

A verification methodology for large mixed-signal SoC designs using NanoSim-VCS

by Raul Salvi, Motorola Semiconductor Products Sector

November 2002

Abstract

With the increased emphasis on trimming production costs and time to market, first-pass silicon success has become a fundamental requirement for system-on-chip (SoC) designs. First-pass silicon success can only be achieved through a comprehensive methodology for verifying the functional, timing, and analog/digital signal interface of a design. This article addresses the challenges associated with verifying large mixed-signal system-on-chip (SoC) designs and offers a detailed verification methodology using the NanoSim™ integration with VCS™ from Synopsys. This article also presents the results achieved on a mixed-signal SoC design using this verification methodology.

Introduction

Top-level SoC verification of analog, RF and digital blocks is now a necessity to evaluate system performance and to ensure functional flow from one block to the next. Many opportunities exist for design errors due to interfacing and timing challenges both at the intra- and inter-chip levels.

Likewise, many aspects of the verification process need to be considered when attempting top-level verification, including:

- Levels of abstraction in a design, Verilog, Verilog-A or C models of blocks.
- Dynamic adjustment of the behavioral level of abstraction.
- Choosing design parameters to ease re-simulation without the need for a new netlist.
- Analog-to-digital and digital-to-analog interface between Verilog and transistor netlists.
- Drive-load interface modeling.
- The use of circuit partitioning to improve simulation time and convergence.
- The use of different transistor models for trading off accuracy and speed (and the use of model lookup tables).
- Efficiently manipulating and analyzing simulation data.

The biggest advantage of top-level SoC design verification is the ability to take large blocks of both digital and analog designs and simulate them as a system. Until this is done, many design flaws can go unnoticed. To achieve top-level SoC verification, designers can leverage the NanoSim analog circuit engine transparently from within the VCS simulator. The mixed-signal simulation process requires that all of the simulators used in the simulation session maintain uniform synchronization. The VCS simulator is the master simulator in this environment, and controls the advancement of time and therefore the synchronization of the two simulator engines. Using this NanoSim integration with VCS enables the designer to access all of the features that VCS and NanoSim offer individually (Figure 1).

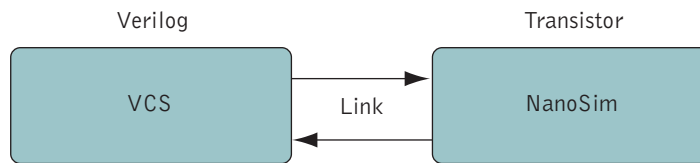


Figure 1. The NanoSim integration with VCS for SoC verification.

Challenges of mixed-signal SoC verification

Timing verification at interfaces

Usually, both static and dynamic timing are investigated extensively at the block level during design construction. Timing is critical with digital circuits, since the designer must contend with clock skew, race conditions, meta-stability and setup/hold requirements. Many of these performance factors are readily evaluated with an all-digital simulation environment.

Likewise, with analog blocks, attributes such as propagation delays, startup/shutdown times and frequency response of mixed-signal closed loops must be evaluated. The problem with past evaluation methods is that system-level block timing was not possible for mixed-signal systems where analog and digital blocks have timing dependencies

A key to evaluating and controlling system timing is being able to evaluate digital-to-analog as well as analog-to-digital signal interfaces in real time. Mixed-signal simulations allow analog blocks to correctly load digital interfaces as well as identify level shifter delays, which may otherwise go unnoticed.

To address these challenges, there are commands available with NanoSim-VCS, such as **set_sim_tres**, **set_sim_aspd**, **set_sim_aesv**, **set_sim_spd**, **set_sim_tup**, **set_sim_esv** and **set_print_tres**, which allow the user to trade off timing accuracy at both the system and block level. These commands also help the user to control the amount of data stored and the frequency of progress reports, and to offer some control over tradeoffs between simulation speed and accuracy.

These commands can reduce both the amount of data stored and the simulation time by several orders of magnitude, while maintaining the required accuracy. Obviously, simulation accuracy and simulation times are inversely proportional and the designer will need to make the tradeoff. For example, simulations done on a large mixed-signal design could take days to run due to the use of high-accuracy switches and high-resolution settings. Many simulations done at Motorola showed that the required accuracy could be decreased with no degradation in test integrity. This reduced simulation time to just hours. Furthermore, changes in time-step and print commands can cut the size of data files by 10 times with no loss in the readability of waveforms. When disk space is at a premium, these options are valuable.

