

Model*Sim*概览



SCHOOL OF SOFTWARE ENGINEERING OF USTC

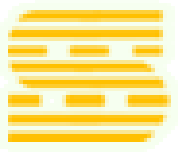


ModelSim 仿真工具

1. 由Model Technology技术公司开发
(Mentor Graphics的子公司)
2. 工业上最通用的仿真器之一
3. 可在Verilog 和 VHDL仿真
4. 具备命令行的操作方式
5. 具有分析代码的能力，可以看出不同的代码段消耗资源的情况，从而可以对代码进行改善，以提高其效率。



Model Technology
A MENTOR GRAPHICS COMPANY



ModelSim 产品

1. Model*Sim*/VHDL 或者 Model*Sim*/Verilog
 - OEM
2. Model*Sim*/LNL
 - 许可 Verilog 或者 VHDL，但是不同时许可
3. Model*Sim*/PLUS
 - 设计者能立刻混合仿真 Verilog 和 VHDL
4. Model*Sim*/SE
 - 首要的版本
 - PLUS的所有功能连同附加功能



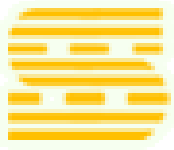
ModelSim OEM 功能

1. 提供完全的标准

- '87 VHDL
- '93 VHDL
- IEEE 1364-' 95 Verilog
- SDF 1.0 - 3.0
- VITAL 2.2b
- VITAL '95

2. 易用的界面

- 通用的平台



用Model*Sim* 仿真



Model Technology
A MENTOR GRAPHICS COMPANY

SCHOOL OF SOFTWARE ENGINEERING OF USTC



Model 技术公司的 ModelSim

1. ModelSim的主窗口（Main window）包括菜单栏、工具栏、工作区和命令行操作区。
2. 在菜单栏**View**下可以打开:source window、list window、wave window、structure window、 signal window、 dataflow window、 process window、 viarables window等窗口。
3. 在**帮助**菜单里有SE的帮助文件和Tcl的帮助文件，是学用ModelSim很好的帮手。

启动可以命令行输入: **modelsim.exe**

SCHOOL OF SOFTWARE ENGINEERING OF USTC



Model 技术公司的 ModelSim

ModelSim SE PLUS 6.5

File Edit View Compile Simulate Add Objects Tools Layout Window Help

Layout: NoDesign

Library

Name	Type	Path
work	Library	work
counter	Module	E:/EDA/experiments
test_counter	Module	E:/EDA/experiments
a	Library	a
floatfixlib	Library	\$MODEL_TECH/./fx
mtiAvm	Library	\$MODEL_TECH/./av
mtiOvm	Library	\$MODEL_TECH/./ov
mtiPA	Library	\$MODEL_TECH/./pa
mtiUPF	Library	\$MODEL_TECH/./up
sv_std	Library	\$MODEL_TECH/./sv
vital2000	Library	\$MODEL_TECH/./vi
ieee	Library	\$MODEL_TECH/./ie
modelsim_jlib	Library	\$MODEL_TECH/./m
std	Library	\$MODEL_TECH/./st
std_developerskit	Library	\$MODEL_TECH/./st
synopsys	Library	\$MODEL_TECH/./sy
verilog	Library	\$MODEL_TECH/./ve

Objects

Name	Value
------	-------

Processes (In Region)

Name	Type (filtered)
------	-----------------

E:/EDA/experiments/lab2/tcounter.v

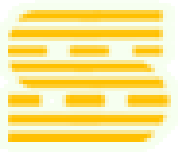
```
Ln#
10 module test_counter;
11
12 reg clk, rst;
13 wire [7:0] count;
14
15 counter #(5,10) dut (count,clk,rst);
16
17 initial // Clock generator
18 begin
19     clk = 0;
20     #10 forever #10 clk = !clk;
21 end
22
23 initial // Test stimulus
24 begin
25     rst = 0;
26     #5 rst = 1;
27     #4 rst = 0;
28     #50000 $stop;
29 end
```

Library Project

Transcript

```
vlib a
vmap a a
# Modifying E:/EDA/experiments/lab2/1.mpf
ModelSim>
```

Project: 1 <No Design Loaded> <No Context>



ModelSim实现方法

1. 交互式的命令行 (Cmd)

- 唯一的界面是控制台的命令行, 没有用户界面

2. 用户界面 (UI)

- 能接受菜单输入和命令行输入
- 课程主要讨论

3. 批处理模式

- 从DOS或UNIX命令行运行批处理文件
- 不讨论



基本仿真步骤

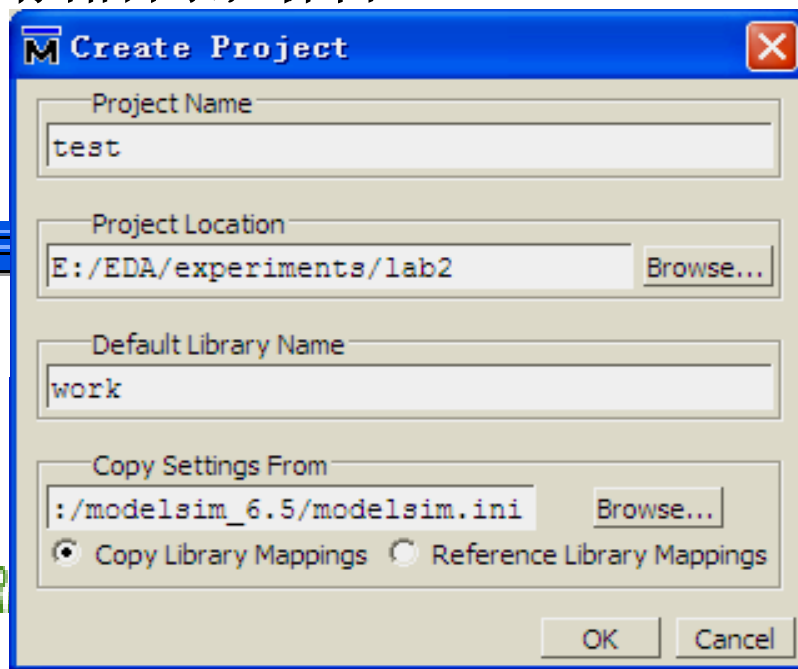
1. Create a Project

□ 第一次打开ModelSim会出现Welcome to ModelSim对话框，选Create a Project,或者选取File\New\Project,然后会打开Create Project对话框。

□ 在Create Project对话框中，填写test作为Project Name；选取路径Project Location作为Project文件的存储目录；保留Default Library Name设置为work。

Transcript

```
vlib a  
vmap a a
```





ModelSim 库

- 包含编译设计单元的目录
 - VHDL 和 Verilog 都被编译到库里
- 两个类型
 - Working (缺省值 *work*)
 - 包含当前被编译的设计单元
 - 编译前必须建立一个working库
 - 每个编译只允许一个
 - Resource
 - 包含能被当前编译引用的设计单元
 - 在编译期间允许多个
 - VHDL库能通过LIBRARY和USE子句引用



