2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

How to test the whole firmware/software when the RTL can't fit the emulator

Horace Chan, Byron Watt – Microchip Technology

a MICROCHIP company

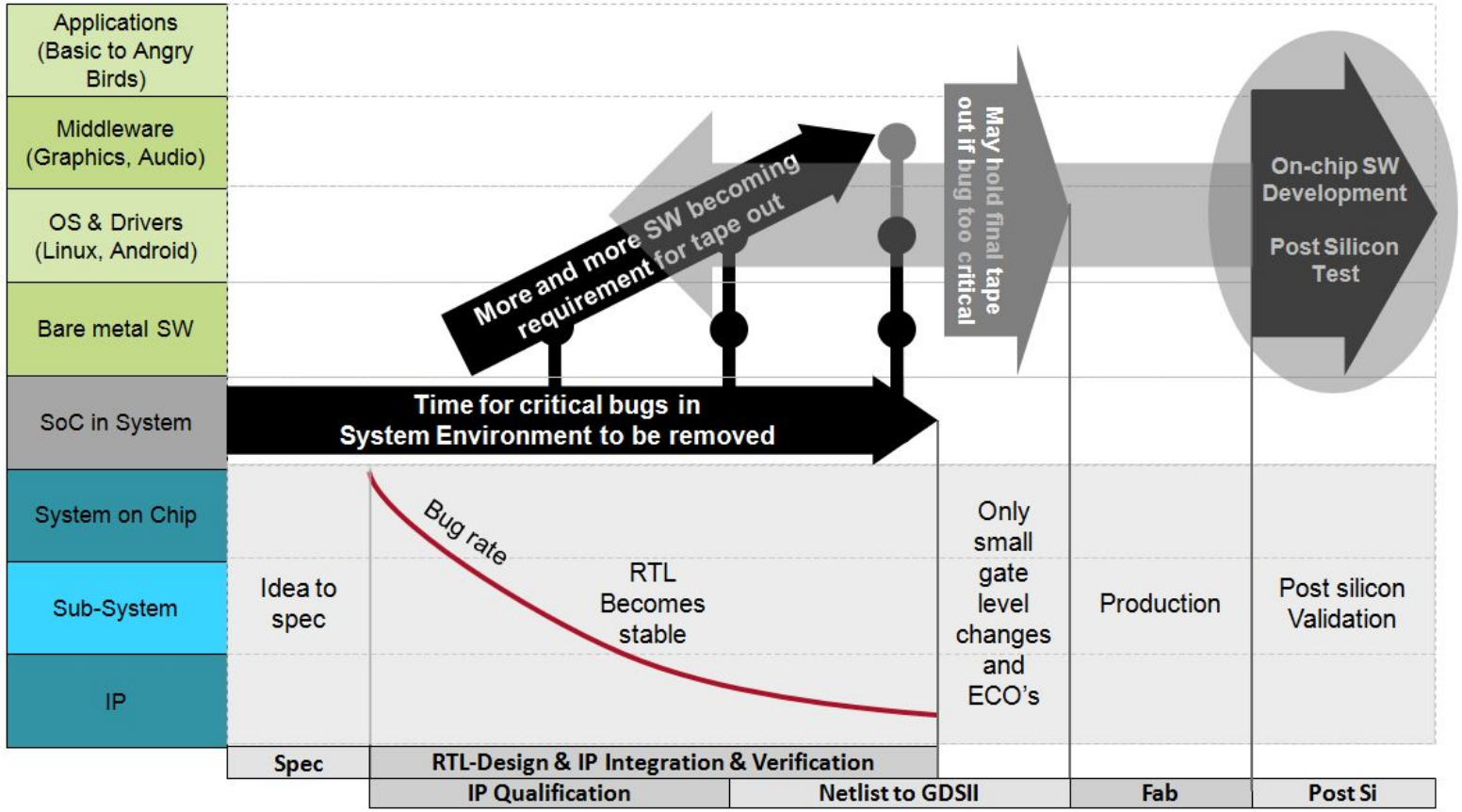
accellera SYSTEMS INITIATIVE

Agenda

- Introduction
- Challenges in Hybrid Emulation
- Existing Solutions
- Architecture of the Hybrid Software Simulator
- Use Mode on Testing User Space FW
- Use Mode on Testing Kernel Space FW
- Results

