



CCS Noise

Technical White Paper

Version 1.1

Abstract

This document describes the Synopsys CCS Noise model for cell-level noise analysis.

Version 1.1

11/21/05
1

Copyright Notice and Proprietary Information

Copyright © 2005 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page: "This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPTSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Arcadia, C Level Design, C2HDL, C2V, C2VHDL, Cadabra, Calaveras Algorithm, CATS, CSim, Design Compiler, DesignPower, DesignWare, EPIC, Formality, HSPICE, Hypermodel, I, iN-Phase, in-Sync, Leda, MAST, Meta, Meta-Software, ModelAccess, ModelTools, NanoSim, OpenVera, PathMill, Photolynx, Physical Compiler, PowerMill, PrimeTime, RailMill, Raphael, RapidScript, Saber, SiVL, SNUG, SolvNet, Stream Driven Simulator, Superlog, System Compiler, Testify, TetraMAX, TimeMill, TMA, VCS, Vera, and Virtual Stepper are registered trademarks of Synopsys, Inc.

Trademarks (™)

abraCAD, abraMAP, Active Parasitics, AFGen, Apollo, Apollo II, Apollo-DPIL, Apollo-GA, ApolloGAIL, Astro, Astro-Rail, Astro-Xtalk, Aurora, AvanTestchip, AvanWaves, BCView, Behavioral Compiler, BOA, BRT, Cedar, ChipPlanner, Circuit Analysis, Columbia, Columbia-CE, Comet 3D, Cosmos, CosmosEnterprise, CosmosLE, CosmosScope, CosmosSE, Cyclelink, Davinci, DC Expert, DC Expert Plus, DC Professional, DC Ultra, DC Ultra Plus, Design Advisor, Design Analyzer, Design Vision, DesignerHDL, DesignTime, DFM-Workbench, DFT Compiler, Direct RTL, Direct Silicon Access, Discovery, DW8051, DWPCI, Dynamic-Macromodeling, Dynamic Model Switcher, ECL Compiler, ECO Compiler, EDAnavigator, Encore, Encore PQ, Evaccess, ExpressModel, Floorplan Manager, Formal Model Checker, FoundryModel, FPGA Compiler II, FPGA Express, Frame Compiler, Galaxy, Gatran, HDL Advisor, HDL Compiler, Hercules, Hercules-Explorer, Hercules-II, Hierarchical Optimization Technology, High Performance Option, HotPlace, HSPICE-Link, iN-Tandem, Integrator, Interactive Waveform Viewer, i-Virtual Stepper, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, JVXtreme, Liberty, Libra-Passport, Library Compiler, Libra-Visa, Magellan, Mars, Mars-Rail, Mars-Xtalk, Medici, Metacapture, Metacircuit, Metamanager, Metamixsim, Milkyway, ModelSource, Module Compiler, MS-3200, MS-3400, Nova Product Family, Nova-ExploreRTL, Nova-Trans, Nova-VeriLint, Nova-VHDLint, Optimum Silicon, Orion_ec, Parasitic View, Passport, Planet, Planet-PL, Planet-RTL, Polaris, Polaris-CBS, Polaris-MT, Power Compiler, PowerCODE, PowerGate, ProFPGA, ProGen, Prospector, Protocol Compiler, PSMGen, Raphael-NES, RoadRunner, RTL Analyzer, Saturn, ScanBand, Schematic Compiler, Scirocco, Scirocco-i, Shadow Debugger, Silicon Blueprint, Silicon Early Access, SinglePass-SoC, Smart Extraction, SmartLicense, SmartModel Library, Software, Source-Level Design, Star, Star-DC, Star-MS, Star-MTB, Star-Power, Star-Rail, Star-RC, Star-RCXT, Star-Sim, Star-SimXT, Star-Time, Star-XP, SWIFT, Taurus, Taurus-Device, Taurus-Layout, Taurus-Lithography, Taurus-Process, Taurus-Topography, Taurus-Visual, Taurus-Workbench, TimeSlice, TimeTracker, Timing Annotator, TopoPlace, TopoRoute, Trace-On-Demand, True-Hspice, TSUPREM-4, TymeWare, VCS Express, VCSi, Venus, Verification Portal, VFormal, VHDL Compiler, VHDL System Simulator, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.
SystemC is a trademark of the Open SystemC Initiative and is used under license.
ARM and AMBA are registered trademarks of ARM Limited.
All other product or company names may be trademarks of their respective owners.
Printed in the U.S.A.

