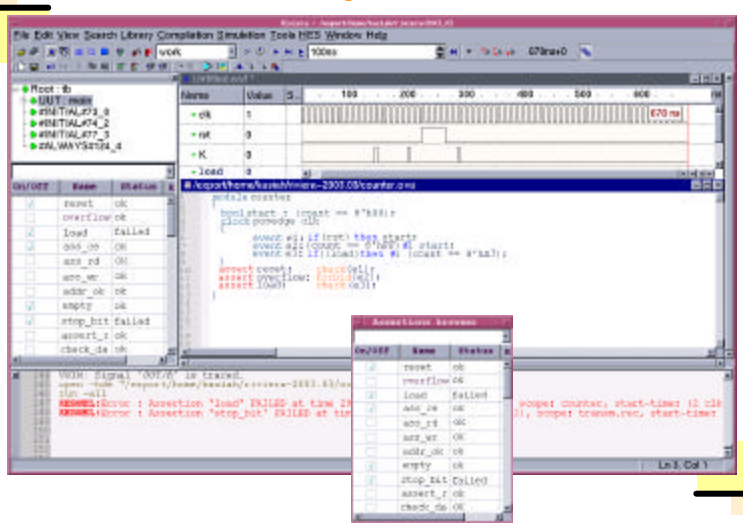
ALDEC, INC

# ASSERTION-BASED VERIFICATION

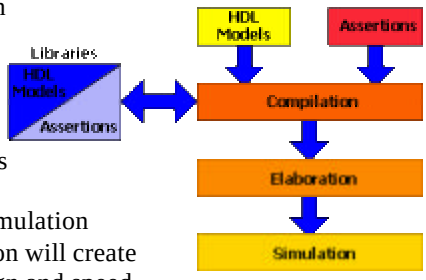


## What is Assertion-based Verification (ABV)

Assertion-based verification is the convergence of design and verification to create an improved design-for-verification methodology. This methodology has been dormant for over a decade in software development and is just now making its' way into hardware design flows. Assertions are quite simply design checks embedded into the module or IP to verify the assumptions about how a specific block should operate, both by itself and in relation to the surrounding design blocks. They explicitly express all information about the functional/behavioral nature of the block just as the designer intended it to be used. This smarter methodology brings together design and verification to improve both the code and the verification process simultaneously. By incorporating these assertion checks early (during coding), designers will see value throughout the process of design, integration, system simulation and tape out.

## Why Assertion-based Verification

It is estimated that over 70 % of the design cycle for any new device is spent in the verification process. As these designs continue to grow in size and complexity the need for improved verification methods at the functional level continues to grow along with it. Implementing assertions as part of the design process along side simulation and formal verification will create a higher quality design and speed time-to-market. Simulating with assertions provides more information, which can improve understanding about the internal working of the design and reduce the number of iterations. Whenever the design does not behave the way it was intended or an assumption is broken, the assertion flags the exact time and location of the problem.




This approach makes pinpointing and correcting the problem at the functional level much more efficient. Additionally, testbenchs created for large designs may be unable to check all possible sub-interfaces and can create the potential danger of the design malfunction in lower level modules. Because assertions are bound to the low level modules violations will always be reported, regardless of whether the testbench sees the violation or not.

## Differences between Simulation and Formal Verification

The main difference between the two methods is when they are applied. Assertions when used with simulation are checking the design block dynamically at the functional level early in the process. Formal verification uses only a subset of these assertion rules statically at the behavioral level after simulation is verified. Based on the amount of time designers spend at the functional level and the limitations with formal verification, checking assertions during simulation offers an early indication of a potential problem and can ultimately reduce the overall debugging time needed. Formal verification further increases the thoroughness but without this early indication, the corrections can take much longer or be missed entirely.

