VCS

The Verilog Compiler Simulator

仿真的过程

- 编译Compile

  VCS对源文件进行编译，生成中间文件和可执行文件

- 仿真Simulate

  运行可执行文件，对设计进行仿真

- 调试

  通过观察波形、设置断点、追踪信号、查看schematic等来发现错误，并进行纠正

- 覆盖率测试

  通过在编译时，加入覆盖率测试的选项、仿真后，生成包含覆盖率信息的中间文件来显示测试平台的正确性和完备性。

一、编译：VCS

一个常见的编译命令如下：

```
vcs design.v -f file.f -y lib_dir +libext+.v -v lib_file pli.c \
  -P pli.tab -Mupdate -o bin_name -l log_file +v2k -R -RI -s \
  -debug_all +vcsd +define+m1+m2 +timopt+<period> -line \
  +incdir+dir1+dir2 +memopt[+2] -sverilog -mhdl +ad \
  -full64 -comp64 +nospecify +notimingcheck -ntb +race \
  -ova_file file_ova +vpdfile+file_vpd +vpdfilesize+nMB \
  +vpdupdate +cli+1|2|3|4 +vcs+initmem+0|1|x|z \
  +vcs+initreg+0|1|x|z +vc
  -cm line|tgl|cond|fsm|path|branch -cm_dir dir \
```

一、编译：VCS

-v *lib_file*　　lib_file是Verilog文件，包含了引用的module的定义，可以是绝对路径，也可以是相对路劲。

-y *lib_dir*　　lib_dir是参考库的目录，vcs从该目录下寻找包含引用的module的Verilog文件，这些文件的文件名必须和引用的module的名一样

+libext+*.v*+*.vhd*+...　　vcs在参考库目录下寻找以.v和.vhd为扩展名的文件。多个扩展名之间用“+”连接。

+incdir+*dir1*+*dir2*+...　　vcs从dir1和dir2等目录下寻找源代码中`include指示的文件。

-full64　　vcs以64位模式编译，生成64位的simv。

-comp64　　vcs以64位模式编译，生成32位的simv。

-file *list_file*　　list_file文件中式源文件的列表以及编译选项。

-debug_pp　　产生vpd文件，enable DVE for post-processing。

-debug　　相对于-debug_pp，多了UCLI调试功能。

-debug_all　　相对于-debug，多了单步调试功能。

-gui　　在仿真时，使用dve调试

-assert dve　　enable Systemverilog assertion tracing in the VPD file

-R　　编译后立即进行仿真

一、编译：VCS

-pvalue+*parameter_hierarchical_name*=*value*　改变设计中的参数值，例如：
vcs -pvalue+test.d1.param1=33

-parameters *filename*　更改*filename*中的参数值

-notice　给出详细的编译信息

-q　不在终端输出编译时的信息

-l *log_file*　将日志写入制定的*log_file*中

+define+*macro1=value*+macro2=value+...　将*macro1*和*macro2*，...传给源文件中同名的宏，如果value是字符串的话，要用双引号括起来