1 Introduction

CCS Noise is a new advanced current-based driver model that enables accurate noise analysis with results very close to SPICE simulation. It not only precisely models injected crosstalk noise bumps, but also allows more advanced analysis, such as propagated noise bumps and the driver weakening, without significant characterization effort.

With CCS Noise, the noise immunity of the cells can also be obtained during the analysis by using the actual noise bump waveforms without the additional need for separate characterization. This dynamic computation of noise propagation and noise immunity enables noise library characterization to be 100 times faster than NLDM noise characterization.

There are three main components to noise calculation: driver modeling, receiver modeling, and reduced order modeling of parasitics. The focus of this paper is the CCS Noise driver model; information about the advanced receiver modeling can be found in the CCS Timing Technical White Paper [1].

2 CCS Noise Driver Model

Current-based driver models are necessary for very accurate crosstalk noise analysis. The desired cell current model for crosstalk noise analysis must be able to interface with arbitrary input noise waveforms and arbitrary coupled load interconnect networks.

CCS Noise is an advanced current driver model that captures both static and transient characteristics of the cell. The static component of the CCS Noise model consist of a current table as a function of input and output voltage levels which can be provided through an efficient DC analysis known as a basic ViVo model (V_{in}/V_{out}).

The key advantage of CCS Noise is that it uses several dynamic parameters to model the dynamic response of the cell that a static current table cannot capture. The dynamic parameters are extracted from transient analysis measurements that record the response of the cell to certain input transitions and noise bumps.

CCS Noise can accurately model all crosstalk noise analysis effects including noise calculation, noise propagation, driver weakening, and combined noise propagation and noise injection. Here, noise calculation

refers to the problem of calculating an aggressor net's injected noise bump, assuming the victim driver is quiet. Noise propagation refers to the problem of propagating a noise bump through a circuit cell, assuming there is no coupling in the cell output net. Combined noise propagation and noise injection analysis refer to the general case where there is noise propagation through the victim driver and noise injections by aggressors of the victim net. Driver weakening can be considered as a special case of noise combination, where the propagated noise is very small by itself, but significantly reduces the effective driving strength of the victim driver due to increased injected noise bump size. Extensive studies have shown that CCS Noise is considerably more accurate than other models for these crosstalk noise analysis tasks, especially because of the CCS Noise dynamic parameters. For example, Fig. 3 shows that propagated noise waveforms computed using CCS Noise match SPICE waveforms much better than those computed using the basic ViVo driver model without the dynamic parameters.

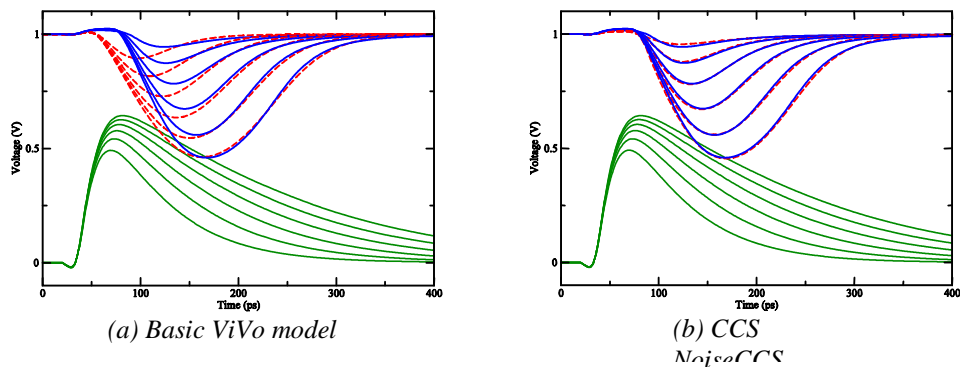


Fig. 3. Waveform propagation comparison (Green curves are input waveforms; blue curves are SPICE output waveforms, and dashed red curves are waveforms computed using models).

