

# RISC-V Heterogeneous Programming Paradigm

Atomic IO Enqueue (AIOE) Extension & AIOE with Virtualization

**GUO REN**

Alibaba Damo Academy, XUANTIE Team

# CONTENTS

---

## 01 Motivation

Heterogeneous Computing Trends & Problems

---

## 02 Solution & Proposals

1. Atomic IO Enqueue (AIOE) Extension
  2. AIOE with Virtualization:  
G-stage table In Process Context (GIPC) Extension
- 

## 03 Final Remarking

---

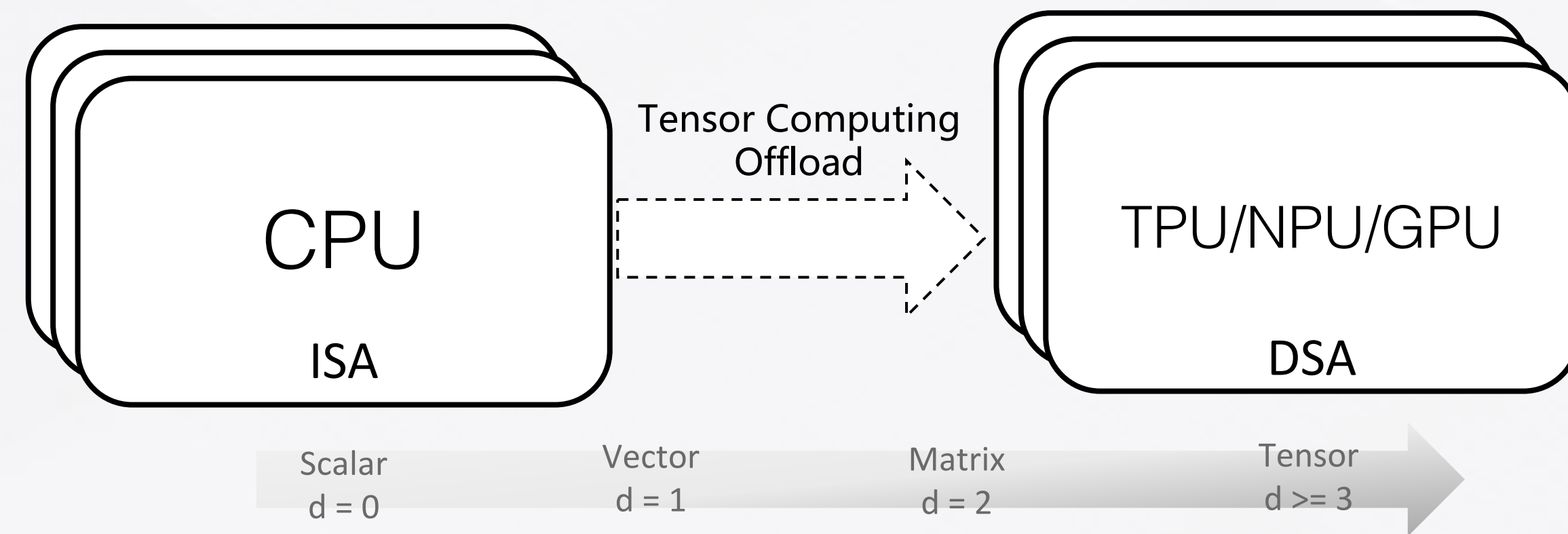
# Motivation

Heterogeneous Computing Trends & Problems

---

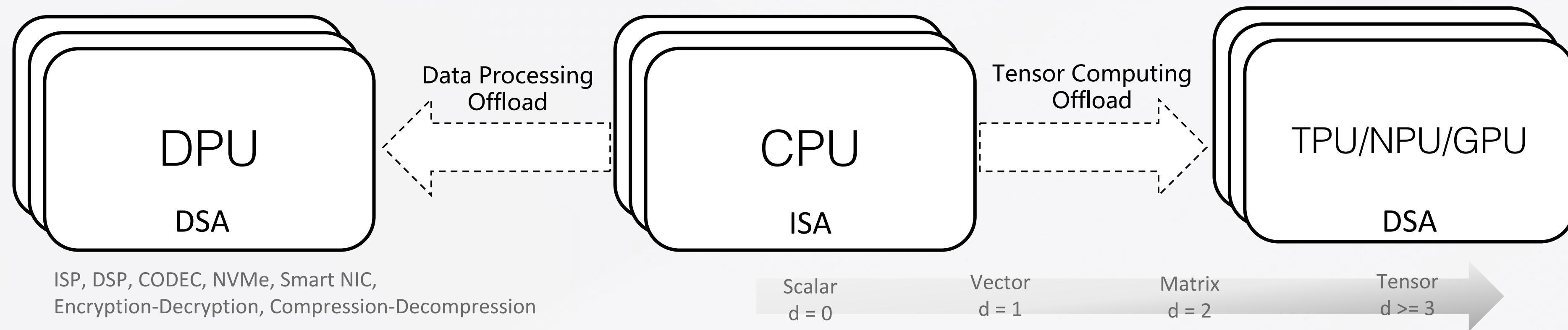


# Heterogeneous Computing Trends

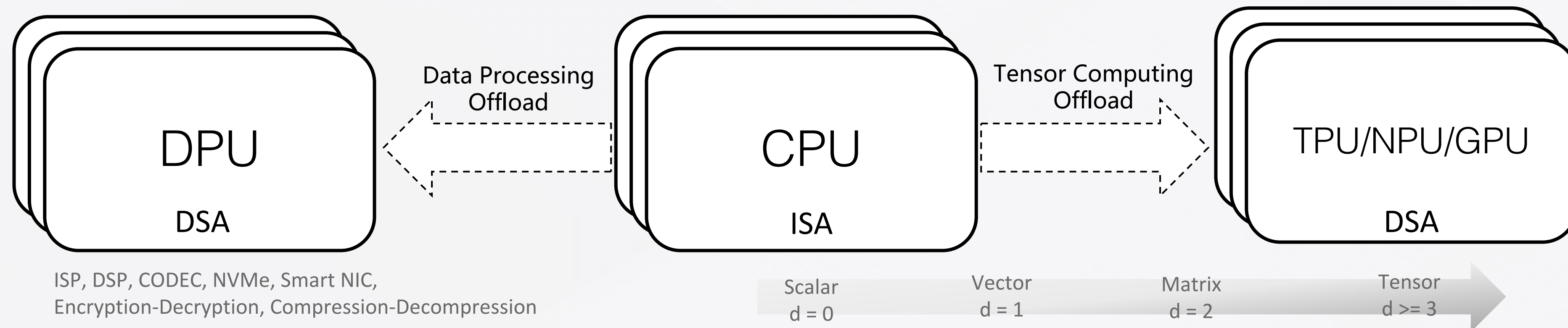




# Heterogeneous Computing Trends



# Heterogeneous Computing Trends

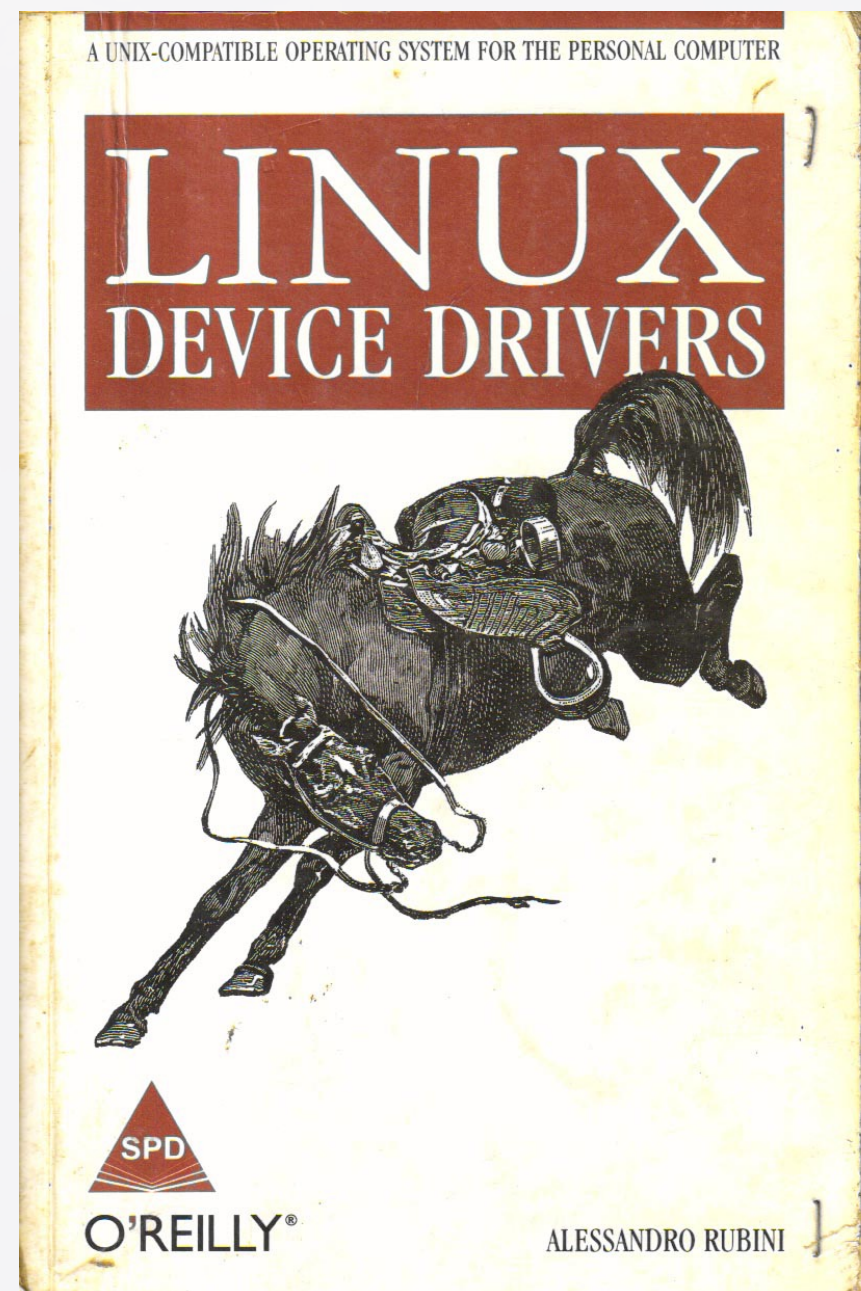


From SoC to Chiplet to CXL, computing systems have become natively heterogeneous.