-o *bin_name*　产生*bin_name*的可执行文件，而不产生simv

+v2k　支持Verilog 2001标准

+vcs+initmem+0|1|x|z　初始化存储器和多位寄存器数组

+vcs+initreg+0|1|x|z　初始化reg变量，不初始化其他寄存器型变量

-xzcheck　当一个条件等于x、z值时，VCS给出警告信息，可以在某些模块中加入$xzcheckoff和$xzcheckon来屏蔽该选项

-RI　执行完编译后，立即运行VirSim，该选项不能和+vcds联用

-s　编译之后，运行simv时，仿真时刻停止0处

+define+*macro1*+...　将宏*macro1*传给源代码。

-sverilog　提供对SystemVerilog的支持

一、编译：VCS

-line 实现单步仿真，将会极大地增加运行时间

-mhdl 实现混合HDL语言的编译和仿真

+ad=<*filename*> 实现混合信号的编译和仿真

-nospecify 禁止模块路径延迟和时序检查，提高仿真速度

+notimingcheck 禁止时序检查任务，可以改善仿真速度

+vpdfile+*filename* 指定要写入的vpd文件名，而不用vcdplus.vpd

+vpdupdate 同时读写vpd文件

+vpdfilesize+*n*MB 指定vpd文件的最大size

+race 自动产生一个race.out文件，列出了竞争

+cli+1|2|3|4 指明仿真时用UCLI模式，后边不同的数字代表了不同的操作级别，一般用3既可

+prof 产生一个名为vcs.prof的文件，报告CPU和memory的使用情况

-parallel +fc[=*No1*]|+sva[=*No2*]|+tgl[=*No3*]|+vpd[=*No4*] 使用多核处理器的选项，若每个参数后不跟数字，则表示使用1级；fc为多核功能覆盖，sva为多核SystemVerilog断言，tgl为多核toggle覆盖，vpd为多核VCD+ dumping。*No*实际上就是使用的cpu核的个数。

-parallel -o *para_bin_name* 指定一个能使用多核技术的可执行文件名。

-timescale=1ns/10ps 指定仿真单位和时间精度

一、编译：VCS

+memcbk　　对存储器进行检查，使用$vcdplusmemon时使用该选项

+rad　　开启仿真时的radiant技术，会增加编译时间

-f *filename*　　*filename*中包含了源文件和编译选项，但是有些编译选项是有限制的，如下：1）除了+comp64，+full64，+memopt外，所有以+开头的options都可以包含在该文件中；2）-f、-gen_asm、-gen_obj、-line、-l、-u、-v、-y可以包含于该文件中，其他一“-”开头的options都不能包含在该文件中；3）不能包含C 源文件和PLI应用程序；4）不能包含$、`、！这3个符号；5）不能有//、/* */这两种注释符。

使用-file *filename*选项可以克服-f选项的限制。

-j*N*　　N为处理器的个数，注意N和j之间没有空白符

-syslib <libs>　指明创建可执行文件时使用的系统库

-timescale=<time_unit>/<time_precision>　指明时间尺度

-ucli　　指明在运行仿真时，进入ucli模式

-Vt　　给出警告信息，和每条命令执行所用的时间

+tetramax　　和teramax混合仿真时，加上该选项

+timopt+<clock_period>　指明设计中最快的时钟的周期

+vcs+dumpvars　代替源代码中没有参数的$dumpvars任务

一、编译：VCS——脉冲控制

VCS默认的路径延迟（module path delay）、sdf文件反标的互联延迟（INTERCONNECT delay）为惯性延迟；而原语门、开关、连续赋值语句和MIPD（module input port delay）的延迟则只能是惯性延迟。

+transport_path_delays +pulse_e/*num1* +pulse_r/*num2*

+transport_int_delays +pulse_int_e/*num1* +pulse_int_r/*num2*

上述两个选项开启了传输延迟模式，后面的两个选项是必须的；num1和num2都是延时的百分比，小于num2的脉冲会被过滤掉（filter out），大于num2但小于num1的脉冲会被x值代替。如果想实现真正的传输延迟，将num1和num2设置为0，即可。

对惯性延迟，是默认的，以下两个选项同前所述。

+pulse_e/*num1* +pulse_r/*num2*

+pulse_int_e/*num1* +pulse_int_r/*num2*

窄脉冲的尾沿安排的时间会取消起始沿安排的时间。若想使sdf反标的线网不使用原语、连续赋值、MIPD的惯性延迟模型，要加上+multisource_int_delays选项，否则使用MIPD的惯性延迟模型。对惯性延迟，num1=0和num2=0并不表示传输延迟，而是指小于延时的脉冲宽度都被忽略。这两个选项对分布延迟不起作用

