，全力推进 RISC-V 软硬全栈技术多领域发展落地，加速实现智能时代的万物互联！



XUANTIE

第19届中国
Linux内核开发者大会

RISC-V IOMMU 的 Scalable 提案 (GIPC) 及 Linux 技术路线

在 IO 虚拟化场景中，处理 PASID 的不同技术流派

郭任

阿里巴巴达摩院　高级技术专家

XUANTIE

目录

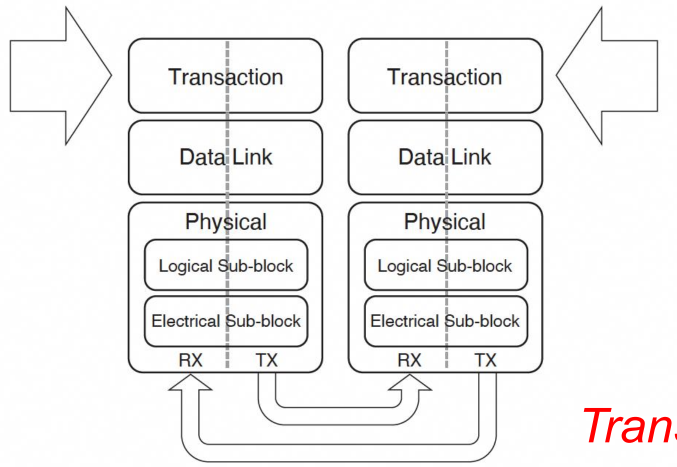
Contents

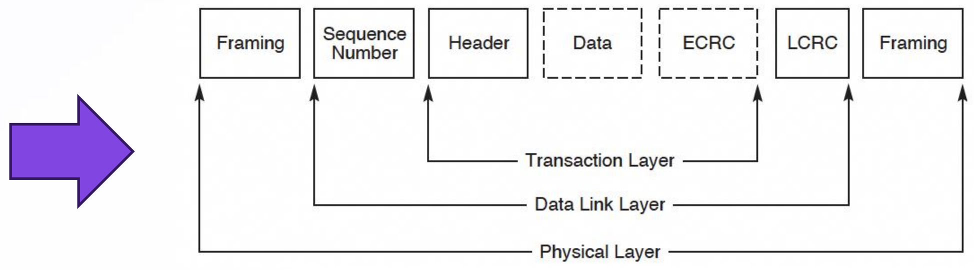
XUANTIE

01 | 什么是 PASID?

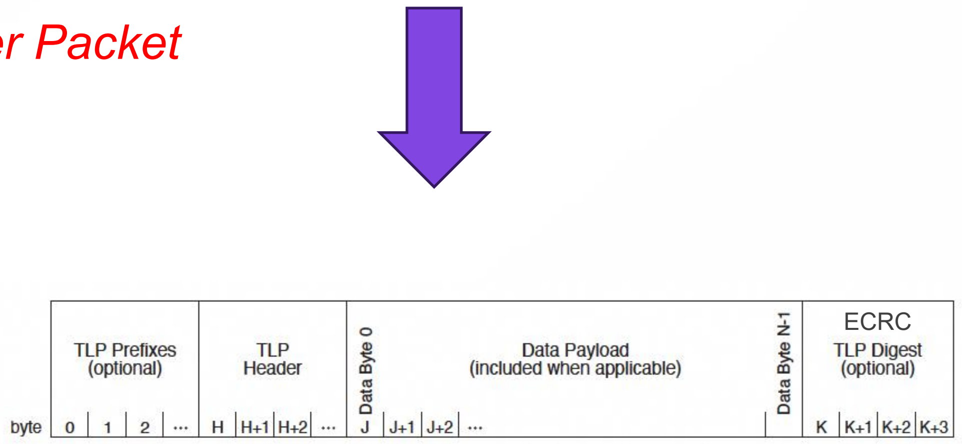
PCI-e TLP - 从 RID 到 PASID

从 PCI-e TLP 开始





Transaction Layer Packet

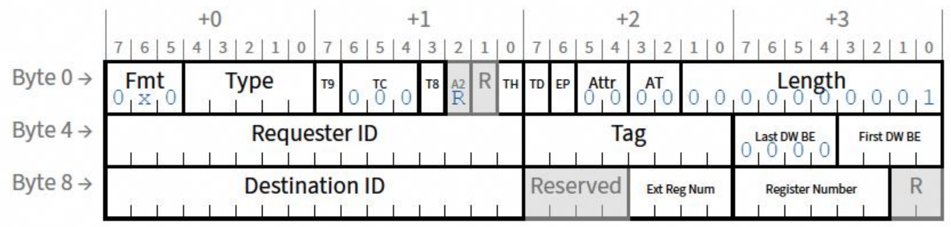


Address Space	Transaction Types	Basic Usage
Memory	Read Request/Completion, Write Request, **Deferrable Write Request/Completion,** AtomicOp Request/Completion	Transfer data to/from a memory-mapped location
I/O	Read Request/Completion, Write Request/Completion	Transfer data to/from an I/O-mapped location
Configuration	Read Request/Completion, Write Request/Completion	Device Function enumerating
Message	Baseline (including Vendor-Defined)	MSI IRQ

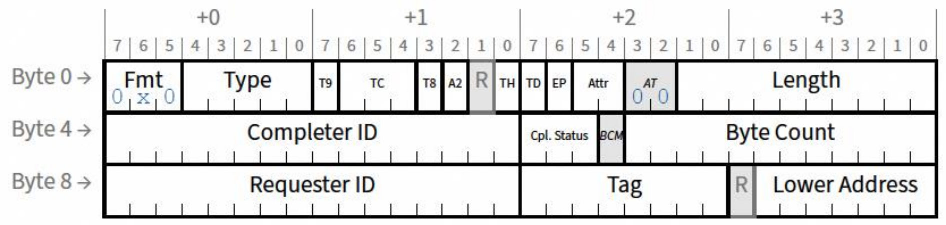
- **TLP Prefixes 可选**
- **TLP Header 必选**
- Data Payload 可选
- ECRC/Digest 可选

Header: Routing and Addressing Rules

ID routing is used with **Configuration Requests**, with **ID Routed Messages**, and with **Completions**.