***But, when focusing on DSA ISA design,  
who cares about the convenience of heterogeneous programming?***



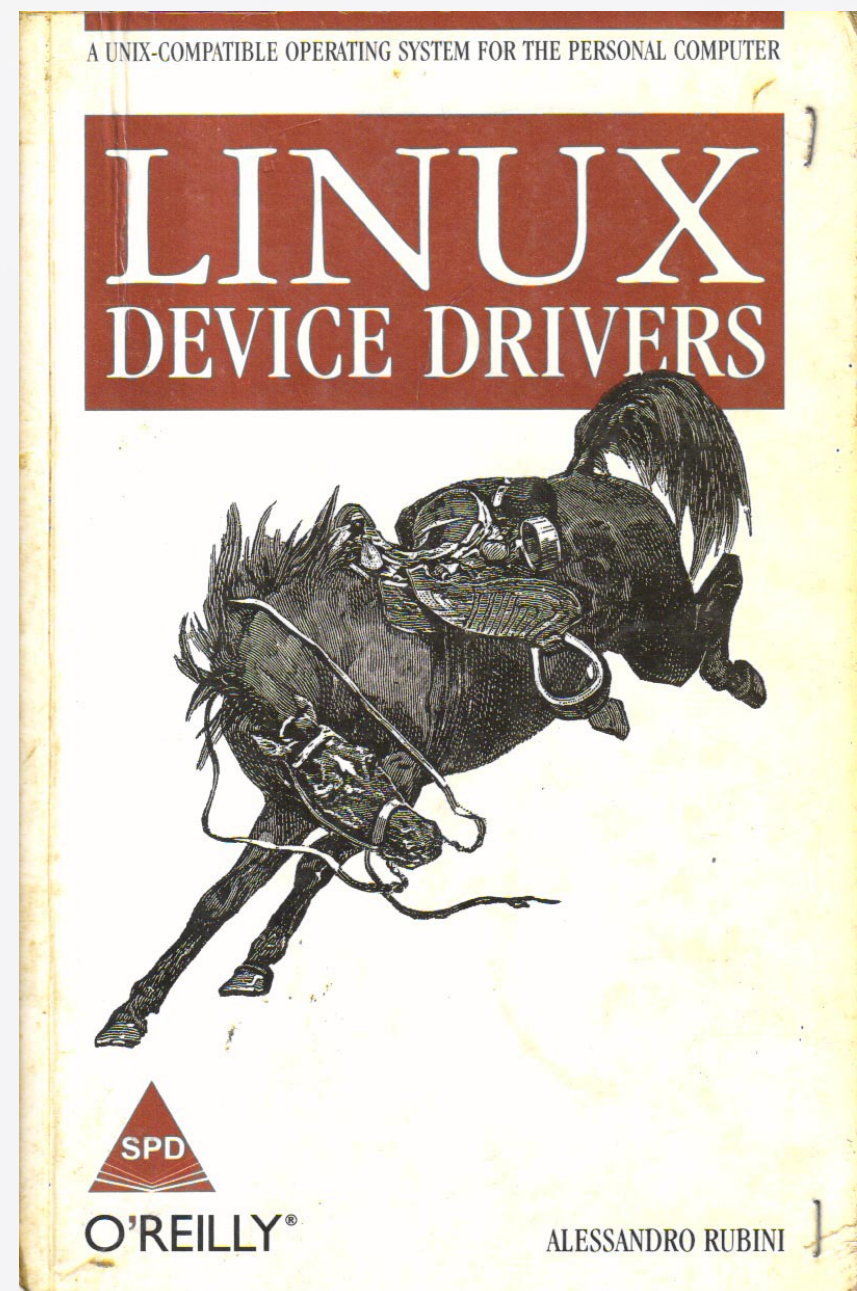
# Traditional Heterogeneous Programming



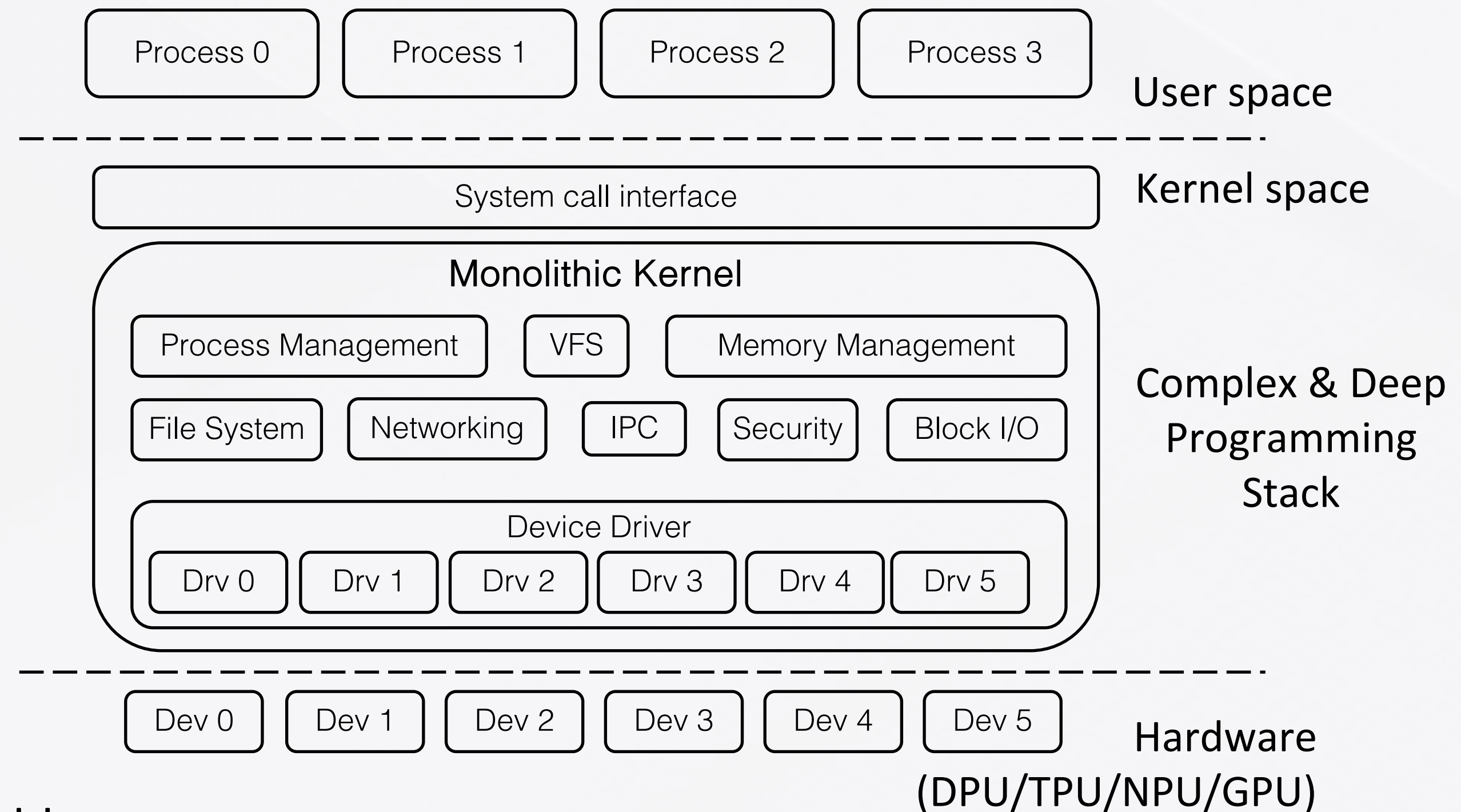
February 1998



# Traditional Heterogeneous Programming



February 1998



## Problems:

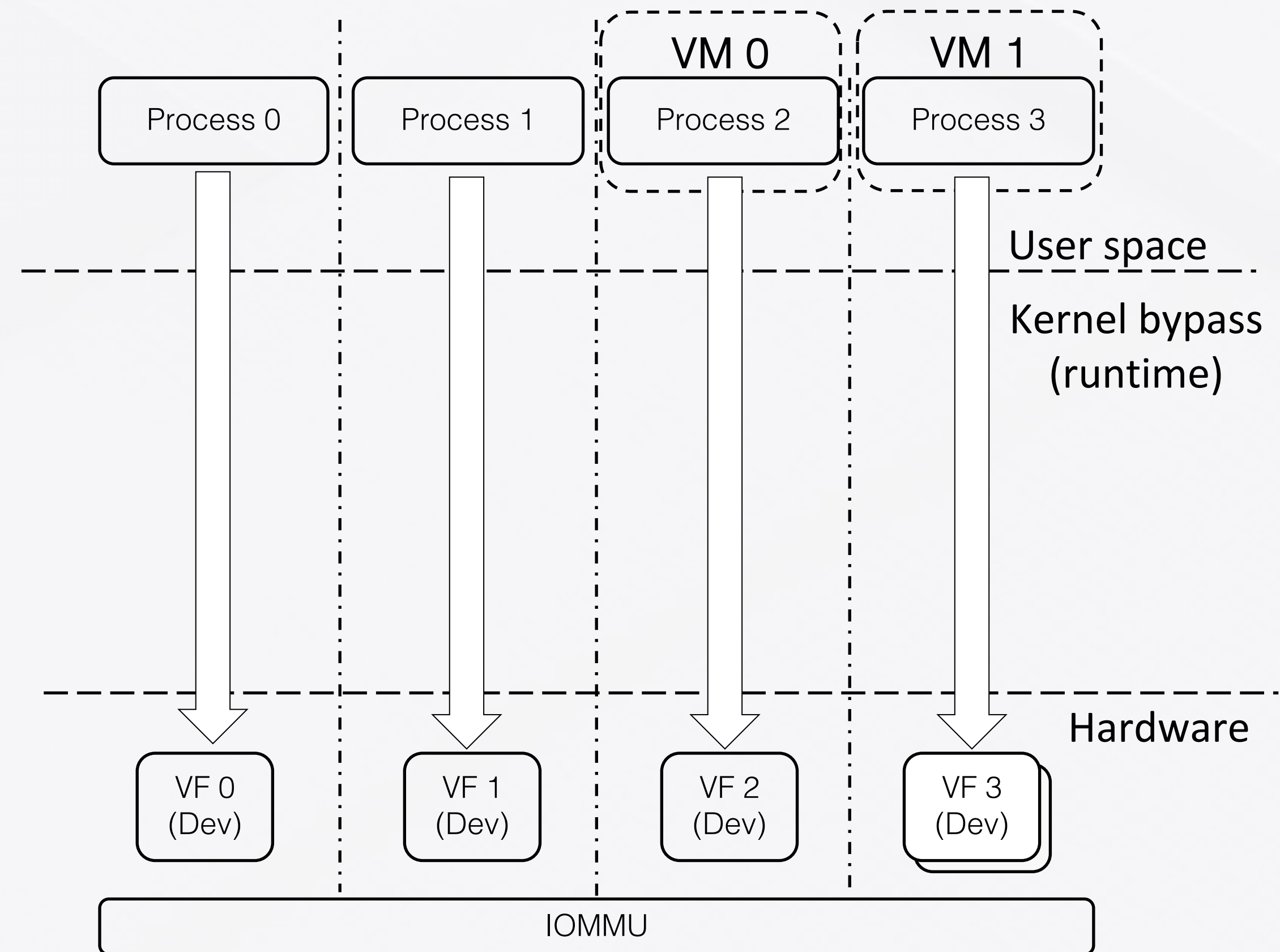
