Software Driven Test of FPGA Prototype

Methods & Use cases

Krzysztof Szczur, Aldec Inc.

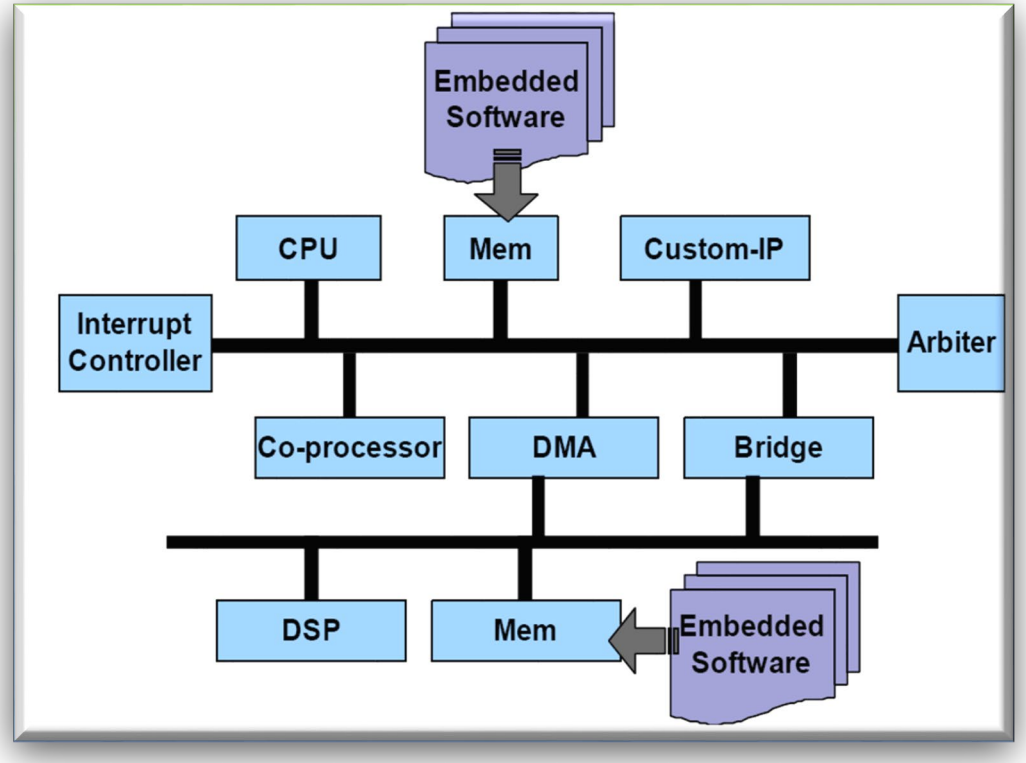
Radosław Nawrot, Aldec Inc.

accellera SYSTEMS INITIATIVE

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Motivation - SoC verification challenge

- SoC = Hardware + Software
  - both need verification
  - software on critical-path (schedule risk)
- Complex architecture
  - Hierarchical Network-on-Chip (NoC)
    - Corner case hardware/software bugs
  - Re-usable IPs: CPU, MEM, DMA, …
    - RTL code not available for FPGA prototyping
  - Custom IPs: Co-processors, Function accelerators
    - Often need late hardware debugging

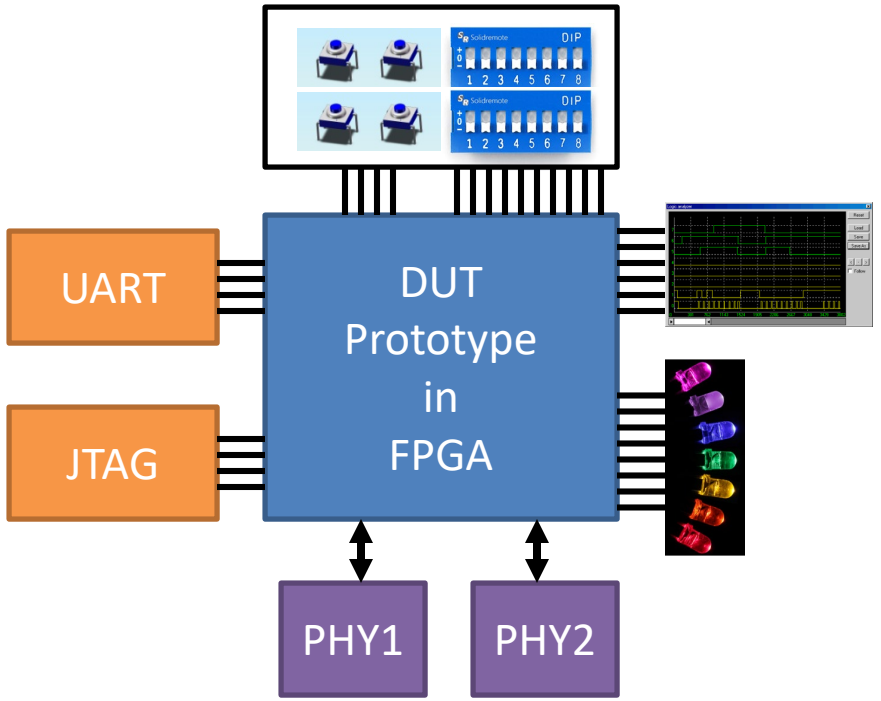


Tutorial Agenda

- Software driven v.s. traditional
- FPGA Technology enabling software driven verification
- Embedded software driven testbench – application examples with HES-US-440
  - Using HES Proto-AXI host interface
  - Bare metal approach
- Software driven testbench in simulation
  - QEMU co-simulation with Riviera-PRO

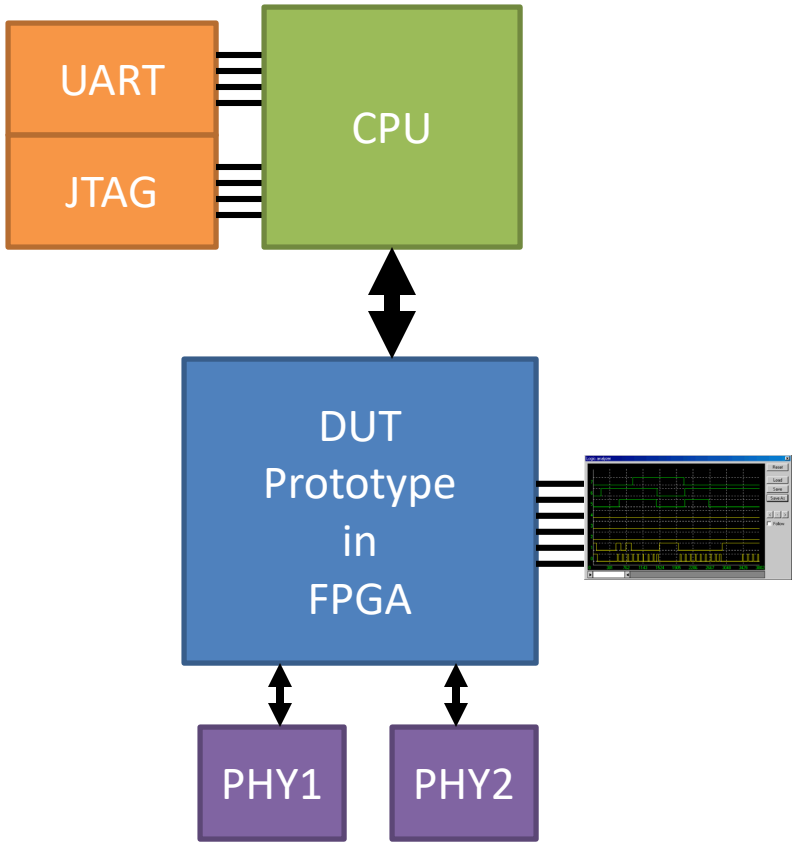
Traditional Prototyping Bench

- Controllability & Configuration
  - Dip switches
  - Push buttons
  - UART or JTAG ports
- Status & Debug
  - LED & Panels
  - Logic analyzers
- Live data streams
  - PHY components
  - Peripheral devices



Software Driven Testbench (SDT)

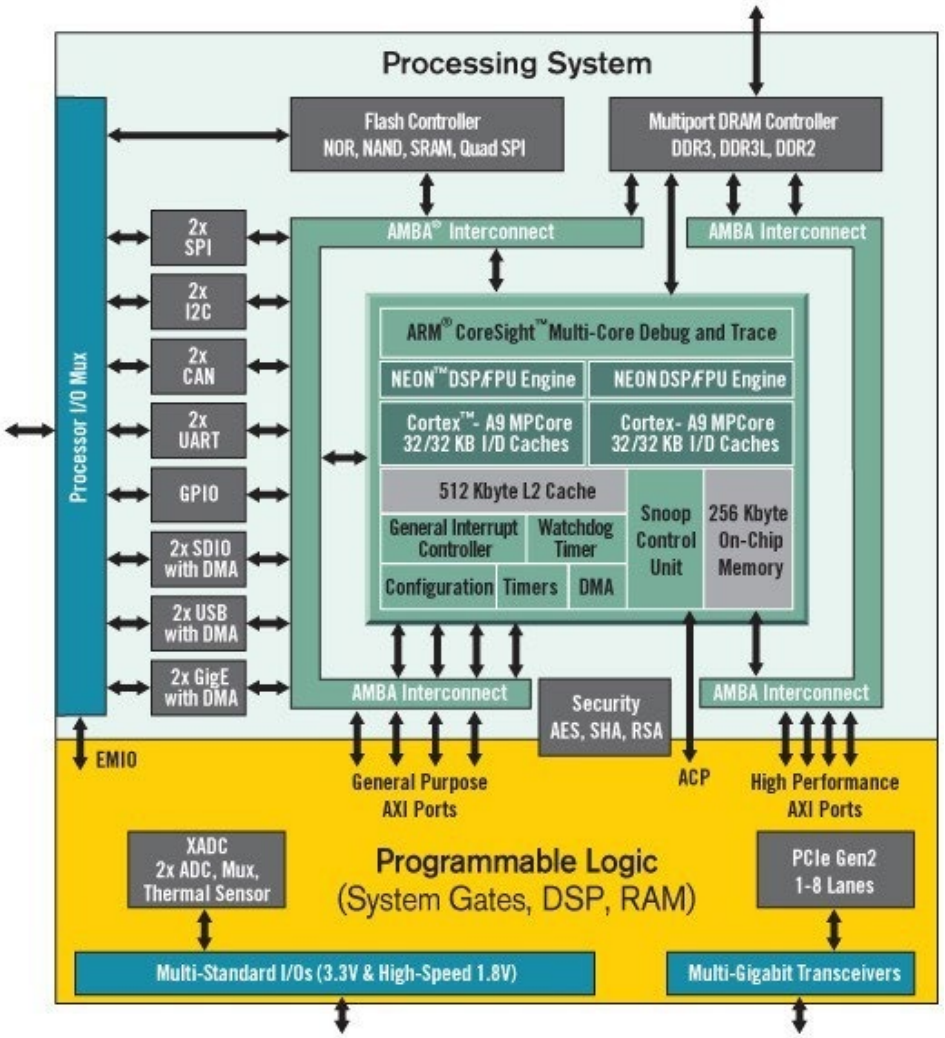
- Controllability & Configuration
  - Use CPU
- Status & Debug
  - Use CPU
  - Logic analyzers (external & on-chip)
- Live data streams
  - PHY components
  - Peripheral devices



