VENTANA MICRO

AIA Virtualization in KVM RISC-V

Anup Patel <apatel@ventanamicro.com>

Outline

- AIA Specification Overview
- AIA Support in KVM RISC-V
- AIA Software Status and Demo

AIA Specification Overview

RISC-V AIA Specification

VENTANA MICRO
RISC-V Performance Leader

- **RISC-V Advanced Interrupt Architecture (AIA)**
  - Addresses limitations of RISC-V PLIC present in existing RISC-V platforms
  - Scalable for system with large number of HARTs
  - Defines functionality as optional modular components
  - Supports message signaled interrupts (MSIs)
  - Supports inter-processor interrupt (IPIs) as software injected MSIs
  - Supports MSI virtualization and IPI virtualization

- **RISC-V AIA specification is in stable state** (Frozen by RISC-V Summit 2022)
  - https://github.com/riscv/riscv-aia/releases/download/0.3.1-draft.32/riscv-interrupts-032.pdf

- **Defines three modular (optional) components:**
  - Extended Local Interrupts (**AIA CSRs**)
  - Incoming Message Signaled Interrupt Controller (**IMSIC**)
  - Advanced Platform Level Interrupt Controller (**APLIC**)

AIA: Extended local interrupts

VENTANA MICRO
RISC-V Performance Leader

- **Smaia ISA extension**: New AIA CSRs for M-mode
- **Ssaia ISA extension**: New AIA CSRs for HS/S-mode
- Supports 64 local interrupts for both RV32 and RV64
- Supports configurable priority for each local interrupt
- Supports local interrupt filtering for M-to-S and HS-to-VS modes
- Backward compatible with existing local interrupts defined by the RISC-V privileged specification

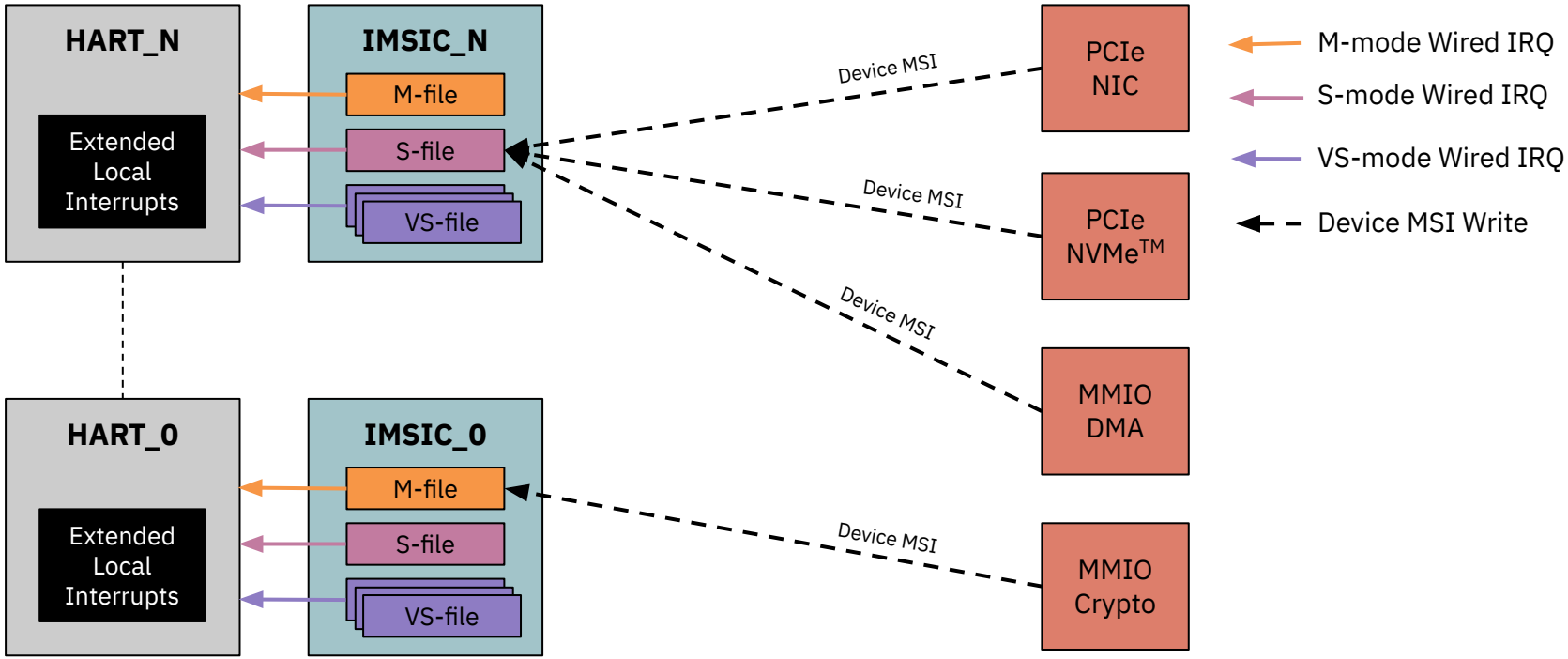
AIA: MSIs using IMSIC

VENTANA MICRO
RISC-V Performance Leader

- One IMSIC instance next to each HART
  - No limit on maximum number of HARTs
- Each IMSIC instance consist of multiple interrupt files
  - **One M-file** (M-level interrupt file), **one S-file** (S-level interrupt file), and **GEILEN guest-files/VS-files** (VS-level interrupt files)
  - Each interrupt file consumes 4KB of physical address space
- Interrupt file configuration done via AIA CSRs
- Each interrupt file supports up to 2047 interrupt identities
- MSI and IPI virtualization supported using VS-files for HARTs with H-extension

