DVT IDE
for Visual Studio Code

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

The complete development environment for hardware design and verification

10 REASONS TO CHOOSE DVT IDE for Visual Studio Code

1. See the errors flagged by incremental compilation as you type
2. Write code faster using autocomplete and error fix suggestions
3. Quickly move around in the source code using hyperlinks
4. Query the project database to accurately locate relevant information
5. Easily create and reuse code templates
6. Continuously improve your source code using refactoring operations
7. Easily understand the project structure using high-level views
8. Visualize the project architecture using UML and HDL diagrams
9. Trace signals throughout the design
10. Work smoothly with mixed language projects

BENEFITS

- Increases productivity and reduces time to market
- Speeds up source code development
- Enables efficient reading and understanding of complex source code
- Simplifies debugging and legacy source code maintenance
- Ensures higher quality source code development
- Streamlines code review
- Accelerates language and methodology learning

OVERVIEW

Design and Verification Tools (DVT) IDE for Visual Studio Code (VS Code) is an integrated development environment (IDE) for SystemVerilog, Verilog, Verilog-AMS, and VHDL. It is similar to well-known programming tools such as Eclipse®, NetBeans®, and Intellij@.

DVT IDE consists of a compiler, a smart code editor, an intuitive graphical user interface, and a comprehensive set of features that help with code writing, inspection, navigation, and debugging. It provides capabilities that are specific to the hardware design and verification domain, such as design diagrams, signal tracing, and verification methodology support.

DVT IDE is a powerful tool that allows your engineers to overcome the limitations of plain text code editors and address today's project complexity more efficiently. It enables faster and smarter code development and simplifies legacy code maintenance for novices and experts alike.

IEEE Standard Compliant Compiler

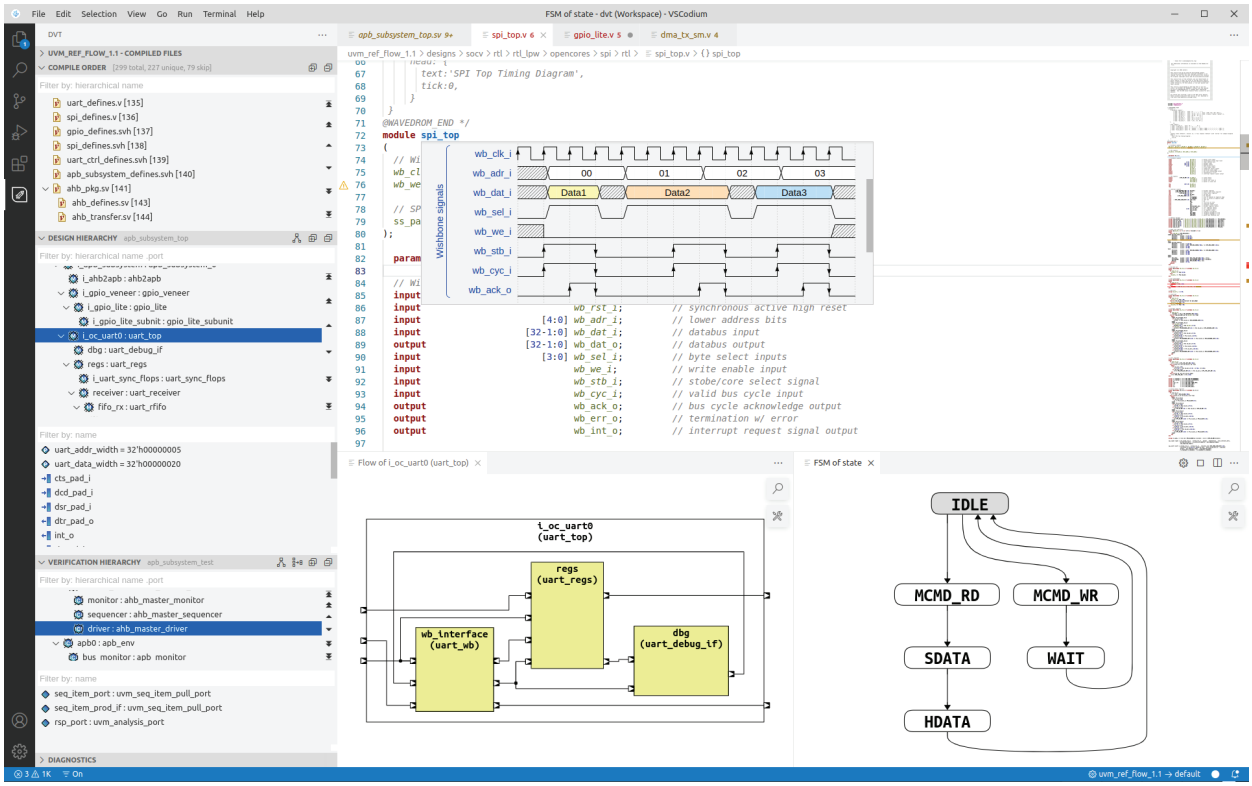
The compiler is compliant with the IEEE 1800™M SystemVerilog and IEEE 1076™ VHDL standards. Besides fully supporting the design and verification languages, DVT IDE also flags the use of non-standard compliant language constructs, which ultimately increases simulator compatibility.

