

Insight • Early RTL Formal Analysis



OVERVIEW

Insight is a new generation of behavioral-level formal analysis that enables engineers to be more productive with formal technologies for early RTL analysis.



Find early RTL bugs without testbench

Analyze cross FSM dependencies

Justify simulation coverage goals

Automatically generate microarchitecture-level assertions and coverage monitors

Uncover actual hardware non-determinism (X states) affecting design behavior

Early estimate of At-Speed DFT coverage at RTL

Increase DFT coverage with formal-based repairs

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 a significantly more complex use model and reduced functionality.

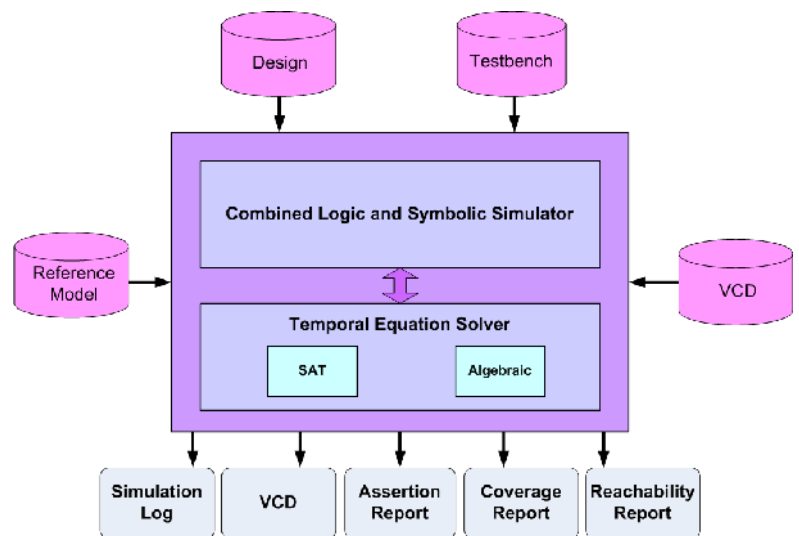
Symbolic simulation enumerates all possible execution paths through a testbench and design in a single pass compared to logic simulation which can only simulate one path at a time. Behavioral-level symbolic simulation enables performing formal analysis using a behavioral testbench, reference models,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

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 automatic guided search and 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. Insight targets reachability of the following from an initial seed state:

- Conditional branch blocks
- Properties (assert and cover)
- All FSM state transitions enumerated in the RTL
- FSM deadlock caused by single or multiple interacting FSMs



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 for more intuitive debug on the sequential logic terms preventing reachability tracing back to their sources.

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

ASSERTION AND COVERAGE SYNTHESIS

Assertion and coverage property synthesis supports a broad range of white box properties based on automatic microarchitecture-level feature recognition. Microarchitecture-level assertions and coverage focus a set of hardware functions and their interactions to isolate problems. Insight augments operational assertions which are manually developed by the verification team. Together microarchitecture-level and operational assertions and coverage models provide the best objective measures for verification closure, and increases the overall quality of functional verification. The range of microarchitecture-level features recognized include:

- FSM
- Queue (FIFO, stack)
- Memory
- Arbiter
- Counter
- Key control functions
- Bus
- Clock domain crossing
- Power controller

Insight supports Tcl commands to navigate the sets of microarchitecture objects to apply assertion and coverage synthesis. Integration with the simulation or formal analysis environment is supported through SystemVerilog binding of assertions and coverage to module instances, or generating separate module that can be included as another top-level module. A query function will show the FSM attributes such as the FSM states and input variables involved in each state transition conditions.

```

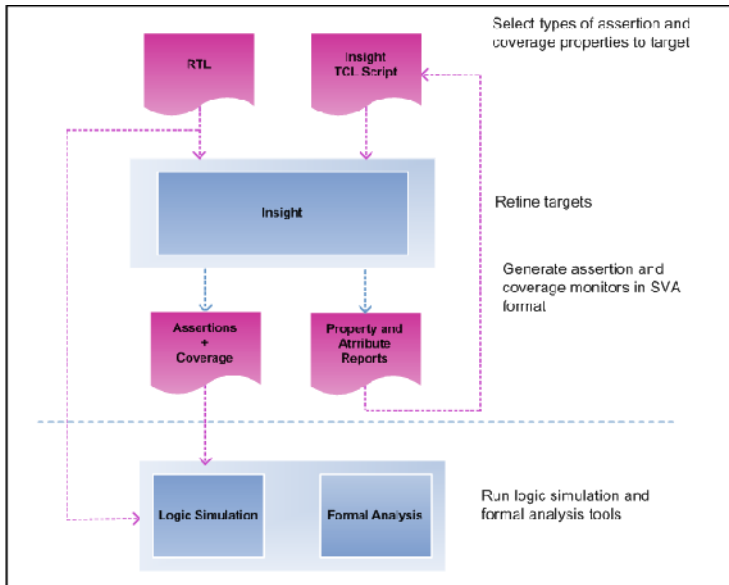
set fsms [insight_get_objects -type fsm]
insight_asyn_script_gen $fsms -file fsm_properties.tcl

fsm_properties.tcl:
insight_asyn_property_gen tb.f1.fifo_state -fsm_state -fsm_transition
insight_asyn_property_gen tb.r2.state_1 -fsm_state -fsm_transition
insight_asyn_property_gen tb.r1.state_0 -fsm_state -fsm_transition

Assertion:
property avy_prop_tb_fsm_state_37;
  @(posedge tb.clk) (tb.fsm.state == 3'D2) |->
  (tb.fsm.state == 3'D2) [*0:7] ##1 (tb.fsm.state != 3'D2);
endproperty
avy_property_tb_fsm_state_36: assert property(avy_prop_tb_fsm_state_37)
  else $warning("AVERY_ASSERT: FSM timeout detected");