The CCS Noise analysis flow can take advantage of the advanced CCS timing receiver model when such data are available in the library. More details about the CCS timing receiver model can be found in the CCS Timing White Paper [1]. For the purpose of noise analysis, the CCS timing receiver model is more accurate than single-valued min/max rise/fall pin capacitance because it includes the dependency of effective receiver pin capacitance on input transition time and receiver output load capacitance. Furthermore, the CCS Noise analysis engine explicitly includes the varying effective input capacitance effect of victim receivers; no extra receiver characterization is required.

As discussed earlier, the ViVo-based current driver models best capture the behavior of a single channel-connected block (CCB). For complex circuit cells having more than one CCB, the transistor-level netlist of the circuit cell needs to be broken into multiple CCBs and a CCS Noise model for each of those CCBs. The transistor-level netlist breaking and CCS Noise model parameter extraction are performed during cell characterization.

Once characterized, CCS Noise model data are stored either on a timing arc or on a pin in the cell library, depending on the topology of the circuit netlist. For circuit cells having a single CCB for an input-output pin pair, one CCS Noise model is extracted and stored on a timing arc. Such single-stage cells include most inverters, NAND gates, NOR gates, AOI gates, OAI gates, etc. For circuit cells having two subsequent CCBs, two CCS Noise models are stored on a timing arc. Such two-stage cells include most of the buffers, AND gates, OR gates, AND-OR gates, and OR-AND gates, etc. For circuit cells having three or more CCBs, including most of the flip-flops, full adders, macro blocks, etc., CCS Noise model data are stored on pins.

3 CCS Noise Analysis

Cross-coupling between a victim net and aggressor nets causes a noise bump on the victim net when the aggressors switch. As shown in Fig. 5, when the noise bump is large enough and the propagated noise bump is latched by the sequential cell, it will cause a functional error by changing the logic value of the sequential cell. It will also increase the dynamic power consumption induced by noise bumps. This issue can be addressed by reporting potential violation sources in *report_at_source* mode. In the *report_at_source* mode, any noise bump that fails the noise immunity criteria is reported. You should fix all the reported violations to address all the potential violation sources.

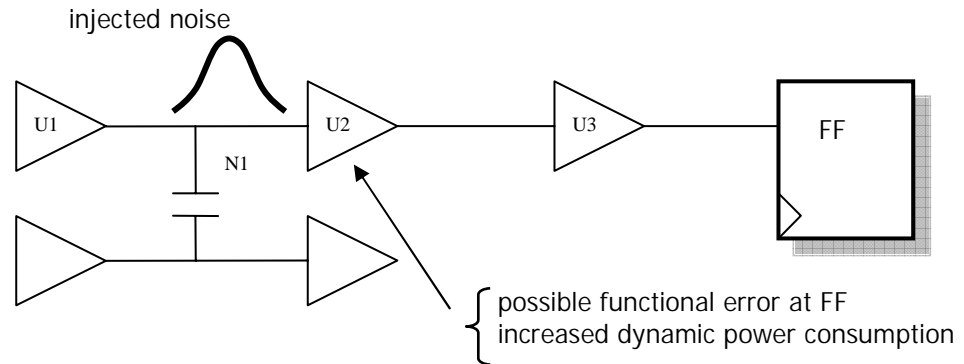


Fig. 5. A violation reported at the *report_at_source* mode.

Alternatively, you could fix just those noise bumps that can propagate all the way to an endpoint such as the D pin of a flip-flop; this method of analysis is referred to as *report_at_endpoint* mode.

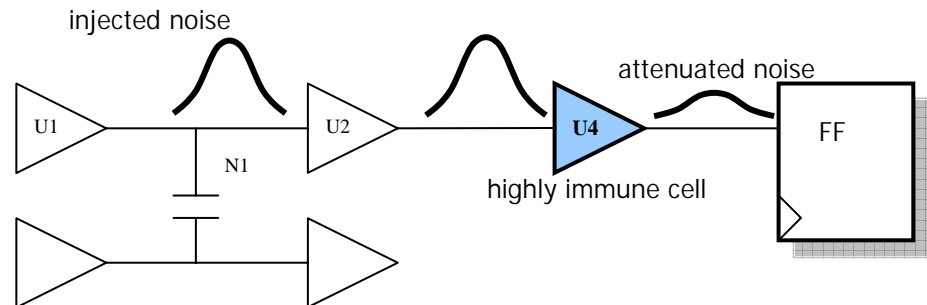


Fig. 6. The injected noise attenuates when it propagates.

