1. 常用仿真命令

vlib work　// 建立 work 仿真库

vmap work wrok　// 映射库

vlog　-cover　bcest　*.v　// 加覆盖率分析的编译

vsim　-coverage　-voptargs="+acc"　-t　ns　test　// 仿真文件为 test.v

add wave　*　// 将所有模块 waveform. dump 出来

add wave sim:/test/t/M2/Reg_out　// 将模块 Reg_out 中的 waveform. dump 出来

delete　wave　/test/i

2. SVA 断言仿真命令

vlog　-sv　a.v

vsim　-assertdebug　test

view　assertions

vsim　-assertdebug　ScaleBlock_tf　-L　xilinxcorelib_ver　-L　unisims_ver　// 加载 xilinxlib 库

3. verror　3601　// 查错

4. 给仿真工具加载 xilinx 库命令

(1)加载之前将 modelsim.ini 改为非“只读”

(2)“运行” cmd，到 xilinx 目录下

(3) C:\Xilinx > compxlib -s mti_se -p c:\Modeltech_6.0\win32 -f all -l verilog -o C:\Modeltech_6.0\Xilinx_lbis

或者　Xilinx 目录下.\bin\nt\下有 compxlib.exe

简单得 modelsim 命令行仿真

用 do 文件进行仿真真得很方便，比写 testbench 方便多了，我是深有感触呀，开始时因为不知道，只知道写 testbence，在小得模块也写 testbench,真得很烦躁！而且信号定义什么得比较多，采用 do 文件得方法就没有那么多信号定义了，管理也比较方便，呵呵，真得很方便，而且采用命令行得形式，感觉特有成就感，呵呵！

1.运行仿真，在主窗口输入命令：vsim work.实体名

2.为时钟信号添加驱动，输入命令：force clk 0 0,1 10 -r 20，将仿真时钟设为 50MHz；(设时间单位为 ns)

3.打开波形窗口，输入命令：view wave

4.为波形窗口添加信号，输入命令：add wave -hex *，这里的*表示添加设计中所有的信号，-hex 表示以十六进制来表示波形窗口中的信号值；

5.开始仿真，输入命令，run 3us，这时候在波形窗口中出现仿真波形

6.退出仿真，输入命令：quit –sim。

modelsim 常用命令

用 do 文件进行仿真真得很方便，比写 testbench 方便多了，采用 do 文件没有那么多信号定义，管理也比较方便.

1.运行仿真，在主窗口输入命令：vsim work.实体名

2.为时钟信号添加驱动，输入命令：force clk 0 0,1 10 -r 20，将仿真时钟设为 50MHz；(设时间单位为 ns)

3.打开波形窗口，输入命令：view wave

4.为波形窗口添加信号，输入命令：add wave -hex *，这里的*表示添加设计中所有的信号，-hex 表示以十六进制来表示波

形窗口中的信号值；

5.开始仿真，输入命令，run 3us，这时候在波形窗口中出现仿真波形

6.退出仿真，输入命令：quit –sim //很常用!!

7.查看错误详细信息: verror **(错误数字代号)

--vlib - 建立一个新的工作库。

如：vlib work

在当前目录建立逻辑库 work，运行后会在当前目录下找到 work 文件夹。

--vmap - 映射逻辑库名到指定的目录

--vsim - 启动仿真

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。

odelSim 之命令行仿真入门

下面是我们的 Tcl 仿真步骤：

启动 ModelSim SE，首先看到在在 ModelSim SE 右边的窗口有 ModelSim> 这样的提示符。在提示符后，顺序运行以下命令：

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。

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 仿真常用命令以及仿真脚本的编写

在我们用 ModelSim 仿真的时候经常是修改一点一点修改代码，这样会造成一个无奈的操作循环：修改代码--->编译代码--->仿真设置--->进入仿真页面--->添加需要观察的波形--->运行仿真。如果仿真结果不理想，还得需要重新修改代码，重复上述的操作。

计算机擅长做重复的事情，为什么不让计算机代劳呢？

我们可以参照 Xilinx ISE 是如何调用 ModelSim 进行仿真的，尤其是脚本的编写。

下面一个脚本是我用 ISE10.1 建立了一个 AES256 的工程，然后在调用 ModelSim6.5 的时候，ISE 会生成这几个脚本文件，AES256_tb.fdo, AES256_tb_wave.fdo 和 AES256_tb.udo。

