

# VCS 学习总结

Author:lu.hongbo

Date:2012-7-23

## 目录

1	VCS工作流程 .....	3
1.1	编译 .....	3
1.2	仿真 .....	6
1.2.1	交互模式 .....	6
1.2.2	batch模式 .....	6
2	编译设计 .....	6
2.1	编译的关键特性 .....	7
2.2	+plusarg_save选项 .....	7
2.3	重定义parameter参数值 .....	8
2.4	条件表达式中的X/Z的检查 .....	8
2.5	+v2k编译选项的使用 .....	8
2.6	VCS V2K的配置和库影射 .....	10
3	仿真 .....	10
3.1	DVE仿真 .....	10
3.2	UCLI仿真 .....	12
4	SDF文件的反标 .....	13
4.1	采用统一的SDF特性 .....	13
4.2	采用\$sdf_annotate 系统任务 .....	13
5	覆盖率 .....	14
5.1	功能覆盖率 .....	14
6	SystemC与verilog联合仿真 .....	14
6.1	应用模型 .....	15
6.2	systemc中的时序精度的控制 .....	15
6.2.1	自动调节时间精度 .....	15
6.2.2	调整verilog/VHDL中的时间精度 .....	15
7	采用VCS图形界面仿真 .....	15
7.1	简单的在服务器上打开vcs的方法 .....	16
	附录A .....	17

本学习总结是在学习 VCS 的用户参考手册基础上完成的，有出入的地方请参见 VCS 参考手册。

# 1 VCS 工作流程

在对一个设计进行仿真的过程中，VCS 主要是通过

- 编译
- 仿真

两个步骤来实现的。

## 1.1 编译

编译其实就是 VCS 根据用户的输入文件（通常是 HDL）编译产生可执行文件（默认是二进制的 simv）。产生的这个二进制文件在之后的仿真中会被执行。

对于 VCS 的编译模式也分为：优化形式的编译和调试模式的编译。但是 Synopsys 公司建议用户先采用完全调试模式/部分调试模式，等到设计达到了一定的正确性以后再切换到优化模式。

优化形式的编译其实就是在 VCS 的 batch 模式下进行工作，在这种情况下用户是通过 VCS 的命令来控制 VCS 实现设计的编译。在这种情况下 VCS 的运行时间会比较快。

调试模式也叫交互模式。这种模式下的 VCS 运行时间和运行性能与 batch 模式比起来要差很多。但是这种模式能够满足你在设计之初/非常需要具有调试功能情况下的需求。其实调试模式就是利用 VCS 的图形界面，通过查看设计的层次化/设计的原理图/仿真之后的波形文件，解决仿真当中遇到的错误。

VCS 的使用

VCS 的语法形式如下：

```
%vcs [compile_options] verilog_files
```

编译阶段常用的编译选项如下：

**-h/-help**

这个选项主要是列出 vcs 的使用技巧，包括一些编译选项的意义等

**-licqueue**

这个选项主要是在系统中的 VCS license 被占用的情况下，通知 VCS 等待 license 空闲的时候再执行，不要直接退出来。

**-v filename**

**指定一个 verilog 的库文件。**VCS 会在这个文件当中查找你的源代码中不存在/没有定义的 module 和 UDP。（其实就类似于各个 Foundry 厂提供的.v 类型仿真文件，它的所有 module 都罗列在一个.v

文件中，这个时候如果你在进行后方针，就只需要将这个库文件通过-v 的形式读入，当然如果你通过读入普通的.v 文件一样读入库文件也是可以的，我这里只不过是举一个例子。)

-y directory

这个命令其实和上一个-v 的选项差不多。只不过这个选项是告诉 VCS 一个 verilog 库的存放路径，当 VCS 在编译的时候发现你提供的源文件中有些例化 module 和 UDP 找不到原形的情况下，它会在这个路径给你查找匹配的 module 和 UDP。VCS 查找的方式是先在这个路径下查找与例化名相同的文件，然后再查看这个文件里面的 module 和 UDP 的名字，如果匹配就直接将这个文件 instance 到你的源文件当中。

注意：如果你的库文件在多个路径当中都有相同的名字，那么 VCS 会采纳第一个路径下的库。

+incdir+directory+

这个选项所指定的路径是告诉 VCS 给文件中使用了`include 语句的文件指定查找路径。如果有不止一个文件路径的时候可以使用"+"继续在上述选项后面累加。注意是在你的设计当中用到了`include 语句的时候。例如：

```
// Test file
`include "parameter.v"
module test (...);
...
endmodule
`include "unparameter.v"
```

+libext+extension+

这个选项是告诉 VCS 在指定的 lib 路径下查找上述选项指定扩展名的文件。当然了你也可以指定多个扩展名，比如：+libext+.V+.v 等等，指定的扩展名的顺序就是 VCS 的查找顺序。这个功能是应用原文件的加载，比如：你不指定原文件的名字，只是通过这个所谓的后缀进行加载。

-full64

这个选项是为 64 位的操作系统准备的，如果你的操作系统是 32 位的就不要操这个心了。

-file filename

指定一个包含了文件列表的文件，并且在这个文件中也可以包含编译的选项。另外-file 可以覆盖掉-f 里面的编译选项。

-f filename

说这个选项和-file 一样未免还是有些太牵强，但是绝大部分内容还是相同的。-f 选项的特性：

-可以使用 verilog 的注释命令符屏蔽掉文件内容，//或/\*\*/

-可以使用 file 中的选项指定到另一个 file

下面是-f 的一些限制：

-可以指定所有+开头的编译选项，但是决不能制定这个选项+memopt

-只能指定如下-开头的编译选项：

-f -gen\_asm -gen\_obj -y -l -u -v

-f 和-file 选项的共同限制:

- -full64 和-comp64 选项不支持在这些文件中。必须要直接通过 vcs 的命令行来指定这两个选项。
- 不能指定换码符
- 不能使用通用字符, 但是可以使用\*和\$

-gui

调用 VCS 的图形化界面。好像很多 EDA 工具都是这样来调用图形界面。

-R

这个指令是告诉 VCS 在编译完成以后直接运行可执行文件, 如果你在编译的时候没有带上这个选项, 编译完成以后 VCS 就退出了, 但是你会发现在相应的目录下产生了以可执行文件, 默认叫 simv。下文会告诉你怎么样更改这个可执行文件的名字。

-pvalue+parameter\_hierarchical\_name=value

这个选项是用来改变源文件中 parameter 参数值的命令。

例如: -pvalue+udut.uchip.ucore.IDLE =0

-parameters filename

通过一个文件的形式改变源文件中很多 parameter 参数的值。这个 file 中的语法结构如下:

assign value path\_to\_parameter

文件的路径要使用"/"作为分隔符。

例如: assign 33 test/d1/param1

-notice

使能更详细的诊断信息。

-q

退出模式; 抑制了例如 VCS 使用的 C 编译器/VCS 分析源文件、top 层、制定的 timescale 的信息。

-V

Verbose 模式; 打印 VCS 在运行的过程中执行的 C compiler, 汇编器和链接器的命令。

-l filename

制定一个 VCS 产生的 log 文件名, 如果你键入了-R 选项, 那么在编译和仿真的时候都会将 log 内容打印到 log 文件中。

+define+macro=value+

给你的源代码中的文本宏定义一个值/一个字符串。你可以在你的 verilog 代码中使用 ifdef 编译进行这个定义的测试。如果你想要定义的字符串中存在空格, 那么你必须要用""把字符串括起来。

例如:

vcs design.v +define+USELIB="dir=dir1 dir=dir2"

## 1.2 仿真

在编译的过程中 VCS 会产生一个二进制的可执行文件，默认叫 simv。根据你的编译选项结果，可以通过：

- 交互模式
- batch 模式

两种模式进行仿真。

### 1.2.1 交互模式

如果你以交互模式编译你的设计，你才能够用 VCS 的交互模式仿真。

那么怎么样才能够实现交互编译呢？那你就**必须要在编译的过程中加入 -debug/debug\_all 选项**。

### 1.2.2 batch 模式

编译采用的 batch 模式，仿真就会使用 batch 模式。

下面的命令就是直接使用 batch 进行仿真的：

```
%simv
```

可执行文件的产生：

**在默认的情况下产生的可执行文件是 simv。但是你也可以在编译的时候使用 -o 选项来制定生成的可执行文件得名字。**

例如：vcs design.v -o design.simv

## 2 编译设计

在 debug 模式下编译/评估设计

**调试模式也叫交互模式**，一般是用在：

- 设计的初始阶段，这个时候需要用调试工具例如：**DVE/UCLI 进行调试**
- 采用了 PLI 的时候
- 采用 UCLI 命令强制了一个信号，将值写入到了 register/net 中。

VCS 在编译的时候有如下选项可以实现调试模式：

`-debug_pp,-debug` 和 `-debug_all`

下面的例子指示如何使用全调试和部分调试的功能

```
%vcs -debug [compile_options] top.v
```

```
%vcs -debug all [compile_options] top.v
```

## 2.1 编译的关键特性

对 verilog 中的 memory 和寄存器进行初始化

可以采用如下 option 对设计中的所有 memory 和 register bits 进行初始化：

`+vcs+initreg+random`

在 0 时刻将设计中的所有状态变量（reg 类型），寄存器和 MDA 中的 memory 初始化成随机的 0/1。

注意：+vcs+initreg+只对 verilog 的设计起作用。

`+vcs+initreg` 初始化正常的 memory 和多维的 reg 类型数组。例如：

```
reg [7:0] mem [7:0] [15:0];
```

+vcs+initreg 不会初始化 register 变量和 reg 类型以外的多维数组。

## 2.2 +plusarg\_save 选项

`+plusarg_save` 与 `+plusarg_ignore` 是相互对立的两个选项，一个是用来使能编译参数，一个是用来非使能编译参数。编译参数要在+plusarg\_save/+plusarg\_ignore 之后，例如：

```
+plusarg_save +test_case0
```

在你的 RTL 中通过+test\_case0 参数来开启某段 event。例如：

```
if ($test$plusargs("+test_case0")) begin
```

```
...
```

```
end
```

