Unified Emulation and Simulation Debug Environment for RISC-V Devices to Reduce Cost and Turnaround Time

Pravin Tavagad, Sharanesh R, Sajosh Janarthanam, Himanshu Suri, Martin Hannon, Rejeesh SB

Ashling Microsystems, Tenstorrent Inc.

Abstract

With increasing design complexity, validating hardware design and software debugger well before tape-out is essential. Traditional methods for hardware verification, software debugger bring-up, and hardware-software compatibility are performed on emulation platforms such as Zebu from Synopsys or Palladium from Cadence using JTAG debuggers. Emulation, while effective for bug discovery with high turnaround time, offers limited debug visibility compared to simulation. Dependency on physical JTAG Debugger caps the number of users to hardware availability. This paper describes mechanism to use the same software debugger on simulation environment too and remove the use of physical hardware debugger. The HW to SW communication utilizes a synthesizable JTAG transactor integrated into the Emulation environment, which interfaces seamlessly with the software debugger via TCP/IP sockets (or any other inter-process communication protocol). This enables issue of software debugger commands to HW, as part of use-case stimuli. The collaterals for communication and stimuli are used in the simulation environment too, facilitating the rerun of emulation failures in simulation platform for bug resolution.

Introduction

Successful post-silicon debugging relies on a hardware and software infrastructure designed to support post-silicon debug use cases. To ensure readiness, post-silicon debug use cases must be thoroughly validated in a pre-silicon environment using the complete hardware and software stack well ahead of chip tape-out.

RISC-V Debug WG defined specification**Error! Reference source not found.** lists support for halting CPUs, accessing registers and memory, defining triggers for breakpoints, and tracing instruction retires. These features are exercised via JTAG debugger from a host CPU running software debugger.

Traditional methods for hardware design and software debugger verification on emulation platforms rely on host-target communication using JTAG debuggers. This paper presents a verification strategy which enables HW/SW verification across simulation and emulation platforms to leverage inherent advantages of each platform. The verification strategy eliminates the need for physical JTAG debuggers, enabling seamless scalability by removing reliance on physical component for hardware-software debugger communication.

Pre Silicon Verification Requirements

The RISC-V debug ecosystem is evolving, with the debug specification formally adopted in 2024. The hardware and associated software debugger supporting this specification represent first-generation implementations. As these implementations are in their early stages, they present challenges such as bugs, specification misinterpretations, and hardware-software incompatibilities. To ensure compatibility and accelerate bug detection, the pre-silicon verification environment must simulate post-silicon use cases using the software stack. Validating these use cases across multiple hardware configurations, along with functional traffic, is critical to minimizing verification gaps. To expedite bug resolution, the pre-silicon platform should enable short build times for reruns with bug fixes. Additionally, the verification environment must scale effectively, enabling geographically dispersed teams of verification engineers to run tests remotely.

Emulation platforms excel in bug discovery with high cycles-per-second (CPS) and debugging capabilities but face longer build times and limited visibility, challenging for immature designs. Simulations offer better visibility and shorter build time but struggle with bug discovery using the software stacks due to low CPS.

To address pre-silicon verification requirements effectively, it is crucial to develop a strategy that combines simulation and emulation platforms, maximizing the unique strengths of each. An effective verification strategy must enable bug discovery on emulation and bug reproduction in simulation while preserving the hardware-software stack and stimuli across the platforms. All verification components and stimuli must be compatible for both simulation (E.g., Synopsys VCS) builds and emulation (E.g., Synopsys Zebu) builds.

Here is a typical emulation setup for debug verification with JTAG debugger.

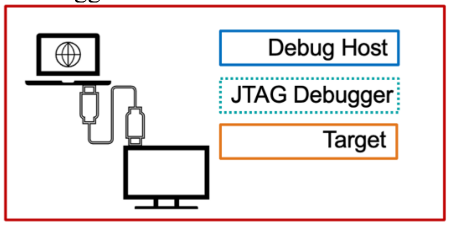

Figure 1: Emulation Setup with Physical JTAG Debugger

The following aspects of traditional emulation platform impede a seamless verification strategy across simulation and emulation platforms.

a) Scale: Host to target communication is over a physical JTAG debugger. The maximum number of simultaneous users is capped by availability of physical JTAG debuggers.
b) Reuse: Software debugger commands from the host are part of the stimuli. However, there is no direct path to replicate these use-cases in a simulation environment, making it difficult to maintain consistent stimuli across platforms - a critical requirement for using simulation as a bug reproduction and root-cause analysis platform.

Scalable Platform Agnostic Testbench Environment

The testbench, designed to fulfil pre-silicon verification requirements while overcoming the limitations of traditional emulation setups, incorporates two key features: the use of TCP/IP sockets for cross-program communication and the utilization of command-line interfaces (CLI) and startup scripts to drive software use-case stimuli. TCP/IP sockets, along with supporting software and verification environment features, replace JTAG debugger functionality, eliminating physical bottlenecks associated with JTAG debuggers. Commands from the CLI and startup scripts are consistently used across both simulation and emulation platforms, enabling root cause of emulation failures in the simulation environment.

The testbench architecture is designed to be independent of the target implementation. It supports both the simulation and emulation builds.

Results and Conclusion

Post Silicon use-cases were exercised on Platform Agnostic testbench. To discover bugs, emulation build was used as the target in the test bench. Failing emulation tests with incorrect hardware responses in logs were rerun using the simulation build. Simulation platform provided faster path to debug with better design visibility.

Software debugger use-case stimuli, along with functional traffic in the scalable platform agnostic environment uncovered implementation bugs, HW-SW compatibility issues, and spec misinterpretations. Using common stimuli for simulation and emulation leveraged the strengths of both platforms. Emulation enabled effective bug discovery, while simulation expedited root cause analysis and resolution. Using TCP/IP sockets for software-to-hardware communication eliminated the need for physical JTAG Debuggers, removing user limitations tied to debugger availability.

References

[1] RISCV Debug Spec: version 1.0.0.