Hardware Software Co-verification in Hybrid QEMU/HDL Environment

Radosław Nawrot, Aldec Inc.

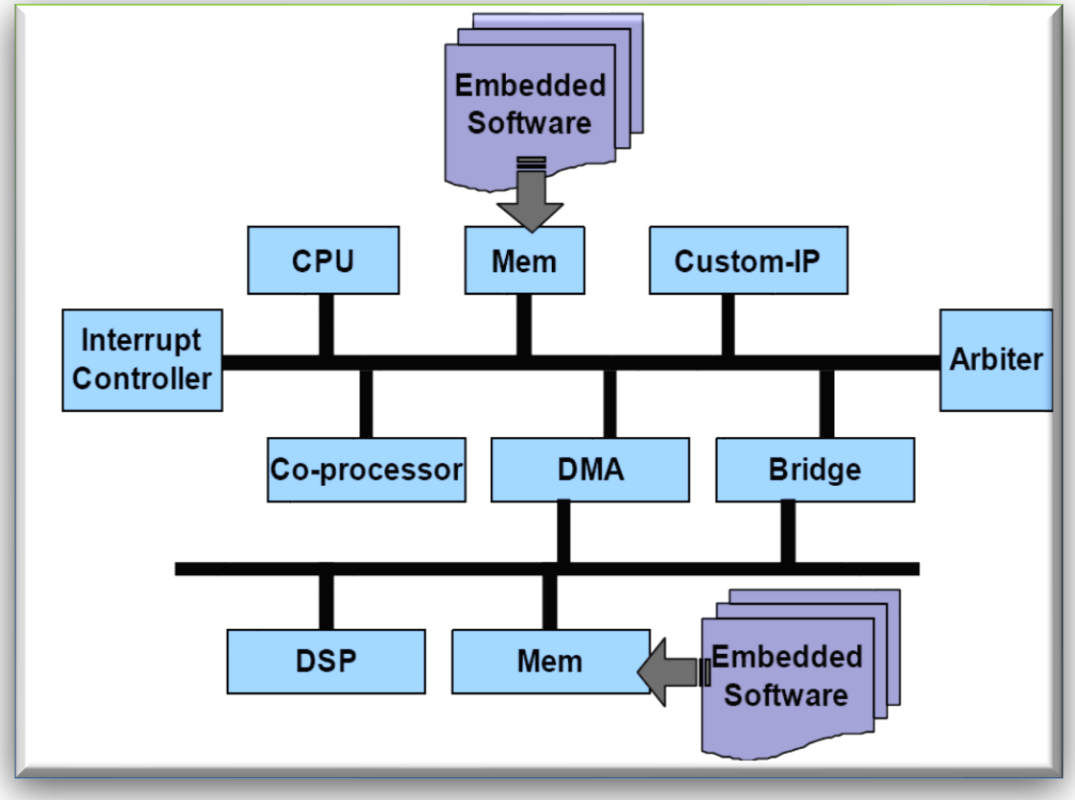
Krzysztof Szczur, Aldec Inc.

accellera SYSTEMS INITIATIVE

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

SoC verification challenges

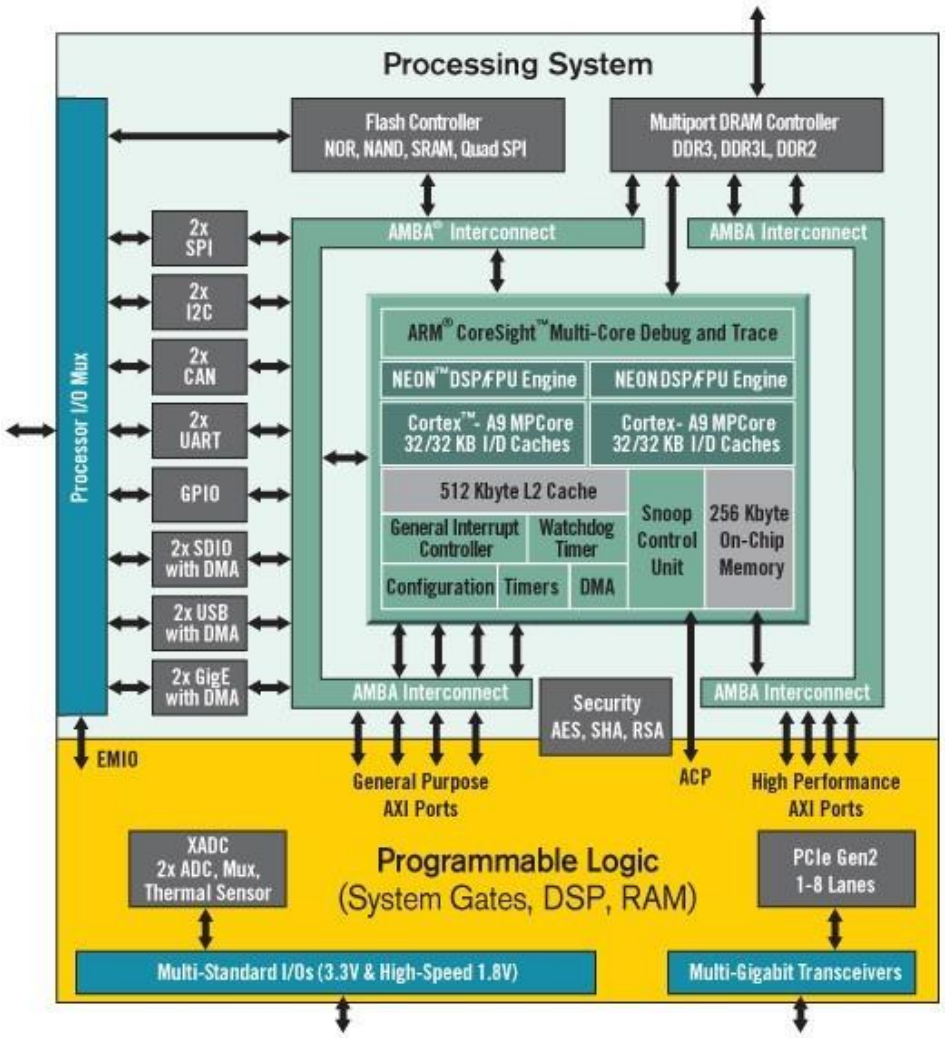
- Contains complex hardware and embedded software
  - both need verification
  - software on critical-path
- Heterogeneous architecture and hierarchical Network-on-Chip (NoC)
  - Corner case hardware problems
  - System level integration problems
- Standard IP reuse: CPU, DMA, MEM … - side effects:
  - Design size fast growth
  - Slow HDL simulation
- Custom IP-Cores
  - Need cycle accuracy for debugging
  - Cannot be fully verified at block level (out of SoC context)



SoC Designs in FPGA Domain

Xilinx Zynq and the like…

- Processing System
  - ARM Cortex CPU
  - Standard I/O peripherals
- Programmable Logic
  - Classic FPGA used for:
    - More peripherals
    - Glue logic & Bridge



Virtual Platforms & Virtual Machines Pros & Cons

(+) Available earlier than system-level RTL code

(+) Fast enough for firmware/software development

(+) Integrated with software development tool-chain

(-) Not cycle accurate

(-) Virtual models not available for Custom-IP

Virtual Machine

QEMU

Versatile – many architectures:

- ARM, RISC-V, SPARC, MIPS
- x86, PowerPC, MIPS

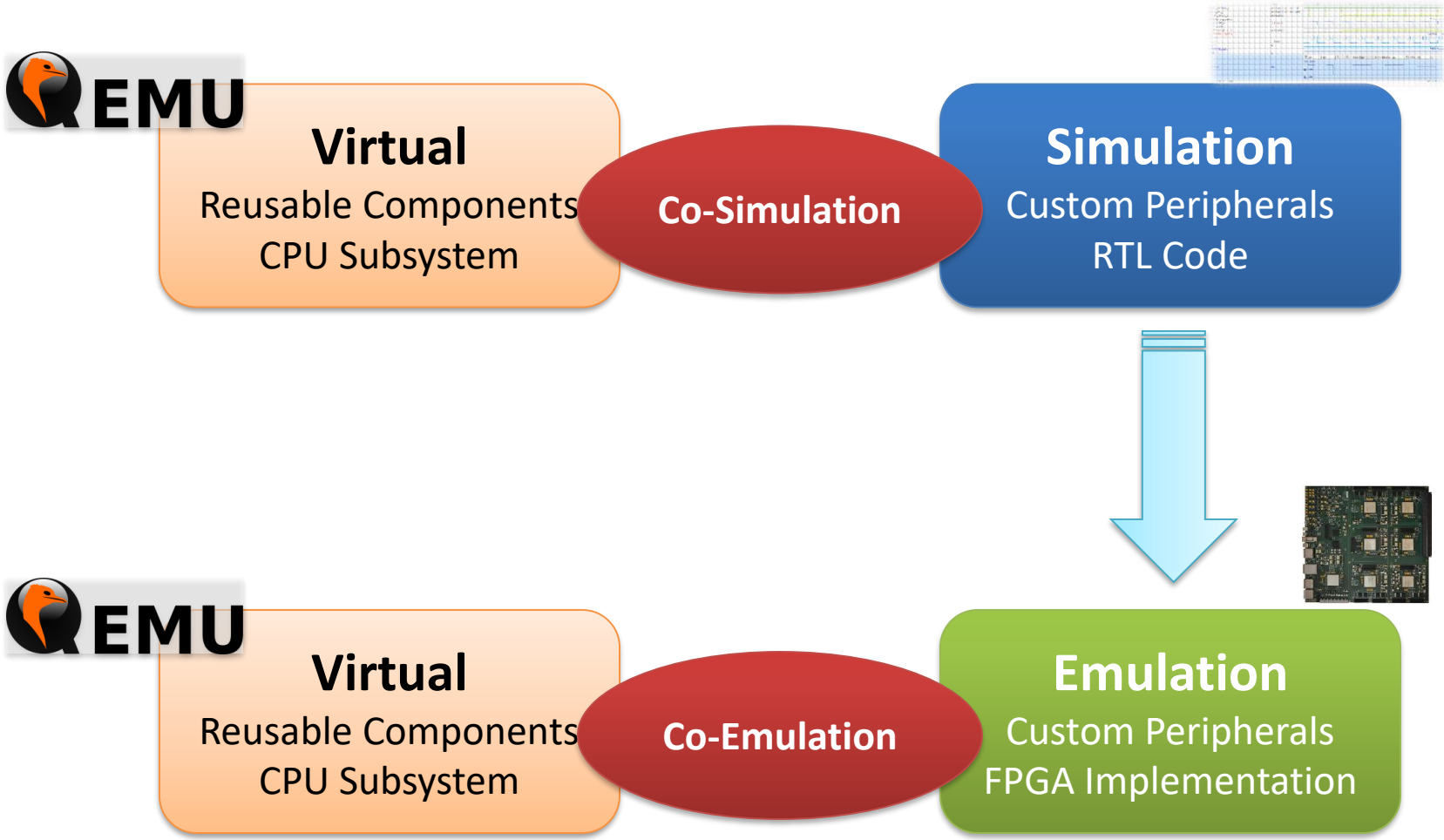
Virtual Platforms

ARM® Fast Models
Virtual Platforms

MULTICORE DESIGN SIMPLIFIED
imperas

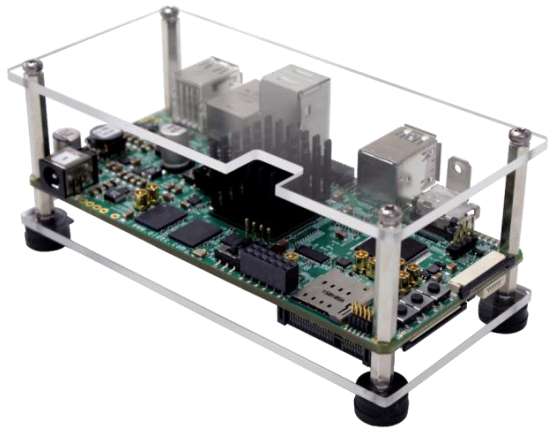
OVP
Open Virtual Platforms

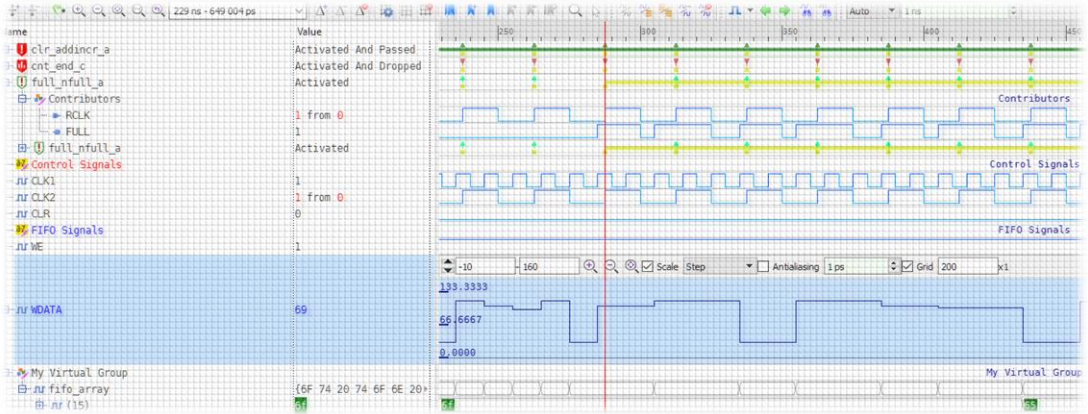
Hybrid co-simulation and co-emulation



- Custom IP development
- Block-level verification
- QEMU – software driven testbench

- SoC Integration & bring up
- Drivers & firmware development
- Complete SoC model for SDK
- QEMU – integral part of SoC model





QEMU co-simulation with Riviera-PRO

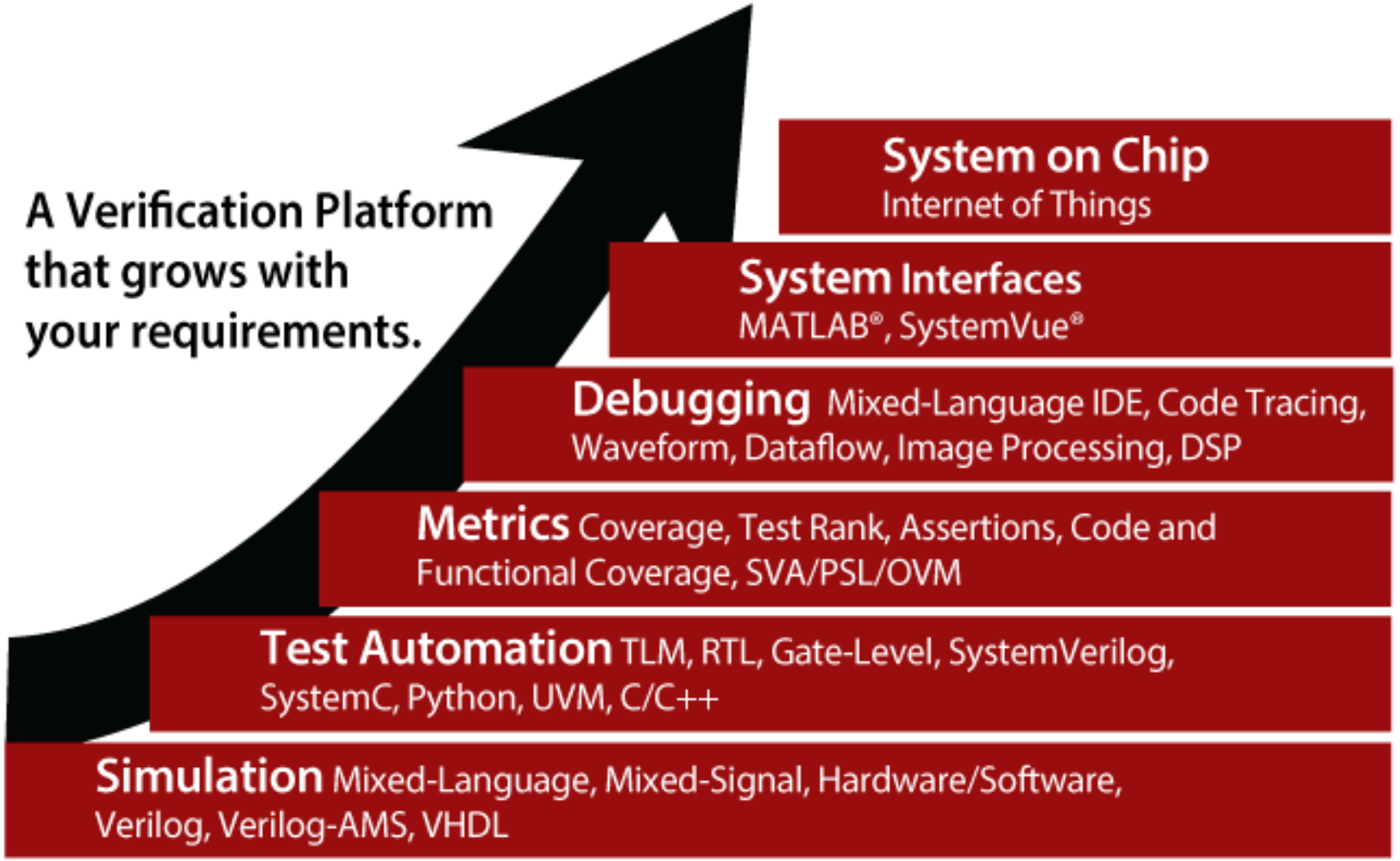
QEMU co-simulation platform

- Parts of the flow:
  - Riviera-PRO Advance Verification Platform
  - Aldec AXI BFM
  - Aldec QEMU Bridge
  - QEMU Emulator

QEMU co-simulation highlights

- Full debug capabilities of RTL IP Core in Riviera-PRO simulator:
  - Waveforms
  - Hardware Breakpoints
  - Hardware steps
  - Transaction based verification and debug
- Kernel and driver debug via GDB
  - Software Breakpoints
  - Variable probing
- Zynq Linux OS ready to use on QEMU without modifications

Riviera-PRO Verification Platform



accellera SYSTEMS INITIATIVE

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

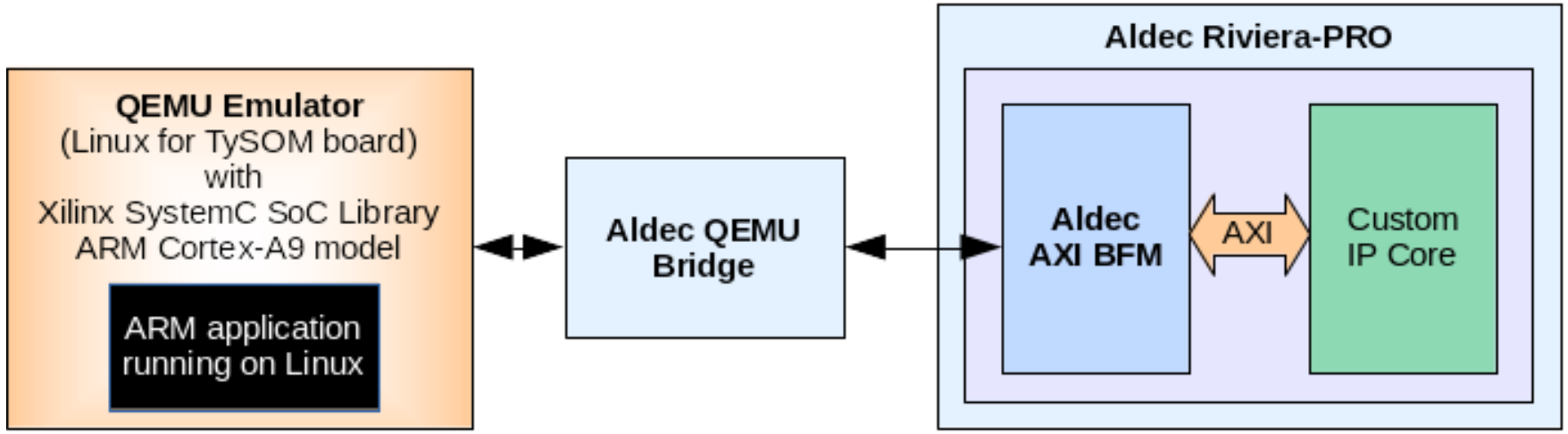
Riviera-PRO Highlights

- High Performance Simulation
  - Extensive simulation optimization algorithms
  - Support for latest Verification Libraries: UVM, OSVVM , UVVM, CocoTB and more
- Advanced Debugging
  - Transaction Level simulation end debug
  - Multi-language debug environment (Verilog, VHDL, SystemVerilog, SystemC, Verilog-AMS)
  - Support for MATLAB and Simulink
  - C/C++ debug environment
  - Support for external C/C++ compilers (GCC, Visual C++)
  - UVM Toolbox, Graph and Class Viewer
  - Code tracing, Waveform, Dataflow, FSM window, Coverage, assertions, memory visualization
  - Comprehensive Assertions-Based Verification (SVA and PSL)
  - Advanced Code and Functional Coverage
  - User-defined test plan linking with coverage database
  - Plot Viewer and Image Viewer