基本仿真步骤

以上三项填写完毕后，点击“OK”按钮，会在指定的位置处创建工程及工作库



在工作库文件夹内，有一个名为“_info”的特殊格式文件，如图所示：

地址 (Q) E:\EDA\experiments\lab2\work



这个文件指定了此文件夹为ModelSim的工作库文件夹。在编译步骤内生成的所有设计单元都会被添加到工作库文件夹内。



基本仿真步骤



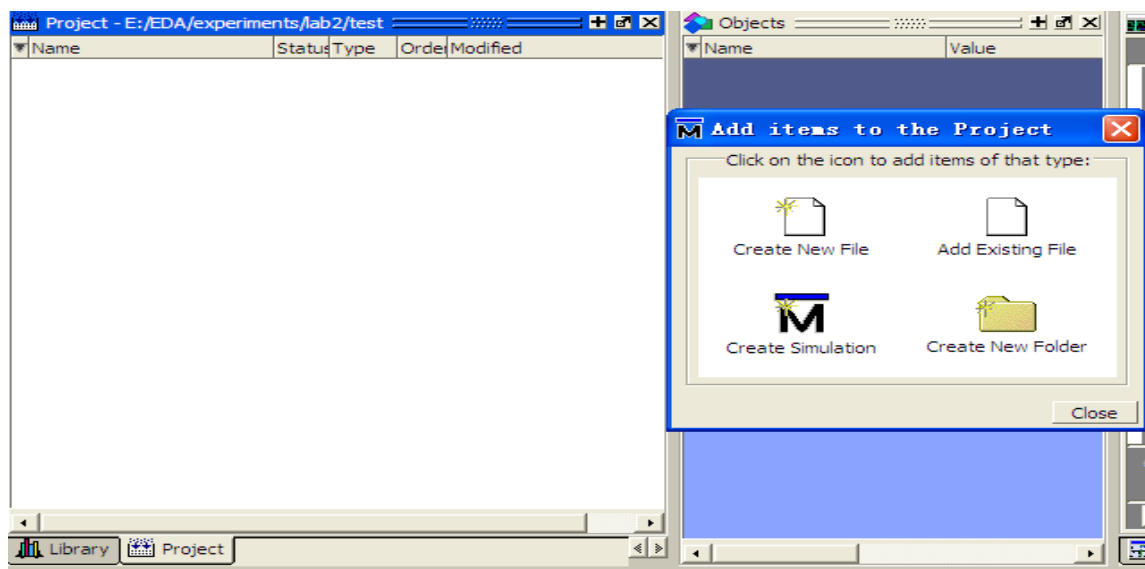
● Where

- `_primary.dat` - Verilog module 或 VHDL entity 的编码格式
- `_primary.vhd` - Verilog 端口的 VHDL entity 陈述
- `verilog.asm` - 执行代码文件



基本仿真步骤

- 选取OK,会看到工作区出现**Project and Library Tab**。



“**Create New File**”项：在工程中创建新的设计文件。

“**Add Existing File**”项：把已经存在的设计文件加入到工程中。

“**Create Simulation**”项：在工程中创建仿真配置文件。

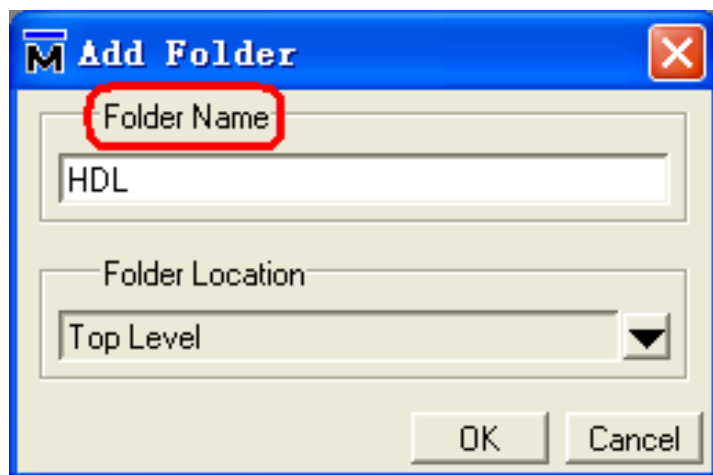
“**Create New Folder**”项：在工程中创建新的文件夹。



基本仿真步骤

- **Create New Folder** 创建文件夹。（*此步骤也可以不要，不过推荐采用此步骤，便于文件管理*）

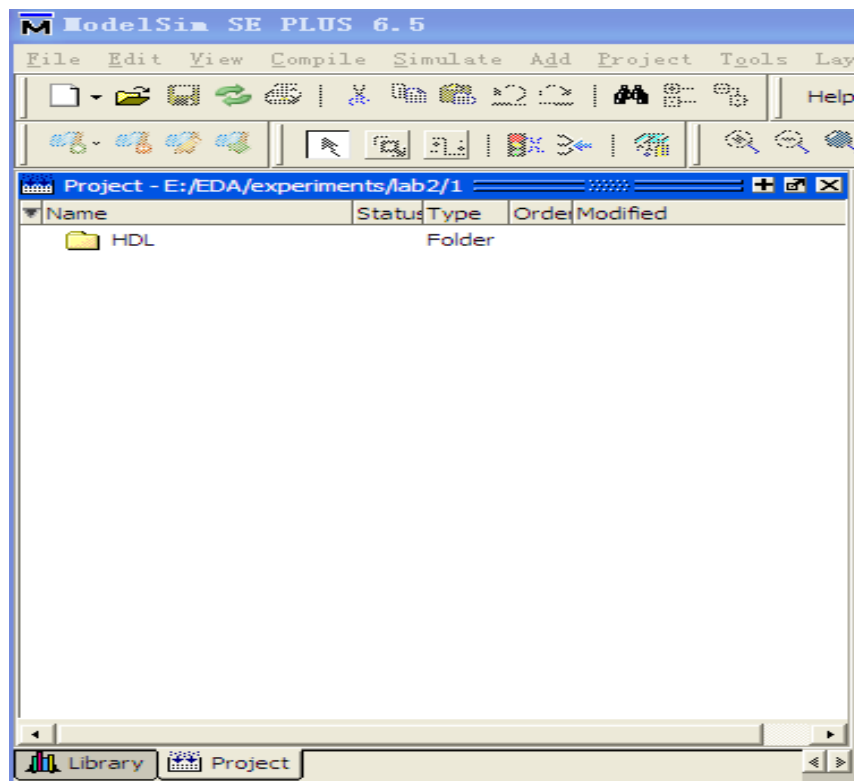
选择“Create New Folder”项，在弹出的对话框中输入要创建的文件夹的名字，如图所示：