Integration with Other Tools

DVT IDE works with revision control systems such as Git, Subversion, and ClearCase, and bug tracking systems such as Jira™.

Visual Studio Code Ready

DVT IDE is built on the powerful Visual Studio Code platform used by millions of engineers worldwide and inherits the best features and practices collected into the platform. The VS Code platform's extensible architecture allows DVT IDE to integrate within a large extension ecosystem and work flawlessly with third-party extensions.



DVT IDE for Visual Studio Code in action (1)

EFFICIENT CODE WRITING AND SIMPLIFIED MAINTENANCE

Advanced Code Editing Features

DVT IDE incorporates advanced code editing features such as:

- On-the-fly syntactic and semantic checking
- Error fix suggestions
- Autocomplete and autoinstance
- Source code refactoring operations
- Customizable code templates
- Macro expansion
- Dedicated wizards to generate getters and setters or override functions
- Highly configurable source code formatting
- Integration with revision control systems
- Emacs and vi emulation

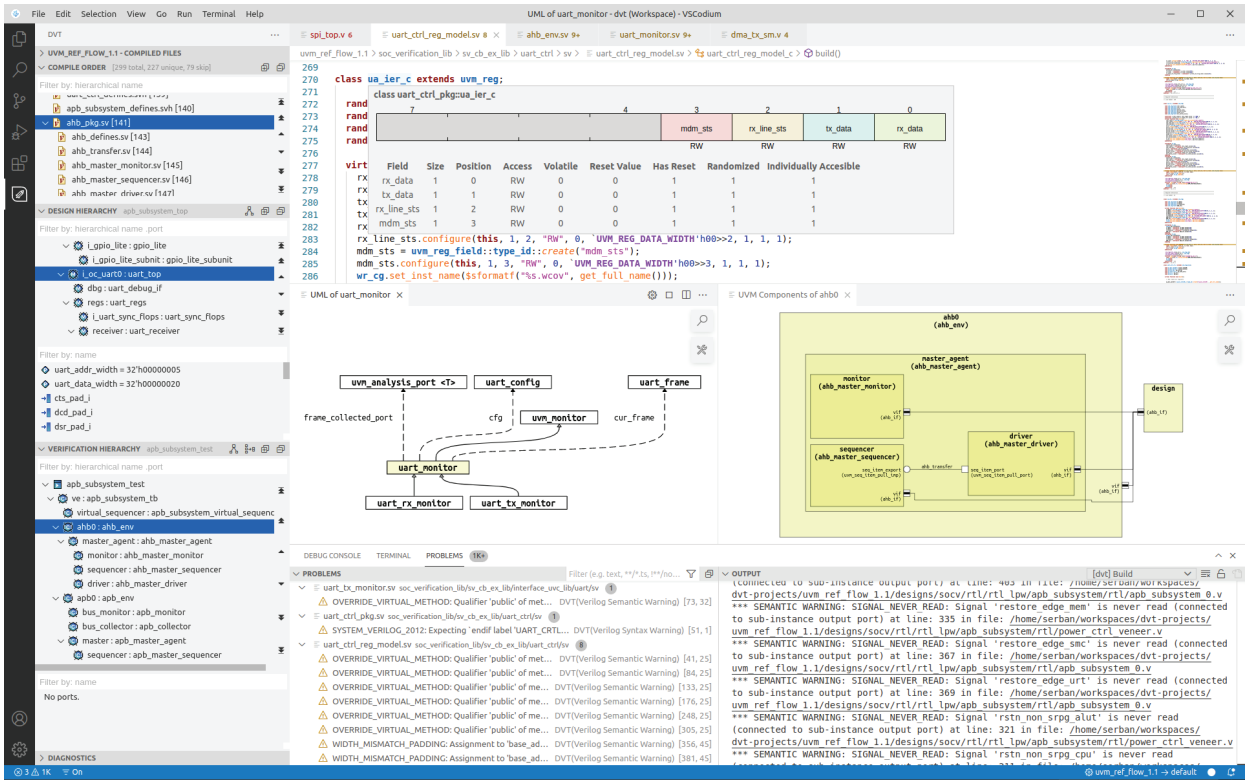
DVT IDE performs on-the-fly incremental compilation. There is no need to invoke the simulator to make sure the code compiles with no errors. Its smart editor highlights the errors in real time, as you type. As a result, users can make the necessary corrections on the spot. To assist with error correction, DVT IDE also provides fix proposals such as "did you mean" when detecting a potential typo and "update instance" when module ports have changed. Moreover, the developer or reviewer can quickly locate and fix various issues spread throughout the code using the Problems View where all errors and warnings are listed.

