Verilog-A Debug Tool: AHDL Linter

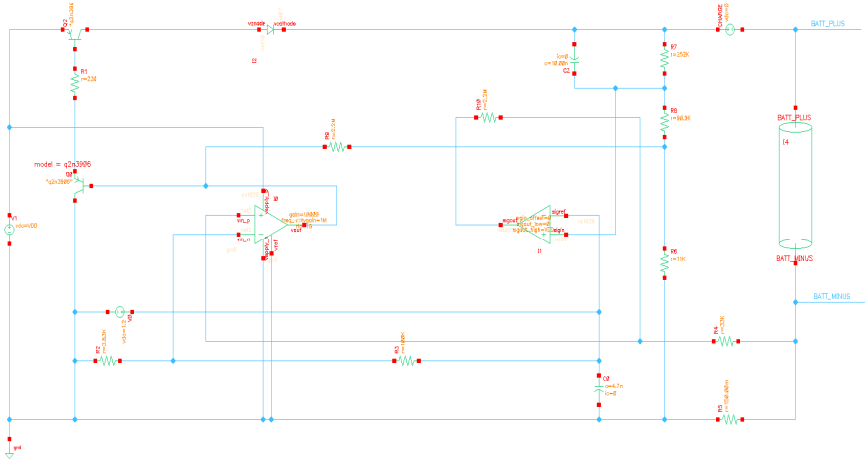
Jushan Xie, Qingping Wu, Art Schaldenbrand, and Andre Baguenier

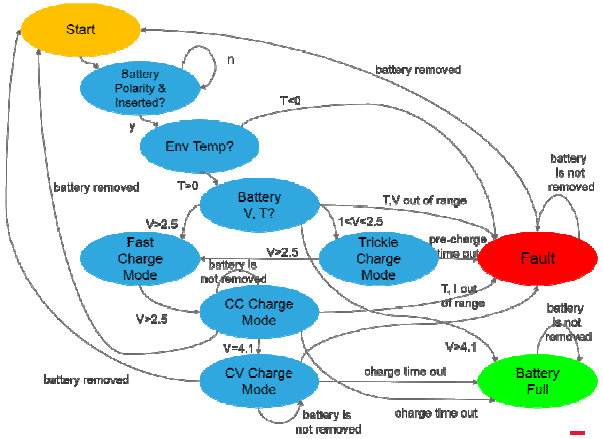
Cadence Design Systems, Inc.
Dec. 6, 2017

cadence®

Beauty of Using Behavioral Modeling

- Behavioral modeling is the enabling technology for analog functional verification
- Consider a Li Ion battery charger, it is an analog circuit and a complex state machine
  - Just performing SPICE simulations to validate datasheet specifications is not enough
  - We need to verify the behavior of the overall system
  - Verifying behavior at transistor level is expensive, using behavioral modeling is the tool that enables functional verification
- Using behavioral modeling languages provides a powerful and flexible way to develop customized models
  - The Verilog-AMS LRM2.4 provides a standard easy readable language which is supported by multiple simulation vendors
  - Automatic calculation of derivatives





cadence

Advantage and Challenge with Verilog-A

- Advantage
  - High level behavior model allows to use simple equation to describe complex issues
  - Standard language, LRM 2.4
  - Easy to learn and easy to use
  - Flexible for both analog and AMS application
  - Much more simple than Spice C code modeling
  - Doesn't need to have derivative (comparing to Spice C code model)
- Challenge
  - Verilog-A cannot auto-detect and fix any model equation issue
    - Model discontinuity, like if-else, event, …
    - Improper function usage, @cross, transition, …
    - Model behavior correctness
  - It is hard to debug
    - No debugger, like C debug tools
- AHDL Linter tries to help user to debug Verilog-A model

Example of Traps when Coding in Verilog-A

- Performance
  - Breakpoints (BoundTimeStep)
    - Timer, transition, $bound_step, etc can set breakpoints and limit step size
    - Improper use of breakpoints may cause too many bound time steps
    - Issues often occur when the time step is set too small or it used on a continuous signal during transition
  - RejectTimeStep
  - Discrete inputs used with cross or above commands may cause lots of rejections

