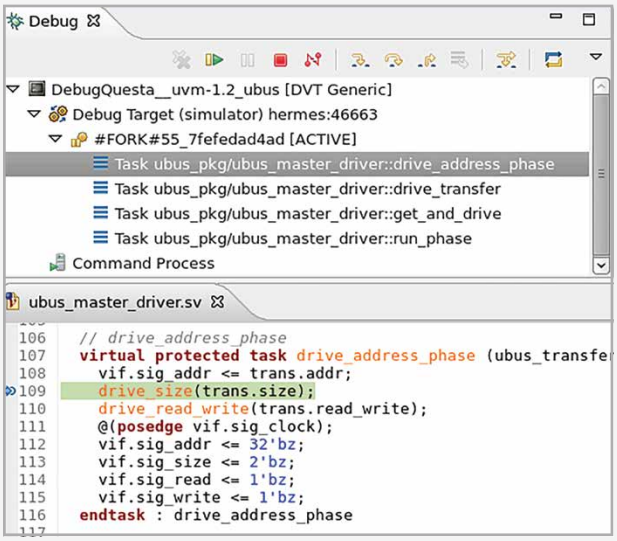
DVT
Debugger

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

Simpler and faster code debugging



BENEFITS

Enables engineers to debug from the same place where they write their code. Commonly used debug operations like adding breakpoints, stepping, moving up and down the call stack, or changing values are a click away in the DVT IDE.

Minimizes the need for explicit printing or other additional commands because the run-time context is automatically fetched from the simulator. The call stack and local variables are available to be analyzed anytime the simulator hits a breakpoint.

Allows users to take advantage of all DVT's features that help navigate and understand the code: hyperlinks, hierarchy browsing, finding usages or tracing, and many others.

OVERVIEW

Design and Verification Tools (DVT) is an integrated development environment (IDE) for the e language, SystemVerilog, Verilog, Verilog-AMS and VHDL. It helps 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.

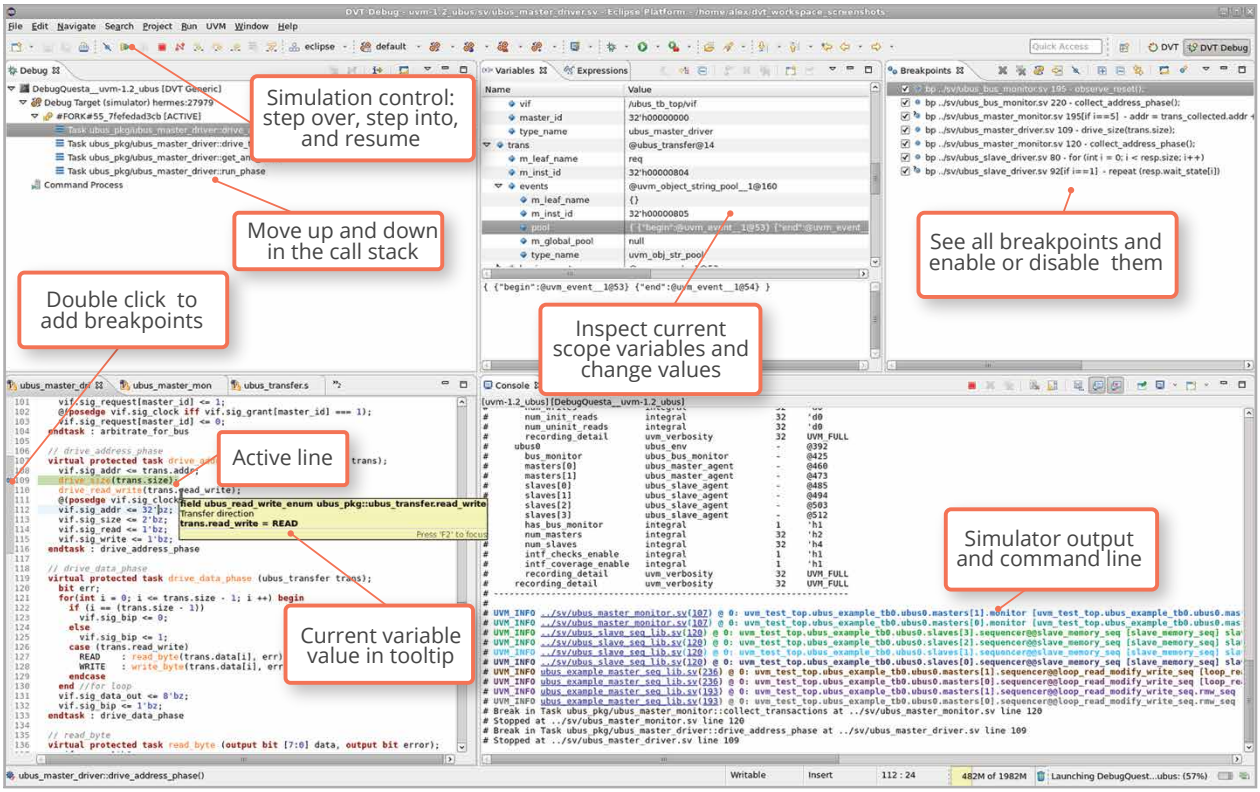
DVT is built on the Eclipse Platform. It comprises an IEEE standard-compliant parser, a smart code editor, an intuitive graphical user interface (GUI), and a comprehensive set of features that help with code writing, inspection, navigation, and debugging and provides UVM support.

The DVT Debugger is an optional add-on module to the DVT 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, to set, enable or disable breakpoints or to advance the simulation.

THE DVT DEBUG PERSPECTIVE

The DVT Debug perspective is a GUI layout focused on debug-specific activities. It provides simulation controls like step over, step into or 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 break point or 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 such as the arguments of the current function, locally declared variables, class members, and module signals. Users can change a variable value at runtime from this view.
- Expression View permits users to define and watch expressions.
- Console View shows the simulation output and it allows users to enter simulator commands.



The DVT Debug Perspective

THE DVT EDITOR

From the DVT's smart editor, debugging becomes simple and faster. Users can add a breakpoint at a specific line 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 the DVT Debugger requires minimal simulation flow changes. Users need to launch the simulation in debug mode and specify the DVT communication library using simulator specific arguments. DVT ships with debugger configuration examples to help your deployment.

REMOTE DEBUGGING

The communication between the DVT IDE 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 use case is to connect from a machine where the source code is available to a machine where the source code is encrypted.

Contact AMIQ

SUPPORT & EVALUATION
support@amiq.com

SALES
sales@amiq.com

WEBSITES
www.dvteclipse.com / www.amiq.com

DBG-0617-A4