+pulse_on_detect 探测到脉冲宽度小于延时，立即在对应时刻给出x，该选项对分布延迟不起作用

+no_pulse_msg 当脉冲宽度小于延时的时候，不给出信息

+pulse_on_event 对上升、下降延时不同的情况是，给出x

一、编译：VCS——延时说明

+delay_mode_path　　模块的延时使用路径延时

+delay_mode_distributed　　模块的延时使用分布延时，对分布延时，只要脉冲宽度小于#后的数字，直接过滤掉（针对惯性延时）。

+delay_mode_unit　　模块的延时使用所有时间精度中的最小值，specify中的延迟不起作用，#后的所有数字变为1，单位使用最小精度

+delay_mode_zero　　模块的延时使用0延时

不指定延时模式时，VCS使用路径延时和分布延时中的最大值。

二、仿真：simv 运行选项

命令：simv *runtime_options*，以下是运行选项的说明：

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

-cm_dir *directory*　指明仿真将中间文件存在哪里

-l *log_file*　记录DVE或VCS的log文件

-gui　启动DVE

-ucli　进入UCLI交互模式

-do *ucli_command_file*　*ucli_command_file*是UCLI命令的列表文件

+vcs+stop+*time*　指定仿真运行中断的的时间

+vcs+finish+*time*　指定仿真运行结束的时间

-cm_glitch <*period*>　对于小于period的脉冲不进行覆盖

-cm_name <*filename*>　指明test文件的文件名

-cm_tglfile <*filename*>

-cm_log <*filename*>　指明仿真期间关于coverage的log文件名

-q　安静模式

-sverilog

-V　verbose mode

二、仿真：simv 运行选项

-vcd <filename>	指明一个VCD文件名
-xzcheck	当检查到一个变量为x或z时，给出warnings
+notimingcheck	不进行时序检查，加快仿真速度
+vcs+dumparrays	在VCD中dump存储器和多维数组，必须和+memcbk编译选项联用
+vcs+dumpon+time	告诉vcs直到time时刻，$dumpvars才能起作用
+vcs+dumpoff+time	告诉vcs直到time时刻，$dumpvars才不起作用
+vcs+dumpvarsoff	关闭$dumpvars系统任务
+vcs+flush+dump	加快将缓冲中的数据写入vcd文件
+vcs+flush+all	加快将缓冲中的数据写入各种文件

三、调试：DVE

$dumpvars　创建一个VCD文件

$vcdplusfile(“*filename*”);　指定一个vpd文件名义代替默认的vcdplus.vpd

$vcdpluson(*level, instance, net_or_reg*)　开始创建一个vpd文件，level表示记录的层次，0或缺省表示所有层；instance为模块的例化名；net_or_reg为net变量或reg变量，缺省时记录所有的net和reg变量

$vcdplusoff(*instanc, net_or_reg*)　停止写vpd文件，这两个系统任务用在initial块中，两个任务之间的时间段内会写vpd文件

$vcdplusautoflushon　当vcs遇到中断时，将仿真结果写入vpd文件中

$vcdplusautoflushoff　关闭自动flush

$test$pluseargs　在testbench中包括该任务，可以在仿真中加入该任务的参数来控制某段代码是否执行。例如：

```
initial
    if($test$plusargs(“postprocess”)) begin
        $vcdpluson;
        #10000 $vcdplusoff;
    end
```

运行时，simv +postprocess就可以执行if中的代码，否则不执行。

三、调试：DVE

$value$plusargs(“format”,signal)　在仿真时将一个参数传入设计中，其返回值必须是integer型，如源代码中有如下语句：

```
reg[31:0] r1; integer status;
initial status=$value$plusargs(“%d’,r1);
```

运行时，simv +r1=10 即可将10传递给status

$vcdplusmemon(*mda_name,n1,n2,n3,...*);　在仿真时记录存储器和多维数组的值变情况，*mda_name*为存储器或多维数组的名字；n1，n2，n3...分别是要记录的范围。如 reg[7:0] mem[1:3] [4:6] [7:9];如要记录第2行第2列第2格，vcdplusmemon(mem,2,5,8);不加数字的话，记录整个存储器或MDA。使用该函数时，必须在编译时加上+memcbk和+v2k选项。

