Open-Source QEMU and RTL Co-simulation

Edgar E. Iglesias

Agenda

- QEMU at AMD/Xilinx
- Emulation technologies
- What, How and why Co-simulation?
- DARPA, POSH and Xilinx
- Co-simulation setups
- Live demo

Modeling Technologies: Software Emulator (QEMU)

Virtual Platform for SW developers

- Open-Source **Scalable distribution & cost model,** Popular in Open-Source community
- @Xilinx from 2009 MicroBlaze, Power PC, **ARM (Zynq, Zynq MPSoC, Versal)**, x86
- Transaction level Fast but not cycle accurate, Linux boot **2sec to user-space, 2min to prompt**
- Debug & Profiling GDB/XSDB, traces, code-coverage and error-injection (-ve testing)
- Co-simulation SystemC/TLM-2, RTL and Hybrid
- Value Shift left (early SW development), Flexibility, Speed, Cost

Users

- Internal BootROM, System Software, SVT
- External Petalinux, Vitis HW-Emulation, GitHub (Roll your own)

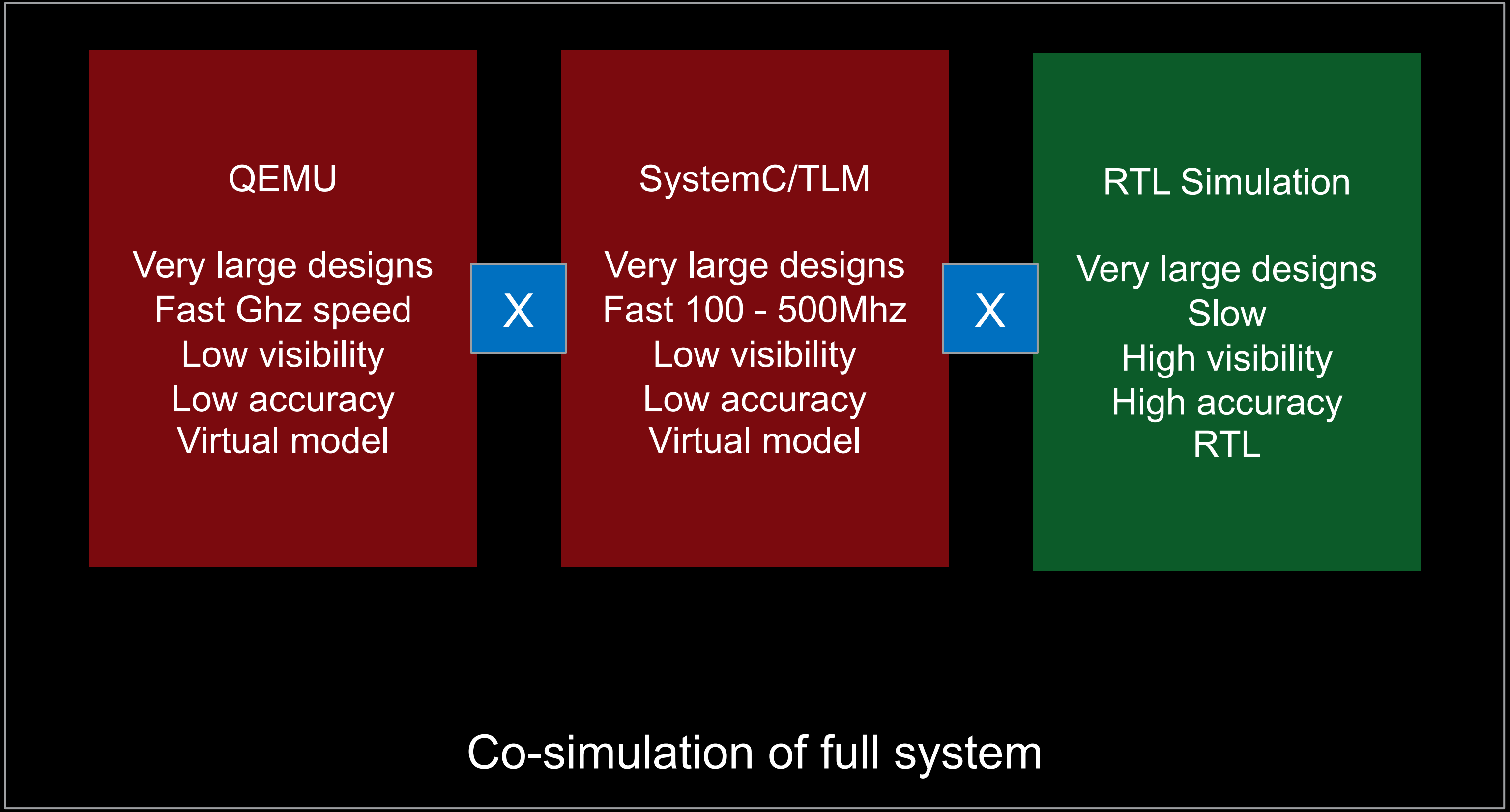
External

- DARPA/POSH AMD/Xilinx chosen for Open-Source QEMU Co-simulation efforts
- Customers/Vendors X (PetaLinux), Y (Github) and Z (Vitis)

Emulation, technologies and trade-offs

QEMU	SystemC/TLM	FPGA Prototyping	HW Emulation	RTL Simulation
Very large designs	Very large designs	Small – Med designs	Large designs	Very large designs
Fast Ghz speed	Fast 100 - 500Mhz	1 - 200 Mhz	Slow 1 Mhz	Slow
Low visibility	Low visibility	Medium visibility	High visibility	High visibility
Low accuracy	Low accuracy	High accuracy	High accuracy	High accuracy
Virtual model	Virtual model	RTL	RTL	RTL

What is Co-simulation?