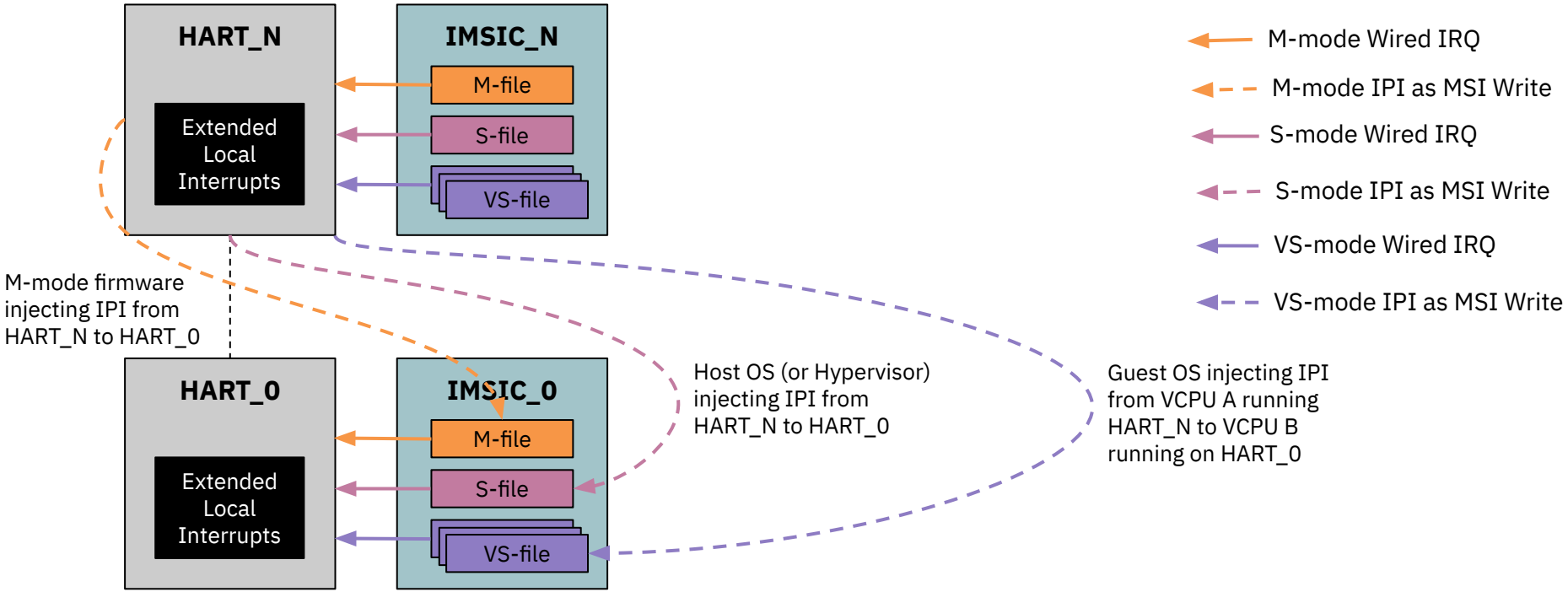
AIA: MSIs using IMSIC (Contd.)