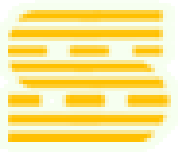


- 点击“OK”按钮，会在“**Workspace**”列表里显示出新创建的文件夹。如图所示：



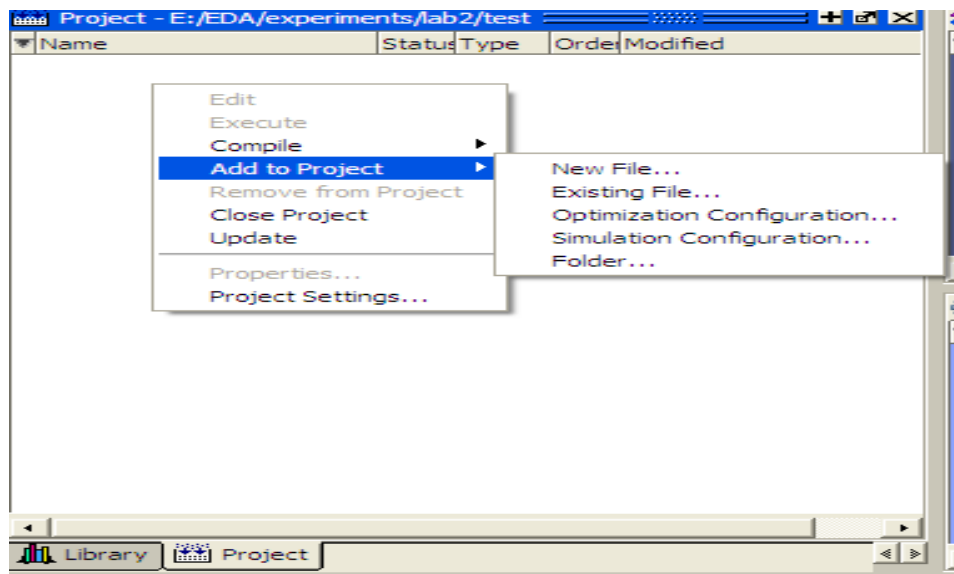
基本仿真步骤





基本仿真步骤

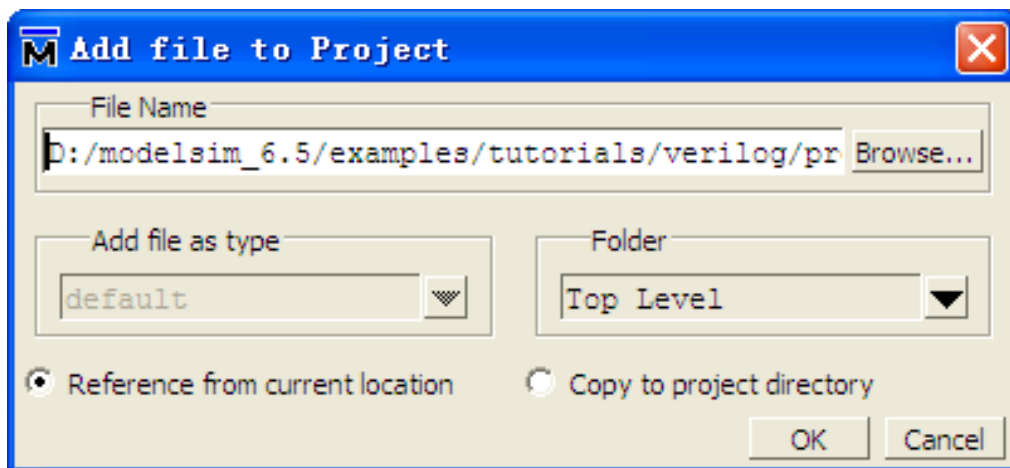
□ 下一步是添加包含设计单元的文件，在工作区的Project page中，点击鼠标右键，选取**Add File to Project**。



□ 点击Add File to Project对话框中的**Browse**按钮，打开ModelSim安装路径中的 **example** 目录，选取**counter.v**和**tcounter.v**，再选取**Reference from current location**，然后点击 OK。

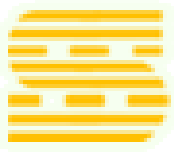


基本仿真步骤



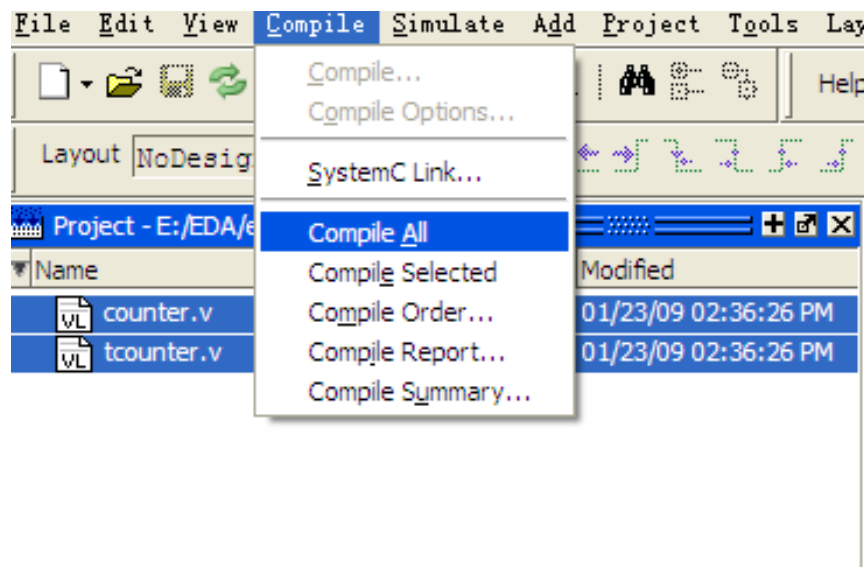
“**Reference from location**”项：表示只将设计文件与工程关联起来。

“**Copy to project directory**”项：表示将设计文件复制一份到工程目录下。存放的具体位置由“Folder”项指定。



基本仿真步骤

□ 在工作区的Project page中，单击右键，选取**Compile All**，顺序并不重要（*命令行: `vlog counter.v tcounter.v`*）





基本仿真步骤

错误信息在 Main
窗口显示

```
ModelSim ALTERA 5.3d Altera
File Edit Design View Run Macro Options Window Help
100000
# Region: /lab2_fifo/u2/lpm_ram_dq_component
# WARNING: E:/Quartus_MS_test/lab2/Verilog/lab2_cnt.v(54): [TFMPC] - Too few port connections.
# Region: /lab2_fifo/u1/lpm_counter_component
vlog -reportprogress 300 -work work {E:/Quartus_MS_test/lab2/Verilog/lab2_fifo_tb.v}
# Model Technology ModelSim ALTERA vlog 5.3d Altera Compiler 2000.04 May 17 2000
# -- Compiling module lab2_fifo_tb
# ERROR: E:/Quartus_MS_test/lab2/Verilog/lab2_fifo_tb.v(43): near "tdata": expecting: ';'
# ERROR: E:/Quartus_MS_test/lab2/Verilog/lab2_fifo_tb.v(45): near "endmodule": expecting: END
vlog -reportprogress 300 -work work {E:/Quartus_MS_test/lab2/Verilog/lab2_fifo_tb.v}
# Model Technology ModelSim ALTERA vlog 5.3d Altera Compiler 2000.04 May 17 2000
# -- Compiling module lab2_fifo_tb
# ERROR: E:/Quartus_MS_test/lab2/Verilog/lab2_fifo_tb.v(45): near "endmodule": expecting: END
vlog -reportprogress 300 -work work {E:/Quartus_MS_test/lab2/Verilog/lab2_fifo_tb.v}
# Model Technology ModelSim ALTERA vlog 5.3d Altera Compiler 2000.04 May 17 2000
# -- Compiling module lab2_fifo_tb
# Top level modules:
# lab2_fifo_tb
vsim -t ns work.lab2_fifo_tb
# vsim -t ns work.lab2_fifo_tb
# Loading work.lab2_fifo_tb
```

```
source_edit_5 - lab2_fifo_tb.v
File Edit Object Options Window
30
31          #50      tdata <= 20;
32
33          #50      tdata <= 24;
34
35          #50      tdata <= 28;
36
37          #50      tdata <= 32;
38
39          #50      tdata <= 36;
40
41          #50      tdata <= 40;
42
43          #200     treset <= 1;
44
45          end
46
47     endmodule
48
```

