

# Insight • Semi-Formal Analysis



## Simulation-Centric Formal Analysis

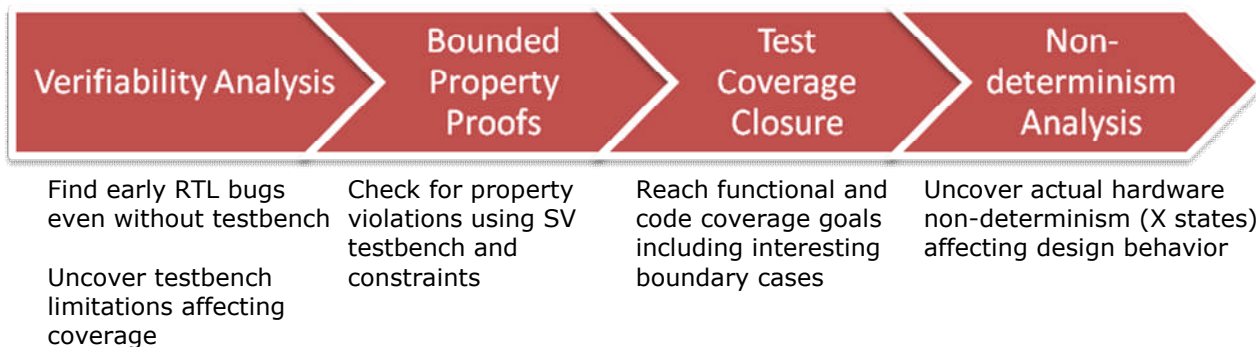
### HIGHLIGHTS

- First behavioral-level formal analysis for Verilog and SystemVerilog
- Better leverage of existing testbench
- Transaction-level counterexamples
- More flexible and less effort than conventional formal tools
- Multiple parallelized formal engines find solutions faster

### OVERVIEW

Insight is a new generation of behavioral-level formal analysis with a simulation-centric viewpoint that enables engineers to be more productive using formal technologies to augment design for verification methodologies.

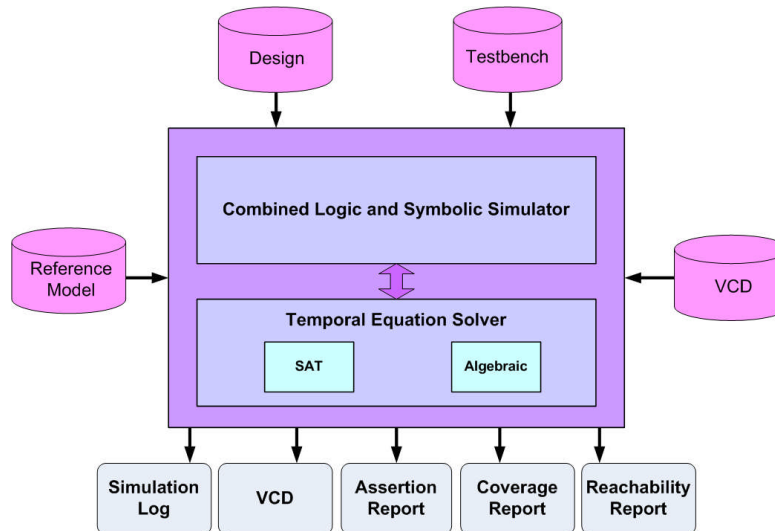
Insight uses behavioral-level symbolic simulation to better leverage an existing testbench infrastructure including behavioral-level transactors, properties and reference models for broader verification. Insight's early verifiability analysis helps you find RTL bugs, generate more realistic code and property coverage goals for formal and simulation methods, and identify the hard-to-verify areas of the design while there is time to do upfront planning and devise appropriate strategies to deal with them. Using Insight for block-level formal proofs of assertions uncovers RTL block bugs. This helps you establish block-level property guarantees enabling a more smooth SoC synthesis process. Test coverage closure allows the designer to evaluate interesting boundary cases using coverage test generation. Insight also addresses some specialized concerns in SoC design involving X state analysis for reset and low power verification which have until now gone unaddressed.



### FUSION SIMULATION

Insight uses a mixed logic and symbolic simulation kernel working in conjunction with formal temporal solver engines to perform formal analysis. Insight has many advantages over traditional approaches for formulating formal proof targets. The most significant innovation is being able to symbolically simulate Verilog and SystemVerilog behavioral-level, RTL, and gate-level constructs. Conceptually it would be best if both logic simulation and formal could fully share models and properties at all abstraction levels. However today most other formal tools only handle synthesizable RTL testbench constructs. This leads to significantly more complex use model and reduced functionality.