FPGA Technology Enabling SDT

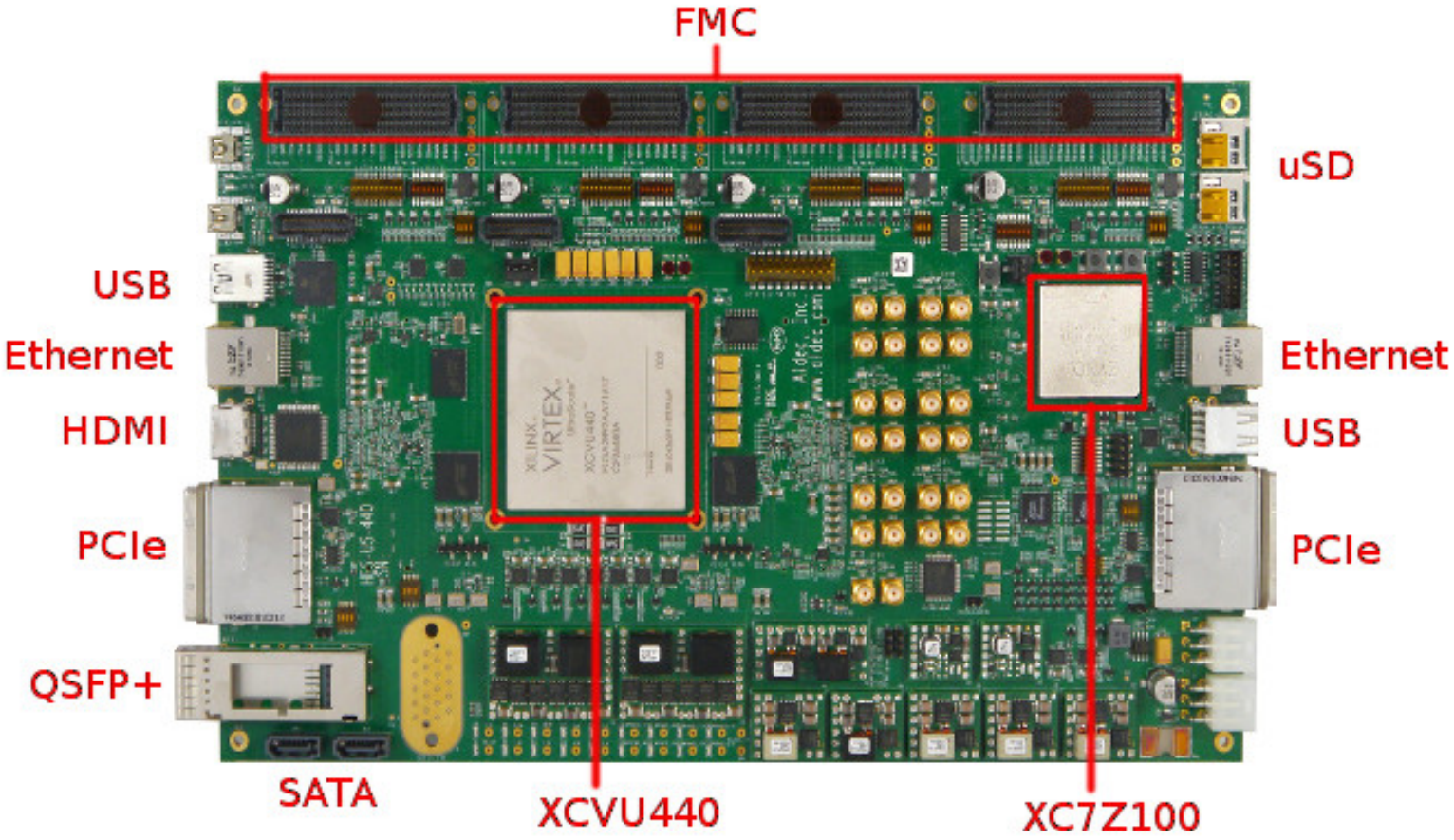
Xilinx Zynq and the like…

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



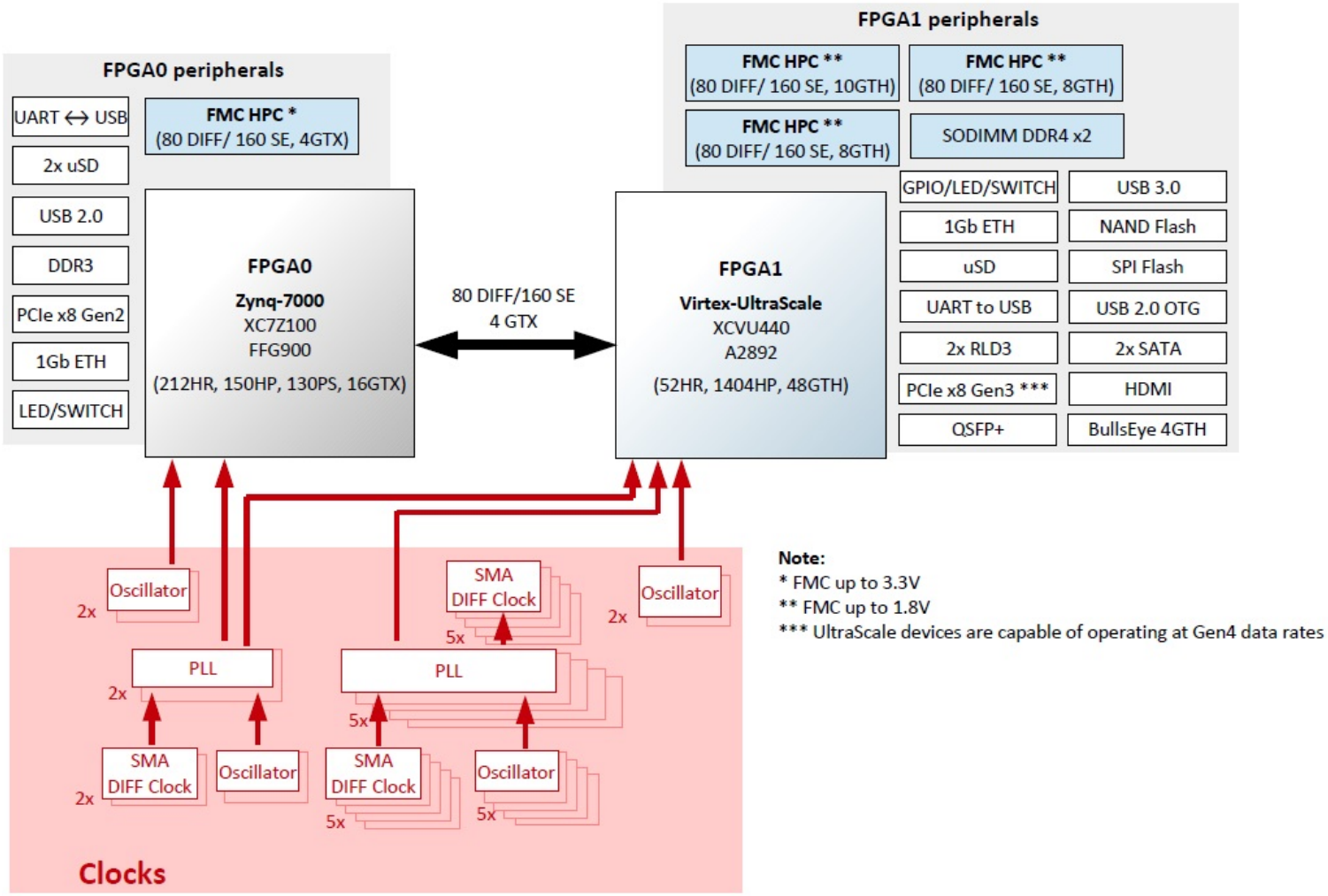
Aldec HES-US-440

- Main FPGA:
  - Virtex UltraScale™ XCVU44
  - 26 Million ASIC Gates
- Host FPGA
  - Zynq-7000, XC7Z100
  - CPU: ARM Cortex A9
  - FPGA: Kintex-7



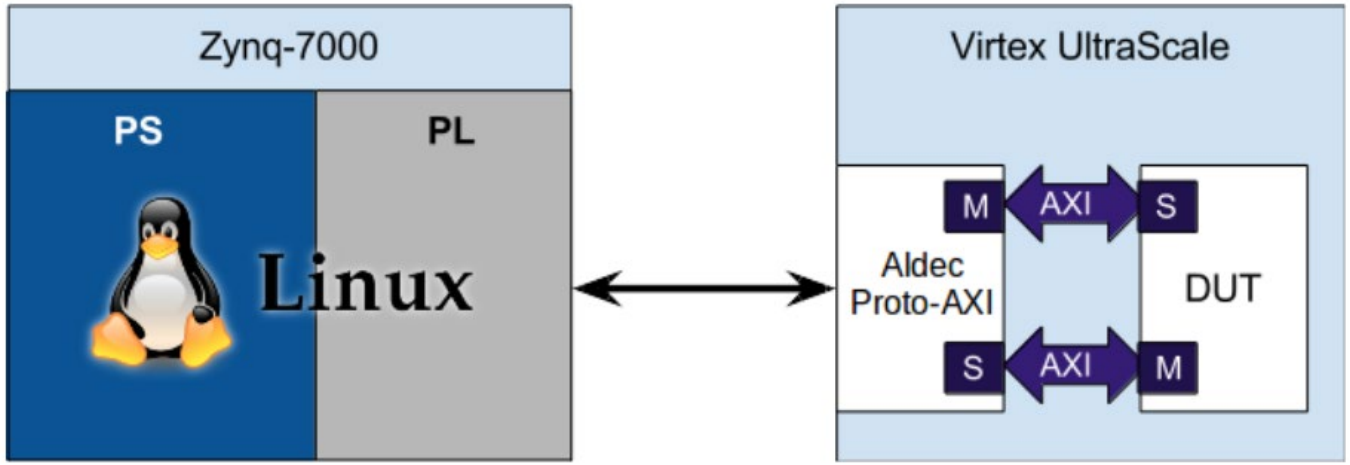
HES-US-440 Diagram

- **Memory**
  - DDR4 (64 GB) : 2 x SO-DIMM
  - RLDRAM-3 (1152 Mb)
  - NAND Flash, SPI Flash
  - 2x MircoSD card slots
- **I/O Connectivity**
  - Inter-FPGA:
    - 80 DIFF / 160 SE, 4 GTH
  - FMC connectors
    - 320 DIFF / 640 SE, 26 GTH
- **Peripherals**
  - PCIe x8 gen3 & gen2
  - USB 3.0 & 2.0 & OTG, SATA
  - Ethernet 1Gb, QSFP+ 40Gb
  - BullsEye
- **Clocks**
  - 7x programmable clock modules

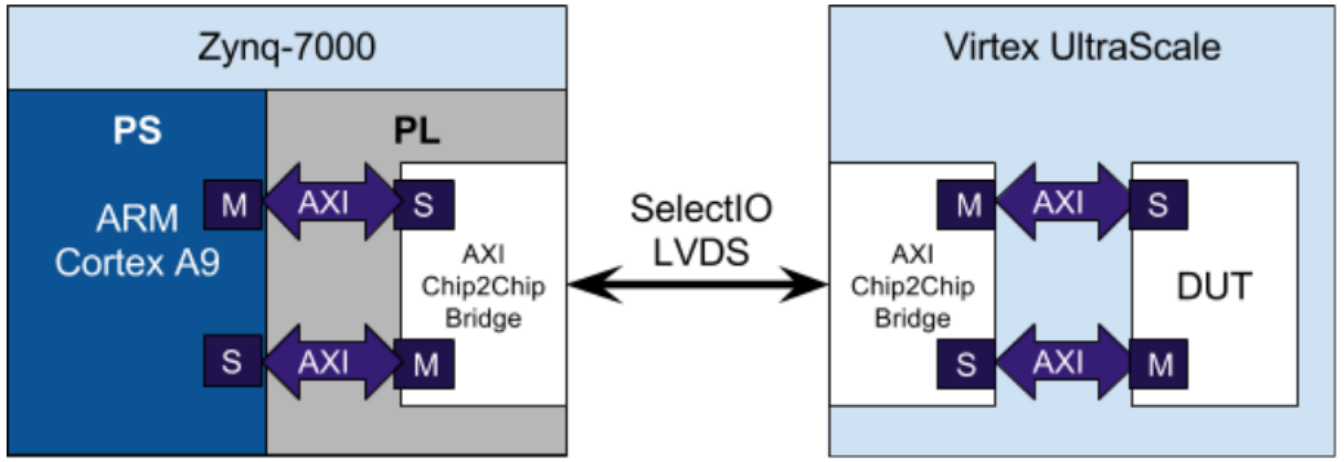


SDT application examples on HES-US-440

- **Case-1:** Aldec HES-ProtoAXI
  - Zynq image ready to use (PL & PS)
  - Use Aldec Proto-AXI IP
  - Ready Embedded Linux env.
  - Simple Proto-AXI C/C++ API
  - Spend all your time to develop SDT

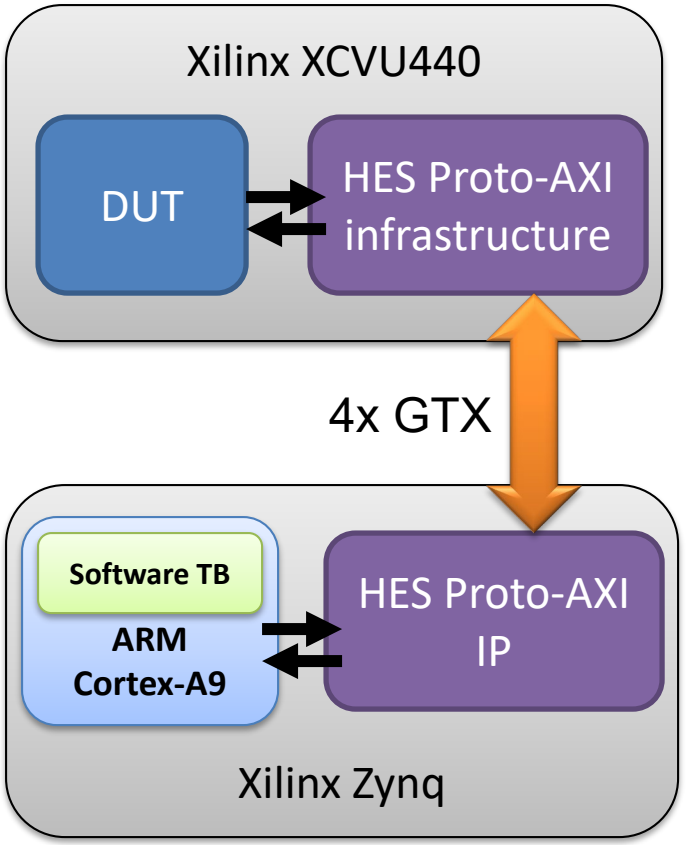


- **Case-2:** Bare metal / Custom
  - Use Xilinx Vivado & SDK
  - Create custom C2C bridge in PL
  - Create custom SDT in PS



