SYNOPSYS®
Silicon to Software™

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

AMD

Extending the RISC-V Verification Interface for Debug Module Co-Simulation

Lee Moore, Aimee Sutton, Synopsys Inc.

Michael Chan, Ravi Shethwala, Richa Singhal,
Advanced Micro Devices Inc.

accellera
SYSTEMS INITIATIVE

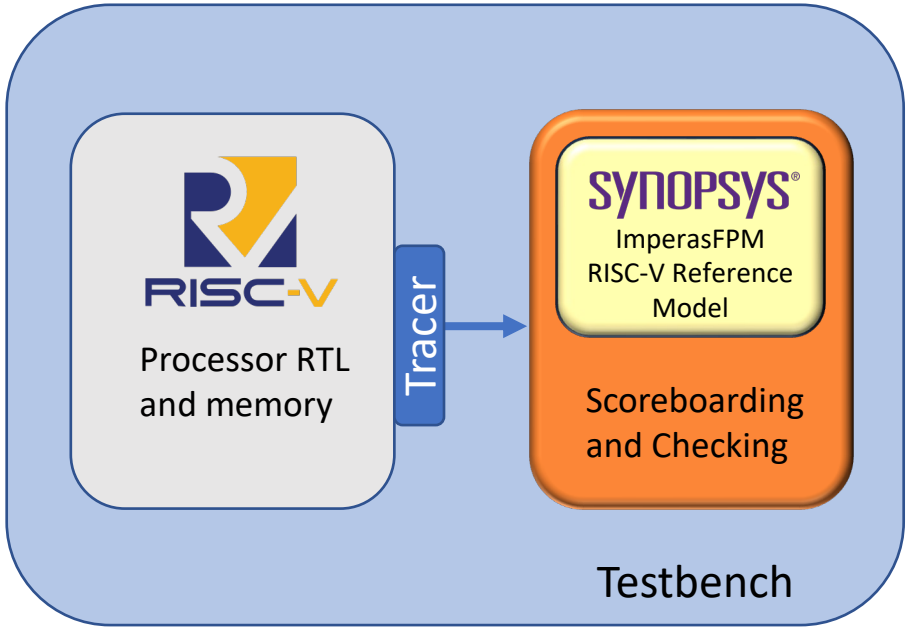
Agenda

- What is the RISC-V Verification Interface (RVVI)?
- What is a debug module?
- AMD's verification challenges
  - Random entry and exit to/from debug mode
  - Verification of debug module abstract commands
- Proposed changes to RVVI
  - debug_mode signal
  - debug interface
- Benefits of adopting RVVI

RISC-V Processor Verification

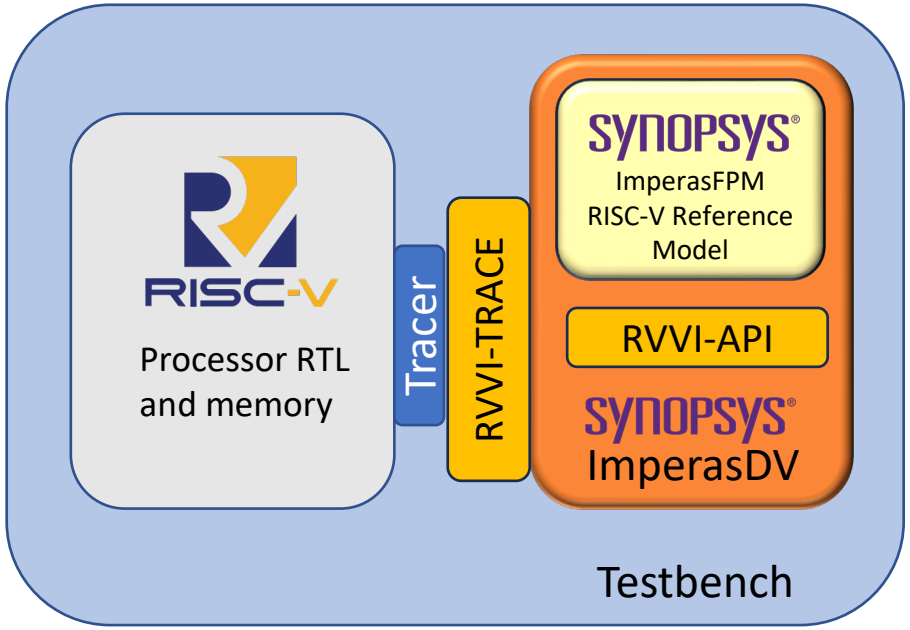
Basic DV environment architecture:

- RISC-V processor RTL and memory
- Tracer to extract DUT state for verification
- Processor reference model
- Testbench component for scoreboarding and checking



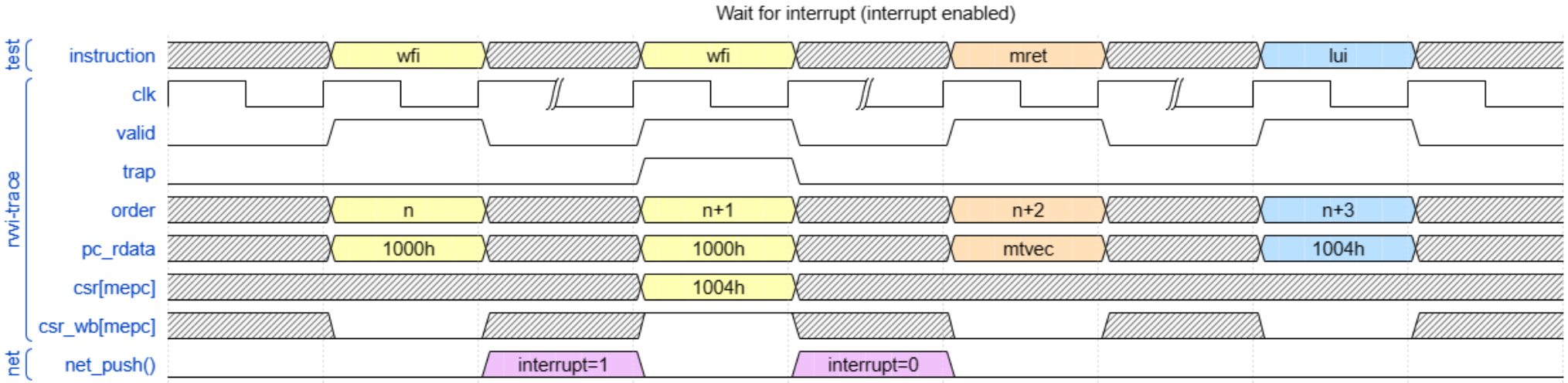
RISC-V Processor Verification using RVVI

- RVVI = RISC-V Verification Interface
  - https://github.com/riscv-verification/RVVI
- Open standard: result of collaboration between industry and open-source
- Standardizes communication between testbench and RISC-V VIP
  - **RVVI-TRACE**: interface between tracer and testbench
  - **RVVI-API**: interface between RISC-V verification component and reference model



