

开组会的时候我提出了一个问题，意思为一个模块的层级多，例如一个模块由 50 多组 FIFO 模块构成，那么对于前端验证来说，这个模块在 debug 过程中会花费大量的体力劳动，如何减少机械的过程从而提高效率。立波哥说了一种思路，让我自行思考，即所有人力点鼠标能解决的东西，用脚本也能实现。

带着这个问题，我就开始查询可以进行工具交互的语言，中间发现这个问题的本质是工具！前端验证使用的是 SYNOPSYS 的系列工具，那么脚本的交互一定是基于工具且工具支持的。带着这个方向在小乌龟查询工具的手册，发现了 Synopsis 工具支持的脚本交互接口--UCLI。

UCLI 即 Unified Command-line Interface，UCLI 功能基于 TCL (Tool Command Language)。UCLI 接口不需要重新编译仿真顶层，使用高级语言接口，容易完成复杂处理，例如传递变量、使用正则表达式等。

拿 VCS 为例，开启 UCLI 接口功能需要在编译时添加 -debug\_access，启用所有 UCLI 命令则在命令后再加上 +all；编译完成后跑仿真的命令，需要添加 -ucli，后面添加参数：

-l	logFilename	捕获仿真输出
-a	logFilename	比-l 参数多了 如果文件不存在则创建
-i	inputFilename	从文件中读取 UCLI 命令，然后输入

-k	keyFilename	写入命令到 <i>inputFilename</i> 中，可以作为后续-i 命令用
----	-------------	---

看我们发现了什么，是不是很眼熟，在我们的 ATB 工具中生成的 Makefile 中见过！看下图 96 131 行，ATB 默认开启了 UCLI 接口，吃的脚本就是-do simrun.do。原来我们在 simrun.do 中干的事情，就是 UCLI 的功能。

```

93 # -----
94 # Elaborate Paramters for VCS
95 # -----
96 DEFAULT_ELAB_OPTS = -lca -repotstats -j8 -check +vpi +acc+2 -check -full64 -CFLAGS -DVCS
  -debug access+all -lca ${UVM_HOME}/src/dpi/uvm_dpi.cc ${VERDI_OPTS}
97 ELAB_CMD = bsub -Is vcs -full64 -sverilog ${ADD} -nc
98 ELAB_LOG = elab.log
99 # -----
131 RUN_OPTS += $(ADD) -lca +notimingchecks -do simrun.do -l simrun.log -ucli -sv_lib dpi_c_lib -sv_lib
  aebp_top_dpi_cfg -sml $(UVM_RUN_OPTS) +UVM_VERBOSITY=$(VERBOSISTY) $(GUT_OPTS) +UVM_TIMEOUT=50000000

```

我们在 simrun.do 脚本经常做的事情是 force 信号，这只是 UCLI 能解决的场景中的冰山一角。我们设想一个案例，一个 SoC 测试用例，跑完需要很久，甚至按天计算，如果仿真过程中服务器宕机导致仿真中断，会造成数据损失，那为了防止时间浪费，需要每隔几个小时保存一下数据，下次仿真时直接从保存的数据开始继续仿真。

这个案例如何用 UCLI 实现呢，我们带着问题看看 UCLI 常用的功能介绍。

abort (中止评估宏文件)

ace (评估模拟模拟器命令)

alias (为命令创建别名)

assertion (断言 (SVA / PSL) 相关命令)

call 执行 Verilog 系统任务/功能, Verilog PLI 任务/功能或 VHDL 外部过程)

cbug 对 C, C++和 SystemC 源文件的调试支持)

checkpoint 检查点/在当前/给定时间加入仿真设计)

config 显示/设置配置变量的当前设置)

constraints 显示设计信息, 禁用/启用/添加/删除/更改约束, 或提取约束的测试用例)

coverage 评估覆盖命令)

detach\_sim 从 Verdi 脱离 simv, 然后回到 UCLI)

do 评估一个 TCL (宏) 脚本; “源” TCL 命令的超集)

drivers 获取信号/变量的驱动程序信息)

**dump 创建/操作/关闭转储值更改文件信息)**

finish 让工具完成, 然后将控制权返回给 UCLI)

**force 强制或将值存入信号/变量)**

fsdb 适用于 VCS (-MX) 的 Debussy FSDB 命令集)

**get 获取信号/变量的值)**

helpdoc 修改或扩展 ucli 帮助页面)

**listing 显示源文本)**

loaddl 在模拟器空间中加载/卸载用户的动态对象

loads 获取信号/变量的负载信息

lp\_show 本机低功耗 (NLP) 相关命令

memory 从文件加载/写入文件的内存值，或使用给定值初始化内存

msglog 设计和测试平台静态和动态数据记录

**next 推进工具逐步完成任务和功能**

onbreak 指定在宏到达停止点时运行的脚本

onerror 指定在宏遇到错误时运行的脚本

onfail 指定在宏遇到故障时运行的脚本

pause 暂停执行宏文件

power 电源统计相关命令(SAIF):

