Development of a modular and fully-digital PCIe-based interface to Real-Time Digital Simulator

Master Thesis

Defense Presentation

Steffen Vogel

ACS | Automation of Complex Power Systems

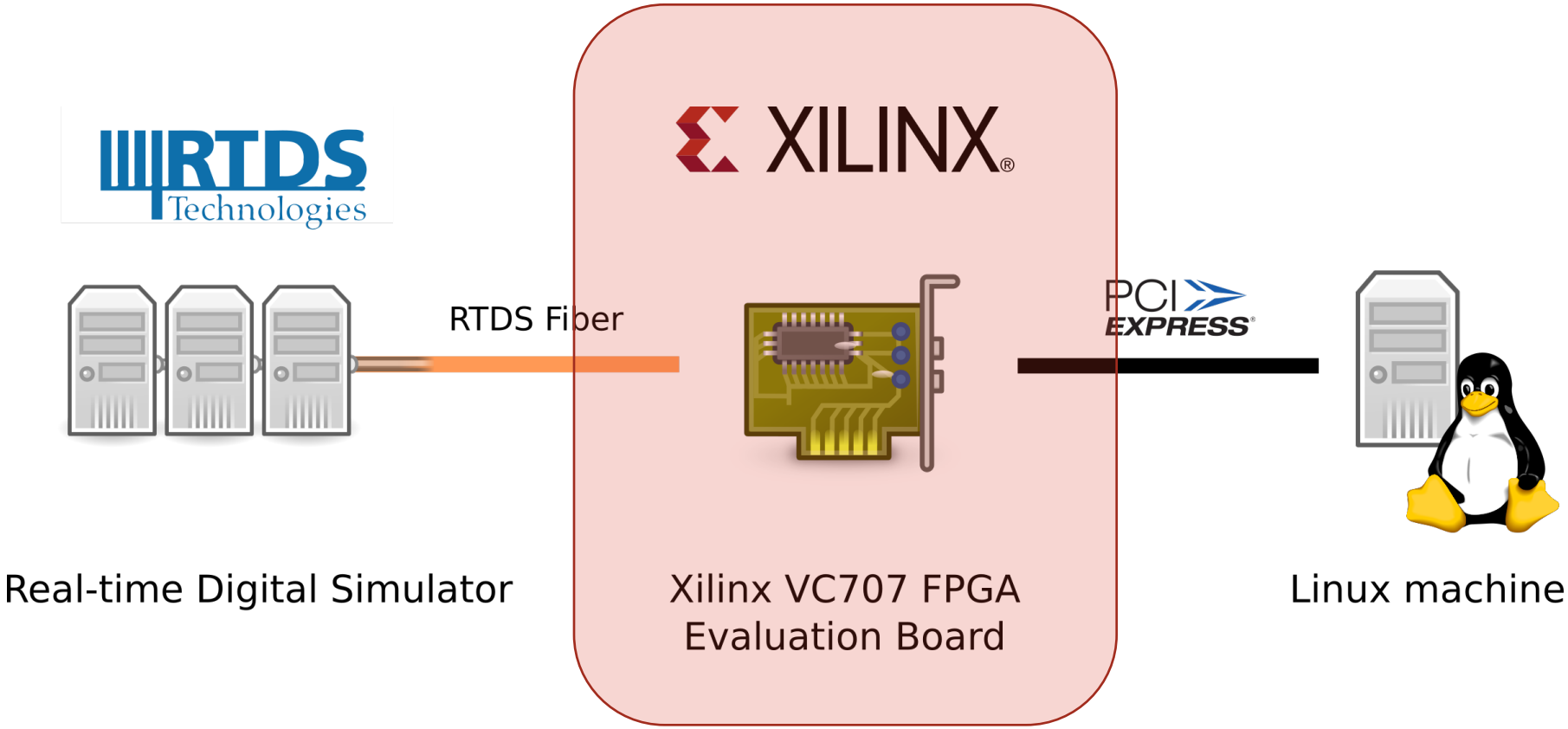
E.ON Energy Research Center

RWTH AACHEN UNIVERSITY

Agenda

- Objectives
- Architecture
  - VILLASfpga
  - VILLASnode
- Linux
  - Real-time Optimizations
  - Driver API
- FPGA
  - Datamovers / Synchronization
  - RTDS Interface
- Example Applications
- Future work

Original Motivation



High-level Architecture

Motivation

- Flexible framework
  - Extensible
  - Reuse of existing intefaces
- Integration into VILLASnode concept
  - Hard / Soft real-time
- No VHDL knowledge
  - C / C++, Simulink models

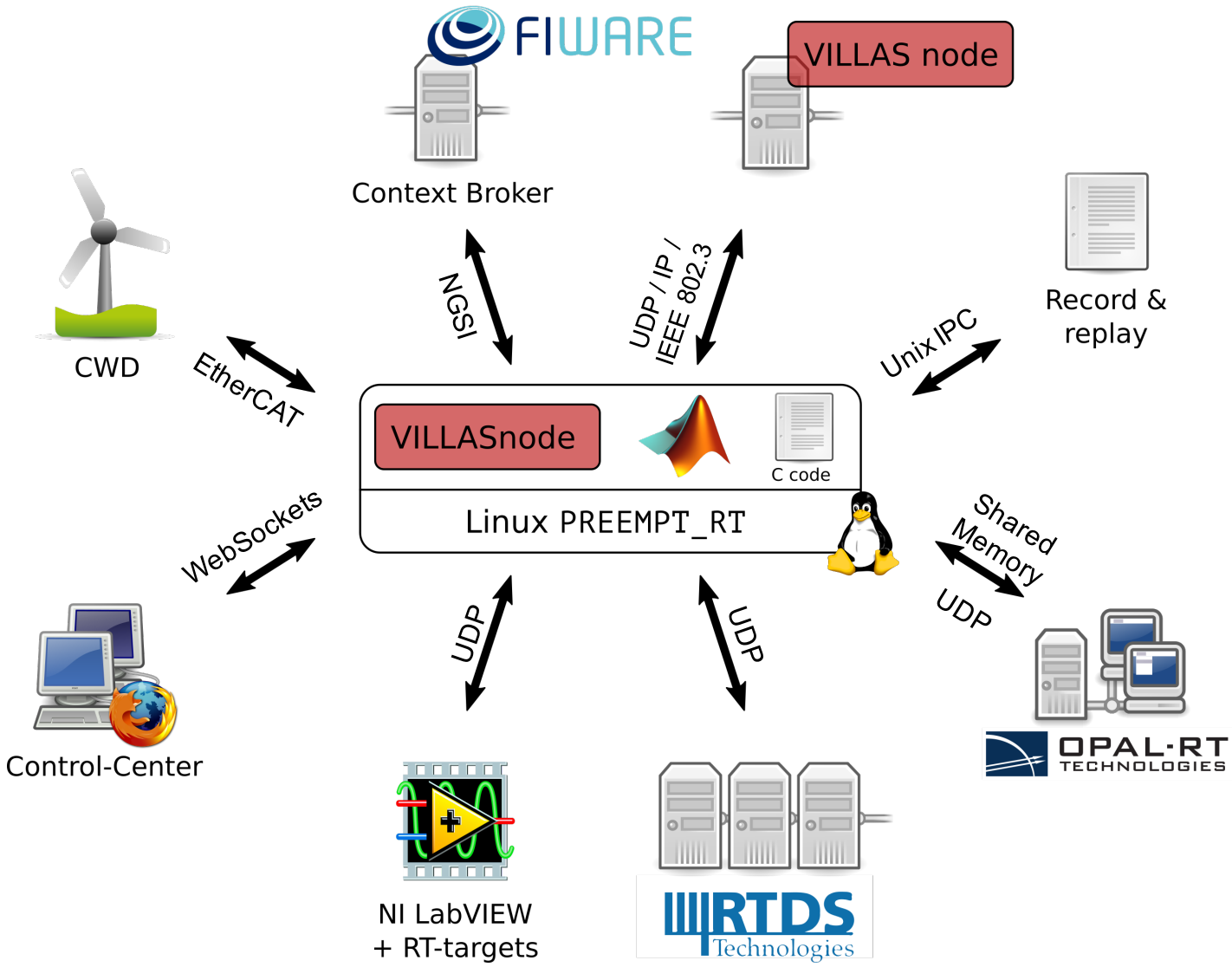
Objectives

- Framework for power system co-simulation on FPGAs
  - Extensible
  - Hard Real Time
- Leverage flexibility of Linux and performance of specialized DRTS
- Example: Interface to RTDS
  - Synchronized data exchange with sub 10 uS time-steps
- Get familiar with latest FPGA design tools & methodologies
  - High Level Synthesis (C / C++)
  - System Generator for DSP (Simulink + Matlab)
  - Electronic System Level Design (ESL)
- Optimize Linux for hard real-time simulation

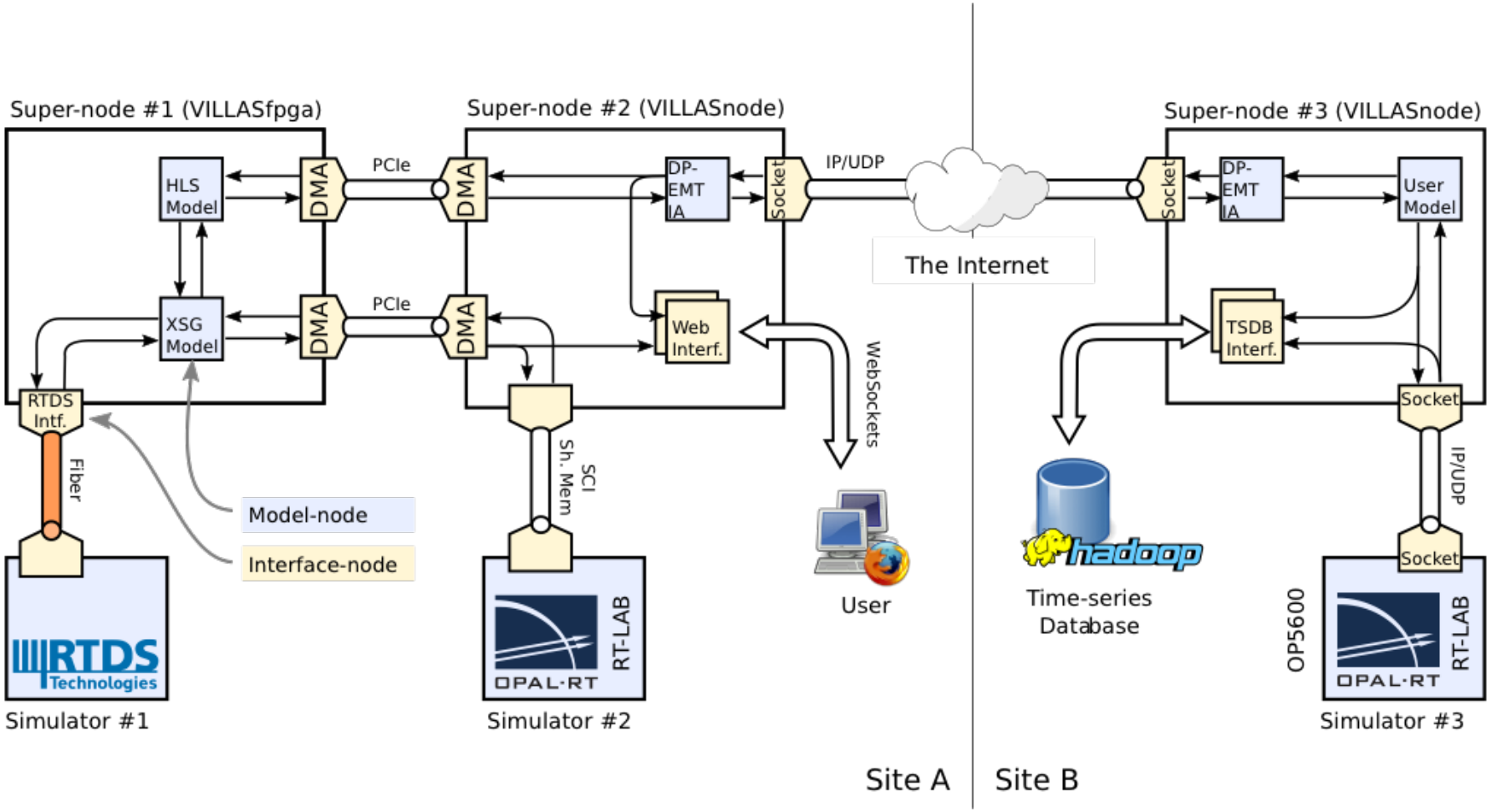
Applications

- Real-time co-simulation
  - One subsystem running on Linux
- Hybrid simulation: EMT + dynamic phasors
  - DFT implementation on FPGA
  - One subsystem is running in phasor domain on Linux
- Interface algorithms for geographically distributed simulation
  - Prediction and compensation of communication delay
  - DFT / wavelet transforms
- Custom communication protocols for RTDS: EtherCat...
- User interaction & monitoring
  - WebSockets, FIWARE
- Data logging & replay
  - Big Data (we could capture **every** timestep)

Architecture: VILLASnode

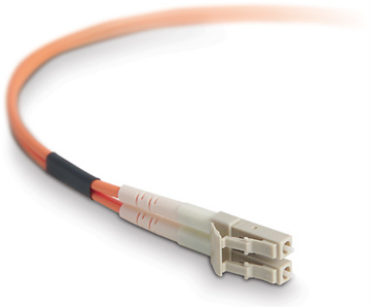


Architecture: VILLASnode + VILLASfpga



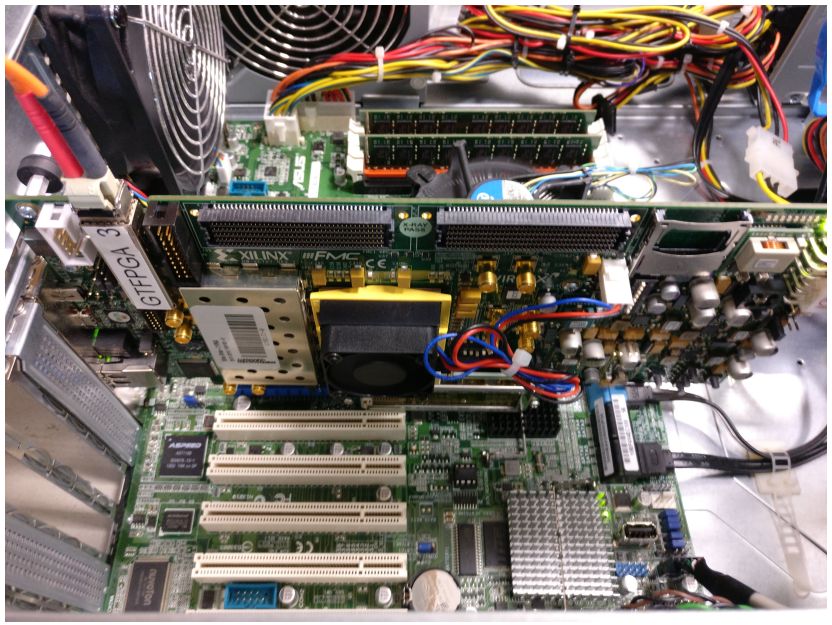
Full picture of possible VILLASnode topology.

