SIEMENS

Efficient debug and trace of RISC-V systems: a hardware/software co-design approach

Oana Lazar, Henrique Mendes & Angelo Maldonado-Liu, Tessent Embedded Analytics, Siemens EDA

Debugging Challenges

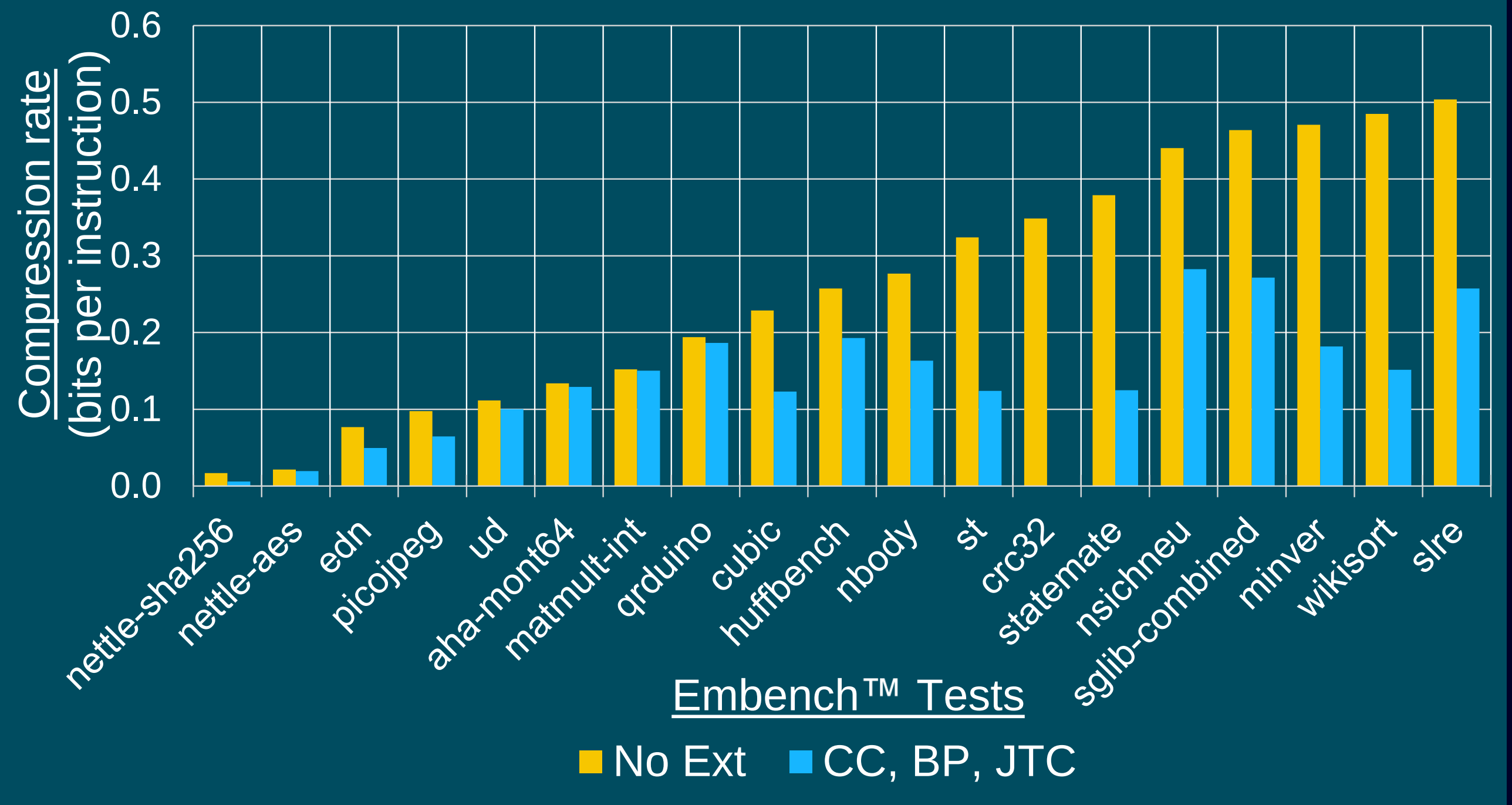
Increasingly complex RISC-V systems reveal increasingly complex issues, such as Heisenbugs and Silent Data Corruption. Existing debugging solutions are intrusive and lack the depth of visibility needed to root-cause timing-sensitive bugs. There is demand for an out-of-the-box solution providing multiple debug methods and insights, for increased debugging efficiency of RISC-V systems. Combining hardware and software provides different levels of visibility in an end-to-end solution.