Introduction

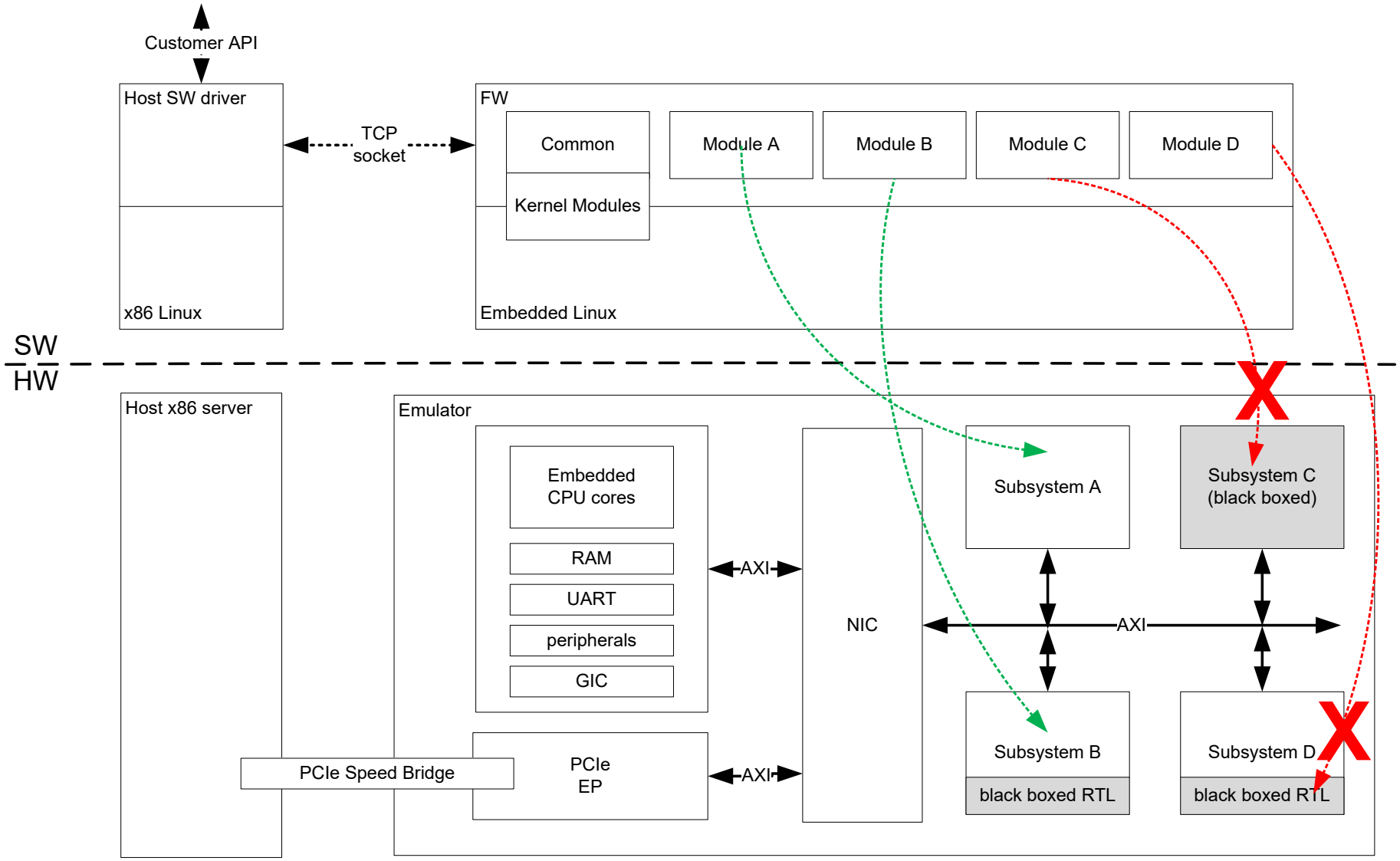
- Pre-silicon FW/SW testing using emulation is a MUST
  - Catch system bugs before tape-out



Blackbox snapshots

- The design is too big to fit in the emulator
  - Buy bigger emulators (if you ask the EDA vendor)
  - $$$$
- Borrow an idea from verification
  - Blackboxes RTL logic not required by the test
  - Build multiple design snapshots for different test scenarios
  - Save up to 50% of gates on average

Challenges in Hybrid Emulation



Existing Solutions

- Run block level FW in verification TB
  - Tried this approach in previous project
  - Can't test the whole FW/SW before silicon
  - Can't find integration bugs
  - The verification team hate it.
    - Extra C code to debug on top of the RTL.
    - The FW is not as verification-friendly as UVM sequences