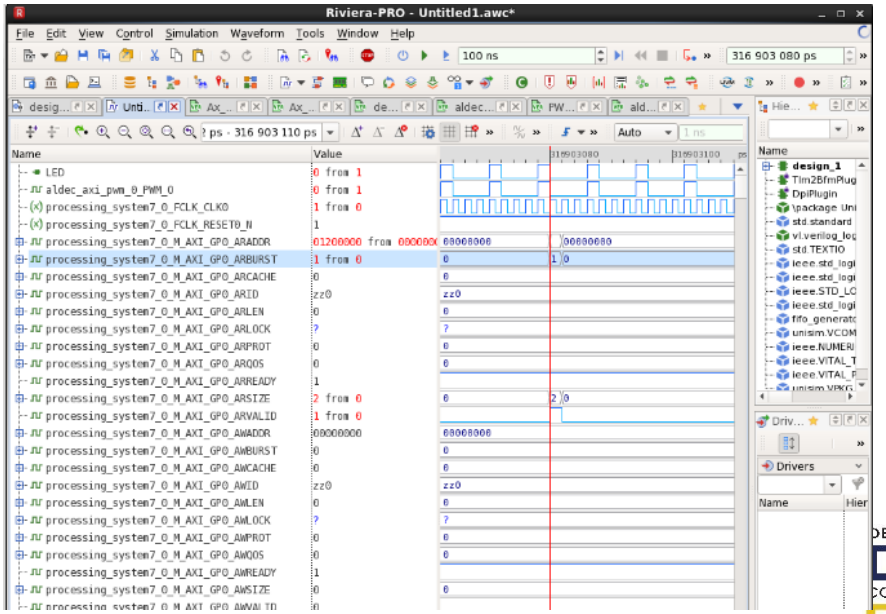
QEMU Highlights

- Free and open-source
- Fast software emulator
- Supports many machines:
  - Xilinx-zynq-a9
- Supports many ARM processors:
  - ARM926
  - ARM946
  - Cortex-A8
  - Cortex-A9
  - Cortex-A15
  - Cortex-A53
  - Cortex-A57

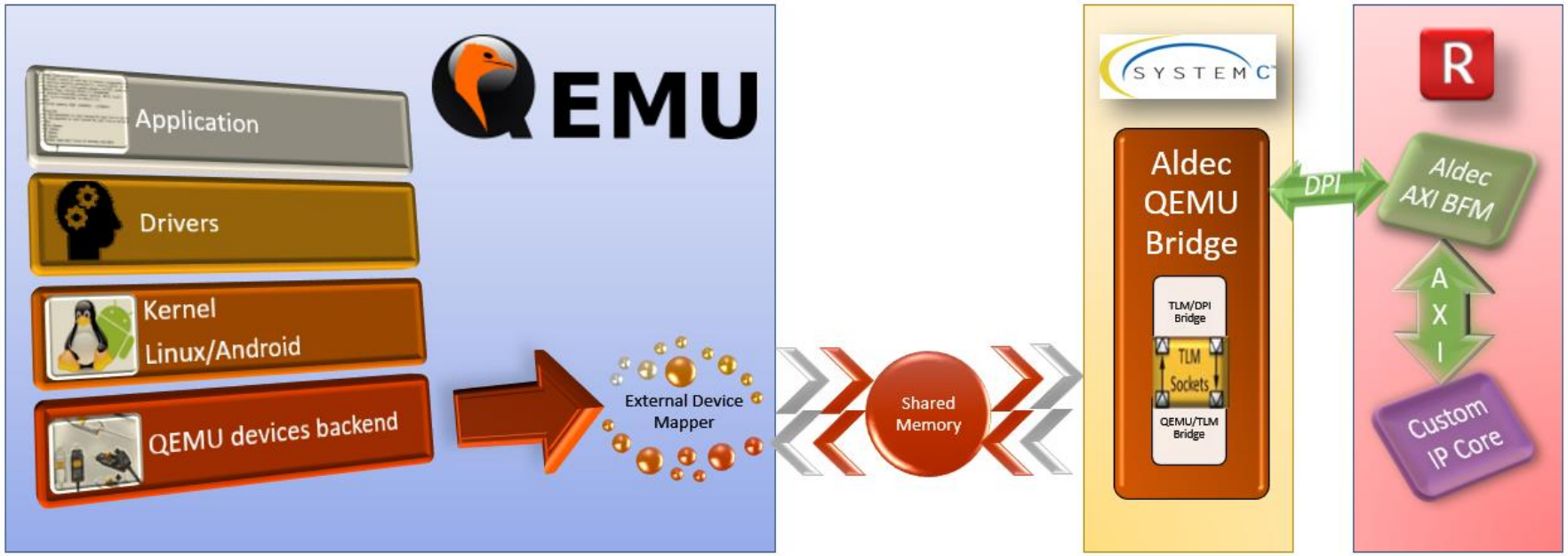
Connecting QEMU with Riviera-PRO

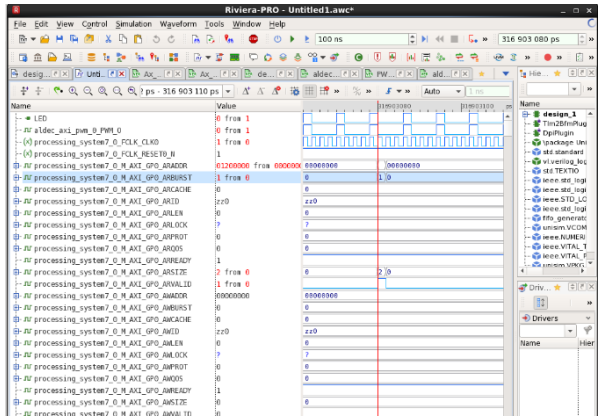



```
++ Starting ftp daemon
++ Starting ssh daemon
random: sshd urandom read with 0 bits of entropy available
rcS Complete
zynq> insmod aldec_pwm.ko
aldec_pwm:init
aldec_pwm: probe
aldec_pwm: probe - misc_register ret = 0
aldec_pwm: probe - of_address_to_resource ret = 0
aldec_pwm: finish with ret = 0
zynq> ./pwm_ioctl.bin
[Warning] pwm_ioctl.bin usage:
        ./pwm_ioctl.bin <duty_cycle> - where duty_cycle is as a

Setting PWM duty cycle of HDL IPcore to 50 %
REMOTEPORT WRITE!!

REMOTEPORT WRITE!!

Reading PWM duty cycle from HDL IPcore
Test passed.
read_duty = 50
```



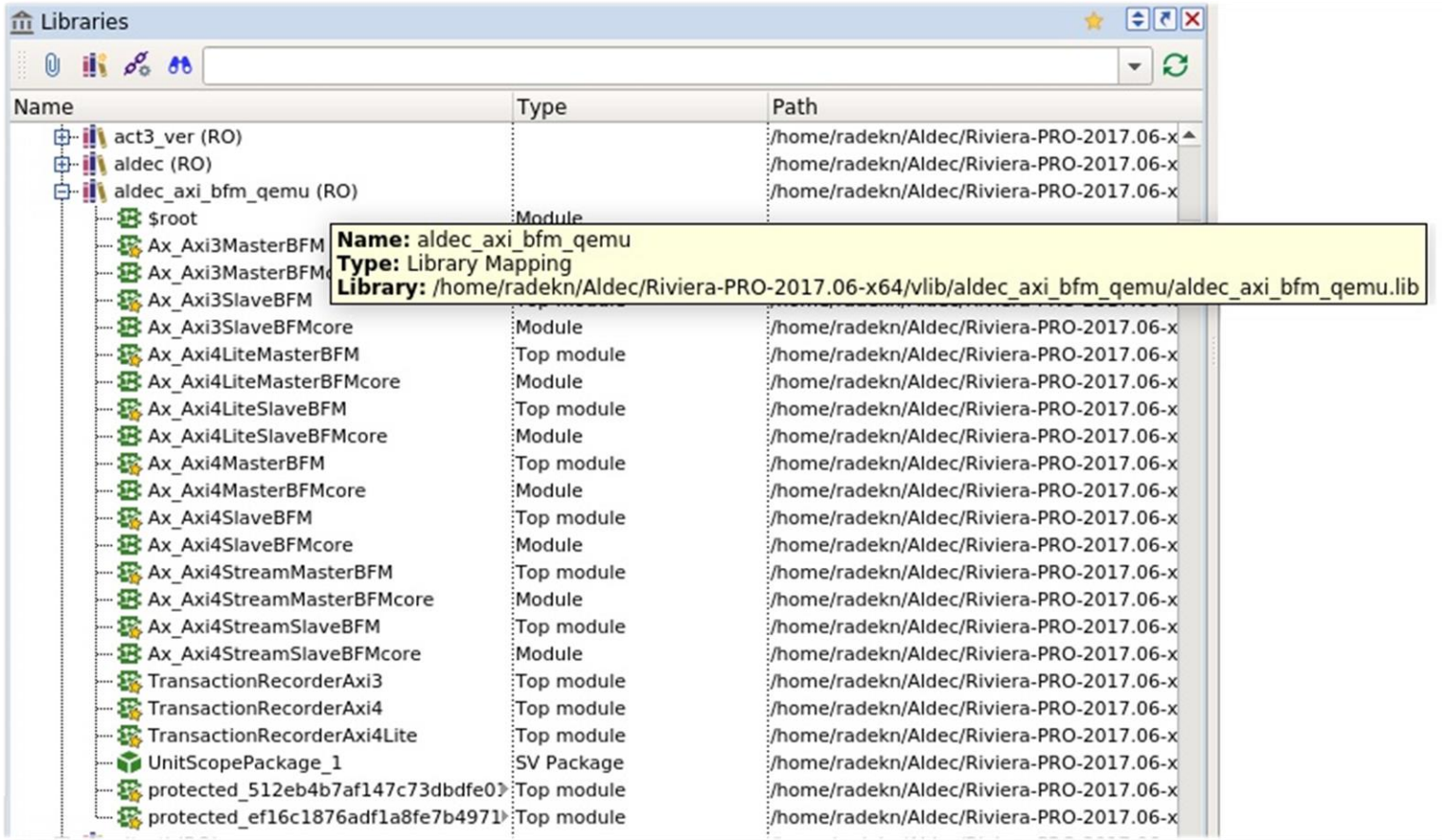
Aldec QEMU Bridge





AXI BFM – HDL site QEMU interface

- AXI 3 Master
- AXI 3 Slave
- AXI 4 Master
- AXI 4 Slave
- AXI 4 Lite Master
- AXI 4 Lite Slave
- AXI 4 Stream Master
- AXI 4 Stream Slave



Aldec AXI BFM usage

Design

- Distributed with precompiled libraries
- Instantiation in HDL part of the project