QEMU

Very large designs
Fast Ghz speed
Low visibility
Low accuracy
Virtual model

X

SystemC/TLM

Very large designs
Fast 100 - 500Mhz
Low visibility
Low accuracy
Virtual model

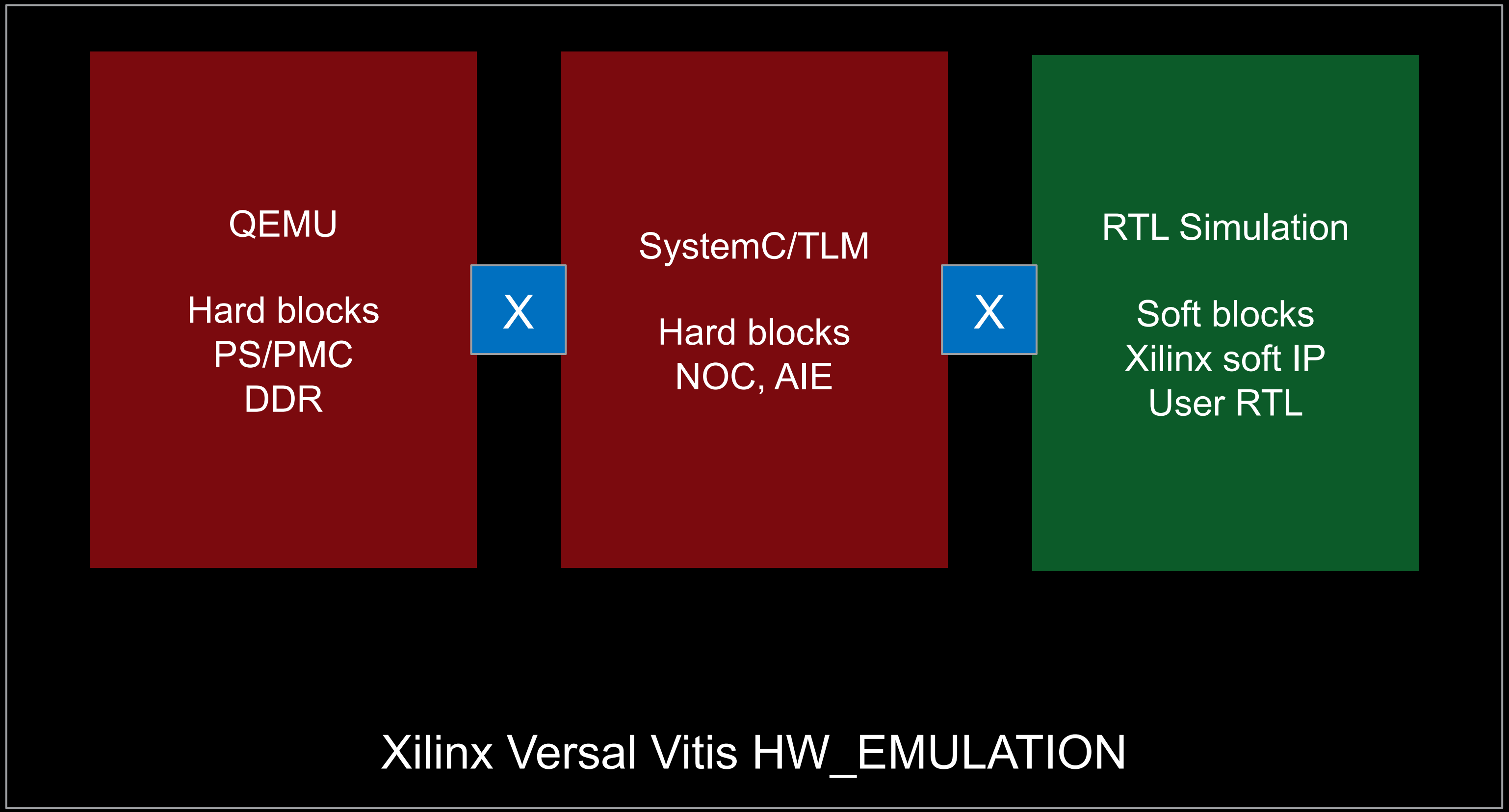
X

RTL Simulation

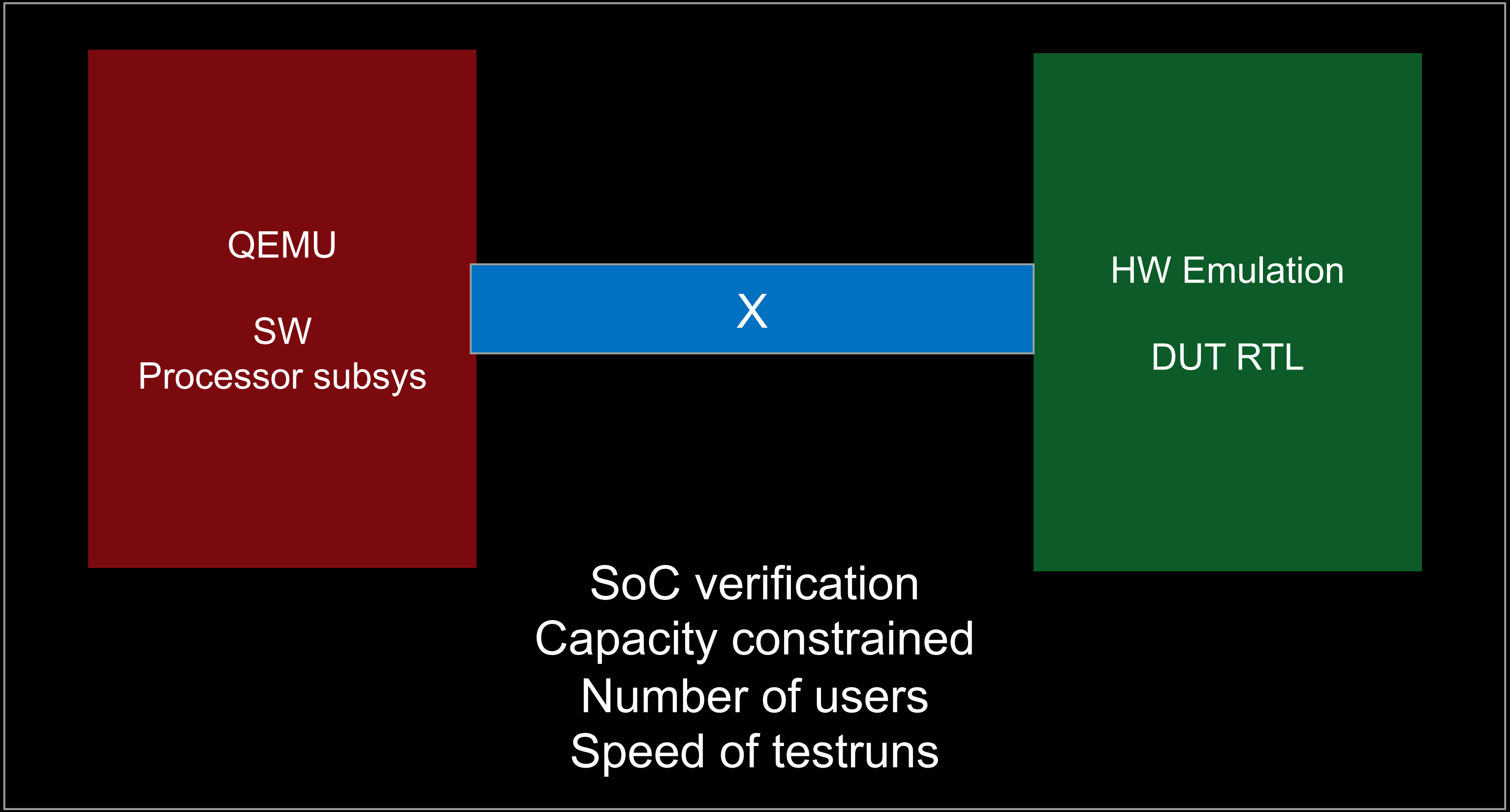
Very large designs
Slow
High visibility
High accuracy
RTL

Co-simulation of full system

Why Co-simulation?