其中+test\_case0 是从编译参数阶段获得的许可。

当然你也可以通过编译过程为一个变量指定一个值，然后在执行阶段将这个值传递给这个变量，这个时候只需要将参数值+在编译列表中，然后在 RTL 中通过\$value\$plusargs，调用这个参数语法如下：

```
integer = $value$plusargs("plusarg_format",signalname);
```

实际使用情况如下：

```
module valueplusargs;
```

```
reg [31:0] r1;
```

```
integer status;
```

```
initial
```

```
begin
$monitor("r1=%0d at %0t",r1,$time);
#1 r1=0;
#1 status=$value$plusargs("r1=%d",r1);
end
endmodule
```

运行阶段执行下面的命令：

```
%simv +r1=10
```

\$monitor 系统任务的输出：

```
r1=x at 0
```

```
r1=0 at 1
```

```
r1=10 at 2
```

## 2.3 重定义 parameter 参数值

在编译的过程中可以使用下述两个选项进行 parameter 类型数据的重定义：

- -pvalue
- -parameter

-pvalue 可以通过下面的语法重定义 parameter 类型参数值：

```
vcs -pvalue+hierarchical_name_of_parameter=value
```

-parameter 可以通过下面的语法调用一个文件（重定义了很多 parameter 类型的参数），**重定义一组 parameter 参数**。

```
vcs -parameter filename
```

调用的文件内的内容格式如下：

```
assign value path_to_the_parameter
```

## 2.4 条件表达式中的 X/Z 的检查

如果你想让 VCS 检查到你的 RTL 代码中**条件表达式的值在出现 X/Z 的情况下报警**。可以使用 **-xzcheck** 选项。

```
warning 'signal_name' within scope hier_name in file_name.v: line_number to x/z at time simulation_time
```

## 2.5 +v2k 编译选项的使用

下表列出了 Std 1364-2001 的结构实现，以及这些结构中是否需要+v2k 参数的使用。我们知道 verilog 的发展到现在最新的标准是 2001，所以为了**兼容 2001 标准的 verilog 语言**，VCS 特设了这个选项。



Std 1364-2001 结构	Require +v2k
逗号分割敏感列表中的变量： always @(r1,r2,r3)	yes
基于名称的参数传递： modname #(.param_name(value)) inst_name(sig1,...);	yes
ANSI 标准端口和参数列表： module dev (output reg [7:0] out1,input wire [7:0] w1);	yes
在寄存器生命阶段进行初始化： reg [15:0] r2 = 0;	yes
条件编译控制： `ifdef ...`elseif	yes
除能默认的线类型： `default_nettype	yes
有符号算法扩展： reg signed [7:0] r1;	no
输出文件任务： \$fopen \$fscan \$scanf...	no
从 runtime 命令行传递参数： \$value\$plusarg 系统功能	yes
部分索引功能： reg [8+:5] = 5'b11111;	yes
多维数组 reg [7:0] r1 [3:0] [3:0];	yes
持续文件名和行号： `line	yes
暗指敏感列表： always @*	yes
幂操作： r1=r2**r3;	yes
属性： (* optimize_power=1 *) module dev (res,out,clk,data1,data2);	yes
generate 段	yes
局部参数声名	yes
自动任务和函数： task automatic t1 ();	需要-sverilog 编译选项
常量函数： localparam lp1 = const_func(p1);	yes
带有位范围的参数： parameter bit [7:0] [31:0] P = {32'd1 , 32'd2 , 2'd3 , 32'd4 , 32'd5 , 32'd6 , 32'd7 ,32'd8};	编译选项中需要加入-sverilog 选项

## 2.6 VCS V2K 的配置和库影射

都知道-v, -y 可以指定库/库的存放路径。但是你也可以通过使用一个文件将多个库/库路径包含到这个文件中, 进行一次性指定。对于 sv 来说 Interface 类型的模块也可以被当作库的一个因素来存在。

### 库影射文件

库影射文件的格式如下:

```
library lib1 /net/design1/design1_1/*.v;
```

```
library lib2 /net/design1/design1_2/*.v;
```

VCS 在调用库的时候就是按照你在影射文件中的定义顺序来调用实际 module 的。

编译过程中可以通过-libmap 来引用可“库影射文件”。

例如: vcs +v2k -libmap lib.lst

vcs -sverilog -libmap lib.lst

上面虽然调用了“库映射文件”, 但是在库中的 verilog 模型中可能还使用了“include”语句调用了别的 module 文件, 所以在你的“库映射文件”中可以使用-incdir 指令来指定查询路径。

例如:

```
library gatelib /net/design1/gatelib/*.v -incdir /net/design1/spec1lib, /net/design1/spec2lib;
```

## 3 仿真

采用的手段如下

- DVE 仿真
- UCLI 仿真
- 关键运行特性

### 3.1 DVE 仿真

下面的命令是用来调用 DVE 工具进行仿真的:

```
% simv -gui
```

调用了 DVE 的图形界面, 但是前提是在编译的时候一定要采用-debug\*\*类型的选项进行编译。

采用如下命令来调用 DVE 实现后处理模式:

```
%dve -vpd[VPD/EVCD_FILENAME]
```

tip

vcd 类型的文件其实就是纪录了信号的翻转信息。你可以通过\$dumpvars 函数打印出 vcd 文件。

vpd 其实是 vcd 的一个升级版, 所以 vpd 的全称是 vcdplus dumping。你可以在你的 verilog 中

调用\$vcpluson 来打印出 VPD 文件。

EVCD 是扩展的 VCD 文件，它只是打印出你设计当中的端口信息。

#### VPD 文件的打印

可以用\$vcpluson 和\$vcplusoff 来开启/关闭 vpd 文件的打印功能。默认的输出文件名称是“vcdplus.vpd”但是你也可以使用\$vcplusfile 来重新定义输出文件名。例如：

```
$vcpluson (level|“LVL=integer”,scope*,signal*);
```

```
level|“LVL=integer”
```

指定打印信号的层次，默认是“0”，打印所有层次的信号。

“LVL=integer”是用来指定打印层次的数量。

#### scope

指定范围（在这个范围中记录了信号值的改变）名称（默认是所有的改变）

#### signal

指定需要打印的信号名（默认打印所有的信号）。

例如：

```
module test();
```

```
...
```

```
initial
```

```
begin
```

```
    $vcpluson;
```

```
    ...
```

```
end
```

```
...
```

```
endmodule
```

\$vcplusoff 是用来关闭波形打印功能的：例如：

```
module test();
```

```
...
```

```
initial
```

```
begin
```

```
    $vcpluson;
```

```
    #5 $vcplusoff;
```

```
    ...
```

```
end
```

```
...
```

```
endmodule
```

\$vcplusfile 用来重定义输出文件名，例如：

```
module test();
```

```
...
```

```
initial
```

```

begin
    $vcdpluson;
    $vcdplusfile("my.vcd");
    ...
end
...
endmodule

```

### 打印多维数组和 memorys

可以使用\$vcplusmemon 和\$vcplusoff 两个参数打印 memory 类型的信息。但是一定要在编译的时候就加上+memcbk 和+v2k 选项。

第二种方法是直接使用，\$vcplusmemorydump 系统函数。

语法结构:

```
system_task( Mda [, dim1Lsb [, dim1Rsb [, dim2Lsb [, dim2Rsb [, ... dimNLSb [, dimNRSb]]]]]] );
```

例如:

```

module tb();
    ...
    reg [3:0] addr1L, addr1R, addr2L, addr2R, addr3L, addr3R;
    reg [7:0] mem01 [1:3] [4:6] [7:9]
    ...
endmodule

module test();
    ...
    initial
        $vcplusmemon( mem01 );
        // Records all elements of mem01 to the VPD file.
    ...
endmodule

```

## 3.2 UCLI 仿真

UCLI仿真其实是建立在TCL语言之上的。所以在UCLI结构中你可以使用TCL命令来控制UCLI的执行。下面的命令就是建立UCLI仿真:

```
% simv [simv_options] -ucli
```

注意：使用上述命令需要在编译的时候使用-debug\_all,debug,debug\_pp选项进行编译哦！

在运行UCLI的时候可以通过命令的形式来控制UCLI的执行。如下:

```

% simv -ucli
ucli % source file.cmds

```

or

```
% simv -ucli -do file.cmds
```

## 4 SDF 文件的反标

首先 SDF 文件里面包含了一些器件的固有延迟，内部连线的延迟，端口延迟，时序确认信息，时序约束信息和脉宽控制信息等内容。

VCS 读取 SDF 文件实际上就是延迟信息的一个反标过程。VCS 通过读取 SDF 文件里面的延迟值，从而改变原文件的默认延迟值（通常是由原文件默认指定，如果原文件没有指定那么就是采用仿真工具默认指定的延迟值）。

### 4.1 采用统一的 SDF 特性

统一的 SDF 特性就是允许你通过编译的时候采用下面的选项来反标 SDF 延迟：

```
-sdf min|type|max:instance_name:file.sdf
```

例如：

```
% vcs -sdf min|type|max:instance_name:file.sdf [compile_options]
% simv
```

### 4.2 采用\$sdf\_annotate 系统任务

可以在你的 verilog 原文件中通过引用\$sdf\_annotate 系统任务来反标延迟信息。

语法结构如下：

```
$sdf_annotate ("sdf_file"[, module_instance] [, "sdf_configfile"] [, "sdf_logfile"] [, "mtm_spec"]
[, "scale_factors"] [, "scale_type"]);
```

"sdf\_file"

指定 sdf 文件及其存放路径。

module\_instance

指定 sdf 文件的反标范围。

"sdf\_configfile"

指定 sdf 配置文件。

"sdf\_logfile"

指定 Sdf 的 log 生成文件，这个文件会保存 vcs 在运行过程中产生的 error 信息和 warning 信息。默认情况下 vcs 不会超过 10 个以上的 error 信息/warning 信息记录到 log 文件中。为了避免这

个限制，在运行的过程中可以通过+sdfverbose 选项来开启 vcs 记录所有信息的功能。

"mtm\_spec"