As an example, consider the scenario shown in Fig. 6, where U4 is a highly noise immune cell; that is, it can tolerate a certain amount of noise without causing a failure. Its propagated noise is also significantly attenuated such that no functional error occurs in FF. In the *report_at_source* mode, the input pin of U2 would be reported as a violation because the noise bump exceeds the noise immunity limit. However, in the *report_at_endpoint* mode, no violation will be reported because the injected noise bump attenuates in a later stage and does not cause any functional error at the endpoint. Therefore, when users care only about the failures at the inputs of sequential cells, the

report_at_endpoint mode analysis can potentially report fewer noise violations .

Fig. 7 shows an example of the *report_at_endpoint* mode analysis. In this figure, a source of violation is the victim net N1, where the noise bump is created due to the cross coupling. A noise endpoint is any input pin of the sequential cell where the noise propagation stops; in this case, the input pin of FF

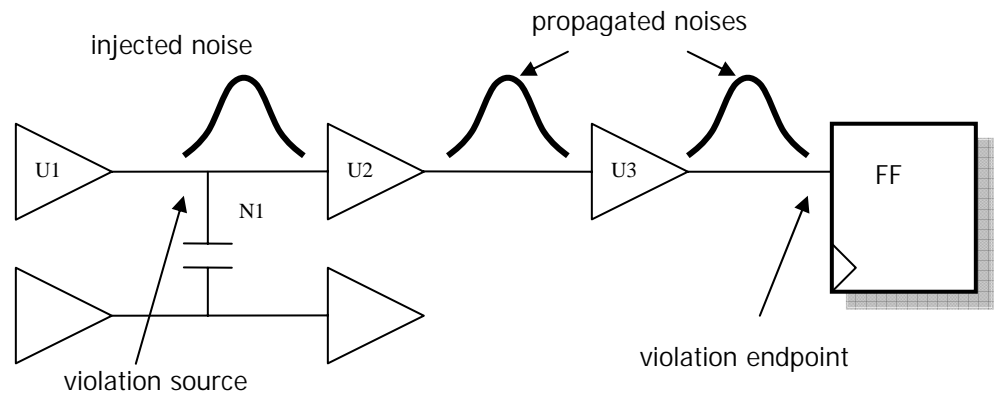


Fig. 7. Source of violation vs. violation at an endpoint.

In the *report_at_endpoint* mode, it may be difficult to determine the violation sources just by viewing the reported violations at noise endpoints, such as the input pins of sequential cells. But the violation sources can be found by searching backward through the fan-in logic cones from the violation endpoints, as shown in Figure 8.