$dumpports(*instance,filename*);　创建一个EVCD文件，*instance*是例化名，*filename*是要创建的EVCD文件的文件名。该任务的使用必须在编译时加上+dumpports+1si编译选项

VCS中有如下3个应用程序：vcd2vpd、vpd2vcd、vpdmerge，用法如下：

vcd2vpd　*vcdfile vpdfile*　　将vcd文件转成vpd文件

vpd2vcd　*vpdfile vcdfile*　　将vpd文件转成vcd文件

vpdmerge -o *merge_file　input_vpd1 input_vpd2*　该命令将两个vpd文件合并为一个vpd文件，但要注意，两个vpd文件中的仿真时刻必须不同。

三、调试：DVE

$sdf_annotate的用法：

$sdf_annotate(“*sdf_file*”, [*module_instance*], [“*sdf_cfgfile*”], [“*sdf_logfile*”], [“*mtm_spec*”], [“*scale_factors*”], [“*scale_type*”]);

sdf_file: sdf文件，可以是相对路径也可以是绝对路径

module_instance: 反标开始的范围，缺省时为调用该函数的instance

sdf_cfgfile: sdf文件的配置文件

sdf_logfile: VCS将错误信息和warnings发送到该文件

mtm_spec: 指明VCS反标的延迟类型，该值可以用如下字符串代替：MINIMUM，TYPICAL，MAXIMUM，TOOL_CONTROLL（缺省），即将反标的延迟用于那一种情况

scale_factors: 比例因子，对三种类型，缺省的情况为“1.0:1.0:1.0”，用户可以设置为”0.8:1.0:1.3”或其他值，语法如见双引号中的情况

scale_type: 可选参数为FROM_TYPICAL、FROM_MINIMUM、FROM_MAXIMUM，如FROM_MAXIMUM表示使用的延迟时sdf文件中的最大延时

三、调试：DVE

- 在DVE的控制台的提示符下，可以输入如下名获得帮助:

help　　列出DVE的基本命令

help -all　　列出DVE的所有命令

- 一个常见的DVE的命令:

dve -vpd *vpd_file* -session *session_file* -cmd "*ucli_cmd*" -script *script_file* -full64

不加任何参数和选项的dve命令等同于simv -gui或vcs -gui -R，*vpd_file*是一个vpd文件名，*session_file*是Tcl格式的文件，保存了dve的配置信息，*ucli_cmd*是一条UCLI命令，*srcipt_file*是一个Tcl脚本，保存了dve的一些命令。-session的优先级高于-script，-full64是对64位平台的支持

- dump signal values

  Simulator > Dump 、Simulator > Add Dump或在Hierarchy Pane中右击某模块或信号，选择Add Dump项

- force signal value

  Simulator > Add Force

- 设置断点：行断点可以在源代码窗口中设置，在行属性那一栏中用右键的弹出菜单来实现enable、disable、delete、delete all 断点。还可以通过下来菜单实现：Simulator > Set Breakpoints，可以设置5类断点

三、调试：DVE

- 要将一个信号移动到想要的时刻，Signal > Shift Time，在弹出的box中输入想要移动的距离（时间段），整数右移，负数左移。
- 在Path Schematic中显示互联信息：1.在path schematic中选择一个对象（objiect）；2.Scope > Add Fanin/Fanout 3.点击Set Selected或Add Selected 4.设置其他属性 5.点击OK来刷新path schematic 6. Signal > Annotate Values来查看信号的值 7.
- 穿越边界来跟踪一个信号：将一个信号拖入schematic view中，选择该信号，右击，选择 Expand Path；当想追踪一个值为X的信号时，在schematic中选中该信号，然后Trace > Trace X

