

VCS®/VCSi™ Release Notes

G-2012.09

September 2012

Comments?

E-mail your comments about this manual to:

vcs_support@synopsys.com.

The Synopsys logo, featuring the word "SYNOPSYS" in a bold, purple, sans-serif font. A registered trademark symbol (®) is located at the top right of the letter "S".

SYNOPSYS®

Copyright Notice and Proprietary Information

Copyright © 2012 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

SYNOPSYS, 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, AEON, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, CHIPit, CoMET, Confirma, CODE V, Design Compiler, DesignWare, EMBED-IT!, Formality, Galaxy Custom Designer, Global Synthesis, HAPS, HapsTrak, HDL Analyst, HSIM, HSPICE, Identify, Leda, LightTools, MAST, METeor, ModelTools, NanoSim, NOVeA, OpenVera, ORA, PathMill, Physical Compiler, PrimeTime, SCOPE, Simply Better Results, SiVL, SNUG, SolvNet, Sonic Focus, STAR Memory System, Syndicated, Synplicity, the Synplicity logo, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, UMRBus, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, ARC, ASAP, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Cosmos, CosmosLE, CosmosScope, CRITIC, CustomExplorer, CustomSim, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, DFTMAX, Direct Silicon Access, Discovery, Eclipse, Encore, EPIC, Galaxy, HANEX, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance ASIC Prototyping System, HSIM^{plus}, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Intelli, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Macro-PLUS, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, ORAengineering, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, RippledMixer, Saturn, Scirocco, Scirocco-i, SiWare, Star-RCXT, Star-SimXT, StarRC, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCSi, VHDL Compiler, VMC, and Worksheet Buffer 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.

Saber is a registered trademark of SabreMark Limited Partnership and is used under license.

All other product or company names may be trademarks of their respective owners.

Contents

1. VCS®/VCSi™ Release Notes

What's New in This Release	3
Diagnostic Features	3
SystemVerilog Features	5
Compilation Features	10
Assertion Features	13
DVE/Debug Features	15
Coverage Features	20
UVM and VMM Features	24
MVSIM Native Low Power Features	26
SystemC Features	30
LCA Features in This Release	33
Signature-based Control for Deferred Assertions and RT Checks	36
Analog and Mixed-Signal Feature: Multi-Driver Support for Wreal	37
UCLI Enhancement in vpd2vcd Utility	37
DVE Viewing the Full Design Hierarchy in a Partially Dumped VPD	37
Tcl Fileevent Support for UCLI and DVE	38

Coverage Report-Time Exclusion	38
Coverage Analysis of Unreachable Verilog Code	38
Echo Features	39
Echo Procedural Sampling Enhancements	40
Hierarchical Cross Coverage	40
Support for Adaptive Exclusion in DVE GUI	41
MVSIM NLP LCA Features	41
Multicore DLP Autopartitioning	42
Multicore ALP FSDB Dumping.	42
The Unified Profiler	43
Constraint Profiling is Integrated into the Unified Profiler . . .	44
PLI/DPI/DirectC Enhancement in the Unified Profiler	45
Single Text Format Report From the Unified Profiler	47
Support for “with” Clause in Cover Groups	47
Profiler Hypertext Links to the Source Files.	47
Extern Task and Function Calls through Virtual Interfaces . .	52
Integrating SystemC with Innovator.	53
Profiling the SystemC Portion of a Design.	53
Partition Compile	54
SAIF Support for SystemVerilog Datatypes.	58
Generating SystemC Profiling Reports	58
SystemC Simulating Virtual Platform Models	59
SystemVerilog Assertions	59
Fast Compilation	61
VMM SystemC (VMM-SC).	61
VMM-AMS	62

Reducing Disk Space for Post-process only Debug	62
Reducing Compile Time for Post-process only Debug	62
Integrating DVE with Protocol Analyzer	62
Verification Planner Doc Annotator	63
Verification Planner XML Doc Plan Generator	63
Verification Planner HVP Interactive Editor	63
IEEE Verilog Std 1364-2005 Encryption	64
The syschk.sh and syschk.dat script	65
Production License Keys	65
Supported Platforms and Products	67
Supported Platforms	67
VCS SystemC Cosimulation Interface	72
Downloading and Installing the VG GNU Package	73
Support for Synopsys SmartModel/FlexModel/Mempro	84
Tested Technologies	85
Third-Party Tools	89
Documentation	90
Documentation in PDF Format	93
Limitations	94
Cap on Number of Command-line Characters	94
Limitations on VCD and VPD Files	95
Coverage Metrics Limitations	95
OVA Limitations	96
SystemC Cosimulation Interface Limitation	97
Limitations on Variable-Sized Arrays	97

OpenOffice Not Supported by Verification Planner Doc Annotator	98
Unified Profiler Limitations	98
Multicore ALP and DLP Current Limitations	99
Known Issues	101
Known Issues in Verilog.	101
SystemC Cosimulation with UCLI Common C++ Library Error	103
Obsolete Switches/Options	104
Error — Obsolete Option Used	104
Warning — Unknown or Obsolete Option Used	105
Obsolete System Tasks	107
Obsolete <code>-assert_disable_file=<file_name></code> Option	108
PDF File Name Changes.	108
Bug Reporting Procedure	109
List of Issues Resolved in VCS and VCS MX	109

1

VCS®/VCSi™ Release Notes

Version G-2012.09

VCS is the Synopsys Verilog/SystemVerilog language verification platform. VCS supports IEEE P1800-2009. These release notes contain the latest information about VCS and VCSi, G-2012.09. These release notes are written for system administrators and end users of Synopsys tools. Please forward this document to the intended audience.

VCS conforms to the common Synopsys versioning system. The version is based on the EDA Consortium roadmap for operating systems foundation letter and the year and month of the release.

These release notes provide information about the following topics:

- [“What’s New in This Release” on page 3](#)
- [“LCA Features in This Release” on page 33](#)

- “The syschk.sh and syschk.dat script” on page 65
- “Production License Keys” on page 65
- “Supported Platforms and Products” on page 67
- “Documentation” on page 90
- “Limitations” on page 94
- “Known Issues” on page 101
- “Obsolete Switches/Options” on page 104
- “Obsolete System Tasks” on page 107
- “Bug Reporting Procedure” on page 109
- “List of Issues Resolved in VCS and VCS MX” on page 109

What's New in This Release

- “Diagnostic Features” on page 3
- “SystemVerilog Features” on page 5
- “Compilation Features” on page 10
- “Assertion Features” on page 13
- “DVE/Debug Features” on page 15
- “Coverage Features” on page 20
- “UVM and VMM Features” on page 24
- “MVSIM Native Low Power Features” on page 26
- “SystemC Features” on page 30

Diagnostic Features

- “Enabling Diagnostics” on page 4
- “Diagnostics for VPI PLI Application” on page 4
- “Using the vpdutil Utility to Generate Statistics” on page 5

Enabling Diagnostics

Use the `-diag` option to enable the libconfig/timescale diagnostic messages at compile-time and VPI diagnostic messages at runtime. The following table lists the supported diagnostic options.

Option	Use Model	Description
libconfig	<code>vcs -diag libconfig</code>	Enables the library binding diagnostics
timescale	<code>vcs -diag timescale</code>	Enables timescale diagnostics
vpi	<code>simv -diag vpi</code>	Enables VPI diagnostics

Diagnostics for VPI PLI Application

By default, as per LRM, VPI remain silent when an error occurs. In order to report any errors, it is the application's responsibility to check for error status and then display errors accordingly. If sufficient error detection mechanisms are not in place, then the application C code must be modified and recompiled. In addition, user may need to recompile the HDL code if required.

However, you can use the following new runtime diagnostic option to make the PLI application report errors without code modification:

```
-diag vpi
```

Furthermore, reporting provides you the information related to the HDL code context (where applicable), to help fix problems with a faster turnaround time.

Using the vpdutil Utility to Generate Statistics

The vpdutil utility generates statistics about the data in the vpd file. The utility takes a single vpd file as input. You can specify options to this utility to query at design, module, instance, and node levels.

This utility supports time ranges and input lists for query on more than one object. Output will be in ascii to stdout with option to redirect to an output file.

SystemVerilog Features

- [“The foreach Iterative Constraint for Packed Arrays” on page 5](#)
- [“Soft Constraints are Enabled by Default” on page 7](#)
- [“Exporting a SystemVerilog Package” on page 7](#)
- [“Randomized Objects in a Structure” on page 8](#)
- [“Using a Package in a SystemVerilog Module, Program, and Interface Header” on page 9](#)

The foreach Iterative Constraint for Packed Arrays

VCS has implemented `foreach` loop variables for entirely packed dimensions of an array in the constraint context in the G-2012.09 release.

In previous releases up to 2011.12-2, a `foreach` loop for the dimensions of a multi-dimensional array in the constraint context required that at least one of the dimensions be unpacked. That restriction is removed. A multi-dimensional packed array in the constraint context is now fully supported.


The following code example illustrates this implementation.

Example 1-1 The foreach Iterative Constraint for Packed Arrays

```
program prog;
class my_class;
    rand reg [2][2][2][2] arr;

    constraint constr {
        foreach (arr[i,j,k,l]) {
            (i==0) -> arr [i][j][k][l] == 1;
            (i==1) -> arr [i][j][k][l] == 0;
        }
    }
endclass

endprogram
```



all dimensions packed

In previous releases at least one of the dimensions of an MDA array needed to be unpacked.

This code example results in the following error message in previous releases:

```
Error-[NYI-UFAIFE] NYI constraint: packed dimensions
doc_ex.sv,9
prog, "this.arr"
  arr has only packed dimensions and no unpacked dimensions.
  Foreach over packed dimensions is supported if the object
  has at least one
  unpacked dimension.

1 error
```

Starting with release F-2011.12-3 and G-2012.09, entirely packed arrays in the constraint context are not an error condition and do not result in this error message.

Soft Constraints are Enabled by Default

Soft constraints are enabled by default, the `-x1rm soft` compile-time option and keyword argument are now obsolete.

Soft constraints prevent randomizations failures when there is a conflict between constraints. VCS disables the soft constraints when they are in conflict with other constraints.

You specify a soft constraint with the keyword `soft`.

Example 1-2 A soft constraint

```
program prog;

class C10;
    rand integer intr0;
    constraint constr0 {
        soft intr0 > 123;
    }
endclass

endprogram
```

Exporting a SystemVerilog Package

VCS has a new implementation of how it exports SystemVerilog packages. This new implementation is less optimistic and is more rigidly compliant with the SystemVerilog IEEE Std 1800-2009 standard. You enable this new implementation with the `-sv_package_export` compile-time option or `vlogan` option.

In the new implementation, declarations imported into a package are not visible by way of subsequent imports of that package.

Package export declarations allow a package to specify those imported declarations to be made visible in subsequent imports.

Randomized Objects in a Structure

VCS has implemented randomized objects in a structure. The following code example illustrates this implementation.

Example 1-3 Randomized Object in a Structure

```
program test;


    class packet;
        randc int addr  = 1;
        int crc;
        rand byte data [] = {1,2,3,4};
    endclass
    class packet_test;
        typedef struct {
            rand packet p1;
        } header;
        header hd;

    function new();
        this.hd.p1 = new;
    endfunction

    endclass

    packet_test pt = new;

    initial begin
        pt.randomize(hd);
    end
endprogram
```



**randomized object
in a structure**

In previous releases, declaring this class in a structure with the `rand` type-modifier keyword resulted in the following error message:

```
Error-[SV-NYI-CRUDST] Rand class object under structure
code_ex_rand_struct.sv, 10
"p1"
  Rand class objects which defined under structure is not
yet supported.

1 error
```

In G-2012.09, the previous code example will compile and run without any errors as `rand` class object inside struct is now supported.

Using a Package in a SystemVerilog Module, Program, and Interface Header

Importing from a package to a module, program, or interface by including the package in the module, program, or interface header is now implemented.

This technique of importing from a package is described in the SystemVerilog LRM IEEE Std 1800-2009 in the section named “26.4 Using packages in module headers” in clause “26 Packages.”

The primary purpose of this syntax and usage is to enable you to imported names in the parameter list or port list, without importing the package into the enclosing scope (`$unit`).

Compilation Features

- “Lint Warning Message for Missing ‘endcelldefine”
- “Error/Warning Message Control”
- “Verilog 2001 is Now Default”
- “IEEE Verilog Std 1364-2005 Encryption” on page 64
- “Lifting Partition Compile Limitations” on page 12
- “Partition Compile Relocatable simv” on page 13

Lint Warning Message for Missing ‘endcelldefine

You can tell VCS to display a lint warning message if your Verilog or SystemVerilog code contains a ``celldefine` compiler directive without a corresponding ``endcelldefine` compiler directive and vice versa.

You enable this warning message with the `+lint=CDUB` VCS compile-time option . The `CDUB` argument stands for “compiler directives unbalanced.” The following is an example of the warning message:

```
Lint-[CDUB] Compiler directive unbalanced
expl.v, 1
  Unbalanced compiler directive is detected : `celldefine
  has no matching `endcelldefine.
  Please make sure that all directives are balanced.
```


Error/Warning Message Control

This release includes the new `-error` and `-suppress` options, and revises the `+lint` and `+warn` options, to control error and warning messages. Their syntax are as follows:

```
-error=[no]message_ID[:max_number],...|none|all
```

```
-error=all,noWarn_ID|noLint_ID
```

```
+warn=[no]message_ID[:max_number],...|none|all
```

```
+lint=[no]message_ID[:max_number],...|none|all
```

```
-suppress[=message_ID,...]
```

Where:

message_ID

Is the character string in the square brackets in a message

maximum_number_of_occurrences

Is a non-negative integer specifying the maximum number of the specified messages VCS can display during compilation.