指定 Vcs 反标哪一类延迟信息。可以使用 "MINIMUM,TYPICAL,MAXIMUM,TOOL\_CONTROL (default)"。

"scale\_factors"

指定 minimum,typical 和 maximum 延迟的因子。通过冒号分割成 3 个字符串，默认值是 "1.0:1.0:1.0"

"scale\_type"

指定在缩放比例之前使用的 SDF 文件里面延迟值。可能的值有 "FROM\_TYPICAL","FROM\_MIMINUM","FROM\_MAXIMUM","FROM\_MTM"(默认值)

如果你在你的 verilog 原文件中添加了 \$sdf\_annotate 系统函数，在你编译和运行的时候就不需要添加什么额外的选项了。

## 5 覆盖率

这一部分内容主要是验证工程师需要关注的。方便在测试之后统计仿真的完成情况。

覆盖率共分为：

- 行覆盖率
- 反转覆盖率
- 条件覆盖率
- 分支覆盖率
- FSM 覆盖率

### 5.1 功能覆盖率

功能覆盖率是在 VCS 支持 NTB 和 Systemverilog 功能覆盖率模型的基础上产生的。为了测试功能覆盖率，你必须要在你的设计中提取功能观测点，以便测试。

覆盖率的选项

-cm line|cond|fsm|tgl|branch|assert

assert:编译 sv 的断言覆盖率

## 6 SystemC 与 verilog 联合仿真

verilog 设计当中包含有 verilog 模型和 systemc 模块。

为了实现联合仿真，那么你必须要给 systemc 模块产生一个 verilog 的 wrapper。

## 6.1 应用模型

verilog 的测试平台，带有 systemc 和 verilog 的例化仿真步骤：

a、产生 wrapper

b、编译

c、仿真

wrapper 产生

```
% syscan [options] file1.cpp:sc_module_name
```

编译

```
% vcs -sysc [compile_options] file1.v file2.v
```

仿真

```
% simv [run_options]
```

## 6.2 systemc 中的时序精度的控制

在 systemc 运行的内核中可以使用 `sc_set_time_resolution()` 和 `sc_set_default_time_unit()` 函数来控制时间精度。默认的设置是 10ns/10ps。

### 6.2.1 自动调节时间精度

如果 systemc 与 verilog 中的实践精度不同，可以通过 `-sysc=adjust_timers` 来自动调节。这个参数会使用两者中分辨率最高的时间精度。

### 6.2.2 调整 verilog/VHDL 中的时间精度

可以在 vcs 中使用 `-timescale=1ns/1ps` 来强制 HDL 内核与 systemc 内核的默认时间精度值相同。

## 7 采用 VCS 图形界面仿真

采用 VCS 的图形界面进行仿真需要注意以下几点：

- 在编译的时候一定要采用 `-debug**` 选项
- 如果是想编译之后立即调出 gui 界面一定要在编译阶段使用 `-R -gui` 选项（关于这两个选项可以参考 `vcs -help` 的解释）

- 使用 vcs 的目录中严禁用“ ”空格符，这样可能会导致打不开 vcs 的图形界面/打开的图形界面没有完全加载仿真文件

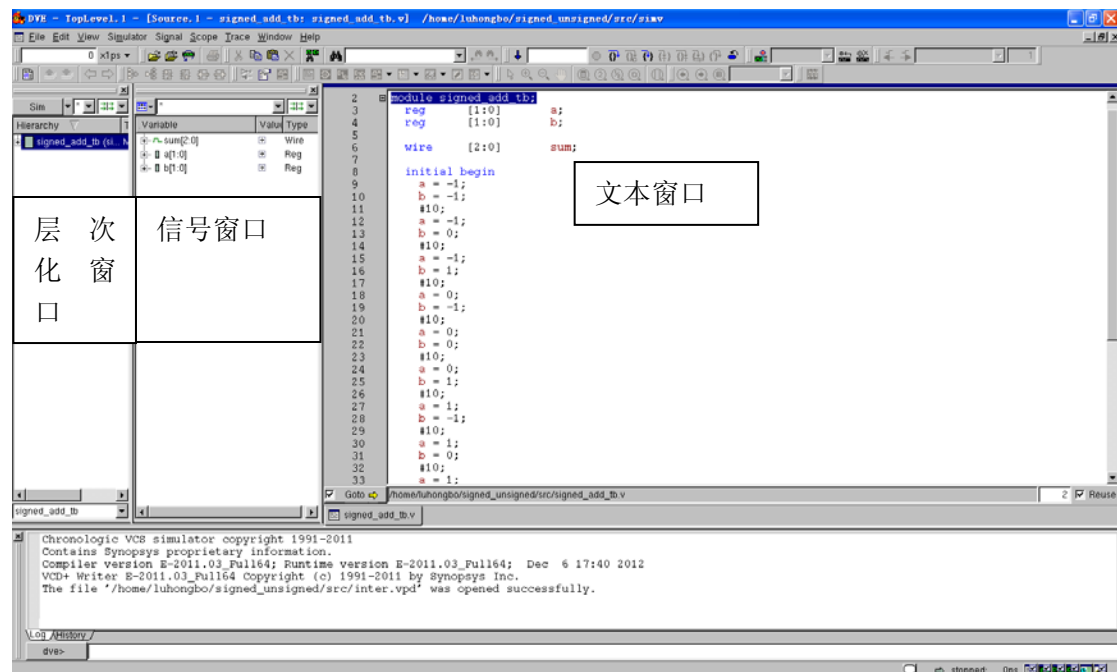
## 7.1 简单的在服务器上打开 vcs 的方法

我们通过使用 signed\_add.v signed\_add\_tb.v 两个 verilog 文件来讲解 vcs 图形界面的使用。  
键入如下命令：




```
vcs -v signed_add.v signed_add_tb.v -full64 -debug_all -R -gui
```

其中-full64、-debug\_all、-R、-gui 选项所发挥的作用可以在附录 A 中查到。

上述命令键入以后会打开 vcs 的图形界面如下：



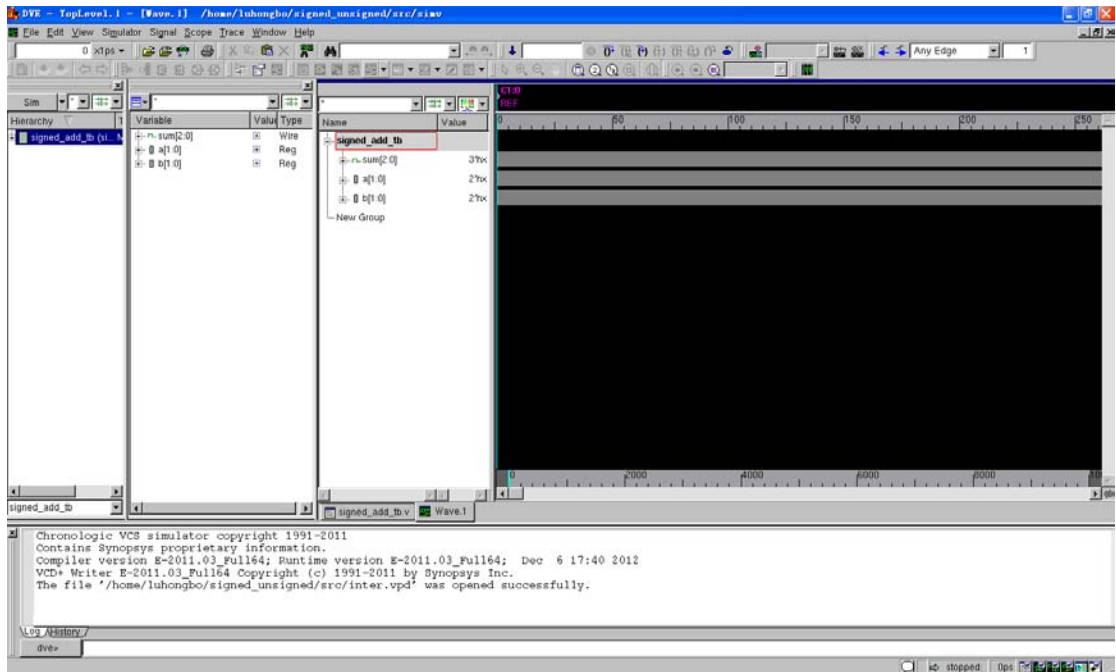
在主界面中可以看到 vcs 会显示出设计的：层次化窗口、信号窗口和文本窗口

选中层次化窗口中的对应层次：工具栏中的    这些工具会自动以可操作的形式显示。

上述的第二个图标是原理图的形式显示设计

第五个图标是显示仿真波形界面点击后如下：



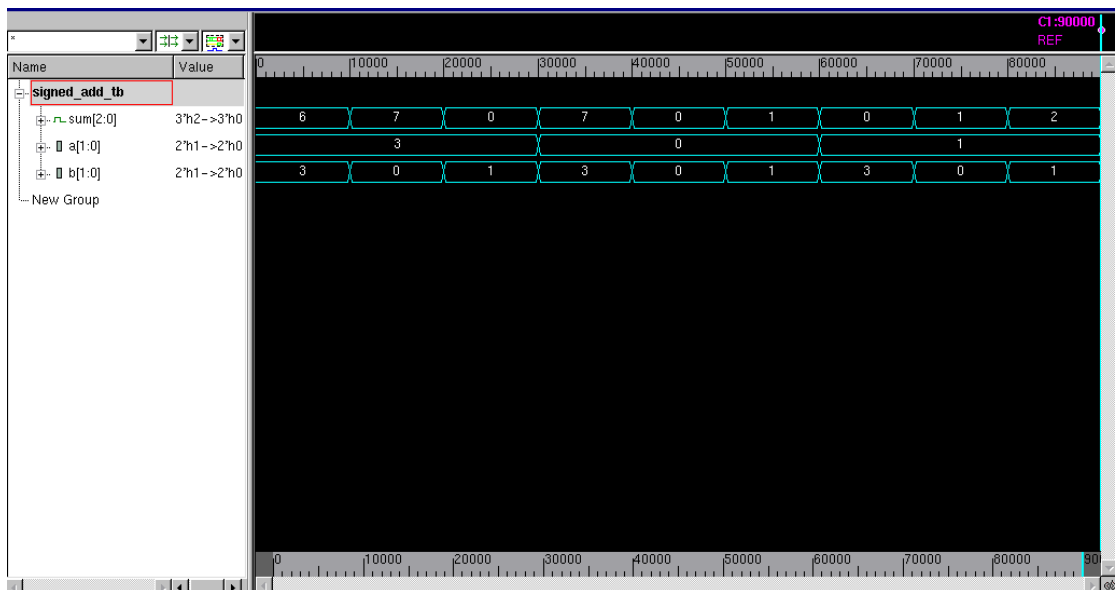


右侧的黑色界面是波形显示窗口，通过点击



这些按钮可以实现连续运行，单步运行等

操作，对应的波形会显示如下：



其余功能，阅者自行开发

## 附录 A

vcs script version : E-2011.03

machine name = autumn

machine type = linux

machine os = Linux 2.6.9-89.ELlargesmp

The FLEXlm host ID of this machine is "00505685644a"

Compiler version = VCS-MX E-2011.03

VCS Build Date = Feb 23 2011 20:58:21

Please Note:

Not all the compile-time and runtime options described in this file work in VCS Basic. For a list of options that do not work in VCS Basic, consult the VCS\_Basic\_Intro.pdf file.