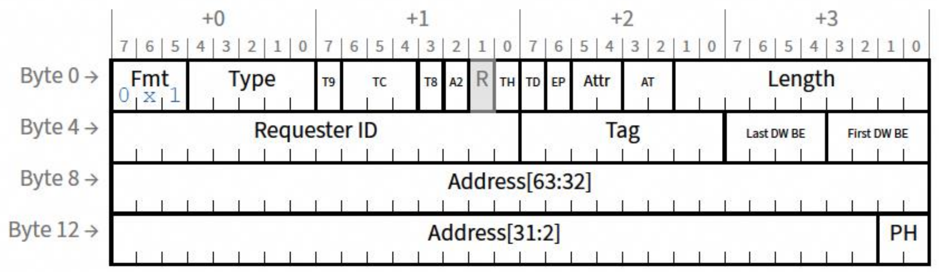


Header Format for Configuration Transactions

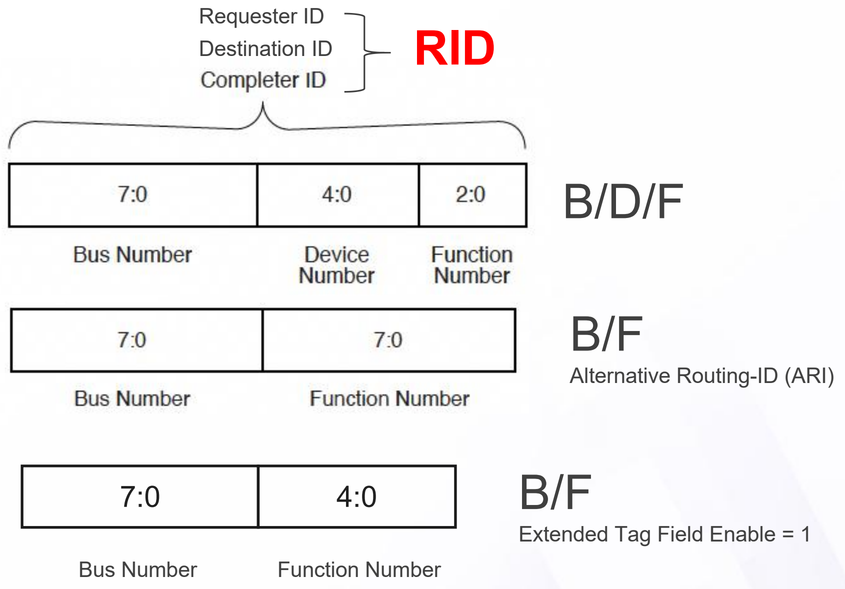


Header Format for Completion Transactions

(ID +) Address routing is used with **Memory, I/O Requests:**



Header Format for Memory Request Transactions



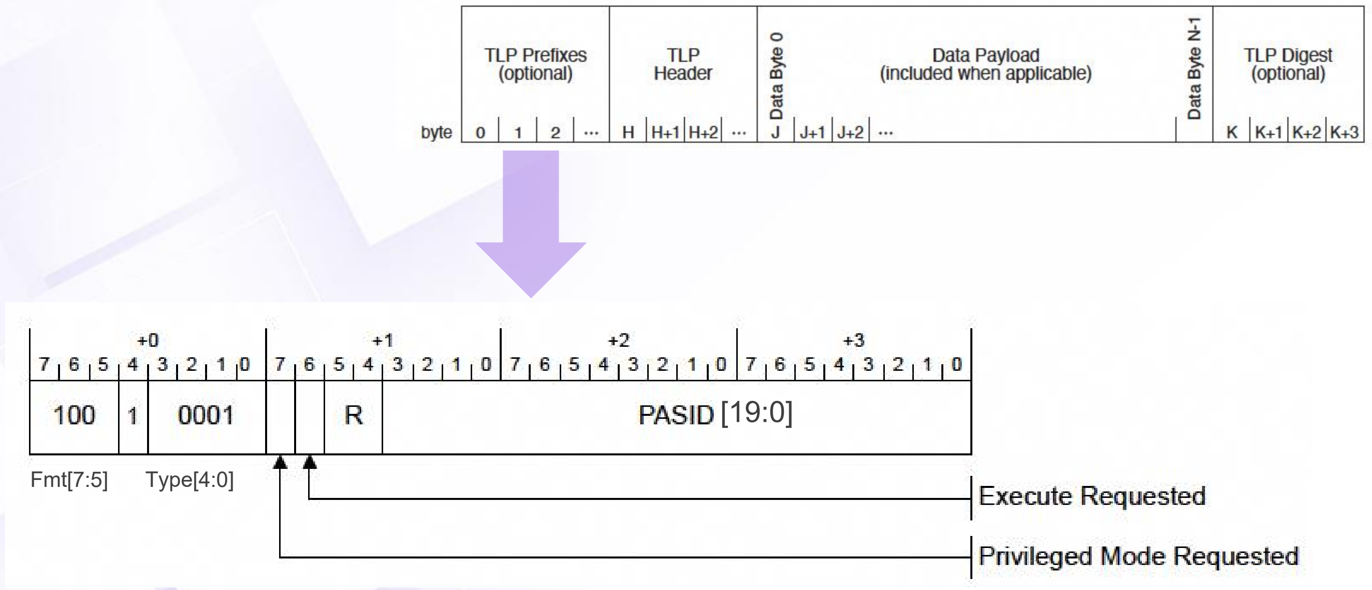
RID 的限制

- Function 是 PCI-e 基于 RID 路由与寻址方式的最小设备单位。
- RID[16]: 最多 256 条总线，每条总线最多 256 个 Functions。
- 当 Extended Tag Field Enable = 1 时，位宽缩减 RID[13]。
- Function 在总线枚举阶段就已确定，每个 PF/VF 必须占用配置空间。
- 基于上述 Function 枚举原则的 SR-IOV (2006) 虚拟设备的成本高，且数量受限。

因此，基于 TLP Requester-ID 的 Scalable 方案正在起草：DRAFT PCI-SIG ECR - Scalable I/O Virtualiztion (by ArchProto WG, AMD, Ampere, Arm, Broadcom, Intel, Microsoft, NVIDIA, RedHat/IBM) (不兼容 SR-IOV)
https://members.pcisig.com/wg/PCIe-Protocol/document/21183

PASID TLP Prefix (Non-Flit Mode)

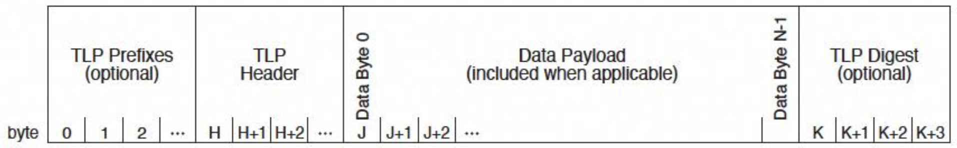
PASID 结合 RID 用于识别请求的地址空间：
- 带有地址类型的 TLP 请求。
- ATS 请求 (MRd with AT=01b), ATS INV, ATS PRI 相关 Messages。

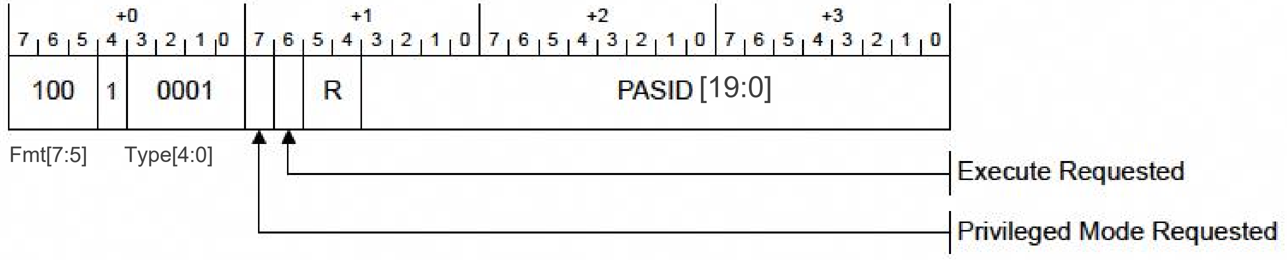


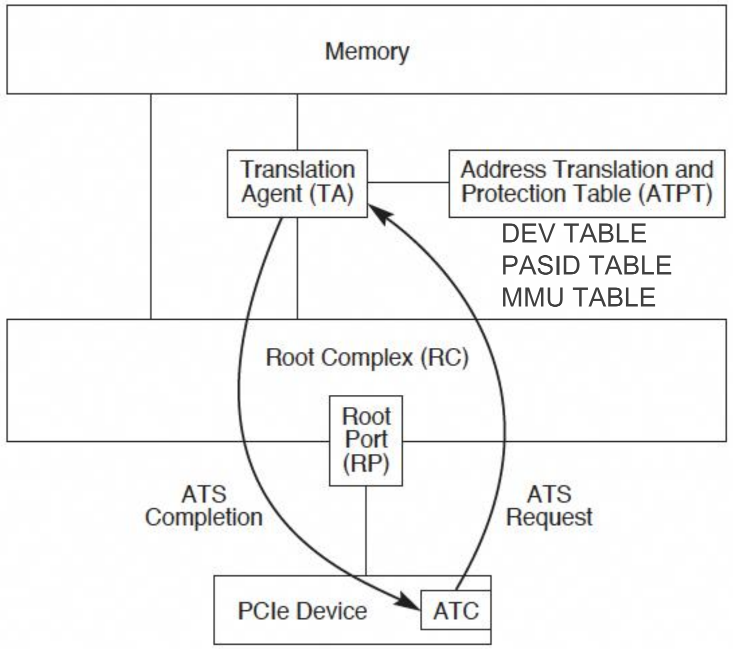
PASID TLP Prefix (Non-Flit Mode)