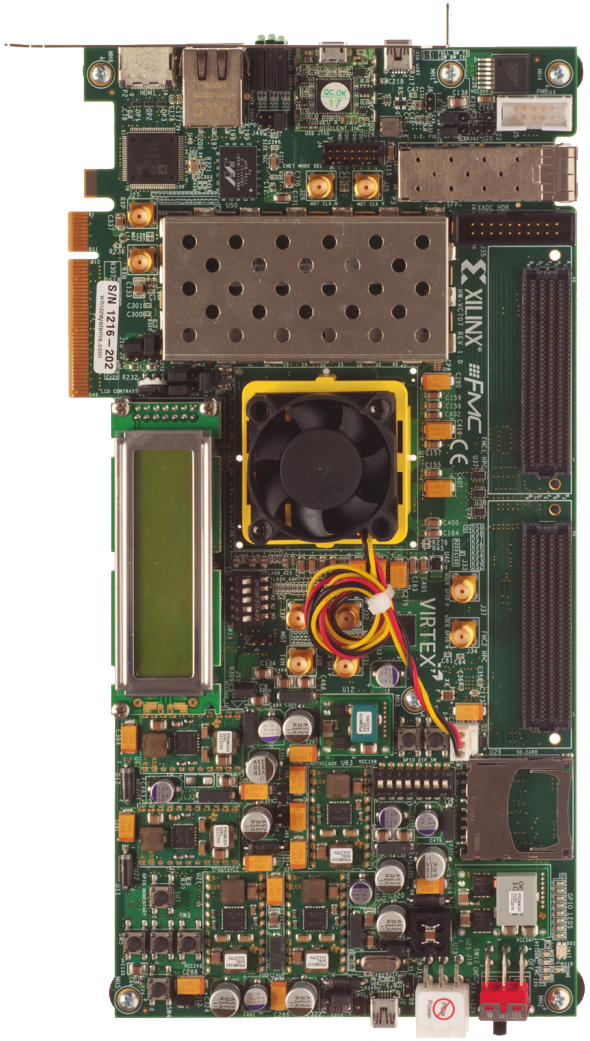
Hardware: Xilinx VC707 Evaluation Board



LC Multi-mode Fiber

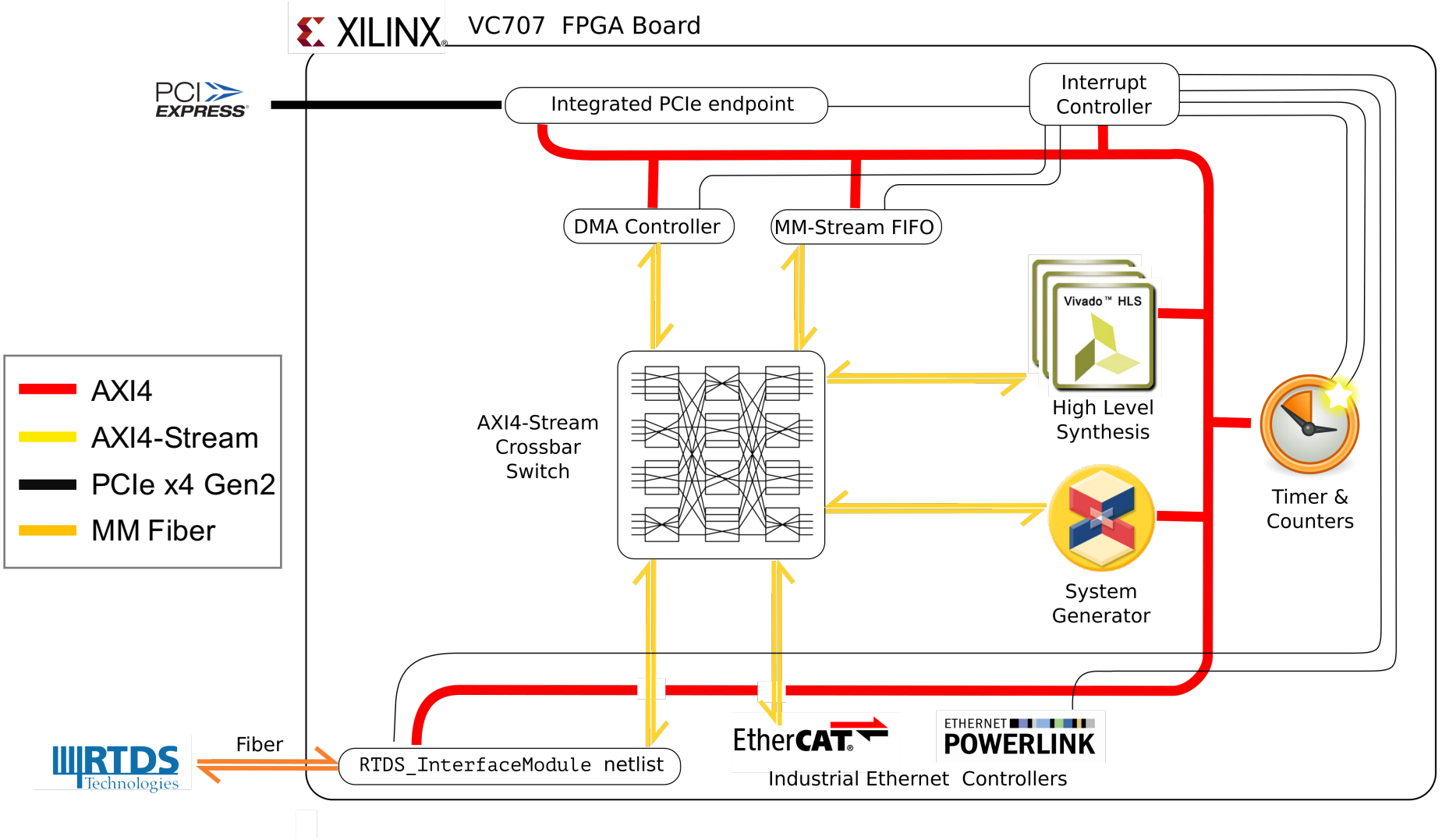
SFP Module



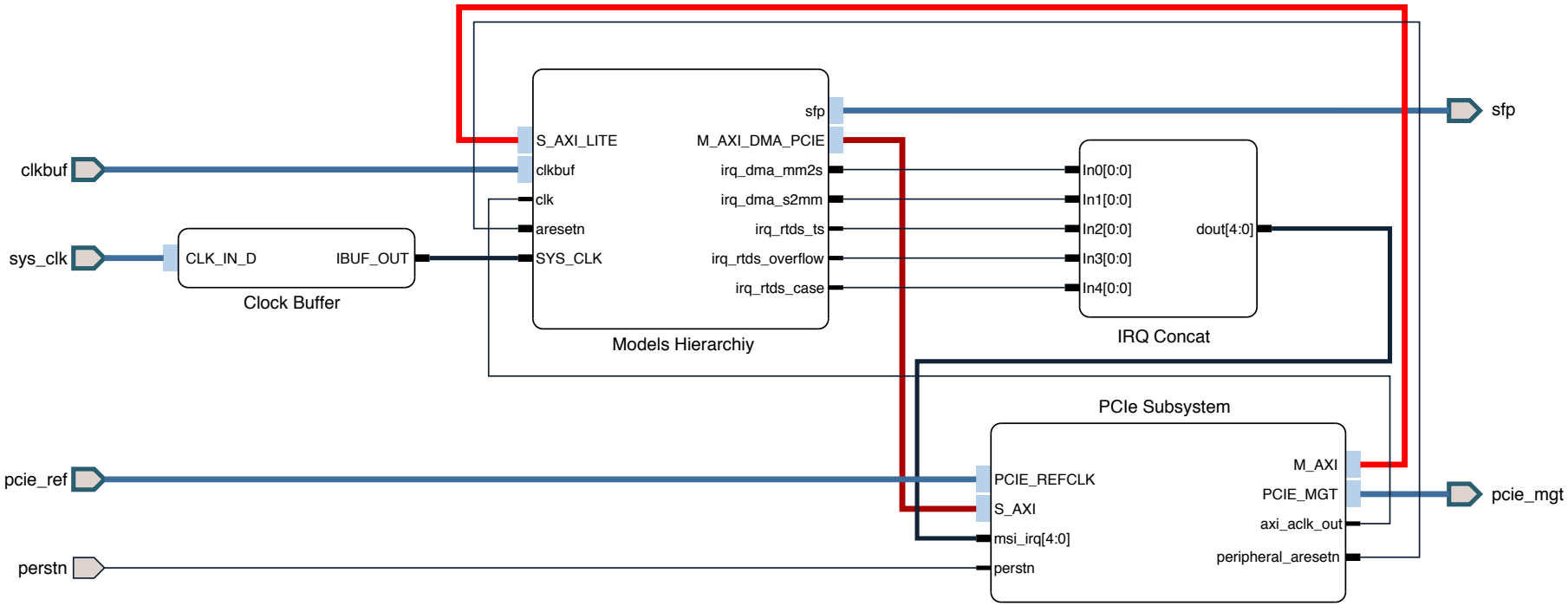


VC707 Evaluation Board © Xilinx Inc.

Architecture: VILLASfpga

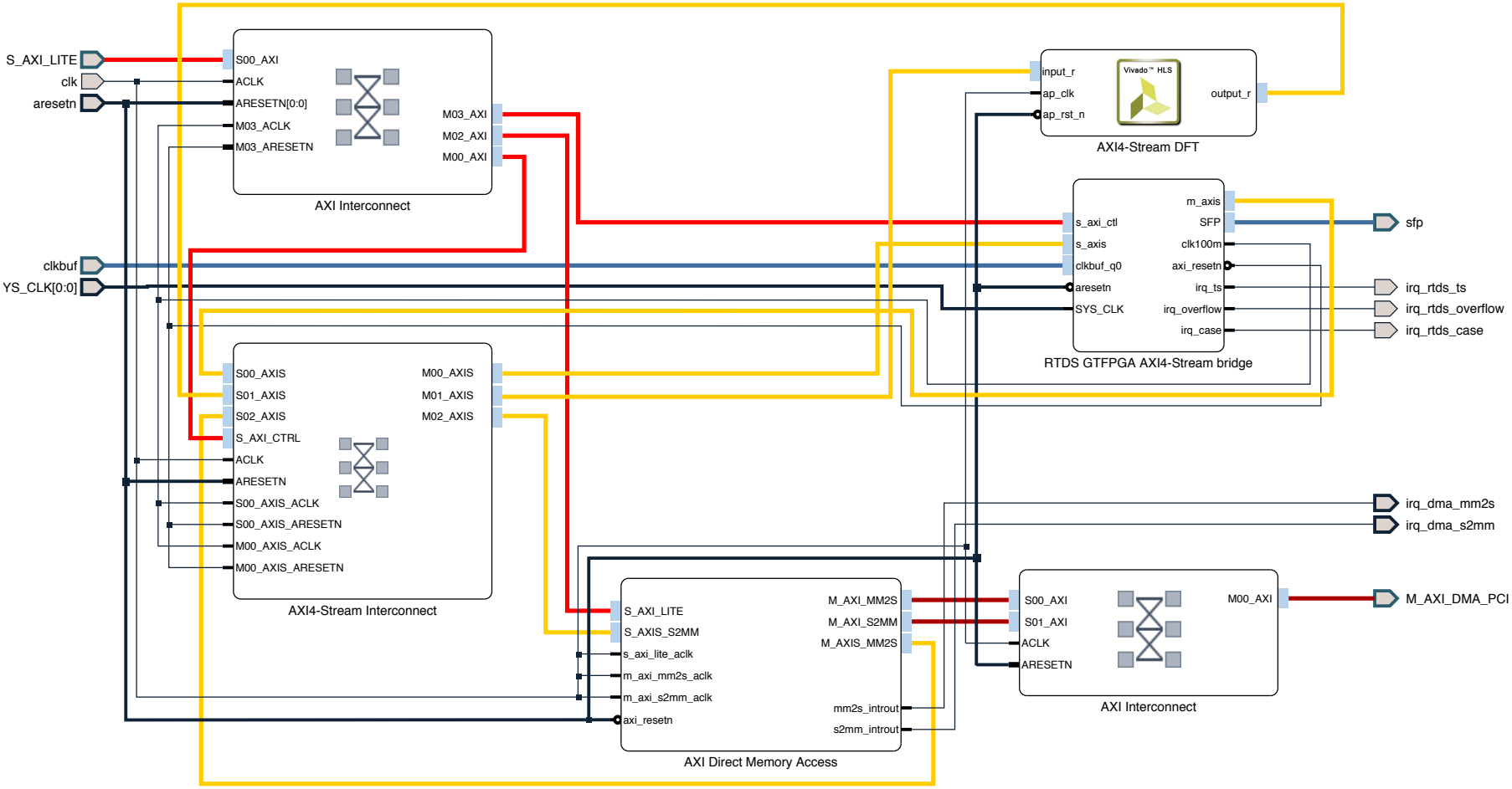


Architecture: IP Integrator



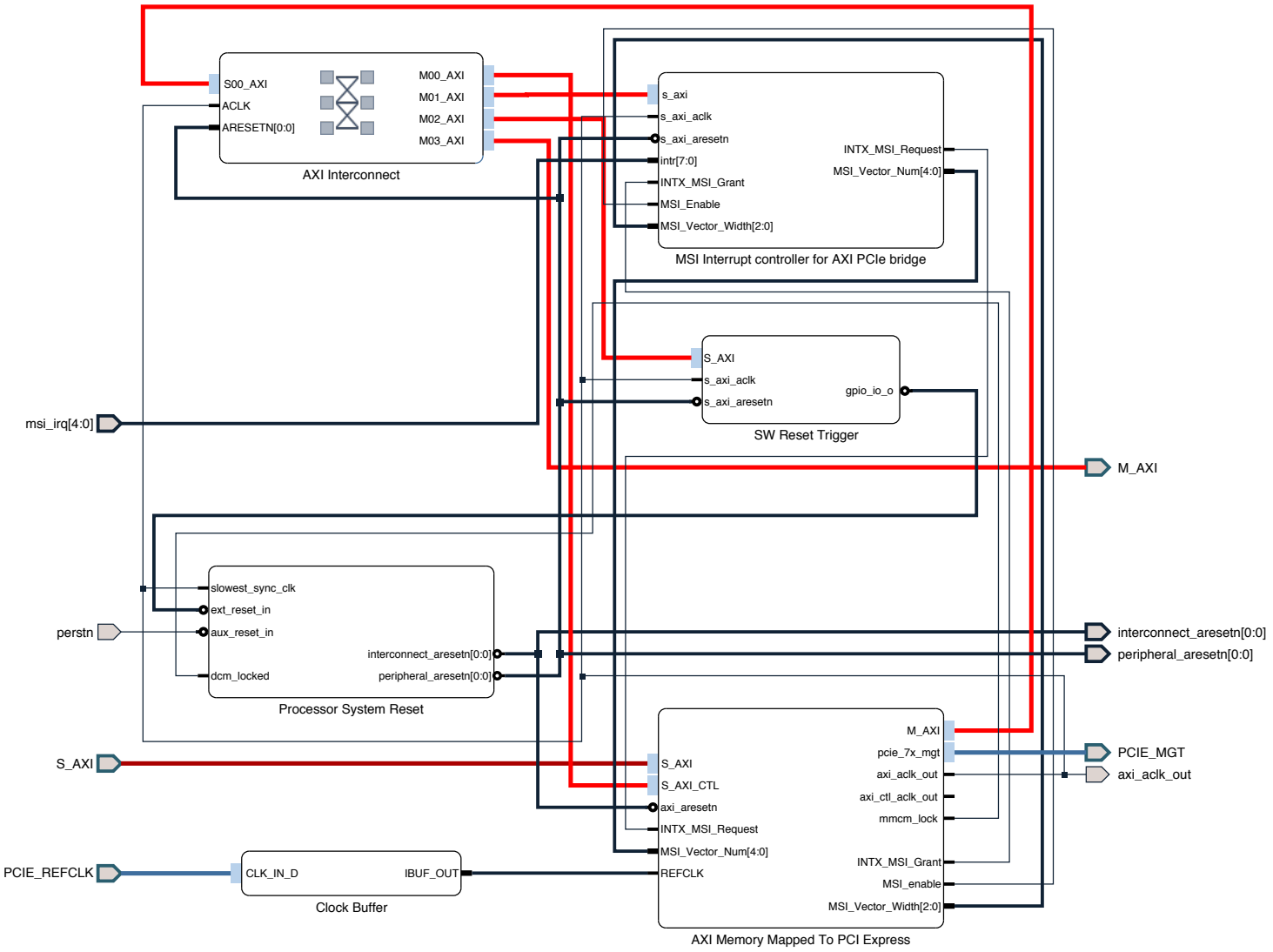
Vivado IP Integrator Block Diagram

Architecture: IP Integrator (Model)



