linaro™ Arm Solutions at Lightspeed

Virtio-msg

Making Virtio Work Where It Does Not Today

Bill Mills, Linaro

Why Virtio-msg?

Virtio

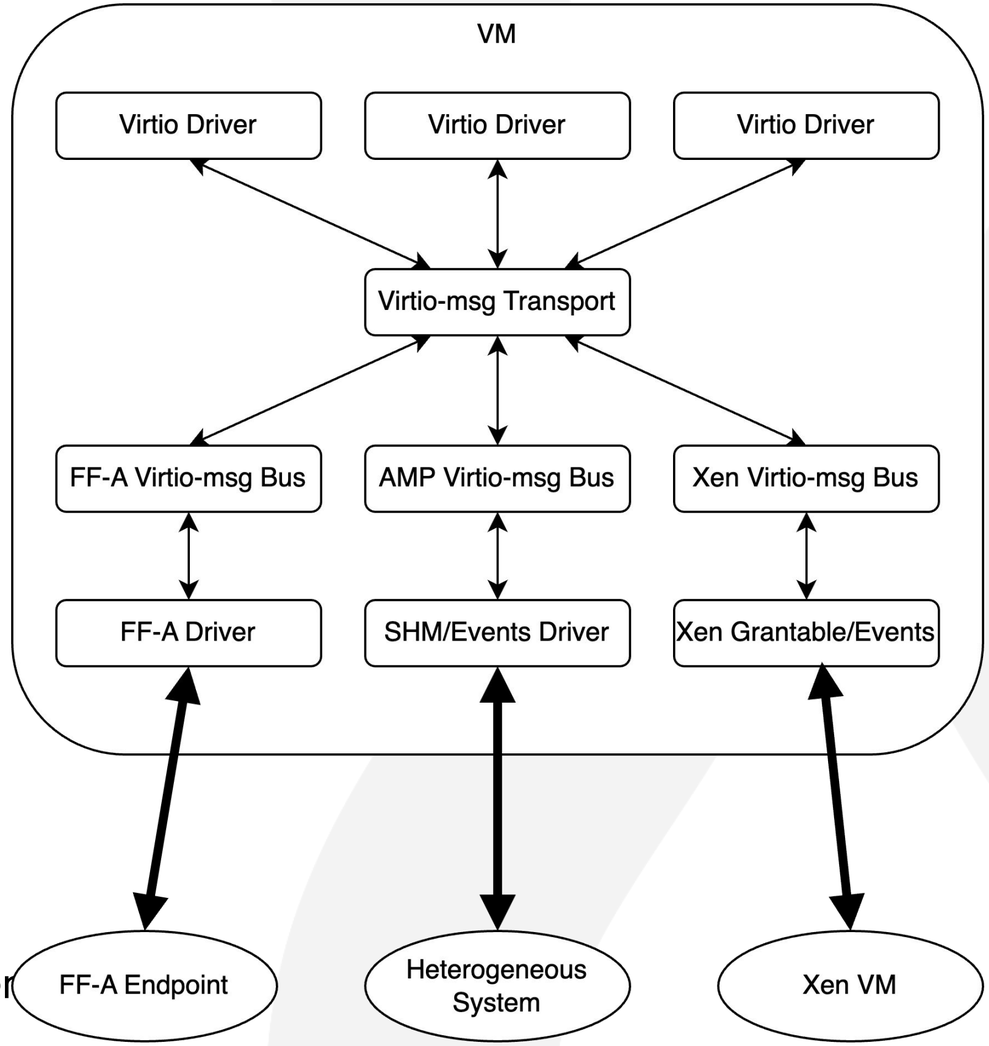
- Already supported by many OS'es, bootloaders, and hypervisors
- New projects can pick one PV device interface on which to focus
- The one to choose when you can only afford to choose one.
- Existing transport layers: virtio-pci, virtio-mmio, virtio-channel (IBM specific)
- Existing transports **assume** a hypervisor environment is available to **trap and emulate read and write accesses**

Virtio-msg

- New transport that removes trap and emulate assumptions by using messages
- Many messages are request/response putting the driver in control of how it waits for the response
- **This transport can be made to work even where existing virtio transports cannot or should not.**

