

---

# Tutorial: NanoSim™ Integration with VCS-MX

---

---

## Overview

The purpose of this tutorial is to help you create a combined simulation environment for both a mixed-signal simulation using NanoSim and VCS, and a mixed-language simulation using Scirocco and VCS. This flow is referred to as *NanoSim Integration with VCS-MX*.

---

## Tutorial

The files described in [Table 1](#) are used in this tutorial:

**Table 1** NanoSim Integration with VCS-MX Files

File name	Description
<i>chargepump.spi</i>	SPICE <i>chargepump</i> subcircuit netlist
<i>chargepump_com.v</i>	Verilog behavioral description file for a <i>chargepump_com</i> module
<i>chargepump_seg.v</i>	Verilog behavioral description file for a <i>chargepump_seg</i> module
<i>bsim3.mod</i>	BSIM model card
<i>comlogic.v</i>	Verilog behavioral description file for a <i>comlogic</i> module
<i>command</i>	Command file for the Scirocco simulation
<i>config</i>	NanoSim configuration file
<i>counter.v</i>	Verilog behavioral description file for a module, counter
<i>dummy.v</i>	<code>`timescale</code> compiler directive for the VCS simulation
<i>file_list.vc</i>	All Verilog description file names used in VCS compilation
<i>run</i>	Run script
<i>seglogic.v</i>	Verilog behavioral description file for a <i>seglogic</i> module

**Table 1** NanoSim Integration with VCS-MX Files (Continued)

File name	Description
<i>signal.rc</i>	turboWave setup file
<i>synopsys_sim.setup</i>	Scirocco setup file
<i>testbench.vhd</i>	VHDL test bench
<i>vcsAD.init</i>	Partition file for the NanoSim-VCS integration
<i>WORK</i>	Directory for the Scirocco physical library

## Procedure

There are two main objectives to create the simulation environment for the *NanoSim Integration with VCS-MX* flow:

- [Setting up Scirocco, VCS, and NanoSim environment variables and paths](#)
- [Running a NanoSim Integration with VCS-MX simulation](#)

---

### Setting up Scirocco, VCS, and NanoSim environment variables and paths

To begin this tutorial, set all necessary environment variables and paths:

#### Steps

##### 1 For Scirocco:

```
setenv SYNOPSYS_SIM
Scirocco_installation_directory

source $SYNOPSYS_SIM/admin/setup/environ.csh
```

##### 2 For VCS:

```
setenv VCS_HOME VCS_installation_directory

set path = ($VCS_HOME/bin $path)
```

```
setenv SCANVLOG_PATH $VCS_HOME/'$VCS_HOME/
bin/vcs -platform'
```

**3** For NanoSim:

```
source NanoSim_installation_directory
CSHRC_platform
```

**4** For Licensing:

```
setenv SNPSLMD_LICENSE_FILE
location_of_license_file
```

**5** As an alternative, you can also create a setup script file:

*EXAMPLE:*

```
setenv SYNOPSISYS_SIM /usr/synopsys/scirocco
source $SYNOPSISYS_SIM/admin/setup/environ.csh
```

```
setenv VCS_HOME /usr/synopsys/vcs
set path = ($VCS_HOME/bin $path)
setenv SCANVLOG_PATH $VCS_HOME/'$VCS_HOME/
bin/vcs -platform'
```

```
source /usr/synopsys/nanosim/CSHRC_sparcOS5
```

```
setenv SNPSLMD_LICENSE_FILE
26585@synopsys:$SNPSLMD_LICENSE_FILE
```

---

## Running a NanoSim Integration with VCS-MX simulation

To begin the second part of this tutorial, following these steps:

### Steps

- 6** Open and read the *run* file (NanoSim Integration with VCS-MX *run* script).

You should see the following (lines 1, 2 and 3, respectively), as shown in [Figure 1](#).

```
vlogan -f file_list.vc
vhdlan -event testbench.vhd
scs WORK.CFG_testbench -verilogcomp "+ad -PP"
scsim -include command
```

**Figure 1** run file script sample

See [Table 2](#) for an explanation of each line of code.