Vivado IP Integrator Block Diagram of Model Hierarchy

Architecture: IP Integrator (PCIe)



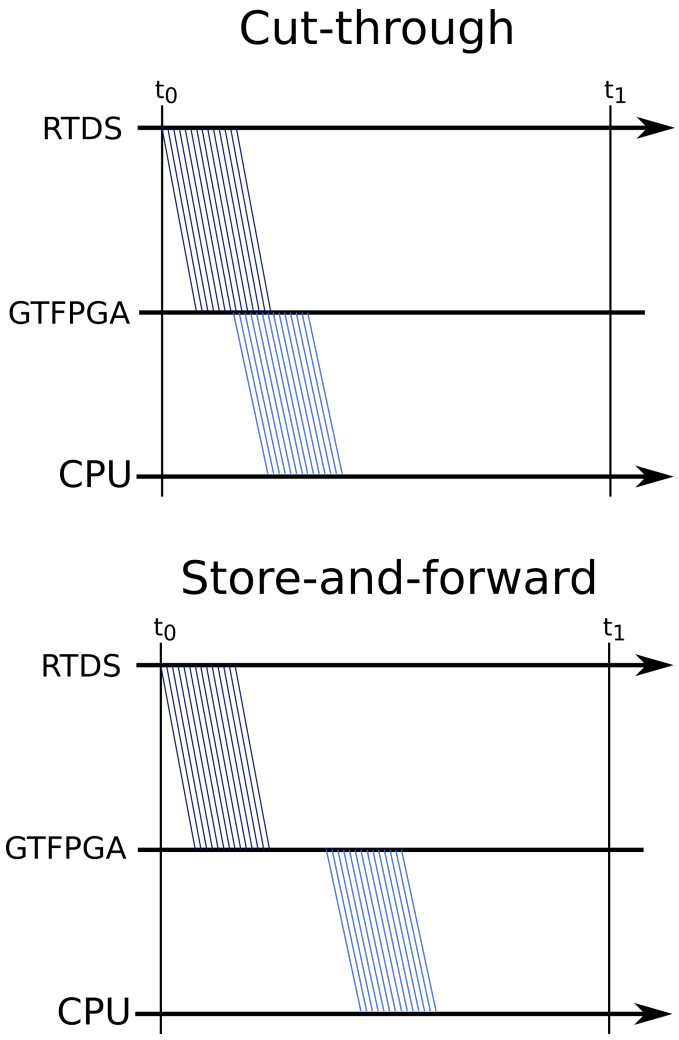
Interconnect

- Virtex 5 (ML507)
  - Memory Mapped: Processor Local Bus (PLB)
  - Streaming: Local Link (LL)
  - Influenced by IBM PowerPC architecture

- Virtex 6/7 (ML605 / VC707)
  - Memory Mapped: AXI4 & AXI4-Lite
  - Streaming: AXI4-Stream
  - Advanced Microcontroller Bus Architecture (AMBA) introduces by ARM Inc.

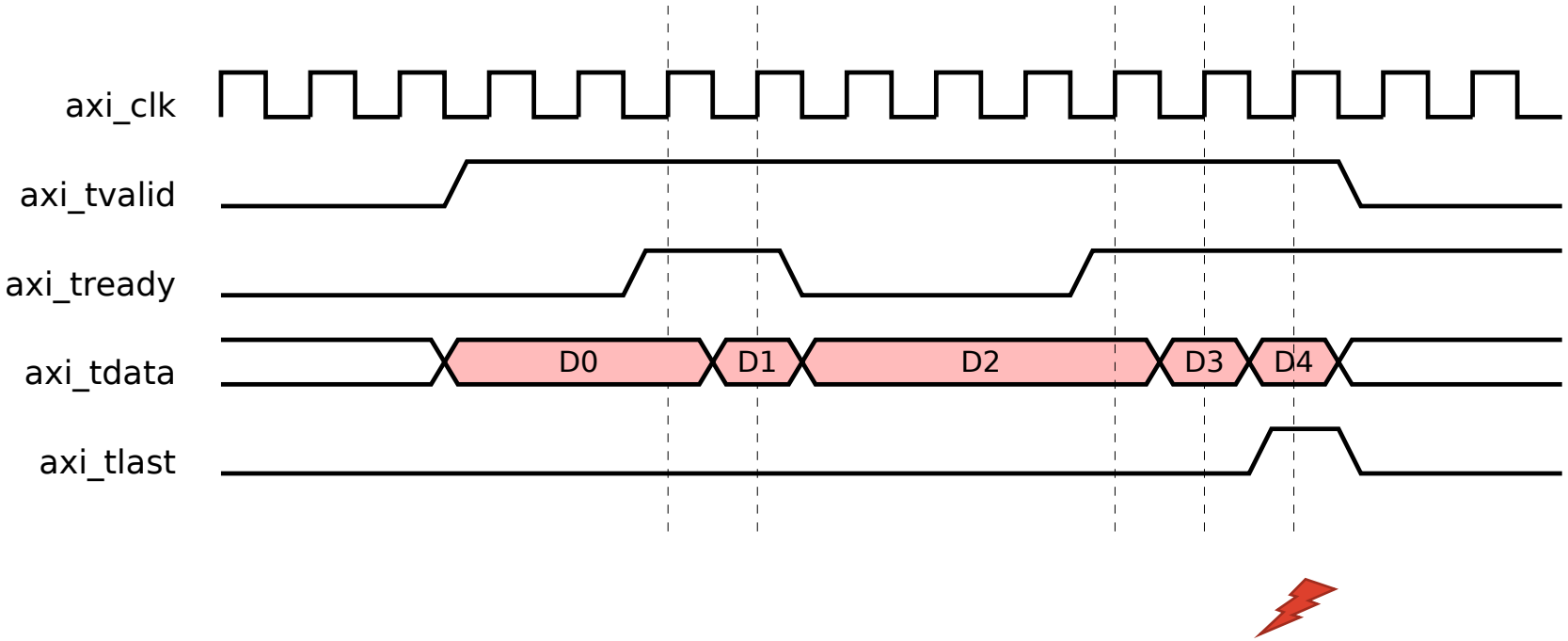
AXI4-Stream

- AXI4-Stream enables **pipelining**
  - FIFOs instead of Addressable Mem
- Cut-through instead of Store-and-forward
  - Reduced latency
- Examples
  - Filters
  - Recursive / Short-term DFT



AXI4-Stream: Protocol

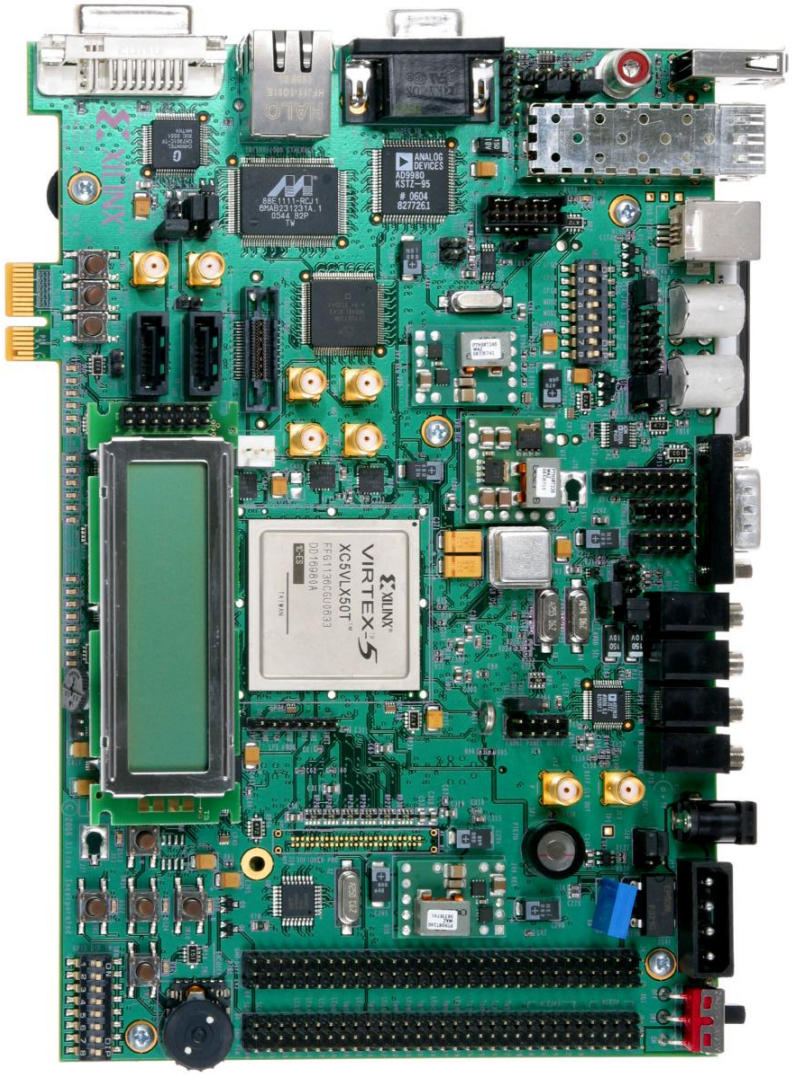
- Implicit synchronization



Xilinx ML507 Evaluation Board

- Integrated PPC440 CPU
- 5th generation Virtex FPGA
- Supported by ISE IDE only

- Two boards are used at ACS



© Xilinx Inc.

Xilinx ML605 Evaluation Board

- 6th generation Virtex FPGA
- Supported by ISE IDE only

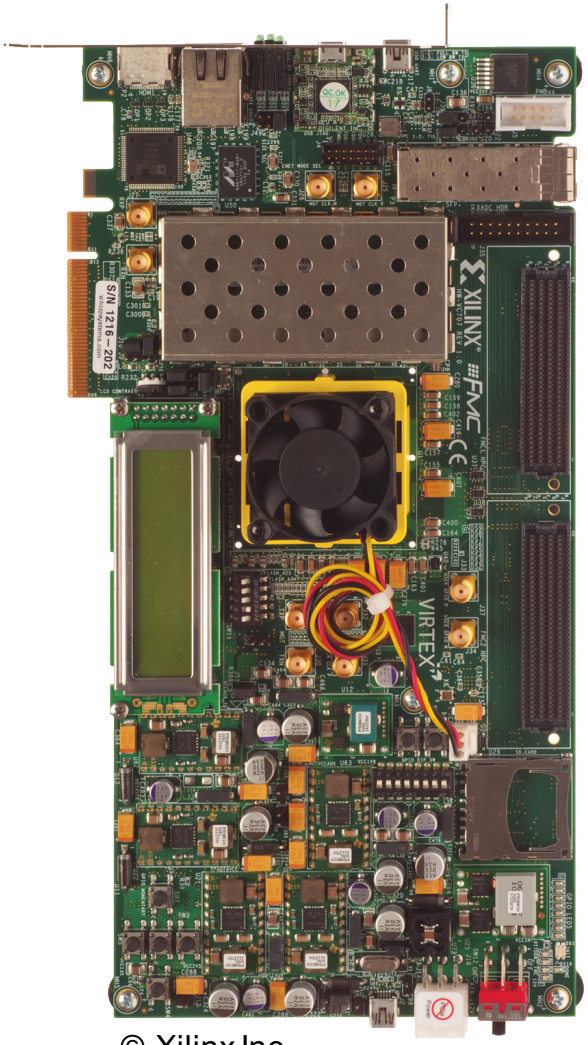
- We have one in OPAL-RT OP5600

© Xilinx Inc.

Xilinx VC707 Evaluation Board

- 7th generation Virtex FPGA
- Supported by Vivado IDE

- We bought one



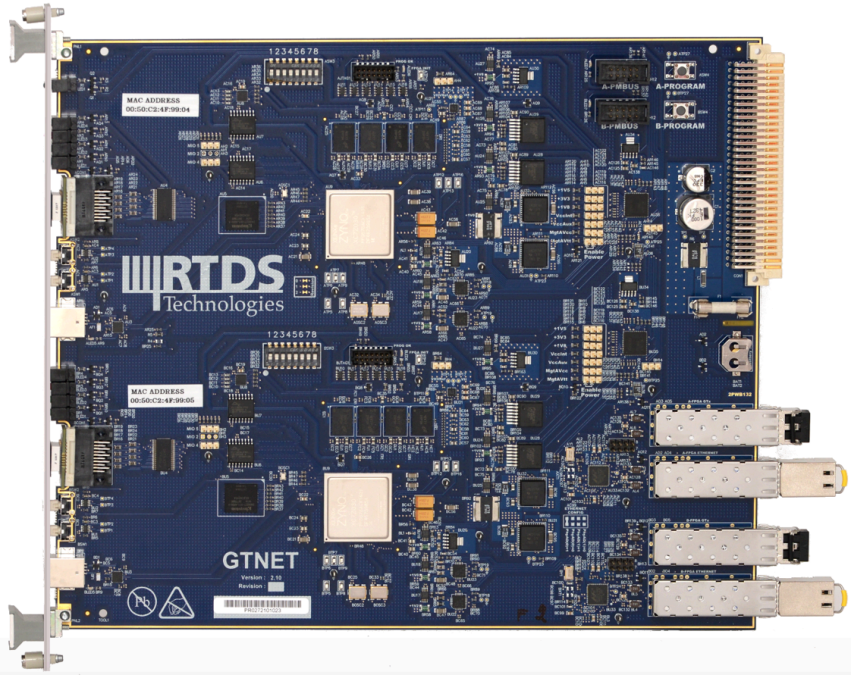
© Xilinx Inc.

Existing RTDS Interfaces: GTNET cards

- Supports a variety of protocols: IEC61850 Goose & SV, IEC60870-5-104, DNP, UDP, TCP
- No time-step synchronization!