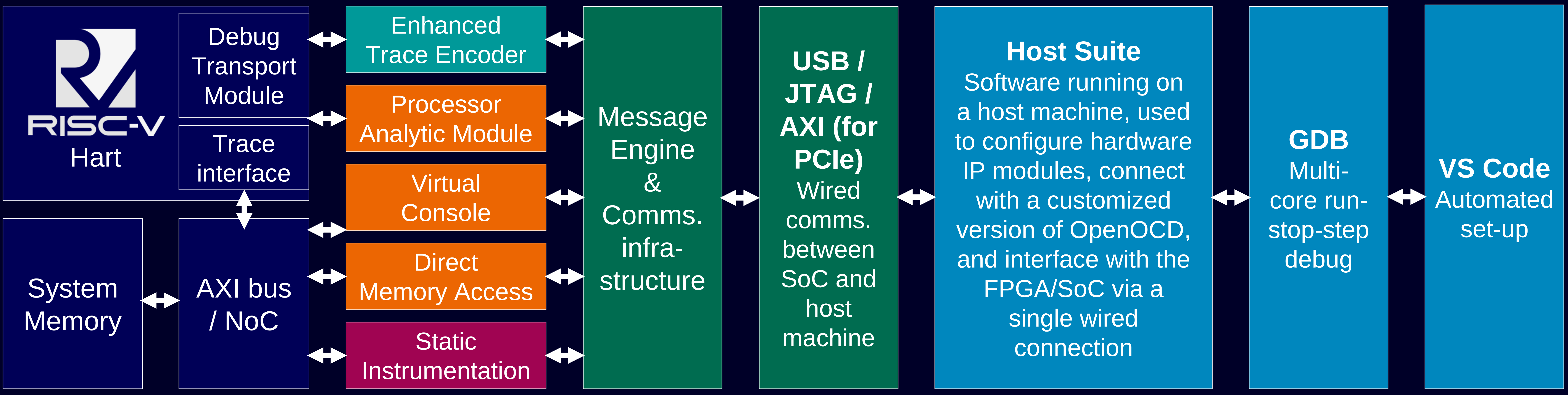
Highly Efficient Trace of Custom Instructions for In-Depth Debugging Visibility

The approach uses an Enhanced Trace Encoder (ETE) to non-intrusively and efficiently trace any RISC-V program's execution path. The ETE is fully compliant with the ratified "Efficient Trace for RISC-V (E-Trace)" specification, with out-of-the-box custom instruction support.

The ETE boasts an average instruction trace compression rate of 0.2317 bits per instruction with no extensions, benefiting large systems with heavy workloads, where bandwidth may be limited.

Additional extensions such as Call Counter (CC), Branch Prediction (BP), and Jump Target Cache (JTC) improve this rate by up to 40%. These allow for more efficient tracing of program execution, to pinpoint the start of unexpected behaviours and help root-cause complex bugs.