VENTANA MICRO
RISC-V Performance Leader



AIA: IPIs as software injected MSIs

VENTANA MICRO
RISC-V Performance Leader



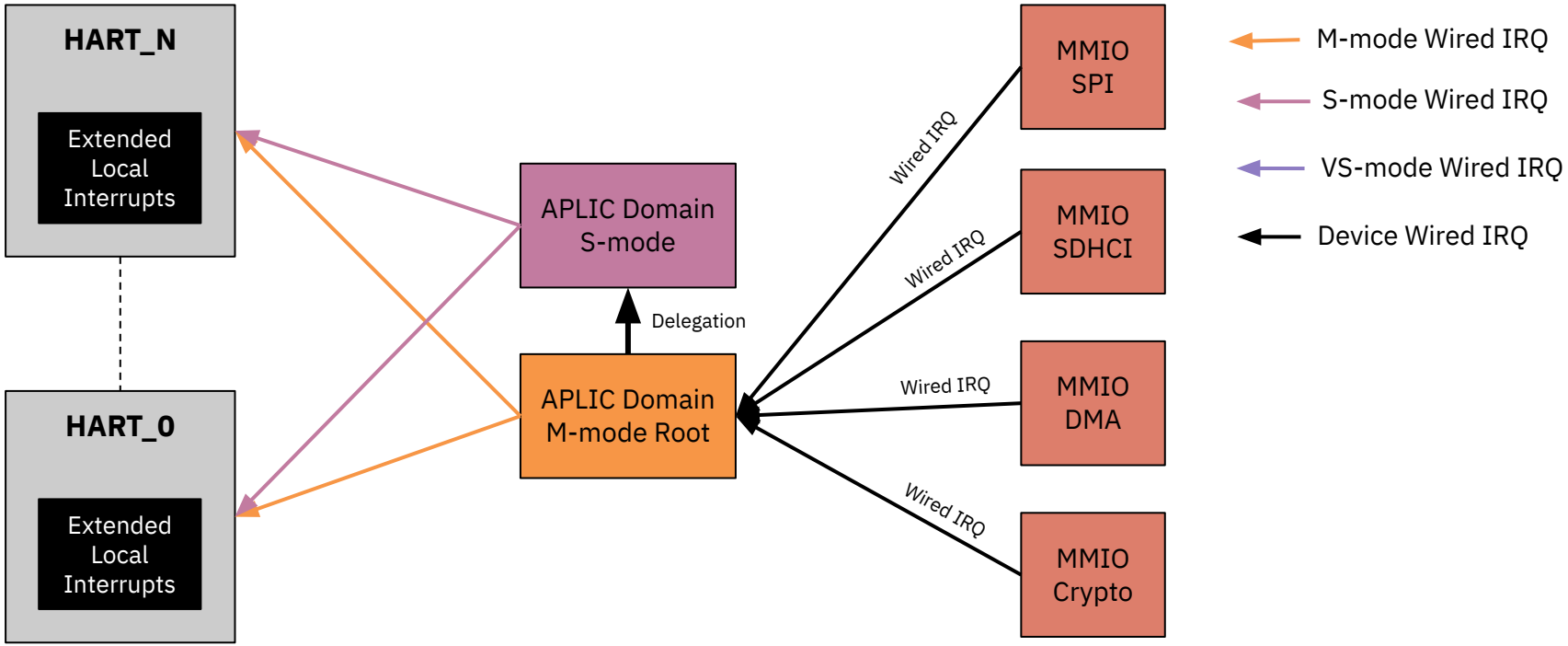
AIA: Wired interrupts using APLIC

VENTANA MICRO
RISC-V Performance Leader

- Hierarchical APLIC domains
  - Wired interrupts from devices only connect to root APLIC domain
  - Each APLIC domain targets a particular privilege level of associated HARTs
  - **An APLIC domain can delegate interrupts** to any of the child APLIC domains
- Configuration done via memory mapped registers (AIA CSRs are not required)
- Configurable line-sensing, priority and target HART for each interrupt source
- Supports up to 1023 interrupt sources and up to 16384 HARTs
- Supports two modes:
  - **Direct mode:** Directly injecting external interrupt to associated HARTs
    - Each APLIC domain consumes physical address space between 16KB to 528KB
  - **MSI mode:** Forward wired interrupt as MSI to associated HARTs
    - Each APLIC domain consumes fixed physical address space of 16KB

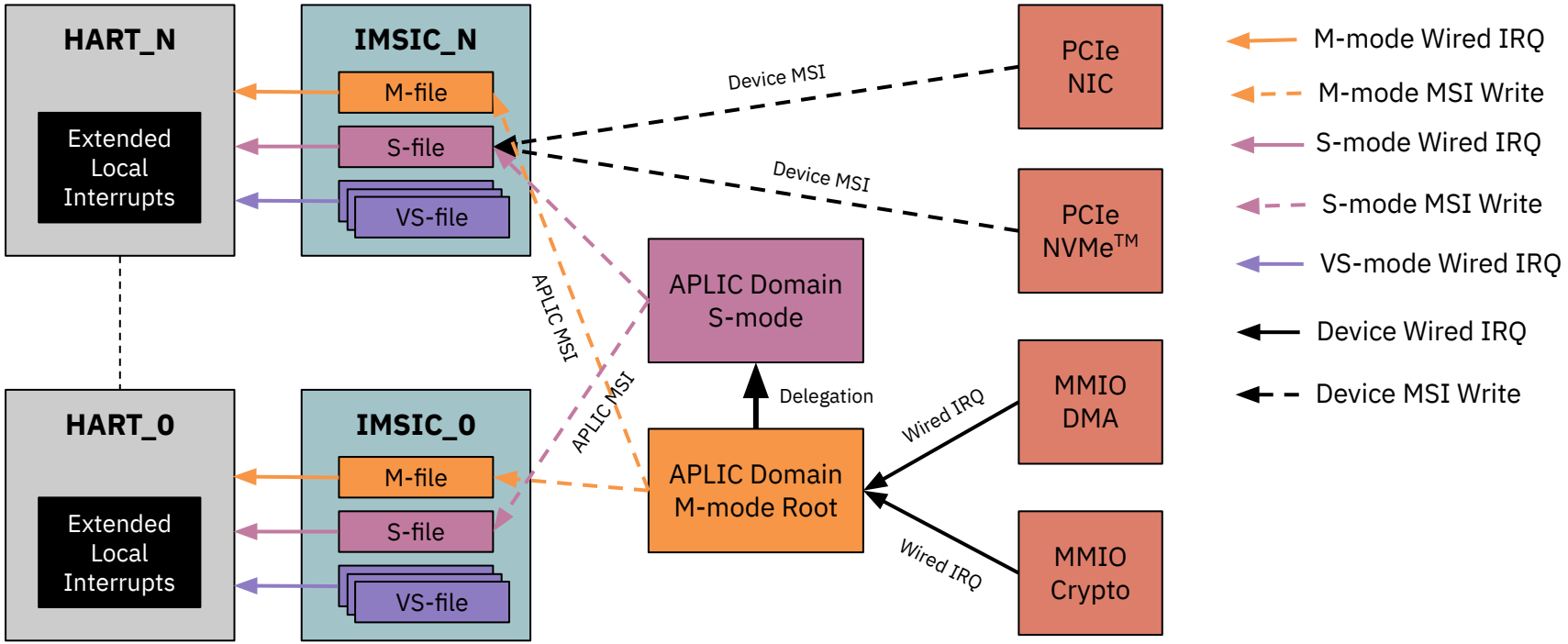
AIA: Wired interrupts using APLIC direct mode