Virtio-msg

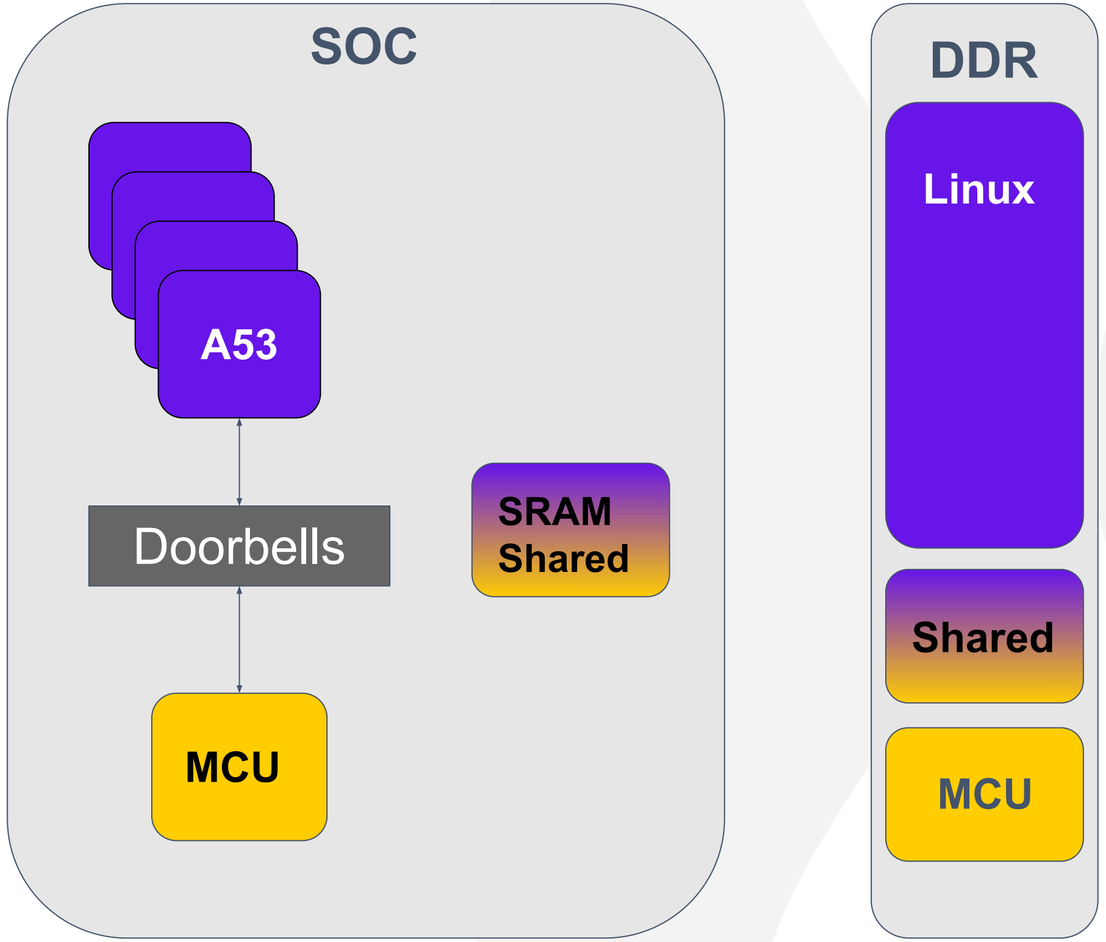
- New Virtio Transport
  - along side PCI, MMIO and Channel I/O
  - Message based
  - Large range of Targets
    - Heterogeneous, Secure/Non-secure, VM to VM, …etc
- Layer based design
  - Virtio-msg Transport
    - API to Message
    - Agnostic of the communication medium
  - Virtio-msg Bus
    - Communication specific
      - IRQ, Shared mem, Hypercall, Xen, FF-A, …etc
    - Roles:
      - Bus and Device discovery
      - Bus and Device lifecycle (hotplug)
      - Optional DMA management
      - Message transport between Frontend and Backer



