# 基本命令

## vlib work

该命令的作用是在该目录下***建立一个work目录***，请注意不要用操作系统来新建一个work的文件夹，因为用操作系统建立的work文件夹并没有ModelSim SE自动生成的\_info文件。

## vmap work work

该命令的作用是将目前的逻辑工作库work和实际工作库work***映射对应***。

## vlog camera.v  camera\_tb.v

该命令的作用是***编译***这些文件，要注意的是文件可以单独分开编译，但是一定要先编译被调用的文件。假如是VHDL文件就可以用vcom file1,file2命令来编译。

## vsim camera\_tb

***仿真***命令, 注意后面的参数必须为camera\_tb.v文件中的模块名。

## add wave /camera\_tb/\*

该命令的作用是将testbench文件camera\_tb.v中模块camera\_tb下所有的信号变量***加到波形文件***中去，注意在“\*”前不要加空格。这时候你也可以看到wave文件被打开。当然也可以***单个信号的添加***，例如添加时钟：add wave clk 等等。

## run 2000

该命令的作用是***运行2000个单位时间的仿真***。也可以用run –all命令来一直仿真下去。这时候就可以在wave窗口文件中看到你的仿真结果。

## View

当然也可以***观察其它窗口的结果***，用view命令显示。view \* 观察包括signals、wave、dataflow等窗口文件。也可以分别打开。例如用view signals来观察信号变量。

  ModelSim SE6..2b还有一个很好用的功能。就是可以看整个文件所形成的***数据流程***，各个模块之间的***逻辑联系***。具体方法是在仿真后执行命令  view datalflow 就可以打开dataflow文件，在dataflow的窗口菜单中点击add中的view all nets就可以观察到各个模块之间的逻辑联系，模块一般都为initial模块、always模块、assign模块等等。点击中一个模块，则这个模块变为红色。这时候在view菜单下点击show wave就可以在窗口下方弹出wave窗口，不同的是这个wave窗口所显示的信号变量仅为点击中的模块所包括的信号变量，这时候也可以点击仿真run –all小图标来仿真有关这个模块的输入输出关系。

## quit -f/-force/-sim

该命令分别是***退出ModelSim***（-f/-force）和***退出仿真***（-sim）。

## 批处理文件

但是在ModelSim SE中开可以执行一种批处理文件，就是file.do文件，相当于DOS中的.bat文件。你可以用***批处理方式来使你仿真简单化***，具体做法为，将你所要执行的命令编辑在一起，以上面所讲为例，我可以编辑一个camera.do文件，文件内容如下：  
vlib work                                 // 建库  
vmap work work                      // 映射  
vlog camer.v camera\_tb.v         // 编译  
vsim camera\_tb                      // 仿真（模块名称）  
add wave/camera\_tb/ \*            // 将camera\_tb下的所有信号变量加入到wave窗口中，注意”\*”前必须有空格  
run 2000        　         // 或者用run –all等。  
view  dataflow             // 用navigate  nets观察dataflow，不想观察就可以不加这条指令

将上述内容保存后，每次用命令do  camera.do 就可以自动执行想要的仿真动作。

### .bat的批处理文件

用于在Windows cmd下运行整个ModelSim仿真。

源码如下：

vlib work

vlog -f vflist

vsim -c -novopt work.counter\_updown\_tb -do "run -all"