**release 从使用 “ force” 分配的值中释放变量**

report\_timing 将实例的计时信息报告给指定的文件或控制台

report\_violations 设置各种与 xprop 相关的报告违规

**restart 重新执行工具； UCLI 将使用以前的设置返回到零时间**

**restore 恢复保存在文件中的模拟状态**

esume 恢复执行宏文件

**run xx 运行 xx 时间后停止，不推荐使用**

**run 运行仿真直到遇到\$stop 或者设置的断点**

**run -posedge xx 运行到信号 xx 的上升沿停止**

**run -change xx 运行到信号 xx 变化时停止**

saif 交换活动交换格式相关命令

save 将模拟状态保存到文件中

scope 显示当前的顶层

scope xxx 进入 xxx 模块

scope -up 回到上一层

search 搜索名称与指定模式匹配的设计对象

senv 显示一个或所有 synopsys :: env 数组元素

sexpr 评估工具中的表达式

show 显示当前模块的信号及子模块

show xxx -value -radix hex/bin/dec 以特定形式显示信号值

stack 显示线程信息或移动调用堆栈

start 开始执行工具

start\_verdi 从 UCLI 提示符启动 Verdi

status 显示宏文件堆栈

step 推进工具一条语句

stop 显示已经设置的断点

stop -posedge xx 在信号的上升沿设置断点

stop -condition {信号表达式} 在信号表达式为真的地方设置断点

stop -delete xx 删除断点 xx, xx 为断点数字编号

tcheck 启用/禁用指定实例/端口的定时检查

Tcl Tcl 内置命令的帮助

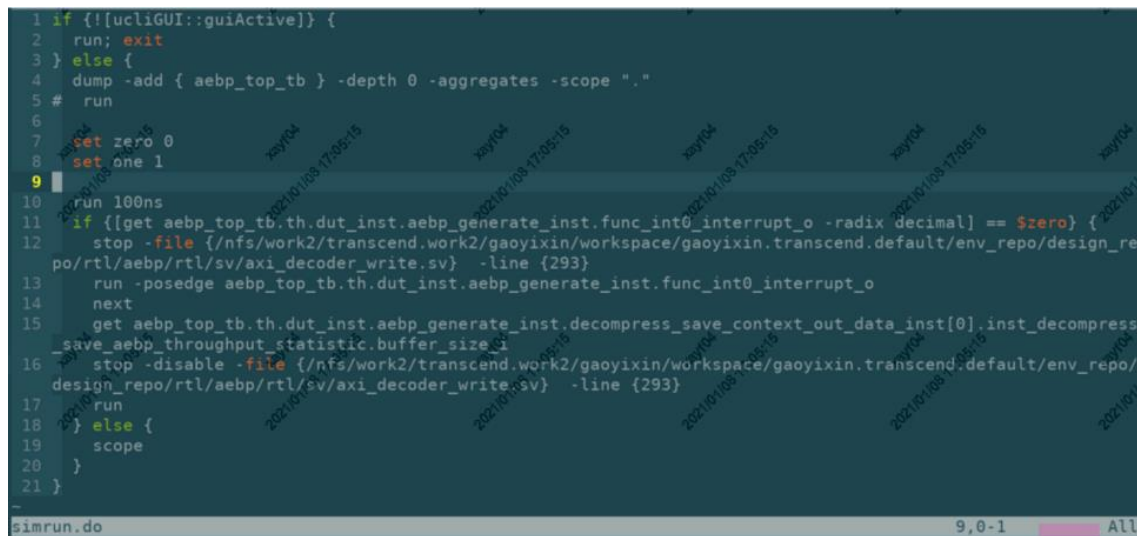
thread 显示线程信息或移动当前线程

unalias 删除一个或多个别名

virtual 创建，删除或显示虚拟对象

xprop 设置或查询 xprop 合并模式

常用的 UCLI 命令我高亮了出来，用这些命令我实现了一个 demo 例子：



```
1 if {[ucliGUI::guiActive]} {
2   run; exit
3 } else {
4   dump -add { aebp_top_tb } -depth 0 -aggregates -scope "."
5   # run
6
7   set zero 0
8   set one 1
9
10  run 100ns
11  if {[get aebp_top_tb.th.dut_inst.aebp_generate_inst.func_int0_interrupt_o -radix decimal] == $zero} {
12    stop -file {/nfs/work2/transcend.work2/gaoyixin/workspace/gaoyixin.transcend.default/env_repo/design_repo/rtl/aebp/rtl/sv/axi_decoder_write.sv} -line {293}
13    run -posedge aebp_top_tb.th.dut_inst.aebp_generate_inst.func_int0_interrupt_o
14    next
15    get aebp_top_tb.th.dut_inst.aebp_generate_inst.decompress_save_context_out_data_inst[0].inst_decompress_save_aebp_throughput_statistic.buffer_size_i
16    stop -disable -file {/nfs/work2/transcend.work2/gaoyixin/workspace/gaoyixin.transcend.default/env_repo/design_repo/rtl/aebp/rtl/sv/axi_decoder_write.sv} -line {293}
17    run
18  } else {
19    scope
20  }
21 }
```

- 第一行第四行是 if - else 判断语句，如果 GUI 界面开启，VCS 工具使能，判断行为非走 else 分支；
- 第四行 dump 命令，depth 为 0 即 dump 所有层级的波形信号，这里我加了 - aggregates， dump MEM 多维数组信号；
- 第十一行又是一个判断，这里我嵌套了 get 命令，如果获取的信号值为 0 则往下走，这里 get 命令加上了参数 -radix 跟 decimal，输出格式就是十进制，通过这个逻辑可以让脚本更加灵活；
- 第十二行第十六行 stop 命令，在 293 行打上了断点，十六行取消断点；
- 第十三行 run 仿真一直跑起来，直到触发该信号上升沿，然后停下来；
- 第十四行 next，相似命令 step，与 VCS gui 界面命令是一致的；
- 第十九行 scope 获取目前的层级。