This argument is optional with the `-suppress` option.

The `+lint` and `+warn` compile-time options are plusargs in this release. You can include a number of messages. They are for disabling lint and warning messages.

You can enter the `-suppress` option with or without arguments. If you do not enter an argument, this option disables the display of all warning and error messages.

Important:

These options cannot suppress the display of the following:

- syntax errors
- fatal error messages, those from error conditions that immediately halt compilation

The `+vcs+error` compile-time option is replaced by the `-error` option.

The `-no_error` option is now obsolete and replaced by the `-error=noID`.

Verilog 2001 is Now Default

The `+v2k` compile-time option is no longer required in VCS to compile Verilog 2001 code.

Verilog 2001 or V2K source code conforms to the Verilog IEEE Std 1364-2001 instead of the Verilog IEEE Std 1364-1995.

If your Verilog code contains a V2K keyword as an identifier, you can tell VCS not to recognize V2K keywords with the `-v95` compile time option, for example:

```
module cell (...,...);
```

The module identifier `cell` is a keyword in Verilog 2001, so to use it as an identifier include the `-v95` compile-time option.

Lifting Partition Compile Limitations

In previous releases Partition Compile could not be used with the following compile-time options:

`-hsopt=gates +rad -cm_noconst`

These restrictions are lifted in this release.

Partition Compile Relocatable simv

The partition compile relocatable `simv` feature enables you to copy the `simv` executable, and Partition Compile related `.so` files, to another location and run the executable there without depending the contents of the source directory.

To use this feature, Synopsys provides script named `simcopy` to copy the files to new location.

With this feature you can, for example, compile a `simv` executable in the `/tmp` or `/localdisk` directories of a machine and then relocate it back to NFS.

This feature also improves the runtime performance of regressions by reducing the NFS load factor.

The script for copying the executable, and partition compile related `.so` files, only works in the partition compile flow.

Assertion Features

- [“Parameter Checks for SVA CG Checker Library” on page 14](#)
- [“Disabling Assertions at Compile Time” on page 14](#)
- [“System Tasks for SVA Control” on page 14](#)
- [“Runtime Reporting for Disabled Assertions” on page 15](#)
- [“Enabling IEEE Std. 1800-2009 Compliant Features” on page 15](#)

Parameter Checks for SVA CG Checker Library

The SystemVerilog assertions (SVA) cover group (CG) library now enables you to parameterize the checkers. To avoid errors due to invalid parameter values at runtime, the checker library is now enhanced to do a compile time check on the parameters.

The parameter checks are implemented using the IEEE1800-2009 elaboration system tasks `$info/$warning/$error/$fatal` to ensure that the identified parameters will have only the valid values passed during the instantiation.

All the invalid values are reported in a single compilation so that it can be corrected in a single iteration.

Disabling Assertions at Compile Time

VCS and VCSMX provides enhanced assertion control capabilities whereby you can disable only the assertions (assert directives) through a compile time switch. This disables only the assert and assume directives without affecting the cover directives.

The new capability complements the existing control options which allows you to disable only cover directives or all the assertions such as `assert/assume/cover`.

System Tasks for SVA Control

VCS now supports the following three system tasks to control the evaluation of assertion statements:

- `$assertoff`
- `$assertkill`

- `$asserton`

These system tasks are now enhanced to provide file name and line number from where these system tasks are called which would otherwise be difficult to track in the absence of this information.

Note:

The runtime switch `-assert old_ctrl_msg` reverts the messaging to the old style for backward compatibility.

Runtime Reporting for Disabled Assertions

VCS provides the enhanced feature to generate a report for the assertions that are disabled during runtime. A report is generated which lists all the disabled assertions using

- System tasks `$asserton/off/kill`
- `-assert hier` at compile/runtime

Enabling IEEE Std. 1800-2009 Compliant Features

You must use the `-assert svaext` compile-time option to enable the new IEEE Std. 1800-2009 compliant SVA features.

DVE/Debug Features

- [“Dynamic Object Browser” on page 16](#)
- [“Creating Nested Signal Groups” on page 16](#)

- “Driver Tracing Support for Virtual Interfaces and Clocking Blocks” on page 17
- “Thread Debugging Enhancements” on page 17
- “Keeping the UCLI/DVE Prompt Active After a Runtime Error” on page 18
- “Constraint Debug Enhancements” on page 18
- “UVM Testbench Design Debugging” on page 18
- “DVE Smartlog” on page 19
- “Low Power Debug Enhancements”

Dynamic Object Browser

DVE now allows you to view and browse all existing class objects and member values using the Object Browser feature. This feature consists of three major pieces of functionality:

- Object Hierarchy Browser which displays current dynamic objects and its values.
- Enhancements to the Class Browser to support viewing of object instances.
- Enhancements to the Local Pane filter to support object search.

Creating Nested Signal Groups

DVE allows you to create nested signal groups in the Wave View and List View. That is, DVE allows a regular signal group to be part of another signal group. You can create nested signal groups by dragging and dropping one signal group to another, or by using Signal Group Manager dialog.

Driver Tracing Support for Virtual Interfaces and Clocking Blocks

DVE now extends the tracing driver functionality by allowing you to view all assignments using virtual interface or clocking block, which caused the value change. DVE displays these assignments as a list of drivers in the Driver Pane.

Thread Debugging Enhancements

DVE extends thread debug functionality by allowing you to:

- View all the threads in your design and the status of the selected thread in the Stack Pane.
- Filter the named and unnamed scopes which are not active call stacks.
- Provide better names for unnamed scopes (of type initial, always or fork) from active call stacks.
- Search for a thread in the Stack Pane
- Set thread-specific breakpoints in the Stack Pane
- Double-click a thread ID in Console Pane to view it in the Stack Pane
- Set different background color for stack frame from user code and UVM (VMM, OVM) library code.

Keeping the UCLI/DVE Prompt Active After a Runtime Error

VCS now allows you to debug an unexpected error/crash of simv by keeping simv alive, and providing a UCLI/DVE command prompt after the failure.

Constraint Debug Enhancements

DVE extends constraint debug functionality by allowing you to:

- Change the radix type of a variable value and values of a constraint expression in the Solver Pane and Relation Pane of the Constraints Dialog.
- Drag-and-drop a variable or constraint block item from the Solver Pane or Relation Pane into the Search field of the Constraints Dialog.
- View object ID information of a class object in the Value column of the Solver Pane.
- Extract testcase of a Partition and Randomize Call item.
- Control rand_mode/constraint_mode and randomization from UCLI/DVE.

UVM Testbench Design Debugging

DVE now supports debugging of UVM testbench designs by allowing you to do the following:

- View all available configurations in your design.
- View the set/get history of a configuration item.

- View all the predefined phases of common and UVM domain.
- Set breakpoints on the important phases or on the `uvm_component`'s phase methods.
- Viewing runtime arguments using the Simulation Arguments dialog.
- Filter UVM object items in the Watch Pane.

DVE Smartlog

Smartlog provides log analysis (diagnostic information) for each line in the log file, in DVE. It takes the compile log and simulation log created by VCS and summarizes the data into easy to understand reports. Smartlog provides the diagnostic information in a separate log file known as smartlog file. Following are the main features of Smartlog:

- Provide log analysis for each line in the log file.
- Hyperlink the log occurrences to the Source View.
- Highlight the words “Error”, “Warning”, and so on, in different colors.
- Display the selected message within a blue rectangle.

Low Power Debug Enhancements

DVE extends low power debug functionality by allowing you to:

Waveform Shading

- View isolation shading
- View isolation power-off shading

- View input toggle shading
- View `set_related_supply_net` (SRSN) shading

Schematic View

- View special schematic symbol for isolation cells

Coverage Features

The following coverage features are available in this release:

- [“Coverage Mapping and Reporting Enhancements” on page 20](#)
- [“Cross of Crosses” on page 21](#)
- [“Cross-of-a-Cross Exclusion Support” on page 23](#)
- [“Filtering Instances with no Coverable Objects” on page 23](#)
- [“Support for Propagating Constants Through Instance Arrays” on page 23](#)
- [“Support for Reporting HVP Annotated Results in XML Format” on page 24](#)
- [“Toggle Coverage MDA Exclusion” on page 24](#)

For more information on the following sections, see *VCS / VCS MX Online Documentation*.

Coverage Mapping and Reporting Enhancements

The following enhancements have been made to URG to support the following features of `assertCovReport`.

- Mapping for assertion coverage using the `-map` and `-mapfile` URG options
- Assertion coverage reporting

For example, you can now perform mapping by which assertions from different hierarchies can be merged together and reported cumulatively.

Cross of Crosses

IEEE 1800 SystemVerilog allows cross specification among two or more coverpoints. However, it does not allow the cross constituents to be a cross itself. Since advanced coverage models require this ability, the cross of crosses extension enables the definition of crosses that include other crosses as elements.

[Example 1-4](#) illustrates a cross of crosses.

Example 1-4

```
//Assuming cp0, cp1, cp2, cp3 are of type bit[3:0]
covergroup cg_1;
  coverpoint cp0;
  coverpoint cp1;
  coverpoint cp2;
  coverpoint cp3;
  cr_0: cross cp0, cp1 {
    bins low_b = binsof(cp0) intersect {[0:7]};
  }
  cr_1: cross cp2, cp3 {
    bins high_b = binsof(cp2) intersect {[8:15]};
  }
  cr_0_X_cr_1: cross cr_0, cr_1 {
    bins one_b = binsof(cr_0.low_b) && binsof(cr_1.high_b);
  }
  cp0_X_cr_1: cross cp0, cr_1 {
    bins two_b = binsof(cr_1.high_b) && binsof(cp0)
    intersect {[0:7]};
  }
}
```

```

    }
endgroup: cg_1;

```

Here, apart from the fact that `cr_0_X_cr_1` is the cross of two crosses (`cr_0` and `cr_1`), it is like any other cross. It can define attributes and bins like regular crosses. However, the `binsof` syntax is restricted to disallow the use of `intersect` with cross bins. It is also possible to have a mix of cross and coverpoint elements, for example, cross `cp0_X_cr_1` crosses `cr_1` with coverpoint `cp0`. Note that the `binsof` construct can use `intersect` when referring to a coverpoint bin.

Limitation: A cross of crosses records and reports its bins in terms of the bins of its elements, as shown in [Table 1-1](#).

Table 1-1 Cross coverage bins for crossed coverpoint values

Coverpoint values				Crosses		Bin hit in <code>cr_0_X_cr_1</code>
<code>cp0</code>	<code>cp1</code>	<code>cp2</code>	<code>cp3</code>	<code>cr_0</code>	<code>cr_1</code>	
1	10	14	15	low_b	high_b	one_b
14	10	14	15	14, 10	high_b	([14, 10], high_b)
8	10	7	15	8, 10	7, 15	([8, 10], [7, 15])

Note:

[14, 10], [8, 10] and [7, 15] are autocross bins in the respective crosses.

Limitation

Currently, this feature cannot be used together with the hierarchical cross feature.

Cross-of-a-Cross Exclusion Support

VCS now supports exclusion of a cross-of-a-cross or its bins in the URG-based exclusion flow. This feature helps you achieve your coverage targets by eliminating unreachable or uninteresting coverage goals.

Filtering Instances with no Coverable Objects

From this release, DVE Coverage GUI, by default, displays only the part of design hierarchy where the coverage data exists, and hides the instances/modules with no coverable objects (instances/modules that have a hierarchical coverable count of zero).

Support for Propagating Constants Through Instance Arrays

Constant filtering (activated with the coverage option `-cm_noconst`) is an important feature in VCS coverage. It identifies compile-time constants in the design and filters them out from the coverage results. While identifying constants, VCS considers the propagation of constants through the port connections between instances. Previously, there was a limitation in this constant identification and propagation mechanism with instance arrays. The previous flow was not able to find constants in a single-instance (each element in an instance array) and constants were not being passed through a single instance.

In this release, this limitation is removed with the introduction of the constant propagation through instance array feature.

This feature helps instance arrays behave like any other normal instance from a constant filtering perspective. Constants are identified from single instances and passed through port connects of single instances. Instance arrays include some special features regarding port connections which are duly regarded during constant.

Support for Reporting HVP Annotated Results in XML Format

In this release, the HVP annotated results are reported in the Unified Report Generator (URG) XML format. With this XML report, you can extract all types of coverage information using a common XML reader and create a coverage report in your own format.

For more information, see the section on “Reporting in Unified Report Generator XML Format” in the *VCS / VCS MX Online Documentation*.

Toggle Coverage MDA Exclusion

DVE Coverage supports Verilog MDA exclusion. You can exclude whole MDA range item or individual bits of an MDA range item, in the Detail Window.

By default, VCS coverage engine does not monitor MDAs as part of toggle coverage. You can use the `-cm_tgl` and `-cm_tgl mda` options, as shown below, to enable monitoring of MDAs.

```
%vcs -cm_tgl -cm_tgl mda
```

UVM and VMM Features

- [“Using the RALF C++ API” on page 25](#)

- [“RALF C++ API Class Reference” on page 25](#)
- [“Using UVM Register C++ Interface” on page 25](#)
- [“Generating RALF and UVM Register Model from IP-XACT” on page 25](#)
- [“Customizing the VMM Performance Analyzer Reports” on page 26](#)

Using the RALF C++ API

From this release, there is full access to parsed RALF data (through C++ routines), which can be used to implement a customized RTL code generator, or any other feature that needs RALF data.