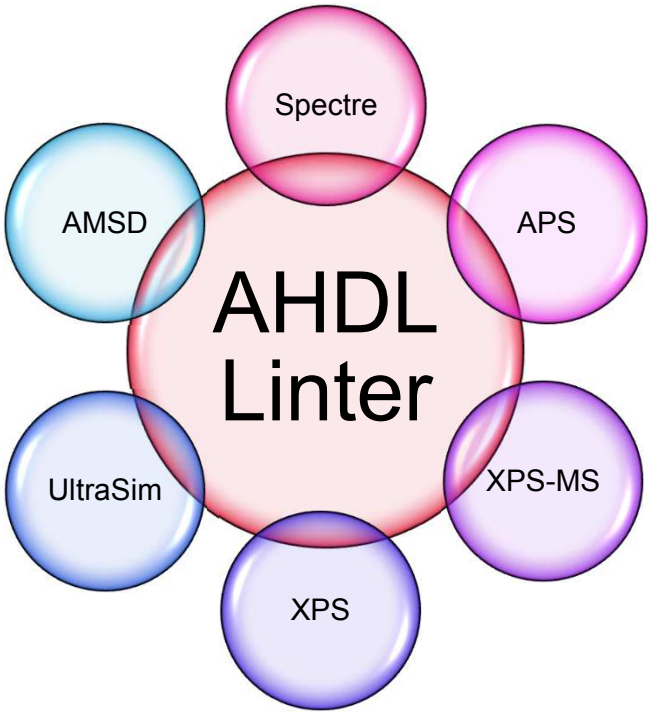
- Discontinuity / convergence
  - If-else, case, events, integer, floor, ceil commands can introduce discontinuities
  - Conductance discontinuity may cause Newton convergence failures

AHDL Linter Overview

What does the AHDL linter do?

- Static and dynamic linter features.
  - –Static linter is done during parser stage (before analysis):
    - – Mainly check local statement and syntax usage
  - –Dynamic linter during analysis
    - – Check various issues that may cause convergence or performance issues
- The current AHDL Linter does not check:
  - Verilog-AMS Real Number Models (wreal)
  - SystemVerilog Real Number Models

Which simulators supports AHDLLinter?

