第一章 介　绍

ModelSim 的简要使用方法

第一课　Create a Project

1. 第一次打开 ModelSim 会出现 Welcome to ModelSim 对话框，选取 Create a Project,或者选取 File\New\Project,然后会打开 Create Project 对话框。
2. 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。
3. 选取 OK,会看到工作区出现 Project and Library Tab。
4. 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。
5. 在这次练习中我们加两个文件,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。
6. 在工作区的 Project page 中，单击右键，选取 Compile All。
7. 两个文件编译了，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。
8. 最后一不是导入一个设计单元，双击 Library Tab 中的 counter，将会出现 Sim Tab,其中显示了 counter 设计单元的结构。也可以 Design\Load design 来导入设计。

到这一步通常就开始运行仿真和分析，以及调试设计，不过这些工作在以后的课程中来完成。结束仿真选取 Design \ End Simulation，结束 Project 选取 File \ Close \ Project。

第二课　Basic VHDL Simulation

准备仿真

1. 为这次练习新建一个目录，然后拷贝 example 目录中所有的 vhd 文件到该目录下。设置该目录为当前工作目录，这一步通过从该目录调用 ModelSim 或是选取 File\Change Directory 命令来完成。
2. 在编译任何 HDL 代码前，要建立一个设计库来存放编译结果。选取 Design \ Create a New Library 生成一个新的设计库。确定选取 Create: a new library and a logical mapping to it，在 Library Name 域中键入 work，然后选取 OK。这就在当前目录中建立了一个子目录，即你的设计库。ModelSim 在这个目录中保存了名为 *_info* 的特殊文件。

   (Prompt : vlib work

   vmap work work)
3. 选取工具栏里的 **Compile** 命令来编译 **counter.vhd** 文件到新库中。这将打开 Compile HDL Source Files 对话框。使用 **vcom** 命令是看不到的。从列表中

选取 **counter.vhd** 再点击 **Compile**，完成后选取 **Done**。可以编译多个文件，按照设计的需要依次选取进行编译。

(Prompt : vcom counter.vhd)

4. 选取工具栏里的 Load design 按钮，导入设计单元。Load design 对话框可以让你选择库和顶级(**top-level**)设计单元来仿真，你也可以为仿真选取 Simulation Resolution 限制。这次仿真运行，下述是缺省的显示：
   - **Simulator Resolution: default (the default is 1 ns)**
   - **Library: work**
   - **Design Unit: counter**

   如果设计单元是一个实体，你可以点击前面的加号，来浏览其关联的结构。
   (Prompt : vsim counter)
5. 选取 counter，然后选择 Load 接受设置。
6. 下面，选取 View \ All 打开所有的窗口，关于窗口的描述，参阅 *ModelSim User' s Manua*l。
   (Prompt : view *)
7. 在 **Signals window** 选取 **View\List\Signals in Region**，这个命令显示 **List window** 中的顶级(**top-level**)信号。
   (Prompt : add list /counter/*)
8. 下步，通过从 **Signals window** 选取 **View\Wave\Signals in Region** 添加顶级(**top-level**)信号到 **Wave window**。
   (Prompt : add wave /counter/*)

运行仿真

通过应用始终输入激励来开始仿真。

1. 点击主窗口，在 **vsim** 提示符下敲如下面的命令：
   (force clk 1 50 , 0 100 –repeat 100)
   (MENU : Signals\Edit\Clock)
   ModelSim 解释 force 命令如下：
   - force clk to the value 1 at 50 ns after the current time
   - then to 0 at 100 ns after the current time
   - repeat this cycle every 100 ns
2. 现在你可以练习来自于主窗口或波形窗口工具条按钮的两个不同的 **Run** 功能。（**Run** 功能在主窗口和波形窗口中定义，即这两个窗口中有 **Run** 功能）。首先选取 **Run** 按钮，运行完成之后选取 **Run All**。
   Run. 运行仿真，在 100ns 后停止。
   (PROMPT: run 100) (MENU: Run \ Run 100ns)
   Run-All. 一直运行仿真，直到选取 **Break**。
   (PROMPT: run -all) (MENU: Run \ Run -All)
3. 选取主窗口或波形窗口的 **Break** 按钮来中断仿真，一旦仿真到达一个可接受的停止点，它就停止运行。
   在源文件窗口中的箭头指向下一条将被执行的语句。（如果暂停发生时，仿真没在评测一个过程，则没有箭头显示在源文件窗口上）。
   下面，你将在 18 行的函数内部设置一个断点。
4. 移动鼠标到源文件窗口，在 18 行上点击设置断点，可以看到紧挨着行号有一