```

Assertion and Coverage Synthesis Flow



The following example shows a FSM has 5 states. The state transition variables for each state transition are summarized. A long form will report the full, multi-level equations for each variable.

TCL command:
insight_query_object -type fsm -info 2

```

===== FSM Analysis Report =====
Variable: cpu8080.state
Type: State register (FSM)
S1, Reset, Signal wait, -> S1 (reset)
  -> S2 (reset)
S2-> S1 (reset)
  -> S3 (reset)
S3, Signal wait, -> S1 (reset)
  -> S4 (reset, waitr)
  -> S3 (reset, waitr)
S4, Signal wait, -> S4 (reset, opcode)
  -> S1 (reset, opcode, sign, parity,
    carry, zero)
  -> S5 (reset, opcode)
S5, Signal wait, -> S1 (reset, intr, ei)
  -> S5 (reset, intr, ei)
=====

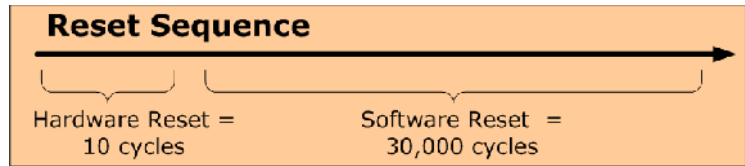
```

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.

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.



This 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 reduction solution. Given a reset state, Insight can find a smaller set of registers to reset that maintains the same reset state as the original design. Reset reduction adds more flexibility to explore different solutions based on later stage place and route considerations.

Low Power Verification

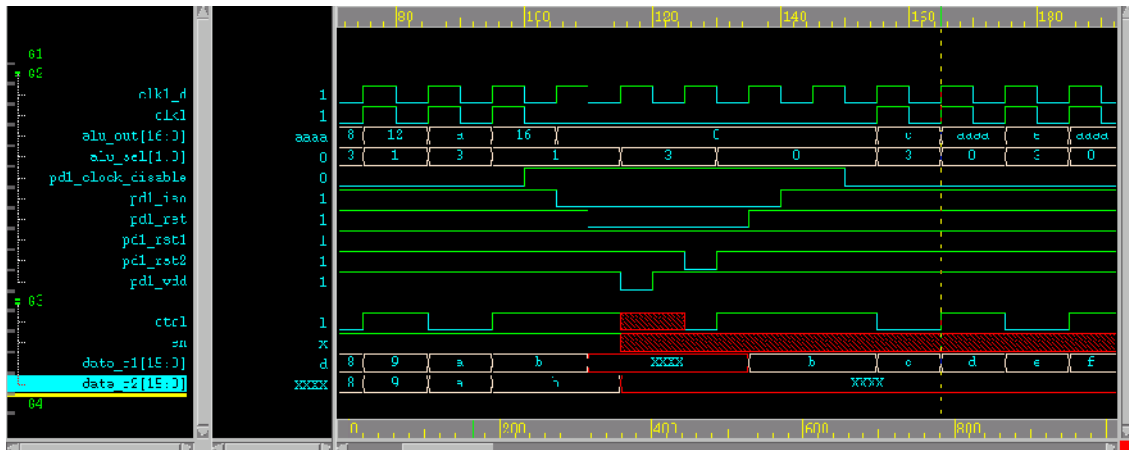
Insight's X verification can also be applied to verifying low power designs. Here the goal is to verify that no unwanted non-determinism affects any powered-on blocks from the power gated blocks. By running through all the power state transitions the design issues with reset, retention, isolation, and power-tolerant control logic can be identified and corrected.