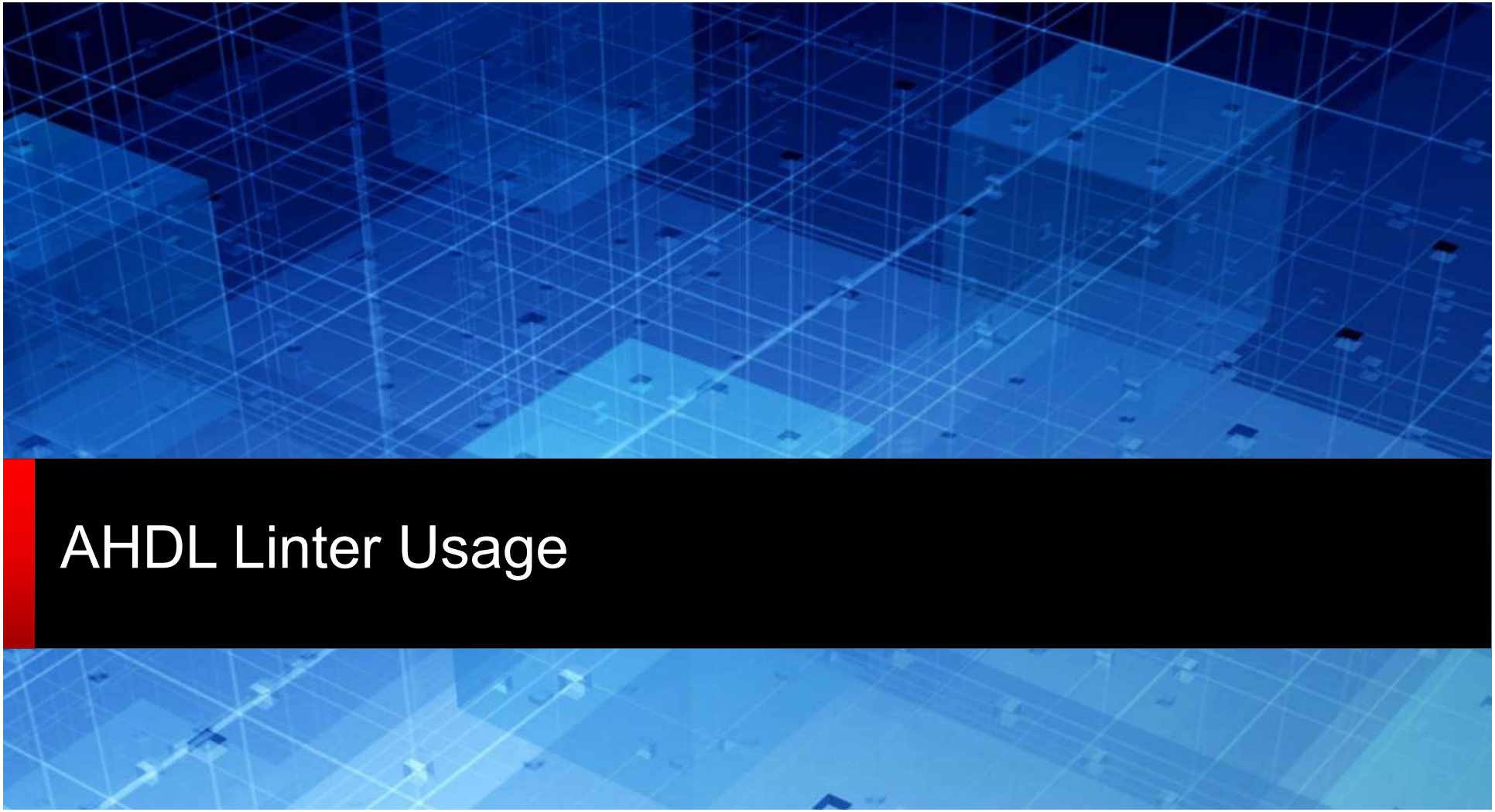


Spectre supports the AHDLLinter, since Spectre 16.1 release.

AHDL Linter Language Support

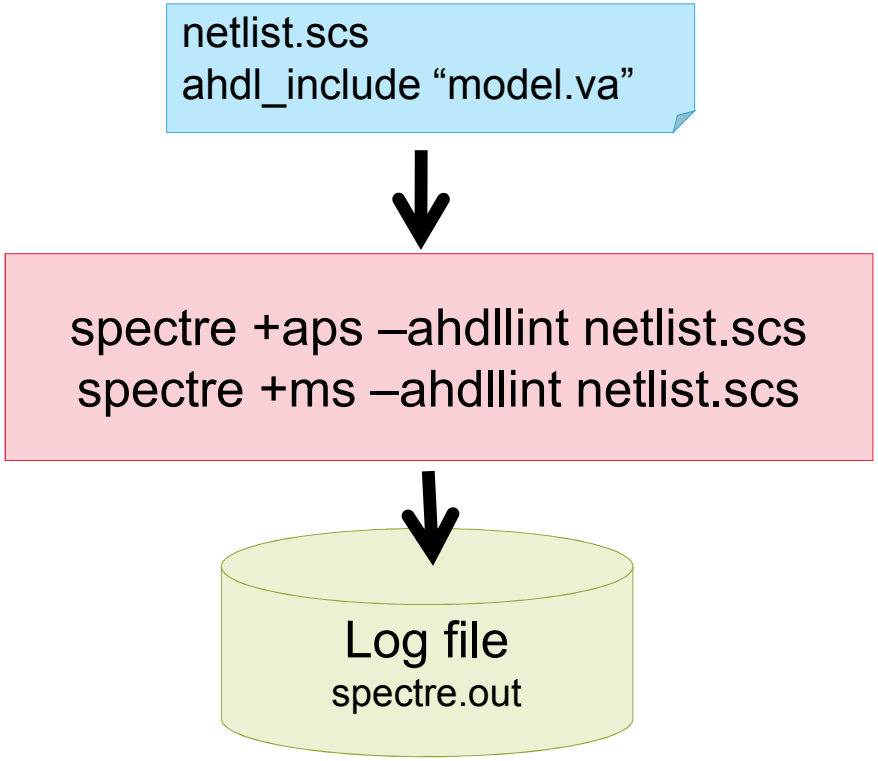
Linter	Verilog-A	Verilog-AMS	VHDL-AMS
Static	✓	✓	
Dynamic	✓	✓	✓