- Privilege context-switch overhead
- Synchronization primitives overhead
- Weak isolation for device safety & security
- Development is complex and expensive



# Bypass Linux Kernel

## Industry Cases:

- DPDK - Data Plane Development Kit
- SPDK - Storage Performance Development Kit



# Bypass Linux Kernel

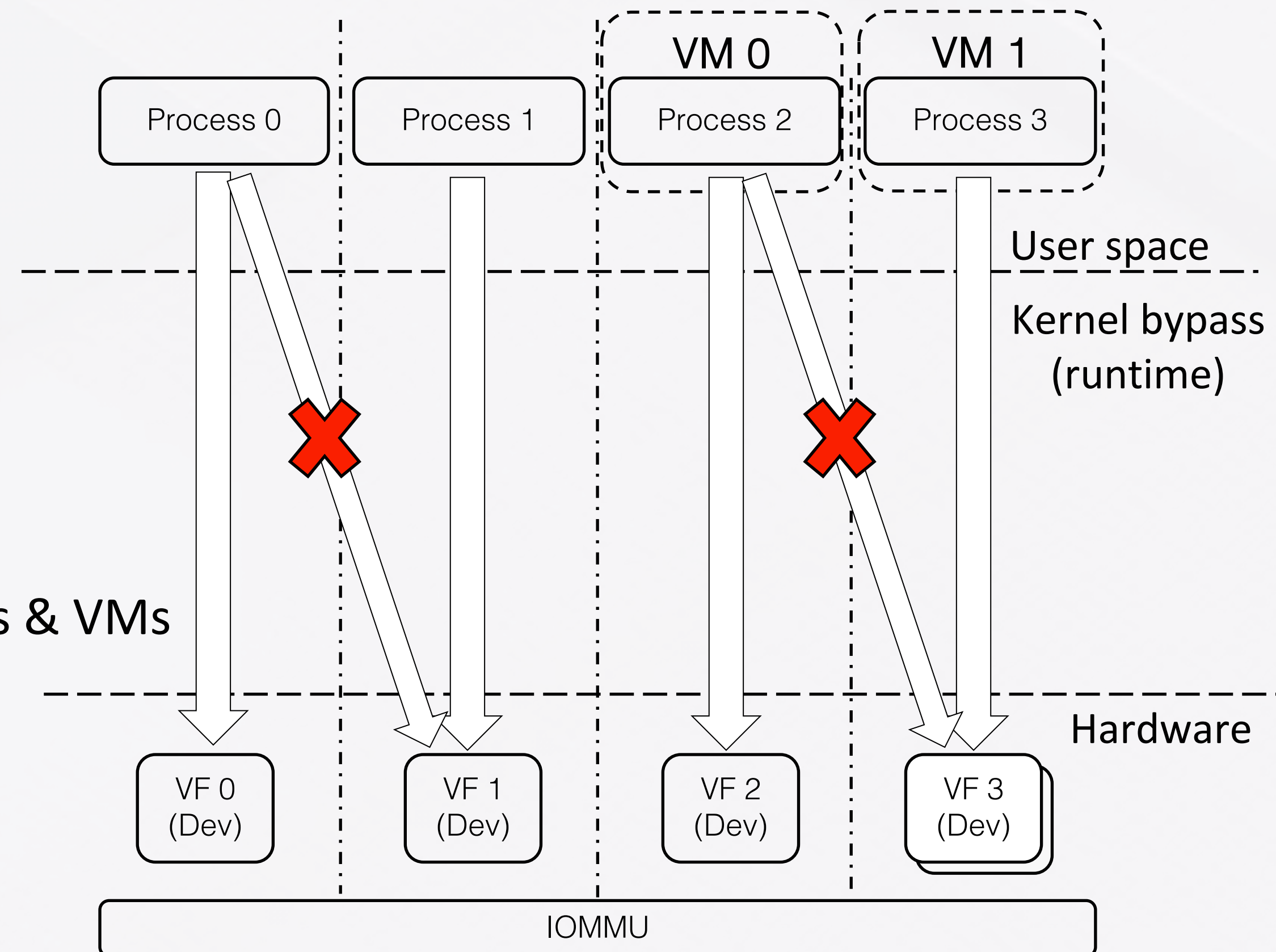
## Industry Cases:

- DPDK - Data Plane Development Kit
- SPDK - Storage Performance Development Kit

## Problems that remain:

- Synchronization primitives overhead
- The device can't be shared across processes & VMs

***What is an ideal heterogeneous programming paradigm?***





---

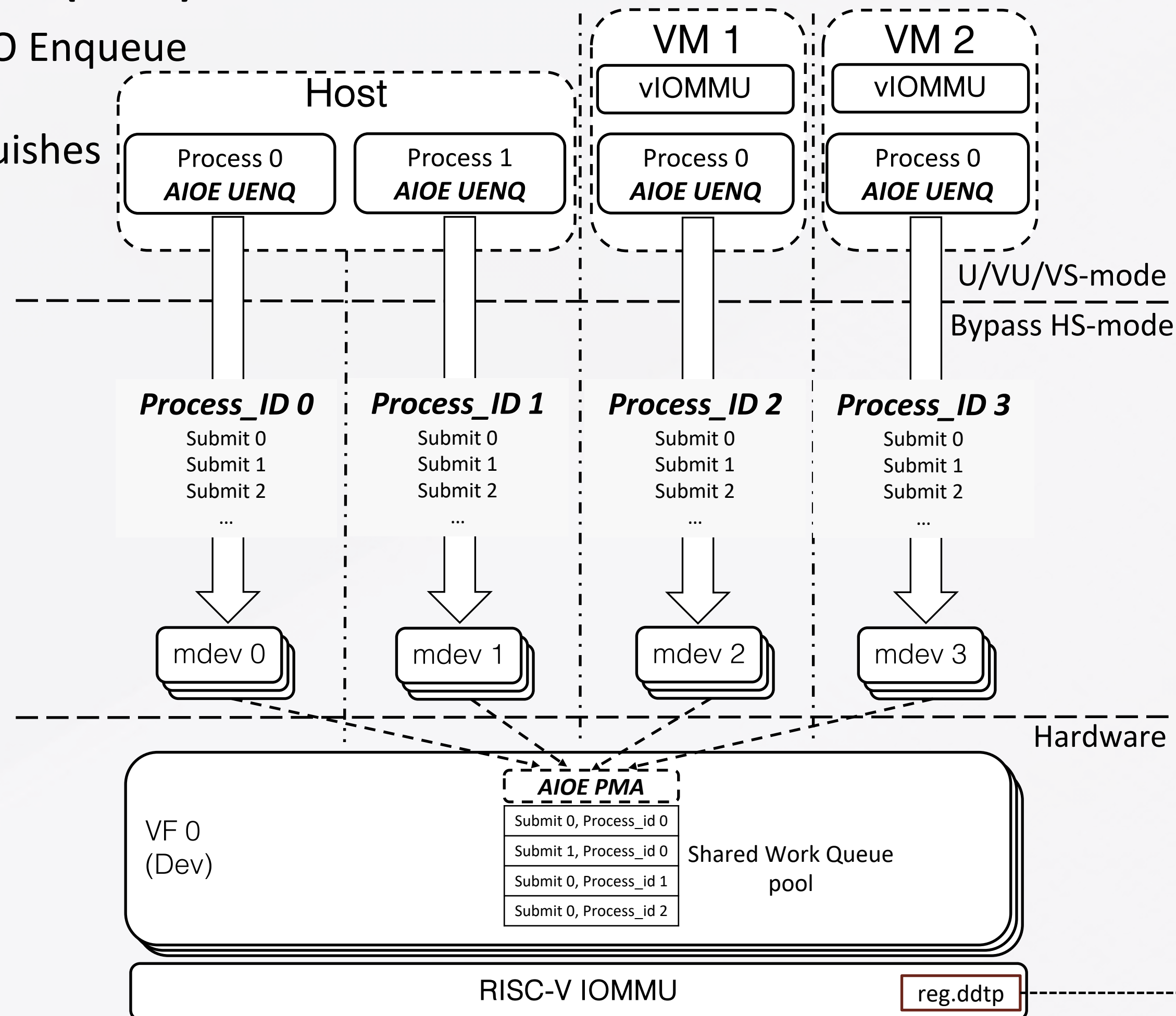
## Solution & Proposals

1. Atomic IO Enqueue (AIOE) Extension
  2. AIOE under Virtualization:  
G-stage table In Process Context (GIPC) Extension
-

# RISC-V Heterogeneous Programming Paradigm

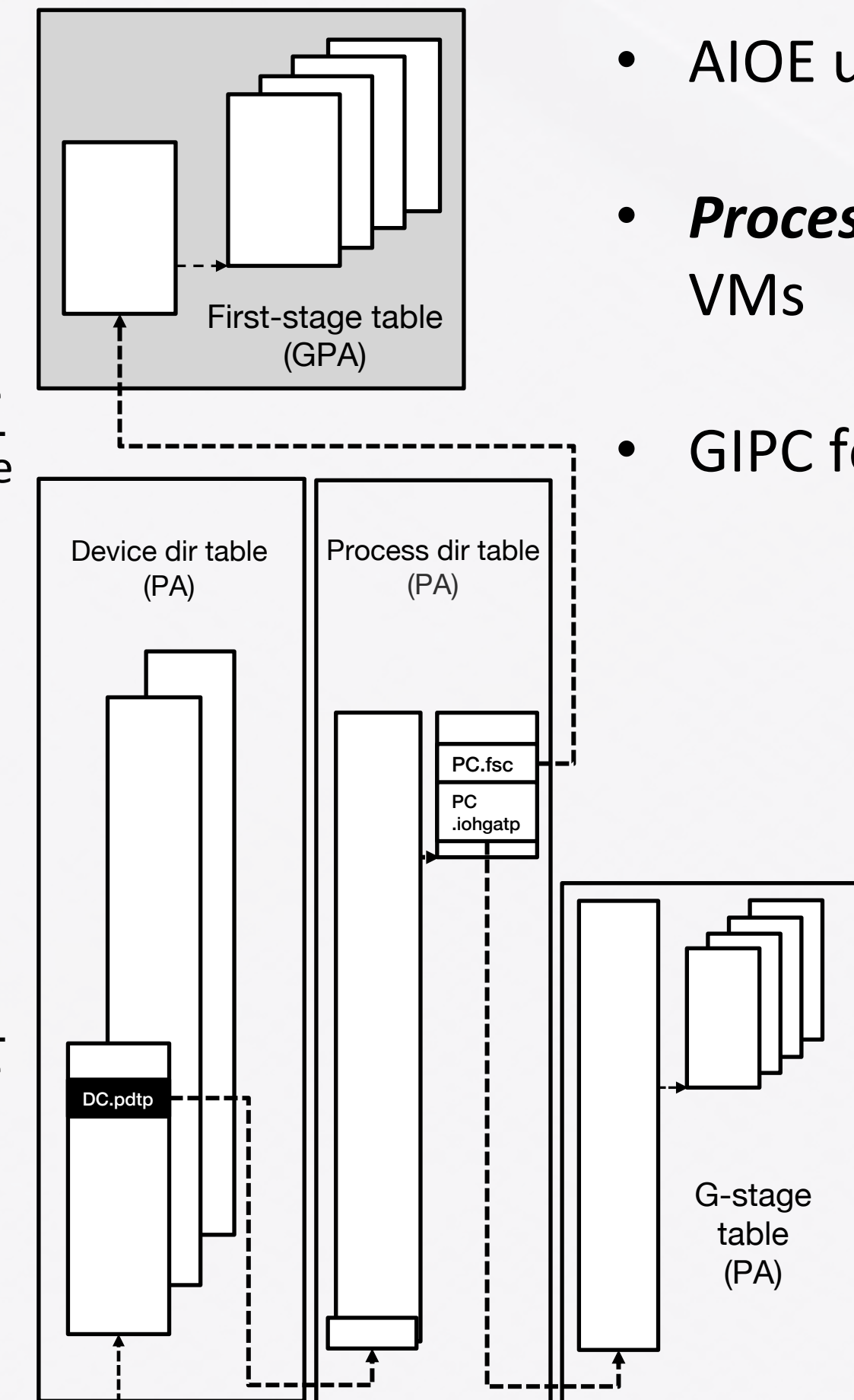
## 1. Atomic IO Enqueue (AIOE) Extension