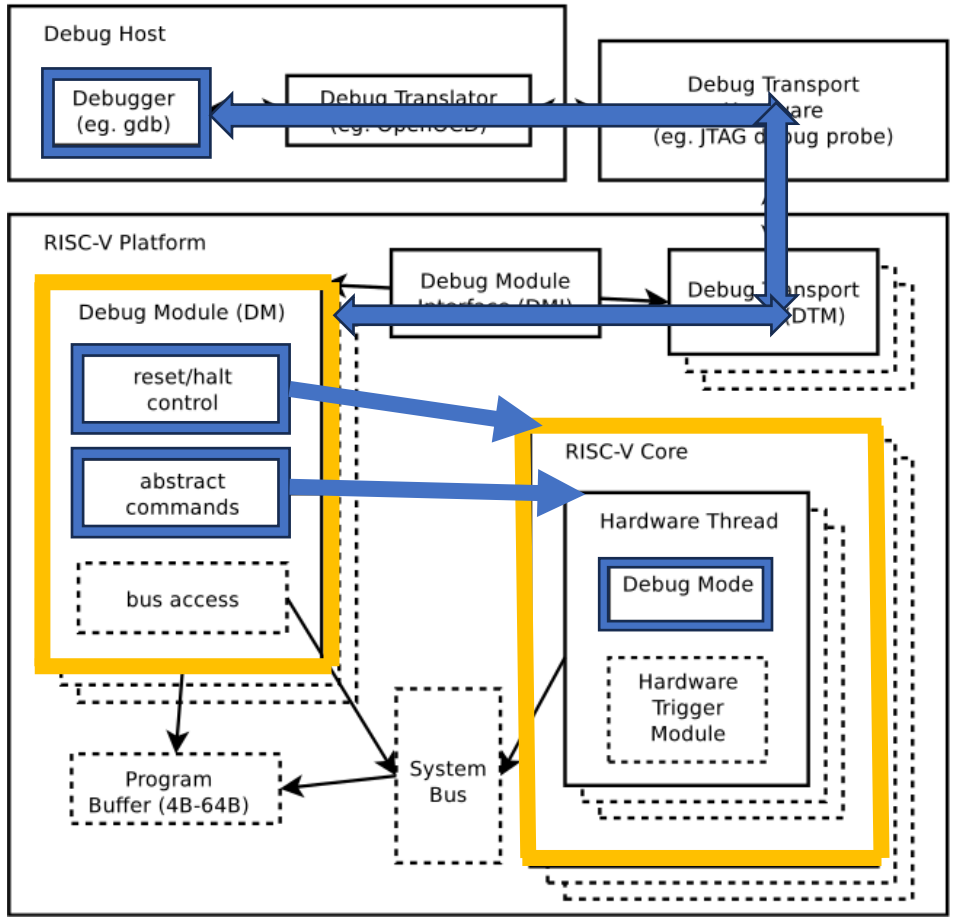
RVVI-TRACE

- Specifies information needed for comprehensive processor DV
- Tracer extracts this info and sends it to the testbench
- Defined as a SystemVerilog interface
- Includes functions for handling asynchronous events



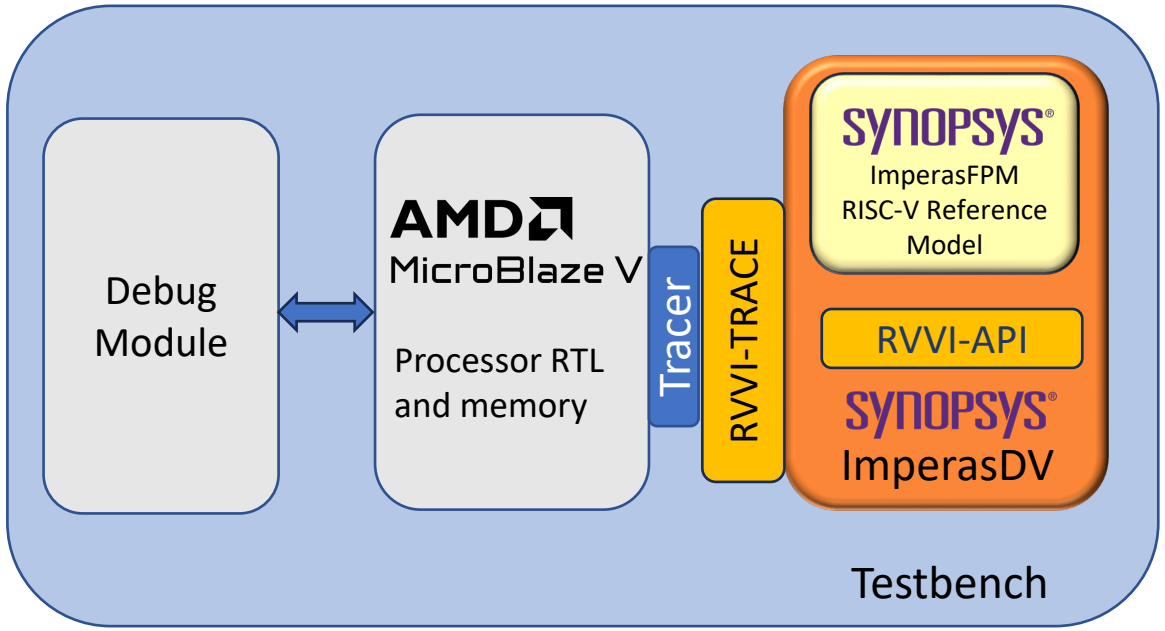
RISC-V Debug Module

- Debug module provides visibility into state of core to enable debug of hardware and software
- Debug module can put the core into a halted state
  - read and write registers and memory using abstract commands
  - execute arbitrary programs



RISC-V Processor + Debug Module Verification

- Verification goal:
  - debug mode entry / exit generated using constrained-random stimulus
  - GPR reads and writes are carried out correctly using abstract commands
  - arbitrary programs are executed correctly using the program buffer



