Tutorial: NanoSim-VCS-MX (NS-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 VCS-MX. This flow is referred to as *NanoSim-VCS-MX*.

Tutorial

The files described in Table 1 are used in this tutorial:

Table 1 NanoSim-VCS-MX Files

File name	Description
chargepump.spi	SPICE *chargepump* subcircuit netlist
chargepump_com.v	Verilog behavioral description file for a *chargepump_com* module
chargepump_seg.v	Verilog behavioral description file for a *chargepump_seg* module
bsim3.mod	BSIM model card
comlogic.v	Verilog behavioral description file for a *comlogic* module
command	Command file for the VHDL-top VCS-MX simulation
config	NanoSim configuration file
counter.v	Verilog behavioral description file for a module, counter
file_list.vc	All Verilog description file names used in VCS compilation
run	Run script
seglogic.v	Verilog behavioral description file for a *seglogic* module
signal.rc	nWave setup file
synopsys_sim.setup	Setup file for mixed-HDL design

Table 1 NanoSim-VCS-MX Files (Continued)

File name	Description
testbench.vhd	VHDL test bench
vcsAD.init	Partition file for the NanoSim-VCS integration
WORK	Directory for the VCS-MX physical library

Procedure

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

- Setting up the NanoSim and VCS-MX Environment Variables and Paths
- Running a NanoSim-VCS-MX simulation

Setting up the NanoSim and VCS-MX Environment Variables and Paths

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

Steps

1. For VCS-MX:

```
setenv VCS_HOME VCS_installation_directory

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

2. For NanoSim:

```
source NanoSim_installation_directory
   CSHRC_platform
```

3. For Licensing:

```
setenv SNPSLMD_LICENSE_FILE
   location_of_license_file
```

4 You can also create a setup script file, as an alternative:

EXAMPLE:

```
setenv VCS_HOME /usr/synopsys/vcs

set path = ($VCS_HOME/bin $path)

source /usr/synopsys/nanosim/CSHRC_sparcOS5

setenv SNPSLMD_LICENSE_FILE
   26585@synopsys:$SNPSLMD_LICENSE_FILE
```

Running a NanoSim-VCS-MX simulation

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

Steps

5 Open and read the *run* file (NanoSim-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_testbanch -mhdl -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

Line	Description
Line 1: `vlogan -f file_list.vc`	Analyzes the Verilog source files listed in the *file_list.vc* file. `-f` has a file name that specifies Verilog description file names to be compiled. `file_list.vc` includes six Verilog description files: *counter.v* *comlogic.v* *seglogic.v* *chargepump_com.v* *chargepump_seg.v*
Line 2: `vhdlan -event testbench.vhd`	Analyzes specified `testbench.vhd` VHDL file. Code is also generated for event-mode simulation using the `-event` option.

Table 2 run file description (Continued)

Line	Description
Line 3: `scs WORK.CFG_testbench -mhdl -verilogcomp “+ad -PP”`	Builds the simulation executable for this tutorial. The top-level design unit for elaboration is `CFG_testbench`, a configuration of the top-level test bench. `WORK` is a logical library name. This library is defined in the VCS-MX *synopsys_sim.setup* setup file. `-mhdl` is required for the Mixed-HDL design. `-verilogcomp` takes Verilog description files that need compilation and the VCS compile time options. `+ad` VCS compile time option enables the NanoSim integration `-PP` enables dumping of signals on ports of the Verilog modules into the VPD files

Table 2 run file description (Continued)

Line	Description

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

6 Open and read the VCS-MX *synopsys_sim.setup* setup file.

You should see a file as shown in Figure 2.

```
WORK > default
default : ./WORK
timebase=ns
time_resolution=10ps
```

Figure 2 VCS-MX synopsys_sim.setup file sample

For a description of each line of code in Figure 2, see Table 3.

Table 3 VCS-MX synopsys_sim.setup file description

Line	Description
Line 1: `WORK > default`	Defines the mapping of the `WORK` logical library to `default` (an intermediate name)
Line 2: `default : ./WORK`	Defines the mapping of the `default` intermediate name to the `./WORK` physical name (a UNIX path name.)

Line 3:
 	
`timebase=ns`
`time_resolution=10ps`
	`timebase` defines the basic unit of time used in VCS-MX. The default is nanoseconds (ns).
	
	`time_resolution` defines the VCS-MX simulation time resolution. To synchronize with NanoSim, 10ps is chosen because the NanoSim default simulation time resolution is 10ps.

7 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
comlogic.v	*testbench.vhd*
seglogic.v	
chargepump_com.v	
chargepump_seg.v	
counter.v	

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.

8 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
Line 1: `partition -cell chargepump_com chargepump_seg;`	Line 1 displays the `partition` command. The `-cell` option takes names of subcircuits to be simulated in NanoSim. These names must exactly match the names used for Verilog modules. View the `chargepump.spi` SPICE netlist file, and the `chargepump_com.v` and *`chargepump_seg.v`* Verilog description files. Verify `chargepump_com` and `chargepump_seg` are the subcircuit names in the `chargepump.spi` SPICE netlist file. Verify `chargepump_com` and `chargepump_seg` are the module names in the *chargepump_com.v* and *chargepump_seg.v* Verilog description files. Check that the node names defined with subcircuit names are used as port names in the `chargepump_com` and `chargepump_seg` modules. They must correspond (except for their bus format).
Line 2: `choose nanosim -n chargepump.spi -C config -o lcd;`	Line 2 displays the `choose` command. This command only takes `nanosim` and its command line options in the *NanoSim-VCS-MX* flow. The `-n`, `-C` and `-o` options are NanoSim command-line options. Here it takes the `chargepump.spi` SPICE netlist file, the `config` NanoSim configuration command file, and the NanoSim output file's prefix name, respectively.

Table 5 vcsAD.init partition file description (Continued)

Line	Description

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

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

You should see the following, as shown in Figure 4.

