



# Testing the Untestable – Using QEMU to test the Linux PCI Endpoint framework

Manivannan Sadhasivam

Senior Staff Engineer, Qualcomm

Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries.  
Qualcomm patents are licensed by Qualcomm Incorporated.



# Testing PCI Endpoint Framework

# Testing PCI Endpoint Framework

- The Linux PCI Endpoint framework
  - Allows Linux Kernel to run on the Endpoint itself and control it
  - Endpoint Controller (EPC) layer controls the hardware Endpoint Controller
  - Endpoint Function (EPF) layer implements the device functionality
  - ConfigFS binds them

# Testing PCI Endpoint Framework

- The Linux PCI Endpoint framework
  - Allows Linux Kernel to run on the Endpoint itself and control it
  - Endpoint Controller (EPC) layer controls the hardware Endpoint Controller
  - Endpoint Function (EPF) layer implements the device functionality
  - ConfigFS binds them
- Kselftests are the standard way to test it
  - But it requires connecting two Linux systems over PCIe and control both
  - Difficult to hook it up in CI

# Testing Endpoint Framework with QEMU

# Testing Endpoint Framework with QEMU

- Using QEMU to test Endpoint Framework was first proposed by Shunsuke Mie
  - QEMU series posted, discussed, not merged
  - [lore.kernel.org/qemu-devel/CANXvt5oKt=AKdqv24LT079e+6URnfqJcfTJh0ajGA17paJUEKw@mail.gmail.com](https://lore.kernel.org/qemu-devel/CANXvt5oKt=AKdqv24LT079e+6URnfqJcfTJh0ajGA17paJUEKw@mail.gmail.com)

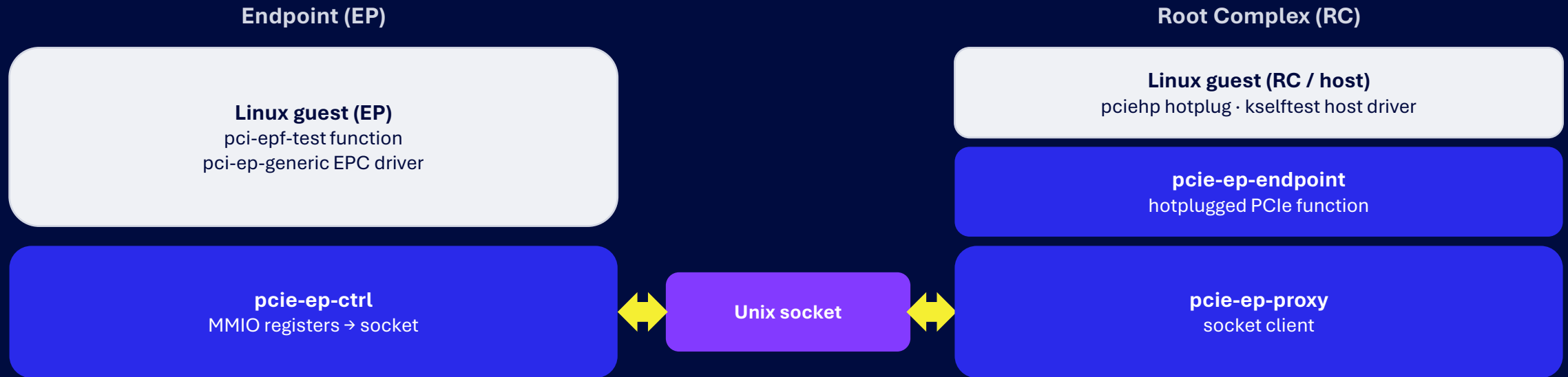
# Testing Endpoint Framework with QEMU

- Using QEMU to test Endpoint Framework was first proposed by Shunsuke Mie
  - QEMU series posted, discussed, not merged
  - [lore.kernel.org/qemu-devel/CANXvt5oKt=AKdqv24LT079e+6URnfqJcfTJh0ajGA17paJUEKw@mail.gmail.com](https://lore.kernel.org/qemu-devel/CANXvt5oKt=AKdqv24LT079e+6URnfqJcfTJh0ajGA17paJUEKw@mail.gmail.com)
- My first take
  - Two QEMU instances over a socket
  - One acting as Endpoint, one acting as host