RALF C++ API Class Reference

RALF C++ API Class Reference describes the C++ classes and methods that comprise the RALF C++ API.

Using UVM Register C++ Interface

UVM register C interface allows firmware and application-level code to be developed and debugged on a simulation of the design.

Generating RALF and UVM Register Model from IP-XACT

IP-XACT 1.5 now allows you to generate UVM RAL model directly from an IP-XACT file in a single step.

Customizing the VMM Performance Analyzer Reports

The `vmm_perf_analyzer` class now allows you to customize your VMM performance analyzer reports by using the following set of protected type variables, which were local type variables in the previous versions:

- `time min_time;`
- `time max_time;`
- `time active_time;`
- `time min_abort_time;`
- `time max_abort_time;`
- `time abort_time;`
- `int n;`

For more information on the above sections, see *VCS / VCS MX Online Documentation*.

MVSIM Native Low Power Features

- [“Automated Low Power Assertions” on page 27](#)
- [“Error for Conflicting Port State Names in the `add_port_state` Command” on page 27](#)
- [“`-simulation_only` Support” on page 28](#)
- [“Error Checking for Isolation Strategy with `-location parent` on Top-level Ports” on page 28](#)
- [“Synopsys Low Power Flow Consistency Check” on page 29](#)

- “Handling of Constants for Corruption and Isolation” on page 29
- “Coverage Support for Low Power Objects” on page 29

Automated Low Power Assertions

Starting from this release, all the low power runtime messages capturing the power events during the simulation will be in the below format:

```
[time] [severity] [msg_id] <Message>
```

The following two modes of message control are supported:

- Runtime Config File
- Function calls from Testbench

Error for Conflicting Port State Names in the `add_port_state` Command

Two Supply Ports which are at two different scopes can be connected using a Supply Net. The Supply Net serves as a way to propagate voltage between these two Supply Ports.

Starting from this release, you cannot use the same port state name with different voltage values for multiple supply ports tied together. VCS generates the `Duplicate Port States` error message for such usage.

-simulation_only Support

With the `-power=sim_only` compile-time option, the below list of UPF commands/options/attributes will result in parsing error, if they are not part of the UPF file that is loaded with `load_upf <upf_file> -simulation_only`.

Table 1-2 Options Supported with -simulation_only

Commands	Options/Attributes
<code>bind_checker</code>	All options and custom options
<code>load_upf</code>	<code>-model</code>
<code>create_power_domain</code>	<code>-simulation_only</code>
<code>set_design_attributes</code>	Simulation-only: UPF_dont_touch SNPS_override_pbp_corruption SNPS_reinit SNPS_dont_reinit Non simulation-only: (SNPS_default_power_upf2sv_vct, SNPS_default_ground_upf2sv_vct, SNPS_default_power_upf2vh_vct, SNPS_default_ground_upf2vh_vct, SNPS_default_power_sv2upf_vct, SNPS_default_ground_sv2upf_vct, iso_source, iso_sink, related_supply_default_primary)

Error Checking for Isolation Strategy with `-location parent` on Top-level Ports

When an isolation strategy applies to a top-level port, the isolation strategy can be implemented only in the “self” location that is visible in the design scope.

MVSIM Native generates an error message when an isolation strategy with location “parent” is specified on top-level ports of the design.

Synopsys Low Power Flow Consistency Check

You can use the `-power=snps_lp_flow` compile-time option to maintain consistency of UPF across the Synopsys Low Power Flow. With this option, the below list of UPF commands/options will result in an error as these are not supported across all the tools in Synopsys Low Power Flow.

Handling of Constants for Corruption and Isolation

MVSIM native mode supports the following behaviors for corruption and isolation of constants. Literals `1'b1` and `1'b0` in port maps or RHS of continuous assignments are treated as constants.

- Default Behavior
- Supply Based Corruption of Constants
- Avoid Corruption for Constants in Continuous Assignments
- Skip Corruption of Constants

Coverage Support for Low Power Objects

You can use the `-power=coverage` compile-time option to automatically create covergroups for low power objects based on the power intent. The covergroups are created for the following low power objects:

- PST states/transitions

- Power Switch State states/transitions
- Supply Net/Port states/transitions (Root Supply)
 - States defined through `add_port_state` of connected port
 - States inferred from `supply_expr` of `add_power_state` for connected SupplySet
- Supply Set Power State states/transitions
 - `primary`
 - `isolation_supply_set`
 - `ret_supply`
- Supply Set simstate states/transitions
 - `primary` (domain simstate)
 - `isolation_supply_set`
 - `ret_supply`
 - `default_isolation`
 - `default_retention`

SystemC Features

- [“Modeling of SystemC Designs with SCV” on page 31](#)
- [“Macros for `sc_main`” on page 31](#)
- [“Reporting of the SystemC `sc_report_handler` Messages in the Log File” on page 31](#)
- [“Renaming Newsync to Deltasync” on page 32](#)

- “Enhanced Debugging of SystemC Kernel Errors” on page 32
- “Automatic Creation of Portmap File” on page 32
- “Improved Debugging in CBug” on page 33
- “Support for SystemC 2.3 and gcc 4.5.2” on page 33

Modeling of SystemC Designs with SCV

VCS now provides an SCV package that you can use to easily model your designs containing SystemC and SCV (SystemC Verification Library).

Macros for `sc_main`

A couple of macros, as described below, are provided to ease implementation of the `sc_main` function. The following are the two macros:

- `SNPS_REGISTER_SC_MAIN`
- `SNPS_REGISTER_SC_MODULE`

Reporting of the SystemC `sc_report_handler` Messages in the Log File

VCS now logs SystemC `sc_report_handler` messages in a log file. Until now, VCS logged only HDL/HVL messages in a log file but not the messages from SystemC. Hereafter, `simv -l logfile` will capture all the messages sent to `sc_report_handler()` in the log file. Log file will now have all the HDL/HVL messages along with the `sc_report_handler` messages.

Renaming Newsync to Deltasync

The 'newsync' loop has been renamed as 'deltasync' loop and has been made default.

As this switch is default, you may see differences in your simulation behavior. The 'oldsync' loop which was default in the previous releases has been introduced as 'blocksync' loop to help you revert to the old flow. To revert to the old flow, use `-sysc=blocksync`.

For more information on the advantages of deltasync loop and the possible backward compatibility issues, refer to the migration helper document.

Enhanced Debugging of SystemC Kernel Errors

VCS now provides an effective mechanism to debug your design issues during elaboration and runtime. Whenever a SystemC kernel error occurs, error messages were not really helpful to enable you identify which part of your source code was causing the error. Debugging such errors was way too tedious.

Now, VCS provides a new function `cbug_stop_here()` that is called whenever a SystemC kernel error occurs. You can make use of this function to place a breakpoint in this function and see the stack trace to know the source code that is causing the error.

Automatic Creation of Portmap File

VCS now writes the portmap file automatically thus making the task of mapping the ports easier across the languages. When SystemC is instantiated in HDL or vice-versa, you must write port map file for mapping the data types between the languages. This is a tedious job when we have many ports.

Now, a default portmap file can be created by using the option “`-sysc=gen_portmap`” while generating the wrapper.

Improved Debugging in CBug

The following debugging capabilities have been provided in CBug to ease troubleshooting issues with your SystemC designs.

- Viewing sc-signal of User-defined struct in Waveform Window
- Driver/Load Support for SystemC Designs in Post Processing

Support for SystemC 2.3 and gcc 4.5.2

From this release onwards, VCS supports the SystemC 2.3 version. This is supported with gcc 4.5.2 and this version of gcc 4.5.2 is also available as part of the VG GNU package only from this release onwards.

By default, VCS still uses SystemC 2.2. To use SystemC 2.3, you must explicitly specify the command option `-sysc=2.3`.

LCA Features in This Release

Limited Customer Availability (LCA) features are newer features that are released for customer testing and feedback. They are not yet considered ready for general release. You can use LCA features without requiring a special license.

To enable LCA features, use the `-lca` compile-time option:

```
% vcs -lca
```

Note:

For this release, a VCS version string is not necessary as in previous releases. When you enter the `-lca` option, VCS issues the following general warning:

```
Warning: LCA features have been enabled by -lca argument
in the command line.
```

Note:

Some exceptions (such as DVE with Cbug) do not require the `-lca` switch.

The following are the new LCA features in this release:

- [“Signature-based Control for Deferred Assertions and RT Checks” on page 36](#)
- [“PLI/DPI/DirectC Enhancement in the Unified Profiler” on page 45](#)
- [“Single Text Format Report From the Unified Profiler” on page 47](#)
- [“Support for “with” Clause in Cover Groups” on page 47](#)
- [“Profiler Hypertext Links to the Source Files” on page 47](#)
- [“Reducing Disk Space for Post-process only Debug” on page 62](#)
- [“Reducing Compile Time for Post-process only Debug” on page 62](#)
- [“Integrating DVE with Protocol Analyzer” on page 62](#)
- [“IEEE Verilog Std 1364-2005 Encryption” on page 64](#)

The following are LCA features in this and previous releases:

- [“Analog and Mixed-Signal Feature: Multi-Driver Support for Wreal” on page 37](#)

- “UCLI Enhancement in vpd2vcd Utility” on page 37
- “DVE Viewing the Full Design Hierarchy in a Partially Dumped VPD” on page 37
- “Tcl Fileevent Support for UCLI and DVE” on page 38
- “Coverage Report-Time Exclusion” on page 38
- “Coverage Analysis of Unreachable Verilog Code” on page 38
- “Echo Features” on page 39
- “Echo Procedural Sampling Enhancements” on page 40
- “Hierarchical Cross Coverage” on page 40
- “Support for Adaptive Exclusion in DVE GUI” on page 41
- “Multicore DLP Autopartitioning” on page 42
- “Multicore ALP FSDB Dumping” on page 42
- “The Unified Profiler” on page 43
- “Constraint Profiling is Integrated into the Unified Profiler” on page 44
- “Extern Task and Function Calls through Virtual Interfaces” on page 52
- “Integrating SystemC with Innovator” on page 53
- “Profiling the SystemC Portion of a Design” on page 53
- “Partition Compile” on page 54
- “SAIF Support for SystemVerilog Datatypes” on page 58
- “Generating SystemC Profiling Reports” on page 58

- “SystemC Simulating Virtual Platform Models” on page 59
- “SystemVerilog Assertions” on page 59
- “Fast Compilation” on page 61
- “VMM SystemC (VMM-SC)” on page 61
- “VMM-AMS” on page 62
- “Verification Planner Doc Annotator” on page 63
- “Verification Planner XML Doc Plan Generator” on page 63
- “Verification Planner HVP Interactive Editor” on page 63

Signature-based Control for Deferred Assertions and RT Checks

VCS now provides an enhanced way to control RT and deferred assertions (labelled assertions) using the “signature” method.

VCS generates a “signature” associated with each error/warning report related to labelled deferred assertion and RT checks.

A signature is a string made up of tokens extracted from the function call stack trace of assertions. The tokens are separated by dots. The signature is not an XMR, but an XMR-like string which is not restricted to legal Verilog syntax.

Analog and Mixed-Signal Feature: Multi-Driver Support for Wreal

A resolution function is required when there are multiple drivers. VCS provides a built-in resolution function which you can select using the `wreal <res_func>` compile-time option, where `<res_func>` can be either `res_def`, `res_sum`, `res_min`, or `res_max`.

UCLI Enhancement in vpd2vcd Utility

You can use the `vpd2vcd` utility `-expand_dumpport_scope` option to generate a VCD file with `$scope` information in a hierarchical format (the default format is flattened).

Syntax

```
% vpd2vcd -f optionfile <vpdfile> <vcdfile> \  
-expand_dumpport_scope
```

DVE Viewing the Full Design Hierarchy in a Partially Dumped VPD

You can view the complete design hierarchy in DVE post-process mode using the **Use design debug library for design hierarchy for post process** preference option, even if you have dumped partial hierarchy in a VPD file. After enabling this option, reload the database in DVE to view the full hierarchy.

Tcl Fileevent Support for UCLI and DVE

DVE and UCLI now support the Tcl fileevent feature (socket-based Tcl communication) between a server program and a client program using the `ucliCore::tempPause` and `ucliCore::tempPauseResume` UCLI commands.

Coverage Report-Time Exclusion

In this release, the granularity of exclusion file version checks is reduced from the existing file level to a module or metric-variant level. This is achieved through changes in the format of exclusion files created by the UCAPI API `covdb_save_exclude_file`.

For more information, see the section on “Coverage Report-Time Exclusion” in the *VCS / VCS MX Online Documentation*.

Coverage Analysis of Unreachable Verilog Code

VCS now detects unreachable conditions by analyzing procedural blocks, evaluating the conditions controlling them, and determining which blocks are actually unreachable. All conditions found to be unreachable because of block-level unreachability are marked as unreachable in the coverage report. Also, from this release onwards, VCS allows you to specify module-specific constant signals within the constant configuration file (`-cm_constfile` option).

Echo Features

The following Echo features are available with this release:

- [“Support for Procedural Sampling” on page 39](#)
- [“Crosses with Non-Random Cover Points” on page 39](#)

Support for Procedural Sampling

Echo analyzes procedural context to find targetable cover points. Echo supports the following two aspects of procedural sampling:

- Sampling a non-random variable that is assigned with a targetable random variable. Echo targets a non-random cover point when the non-random variable is assigned with an expression of random variables.
- Randomizing a sibling class of sampled variables. At runtime, Echo collects targetable holes from the sibling classes. Echo currently supports only one level of inheritance. Note that all the variables involved in the expression of a cover point must be inherited from the targeted base class.