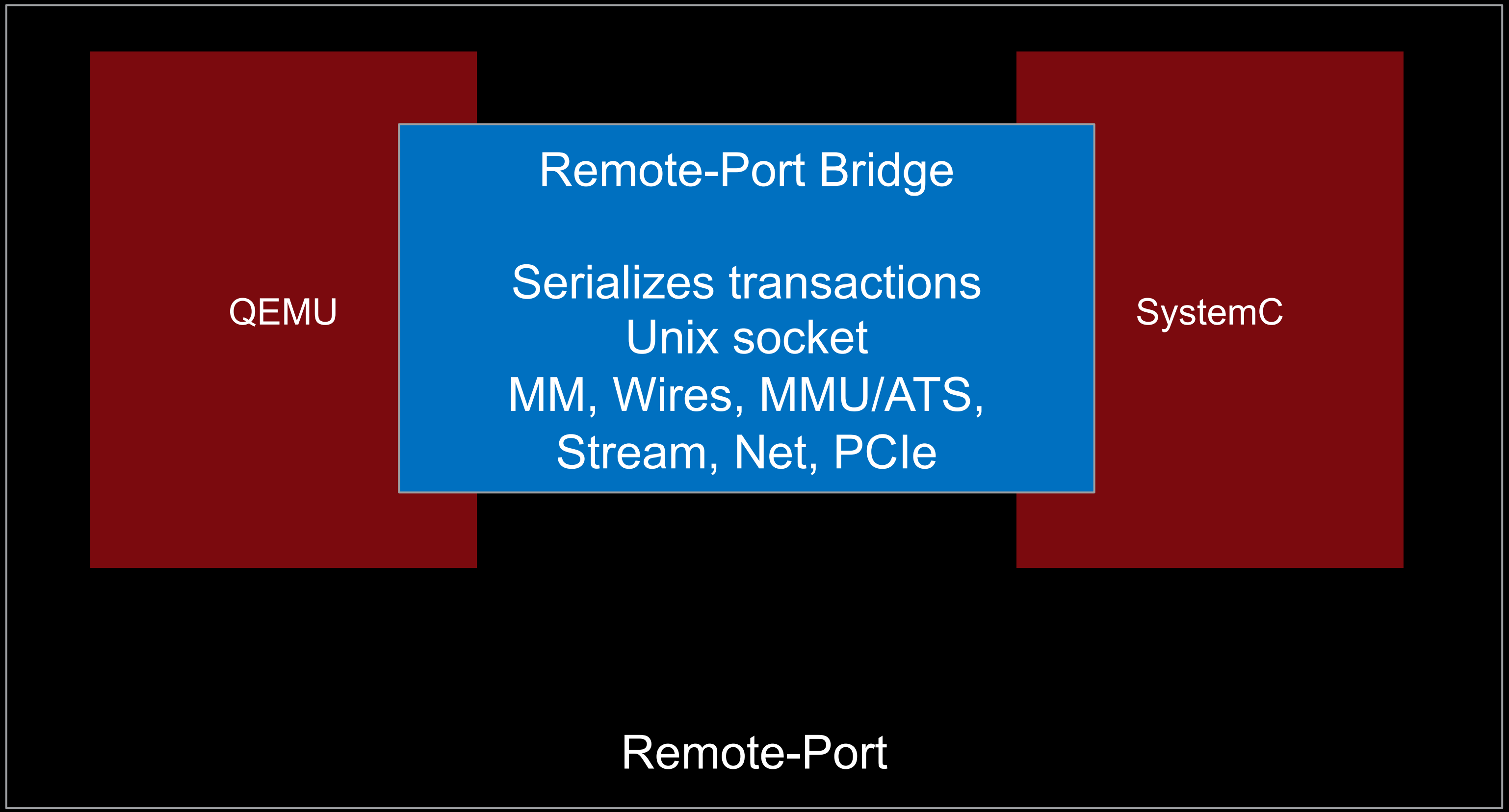


Why Co-simulation?