- UENQ for Atomic IO Enqueue
- **Process\_ID** distinguishes Processes
- AIOE for ISA



## 2. G-stage table In Process Context (GIPC) Extension

- AIOE under Virtualization
- **Process\_ID** distinguishes VMs
- GIPC for IOMMU





# Atomic IO Enqueue (AIOE) Extension

## UENQ.64B (64-byte Atomic IO Store)

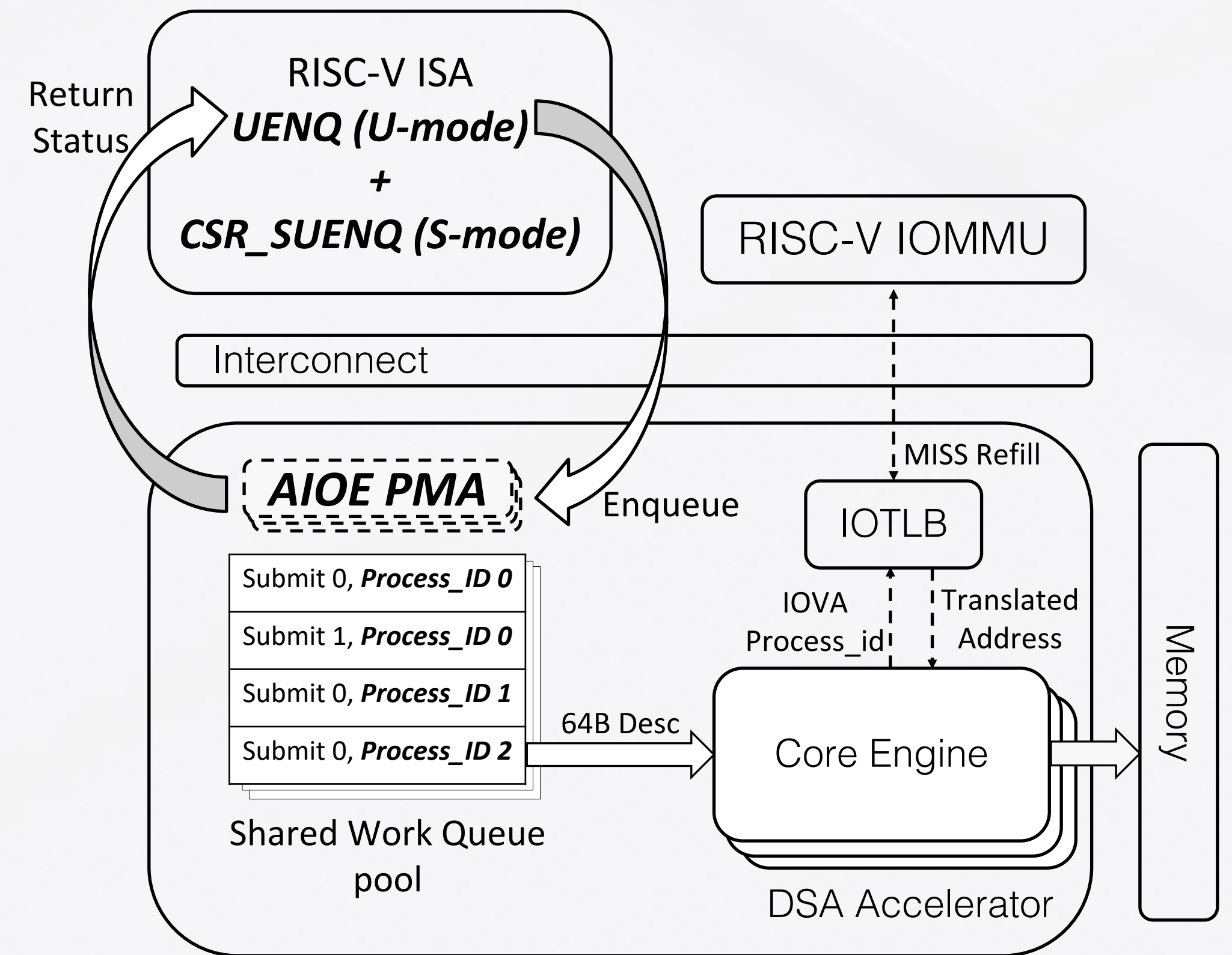
Unprivileged Enqueue (UENQ) instruction of the atomic IO single-write 64-Byte with/without the status result. The 64-Byte store data is formed as data[511:32]:<CSR\_SUENQ>[31:0] from 8 consecutive registers.

## AIOE PMA (for Security)

Atomic IO Enqueue (AIOE) Physical Memory Attributes (PMA) defines SENQ and UENQ target address attribute.

## CSR\_SUENQ (for Process\_ID)

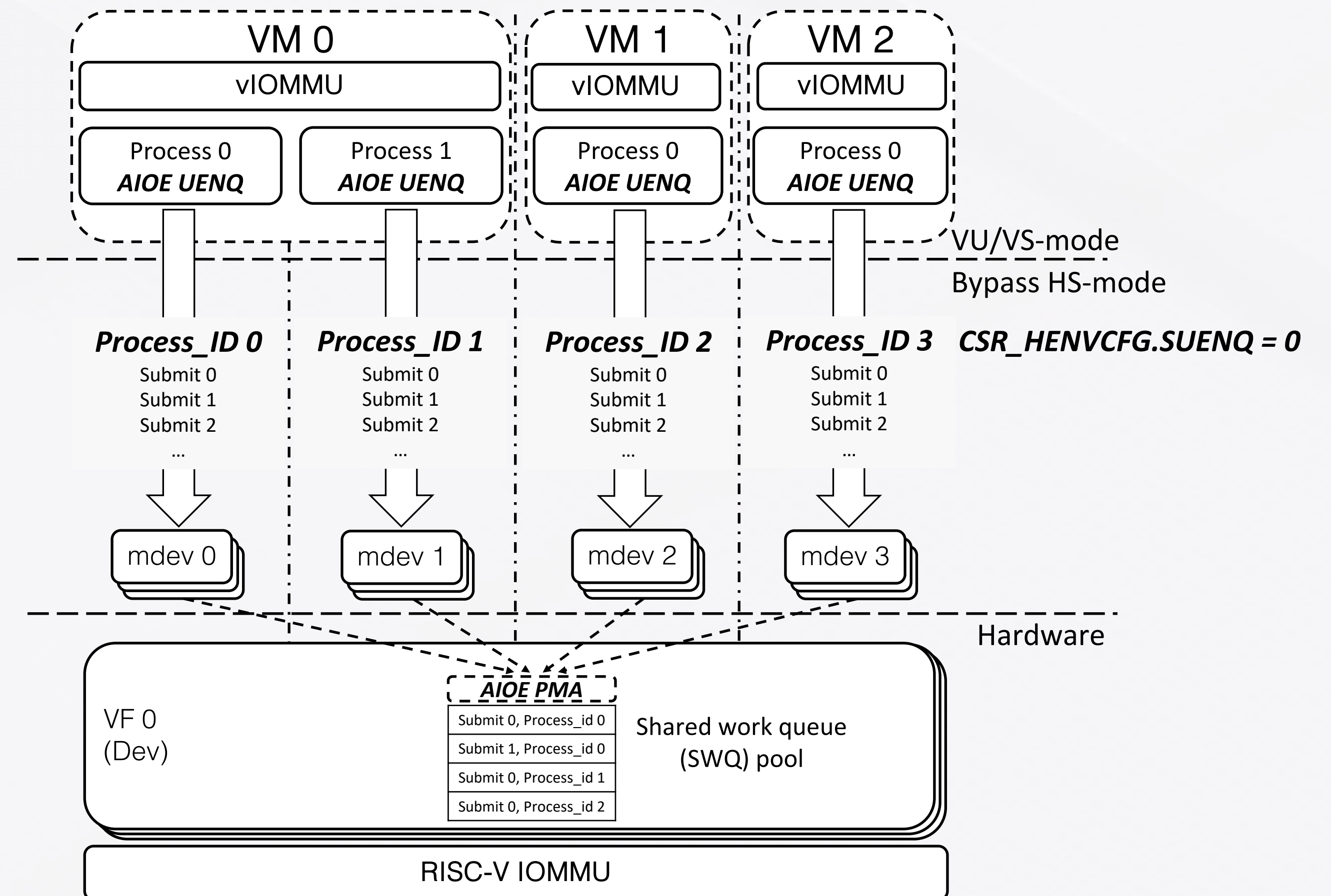
Privileged CSR register that replaces the lowest bits of the store data of UENQ.64B as Process\_ID.