下面的代码是 AES256_tb.fdo 文件的内容。

```
1: vlib work #创建名字是 work 的库，这个仿真之前必须做的
 2: vcom -explicit  -93 "SBOX_ROM.vhd"  #编译 vhd 代码
 3: vcom -explicit  -93 "Inv_SBOX_ROM.vhd"
 4: vcom -explicit  -93 "AES_package.vhd"
 5: vcom -explicit  -93 "subbytes.vhd"
 6: vcom -explicit  -93 "SBOX_ROM_reg.vhd"
 7: vcom -explicit  -93 "round_key_BRAM.vhd"
 8: vcom -explicit  -93 "Inv_subbytes.vhd"
 9: vcom -explicit  -93 "Keyexpansion_Yao.vhd"
10: vcom -explicit  -93 "AES256_ENC_DEC.vhd"
11: vcom -explicit  -93 "AES256_TOP.vhd"
12: vcom -explicit  -93 "AES256_tb.vhd"
13: vsim -t lps   -lib work AES256_tb
```

```
#进入仿真设置，时间单位为 1ps，库指定为 work，AES256_tb 就是指你的 top 层设计的名字
 14: do {AES256_tb_wave.fdo} #执行*.fdo 文件，用来添加信号和变量或者内部的寄存器到波形（WAVE）窗口
 15: view wave #打开波形窗口
 16: view structure #打开架构（structure）窗口
 17: view signals   #打开信号列表窗口
 18: run 1000ns     #运行 1000ns
 19: do {AES256_tb.udo} #运行用户定义的脚本
```

只要编译的时候没有出现语法错误或者是找不到定义的库文件等错误提示，一般会很容易的看到仿真的波形，而不用手动进行操作。这样方便了仿真的整个过程，而无需用很多鼠标点击操作。