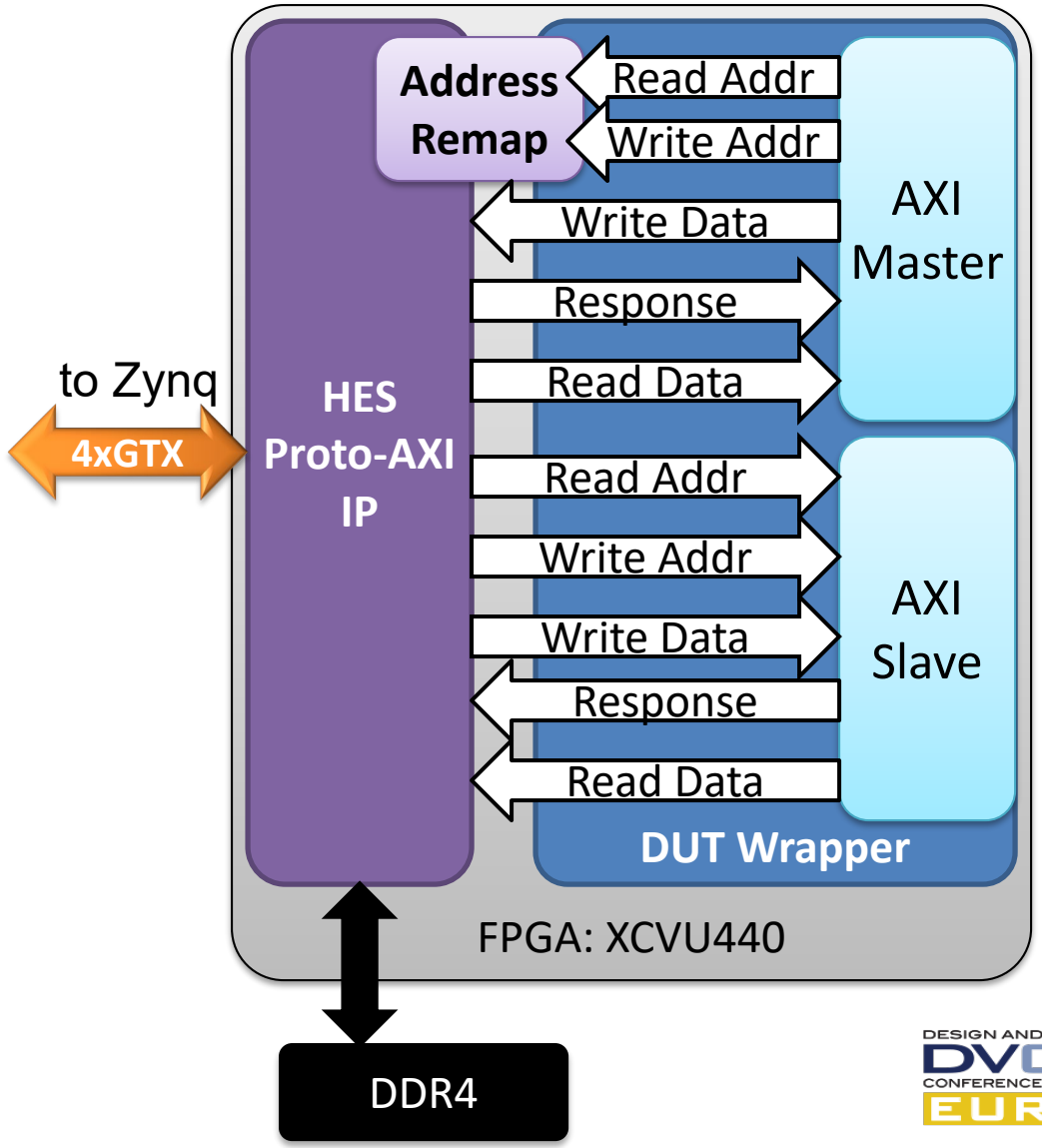
Use Case-1 with HES Proto-AXI

1. Connect DUT to HES Proto-AXI & implement FPGA bitstream
2. Develop & build software TB
3. Run test



Step-1: Connecting DUT to HES-Proto-AXI

- Connect HES Proto-AXI
  - Use any HDL Editor
  - VHDL or Verilog templates available
  - Fixed size AXI4 memory mapped interface
  - Master interface is optional
- Configure address translation parameters (Address Remapper)
- DUT gets access to on-board DDR4 through HES Proto-AXI
- Compile and implement project in Xilinx Vivado



Step-2: Developing & building testbench

- Develop testbench as C/C++ code
  - Use HES Proto-AXI C API
  - `#include “hesprotoaxiapi.h”`
  - Simple, user app level API
    - needn’t know Zynq or Linux kernel
- Compile and build
  - Cross-compilation on host PC
  - Use GCC toolchain from Xilinx SDK
    - `arm-xilinx-linux-gnueabi-g++`

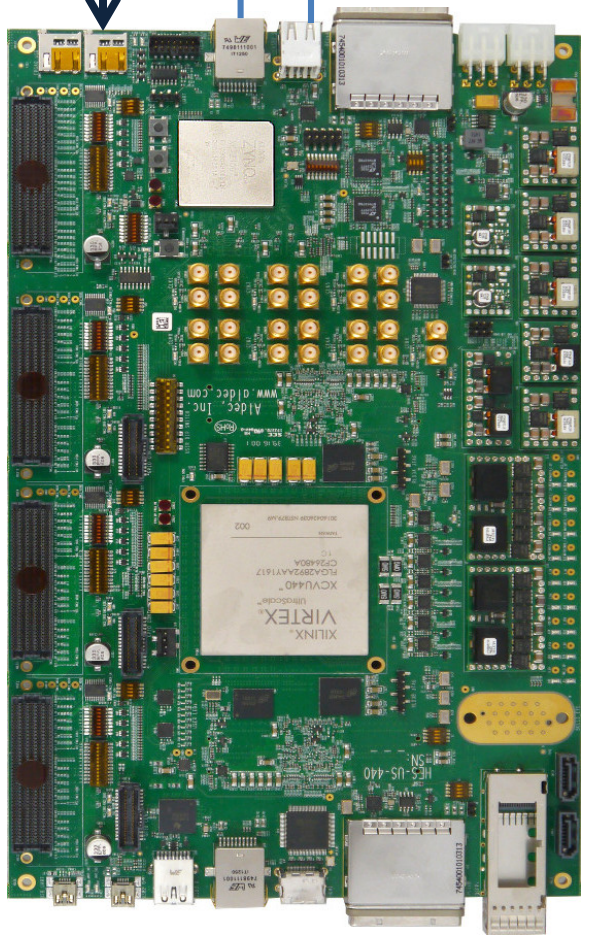
```
// Get Proto-AXI handler
HesProtoAxiHandler HesProtoGetAxiInterface(
    const char *_axisimconfig = 0);

// AXI write (blocking transaction)
HesProtoAxiTrans HesProtoAxiWrite(
    HesProtoAxiHandler  _handler,
    HesProtoAxiLocation _location,
    HesProtoAxiU64      _address32,
    HesProtoAxiBuffer   _buffer,
    HesProtoAxiU64      _bufferSize32);

// AXI read (non blocking transaction)
bool HesProtoAxiReadNoCheck(
    HesProtoAxiHandler  _handler,
    HesProtoAxiLocation _location,
    HesProtoAxiU64      _address32,
    HesProtoAxiBuffer   _buffer,
    HesProtoAxiU64      _bufferSize32);

// GPI write
HesProtoAxiStatus HesProtoAxiWriteGpi(
    HesProtoAxiHandler _handler,
    HesProtoAxiU16     _value);

// GPIO read
HesProtoAxiStatus HesProtoAxiReadGpio(
    HesProtoAxiHandler _handler,
    HesProtoAxiU16   * _out_value);
```

Step-3: Running Test

Proto-AXI Zynq
Images
Linux & FPGA Config

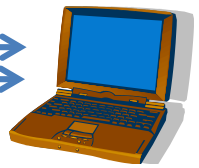
DUT
Bitstream
XCVU440

Software TB
APP
tb.elf



Ethernet

UART/USB

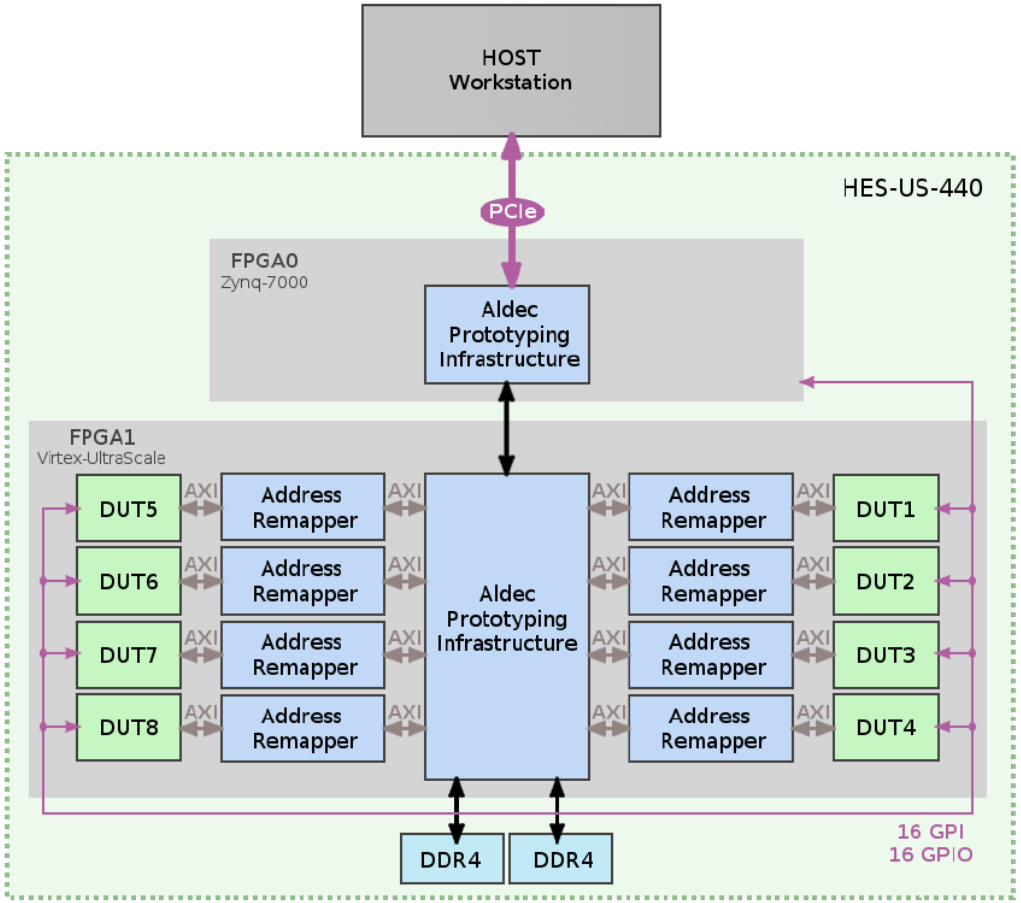


```
Welcome to minicom 2.3

OPTIONS: I18n
Compiled on Mar 22 2017, 08:13:25.
Port /dev/ttyUSB0

                Press CTRL-A Z for help on special keys


aldec_hes:/media/card/v1.1/bin# AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
[1] 1283
-sh: AT: command not found
-sh: c1: command not found
[1]+  Done(127)                 AT S7=45 S0=0 L1 V1 X4
aldec_hes:/media/card/v1.1/bin#
```

HES Proto-AXI Highlights

Data for HES-US-440

- AMBA AXI4 memory mapped
- 8x Master & Slave interfaces
- Data width: 256, Burst length: 256
- Local clock: 160 MHz
- RAM space: 16GB DDR4 + 64MB RLD
- Customizable address translation
- 16x GPI and GPIO additional lines
- Simple C API
- HES Proto-AXI simulation model available
- Embedded & PC Host modes