Crosses with Non-Random Cover Points

You can target a cross with non-random cover points using Echo. The cross must contain at least one random cover point.

While collecting Echo targetable cross holes for non-random cover points, VCS ignores cross-bin combinations with bins that are not sampled. However, if the non-random cover point is properly sampled during the first few cycles, it is collected as a targetable hole at the next collection.

If transition bins are involved in cover points that contribute crosses, VCS ignores the cross bins (including transition bins) when collecting the holes.

Echo Procedural Sampling Enhancements

Echo procedural sampling technique supports the following three new coding styles:

- Sampling a non-random variable that has been assigned a targetable random variable present as a member of another class.
- Passing a randomized object through function/task parameter.
- Stand-alone covergroups.

For more information about Echo procedural sampling, see the Echo chapter in the *VCS / VCS MX Online Documentation*.

Hierarchical Cross Coverage

SystemVerilog supports the covergroup construct to capture the functional coverage model. One of the requirements for a cross is that all the crossed elements must be declared within the same covergroup. This allows seamless and unambiguous sampling, among other benefits.

However, advanced coverage modeling requires the ability to specify crosses of elements in different covergroups. These are called hierarchical covergroup features. The VCS SystemVerilog implementation extends the IEEE 1800 standard to enable this feature.

Support for Adaptive Exclusion in DVE GUI

The adaptive exclusion feature enables you to reuse the exclusion file when the design and coverage database change. Reusing the exclusion file improves productivity in the coverage flow.

MVSIM NLP LCA Features

The MVSIM NLP LCA features in this release are as follows:

- [“Support for Isolation of SVD Constructs” on page 41](#)
- [“MVSIM Native Mode Partition Compile Support” on page 41](#)

Support for Isolation of SVD Constructs

MVSIM native mode now supports virtual isolation for the following SystemVerilog Design (SVD) constructs:

- logic, int, struct, enum, reg, shortint, and union
- Multidimensional arrays

MVSIM Native Mode Partition Compile Support

You can apply the incremental changes to the design source or IEEE 1801, also known as the Unified Power Format (UPF) files, or you can modify the low-power options.

Make sure the `synopsys_sim.setup` file has the UPF library mapping:

```
upf: $VCS_HOME/$platform/packages/upf
```

Where, `$platform` is RHEL32 for a 32-bit VCS build and RHEL64 for a 64-bit VCS build.

Multicore DLP Autopartitioning

If your design is suitable for multicore simulation, VCS can automatically generate a multicore DLP partition file when you enter the `-parallel+autopart=N` compile-time option, where N is the number of cores (or multicore DLP slave partitions) in your design. The value of N should be at least 2.

The partition file VCS writes is named `autopart.cfg`. VCS writes this file in the directory where you enter the `vcs` command.

Multicore ALP FSDB Dumping

FSDB is a simulation history file format for Novas (SpringSoft) tools.

To enable the parallel writing of this simulation history file, use the `-parallel+mtfsdb` compilation or runtime option. The `mt` in `+mtfsdb` stands for multi-threaded.

To enable the same executable for both serial and parallel FSDB dumping, enter the runtime option instead of the compilation option.

You can use the same system tasks, beginning with `$fsdb`, in your Verilog source code.

The Unified Profiler

The new version of the unified profiler makes it easier to see the cause of increases in the amount of CPU time or machine memory needed in different simulation runs. With this information, you can make the adjustments needed to improve simulation performance. The new profiler lets you look at accumulative and comparative views of multiple profile databases:

- The accumulative view shows you how the CPU time or machine memory usage varies in a series of profile databases for different simulation runs.
- The comparative view shows you this profile data when comparing two profile databases.

This version also has new PLI/DPI/DirectC views that tell you the CPU time or machine memory usage of each C/C++ function call.

There is also a new layout for the HTML profile information.

To use the unified simulation profiler, follow these steps:

1. Compile your Verilog design, or elaborate your VHDL or mixed-HDL design, using the `-simprofile` compile-time option.
2. At runtime, enter the `-simprofile` runtime option with a keyword argument or suboption `time` or `mem` to specify the type of data VCS collects during the simulation.
3. Run the profiler using the `profrpt` command and its command-line options. Then review the reports created by the profiler.

For more information on the unified profiler, see the *VCS LCA Features Guide*.

See “Unified Profiler Limitations” on page 98.

Constraint Profiling is Integrated into the Unified Profiler

Constraint profiling is integrated in the unified profiler. This integration is a new LCA feature in this G-2012.09.

This integration adds the following views to the profile reports:

- the Time Constraint Solver view
- the Memory Constraint Solver view

These views tell you, in detail, the calls to the `randomize()` method that use the most CPU time or the most machine memory. With this information you can consider revising your constraints on the random variables to use less of these resources.

To tell `profrpt` to generate these views the following is added to the use model:

The `profrpt -view` option’s arguments now include:

- `time_solver` to specify generating the Time Constraint Solver view
- `mem_solver` to specify generating the Memory Constraint Solver view.

The `time_all` and `mem_all` arguments also generate these views.

Note:

The `profileReport.html` file does **not** contain a new component for constraints, The `CONSTRAINTS` component was in this file in previous releases.

As in previous releases, the left pane of the profileReport.html file, after selecting a profile database, contains a drop down menu for views. This menu now contains the the following for constraint profiling:

- the Time Constraint Solver view
- the Memory Constraint Solver view

PLI/DPI/DirectC Enhancement in the Unified Profiler

The Time and Memory PLI/DPI/DirectC views are enhanced in the G-2012.09 release to include, along with the name of the PLI user-defined system task or DPI/DirectC function call, the following:

- the path name of the Verilog or SystemVerilog source file that contains that system task or function call
- the line number in that source file that contains that system task or function call

The pathname and line number are blue because they are a hypertext link, as shown in [Figure 1-1 on page 46](#).

Figure 1-1 The Time PLI/DPI/DirectC View for a DPI Function Call

Time PLI/DPI/DirectC View

Name	Time	Percentage
DPI	1.36 s	97.46 %
stage_cfunc	920.79 ms	65.77 %
stage_cfunc	334.91 ms	23.92 %
/file_system/big_design/VCS_user_files/dpi1.v:38		
stage_cfunc	307.67 ms	21.98 %
/file_system/big_design/VCS_user_files/dpi1.v:42		
stage_cfunc	278.21 ms	19.87 %
/file_system/big_design/VCS_user_files/dpi1.v:34		
run_all_stage	443.68 ms	31.69 %
run_all_stage	443.68 ms	31.69 %
/file_system/big_design/VCS_user_files/dpi1.v:13		
total	1.36 s	97.46 %

Page: 1

hypertext link

This view reports the CPU time used by:

- Three calls of the DPI function named stage_cfunc, the view reports the total CPU time and percentages of all three of these calls collectively and individually. The hypertext links are for the individual calls.
- One call of the DPI function named run_all_stage

When you click on one of this links, the browser opens another window containing the source file with the selected line. So if, for example, we click the hypertext link for:

[/file_system/big_design/VCS_user_files/dpi1.v:38](#)

The browser opens another window to show the contents of `dpi1.v`, highlighting line 38.

Single Text Format Report From the Unified Profiler

Text format views are merged together into a text file named `profileReport.txt` in the current directory. This is a new feature in G-2012.09.

You specify text format reports with the `-format text` or `-format all` option and argument on the `profrpt` command line.

If you run the `profrpt` report generator more than once, the utility overwrites the `profileReport.txt` file in the current directory so that its profile information is from the last run.

When you specify text format reports the `profrpt` utility also creates separate text files for each view in the profile report directory. These separate text files for each view have names such as `PeakMemInstanceView.txt` or `TimeConstr.txt`.

Support for “with” Clause in Cover Groups

VCS now extends the syntax of cover point and cross bins to support “with” clause. The “with” clauses help you to define cover points clearly, which otherwise would have needed a lot of writing in cover group specification

Profiler Hypertext Links to the Source Files

This is a new feature in G-2012.09

The pathnames of source files in any of the HTML views are hypertext links. Clicking on one of these links opens a new window of the browser to display that source file. This section describes and illustrates this feature.

To use this feature do the following:

1. Compile a design with the `-lca` and `-simprofile` options.
2. Run the simulation with the `-simprofile time/mem/time+mem` option and keyword argument to enable VCS to collect time/memory/time&memory profile information.
3. Run the `profrpt` utility to create the HTML views.
4. Open the `profileReport.1.html` file.
5. Select a profile database in the left pane.
6. Select the Time Instance view.

Figure 1-2 Time Instance View

click here

Time Instance View

Instance	Inclusive Time	Percentage	Exclusive Time	Percentage
tb_top	3.52 s	23.46 %	3.52 s	23.46 %
total	3.52 s	23.46 %	3.52 s	23.46 %

Page: 1

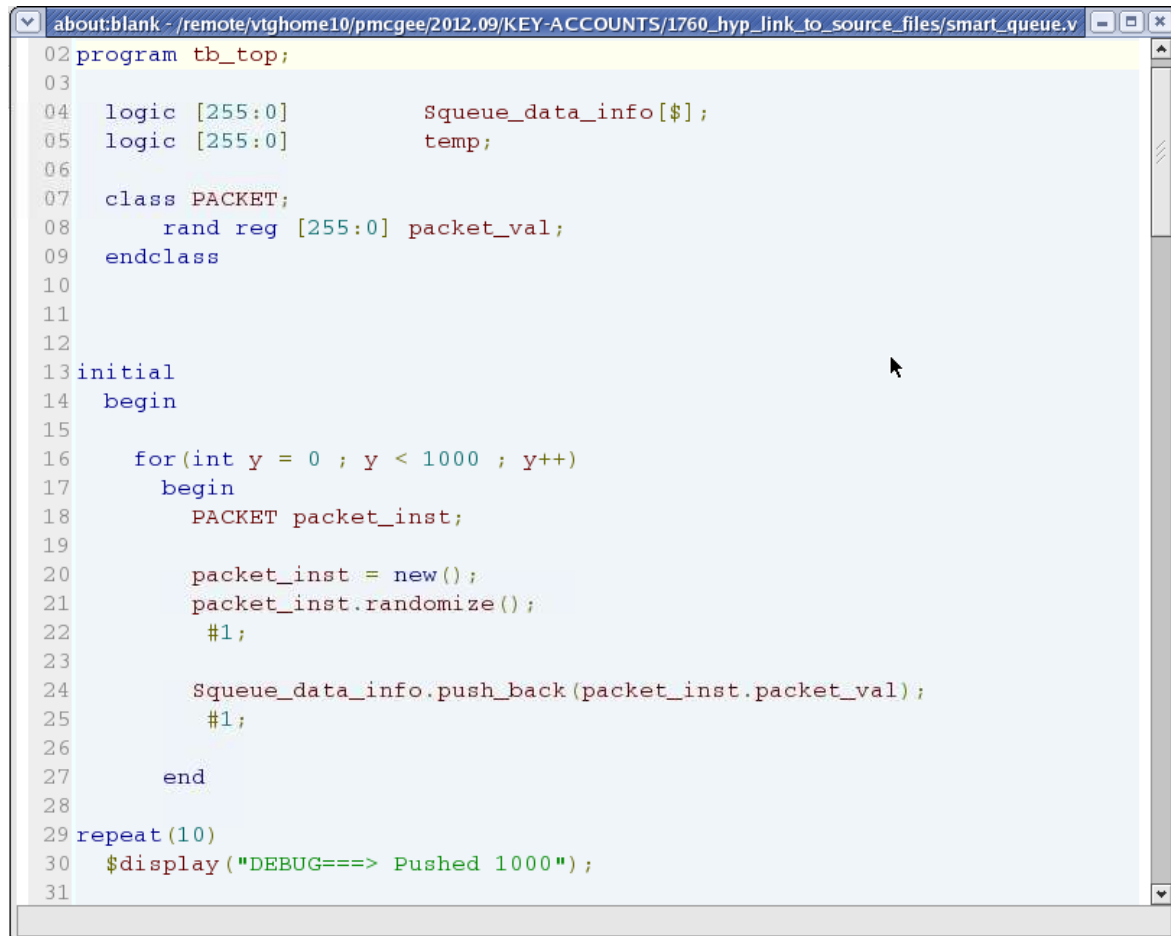
- Click on an instance in the view. This adds information about the instance to the bottom of the HTML page, including the path to the source file that contains the module instance, see [Figure 1-3 on page 50](#).

Figure 1-3 Time Instance View Expanded

Instance Information		
Instance Name		tb_top
Exclusive Time		3.52 s
Exclusive Percentage	click here	23.46 %
Inclusive Time		3.52 s
Inclusive Percentage		23.46 %
Master Module		tb_top
Child Instance Number		0
Source Information	/file_system/big_design/VCS_user_files/smart_queue.v:2	

8. Click on the blue path name of the source file, this is a hypertext link. The browser opens a new window to display the source file, see [Figure 1-4 on page 51](#).

Figure 1-4 The New Source File window



The image shows a window titled "about:blank - /remote/vtghome10/pmcgee/2012.09/KEY-ACCOUNTS/1760_hyp_link_to_source_files/smart_queue.v". The window contains Verilog code for a testbench. The code is as follows:

```
02 program tb_top;
03
04   logic [255:0]      Squeue_data_info[$];
05   logic [255:0]      temp;
06
07   class PACKET;
08     rand reg [255:0] packet_val;
09   endclass
10
11
12
13 initial
14   begin
15
16     for(int y = 0 ; y < 1000 ; y++)
17       begin
18         PACKET packet_inst;
19
20         packet_inst = new();
21         packet_inst.randomize();
22         #1;
23
24         Squeue_data_info.push_back(packet_inst.packet_val);
25         #1;
26
27       end
28
29 repeat(10)
30   $display("DEBUG==> Pushed 1000");
31
```

Extern Task and Function Calls through Virtual Interfaces

You can define tasks and functions in an interface with one or more of the modules connected by the interface. You declare them as export in a modport or as extern in an interface. When they are called through virtual interfaces, the actual task or function that VCS executes depends on the interface instance of the virtual interface.