个红点，可以用鼠标点击切换断点的使能与否，断点禁止后看到是一个小的红色的园环。可以在断点上点击鼠标右键，选取 **Remove BreakPoint 18** 来取消断点。

(PROMPT ： bp counter.vhd 18)

5. 选取 Continue Run 按钮恢复中断了的运行，**ModelSim** 会碰上断点，通过源文件中的一个箭头或是在主窗口中的一条中断信息来显示出来。
   (PROMPT: run -continue) (MENU: Run \ Continue)
6. 点击 Step 按钮可以单步执行仿真，注意 Variables window 中值的变化。如果你愿意可以持续点击 Step。
   (PROMPT: run -step) (MENU: Step)
7. 当你完成了，敲入以下命令结束仿真。
   quit -force
8. 命令没有寻求确认就结束了 **ModelSim**。

第三课　Basic verilog Simulation

1. 新建一个目录，并设置该目录为当前工作目录，通过从该目录调用 ModelSim 或是选取 File\Change Directory 命令来完成。
2. 拷贝 example 目录中 verilog 文件到当前目录下。在你编译 verilog 文件前，你需要在新目录下生成一个设计库。如果你仅仅熟悉解释性 verilog 仿真器，诸如 Cadence Verilog-XL，那么对于你来说这是一个新的方法。因为 ModelSim 是一个编译性 Verilog 仿真器，对于编译它需要一个目标设计库。如果需要的话，ModelSim 能够编译 VHDL 和 Verilog 代码到同一个库中。
3. 在编译任何 HDL 代码前，要建立一个设计库来存放编译结果。选取 Design \ Create a New Library 生成一个新的设计库。确定选取 Create: a new library and a logical mapping to it，在 Library Name 域中键入 work，然后选取 OK。这就在当前目录中建立了一个子目录，即你的设计库。ModelSim 在这个目录中保存了名为 *_info* 的特殊文件。
   (Prompt : vlib work
   vmap work work)
4. 下面你将编译 Verilog 设计。
   这个设计例子由两个 Verilog 源文件组成，每一个都包含一个唯一的模块。文件 counter.v 包含一个名为 counter 的模块，它执行一个简单的八位加法计数器。另一个文件 tcounter.v 是一个测试台模块(test_counter)，通常用来校验 counter。在仿真下，你可以看到这两个文件，通过一个被测试台例示了的模块 counter 的一个简单的实例（名为 dut 的实例），来层次化的设置了。稍候你将有机会看一下这个代码的结构，现在，你需要编译两个文件到 work 设计库。
5. 通过选取工具条中的 Compile 按钮来编译两个文件。
   (PROMPT ： vlog counter.v tcounter.v)
   这就打开了 Compile HDL Source Files 对话框。
   选取两个文件后，选择 **Compile**,编译完成后选取 **Done**。
6. 选取工具条中的 Load Design 按钮开始仿真。
   (PROMPT ： vsim test_counter)

Load Design 对话框允许你从指定的库中选取一个设计单元仿真。你也可以为仿真选取 Simulation Resolution 限制，缺省的库是 work,缺省的 Simulation Resolution 是 1ns。

7. 选取 test_counter，点击 Load 接受这些设置。
8. 通过在主窗口下的 vsim 提示符下敲入下述命令来调出 Signals、List and Wave window:
   view signals list wave
   (MENU: View\<window name\>)
9. 为了列示顶级(**top-level**)信号，移动鼠标到 Signals window，选取 **View\List\Signals in Region**。
   (Prompt : add list /test_counter/*)
10. 现在向 Wave window 添加信号。在 Signals window 选取 Edit\Select All 选择三个信号，拖动三个信号到 Wave window 的路径名或是数值窗格的任一个中。
    HDL 条目也能够从一个窗口拷贝到另一个窗口（或者是在 Wave and List window 内部），通过 Edit \ Copy 和 Edit \ Paste 菜单命令。也能删除选取的条目 Edit \ Delete。
11. 下面打开 Source window,选取 View \ Source
    (Prompt : view source)
12. 导入设计的时候会在工作去开出一个新的 Sim Tab 栏。这个 Structure Pane 展示了设计的层次结构。你可以点即“+”（expand）或“-”（contract）来观察。
13. 点击其中的Function increment 可以注意到其他窗口是怎么适当的自动更新的。明确地说，Source window 显示了你在 Structure window 所选的层次水平的 Verilog 代码.在这种方式下使用 Structure Pane 类似于解释性 Verilog 的范围命令。现在，点击 Structure Pane 的顶层线，确定 test_counter 模块显示在 Source window。