在信息上双击，引起
错误的代码在 Source
窗口被点亮



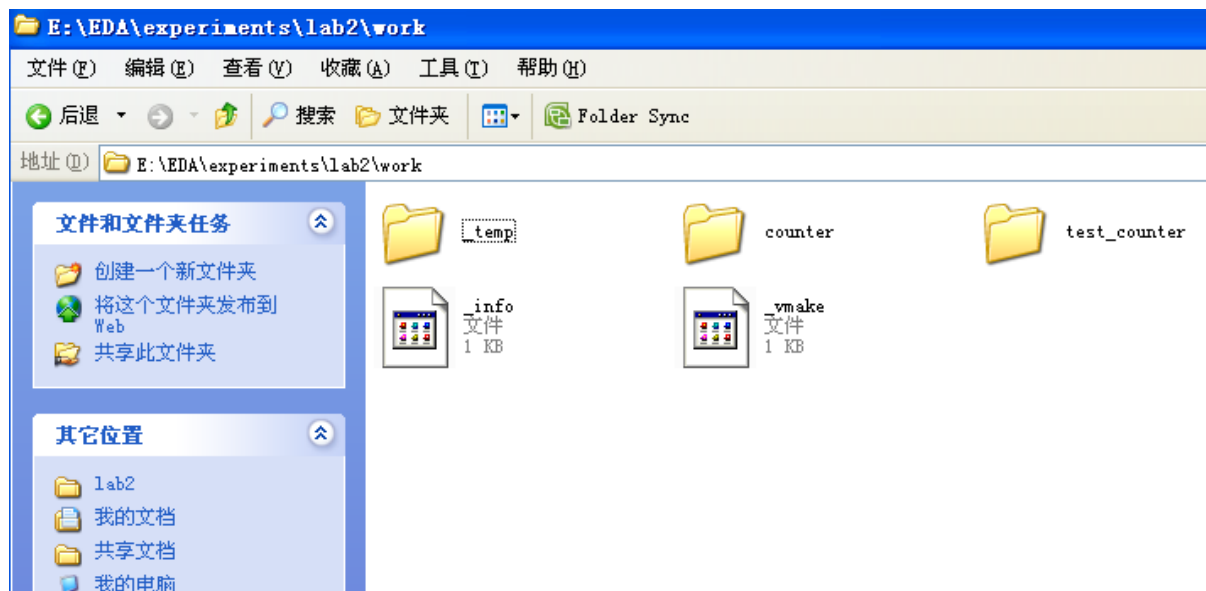
基本仿真步骤

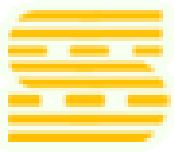
□ 如果设计文件编译无误，每个设计文件后面的“Status”栏会有**绿色**的**对勾**，否则会有**红色的错叉**出现。当有错叉出现时，需要根据ModelSim的“**Transcript**”栏内的提示信息修改设计文件，并重新编译，直到编译通过为止。



基本仿真步骤

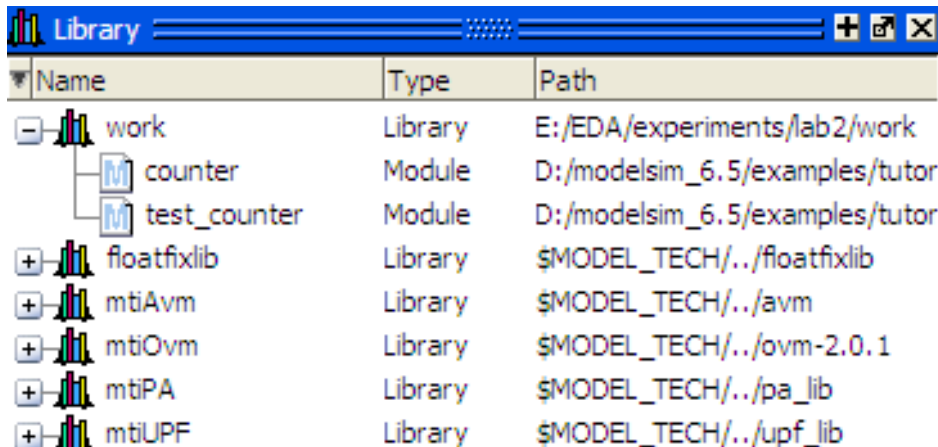
□ 编译通过后，由设计文件编译成的设计单元会被加入到工作库中，如图所示：





基本仿真步骤

□ 查看编译后的设计单元，鼠标点击**Library Tab**栏，将会看到两个编译了的设计单元列了出来。*(看不到就要把Library的工作域设为work)*



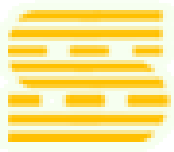
Name	Type	Path
work	Library	E:/EDA/experiments/lab2/work
counter	Module	D:/modelsim_6.5/examples/tutor
test_counter	Module	D:/modelsim_6.5/examples/tutor
floatfixlib	Library	\$MODEL_TECH/./floatfixlib
mtiAvm	Library	\$MODEL_TECH/./avm
mtiOvm	Library	\$MODEL_TECH/./ovm-2.0.1
mtiPA	Library	\$MODEL_TECH/./pa_lib
mtiUPF	Library	\$MODEL_TECH/./upf_lib

(命令行输入: `vsim test_counter`

仿真多个top级模块。

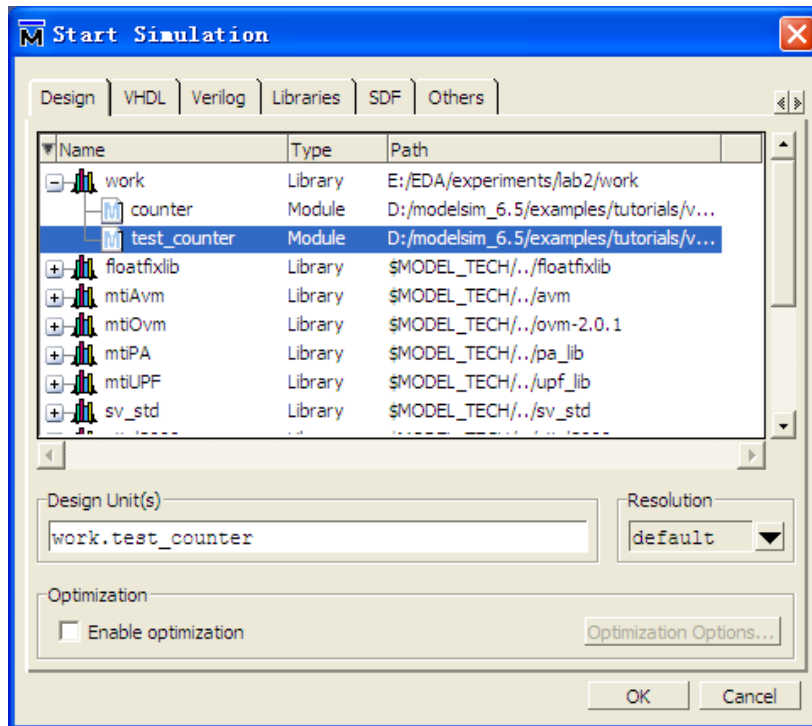
`vsim top_level1 top_level2`)

最后, 将编译后生成的设计单元加入到仿真器中 (`load design`) , 由于**测试文件** (testbench) 例化了要仿真的模块, 所以只需要将测试文件生成的设计单元加载到仿真器中。双击**Library Tab**中的工作库下**测试文件生成的设计单元** , 将会出现**Sim Tab**, 其中显示了counter设计单元的结构。