**Table 2** run file description

<b>Line</b>	<b>Description</b>
<p><b>Line 1:</b></p> <pre>vlogan -f file_list.vc</pre>	<p>Analyzes the Verilog source files listed in the <i>file_list.vc</i> file.</p> <p><code>-f</code> has a file name that specifies Verilog description file names to be compiled.</p> <p><i>file_list.vc</i> includes six Verilog description files:</p> <ul style="list-style-type: none"> <li><i>dummy.v</i></li> <li><i>counter.v</i></li> <li><i>comlogic.v</i></li> <li><i>seglogic.v</i></li> <li><i>chargepump_com.v</i></li> <li><i>chargepump_seg.v</i></li> </ul>
<p><b>Line 2:</b></p> <pre>vhdlan -event testbench.vhd</pre>	<p>Analyzes specified <i>testbench.vhd</i> VHDL file. Code is also generated for event-mode simulation using the <code>-event</code> option.</p>
<p><b>Line 3:</b></p> <pre>scs WORK.CFG_testbench -verilogcomp "+ad -PP"</pre>	<p>Builds the simulation executable for this tutorial. The top-level design unit for elaboration is <i>CFG_testbench</i>, a configuration of the top-level test bench.</p> <p><i>WORK</i> is a logical library name. This library is defined in the Scirocco <i>synopsys_sim.setup</i> setup file.</p> <p><code>-verilogcomp</code> takes Verilog description files that need compilation and the VCS compile time options.</p> <p><code>+ad</code> VCS compile time option enables the NanoSim integration</p> <p><code>-PP</code> enables dumping of signals on ports of the Verilog modules into the VPD files</p>

**Table 2** run file description (Continued)

Line	Description
<p>Line 4:</p> <pre>scsim -include command</pre>	<p><b>scsim is the simulation executable:</b></p> <p><b>-include contains the Scirocco simulation control file.</b></p> <p><b>command includes the following simulation control commands (lines 1-3):</b></p> <pre>dump -deep /TESTBENCH -o     lcd.vpd run quit</pre> <p><b>Line 1 dump shows that all signals under /TESTBENCH are dumped to the lcd.vpd file. The signals simulated in NanoSim are excluded.</b></p> <p><b>Line 2 run shows that the simulator runs until either an assertion stop or failure is detected.</b></p> <p><b>Line 3 quit shows quitting of the simulation.</b></p>

- 7** Open and read the Scirocco *synopsys\_sim.setup* setup file. You should see a file as shown in [Figure 2](#).

```

WORK > default
default : ./WORK
timebase=ps

```

**Figure 2** Scirocco synopsys\_sim.setup file sample

For a description of each line of code in [Figure 2](#), see [Table 3](#).

**Table 3** Scirocco synopsys\_sim.setup file description

Line	Description
<b>Line 1:</b> WORK > default	Defines the mapping of the WORK logical library to default (an intermediate name)
<b>Line 2:</b> default : ./WORK	Defines the mapping of the default intermediate name to the ./WORK physical name (a UNIX path name.)
<b>Line 3:</b> timebase=ps	Defines the basic unit of time used in Scirocco. The default is nanoseconds (ns).  To synchronize all three simulators, 1ps is chosen for NanoSim and VCS. NanoSim uses the <b>set_sim_tres 1p</b> NanoSim config command, specified in the <i>config</i> file. VCS uses the <code>`timescale 1ns / 1ps compiler</code> directive, specified in the <i>dummy.v</i> Verilog description file.

- 8** View the VHDL/Verilog description files and familiarize yourself with the circuit.

The tutorial design involves an LCD driver. The LCD driver contains two parts:

- ◆ Behavioral digital control logic circuits that generate digital stimuli
- ◆ Transistor-level charge pump circuits that use digital stimuli and generate voltage outputs for an LCD

The design essentially comprises the following Verilog and VHDL description files, as shown in [Table 4](#).

**Table 4** Verilog and VHDL description files

Verilog Description Files	VHDL Description Files
<i>comlogic.v</i>	<i>testbench.vhd</i>
<i>seglogic.v</i>	
<i>chargepump_com.v</i>	
<i>chargepump_seg.v</i>	
<i>counter.v</i>	

The entity testbench is the top-level that instantiates the following modules:

- ◆ *counter* (I0)
- ◆ *seglogic* (I1)
- ◆ *comlogic* (I2)
- ◆ *chargepump\_com* (I3)
- ◆ *chargepump\_seg* (I4)

The *chargepump\_com* and *chargepump\_seg* modules are the Verilog wrappers (containing only module names,

port lists, and port declarations) that wrap around the transistor-level circuits that are simulated in NanoSim.