#### Compile-Time Options

\*\*\*\*\*

编译阶段

-ams

Enables the use of Verilog-AMS code in VCS 2-step mode.

-ams\_discipline <discipline\_name>

Specifies the default discrete discipline in VerilogAMS in VCS 2-step mode.

-ams\_iereport

Provides the auto-inserted connect modules (AICMs) information in VCS 2-step mode.

-as <assembler>

Specifies an alternative assembler. Only applicable in incremental compile mode, which is the default. Not supported on IBM RS/6000 AIX.

-ASFLAGS <options>

Passes options to assembler. Not supported on IBM RS/6000 AIX.

-assert <keyword\_argument>

The keyword arguments and what they do are as follows:

disable\_cover

Disables coverage for SVA cover statements.

dumpoff

Disables the dumping of SVA information in the VPD file.

dve

Enables SystemVerilog assertions tracing in the VPD file that you load into DVE. This tracing enables you to see assertion attempts.

enable\_diag

Enables further control of SystemVerilog assertions result reporting with runtime options.

filter\_past

Ignores SystemVerilog assertion subsequences containing past operators that have not yet eclipsed the history threshold.

vpiSeqBeginTime

Enables you to see the simulation time that a SystemVerilog assertion sequence starts when using Debussy.

vpiSeqFail

Enables you to see the simulation time that a SystemVerilog assertion sequence doesn't match when using Debussy.

-C

Stops after generating the intermediate C or assembly code.

-cc <compiler>

Specifies an alternative C compiler.

-CC <options>

Works the same as -CFLAGS.

-CFLAGS <options>

Pass options to C compiler. Multiple -CFLAGS are allowed. Allows passing of C compiler optimization levels.

-cm line|cond|fsm|tgl|path|branch|assert

Specifies compiling for the specified type or types of coverage.

The arguments specify the types of coverage:

line     Compile for line or statement coverage.

cond     Compile for condition coverage.

fsm      Compile for FSM coverage.

tgl      Compile for toggle coverage.

path     Compile for path coverage.

branch   Compile for branch coverage

assert   Compile for SystemVerilog assertion coverage.

If you want VCS to compile for more than one type of coverage, use the plus (+) character as a delimiter between arguments, for example:

-cm line+cond+fsm+tgl

-cm\_assert\_hier <filename>

Limits SystemVerilog assertions coverage to the module instances

listed in the specified file.

#### `-cm_cond <arguments>`

Modifies condition coverage as specified by the argument or arguments:

basic	Only logical conditions and no multiple conditions.
std	The default: only logical, multiple, sensitized conditions.
full	Logical and non-logical, multiple conditions, no sensitized conditions.
allops	Logical and non-logical conditions.
event	Signals in event controls in the sensitivity list position are conditions.
anywidth	Enables conditions that need more than 32 bits.
for	Enables conditions if for loops.
tf	Enables conditions in user-defined tasks and functions.
sop	Condition SOP coverage instead of sensitized conditions. also tells VCS that when it reads conditional expressions that contain the ^ bitwise XOR and ~^ bitwise XNOR operators, it reduces the expression to negation and logical AND or OR.

You can specify more than one argument. If you do use the + plus delimiter between arguments, for example:

`-cm_cond basic+allops`

#### `-cm_constfile <filename>`

Specifies a file listing signals and 0 or 1 values. VCS compiles for line and condition coverage as if these signals were permanently at the specified values and you included the `-cm_noconst` option.

#### `-cm_count`

Enables cmView to do the following:

In toggle coverage, not just whether a signal toggled from 0 to 1 and 1 to 0, but also the number of times it so toggled.

In FSM coverage, not just whether an FSM reached a state, had such a transition, but also the number of times it did.

In condition coverage, not just whether a condition was met or not, but also the number of times the condition was met.

In Line Coverage, not just whether a line was executed, but how many times.

#### `-cm_dir <directory_path_name>`

Specifies an alternative name and location for the coverage database directory.

-cm\_fsmcfg <filename>

Specifies an FSM coverage configuration file.

-cm\_fsmopt <keyword\_argument>

The keyword arguments are as follows:

allowTemp

Allows FSM extraction when there is indirect assignment to the variable that holds the current state.

optimist

Specifies identifying illegal transitions when VCS extracts FSMs in FSM coverage. cmView then reports illegal transitions in report files.

report2StateFsms

By default VCS does not extract two state FSMs. This keyword tells VCS to extract them.

reportvalues

Specifies reporting the value transitions of the reg that holds the current state of a One Hot or Hot Bit FSM where there are parameters for the bit numbers of the signals that hold the current and next state.

reportWait

Enables VCS to monitor transitions when the signal holding the current state is assigned the same state value.

reportXassign

Enables the extraction of FSMs in which a state contains the X (unknown) value.

-cm\_fsmresetfilter <filename>

Filters out transitions in assignment statements controlled by if statements where the conditional expression (following the keyword if) is a signal you specify in the file.

-cm\_hier <filename>

When compiling for line, condition, FSM or toggle coverage, specifies a configuration file that specifies module definitions, source files, or module instances and their subhierarchies, that you want VCS to exclude from coverage or be the only parts of the design compiled for coverage.

-cm\_ignorepragmas

Tells VCS to ignore pragmas for coverage metrics.

-cm\_libs yv|celldefine

Specifies compiling for coverage source files in Verilog libraries when you include the yv argument. Specifies compiling for coverage module definitions that are under the `celldefine compiler directive when you include the celldefine argument. You can specify both arguments using the plus (+) delimiter.

-cm\_line contassign

Enables line coverage for continuous assignments.

-cm\_name <filename>

As a compile-time or runtime option, specifies the name of the intermediate data files.

-cm\_noconst

Tells VCS not to monitor for conditions that can never be met or lines that can never execute because a signal is permanently at a 1 or 0 value.

-cm\_pp [gui][batch]

Tells VCS to start cmView, it starts cmView in batch mode to write reports by default.

gui

Specifies starting the cmView graphical user interface instead of writing reports.

batch

Specifies starting cmView in batch mode to write reports. This keyword argument is not necessary because batch mode is the default condition.

You enter cmView command line options to the right of this option and its argument.

-cm\_resetfilter

You can filter out of FSM coverage transitions in assignments controlled by if statements where the conditional expression (following the if keyword) is a signal you specify in the file. This filtering out can be on the specified signal

in any module or the module you specify in the file. You can also specify the FSM and whether the signal is true or false.

**-cm\_tglfile <filename>**

Specifies displaying at runtime a total toggle count for one or more subhierarchies specified by the top-level module instance entered in the file.

**-cm\_tgl mda**

Enables toggle coverage for Verilog 2001 and SystemVerilog unpacked multidimensional arrays.

**-cpp**

Specifies a C++ compiler.

**-comp64**

Compiles the design in 64 bit mode and creates a 32 bit executable for simulating in 32 bit mode.

**-debug**

Enables the use of UCLI commands and DVE.

**-debug\_all**

Enables the use of UCLI and DVE. Also enables line stepping.

**-doc**

Starts browser to display the HTML files for the VCS/VCSi documentation.

**-dve\_opt <dve\_option>**

You can use the argument called -dve\_opt to pass DVE arguments from simv to DVE. Each DVE argument must be preceded by -dve\_opt argument. In cases where the argument requires an additional option, the = sign needs to be used. (E.g. -dve\_opt -session=file.tcl)

**-e <new\_name\_for\_main>**

Specifies the name of your main() routine in your PLI application.

**-f <filename>**

Specifies a file that contains a list of pathnames to source files and compile-time options.

**-F <filename>**

Same as the -f option but allows you to specify a path to the file and the source files listed in the file do not have to be absolute

pathnames.

**-file filename**

This option is for problems you might encounter with entries in files specified with the -f or -F options. This file can contain more compile-time options and different kinds of files. It can contain options for controlling compilation and PLI options and object files. You can also use escape characters and meta-characters in this file, like \$, `, and ! and they will expand, for example:

```
-CFLAGS '-I$VCS_HOME/include'  
/my/pli/code/$PROJECT/treewalker.o  
-P /my/pli/code/$PROJECT/treewalker.tab
```

You can comment out entries in this file with the Verilog // and /\* \*/ comment characters.

**-full64**

Compiles the design in 64 bit mode and creates a 64 bit executable for simulating in 64 bit mode.

**-gen\_asm**

Specifies generating intermediate assembly code. Not supported on IBM RS/6000 AIX.

**-gen\_c**

Specifies generating intermediate C code. This is the default in IBM RS/6000 AIX.

**-gen\_obj**

Generate object code; default on Linux, Solaris, and HP platforms. Not supported on IBM RS/6000 AIX.

**-h or -help**

Lists descriptions of the most commonly used compile-time and runtime options.

**-ID**

Displays the hostid or dongle ID for your machine.