Autocomplete provides a context-sensitive list of proposals for partially entered text. This capability helps avoid typos and eliminates the need to search for definitions in other files. Autoinstance allows engineers to quickly instantiate and connect a module or entity when needed.

Refactoring allows users to perform semantic changes in code. While a plain text editor or grep/sed utility is limited to simple search and replace actions, DVT IDE can accurately perform powerful operations like “rename method foo() of class bar” and “rename signal x of module y.” All the definitions and places where the method or signal is used are precisely updated. Users can also “extract this piece of code into a separate function” and “add a new port p to module m.” Refactoring helps engineers avoid tedious and error-prone operations such as scrolling through long lists of irrelevant plain text matches or repetitive copy and paste.

Code templates are parameterized code snippets. Combined with TODO tags, this capability enables your users to easily follow the project development guidelines.

Errors inside macros are difficult to investigate. With DVT IDE's macro expansion feature, engineers can examine and debug macro code fragments in context with the source code.



DVT IDE for Visual Studio Code in action (2)

Code and Project Navigation Features

Maintaining tens of thousands of lines of code can be challenging. DVT IDE simplifies maintenance by providing capabilities such as:

- Hyperlinks
- Breadcrumb navigation bar
- Project database queries
- Semantic search for usage, writers, or readers
- Structural views
- Signal tracing
- UML and HDL diagrams
- Comments and macro or parameter values in tooltips
- Semantic source code coloring

These features enable users to navigate easily through complex code, locate the relevant information faster, and understand the source code quickly. They also reduce project costs, by allowing users to avoid locking a simulator license just to inspect the design hierarchy or the verification environment architecture.

Hyperlinks help navigate faster through multiple project files. This practically eliminates the need for using the grep command or memorizing details such as file names and locations. To look up the definition of an element, users can simply hover the mouse over the element name to turn it into a hyperlink. This saves time by jumping directly to the element definition instead of having to search for it.

The editor and diagrams show a breadcrumb navigation bar that clearly indicates the current location in the design hierarchy. It enables engineers to quickly find their way and easily move up and down in the design as needed.

Project database queries allow users to quickly locate specific elements. For example, typing a few letters in a search bar locates a specific module, entity, class, macro, assertion, or coverage item.

Semantic search for usage lets users quickly find out who is calling "method foo," who is using "signal clk of module fifo," or "what are all the constraints on packet size.". It is also very easy to quickly locate all places where a variable, signal or parameter is written or read, as well as where events are emitted.

Unlike plain text grep/sed searches, the semantic search results are accurate. For example, a search for "Who is calling method foo of class a" will not match calls to "method foo of some other class b."

To help in understanding the project architecture, DVT IDE offers structural views for examining function call, design, and verification hierarchies.

Using the signal tracing functionality, designers can effortlessly locate the signal source, an operation called "show writers," or the signal destination, an operation called "show readers." The signal trace can also be visualized as a diagram.