- The AHDL Linter performs linter checks on behavioral models written in both analog and mixed-signal languages

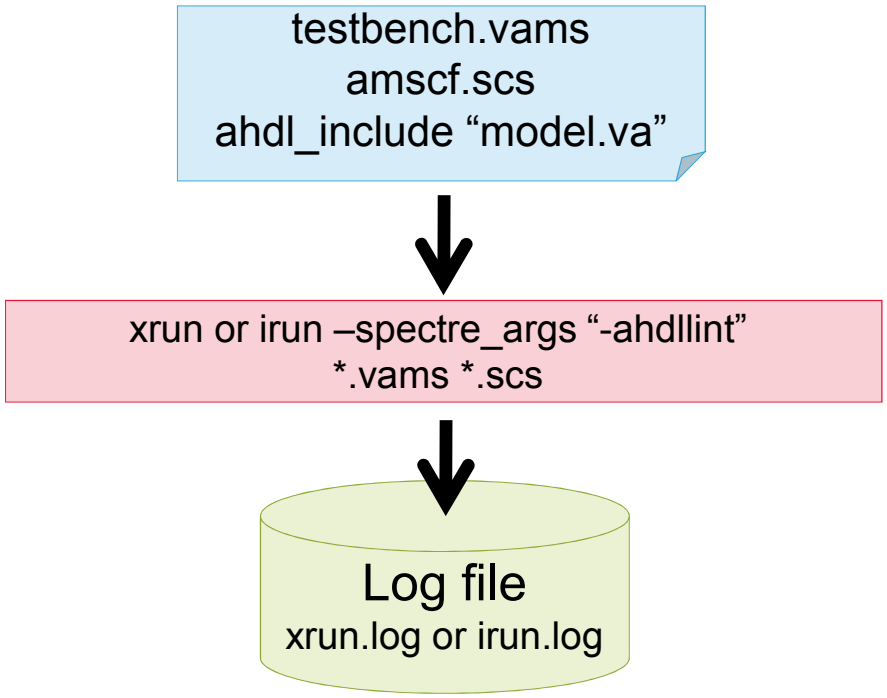


AHDL Linter Usage

Running AHDL Linter in Spectre, APS, XPS MS, XPS



Running AHDL Linter in AMS-D



How to run AHDL Linter

- Both static and dynamic check,
  - Add “-ahdllint” option to your original command line

    % spectre -ahdllint netlist.scs
    % spectre +aps -ahdllint netlist.scs
    % spectre +xps +cktpreset=sram -ahdllint netlist.scs
    % ultrasim -ahdllint netlist.scs
    % xrun or irun –spectre_args “-ahdllint” *.vams *.scs

- Static only

  spectre -ahdllint=static test.va
  test.va is a pure Verilog-A model
  Just check a Verilog A module without running a simulation

More AHDL Linter Usage

- AHDL Linter help:

  spectre –help ahdllint

- More options to run AHDL Linter

  spectre –ahdllint= [no | warn | error | force] test.scs

  -ahdllint=no/warn/error/force
  - no turn off ahdl linter
  - warn treat issues as warning
  - error treat issues as error
  - force see details in manual

