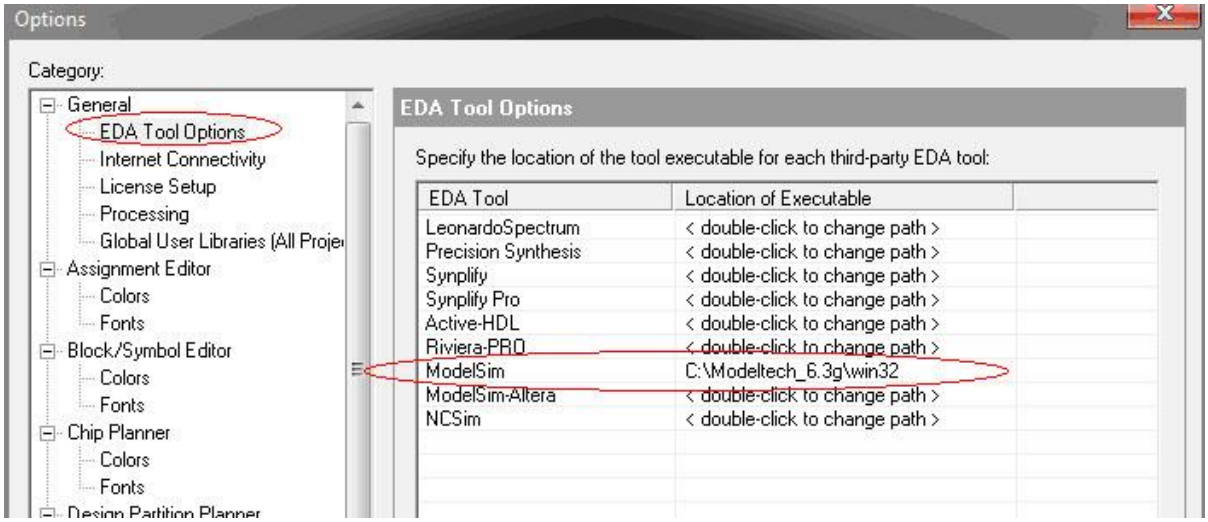
Modelsim 仿真方法 前仿真和后仿真的区别

仿真的目的是在软件环境下，验证电路的行为和设想中的是否一致。使用 Modelsim 仿真工具进行操作，**Modelsim 需要和 Quartus II 建好关联！**

建立 Quartus II 和 modelsim 的联系

① 完成上述工作之后需要在 Quartus II 中设置 modelsim 路径，Quartus II 菜单 Tools ——>General——>EDA Tool Options，进行相关设置，如 modelsim：C:\Modeltech_6.3g\win32 如图 2 所示。



modelsim 功能仿真步骤

一、启动软件

建立文件夹，在 modelsim 中选 file 下的 change directory，在其中的 choose folder 对话框中设置目录路径。

二、建立工程

选 file 的 new 下面的 project，在 project name 中填写项目名字。

项目名最好与顶层文件名一致。

project location 是工作目录，可自己选择。

default library name 可直接采用默认的 work，如此，workspace 窗口中的 library 中就会出现 work 库。

三、为工程添加文件

选择 add exsiting file 后，根据相应提示将文件添加打牌 projcet 中，包括设计顶层文件.v 以及测试向量文件.v/.vt。

四、编译文件

compile 中的 compile all。

五、装载文件

选择 library 中的 work 里面的测试向量文件，点击 simulate 下的 start simulation。

六、开始仿真

在 workspace 下的 sim 中，选择测试向量，右击，选 add 中的 add to wave，然后在 simulate 中选择 run all。

七、退出仿真

simulate 中的 end simulation。

注:

1 亦可不添加 testbench，开始的步骤均同上，在装载文件时，选中设计文件，在 sim 中点 add 里面的 add to wave。

直接对输入信号编辑测试波形（右击要编辑的信号，选中 create wave），然后点击 run 里面的 run all。

2 在 modelsim 中直接写 testbench。

先在 file 中选中 new 里面的 source，则在菜单栏里面会出现一个 source 栏目，在 source 底下勾选 show language templates，则会出现该加载项，双击其中的 create testbench，会出现一个 create testbench wizard，在 work 中选中设计文件，再点击 next，可以指定 testbench 名及要编译到的库等，默认即可，点击 finish 后自己添加内容就 ok 了。保存文件后缀为.v.

完了之后同前步骤将 testbench 文件编译到工作库里面即可。

Modelsim 的仿真分为前仿真和后仿真，下面先具体介绍一下两者的区别。

1 前仿真

前仿真也称为功能仿真，主旨在于验证电路的功能是否符合设计要求，其特点是不考虑电路门延迟与线延迟，主要是验证电路与理想情况是否一致。可综合 FPGA 代码是用 RTL 级代码语言描述的，其输入为 RTL 级代码与 Testbench.

2 后仿真

后仿真也称为时序仿真或者布局布线后仿真，是指电路已经映射到特定的工艺环境以后，综合考虑电路的路径延迟与门延迟的影响，验证电路能否在一定时序条件下满足设计构想的过程，是否存在时序违规。其输入文件为从布局布线结果中抽象出来的门级网表、Testbench 和扩展名为 SDO 或 SDF 的标准时延文件。SDO 或 SDF 的标准时延文件不仅包含门延迟,还包括实际布线延迟,能较好地反映芯片的实际工作情况。一般来说后仿真是必选的，检查设计时序与实际的 FPGA 运行情况是否一致，确保设计的可靠性和稳定性。选定了器件分配引脚后在做后仿真。