VENTANA MICRO
RISC-V Performance Leader



AIA: Wired interrupts using APLIC MSI mode

VENTANA MICRO
RISC-V Performance Leader



AIA Virtualization support

VENTANA MICRO
RISC-V Performance Leader

- **AIA CSR virtualization**
  - Separate VS-mode CSRs for Guest/VM
  - Local interrupt priorities for VS-mode virtualized using hvictl, hviprio1 and hviprio2 CSRs

- **IMSIC virtualization**
  - Multiple guest-files (or VS-files) for each HART for virtualizing interrupt file for Guest/VM
  - VS-file assigned to a Guest VCPU is mapped in G-stage and selected using hstatus.VGEIN
  - No traps when Guest VCPU uses VS-file
  - **Hypervisor can:**
    - Inject emulated IRQs by writing to MMIO register of VS-file assigned to Guest VCPU
    - Route/forward device MSIs to MMIO register of VS-files using IOMMU
    - Take VS-file interrupt by setting appropriate bit in hgeie CSR with hie.SGEI == 1

- **APLIC only supports virtualization partly**
  - In MSI mode, there will be no MMIO traps at time of handling interrupts
  - In direct mode, there will be MMIO traps at time of handling interrupts

AIA Support in KVM RISC-V

AIA Support in KVM RISC-V

- Two parts of AIA Support in KVM RISC-V
  - AIA CSR virtualization
  - AIA in-kernel irqchip

- **AIA CSR virtualization**
  - **Always available** when Host has Ssaia extension
  - KVM user-space can access Guest VCPU AIA state using ONE_REG ioctls()

- **AIA in-kernel irqchip**
  - Consist of:
    - An optional APLIC with only MSI delivery mode
    - One IMSIC file for each Guest VCPU
  - It is **an optional feature** of KVM RISC-V
    - KVM user-space can always emulate the irqchip itself

KVM: In-kernel AIA irqchip

VENTANA MICRO
RISC-V Performance Leader

- At any point in time, a Guest VCPU uses one of the following:
  - **IMSIC SW-file** (trap-n-emulated by software)
  - **IMSIC VS-file** (virtualized by hardware)

- **IMSIC VS-file assigned to Guest VCPU must be updated when HART changes**

- Three modes of operation for in-kernel AIA irqchip
  - **Emulation (EMUL)**
    - Always use IMSIC SW-file (i.e. trap-n-emulate) for each Guest VCPU
  - **HW Acceleration (HWACCEL)**
    - Always use IMSIC VS-file (i.e. hardware virtualization) for each Guest VCPU
    - Only available when underlying host has VS-files in IMSIC of each HART
  - **Automatic (AUTO)**
    - Use IMSIC VS-file for Guest VCPU when available otherwise use IMSIC SW-file
    - Only available when underlying host has VS-files in IMSIC of each HART
    - Allows running more VCPUs (> GEILEN) on same HART

KVM: Setup in-kernel AIA irqchip in user-space

VENTANA MICRO
RISC-V Performance Leader

1. Create AIA device file using **KVM_CREATE_DEVICE** ioctl()
   - *type=KVM_DEV_TYPE_RISCV_AIA, flags=0*

2. Configure using **KVM_SET_DEVICE_ATTR** ioctl() on the AIA device file
   - If APLIC is required then must set **AIA interrupt sources** (i.e. number of wired lines)
     - *group=KVM_DEV_RISCV_AIA_GRP_CONFIG, attr=KVM_DEV_RISCV_AIA_CONFIG_SRCS*
   - If APLIC is required then must set **APLIC base address**
     - *group=KVM_DEV_RISCV_AIA_GRP_ADDR, attr=KVM_DEV_RISCV_AIA_ADDR_APLIC*
   - Must set **AIA interrupt identities** (i.e. number of MSI identities)
     - *group=KVM_DEV_RISCV_AIA_GRP_CONFIG, attr=KVM_DEV_RISCV_AIA_CONFIG_IDS*
   - Must set **HART index bits** in IMSIC base addresses
     - *group=KVM_DEV_RISCV_AIA_GRP_CONFIG, attr=KVM_DEV_RISCV_AIA_CONFIG_HART_BITS*
   - Must set **IMSIC base address for each VCPU**
     - *group=KVM_DEV_RISCV_AIA_GRP_ADDR, attr=KVM_DEV_RISCV_AIA_ADDR_IMSIC(<vcpu_id>)*

3. Finalize using **KVM_SET_DEVICE_ATTR** ioctl() on the AIA device file
   - *group=KVM_DEV_RISCV_AIA_GRP_CTRL, attr=KVM_DEV_RISCV_AIA_CTRL_INIT*

KVM: Access in-kernel AIA irqchip in user-space

VENTANA MICRO
RISC-V Performance Leader

- Inject emulated IRQs
  - Update wired interrupt line state using **KVM_IRQ_LINE** ioctl()
  - Signal MSI using **KVM_SIGNAL_MSI** ioctl()