Digital/analog signal interfacing

NanoSim as a standalone product will handle digital/analog interfaces transparently. However, as SoC designs get larger and larger, memory requirements and simulation speed can become excessive. The NanoSim integration with VCS allows the digital portions of the design to be simulated by the VCS engine, while the custom or analog portions of the design can be simulated at the transistor level with the NanoSim engine.

The interface between analog and digital blocks is an important one. The first step required to perform NanoSim-VCS simulations is to partition the design. A partition file is used to configure the mixed-signal design. The partition file allows the designer to specify one or more transistor-level blocks to be simulated with a transistor-level simulator like NanoSim. These blocks can be anywhere in the design hierarchy, except at the top level, and have to be instantiated by a Verilog block with a Verilog wrapper. The modules partitioned to the Synopsys transistor-level design simulator are ignored in the Verilog design database. The blocks chosen to partition to the transistor-level simulator will be instantiated as NanoSim components using the supplied netlist.

A Verilog wrapper (dummy module) is required for each block being simulated by the transistor-level simulator. The wrapper module(s) then must be instantiated inside a Verilog module. The instantiation of the wrapper module allows for port mapping, for example: **PLL I1(.reset(rst), .fanin(fin), .fanout(fout)).**

The wrapper module port names must match the port names in the **subckt** description of the transistor-level block to preserve the original hierarchy and connectivity through the Verilog wrapper.

Verilog modules not partitioned to NanoSim can be behavioral (Verilog-A or C), register-transfer level (RTL), or gate level. The syntax for VCS is the same as the syntax you use to run a pure, Verilog design simulation using VCS. All the same options and switches are available on the VCS command line (PLI calls, etc.). The only difference is the inclusion of the **+ad** argument to identify an analog/digital simulation.

A typical VCS command line calls the partition file (e.g. **vcsAD.init**) as:

```
vcs +ad[=vcsAD.init] top.v [-o=simv] [-R]
```

A typical partition file would then identify the analog simulator and block to be partitioned to the transistor simulator as:

```
choose nanosim -A -c config.nano -nspice options -nmcspice mylib.lib \  
-nmcspice .././spice/Tx_analog_full.ckt  
partition -cell subckt1
```

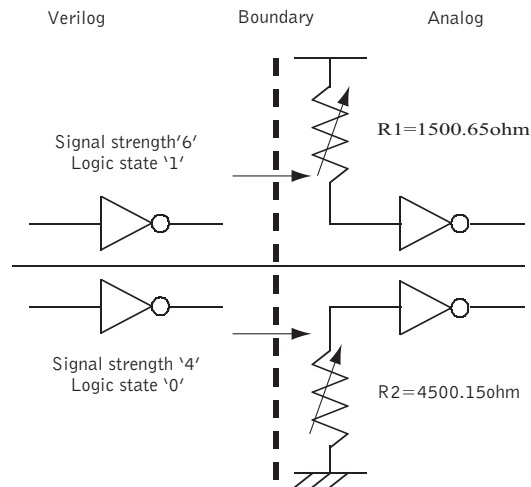


Figure 2. Using resistance map file.

The **choose** line specifies the transistor simulation engine, library, configuration files, and the netlist to partition. The partition file contains all the information and file references needed to partition and run a VCS-NanoSim mixed-signal simulation. During co-simulation, the run options specified on the **choose** command line are passed to the transistor-level simulator. NanoSim integration with VCS recognizes the partition file during the compile or partitioning stage and partitions the modules specified with regard to transistor-level design simulation. Multiple analog blocks can be partitioned at any level of the hierarchy except top level allowing for great flexibility. When creating Verilog wrappers for analog blocks, care must be taken when mapping digital nets (or wires) to analog signals to ensure there is no contention between digital and analog sources. Inadvertently mapping a digital driver to an analog source will not be flagged as an error since there is an interface resistor between the ports and can cause unexpected results. A mapping file (**names.map**) is created at compilation time, which helps the designer see signal interface points easily. A few example lines of interconnect points between analog and digital blocks – with a colon (:) in between – are shown below.

```
testbench.Harmony_top.TX_ADC_IOA_bias_1:testbench.Tx.tx_spi_adc_ioa_bias[1]
testbench.Harmony_top.TX_ADC_IOA_bias_0:testbench.Tx.tx_spi_adc_ioa_bias[0]
testbench.Harmony_top.Tx_DAC_vcc_bias:testbench.Tx_DAC_Bias_Vcc_i
testbench.Harmony_top.Tx_DAC_vcc_buf:testbench.Tx_DAC_Buffer_Vcc_I
```

Notice how buses are handled between analog and digital blocks. Bus notations can be set/modified with the **set_bus_format** command.