现在通过一个具体的实例来说明如何运用脚本来实现 ModelSim 的仿真。

工具版本：ISE10.1 ， ModelSim 6.5a

1. 创建 ISE 工程

首先通过 ISE 创建 test.vhd 和 test_tb.vhd 文件并添加到工程中，这里不细说如何创建 HDL 源代码。

然后配置 ISE 的仿真器，右键选中 FPGA 芯片，点击 properties，然后在 Simulator 选择 Modelsim-SE VHDL。

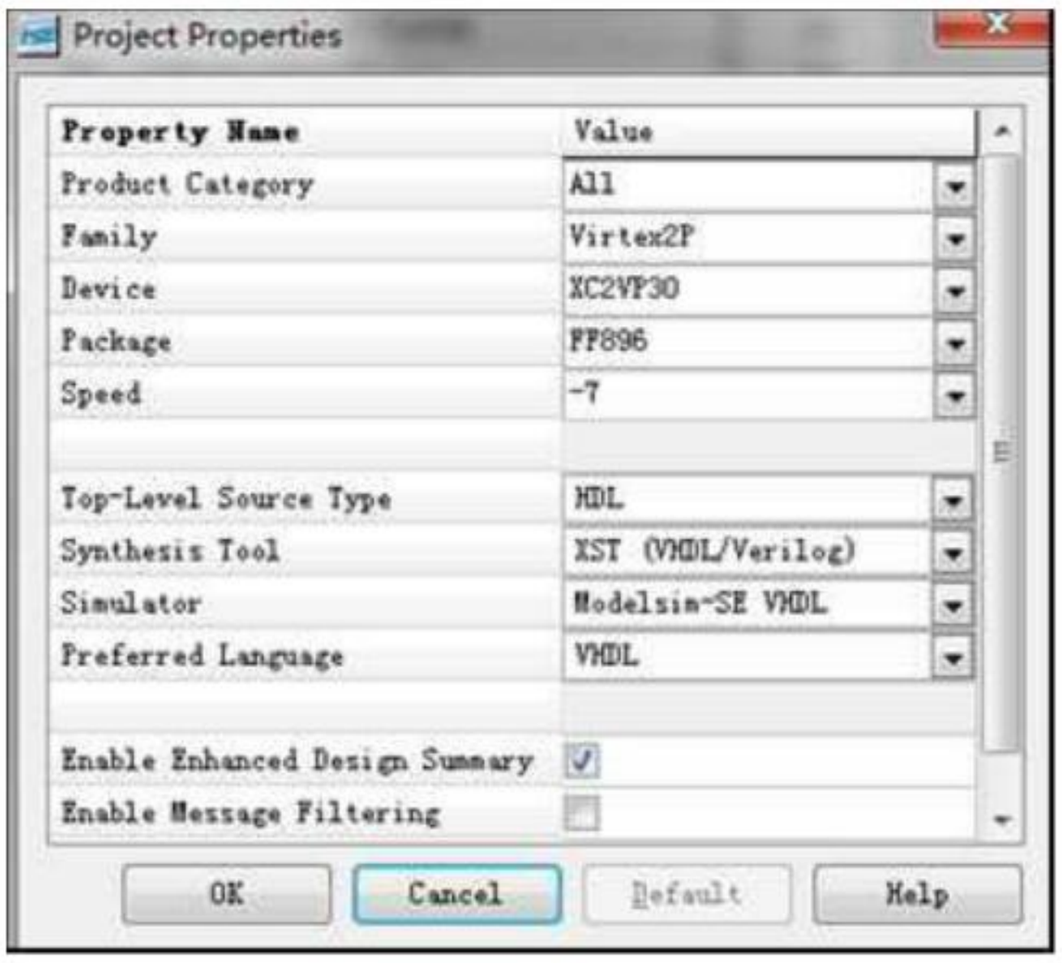

图 1. 工程属性

在 source 窗口选择“Behavioral Simulation”，可以看到工程中的 test_tb.vhd（testbench 文件）。选中 test_tb.vhd 文件，在 Process 窗口中双击 Simulate

Behavioral Model，ISE 开始调用 ModelSim，这是 ISE 自动生成了三个脚本文件:test_tb.fdo, test_tb_wave.fdo 和 test_tb.udo。

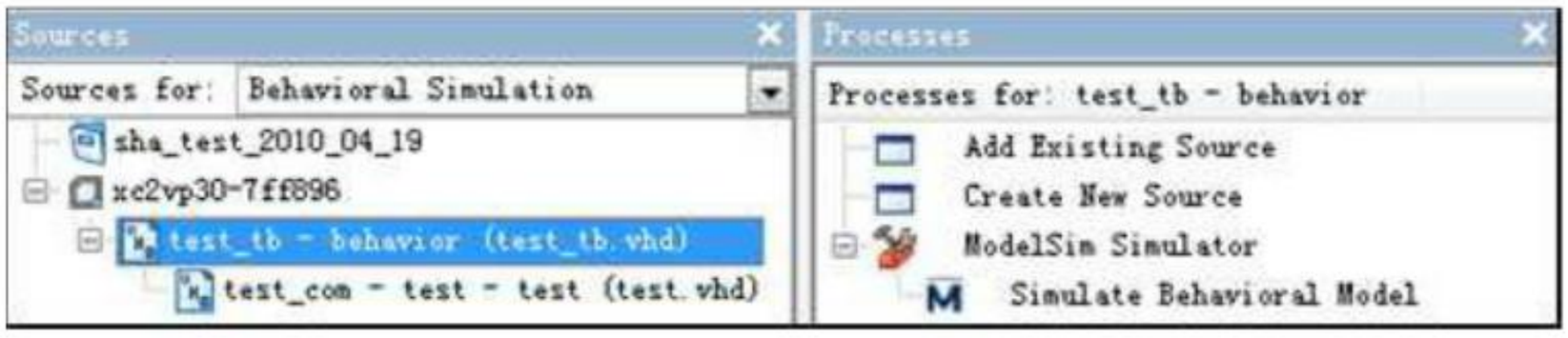

图 2. Behavioral Simulation 窗口

看看三个脚本文件的内容吧。

test_tb.fdo 内容如下：

```
1: vlib work    #添加 library
2: vcom -explicit  -93 "test.vhd"  #编译 test 的顶层文件
3: vcom -explicit  -93 "test_tb.vhd" #编译 test_tb 的 testbench 文件
4: vsim -t 1ps   -lib work test_tb   #进入仿真，时间单位 1ps，仿真 test_tb
5: do {test_tb_wave.fdo}            #执行*.fdo 脚本文件，用于添加仿真波形
6: view wave                       #打开波形窗口
7: view structure                   #打开 structure 窗口
8: view signals                    #打开信号窗口
9: run 1000ns                      #仿真运行 1000ns
10: do {test_tb.udo}                #执行*.udo 脚本文件，用于执行用户定义的脚本命令
```