Existing Solutions

- Use of #ifdef or makefile parameters to partition the FW
  - Tried this approach in previous project
  - Spent lots of unproductive hours hacking the FW code
    - add #ifdef all over the place
    - create mock function to bypass register access into blackboxed addresses
  - It is a maintenance nightmare on RevB
  - This solution doesn’t scale

Existing Solutions

- Insert dummy AXI/AHB slave components in the blackboxed RTL
  - Investigated this approach in the beginning of this project
  - Too much work to create the blackboxed SV modules
  - Too much work to hack the RTL compile script for emulation
  - Can’t model the blackboxed RTL behavior
    - AXI/AHB dummy only returns zero on read

Existing Solutions

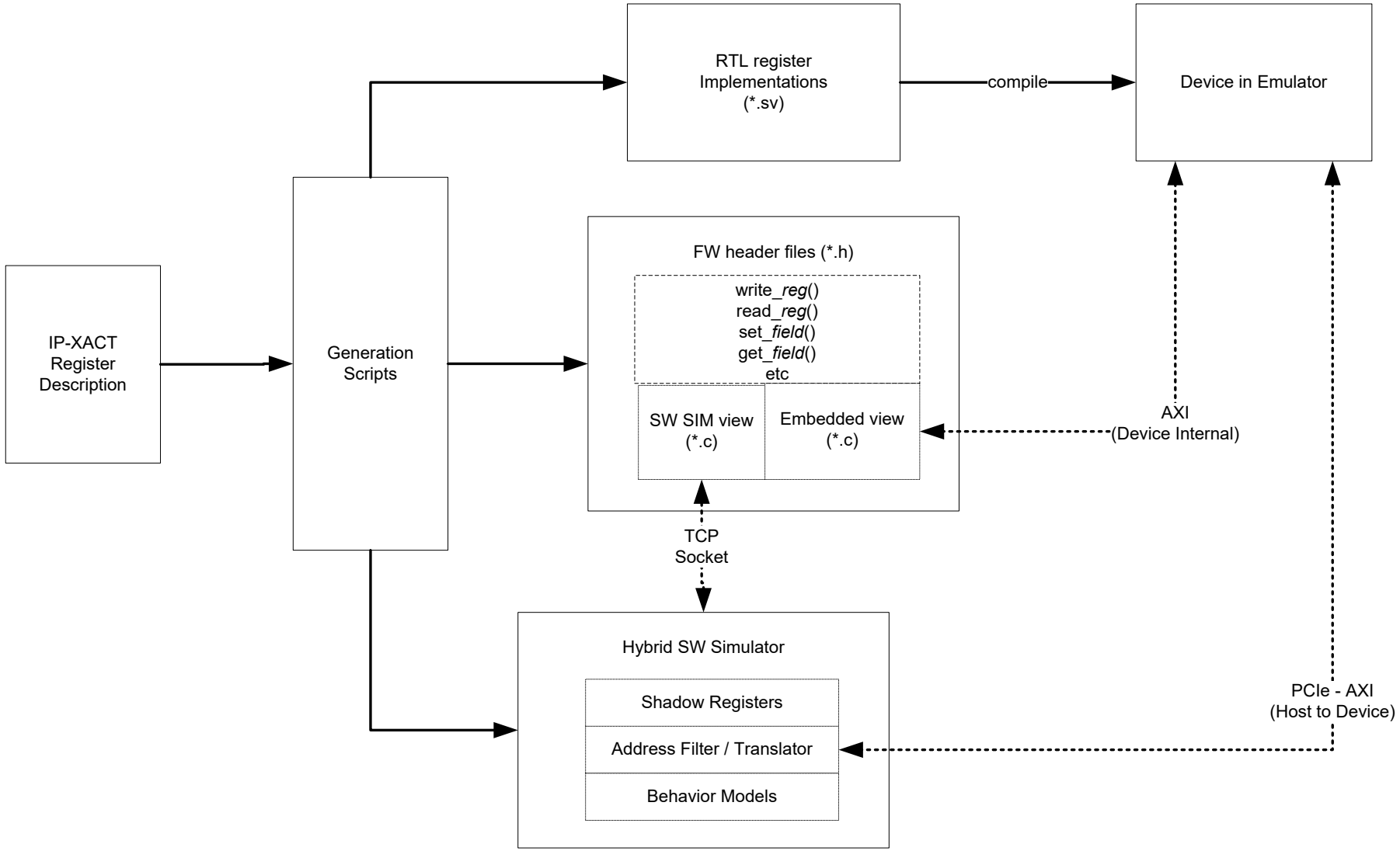
- Use of a Virtual Platform
  - Investigated this approach in the beginning of this project
  - Too time consuming to setup QEMU proxy and SystemC models
  - Inconvenient to debug the FW code compare to our approach