The transistor-level circuits are in the *chargepump.spi* SPICE netlist file. This file has transistor-level circuits for *chargepump\_com* and *chargepump\_seg*.

VCS is given the information about handing-off *chargepump\_com* and *chargepump\_seg* to NanoSim through the *vcsAD.init* file.

- 9 Open and read the *vcsAD.init* partition file for NanoSim integration. This is the required file for the +ad option.

You should see the following, as shown in [Figure 3](#).

```
partition -cell chargepump_com chargepump_seg;  
choose nanosim -n chargepump.spi -c config -o lcd;  
set bus_format <%d>;
```

**Figure 3** vcsAD.init partition file sample

For a description of each line of code from [Figure 3](#), see [Table 5](#).

**Table 5** vcsAD.init partition file description

Line	Description
<p><b>Line 1:</b></p> <pre>partition -cell chargepump_com chargepump_seg;</pre>	<p>Line 1 displays the <code>partition</code> command. The <code>-cell</code> option takes names of subcircuits to be simulated in NanoSim. These names must exactly match the names used for Verilog modules.</p> <p>View the <code>chargepump.spi</code> SPICE netlist file, and the <code>chargepump_com.v</code> and <code>chargepump_seg.v</code> Verilog description files.</p> <p>Verify <code>chargepump_com</code> and <code>chargepump_seg</code> are the subcircuit names in the <code>chargepump.spi</code> SPICE netlist file.</p> <p>Verify <code>chargepump_com</code> and <code>chargepump_seg</code> are the module names in the <code>chargepump_com.v</code> and <code>chargepump_seg.v</code> Verilog description files.</p> <p>Check that the node names defined with subcircuit names are used as port names in the <code>chargepump_com</code> and <code>chargepump_seg</code> modules. They must correspond (except for their bus format).</p>
<p><b>Line 2:</b></p> <pre>choose nanosim -n chargepump.spi -c config -o lcd;</pre>	<p>Line 2 displays the <code>choose</code> command. This command only takes <code>nanosim</code> and its command line options in the <i>NanoSim Integration with VCS-MX</i> flow. PowerMill and TimeMill are not supported.</p> <p>The <code>-n</code>, <code>-c</code> and <code>-o</code> options are NanoSim command-line options. Here it takes the <code>chargepump.spi</code> SPICE netlist file, the <code>config</code> NanoSim configuration command file, and the NanoSim output file's prefix name, respectively.</p>

**Table 5** vcsAD.init partition file description (Continued)

Line	Description
<p>Line 3:</p> <pre>set bus_format &lt;%d&gt;;</pre>	<p>Line 3 displays the set bus_format command. This command instructs VCS for the bus format used in the chargepump.spi SPICE netlist file.</p> <p>Check the chargepump.spi SPICE netlist file. You should see the following:</p> <pre>.subckt chargepump_com + com_inv&lt;3&gt; com_inv&lt;2&gt; com_inv&lt;1&gt; com_inv&lt;0&gt; + com_rsh&lt;3&gt; com_rsh&lt;2&gt; com_rsh&lt;1&gt; com_rsh&lt;0&gt; + clk</pre> <p>In this case, com_inv&lt;3&gt; , com_inv&lt;2&gt;, com_inv&lt;1&gt;, and com_inv&lt;0&gt; nodes correspond to the com_inv[3:0] port defined in the chargepump_com module in the chargepump_com.v Verilog description file.</p>

**10** Open and read the *config* NanoSim configuration command file.

You should see the following, as shown in [Figure 4](#).

```
use_sim_case
set_sim_tres lps
Print_node_v TESTBENCH.I4.com_out_* TESTBENCH.I3.seg_out_*
set_print_uod out=all
```

**Figure 4** NanoSim config command file sample

For a description of each line of code from [Figure 4](#), see [Table 6](#).