The definition of extern subroutines within an interface shall observe the following rules:

- Each interface instance may have different implementations of its extern subroutines:
 - The same extern subroutine of different interface instances can be defined in different modules.
 - Different extern subroutines of the same interface instance can be defined in different modules.
- Every interface instance must have one and only one definition of its extern subroutines:
 - If an interface instance contains an extern subroutine, one of the modules connected must define that subroutine.
 - An extern subroutine of an interface instance cannot be defined in more than one module.
 - The module implementing any extern subroutine can be instantiated only once.
- These rules also apply for exported subroutines in modports.

Note the following limitations:

- Extern task and function calls through virtual interfaces are not supported in the separate compile flow.
- Calling extern functions through virtual interfaces is not supported in constraints.

Integrating SystemC with Innovator

Innovator is an integrated virtual platform development environment provided by Synopsys. It supports virtual platform assembly from SystemC TLM-2 hardware models and software development on top of it. The integration of SystemC with Innovator provides debugger integration that is focused mainly on Platform Analyzer.

This feature enables interactive Platform Analyzer functionality in the context of VCS simulation. This allows VCS to use the Innovator SystemC kernel while retaining VCS/DVE HDL visualization capabilities.

Profiling the SystemC Portion of a Design

VCS unified profiler supports SystemC. With this feature you can see the percentage of simulation time spent in all SystemC related code in a VCS-SystemC simulation. This includes the overhead of SystemC scheduling, as well as value updates between SystemC and HDL.

Partition Compile

Partition compile is a VCS feature that allows you to compile portions of the design and get significantly faster turnaround time during the iterative process of compile and recompile.

With partition compile you specify partitions in your design that you expect to revise and recompile often. VCS MX only recompiles the modified partitions.

The partition compile feature requires a VCS MX installation. However, you don't need any additional VCS MX licenses or a special license. Verilog-only users can use their existing licenses for partition compile.

You can specify partitions two ways:

- In an `+optconfigfile` configuration file (for Verilog/SV designs)
- In a V2K or SV configuration (for Verilog/SV/VHDL/SystemC and MX designs)

Some of the partition compile features are:

- Compile partitions independently
- Support for XMRs across partitions
- Reduced disk space across multiple partitions
- Compile partitions in parallel
- Faster IP integration in SOC environments

Partition compile is very similar to the existing VCS MX regular compile use model. No changes are required to the source code of the DUT and testbench when migrating to partition compile and runtime performance is not compromised when compared to the regular VCS compile flow.

With partition compile, you partition the DUT so that separate partitions are created for source code that is frequently recompiled. Partition compile provides faster turnaround time by recompiling only the modified partitions of the design, while the remainder of the design (in one or more partitions) is not recompiled.

You can also compile these partitions in parallel on multicore machines. This improves compile-time performance from scratch compile in a regression mode.

Partition compile also helps reduce disk space for multiple tests if the design and test are in separate partitions.

For more information, see the “Partition Compile” chapter in the *VCS / VCS MX Online Documentation*, including these sections:

- Instance or Cell Based Partitions
- Package Based Partitions
- 2-Step Partition Compile for Verilog/SystemVerilog Designs
- Specifying a Location for the Partition Data Generation
- Parallel Compilation of Tests
- Parallel Compilation of Partitions
- Cross-Module References (XMRs)
- Partition Compile Flow for SystemC-on-Top Designs

- Specifying Partitions in a VHDL Configuration File
- MVSIM Native Mode in Partition Compile
- Scenarios Causing a Recompilation of a Partition
- Achieving the Best Turnaround Time (TAT) with Partition Compile

Compiling SystemC Designs in Partition Compile Flow

Starting with this release, you can compile your SystemC-on-top designs with VCS in the partition compile flow. Earlier releases allowed the partition compile flow only for SystemC-down designs.

Partition Compile Limitations

The following technologies are not supported with partition compile:

- Gate-level performance limitation. The `-hsopt=gates` compile-time option for gate-level performance.
- OpenVera testbench shell and shared object files, as specified with the `-ntb_cmp` and `-ntb_vl` elaboration or compile-time options.
- The timing optimizer (`+timopt`).
- Radiant technology performance optimizations (`+rad`).
- Searching for the `PATHPULSE$ specparam` in specify blocks with `+pathpulse`.

You cannot use partition compile with self instances of virtual interfaces `interface::self()`.

For SDF backannotation with partition compile, the following limitations apply:

- The delays in any SDF file must be for a single partition; they cannot cross partition boundaries.
- Two or more instances of a partition cannot have different SDF files.
- Conditional and delayed SDF annotation are not supported.
- For mixed-HDL designs, the VHDL part cannot have SDF backannotation.

Code coverage has the following limitation:

- Detecting coverable objects (statements, conditions, toggles) that can never execute and automatically excluding them from coverage with `-cm_noconst`.

AMS is not supported, but mixed-signal simulation with FastSPICE, NanoSim, or XA is supported.

Using the `+optconfigfile` method of specifying partitions is only enabled for Verilog and SystemVerilog code.

For mixed-HDL designs:

- A partition that includes VHDL inside Verilog or Verilog inside VHDL cannot be specified with the `+optconfigfile` method.
- A partition for Verilog that is inside VHDL must be specified with a VHDL configuration to specify the Verilog partitions.
- A partition for VHDL that is inside Verilog must be specified with a Verilog configuration to specify the VHDL partitions.

The following limitations apply in MVSIM native mode:

- Low-power isolation for MX designs is not supported .

- The UPF command `-set_level_shifter` is not supported.
- For VHDL, you cannot specify an `always_on` signal in a file and cannot use the `UPF_dont_touch` attribute.
- `set_related_supply_net` usage is not supported for MX designs.
- The voltage-aware flow is not supported.
- Isolation cells are not shown in the DVE Hierarchy Pane.

SAIF Support for SystemVerilog Datatypes

SAIF supports monitoring of SystemVerilog datatypes in SystemVerilog designs. To make this work, pass the `sv` keyword to the `$set_gate_level_monitoring` system task.

Generating SystemC Profiling Reports

VCS now enables you to generate profiling reports for SystemC designs. Traditionally, identifying the time taken by the SystemC design during a simulation run was a problem. In the previous release, VCS provided the complete time taken by your SystemC design. This release enhances this support and provides more detail on the time taken by your SystemC design.

SystemC Simulating Virtual Platform Models

Starting with this release, you can simulate virtual platform models from SG (formerly Coware/Innovator platform models) with VCS. After simulating, you can debug your issues with full debug capabilities using SG debug tools, while limited debug capability is also provided with DVE.

SystemVerilog Assertions

The following SystemVerilog features are LCA for this release:

- [“Using SystemVerilog Constructs Inside vunits” on page 59](#)
- [“Using Fail-only Assertion Evaluation Mode” on page 60](#)
- [“SVA Extensions” on page 60](#)

Using SystemVerilog Constructs Inside vunits

VCS now supports using SystemVerilog (SV) and SystemVerilog Assertions (SVA) inside a verification unit (vunit). This feature:

- Allows vunits in your code to be interpreted as SV or SVA compliant code.
- Allows you to bind checkers containing assertions and modeling code in a vunit to a design more conveniently than with other use models.

Using Fail-only Assertion Evaluation Mode

Fail-only is a new assertion evaluation mode you can use to improve assertion runtime performance. To use this optimization, specify the `-assert failonly` or `-assert concfail` compile-time option.

Immediate/deferred assertions and concurrent assertions without pass action blocks, local variables, match operators, or multiple clocks tend to benefit from this evaluation mode.

SVA Extensions

This section describes VCS extensions to SystemVerilog Assertions that are not standard LRM features.

Functions for Counting Unknown Bits

Use the following system functions to count the number of X and Z values in a vector.

- `$countx(expression)` returns the number of expression bits set to X.
- `$countz(expression)` returns the number of expression bits set to Z.
- `$countunknown(expression)` returns the number of expression bits set to either X or Z.
- `$onedriven` returns true if only one bit of the expression is not Z and its value is defined (not X).
- `$onedriven0` returns true if at most one bit of the expression is not Z, and if such a bit exists, its value is defined (not X).

Fast Compilation

VCS has new compile-time performance optimizations called fast compilation that you can use to reduce compile time for your design (and therefore overall turnaround time). You enable fast compilation with the `-fastcomp` compile-time option.

There are two levels of fast compilation, specified by the `-fastcomp=0` and `-fastcomp=1` compile-time options:

- `-fastcomp=0` is the same as `-fastcomp` (without the argument) and is the generally preferred option.
- `-fastcomp=1` applies more aggressive compile-time performance optimizations.

VMM SystemC (VMM-SC)

VMM now supports the development of verification environments in SystemC by providing a set of base classes in SystemC. The VMM SystemC library also provides an infrastructure that offers greater productivity and easier integration with VMM-based verification environments written in SystemVerilog.

VMM-SC is composed of a set of base classes, utility classes, and APIs. It provides a unified reference verification methodology in conjunction with VMM-SV (VMM SystemVerilog).

For more information on VMM-SC support, see the *LCA Features* in the *VCS / VCS MX Online Documentation*.

VMM-AMS

VMM enables you to create reusable verification environments for analog and mixed-signal DUTs by providing a set of base classes, utility classes, and APIs for the same. For more information, see the *VCS / VCS MX Online Documentation*.

Reducing Disk Space for Post-process only Debug

If you want to perform only post-process debug, then you can use the `-debug_perf=splitdebugdir` compile-time option to significantly reduce the `simv.daidir` disk space. When this option is used, VCS creates the `debug_dump` directory in the `.daidir` directory at compile-time.

Reducing Compile Time for Post-process only Debug

If you want to perform only post-process debug with an existing VPD file and recreate `simv.daidir`, then you can use the `-static_dbgen_only` compile-time option and significantly reduce the compilation time.

Integrating DVE with Protocol Analyzer

Protocol Analyzer Integrated Into DVE

VCS enables you to invoke Protocol Analyzer from DVE, and connect it automatically to DVE. You can directly load VIP configured session files from Protocol Analyzer to DVE.

Verification Planner Doc Annotator

Verification Planner doc annotator enables creation of verification plans using Microsoft Office (.doc) documents, and back-annotation of coverage and other scores into the .doc plan. The .doc verification plan must be formatted properly with predefined styles and keywords built into the Verification Planner doc annotator.

You can use MS-Word 2003 and later versions to create the .doc verification plan. For more information, see the LCA features book in the *VCS / VCS MX Online Documentation*.

Verification Planner XML Doc Plan Generator

Verification Planner XML doc plan generator provides a way to convert verification plans in HVP language format to Word doc XML plan format. The doc plans generated from the Word plan generator can be fed into the Verification Planner word annotator to annotate scores. This application currently supports MS-Word 2003 and later versions only. For more information, see the LCA features book in the *VCS / VCS MX Online Documentation*.

Verification Planner HVP Interactive Editor

Verification Planner HVP editor allows you to interactively create a verification plan. Verification Planner is a technology you can use to create a comprehensive model to track the progress of verification projects. Once you have created a verification plan, you can then annotate that plan with live verification data using the Unified Report Generator (URG) reporting tool. For more information, see the LCA features book in the *VCS / VCS MX Online Documentation*.

IEEE Verilog Std 1364-2005 Encryption

IEEE Std 1364-2005 includes as standard for encrypting and decrypting Verilog and SystemVerilog source files.

In addition, VCS supports the recommendations from the IEEE P1735 working group for encryption interoperability between different encryption and decryption tools, denoted as “version 1” by P1735.

You enable this encryption mode with the `-ipprotect` option on the `vcs` command line and specifying with it a protection header file. This file contains:

- encryption envelope information about VCS and how VCS encrypts a source file. You enter this information with ``pragma protect` expressions as specified in IEEE Std 1364-2005.
- The VCS base64 encoded RSA public key

Note:

The protection header file can contain any of the supported pragmas and also any RSA public key for any vendor not just VCS.

VCS encrypts:

- the source files on the `vcs` command line
- the source files specified in ``include` compiler directives.

VCS decrypts encrypted source files that were encrypted with an RSA `public_key` for VCS.

With this release the `-ipkey` options becomes obsolete.

The syschk.sh and syschk.dat script

Use the `syschk.sh` script to check if a system and environment match the QSC requirements for a given release of a Synopsys product.

For information about using the `syschk.sh` utility, see the *VCS / VCS MX Online Documentation*.

Production License Keys

The license keys are shown in the following table.