AHDL Linter Outputs

AHDL Linter Results Report

- The AHDL Linter provides a summary at the end the log file
  - The summary indicates all issues identify during simulation and the number of occurrences of the issue simplifying the
  - The summary allows designers to easily find most problematic issues in their Verilog A modules

```
Warning from spectre at time = 200 us during transient analysis `tran1'.
    WARNING (AHDLLINT-8007): "test4_6.va" 18: qam: The transition function in
the model has imposed a time step size of (1 ps), which is too small and might
slow down the simulation. Increase the applicable tolerances and/or
        expression values of the function to avoid such small step size, and for
faster simulation.
************ AHDL Lint Summary ************
Number of accepted steps = 101
#Steps Type         Instance:File:Line No.
   118 transition   qam:test4_6.va:19
     4 transition   qam:test4_6.va:19
     4 transition   qam:test4_6.va:19
     4 transition   qam:test4_6.va:18
```

Understanding the AHDL linter messages

- There are 2 types of linter message
  - Static linter message
    - Issue found during compilation stage
    - Check for local statement, with no dependency check
    - Their message ID starts from #5000

    WARNING (AHDLLINT-5008): "/home/andre/lab_ahdllinter/source/clk_gen.vams", line 26: Detected discrete expression on the right hand side of the a contribution statement. It is recommended that you apply the transition function on discrete values by specifying the transition rise and fall time when the logic of discrete expression value changes.

  - Dynamic linter message
    - Issue found during simulation stage
    - Their message ID starts from #8000
    - Check for offending issues during simulation, that causes convergence or performance issues

    WARNING (AHDLLINT-8006): "/home/andre/lab_ahdllinter/source/clk_gen.vams", line 29: top.10._cds_internal_clock_gen_: Detected signal glitches/pulses in transition expr with pulse width smaller than tr or tf.

Help Using the AHDL Linter

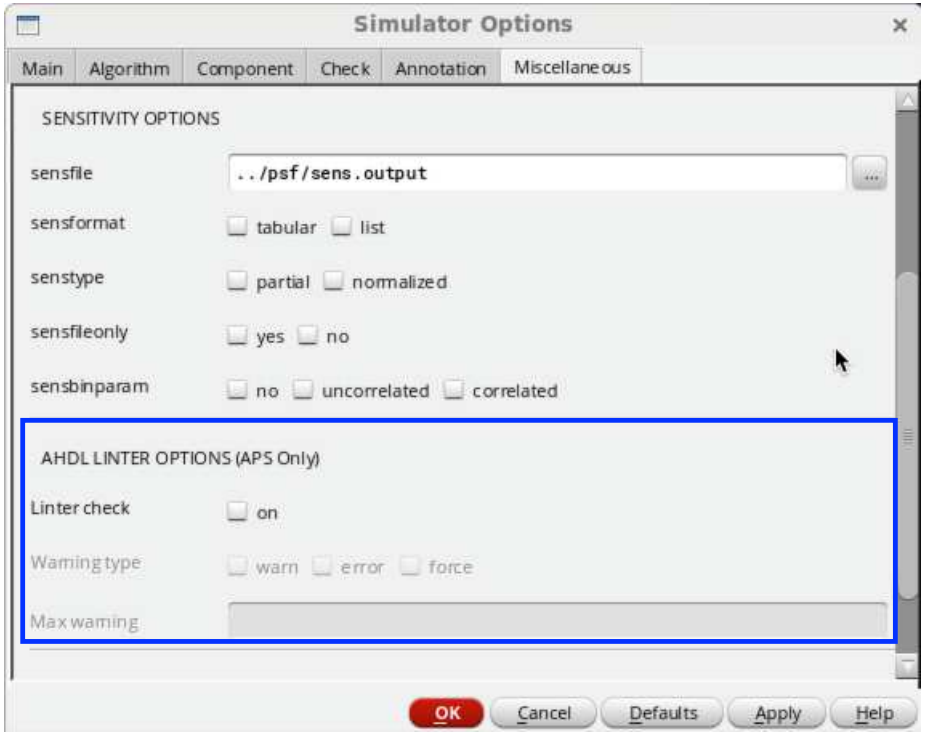
Command line support with the ‘ahdlhelp’ utility

- You can use the ‘ahdlhelp’ utility to view the extended help on various messages reported by AHDL Linter. Give the Linter ID as the argument.
- Example:

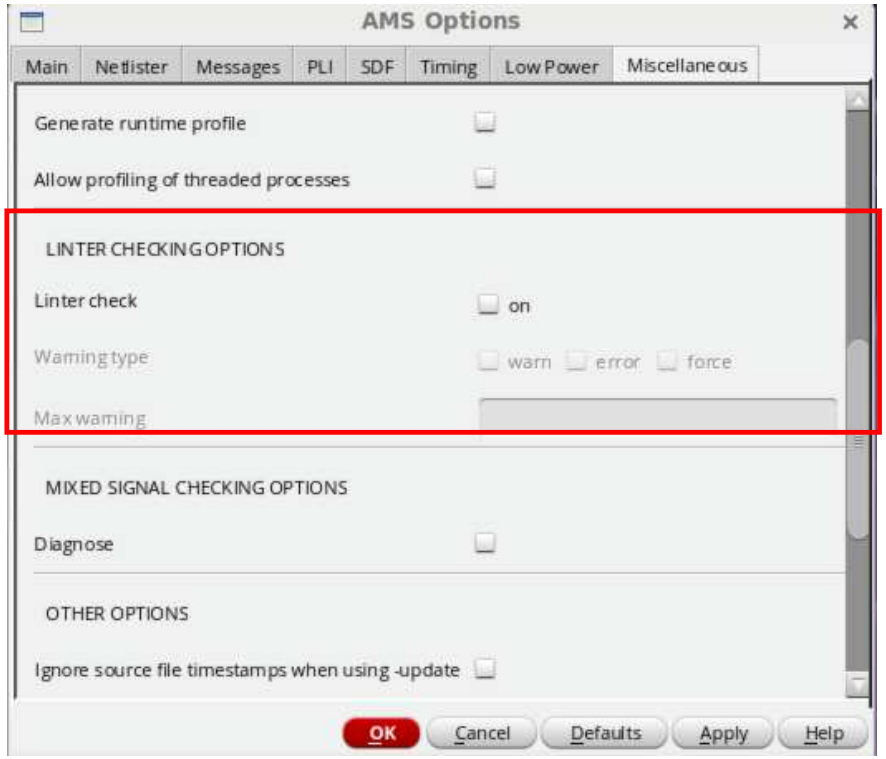
```
% ahdlhelp 5012

AHDLLINT-5012: Detected $abstime in an equality expression inside a conditional
statement.
Analog simulations select timepoints using a variable timestep size algorithm based
on accuracy considerations, so the value of "time" only changes between discrete
real values. In order to perform an event at a particular time, the @(timer)
construct must be used to tell the simulator to step to a particular timepoint and
execute the desired code when that event actually occurs.
example:
if ($abstime==50n) xval=2;
solution:
@(timer(50n)) xval=2;
```

AHDL Linter Integration

Enable the AHDL Linter in ADE



- Spectre, Spectre APS, Virtuoso UltraSim, Spectre XPS-MS, and Spectre XPS
- Linter of Verilog A