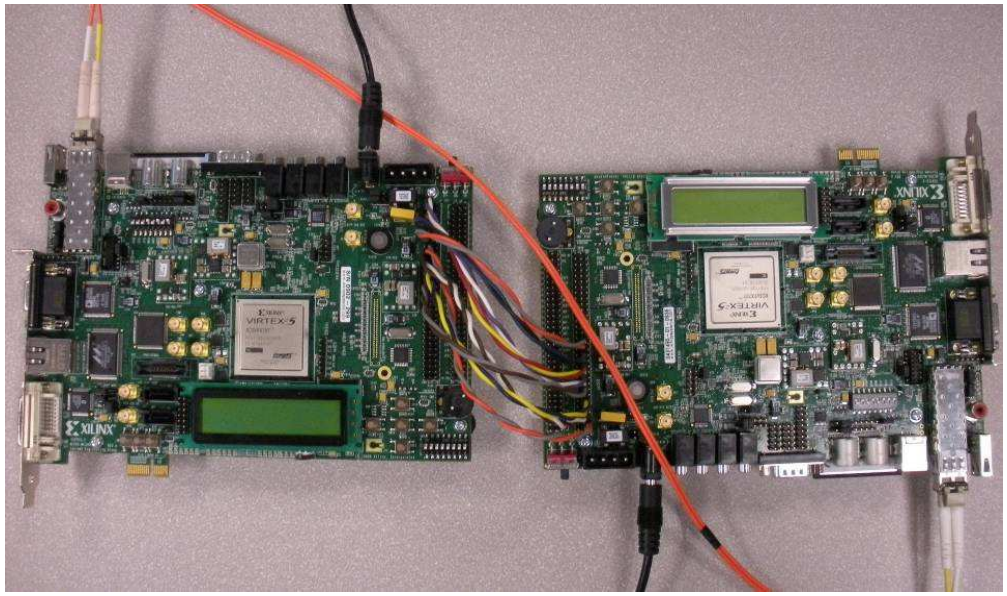
- GTNET-SV: IEC 61850-9-2 Sampled Values
  - Up to 256 samples / cycle (12.8 kHz for f=50 Hz)

- GTNETv2-SKT
  - Up to 300 values at 1 kHz
  - Less then 60 values at 5 kHz
  - GTNETx1 limited to 100 Hz



Existing RTDS Interfaces: GTFPGA / MMC support unit

- First digital RTDS interface was demonstrated by CAPS / ABB in 2008
  - 16-bit LVDS digital interface between to GTFPGA cards
- Closed source IP (compiled NGC netlist)
- Up to 64 values per direction and time step (FP or Integer)



© CAPS, Michael Sloderbeck

Linux, Drivers & Optimization

Steffen Vogel

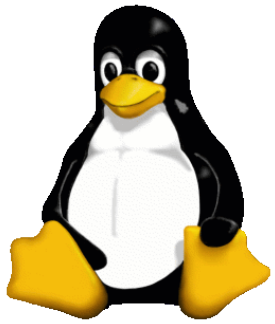
ACS | Automation of Complex Power Systems

E.ON Energy Research Center

RWTH AACHEN UNIVERSITY

Linux Real-time Optimizations

- Linux
  - Tickless kernel!
  - Pinned threads and IRQs (`isolcpus`)
  - Preemptible Kernel: `PREEMPT_RT` patchset
- System
  - Avoid System Management Interrupts
  - Reduce PCIe bus contention and auxiliary IRQs
- Lock-less datastructures: queues, lists, stacks



SW Driver

- Linux provides APIs to implement drivers in userspace
  - Userspace IO (UIO)
  - Virtual Function IO (VFIO)
  - /sys/bus/pci/[bdf]/resources
  - /dev/mem + /proc/[pid]/pagemaps

- Ideally we would implement the driver as a custom kernel module
  - Disadvantages:
    - Context-switches become critical for small timestep simulations
    - Maintenance of out-of-tree module
    - Compilation of kernel / module

VFIO has been choosen

Comparison

- Userspace IO (UIO)
  - No bus-mastering
    - = No DMA
    - = No MSI interrupts
- /dev/mem + sysfs
  - Bus-mastering possible with dirty hacks
  - No device isolation by IOMMU
  - DMA continuous allocation only with hugepages
  - No MSI support
  - Requires superuser privileges
- Virtual Function IO (VFIO)
  - „Successor“ of UIO
  - Mostly used by KVM and DPDK
  - Safe bus-mastering by isolating devices in IOMMU containers.
  - No superuser privileges required

Driver Comparison

	UIO Userspace IO	VFIO Virtual Function IO	Sysfs /sys/bus/pci + /proc/pagemap	LKM Loadable Kernel Module
Bus-mastering	☐	✔	ugly	✔
Interrupts	Legacy only	Legacy + MSI	☐	✔
DMA	☐	✔	ugly	✔
IOMMU Isolation	☐	✔	☐	(☐)
Req. permissions	Superuser	flexible	Superuser	flexible
Users	DPDK, Closed Source Drivers	KVM, DPDK	Nobody	Upstream Drivers
Userspace	✔	✔	✔	(☐)
IPC	mmap()	mmap() ioctl()	read() write() mmap()	flexible

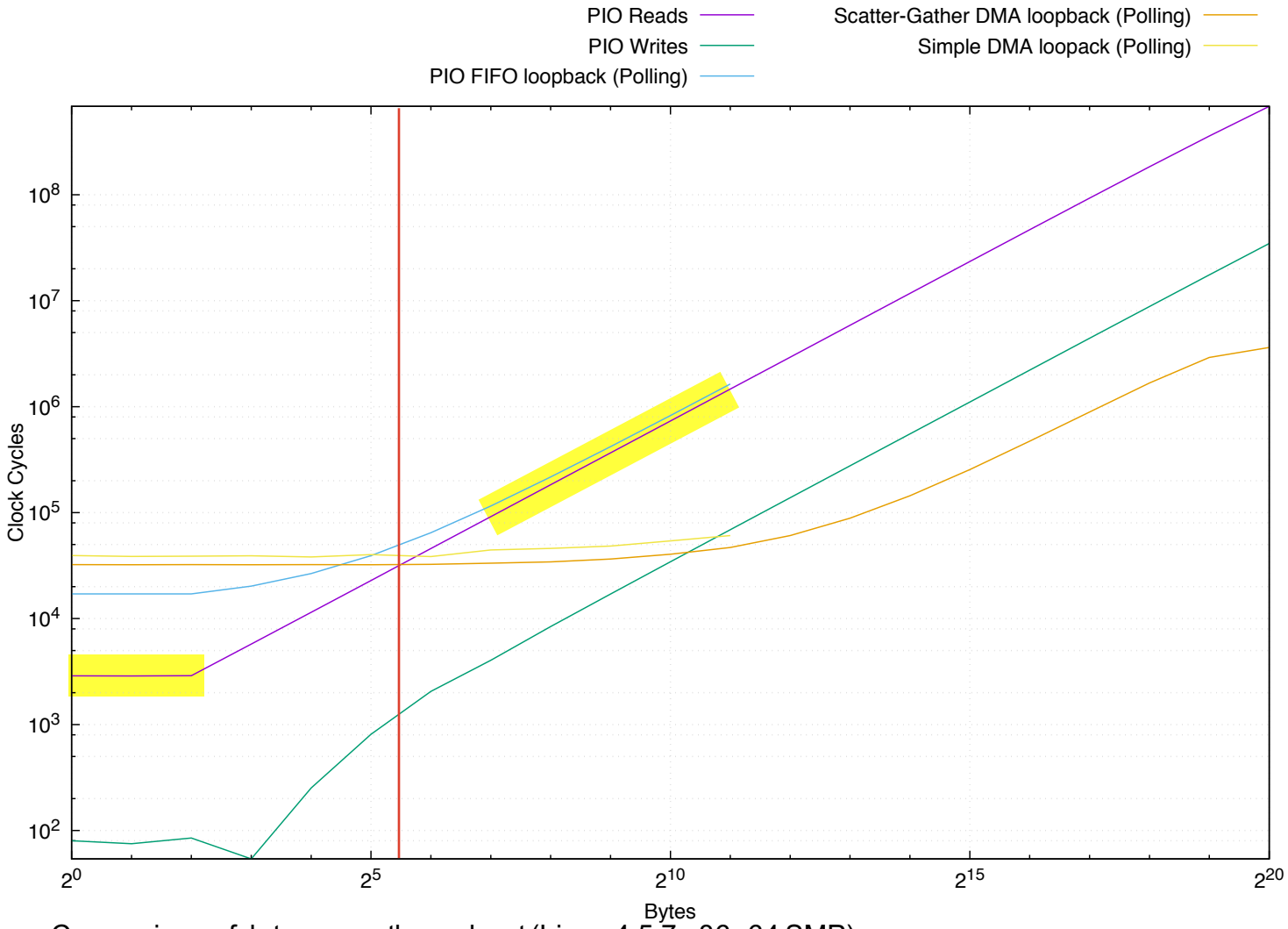
Datamovers: CPU <-> FPGA

- Transfer data between FPGA and CPU
- Convert from memory-mapped to packet streaming bus
- Usually two channels:
  - Device to Host: Stream to Memory-mapped (S2MM)
  - Host to Device: Memory-mapped to Stream (MM2S)

- Direct Memory Access (DMA)
  - Data transfer is delegated to DMA controller on FPGA
  - CPU is not occupied
  - PCI Bus Mastering: Device writes to host memory

- Programmed / Memory Mapped IO (MMIO / PIO)
  - Direct register access
  - FIFO

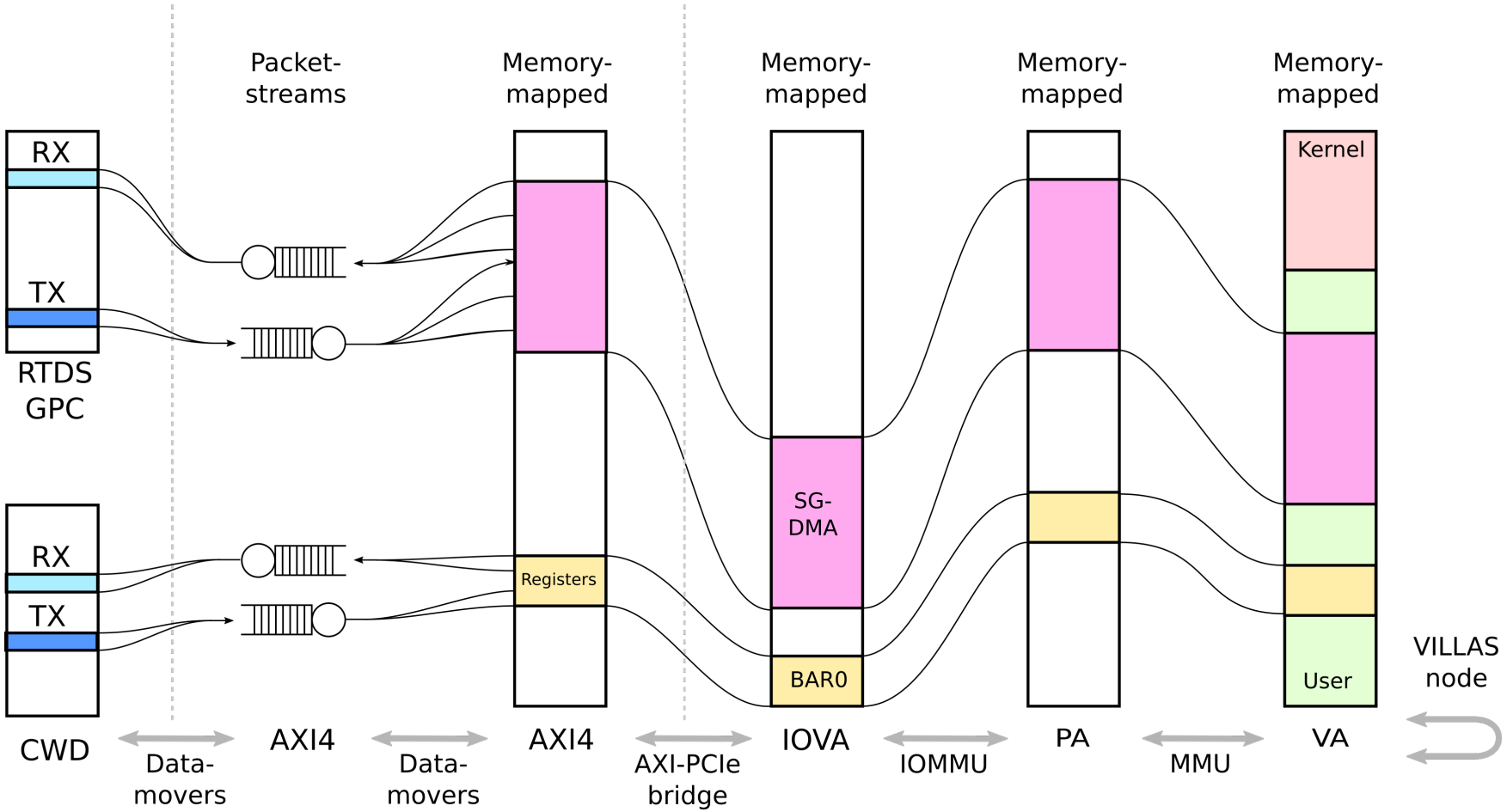
PCIe Interface: Datamovers



~ 1 us

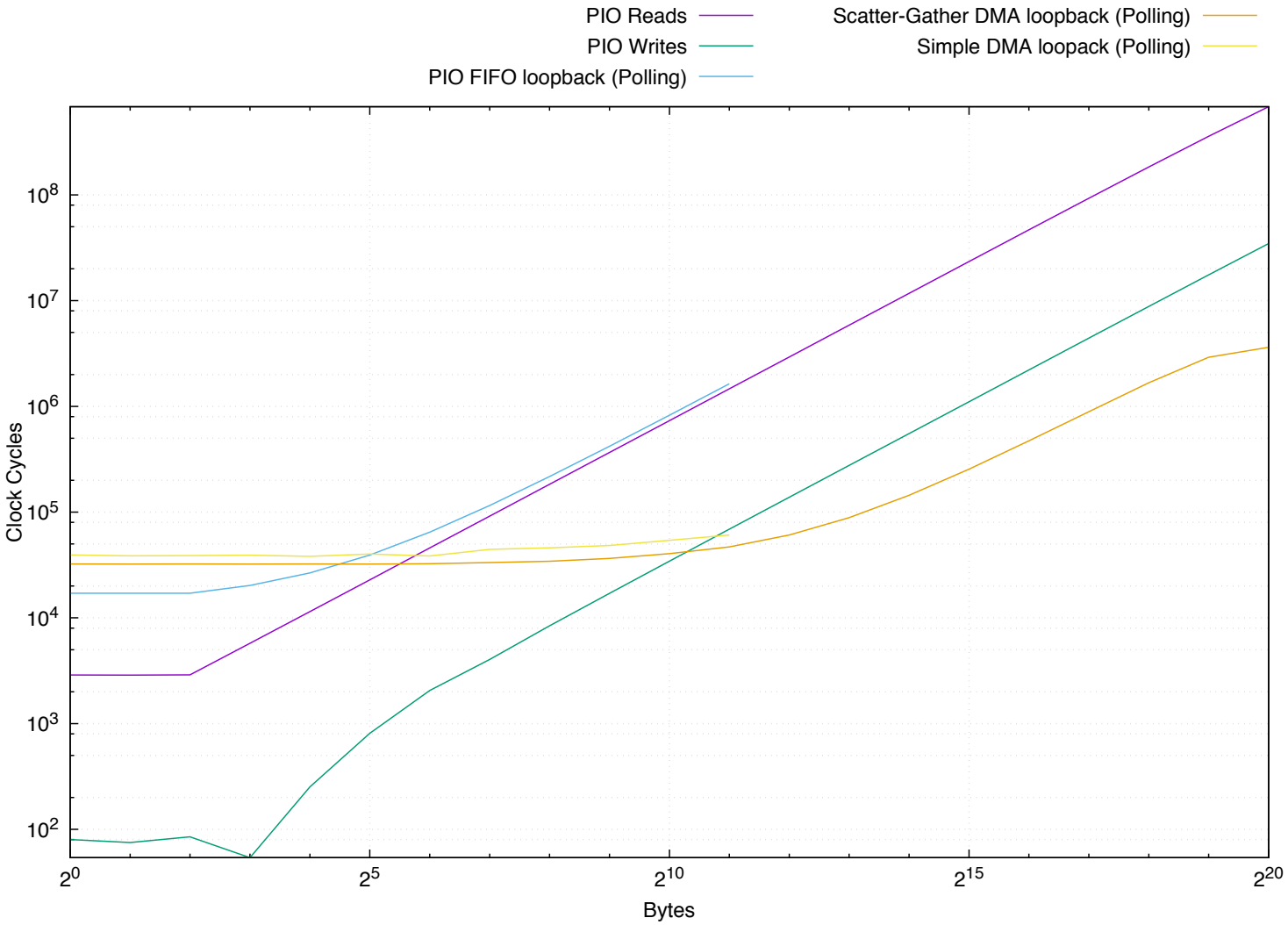
Comparison of data mover throughput (Linux 4.5.7 x86_64 SMP)