pause

camera.do 保存文件为camera.bat。在工程目录下直接点击sram\_controller.bat运行即可。

**vsim -c -do sim.do counter -wIf counter.wlf**

**这里-c表示不打开GUI界面，-wlf表示保存仿真结果为wlf文件**

### 在GUI下查看结果

如果在counter.wlf保存了仿真结果，就可在GUI下通过-view参数查看。

注意：

确定你的环境变量的PATH设置的路径为当前ModelSim的路径。

a）在系统提示符后输入vsim-view counter.wlf。

打开GUI和一个名为“counter"资料组标签（图8-1）。

b）右击counter实例并选择Add>To Wave>All items in region。

波形窗口显示波形。

## 整理：

### 1、编译（它们的效果是等效的）：

    vlog div.v div\_tb.v

    vlog -work work div.v div\_tb.v

   或者将它们分开进行单独编译

### 2、仿真（效果等效）：

    vsim div\_tb

    vsim -lib work div\_tb

    vsim work.div\_tb

    sim -L D:/Modelsim/Installfiles/altera/altera\_mf work.videoin\_tb

### 3、添加波形

    add wave/div\_tb/ \*

    add wave sim:/div\_tb/\*       //\*前面没有空格

也可以单独添加波形（可以将它们直接写成一个wave.do文件，以便直接执行do wave.do就可以完成各种波形的编辑设置）：

add wave -divider Input  
add wave sim:/divider\_tb/rst\_n  
add wave sim:/divider\_tb/clk\_in  
add wave -divider Ouput  
add wave sim:/divider\_tb/clk\_out  
#配置wave相关命令  
WaveRestoreCursors {{Cursors 1} {200ns} 0} {{Cursors 2} {400ns} 0}  
configure wave -rowmargin 4  
configure wave -childrowmargin 2  
configure wave -namecolwidth 150  
configure wave -valuecolwidth 100  
configure wave -signalnamewidth 0  
configure wave -justifyvalue left  
configure wave -snapdistance 10  
configure wave -datasetprefix 0  
configure wave -gridoffset 0  
configure wave -gridperiod 1  
configure wave -griddelta 40  
configure wave -timeline 0

update

WaveRestoreZoom {0 ns} {1000 ns}

## +incdir+：

如：vlog +incdir+YOUR\_SOURCE\_PATH  foo.v

  +incdir+YOUR\_SOURCE\_PATH 选项是指在verilog文件中出现`include "xxx.v"

时，包含文件的搜索路径。

缺省是搜索当前路径，然后是 YOUR\_SOURCE\_PATH 指定的路径。

## +define+：

+define+<macro\_name>[=<macro\_text>]

  允许用户在命令行中定义宏定义，等效于编译器指令:

`define <macro\_name> <macro\_text>

  用户可以指定多个宏定义，如下：

vlog +define+one=r1 +two=r2 +three=r3 **test.v**

  命令行的宏定义会**覆盖**在源文件中用`define定义的**相同名字的宏定义**

## -c -l

如：vsim -c -l vsim.log -do ./YourDo.do -L ./work work.foo

   开始仿真:

-c 选项让vsim工作在commandline模式；

**-l 选项是输出log文件到vsim.log；**

**-do 选项是开始仿真后运行tcl脚本文件**；

-L 选项是指定工作逻辑库；

work.foo是仿真的top level module。

## .main clear

清屏transcript

# 二、do文件编写

## 1、编写好源文件。

包括sram\_controller.v和它的Testbench文件sram\_controller\_tb.v

## 2、编写.do文件（sram\_controller.do），内容如下：

### # Create the work library

vlib work

vmap work work

### # Compile the verilog files

vlog  sram\_controller.v

vlog  sram\_controller\_tb.v

### # Run simulation

vsim  sram\_controller\_tb

### #Add all the top signals

add  wave \*

### #The time of run

run 1us

do文件的注释是由#开始的，但不可以在代码行后面添加，只能另起一行。

如果在仿真的时候要修改.do文件，需要现在modelsim里运行quit -sim，退出仿真，然后修改.do文件，再保存，然后再重新执行do filename.do指令即可。

## 3、打开ModelSim

修改工作路径到当前工程目录下。在命令窗口中输入do sram\_controller.do，软件便开始自动运行进行仿真了。

## 4、如果对设计文件进行了修改，需要进行重新仿真。

那么，只要在命令行重新输入do sram\_controller.do回车即可完成仿真，非常方便，一劳永逸。

## 5、.bat的批处理文件

用于在Windows cmd下运行整个ModelSim仿真。

源码如下：   
vsim -do sram\_controller.do 保存文件为sram\_controller.bat。在工程目录下直接点击sram\_controller.bat运行即可。