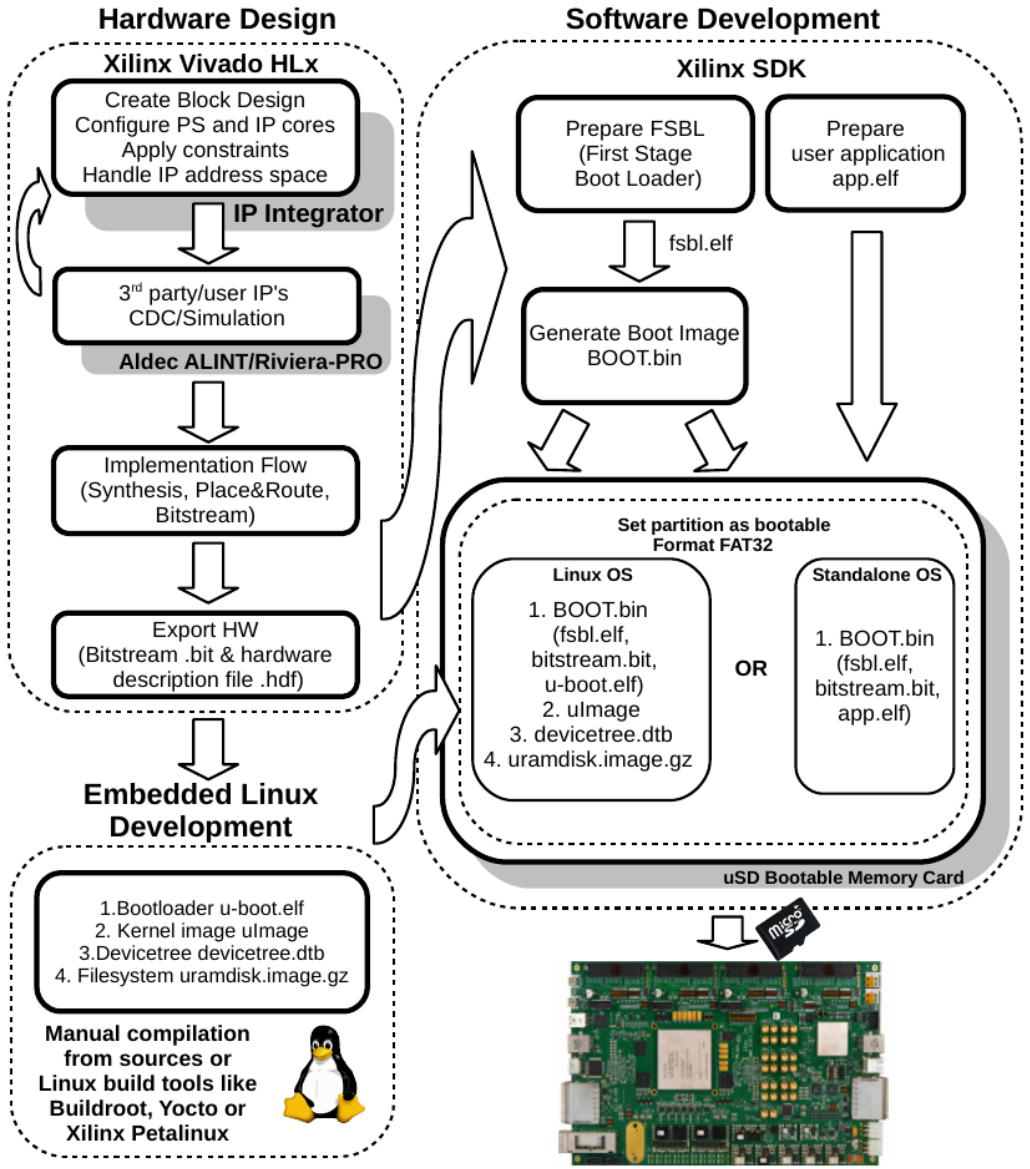


Questions

Next: Case-2: Bare metal approach

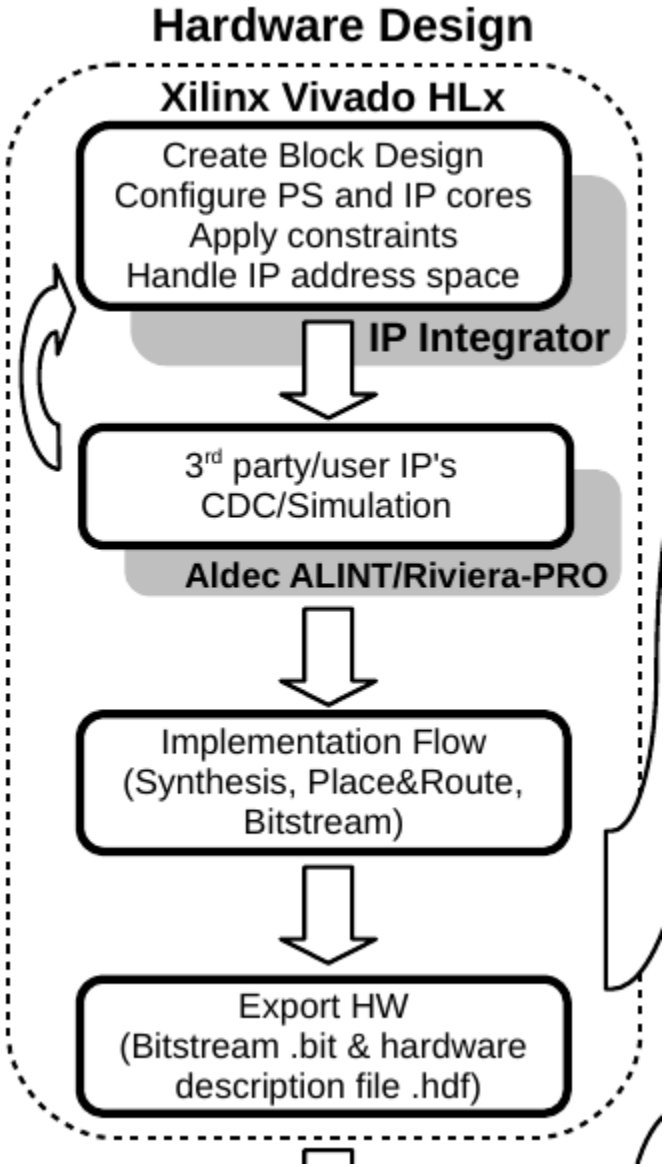
Use Case-2 – Bare metal approach

1. Test controller (Zynq)
   A. Hardware design
   B. Embedded Linux development
   C. Software development
2. Design connectivity (U440)
   A. Custom interconnect
   B. FPGA implementation



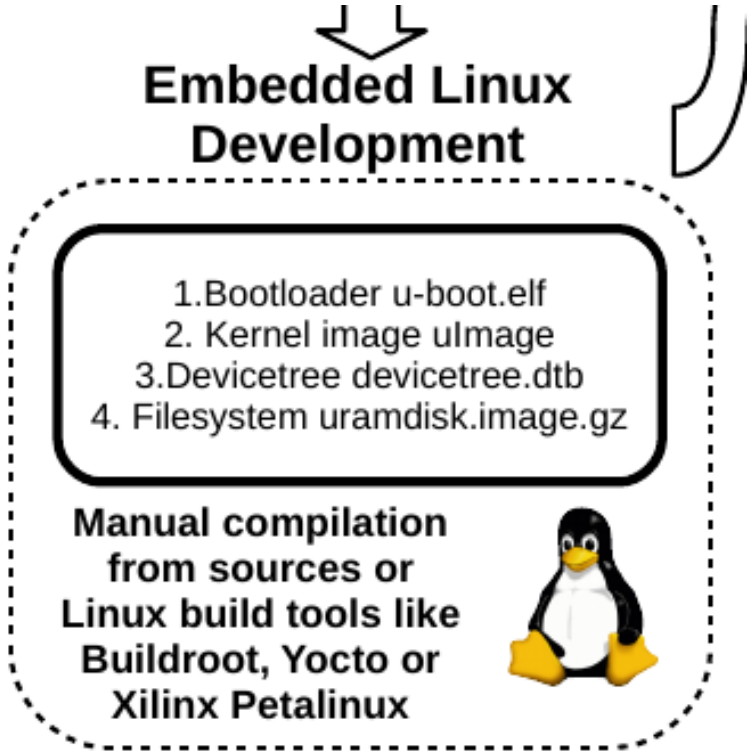
Step 1.A: Hardware design

- Short path: start with reference design from Aldec
- Customize reference design
  - Add your custom chip-2-chip connectivity
- Use Aldec Riviera-PRO with QEMU to verify your changes
- Run Vivado to implement bitfiles and export hardware description



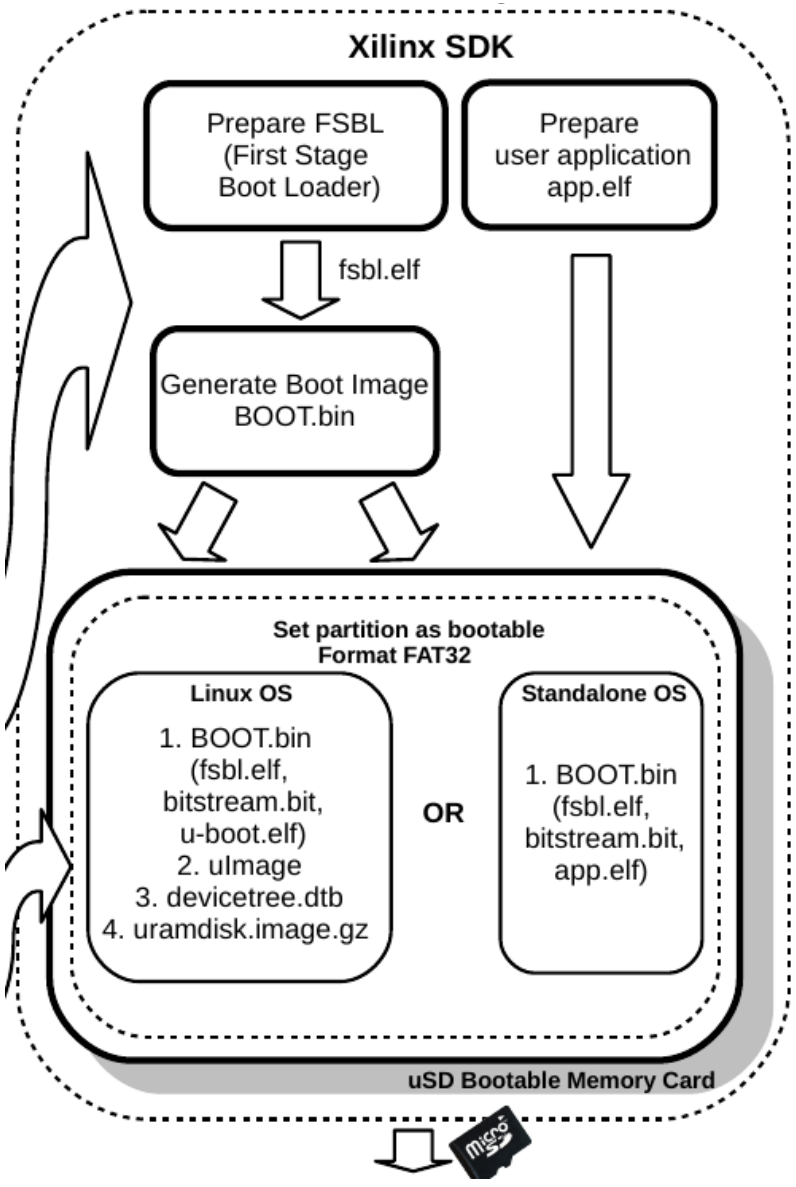
Step 1.B: Embedded Linux development

- Embedded Linux projects available:
  - Yocto
  - Petalinux
- Other than Linux:
  - Free RTOS
  - Bare metal – no OS
    - Xilinx SDK provides toolchain for such mode



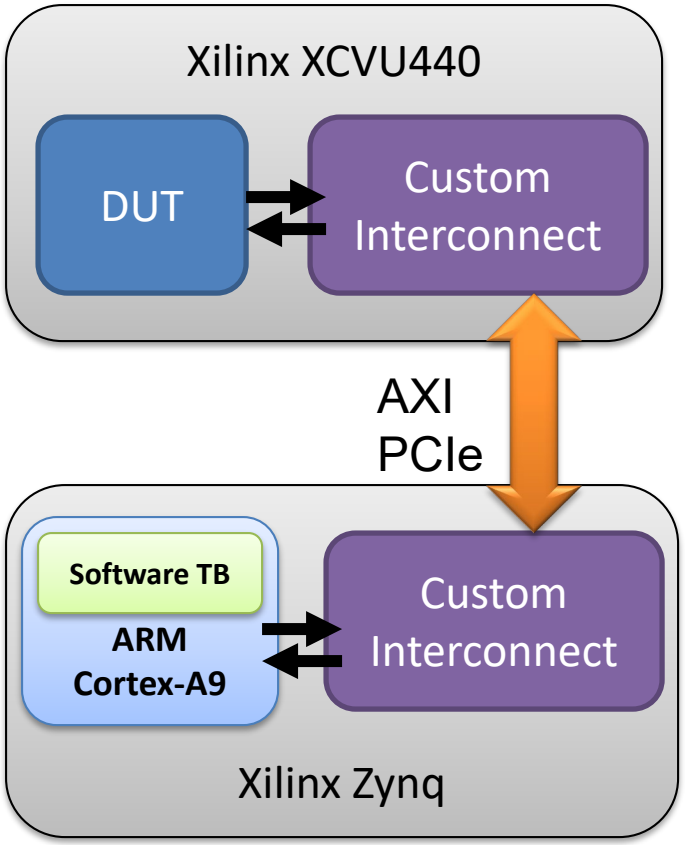
Step 1.C: Software development

- Use Xilinx SDK
  - to develop test application
  - create bootable Linux image
- Copy files to Micro-SD
  - Bitstream for FPGA (PL configuration)
  - Bootloader for ARM (PS configuration)
  - Linux image & device tree
  - User application (app.elf)