PASID 结合 RID 用于识别请求的地址空间：

- 带有地址类型的 TLP 请求。
- ATS 请求 (MRd with AT=01b), ATS INV, ATS PRI 相关 Messages。



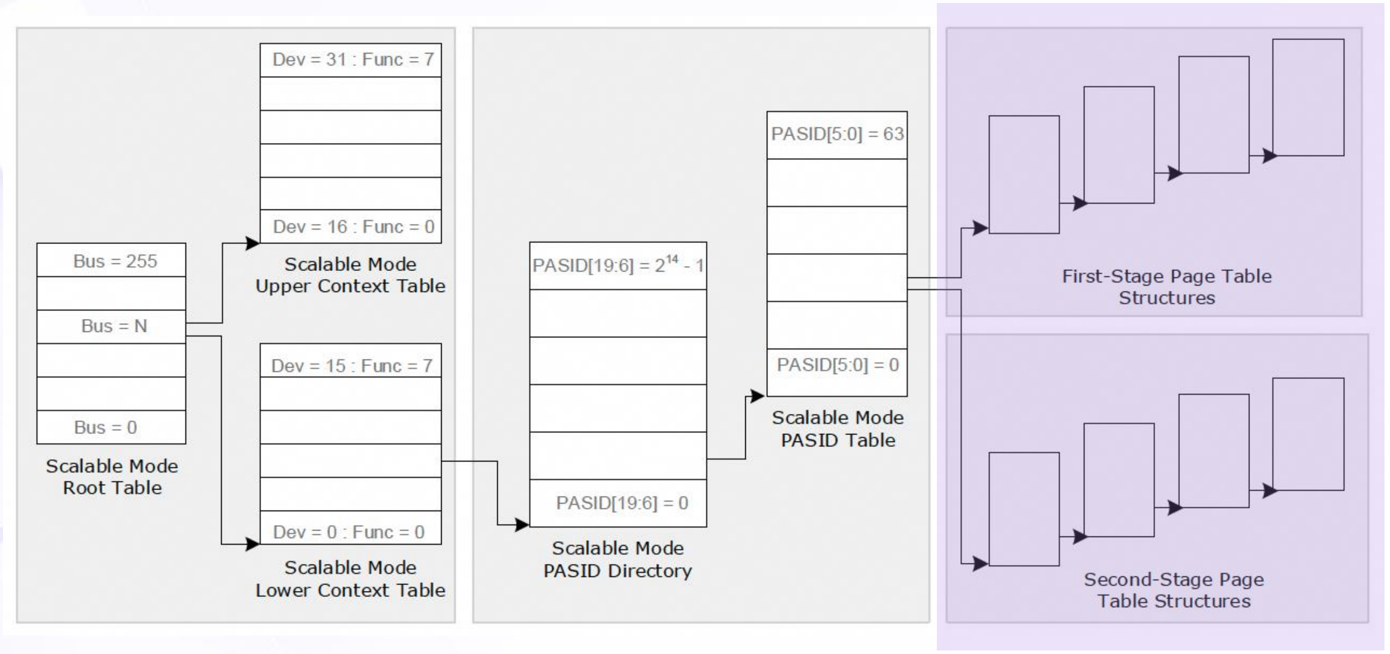




各家 TA 对 PASID 的处理形成两大技术流派：

1. Intel VT-d 的 Scalable Mode 方案
2. ARM-SMMU 和 AMD-Vi 的方案

When TA = Intel VT-d **Scalable Mode**



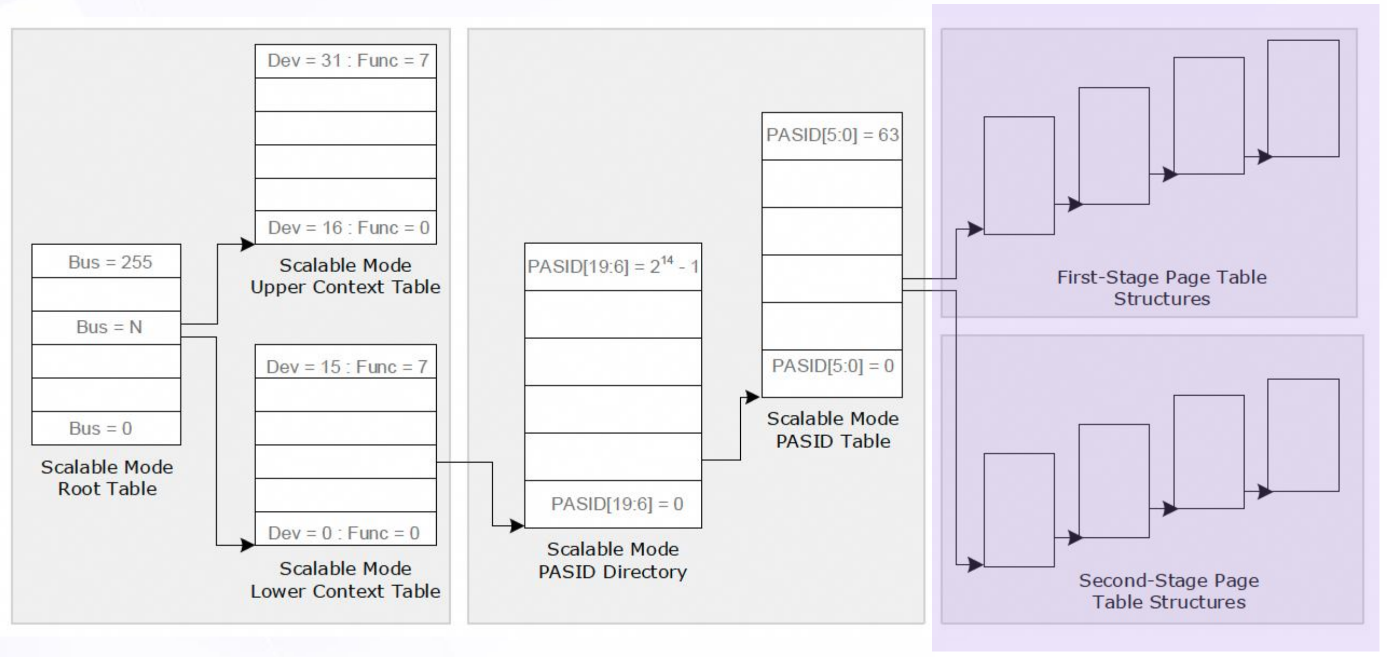
Intel VT-d **Scalable Mode** Address Translation

June 2018, Intel VT-d Revision 3.0:

- Added support for scalable-mode translation for DMA Remapping, that enables PASID-granular first-level, second-level, nested and pass-through translation functions.

每一个 PASID Entry 拥有独立的 First-Stage Page Table 和 Second-Stage Page Table，拥有完整的两级地址空间。

When TA = Intel VT-d **Scalable Mode**



Intel VT-d **Scalable Mode** Address Translation

June 2018, Intel VT-d Revision 3.0:
- Added support for scalable-mode translation for DMA Remapping, that enables PASID-granular first-level, second-level, nested and pass-through translation functions.

OPEN Compute Project

Scalable I/O Virtualization Revision 1.0

Version 1.3

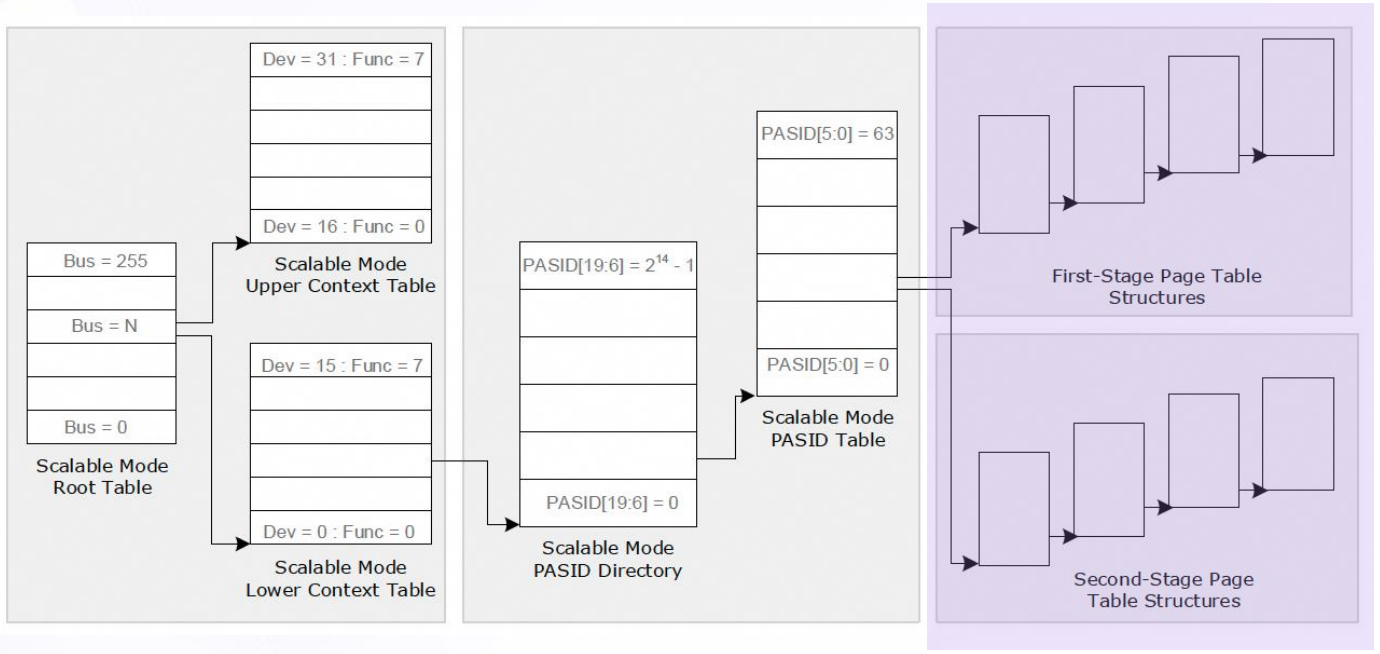
June 2022

Authors:
Intel Corporation (Contact Point: Tom Stachura)
Microsoft Corporation (Contact Point: Renee L'Heureux)