```
use_sim_case
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
Line 1: `use_sim_case`	Keeps case-sensitivity in the transistor-level netlist
Line 2: `Print_node_v TESTBENCH.I4.com_out_* TESTBENCH.I3.seg_out_*`	Prints output node voltage signals in the *chargepump_com* and *chargepump_seg* subcircuits
Line 3: `set_print_uod out=all`	Generates the *lcd_uod.out* unified output display (UOD) file. This UOD file contains data from both the *lcd.vpd* and *lcd.out* files. The out=all option maintains the *lcd.vpd*, *lcd.out*, and *lcd_uod.out* files.

10 Execute the *run* script.

11 Invoke the nWave waveform viewer by entering:

```
nWave -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.

12 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 nWave menu bar, as shown in Figure 6.

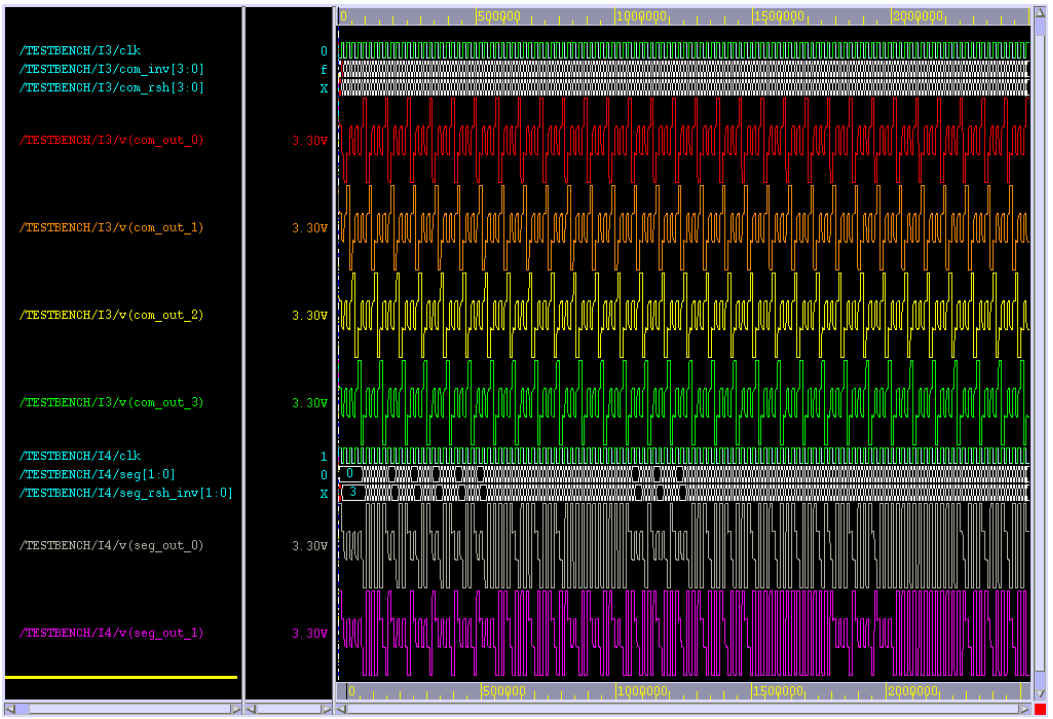

Figure 5 nWave display

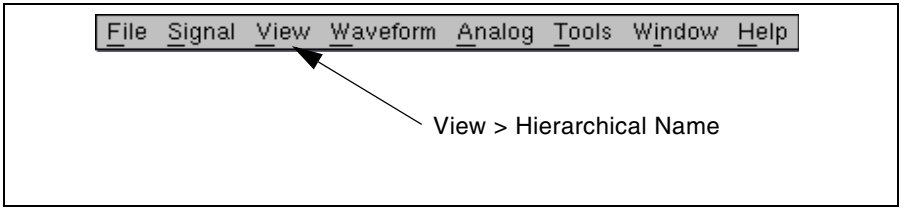

Figure 6 nWave menu bar

You have now completed the tutorial. Thank you!