Virtio-msg Examples – AMP SOC

AMP SOC

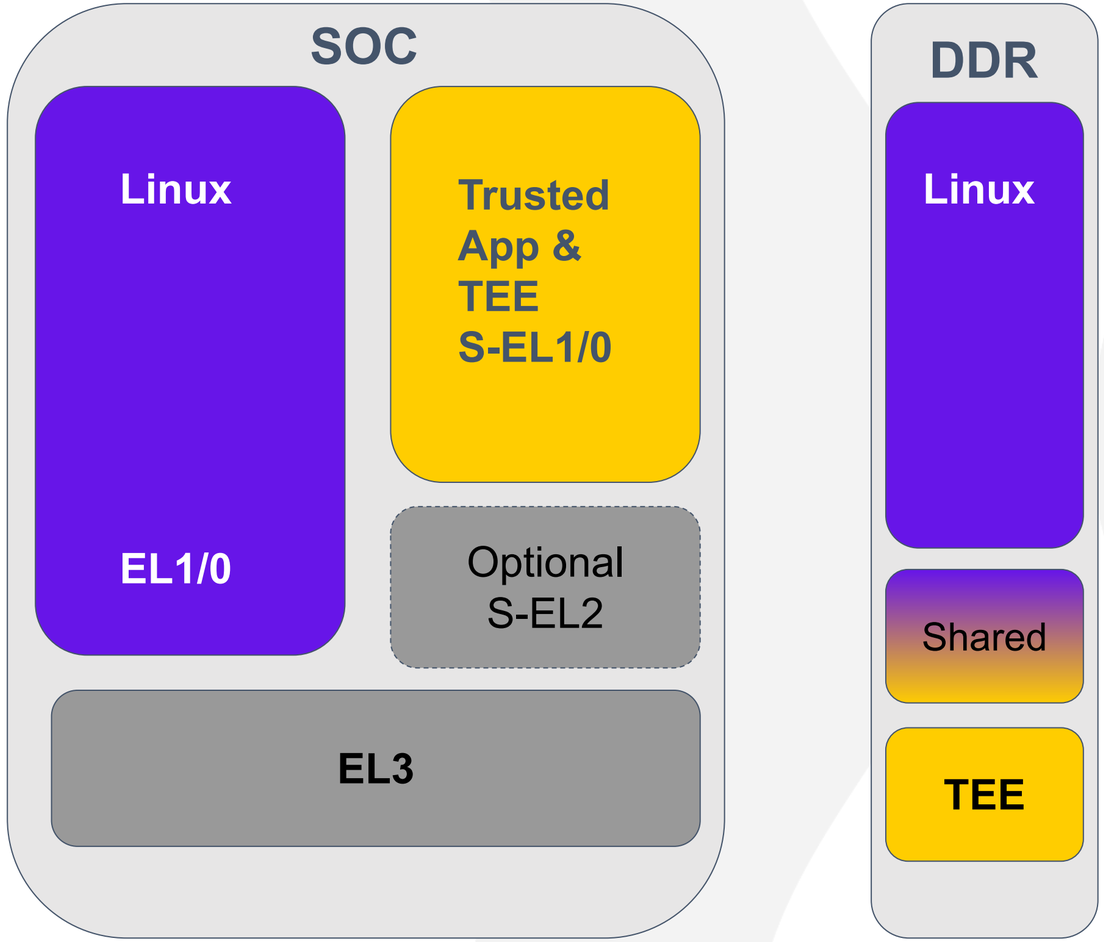
- A single SOC
- CPUs that are SMP Linux capable
- Examples: MCU device
  - MCU owned I^2C, SPI, GPIO shared w/ Linux
  - Multiple virtual ttys for each MCU
  - Co-exists with RPMsg or VSOCK
- Examples: MCU driver
  - MCU access to Linux resources
  - Filesystem, block devices, network
- Example SOCs:
  - ZynqMP: 4x A53s + 2x R5s [+FPGA]
  - STM32MP15: 2x A7s + 1 M4
  - NXP iMX8M+: 4x A53s + 1 M7 + DSP
  - TI TDA4VM: 2x A72s + 6x R5s + 3 DSPs



Demo running on ST SOC now!