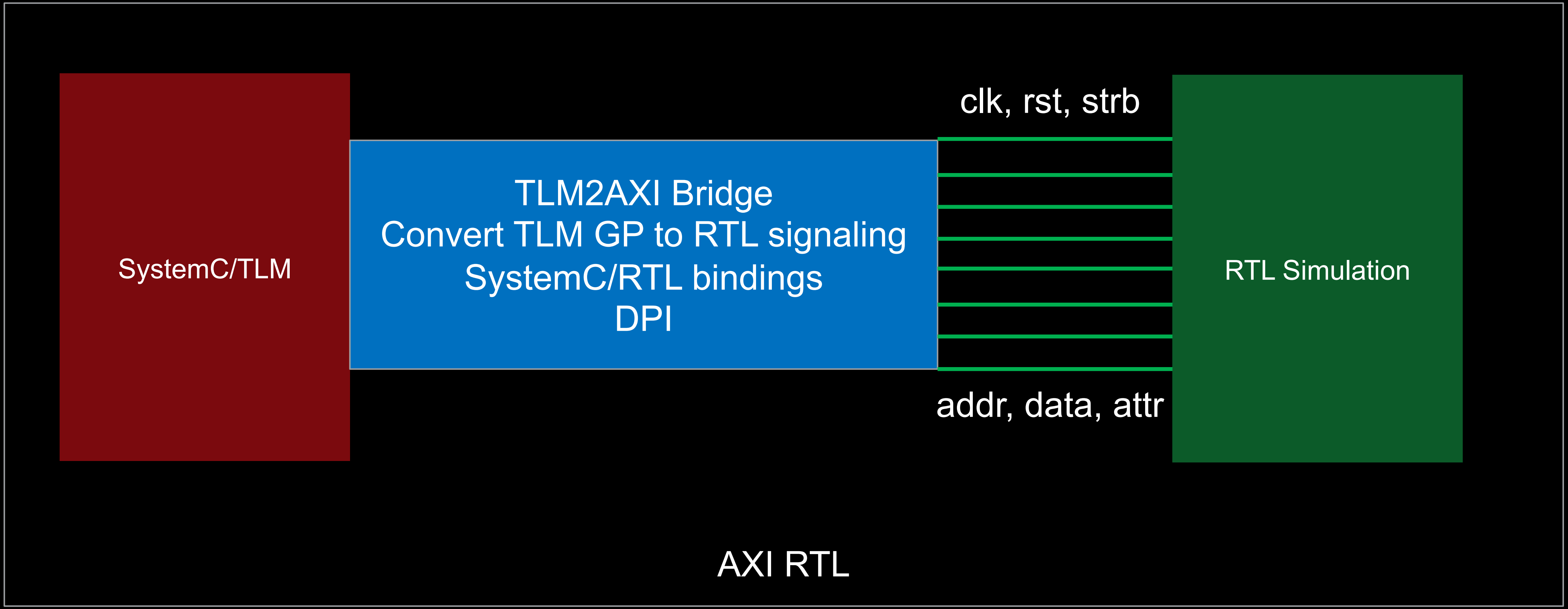


SoC verification
Capacity constrained
Number of users
Speed of testruns

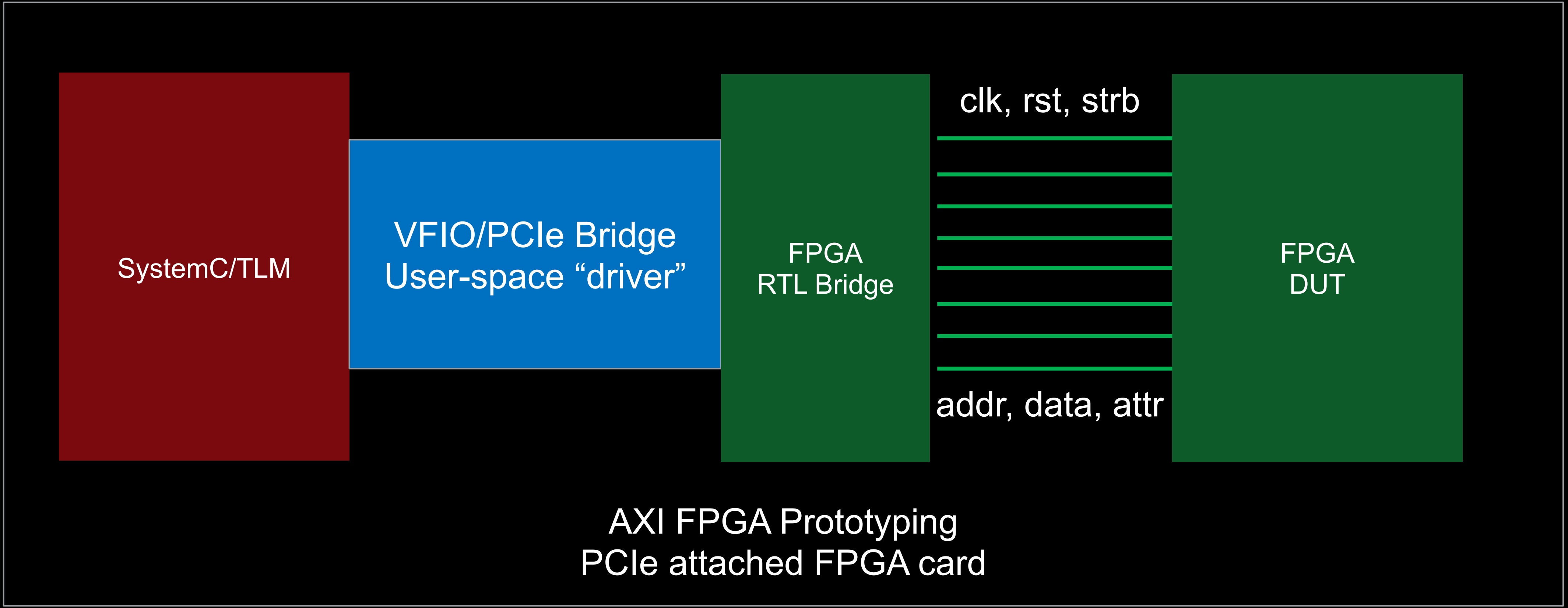
How is bridging done?



How is bridging done?



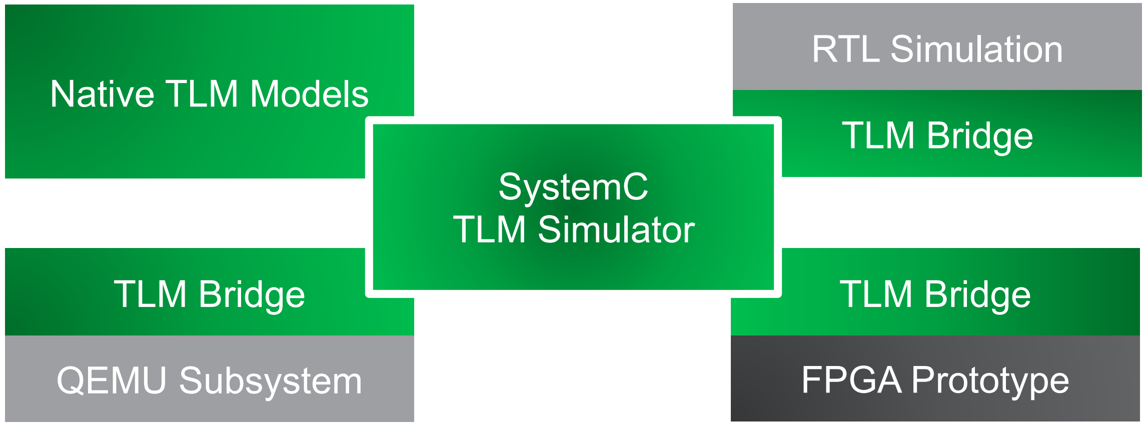
How is bridging done?



Open Source Mixed Simulation Environment (DARPA/POSH)