test_tb_wave.fdo 内容如下（）：

```
1: ## Project Navigator simulation template: test_tb_wave.fdo
2: ## You may edit this file to control your simulation.
3: add wave *  ##添加 Top 层所有的端口信号
```

test_tb.ufo 内容为空白，是留着给用户自己添加。

对我们来说最有用的就是 test_tb.fdo 文件了，只要稍微修改就可以成为一个针对这个工程的很好的用于仿真的脚本。当然我们可以自己手动来编写类似的脚本，用 ISE 自动生成主要是为了涂个省事。

2.调用 ModelSim 进行仿真

在图 2 中，Process 窗口中双击 Simulate Behavioral Model，进入 ModelSim 仿真环境。请仔细观察 ModelSim Transcript 窗口中消息的输出。内容如下：

```
1: # do {test_tb.fdo}   #<--------执行脚本文件
2: # ** Warning: (vlib-34) Library already exists at "work".
```

```
  3: # Model Technology ModelSim SE vcom 6.5 Compiler 2009.01 Jan 22
2009  #<-----编译 test.vhd 得到的 message
  4: # -- Loading package standard
  5: # -- Loading package std_logic_1164
  6: # -- Loading package std_logic_arith
  7: # -- Loading package std_logic_unsigned
  8: # -- Compiling entity test
  9: # -- Compiling architecture test of test
 10: # Model Technology ModelSim SE vcom 6.5 Compiler 2009.01 Jan 22
2009 #<-----编译 test.vhd 得到的 message
 11: # -- Loading package standard
 12: # -- Loading package std_logic_1164
 13: # -- Loading package std_logic_arith
 14: # -- Loading package std_logic_unsigned
 15: # -- Compiling entity test_tb
 16: # -- Compiling architecture behavior of test_tb
 17: # vsim -lib work -t lps test_tb              #仿真设置命令行
 18: # Loading std.standard
 19: # Loading ieee.std_logic_1164(body)
 20: # Loading ieee.std_logic_arith(body)
 21: # Loading ieee.std_logic_unsigned(body)
 22: # Loading work.test_tb(behavior)#1          #加载 test_tb
 23: # .main_pane.wave.interior.cs.body.pw.wf      #打开 wave 窗口
 24: # .main_pane.structure.interior.cs.body.struct #列出 structure
 25: # .main_pane.objects.interior.cs.body
```

注：在 test_tb.fdo 的每一行都可以在 ModelSim Transcript 窗口中分开一行一行执行。

然后可以观察到仿真波形文件。

图 3.仿真波形

3. 调试代码

或许仿真出来的结果不是我们想要的，必须的修改代码，然后再仿真。那我们应该怎么办呢？

在 trasncript 窗口输入：

```
1: quit -sim      #退出仿真
2: do test_tb.fdo #修改代码完成后重新执行该脚本进行仿真
```

在仿真的波形的时候，我们不仅仅希望看到 Top 层设计的端口信号的波形，还希望能观察到内部信号是如何变化的，所以我们在 Sim 窗口中找到内部的信号放到 wave 中进行观察。但是如果我们重新运行当初 test_tb.fdo 文件，我们又再一次只能看到 Top 层设计的端口信号的波形，而内部信号的波形已经被删除。那我们应该如何保存和运用我们仿真波形文件呢？

我现在需要观察内部的一个寄存器输出：tmp 信号，见图 4。Ctrl+S，保存波形文件。默认保存为工程目录下/wave.do，但是我们将用波形文件保存为 test_tb_wave.fdo 文件，这样做的好处可以不用修改 test_tb.fdo 文件。