Address Spaces & Methods



Address spaces and Access methods

PCIe Interface: Datamovers



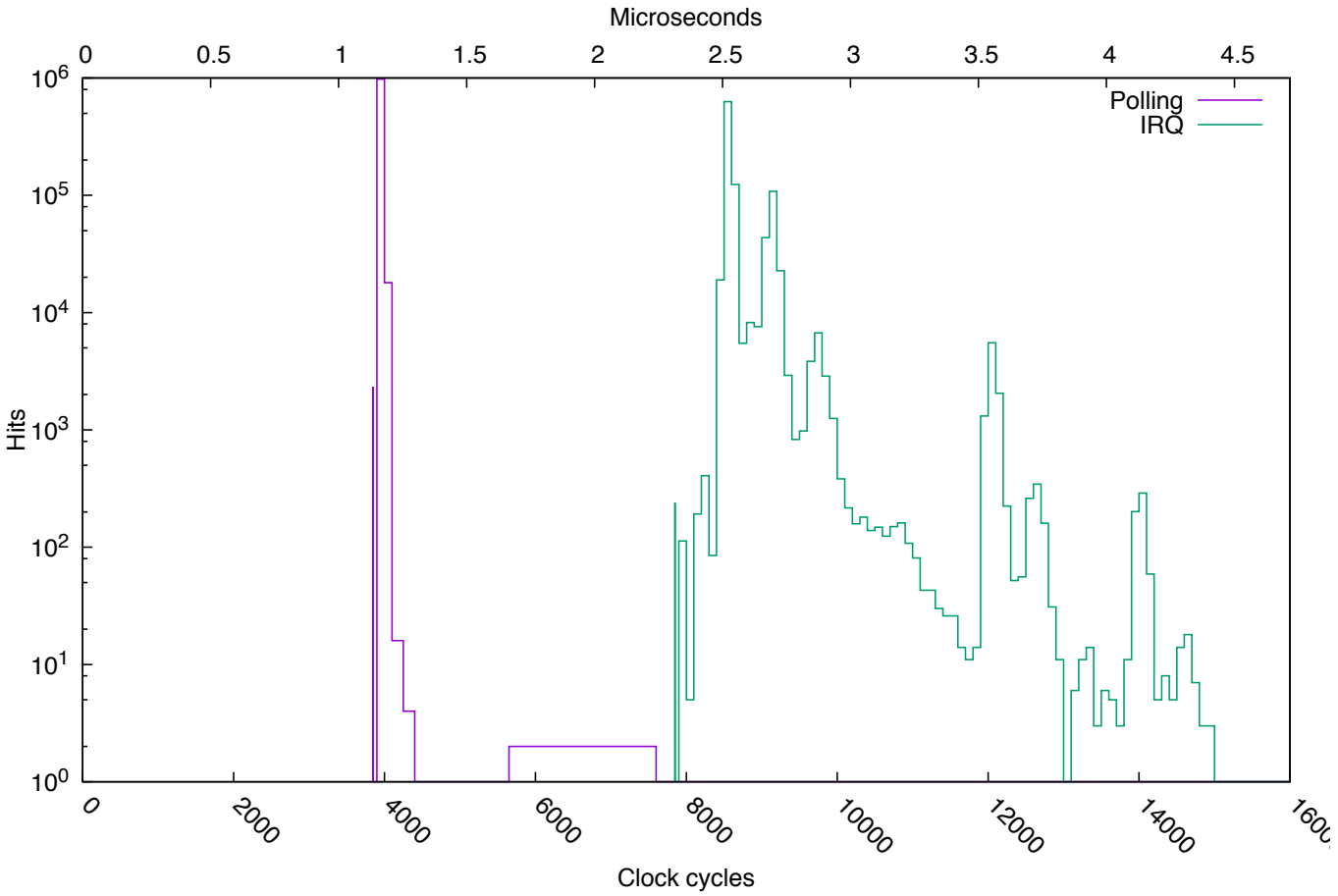
Comparison of data mover throughput (Linux 4.5.7 x86_64 SMP)

Synchronization Methods

- Interrupts
  - Legacy (INTX)
    - Out-of-band: Data-races possible
    - Only 4 physical interrupt lines
  - Message signalled (MSI)
    - In-band: ordering with data preserved
    - Lower latency
    - Supports up to 32 vectors
- Polling
  - Host RAM
    - Directly poll on DMA buffer descriptors
  - PCI mapped memory (BARs)
    - Poll Interrupts status registers
  - Network Interface Quibbles (NIQ)
    - long polling
    - delayed PCI read-completion TLPs (up to 50ms)

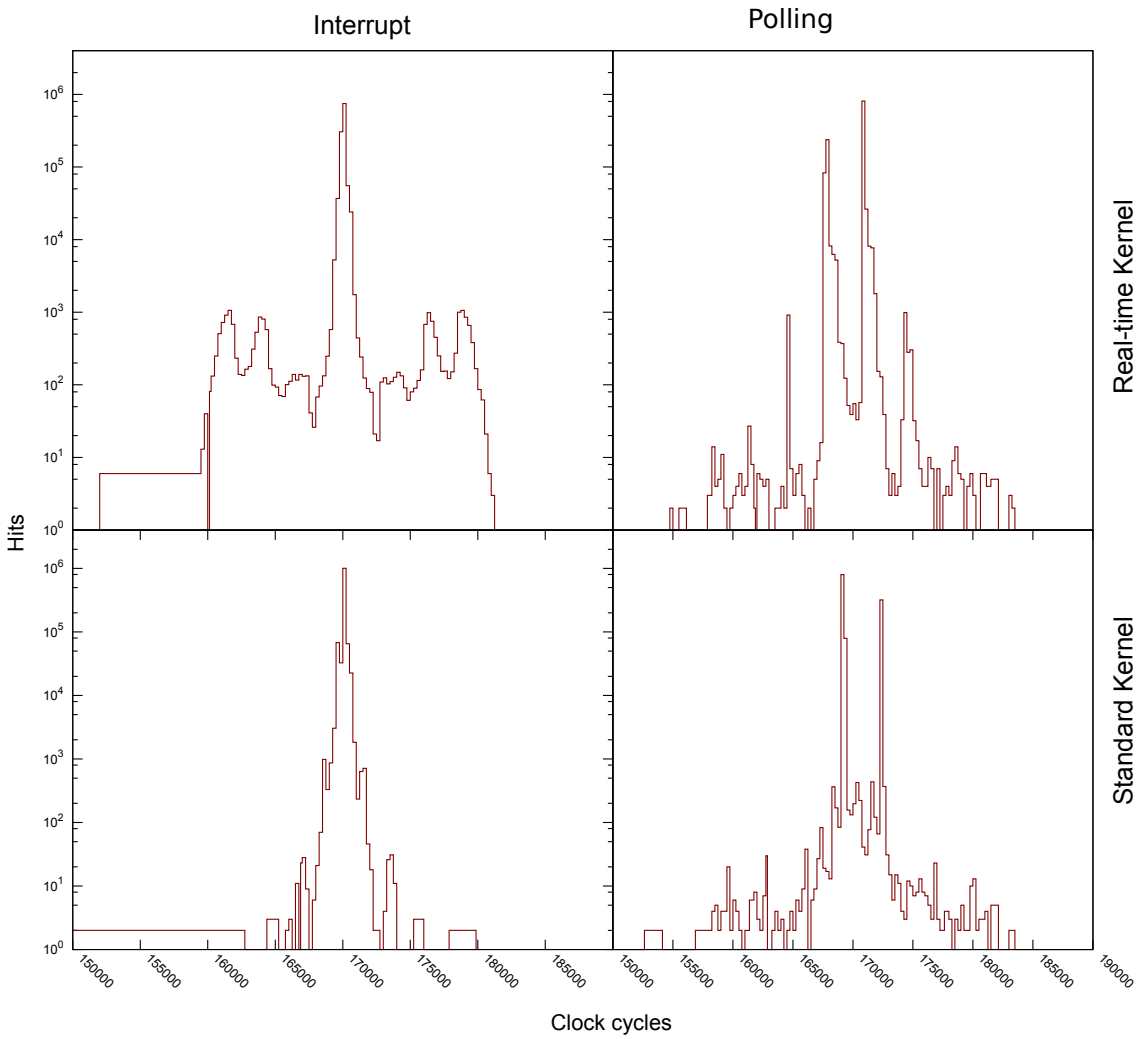
PCIe Interface: Latency

- Response time to interrupt generated on FPGA
  - Polling: ~ 1 uS
  - Interrupts: > 2 uS



PCIe Interface: Jitter

- Periodic timer IRQ every 50 uS ~ 170k clks
- Similiar results for polling on real-time and non real-time kernel
- Big differences for IRQs
  - Differences in PREEMPT_RT interrupt handling



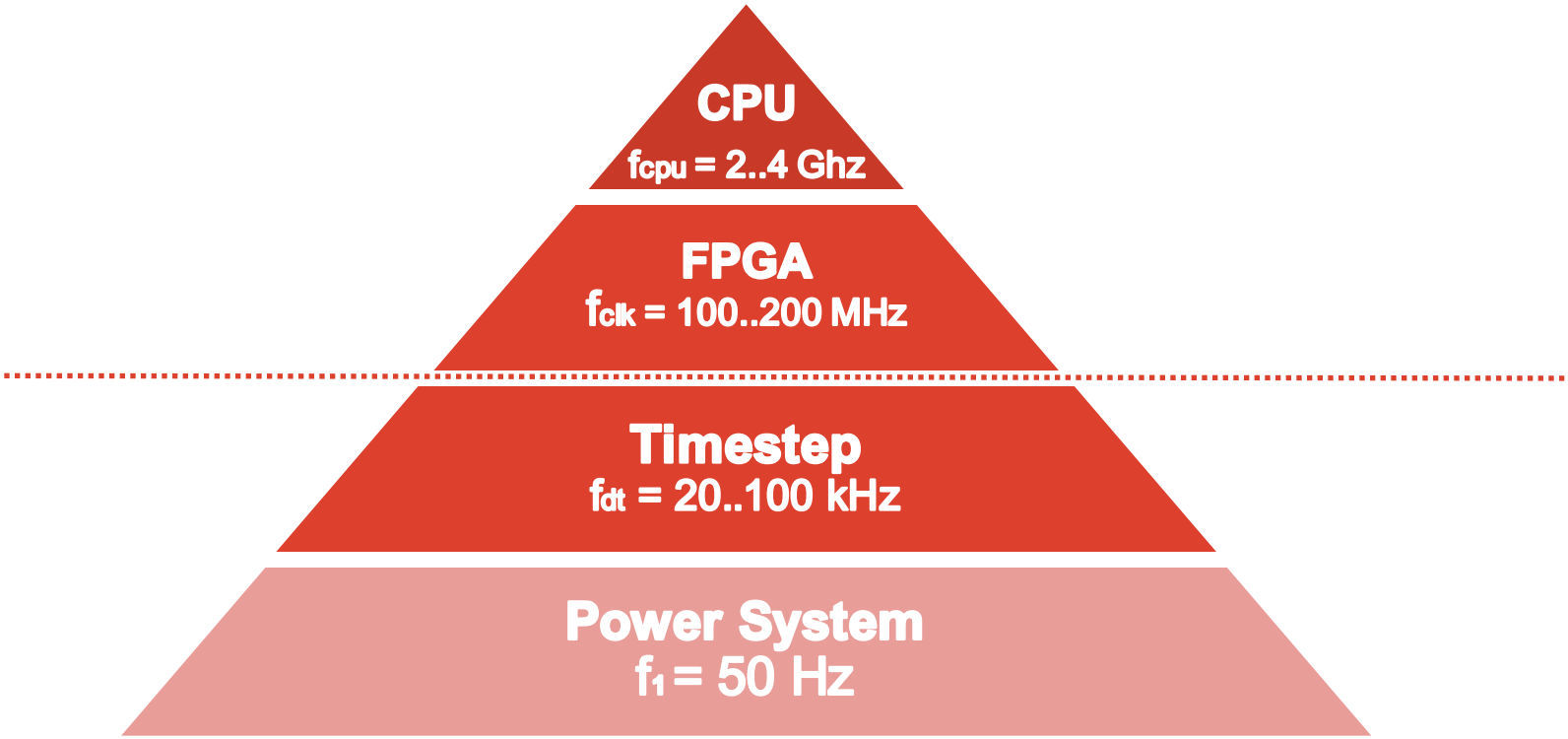
Comparison of interrupt jitter (Linux 4.5.7 vs 4.0.4-rt x86_64 SMP)

Synchronization

- Events require synchronization
  - Simulation started / stopped
  - New timestep started
  - Deadline missed / overrun
  - Data transfer completed

- „Hard“ synchronized
  - Reference to global time
  - Attached HIL devices
  - Multiple HIL devices
- „Soft“ synchronized
  - Only logical order is preserved