# AIOE with Virtualization

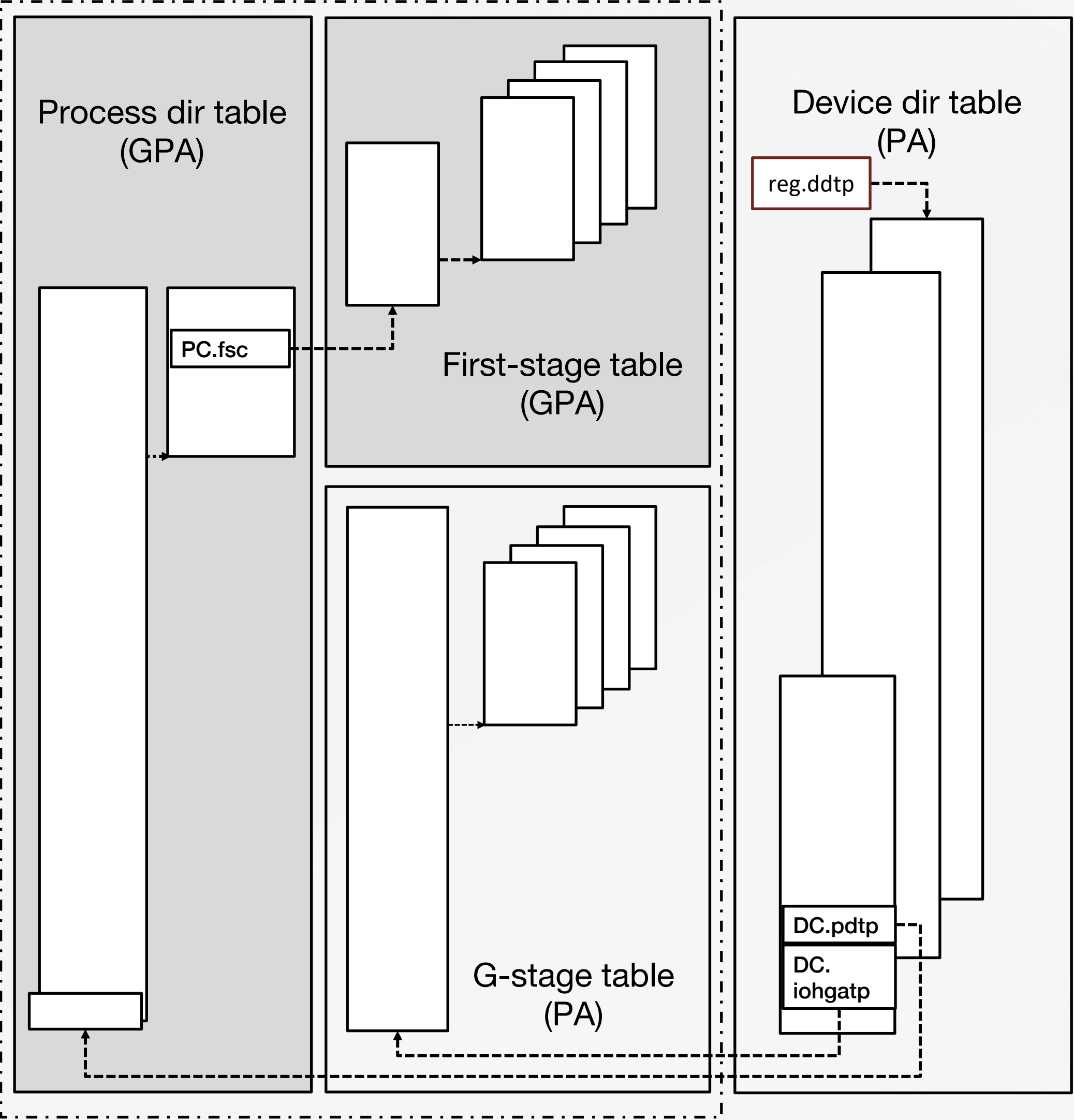
- ***CSR\_HENVCFG.SUENQ = 0***
- ***Process\_ID distinguishes VMs***
- The vIOMMU maintains First-stage table
- VMM maintains Device dir tables, Process dir tables, G-stage tables

*However, the current RISC-V IOMMU distinguishes VM domains by Device\_ID:*  
***G-stage table In Device Context (GIDC)***