运行仿真

1. Run 运行 100ns，缺省设置。
   (PROMPT: run) (MENU: Run\Run 100ns)
2. 设置 Run Length 为 500ns,然后 Run。
   现在仿真运行了 600ns，在工作取底部状态栏可以看到这些信息。
3. 上个命令使仿真器前进了 500ns，也可以设置仿真器推进的时间
   run @ 3000
   实际仿真器向前推进了 2400ns(3000-600)
4. 选取主窗口 Run All。
   (PROMPT: run -all) (MENU: Run\Run -All)
5. 选取 Break 中断运行。
   看 Source window，察看中断执行的语句。

调试仿真

1. 在List window选取/test_counter/count。从List window菜单条中选取 Prop \ Signal Props。Modify Signal Properties (list) 对话框打开了。

为信号counter选取十进制（在Radix）,相应的List window的输出也发生改变，成为十进制数，而不是缺省的二进制了。

2. 我们选取工作区Structure Pane中的dut:counter，然后在counter.v中的30行（这里包含一个到Verilog功能增量的调用）设置断点。
3. 选取Restart按钮，重载设计组件和重置仿真时间为零。
   (PROMPT: restart) (MENU: File\Restart)
   确认Restart对话框中所有条目被选中，然后点击Restart。
   例子中的Verilog代码中19行有一个stop语句，如果不Restart的话，将会停在这一句上。
4. 选取Run -All（主窗口），恢复执行仿真。中断后看Source window。
   (PROMPT: run -all) (MENU: Run\Run -All)
5. 正常的，当中断到达后你对一个或多个信号的值感兴趣，你有几个选项可以检测这些值。你能看显示在Signals window中的值；可以在Source window中，在变量上点右键；或者使用examine命令。
   examine count
   命令的结果是，值会输出在主窗口。
6. 执行单步跳使命令Step，遍历Verilog源函数。
7. 结束仿真的命令为：quit -force。

第四课 Mixed VHDL/verilog simulation

准备仿真

1. 生成一个新的工作目录，拷贝..\examples\mixedhdl\下的*.vhd 和*.v 文件到新目录中。设置为为当前工作目录。运行软件，如果 Welcome 对话框出现，选取 Proceed to ModelSim。
2. Select Design\Create a New Library
   （PROMPT:vlib work）
   Type Library Name:work
   Select OK!
3. 编译文件
   (PROMPT : vlog cache.v memory.v proc.v)
   (PROMPT : vcom util.vhd set.vhd top.vhd)
   打开 Compile HDL Source Files 对话框。 逐个编译 Verilog 文件。
   cache.v memeory.v proc.v
4. 依赖设计，VHDL 的编译次序是特定的。在这个例子中，top.vhd 文件必须最后编译。按照下面的顺序编译文件：
   util.vhd set.vhd top.vhd
5. 编译完成，点 Done。

运行仿真

1. 选取 Load Design 开始仿真。Load Design 对话框打开，选取 top 实体点击 Load。
   (PROMPT : vsim top)
2. View\All, (PROMPT : view *)

3. add list *
   add wave *
   (Signals MENU: View\List\Signals in Region)
   (Signals MENU: View\Wave\Signals in Region)
4. 观察一下工作区的 Structure pane。注意设计中两者的层次混合，VHDL 级的用一个方框前缀指示，Verilog 级的用一个圆形前缀指示。
5. 在 Structure pane 中点击模块 c:cache，它的源代码出现在源文件窗口。
6. 用查找功能定位 cache.v 文件中 cache_set 的声明。
   Edit\Find。
   找到了可以发现，cache_set 是 cache.v 文件内例示了的 VHDL 实体。
7. 在 Structure window，点击行“s0:cache_set(only)”。则 Source window 显示了 cache_set 实体的 VHDL 代码。
8. Quit -force

