OHW2013 workshop

An open source PCIe device virtualization framework

CRISP
The Cluster of Research Infrastructures for Synergies in Physics

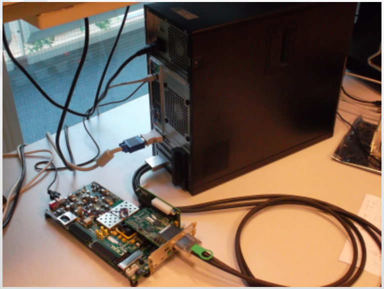
Plan

- Context and objectives
- Design and implementation
- Future directions
- Questions

Context - ESRF and the ISDD electronic laboratory

The ESRF is an XRAY light source for Europe located in Grenoble, France. The ISDD electronic laboratory mission is to develop and investigate **XRAY instrumentation electronics**.

Context - RASHPA



- high bandwidth data acquisition framework for 2D XRAY detectors
- Intel X86_64 workstation
- KC705 prototype board, PCIe over cable
- see ICALEPCS poster session

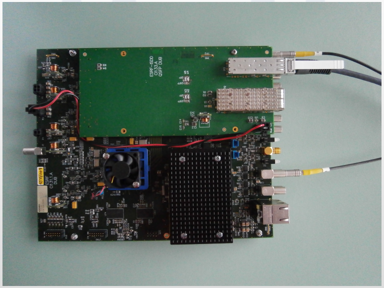
CRISP
The Cluster of Research Infrastructures for Synergies in Physics

Context - DCORE

- generic data acquisition and control boards
- Intel ATOM CPU
- Xilinx Spartan6 FPGA

CRISP
The Cluster of Research Infrastructures for Synergies in Physics

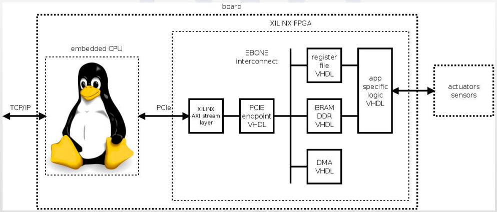
Context - DUB



- project specific readout electronics
- Intel ATOM CPU
- Xilinx Virtex6 FPGA

CRISP
The Cluster of Research Infrastructures for Synergies in Physics

VPCIe - rationale



The above boards rely on **PCIe** as the default CPU to FPGA communication link. VPCIe is a framework made to **virtualize** these platforms on a standard desktop PC.

VPCIe - objectives

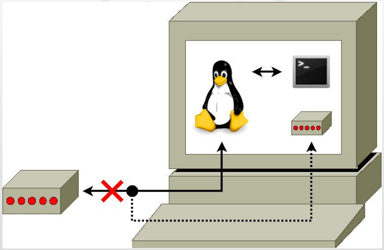
- CPU software must run **unmodified** (including **drivers**)
- few modifications allowed for **functional VHDL simulation**
- run **multiple PCIe devices** concurrently
- performances not critical

VPCIe - benefits

- hardware software **codesign**
- **reduce** development cycle time
- platform **scaling** (multiple PCIe devices)
- test with **different CPU architectures** (INTEL, ARM ...)
- investigate **unavailable technologies** (NVM EXPRESS ...)
- **testing** (fault injection ...)

CRISP
The Cluster of Research Infrastructures for Synergies in Physics

VPCIe - concepts



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**.

VPCIe - building blocks

VPCIe relies on opensource projects

QEMU

- 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

CRISP
The Cluster of Research Infrastructures for Synergies in Physics

VPCIe - CPU virtualization

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**
  - maintainers contacted for a merge

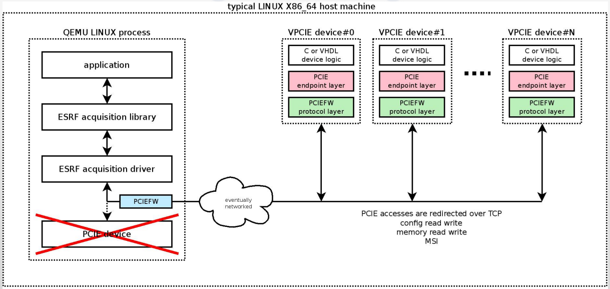
VPCIe - device virtualization

Virtual devices

- can be implemented in **C** or **VHDL**
  - GHDL is used to compile VHDL into a **native executable**
  - a glue interfaces the executable to the VPCIe runtime
- run as a LINUX processes, can be **duplicated** at will
- PCIe made **simple**, focus on device logic
  - but close to the XILINX PCIe transaction layer

