IDEs SHOULD BE AVAILABLE TO HARDWARE ENGINEERS TOO!

Author: Syed Daniyal Khurram, Horace Chan
Speaker: Syed Daniyal Khurram
Contribution

• Illustrates productivity benefits of using IDEs in keeping with the demand of a SystemVerilog UVM TB
  – Internal feature evaluation of four popular IDEs: DVT, Sigasi, SlickEdit, SVEditor

• Analyzes application usability and addresses inhibitions towards IDE adoption
  – Helps eliminate/reduce application assessment costs

UVM (1.2) “UBus” example verification environment will be used for feature demonstration purposes
Motivation

• “One tool to rule them all”
• Automation of code development and simplification of debug for modern design sizes
Outline

IDE Feature Overview
- Code Navigation
- Code Development and Refactoring
- Macros
- Advanced Features

Usability and Concerns
IDE Feature Overview
Code Navigation

- Symbol Lookup
- File Browsing
- Class Browsing
- Design Browsing

Reduce time spent on understanding source code and locating design information!
Symbol Lookup :: Search by Definition

Jump from a symbol to its declaration!

Localized to workspace and not just current file
Symbol Lookup :: Search by Reference

List all occurrences of the symbol!

Localized to workspace, project or single file
File Browsing

Visually Copy/Paste, Add, Delete and Rename files!
Class Browsing

Removes need to open up multiple files/tabs to understand class relationships.

Presents hierarchical view of object oriented programming code!
Design Browsing

Display recursively all instances of a Verilog/SystemVerilog module, SystemVerilog instances, instances of VHDL entity and VHDL components!
Code Navigation :: Takeaways

• Symbol Lookup
  – *Push-button alternative to external/built-in search plugins such as ‘Grep’*

• File Browsing
  – *File management through an easy to use interface*

• Class Browsing
  – *Visualization of the hierarchy makes it easier to understand class-based TB organization and relationships*

• Design Browsing
  – *Design engineers benefit when analyzing external IP or during design audits.*
Automate common programming tasks through intelligent code formatting and code prediction!
Auto-Editing

- Auto-completion
- Auto-parametrization
- Inline Expansion
- Smart Indentation
Intelligent Refactoring

Changes only relevant occurrences of renamed element based on the results!

Resolves symbol and its references first!
Code Collapse

Selective hiding or displaying of relevant code!

Highlights functional information during code reviews and audits.
Code Development & Refactoring ::
Takeaways

• Auto-Editing
  – Solution to typical questions that arise during code development:
    • What is the name of the method that you wish to use?
    • What is a methods order of arguments?
    • What are the possible values of an enumerated type?

• Intelligent Refactoring
  – Code transformations that maintain the behavior of the design
    • Especially handy when dealing with port related changes in design modules!

• Code Collapse
  – Useful for masking irrelevant information
Enter the Macro

Macro Recording/Keyboard Macros

Macro Expansion

Automate repetitive actions performed frequently while writing code

Open pre-existing Macros in an interactive window and trace line by line for debug and analysis
Macro recording/Keyboard Macros

SlickEdit Pro lets you create a Macro simply by recording a series of user interactions! The recorded Macro is saved as proprietary Slick-C source code, and can be re-opened and modified afterwards.

Once recorded the Macro can be saved and bound to a key for ease of access.
Macro Expansion

UVM Macros can be unwrapped line by line and analyzed.
Macros:: Takeaways

• Macro Recording/Keyboard Macros
  – Save time spent on typing re-use code
  – Extend existing command functionality or add new commands e.g Macros to display duplicate lines of text, file attributes etc

• Macro Expansion
  – Expand and analyze:
    • Proprietary simulator pre-processing code
    • Macros included as part of a verification methodology standard such as UVM
    • Macros created in a tool specific programming language such as Slick-C

Emacs still reigns supreme in this category! As macro programming in Emacs Lisp can be remarkably powerful in the hands of an experienced user.

Advanced Macro debug features such as breakpoint insertion can be utilized in IDEs that support compiler/simulator integration capabilities.
Advanced Features

Integration with UVM

• Each IDE integrates with UVM
• Integration with popular compilers and simulators through the use of add-ons/licenses/tools & build configuration

Integration with Simulation Tools

Revision Control Integration

• Revision Control from within the IDE
• Eclipse based IDEs require plugins e.g Subclipse
• SlickEdit Pro has inbuilt VCS support
IDE Integration with Tools and Standards

• Additional UVM debug features available in DVT:
  – UVM Factory queries
  – UVM Templates
  – UVM Browser & Sequence Tree

While UVM debug features are supported in most advanced simulators available in the market, they can only be used post-compilation!

• Integration with simulation tools:
  – External tools and build configuration in Eclipse based IDEs. However this requires strenuous effort and may not work for all tools
  – DVT: Add-on tool (DVTDebugger)
  – Sigasi: Sigasi Studio Creator and higher

Refer to the product website or the full paper for a list of supported simulators!
Revision Control Integration

All features common to SVN can be used:

- Update files
- Add files
- Commit
- Check in (with comments)/Check out files
- Revert/Remove files
- "Diff" against a file or history, and view changes or annotations to your code from within the IDE!
Advanced Features:: Takeaways

• Integration with UVM
  – *Significant in modern ASIC verification*

• Integration with simulation tools
  – *On the fly debug!*
  – *Invoke compiler/simulator from within tool GUI and trace warnings/errors to problematic source code*

• Revision Control Integration
  – *Removes time spent switching between command line and text editor*
  – *Visual ‘diff’ is powerful and interactive*
Usability and Concerns
Usability and Concerns

• Learning Curve
  – User-friendly and easy to pick-up by junior engineers
  – Prior experience in using established IDEs (Eclipse, Visual Studio) reduces training time

• Reduction of Tools
  – Vast array of features in a centralized environment
  – External plugins are supported

• Support
  – Customer specific support available for all commercial IDEs

• Setup Flow
  – Possibly the biggest adherence towards IDE adoption
  – Quite simple actually with clear instructions

Note: Not true!
Conclusion

• There are tools!
  – Established tools exist in the marketspace
  – Choose what best fits your needs
• Worth your time!
  – Invest to save time
• Less is more!
  – Centralize your environment
  – Reduce resource consumption

A comprehensive summary of features available per tool is tabulated in the full paper as a reference!

Questions?