**-ignore <keyword\_argument>**

The keyword arguments are as follows:



unique\_checks

Suppresses warning messages about SystemVerilog unique if and unique case statements.

priority\_checks

Suppresses warning messages about SystemVerilog priority if and priority case statements.

all

Suppresses warning messages about SystemVerilog unique if, unique case, priority if and priority case statements.

-j<number\_of\_processes>

Specifies the number of processes to use for parallel compilation.  
There is no space between the j character and the number.

-l <filename>

(lower case L) Specifies a log file where VCS records compilation messages and runtime messages if you include the -R, -RI, or -RIG options.

-ld <linker>

Specifies an alternative linker.

-LDFLAGS <options>

Pass options to the linker.  
Only applicable in incremental compile mode.

-line

Enables stepping through the code and source line breakpoints in DVE.

-lmc-swift

Enables the LMC SWIFT interface.

-lmc-swift-template <swift\_model\_name>

Generates a Verilog template for a SWIFT Model.

-l<name>

Links the <name> library to the resulting executable.

-load <shared\_VPI\_library>:<registration\_routine>

Specifies the registration routine in a shared library for a VPI application.

**-Marchive=<number\_of\_module\_definitions>**

Tells the linker to create temporary object files that contain the specified number of module definitions. Use this option if there is a command line buffer overflow caused by too many object files on the linker command line.

**-Mdelete**

Use this option for the rare occurrence when the `chmod -x simv` command in the make file can't change the permissions on an old `simv` executable. This option replaces this command with the `rm -f simv` command in the make file.

**-mhdl**

Enables VCS MX - Mixed HDL simulation.

**-Mlib=<directory>**

Specifies the directory where VCS looks for descriptor information to see if a module needs to be recompiled. Also specifies a central place for object files. You use this option for shared incremental compilation.

**-Mmakeprogram=<program>**

Program used to make object (default is `make`).

**-Mupdate[=0]**

By default VCS overwrites the Makefile between compilations. If you wish to preserve the Makefile between compilations, enter this option with the 0 argument.

Entering this argument without the 0 argument, specifies the the default condition, incremental compilation and updating the Makefile.

**-negdelay**

Enables the use of negative values in IOPATH and INTERCONNECT entries in SDF files.

**-noIncrComp**

Disables incremental compilation.

**-notice**

Enables verbose diagnostic messages.

**-ntb**

Enables the use of the OpenVera Testbench language constructs described

in the OpenVera Language Reference Manual: Native TestBench.

**-ntb\_cmp**

Compiles and generates the testbench shell (file.vshell) and shared object files. Use this option when compiling the .vr file separately from the design files.

**-ntb\_define <macro>**

Specifies any OpenVera macro name on the command line. You can specify multiple macro names using the + delimiter.

**-ntb\_filext <.ext>**

Specifies an OpenVera file extension. You can specify multiple filename extensions using the + delimiter.

**-ntb\_incdir <directory\_path>**

Specifies the include directory path for OpenVera files. You can specify multiple include directories using the + delimiter.

**-ntb\_noshell**

Tells VCS not to generate the shell file. Use this option when you recompile a testbench.

**-ntb\_opts <keyword\_argument>**

The keyword arguments are as follows:

**check**

Reports error, during compilation or simulation, when there is an out-of-bound or illegal array access.

**dep\_check**

Enables dependency analysis and incremental compilation. Detects files with circular dependencies and issues an error message when VCS cannot determine which file to compile first.

**no\_file\_by\_file\_pp**

By default, VCS does file by file preprocessing on each input file, feeding the concatenated result to the parser. This argument disables this behavior.

**print\_deps[=<filename>]**

Enter this argument with the dep\_check argument. This argument tells VCS to display the dependencies for the source files on the screen or in the file that you specify.

tb\_timescale=<value>

Specifies an overriding timescale for the testbench. The timescale is in the Verilog format (for example, 10ns/10ns).

use\_sigprop

Enables the signal property access functions. (for example, vera\_get\_ifc\_name()).

vera\_portname

Specifies the following:

The Vera shell module name is named vera\_shell.

The interface ports are named ifc\_signal.

Bind signals are named, for example, as: \if\_signal[3:0].

You can enter more than one keyword argument, using the + delimiter, for example:

-ntb\_opts use\_sigprop+vera\_portname

-ntb\_shell\_only

Generates only a .vshell file. Use this option when compiling a testbench separately from the design file.

-ntb\_sfname <filename>

Specifies the filename of the testbench shell.

-ntb\_sname <module\_name>

Specifies the name and directory where VCS writes the testbench shell module.

-ntb\_spath

Specifies the directory where VCS writes the testbench shell and shared object files. The default is the compilation directory.

-ntb\_vipext <.ext>

Specifies an OpenVera encrypted-mode file extension to mark files for processing in OpenVera encrypted IP mode. Unlike the -ntb\_filext option, the default encrypted-mode extensions .vrp, .vrhp are not overridden, and will always be in effect. You can pass multiple file extensions at the same time using the + delimiter.

-ntb\_vl

Specifies the compilation of all Verilog files, including the design,

the testbench shell file and the top-level Verilog module.

-o <name>

Specifies the name of the executable file that is the product of compilation. The default name is simv.

-ovac

Starts the OVA compiler to check the syntax of OVA files on the vcs command line.

-ova\_cov

Enables functional coverage.

-ova\_cov\_events

Enables functional coverage reporting of expressions.

-ova\_cov\_hier <filename>

Limits functional coverage to the module instances listed in the specified file.

-ova\_debug or -ova\_debug\_vpd

Enables OVA attempt dumping into VPD.

-ova\_file <filename>

Specifies an OpenVera Assertions file. Not required if the filename has an .ova extension.

-ova\_filter\_past

For assertions that are defined with the past operator, ignore these assertions where the past history buffer is empty. For instance, at the very beginning of the simulation the past history buffer is empty. So, a check/forbid at the first sampling point and subsequent sampling points should be ignored until the past buffer has been filled with respect to the sampling point.

-ova\_filter\_last

Ignores assertion subsequences containing past operators that have not yet eclipsed the history threshold.

-ova\_enable\_diag

Enables runtime options for controlling functional coverage reports.

-ova\_inline

Enables compiling OVA code that is written in Verilog source files.

-ova\_lint

Enables general rules for the OVA linter

-ova\_lint\_magellan

Enables Magellan rules for the OVA linter.

-override-cflags

Tells VCS not to pass its default options to the C compiler.

-override\_timescale=<time\_unit>/<time\_precision>

Overrides the time unit and precision unit of all the ``timescale` compiler directives in the source code and, like the `-timescale` option, provides a timescale for all module definitions that precede the first ``timescale` compiler directive.

-P <pli.tab>

Specifies a PLI table file.

-parameters <filename>

Changes parameters specified in the file to values specified in the file. The syntax for a line in the file is as follows:

assign <value> <path\_to\_parameter>

The path to the parameter is similar to a hierarchical name except that you use slash characters (/) instead of periods as delimiters.

-platform

Returns the name of the platform directory in your VCS installation directory.

-pvalue+<parameter\_hierarchical\_name>=<value>

Changes the specified parameter to the specified value.

-q

Suppresses VCS compiler messages.

-R

Run the executable file immediately after VCS links together the executable file. You can add any runtime option to the `vcs` command line.

-S

Stop simulation just as it begins. Use this option with the -R and +cli options.

-sim\_res=<time\_precision>

Defines simulation resolution. It also defines timescales for modules which don't have timescales after analysis.

-sv\_pragma

Tells VCS to compile the SystemVerilog assertions code that follows the sv\_pragma keyword in a single line or multi-line comment.

-sysc

Tells VCS to look in the ./csrc directory for the subdirectories containing the wrapper and interface files needed by the VCS/SystemC cosimulation interface to connect the Verilog and SystemC parts of a mixed Verilog and SystemC design.

-syslib <libs>

Specifies system libraries to be linked with the runtime executable.

-timescale=<time\_unit>/<time\_precision>

If only some source files contain the `timescale compiler directive and the ones that don't appear first on the vcs command line, use this option to specify the time scale for these source files.

-u

Changes all characters in identifiers to uppercase.

-ucli

Specifies UCLI mode at runtime.

-use\_vpiobj

Used to specify the vpi\_user.c file that enables you to use the vpi\_register\_systf VPI access routine.

-V

Enables the verbose mode.

-v <filename>

Specifies a Verilog library file to search for module definitions.

-vera

Specifies the standard VERA PLI table file and object library.

-vera\_dbind

Specifies the VERA PLI table file and object library for dynamic binding.

-vhdlclab "<command\_line\_options>"

Specifies elaboration and compilation options for scs, the VHDL compiler.

-Vt

Enables warning messages and displays the time used by each command.

-y <directory\_pathname>

Specifies a Verilog library directory to search for module definitions.

+acc+1|2|3|4

Old style method to enable PLI ACC capabilities for the entire design.

1 enables all capabilities but breakpoints and delay annotation.

2 enables what 1 enables plus breakpoints on value changes of nets and registers.

3 enables what 2 enables plus module path delay annotation.

4 enables what 3 enables plus gate delay annotation.

+ad=<partition\_filename>

Specifies the partition files used in mixed signal simulation.

+allmtm

Allows you to specify at runtime which values in min:typ:max delay value triplets in compiled SDF files using the +mindelays, +maxdelays, or +typdelays runtime options.

+applylearn[+<filename>]

Compiles your design to enable only the ACC capabilities that you needed for the debugging operations you did during a previous simulation of the design.

The +vcs+learn+pli runtime option records where you used ACC capabilities in a file named pli\_learn.tab. If you do not change the file's name or location, you can omit +<filename> from this option.

+autoprotect[<file\_suffix>]

Creates a protected source file; all modules are encrypted.

+auto2protect[<file\_suffix>]



Create a protected source file that does not encrypt the port connection list in the module header; all modules are encrypted.

`+auto3protect[<file_suffix>]`

Creates a protected source file that does not encrypt the port connection list in the module header or any parameter declarations that precede the first port declaration; all modules are encrypted.