三、调试：UCLI

进入ucli的方式：simv -ucli [simv_options]

help *cmd_name*　　*cmd_name*是ucli命令名

run *10000ns*　　运行10000ns，无时间参数，表示一直运行

step　　运行下一行

dump -add -depth *no* -file *filename*　　dump一个文件名为*filename*的vpd文件，深度为*no*

更进一步的UCLI命令见ucli_userguide.pdf

四、分析覆盖报告：cmView

vcs -cm_pp -cm line -cm_dir directory -cm_hier *hieroptions* -gui

打开cmView，对vcs

vcs -cm_pp -cm line -cm_dir directory -cm_hier *hieroptions*

batch mode 批处理模式

cmView -b -mc_dir *directory* -cm hier [*heiroptions*] 批处理模式，对vcs-mx

cmView -mc_dir *directory* -cm hier [*heiroptions*] 图形处理模式，对vcs-mx

urg -help|h

urg -dir *directory_name* 指明覆盖率数据的目录

Coverage Metrics：覆盖率测试

- Line/Statement Coverage: 检测哪一行、哪一条语句没有执行
- Condition Coverage: 检测代码中的条件为1或0是否都出现过。默认情况下不对for循环和用户定义的task和function进行检测。加-cm_cond tf可以改变上述情况。
- FSM Coverage: 检测状态的跳转和哪一个状态没有被跳转到，以及错误的状态跳转
- Toggle Coverage: 检测net、reg和向量中的每一位是否都经过了0->1和1->0的跳转
- Path Coverage: 显示initial和always块中所有条件的组合是否实现。

cmView不能图形化得显示path覆盖的相关数据。故不能有gui选项。

- Branch Coverage: 检测if、case、？：中的每条分支是否执行。

Coverage Metrics：覆盖率测试

VCS在有覆盖率测试的选项，经编译后，默认情况会建立一个simv.cm的目录，该目录下又有3个目录，分别是：db、coverage、reports，其中db、coverage目录下各有2个子目录，分别是verilog、vhdl。

仿真后，在db/verilog下会建立一个文件cm.decl_info的文件，cmView通过读取该文件来显示覆盖率或写覆盖率的report文件到reports目录中。

VCS将默认的test.line，test.cond，test.tgl，test.fsm，test.path，test.branch这些文本文件写入coverage/verilog目录下。

对VCS，启动cmView的命令是:vcs -cm_pp -gui|batch；对VCS-MX，启动cmView的命令是：cmView -b。-b是指以批处理的方式运行，即写覆盖率报告。

以下将对一些有关覆盖率的编译选项、运行选项、以及cmView选项说明:

-cm_dir *directory* 既是编译选项、仿真选项，也是cmView选项，*directory*是该选项的参数。编译时告诉VCS建立一个*directory.cm*的目录以代替默认的*simv.cm*目录；仿真时可以不加该选项及其参数；但查看时必须告诉cmView到哪里去寻找所需要的中间文件。若仿真时也有该参数及其参数，则在运行cmView的时候加上两个-cm_dir *directory2* -cm_dir *directory2*。该选项的优先级高于-o选项。

Coverage Metrics：覆盖率测试

-o *bin_name* ： 编译选项，生成一个名为*bin_name*的目录来代替默认的*simv.cm*目录，同时生成一个二进制可执行文件*bin_name*来代替默认的**simv**。

-cm_libs vy+celldefine: 编译选项。VCS默认不对参考库中的设计和cell进行覆盖报告，加入该选项可以实现上述功能，

-cm_noconst: 编译选项。指示VCS对代码中恒为1或0所导致的非执行代码也进行覆盖报告。

-cm_name *testname*: 编译选项。将默认的test.line、test.tgl等的名字改为testname.line、testname.tgl等。

-cm_hier +tree *instance_name* [*level_number*]

- tree instance_name [*level_number*]

+module *module_name* -module *module_name*

+file *file_name* -file *file_name*

+filelist *listfile* -filelist *listfile*

+library *lib_name* -library *lib_name*

既是编译选项，也是cmView选项。指示VCS: +: 只进行指定的instanc、module、file、library的覆盖编译; -: 不进行只等的instance、module、file、library的编译覆盖。Level_number缺省表示对应的instance的所有层都进行覆盖编译，0表示所有层，1表示第一层，以此类推。