Resistance map files are used to establish drive strengths between analog and digital blocks. The resistance map file can be created by the user or the default resistance map file can be used. A user-created resistance map can be identified in the partition file. Otherwise, NanoSim-VCS will look for the default file with the name **rmapAD.init** in the directory **\$EPIC_HOME/etc/vcsace**. A graphical example of how this mapping works is shown in Figure 2.

Another aspect of particular importance is level shifting between blocks. NanoSim-VCS allows for mixed supplies of different blocks and allows for the verification of different signal levels between blocks. Be sure that power supplies are defined for the transistor-level blocks in the transistor netlist to avoid unexpected results. Signal levels for the digital-to-analog interfaces are automatically set, based on the supply of the transistor-level block. If desired, the **set_vec_opt** and **set_node_stv** commands can be used to modify these signal levels to reflect the supply of the digital block. This turns out to be a very powerful tool in analyzing correct signal levels between analog and digital blocks, and helps avoid floating-gate and forward-biased diode errors in the design.

Functional verification opportunities

With a large mixed-signal system, interaction between several digital and analog blocks must be verified for correct functionality. Many times, designers of individual blocks may overlook some interface or algorithm issue between one block and another. This is usually because individual designers are simulating individual circuits while taking note of system specifications. The problem with this approach is that system specifications are not always in tune with the realities of the individual block designs and thus interface issues may arise.

The methods outlined in this article allow the designer to approach all the blocks (both digital and analog) as a system with interdependencies that can only be verified from a top-level, mixed-signal viewpoint. Functionality of several mixed-signal blocks interacting together is key in verifying that silicon will be operational on a first pass. With today's time-to-market requirements, full functional performance is now a requirement for first-pass silicon.

Block and system-level simulations

Block-level abstraction

NanoSim integration with VCS provides the SoC designer with many options for modeling different blocks. NanoSim makes numerous tools available to ease trade offs between simulation accuracy and speed of the transistor blocks, and supports different levels of model abstractions. In addition to transistor and Verilog models, Verilog-A and C models for highly complex analog blocks are also available. Both languages allow the designer to write specific code that can model the analog behavior of blocks more efficiently with enough accuracy to verify other blocks in the system or systems. Figure 3 illustrates how these models are incorporated into the SoC verification flow using NanoSim integration with VCS.

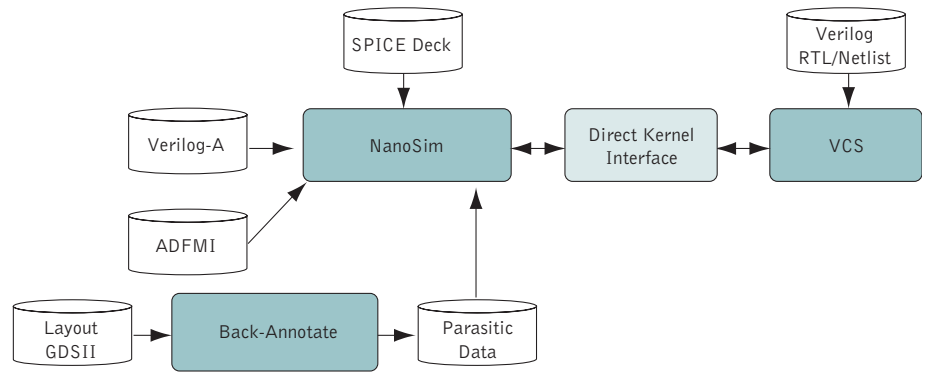


Figure 3. Incorporating models in the SoC verification flow.