Virtio-msg Examples – Normal and Secure World

- Secure and Normal worlds do not share a hypervisor, trap and emulate is not possible
- Virtio-msg over hypercalls solves this and keeps EL2 & EL3 simple
- Arm's FF-A interface standardizes these interfaces and can make VM to VM look similar as well
- Use cases:
  - Rich easy communication via VSOCK
  - Access to Secure world owned I^2C, SPI, GPIO, RNG, RTC, etc

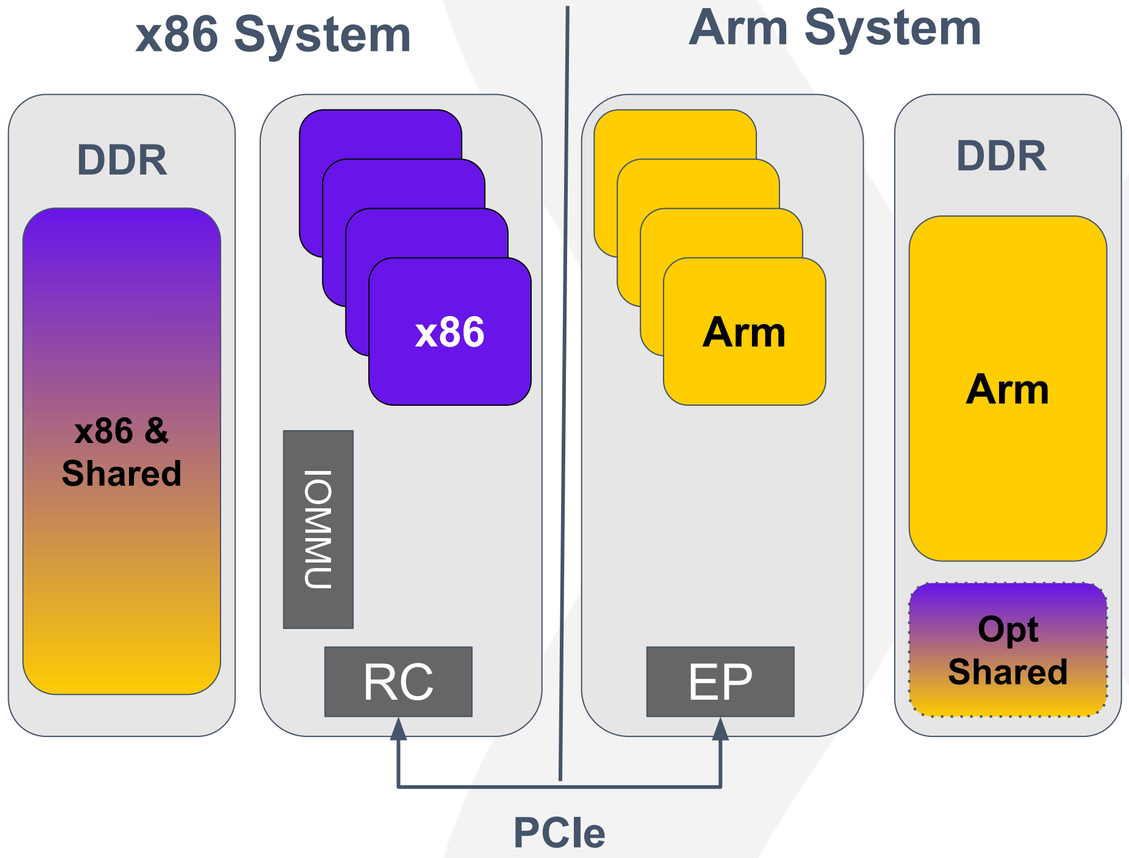


Google, Arm, and Linaro have demoed this running between Trusty Unikernel pVM and Trusty TEE!