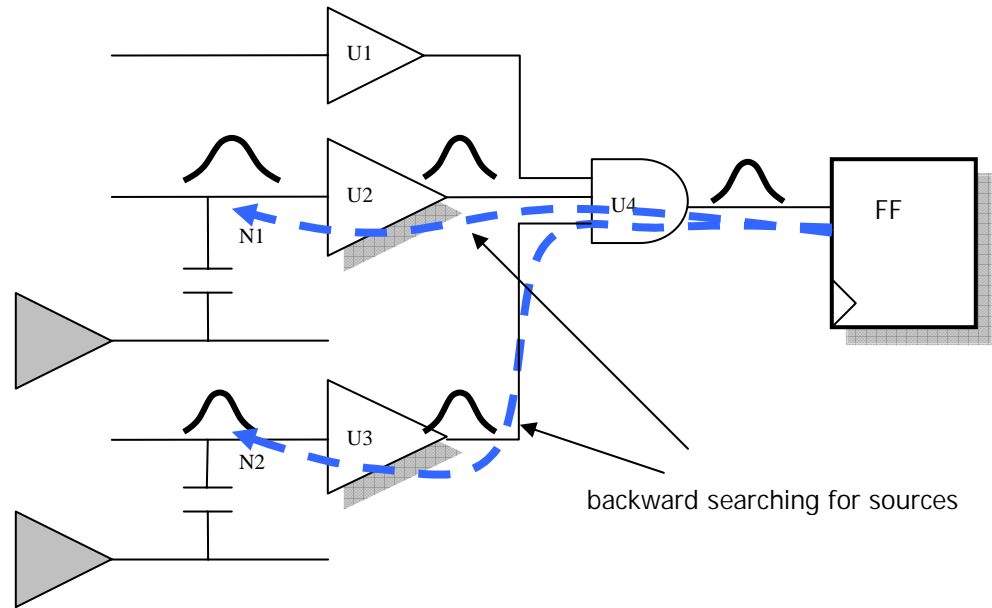


Fig. 8. Reporting the sources of violations in the `report_at_endpoint` mode analysis.

Fig. 8 shows how the violation sources can be found given the violation reported at the endpoint. The backward search starts from the endpoint, and traces the fan-in logic cone until it hits the first net where the injected noise exceeds the noise immunity. In this way, all possible sources of violations can be found. In Figure 8, the violation has occurred at the D pin of FF and `report_at_endpoint` mode reports nets N1 and N2 as the violation sources. So when you fix the violations on N1 and N2, the violation at the endpoint will be fixed as well.

4 Noise Slack Calculation

A CMOS circuit cell can tolerate a certain amount of noise without causing a failure at the cell output; this characteristic is called noise immunity. CCS Noise information can be used to calculate the noise immunity of a cell on the fly.

Using CCS Noise and the actual shape of the input noise bump, the noise immunity of the cell can be computed as well as the noise slack;

that is, the amount of noise height that needs to be added to the noise bump to cause a failure. When the noise slack is negative, it means that the noise bump has exceeded the noise immunity parameters for the cell. When the noise slack is positive, the noise bump is within the noise immunity parameters.

5 Results

Fig. 9 shows the results of a crosstalk noise correlation with SPICE. The test circuit has two aggressor nets coupled to a victim net and the victim net has a fan-out net for noise propagation analysis.

Fig. 9 (b) compares CCS Noise and SPICE waveforms for a typical case. The difference in waveforms is barely discernible at the resolution of the plot. The cell driving strengths: that is, the input transition times of the aggressor drivers and the lengths of interconnects, have also been varied to cover all reasonable scenarios for accuracy correlation.

The noise bump height correlation at point A (for noise calculation) is shown in Fig. 9 (c). The correlation at point B (for both noise calculation and propagation) is shown in Fig. 9 (d).