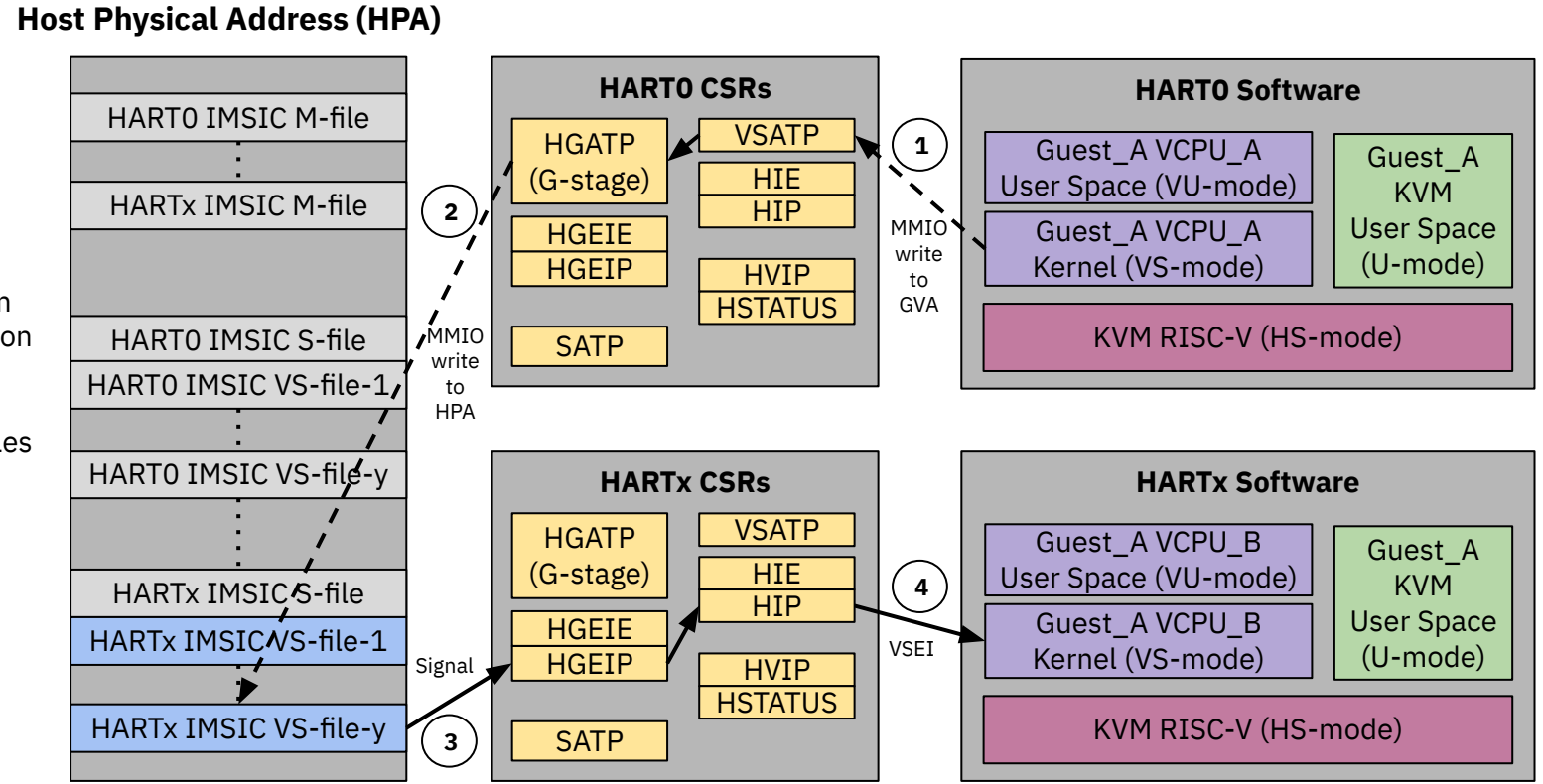
- Access 32-bit wide APLIC registers
  - Use **KVM_GET/SET_DEVICE_ATTR** ioctl() to read/write APLIC register
    - *group=KVM_DEV_RISCV_AIA_GRP_APLIC*
    - *attr=<aplic_register_offset>*

- Access XLEN-bit wide IMSIC registers
  - Use **KVM_GET/SET_DEVICE_ATTR** ioctl() to read/write IMSIC register
    - *group=KVM_DEV_RISCV_AIA_GRP_IMSIC*
    - *attr=KVM_DEV_RISCV_AIA_IMSIC_MKATTR(<vcpu_id>, <imsic_isel>)*