CRISP The Cluster of Research Infrastructures for Synergies in Physics

VPCIe - implementation



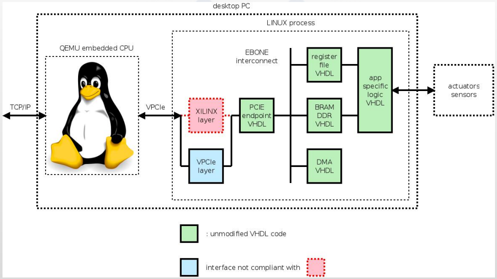
🟦 : PCIEFW, a patch for QEMU implementing PCIe request forwarding over TCP

🟩 : PCIEFW protocol handling on the device side

🟥 : runtime to simplify PCIe endpoint implementation and glue to upper layer

CRISP
The Cluster of Research Infrastructures for Synergies in Physics

VPCIe - EBONE based example



EBONE is a PCIe centric FPGA core interconnect developped at the ESRF and recently released on OHR (http://www.ohwr.org/projects/e-bone).

Excluding the PCIe layer, most of the VHDL remains **unchanged** in a typical EBONE based design.

VPCIe - virtualized device VHDL interface

```
entity endpoint is
 port
 (
  rst:          in std_ulogic;
  clk:          in std_ulogic;

  req_en:       out std_ulogic;
  req_wr:       out std_ulogic;
  req_bar:      out std_ulogic_vector(pcie.BAR_WIDTH - 1 downto 0);
  req_addr:     out std_ulogic_vector(pcie.ADDR_WIDTH - 1 downto 0);
  req_data:     out std_ulogic_vector(pcie.DATA_WIDTH - 1 downto 0);

  rep_en:       in std_ulogic;
  rep_data:     in std_ulogic_vector(pcie.DATA_WIDTH - 1 downto 0);

  mwr_en:       in std_ulogic;
  mwr_addr:     in std_ulogic_vector(pcie.ADDR_WIDTH - 1 downto 0);
  mwr_data:     in std_ulogic_vector(pcie.PAYLOAD_WIDTH - 1 downto 0);
  mwr_size:     in std_ulogic_vector(pcie.SIZE_WIDTH - 1 downto 0);

  msi_en:       in std_ulogic
 );
end entity;
```

The Cluster of Research Infrastructures for Synergies in Physics

VPCIe - virtualized device C interface

```
/* runtime initialization */
int      pcie_init_net(pcie_dev_t*, ...);
int      pcie_fini(pcie_dev_t*);
int      pcie_loop(pcie_dev_t*);

/* misc config byte accessors */
int      pcie_set_deviceid(pcie_dev_t*, ...);
int      pcie_set_vendorid(pcie_dev_t*, ...);

/* PCIe BAR access handlers */
typedef void (*pcie_readfn_t)(uint64_t, void*, size_t, void*);
typedef void (*pcie_writefn_t)(uint64_t, const void*, size_t, void*);
int      pcie_set_bar(pcie_dev_t*, ..., pcie_readfn_t, pcie_writefn_t, ...);

/* host memory read write operations */
int      pcie_write_host_mem(pcie_dev_t*, uint64_t, size_t*);
int      pcie_read_host_mem(pcie_dev_t*, uint64_t, size_t*);

/* send an MSI */
int      pcie_send_msi(pcie_dev_t*);
```

CRISP
The Cluster of Research Infrastructures for Synergies in Physics

VPCIe - future directions

- GHDL no longer maintained
- merge PCIe forwarder in QEMU
- XILINX AXI stream compatible PCIe layer
- reimplement PCIe forwarding and protocol
- NVM Express integration testing
- licensing

CRISP The Cluster of Research Infrastructures for Synergies in Physics

VPCIe - availability

VPCIe source is available online

- https://github.com/texane/vpcie
- documentation still poor, but clear examples
- feedbacks or contributions are welcome

VPCIe - questions

Thanks for your attention.

Any question?