## 6、add wave \*

这条命令就将test bench顶层的所有信号加入到wave窗口中。对我们来说，在调试阶段，有很多底层信号都是想观测的，所以需要再手动修改一下命令。比如我们想将实例中的状态机信号加进去，可以在do文件中增加一条命令：add wave /sram\_controller\_tb/i1/state 。当然也可以通过软件操作完成此功能。

## 7、借助软件的UI操作来查看并记录这些命令。

例如，为了区分仿真波形窗口中的各种信号线，需要信号波形作设置，如不同信号线的颜色、显示基数、显示方式等要有区别，这时就需要在仿真波形窗口单独对每一个信号线手动进行设置，这对于不断修改源代码然后再不断地进行仿真来说，非常麻烦。我们可以先进行一次仿真，然后对颜色等进行设置，最后保存波形格式为wave.do。

这时候建议将前面的sram\_controller.do文件**拆分为sim.do和wave.do**，将波形添加和设置等操作放到wave.do中，分别运行这两个do 文件即可完成仿真，且如上图波形的颜色设置等不需要重复劳动。

总之，如果你对命令不熟悉，可以先进行UI操作，观察命令窗口中对应的命令，就能理解命令的操作。其他还有很多TCL命令，这里就不赘述了。

## 8、综合

quit -sim *#退出仿真*

.main clear

*# Create the work library*

vlib work

vmap work work

*# Compile the verilog files*

vlog key\_counter\_scan.v

*#编译.v文件，文件名用自己的.v文件替代*

vlog key\_counter\_scan\_TB.v

*# Run simulation*

vsim -voptargs=+acc key\_counter\_scan\_TB

log -r /\*

log -r /\*：对所有的信号记录，运行这个命令后即使在仿真前没有把信号加入wave窗口，仿真完成后直接加入wave窗口就可以查看波形，比较方便，但是缺点是当工程较大和仿真时间很长时仿真速度较慢，占用内存也较大。

+acc意思是设计中所有的信号都提供入口，可以观察，而log命令的意思就是在运行仿真的时候信号不用加入wave窗口在开始仿真后直接可以看波形，意思就是他记录了所有的信号波形，类似于debussy中的fsdbDump这个命令。

 如果有 log -r /\* 而没有 +acc，可能有一些信号看不到

*#仿真，不优化，否则没有波形*

*#Add all the top signals*

add wave \*

*#The time of run*

run 800us

## 9、常用的一些命令：

退出当前仿真quit -sim  
vlib work  
#编译修改后的文件，我这里把设计文件和仿真文件分开放了，所以写两个。vlog "../Src/\*.v"  
vlog "../Sim/\*.v"  
#开始仿真  
vsim -voptargs=+acc work.tb\_Cordic\_Cos\_Sin  
#添加指定信号  
#添加顶层所有的信号  
# Set the window types  
# 打开波形窗口  
view wave  
view structure  
# 打开信号窗口  
view signals  
# 添加波形模板  
add wave -divider {tb\_Cordic\_Cos\_Sin}  
add wave tb\_Cordic\_Cos\_Sin/\*  
add wave -divider {tb\_Cordic\_Cos\_Sin}  
add wave tb\_Cordic\_Cos\_Sin/uut/\*  
.main clear  
#运行xxms  
run 100us  
**不必每次都输入指定do文件的命令**，在脚本控制台直接按上键就会**显示上一个执行过的命令**。