# Testing Endpoint Framework with QEMU

- Using QEMU to test Endpoint Framework was first proposed by Shunsuke Mie
  - QEMU series posted, discussed, not merged
  - [lore.kernel.org/qemu-devel/CANXvt5oKt=AKdqv24LT079e+6URnfqJcfTJh0ajGA17paJUEKw@mail.gmail.com](https://lore.kernel.org/qemu-devel/CANXvt5oKt=AKdqv24LT079e+6URnfqJcfTJh0ajGA17paJUEKw@mail.gmail.com)
- My first take
  - Two QEMU instances over a socket
  - One acting as Endpoint, one acting as host
- Kselftests Result
  - pass:10 fail:10 skip:3 (filtered pass:10 fail:0)
  - No DMA, INTx, Doorbell

# First design: Two QEMU instances over a socket



# Rethinking the design: One QEMU instance

# Rethinking the design: One QEMU instance

- Suggested by QEMU maintainers
  - Socket + two address spaces add complexity

# Rethinking the design: One QEMU instance

- Suggested by QEMU maintainers
  - Socket + two address spaces add complexity
- New design: RC and EP in one QEMU
  - One kernel, one guest address space
  - In-process calls, BARs are memory aliases

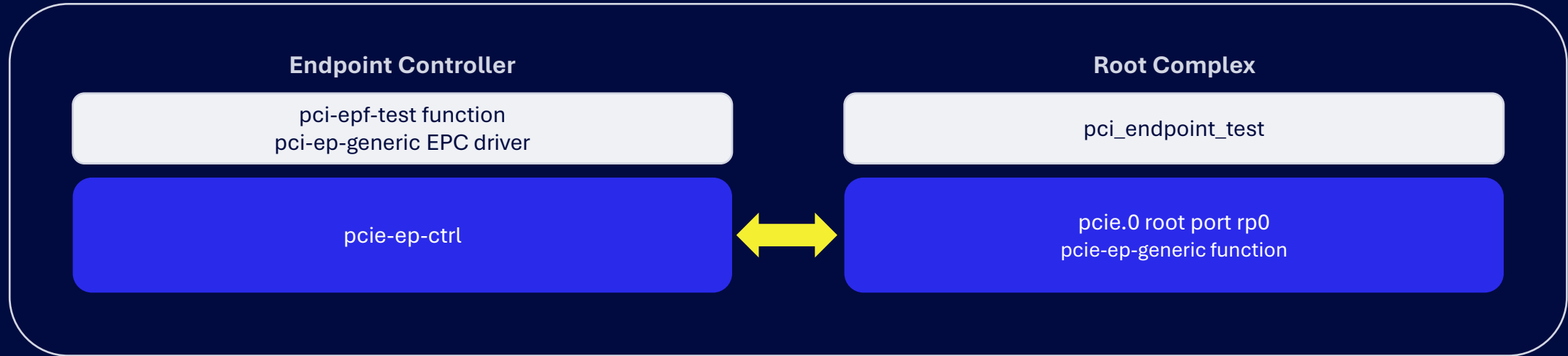
# Rethinking the design: One QEMU instance

- Suggested by QEMU maintainers
  - Socket + two address spaces add complexity
- New design: RC and EP in one QEMU
  - One kernel, one guest address space
  - In-process calls, BARs are memory aliases
- Simpler design, much less code
  - [lore.kernel.org/qemu-devel/20260913-pcie-ep-emulation-v1-0-b8e8f74a5842@kernel.org](https://lore.kernel.org/qemu-devel/20260913-pcie-ep-emulation-v1-0-b8e8f74a5842@kernel.org)