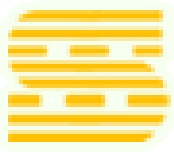
基本仿真步骤

也可以选择“**Simulate > Start Simulation**”，会弹出一个对话框，展开“Work”工作库，选中测试文件生成的设计单元，如图所示：



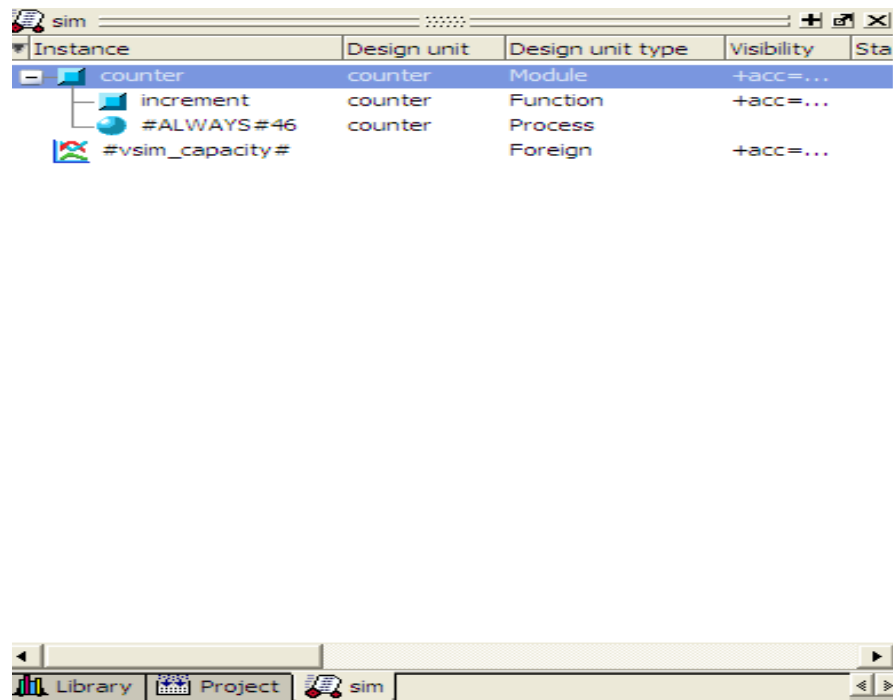
(命令行输入：
view signals list wave
出现信号窗口，列表窗口
和波形窗口)

点击“OK”按钮即可将测试文件生成的设计单元加载到仿真器中。



基本仿真步骤

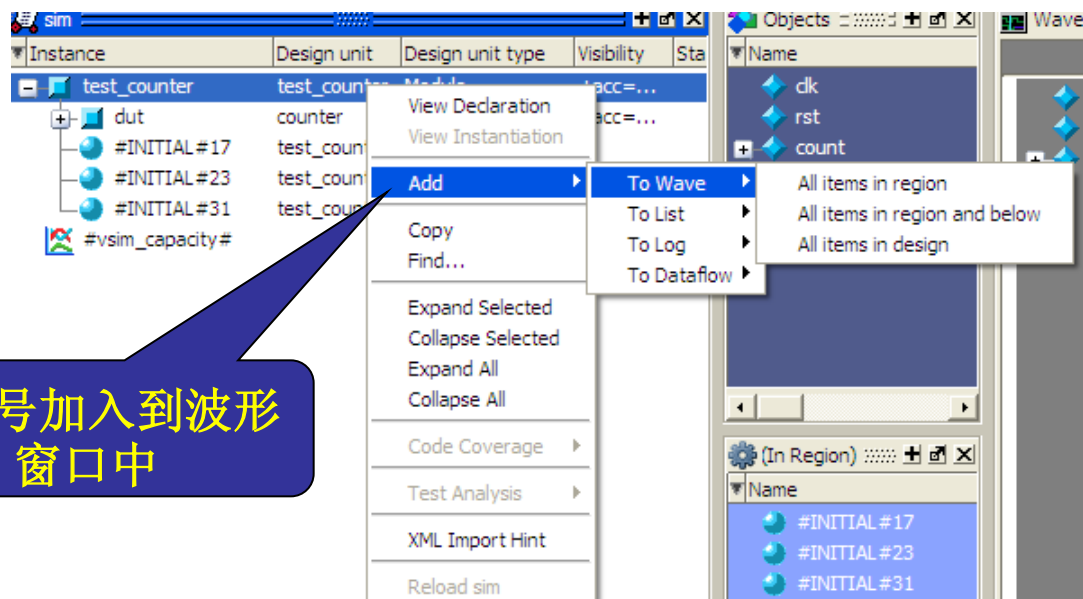
设计单元成功加载到仿真器后，ModelSim会自动弹出仿真器，并将“Workspace”列表切换成“sim”栏，如图所示：



□ 开始运行仿真和分析



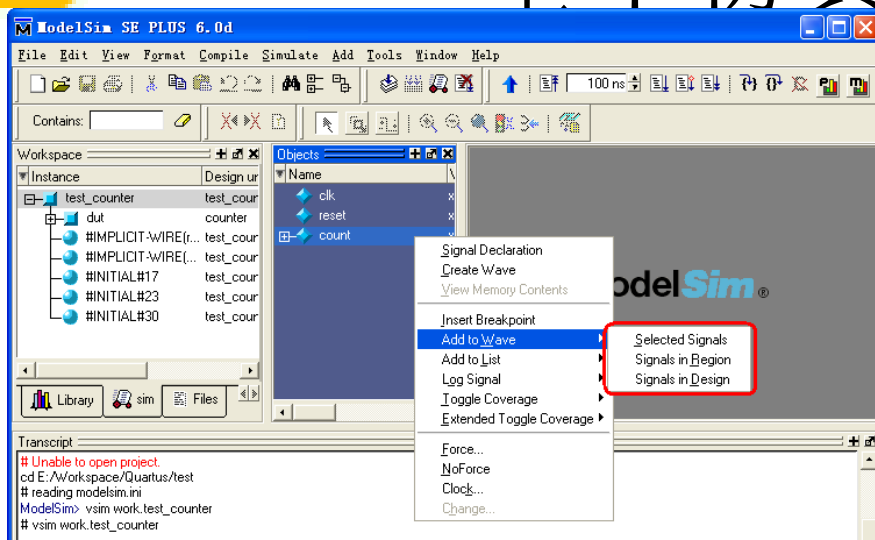
基本仿真步骤



也可以在“**Objects**”列表里选择感兴趣的信号加入到波形窗口里。方法是选中信号，**右键单击此信号**，在弹出的列表里选择“**Add to Wave**”，会弹出三个选项，如图所示：



基本仿真步骤



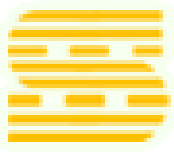
“Selected Signals”项：表示加入到波形窗口中的信号是**被选中的信号**，
即图中“Objects”列表里的“count”信号。

“Signals in Region”项：表示加入到波形窗口中的信号是**“Objects”列表里的所有信号**，
即图中“Objects”列表里的“clk”、“reset”和“count”信号。