**Table 6** NanoSim config command file description

Line	Description
<b>Line 1:</b>  <code>use_sim_case</code>	Keeps case-sensitivity in the transistor-level netlist
<b>Line 2:</b>  <code>set_sim_tres 1ps</code>	Sets 1ps as NanoSim simulation time resolution value. This value is used to synchronize Scirocco, VCS and NanoSim.
<b>Line 3:</b>  <code>Print_node_v</code> <code>TESTBENCH.I4.com_out_*</code> <code>TESTBENCH.I3.seg_out_*</code>	Prints output node voltage signals in the <i>chargepump_com</i> and <i>chargepump_seg</i> subcircuits
<b>Line 4:</b>  <code>set_print_uod out=all</code>	Generates the <i>lcd_uod.out</i> unified output display (UOD) file. This UOD file contains data from both the <i>lcd.vpd</i> and <i>lcd.out</i> files. The <code>out=all</code> option maintains the <i>lcd.vpd</i> , <i>lcd.out</i> , and <i>lcd_uod.out</i> files.

**11** Execute the *run* script.

**12** Invoke the turboWave waveform viewer by entering:

```
turboWave -ssr signal.rc
```

You see input digital stimuli and output voltage signals in both the *chargepump\_com* and *chargepump\_seg* subcircuits, as shown in [Figure 5](#).

**13** Select one of two options (command-line or GUI) to view hierarchical names in the waveform viewer:

- ◆ Press `h` on your keyboard
- ◆ Choose **View > Hierarchical Name** from the turboWave menu bar, as shown in [Figure 6](#).

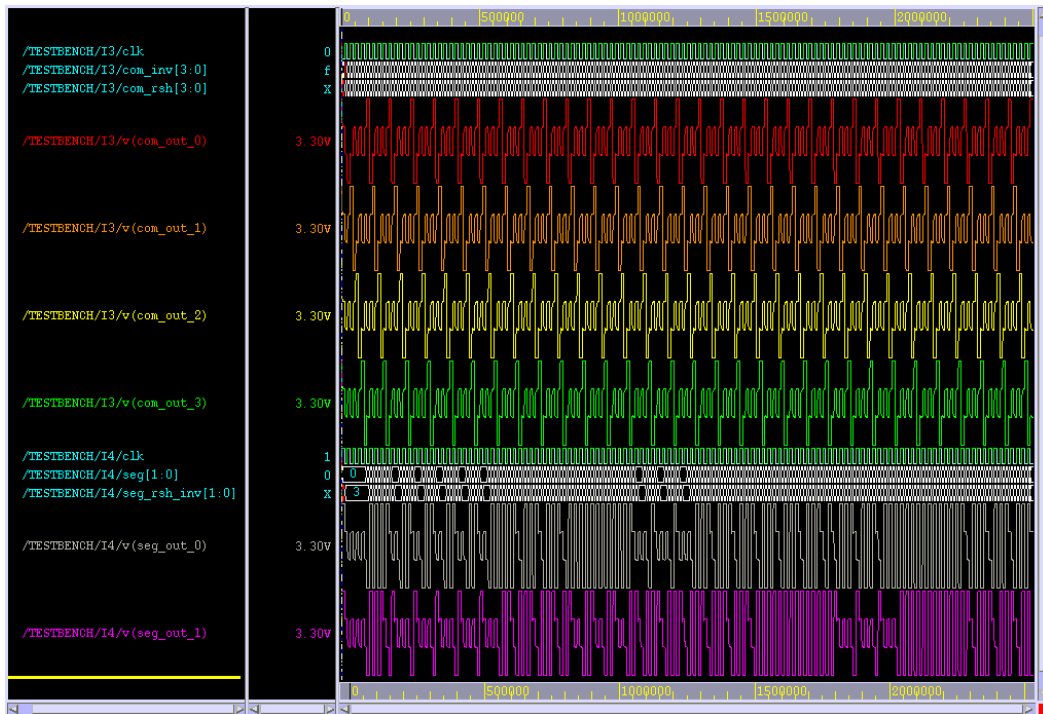


Figure 5 turboWave display

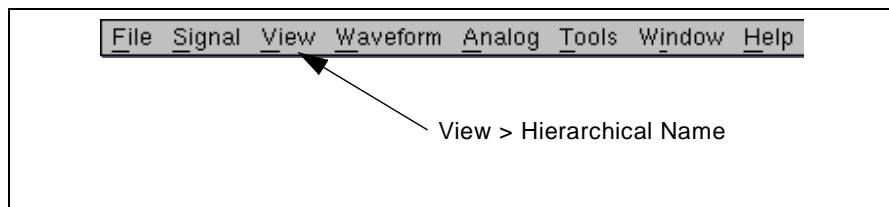


Figure 6 turboWave menu bar

**You have now completed the tutorial. Thank you!**