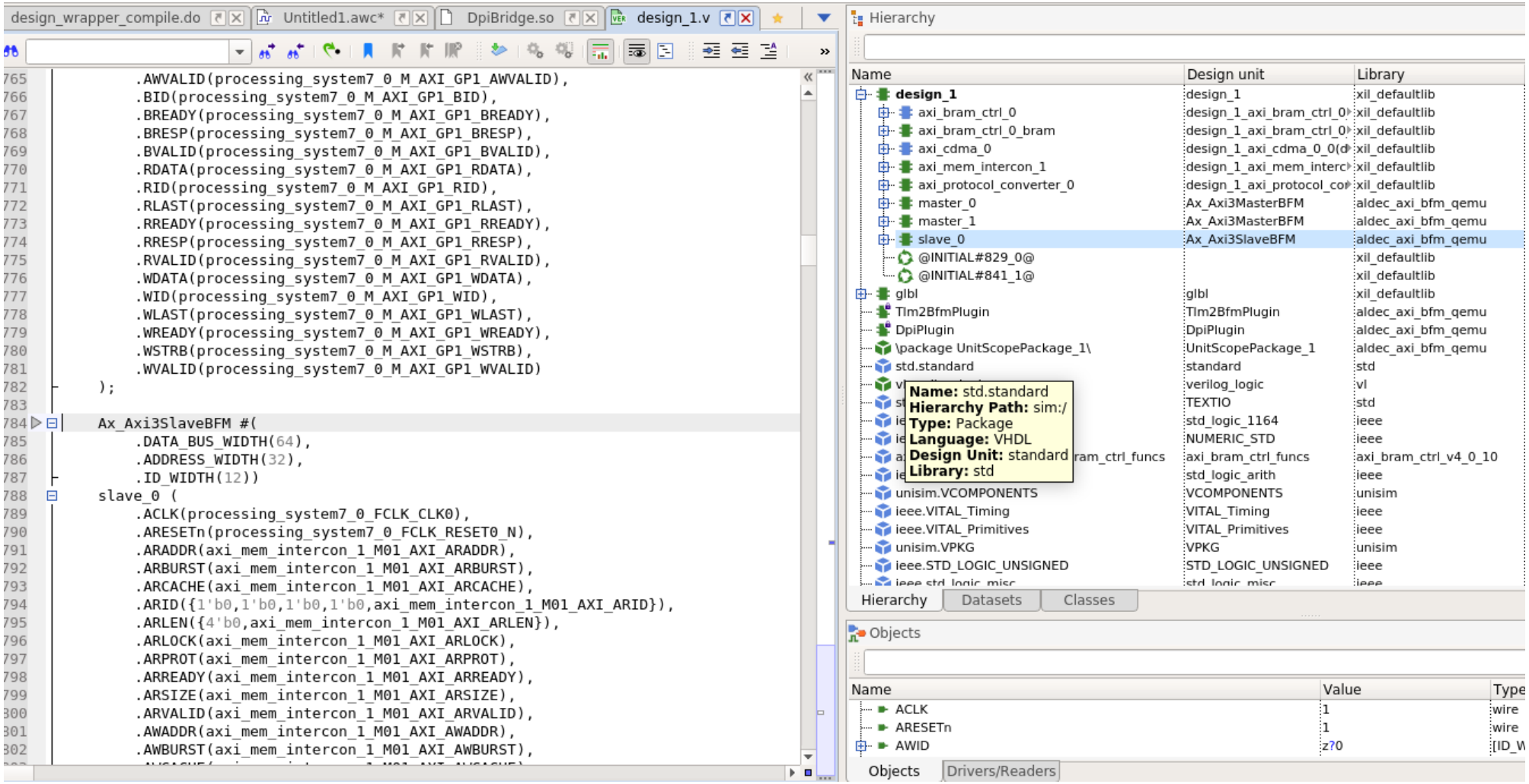
Compilation and Configuration

- Configuration set by parameters
- Turn on build-in transaction recorder

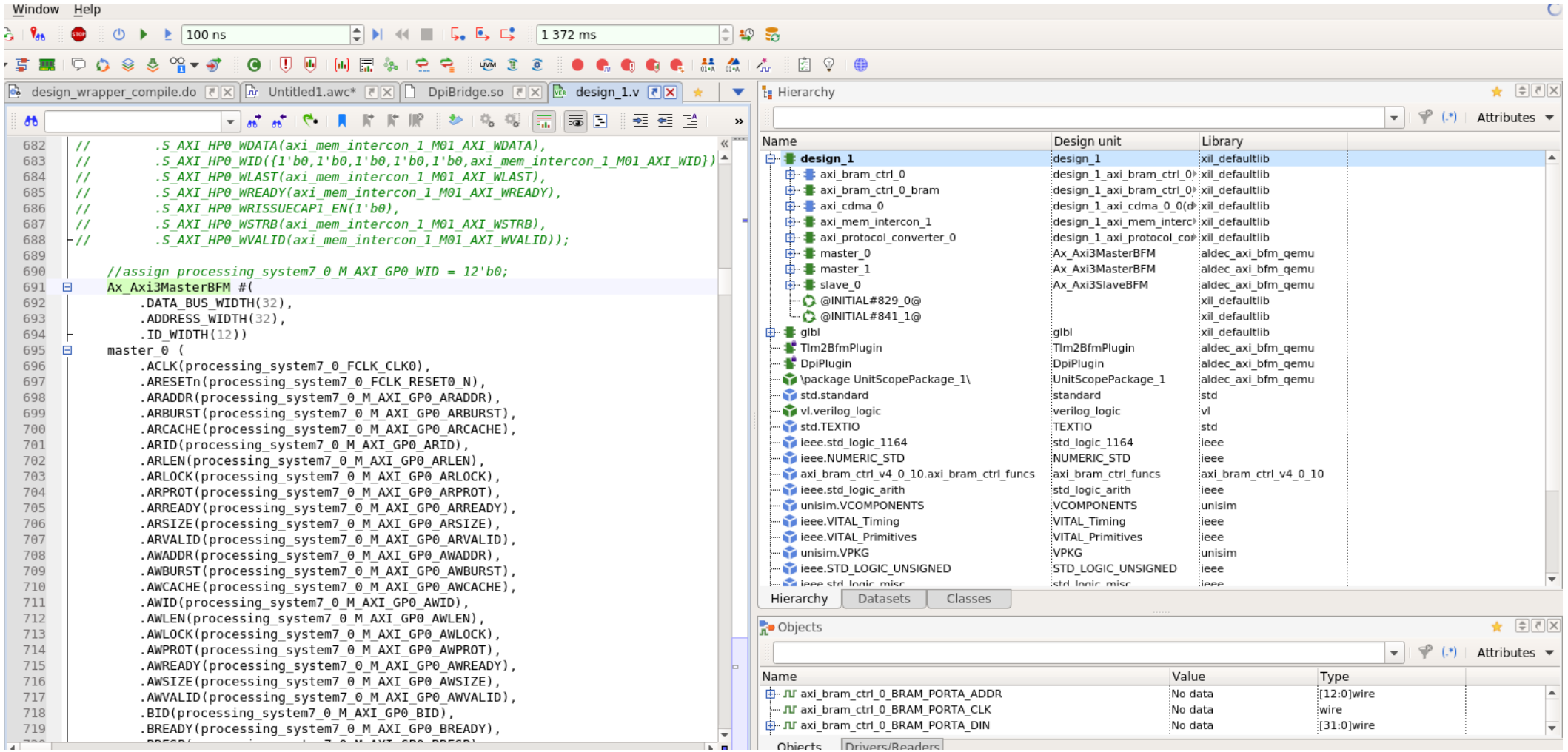
Simulation

- Automatic connection with Aldec QEMU Bridge process done at simulation stage

AXI BFM instance in HDL Code

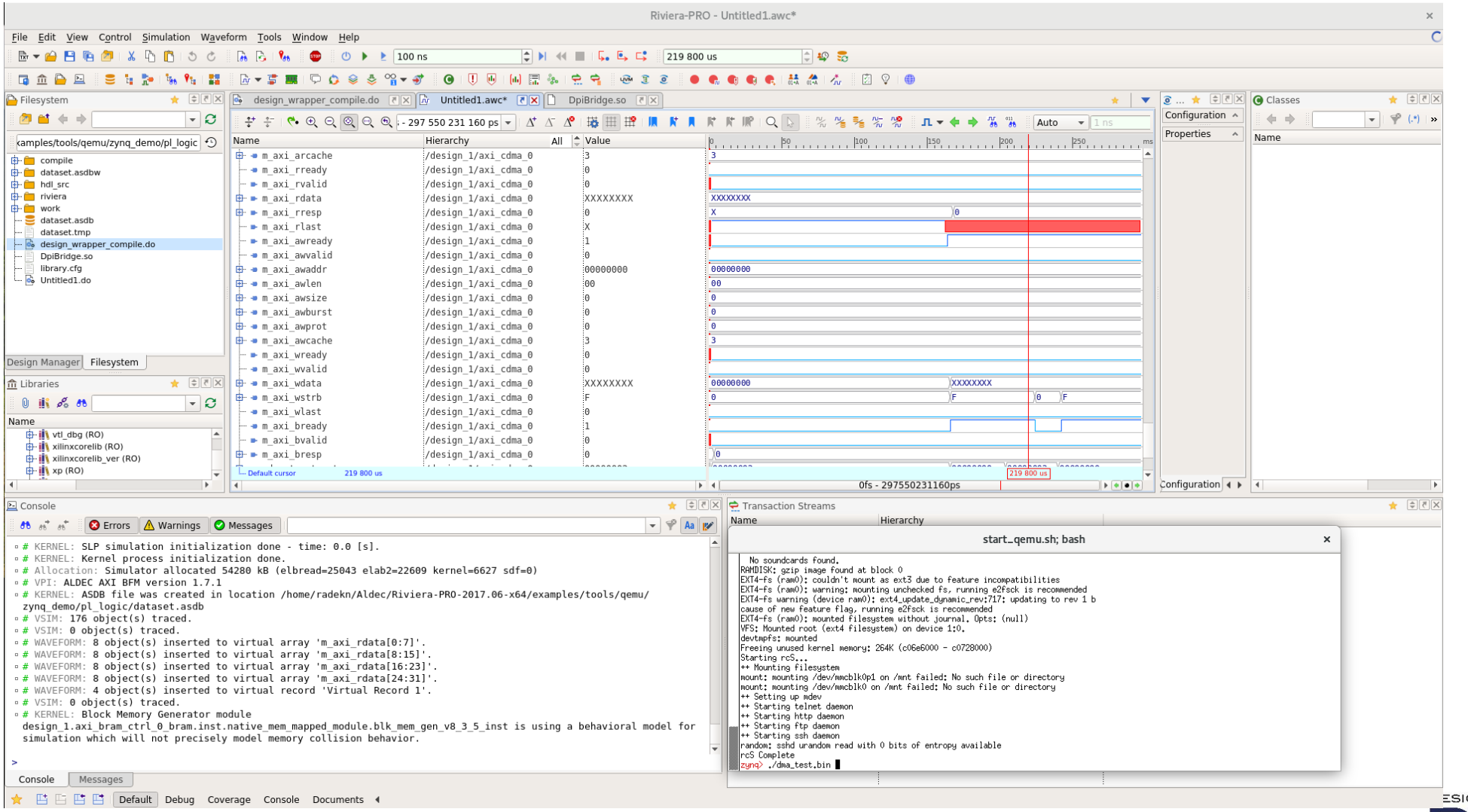


AXI BFM instance in HDL Code

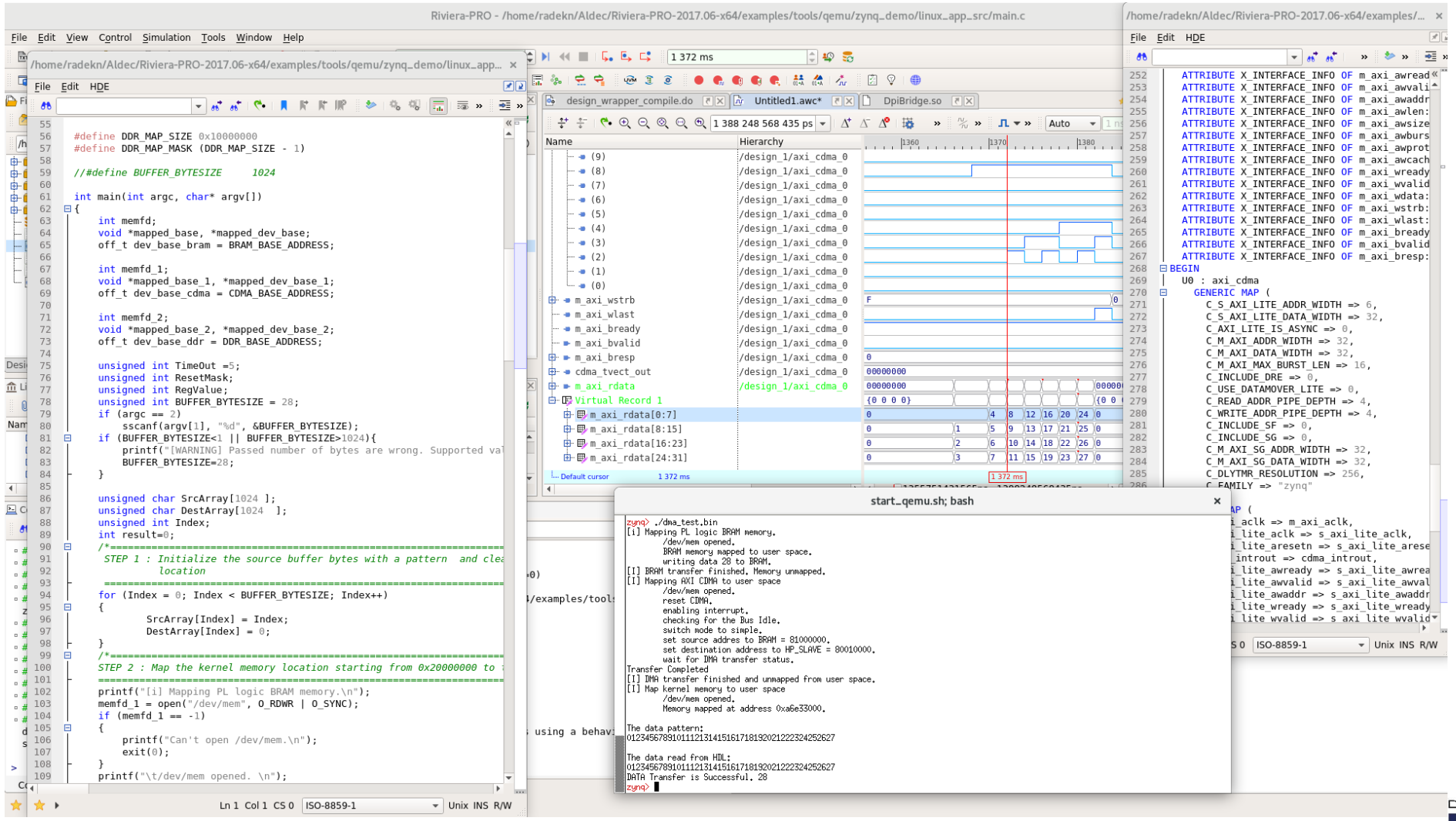



```
//        .S_AXI_HP0_WDATA(axi_mem_intercon_1_M01_AXI_WDATA),
//        .S_AXI_HP0_WID({1'b0,1'b0,1'b0,1'b0,1'b0,axi_mem_intercon_1_M01_AXI_WID}),
//        .S_AXI_HP0_WLAST(axi_mem_intercon_1_M01_AXI_WLAST),
//        .S_AXI_HP0_WREADY(axi_mem_intercon_1_M01_AXI_WREADY),
//        .S_AXI_HP0_WRISSUECAP1_EN(1'b0),
//        .S_AXI_HP0_WSTRB(axi_mem_intercon_1_M01_AXI_WSTRB),
//        .S_AXI_HP0_WVALID(axi_mem_intercon_1_M01_AXI_WVALID));

    //assign processing_system7_0_M_AXI_GP0_WID = 12'b0;
    Ax_Axi3MasterBFM #(
        .DATA_BUS_WIDTH(32),
        .ADDRESS_WIDTH(32),
        .ID_WIDTH(12))
    master_0 (
        .ACLK(processing_system7_0_FCLK_CLK0),
        .ARESETn(processing_system7_0_FCLK_RESET0_N),
        .ARADDR(processing_system7_0_M_AXI_GP0_ARADDR),