Table 1-3 Production License Keys

Feature Name	VCS	VCSi
FusionVantageLmcInterface	X	X
LMCSwift_Net	X	X
VCSPostProcDebugger_Net	X	X
VCSTools_Net	X	X
VCSCompiler_Net	X	-
VCSRuntime_Net	X	-
VCSiCompiler_Net	-	X
VCSiRuntime_Net	-	X
XVCSDebugger	X	X
VT_Visual	X	X
VCSAMSCompiler_Net	X	X
VCSAMSRuntime_Net	X	X
VT_Assertions	X	X
VT_AssertionsRuntime	X	X
VT_Coverage	X	X
VT_CoverageRuntime	X	X

Table 1-3 Production License Keys (continued)

VT_Testbench	X	X
VT_TestbenchRuntime	X	X
SNPS-Assertions	X	X

Supported Platforms and Products

This section describes the platforms, products, and flows supported in this release, in the following sections:

- [“Supported Platforms” on page 67](#)
- [“VCS SystemC Cosimulation Interface” on page 72](#)
- [“Downloading and Installing the VG GNU Package” on page 73](#)
- [“Support for Synopsys SmartModel/FlexModel/Mempro” on page 84](#)
- [“Tested Technologies” on page 85](#)
- [“Third-Party Tools” on page 89](#)

Supported Platforms

[Table 1-4](#) shows the supported platforms, compilers, and linkers.

Table 1-4 Supported Platforms, Compilers, and Linkers

Platform / OS	Compiler Version	Linker Version
Solaris 10 (also known as 5.10 and 2.10) (32- and 64-bit)	Sun Studio 12	Solaris 5.10-1.489
Solaris x86 10 (32- and 64-bit)	Sun Studio 12	Solaris 5.10-1.489
RHEL32 and RHEL64 v4 (32- and 64-bit)	gcc 4.5.2	2.18
SUSE RHEL32 and RHEL64 Enterprise Server 10 (32- and 64-bit)	gcc 4.5.2	2.18
AIX 5.3-TL7-SP2	C/C++ Enterprise Edition v9	Default linker in compiler version

Note:

- For platforms RH6.1 and SuSE11, the gcc 4.5.2 version used for generating binary files should be built on these platforms respectively. Using a gcc 4.5.2 version built on an older platform such as RH4.8 or SuSE10 may lead to unexpected termination and/or linker errors.
- Support for the Solaris x86 platform will be discontinued after the H-2013.06 release.

Purify Version Used

This software release was checked with Purify: Version 7.0.1.0-000.L. You can use this version of Purify when debugging any C code issues.

GDB Path to Use

You can use the GNU Project Debugger (GDB) in the following location in the installation to debug user-developed C++/SystemC code compiled with gcc:

`$VCS_HOME/$ARCH/bin/cbug-gdb[-64]/bin/gdb`

RHEL32 and RHEL64 Platform Support Notes

On RHEL32 and RHEL64 platforms, note the following limitations:

- On RHEL32 and RHEL64 v4, the 32- and 64-bit platforms both support the AMD Opteron and Intel EM64T architectures.
- For RHEL32, use RHEL v5 in binary compatibility mode.
- For RHEL32, use RHEL v6 in binary compatibility mode.

- On SUSE10, the 32- and 64-bit platforms both support the AMD Opteron and Intel EM64T architectures.
- SUSE11 is supported.
- AIX is supported.
- RH 3.0 is not supported.
- Red Hat Fedora is not supported.

Note:

For information about using the correct glibc version for the SWIFT interface on RHEL v4, send an e-mail to support_center@synopsys.com.

Supported and Unsupported Features in RHEL64 64-Bit

Not all features supported on other platforms are available on RHEL64 64-bit. The supported features include the following:

- OpenVera Assertions (OVA)
- Verilog-2001 (V2K)
- SystemVerilog design, assertion, and testbench constructs
- VCD and VPD file dumping
- Coverage metrics (Opteron only)
- NTB (OpenVera)
- VMC
- DVE
- UCLI
- `-comp64`

- -gen_asm
- -gen_c
- -gen_obj
- +rad
- +vpi
- vcdiff
- vcat

Unsupported Features on AIX

The following features are not supported on AIX:

- DVE
- UCLI
- SystemC/Cbug
- AMS
- Master slave mode
- Verification planner
- DPI time-consuming functions
- -gen_obj
- -gen_asm
- Partition compile
- Separate compile

- Multicore
- Low power
- -mode64
- -comp64

Unsupported Features on SolarisX86

The following features are not supported on SolarisX86:

- Cbug
- -comp64

Environment Setup

To run VCS, you must set the environment variables shown in the following examples.

Note:

When you set the `SNPSLMD_LICENSE_FILE` variable, VCSignores any settings for the `LM_LICENSE_FILE` variable.

```
% setenv VCS_HOME <VCS_installation_directory>

% setenv SNPSLMD_LICENSE_FILE <port_number@server>:\
<path_to_license.dat>

% setenv PATH <path_to_gcc-4.2.2>/bin:\
<path_to_gnu/binutils-2.18>/bin:$PATH

% setenv LD_LIBRARY_PATH <path_to_gcc-4.2.2>/lib:\
<path_to_gnu/binutils-2.18>/lib

% set path=($VCS_HOME/bin \
$VCS_HOME/`$VCS_HOME/bin/vcs -platform`/bin/$path)
```

VCS SystemC Cosimulation Interface

[Table 1-6](#) shows the supported platforms for the VCS SystemC cosimulation interface.

Table 1-5 VCS SystemC Cosimulation Supported Platforms

Platform / OS	Compiler Version	Linker Version
Solaris/RHEL64 (solarisX86, x86sol64) (32- and 64-bit)	gcc 4.2.2	2.18 or up
RHEL32 and RHEL64 v4 (32- and 64-bit)	gcc 4.5.2	2.18 or up
RHEL32 and RHEL64 v5 (32- and 64-bit)	gcc 4.5.2	2.21 or up
RHEL32 and RHEL64 v6 (32- and 64-bit)	gcc 4.5.2	2.21 or up
SUSE RHEL32 and RHEL64 Enterprise Server 10 (32- and 64-bit)	gcc 4.5.2	2.18 or up
SUSE RHEL32 and RHEL64 Enterprise Server 11 (32- and 64-bit)	gcc 4.5.2	2.21 or up

Note:

Solaris Sparc (64bit) and Solaris SPARC64 are not supported.

Important:

To use the VCS SystemC cosimulation interface, you must download and install the VG GNU package, as explained in [“Downloading and Installing the VG GNU Package”](#)

Downloading and Installing the VG GNU Package

Several GNU gcc compiler versions are provided in the VG GNU package, together with GNU tools (linker, assembler, etc.). Synopsys STRONGLY recommends using the VG GNU package for SystemC.

There are three different ways to use the VG GNU package:

1. [“Installing and Setting up Outside a VCS Image” on page 73](#)
2. [“Using the VG GNU Package Without Changing PATH \(Not for Solaris/SPARC\)” on page 78](#)
3. [“Installing and Setting Up Inside a VCS Image” on page 79](#)

In most cases, the RHEL32 gcc 4.5.2 used with method (1) is what you need. If so, just download the RHEL32 default tar ball, untar it to a convenient place, and use the `source_me.*` scripts to activate it. If you want to work with shared libraries, see [“Hints for Using Shared Archives” on page 82](#).

Installing and Setting up Outside a VCS Image

Follow these instructions when:

- You do not want to install the VG GNU package inside the VCS image, or
- You want to use the package for all VCS applications, not just for SystemC.

You need to add both `gcc` and `binutils` to the `PATH`. This is easy when you use the `source_me` scripts.

To install the default gcc version, following these steps:

1. Download the package for your platform from EST:

RHEL32

`VG_GNU_PACKAGE_2012.09/linux_gcc4_default.tar.gz`

Solaris/SPARC (sparcOS5, sparc64)

`VG_GNU_PACKAGE_2012.09/solaris_gcc3.tar.gz`

Solaris/RHEL64 (solarisx86, x86sol64)

`VG_GNU_PACKAGE_2012.09/solaris_gcc4.tar.gz`

Using the command-line FTP:

- Start an ftp session to “ftp.synopsys.com”

```
% ftp ftp.synopsys.com
```

- Enter anonymous as the login name.

- Enter your *<email_address>* as the password.

- Type `binary` at the ftp prompt to set the transfer mode to binary:

```
ftp> binary
```

- Type the following command(s) to receive the files:

```
ftp> cd pub
ftp> cd VG_GNU_PACKAGE_2012.09
ftp> get solaris_gcc4.tar.gz
ftp> get solaris_gcc3.tar.gz
ftp> get linux_gcc4_default.tar.gz
ftp> get linux_gcc34.tar.gz
ftp> get linux_gcc42.tar.gz
```

- To logoff the ftp server, type `quit`.

2. Set up your environment:

```
% setenv VCS_HOME <VCS_installation_directory>
% cd $VCS_HOME
% mkdir gnu
% cd gnu
```

Note:

Installing the VG GNU package in a directory named \$VCS_HOME/gnu is just a suggestion. You can install the package anywhere that is convenient.

3. Untar the package:

RHEL32

```
% gtar -zxvf linux_gcc4_default.tar.gz
```

Solaris/SPARC

```
% gtar -zxvf solaris_gcc3.tar.gz
```

Solaris/RHEL64

```
% gtar -zxvf solaris_gcc4.tar.gz
```

4. Set the VG_GNU_PACKAGE environment variable:

sh users

```
% VG_GNU_PACKAGE=<path_to_this_directory>
% export VG_GNU_PACKAGE .
% $VG_GNU_PACKAGE/source_me.sh
```

csh or tcsh users

```
% setenv VG_GNU_PACKAGE <path_to_this_directory>
% source $VG_GNU_PACKAGE/source_me.csh
```

Use the 64-bit version only if:

- You really want to compile code in 64-bit mode (for example, with `vcs -full64`).
- The code is compiled in 32-bit mode but contains pthreads and the CPU itself has an x86_64 architecture.

sh users

```
% VG_GNU_PACKAGE=<path_to_this_directory>
% export VG_GNU_PACKAGE .
% $VG_GNU_PACKAGE/source_me_64.sh
```

csh or tcsh users

```
% setenv VG_GNU_PACKAGE <path_to_this_directory>
% source $VG_GNU_PACKAGE/source_me_64.csh
```

Note: Default GCC version on RHEL64 (4.5.2) comes only in 64-bit shared version. You can use the same one for 32-bit mode static as well.

Installing and Using gcc 3.4.6

The VCS default gcc version on RHEL32 and Solaris/RHEL64 is gcc 4.5.2. The VCS default gcc version on Solaris/SPARC is gcc 3.3.2. For detailed download, installation, and setup instructions, see [“Installing and Setting up Outside a VCS Image” on page 73](#).

On RHEL32 and RHEL64 only, this package also contains a gcc 3.4.6 installation, for both 32-bit and 64-bit. To use one of these compilers, install them first:

1. Download the package for your platform from EST:

RHEL32

```
VG_GNU_PACKAGE_2012.09/linux_gcc34.tar.gz
```

```
% setenv VCS_HOME <VCS_Installation_Directory>
% cd $VCS_HOME
% mkdir gnu
% cd gnu
```

2. Untar the package:

```
% gtar -zxvf linux_gcc34.tar.gz
```

csh or tcsh users

```
% setenv VG_GNU_PACKAGE <path_to_this_directory>
% source $VG_GNU_PACKAGE/source_me_gcc34_32.csh
```

Use the 64-bit version only if:

- You really want to compile code in 64-bit mode (for example, with `vcs -full64`).
- The code is compiled in 32-bit mode but contains pthreads and the CPU itself has an x86_64 architecture.

```
% setenv VG_GNU_PACKAGE <path_to_this_directory>
% source $VG_GNU_PACKAGE/source_me_gcc34_64.csh
```

Installing and Using gcc 4.2.2

The VCS default gcc version on RHEL32 and Solaris/RHEL64 is gcc 4.5.2. The VCS default gcc version on Solaris/SPARC is gcc 3.3.2. For detailed download, installation, and setup instructions, see [“Installing and Setting up Outside a VCS Image” on page 73](#).

On RHEL32 and RHEL64 only, this package also contains a gcc 4.2.2 installation, for both 32-bit and 64-bit. To use one of these compilers, install them first:

1. Download the package for your platform from EST:

RHEL32

VG_GNU_PACKAGE_2012.09/linux_gcc42.tar.gz

```
% setenv VCS_HOME <VCS_Installation_Directory>
% cd $VCS_HOME
% mkdir gnu
% cd gnu
```

2. Untar the package:

```
% gtar -zxvf linux_gcc42.tar.gz
```

csh or tcsh users

```
% setenv VG_GNU_PACKAGE <path_to_this_directory>
% source $VG_GNU_PACKAGE/source_me_gcc42_32.csh
```

Use the 64-bit version only if:

- You really want to compile code in 64-bit mode (for example, with `vcs -full64`).
- The code is compiled in 32-bit mode but contains pthreads and the CPU itself has an x86_64 architecture.

```
% setenv VG_GNU_PACKAGE <path_to_this_directory>
% source $VG_GNU_PACKAGE/source_me_gcc42_64.csh
```

Using the VG GNU Package Without Changing PATH (Not for Solaris/SPARC)

This setup applies to RHEL32 and Solaris/RHEL64 only. It is not available on Solaris/SPARC (sparcOS5, sparc64).

First, download and install the tar file for your platform, as explained in [“Installing and Setting up Outside a VCS Image” on page 73](#).

If you cannot source the `source_me` files, use the `xbin` directory as an alternative. You don't have to set or modify any environment variables (including the `PATH` variable). Instead, you can call `gcc` directly. For example:

```
% <path>/gcc-4.2.2_32/xbin/gcc myfile.c
```

-Or-

```
% syscan -cpp <path>/gcc-4.2.2_32/xbin/g++ model.cpp
```

```
% vcs -sysc top.v \  
  -cpp <path>/gcc-4.2.2_32/xbin/g++ \  
  -gcc <path>/gcc-4.2.2_32/xbin/gcc
```

Notice that `xbin` is used (not `bin`). When you call the compiler via the `xbin` directory, it automatically selects the corresponding `binutils` (that is, assembler, linker ...). You cannot override the location of `binutils`.

