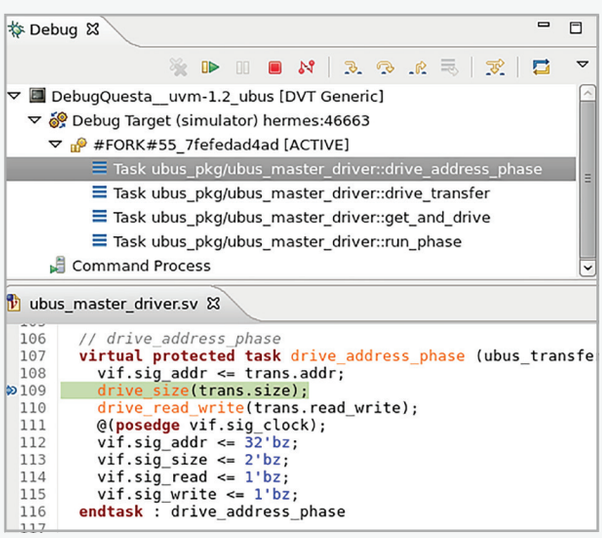
DVT
Debugger

- For SystemVerilog, Verilog, Verilog-AMS, e, and VHDL

Simpler and faster code debugging



BENEFITS

- Enables engineers to debug from the same place where they write their code
- Supports commonly used debug operations such as adding breakpoints, stepping, moving up and down the call stack, and changing values with a single click
- Minimizes the need for explicit printing or other additional commands because the run-time context is automatically fetched from the simulator
- Makes the call stack and local variables available for analysis anytime the simulator hits a breakpoint
- Provides all DVT Eclipse IDE features that help users navigate and understand the code: hyperlinks, hierarchy browsing, finding usages, tracing, and many others

OVERVIEW

Design and Verification Tools (DVT) Eclipse IDE is an integrated development environment for SystemVerilog, Verilog, Verilog-AMS, the e language, and VHDL. It helps your design and verification engineers increase the speed and quality of new code development, easily understand complex source code, simplify the maintenance of legacy code and reusable libraries, and accelerate language and methodology learning.

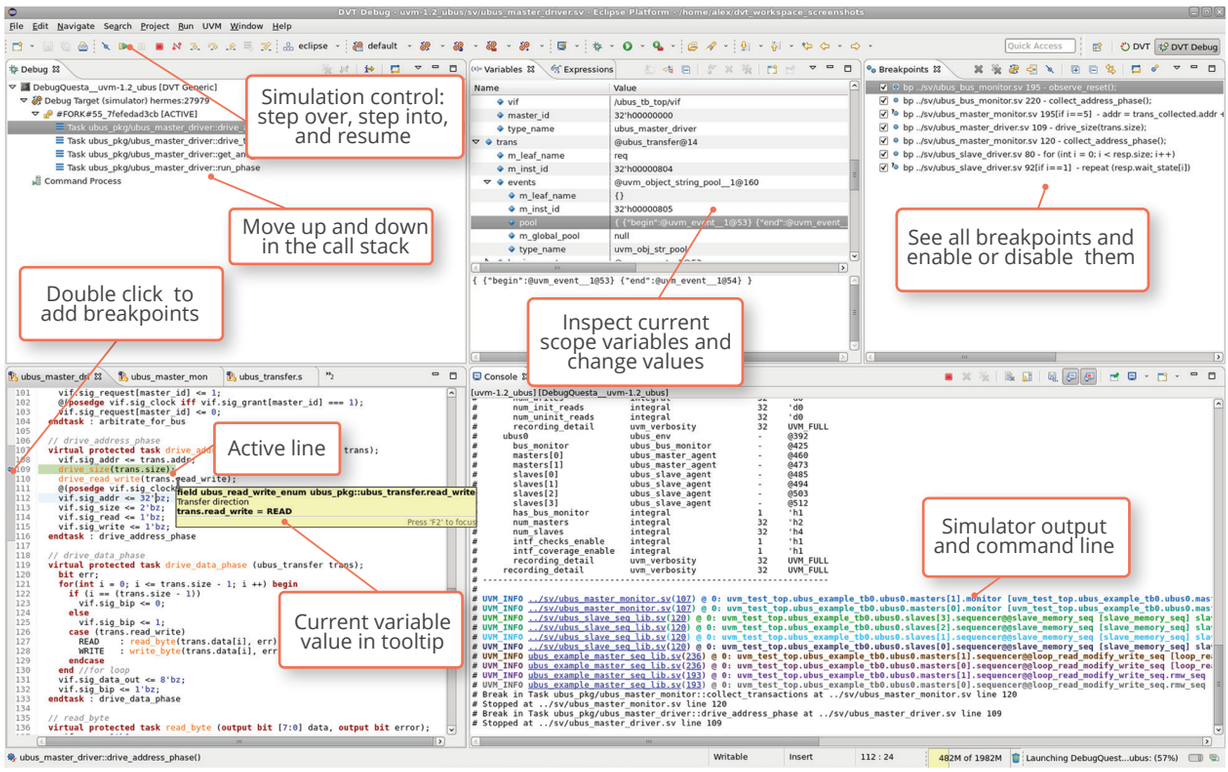
The IDE is built on the Eclipse Platform. It consists of an IEEE standard-compliant parser, a smart code editor with built-in Universal Verification Methodology (UVM) support, an intuitive graphical user interface (GUI), and a comprehensive set of features that help with code writing, inspection, navigation, and debugging.