June 2018: v1.0 by Intel
June 2022: v1.3 by OCP

PASID based ADI (Assignable Device Interfaces) 兼容 SR-IOV

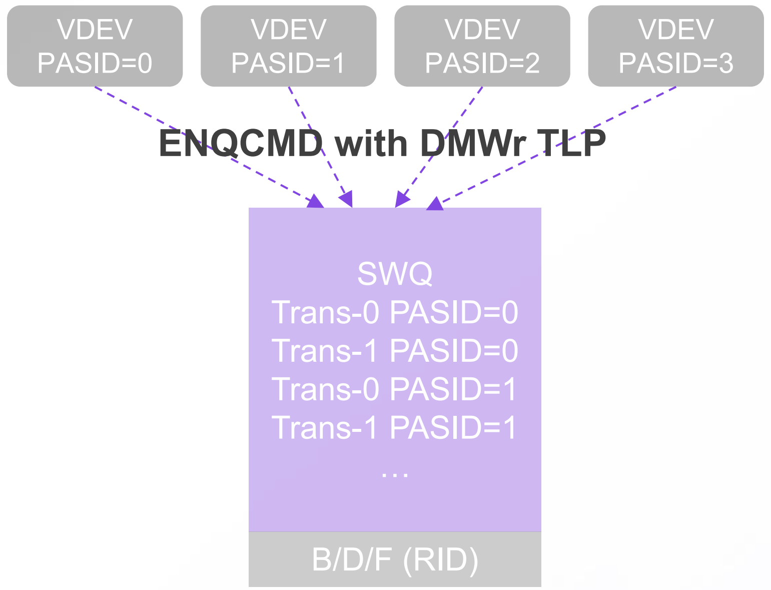
When TA = Intel VT-d **Scalable Mode**



Intel VT-d **Scalable Mode** Address Translation

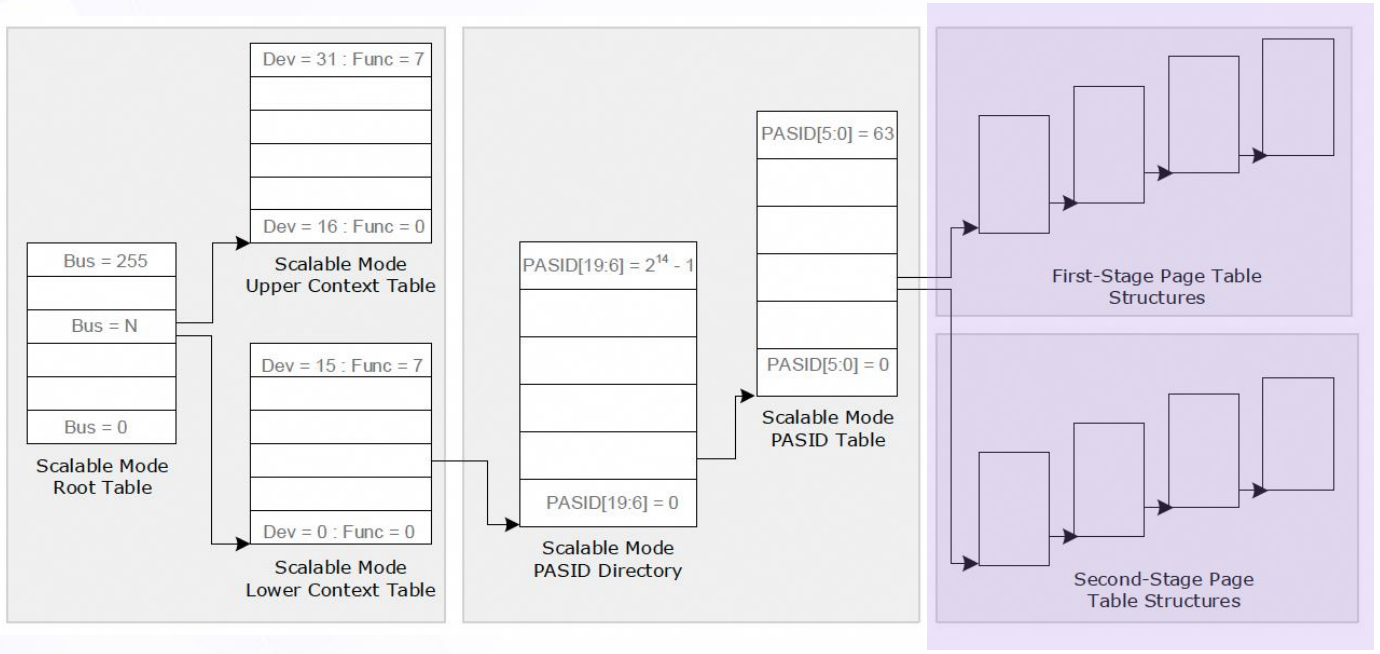
June 2018, Intel VT-d Revision 3.0:
- Added support for scalable-mode translation for DMA Remapping, that enables PASID-granular first-level, second-level, nested and pass-through translation functions.

Scalable I/O Virtualization Spec:
7.3 ADIs Using Shared Work Queues



VT-d Scalable Mode 让同一个 Func 中的 VDEVs 可以自由分配给属于不同虚机的进程和容器。

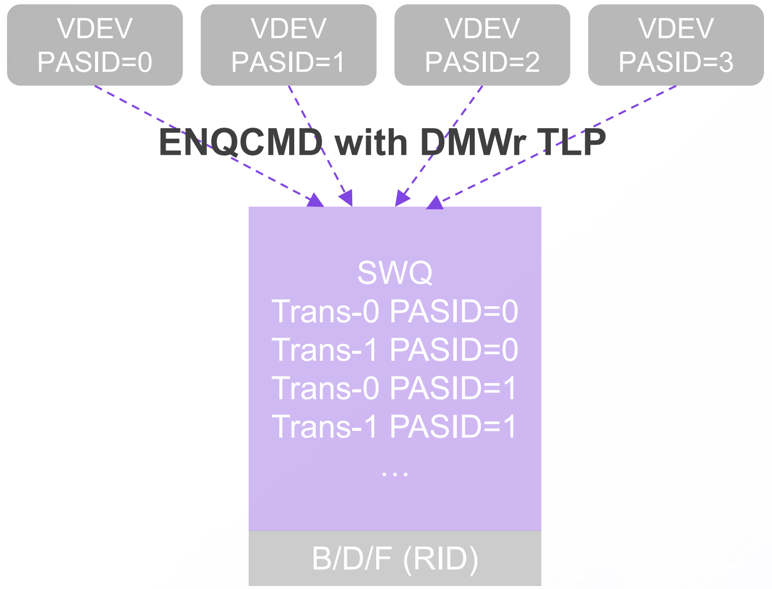
When TA = Intel VT-d **Scalable Mode**



Intel VT-d **Scalable Mode** Address Translation

June 2018, Intel VT-d Revision 3.0:
- Added support for scalable-mode translation for DMA Remapping, that enables PASID-granular first-level, second-level, nested and pass-through translation functions.

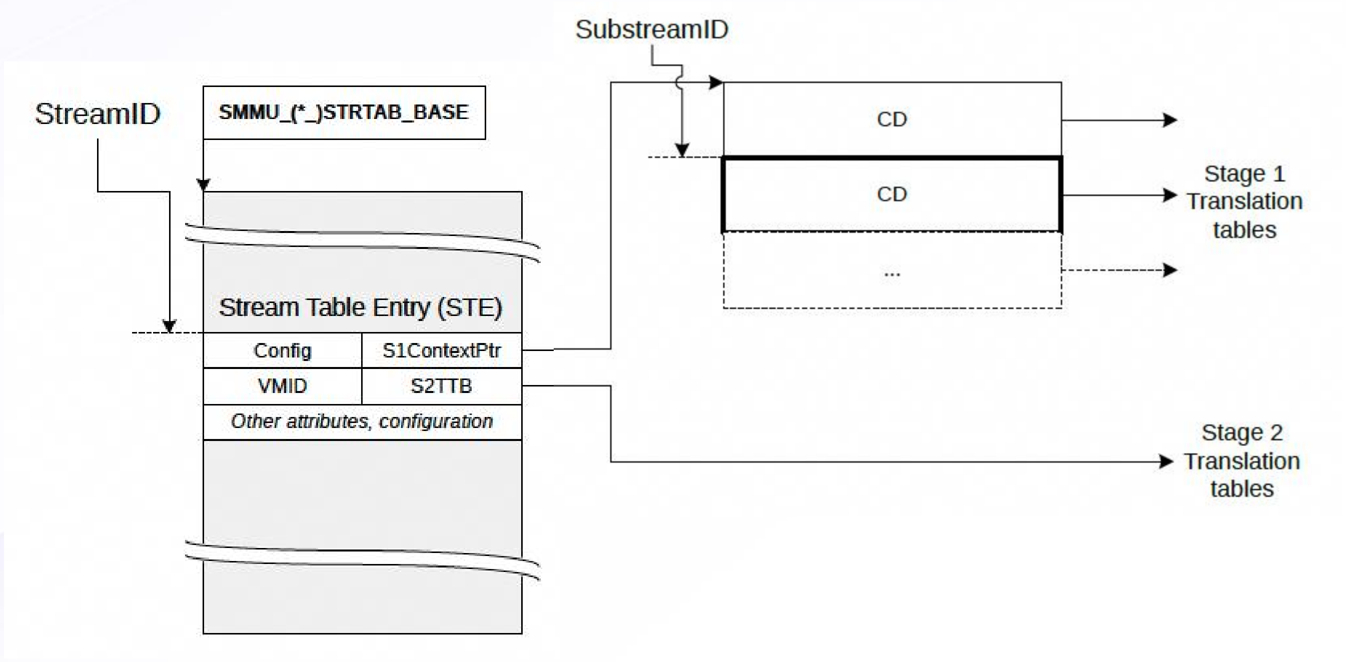
Scalable I/O Virtualization Spec:
7.3 ADIs Using Shared Work Queues



VT-d Scalable Mode 让同一个 Func 中的 VDEVs 可以自由分配给属于不同虚机的进程和容器。

对于同一个 Function 中的 VDEVs 进行跨虚机分配，ARM 和 AMD 有不同的见解 …

When TA = ARM SMMUv3



StreamID Table	-> PCI-e RID (B/D/F) index, Root Table
SubstreamID Table	-> PCI-e PASID Directory