LibSystemCTLM-SoC

https://github.com/Xilinx/libsystemctlm-soc



Phase 1a

Phase 1b

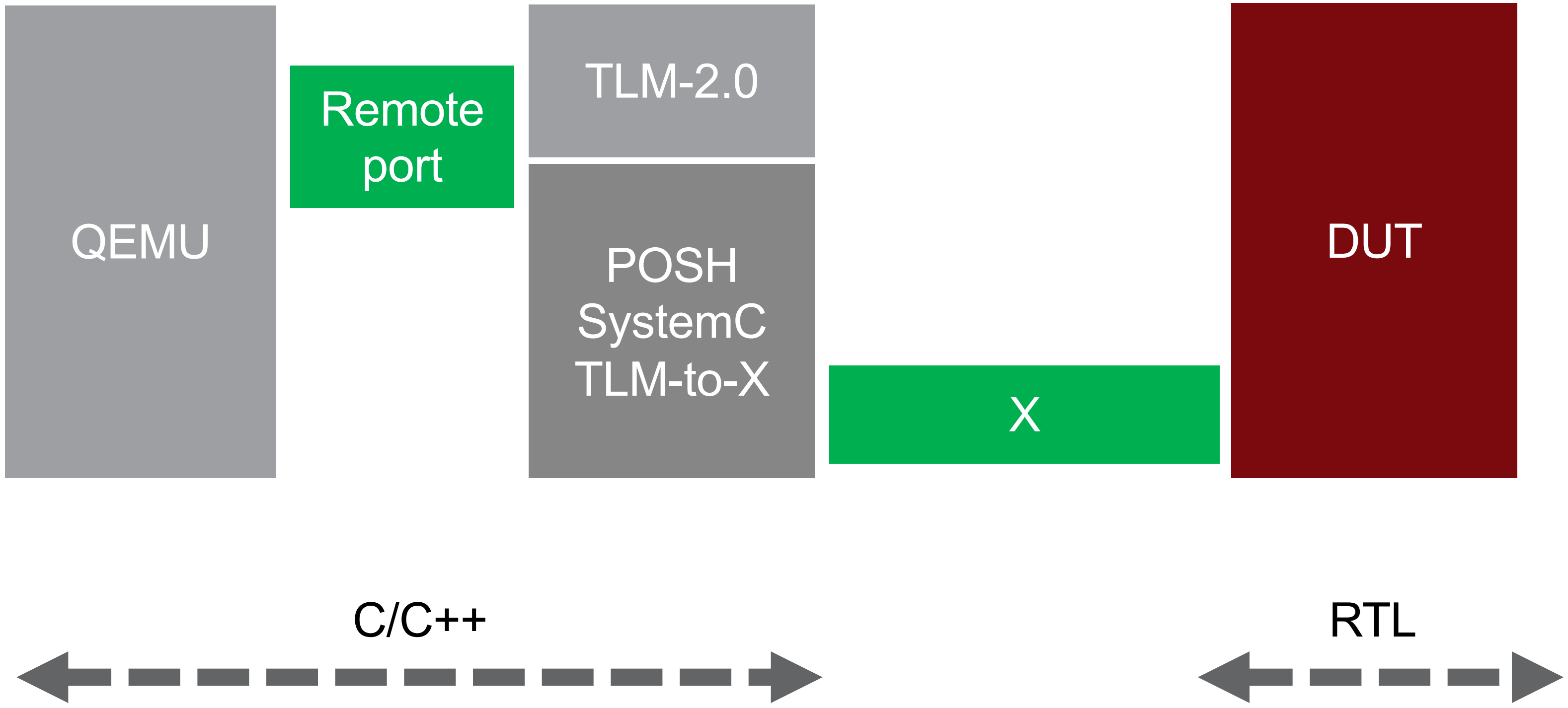
Based on Open-Source Projects

QEMU · accellera · eclipse · VERILATOR · Kactus2 · pytest · GNU Debugger

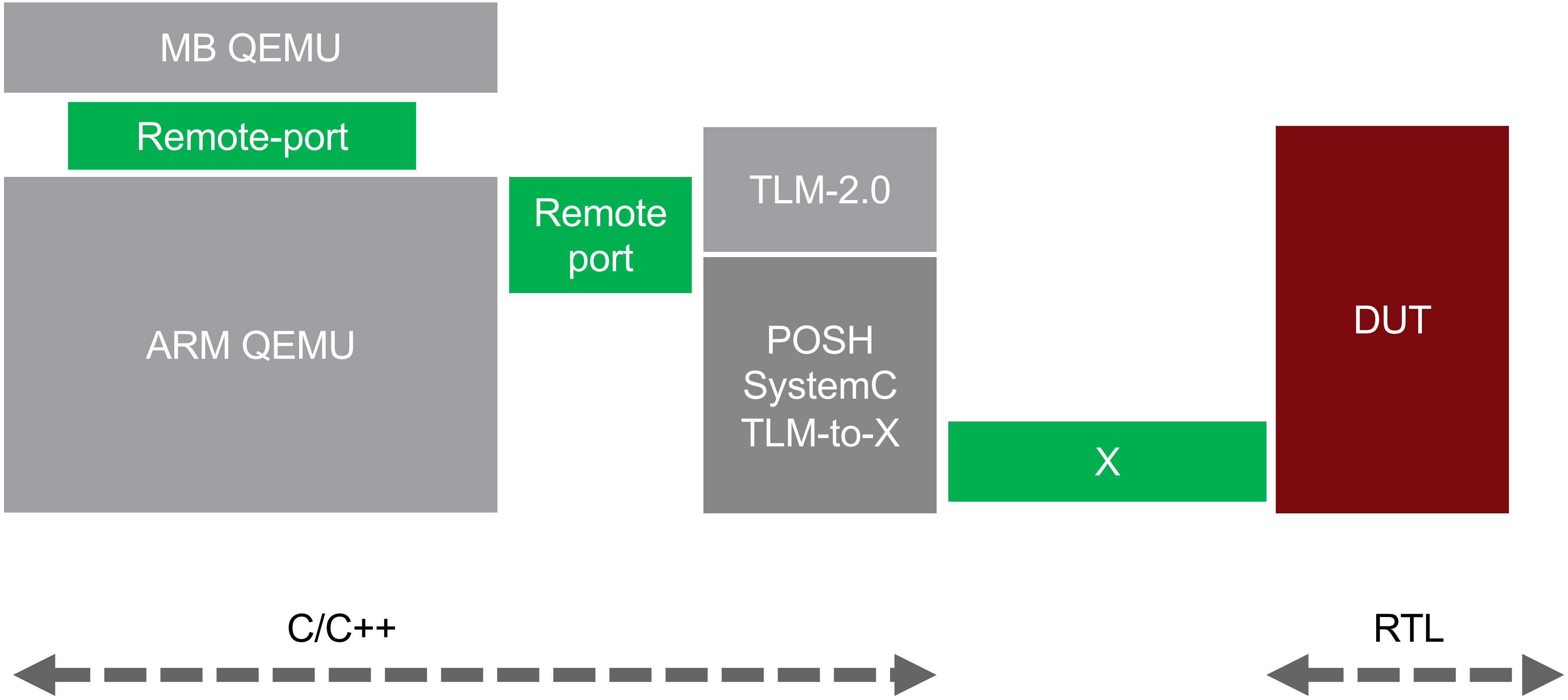
POSH – Xilinx – Libsystemctlm-soc

- TLM simulation bridges, protocol checkers and traffic generators
  - APB, AXI (3, 4, Lite, Stream), ACE, CHI, CCIX, CXS, PCIe, XGMII, Native, VFIO
- TLM RTL/FPGA prototyping bridges
  - AXI (3, 4, Lite), ACE, CHI, CCIX, PCIe
- GDB + GTKWave debugging view
- SoC Emulator auto-stiching from IP-XACT (Kactus2, QEMU, SystemC, Verilator, Pysimgen)