KVM: Virtual IPIs using IMSIC VS-files

VENTANA MICRO
RISC-V Performance Leader

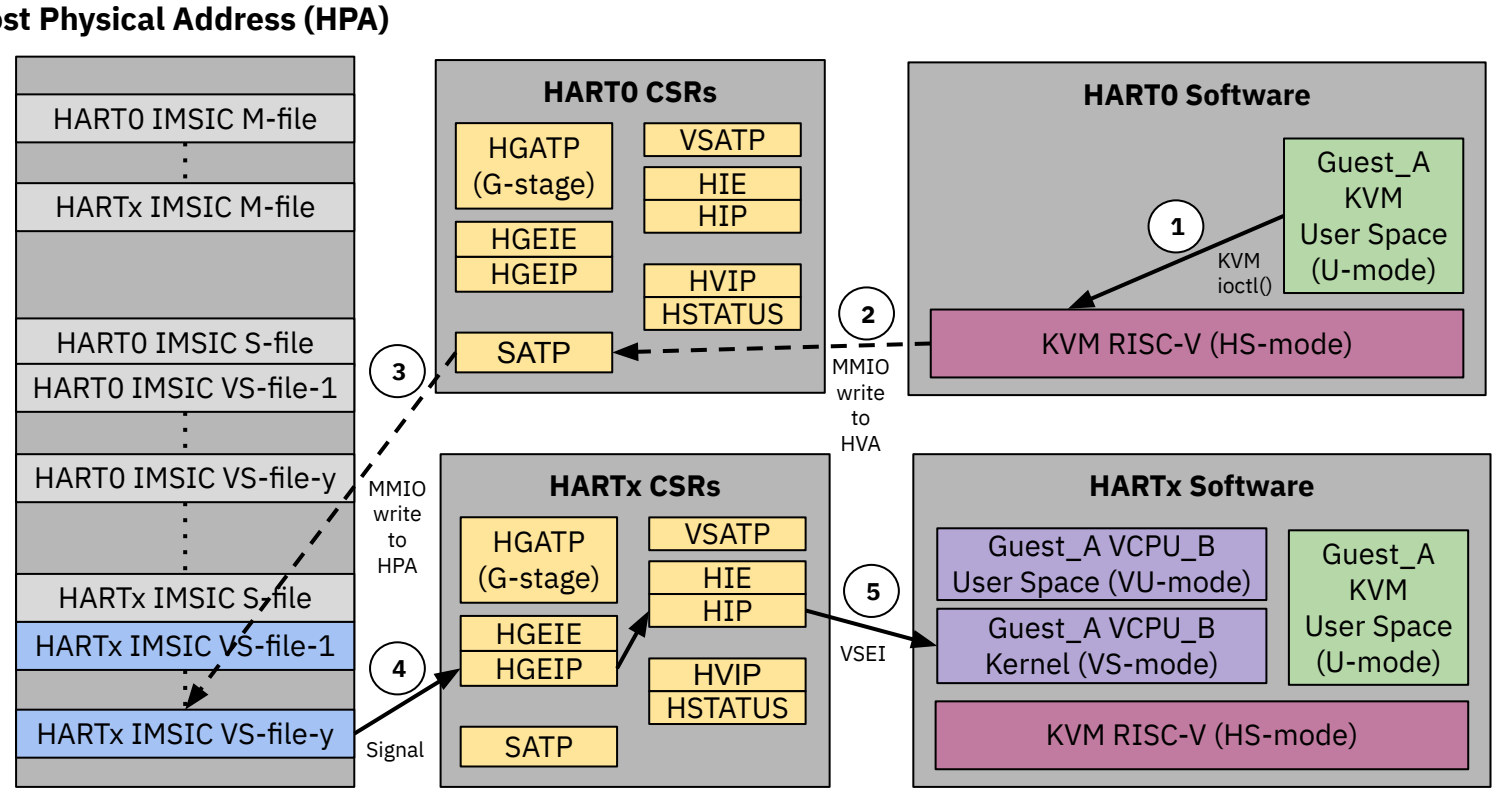
Example: IPI from Guest_A VCPU_A on HART0 targeting Guest_A VCPU_B using IMSIC VS-files on HARTx