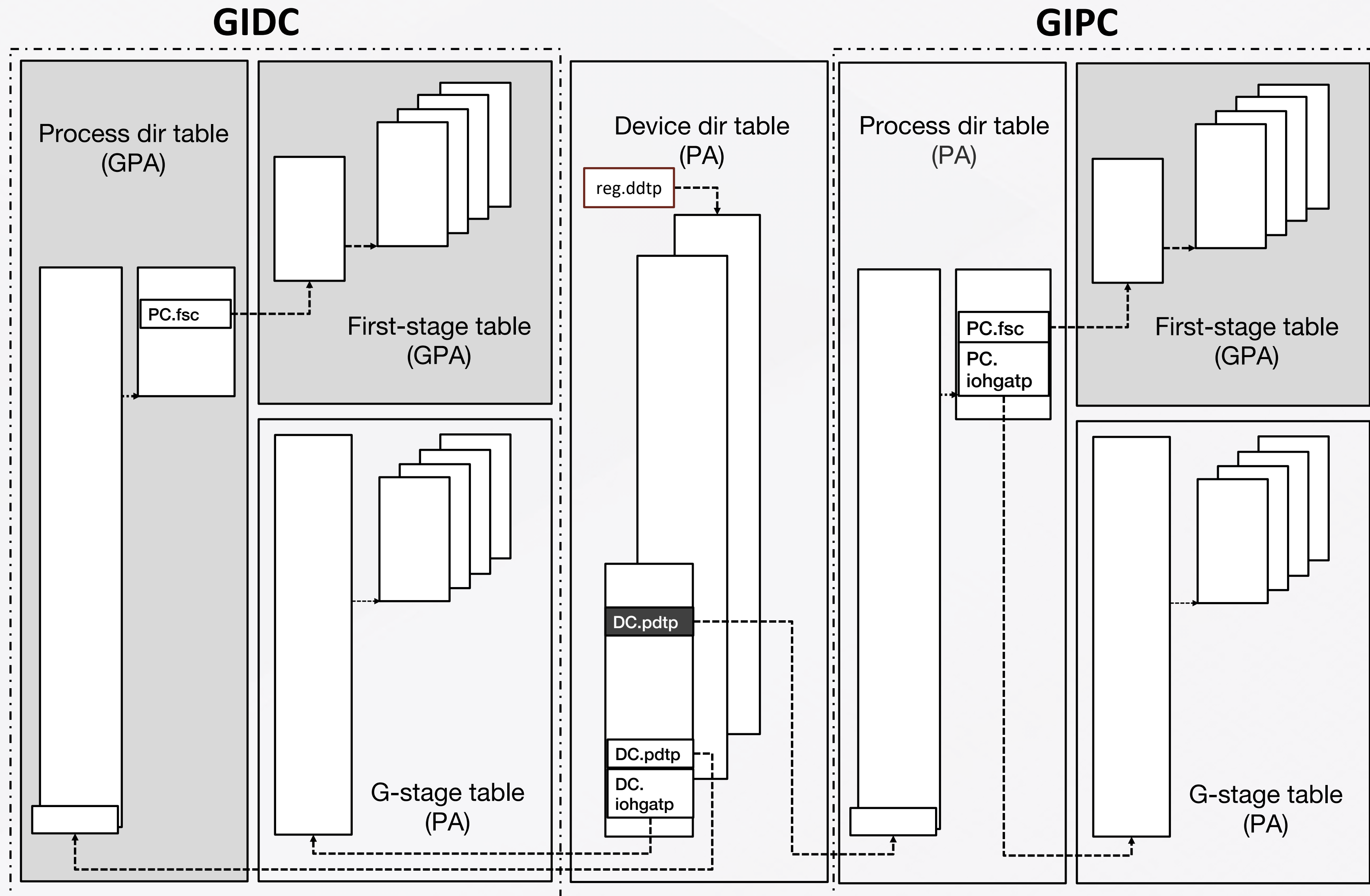




GIDC



# G-stage table In Process Context (GIPC)



***Process\_ID*** distinguishes VMs:

- *Process dir table is based on PA and maintained by the VMM*
- *No Nested Page Table Walk*
- *No GPA->PA TLB entries*

**VM domain**

**VMM domain**



# Conclusion

1. **AIOE Extension** eliminates the overhead of synchronization primitives.
2. **GIPC Extension** enables AIOE to function across processes of different VMs & domains.

With the help of AIOE and GIPC, we could explore a new heterogeneous programming paradigm from HPC to embedded scenarios.

## Opportunity

Discuss HPC accelerator scenario in datacenter SIG.

Link: <https://lists.riscv.org/g/sig-datacenter>

---

# Final Remarking

---



# Final Remarking

## RVI Community Progress:

- AIOE extension published at [tech-privileged@lists.riscv.org](mailto:tech-privileged@lists.riscv.org) has now been updated to v4 with the help of the community's reviewing. Thanks to Allen, Ved, Paul, and Greg for their reviews, and welcome more feedback from the community.

Link: <https://lists.riscv.org/g/tech-privileged/message/2320>

- GIPC has been open for review, and welcome feedback.

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

## Welcome to visit our Poster & Booth

Booth Sign:  XUANTIE

 XUANTIE

### RISC-V Heterogeneous Programming Paradigm

#### Atomic IO Enqueue (AIOE) Extension & AIOE with Virtualization

Author: Guo Ren, Alibaba Damo Academy  
Email: guoren.gr@alibaba-inc.com

#### Abstract

In the era of artificial intelligence, a single CPU architecture can no longer meet the demands of diverse intelligent computing workloads. Consequently, heterogeneous computing has emerged as the mainstream approach. Integrating different Domain-Specific Accelerators (DSAs) into computing systems could enhance overall computational efficiency. For instance, high-dimensional tensor computing tasks can be offloaded to TPUs/NPUs/GPGPUs, while data stream processing tasks are delegated to DPUs. The challenge of efficiently managing DSAs in heterogeneous systems has become a prominent industry focus, driving several technological advancements:

- PCI-e 5.0 and CXL 2.0 have introduced Deferred Memory Write (DWM) TLPs
- Armv8.7/9.2 has incorporated ST64BVD instructions for 64-byte atomic I/O enqueue operations
- The x86 architecture has implemented ENQCMD instructions with comparable functionality

These innovations collectively reduce control latency and optimize system resource utilization.

To help RISC-V adapt to this trend, the presentation introduces the Atomic IO Enqueue Extension (AIOE) and its usage with virtualization. The AIOE extension is designed for RV64 ISA, which includes one PMA definition, two U-mode instructions, two S-mode instructions, a single S-mode CSR, and two envcfg control bits. The presentation also introduces how to use AIOE under the virtualization scenario, which involves the new proposal for RISC-V IOMMU: G-stage table in Process Context (GIPC). With the help of AIOE and GIPC, RISC-V could explore a new heterogeneous programming paradigm from HPC to embedded scenarios.

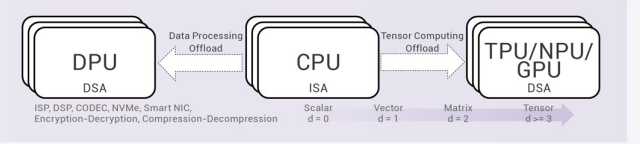


Diagram illustrating the heterogeneous computing paradigm. It shows a DPU (Data Processing Unit) with a DSA (Domain-Specific Accelerator) for operations like Encryption-Decryption, Compression-Decompression, and Matrix. Data is offloaded to a CPU (Central Processing Unit) with an ISA (Instruction Set Architecture). The CPU then offloads data to a TPU/NPU/GPU (Tensor Processing Unit/Neural Processing Unit/GPGPU) with a DSA for operations like Matrix and Vector. The diagram also shows various enqueue and dequeue operations between these units.

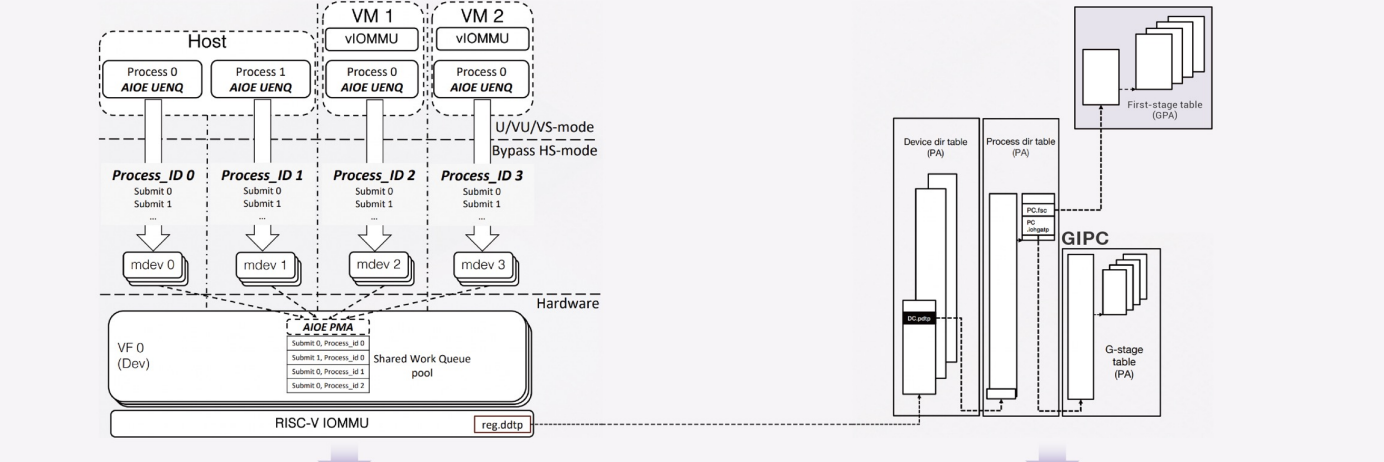


Diagram illustrating the RISC-V IOMMU architecture. It shows a Host with multiple VMs (Virtual Machines) running on it. Each VM has its own IOMMU (Input/Output Memory Management Unit). The Host IOMMU manages the mapping between VMs and hardware devices. The hardware devices are connected to the Host IOMMU via a Shared Work Queue pool. The Host IOMMU also manages the mapping between VMs and hardware devices. The Host IOMMU also manages the mapping between VMs and hardware devices. The Host IOMMU also manages the mapping between VMs and hardware devices.