Symbolic simulation propagates expressions of all possible execution paths through a testbench and design in a single pass compared to logic simulation which can only simulate one path at a time. Behavioral-level symbolic simulation enables performing formal analysis by using a behavioral testbench, reference models, and properties to describe the permissible inputs and intended behavior. The use of constrained random variables in verification environments provides a convenient way to model symbolic representations of transactions.



### Insight Simulation-Centric Formal Analysis

When partial logic and symbolic simulation are used together, Insight can perform deeper sequential analysis while still comprehensively evaluating possible execution paths or limiting the state space through case analysis.

### FORMAL REACHABILITY ANALYSIS

Reachability analysis determines how much of the RTL, properties, and cover points can be exercised. Compared to simulation code coverage or structural lint tools it is more comprehensive and accurate because it tests all conditional execution paths for formal satisfiability. Used very early in design process you can find RTL bugs, generate more realistic code and property coverage goals for formal and simulation methods, and identify the hard-to-verify areas of the design where more special attention will need to be applied. Later when a testbench is available, reachability analysis allows you to reassess coverage based on constrained stimulus and possibly identify testbench limitations.

#### RTL Source Code:

```

414 always @(posedge clk or negedge resetN)
415   if(~resetN)
416     pswitch <= `D 1'b0;
417   else if ((fifo_empty & fifo_rd0 &
             !fifo_rd1 & (config_mode == 2))
418     pswitch <= `D 1'b1;
419   else if ((fifo_rd1 & fifo_alm_empty) | dl_down)
420     pswitch <= `D 1'b0;

```

#### Reachability Report:

```

instance at mod.inst1
  ../rtl/avy_checker.v, line 156: 1
  ../rtl/avy_checker.v, line 418: 0
  ../rtl/avy_checker.v, line 445: 0 (symbolic hit = 1)

```

*"symbolic\_hit" indicates that formal analysis determines this code or property is reachable*

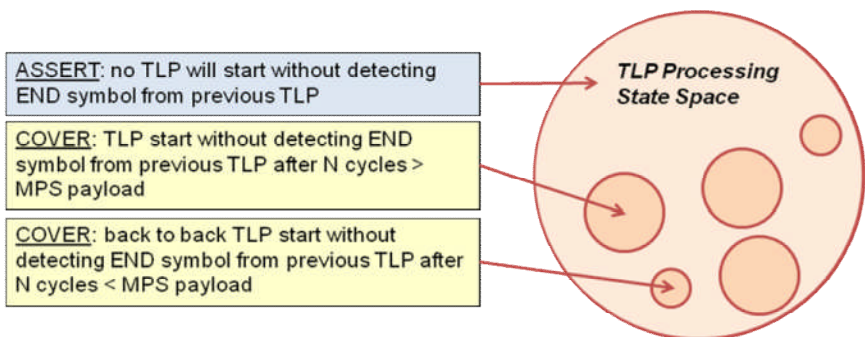
When conditional statements or properties are unreachable it is important to understand the reasons. Insight provides causal analysis on these terms back to their origins for intuitive debug.

Support is planned for the Unified Coverage Interoperability Standard from Accellera enabling sharing and merging reachability analysis coverage data with simulation and lint tools.

### BUG HUNTING

Bug hunting is the process of sequentially analyzing the design over a bounded symbolic simulation period to determine whether any design properties can be violated under legal input conditions.

Insight supports a wide range of design and verification constructs which are automatically analyzed including scoreboard checks or assertions. In most cases only a simple directive is added to an existing testbench to enable a switch from logic simulation to symbolic simulation and formal analysis.