第五课　Debugging a VHDL simulation

准备仿真

1. 拷贝..\example\下的 gates.vhd，adder.vhd，testadder.vhd 文件到新建的工作目录，并定位为当前工作目录。
2. 生成一个新库：vlib library_2。
3. 在命令行的方式下敲入以下命令将源文件编译到新库中
   vcom -work library_2 gates.vhd adder.vhd testadder.vhd
4. 下一步是映射新库到工作库，可以编辑 *modelsim.ini* 文件来生成映射，或者用 vmap 命令生成一个逻辑库名字来完成。
   vmap work library_2
   ModelSim 为你修改 modelsim.ini 文件。
5. 选取 Design\Load Design，打开 Load Design 对话框。
6. 确认 simulation resolution 为缺省；在设计单元中选取名为 test_adder_structural 的配置；单击 Load 接受设置。
   (PROMPT : vsim -t ns work.test_adder_structural)
7. 打开所有的窗口。(PROMPT:View *)(MENU:View\All)
8. 在 Signals window 中选区所有信号 Edit\Select All 然后拖到 List window 中。
   (MENU:View\List\Signals in Region) (PROMPT:add list *)
9. 同样地，把信号加到 Wave window 中。键入命令：add wave *
   (MENU:View\Wave\Signals in Region)(DRAG&DROP)
10. 在主工具条上的运行时间选择器中，改变运行时间设置为 1000ns。
    (MENU:Option\Simulation\Defaults)

运行调试仿真

1. 选取 Run，运行仿真。(PROMPT：run)
   主窗口中的一条消息将通报你有一个判断错误。执行下面步骤查找错误。
2. 首先，改变仿真判断选项。选取 Option\Simulation。
3. 选取 Assertions 页面。改变选择为 Break on Assertion to Error 并点击

OK。这将使仿真停在 HDL 判断语句上。

4. 选取 Restart。(MENU:File\Restart) (PROMPT:restart)
   确定 Restart 对话框中所有条目被选，然后点击 Restart。
5. 选取 Run。可以看到 Source window 中的箭头指向判断语句。
   (MENU:Run\Run 1000 ns) (PROMPT:run)
6. 在 Variables window 中，你可以看到 i=6。这表示仿真停留在测试模式环路的第六次重复中。
7. 点击加号“+”展开名为 test_patterns 的变量。
8. 也要展开排列 test_patterns (6) 的第六次纪录。
   判断表明了 Signal window 中的 sum 不等于 Variables window 中的 sum 字段。输入 a,b 和 cin 的和应该等于输出 sum。但是在测试向量内有一个错误。为了改正这个错误，你需要重仿真且修改测试向量的初始值。
9. 执行 restart -f 命令
   参数-f 使 ModelSim 不出现确认对话框就重新仿真。
10. 在 test Process window 中选取 testbench process 更新 Variables window。
11. 再次展开 Variables window 中的 test_patterns 和 test_patterns(6)。点击变量名字，高亮显示.sum 纪录，然后选取 Edit\Change。
12. 把 value 中数值的最后四位（1000），替换为 0111，并点击 Change。（这只是暂时编辑，你必须用文本编辑器永久地改变源代码。）
13. 选取 Run。
    (MENU:Run\Run 1 us) (PROMPT:run)
    这样，仿真运行时就不会报错了。

改变 new-line 触发

缺省的，对于列出信号的每一次变化一条新线显示在 List window 中。下面的步骤将改变触发因而每 100ns 就列出这些值。

1. 在 List window 中，选取 Prop\Display Props。
2. 在 Triggers 页面完成这些步骤。
   - 取消选取 Trigger On:Signals 以禁止在信号上触发
   - 选取 Trigger On:Strobe 以开启 strobe
   - 在 Strobe Period 域键入 100
   - 在 First Strobe at 域键入 70
   - 单击 OK 接受设置
3. 最后一步将把信号 a,b 和 sum 改为十进制。选取 Prop\Signal Props，打开 Modify Signal Properties(list)对话框。
4. 选取信号，改变其属性。然后结束 ModelSim，quit -force。

第六课 Running a batch-mode simulation

批处理模式仿真必须运行在 DOS 或 UNIX 提示符下。

1. 生成一个新目录，设置成当前工作目录。拷贝..\examples\counter.vhd 到该目录下。
2. 生成一个新的设计库：vlib work
3. 映射库：vmap work work

4. 编译源文件：vcom counter.vhd
5. 使用宏文件为计数器提供激励。拷贝..\example\stim.do 文件到当前工作目录中。
6. 生成批处理文件，内容为：
   add list -decimal *
   do stim.do
   write list counter.lst
7. 执行下面的命令，运行批处理模式仿真：
   vsim -do yourfile -wlf saved.wlf counter
   - 在名为“counter”的设计单元调用 vsim 仿真器
   - 通过-wlf 这个可选项通知仿真器在名为 saved.wlf 的日志文件中保存仿真结果
   - 运行 yourfile 指定：值以十进制的方式列示出来；执行名为 stim.do 的激励；并将结果写到名为 counter.lst 的文件中。缺省的设计名为 counter。
8. 浏览仿真结果 vsim -view saved.wlf
9. 打开一些窗口 view signals list wave
10. 在窗口中放置信号 add wave *
    add list *
11. 运用 Variables windows 实验保存的仿真结果。完成了结束仿真：
    quit -f