`+bidir+1`

Tells VCS to finish compilation when it finds a bidirectional registered mixed-signal net.

`+charge_decay`

Enables charge decay in trireg nets. Charge decay will not work if you connect the trireg to a transistor (bidirectional pass) switch such as `tran`, `rtran`, `tranif1`, or `rtranif0`.

`+cli+[<module_name>=]1|2|3|4`

Enable CLI debugging.

1 enables you to see the values of nets and registers and deposit values to registers.

2 also enables breakpoints on value changes of nets and registers.

3 also enables you to force a value on nets.

4 also enables you to force a value on a register.

You can specify a module to enable CLI debugging only for instances of the module.

`+cliedit`

Enables you to use the UNIX GNU command line editing interface for entering CLI commands. See the VCS/VCSi User Guide for download and setup information.

`+csdf+precompile`

Precompiles your SDF file into a format that is for VCS to parse when it is compiling your Verilog code.

`+csdf+precomp+dir+<directory>`

Specifies the directory path where you want VCS to write the precompiled SDF file.

`+csdf+precomp+ext+<ext>`

Specifies an alternative to the `"_c"` character string addition to the filename extension of the precompiled SDF file.

**+define+<macro\_name>=<value>**

Defines a text macro. Test for this definition in your Verilog source code using the ``ifdef` compiler directive.

**+delay\_mode\_distributed**

Specifies ignoring the module path delays and use only the delay specifications on all gates, switches, and continuous assignments.

**+delay\_mode\_path**

For modules with specify blocks, specifies ignoring the delay specifications on all gates and switches and use only the module path delays and the delay specifications on continuous assignments.

**+delay\_mode\_unit**

Specifies ignoring the module path delays and change all the delay specifications on all gates, switches, and continuous assignments to the shortest time precision argument of all the ``timescale` compiler directives in the source code.

**+delay\_mode\_zero**

Change all the delay specifications on all gates, switches, and continuous assignments to zero and change all module path delays to zero.

**+deleteprotected**

Allows overwriting of existing files when doing source protection.

**+error+<n>**

Enables you to increase the maximum number of NTB errors at compile-time to <n>.

**+incdir+<directory>**

Specifies the directories that contain the files you specified with the ``include` compiler directive. You can specify more than one directory, separating each path name with the + character.

**+libext+<extension>**

Specifies that VCS only search the source files in a Verilog library directory with the specified extension. You can specify more than one extension, separating each extension with the + character.

For example, `+libext+.v` specifies searches library files with no extension and library files with the `.v` extension.

Enter this option when you enter the `-y` option.

#### +liborder

Specifies searching for module definitions in the libraries that follow, on the vcs command line, a library that contains an unresolved instance before searching the libraries that precede the library with the unresolved instance.

#### +librescan

Specifies always starting the search for unresolved module definitions with the first library specified on the vcs command line.

#### +libverbose

Tells VCS to display a message when it finds a module definition in a source file in a Verilog library directory that resolves a module instantiation statement that VCS read in your source files, a library file, or in another file in a library directory.

#### +lint=[no]ID|none|all,...

Enables or disables Lint messages about your Verilog code.

#### +maxdelays

Use maximum value when min:typ:max values are encountered in delay specifications SDF files.

#### +memcbk

Enables callbacks for memories and multidimensional arrays (MDAs). Use this option if your design has memories or MDAs and you are doing any of the following:

- o Writing a VCD or VPD file during simulation. For VCD files, at runtime, you must also enter the +vcs+dumparrays runtime option. For VPD files you must enter the \$vcdplusmemon system task. VCD and VPD files are used for post-processing with DVE or debugging using SmartDebug.
- o Using the VCS/SystemC Interface
- o Interactive debugging with DVE
- o Writing an FSDB file for Debussy
- o Using any debugging interface application - VCSD/PLI (acc/pli) that needs to use value change callbacks on memories or MDAs. APIs like acc\_add\_callback, vcsd\_add\_callback, and vpi\_register\_cb need this option if these APIs are used on memories or MDAs.

**+memopt**

Applies optimizations to reduce memory. For more information see the VCS/VCSi User Guide.

**+mindelays**

Use minimum value when min:typ:max values are encountered in delay specifications and SDF files.

**+multisource\_int\_delays**

Enables multisource interconnect delays.

**+nbaopt**

Removes the intra-assignment delays from all the nonblocking assignment statements in your design.

**+neg\_tchk**

Enables negative values in timing checks.

**+nocelldefinepli+0|1|2**

For specifying what VCS records in the VPD file about nets and registers defined under the ``celldefine` compiler directive.

0 enables recording the transition times and values of nets and registers in all modules defined under the ``celldefine` compiler directive or defined in a library that you specify with the `-v` or `-y` compile-time options.

1 disables recording the transition times and values of nets and registers in all modules defined under the ``celldefine` compiler directive.

2 disables recording the transition times and values of nets and registers in all modules defined under the ``celldefine` compiler directive or defined in a library that you specify with the `-v` or `-y` compile-time options whether the modules in these libraries are defined under the ``celldefine` compiler directive or not.

**+noerrorIOPCWM**

Changes the error condition, when a signal is wider or narrower than the inout port to which it is connected, to a warning condition, thus allowing VCS to create the `simv` executable after displaying the warning message.

**+nolibcell**

Specifies not defining modules in libraries as cells unless they are under the ``celldefine` compiler directive.

+nospecify

Suppresses module path delays and timing checks in specify blocks.

+notimingcheck

Suppresses timing checks in specify blocks.

+nowarnTFMPC

Suppress the "Too few module port connections" warning messages during Verilog Compilation.

+no\_notifier

Disables the toggling of the notifier register that you specify in some timing check system tasks.

+no\_tchk\_msg

Disables the display of timing check warning messages but does not disable the toggling of notifier registers in timing checks. This is also a runtime option.

+optconfigfile+<filename>

Specifies the VCS configuration file.

+overlap

Enables accurate simulation of multiple non-overlapping violation windows for the same signals specified with negative delay values in timing checks.

See the section on "Using Multiple Non-Overlapping Windows" in the VCS/VCSi User Guide.

+pathpulse

Enables the search for the PATHPULSE\$ specparam in specify blocks.

+pli\_unprotected

Enables PLI and CLI access to the modules in the protected source file being created (PLI and CLI access is normally disabled for protected modules).

+plusarg\_save

Enter this option in the file that you specify with the -f option so that VCS passes to the simv executable the options beginning with a plus + character that follow in the file.

+plusarg\_ignore

Also enter this option in the file that you specify with the -f option so that VCS does not pass to the simv executable the options that follow in the file. Use this option with the +plusarg\_save option to specify that other options should not be passed.

+print+bidir+warn

Tells VCS to display a list of bidirectional registered mixed-signal nets.

+prof

Specifies that VCS writes the vcs.prof file during simulation that tells you the module definitions, module instances, and Verilog constructs in your design that use the most CPU time.

+protect[<file\_suffix>]

Creates a protected source file; only encrypting `protect/`endprotect regions.

+pulse\_e/<number>

Specifies flagging as error and drive X for any path pulse whose width is less than or equal to the percentage of the module path delay specified by the number argument.

+pulse\_int\_e/<number>

Same as the +pulse\_e option but only applies to interconnect delays.

+pulse\_int\_r/<number>

Same as the +pulse\_r option but only applies to interconnect delays.

+pulse\_on\_event

Specifies that when VCS encounters a pulse shorter than the module path delay, VCS waits until the module path delay elapses and then drives an X value on the module output port and displays an error message.

+pulse\_on\_detect

Specifies that when VCS encounters a pulse shorter than the module path delay, VCS immediately drives an X value on the module output port, and displays an error message. It does not wait until the module path delay elapses.

+pulse\_r/<number>

Reject any pulse whose width is less than number percent of module path delay.

+putprotect+<target\_dir>

Specifies the target directory for protected files.

+race

Specifies that VCS generate a report, during simulation, of all the race conditions in the design and write this report in the race.out file.

+race=all

Analyzes the source code during compilation to look for coding styles that cause race conditions.

+racecd

Specifies that VCS generate a report, during simulation, of the race conditions in the design between the ``race` and ``endrace` compiler directives and write this report in the race.out file.

+race\_maxvecsize=<size>

Specifies the largest vector signal for which the dynamic race detection tool looks for race conditions.

+rad

Performs Radiant Technology optimizations on your design.

+sdfprotect[<file\_suffix>]

Creates a protected SDF file.

+sdf\_nocheck\_celltype

Tells VCs not to check to make sure that the CELLTYPE entry in the SDF file does not match the module identifier for a module instance before back annotating delay values from the SDF file to the module instance.

+sdfverbose

Enables the display of more than ten warning and more than ten error messages about SDF back annotation.

+spl\_read

Tells VCS to treat output ports as inout ports in order to facilitate more accurate multi-driver contention analysis across module boundaries. This option can have an adverse impact on runtime performance.

`+systemverilogext+<ext>`

Specifies a filename extension for source files containing SystemVerilog source code.

`+tetramax`

Enter this option when simulating TetraMAX's testbench in zero delay mode.

TetraMAX can run the `simv` executable file. This option tell VCS to prepare the `simv` executable for use by TetraMAX.

`+timopt+<clock_period>`

Enables Timing Check Optimizations, the `+<clock_period>` argument specifies the clock period of the fastest clock in the design. Please refer to the VCS/VCSi User Guide for more information on this option.

Starts the Timopt timing optimizer. the `+<clock_period>` argument specifies the clock period of the fastest clock in the design. Timopt applies timing optimizations to your design. Timopt also writes a `timopt.cfg` file in the current directory. This file contains clock signals and module definitions of sequential devices it's not sure of. You edit this file and recompile without the `+<clock_period>` argument to obtain more Timopt optimizations.

`+transport_int_delays`

Enables transport delays with full pulse control for single source nets.

`+transport_path_delays`

Turns on the transport behavior for I/O paths.

`+typdelays`

Use typical value when min:typ:max values are encountered in delay specifications and SDF files.

`+v2k`

Enables the use of new Verilog constructs in the 1364-2001 standard.