## 10、模板

**#quit last sim**

quit -sim

**# Create the work library**

vlib work

**#vmap the work lib**

vmap work work

**#include the head files**

vlog +incdir+ *YOURPATH* sfifo\_def.v

#eg: vlog +incdir+ sfifo\_def.v

**# Compile the verilog files**

vlog -work work sfifo.v

vlog -work work sfifo\_tb.v

#Or you can merge them, but be sure to compile the called file first

vlog -work work sfifo.v sfifo\_tb.v

#You can also ignore the ‘-work work’, such as vlog sfifo.v sfifo\_tb.v

**# Run simulation**

vsim sfifo\_tb

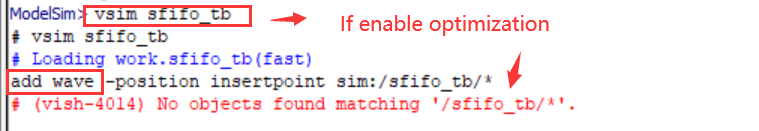
vsim -lib work sfifo\_tb

vsim work. sfifo\_tb

**#prohibit to optimize**

vsim -novopt work.sfifo\_tb

vsim -voptargs=+acc -novopt work.sfifo\_tb

**#Attention:** If you don’t add ‘-novopt’ to disable optimization in your commands, there will be error when add wave to window: *# (vish-4014) No objects found matching '/sfifo\_tb/\*'*. 

#specified time pricision is 1ns

vsim -novopt work.sfifo\_tb -t 1ns

**# Set the window types**

# Open the wave window

view wave

#Open the instance structure window

view structure

# Open the signals list window

view signals

#Open the source window

view source

#Open the list window

view list

#Open the variables window

view variables

#Open the dataflow window

view dataflow

#Add all signals to wave window

add wave -r /\*

**#Attention:** -r should be followed by a space

**#add the specific test signal to wave window, eg: add wave sim:/ sfifo\_tb /clock**

add wave sim:/ sfifo\_tb /signal

**#Attention:** ‘sim:’ can be ignored, which means ‘add wave / sfifo\_tb /signal’ is also right

**#add the tb signals to wave window**

add wave sim:/sfifo\_tb /\*

add wave /sfifo\_tb /\*

add wave \*

**#Display in hexadecima**l **or binary or decimal or unsigned or others**

add wave -hex/-decimal/-unsigned/-binary \*

**#format: Logic/Literal/Event/analogautomatic/analogcustom**

#add wave -noupdate -format Logic -radix decimal -color Red /sfifo\_tb/sfifo1/clock

#add wave -radix unsigned -color pink -format analogautomatic sim: sfifo\_tb/sfifo1/clock

add wave -unsigned sim: sfifo\_tb/sfifo1/clock

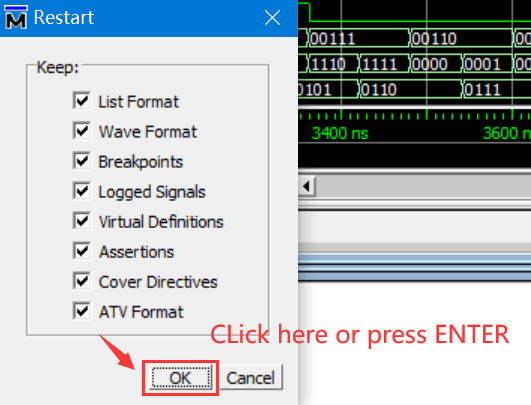
add wave -noupdate -color pink -format Analog-Step -height 74 -max 4094.9999999999995 -radix unsigned sfifo\_tb/sfifo1/clock

