Use of the Imperas reference model, RVVI, and ImperasDV for the OpenHW CV32E20 project

9 March 2022

Agenda

- OpenHW DV Status and CV32E20 Collaboration
- ImperasDV Overview and Demo
- Next Steps

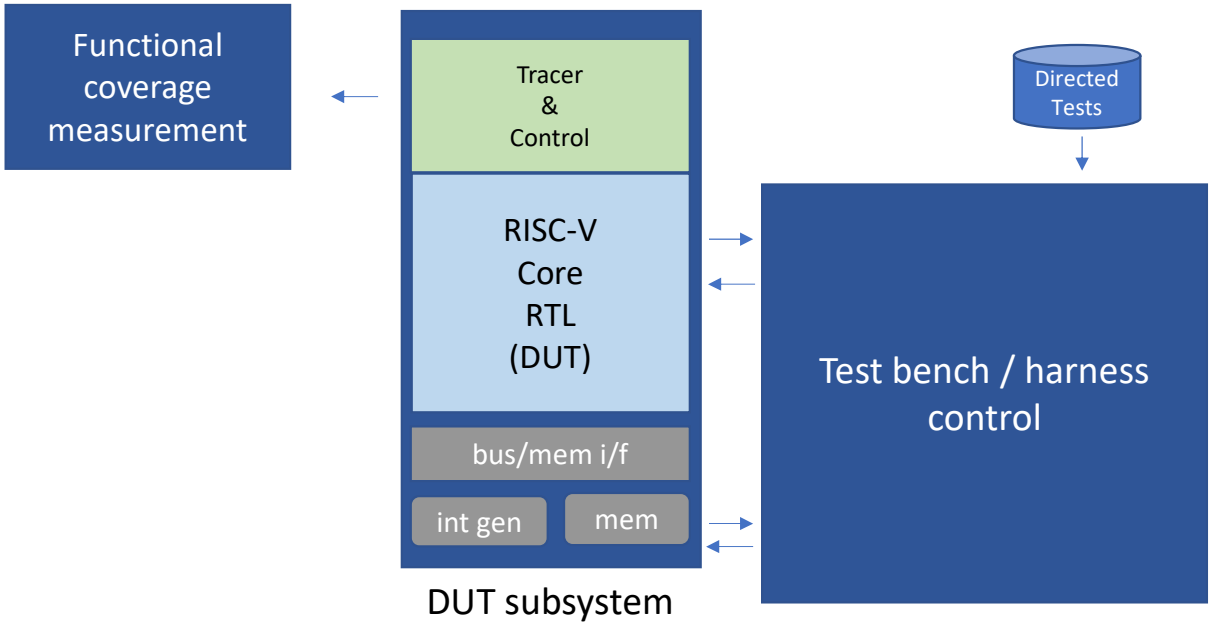
Imperas: OpenHW DV Status

Core	RTL Tracer	Test Bench	Reference Model Subsystem
E40P (SiLabs, Imperas, OpenHW)	Ad hoc	SystemVerilog	M*DEV (ref. model + simulator)
E40S (SiLabs, Imperas, OpenHW)	RVFI + extensions	SystemVerilog	M*DEV (ref. model + simulator)
E40X (SiLabs, Imperas, OpenHW)	RVFI + extensions	SystemVerilog	M*DEV (ref. model + simulator)
E40Pv2 (Dolphin, Imperas, OpenHW)	RVFI + extensions + RVVI-VLG (Dolphin modifying tracer to support RVFI. Imperas to adding RVFI -> RVVI-VLG)	SystemVerilog Dolphin/ OpenHW modifying existing test bench	ImperasDV (ref. model + simulator + verification IP)
Ibex (Imperas only)	RVFI (from Ibex) (Imperas added RVFI -> RVVI-VLG)	SystemVerilog or C++	ImperasDV (ref. model + simulator + verification IP)
E40X (Imperas only)	RVFI + RVVI-VLG	SystemVerilog	ImperasDV (ref. model + simulator + verification IP)

Proposed NXP, Intrinsix, Imperas, OpenHW Collaboration on CV32E20:

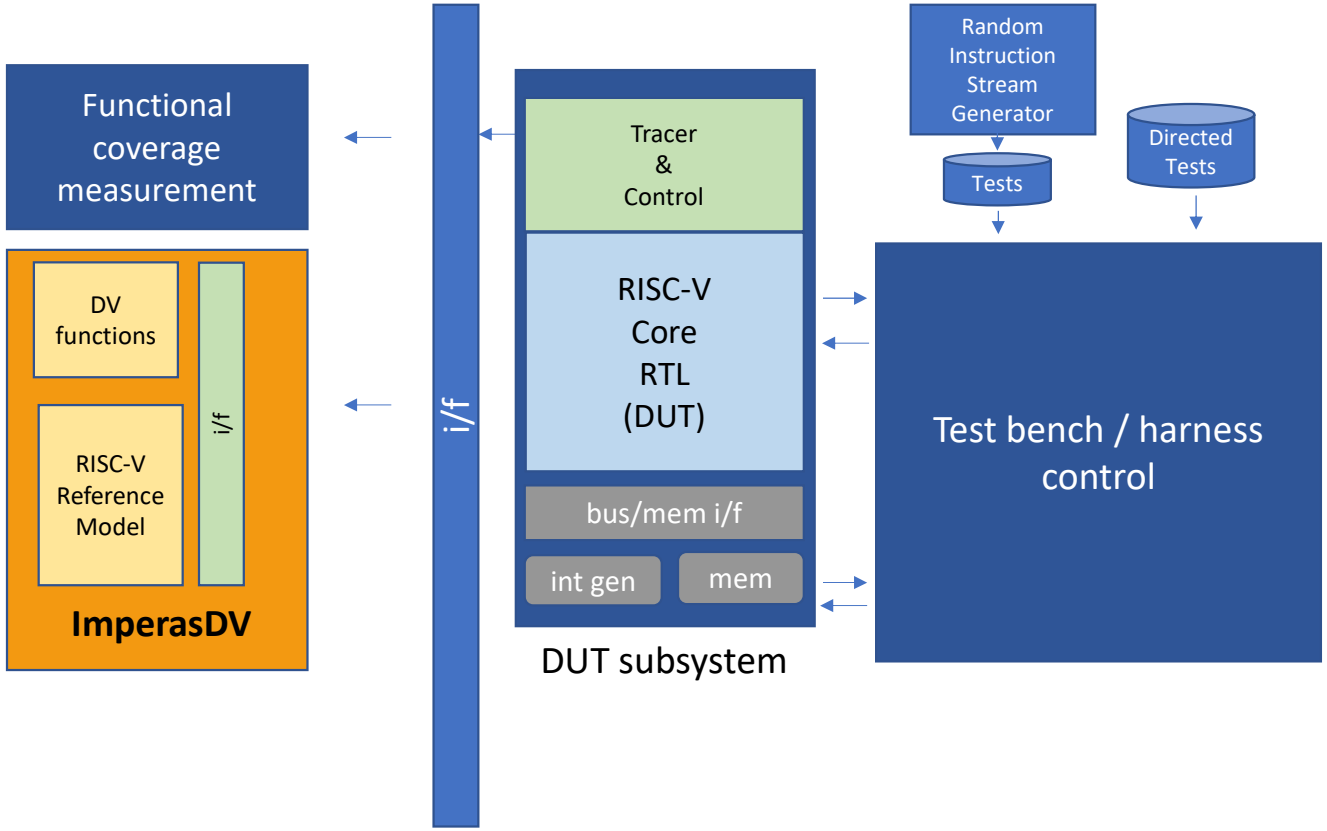
- Imperas would provide the reference model/simulator, plus verification IP (ImperasDV product)
- Could potentially build on Imperas internal development work on DV environment for Ibex core
  - Key pieces needing development are RTL tracer, test bench, E20 model

RISC-V Processor “Simulation” Environment



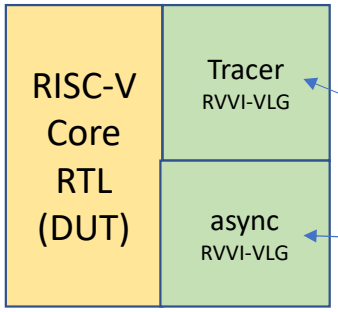
- Basic simulation environment
- Runs programs
- Directed – maybe self checking tests
- Requires ‘tracer’ for functional coverage and file logging
- Maybe measure functional coverage

RISC-V Processor “DV” Environment (includes reference subsystem)

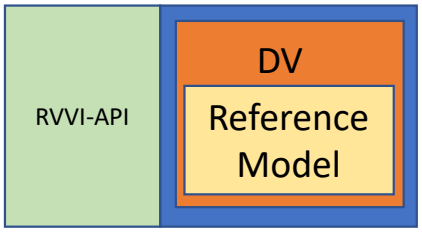


- Includes ref. model subsystem
  - Runs as Verification-IP
  - Passively monitors activity (piggy-back)
  - Compares in lock-step with all processor activity
- Requires ‘tracer’ interface from RTL
- No change to existing testbenches
- With ref. model compare – can use random tests (as no need for known good result as comparing to ref. model every event)

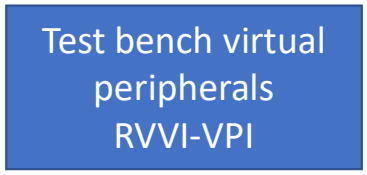
RVVI: RISC-V Verification Interface (driven by RISC-V DV usage)