有关批处理和命令行模式更多的信息，请参阅 *ModelSim User’s Manual*。

第七课 Executing Commands at startup

本课与第六课所介绍的工作于相同的目录，也是以命令行方式操作。

1. 这里将用到宏文件（DO）提供启动信息。拷贝..\examples\startup.do 到当前工作目录。
2. 拷贝 *modeltech* 目录下的 modelsim.ini 文件到当前工作目录。然后编辑该文件，指定一个在设计导入之后被执行的命令。用 notepad 打开 ini 文件，取消下属语句的注释，它位于文件的[vsim]部分：（修改后保存）
   Startup=do startup.do
3. 浏览这个 DO 文件，可以发现它用了一个预定义变量$entity 来为不同的设计在启动时作不同的事情。
4. 键入以下指令指定将被仿真的顶级设计单元，开始仿真：vsim counter
   注意到没有显示对话框仿真器就导入了设计单元。对于一再地仿真同一个设计单元，这样做是很便捷的。还可以注意到所有的窗口都打开了，这是因为命令 view *包括在启动宏里面。
5. 结束 ModelSim，执行 quit -f 命令。
6. 在其他例子中是不需要 startup.do 文件的，所以用文本编辑器注释掉 modelsim.ini 文件中的 Startup 这一行。

第八课 Finding names and values

Finding items by name in tree windows

你可以使用各个窗口(List,Process,Signal,Source,Structure,Variables,and

Wave window）中的查找对话框找寻你需要的 HDL 条目。Edit\Find

Searching for item values in the List and Wave windows

你可以在 List and Wave windows 中搜寻 HDL 条目的值。Edit\Search
你能够为 Signal Name(s)定位值，搜寻基于以下的选项：

- Search Type:Any Transition 搜寻选取信号的任何变化
- Search Type:Rising Edge 搜寻选取信号的上升沿
- Search Type:Falling Edge 搜寻选取信号的下降沿
- Search Type:Search for Signal Value
  搜寻 Value 域中指定的值（***符合 VHDL or Verilog 的数值格式***）
- Search Type:Search for Expression
  搜寻评测一个布尔真值的 Expression 域中的表达式

表达式可以调用一个以上的信号，但是限制在纪录于List windows上的信号。表达式可以包括常量、变量和Tcl宏。如果没有指定表达式，搜寻将返回一个错误。参阅*ModelSim Command Reference*以获取更多关于表达式语法的信息。

- Search Options:Match Count
  你能够搜寻关于值的第 n 个变化或者是第 n 个匹配。Match Count 指示了搜寻到的变化或匹配的数量。
- Search Options:Ignore Glitches
  忽略 VHDL 信号和 Verilog 网表中的零宽度的脉冲干扰。

搜寻的结果显示在对话框的底部。

第二章　使用中的注意事项

1. 如果打开ModelSim，没有出现*Welcom to ModelSim*对话框，可以在主窗口点击Help \ Enable Welcome，则以后打开ModelSim就会出现该对话框。
2. 在工作区底部的状态栏里会显示一些有用的信息。
3. 操作哪个目录中的文件一定要定位到该目录，或者是设置为当前工作目录。
4. 不能用UNIX或window命令来生成work子目录，因为里面没有*_info*文件，只能用菜单或vlib命令。
5. 断点只能设置在可执行的行上，这些行以绿色行号指示
6. 在Basic Verilog Simulation一课里面，编译两个文件的次序是不重要的（不同于被编译器指示生成的源码的从属性）。Verilog-Xl的用户可能再次感到奇怪，他们了解设计单元之间的接口检测或是编译器指示的继承关系上可能存在的问题。ModelSim推迟了这样的检测，直到设计被导入。所以在这里，如果你选择在tcounter.v之前或之后编译counter.v不存在任何问题。
7. 一组Verilog文件可以以任意次序编译，但是在一个混合VHDL/Verilog设计中，Verilog文件必须在VHDL文件值前编译。
8. force命令可以驱动clk，相当于给仿真初始化。