DVT Debugger is an optional add-on module to DVT Eclipse IDE. It integrates with all major simulators and provides advanced debugging capabilities. It is unique because it allows users to perform debugging from the same place where they develop their code. It practically eliminates the need to continuously switch between the editor to understand the source code, and the simulator to inspect variable values, set, enable or disable breakpoints, or advance the simulation.

THE DVT DEBUG PERSPECTIVE

The DVT Debug perspective is a GUI layout focused on debug-specific activities. It provides simulation controls such as step over, step into, and resume. It also shows the simulation context in the editor and several dedicated views:

- Breakpoints View enables users to quickly inspect all breakpoints, enable or disable a specific breakpoint, and define conditional breakpoints
- Debug View allows users to move up and down the call stack where the simulator stopped
- Variables View displays the variables associated with the stack frame selected in the Debug View, including the arguments of the current function, locally declared variables, class members, and module signals, and allows users to change variable values at runtime
- Expressions View permits users to define and watch expressions
- Console View shows the simulation output and allows users to enter simulator commands



The DVT Debug Perspective

THE DVT EDITOR

From DVT Debugger's smart editor, your debugging becomes simple and fast. Users can add a breakpoint at a specific line simply by double clicking in the editor.

Whenever the simulator hits a breakpoint, the editor highlights the corresponding line.

The Debug View and the editor are always synchronized. In the Debug View, when the user moves up and down the call stack, the active line corresponding to the selected stack frame is automatically highlighted.

Users can quickly see a variable value in the tooltip by hovering over its name. They can also inspect a complex expression by selecting it in the editor, and then adding a watch to the Expressions View.

FLOW INTEGRATION

Deploying DVT Debugger requires minimal simulation flow changes. Users need to launch the simulation in debug mode and specify the communication library using simulator-specific arguments. DVT Debugger ships with configuration examples to help your deployment.

REMOTE DEBUGGING

The communication between DVT Debugger and simulator is done through network sockets. This allows users to connect to a simulation running on another machine, for example from a "GUI jobs" machine to a more powerful "batch jobs" machine that executes the simulation. Another common use case is connecting from a machine where the source code is available to a machine where the source code is encrypted.

TECHNICAL SUPPORT

The technical support team is available to promptly answer your questions, provide you with training, and work with you to determine your needs.

Your requirements and feedback are important. Whether you are looking for technical support or new features to improve your design and verification flow, AMIQ's technical support team strives to answer your requests in a timely manner.

CONTACT AMIQ

SUPPORT & EVALUATION
support@amiq.com

SALES
sales@amiq.com

WEBSITES
www.eda.amiq.com / www.amiq.com

Copyright 2025 AMIQ EDA S.R.L.
All rights reserved.

The information contained herein is subject to change without notice.

DBG-0125-A4