# Rethinking the design: One QEMU instance

- Suggested by QEMU maintainers
  - Socket + two address spaces add complexity
- New design: RC and EP in one QEMU
  - One kernel, one guest address space
  - In-process calls, BARs are memory aliases
- Simpler design, much less code
  - [lore.kernel.org/qemu-devel/20260913-pcie-ep-emulation-v1-0-b8e8f74a5842@kernel.org](https://lore.kernel.org/qemu-devel/20260913-pcie-ep-emulation-v1-0-b8e8f74a5842@kernel.org)
- Kselftests Result
  - pass:10 fail:10 skip:3 (filtered pass:10 fail:0)
  - No DMA, INTx, Doorbell

## A simpler model: Single QEMU instance



What single instance gives up

# What single instance gives up

- Independent endpoint lifecycle
  - EP reboot or crash while host stays up

# What single instance gives up

- Independent endpoint lifecycle
  - EP reboot or crash while host stays up
- Link events
  - Link down, retrain, hot reset

# What single instance gives up

- Independent endpoint lifecycle
  - EP reboot or crash while host stays up
- Link events
  - Link down, retrain, hot reset
- Surprise removal

# What single instance gives up

- Independent endpoint lifecycle
  - EP reboot or crash while host stays up
- Link events
  - Link down, retrain, hot reset
- Surprise removal
- Address space isolation
  - Aliasing can hide bad translations

# What single instance gives up

- Independent endpoint lifecycle
  - EP reboot or crash while host stays up
- Link events
  - Link down, retrain, hot reset
- Surprise removal
- Address space isolation
  - Aliasing can hide bad translations
- Heterogeneous setup
  - Different host and EP kernels

TODO

# TODO

- Add QEMU functional tests
  - Automate the kselftest run for CI

# TODO

- Add QEMU functional tests
  - Automate the kselftest run for CI
- Extend to more Endpoint functions
  - MHI, NVMe, beyond pci-epf-test

# TODO

- Add QEMU functional tests
  - Automate the kselftest run for CI
- Extend to more Endpoint functions
  - MHI, NVMe, beyond pci-epf-test
- Upstream generic EPC Linux Kernel driver
  - <https://github.com/Mani-Sadhasivam/linux/commit/d23bc106b79579ddb979be6b141e71712a70acfa>

# TODO

- Add QEMU functional tests
  - Automate the kselftest run for CI
- Extend to more Endpoint functions
  - MHI, NVMe, beyond pci-epf-test
- Upstream generic EPC Linux Kernel driver
  - <https://github.com/Mani-Sadhasivam/linux/commit/d23bc106b79579ddb979be6b141e71712a70acfa>
- Fill functionality gaps
  - DMA, INTx routing, Doorbell

# Thank you

Nothing in these materials is an offer to sell any of the components or devices referenced herein.

© Qualcomm Technologies, Inc. and/or its affiliated companies. All Rights Reserved.

Qualcomm and Snapdragon are trademarks or registered trademarks of Qualcomm Incorporated.  
Other products and brand names may be trademarks or registered trademarks of their respective owners.

References in this presentation to “Qualcomm” may mean Qualcomm Incorporated, Qualcomm Technologies, Inc., and/or other subsidiaries or business units within the Qualcomm corporate structure, as applicable. Qualcomm Incorporated includes our licensing business, QTL, and the vast majority of our patent portfolio. Qualcomm Technologies, Inc., a subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of our engineering, research and development functions, and substantially all of our products and services businesses, including our QCT semiconductor business.

Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patents are licensed by Qualcomm Incorporated.

Follow us on: [in](#) [X](#) [@](#) [v](#) [f](#)

For more information, visit us at [qualcomm.com](https://www.qualcomm.com) & [qualcomm.com/blog](https://www.qualcomm.com/blog)