        .ARBURST(processing_system7_0_M_AXI_GP0_ARBURST),
        .ARCACHE(processing_system7_0_M_AXI_GP0_ARCACHE),
        .ARID(processing_system7_0_M_AXI_GP0_ARID),
        .ARLEN(processing_system7_0_M_AXI_GP0_ARLEN),
        .ARLOCK(processing_system7_0_M_AXI_GP0_ARLOCK),
        .ARPROT(processing_system7_0_M_AXI_GP0_ARPROT),
        .ARREADY(processing_system7_0_M_AXI_GP0_ARREADY),
        .ARSIZE(processing_system7_0_M_AXI_GP0_ARSIZE),
        .ARVALID(processing_system7_0_M_AXI_GP0_ARVALID),
        .AWADDR(processing_system7_0_M_AXI_GP0_AWADDR),
        .AWBURST(processing_system7_0_M_AXI_GP0_AWBURST),
        .AWCACHE(processing_system7_0_M_AXI_GP0_AWCACHE),
        .AWID(processing_system7_0_M_AXI_GP0_AWID),
        .AWLEN(processing_system7_0_M_AXI_GP0_AWLEN),
        .AWLOCK(processing_system7_0_M_AXI_GP0_AWLOCK),
        .AWPROT(processing_system7_0_M_AXI_GP0_AWPROT),
        .AWREADY(processing_system7_0_M_AXI_GP0_AWREADY),
        .AWSIZE(processing_system7_0_M_AXI_GP0_AWSIZE),
        .AWVALID(processing_system7_0_M_AXI_GP0_AWVALID),
        .BID(processing_system7_0_M_AXI_GP0_BID),
        .BREADY(processing_system7_0_M_AXI_GP0_BREADY),
```