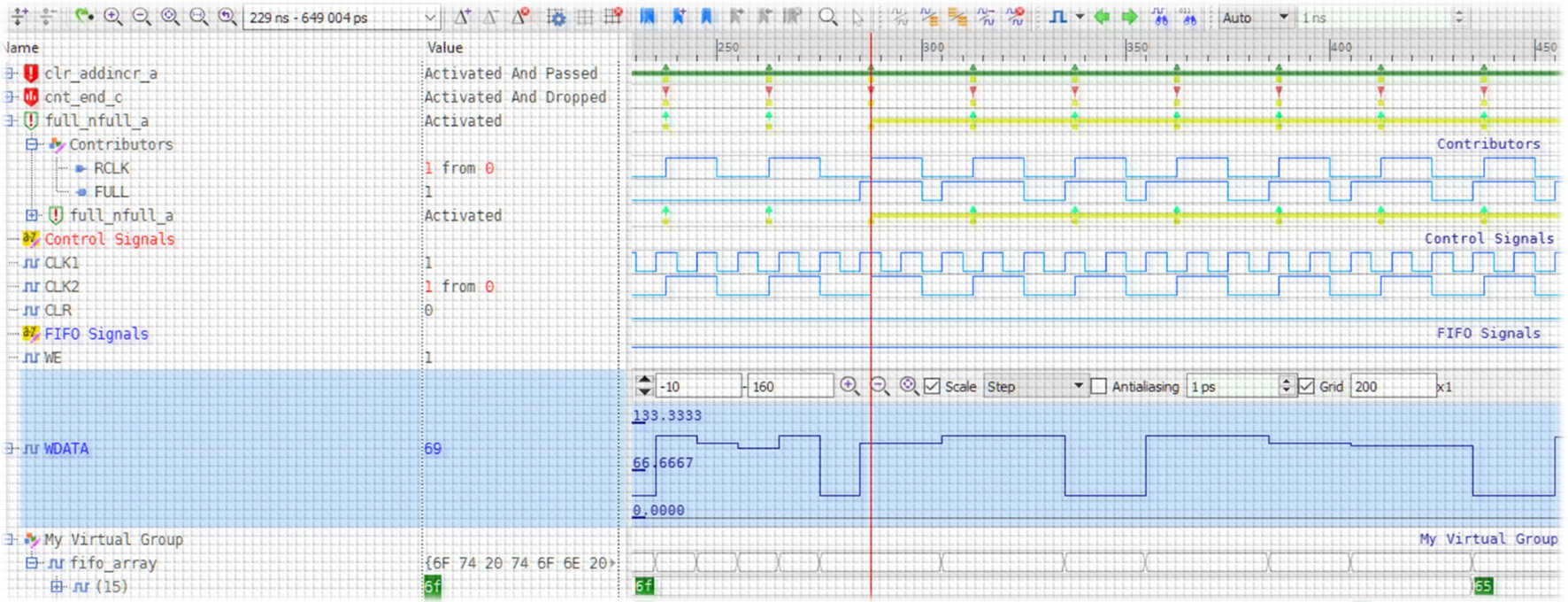
Step 2: Design connectivity

- Use Xilinx Vivado to implement XCVU440
  - Create custom interconnect
    - Coupling with one created in Zynq
    - Use any standard (AXI, PCIe)
    - Use any signaling (LVDS, GTX)
  - Connect with the DUT
  - Run FPGA implementation



Questions

Next:
Software driven testbench in simulation:
QEMU co-simulation with Riviera-PRO



QEMU CO-SIMULATION WITH RIVIERA-PRO

Radosław Nawrot
Software Products Manager, Aldec Inc.

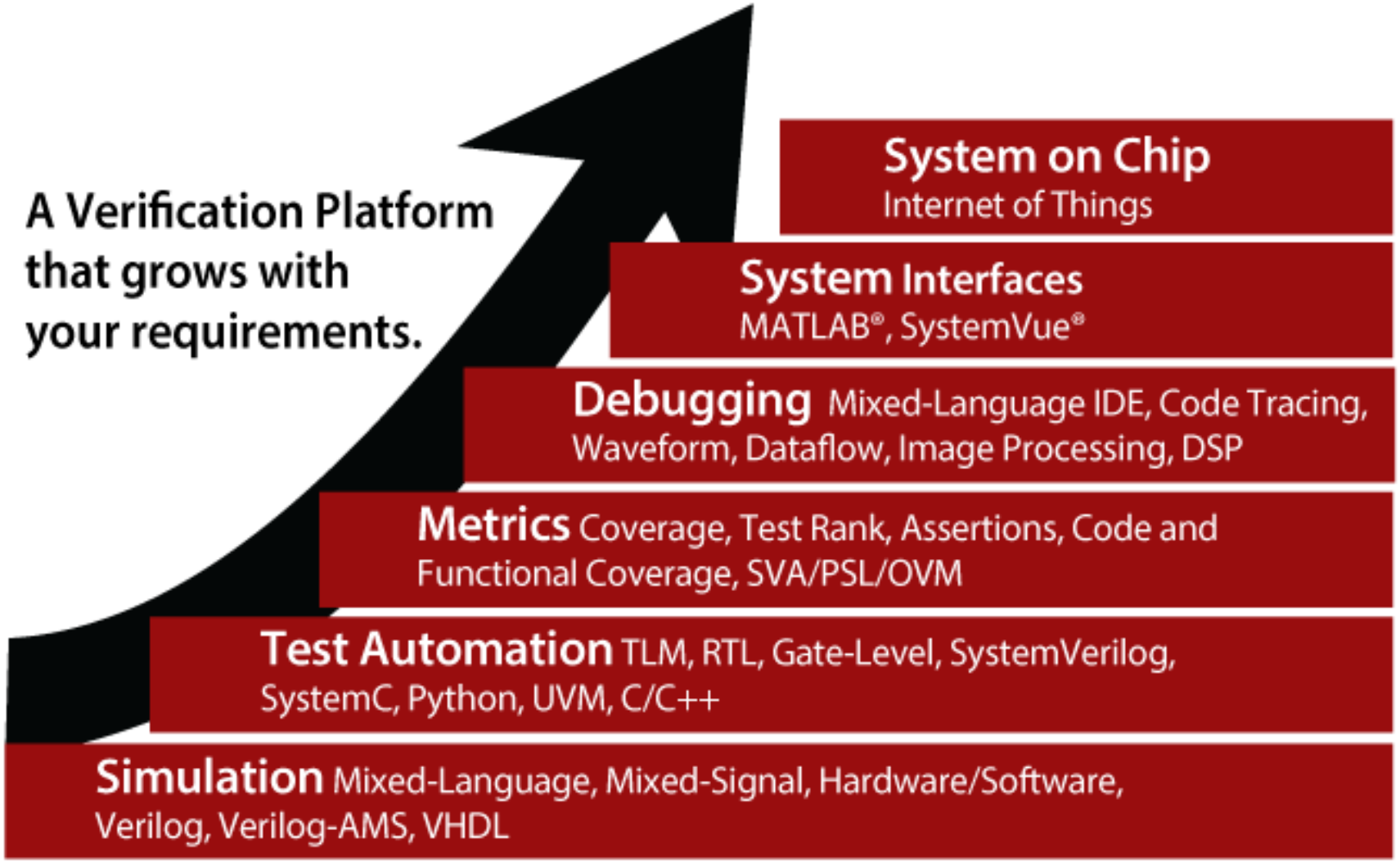
QEMU co-simulation platform

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

QEMU co-simulation platform

- 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

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

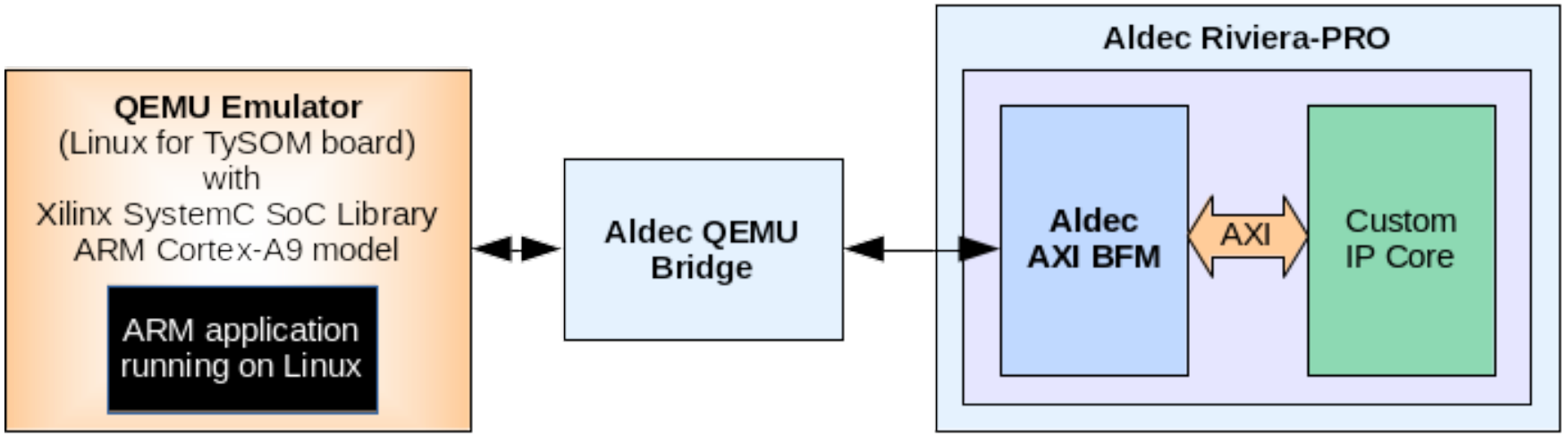
Riviera-PRO Highlights

- 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

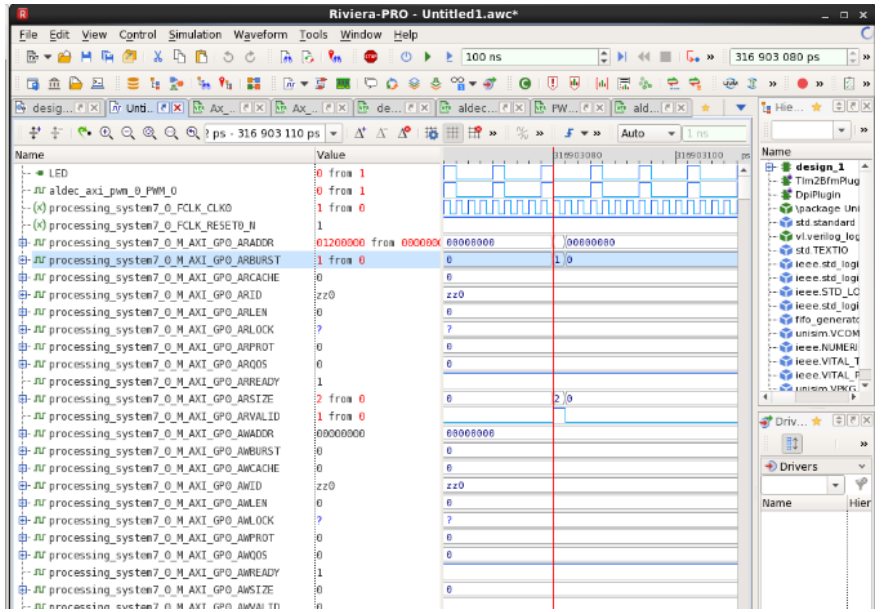
QEMU – Virtual Platform

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

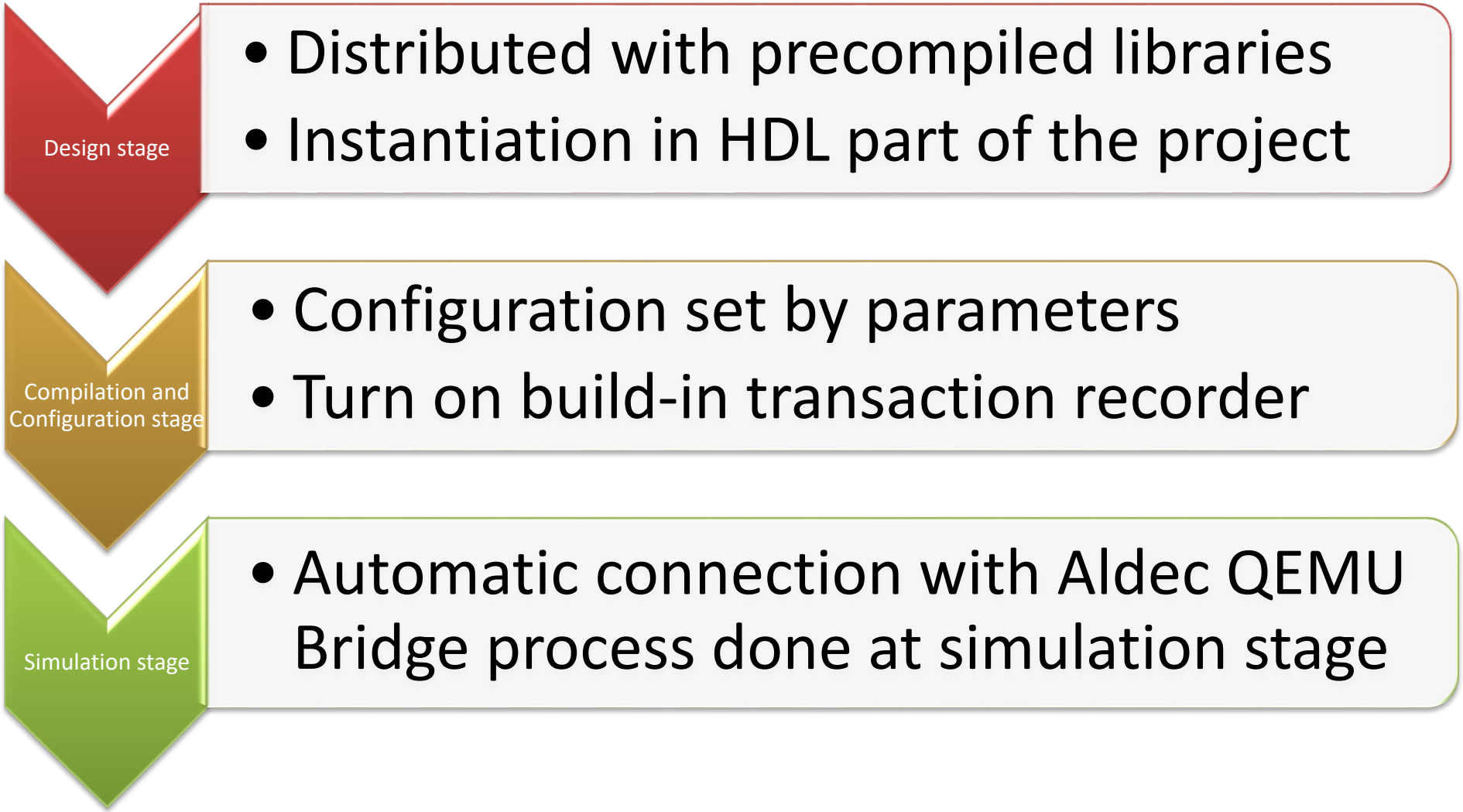
Riviera-PRO and QEMU




```
++ 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
```