Virtio-msg Examples – AMP System via PCIe

Virtio-msg via PCIe (and similar)

- x86 host with Arm SOC on a PCIe card
- x86 side is driver side and is standard except for the message transport
  - x86 memory is shared and accessed via IOMMU to the PCI card
- Other systems like this:
  - PCIe systems w/ non-transparent bridge
  - UCIe and Chiplet ecosystem
  - SmartNIC PCIe card
- Use cases
  - net, blk, filesystem, vsock, consoles, shared display etc



AMD and Linaro demoed this system at Embedded World 2025

Also see "Virtio for PCI Endpoint Subsystem in Linux Kernel" here at OSS NA (was Tuesday 11:55am)

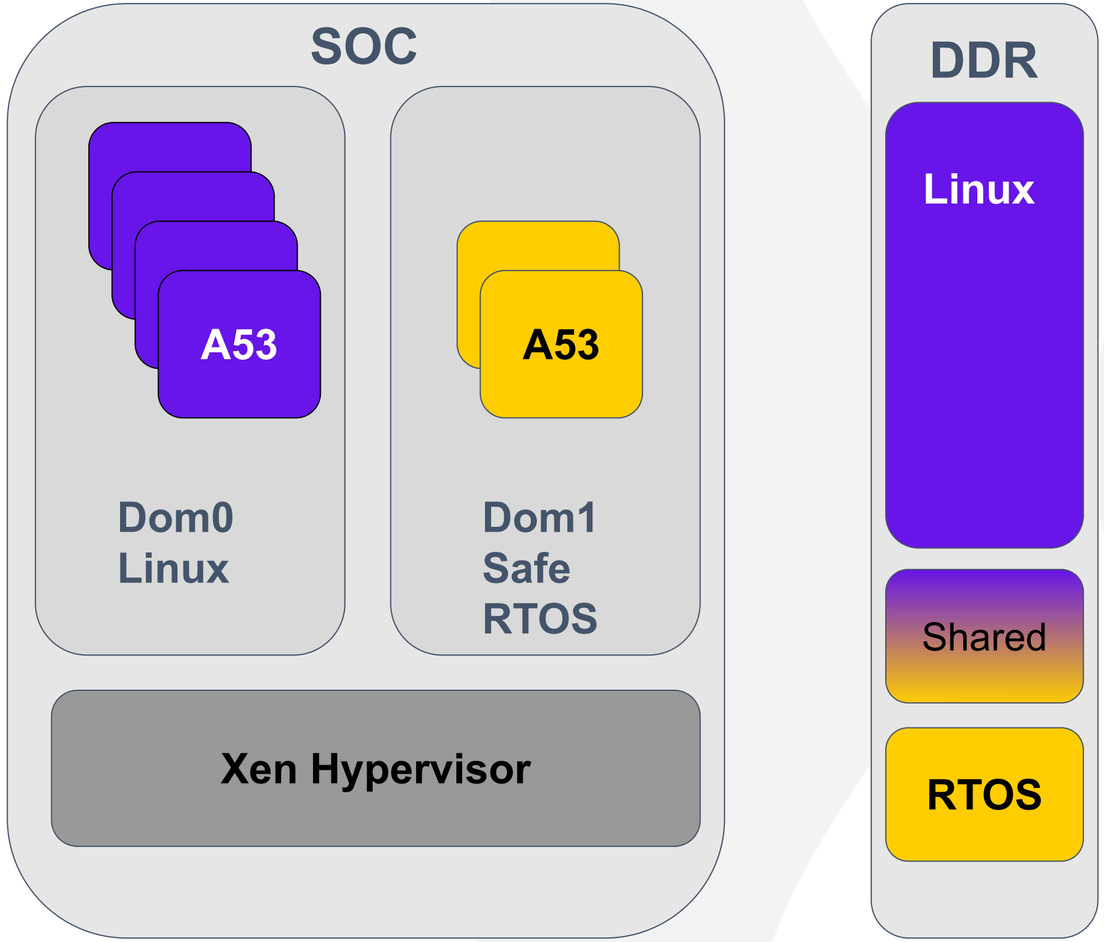
Virtio-msg Examples – VM to VM

Virtio-msg in Xen and other hypervisors

- Allow Virtio in Mixed Critical system
  - Dom1 is secure and safety certified
  - Dom0 is not, it is “Quality Managed”
- Virtio-msg does NOT steal cpu cycles from safe or secure VM and give them to Linux
  - Critical to Freedom from Interference
  - Reduces Trusted Compute Base
  - Important for Real Time latency
- Dom1 can use devices provided by Dom0 but can continue critical functions if they fail

Demo using Xen with FF-A running today!