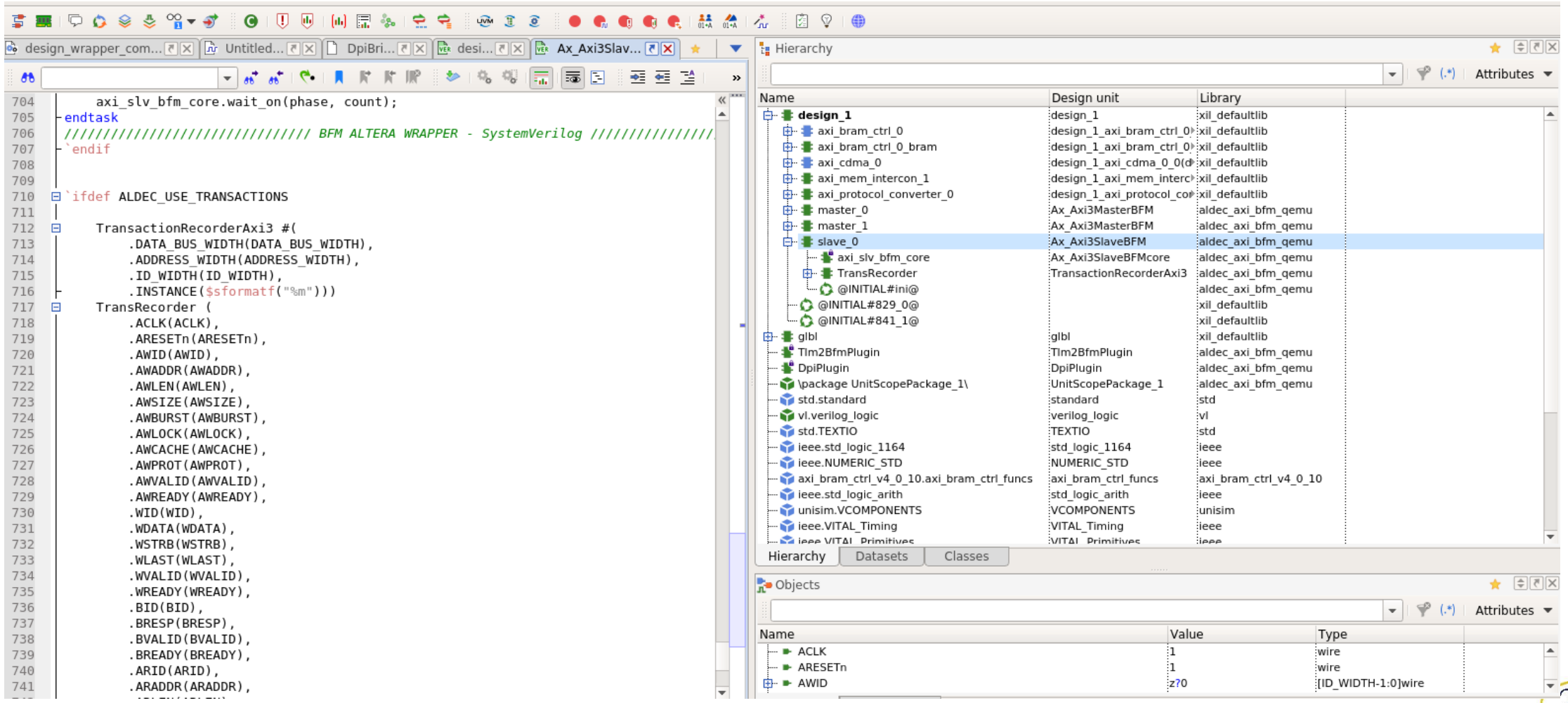
Riviera-PRO QEMU environment



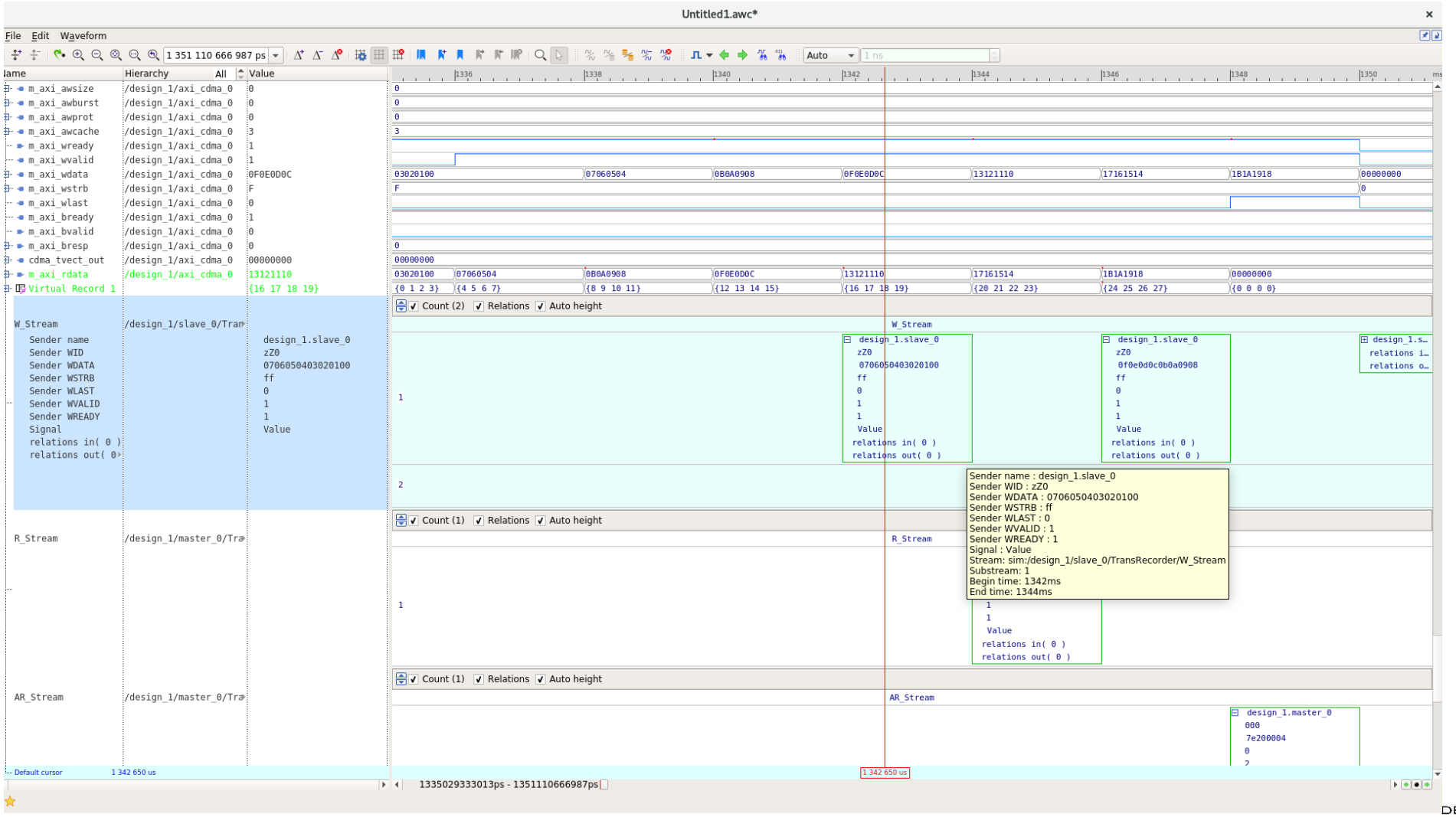
Hardware and Software co-sim



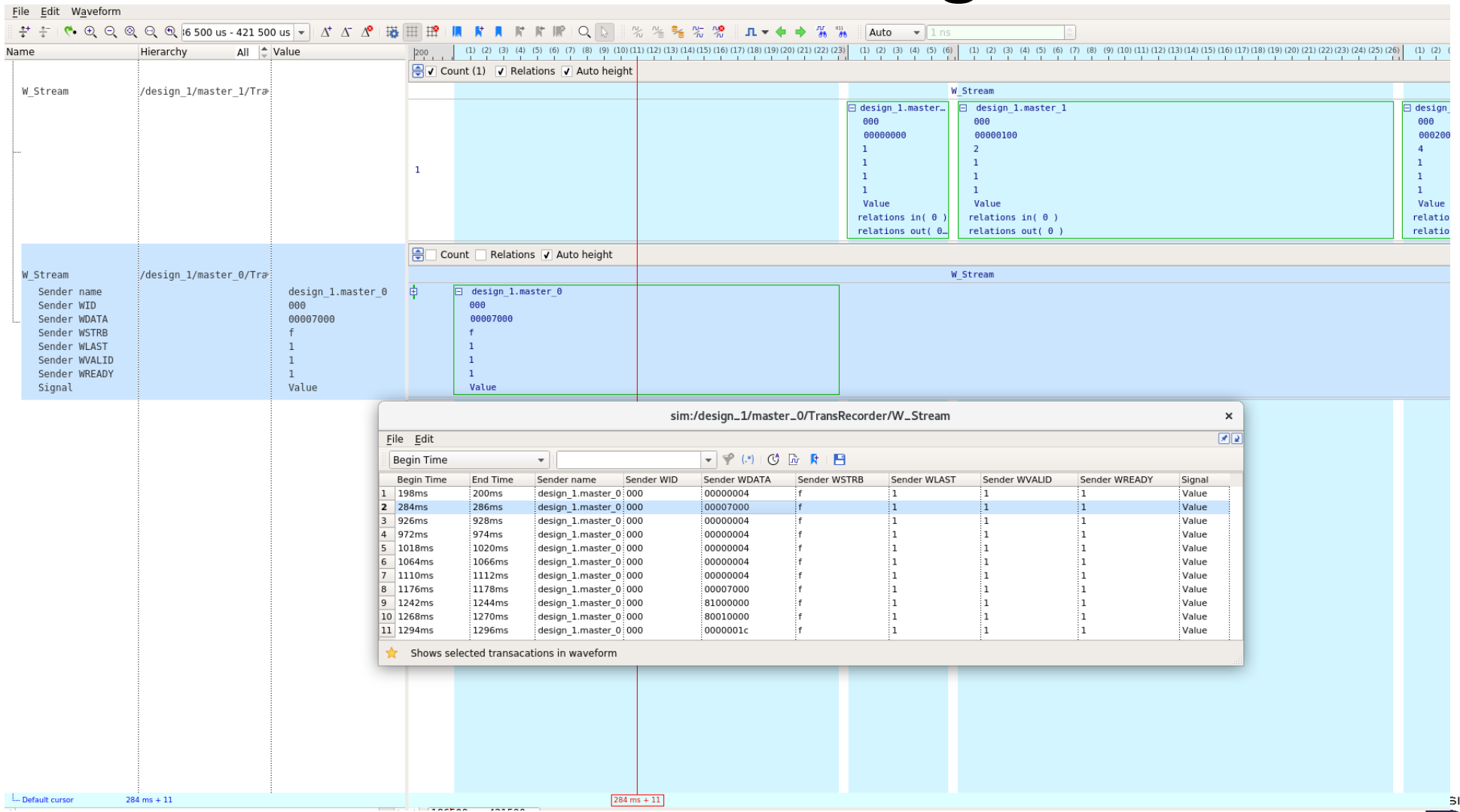
Built-in Transaction Recorder



Transaction based debug



Transaction logs



Summary

- SoC FPGA Verification Platform
- Processor modeling based on QEMU emulator (instead of simplified BFM model)
- Hardware and Software debugging
- Fast RTL IP Core Verification and bugs fixing
- Driver and Firmware Verification
- Built-in Transaction Recorder
- Support for many platforms and processors
- Support for different Linux versions
  - Yocto project

QEMU co-emulation with HES-DVM

Emulation Pros & Cons

(+) Fast – 10+ MHz speed

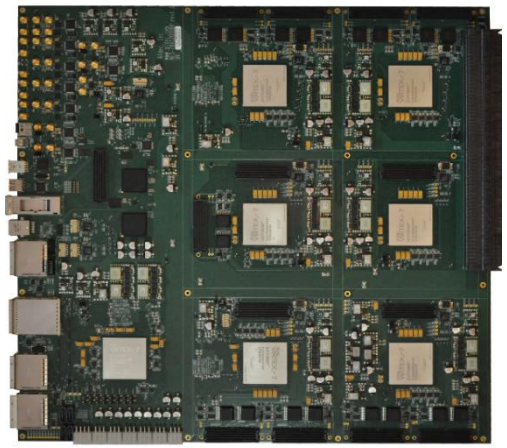
(+) Cycle Accurate hardware model

(+) Advanced Hardware Debugging

(+) Enables testing with real devices/live data

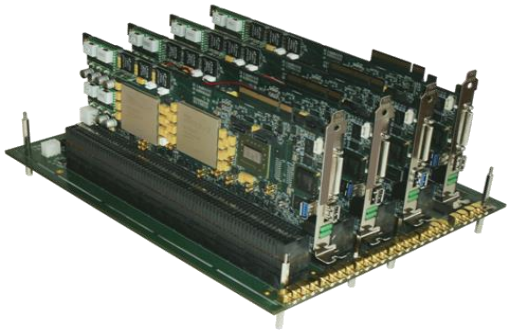
(-) Emulation available later than RTL Simulation

(-) Usually limited emulation resources



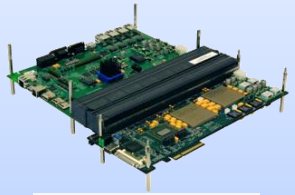
HES

Hardware Emulation Solutions



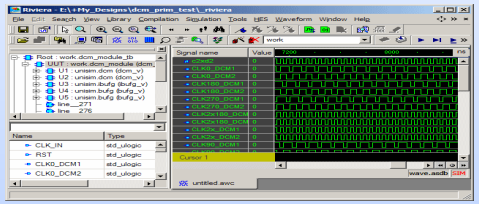
HES-DVM™ Versatile Emulation Platform

In Circuit Emulation



Speed Adapter

Simulation Acceleration



UVM OVM VHDL SystemVerilog SYSTEMC

SCE-MI PLI/VHPI

Hybrid Co-emulation

Virtual Machine

QEMU

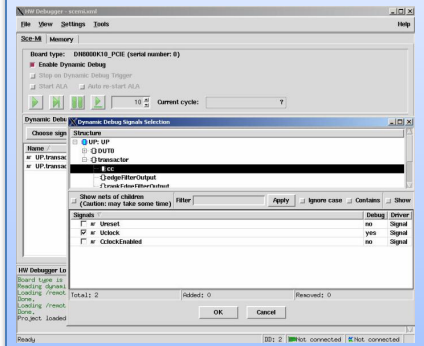
Virtual Platforms

ARM Fast Models Virtual Platforms

SYSTEMC OVP Open Virtual Platforms

SCE-MI TLM

Hardware Debugging



TCP-IP

Break

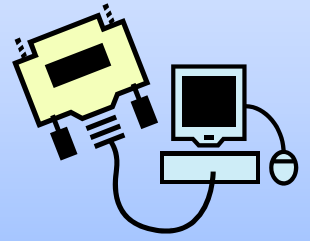
Capture

Run

SoC Integration Software Debugging

JTAG

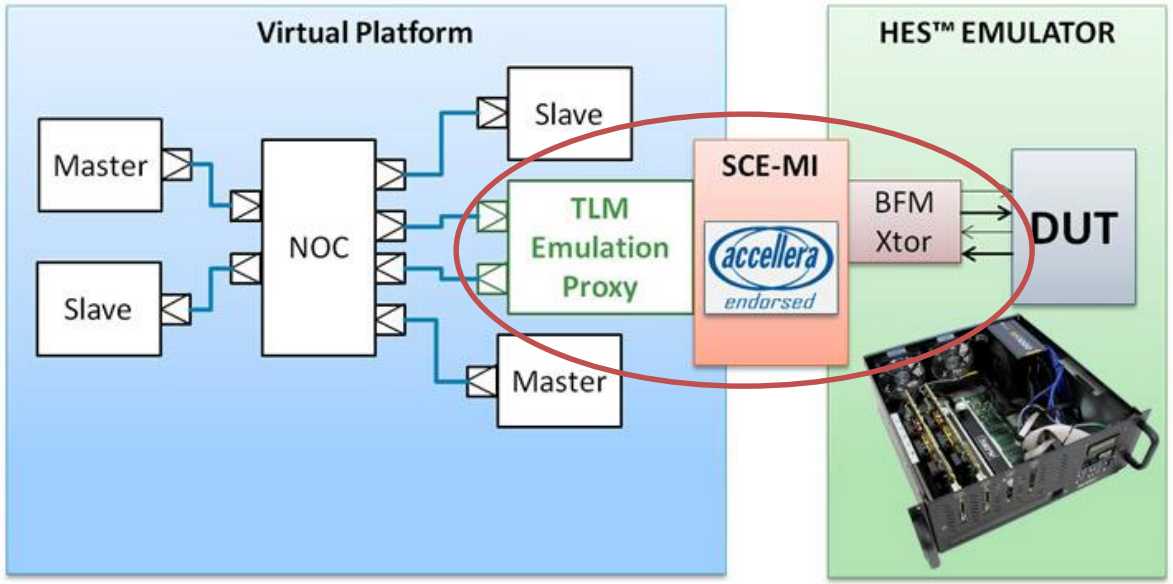
- JTAG Port
- vJTAG Xtor



Connecting Virtual Platforms

SoC Bus Transactors (BFM with SCE-MI)