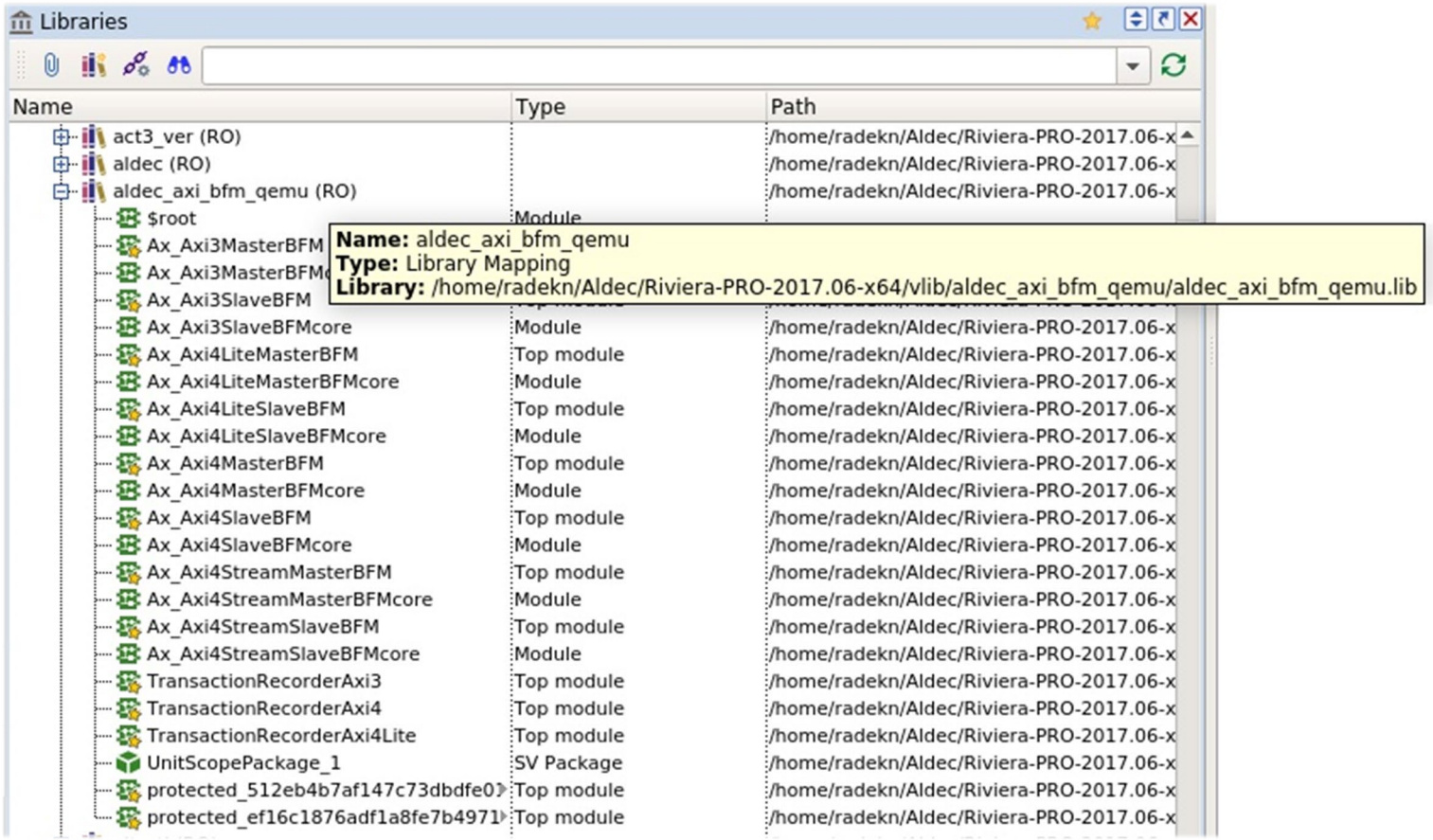


Aldec AXI BFM usage

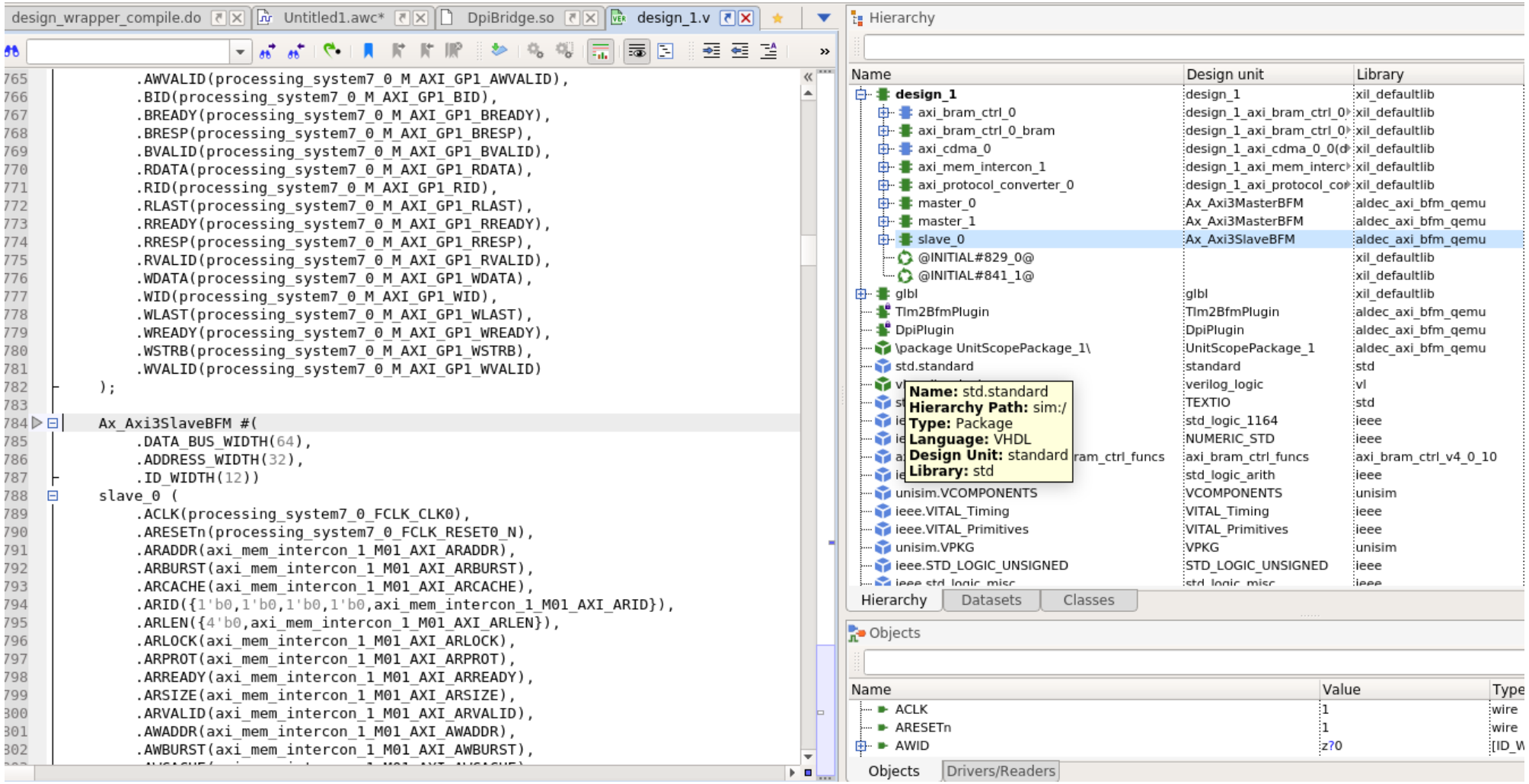


Aldec AXI BFM

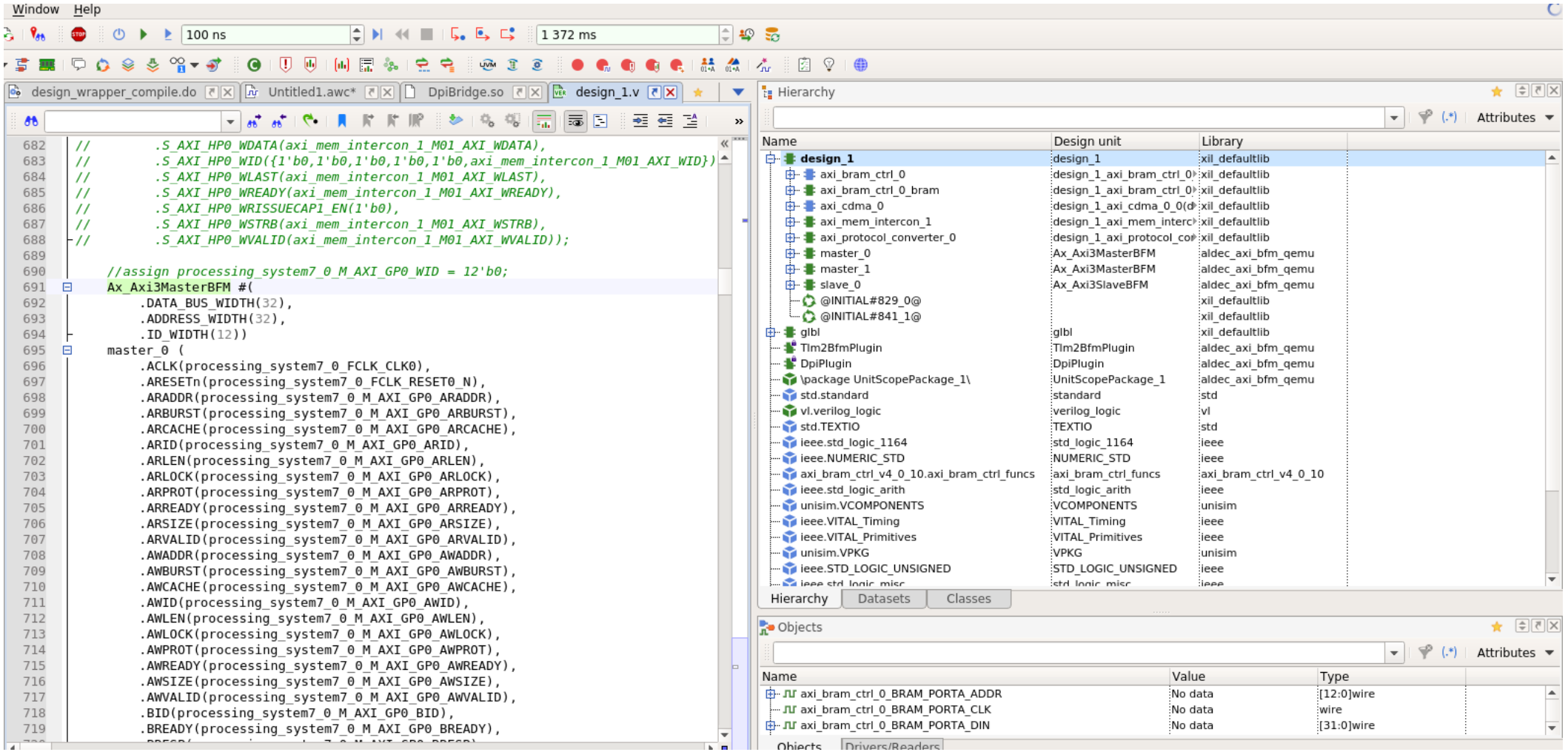
- AXI 3 Master
- AXI 3 Slave
- AXI 4 Master
- AXI 4 Slave
- AXI 4 Lite Master
- AXI 4 Lite Slave
- AXI 4 Stream Master
- AXI 4 Stream Slave