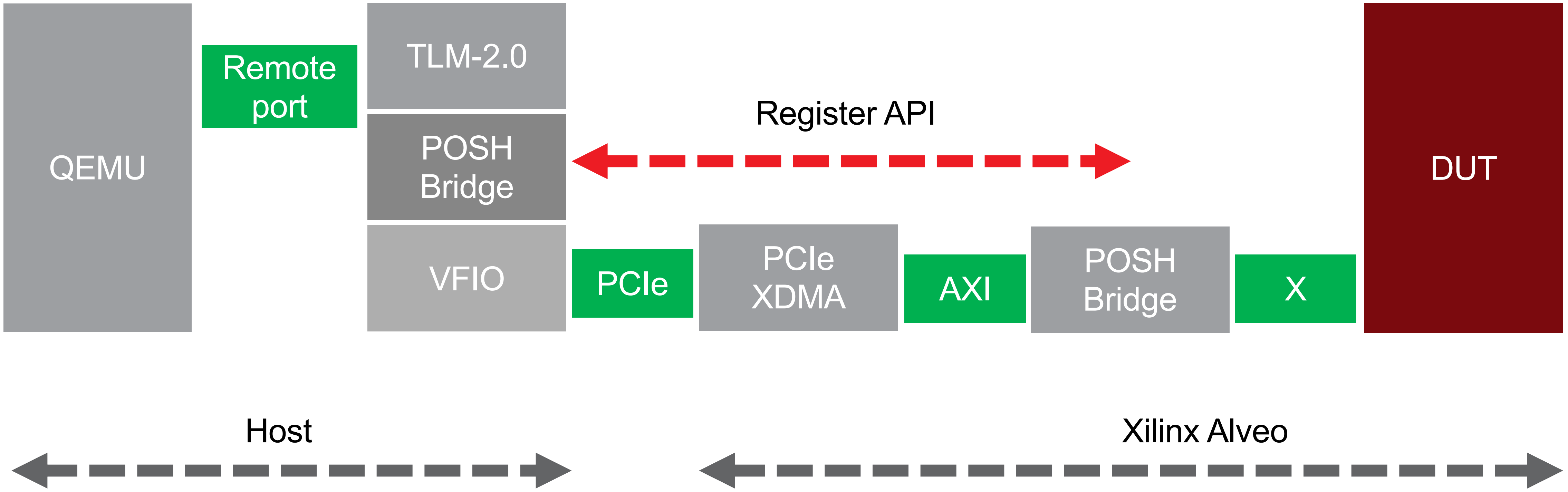
TLM2 bridges RTL simulation



QEMU heterogeneous

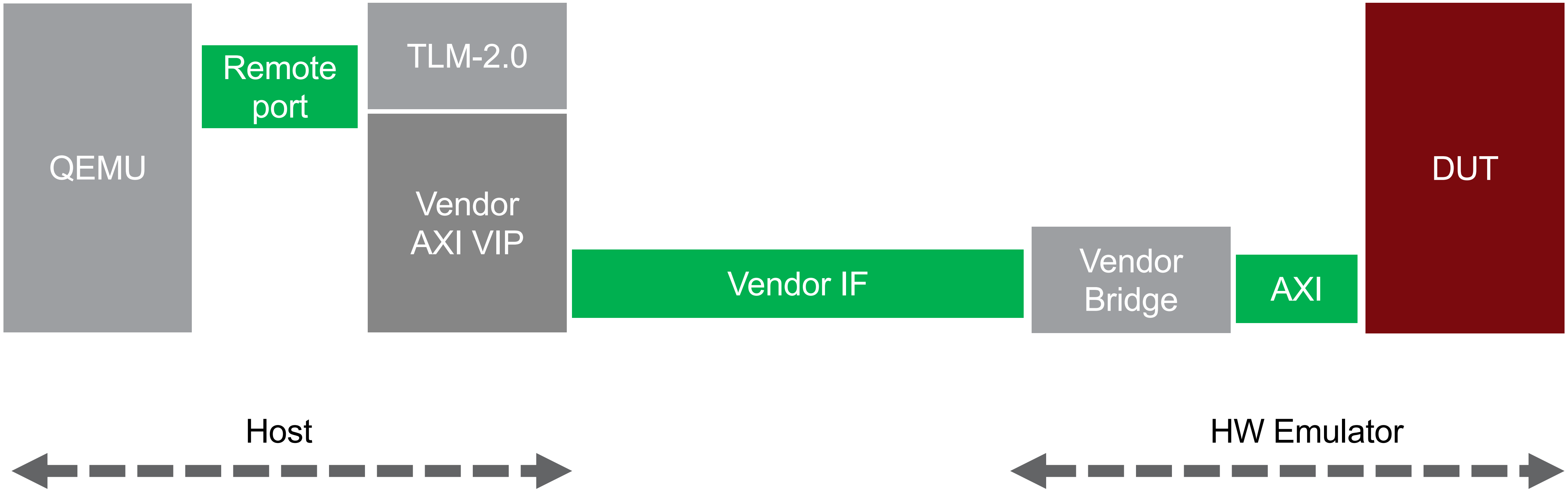


TLM2 bridges FPGA Prototyping



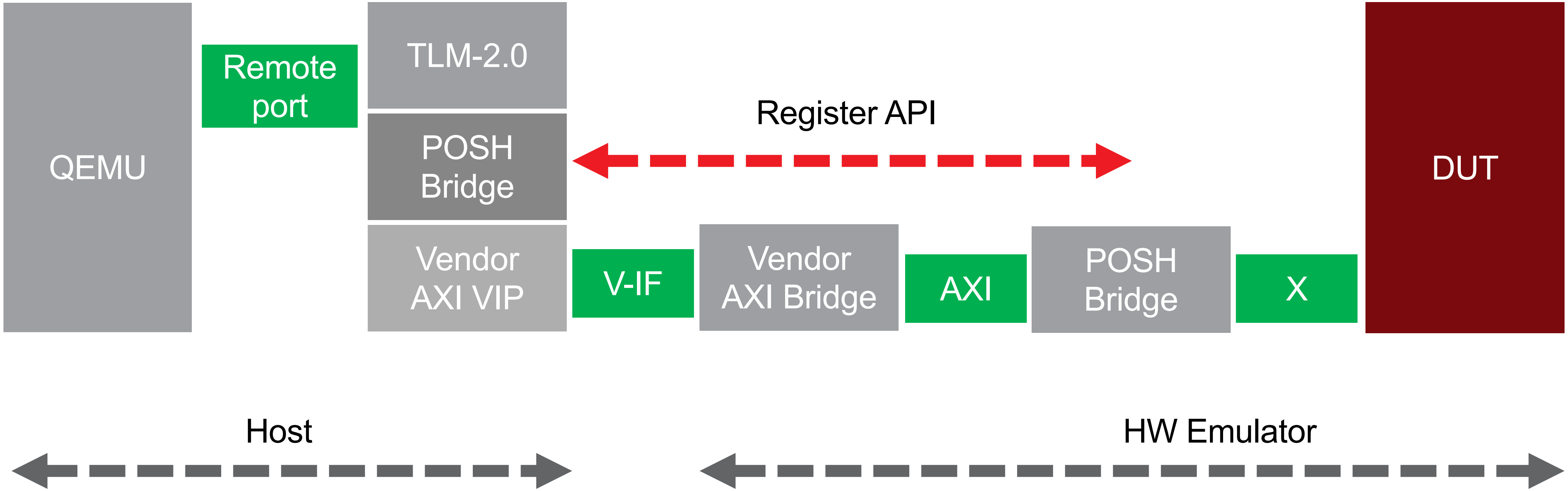
TLM2 bridges Emulator #1

Pros: Fast, sort of “Vendor neutral”
