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
### 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**
  - Very large designs
  - Fast Ghz speed
  - Low visibility
  - Low accuracy
  - Virtual model

- **SystemC/TLM**
  - Very large designs
  - Fast 100 - 500Mhz
  - Low visibility
  - Low accuracy
  - Virtual model

- **FPGA Prototyping**
  - Small – Med designs
    - 1 - 200 Mhz
    - Medium visibility
    - High accuracy
    - RTL

- **HW Emulation**
  - Large designs
    - Slow 1 Mhz
    - High visibility
    - High accuracy
    - RTL

- **RTL Simulation**
  - Very large designs
    - Slow
    - High visibility
    - High accuracy
    - RTL
What is Co-simulation?

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

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

RTL Simulation
Very large designs
Slow
High visibility
High accuracy
RTL

Co-simulation of full system
Why Co-simulation?

- QEMU
  - Hard blocks
  - PS/PMC
  - DDR

- SystemC/TLM
  - Hard blocks
  - NOC, AIE

- RTL Simulation
  - Soft blocks
  - Xilinx soft IP
  - User RTL

Xilinx Versal Vitis HW_EMULATION
Why Co-simulation?

QEMU

SW Processor subsys

X

HW Emulation

DUT RTL

SoC verification
Capacity constrained
Number of users
Speed of testruns
How is bridging done?

Remote-Port Bridge

Serializes transactions
 Unix socket
 MM, Wires, MMU/ATS,
 Stream, Net, PCIe
How is bridging done?

SystemC/TLM

TLM2AXI Bridge
Convert TLM GP to RTL signaling
SystemC/RTL bindings
DPI

clk, rst, strb

addr, data, attr

RTL Simulation

AXI RTL
How is bridging done?

- SystemC/TLM
- VFIO/PCIe Bridge User-space “driver”
- FPGA RTL Bridge
- clk, rst, strb
- addr, data, attr
- FPGA DUT
- AXI FPGA Prototyping
- PCIe attached FPGA card
Open Source Mixed Simulation Environment (DARPA/POSH)
LibSystemCTLM-SoC
https://github.com/Xilinx/libsystemctlm-soC

Based on Open-Source Projects
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

MB QEMU
Remote-port

ARM QEMU
Remote port

TLM-2.0
POSH SystemC TLM-to-X

DUT

C/C++
RTL
TLM2 bridges FPGA Prototyping

QEMU | Remote port | TLM-2.0 | Register API | DUT

POSHT Bridge | VFIO | PCIe | PCIe XDMA | AXI | POSH Bridge | X

Host | Xilinx Alveo
TLM2 bridges Emulator #1

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

**Vendor IF**

**Vendor Bridge**

**AXI**

**DUT**

**Remote port**

**QEMU**

**TLM-2.0**

**Vendor AXI VIP**

**Host**

**HW Emulator**
TLM2 bridges Emulator #2

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

QEMU
Remote port
TLM-2.0
POSH Bridge
Vendor AXI VIP
V-IF
Vendor AXI Bridge
AXI
POSH Bridge
X

Register API

Host
HW Emulator
DUT
LeWiz Communications Ethernet MAC Core2 10G/5G/2.5G/1G

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