KVM: Emulated IRQs using IMSIC VS-files

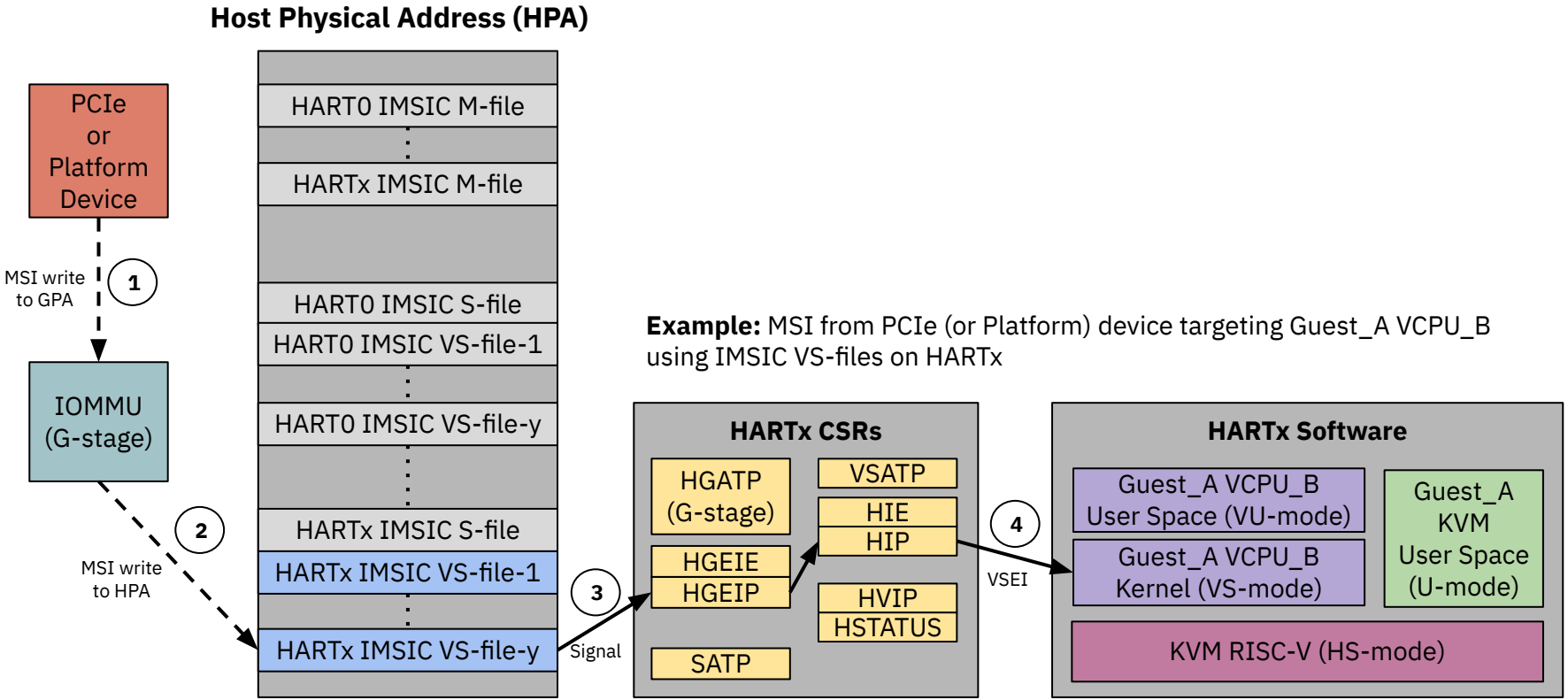
VENTANA MICRO
RISC-V Performance Leader

Example: MSI from software emulated (or paravirt) device on HART0 targeting Guest_A VCPU_B using IMSIC VS-file-y on HARTx



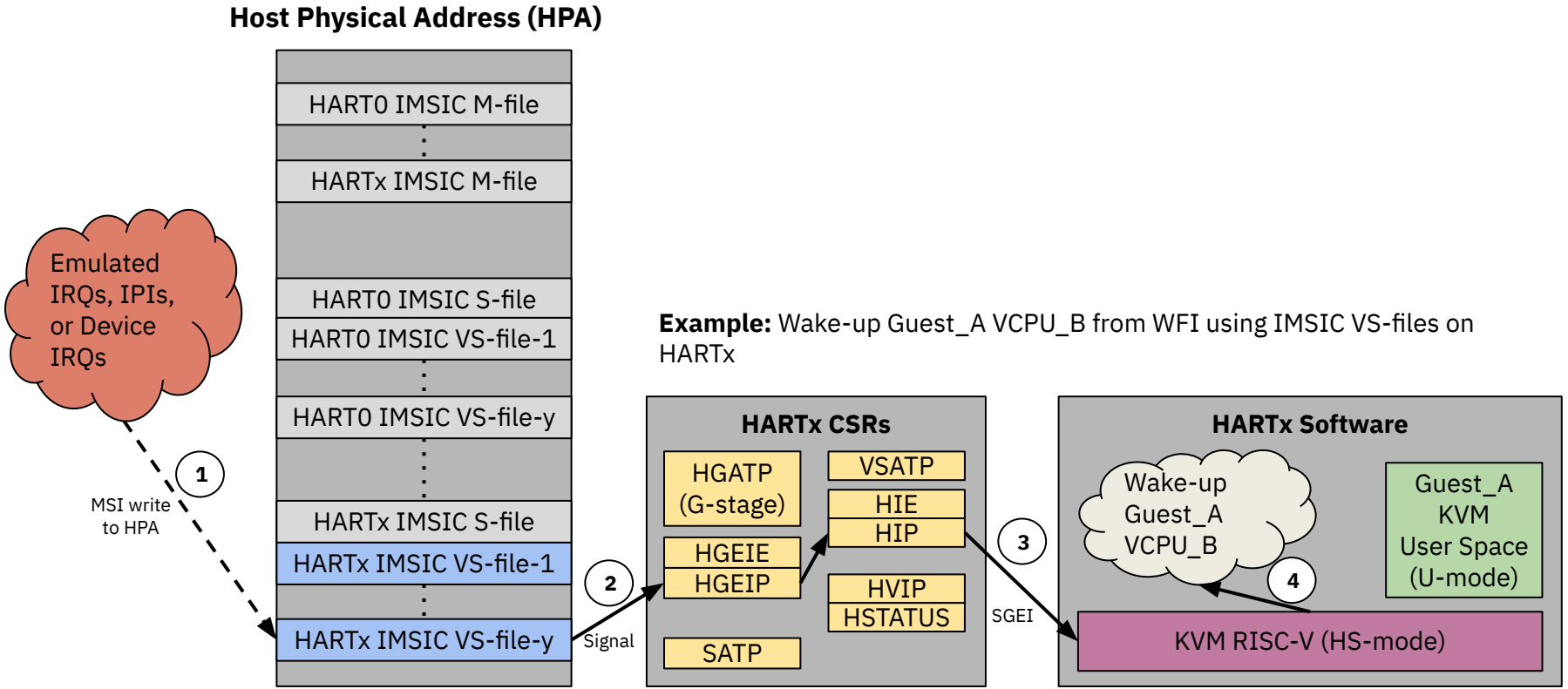
KVM: Device MSIs using IMSIC VS-files

VENTANA MICRO
RISC-V Performance Leader



Example: MSI from PCIe (or Platform) device targeting Guest_A VCPU_B using IMSIC VS-files on HARTx