Synchronization: „Hierarchy“

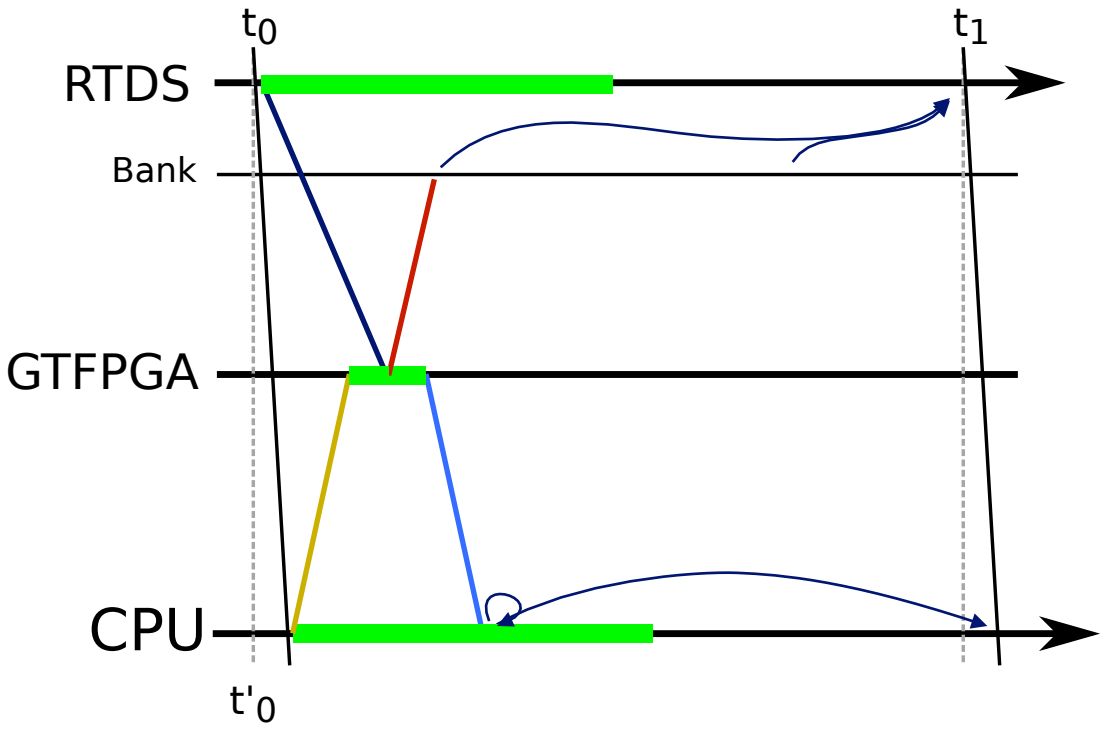


FPGA implementation

- 8 clocks
- 4 clock domains

→ Clock Domain Synchronization

Timing

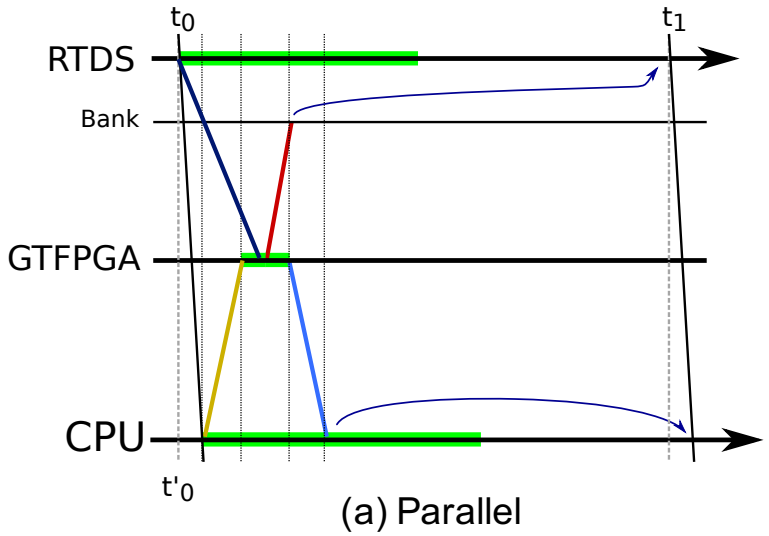


Computation

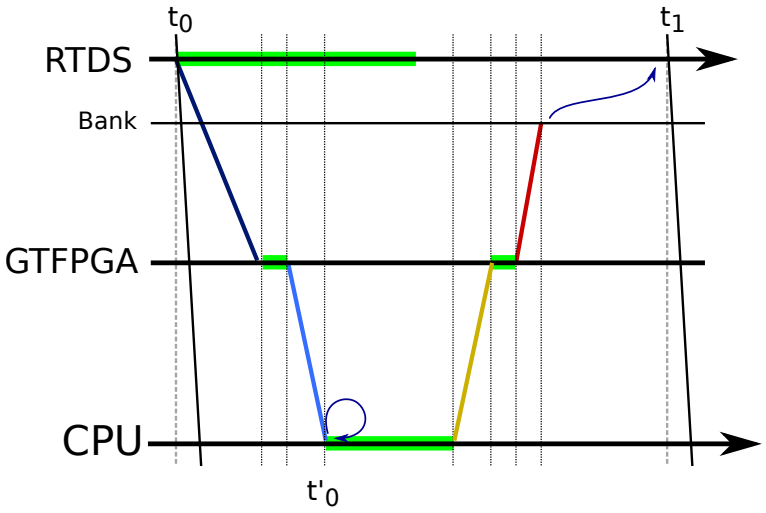
Update

Timing 2

- Different communication schemes are possible
- Limitations imposed by capabilities of RTDS_InterfaceModule
  - Sends data only at beginning of TS
  - Sends data only once per TS



(a) Parallel



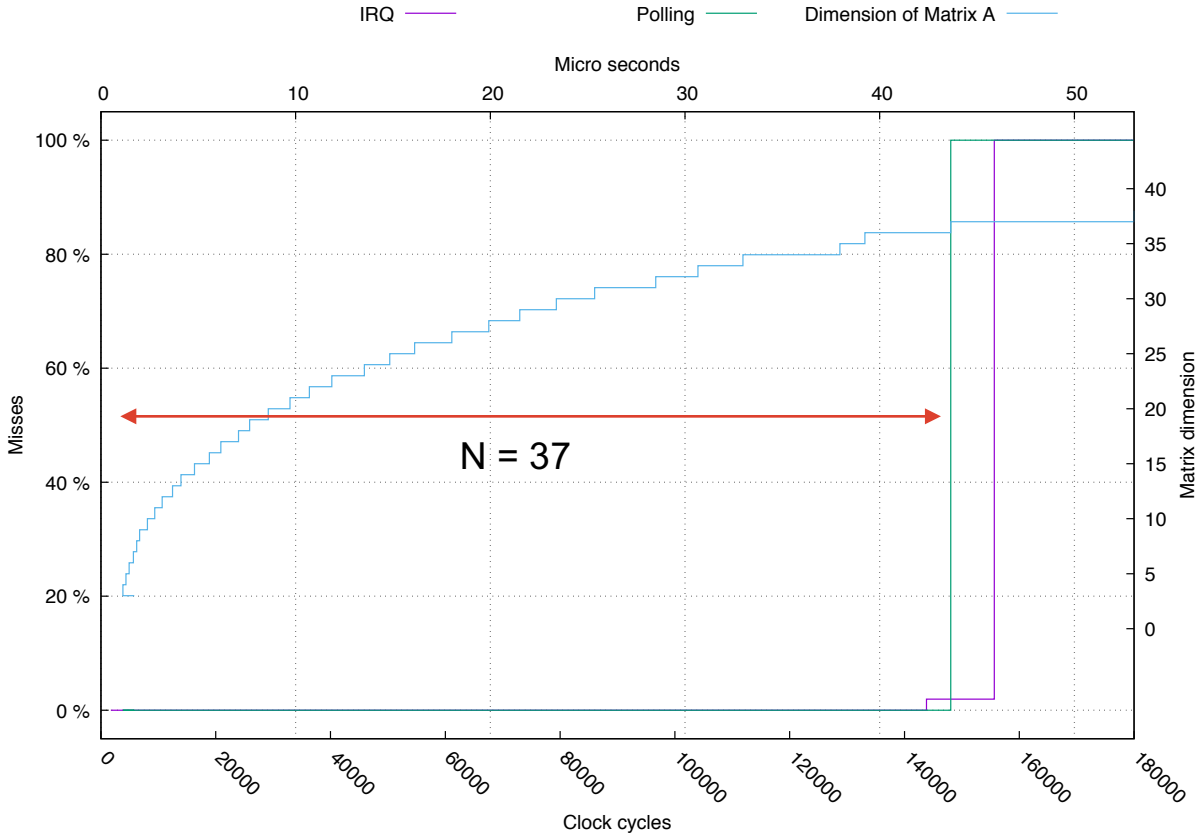
(a) Serial

Computation

Update

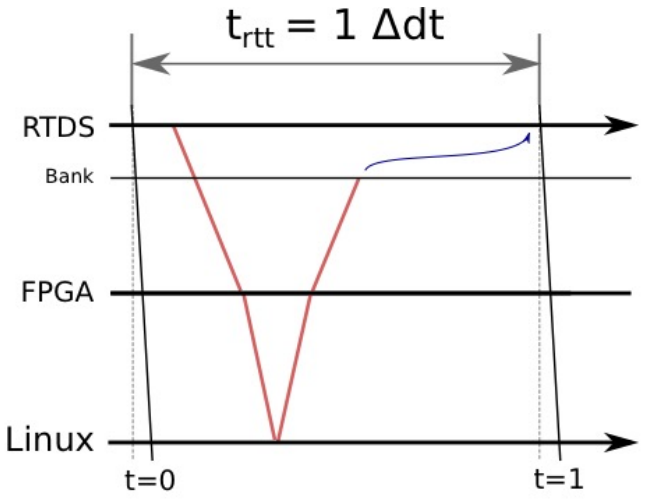
PCIe / RTDS Interface: Overruns

- Biggest matrix dimension which can be solved by LU decomposition?
  - Net-lib's LAPACK
- Detecting overruns by abnormal RTT
- Parallel communication pattern, time step: 50 us