“Signals in Design”项：表示加入到波形窗口中的信号是**测试文件生成的设计单元里的所有信号**，

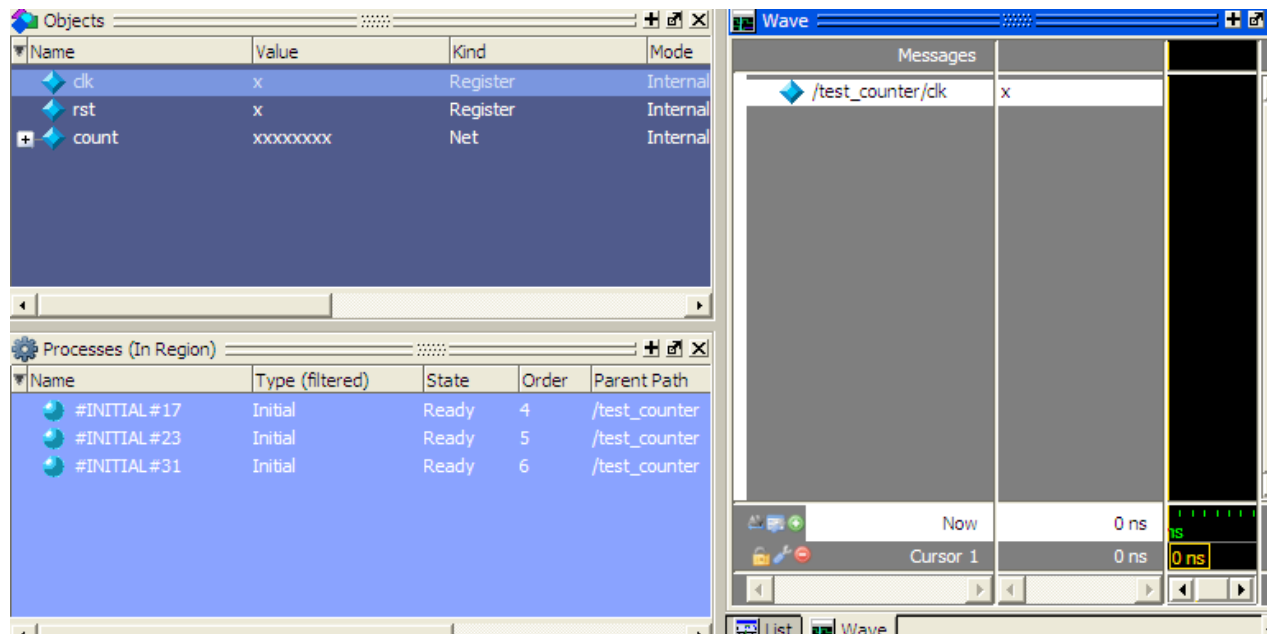
即图中“Workspace”列表里的“test_counter”下面展开的所有设计单元的信号。

其中“Signals in Design”的效果与上一种方法效果是一样的。



基本仿真步骤

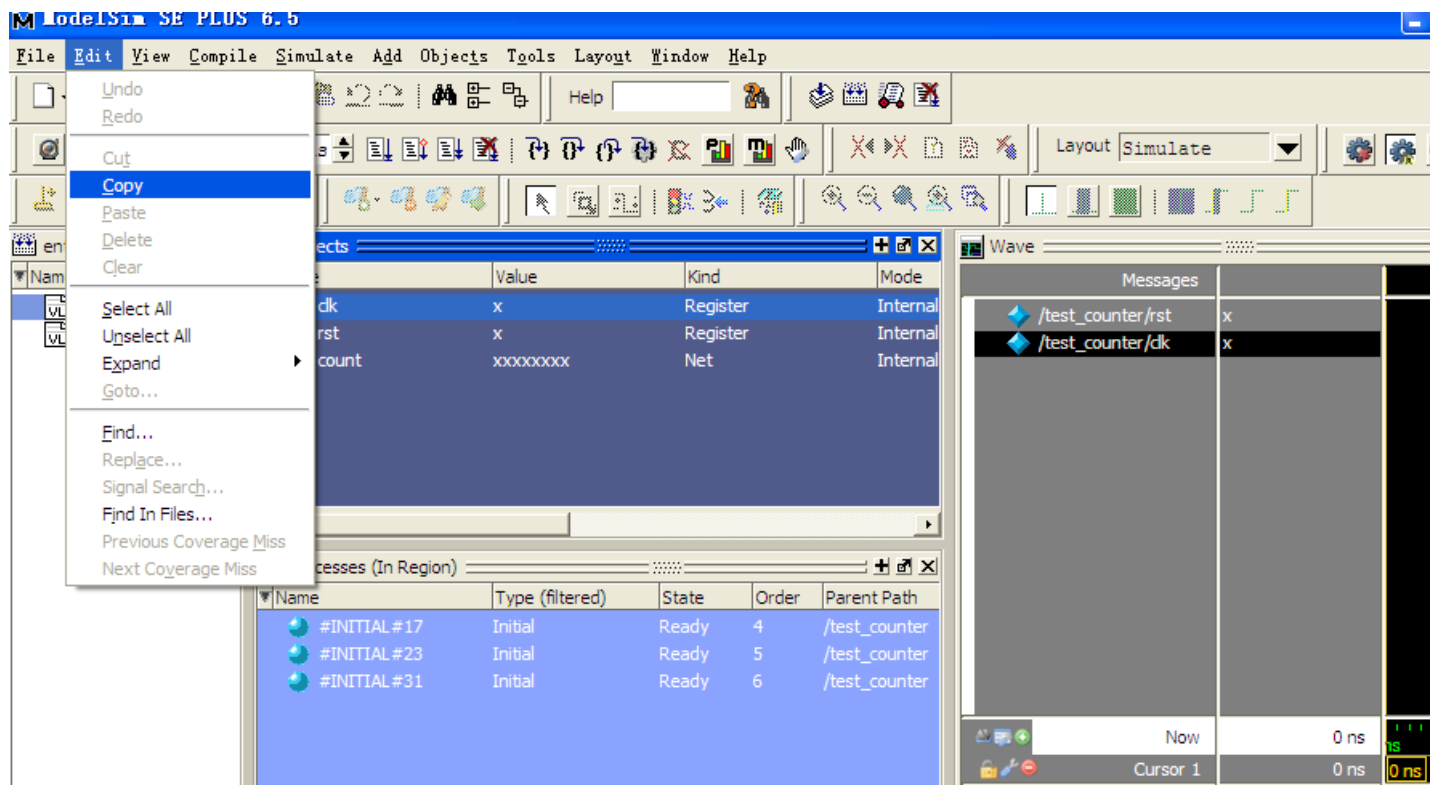
也可以在信号窗口中，选中每个 clk ， rst 和 $count$ 信号组成一个组的选项，点击这个组不放并保持一段时间，然后将它拖动到路径名或波形窗口的值处。

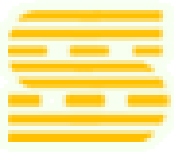




基本仿真步骤

HDL的条目也能够用**Edit>Copy**和**Edit>Paste**菜单选项从一个窗口复制到另一个窗口（或者在波形和列表窗口中）。也可以用**Edit>Delete**选项删除所选条目。