`+vc[+abstract][+allhdrs][+list]`

Enables the direct call of C/C++ functions in your Verilog code using the DirectC interface. The optional suffixes specify the following:

`+abstract`



Specifies that you are using abstract access through `vc_handles` to the data structures for the Verilog arguments.

`+allhdrs`

Writes the `vc_hdrs.h` file that contains external function declarations that you can use in your Verilog code.

`+list`

Displays on the screen all the functions that you called in your Verilog source code.

`+vcs+dumpvars`

A substitute for entering `$dumpvars`, without arguments, in your Verilog code.

`+vcs+flush+log`

Increases the frequency of flushing both the compilation and simulation log file buffers.

`+vcs+flush+all`

Shortcut option for entering all three of the `+vcs+flush+log`, `+vcs+flush+dump`, and `+vcs+flush+fopen` options.

`+vcs+initmem+0|1|x|z`

Initializes all bits of all memories in the design.

`+vcs+initreg+0|1|x|z`

Initializes all bits of all regs in the design.

`+vcs+lic+vcsi`

Checks out three VCSi licenses to run VCS.

`+vcsi+lic+vcs`

Checks out a VCS license to run VCSi when all VCSi licenses are in use.

`+vcs+lic+wait`

Tells VCS to wait for a network license if none is available.

`+vcsi+lic+wait`

Tells VCSi to wait for a network license if none is available.

`+vcs+vcdpluson`

A compile-time substitute for `$vcdpluson` option. The `+vcs+vcdpluson`

switch enables dumping for the entire design. You would however need to use a debug switch (example -debug\_pp) to dump the data.

#### +vcs+mipdexpand

Intended to use with +oldsdf. When back annotating SDF delay values from an ASCII text SDF file at run-time, if the SDF file contains PORT entries for the individual bits of a port, using this compile-time option enables VCS to backannotate these PORT entry delay values. Similarly, using this compile-time option enables a PLI application to pass delay values to individual bits of a port.

#### +verilog1995ext+<ext>

Specifies a filename extension for source files containing Verilog 1995 source code.

#### +verilog2001ext+<ext>

Specifies a filename extension for source files containing Verilog 2001 source code.

#### +vhdl+<logical\_libname>

This option specifies the VHDL logical library to use for VHDL design entity instances that you instantiate in your Verilog design.

#### +vpi

Enables the use of VPI PLI access routines.

#### +warn=[no]ID|none|all,...

Enables or disables warning messages.

#### Runtime Options

\*\*\*\*\*

运行阶段

#### -a <filename>

Specifies appending all messages from simulation to the bottom of the text in the specified file as well as displaying these messages to the standard output.

#### -assert <keyword\_argument>

The keyword arguments and what they do are as follows:

#### dumpoff

Disables the dumping of SVA information in the VPD file during simulation.

filter

Blocks reporting of trivial SystemVerilog assertion implication successes. These happen when an implication construct registers a success only because the precondition (antecedent) portion is false (and so the consequence portion is not checked). With this option, reporting only shows successes in which the whole expression matched.

finish\_maxfail=<N>

Terminates the simulation if the number of SystemVerilog assertion failures for any assertion reaches N. N must be supplied, otherwise no limit is set.

global\_finish\_maxfail=<N>

Stops the simulation when the total number of failures, from all SystemVerilog Assertions, reaches N.

maxcover=<N>

Disables the collection of coverage information for cover statements after the cover statements are covered N number of times. <N> must be a positive integer, it can't be 0.

maxfail=<N>

Limits the number of SystemVerilog assertion failures for each assertion to N. When the limit is reached, the assertion is disabled. N must be supplied, otherwise no limit is set.

maxsuccess=<N>

Limits the total number of reported SystemVerilog assertion successes to N. N must be supplied, otherwise no limit is set. The monitoring of assertions continues, even after the limit is reached.

nocovdb

Tells VCS not to write the <program\_name>.db file for assertion coverage.

nopostproc

Disables the display of the SVA coverage summary at the end of simulation.

quiet0|1

quiet0 Disables messages, in standard output, about assertion failures.  
quiet1 Disables messages, in standard output, about assertion failures,

but displays the summary of them at the end of simulation.  
The never triggered assertions are also reported.

report[=<filename>]

Generates a SystemVerilog assertion report file in addition to displaying results on your screen. By default the file's name and location is `./simv.vdb/report/ova.report`, but you can change this by entering the filename pathname argument.

success

Enables reporting of successful SystemVerilog assertion matches in addition to failures. The default is to report only failures.

verbose

Adds more information to the report specified by the `report=<filename>` keyword, including assertions that never triggered and attempts that did not finish, and a summary with the number of assertions present, attempted, and failed.

You can enter more than one keyword, using the plus + separator, for example:

`-assert maxfail=10+maxsuccess=20+success+filter`

`-cm line|cond|fsm|tgl|path|branch|assert`

Specifies monitoring for the specified type or types of coverage.

The arguments specifies the types of coverage:

`line` Monitor for line or statement coverage.

`cond` Monitor for condition coverage.

`fsm` Monitor for FSM coverage.

`tgl` Monitor for toggle coverage.

`path` Monitor for path coverage.

`branch` Monitor for branch coverage.

`assert` Monitor for SystemVerilog assertions coverage.

If you want VCS to monitor for more than one type of coverage, use the plus + character as a delimiter between arguments, for example:

`-cm line+cond+fsm+tgl`

`-cm_dir <directory_path_name>`

Specifies and alternative name and location for the coverage database directory.

`-cm_glitch <period>`

Specifies a glitch period during which VCS does not monitor for coverage caused by value changes. The period is an interval of simulation time specified with a non-negative integer.

`-cm_log <filename>`

Specifies a log file for monitoring for coverage during simulation.

`-cm_name <filename>`

As a compile-time or runtime option, specifies the name of the intermediate data files. On the cmView command line, specifies the name of the report files.

`-cm_tglfile <filename>`

Specifies displaying at runtime a total toggle count for one or more subhierarchies specified by the top-level module instance entered in the file.

`-E <program>`

Starts the program that displays the compile-time options that were on the vcs command line when you created the simv (or simv.exe or some other name specified with the -o option) executable file.

`-grw <filename>`

Sets the name of the \$gr\_waves output file to the specified file. The default filename is grw.dump.

`-gui`

Starts the Discovery Visual Environment (DVE) graphical user interface.

`-i <filename>`

Specifies a file containing CLI commands that VCS executes when simulation starts.

`-k <filename> | off`

Specifies an alternative name or location for the vcs.key file into which VCS writes the CLI interactive commands that you enter during simulation. The off argument tells VCS not to write this file.

`-l <filename>`

Specifies writing all messages from simulation to the specified file as well as displaying these messages in the standard output. This option begins with the letter "l" (lowercase "L") for log file.

-ova\_cov

Enables functional coverage reporting.

-ova\_cov\_name <filename>

Specifies the file name or the full path name of the functional coverage report file. This option overrides the default report name and location.

If only a file name is given, the default location is used resulting in `./simv.vdb/fcov/<filename>.db`.

-ova\_cov\_db <filename>

Specifies the path name of the initial coverage file. The initial coverage file is needed to set up the database. By default, an empty coverage file is loaded from `simv.vdb/snps/fcov/default.db`.

-ova\_filter

Blocks reporting of trivial if-then successes. These happen when an if-then construct registers a success only because the if portion is false (and so the then portion is not checked). With this option, reporting only shows successes in which the whole expression matched. This option is enabled by the `-ova_enable_diag` compile-time option.

-ova\_max\_fail <N>

Limits the number of reported failures for each assertion to N. The monitoring of assertions continues, even after this limit is reached. This option is enabled by the `-ova_enable_diag` compile-time option.

-ova\_max\_success <N>

Limits the number of successes for each assertion to N. The monitoring of assertions continues, even after the limit is reached. This option is enabled by the `-ova_enable_diag` compile-time option.

-ova\_name <name | /<pathname>/<name>

Specifies an alternative name or location and name for the `./simv.vdb/scov/results.db` and `./simv.vdb/reports/ova.report` files. You use this option if you want data and reports from a series of simulation runs. It's a way of keeping VCS from overwriting these files from a previous simulation.

If you just specify a name the alternatively named files will be in the default directories. If you specify a pathname, with an argument containing the slash character `/`, you specify a different location and name for these files, for example:

`-ova_name /net/design1/ova/run2`

This example tells VCS to write run2.db and run2.report in the /net/design1/ova directory.

-ova\_report [<filename>]

Specifies writing an OpenVera Assertions report file. The default file name and location is simv.vdb/report/ova.report but you can specify a different name and location as an argument to this option.

-ova\_simend\_max\_fail <N>

Terminates the simulation if the number of failures for any assertion is reached. This option is enabled by the -ova\_enable\_diag compile-time option.

-ova\_success

Enables the reporting of successful matches. This option is enabled by the -ova\_enable\_diag compile-time option.

-ova\_quiet [1]

Disables displaying functional coverage results on the screen. The optional 1 argument specifies displaying a summary of these results.

-ova\_verbose

Adds more information to the end of the report including assertions that never triggered and attempts that did not finish, and a summary with the number of assertions present, attempted, and failed.

-q

Quiet mode. Suppress printing of VCS header and summary information, the proprietary message at the beginning of simulation, and the VCS Simulation Report at the end of simulation (time, CPU time, data structure size, and date)

-s

Stops simulation just as it begins, and enters interactive mode. Use with the +cli+<number> option.

-sverilog

Enables the use of the Verilog language extensions in the Accellera SystemVerilog specification.

-ucli

Enables the use of UCLI commands.

-V

Verbose mode. Print VCS version and extended summary information.  
Prints VCS compile and run-time version numbers, and copyright information, at start of simulation.

`-vcd <filename>`

Sets the output VCD file name to the specified file.  
The default filename is `verilog.dump`.

A `$dumpfile` system task in the Verilog source code will override this option.

`+vcdfile+<filename>`

Specifies the VCD file you want to use for post-processing.

`-vhdlrun "<scsim command line options>"`

VCS-MX option to pass scsim command line options for the VHDL part of a mixed HDL design.