- **AMBA** – Advanced Microcontroller Bus Architecture
  - AXI Master & Slave
  - AHB Master & Slave
  - APB Master & Slave
- **OCP** – Open Core Protocol
  - Master & Slave



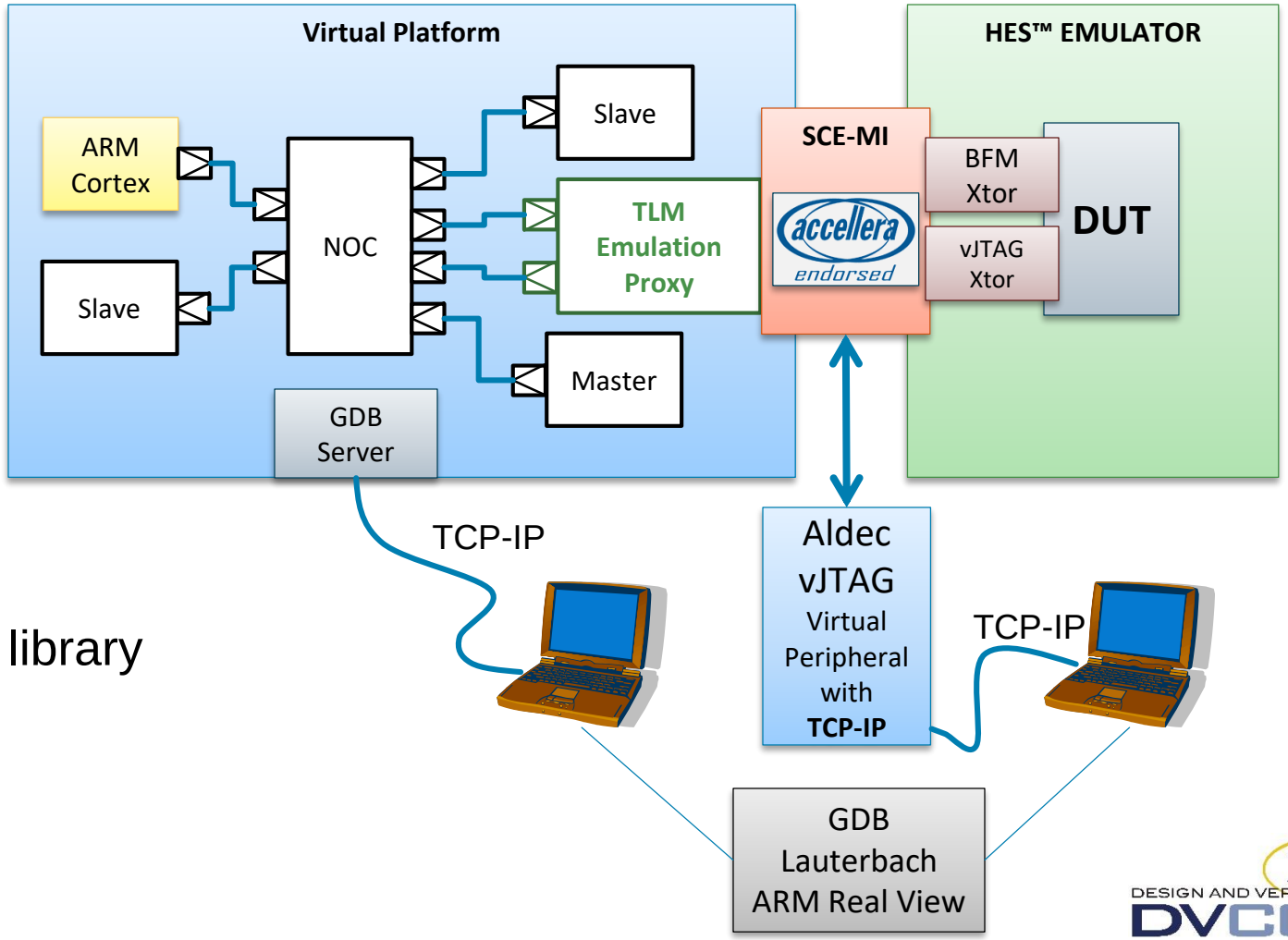
Virtual Platform Plug & Play

- **HES PCIe host driver** (Linux/Windows)
- **SystemC/TLM Proxy** for each bus transactor
- **SCE-MI** – Accellera Standard Co-Emulation Modeling Interface

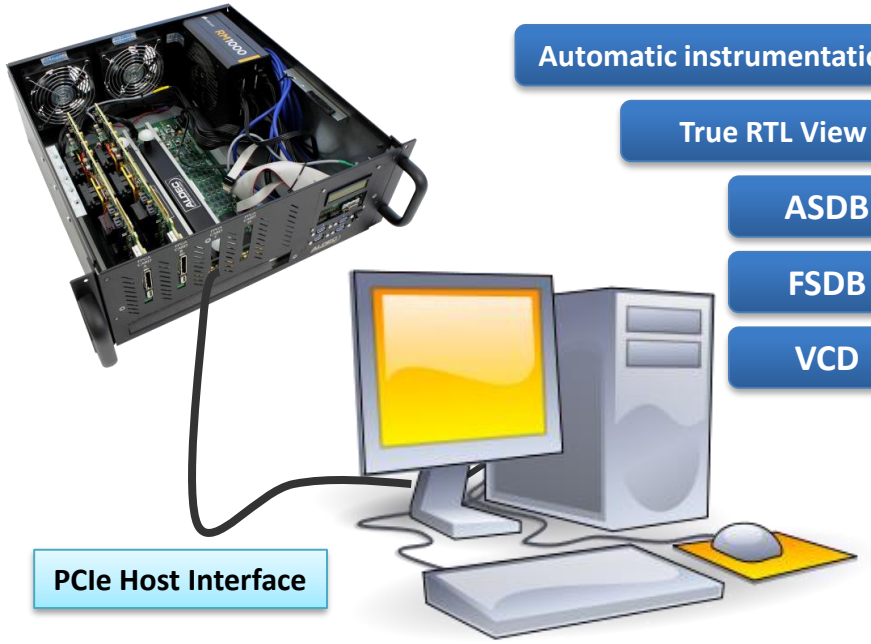
Software debugging

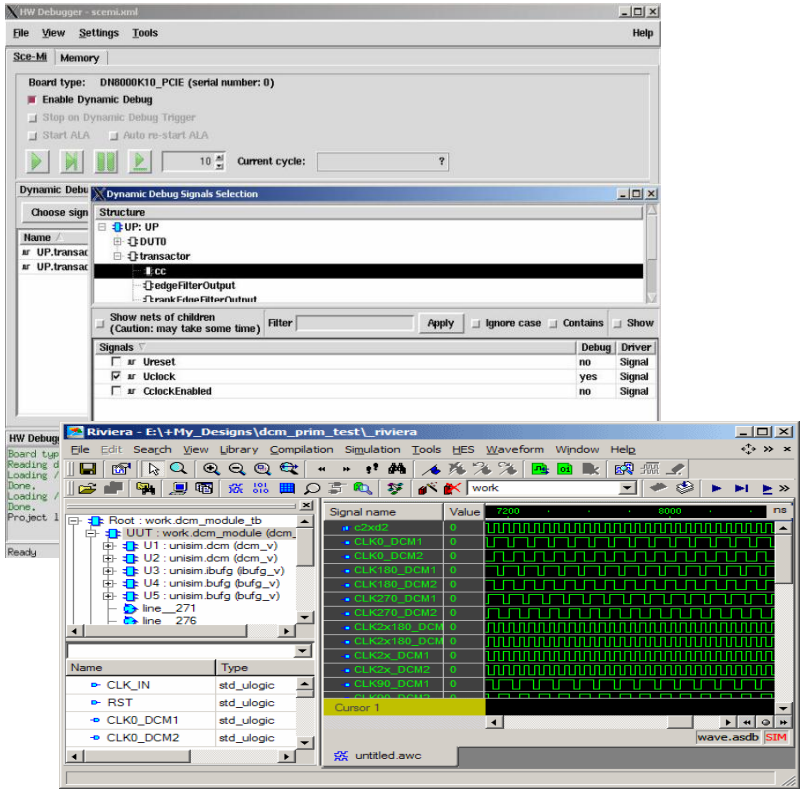
Where is the CPU core?