基本仿真步骤

打开结构窗口，View>Structure *（重新整理窗口使你清楚的查看所有打开的窗口，试用菜单选项Layout>Reset）*。

结构窗口显示了设计的分级结构。默认只有层级的顶层可扩展。可以通过

过点击任何行的符号“+”（展开）或“-”（缩短）操控层级。

点击“+”可以看到全部的三个层级：test_counter, counter和一个

名为increment的函数。（如果不显示test_counter, 那么所仿真的counter就替代test_counter。）



基本仿真步骤

Instance	Design unit	Des
[-] counter	counter	Mod
[-] increment	counter	Fun
#ALWAYS#46	counter	Pro
[-] test_counter	test_counter	Mod
[-] dut	counter	Mod
[-] increment	counter	Fun
#ALWAYS#46	counter	Pro
#INITIAL#17	test_counter	Pro
#INITIAL#23	test_counter	Pro
#INITIAL#31	test_counter	Pro
#vsim_capacity#		For



基本仿真步骤

□ 点击“**run**”按钮，就可以进行仿真了，需要注意的是，“**run**”按钮每次只能仿真**100ns**，如果要一直仿真下去的话，点击“**run all**”按钮可以实现。

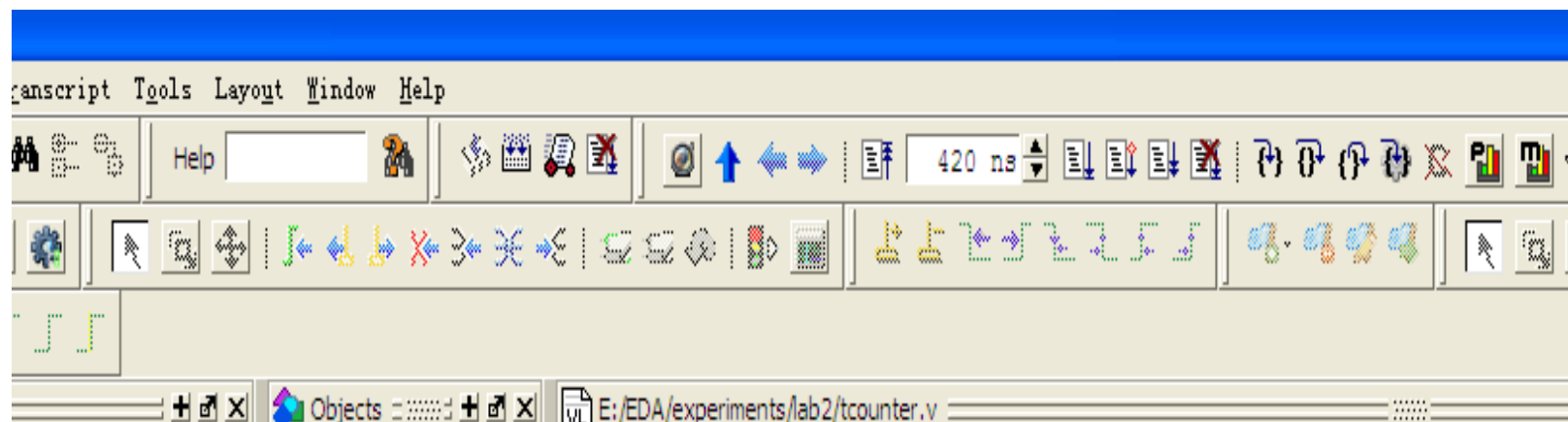


如果要停止仿真，可以按“**break**”按钮；如果要重新仿真，可以按“**restart**”按钮

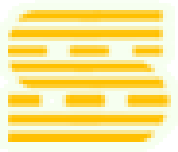


基本仿真步骤

改变Run Length的值。



- **run 1000**
 - 从当前位置运行仿真 1000 timesteps
- **run 2500 ns**
 - 从当前位置运行仿真2500 ns
- **run @3000**
 - 运行仿真到 timestep 3000



基本仿真步骤

对仿真波形进行分析，确定逻辑是否正确。如果没有得到你要的仿真结果，则需要修改设计文件，并**重新编译**，加载到仿真器中进行仿真。

注：

*ModelSim*的编译器在对设计文件进行编译时，并不会对设计文件进行综合，即将设计文件与硬件对应（上面的仿真步骤并没有选择器件型号）。因此编译时生成的设计单元并没有包含硬件信息，即不会存在器件延时，所以只用*ModelSim*作的仿真是一种功能仿真，有时也称为“前仿真”。

如果想用*ModelSim*作仿真时加入器件的延时信息，模拟硬件来仿真，则需要*Quartus*的配合才能完成。加入器件延时信息的仿真称为“后仿真”，仿真结果接近真实器件的运行结果。



调试仿真器

1. 什么时候调试?

- 编译失败
- 不正确或意外的仿真结果

2. ModelSim 调试能力举例

- 信号监视
- 断点



监视更多的信号

给跟踪加入附加的信号或变量

1. 在Structure 窗口选择层

2. 从Source, Signals 或Variables窗口 “拖放” 到:

- Wave窗口

- List 窗口



断点

1. 支持两种类型的断点

□ 在源代码窗口设置断点

1. Toggles – 再次点击删除断点
2. 没有断点数量的限制
3. 用 bp 命令

bp <file_name> <line#>

□ 条件断点

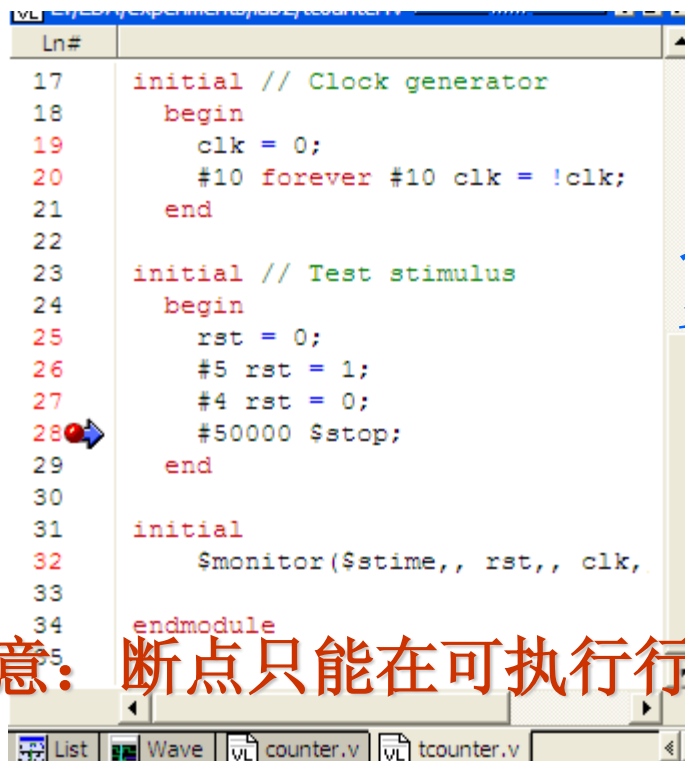
1. *when* <condition> <action>
2. *when* {b=1 and c/=0}
3. 与VHDL信号和Verilog 线网和寄存器一起使用
4. 也可用 bp 命令

bp <file_name> <line#> {if{\$now/=100}then{cont}}



断点

在counter.v文件的第30行设置一个断点。首先，在源代码窗口中移动光标到第30行，点击行号30处设置一个断点。在断点设置处的行号边可以看到一个红点出现。



```
Ln#
17  initial // Clock generator
18      begin
19          clk = 0;
20          #10 forever #10 clk = !clk;
21      end
22
23  initial // Test stimulus
24      begin
25          rst = 0;
26          #5 rst = 1;
27          #4 rst = 0;
28          #50000 $stop;
29      end
30
31  initial
32      $monitor($stime,, rst,, clk,
33
34  endmodule
35
```