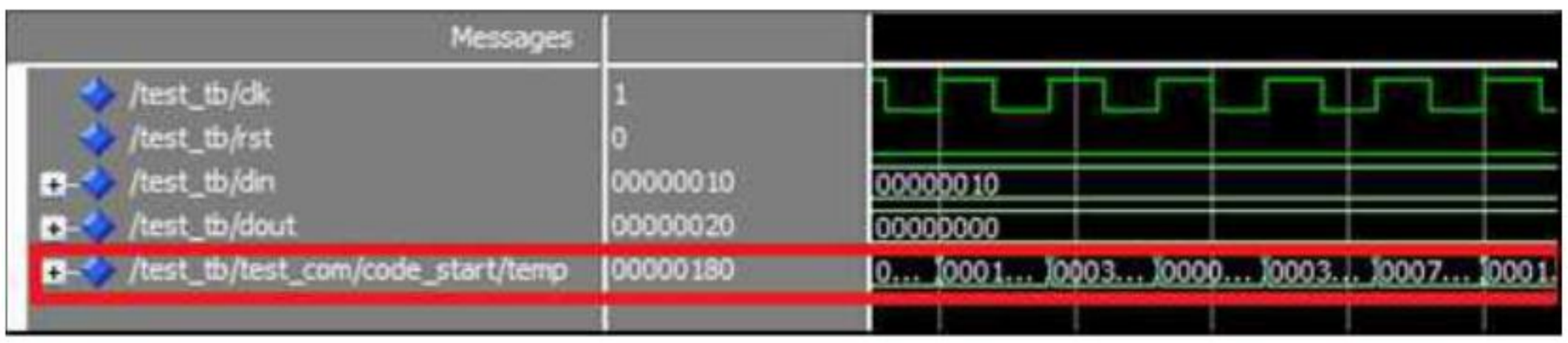

图 4.内部信号观察

我们重新运行 test_tb.fdo 脚本文件之后，输出的波形是图 4，而不是图 3。

差不多常用的 VHDL 仿真命令都用上了，我相信这样做肯定会提高仿真的效率。如果想要熟悉更多的 ModelSim 的仿真命令，请查看 ModelSim 的用书手册。

【翻译】ModelSim 指南 VI （ModelSim）（Verilog）（Digital Logic）

作者：yf.x 来源：博客园 发布时间：2011-03-09 16:20 阅读：307 次 原文链接 [收藏]

8

自动仿真

简介

前面的课程主要讲使用 ModelSim 的交互模式：通过图形界面或主窗口的命令行一条条的执行单一的命令。当需要完成重复的任务时，可用 DO 文件提高效率。

DO 文件是一次执行多条命令的脚本。这个脚本可以像带有相关参数的一系列 ModelSim 命令一样简单，或者是带有变量，执行条件等等的 Tcl 程序。可在 GUI 里或系统命令提示符后执行 Do 文件。

注意：

本课假设你已经添加<install_dir>/modeltech/<platform>到你的环境变量的 PATH。否则，要指定工具（如，vlib，vmap, vlog，vcom，和 vsim）的绝对路径。

扩展阅读

用户手册：Tcl and Macros (DO Files)。

Practical Programming in Tcl and Tk, Brent B. Welch, Copyright 1997

创建一个简单的 **DO** 文件

创建 DO 文件就像在文本文件里输入命令一样。或者，可保存主窗口的 transcript 作为一个 DO 文件。本练习将用在主窗口 transcript 输入的命令来创建一个 DO 文件以添加信号到波形窗口，添加激励，然后仿真。

1. 加载 test_counter 设计单元。

a) 启动 ModelSim。

b) 切换目录至在基本仿真那课创建的目录。

c) 输入 vsim test_counter 来加载设计单元。

2. 输入命令给波形窗口添加信号，激励信号，然后运行仿真。

选择 File > New > Source > Do 来创建一个新的 DO 文件。

在源码窗口输入以下命令：

```
add wave count

add wave clk

add wave reset

force –freeze clk 0 0, 1 {50 ns} –r 100

force reset 1

run 100

force reset 0

run 300

force reset 1

run 400

force reset 0

run 200
```