AXI BFM instance in HDL Code

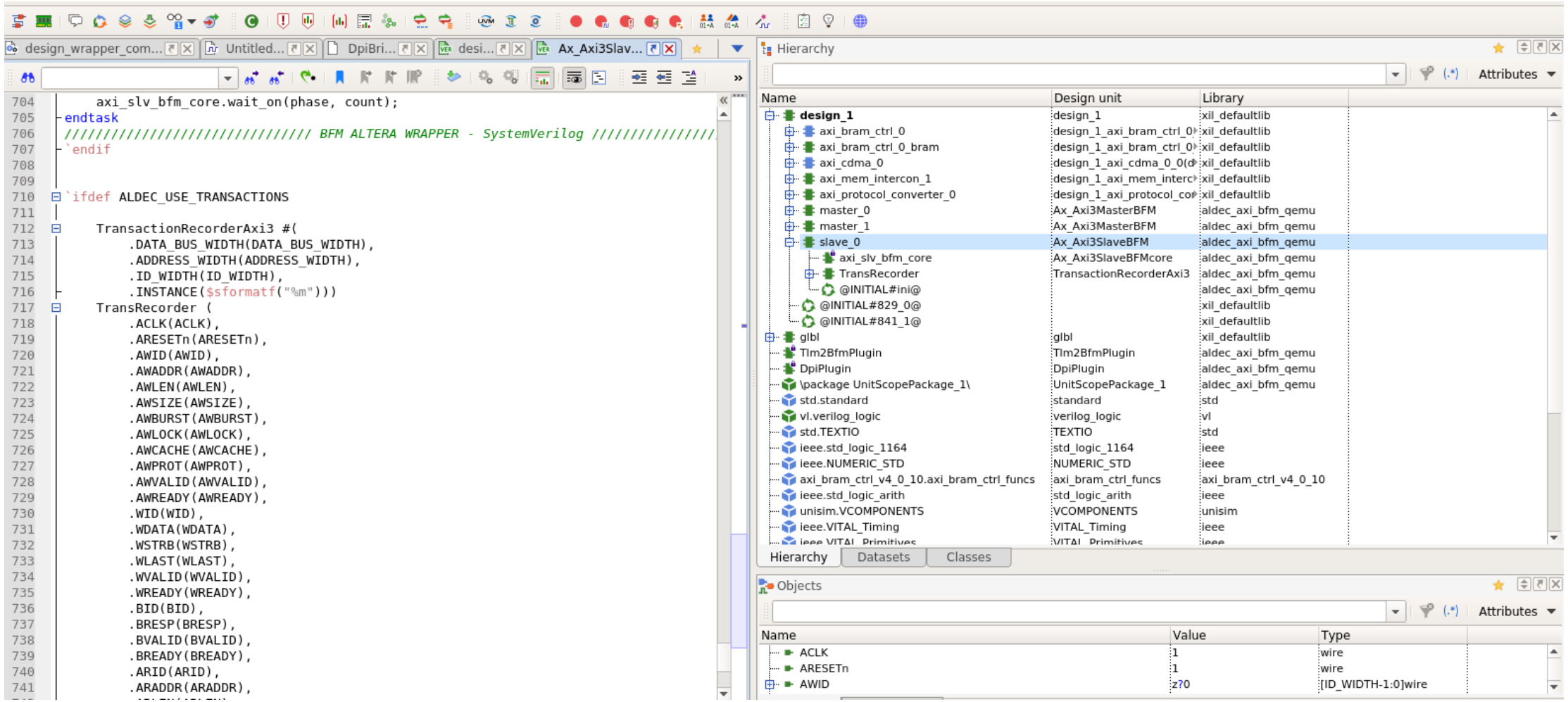


AXI BFM instance in HDL Code

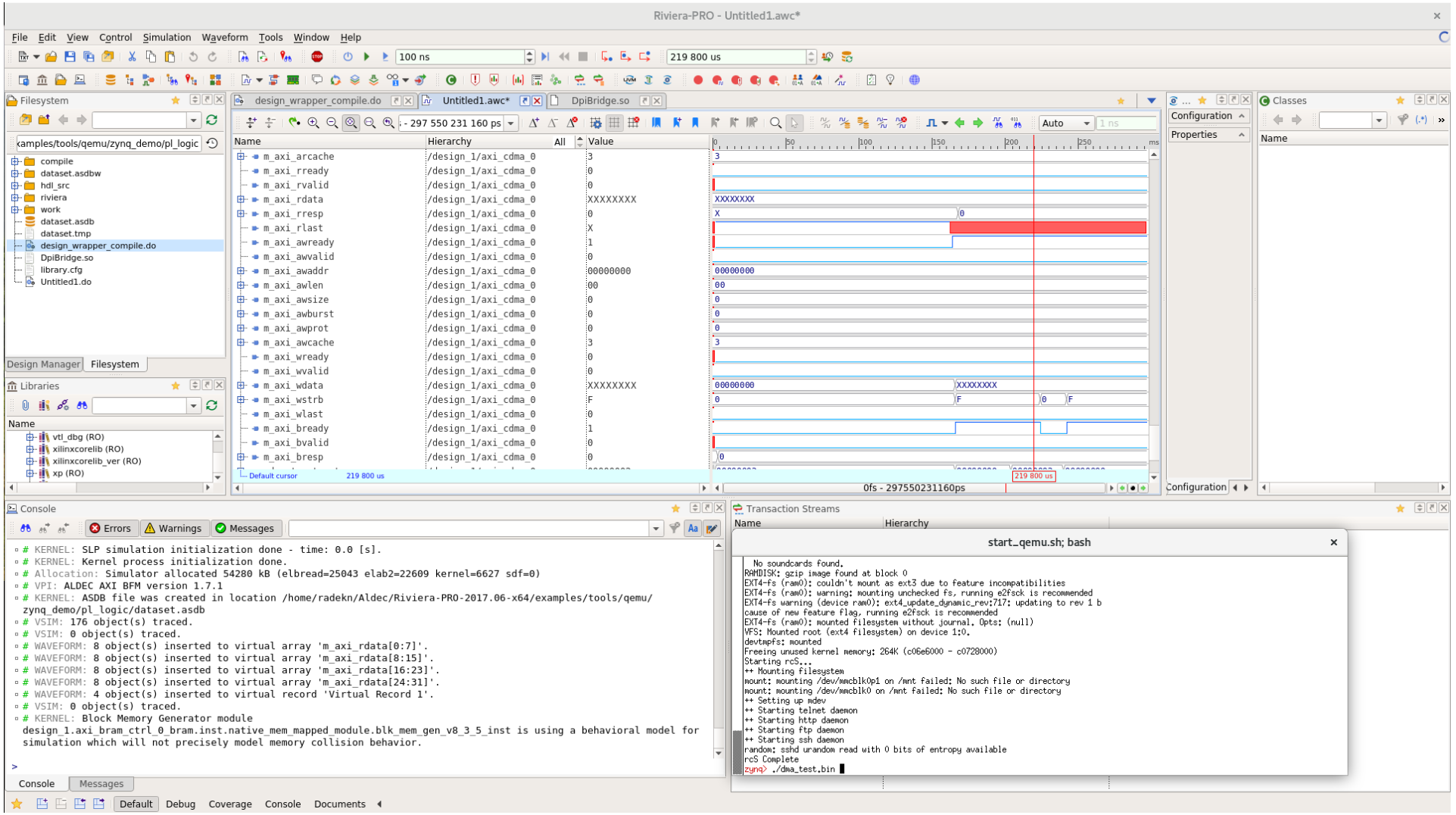



```
//        .S_AXI_HP0_WDATA(axi_mem_intercon_1_M01_AXI_WDATA),
//        .S_AXI_HP0_WID({1'b0,1'b0,1'b0,1'b0,1'b0,axi_mem_intercon_1_M01_AXI_WID}),
//        .S_AXI_HP0_WLAST(axi_mem_intercon_1_M01_AXI_WLAST),
//        .S_AXI_HP0_WREADY(axi_mem_intercon_1_M01_AXI_WREADY),
//        .S_AXI_HP0_WRISSUECAP1_EN(1'b0),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

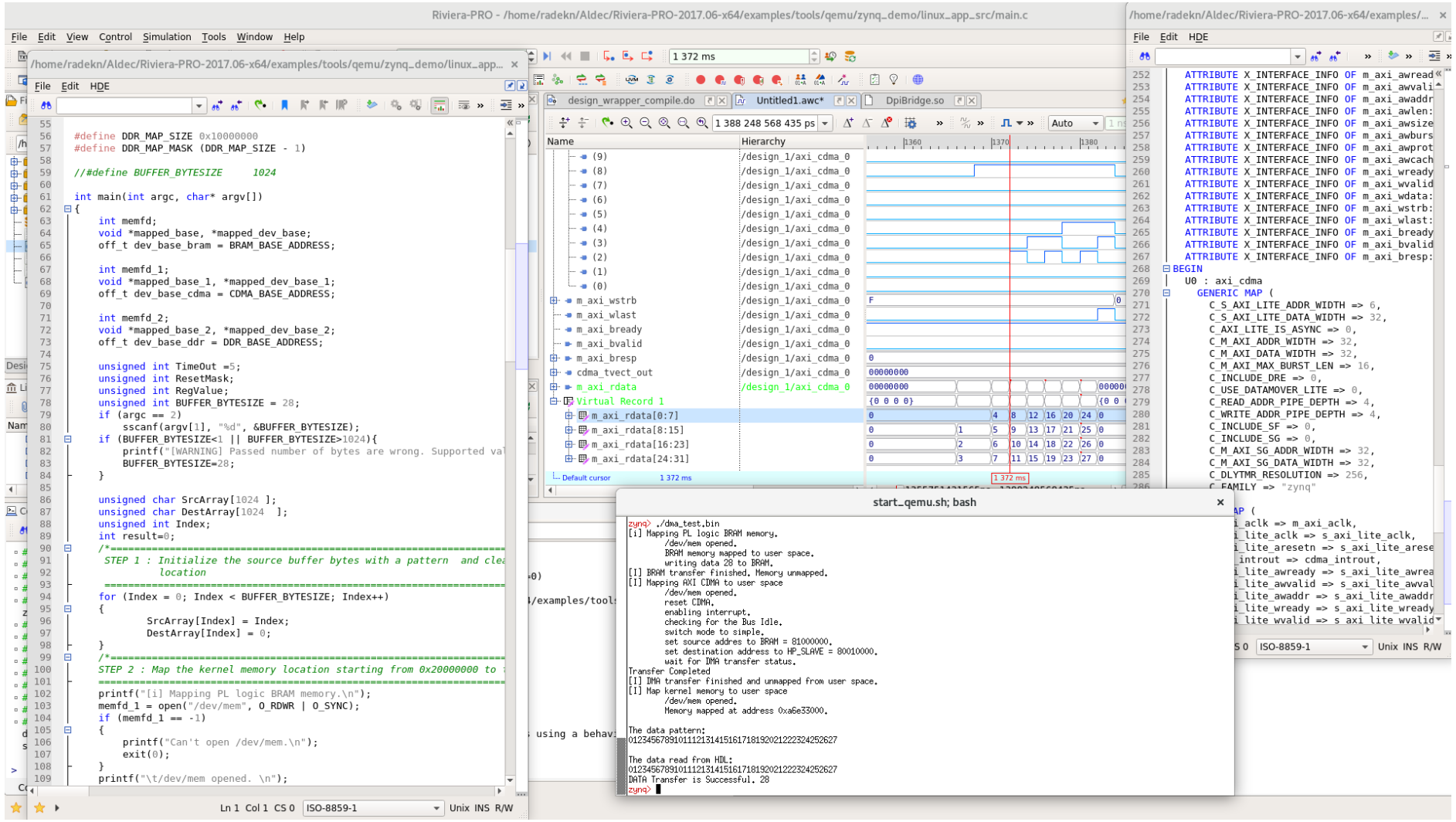
Built-in Transaction Recorder



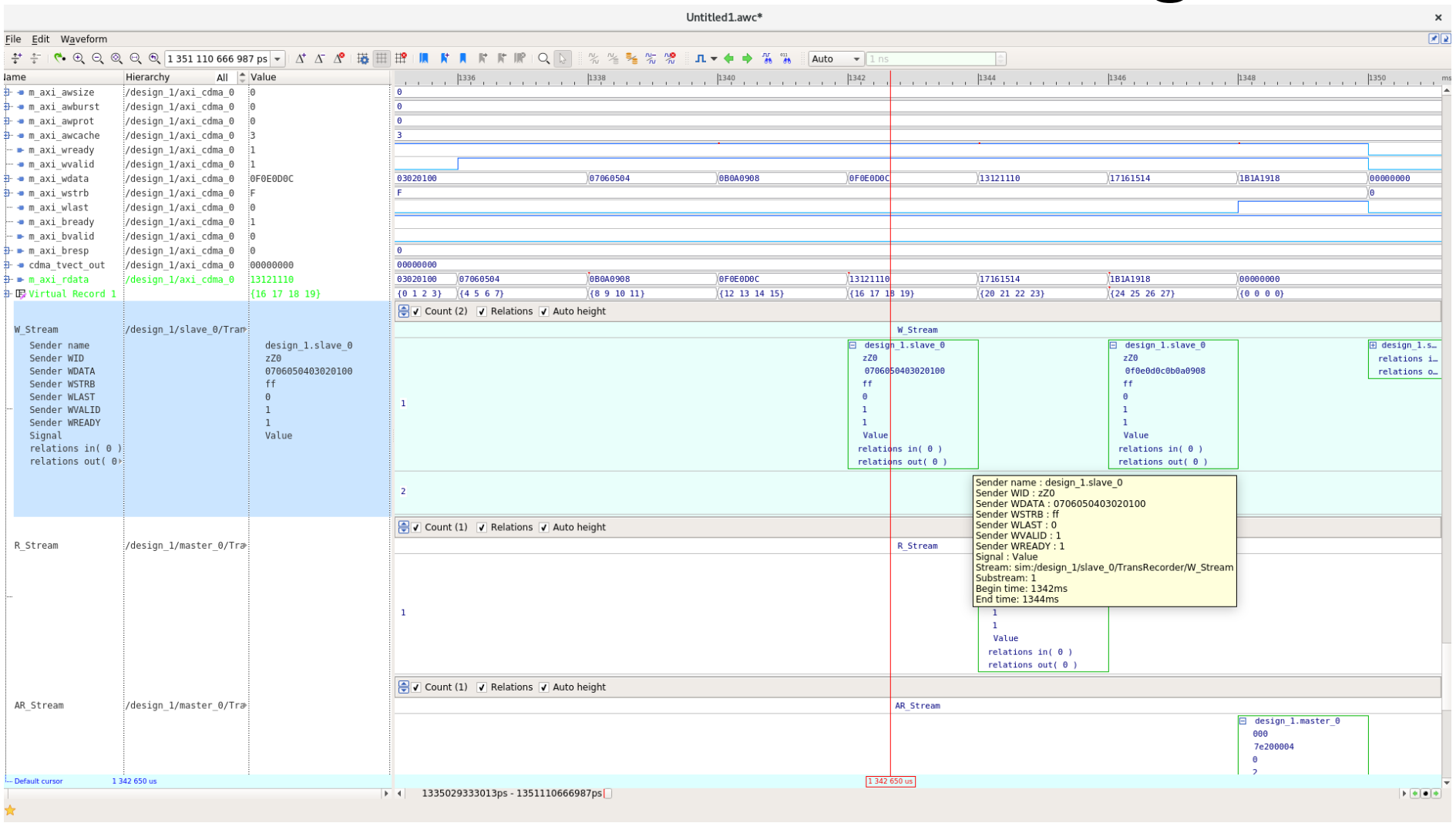
Riviera-PRO QEMU environment



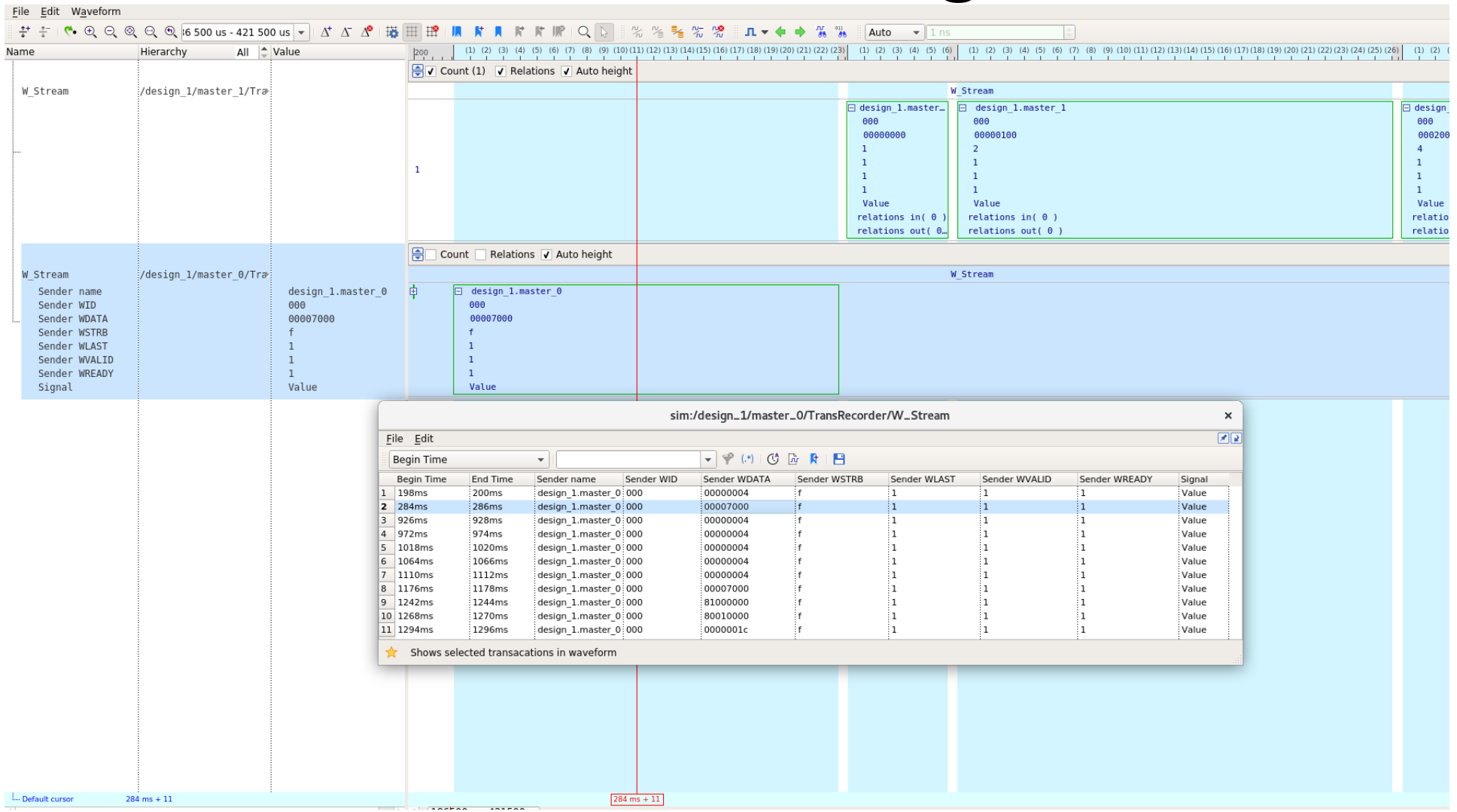
Hardware and Software co-sim



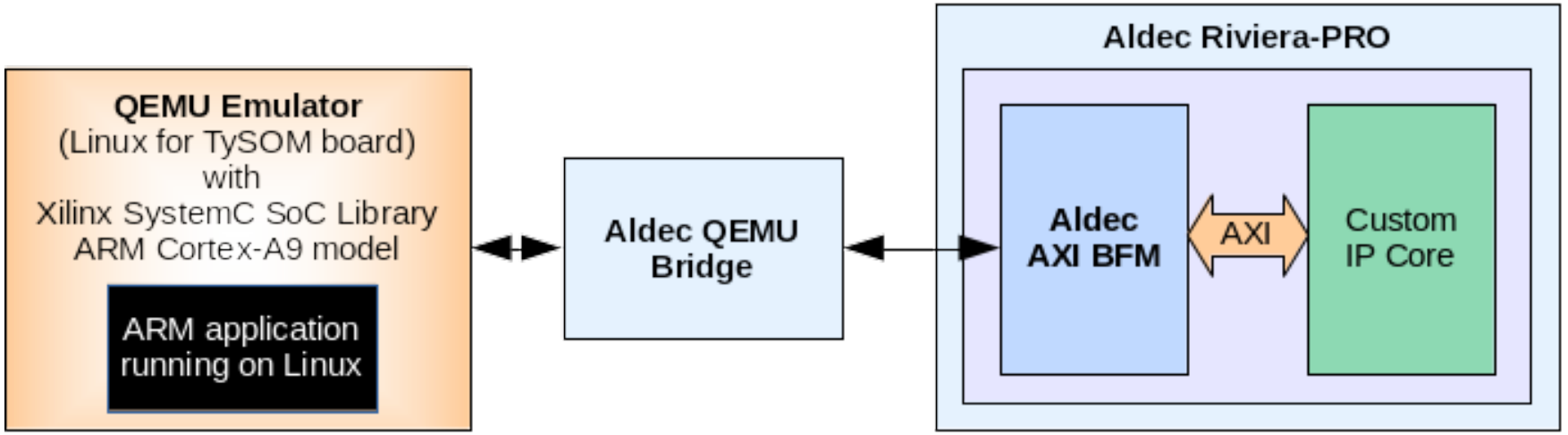
Transaction based debug



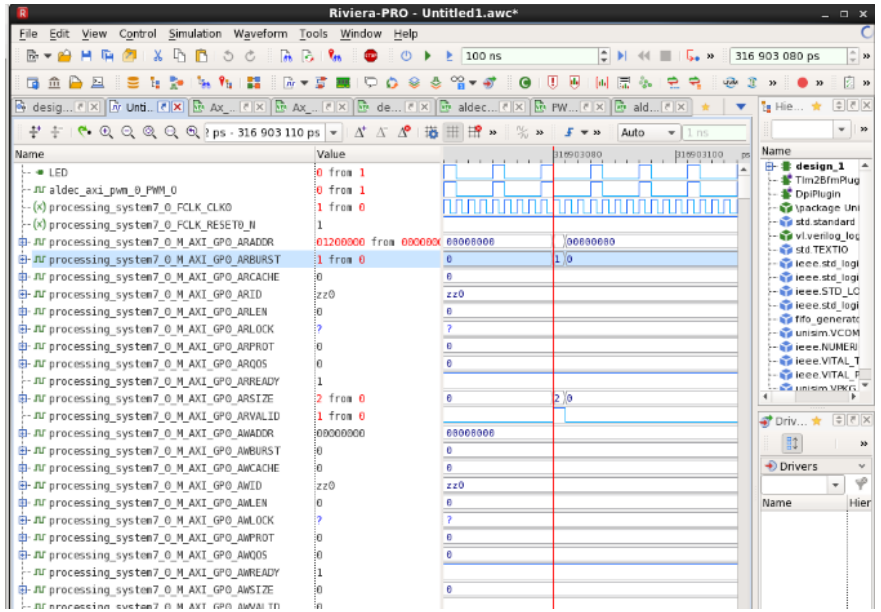
Transaction logs



Riviera-PRO and QEMU – project flow




```
++ 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
```



TySOM – Zynq Based EDK

Specs \ Boards	TySOM-1	TySOM-2	TySOM-2A
			
SoC	XC7Z030 FBG484	XC7Z045/100 FFG900	XC7Z030 2FFG676I
DDR3	512MB	1GB	1GB
SPI Flash	16MB	16MB	16MB
uSD/SD/EMMC	uSD	uSD	uSD
I2C EEPROM	64Kb	64Kb	64Kb
USB 2.0	2	4	4
USB 3.0	2	-	-
UART/USB → UART	2 UART	1 USB → UART	1 USB → UART
JTAG PL Header	1	1	1
1 Gb Ethernet	V	V	V
Ethernet IEEE1588	-	-	V
Audio IN/OUT	V	-	-
HDMI/VGA	HDMI	HDMI	HDMI
Camera Con	V	-	-
mPCIe	V	-	
FMC	-	× 2	× 1
Pmod	2 × 6	-	-