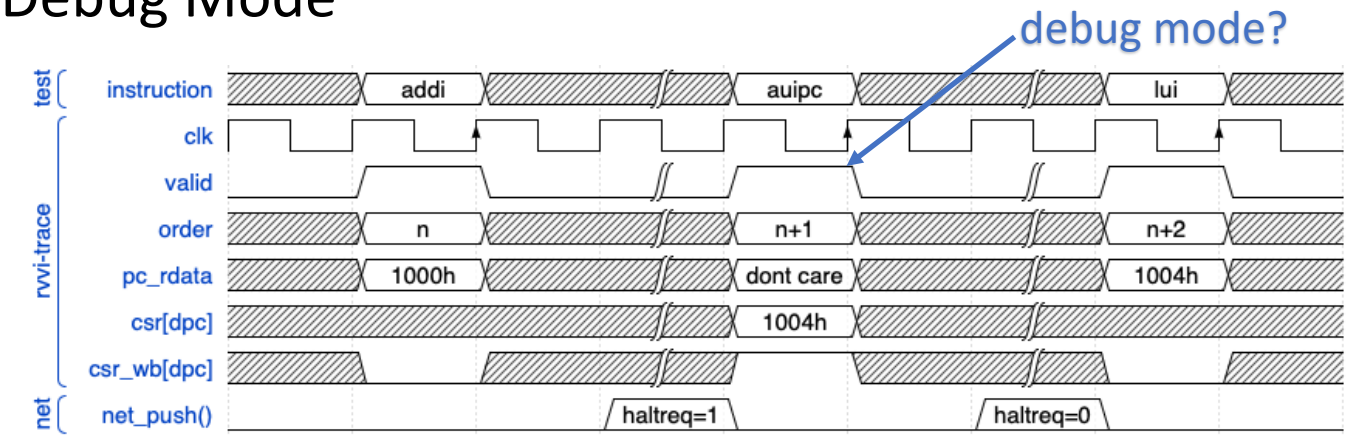
Debug Mode Verification challenges

Recognizing Debug Mode entry and exit

Verification of Debug Module abstract commands

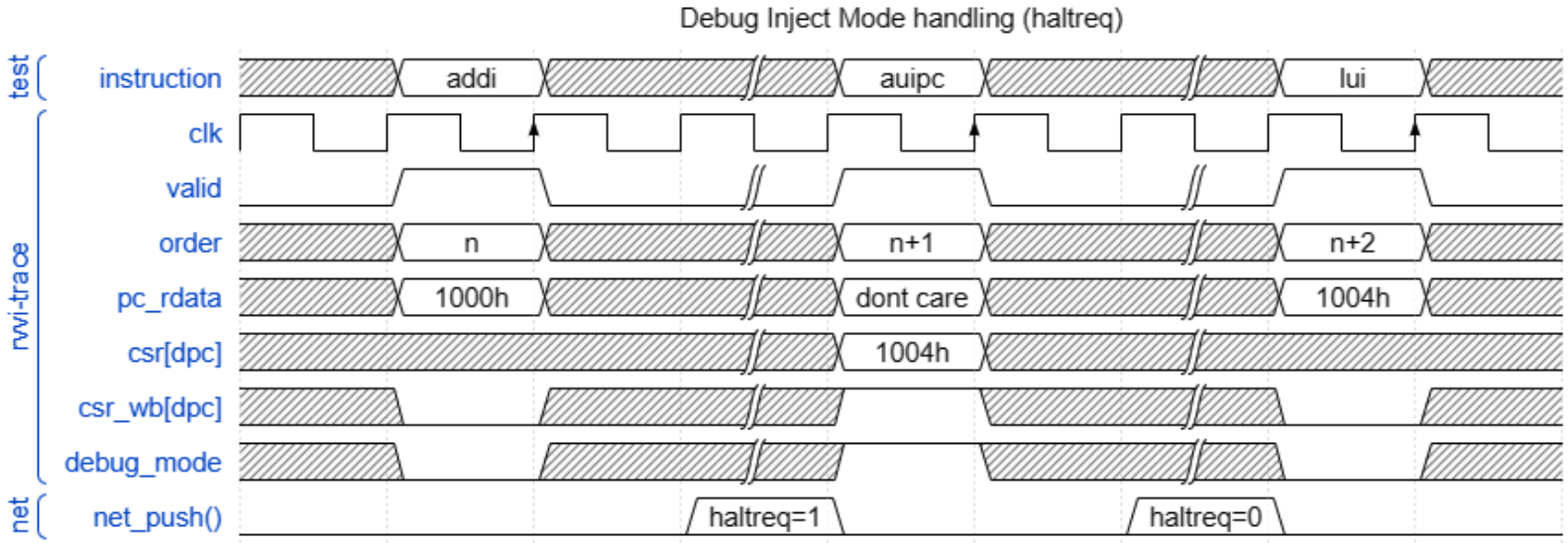
Recognizing Debug Mode entry/exit (1)