Coverage Metrics：覆盖率测试

-cm_glitch *integer*: 编译选项。该选项只对line、toggle、condition起作用。指示VCS对小于integer的宽度的毛刺或脉冲不进行覆盖编译。

-cm_nocasedef: 编译选项，对case中的default不进行覆盖编译。

-cm_verbose 2|1: cmView选项，是否详细的写summary文件，2详细。

-cm_report *summary*: cmView选项，写一个名为*summary*的总结文件。

-cm_report annotate: cmView选项，写反标了覆盖信息的源文件。

-cm_summary noemail: cmView选项，写完后不发email，以节省磁盘。

Line coverage相对应的选项:

-cm_line contassign: 编译选项，使VCS对连续赋值语句也进行行覆盖编译。

Toggle coverage相对应的选项:

-cm_tgl mda: 编译选项，使VCS对多维数组和存储器也进行toggle覆盖测试。

-cm_tgl portsonly: 编译选项，只对ports进行toggle覆盖测试。

-cm_tgl_uncovef *uncovfile*: cmView选项。将没有toggle的信号，即没有覆盖的信号写入*uncovfile*文件。

Condition coverage对应的选项:

-cm_cond 编译选项，只后可以跟的参数如下:

full: 除了读&&、||、?:条件进行检测外，增加其他操作符的条件检测，如: ==、&、 等。

Coverage Metrics：覆盖率测试

std: basic: sop: 积之和; for: 循环; event: 用户定义的函数和任务tf: allops: ports: 检测端口的变化; allvectors: scalarbitwise: 多个参数用+号来连接。

其中，不能同时使用full、std和basic参数。常用的是full。

-cm_cond_branch: 编译选项。使条件覆盖的报告更明晰。

FSM的编译选项:

-cm_fsmopt allowTemp:编译选项。使能 状态寄存器没有被直接赋值的覆盖编译。

-cm_fsmopt report2StateFsms:编译选项。使能只有两个状态的覆盖编译。

-cm_fsmopt reportXassign: 编译选项。

-cm_fsmopt sequence: 编译选项。使能 1->2->3->4类似的覆盖编译。

-cm_fsmopt optimist: 指明错误的状态跳转次数

-cm_count 记录状态跳转的次数、条件满足的次数、line执行的次数、toggle的次数

Coverage Metrics：覆盖率测试

cmView自动将各次的仿真时的中间文件合并。所以，只需要使用同一个testbench，编译一次，但多次仿真，就可以得到多次仿真的覆盖报告。如如下几种方法:

1）编译一次，运行N次，每次运行前更改$readmemb任务的文本文件，就等同于更改了输入，得到多个test文件（test1.*test2.*...testN.*）；

simv -cm line -cm_name *test1*

simv -cm line -cm_name *test2*

最后运行cmView。Vcs -cm_pp -cm *merge_name*，cmView就会在simv.cm/reports目录下写一个混合后的报告。

2)编译一次，运行N次，每次更改$test$plusargs认为的参数，因为每个该任务的参数对应不同$readmemb的输入文件，等于更改了输入。

simv -cm_name *test1* -cm *mac1*

simv -cm_name *test2* -cm *mac2*

最后运行cmView。

3)编译多次，建立多个simv，运行多次，每次用的testbench不同。

vcs -o *simv1* -cm ...

vcs -o *simv2* -cm ...　　运行simv1，simv2等。最后执行cmView

vcs -cm_pp -cm line -cm_dir *./simv1.cm* -cm_dir *./simv2.cm* *merge_name*

transport delay和inertial dealy

传输延迟和惯性延迟

transport delay: 小于该延迟的脉冲仍然可以通过。

inertial delay: 小于该延迟的脉冲无法通过，默认的sdf文件中的gates、switchs、UDP、MIPD（module input port delay）和连续赋值的延迟都是惯性延迟。