The key elements of Insight's low power formal verification are:

- reads UPF 2.0 for power related information
- performs power aware symbolic simulation to add symbolic X during corruption phase
- analyzes built-in power gating assertions from UPF and user-supplied assertions for voltage scaling
- analyzes and reports all possible X propagations to module outputs internal registers
- generates counterexamples and testbench to confirm operation in normal logic simulation

M2159: Entering symbolic simulation after time 0 [Simulat]
M2159: Leaving symbolic simulation at time 200. [Simulat]
Symbolic X values propagate to top.alu_out at time 200 due to the following variables and values:
top.data_r2_i115 at ret_ff2.v: 164:
value1= 16'h2a09, value2= 16'h8dc2
top.en_i115 at ret_ff2.v: 166:
value1= 1'b0, value2= 1'b1
Symbolic X values do not propagate to top.ctrl

Here we show an example of an X issue that does not show up in logic simulation but which could have adverse effect on operation of a powered-on block in real life. Here the output of the power gated block, alu_out, is connected to a powered-on block. Because of X-optimism issues the alu_out appears as though isolation has been applied. X verification analysis on one of the power state transition cycles that alu_out flags possible non-determinism and the source registers, en and data_r2, are identified as the source of the X. After adding reset or retention to these registers the issue is resolved. Insight can augment design rule checking with dynamic X verification.

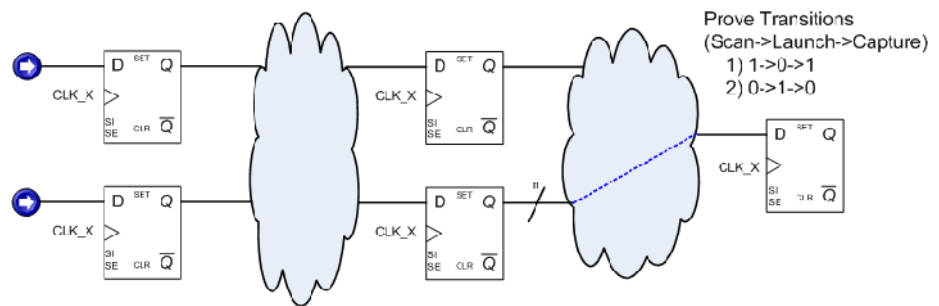


Waveform showing power cycle sequence. Power corruption is applied at time 260 when power is turned off for domain pd1. alu_out is a function of ctrl, en, data_r1, and data_r2 and should be X.

DFT ANALYSIS

Achieving high at-speed path transition testability coverage is becoming a must for sub 65nm designs. However waiting until after synthesis and ATPG to consider how to handle untestable logic may result in dire consequences. Insight can perform at-speed path transition testability analysis at the RT-level prior to synthesis and enable designers to ensure designs are highly testable. Insight proves that the path transitions are possible so later on ATPG can reliably generate the scan patterns.

RT-level at-speed path delay fault analysis, coverage estimation, and repair enables designers to target higher initial ATPG results by finding and resolving testability problems during initial design stage. Upfront at-speed testability analysis identifies RTL code that will limit coverage because of launch/capture limitations. After initial problems are resolved, Insight will perform critical path identification and sampling on the RTL and then determine individual path delay fault testability including root cause analysis as well as generate overall cover reports. Finally Insight will generate repair solutions ranked by the highest incremental % coverage improvement. The designer will then select a number of these RTL repairs to make to reach the desired coverage level. Insight allows you to control the scan design intent by controlling which registers are scannable, memory test bypass modes, and hierarchical test methods for shared isolation scan wrapper, and X generator sources.



All untestable paths are categorized and grouped for easy analysis. Furthermore Insight can analyze the untestable paths and propose repair solutions ranked by largest testability improvement. Insight DFT supports designs with multiple clock domains, multicycle paths, synchronous and asynchronous memory write through modes, and partial scan.

Insight supports an automated flow that includes block-level testbench generation for the initial analysis and repair analysis. In both cases testability reports are generated including testable and untestable transition paths and coverage results.

```

=== Begin of At-Speed Path Report ===
top.d => top.f: OB
top.x => top.f: OB
top.e => top.f: OB
top.a => top.a_1: 010 (UP), 101 (UP)
top.c => top.e: 010 (UP), 101 (UP)
top.a_1 => top.d: 010 (T), 101 (UP) UC
top.b => top.d: 010 (UP), 101 (UP) UC
Number of paths: 14, testable: 1, coverage: 7.14
=== End of At-Speed Path Report ===

```

- Untestable Fault Categories
- XG: fanin has X generator
 - UC: uncontrollable register
 - UB: destination register unobservable
 - CD: clock domain crossing
 - UP: uncontrollable path transition

Testability report lists transition paths and testable and untestable status

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)

978 689 7286
408 625 7618 (US West)
+49-2132-137485 (Europe)

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

Trademarks/Copyright ©2010 Avery Design Systems, Inc. All Rights Reserved.