整张 SubstreamID (PASID) 表的 GPA-> PA 翻译，被绑定在 STE.S2TTB 上，因此**无法跨虚拟机分配 PASID**。

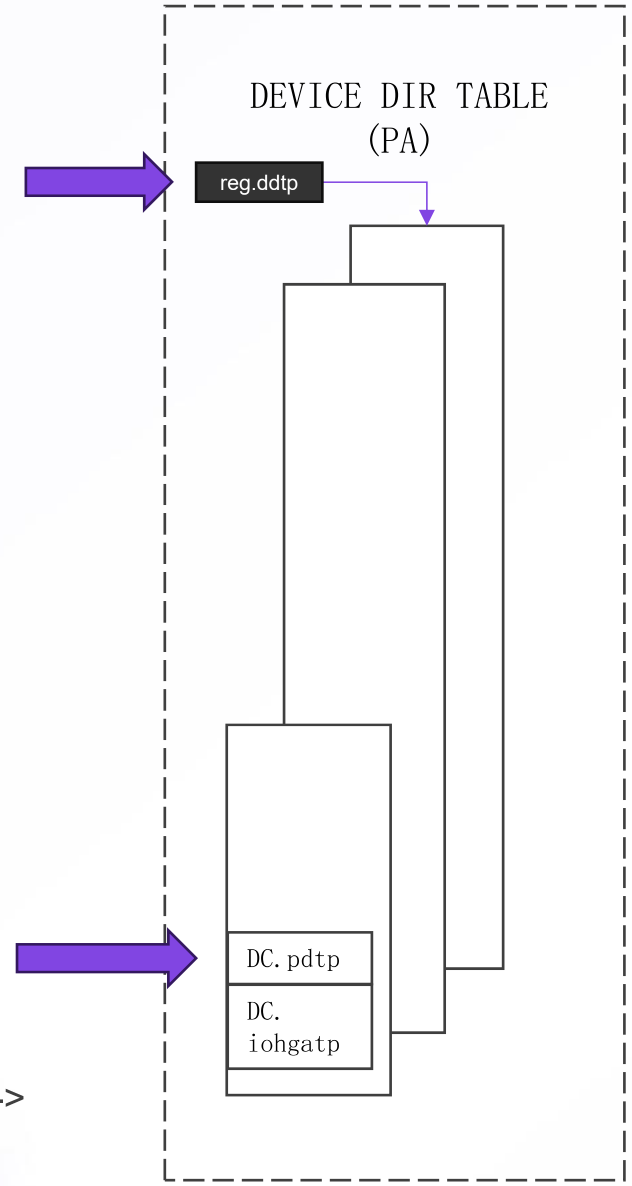
XUANTIE

02 | Scalable Mode 提案 （RISC-V）

GIPC - Guest page table In Process Context

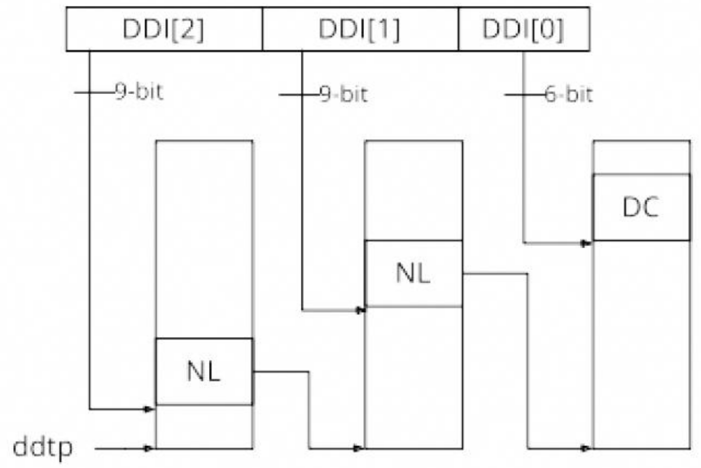
RISC-V IOMMU 表项设计

The PPN of the root device-directory-table is held in a memory-mapped register called the device-directory-table pointer (ddtp):



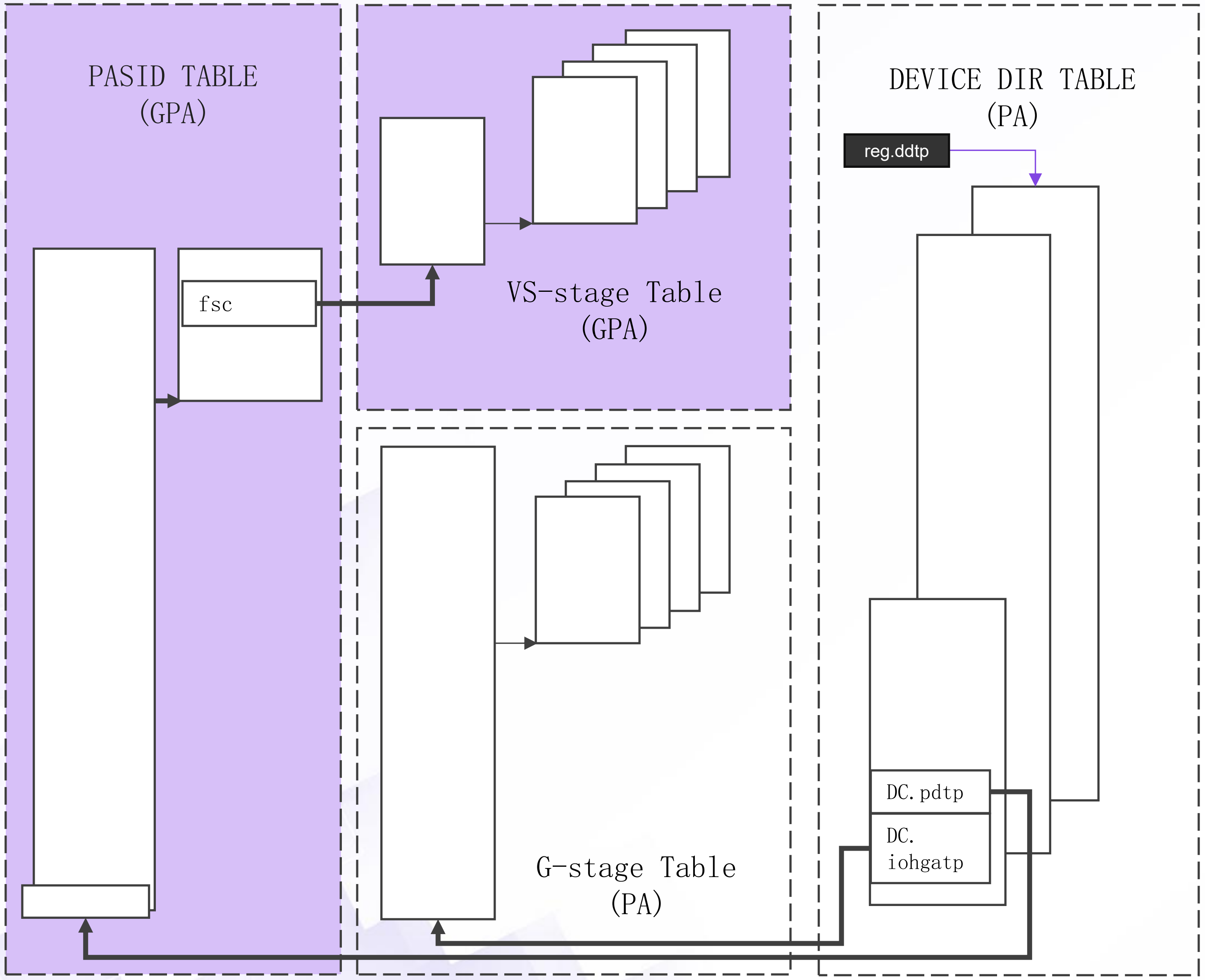
每一个 PCI-e Function，对应 Device-Directory-Talbe (DDT) 中的一个 Device-Context (DC):

- Process-directory table pointer (pdtp) - PASID Table
- IO hypervisor guest address translation and protection (iohgatp) - Guest Page Table (GPA -> PA)



Three-level device directory with extended format DC.

RISC-V IOMMU 表项设计



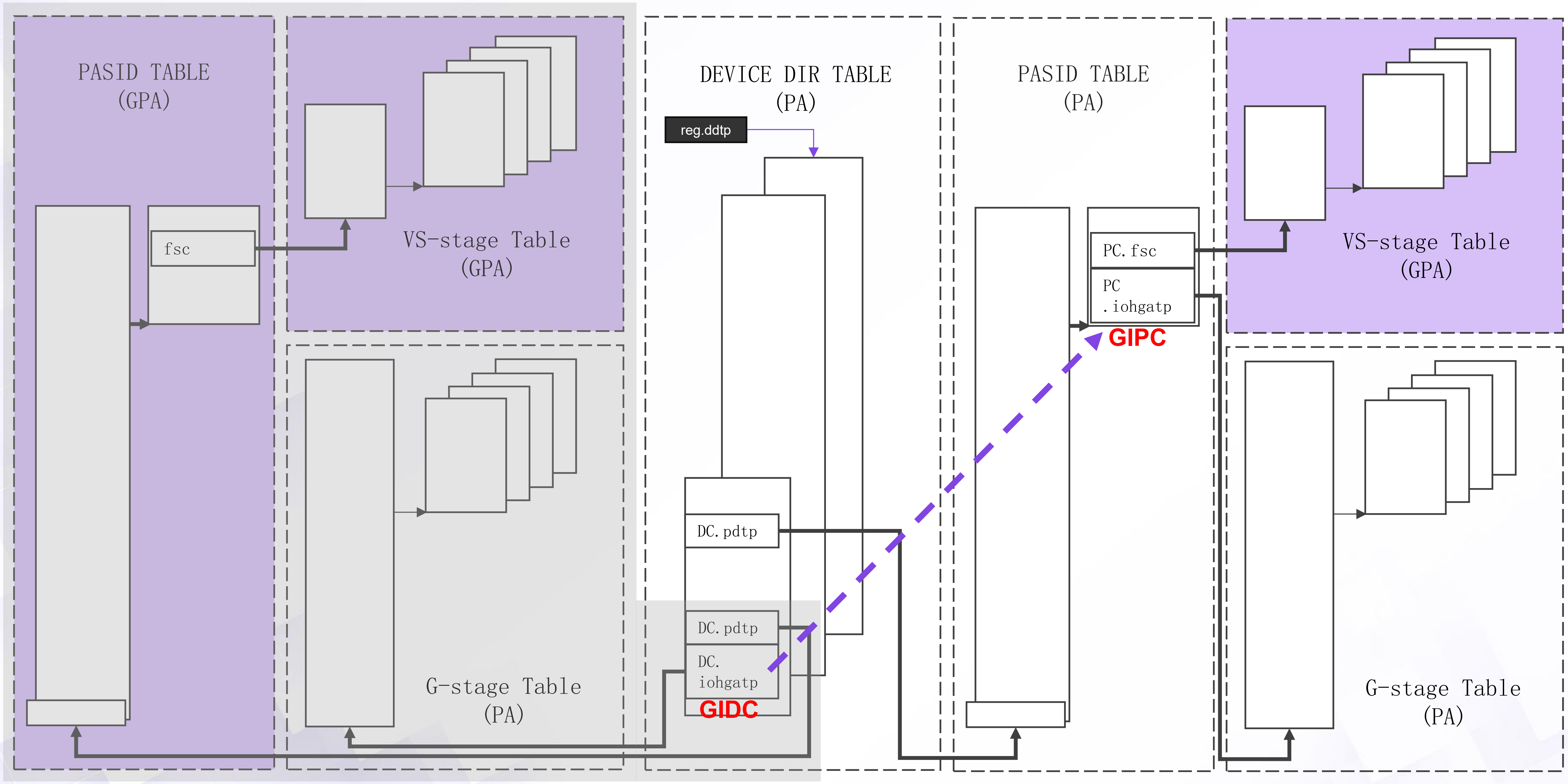
类似 ARM-SMMU & AMD-Vi:

- 客户机管理整张 PASID 表
- PASID 表基于 GPA 寻址
- IOMMU 遍历 PASID 表需要类似 VS-stage 页表的二级地址翻译。

GIDC - Guest page table In Device Context

GIPC - Guest page table In Process Context

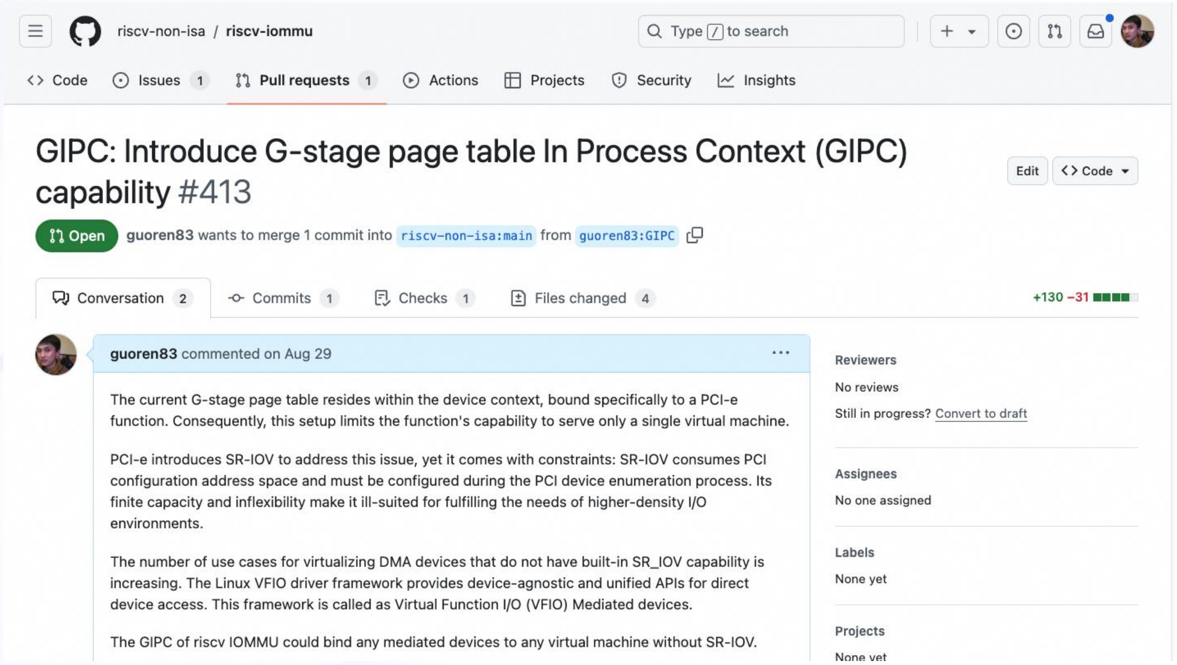
XUANTIE



RISC-V Scalable 提案：GIPC

XUANTIE

- 达摩院玄铁团队参与 RISC-V IOMMU Architecture Specification 初版制定，被列入贡献者名单。
- 如今，为满足高密 IO 场景需求，再次提出 Scalable 模式，以完善 RISC-V IOMMU 功能。



GIPC: Introduce G-stage page table In Process Context (GIPC) capability #413

Open guoren83 wants to merge 1 commit into riscv-non-isa:main from guoren83:GIPC

guoren83 commented on Aug 29

The current G-stage page table resides within the device context, bound specifically to a PCI-e function. Consequently, this setup limits the function's capability to serve only a single virtual machine.

PCI-e introduces SR-IOV to address this issue, yet it comes with constraints: SR-IOV consumes PCI configuration address space and must be configured during the PCI device enumeration process. Its finite capacity and inflexibility make it ill-suited for fulfilling the needs of higher-density I/O environments.

The number of use cases for virtualizing DMA devices that do not have built-in SR_IOV capability is increasing. The Linux VFIO driver framework provides device-agnostic and unified APIs for direct device access. This framework is called as Virtual Function I/O (VFIO) Mediated devices.

The GIPC of riscv IOMMU could bind any mediated devices to any virtual machine without SR-IOV.

Reviewers
No reviews
Still in progress? Convert to draft

Assignees
No one assigned

Labels
None yet

Projects
None yet

https://github.com/riscv-non-isa/riscv-iommu/pull/413

Authors:
Hao Ziyi <haoziyi@zhcomputing.com>
Guo Ren <guoren@linux.alibaba.com>
Song Zhuo <zhuo.song@linux.alibaba.com>
Dust Li <dust.li@linux.alibaba.com>
Shuai Xue <xueshuai@linux.alibaba.com>
Feng Guanghui
<guanghuifeng@linux.alibaba.com>
Kaige Fu <kaige.fu@linux.alibaba.com>
Hao Xiang <hao.xiang@linux.alibaba.com>

XuanTie RISC-V Virtualization Solution

CPU

H Extension
HS/VS/VU
2-stage memory mapping MMU in runtime

vCPU Performance Overhead from core binding < 1%

Interrupt

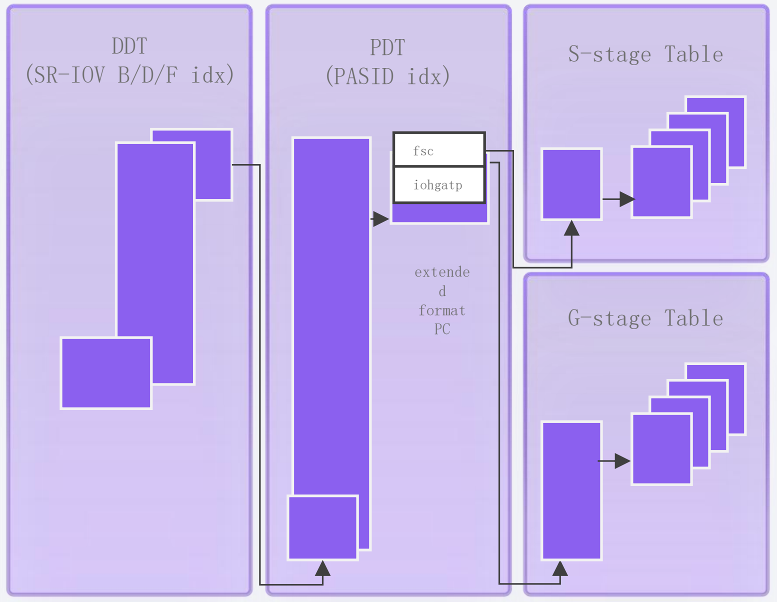
AIA
Hyper CSR Virtual Interrupts
AIA Passthrough Feature

Interrupt Forwarding Overhead in Virtual Machines ~ 0%

IO

RISC-V IOMMU
Hyper passthrough Mechanism

DMA Performance Overhead < 5%



Scalable IOMMU - Extension to Meet the Demands of High-Density I/O Scenarios

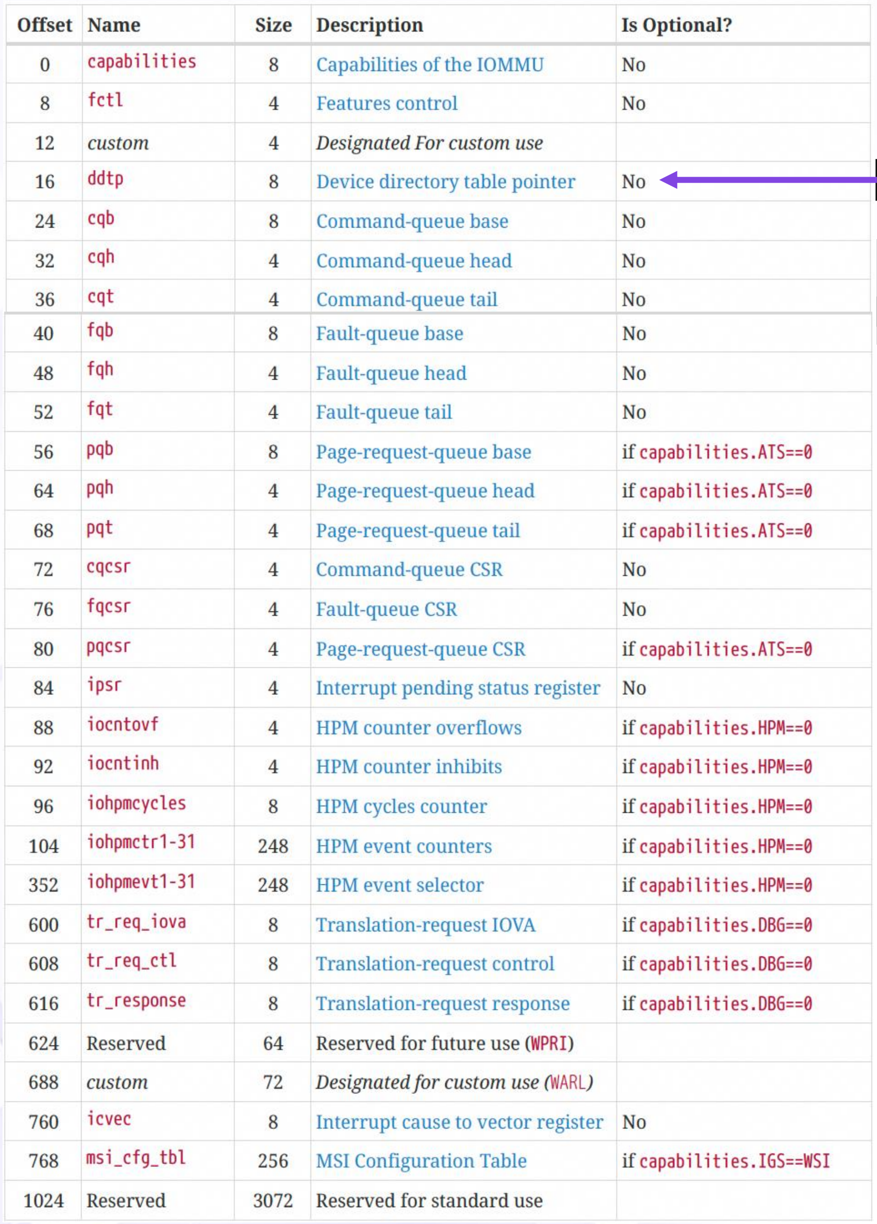
· Implementation of Scalable IOMMU · IRQ-Remapping · RISC-V virtio-IOMMU