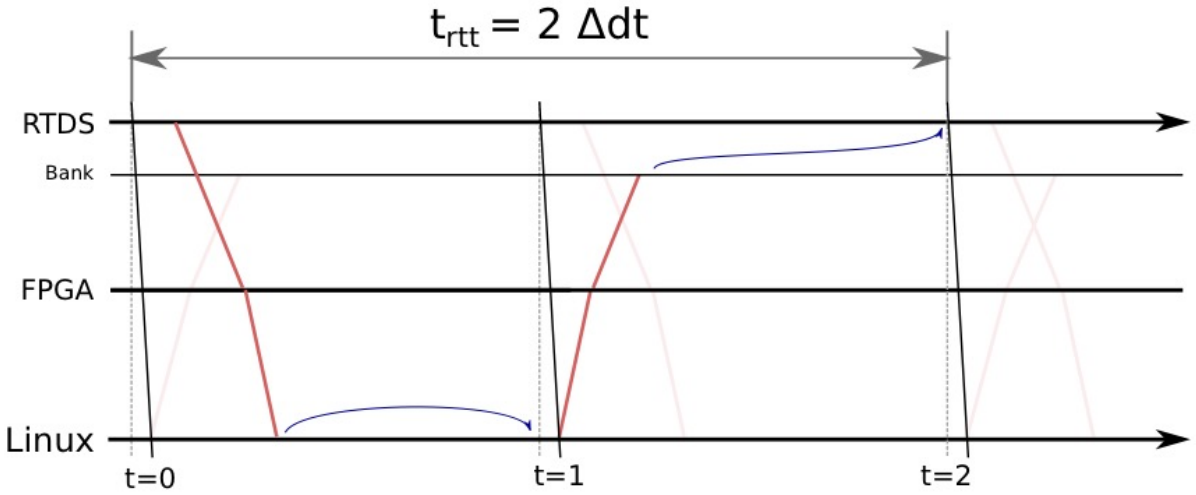


Round-trip Time: RTDS-Linux-RTDS

RTDS-Linux-RTDS Serial

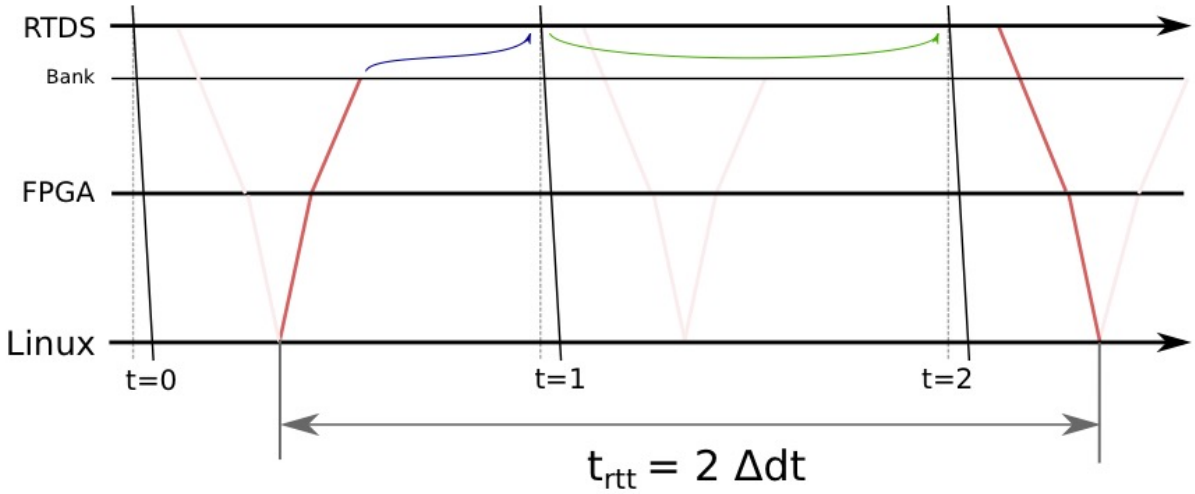


RTDS-Linux-RTDS Parallel

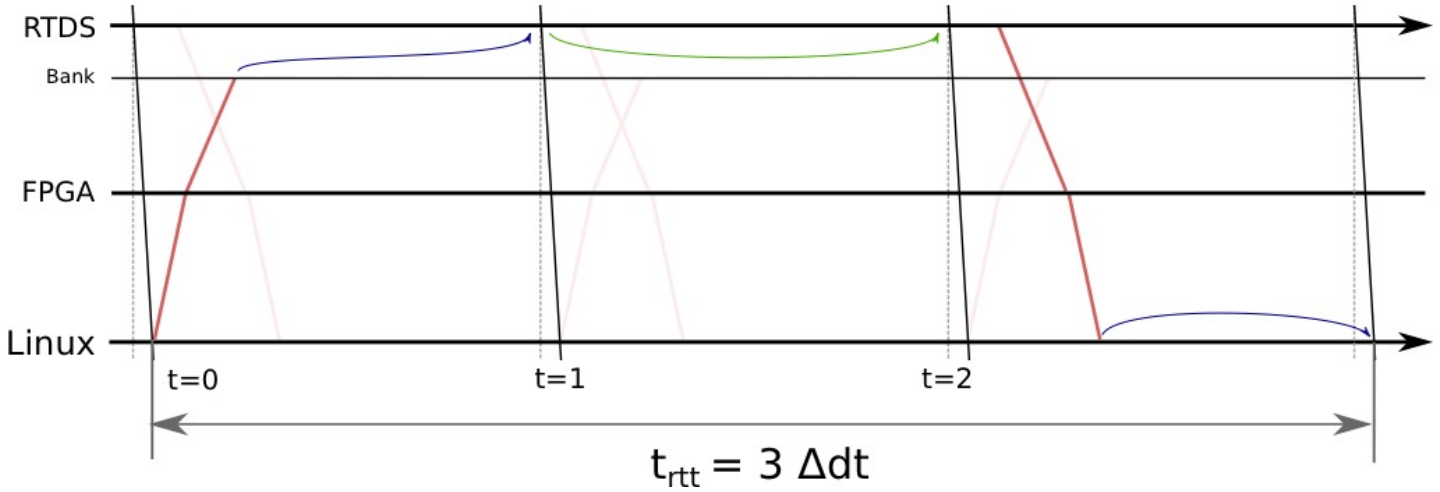


Round-trip Time: Linux-RTDS-Linux

Linux-RTDS-Linux Serial



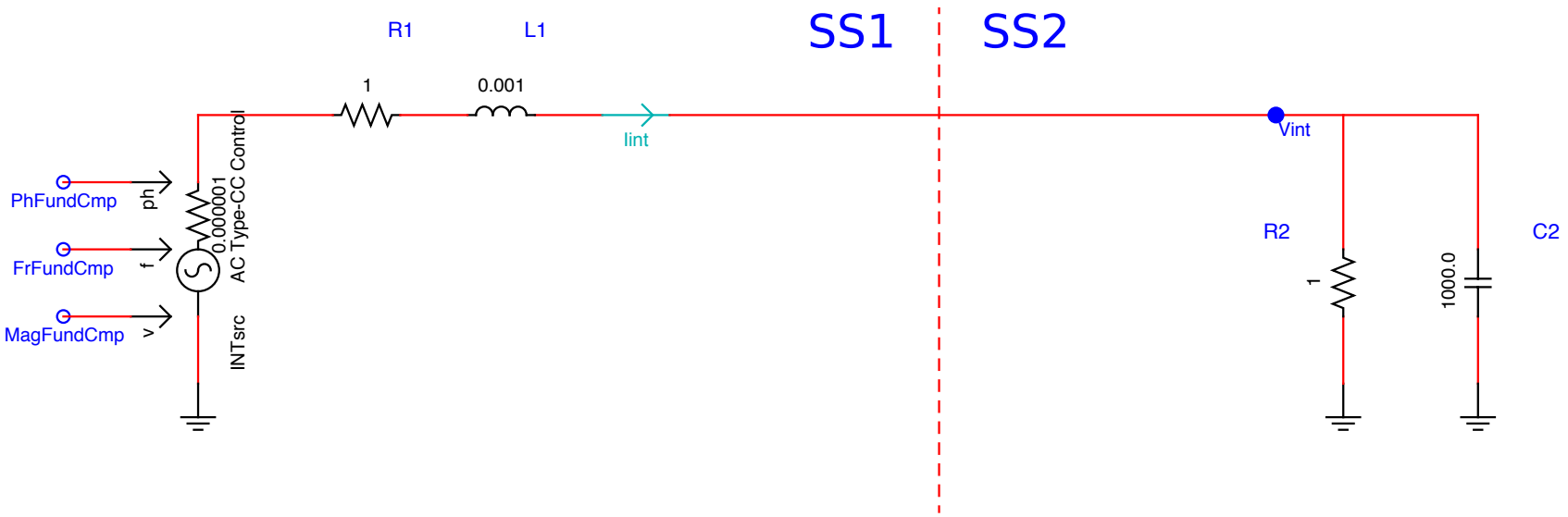
Linux-RTDS-Linux Parallel



Example Applications

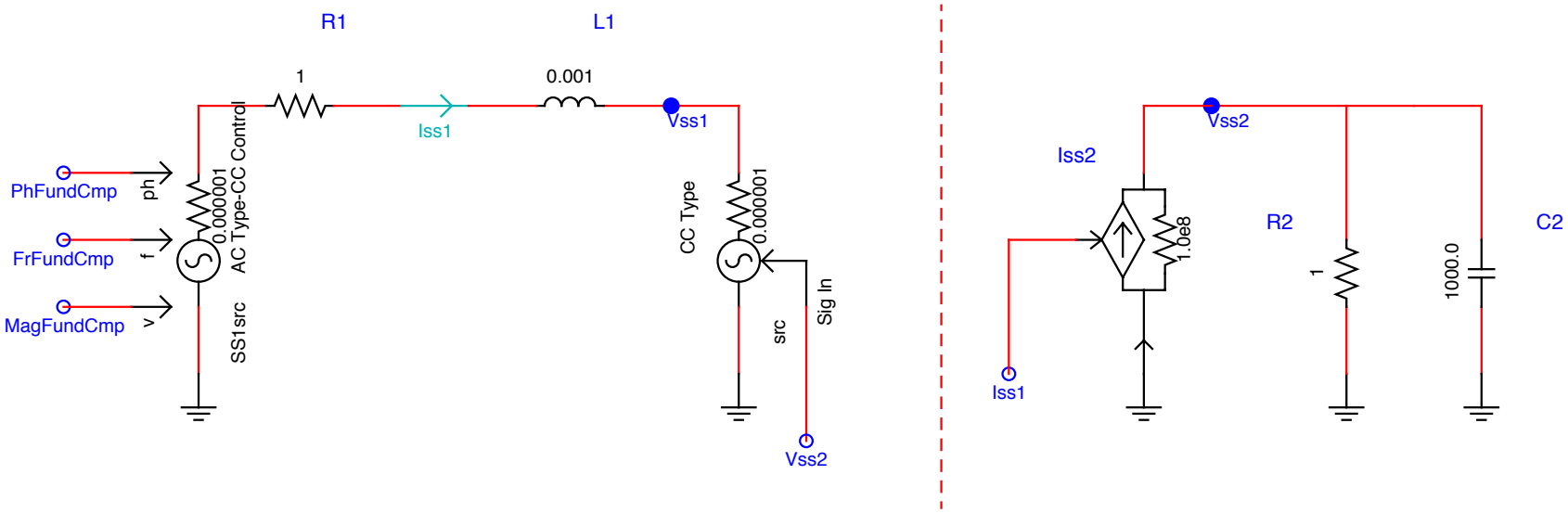
Simple electrical model

- Original network
- To be decoupled into two subsystems (SS1 + SS2)



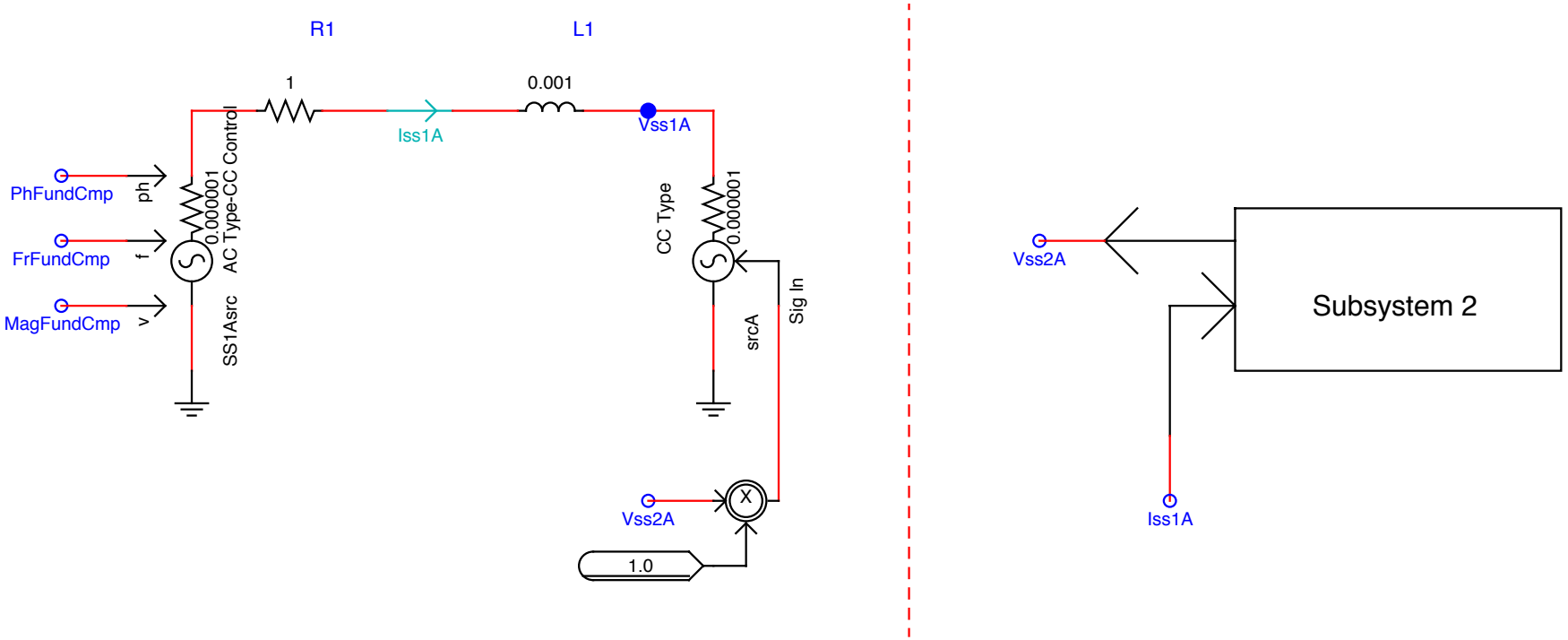
Simple electrical model

- Decoupled network by Ideal Transformer Model (ITM)



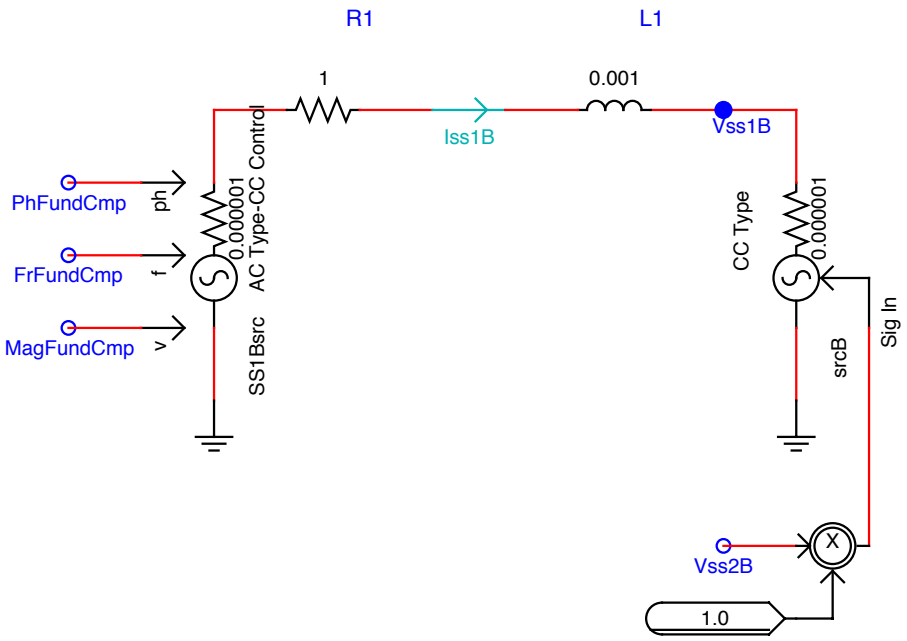
Simple electrical model

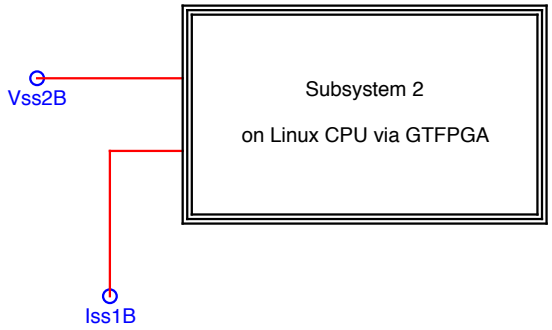
- SS2 is replaced by CBuilder component
- CBuilder component includes solver
- Added Δdt delay for control signals Vss2A & Iss1A



Simple electrical model

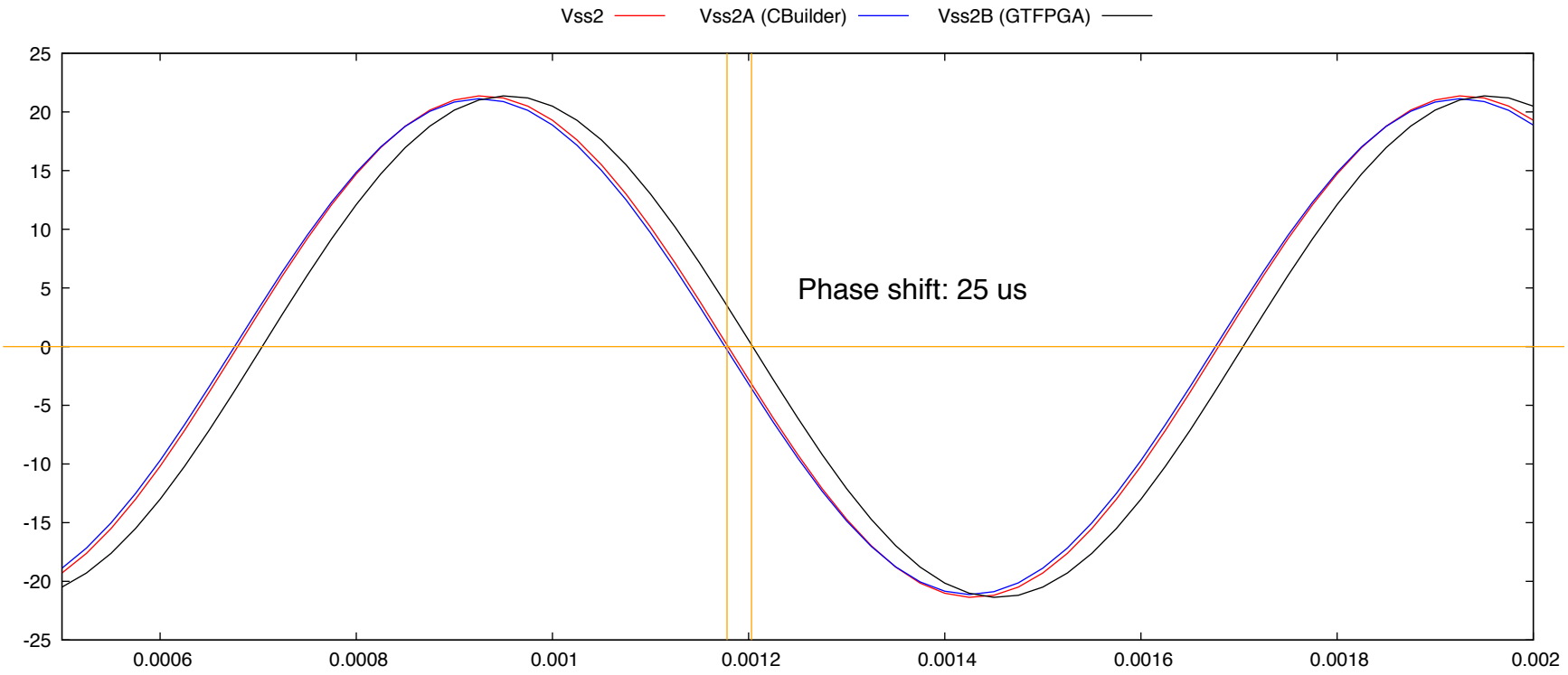
- With GTFPGA interface
- CBuilder code is cross-compiled on Linux
  - Some C and header files are used with some tricks