DVT IDE enables your engineers to inspect a project through diagrams. Designers can use HDL diagrams such as schematic, state machine, and flow diagrams. Verification engineers can use UML diagrams such as inheritance and collaboration diagrams. Bit field diagrams for packed data structures and UVM registers are also automatically generated. Diagrams are hyperlinked and synchronized with the source code and can be saved for documentation purposes. Users can easily search and filter diagrams as needed, for example, visualizing only the clock and reset signals in a schematic diagram.

Semantic source code coloring simplifies reading. For example, inactive pre-processing regions are grayed-out, input ports are visibly distinct from output ports, and local variables and class variables have different colors.

In the design or verification source code, users can include waveform specifications using the popular open-source WaveDrom format which are automatically rendered as waveforms.

PREPROCESSED FILES SUPPORT

DVT IDE provides full IDE capabilities when working with files that contain "preprocessor" statements in other languages such as Perl and Python's Jinja2 library, or in proprietary languages. Even if the actual code for simulation is generated in subsequent steps from such files, all the features like navigational hyperlinks, autocomplete, on-the-fly error detection, quick fixes, refactoring, etc. work smoothly as if using files written using only a standard language.

VERIFICATION METHODOLOGY SUPPORT

DVT IDE supports the Universal Verification Methodology (UVM). Its powerful UVM-oriented features help users learn UVM faster, accelerate adoption, and build UVM verification environments with ease.

Users can easily browse through UVM-based classes such as agents, monitors, and sequences, examine component trees, visualize architecture diagrams including TLM port connections, browse register maps, search for factory related constructs or config db getters and setters that may influence the testbench behavior, and generate code using UVM specific code templates and wizards.

DVT IDE users can interactively run the Verissimo SystemVerilog Linter product, including its checks for SystemVerilog and UVM compliance. Verissimo SystemVerilog Linter provides hundreds of customizable rules and advanced capabilities for a thorough audit of testbenches.

SIMULATOR INTEGRATION

Users can start the simulation or synthesis directly from DVT IDE and browse its output in the built-in terminal. Customizable problem matchers grep the standard output in order to detect error or warning messages and connect them with the source code. All errors and warnings are presented in the Problems View, allowing users to quickly jump to the source of the error or warning. By providing simulator log recognition, DVT IDE significantly simplifies and speeds up the simulation analysis and debugging.

CROSS-LANGUAGE CAPABILITIES FOR MIXED-LANGUAGE PROJECTS

The cross-language capabilities allow users to work with source code written in multiple languages and easily understand the whole design.

Features such as hyperlinks, design hierarchy browsing, HDL diagrams, and signal tracing work across SystemVerilog/Verilog and VHDL. For example, a user can click on an instance in Verilog and jump to its VHDL definition.

INCREASING DESIGN AND VERIFICATION PRODUCTIVITY AND QUALITY

Faster and Smarter Code Development

DVT IDE was developed with maximizing the design and verification productivity in mind. Users do not need to switch from the editor to the simulator, browser, or console and therefore they can focus on code writing and review. Moreover, DVT IDE substantially reduces the time spent performing repetitive tasks such as locating a class or module definition, finding all places where a function is called, renaming a variable, and searching for relevant information in large source code files or documentation.

By using hyperlinks, autocomplete, in-line documentation, and semantic search features users can find what they need through a single click or shortcut. As a result, the speed and quality of code development increase significantly.

Efficient Project Management

DVT IDE helps manage your design and verification projects more efficiently. The ability to easily review the source code using features such as hyperlinks, project database queries, structural views, and HDL or UML diagrams enables both managers and engineers to see the whole picture clearly and control a project from a higher perspective.

Lower Language Learning Curve

Beginners feel comfortable with the DVT IDE friendly user interface. In addition, the combination of features such as compilation errors highlighted as you type, error fix proposals, autocomplete, and code templates together with the access to integrated documentation speed up the learning process.

INTEGRATED SOLUTION

DVT IDE for VS Code is closely integrated with the other design and verification products available from AMIQ EDA, including **DVT Eclipse IDE, DVT Debugger, Verissimo SystemVerilog Linter, and Specador Documentation Generator.**

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.dvteclipse.com / www.amiq.com

DVT-VSCODE-0224-A4