Installing and Setting Up Inside a VCS Image

The easiest way to use the VG GNU package for VCS simulations that contain SystemC is to install the package inside a VCS image. Once installed, VCS automatically uses this package for the VCS SystemC flow (for example, when you use the `syscan` command).

Install the image at `$VCS_HOME/gnu`. The directory structure should look like [Figure 1-5](#).

Figure 1-5 \$VCS_HOME/gnu Directory Structure

```
$VCS_HOME/gnu/  
linux/  
    gcc-32  
    gcc-64  
    binutils-32  
    binutils-64  
x86sol/  
    gcc-32  
    gcc-64  
    binutils-32  
    binutils-64  
solaris/  
    gcc-5.9  
    gcc-5.10
```

On RHEL32 and Solaris/RHEL64, VCS uses the static version of gcc 4.2.2 by default. On Solaris/SPARC, VCS uses gcc 3.3.2. You cannot select a different gcc version as the default compiler (for example, 3.4.6 or the shared version of 4.2.2).

1. Download the package for your platform from EST:

RHEL32

```
VG_GNU_PACKAGE_2012.09/linux_gcc4_default.tar.gz
```

Solaris/SPARC

```
VG_GNU_PACKAGE_2012.09/solaris_gcc3.tar.gz
```

Solaris/RHEL64

```
VG_GNU_PACKAGE_2012.09/solaris_gcc4.tar.gz
```

```
% setenv VCS_HOME <VCS_installation_directory>  
% cd $VCS_HOME  
% mkdir gnu  
% cd gnu
```

2. Untar the package:

```
% gtar -zxvf linux_gcc4_default.tar.gz
% gtar -zxvf solaris_gcc3.tar.gz
% gtar -zxvf solaris_gcc4.tar.gz
```

You don't have to set the `VG_GNU_PACKAGE` environment variable or source any setup script. The compiler in the current path doesn't matter. When the VG GNU package is installed this way, VCS uses it automatically for the VCS SystemC flow.

However, if you set the `VG_GNU_PACKAGE` environment variable to any value, this disables automatic selection of gcc from `$VCS_HOME/gnu`. So, before you use this feature, unset that environment variable. For example:

```
% unsetenv VG_GNU_PACKAGE
```

You can override this default with explicit command-line arguments such as `-cc`, `-cpp`, or `ld`. Command-line arguments always take precedence over the default GNU package, and as such, they are the only way to override the default compiler version.

VCS does not use the VG GNU package when you are outside of the VCS SystemC flow. For example, calling:

```
% vcs top.v
```

without the `-sysc` argument does not automatically infer the VG GNU package. Instead, VCS uses the compiler in the current path, as usual.

Hints for Using Shared Archives

The gcc42 compiler in this package is configured to create static executables (not dynamic executables). An executable created by the compiler contains the static library `libstdc++.a`, but does not use the shared (dynamic) `libstdc++.so`. This means that the executable does not rely on `LD_LIBRARY_PATH` to find `libstdc++.so` (because `libstdc++.so` is not used).

There is an alternative gcc4 compiler in this package which is configured to create dynamic executables (instead of static executables). This compiler uses `libstdc++.so` and other shared libraries. You must use this compiler in the following situations:

- When using `syscan -shared` of the SystemC incremental compiler.
- You want to create a shared simulation (`simv.so`). For example, when creating a slaveable simulation.
- When you link a shared third-party library into your simulation that itself needs `libstdc++.so`.

To use the shared gcc42 compiler, use one of these scripts:

```
% source_me_gcc42_shared.sh
% source_me_gcc42_32_shared.csh
% source_me_gcc42_64_shared.sh
% source_me_gcc42_64_shared.csh
```

or invoke directly:

```
% gcc-4.2.2_32-shared
% gcc-4.2.2_64-shared
```

On RHEL32, static and shared compiler installations are also provided for gcc 3.4.6 (besides gcc 4.2.2).

On Solaris/RHEL64, both static and shared compiler installations for gcc 4.2.2 are provided but not for other gcc versions.

Only static compilers are provided on Solaris/SPARC.

Support for 64-Bit Compilation and Simulation

VCS 64-bit compilation and simulation (`-full64`) is available on Solaris 9 (also known as 5.9 and 2.9) and RHEL64 machines.

VCS 64-bit compilation with 32-bit simulation (`-comp64`) is available on Solaris 9 (also known as 5.9 and 2.9) and RHEL64 machines.

On Solaris, 64-bit compilation for both 64 (`-full64`) and 32 (`-comp32`) modes, the simulation requires the following C++ shared library patches:

- 106300-10 or above (106300-10 is included in Solaris 7 Patch Cluster version 08/16/01).
- 106327-09 or above.

Note:

Not all the tools and models that VCS interfaces with can work in 64-bit compilation and simulation or 64-bit compilation and 32-bit simulation.

Support for Synopsys SmartModel/FlexModel/Mempro

Table 1-6 shows the current support for Synopsys SmartModel/FlexModel/MemPro and hardware models.

Table 1-6 Support for SWIFT Models and Hardware Models

Supported Platform	SWIFT Models	Hardware Model Testing Static and Static Deb. Libraries	
		HWM RMS 3.5a	HWM RMS 3.6a
Solaris 32-bit	Yes	Yes	Yes
Solaris 64-bit	Yes	Not supported	Not supported
RHEL32 v4 32-bit	Yes	Not supported	Yes
RHEL64 v4 64-bit	Yes	Not supported	Not supported

SWIFT models are not supported on SUSE RHEL Enterprise Server.

Some SmartModels or FlexModels may be available on RHEL32.
For more information, check with your Synopsys representative.

Note:

For more information on platform support for SWIFT models, FlexModels, and SmartModels, send an e-mail to support_center@synopsys.com.

Tested Technologies

Synopsys has a number of technologies that are developed to work with VCS. [Table 1-7](#) shows the latest versions of those technologies that have been successfully tested with this version of the software.

Table 1-7 Tested Technologies and Versions

Technology	Version
SCL	11.1
HSIM	G-2012.06-SP1
LMC	4.1
Magellan	G-2012.09
MVtools	G-2012.09
NanoSim	G-2012.06-SP1
SWIFT	2.23
TetraMAX	G-2012.06
Vera	D-2009.12
VIP Package VMT	3.90a
VMC	G-2012.09
XA	G-2012.06-SP1

Table 1-8 lists the DesignWare IIP technologies and their versions.

Table 1-8 DesignWare IIP Technologies and Versions

Technology	Version
DW8051_core	3.80a
DWC_ahsata	3.30a
DWC_d20ahb	2.45b
DWC_d20vci	1.94a
DWC_ddr21_mctl	1.50a
DWC_ddr23l_mctl	2.20a
DWC_ddr3_mctl	1.30a
DWC_ddr3_pctl	2.50a
DWC_ddr3_pub	1.30a
DWC_ddr3l_pctl	1.40a
DWC_ddr_umctl	2.40a
DWC_ddr_umctl2	1.00a
DWC_ddr_upctl	2.30a
DWC_dsata	1.33a
DWC_gmac	3.71a
DWC_h20ahb	2.98a
DWC_hdmi_rx	1.30a
DWC_hdmi_tx	1.31a
DWC_hub	2.20a
DWC_mipi_csi2_host	1.03a
DWC_mipi_drf3gm	1.03a
DWC_mipi_drf3gs	1.03a
DWC_mipi_drf4gm	1.04a
DWC_mipi_dsi_host	1.10a

Table 1-8 DesignWare IIP Technologies and Versions (continued)

Technology	Version
DWC_mobile_storage	2.50a
DWC_n2p	1.20a
DWC_otg	3.00a
DWC_pci-x	2.70a
DWC_pcie_dm	4.01a
DWC_pcie_ep	4.01a
DWC_pcie_ep_le	3.90a
DWC_pcie_ep_se	3.90a
DWC_pcie_rc	4.01a
DWC_pcie_sw	4.00a
DWC_usb3	2.20a
DWC_uwb	1.60a
DWC_xgmac	1.20a
DWC_xgxs_pcs	2.11a
DWC_xpcs	3.00a
DW_ahb	2.10b
DW_ahb_dmac	2.17d
DW_ahb_eh2h	1.07e
DW_ahb_h2h	1.06c
DW_ahb_icm	1.13c
DW_ahb_ictl	2.09d
DW_apb	2.02b
DW_apb_gpio	2.09c
DW_apb_i2c	1.17a
DW_apb_i2s	1.06e

Table 1-8 DesignWare IIP Technologies and Versions (continued)

Technology	Version
DW_apb_ictl	2.096
DW_apb_rap	2.04d
DW_apb_rtc	2.03d
DW_apb_ssi	3.22a
DW_apb_timers	2.06c
DW_apb_uart	3.13a
DW_apb_wdt	1.07c
DW_axi	2.13a
DW_axi_a2x	1.00a
DW_axi_gm	1.04b
DW_axi_gs	1.09a
DW_axi_hmx	1.07b
DW_axi_rs	1.02b
DW_axi_x2h	1.06b
DW_axi_x2p	1.07b
DW_axi_x2x	1.04b
DW_hsata	1.91a
DW_memctl	2.79c
DW_rambist	3.00a

[Table 1-9](#) lists the Verification IP technologies and their versions.

Table 1-9 Verification IP Technologies and Versions

Technology	Version
AMBA	6.50a
AMBA_SVT	1.20a
Ethernet	6.21d
HDMI_SVT	1.10a
I2C	2.70a
MIPI_SVT	1.00a
OCP_VRT	1.91a
PCIe	7.58c
SATA	3.21a
SIO	1.53a
USB	8.30b
USB_SVT	2.02a

Third-Party Tools

[Table 1-10](#) shows the latest versions of third-party tools that have been successfully tested with this version of the software.

Table 1-10 Third-Party Tools

Tool	Version
Denali	v3.3.009
Specman	9.2
Verdi	2011_04

Documentation

Documentation is available in both HTML and PDF formats.

To view the documentation in PDF format, see [“Documentation in PDF Format”](#).

To view the documentation in HTML format, enter the following command:

```
% vcs -doc
```

The `vcs -doc` command opens a Web browser and displays the start page of the online documentation.

The HTML online documentation is easily searchable, with search results weighted according to relevance. Navigation features on each page make it easy to move around in the documentation.

Every page of the HTML online documentation system has a “How to Use this Online Documentation” link. Click this link to find tips and pointers on navigating and searching in the HTML online documentation system.

The following sections contain information about Web browser support for the HTML online documentation:

- [“Supported Web Browsers for the Online Documentation” on page 91](#)
- [“Choosing a Web Browser” on page 91](#)

Supported Web Browsers for the Online Documentation

To use this online documentation system, Synopsys recommends using the Web browsers shown in [Table 1-11](#) on the Synopsys-supported platforms.

Table 1-11 Supported Web Browsers for Online Documentation

Platform	Operating System	Supported Browsers
RHEL (Opteron) 64-bit	RHEL64 3 or 4 SUSE RHEL Enterprise Server 10	Firefox 1.5 Mozilla 1.7 Netscape Navigator 7.0
Intel EM64T (Xeon64) 64-bit	RHEL64 3 or 4 SUSE RHEL Enterprise Server 10	Firefox 1.5 Mozilla 1.7 Netscape Navigator 7.0
Itanium 2 (IPF-2) RHEL64 64-bit	RHEL64 2.1	Firefox 1.5 Mozilla 1.4 Netscape Navigator 7.0
SunSPARC Solaris 32- and 64-bit	Solaris 9 or 10	Mozilla 1.7

Choosing a Web Browser

While most modern Web browsers display online documentation satisfactorily, some browsers display search results better than others. For example:

- Internet Explorer (version 6 and later) highlights search strings by default. This makes search results easier to browse.
- On RHEL32 and RHEL64 platforms, Firefox highlights search results when the **Highlight All** Find toolbar option is selected, making search results easier to browse.

The following sections describe the behavior of some popular browsers:

- “Firefox 1.5 on RHEL32” on page 92
- “Mozilla 1.7 on RHEL32” on page 92
- “Netscape 4.8 on RHEL32” on page 93

Firefox 1.5 on RHEL32

You can use **Edit > Find in This Page** to highlight the string you are looking for in the results. This menu sequence opens a Find toolbar at the bottom of the browser window. There, you can select **Highlight All**. So, even though Firefox does not initially highlight the search strings in the HTML pages that display when using the Search function, you can use this additional step to highlight search strings.

Once you use **Edit > Find in this Page**, the Find toolbar remains at the bottom of the browser window. To see where the search string occurs in the current browser window, click the **Highlight All** button in the Find toolbar. Firefox remembers the search string until you replace the search string with a new one.

Mozilla 1.7 on RHEL32

Mozilla also has an **Edit > Find in This Page** selection available from its menu. However, with this browser you get a dialog box instead of a persistent toolbar at the bottom of the browser window. There is no Highlight All feature. You have to go through the search hits serially in an HTML page.

Netscape 4.8 on RHEL32

Netscape also has an **Edit > Find in This Page** selection available from its menu. However, with this browser you get a dialog box instead of a persistent toolbar at the bottom of the browser window. There is no Highlight All feature. You have to go through the search hits serially in an HTML page.

Documentation in PDF Format

To view the documentation in PDF format, use the Acrobat Reader to open the `$VCS_HOME/doc/pdf/navigator.pdf` file.

The navigator.pdf file contains hypertext links to all the PDF files in the VCS documentation set.

Important:

VCS supports version 8.0 of Acrobat on all supported platforms.