We hope to show more hypervisors soon



Architecture

Virtio-msg Common Transport Level

Request / Response Messages

- **GET_DEVICE_INFO**:
  - Get Vendor/Device type & other static parameters
- **{GET,SET}_DEVICE_STATUS**:
  - Driver: report init progress or reset
  - Device: request reset
- **GET_DEVICE_FEATURES**, **SET_DRIVER_FEATURES**:
  - Negotiate features
- **{GET,SET}_CONFIG**:
  - Access config data area
- **{GET,SET,RESET}_VQUEUE**:
  - Configure the virtqueues
- **GET_SHM**:
  - Support virtio 1.3 Shared memory

Notification Messages

- **EVENT_CONFIG**:
  - Device -> Driver
  - Status Change
  - Config Change (w/ optional data)
- **EVENT_AVAIL**:
  - Driver -> Device
  - New data in virtqueue
- **EVENT_USED**:
  - Device -> Driver
  - Returned data in virtqueue

Virtio-msg Bus Level

Bus Operations

- Bus discovery
  - Example: DTB or PCI VID:PID
  - One system can have multiple busses
- Send and Receive Messages
  - Core of a bus implementation
- Bus Level Messages
- Virtio Device enumeration & creation
  - Via DT or via Bus messages
  - One bus can support multiple device
- Setup DMA layer
  - *see next slide*
- Monitor health *(optional)*
  - Check other side health and teardown if needed

Notification Methods

- Transport event messages in-band
- Convert messages to **out-of-band notifications** and vis-veras: Example: use one MSIX for each virtqueue
- As a last resort, periodically create event messages to cause polling

Bus Level Messages

- co-exist in message queue
- A few standardized messages exist
- Bus specific ID range is available

Virtio-msg Memory Handling (DMA Layer)

Driver Memory Placement:

- Unconstrained Memory
  - Driver puts virtqueues and buffers anywhere in its memory space
- Slightly Constrained Memory
  - Driver puts virtqueues and buffers anywhere in memory below a specific limit
  - *dma_set_{,coherent_}mask*
- Constrained Memory
  - Only a specific set of memory areas are shared between driver and device
  - virtqueues must be placed in these areas
  - buffers outside of these areas will be *“bounced”* into them (ex: via *swiotlb*)

Driver Memory Access Control:

- Driver may need to enable access via IOMMU or equivalent (Xen Page Grants)

Memory coherency:

- Normally virtio operates with full memory coherency
- This may or may not be true for virtio-msg
- Constrained memory areas may be mapped uncached
- SWIOTLB may need to do cache flush and invalidate

Memory addresses:

- Virtio normally uses driver side PA for:
  - Virtqueue Addresses
  - Buffer Addresses
- The DMA layer can arrange other forms:
  - IOVA from IOMMU
  - Area ID: Offset stuffed into 64 bits
  - In any case the driver must ensure the device understands the addresses used and may use bus messages to inform the device

