



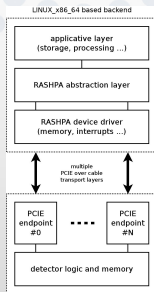
## The use of hardware virtualization in RASHPA

# Contents



- RASHPA context reminder
- Device virtualization
- The VPCIE framework

# Context - RASHPA reminder



RASHPA basic goal is to transfer **detector** memory contents into a **backend** for further processing, visualisation or storage. To do so, RASHPA relies on low level hardware and software components

- a data transport layer, currently PCI Express over cable
- a data transmission engine implemented on XILINX FPGA
- LINUX kernel device driver and userland software

# Context - RASHPA project issues



Still being prototyped

- how to simplify hardware software codesign ?

Aims at using multiple PCIE links. Yet, the first prototype is single link

- how to scale the prototype platform ?

Well positionned to interface with PCIE based accelerating technologies

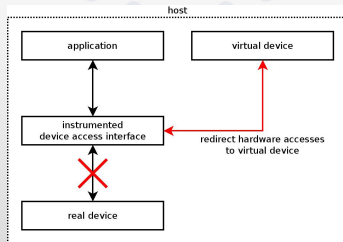
- how to test unavailable (too recent or expensive) hardware ?

# Context - solution



Solution: **device virtualization**

# Device virtualization - main idea



Applications on a **host** machine access hardware via interfaces. By **instrumenting** these interfaces, one can **redirect** the accesses to a software implementing the device. The device is said to be **virtualized**.

# Device virtualization - hardware access redirection (0)



Virtualization assumes device accesses are made using a known interface

- software library API (system calls)
- memory mapped registers
- CPU specific instructions (in, out)

# Device virtualization - hardware access redirection (1)



The interface is then instrumented to redirect access to the virtual device

- software hooking (code instrumentation, library replacing)
- instruction emulation (QEMU)
- hardware traps (page protection)
- architecture support (INTEL VT)
- paravirtualization (XEN)



# Device virtualization - applications



## Device virtualization example applications

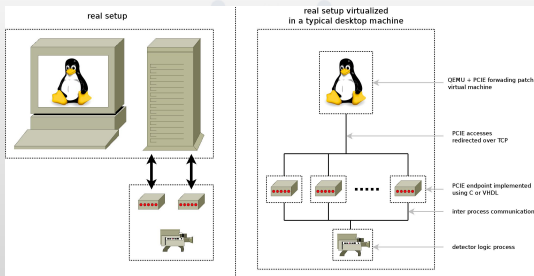
- support: application running on unmaintained platform
- security: sandboxing, reverse engineering
- quality: debugging, fault injection
- testing: milkymist, zinq, android

# VPCIE - goals

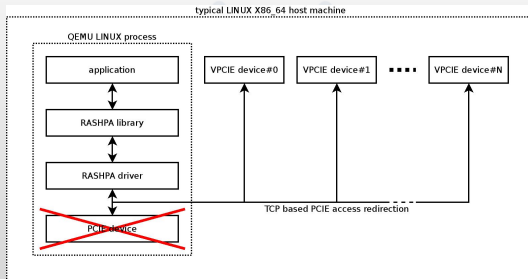


- virtualize PCIE endpoints for RASHPA
- made generic enough to fit other projects
- the backend software must run **without any modification**
- should be possible to use VHDL for device simulation

# VPCIE - RASHPA possible setup (0)



# VPCIE - RASHPA possible setup (1)



# VPCIE - opensource building blocks



## QEMU

- [http://wiki.qemu.org/Main\\_Page](http://wiki.qemu.org/Main_Page)
- architecture emulator (X86\_64, ARM ...)
- used to trap PCIE hardware accesses

## GHDL

- <http://ghdl.free.fr>
- VHDL frontend for GCC
- used to implement device in VHDL

# VPCIE - RASHPA backend



RASHPA backend is a full featured LINUX system

- runs in a QEMU virtual machine
- PCIE accesses are trapped and sent over TCP to the devices
- PCIE forwarder is available as a QEMU patch

# VPCIE - RASHPA devices



## RASHPA devices

- run as a LINUX processes
- can be implemented in C or VHDL
- can be duplicated at will

# VPCIE - benefits

## Hardware software codesign

- reduce development time
- no **modification** on the backend software (esp. driver)
- act as a stimuli generator for VHDL simulations

## Platform scaling

- one PCIE endpoint per LINUX process

## Investigate unavailable technologies

- NVM Express support for QEMU is available



# VPCIE - availability



<https://github.com/texane/vpcie>

# VPCIE - demo



demo