Cons: Not debug-portable, Licensing AXI VIP

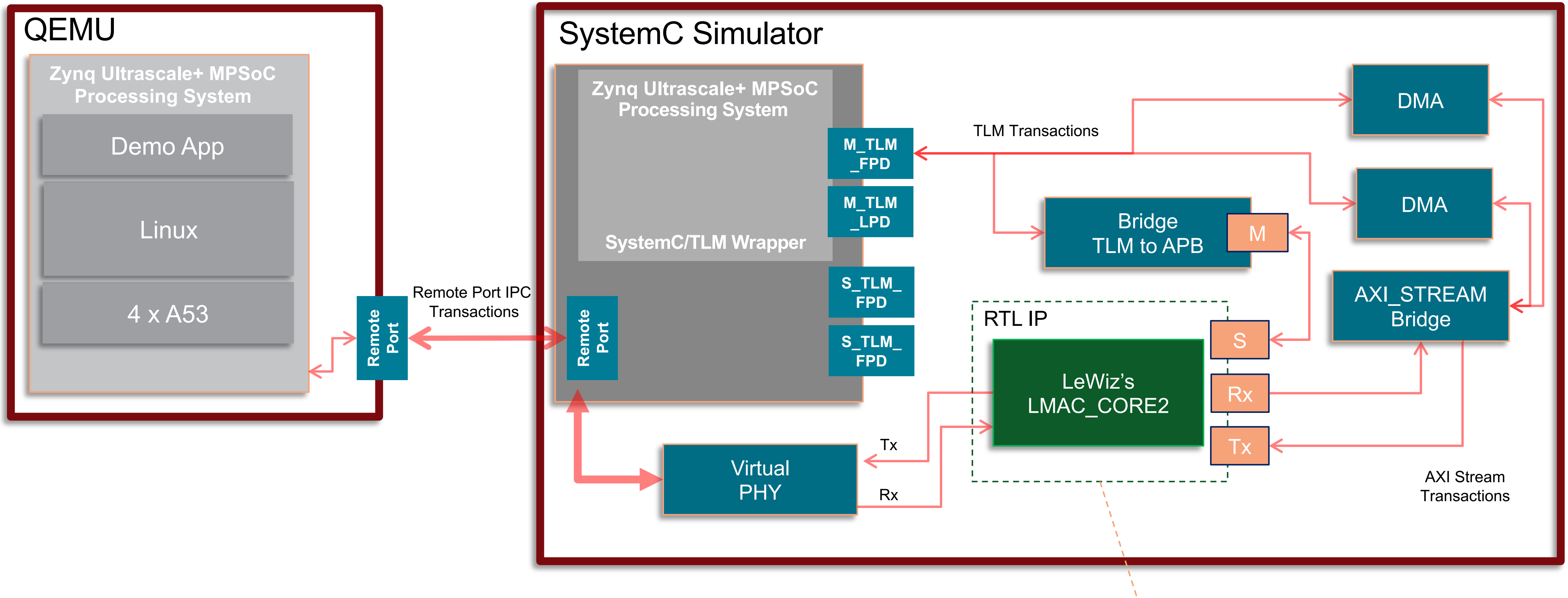


TLM2 bridges Emulator #2

Pros: Vendor neutral, debug-portable
Cons: Slow, Licensing AXI VIP



LMAC demo



LeWiz Communications Ethernet MAC Core2 10G/5G/2.5G/1G

https://github.com/lewiz-support/LMAC_CORE2

Thank you