Tcl 的全称是 Tool Command Language，它是当今 EDA 软件系统中普遍采用的一种脚本语言，如 Synopsys DC 中的 dc_shell-t>和 Synopsys PrimeTime 中的 pt_shell>就是基于这种脚本的，实际上它已经成为了一种工业标准。

以 DC 为例，使用 Tcl 比 dc_shell 具有以下一些优点：

有更多的在线帮助；

支持 lists 和 arrays；

用户可以定义 procedures；

case 结构、string 的操纵与比较、文件操纵；等等。

Synopsys 中使用 Tcl 的工具有：Design Compiler, Formality, PrimeTime, Physical Compiler, Chip Architect 等等。

一、Tcl 语法

Tcl 是基于字符串的命令语言。每一个 Tcl 脚本被看作是一系列命令的组合，用换行符或分号来分隔。每一条命令语句由一个命令和一系列参数组成。

注释：

用#表示这一行后面的内容为注释，例如：

Same line comment.

通配符：

？表示一个字符(PrimeTime only)

* 表示任意多个(包括 0 个)字符

变量：

给一个变量附值的 set 命令，例如：

set x 27

删除一个变量的 unset 命令，例如：

unset x

用$符号对一个变量取值，例如：

set x 27

set y $x

用“\”引用一些特殊符号，例如：

set x 7

set y \$x

注：这里 y 的值是$x，而不是 7。

命令嵌套和引用：

“ ”表示弱引用，嵌套命令和变量替代仍旧有效，例如：

set a 5

set s “temp = data[$a]”

注：s 的值为 temp = data[5]

{ }表示强制引用，其间没有任何替代，例如：

set a 5

set s {temp = data[$a]}
注：s 的值为 temp = data[$a]

用 expr 进行算术运算，例如：
dc_shell-t> set period 10.0
10.0

dc_shell-t> set freq [expr (1 / $period)]
0.1

dc_shell-t> echo “freq = [expr $f req* 1000] MHz”
freq = 100.0 MHz

dc_shell-t> set_load [expr [load_of cba_core/and2a0/A] * 5] \
[all_output]
注：这里的“\”表示续行。

二、Lists 的用法

Lists 是 Tcl 的一个重要部分。Lists 用来表示不同对象的分组。Lists 元素可以包含 strings 或其它 lists。

下面表格列出了相关的命令：

Command	Task
concat	Concatenates two lists and returns a new list.
join	Joins elements of a list into a string.
lappend	Creates a new list by appending elements to a list (modifies the original list).
lindex	Returns a specific element from a list; it returns the index of the element if it is there or -l if it is not there.
linsert	Creates a new list by inserting elements into a list (it does not otherwise modify the list).
list	Returns a list formed from its arguments.
llength	Returns the number of elements in a list.
lrange	Extracts elements from a list.
lreplace	Replaces a specified range of elements in a list.
lsearch	Searches a list for a regular expression.
lsort	Sorts a list.
split	Splits a string into a list.

现举例说明：
dc_shell-t> set L1 {el1 el2 el3}
el1 el2 el3

dc_shell-t>echo $L1
el1 el2 el3

```
dc_shell-t>set    Num_of_List_Elements [llength $L1]
3
```

还可以这样用：

```
dc_shell-t> set a 5
5

dc_shell-t> set b {c d $a [list $a z]}
c d $a [list $a z]

dc_shell-t> set b [list c d $a [list $a z]]
c d 5 {5 z}
```

在 DC 中可以这样配置 link_library：

```
dc_shell-t> set link_library {*}
*

dc_shell-t> lappend link_library tc6a.db opcon.db
* tc6a.db opcon.db

dc_shell-t> echo $link_library
* tc6a.db opcon.db
```

三、控制流(Control Flow)命令的使用：

if 语句，例如：

```
if {$x ==0} {
   echo “Equal”
} elseif {$x > 0} {
   echo “Greater”
} else {
   echo “Less”
}
```

while 语句，例如：

```
set p 0
while {$p <= 10} {
     echo “$p squared is: [expr $p*$p]”    incr p
}
```

循环语句，例如：

```
for {set p 0} {$p <= 10} {incr p} {
echo   “$p squared is: [expr $p * $p]”
}
```

foreach 语句，例如：

```
set i 1
foreach value {1 3 5 2} {
    set i [expr $i * $value]
}
echo $i
30
```