RISC-V Hart

Debug Transport Module

Trace interface

System Memory

AXI bus / NoC

Enhanced Trace Encoder

Processor Analytic Module

Virtual Console

Direct Memory Access

Static Instrumentation

Message Engine & Comms. infra-structure

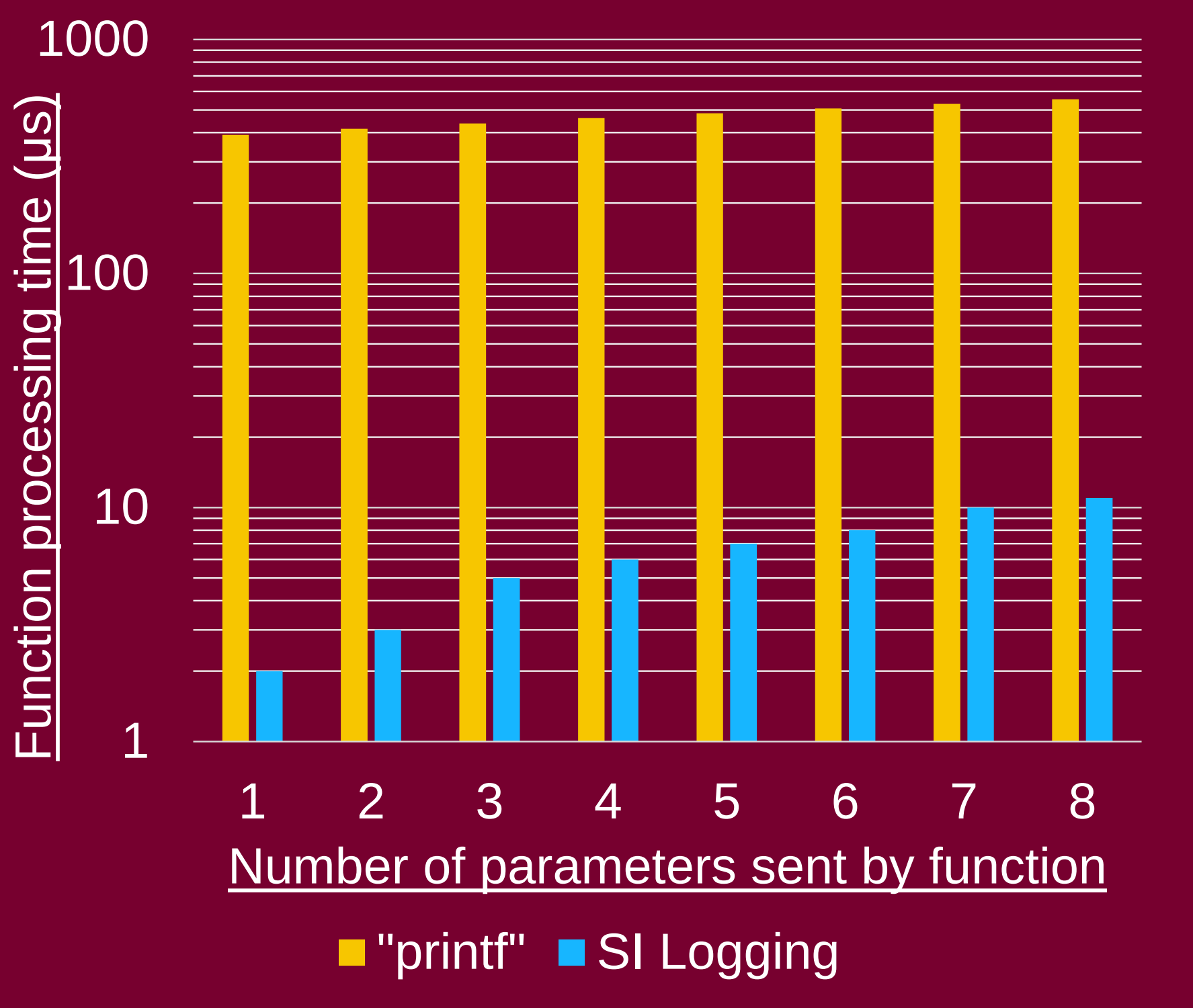
USB / JTAG / AXI (for PCIe) Wired comms. between SoC and host machine

Host Suite Software running on a host machine, used to configure hardware IP modules, connect with a customized version of OpenOCD, and interface with the FPGA/SoC via a single wired connection