#### ① Atomic IO Enqueue (AIOE) Extension

"Atomic IO Enqueue" (AIOE) extension is designed for the RV64 ISA, which contains one PMA definition, two user instructions, two supervisor instructions, one single S-mode CSR, and two envcfg control bits:

| AIOE PMA          | Atomic IO Enqueue Physical Memory Attribute                             |
|-------------------|-------------------------------------------------------------------------|
| CSR_SUENQ         | Supervisor Read Write CSR for UENQ instructions                         |
| UENQ.64B          | User Enqueue Instruction for 64-byte                                    |
| UENQ.32B          | User Enqueue Instruction for 32-byte (Optional)                         |
| SENQ.64B          | Supervisor Enqueue Instruction for 64-byte                              |
| SENQ.32B          | Supervisor Enqueue Instruction for 32-byte (Optional)                   |
| CSR_MENVCFG_SUENQ | Control bit for SENQ & CSR_SUENQ in S/H/S/VS-mode                       |
| CSR_HENVCFG_SUENQ | Control bit for SENQ & CSR_SUENQ in VS-mode (AIOE under Virtualization) |

② UENQ.64B (64-byte Atomic IO Store)

Unprivileged Enqueue (UENQ) instruction of the atomic IO single-write 64-Byte with/without the status result. The 64-Byte store data is formed as data [61:1/255:32]<CSR\_SUENQ[31:0] from 8 consecutive registers.

③ CSR\_SUENQ (for Process\_ID)

Privileged CSR register that replaces the lowest bits of the store data of UENQ.64B as Process\_ID.

④ AIOE PMA (for Security)

Atomic IO Enqueue (AIOE) Physical Memory Attributes (PMA) defines SENQ and UENQ target address attribute.

#### ② AIOE with Virtualization

• CSR\_HENVCFG\_SUENQ = 0

- The vIOMMU maintains First-stage table
- vIOMMU maintains Device dir tables, Process dir tables, G-stage tables
- Process\_ID distinguishes VMs

However, the current RISC-V IOMMU distinguishes VM domains by Device\_ID.

Diagram illustrating the AIOE with Virtualization. It shows a Host with multiple VMs running on it. Each VM has its own IOMMU (Input/Output Memory Management Unit). The Host IOMMU manages the mapping between VMs and hardware devices. The hardware devices are connected to the Host IOMMU via a Shared Work Queue pool. The Host IOMMU also manages the mapping between VMs and hardware devices. The Host IOMMU also manages the mapping between VMs and hardware devices.

#### G-stage table In Process Context (GIPC) Extension

Diagram illustrating the G-stage table In Process Context (GIPC) Extension. It shows a Host with multiple VMs running on it. Each VM has its own IOMMU (Input/Output Memory Management Unit). The Host IOMMU manages the mapping between VMs and hardware devices. The hardware devices are connected to the Host IOMMU via a Shared Work Queue pool. The Host IOMMU also manages the mapping between VMs and hardware devices. The Host IOMMU also manages the mapping between VMs and hardware devices.

► For more information, please visit [xrv.com](http://xrv.com)



---

# backup

---

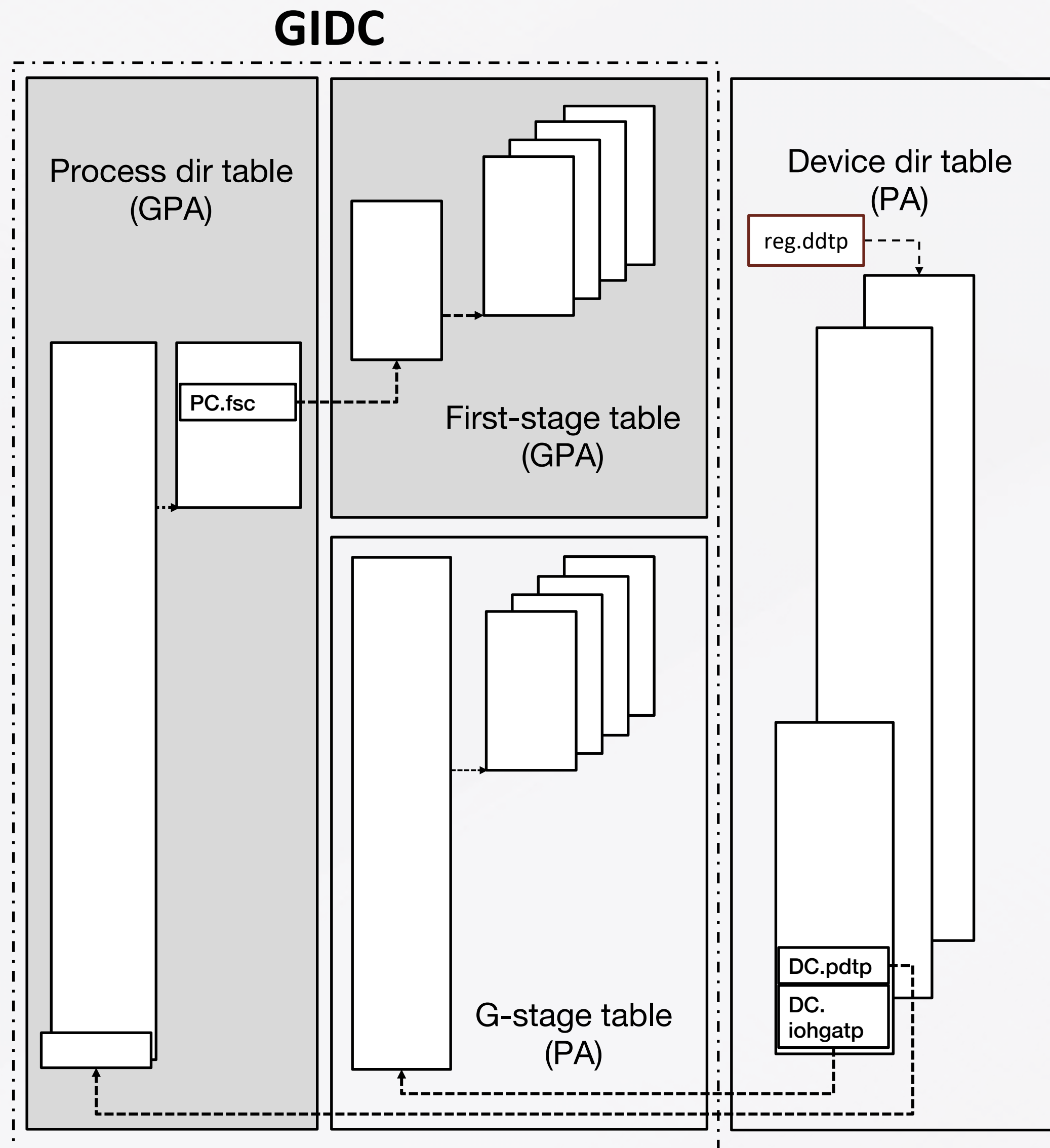


# AIOE Summary

“Atomic IO Enqueue” (AIOE) extension is designed for the RV64 ISA, which contains one PMA definition, two user instructions, two supervisor instructions, one single S-mode CSR, and two envcfg control bits:

|                          |                                                                                        |
|--------------------------|----------------------------------------------------------------------------------------|
| AIOE PMA                 | Atomic IO Enqueue Physical Memory Attribute                                            |
| CSR_SUENQ                | Supervisor Read Write CSR for UENQ instructions                                        |
| UENQ.64B                 | User Enqueue Instruction for 64-byte                                                   |
| UENQ.32B                 | User Enqueue Instruction for 32-byte (Optional)                                        |
| SENQ.64B                 | Supervisor Enqueue Instruction for 64-byte                                             |
| SENQ.32B                 | Supervisor Enqueue Instruction for 32-byte (Optional)                                  |
| CSR_MENVCFG.SUENQ        | Control bit for SENQ & CSR_SUENQ in S/HS/VS-mode                                       |
| <b>CSR_HENVCFG.SUENQ</b> | <b>Control bit for SENQ &amp; CSR_SUENQ in VS-mode<br/>(AIOE under Virtualization)</b> |

# RISC-V IOMMU Limitation



*Current RISC-V IOMMU implementation:*

***G-stage table In Device Context (GIDC)***

***Device\_ID*** distinguishes VMs:

- *Process dir table is based on GPA and maintained by the VM*
- *GPA causes Nested Page Table Walk*
- *GPA causes GPA->PA TLB entries*

**VM domain**

**VMM domain**