XUANTIE

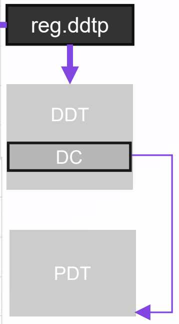
03 Linux 技术路线

RISC-V IOMMU、emulated-IOMMU 和 virtio-IOMMU

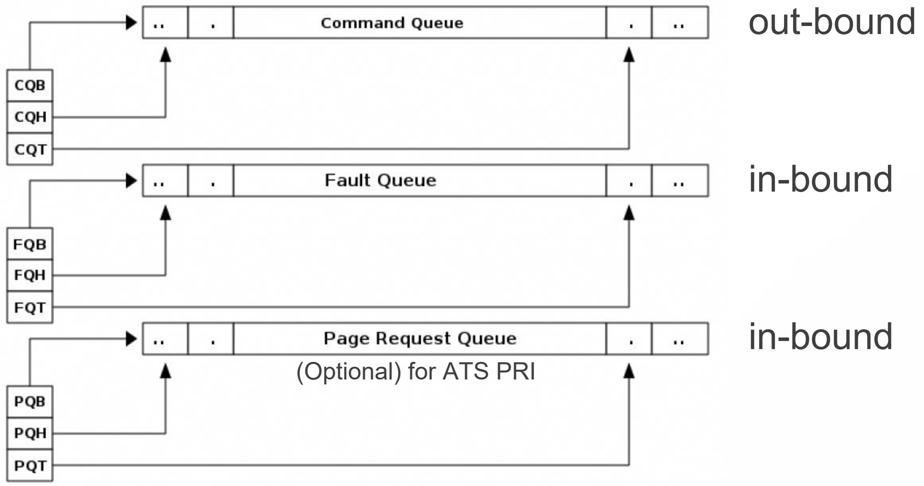
RISC-V IOMMU & Linux Driver



Offset	Name	Size	Description	Is Optional?
0	capabilities	8	Capabilities of the IOMMU	No
8	fctl	4	Features control	No
12	*custom*	4	*Designated For custom use*	
16	ddtp	8	Device directory table pointer	No
24	cqb	8	Command-queue base	No
32	cqh	4	Command-queue head	No
36	cqt	4	Command-queue tail	No
40	fqb	8	Fault-queue base	No
48	fqh	4	Fault-queue head	No
52	fqt	4	Fault-queue tail	No
56	pqb	8	Page-request-queue base	if capabilities.ATS==0
64	pqh	4	Page-request-queue head	if capabilities.ATS==0
68	pqt	4	Page-request-queue tail	if capabilities.ATS==0
72	cqcsr	4	Command-queue CSR	No
76	fqcsr	4	Fault-queue CSR	No
80	pqcsr	4	Page-request-queue CSR	if capabilities.ATS==0
84	ipsr	4	Interrupt pending status register	No
88	iocntovf	4	HPM counter overflows	if capabilities.HPM==0
92	iocntinh	4	HPM counter inhibits	if capabilities.HPM==0
96	iohpmcycles	8	HPM cycles counter	if capabilities.HPM==0
104	iohpmctr1-31	248	HPM event counters	if capabilities.HPM==0
352	iohpmevt1-31	248	HPM event selector	if capabilities.HPM==0
600	tr_req_iova	8	Translation-request IOVA	if capabilities.DBG==0
608	tr_req_ctl	8	Translation-request control	if capabilities.DBG==0
616	tr_response	8	Translation-request response	if capabilities.DBG==0
624	Reserved	64	Reserved for future use (WPRI)	
688	*custom*	72	*Designated for custom use (WARL)*	
760	icvec	8	Interrupt cause to vector register	No
768	msi_cfg_tbl	256	MSI Configuration Table	if capabilities.IGS==WSI
1024	Reserved	3072	Reserved for standard use	



- Data Structures (DDT, DC, PDT, PC …)
- In-memory queue interface (CQ, FQ, PQ)
- Memory-mapped register interface



[PATCH v10 0/7] Linux RISC-V IOMMU Support
by Tomasz Jeznach <tjeznach@rivosinc.com>
https://lore.kernel.org/linux-riscv/cover.1729059707.git.tjeznach@rivosinc.com/

- Support platform device driver
- Support PCIe device driver

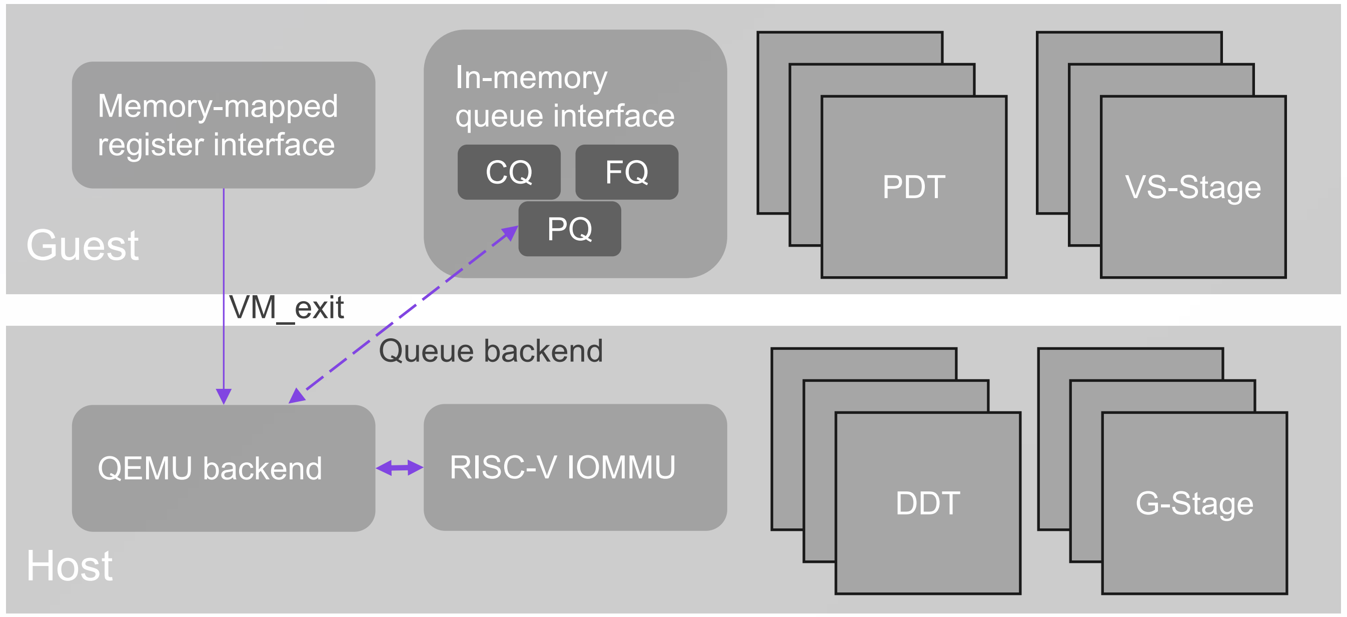
Emulated-IOMMU

- Guest 对 Memory-mapped register 访问，都会陷入 Hypervisor 处理
- QEMU 后端填充/读取 CQ, FQ, PQ。

[RFC PATCH v2 00/10] RISC-V IOMMU HPM and nested IOMMU support
by Zong Li <zong.li@sifive.com>
https://lore.kernel.org/lkml/20240614142156.29420-1-zong.li@sifive.com/

- Support GPA -> SPA (Tested)
- Support nested iommu (Not ready for Test)