GDB Multi-core run-stop-step debug

VS Code Automated set-up

Minimally Intrusive Program Flow Logging



Run-control debuggers and "printf-style debugging" can interfere with program execution order and timing, making it easy to miss timing-sensitive Heisenbugs. The approach uses a Static Instrumentation (SI) module for minimally intrusive logging of program flow, via intuitive "printf"-like APIs:

```
UST_LOGGING_DEBUG_PARAM3("Pixel write y=%d, x=%d,
                          colour=%u", y, x, colour);
```

FPGA implementations of SI show its logs only take 0.2-0.3% of the processing time that their "printf" equivalents take to run, minimising interference and increasing the chances of catching timing-sensitive bugs.

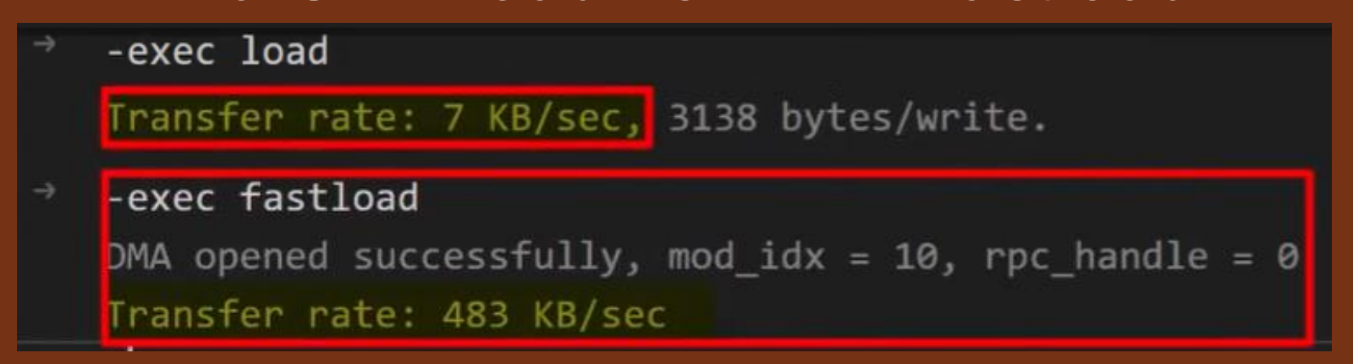
Harnessing Hardware/Software Benefits

Dedicated hardware is faster at completing specific tasks, at the cost of silicon area. Inversely, software can be leveraged to mitigate this cost.

The approach maximizes the benefits of both hardware and software, with highly-configurable hardware IP providing flexibility for various silicon area requirements:

- Virtual Console module: uses software multiplexing to provide access to multiple "virtual" bi-directional off-chip communication channels, without needing additional pins as in a hardware-only solution.

ELF file GDB load vs DMA "fastload":

