FIRSTEDA
ENABLING DESIGN
FirstEDA Limited, Eastlands II, London Rd, Basingstoke, RG21 4AW, UK
t: +44 (1295) 201250 / e: info@firsteda.com / w: www.firsteda.com

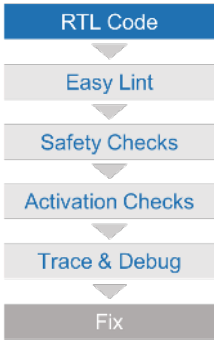
onespin
making electronics reliable

DATASHEET

OneSpin® 360 DV-Inspect™

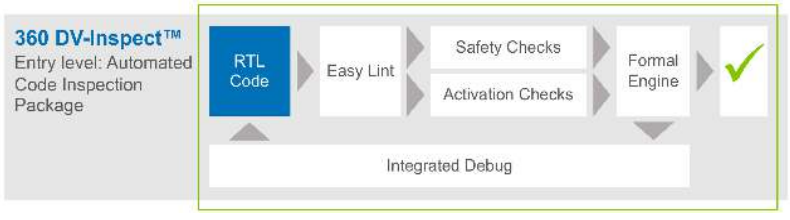
Add-On to OneSpin 360 DV Property Checker™

AUTOMATIC CODE INSPECTION BEFORE FUNCTIONAL VERIFICATION



360 DV-Inspect automatically inspects the RTL code once it has been compiled, without the need for any testbench setup. This code inspection comprises Easy Lint and exhaustive formal analysis with interactive debugging, eliminating an extensive amount of early simulation and stimulus creation while identifying issues before they result in serious bugs. Thus, DV-Inspect lends itself as a sign-off criterion for handing off an RTL block to functional verification and integration into automated daily regression runs.

Easy Lint identifies potential issues without formal analysis. It reports typical issues like value truncation, latches, and inconsistent sensitivity lists, but also more advanced issues such as a combinational feedback loop from a FF output to its asynchronous reset. The reported issues are visualized in the GUI with a differential view, allowing to easily identify fixed or newly introduced issues in a new code revision. Issues can be waived and are fully accessible for custom TCL-scripting. Moving from Easy Lint to formal analysis is a seamless step in DV-Inspect as no user input is required. The automated formal analysis is handled by two kinds of checks: Safety Checks focusing on synthesis-simulation mismatches and other issues like bus contention while the Activation Checks focus on code coverage. All these checks are automatically created and the exhaustive verification of these assertions is triggered by pushing a single button or running a tool command in batch mode. This leverages the full formal power of the OneSpin 360 DV platform to either exhaustively verify these assertions or quickly find a violation. An exhaustively proven assertion gives a level of confidence impossible to achieve in simulation – you are sure that it cannot be violated in any simulation run. A violation, on the other hand, is also very intuitive to analyze – after all, it is the same as if you had actually run the exact same input stimuli identified by the formal engines in a simulator. Similar to Easy Lint, failures can be waived and all results are fully accessible in TCL for creating custom regression scripts.



Activation Checks automatically produce simulation traces to actually cover certain code branches or FSM transitions. These simulation traces can be exported as testbenches to a simulation environment to increase coverage in later stages of the functional verification flow. In addition, unreachable code branches are identified which can either point to design errors or indicate potential coverage excludes for the simulation environment, again increasing the overall code coverage. In case the inputs of the DUT need to be restricted to realistic ones for the formal analysis, DV-Inspect fully supports SVA and PSL for formal constraints. For standard protocols like the AMBA™ family, the corresponding constraints can be taken automatically from the corresponding Verification IP on top of DV-Inspect, essentially allowing a properly constrained DV-Inspect analysis right after compiling a design without any need for formal knowledge.

ONESPIN 360 DV-INSPECT FEATURE OVERVIEW

PLATFORM

- Linux 64 bit: RHEL 5-7, SLES 11
- Windows 64 bit: 7/8.x/10 Server 2003/2008/2012

LANGUAGE SUPPORT

- Verilog, System Verilog, VHDL, SystemC
- Mixed Language with Verilog/System Verilog/VHDL/SystemC

EASY LINT

- Over 300 warnings for potential issues with Differential View and Waivers
- Synthesis Simulation Mismatches like incomplete sensitivity list or full case pragma violations
- Design errors like value truncation, uninitialized registers, or port size mismatch
- Combinational feedback loops

FORMAL AUTO CHECKS / SUPER LINTING

- Initialization/Reset Checks
- Range and condition Checks, e.g. array access out of bounds, division by zero
- Race detection between processes, e.g. write-write races.
- Arithmetic checks, e.g., negative remainder
- Bus contention checks

FORMAL COVERAGE ANALYSIS

- Dead Code detection, line/block coverage
- FSM deadlock detection, state/transition coverage
- Stuck-at bit detection, toggle coverage
- Generation of System Verilog and VHDL test benches.

FORMAL ENGINES

- Multiple provers with heuristic engine director
- Speedup by parallel execution and network distribution, e.g., LSF
- Full prover control possible

INTEGRATED DEBUGGER

- Waveform Debugger, Source Code Debugger, Design Instance/Hierarchy Browser
- Fan-in/Fan-out /Driver Tracing through source code
- Active Value and Active Code Annotation
- Automated transaction wave view for Apps

ONESPIN SOLUTIONS

OneSpin Solutions has established itself as a leader in formal verification through a range of advanced electronic design automation (EDA) solutions for digital integrated circuits. Headquartered in Munich, Germany, we are passionate about enabling our users to address design challenges in areas where reliability really counts: safety-critical verification, SystemC/C++ high-level synthesis (HLS) code analysis, and FPGA equivalence checking. Our advanced formal verification platform and dedication to getting it right the first time have fueled dramatic growth over the past four years as we forge partnerships with leading electronics companies to pursue design perfection. OneSpin: Making Electronics Reliable.

YOUR NOTES:

Contact · info@onespin.com · www.onespin.com
USA: +1 408 734 1900 · Europe: +49 89 99013-0 · Japan: +81 3 4530 3865

2018-04 V. 1