3. 保存文件。

a) 选择 File > Save As。

b) 在 File name 栏输入 sim.do 并保存到当前目录。

4. 再次加载仿真并使用 DO 文件。

a) 在 VSIM>后输入 quit –sim。

b) 在 ModelSim>后输入 vsim test_counter。

c) 在 VSIM>后输入 do sim.do。

ModelSim 执行保存的命令并在波形窗口生成波形。

5. 当完成本练习，选择 File > Quit 退出 ModelSim。

在命令行模式运行

1. 创建一个新目录并拷贝文件。

开始本练习前，创建一个新目录，并拷贝以下文件：

l /<install_dir>/examples/tutorials/verilog/automation/counter.v

l /<install_dir>/examples/tutorials/verilog/automation/stim.do

2. 创建一个新的设计库并编译源文件。

在系统提示符后输入以下命令。

a) 输入 vlib work。

b) 输入 vlog counter.v。

3. 创建一个 DO 文件。

a) 打开文本编辑器。

b) 输入以下内容：

```
# list all signals in decimal format

add list –decimal *

# read in stimulus

do stim.do

# output results

write list counter.lst

#quit the simulation

quit –f
```

c) 把文件命名为 sim.do，然后保存在当前文件夹。

4. 运行批处理仿真

a) 在提示符后输入以下命令：

vsim –c –do sim.do counter –wlf counter.wlf

这里-c 表示不打开 GUI。-wlf 保存仿真结果为一个 WLF 文件。调试时可在 GUI 查看仿真结果。

5. 查看输出列表。

a) 打开 counter.lst 并查看仿真结果。类似下图：

```
ns          /counter/count
 delta          /counter/clk
                /counter/reset
  0   +0                    x z *
  1   +0                    0 z *
 50   +0                    0 * *
100   +0                    0 0 *
100   +1                    0 0 0
150   +0                    0 * 0
151   +0                    1 * 0
200   +0                    1 0 0
250   +0                    1 * 0
.
.
```

6. 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）。

Figure 8-1. A Dataset in the Main Window Workspace

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

波形窗口显示波形。

7. 当查看完结果。选择 File > Quit 关闭 ModelSim。

在仿真器里使用 Tcl

前面的练习的 DO 文件只使用了 ModelSim 的命令。但是，DO 文件实质上是 Tcl 脚本。可以包含很多 Tcl 结构，比如程序，条件运算符，数学和三角函数，正则表达式等等。

本练习创建一个 Tcl 脚本，用来识别信号的确定值，如果存在，在波形窗口放置标签。标签可将波形窗口的滑动和放大区域记录下来。

1. 创建脚本。

a) 在文本编辑器，新建一个文件并输入以下行：

```
proc add_wave_zoom {stim num} {
echo “Bookmaking wave $num”
bookmark add wave “bk$num” “[expr $stime – 50] [expr $stime +
100]” 0
}
```

这些命令的作用：

l 创建一个名为“add_wave_zoom”的程序，它有 stime 和 num 两个参数。

l 从当前仿真时间的前 50 个单位时间到后 100 个单位时间的范围创建一个书签。

b) 在脚本末添加以下行：

```
add wave –r /*
when {clk’event and clk=”1”}{
echo “Count is [xa count]”
if {[examine count]==”00100111”}{
add wave_zoom $now 1
} elseif {[examine count]==”01000111”}{
add_wave_zoom $now 2
}
}
```

这些命令的作用：

l 给波形窗口添加所有的信号。

l 使用 when 语句识别 clk 跳变为 1 的时刻。

l 在这些跳变检测 count 的值，如果它是某个确定值，添加一个书签。

c) 将这个脚本以”add_bkmrk.do”为名存到在基本仿真那课创建的目录里。

2. 加载 test_counter 设计单元。

a) 启动 ModelSim。

b) 选择 File > Change Directory 切换至上面第一步保存 DO 文件的目录。

c) 在 QuestaSim>后输入 vsim test_counter

3. 执行 DO 文件并运行设计。

a) 在 VSIM>后输入 do add_bkmrk.do。

b) 在 VSIM>后输入 run 1500 ns

仿真运行，DO 文件创建两个书签。

c) 如果波形窗口停靠在主窗口，激活它，然后选择 Wave > Bookmarks >bk1。如果那个窗口独立出来了，选择 View > Bookmarks > bk1.

查看波形窗口，放大并滑到 count 是 00100111 的时刻。

同样试试 bk2。

本课小结

本课到此结束。

1. 选择 File > Quit 关闭 ModelSim.