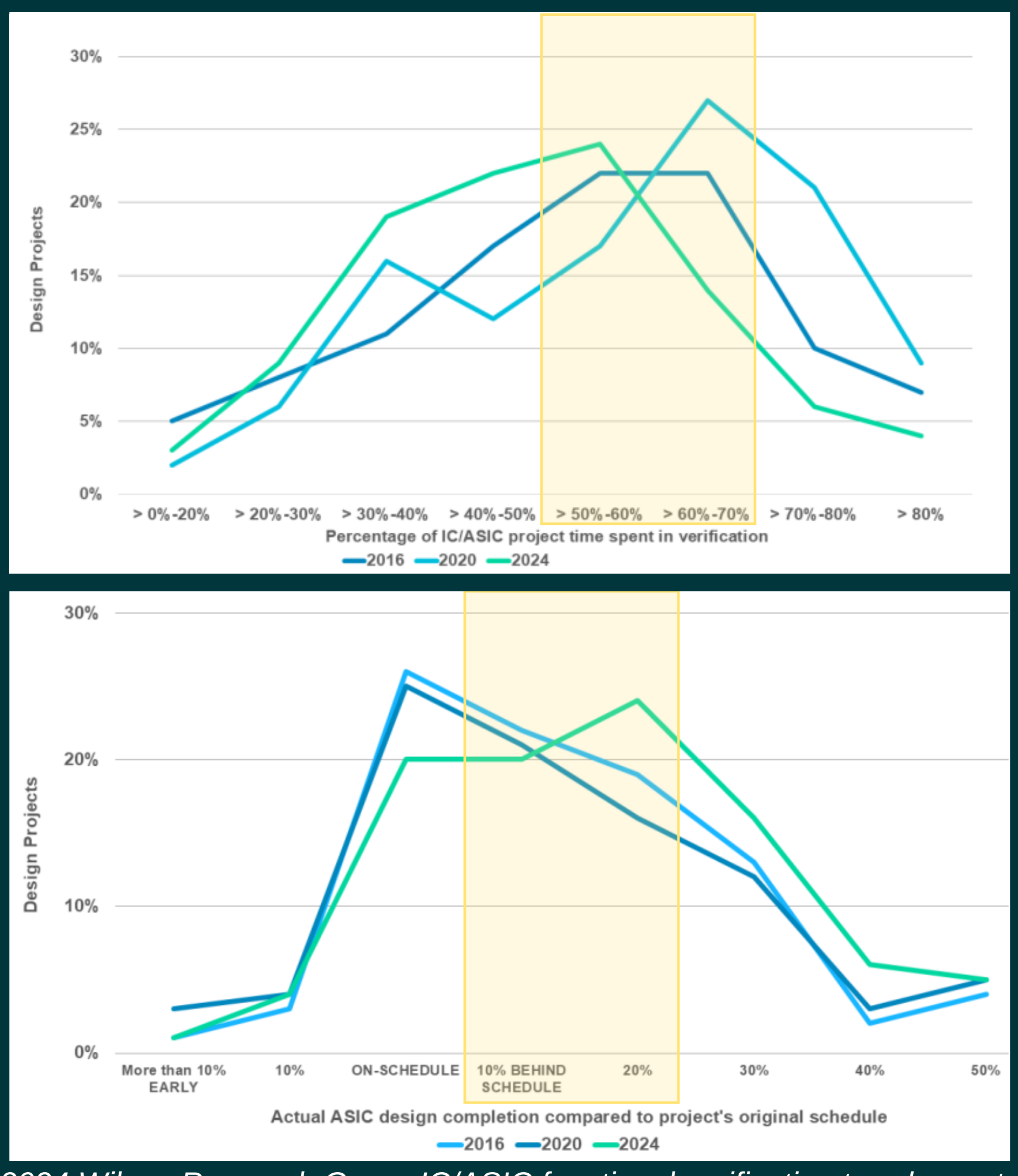


```
-exec load
Transfer rate: 7 KB/sec, 3138 bytes/write.
-exec fastload
DMA opened successfully, mod_idx = 10, rpc_handle = 0
Transfer rate: 483 KB/sec
```

- Direct Memory Access module: offers a 100x speed-up in on-chip ELF file loads over software-only GDB loads.
- Processor Analytic Module: allows RISC-V harts to be debugged via a single wired connection to a host.