GPIO	43 (*)	-	14(**)
GTX	1 on MMCX con	16 on FMC con	4 on FMC con
ADC	V	V	V
Wi-Fi/Bluetooth	-	-	V
Sensors	Temp. Accelerometer	Temp. Accelerometer	Temp. Accelerometer
RTC	V	V	V
TouchPannel/LCD Interface	T.P & LCD	-	-
User DIP switch	8	8	8
User LED	8	4	4

* Available on mother boards / expansion carrier board.
**Available on XADC Header.

Specs \ Boards	FMC – IoT	FMC – Vision	FMC – ADAS	FMC – INTF
				
Memory	64 Kb EEPROM, uSD, USER SATA	64 Kb EEPROM	64 Kb EEPROM	4 Gb NAND Flash, 2× 32 Mb SPI Flash Memory, 64 Kb EEPROM, uSD
Communication interfaces	FMC, 1× Ethernet IEEE1588, Wi-Fi, Bluetooth 4.1, ZigBee, 6LoWPAN, RF4CE, SP100, WirelessHART, Sigfox, WMBus, mPCIe, USB 2.0, Micro Sim card, RS-485, 2×6 Digilent Pmod	FMC, 1× HDMI IN, 1× HDMI OUT, 1× Display Port IN, 1× Display Port OUT, MIPI DSI, MIPI CSI, Audio Codec, 2× USB 3.0	FMC, 5× FPD-Link lll (LVDS interface), 1× URM37_V4.0 Ultrasonic sensor connector, 1× LIDAR-Lite_V2 Connector	FMC, 1× Ethernet, PCI-Express connector, QSFP+ connector, 1× USB 2.0, 1× UART to USB
Miscellaneous	2× user clock, 1× clock for SATA, Dedicated clock to PCIE	2× Oscillators, 2× Clock for transceivers	Buzzer, 2× Oscillator	3× Oscillator, 8× User DIP switch, 8× User LED, GPIO (20 IOs, 2×20 gold pins connector)

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

Resources

Training	www.aldec.com/training
Support	www.aldec.com/support
Blog	www.aldec.com/company/blog

Questions