`-xzcheck [nofalseneg]`

Checks all the conditional expressions in the design and displays a warning message every time VCS evaluates a conditional expression to have an X or Z value.

`nofalseneg`

Suppresses the warning message when the value of a conditional expression transitions to X or Z and then to 0 or 1 in the same simulation time step.

`+cliecho`

Specifies that VCS displays CLI commands in a file that you specify with the `-i` option as VCS executes these commands. UNIX only.

`+maxdelays`

Species using the compiled SDF file for maximum delays generated by the `+allmtm` compile-time option.

Also specifies using maximum delays for SWIFT VMC or SmartModels or Synopsys hardware models if you also enter the `+override_model_delays` runtime option.

`+mindelays`

Specifies using the compiled SDF file for minimum delays generated by the `+allmtm` compile-time option.

Also specifies using minimum delays for SWIFT VMC or SmartModels or Synopsys hardware models if you also enter the `+override_model_delays` runtime option.



+no\_notifier

Suppresses the toggling of notifier registers that are optional arguments of timing check system tasks.

+no\_pulse\_msg

Suppresses pulse error messages, but not the generation of StX values at module path outputs when a pulse error condition occurs.

+no\_tchk\_msg

Disables the display of timing check warning messages but does not disable the toggling of notifier registers in timing checks. This is also a compile-time option.

+notimingcheck

Suppress timing checks.

+ntb\_cache\_dir=<path\_name\_to\_directory>

Specifies the directory location of the cache that VCS maintains as an internal disk cache for randomization.

+ntb\_debug\_on\_error

Causes the simulation to stop immediately when a simulation error is encountered. In addition to normal verification errors, This option halts the simulation in case of runtime errors as well.

+ntb\_enable\_solver\_trace=<value>

Enables a debug mode that displays diagnostics when VCS executes a randomize() method call. Allowed values are:

0 - Do not display (default).

1 - Displays the constraints VCS is solving.

2 - Displays the entire constraint set.

+ntb\_enable\_solver\_trace\_on\_failure=<value>

Enables a mode that displays trace information only when the VCS constraint solver fails to compute a solution, usually due to inconsistent constraints. When the value of the option is 2, the analysis narrows down to the smallest set of inconsistent constraints, thus aiding the debugging process. Allowed values are 0, 1, 2. The default value is 2.

+ntb\_exit\_on\_error[=<value>]

Causes VCS to exit when value is less than 0. The value can be:

0: continue

1: exit on first error (default value)

N: exit on nth error.

When value = 0, the simulation finishes regardless of the number of errors.

+ntb\_load=path\_name\_to\_libtb.so

Specifies loading the testbench shared object file libtb.so.

+ntb\_random\_seed=<value>

Sets the seed value used by the top level random number generator at the start of simulation. The random(seed) system function call overrides this setting. The value can be any integer number.

+ntb\_solver\_mode=<value>

Allows choosing between one of two constraint solver modes. When set to 1, the solver spends more pre-processing time in analyzing the constraints, during the first call to randomize() on each class. When set to 2, the solver does minimal pre-processing, and analyzes the constraint in each call to randomize(). Default value is 2.

+ntb\_stop\_on\_error

Causes the simulation to stop immediately when a simulation error is encountered, turning it into a cli debugging environment. In addition to normal verification errors, ntb\_stop\_on\_error halts the simulation in case of run time errors. The default setting is to execute the remaining code within the present simulation time.

+override\_model\_delays

Enables you to use the +mindelays, +typdelays, or +maxdelays runtime options to specify timing for SWIFT SmartModels or Synopsys hardware models.

+sdfverbose

Enables the display of more than ten warning and ten error messages about SDF back annotation.

+typdelays

Specifies using the compiled SDF file for typical delays generated by the +allmtm compile-time option.

Also specifies using typical delays for SWIFT VMC or SmartModels or Synopsys hardware models if you also enter the +override\_model\_delays runtime option.

+vcs+dumparrays

Enables dumping memory and multi-dimensional array values in the

VCD file. You must also have use the +memcbk compile-time option.

+vcs+dumpoff+<t>+<ht>

Turn off value change dumping (\$dumpvars system task) at time <t>.

<ht> is the high 32 bits of a time value greater than 32 bits.

+vcs+dumpon+<t>+<ht>

Suppress \$dumpvars system task until time <t>.

<ht> is the high 32 bits of a time value greater than 32 bits.

+vcs+dumpvarsoff

Suppress \$dumpvars system tasks.

+vcs+finish+<t>+<ht>

Finish simulation at time <t>.

<ht> is the high 32 bits of a time value greater than 32 bits.

+vcs+grwavesoff

Suppress \$gr\_waves system tasks.

+vcs+ignorestop

Tells VCS to ignore the \$stop system tasks in your source code.

+vcs+flush+log

Increases the frequency of flushing both the compilation and simulation log file buffers.

+vcs+flush+dump

Increases the frequency of flushing all the buffers for VCD files.

+vcs+flush+fopen

Increases the frequency of flushing all the buffers for files opened by the \$fopen system function.

+vcs+flush+all

Shortcut option for entering all three of the +vcs+flush+log, +vcs+flush+dump, and +vcs+flush+fopen options.

+vcs+learn+pli

Keeps track of where you use ACC capabilities for debugging operations so that you can recompile your design and in the next simulation enable them only where you need them.

With this option VCS writes the pli\_learn.tab secondary PLI table file. You input this file when you recompile your design

with the +applylearn compile-time option.

+vcs+lic+vcsi

Checks out three VCSi licenses to run VCS.

+vcsi+lic+vcs

Checks out a VCS license to run VCSi when all VCSi licenses are in use.

+vcs+lic+wait

Wait for network license if none is available when the job starts.

+vcsi+lic+wait

Tells VCSi to wait for a network license if none is available.

+vcs+mipd+noalias

If during a simulation run, acc\_handle\_simulated\_net is called before MIPD annotation happens, a warning message is issued. When this happens you can use this option to disable such aliasing for all ports whenever mip, mipb capabilities have been specified. This option works for regular sdf annotation and not for compiled SDF.

+vcs+nostdout

Disables all text output from VCS including messages and text from \$monitor and \$display and other system tasks. VCS still writes this output to the log file if you include the -l option.

+vcs+stop+<t>+<ht>

Stop simulation at time <t>. <ht> is the high 32 bits of a time value greater than 32 bits (optional). See the section on "Specifying A Long Time Before Stopping Simulation" in the VCS/VCSi User Guide.

+vera\_load=<filename.vro>

Specifies the VERA object file.

+vera\_mload=<filename>

Specifies a text file that contains a list of VERA object files.

Options for Specifying How VCS Writes the VPD File

\*\*\*\*\*

提高

+vpdfile+<filename>

At runtime, defines an alternative name of the VPD file that VCS writes instead of the default name vcdplus.vpd.

+vpdfileswitchsize+<number\_in\_MB>

Specifies a size for the VPD file. When the VPD file reaches this size, VCS closes the VPD file and opens a new one with the same design hierarchy as the previous VPD file. There is a number suffix added to the VPD file name to differentiate them.

-PP

Enables you to enter in your Verilog source code system tasks like \$vcdpluson to create a VPD file during simulation. This option minimizes net data details for faster post-processing.

+vpdbufsize+<MB>

VCS uses an internal buffer to store value changes before it writes them to the VPD file on disk. VCS makes this buffer size either 5 MB or large enough to record 15 value changes for all nets and registers in your design, whichever is larger.

You can use this option to override the buffer size that VCS calculates for the buffer size. You specify a buffer size in megabytes.

+vpddrivers

Tells VCS to record the values of all the drivers of all the nets.

+vpdfilesize+<MB>

Specifies the maximum size of the VPD file. When VCS reaches this limit, VCS overwrites the oldest simulation history data in the file with the newest.

+vpdignore

Tells VCS to ignore \$vcdplus system tasks so VCS does not write a VPD file.

+vpdports

Tells VCS to record, in the VPD file, the port direction of signals that are ports.

+vpdnocompress

Disables the automatic compressing of the data in VPD files.

+vpdupdate

If VCS is writing a VPD file during simulation, this option enables you to have VCS halt writing to the VPD file while the simulation is running and so that you can view the recorded results in DVE. This option enables you to use the update feature in DVE.

+vpdnostrengths

Disables recording strength information in the VPD file.

#### Options For Calling The vcd2vpd and vpd2vcd Utilities

\*\*\*\*\*

波形转换

-vcd2vpd <vcd\_filename> <vcdplus\_filename>

Tells VCS to find and run the vcd2vpd utility that converts a VCD file to a VPD file. VCS inputs to the utility the specified VCD file and the utility outputs the specified VPD file.

-vpd2vcd <vcdplus\_filename> <vcd\_filename>

Tells VCS to find and run the vpd2vcd utility that converts a VPD file to a VCD file. VCS inputs to the utility the specified VPD file and the utility outputs the specified VCD file.

#### Environment Variables

\*\*\*\*\*

环境变量

DISPLAY\_VCS\_HOME

Enables the display at compile time if the path to the directory specifies with the VCS\_HOME environment.

LM\_LICENSE\_FILE

The complete path of the VCS license file or port@host.

PATH

On UNIX add \$VCS\_HOME/bin to this environment variable.

VCS\_HOME

Specifies the directory where you installed VCS.

VCSI\_HOME

Specifies the directory where you installed VCSi.

TMPDIR

Specifies the directory for temporary compilation files.

VCS\_CC

Specifies the C compiler.

#### VCS\_COM

Specifies the path to the VCS compiler executable named vcs1 (or vcs1.exe).

#### VCS\_LOG

Specifies the runtime log file name.

#### VCS\_RUNTIME

Specifies which runtime library named libvcs.a VCS uses.

#### VCS\_SWIFT\_NOTES

Enables the printf PCL command.

#### VCS\_WARNING\_ATSTAR

Specifies the number of signals in a Verilog-2001 implicit sensitivity list that must be exceeded before VCS displays a warning. The default limit is 100 signals.

PDF files for the VCS documentation set are in \$VCS\_HOME/doc/UserGuide.