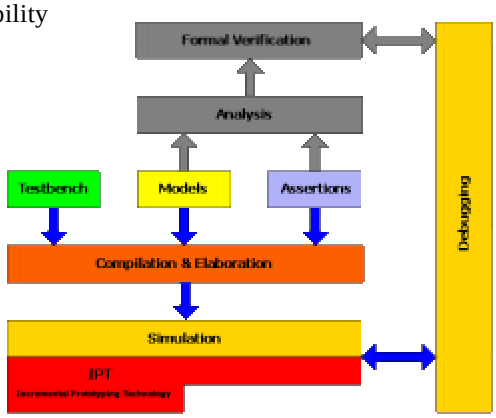
## Assertions Languages

There are currently several Assertion-based Verification languages available. Aldec has implemented a large portion of the individual specification and will continue to add additional support as ABV becomes an IEEE standard. The following is a brief description of each:

- OpenVera™ Assertions (OVA) language has been donated to the public domain by Synopsys™, it is based on VERA and provide comprehensive support for assertions.
- Property Specific Language (PSL) was donated by IBM® and is based on the Sugar formal property language, PSL provides the most advanced and complex assertions checking capability
- Accelera Open Verification Library (OVL) provides ready to use assertion functions in the form of VHDL and Verilog HDL libraries.
- SystemVerilog is a next generation language standard based on many of the best features of the SUPERLOG, VHDL, VERA, C, C++, OVA, PSL/Sugar languages, added to the core Verilog HDL. SystemVerilog is aimed at becoming the next standard approved by IEEE.

## Riviera and Riviera-IPT support

Riviera and Riviera-IPT have the unique ability to utilize Assertion-based Verification in the mixed language software simulator as well as in the hardware accelerator. The assertion compiler from Aldec produces these module checks in the form of RTL code added to the synthesizable portion of the design. Once assertions are implemented into the design they can be verified at the behavioral (dynamic) level in the software simulator, and at the structural (static) level in the hardware accelerator. In addition to the improved verification time and flexibility of using assertions for verification the designer can utilize these checks during design prototyping as well as in the final product. Assertions used in prototyping can detect any functional problems in real time. They become part of the design for monitoring the desired signals and can flag an error or exception whenever there is a violation.




## Additional Benefits of using ABV with Riviera and Riviera-IPT:

### ♦ Less Simulation Overhead

Because assertion support is built into Riviera's simulation kernel less overhead exists compared to using an interfaced 3rd party checker. This can equal a performance improvement of over 20% when compared to the competition.

### ♦ Better than Testbench alone

Because assertions are part of the source code and describe the internal functionality of the module they create a faster and better way to check the system. Implementing assertions with a typical testbench creates improved system-level results by improving the overall coverage with less effort.

### ♦ Better Debugging

Using assertions in a design flow can produce improved verification results earlier in the cycle during simulation. This adds value by providing the exact location of the bug quickly and concisely and with less effort than typical debug techniques.

### ♦ Improved Design Re-Use

Assertions are quite simply assumptions about how a specific block should operate, both by itself and in relation to the surrounding design blocks. Because of this, subsequent re-use of the specific modules becomes much easier.

### ♦ Better Design Outsourcing

Because assertions provide an exact blueprint and intended use of the design module, outsourcing to other team members becomes more reliable and efficient.

### ♦ Design Monitoring

Assertions can remain in the final ASIC or FPGA as part of a designer specified monitoring system. Once implemented these assertions can be used to detect and react to protocol or sequence errors.

## Platform Support

- Sun Sparc running Solaris 7 or 8
- PC running Microsoft Windows 2000/NT/XP
- Linux kernel 2.4 (Red Hat 7.0 and higher)

## Industry Standard Support

### IEEE

- VHDL 1076-87/93
- Verilog 1364-95/2001 (partial)
- VITAL 1076.4-95/2000
- SDF 1.0, 2.0, 3.0 and 4.0
- SystemVerilog (Q1 2004)

## Interface Protocols

- Tcl/Tk
- PERL
- SWIFT (including LMTV and MemPro)
- PLI
- VHPI + CHPI

ALDEC
The Design Verification Company
2230 Corporate Circle
Henderson, NV 89074
United States
Phone: 702-990-4400
Fax: 702-990-4414
E-mail: info@aldec.com