  - Doesn’t work for us, but it might be the right choice for a different project

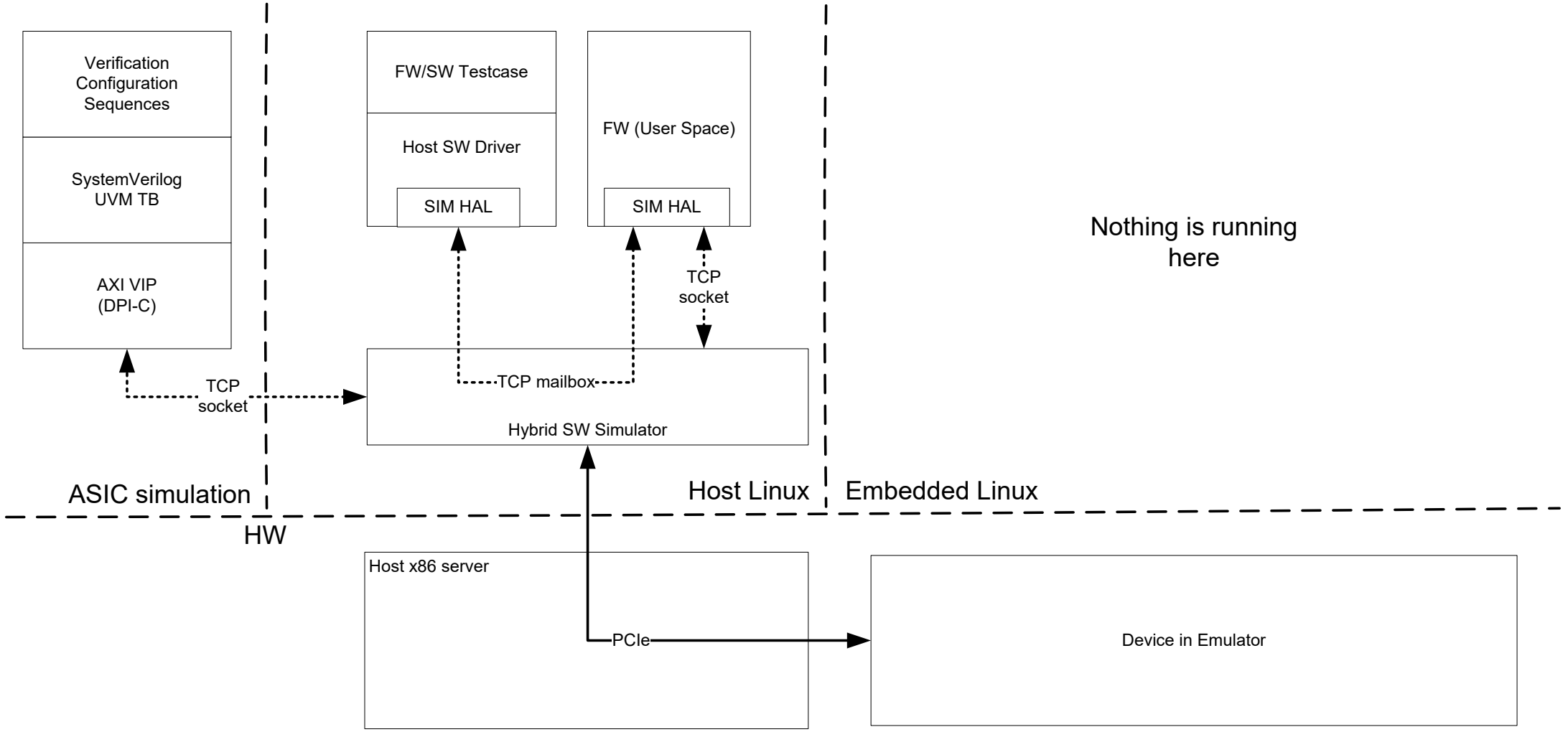
Our Hybrid Emulation Scenario

- A x86 Linux server controls multiple chips via PCIe bus in the production environment
- The PCIe bus is replaced by the PCIe speed bridge in the emulation environment
- The embedded CPU inside the chip is also running Linux
- The SW running on the host Linux communicates with the FW running inside the embedded Linux via TCP sockets or PCIe mailbox.
- The end-user has no direct access into the FW. The end-user interacts with the FW through the public API of the SW running on the host Linux