- https://github.com/riscv-verification/RVVI (Public Open Standard)
- RVVI-VLG
  - Verilog DUT interfaces
    - RVVI-VLG state – streaming 'tracer' data
    - RVVI-VLG nets – implementation dependent (Interrupts, Debug)
    - Handles multi-hart, multi-issue, Out-of-Order

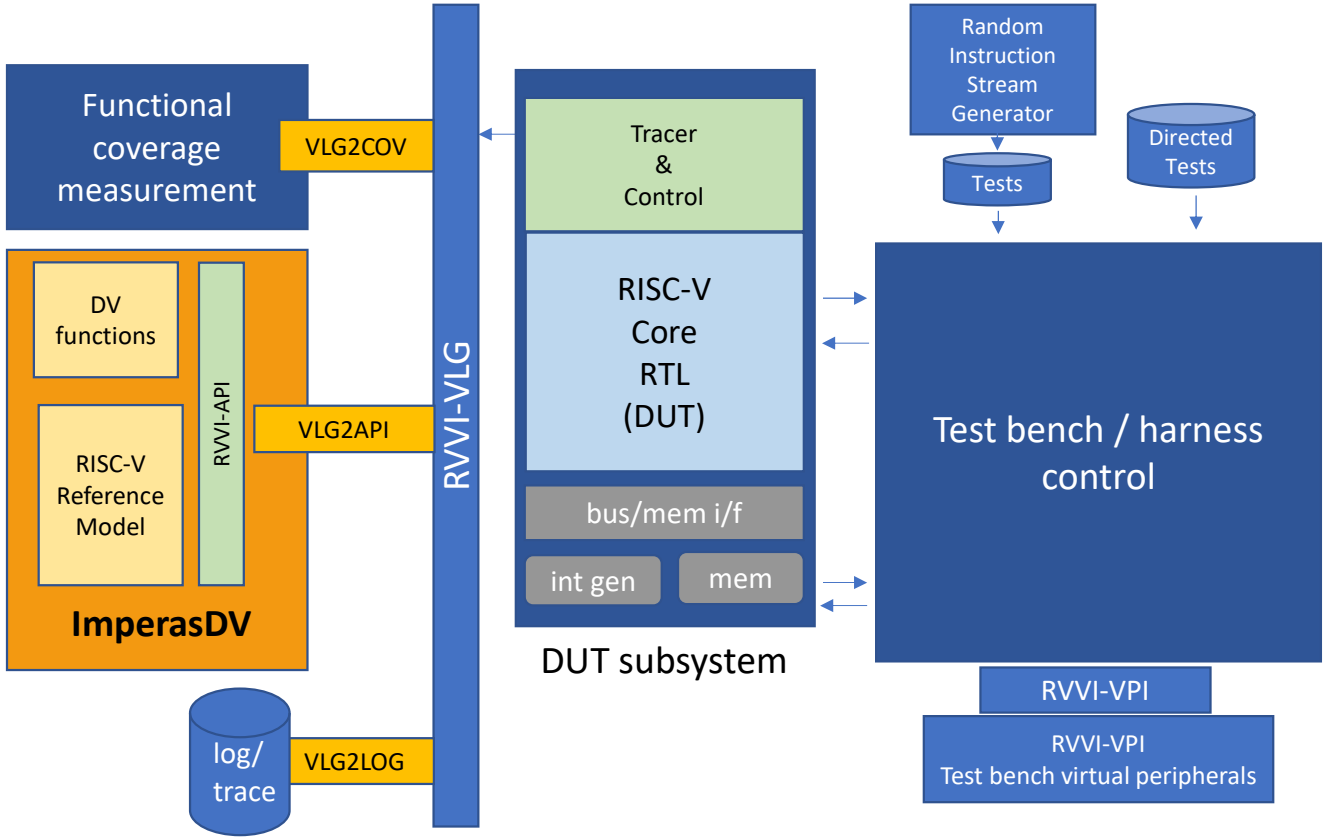


- RVVI-API
  - Controls DV subsystem and reference model
    - C/C++
    - SystemVerilog



- RVVI-VPI (work-in-progress (Mar. 2022))
  - Virtual Peripheral Interfaces
    - timers, interrupts, debug, random, printer/uart, …
  - Verilog and C macros & examples

RISC-V Processor "DV" Environment (using RVVI for flexibility and reuse)



- Using RVVI allows reuse of verification components
- Imperas provides source of interfaces from RVVI to verification components
  - Ref. model / DV
  - Functional coverage
  - Log file writer
- ImperasDV
  - 'works-out-of-the-box'
  - Just configure ref. model and DV capabilities