**ASSERT:** no TLP will start without detecting END symbol from previous TLP

**COVER:** TLP start without detecting END symbol from previous TLP after N cycles > MPS payload

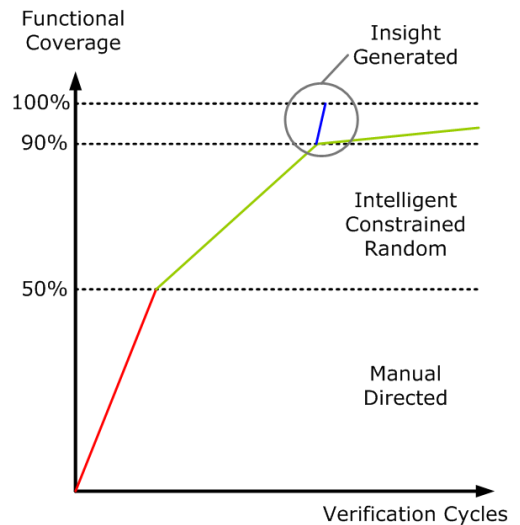
**COVER:** back to back TLP start without detecting END symbol from previous TLP after N cycles < MPS payload

**TLP Processing State Space**

When an assertion is falsified or a coverage item is solved, the resulting counterexample is then replayed in logic simulation mode through the testbench for normal debug purposes and VCD file generation. After symbolic simulation is resumed additional assertions and coverage items can be analyzed. Assertion and coverage reports can be generated by Insight.

**TEST COVERAGE CLOSURE**

Test coverage closure targets generating interesting or missed boundary cases specifically for simulation replay. This allows the engineer to visualize the design operation through these cases of interest. Using constrained random test methods to achieve property, coverpoint, and code coverage targets can be very inefficient and require significant effort to manually over constrain random parameters. Insight generates coverage report which can then be compared to coverage goals from formal reachability analysis.



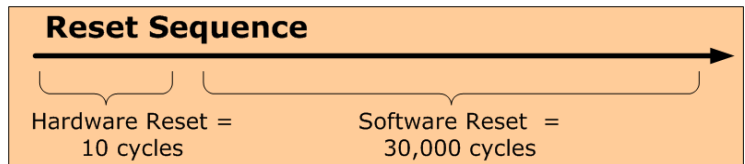
```

===== Start of Coverage Report =====
##### Cover Property Report #####
Property: avy_pcie_x8_test.avy_pcie_x8.seq.DLL_P3
Attempted: 46, succeeded: 1, failed: 43, succeeded vacuously: 0.
Property: avy_pcie_x8_test.avy_pcie_x8.seq.DLL_P4
Attempted: 46, succeeded: 1, failed: 43, succeeded vacuously: 0.
##### End of Cover Property Report #####
===== End of Coverage Report =====

```

**NONDETERMINISM ANALYSIS**

When registers are uninitialized or when don't care assignments are used for synthesis optimizations, non-deterministic design behavior may result. Logic simulation cannot accurately reflect the actual 2 state hardware semantics of don't care. Simulation has to pick one valuation and follow that execution path even through it is not accurate. X verification is a formal method to find unwanted non-determinism in the design which can result in unexpected behavior, potentially only found later during gate-level simulation or worst case in the lab. Using symbolic simulation Insight will propagate X's through the design and evaluate all execution paths to determine if observation points illustrate non-determinism. X verification can also be applied to various sources of non-determinism such as reset, lower power design using power gating and clock domain crossing verification.



*Reset Verification and Repair*

One application of X verification is verifying the reset sequence which usually applies hardware reset and software initialization to a subset of registers and memories. These leaves a lot of X's in the post reset state, some of which could result in unexpected behavior. Insight will find real X's for you and even generate a repair solution to eliminate the X's. In order to process full chips Insight's reset verification flow utilizes automatic design partitioning and interval analysis to handle long SW initialization sequences. Insight also utilizes a logic simulation VCD file for the actual reset and initialization sequence and picking registers which need to be analyzed. Insight can also generate a reset repair solution. Given a set of uninitialized registers causing adverse behavior due to X propagation, Insight can find a minimal set to hardware reset to fix the problem. The impact of this approach is found in cell and routing area savings. Reset repair adds more flexibility to explore different solutions based on later stage place and route considerations.

**LOCATIONS AND FACILITIES**

**U.S. Headquarters:** 2 Atwood Lane, Andover, MA 01810, Tel: 978 689 7286, Fax: 866 457 1388

**International Field Office:** 76, 1st Section, Chung-Hsiao E. Rd., suite 1203, Taipei, Taiwan, ROC, Tel +886-2-23278766

**Sales**

Avery Sales and Support

Saphirus

BlackForest EDA ([blackforest-eda.de](http://blackforest-eda.de))

978 689 7286

408 625 7618 (West)

+49-2132-137485 (Europe)

**WEBSITE:** <http://www.avery-design.com>

Trademarks/Copyright ©2009 Avery Design Systems, Inc. All Rights Reserved.