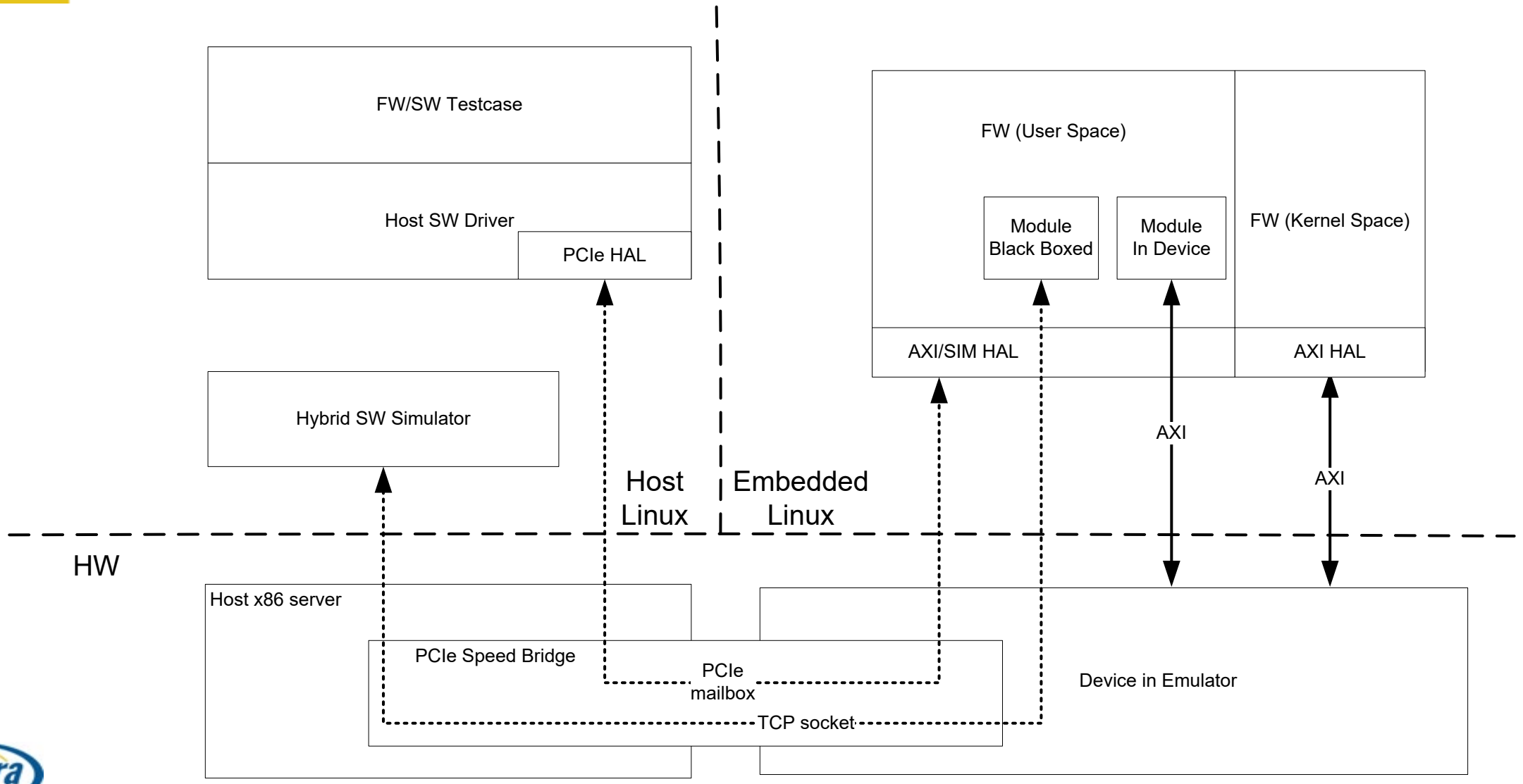
Hybrid SoftWare SIMulator



Test User Space FW



Test Kernel Space FW



Results (pre tape-out)

- All major FW features are tested in emulator
  - Design at 5Mhz in the emulator
  - Linux boots up in less than 5 minutes
  - Found 3 RTL bugs before tape-out

Results (Silicon)

- Smoothest bring up for a Rev A device
  - Linux and FW running on day 1
  - First major feature running on day 3 (debug non-digital FW code)
  - All major features running on the first week
  - First FW release to customers in less than a month

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Q & A

accellera SYSTEMS INITIATIVE