UVM System Verification

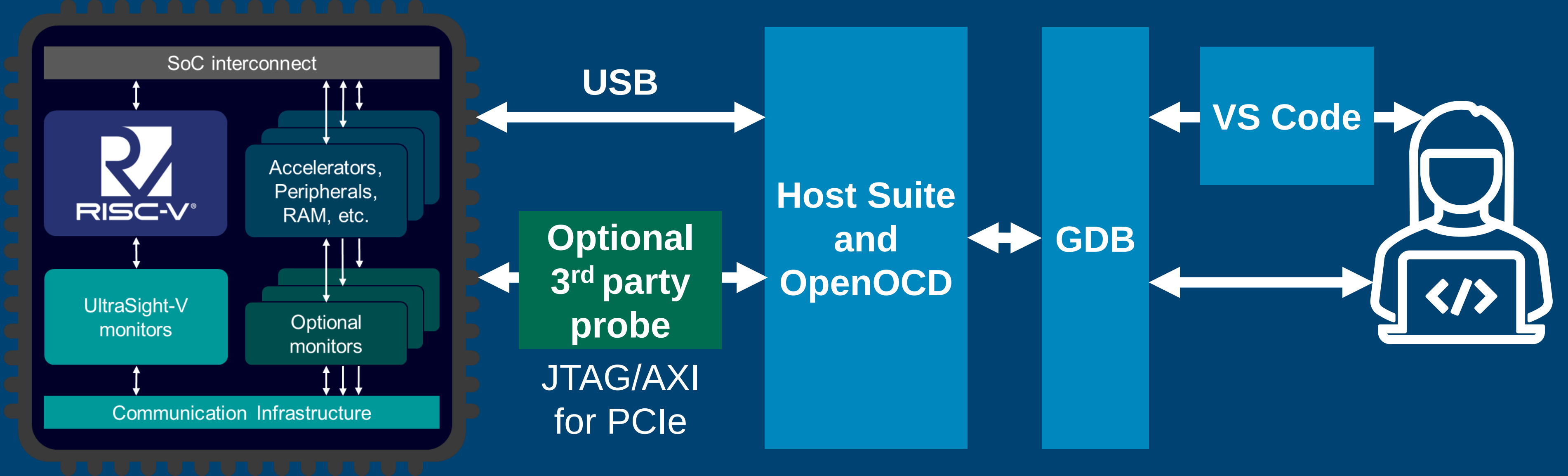
Almost 50% of design engineers' time is spent on verification instead of design, potentially increasing delays. The approach includes a UVM environment for driving stimuli to system communicators, to easily verify the integration of hardware modules within RISC-V systems. This prepackaged solution allows engineers to refocus their time back to designing.



2024 Wilson Research Group IC/ASIC functional verification trend report

Tessent UltraSight-V: A Complete End-to-End Debug and Trace Solution

FPGA and SoC implementations demonstrate that approaches using hardware and software to their full extent can provide efficient and minimally intrusive RISC-V debug and trace. This approach is implemented by UltraSight-V, a comprehensive end-to-end solution for debug, trace, and testing of any RISC-V system, maximising developers' usage of customizations and extensions and out-of-the-box. Optional system-level functional monitors can be added to the scalable messaging fabric, with these pre-verified IPs being silicon-proven at 3nm and 3GHz. A provided UVM environment verifies hardware modules' integration within a RISC-V system. UltraSight-V is compatible with 3rd party tools including: GDB, VS Code, Lauterbach TRACE32®.



SoC interconnect

RISC-V

Accelerators, Peripherals, RAM, etc.

UltraSight-V monitors

Optional monitors

Communication Infrastructure

USB

Optional 3rd party probe

JTAG/AXI for PCIe

Host Suite and OpenOCD

GDB

VS Code

UltraSight-V features:

Code debugging

- Multi-core run/stop control
- Fast memory access
- Instrumentation

Efficient trace

- Highly-compressed instruction trace (E-Trace)
- Custom instruction support

Interface options

- USB and JTAG
- AXI (for PCIe access)
- Lauterbach TRACE32®

Software toolkit

- Host Suite with OpenOCD
- GDB compatibility
- VS Code IDE support
- UVM Verification IP

Acknowledgement of contributions

Special thanks to:

- Dr. Francisca Tan[1], Senior Product Manager
- Rod Boyce[1], Principal Software Engineer
- Dr. Luke Southwell[1], Software Engineer
- Marcel Zak[1], R&D Software Engineer
- Iain Robertson[1], Senior Engineering Director
- Harry Foster[2], Chief Scientist Verification

1: Tessent Embedded Analytics, Siemens EDA
2: Siemens AG

RISC-V SUMMIT EUROPE 2025