Limitations

This section contains information on known limitations and workarounds (if available) for the current release of the software, organized in the following subsections:

- [“Cap on Number of Command-line Characters” on page 94](#)
- [“Limitations on VCD and VPD Files” on page 95](#)
- [“Coverage Metrics Limitations” on page 95](#)
- [“OVA Limitations” on page 96](#)
- [“SystemC Cosimulation Interface Limitation” on page 97](#)
- [“Limitations on Variable-Sized Arrays” on page 97](#)
- [“OpenOffice Not Supported by Verification Planner Doc Annotator” on page 98](#)
- [“Unified Profiler Limitations” on page 98](#)
- [“Multicore ALP and DLP Current Limitations” on page 99](#)

Cap on Number of Command-line Characters

There is a known limit to the number of characters that can appear on a `vcs` command line. This limit varies from platform to platform, but all platforms have a limit. To work around this limitation, use these commands with the `-f` or `-F` options. These options specify a file that can contain compile-time options and source file names. If you find that you reach this limit, consider using these options.

Limitations on VCD and VPD Files

VCD files (specified with the `$dumpvars` system task) and VPD files (specified with the `$vcdpluson` system task) have the following limitations:

- VCS cannot record value changes for variables declared within `automatic` tasks and functions.
- VCS cannot record Native Testbench code behavior in these files.

Coverage Metrics Limitations

VCS does not monitor some of the language constructs in the IEEE Std 1364-1995 and 1364-2001 for condition or FSM coverage.

Memories

VCS does not monitor for condition coverage subexpressions that contain memory elements. VCS also does not extract FSMs where the current and next state of the FSM are in memory elements.

Multidimensional Arrays

Just like memory elements, VCS does not monitor for condition coverage or extract for FSM coverage multidimensional array elements.

OVA Limitations

The following limitations apply to OVA:

- Declaring an event in one clock domain and using it in another clock domain (through matched) is okay. But both clock domains using each others' events (cross dependency through matched) is not supported.
- The `past` operator does not store history information when `$ova_stop` is in effect. Therefore, when `$ova_start` is issued, there may be some initial failures in assertions using the operator before the history information is re-established as time advances. (This is similar to the situation at time 0 where no past history is yet established.)

One way to work around this problem is to use the `-ova_filter_past` compile-time option. It removes the initial failures before the history information is established.

- The following Verilog 2001 operators do not currently work with OVA:
 - arithmetic shift `<<<`, `>>>`
 - exponentiation `**`
- You cannot enter OVA compile-time options such as `-ova_cov` or `-ova_debug` or files containing the OVA language in a file specified with the `-f` compile-time option.

SystemC Cosimulation Interface Limitation

The VCS/SystemC cosimulation interface has the following limitations:

- The option `-e<name>` is used for specifying an alternate entry function for the slave-API. The slave-API cannot be used for designs containing SystemC. However, when the `-sysc=newsync` option is used for elaboration, then the slave-API can be used, and the `-e<name>` option can be used as well.
- The UCLI save/restore command is not supported for designs containing SystemC models.

To see the full list of SystemC-specific limitations, see the *VCS / VCS MX Online Documentation*.

Limitations on Variable-Sized Arrays

The following features are not supported:

- `{default: value}` for associative arrays
- Identifiers imported from packages, in associative arrays
- Only the following types of associative array indexes are supported:
 - integer literal
 - string literal
 - identifier

An associative array declared with a non-constant RHS initializer may be initialized before expressions referred to in the RHS are initialized.

Assignments such as “aa = '{0:aa[1], 1:aa[0]}’” are executed as if they were separate assignments. This can have unexpected results.

OpenOffice Not Supported by Verification Planner Doc Annotator

The Verification Planner Word Doc Annotator does not currently support OpenOffice XML input. Only MS Word 2003 or new versions are supported. This applies only to the Word flow; the spreadsheet flow supports both MS Excel and OpenOffice.

Unified Profiler Limitations

The unified profiler is an LCA feature. The following technologies are not supported in the unified profiler:

- Multicore — Both for Application Level Parallelism (ALP) and Design Level Parallelism (DLP is an LCA feature).
- The behavior is unpredictable if you fork child processes or threads in your C code which might be called through PLI/DPI/DirectC interfaces.
- Incremental compilation — Not supported yet for the unified profiler.

- OpenVera is not officially supported. VCS provides some information for reference, but the names of the programs and constructs might be a bit different from the original ones.
- Code coverage is not supported yet; the time and memory used by code coverage would be counted to corresponding HDL code.
- Only black box information for SystemVerilog constraints and assertions and code and functional coverage (no breakdown of information for instances, module, constructs, and so forth).

Multicore ALP and DLP Current Limitations

The limitations of Multicore ALP and DLP are:

- SystemVerilog constructs, such as, but not limited to, classes, covergroups, associative and dynamic arrays, strings and events have the following limitations in Multicore DLP:
 - Their use throughout the design can prevent autopartitioning from generating a valid autopart.cfg file.
 - Their use is prohibited in cores that are slave partitions.

SystemVerilog covergroups, however, are supported by Multicore DLP but only if you also run Multicore ALP with a core (or a consumer) for multicore functional coverage (PFC) for these covergroups.

The SystemVerilog data types `logic` and `bit` are supported throughout the design with Multicore ALP and DLP.

- Multicore ALP Parallel SAIF is not supported with VCS Multicore DLP.

- The unified profiler (`-simprofile`), an LCA feature, is not supported in Multicore ALP or DLP.
- SystemC-on-top is not supported for VCS Multicore DLP.
- The UCLI `dump` command with `-add` and `-ports` is not supported for Multicore ALP parallel VPD file dumping.
- Multicore ALP parallel VPD file dumping does not support the `$msglog` or `$tblog` system tasks.
- The dynamic race detection tool (`-race` and `-racecd`) is not supported with Multicore DLP.
- Saving and restarting the simulation with checkpoint files (`$save` system task) is not supported with Multicore DLP.
- For SystemC cosimulation the following are not supported in Multicore DLP:
 - partial build with shared libraries
 - partial build with object files
 - SystemC and C/C++ cosimulation
- VMC is not supported with Multicore DLP.
- Analog mixed-signal simulation is not supported with Multicore DLP.

Known Issues

This section explains known issues with the current release of the software.

Known Issues in Verilog

This section lists the known Verilog issues in this release.

Using Threaded Libraries on Solaris

Using threaded libraries on Solaris is not yet supported, but this release includes the `-Xstrict=0x01` compile-time option that enables you to use them. The `-Xstrict=0x01` option prevents VCS MX from using the same resources as the threaded library. Use the `-Xstrict=0x01` option with the `-syslib` option for specifying a threaded library. For example:

```
% vcs -Xstrict=0x01 -syslib "-lpthread" ...
```

Incremental Binding

STAR STS0158858 — Incremental port binding does not work properly in cycle mode.

Using an External Memory Manager

The internal memory manager in VCS MX relies on this documented feature of malloc:

```
void *malloc(size_t size);
```

The `malloc()` and `free()` functions provide a simple general-purpose memory allocation package. The `malloc()` function returns a pointer to a block of at least *size* bytes suitably aligned for any use.

Certain restrictive memory managers may choose to enforce a fixed memory alignment. This can lead to unaligned accesses in some situations, which may produce bus errors on Solaris-based systems.

If you want to use an external memory manager, Synopsys recommends thoroughly reviewing the memory manager's documentation for issues related to the alignment of memory returned.

Displaying For Loop Variable Values in DVE and UCLI

SystemVerilog enables you to declare the `for` loop control variable within the `for` loop. This creates a local variable within the loop. In some cases, DVE and the UCLI cannot display the value of a control variable declared within a `for` loop statement, though it can be printed using the `$display` system task. The workaround is to move the declaration outside the `for` loop.

For example:

```
initial begin
  for (int j=0;j<5;j++) begin
    $display("j=%0d\n",j);
  end
end
```

Workaround code:

```
initial begin
    int j;
    for (j=0;j<5;j++) begin
        $display("j=%0d\n",j);
    end
end
```

SystemC Cosimulation with UCLI Common C++ Library Error

The following error message is displayed only when the g++ compiler cannot find the correct `libstdc++` library, which is shipped with the compiler and can be found in the same directory as that of your compiler.

```
libvcsnew.so: undefined reference to
`_Unwind_Resume_or_Rethrow@GCC_3.3'
```

This is typically caused by loading the wrong `libstdc++` libraries. When you see the above error message, you must set your `LD_LIBRARY_PATH` to */<path to the compiler>/lib*.

For information on the default compiler for this release, see [“VCS SystemC Cosimulation Interface” on page 72](#).

Obsolete Switches/Options

The following switches or options are obsolete from this release onwards. There are two categories:

- [“Error — Obsolete Option Used” on page 104](#)
- [“Warning — Unknown or Obsolete Option Used” on page 105](#)

Error — Obsolete Option Used

Using these options causes the simulator to issue an error message and exit:

- `+dmprof`
- `+oldsdf`
- `-commonsdf`
- `-profile`
- `-uniprofile`
- `-vhdlelab`
- `-cm_olddb`
- `-ntb_opts covg_compat`
- `+2state`

Note:

Older Vera semantics (pre Vera-2005.12) for auto-bins and cross-bins are not supported. Therefore, you should not use the `-ntb_opts covg_compt` option from the VCS command line. You should convert older style cross-bins to the newer cross-bins syntax. For more information, see the section on “Functional Coverage” in the *OpenVera User Guide*.

Warning — Unknown or Obsolete Option Used

Using these options causes the simulator to issue a warning message and ignore the option, but processing continues:

- `-assert disable_file=<file_name>`
- `-noerrorIOPCWM`
- `-smart_analysis`
- `-oldassocarray`
- `-newassocarray`
- `-Xoldassocarray`
- `-Xnewassocarray`
- `-ntc_alt_solver`
- `+ntccopt`
- `+pulse_port_e/`
- `+pulse_port_r/`
- `+transport_port_delays`
- `-xdelay_large`

- -nonunielab
- -mhdlnewsync
- -disable_mdb_vlogan_internal
- -ctopt
- -ctverilog
- -ctpli
- -ctdirectc
- -ctvhdl
- -ctvhdl93
- -ctvcd
- +cmod
- +cmodext
- +cmoddefine
- +cmodincdir
- -auto4protect
- +defererrors
- -no_swift_bus
- -ptime
- +vcs+nocawm
- -dut_tb
- -ignore_uselib_lib

- +purecoverun
- +purifyrun

Obsolete System Tasks

The following system tasks are obsolete starting with this release. From this release onwards, only the new system task will work.

- \$tblog (to be replaced by \$vcdplustblog)
- \$tblogon (to be replaced by \$vcdplustblogon)
- \$tblogoff (to be replaced by \$vcdplustblogoff)
- \$msglog (to be replaced by \$vcdplusmsglog)
- \$msglogon (to be replaced by \$vcdplusmsglogon)
- \$msglogoff (to be replaced by \$vcdplusmsglogoff)

For more information, see the *VCS / VCS MX Online Documentation*.

Note:

The HTML-based product documentation no longer includes links to PDF versions. To view the PDF versions of the same documentation content, use Acrobat to bring up the navigator.pdf document located at \$VCS_HOME/docs/UserGuide/pdf/navigator.pdf.

Obsolete `-assert disable_file=<file_name>` Option

The `-assert disable_file=<file_name>` option is obsolete in this release. Use of this option causes VCS to issue a warning message. The same functionality can be achieved by making use of `-assert hier=<file_name>` option to both disable and enable assertions.

PDF File Name Changes

In this release, the following PDF files were renamed:

- `sva_quickref_cg.pdf` is now named `sva_cg_quickref.pdf`
- `vcs_quickstart.pdf` is now named `vcs_tb_qs.pdf`

For this version, soft links are in place for backward compatibility:

- `sva_quickref_cg.pdf` links to `sva_cg_quickref.pdf`
- `vcs_quickstart.pdf` links to `vcs_tb_qs.pdf`

In future versions, these links will be removed.

Bug Reporting Procedure

We are interested in hearing your feedback related to the VCS G-2012.09 Release. Please report all comments and issues related to this release through an e-mail to:

vcs_support@synopsys.com.

Your e-mail will be acknowledged by automatic reply and assigned a case number. In general, we need to be able to reproduce the problem in order to fix it, so a simple model demonstrating the error is the most effective way for us to identify the bug. If that is not possible, then please provide a detailed explanation of the problem.

You can call for support at 1-800-VERILOG (1-800-837-4564). Press 1 for VCS Technical Support. Support hours are Monday to Friday 8:30 a.m. – 6:00 p.m. Pacific time.

List of Issues Resolved in VCS and VCS MX

version G-2012.09

9000433357	9000496432	9000506924	9000508101	9000510464
9000513104	9000515648	9000530310	9000531659	9000533544
9000534317	9000534827	9000537161	9000543467	9000547034
9000548935	9000549113	9000549466	9000549552	9000549855
9000550312	9000551111	9000551320	9000551486	9000552432
9000552652	9000552925	9000553030	9000553209	9000553319
9000553350	9000553359	9000553394	9000553397	9000553408
9000553409	9000553413	9000553414	9000553416	9000553417
9000553418	9000553434	9000553624	9000553903	9000554089
9000554815	9000555306	9000555376	9000555377	9000555787

9000555903	9000555920	9000556080	9000556120	9000556372
9000556578	9000556646	9000556698	9000557158	9000557171
9000557541	9000557740	9000558001	9000558006	9000558022
9000558127	9000558152	9000558553	9000558909	9000559110
9000561472	9000561583	9000562373	9000562990	9000563230
9000564226				