**#delete signal**

delete  wave  /test/i

**#restart simulation, then click OK or hit ENTER**

restart



**#run simulation**

run 100 us

run -all

**#output results**

write list counter.lst

**#quit -f/-force to quit Modelsim**

**#quit -sim to quit simulation**

quit -f/-force/-sim

#clear transcript window

.main clear

#配置wave相关命令

WaveRestoreCursors {{Cursors 1} {200ns} 0} {{Cursors 2} {400ns} 0}

configure wave -rowmargin 4

configure wave -childrowmargin 2

configure wave -namecolwidth 150

configure wave -valuecolwidth 100

configure wave -signalnamewidth 0

configure wave -justifyvalue left

configure wave -snapdistance 10

configure wave -datasetprefix 0

configure wave -gridoffset 0

configure wave -gridperiod 1

configure wave -griddelta 40

configure wave -timeline 0

update

WaveRestoreZoom {0 ns} {100 us}

# **三、如何让状态机显示状态的名字**

在使用Verilog编写有限状态机等逻辑的时候，状态机的各个状态通常以参数表示，但当使用ModelSim仿真的时候，状态机变量在wave窗口中以二进制编码的形式显示，如上面第一部分中6添加状态机的状态，这种显示形式不是很直观，但我们可以使用ModelSim提供的命令将状态机变量以“文本”形式的参数名显示，从而有利于调试。

## 1.定义枚举变量

首先使用ModelSim的virtual type命令定义一个新的枚举类型（FSM\_TYPE）：

virtual type {   
{0x0 idle} {0x1 write\_U9} {0x2 write\_U6} {0x4 start\_1} {0x8 rdU6\_wrU5} {0x10 update\_1} {0x20 read\_end\_1} {0x40 waiting\_1}   
{0x80 ARM\_com\_1} {0x100 start\_2} {0x200 rdU5\_wrU6} {0x400 update\_2} {0x800 read\_end\_2} {0x1000 waiting\_2} {0x2000 ARM\_com\_2}   
} FSM\_TYPE

注意：状态变量之间需要一个空格。

## 2.类型转换

然后我们将需要显示的信号（/sram\_controller\_tb/i1/state，注意信号在wave窗口中的完整名字）进行类型转换，转换成一个新的信号（state\_new）

virtual function {(FSM\_TYPE)/sram\_controller\_tb/i1/state} state\_new

## 3.添加信号

最后我们将新的信号加入到wave窗口中

add wave -color pink /sram\_controller\_tb/i1/state\_new

# 四．交互式命令

通过在主窗口的命令窗口输入命令来实现，具有更好的调试和交互功能，提供多种指令，既可以是单步指令，也可以构成批处理文件，用来控制编辑、编译和仿真流程；

常见交互式命令如下：

## 1.force-repeat指令

指令格式：force 开始时间开始电平值，结束电平值忽略时间（即0电平保持时间） -repeat 周期

force clk 0 0,1 30 -repeat 100 表示强制clk从0时间单元开始，起始电平为0，结束电平为1，0电平保持时间为30个默认时间单元，周期为100个默认时间单元，占空比为70%。