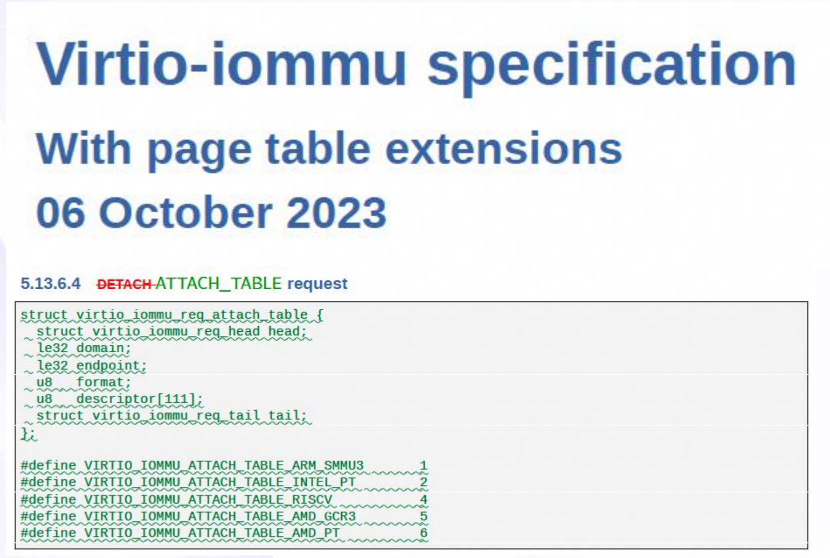
但是，尚未给出 QEMU backend 代码，所以暂时无法测试 nested-translation。



缺点：
- 访问 Memory-mapped 寄存器效率低，产生 VM_exit。
- Guest 复用 RISC-V IOMMU 驱动，Nested 设计增加驱动复杂性。
- 各个架构都要实现自己的 Qemu 后端驱动，工作量大，且不统一。

短期不会支持 GIPC，我们认为有更好的选择：

Virtio-IOMMU



Virtio-iommu specification

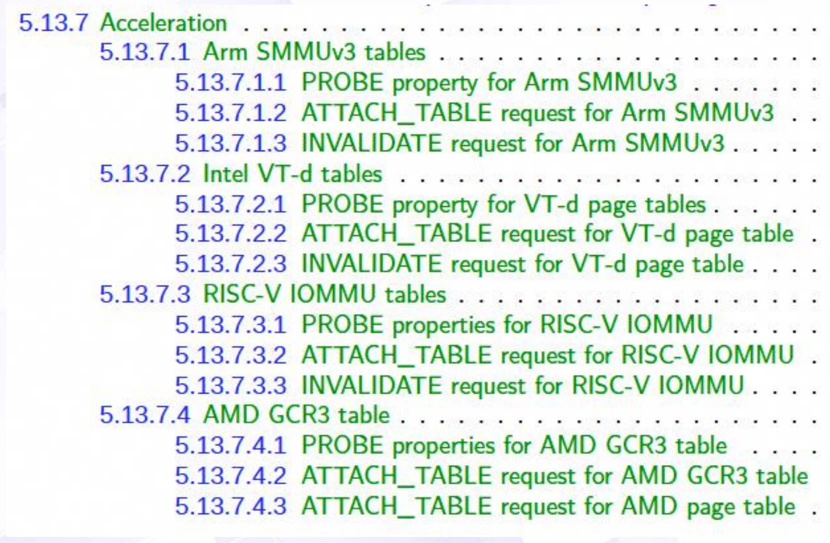
With page table extensions

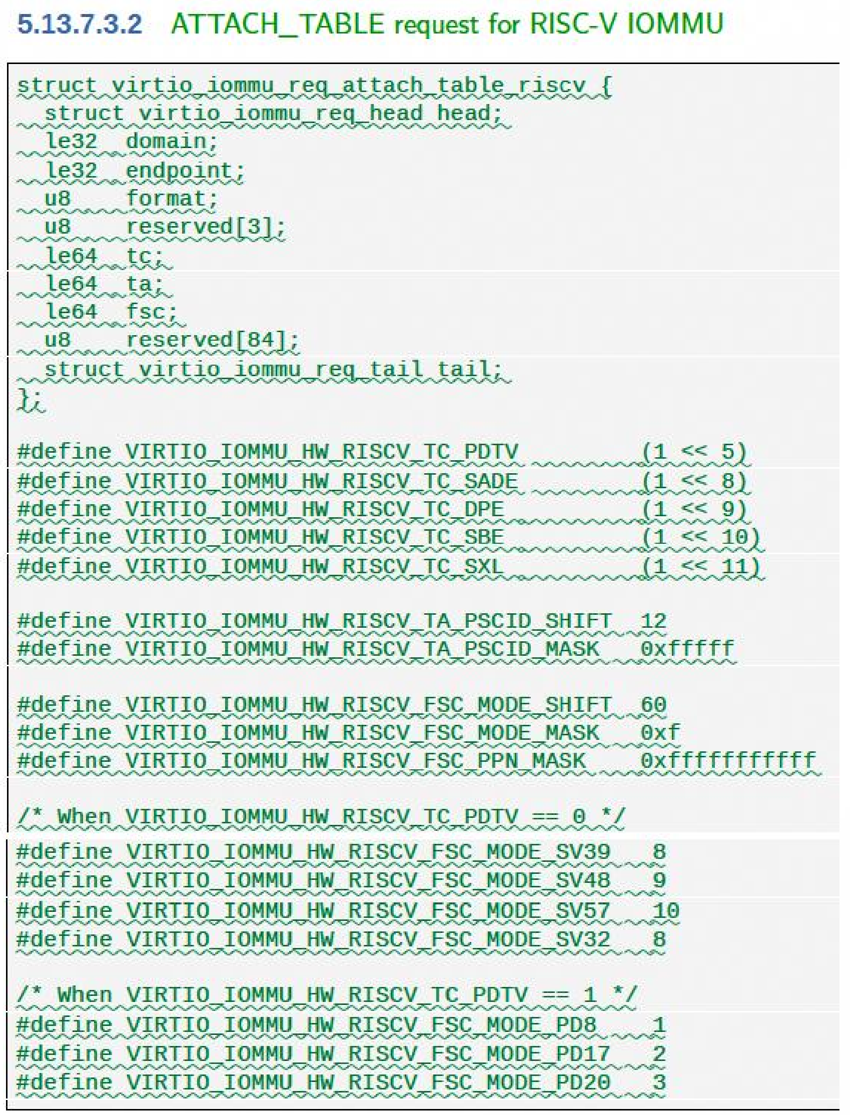
06 October 2023

5.13.6.4 ~~DETACH~~ ATTACH_TABLE request

```
struct virtio_iommu_req_attach_table {
  struct virtio_iommu_req_head head;
  le32 domain;
  le32 endpoint;
  u8   format;
  u8   descriptor[111];
  struct virtio_iommu_req_tail tail;
};

#define VIRTIO_IOMMU_ATTACH_TABLE_ARM_SMMU3        1
#define VIRTIO_IOMMU_ATTACH_TABLE_INTEL_PT         2
#define VIRTIO_IOMMU_ATTACH_TABLE_RISCV            4
#define VIRTIO_IOMMU_ATTACH_TABLE_AMD_GCR3         5
#define VIRTIO_IOMMU_ATTACH_TABLE_AMD_PT           6
```





5.13.7.3.2 ATTACH_TABLE request for RISC-V IOMMU

```
struct virtio_iommu_req_attach_table_riscv {
  struct virtio_iommu_req_head head;
  le32  domain;
  le32  endpoint;
  u8    format;
  u8    reserved[3];
  le64  tc;
  le64  ta;
  le64  fsc;
  u8    reserved[84];
  struct virtio_iommu_req_tail tail;
};

#define VIRTIO_IOMMU_HW_RISCV_TC_PDTV           (1 << 5)
#define VIRTIO_IOMMU_HW_RISCV_TC_SADE           (1 << 8)
#define VIRTIO_IOMMU_HW_RISCV_TC_DPE            (1 << 9)
#define VIRTIO_IOMMU_HW_RISCV_TC_SBE            (1 << 10)
#define VIRTIO_IOMMU_HW_RISCV_TC_SXL            (1 << 11)

#define VIRTIO_IOMMU_HW_RISCV_TA_PSCID_SHIFT   12
#define VIRTIO_IOMMU_HW_RISCV_TA_PSCID_MASK    0xfffff

#define VIRTIO_IOMMU_HW_RISCV_FSC_MODE_SHIFT   60
#define VIRTIO_IOMMU_HW_RISCV_FSC_MODE_MASK    0xf
#define VIRTIO_IOMMU_HW_RISCV_FSC_PPN_MASK     0xfffffffffff

/* When VIRTIO_IOMMU_HW_RISCV_TC_PDTV == 0 */
#define VIRTIO_IOMMU_HW_RISCV_FSC_MODE_SV39    8
#define VIRTIO_IOMMU_HW_RISCV_FSC_MODE_SV48    9
#define VIRTIO_IOMMU_HW_RISCV_FSC_MODE_SV57    10
#define VIRTIO_IOMMU_HW_RISCV_FSC_MODE_SV32    8

/* When VIRTIO_IOMMU_HW_RISCV_TC_PDTV == 1 */
#define VIRTIO_IOMMU_HW_RISCV_FSC_MODE_PD8     1
#define VIRTIO_IOMMU_HW_RISCV_FSC_MODE_PD17    2
#define VIRTIO_IOMMU_HW_RISCV_FSC_MODE_PD20    3
```

iommu: Add virtio-iommu driver (2019.1)
Author: Jean-Philippe Brucker
<jean-philippe@linaro.org>

优势：

- Linux 五年前已支持 virtio-IOMMU
- QEMU 后端框架已实现
- 无需模拟 RISC-V IOMMU 实现
- virtio 命令队列更高效
- 跨 CPU ISA 支持
- 支持 GIPC 和 GIDC

总结

XUANTIE

- RISC-V IOMMU 仅支持 GIDC
- GIPC 提案使 RISC-V IOMMU 可以同时支持 GIPC 和 GIDC 的混合部署。
- 支持 Emulated-IOMMU
- 支持 Virtio-IOMMU（欢迎一起共建）

XUANTIE

Thank you

玄铁公众号 玄铁中文站 玄铁海外站