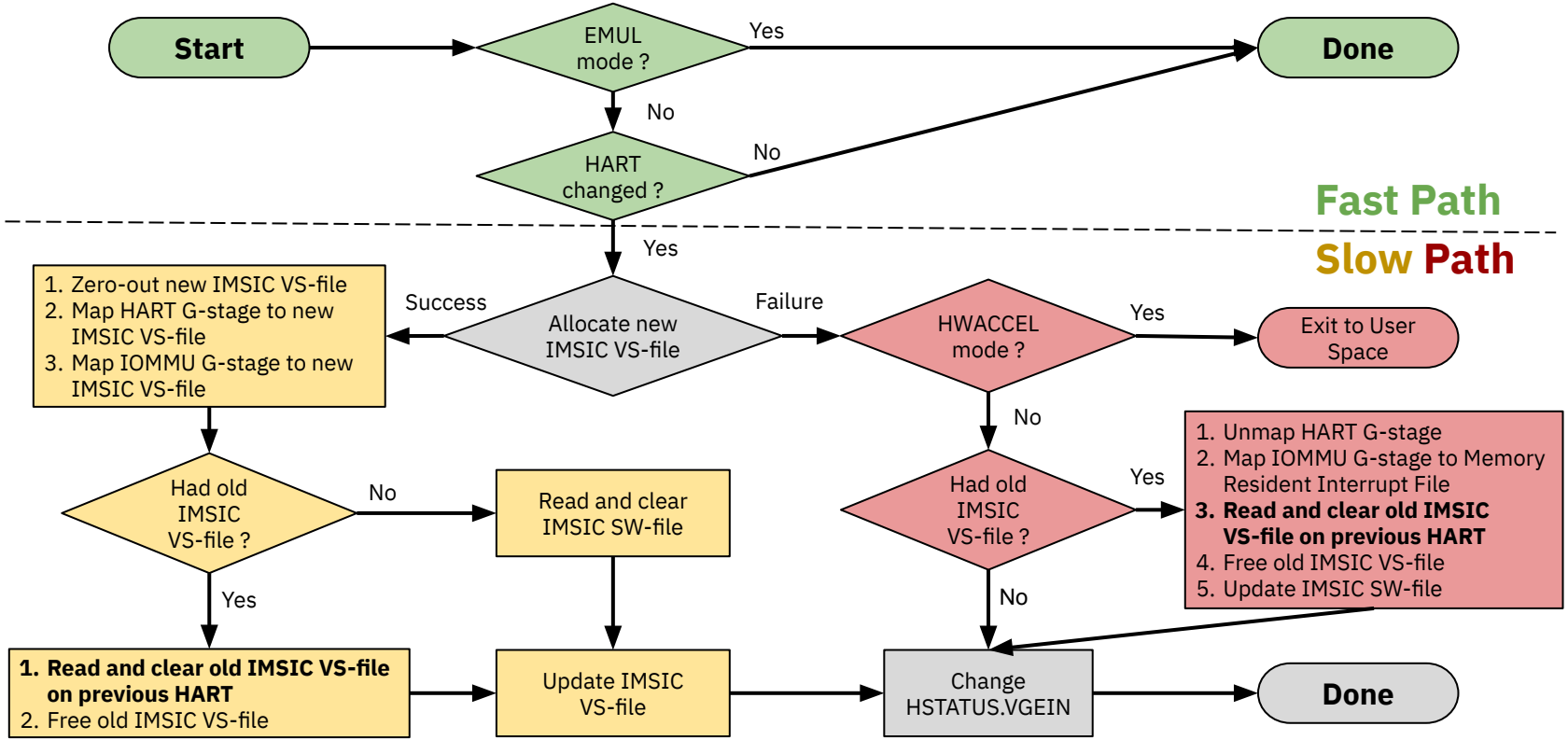
KVM: WFI wake-up using IMSIC VS-files



Example: Wake-up Guest_A VCPU_B from WFI using IMSIC VS-files on HARTx

KVM: Updating IMSIC file in VCPU run-loop

VENTANA MICRO
RISC-V Performance Leader



AIA Software Status and Demo

AIA Software Status

VENTANA MICRO
RISC-V Performance Leader

- Complete proof-of-concept done (QEMU, OpenSBI, Linux, KVM, and KVMTOOL)
- Device tree and ACPI support
  - Already reviewed on RISC-V AIA TG and RISC-V Hypervisors SIG mailing lists
  - Need to send out RFC PATCHES for review on Linux mailing lists
  - Need to send out ACPI ECRs to UEFI forum
- QEMU and OpenSBI patches already upstreamed
- Linux, KVM, and KVMTOOL patches yet to be sent for review
  - Branch riscv_aia_v1 at https://github.com/avpatel/linux (Linux AIA Drivers)
  - Branch riscv_kvm_aia_v1 at https://github.com/avpatel/linux (KVM AIA support)
  - Branch riscv_aia_v1 at https://github.com/avpatel/kvmtool (KVM AIA user-space support)
- **Live Demo !!!**

Thank You !!!