可以通过点击断点使它在有效和无效之间切换。当断点无效时，循环中断。要删除断点，用鼠标右键点击断点行号并在菜单里选择删除。

注意：断点只能在可执行行（有绿色或红色行号标识）设置。



断点

命令行设置断点:

bp <filename> <line_number>

```
ModelSim> bp counter.v 28
# ** Error: (vish-4000) A design must be loaded before the bp command can be used.
ModelSim> vsim work.counter
# vsim work.counter
# Loading work.counter
VSIM 73> bp counter.v 30
# ** Error: (vsim-3330) Line 30 in file "counter.v" is not executable.
#
VSIM 74> bp counter.v 29
VSIM 75> ]
```



断点

当仿真运行到断点时，停止运行，用一个深蓝色箭头标显Source窗口并在主窗口中发布一条“**中断**”信息。

The screenshot displays a Verilog simulation environment. The top-left pane shows the project hierarchy with files like #INITIAL#23, #INITIAL#31, and #vsim_capacity#. The top-right pane shows the Verilog source code for tcounter.v, with a blue arrow pointing to line 28: `#50000 $stop;`. The bottom-left pane shows the 'Processes (In Region)' table:

Name	Type (filtered)	State	Order
/test_counter/#INI...	Initial	Wait	-
/test_counter/#INI...	Initial	Active	1

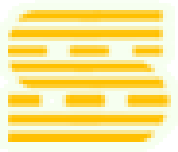
The bottom-right pane shows the 'Transcript' window with the following output:

```
49990 0 0 195
50000 0 1 196
Break in Module test_counter at E:/EDA/experiments/lab2/tcounter.v line 28
```



Model*Sim* 项目

1. 一个集合:
 - 根目录和子目录
 - HDL 仿真文件
 - 库
 - 仿真器设置
2. 允许你保留当前工作
3. 多用 UI
4. 在项目目录里保存为 .MPF 文件
 - 仍然支持.INI文件
5. 项目操作
 - **File -> New / Open / Delete**



force 命令

1. 允许用户给VHDL信号和Verilog线网予以激励

2. 常规语法:

□ *force* <item_name> <value> <time>, <value> <time>

3. 参数

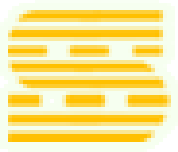
□ *item_name*

1. 被激励的HDL项的名称

2. 必需的

3. Must be a **scalar** (标量) or **one-dimensional array of characters**. Can be an indexed array, array slice, or record sub-element as long as its of the above type

4. Can use **wildcards** (通配符) as long as only one match is obtained



force 命令

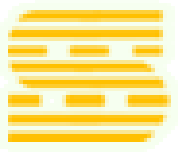
□ **value**

- 1.被强制的项的值
- 2.必须适合项的数据类型
- 3.必需的

□ **time**

- 1.指定值的时间单位
- 2.相对于当前的仿真时间
 - 1.用 @ character指定绝对时间
- 3.时间单位能被指定
 - 1.缺省值是仿真分辨率
- 4.可选的

Value	Description
1111	character sequence
2#1111	binary radix
10#15	decimal radix
16#F	hexadecimal radix



force 命令

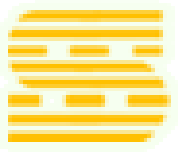
4. 其他参数

- **-r[epeat] <period>**

1. 在指定周期重复**force**命令
2. 可选的

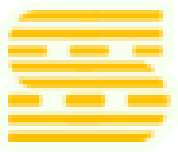
- **-cancel <period>**

1. 在指定周期后取消强制**force**命令
2. 可选的



force 命令

- *force* **clr 0**
 - 在当前仿真时间强制 **clr** 到 0
- *force* **bus1 01XZ 100 ns**
 - 在当前仿真时间后100ns强制 **bus1**到 01XZ
- *force* **bus2 16#4F @200**
 - 仿真启动后强制 **bus2**到 4F直到200时间单位，分辨率在仿真启动时选择
- *force* **clk 0 0, 1 20 -repeat 50 -cancel 1000**
 - 在当前仿真后0时间单位强制**clk**到0和在20时间单位强制到1. 每50时间单位重复直到1000. 因此, 下一个 1 将在70时间单位发生
- *force* **clk2 1 10 ns, 0 {20 ns} -r 100 ns**
 - 和上一个例子相似。-r前面的时间单位表达式必须放在大括号里



DO 文件

1. 自动完成仿真步骤的宏文件

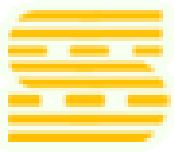
- 库设置
- 编译
- 仿真
- 强制仿真激励

2. 能在所有的ModelSim 模式里被调用

- UI) *Macro* -> *Execute*
- COM) *do* <filename>.do

3. 能调用其他的DO文件

```
cd c:\mydir
vlib work
vcom counter.vhd
vsim counter
view *
add wave /*
add list /*
do run.do
```



DO 文件

my_sim.do

```
cd c:\mydir  
vlib work  
vcom counter.vhd  
vsim counter  
view *  
do stimulus.do
```

stimulus.do

```
add wave /clk  
add wave /clr  
add wave /load  
add wave -hex /data  
add wave /q  
force /clk 0 0, 1 50 -repeat 100  
force /clr 0 0, 1 100  
run 500  
force /load 1 0, 0 100  
force /data 16#A5 0  
force /clk 0 0, 1 50 -repeat 100  
run 1000
```



startup.do 文件

1. 一个 DO 脚本自动执行通过 *vsim*
2. 一个例文件 *startup.do* 就象这样:
view source
view structure
view wave
do wave.do
3. 调用一个启动文件, 未注释 (移去 “;”) 下面 *modelsim.ini* 文件行给 do 文件提供路径:
;Startup = do /path /startup.do



Modelsim仿真（使用do文件）

在modelsim中使用do文件是非常方便的进行仿真的一种方法，尤其项目比较大，是几个人分开做的，所以前后模块的联合仿真比较重要，并且WAVE窗口里要引入的信号也很多，不同人开发的代码放在了单独的文件夹中。

如果还是像原来那样仿真，单单是编译源文件，添加查看信号就要花费不少时间，且仿真需要比较长的时间，需要观察不同时间段内的仿真结果，因此使用do文件。



Modelsim仿真（使用do文件）

以带时钟和复位信号的加法器为例，Verilog 代码如下：

```
module counter (    clk, n_rst,  a,  b,  d  );
input [7:0] a;
input [7:0] b;
input clk;
input n_rst;
output [8:0] d;
reg [8:0] d;
always @ ( posedge clk)
begin
    if ( !n_rst)
        d <= 9'b0000000000;
    else
        d <= a + b;
end
endmodule
```



Modelsim仿真（使用do文件）

编写名为 counter.do 文件，其内容如下：

```
1) vlib work
2) vmap work
3) vlog counter.v
4) vsim -L work work.counter -t 1ns
5) add wave -r /*
6) force -repeat 10 clk 0 0,1 5
7) force a 16#aa
8) force b 16#bb
9) force n_rst 1
10) run 40
11) force n_rst 0
12) run 20
13) force n_rst 1
14) force a 16#55
15) force b 16#aa
16) run 20
```



Modelsim仿真（使用do文件）

在 Modelsim 中，将工作目录切换到 counter.v 和 counter.do 两个文件所在的目录下，然后在命令行中输入 do counter.do，仿真结果如下图