四、过程(Procedures)

用户通过过程可以扩充 Tcl 的命令，并且可以有不同的参数。

语法为：

proc *name arguments body*

name：过程名

arguments：过程的参数，可以为空

body：过程的脚本

例如：

```
dc_shell-t> proc plus {a b} { return [expr $a + $b]}
dc_shell-t> plus 5 6
11
```

五、DC 中的一些 Tcl 说明

在 UNIX 提示符下启动 DC 的 Tcl 模式：

```
UNIX%   dc_shell –tcl_mode
```

每个设计中都有以下一些目标对象(objects)：

designs, cells, references, ports, pins, clocks, nets

其中每个目标对象都有其相应的属性，例如：

ports 拥有的属性有：direction, driving_cell, max_capacitance, others…

designs 拥有的属性有：area, operating_conditions_max, max_area, others…

在标准的 Tcl 中则没有这些内容。

集合(Collection)：

以下是在 DC-Tcl 中生成集合的一部分命令：

```
get_cells
get_clocks
get_designs
get_libs
get_nets
get_pins
get_ports
all_clocks
```

all_designs
all_inputs
all_outputs
all_registers
举例说明：

```
# Constrain a design for timing, using a time budget
set CLK_PER    10.0;          # clock period (ns)
set time_budget    40.0;       # percentage of clock period allowed for input/output logic
# calculate intermediate variables
set IO_DELAY    [expr ( (1-$time_budget/100.0) * $CLK_PER)]
set all_except_clk    [remove_from_collection [all_inputs] [get_ports Clk] ]
# constrain the design for timing
create_clock –period $CLK_PER –name MY_CLOCK [get_ports Clk]
set_input_delay $IO_DELAY –max –clock MY_CLOCK $all_except_clk
set_output_delay $IO_DELAY –max clock MY_CLOCK [all_outputs]
```

集合过滤(Filtering Collections)：
用 filter_collection 命令，例如：

```
filter_collection [get_cells] “ref_name == AN2”;
```

注：从 cell 集合中去除名字为 AN2 的 cell

六、RISC_CORE 的 Tcl 脚本
runit.tcl 文件：

```
#
set_min_library core_slow.db -min_version core_fast.db
# Directory Structure
set source_dir unmapped
set script_dir scripts
set mapped_dir mapped
set reports_dir reports

# List of designs to be compiled
set designs_to_build {RISC_CORE}

foreach module $designs_to_build {
    set fname $source_dir/$module.db
    read_db $fname
    set current_design $module
    link
    source scripts/top_level.tcl
    uniquify
    compile

    set fname $mapped_dir/$module.db
```

```
    write -hierarchy -output $fname
    set fname $reports_dir/$module.rpt
    report_constraint -all_violators > $fname
    report_timing > reports/default_timing.rpt
    report_timing -input_pins > reports/pins_timing.rpt
    report_timing -nets > reports/nets_timing.rpt
    report_timing -delay min > reports/min_timng.rpt
}
```

top_level.tcl 文件：

```
#
# Create user defined variables
set clk_port [get_ports Clk]
set clk_period 4.0
set clk_skew 0.3
set drive_cell buf1a3
set drive_pin "Y"
set max_input_load [load_of ssc_core_slow/buf1a2/A]
set clk_to_q 1.5
set output_delay 1.5
set all_ins_ex_clk [remove_from_collection [all_inputs] [get_ports Clk]]

# Reset the design
reset_design

# Operating Environment
create_clock -period $clk_period -name my_clock $clk_port
set_clock_uncertainty 0.3 [get_clocks my_clock]
set_dont_touch_network [get_clocks my_clock]

# Time budget
set_input_delay $clk_to_q -max -clock my_clock $all_ins_ex_clk
set_output_delay $output_delay -max -clock my_clock [all_outputs]

# Load Budget (Assumes output ports will only fan out to 3 other designs)
set_driving_cell -lib_cell $drive_cell -pin $drive_pin $all_ins_ex_clk
#set_max_capacitance $max_input_load $all_ins_ex_clk
set_load $max_input_load [all_outputs]
```