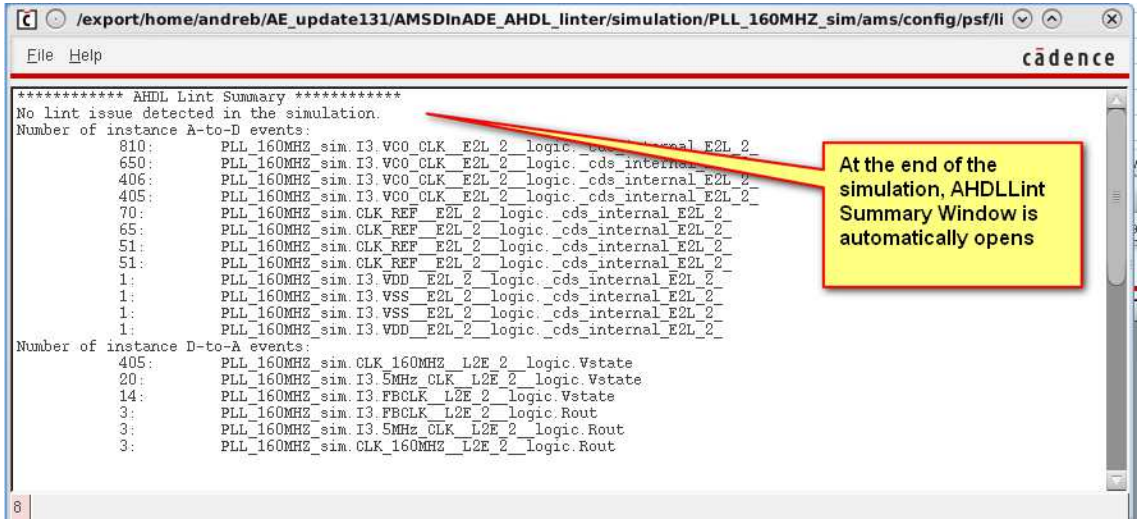


- Spectre AMS Designer
- Linter of Verilog A, Verilog AMS, and VHDL AMS

Using the AHDL Linter with AMS Designer

1. Simulation setup for AMS Designer in ADE Batch mode:

2. Example of a lint report, found at the end of the log file



cadence

Samples

Static linter sample

```
@(cross(V(in) - vth1,0)) begin
      if ((V(in) - vth1) >0)
         tmp1=1;
      else
         tmp1=0;
   end
```

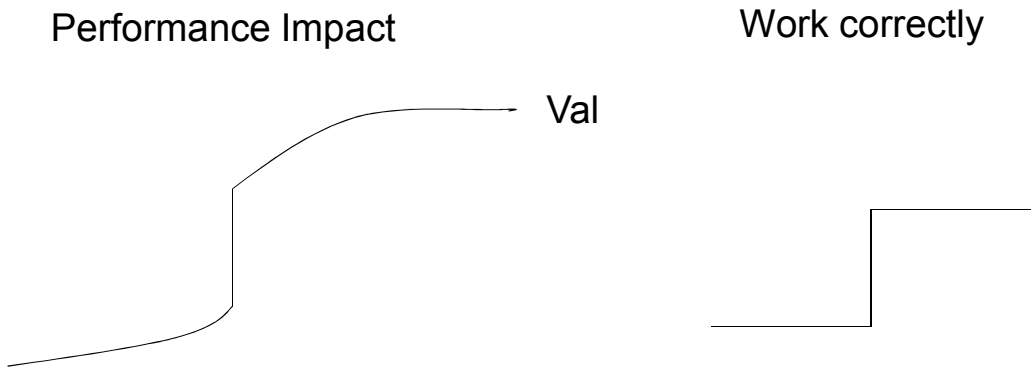
Suggested:

```
@(cross(V(in) - vth1,1)) tmp1=1;
@(cross(V(in) - vth1,-1)) tmp1=0;
```

WARNING (AHDLLINT-5022): "input.va", line 54: Encountered an if-else statement (a level trigger) inside a cross/above event (an edge trigger) for the same branch, 'in'. When a cross/above event is triggered and the value of the analog signal equals the threshold value, the if-else statement may return unexpected results.
'ahdlhelp 5022' will give suggestions on how to modify in details.

Dynamic linter sample

```
if ((V(ref) >0.5) )
    val=V(in1);
else
    val=V(in2);
end
V(out)<+transition(val,0,trise);
```



- WARNING (AHDLLINT-8004): “input.va" 12: ts1: Non-piecewise constant argument is detected in the transition filter. The transition filter can only be applied to piecewise constant signals. The argument must be redefined to ensure that the transition filter applies only to a piecewise constant signals.

Conclusion

Conclusion

- Using both diagnose and AHDLLinter you can speed up
  - Spectre APS and XPS MS verification
  - AMS Verification
- Diagnose and AHDLLinter enable anyone to analyze the coding style and provide suggestions for model coding style enhancement, for
  - Verilog-A,
  - Verilog-AMS,
  - VHDL-AMS behavioral models.
- Diagnose operates already for SystemVerilog-AMS connect module.
- Diagnose and AHDLLinter enable deterministic profiling during TB verification.
- Customers reported multiple times Cadence is the only one delivering such technology.

Questions ?


cadence®