指令功能：每隔一段的周期重复一定的force命令，用来产生时钟信号，也可用来产生周期的输入信号，如01010101,00110011等。

force clk 0 10,1 20 -r 20

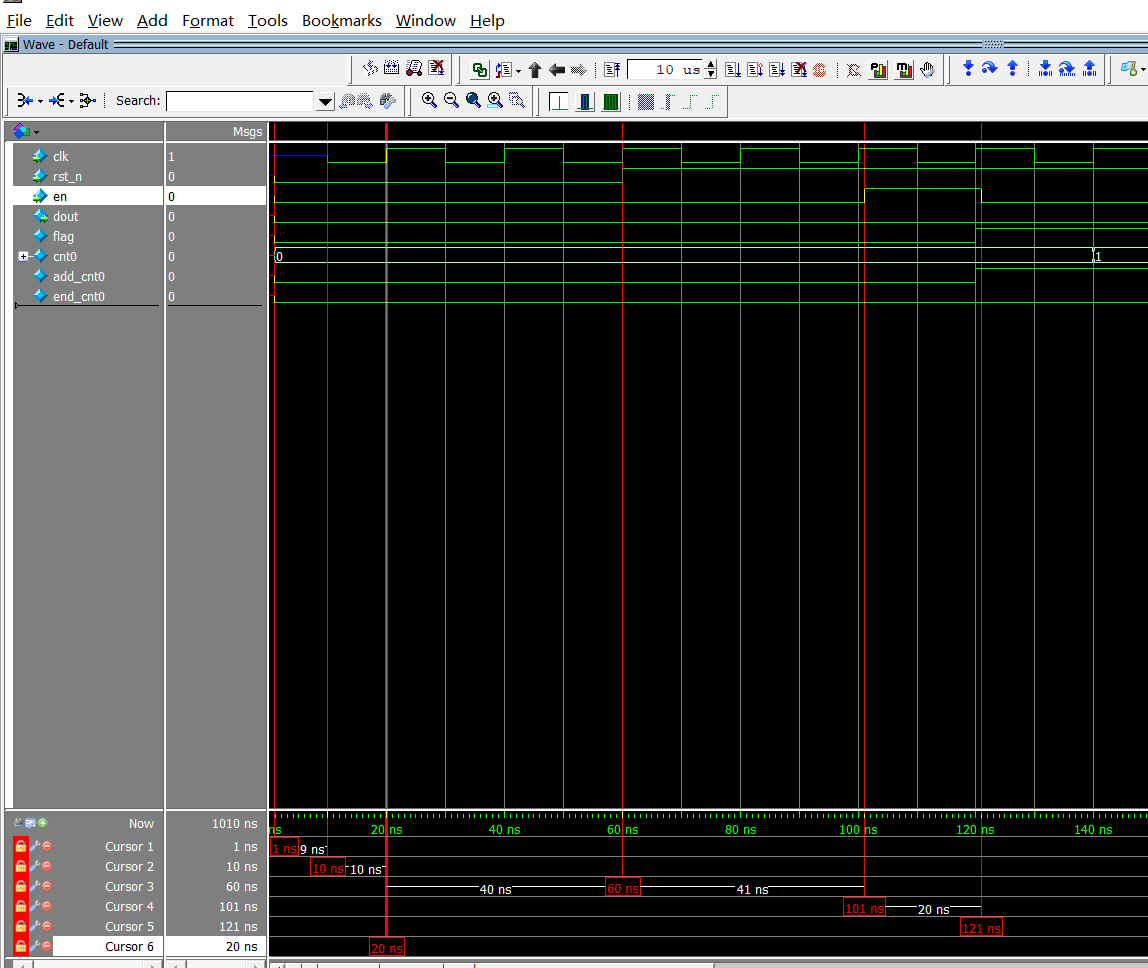
解释：在第10ns时，clk = 0，在第20ns，clk=1，-r 20从第20个时间单位单位开始重复循环clk值的变化规律。

force rst\_n 0 1,1 60

解释：在第1ns，rst\_n = 0，在第60ns，rst\_n = 1;

force en 0 1,1 101,0 121

解释: 在第1ns，en = 0，在第101ns，en = 1，在第121ns，en = 0;



## 2.force指令

指令格式：force item\_name value time,value time；item\_name为端口信号或内部信号，支持通配符号，但只能匹配一个；value不能默认，time，可选项，支持时间单元；

force din 16#40900000 从**当前时刻起**给din赋值16进制40900000；

force bus 16#F @100ns 在**100ns时刻**给bus赋值16进制F；

force clr 1 100 经历100个默认时间单元延迟后为clr赋值1；

force clr 1,0 100 表示clr赋值1后，经历100个默认时间单元延迟后为clr赋值为0；

## 3.run指令

指令格式：run timesteps time\_unit,timesteps时间步长，time\_unit时间单元，可以是fs、ps、ns、us、ms、sec；

指令功能：运行（仿真）并指定时间及单元；

run 100，表示运行100个默认时间单元；

run 2500ns，表示运行2500ns；

run -all，表示运行全过程；

run -continue，表示继续运行

## 4.force-cancel指令

指令格式:force-cancel period

指令功能：执行period周期时间后取消force命令；

force clk 0 0,1 30 -repeat 60-cancel 1000，表示强制clk从0时间单元开始，直到1000个时间单元结束；

## 5.view

参考二. 10. 模板