To support additional trade offs, the NanoSim engine is capable of reading Verilog-A and C models created with NanoSim's Analog Digital Functional Modeling Interface (ADFMI). Therefore, blocks like a VCO in a control loop system such as a PLL can be modeled easily using Verilog-A. The netlist with a Verilog-A call such as **.hdl "blockname"** is all that is required since the simulator will then look for a Verilog-A description file called **blockname.va**. Obviously, all I/O parameters must match between the Verilog-A file and the netlist. The same holds true for the ADFMI models. ADFMI is a "C" based modeling language. The use of these higher level models can improve the speed of mixed clock rate systems considerably, and thus allow the designer to focus on certain critical blocks while not having to wait days for other modules to simulate at full transistor level in a co-environment. The use of a good hierarchical editor for choosing the level of abstraction of analog blocks makes this job a mouse click away. Even simple, less functional schematics can be chosen for speed optimization.

It is, however, important that higher level models be verified for needed accuracy. This can be accomplished by comparing the behavioral model to a more complex transistor-level model in a simple specific simulation to evaluate the model accuracy.

Block, intra-chip and inter-chip simulations

The size of the simulation environment can change drastically depending on the size and number of blocks to be simulated. This, of course, affects simulation time, which reveals the importance of using model and block abstracting to improve speeds. When simulating big mixed-signal designs such as closed loop systems where part of the loop is done with digital RTL and part of the loop is RF/Analog blocks, much insight can be gained on response times, stability and loop noise. The speed improvement that NanoSim's integration with VCS allows makes these simulations possible.

Another big plus with such high-level simulations is the ease with which the software to model and control these SoC designs can be developed and tested. By having a "virtual chip" months before actual silicon has arrived, software testing and debug can get a head start. Another advantage is the ability to accurately assess current drains in many modes of operation and thus give accurate battery life estimates early in the design cycle.

Results and conclusions

To date, Motorola has used NanoSim integration with VCS on many designs successfully. NanoSim-VCS has caught many costly bugs early in the design process, which reduced time to market and lowered design costs by eliminating multiple design passes. Furthermore, by catching these bugs early, software changes can be anticipated before silicon arrives, to accommodate known problems that will not be fixed in silicon.

Following is a list of some of the mixed-signal bugs identified before tape out of the last chipset:

- Transmit DAC signed digital input reversed
- Several SPI POR settings incorrect
- Several battery save leakage paths and floating gates
- Correction loop inversion in Chip2 block bit in digital control
- TX DACs and AFC DAC test lines were merged incorrectly
- AFC DAC digital control (RTL) SPI override bit 1F2 incorrect
- TXMON trigger digital control SPI 02E incorrect active state
- TXMON ADC output glitches when presented with a ramp input

The designs came back in silicon over 99 percent functional. Areas of non-functionality were completely due to poor test coverage. In other words, we did not ask the simulator the right questions. The value of these tools cannot be overstated, however many design schedules still fail to properly allot time for this critical phase in the development cycle. Many times, the time for system simulations is free since after schematic and digital RTL freeze dates, a month or more is required for analog and digital routing and timing verification. This is time well spent using this powerful engine to uncover system errors left unseen by specific block simulations. NanoSim integration with VCS enabled us to find and fix the problems above with fewer design cycles, allowing us to shorten our time to market.

By far the most powerful utilization of the tool is in signal interfaces and algorithm performance. This is usually the area that goes unnoticed in the lower level block simulations. NanoSim integration with VCS is able to find these issues due to the fact that they are not covered in lower level simulations.

Acknowledgements

Many thanks go to Rick Fernandez and Tom Quiroga of Motorola SPS for their much-needed guidance on using NanoSim-VCS on large SoC systems embedded in the Motorola Digital flow. The same applies to Pat Donahue and Roddy Cooper of Synopsys, who offered valuable help in the debugging of our simulation setups.

References

- [1] Synopsys, "NanoSim-VCS Direct Link Manual," TLD-2001.06, June 2001.
- [2] Donahue, Pat, "NanoSim, NanoSim-VCS, Verilog-A," SpringTech 2001 Seminar, May 2001.
- [3] Salvi, Raul, "Mixed-Signal Top-Level Functional Verification," Radio Hardware Architecture Steering Committee, May 2002.

SYNOPSYS®

700 East Middlefield Road, Mountain View, CA 94043 T 650 584 5000 www.synopsys.com
For product related training, call 1-800-793-3448 or visit the Web at www.synopsys.com/services