Results: Simple Circuit

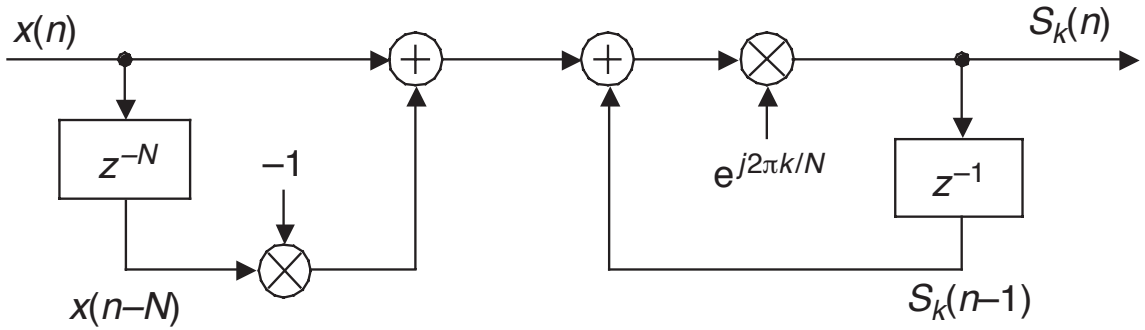
- $f_0 = 1\,\mathrm{kHz},\ \Delta t = 25\,\mu\mathrm{s}$
- 1 time step latency
- 1 additional time step due to coupling of control and power system solution (T2 vs T0 zone)



Example: Sliding DFT

- Simple Recursive implementation
- Window over fundamental period: z^{-N}
- 3 Additions, 2 Multiplications per frequency and sample

$$S_k(n) = e^{j2\pi k/N}[S_k(n-1) + x(n) - x(n-N)]$$

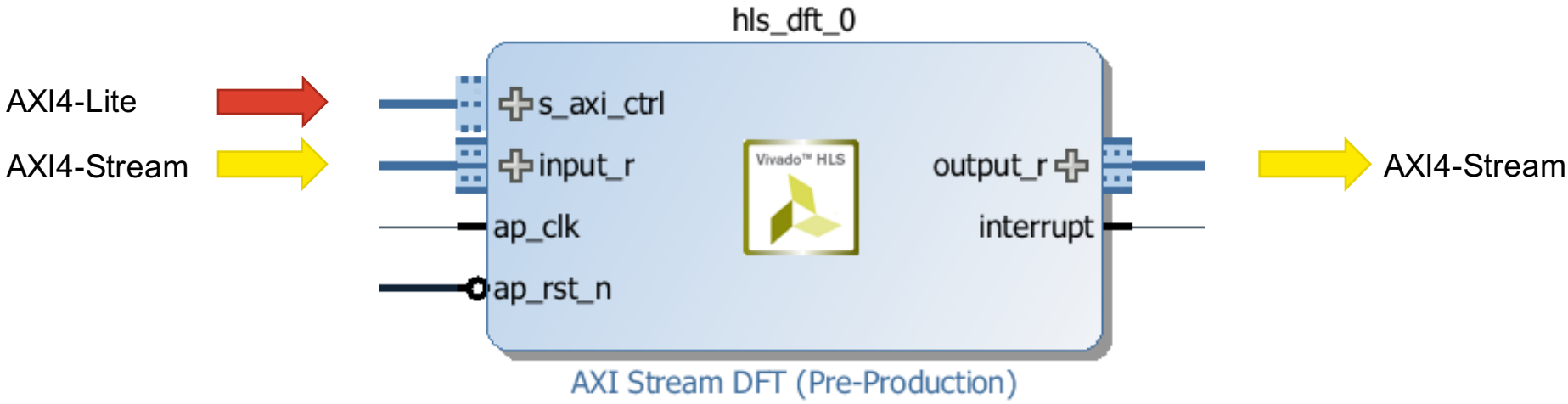


- Performance: 68 clk cycles + 2 clk cycles per harmonic (pipelined)
  - IEEE 754 Single Precission FP

E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 74–80, Mar. 2003.

High-level Synthesis

- IP Integrator

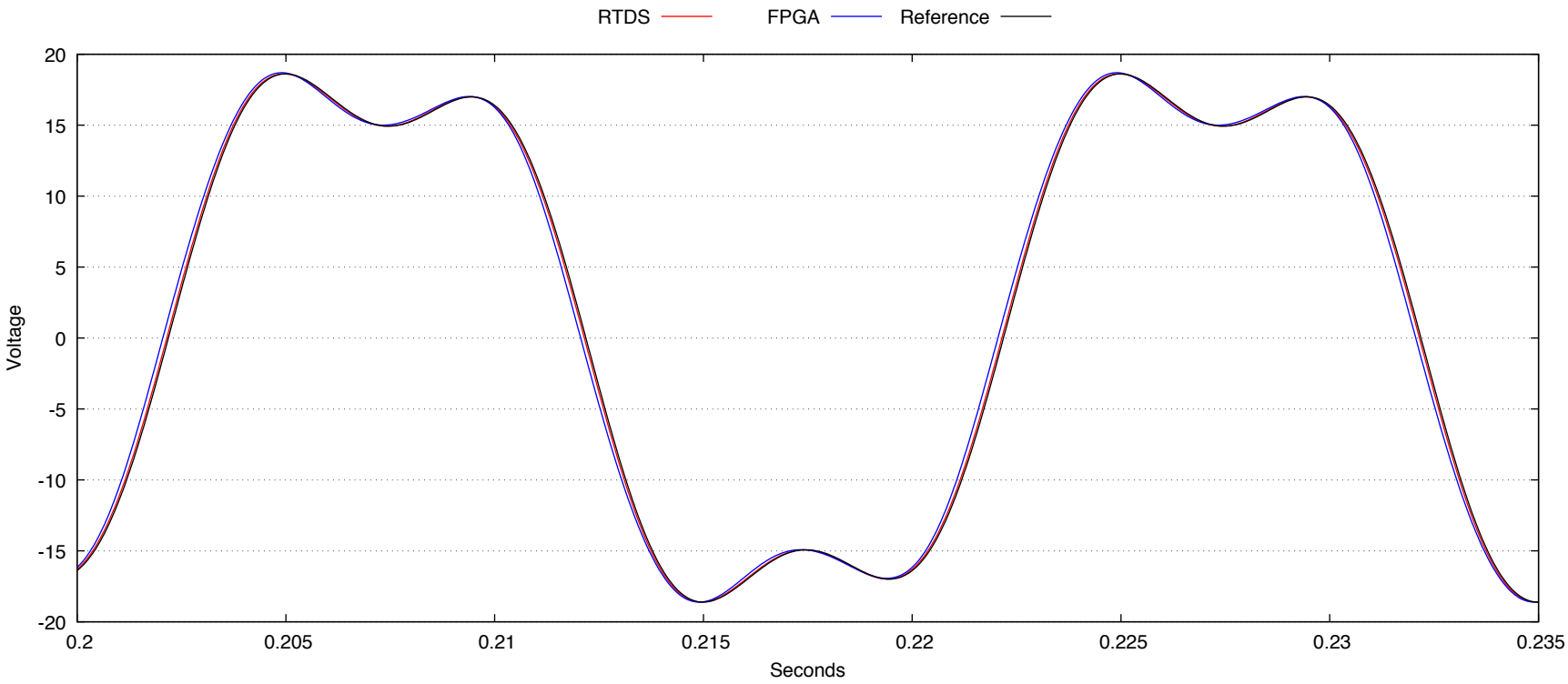


High-level Synthesis

```
void hls_dft(hls::stream<axis> &input, hls::stream<axis> &output,
             float fharmonics[MAX_HARMONICS], ap_int<8> num_harmonics, ap_int<8> decimation) {
        #pragma HLS INTERFACE s_axilite port=return,fharmonics,num_harmonics,decimation bundle=ctrl
        #pragma HLS INTERFACE axis port=input,output
        #pragma HLS STREAM depth=64 variable=input,output
        #pragma HLS DATAFLOW

        /** Previous coefficients for incremental update */
        static std::complex<float> coeffs[MAX_HARMONICS];

        /* Time */
        static float t;
        static ap_int<32> decimation_cnt;

        /** Sliding window of samples */
        static ap_shift_reg<float,NSAMPLES> windows[MAX_VALUES];

        /** AXI Stream signals */
        axis real, imag, refph;

        for (int index = 0; index < MAX_VALUES; index++) {
                /* Read real-valued time-domain data from AXI Stream interface */
                axis in = input.read();

                /* Shift and get data from SLR */
                float newest = in.data;
                float oldest = windows[index].shift(newest, NSAMPLES-1);

                for (int i = 0; i < num_harmonics; i++) {
                        #pragma HLS PIPELINE II=2
```

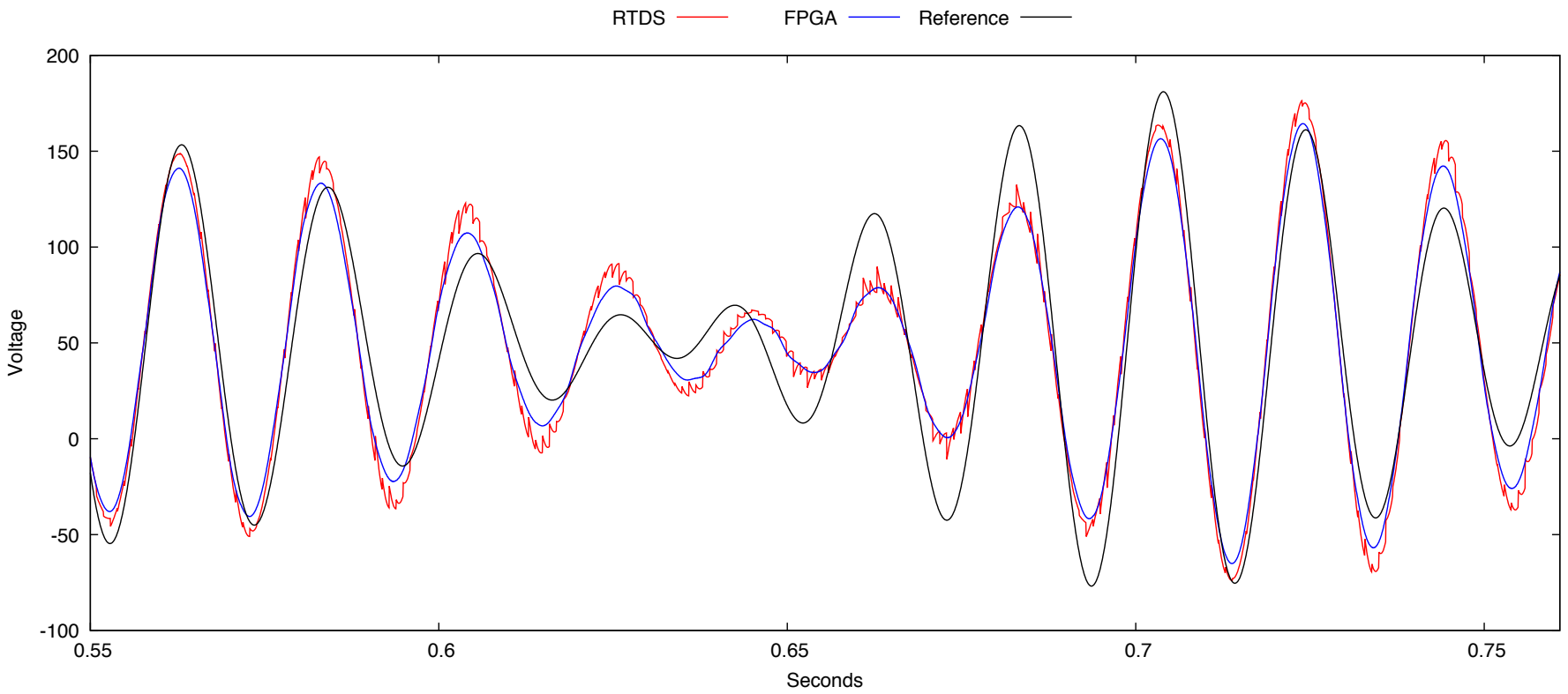
Results: Dynamic Phasor – EMT transformation

- Alignment of reference phase can be problematic
  - Simulations must be started simulateously



Results: Dynamic Phasor – EMT transformation

- DFT32 block on RTDS is subject to distortions in case of fast transients



Conclusion

- Real-time co-simulation interfaces with latencies below 1 time-step are feasible
  - 2 time step round-trip (parallel, 1 time step per direction)
  - 1 time step round-trip (serial, RTDS->Linux->RTDS)
  - Time-step periods **over 25 uS** are pretty stable
- Results align with first tests made between OPAL-RT and RTDS
- Presented architecture is flexible and extensible
  - Comes at the cost of increased complexity and step learning curve
- Interaction of control system and power system solution of RTDS may add an additional time-step latency
- Hard real-time on Linux is possible but fragile
  - Already small changes, different kernel versions or improper coding style will cause issues

Future work: Applications

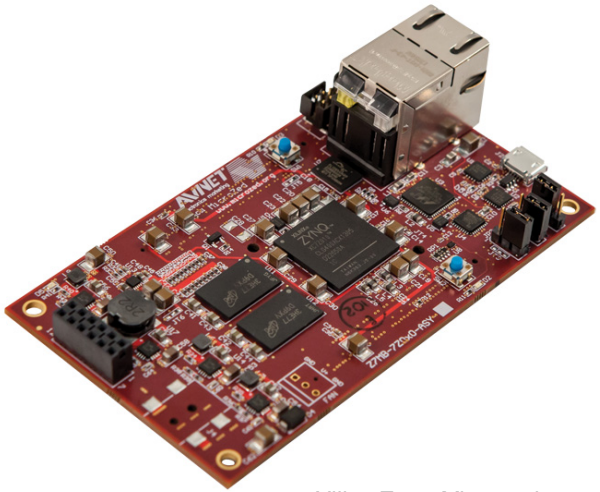
- Use the framework: It‘s there.

- Additional interfaces
  - Industrial ethernet: EtherCAT, PowerLink, ...
    - = Connection to PGS, CWD, EBS labs
  - PHIL amplifiers:
    - = OPAL RT-Link ORION, Aurora
- FPGA based models: C/C++, Simulink
- Real-time co-simulation of power and communication systems
  - Linux runs NS3 as a communication emulator
- Multi-physics co-simulation
  - RTDS: electrical
  - Linux: thermal
  - Multi-rate interface is required
- Flexible control algorithms running on Linux

Future work: Framework

- Add FMC extension module for more fiber optic connections
  - FPGA-to-FPGA
  - FPGA-to-RTDS
- Sync RTDS as Slave
- Small time-step mode (< 10 uS)
- Test other targets / OS‘s
  - RTOS: VxWorks
  - AMP: HermitCore? Baremetal?
  - Architectures: NUMA, ARM (Zynq)
- Simplify usage by adding more automation (ESL)

FMC Quad-SFP Extension Module



Xilinx Zynq Microzed

Acknowledgments

Thanks to my supervisors Marija Stevic & Stefan Lankes
for their advise and support.

E.ON Energy Research Center

Thanks to RTDS Technologies
for their support and access to the GTFPGA interface.

RTDS Technologies

Thanks to OPAL-RT Technologies for testing
and support of the RTDS-OPAL co-simulation interface.

OPAL-RT TECHNOLOGIES

Thanks to Xilinx Inc. for sponsoring software licenses
within the XUP and their excellent tools which made this
work possible.

XILINX
ALL PROGRAMMABLE™

Contact

E.ON Energy Research Center
Mathieustraße 10
52074 Aachen
Germany

Steffen Vogel
stvogel@eonerc.rwth-aachen.de
http://www.eonerc.rwth-aachen.de

http://www.steffenvogel.de

ACS | Automation of Complex Power Systems

E.ON Energy Research Center

RWTH AACHEN UNIVERSITY