Status

Status

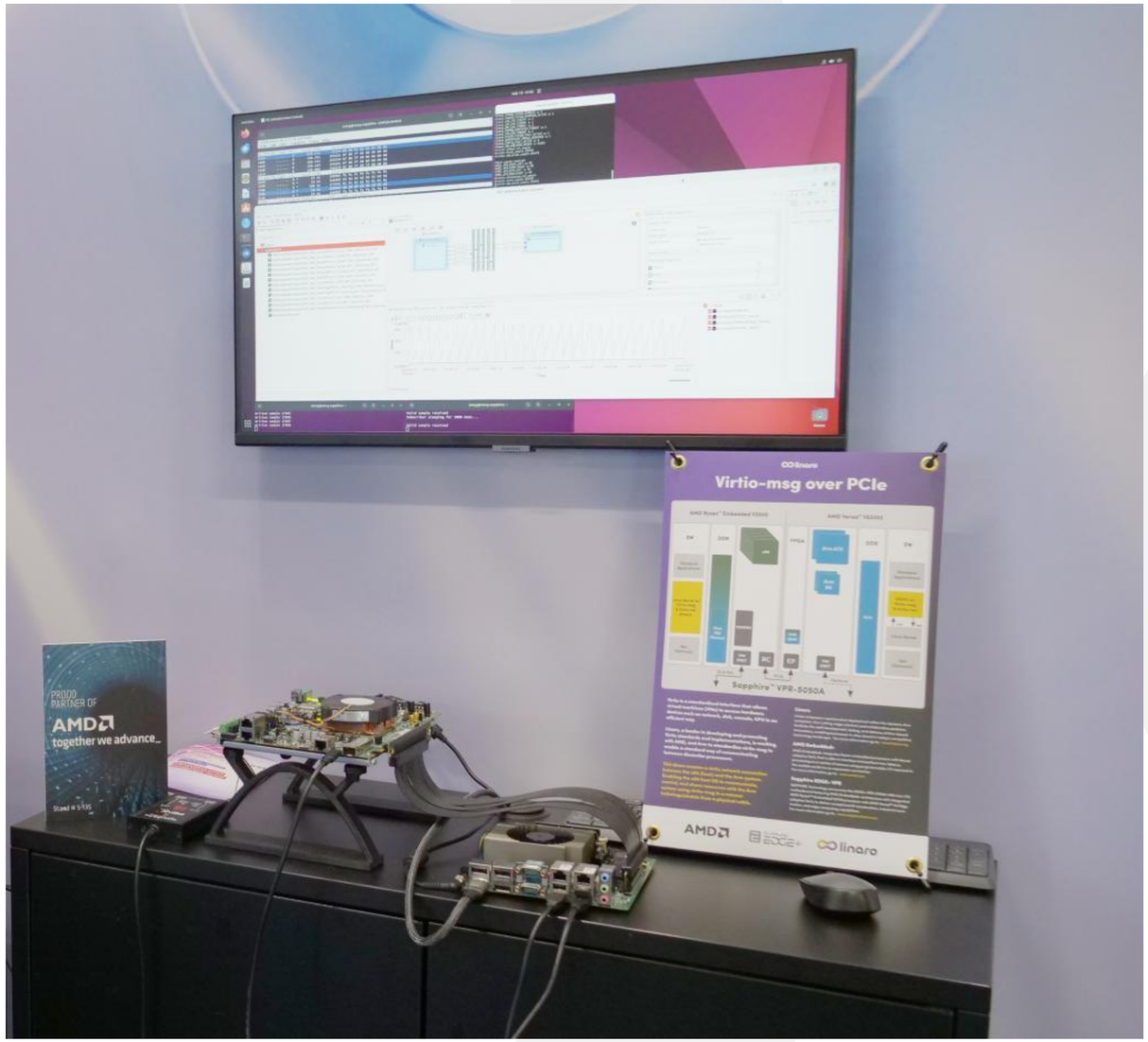
- Home page for project HVAC
  - (used to coordinate work on virtio-msg)
- Work in progress Spec:
  - Latest PDF repo
  - RFC2 was sent June 20th
- Run the emulated demos:
  - **`podman run -it --rm docker.io/wmills/hvac-demo`**
  - or clone and run: https://github.com/wmamills/hvac-demo
- AMD HW demo X86 to ARM via PCI
  - Shown at Embedded World 2025
- ST HW demo
  - M4 Zephyr I2C device
  - A7 Linux driver
  - Zephyr/RTOS work for virtio-msg is done in the context of the OpenAMP project
- Android AOSP w/ pKVM and Trusty
  - *See Virtio Meets Arm FF-A*

Thanks and Credits

Thank you for your attention!

Thanks to the all the virtio-msg contributors:

Bertrand Marquis @ ARM
Edgar Iglesias @ AMD
Armelle Laine and others @ Google
Viresh Kumar @ Linaro
Arnaud Pouliquen @ STMicroelectronics



AMD X86 + Arm Versal at EW2025

Linaro HVAC Project Home Page

Ask about the OpenAMP stickers

linaro.org

Thank you