- Problem:
  - haltreq signal asserted randomly during program execution
  - debug program instructions injected directly into pipeline
  - impossible to determine whether or not an instruction is being retired in Debug Mode



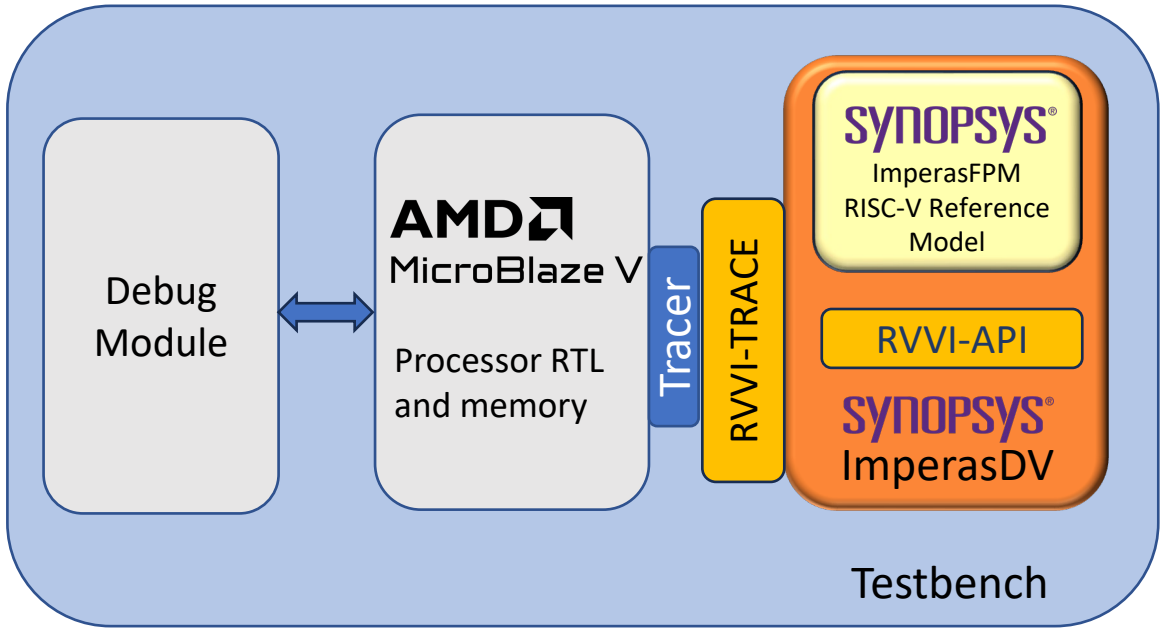
Recognizing Debug Mode entry/exit (2)

- Solution: debug_mode signal added to RVVI-TRACE
- Asserted by tracer for every instruction retired in Debug Mode



Verifying abstract command execution (1)

- Problem:
  - Verification environment has no visibility of the interface between debug module and processor
  - Reference model has the information needed (e.g. register values) to verify abstract commands
  - Changes to RVVI and ImperasDV are required



Verifying abstract command execution (2)

- Solution:
  - Debug module interface added to RVVI-TRACE
  - Tracer is updated to inform testbench about read/write accesses to debug module registers
  - DM registers modeled in the *store* array

```
interface dm;

wire    clk;    // Interface clock
wire    rd;     // read
wire    wr;     // write
wire [31:0]     address;
wire [31:0]     data;

// Storage for DM registers
bit  [(XLEN-1):0]  store      [127:0];

endinterface
```

Future work

- Support other abstract commands
  - quick access
  - access memory
- Update the public RVVI standard

Benefits of adopting RVVI

- RVVI-TRACE:
  - tracer requirements are known upfront
  - enables re-use in tracer
  - enables use of RVVI-compliant RISC-V verification solutions
    - e.g. Synopsys ImperasDV, riscvISACOV (functional coverage)
  - benefit from experience of others == best practices
- RVVI-API:
  - requirements for processor verification are specified
  - enables the creation of reusable RISC-V processor verification solutions

Questions

- Thank you!

Aimee Sutton aimees@synopsys.com

Lee Moore moore@synopsys.com

Michael Chan, Ravi Shethwala, Richa Singhal