- **In Virtual Platform**
  - VP provides interface
- **In HES Emulator**
  - Aldec provides solution:
    - vJTAG Transactor
    - vJTAG Virtual Peripheral library with TCP-IP



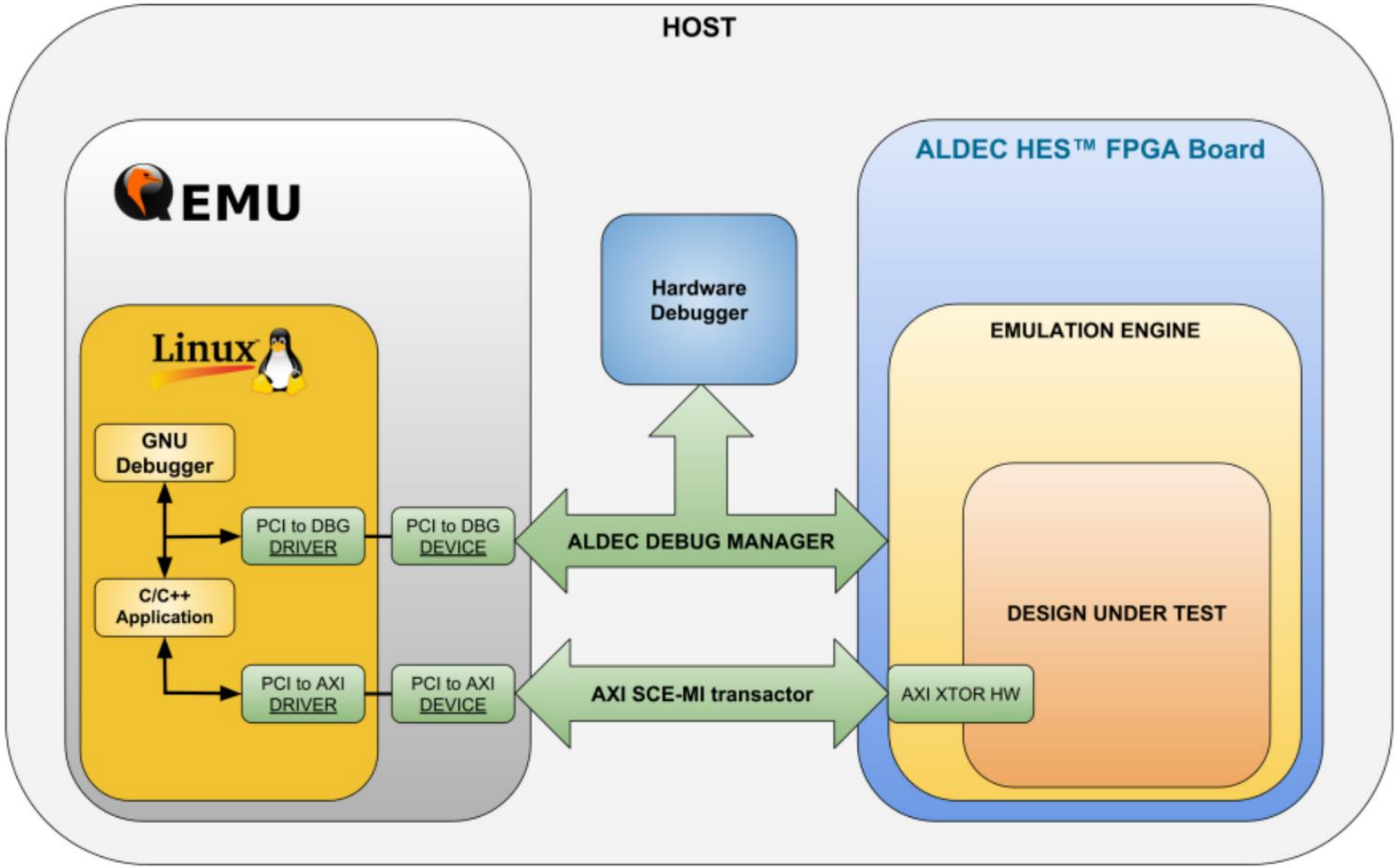
Hardware debugging



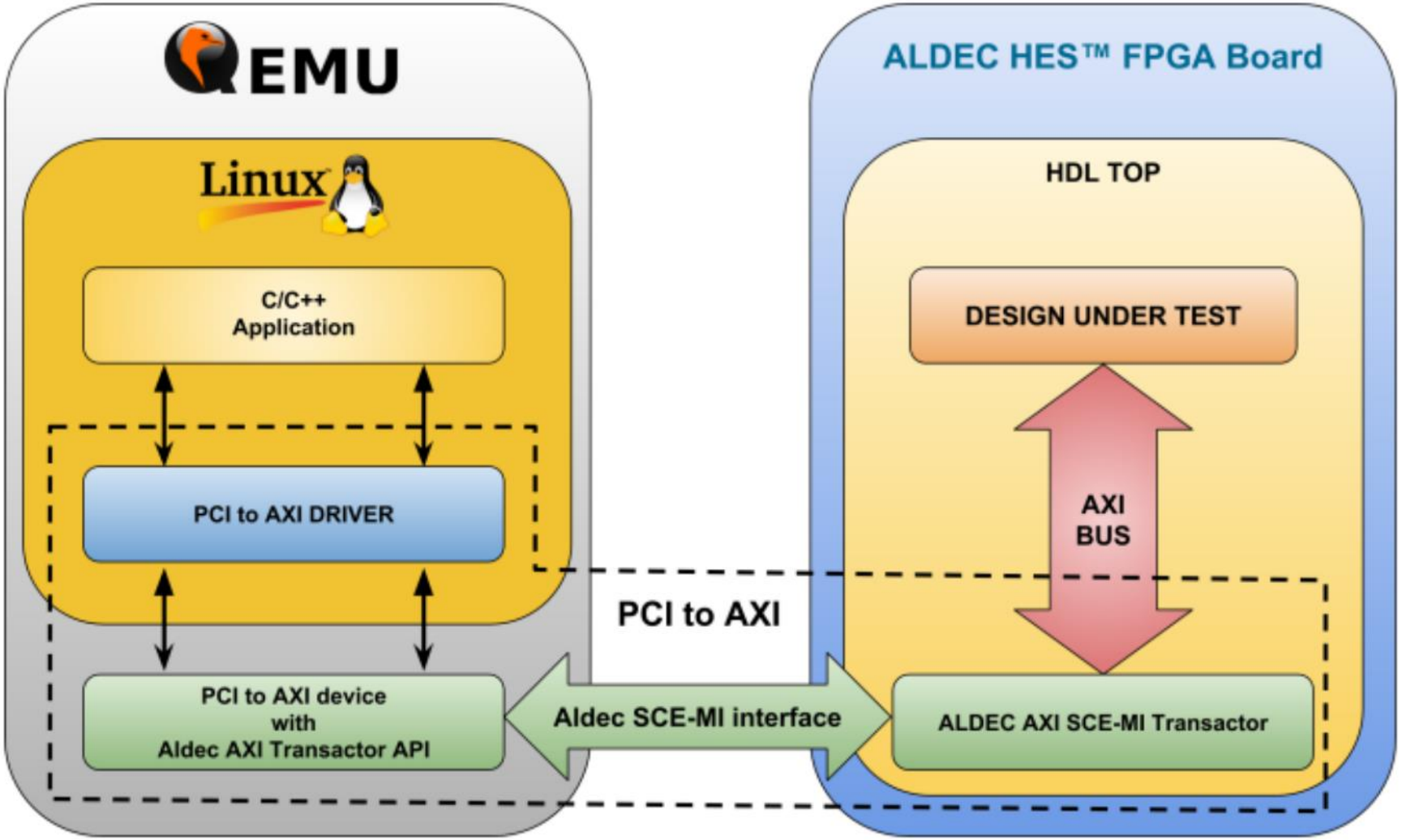


- Debug probes – 100% visibility
- Breakpoints and Triggers
- Clock control
- True RTL debugging
- Memory visibility

QEMU Co-Emulation



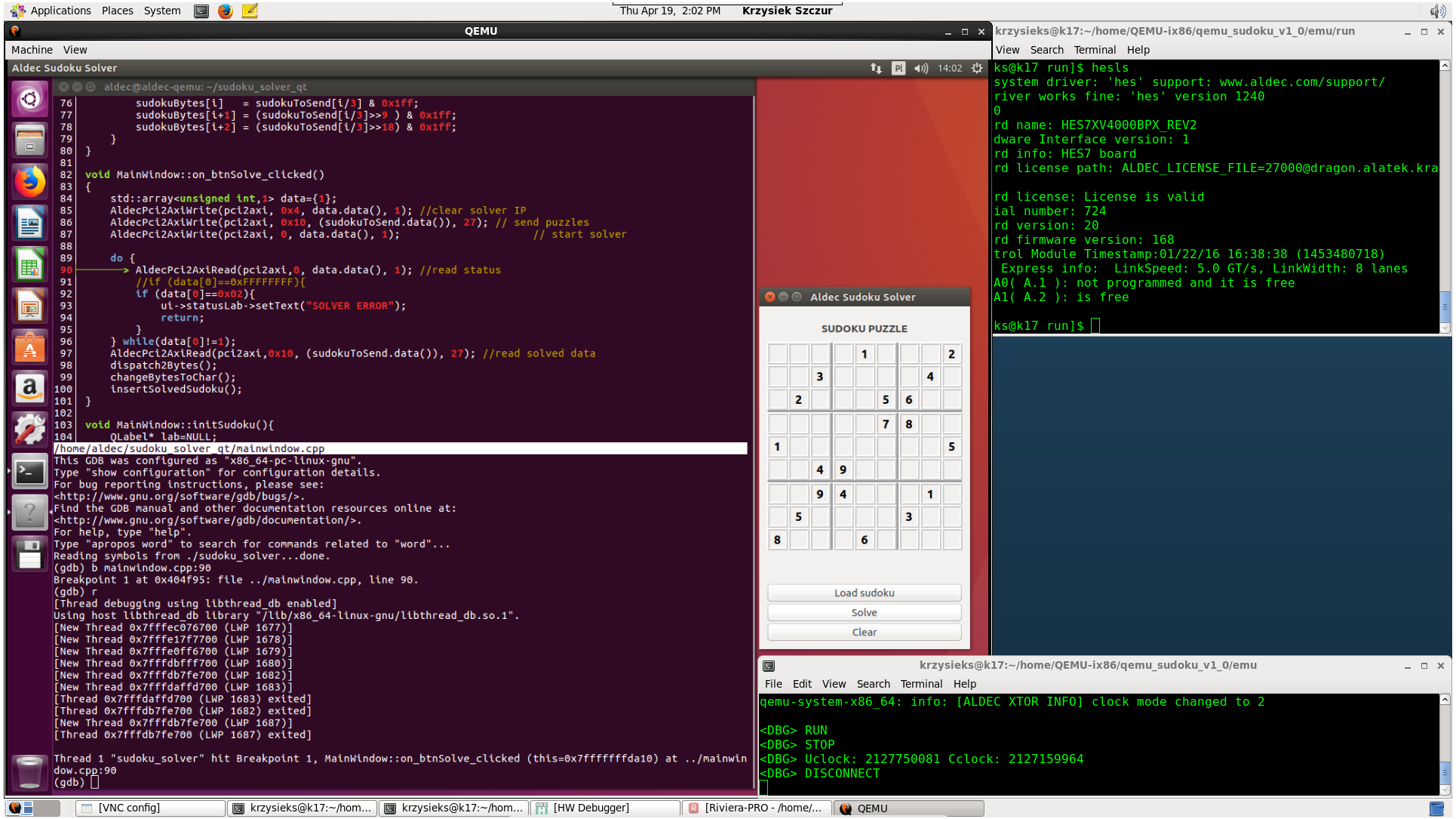
QEMU PCI to AXI bridge



QEMU Co-Emulation Highlights

- Synchronized Hardware/Software Debug
  - ***Force Break*** – stop hardware clocks on software breakpoint
  - Software debugger integration (GDB), custom commands
- AMBA AXI interface support
  - Aldec AXI Master transactor VIP
- PCI to AXI Bridge
- QEMU virtual machine architectures with PCI device tree (ARM, x86 and others)
- Integrated with Linux Guest OS in QEMU
- Automated design compilation in HES-DVM platform

QEMU Co-Emulation Live Demo



Questions