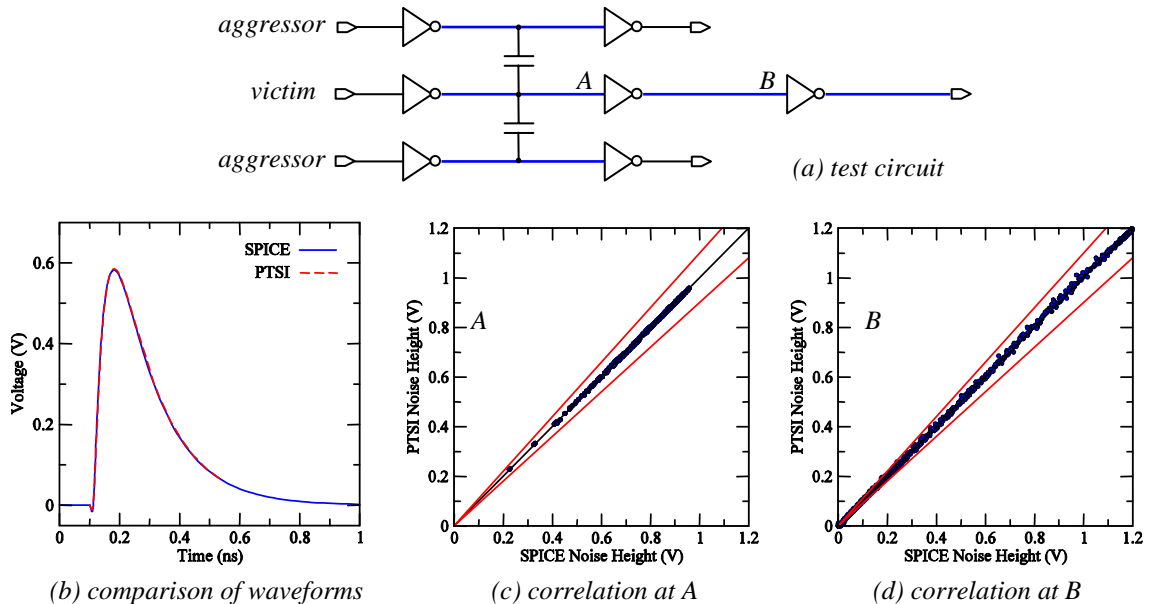


Fig. 9. Comparison of CCS Noise analysis results with SPICE.

Note that a coupled interconnect network for crosstalk noise analysis is a nonlinear system, therefore the law of linear superposition does not apply. CCS Noise enables the combined noise bump by switching all aggressors. The sum of individual noise bumps may not match the total noise bump.

Table 1 and Fig. 10 show details of static noise analysis results with customer designs.

Table 1 shows how the pessimism is reduced in different noise analysis modes by counting the number of violations reported. The fourth column shows the number of noise bumps whose heights are higher than 40% of V_{DD} . However, after the noise immunity and noise slack for those noise bumps are calculated, no violations are reported because the noise bumps do not exceed their noise immunity. Only the noise bumps whose noise slacks are negative are reported in the *report_at_source* column. Moreover, if you exclude any noise bumps that attenuate during their propagation and instead report only the noise bumps that reach the endpoints and cause functional errors, pessimism can be further reduced. These numbers are reported in the *report_at_endpoint* column. As shown in the table, *report_at_endpoint* mode analysis reports fewer violations compared with the number of noise bumps with their heights greater than 40% of V_{DD} . When the *report_at_endpoint* mode analysis is used, this pessimism reduction can save lots of time during ECO fixing stage, and significantly reduces the number of ECO fixing iterations.

Table 1. Number of static noise violations in customer designs.

Design	Number of gates	Library	Number of violations		
			height > 40% VDD	report_at_source	report_at_endpoint
Design A	325K	90 nm	288	31	4
Design B	71K	90 nm	353	51	12
Design C	195K	90 nm	161	45	1

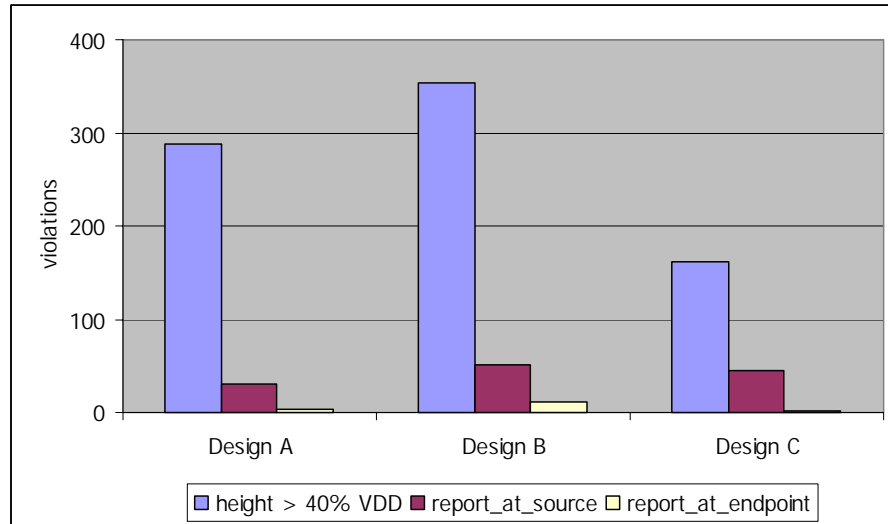


Fig. 10. Comparison of noise violation counts.

6 Summary

In this paper we have introduced the new advanced CCS Noise model and shown how it can be used to improve static noise analysis. CCS Noise provides for accurate noise calculation including the effects of noise propagation, so that the gate model can be characterized significantly faster than before. Results with various designs show that CCS Noise matches SPICE simulation results with excellent correlation.

7 References

[1] CCS Timing Technical White Paper.