3 Modelsim 仿真的基本步骤

Modelsim 的仿真主要有以下几个步骤：

(1)建立库并映射库到物理目录；

(2)编译原代码（包括 Testbench）；

(3)执行仿真。

上述 3 个步骤是大的框架，前仿真和后仿真均是按照这个框架进行的，建立 modelsim 工程对前后仿真来说都不是必须的。

3.1 建立库

在执行一个仿真前先建立一个单独的文件夹，后面的操作都在此文件下进行，以防止文件间的误操作。然后启动 Modelsim 将当前路径修改到该文件夹下，修改的方法是点 File->Change Directory 选择刚刚新建的文件夹见下图。

图 3 新建文件夹

做前仿真的时候，推荐按上述建立新的文件夹。

做后仿真的时候，在 Quartus II 工程文件夹下会出现一个文件夹：工程文件夹\simulation\modelsim，前提是正确编译 Quartus II 工程；因此，不必再建立新的文件夹了。

仿真库是存储已编译设计单元的目录，modelsim 中有两类仿真库，一种是工作库，默认的库名为 work,另一种是资源库。Work 库下包含当前工程下所有已经编译过的文件。所以编译前一定要建一个 work 库，而且只能建一个 work 库。资源库存放 work 库中已经编译文件所要调用的资源，这样的资源可能有很多，它们被放在不同的资源库内。例如想要对综合在 cyclone 芯片中的设计做后仿真，就需要有一个名为 cyclone_ver 的资源库。

映射库用于将已经预编译好的文件所在的目录映射为一个 modelsim 可识别的库，库内的文件应该是已经编译过的，在 Workspace 窗口内展开该库应该能看见这些文件，如果是没有编译过的文件在库内是看不见的。

建立仿真库的方法有两种。一种是在用户界面模式下，点 File->New->Library 出现下面的对话框，选择 a new library and a logical mapping to it,在 Library Name 内输入要创建库的名称，然后 OK，即可生成一个已经映射的新库。另一种方法是在 Transcript 窗口输入以下命令：

vlib work

vmap work work

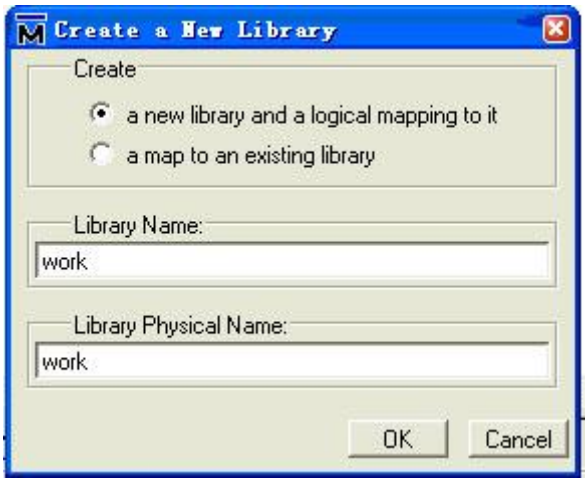


图 4 建立仿真库

如果要删除某库，只需选中该库名，点右键选择 Delete 即可。

需要注意的是不要在 modelsim 外部的系统盘内手动创建库或者添加文件到库里；也不要 modelsim 用到的路径名或文件名中使用汉字，因为 modelsim 可能无法识别汉字而导致莫名其妙的错误。

3.2 编写与编译测试文件

在编写 Testbench 之前，最好先将要仿真的目标文件编译到工作库中，点 Compile->Compile 或 ，将出现下面的对话框，

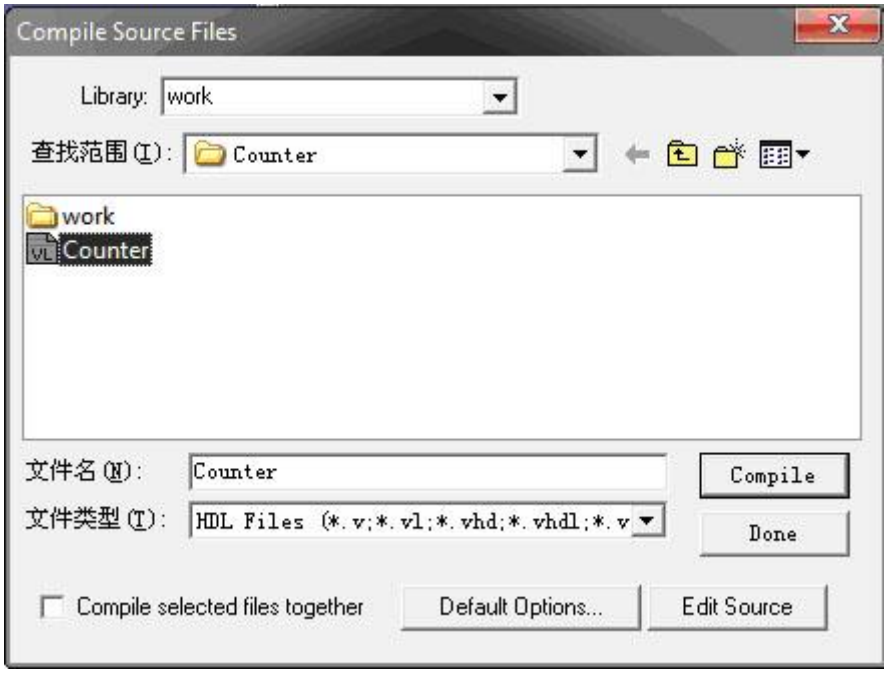


图 5 编译目标文件

在 Library 中选择工作库，在查找范围内找到要仿真的目标文件(Library 选择刚才建立的库，查找范围选择目标文件所在的文件夹)，然后点 Compile 和 Done；或在命令行输入

vlog Counter.v。此时目标文件已经编译到工作库中，在 Library 中展开 work 工作库会发现该文件。

当对要仿真的目标文件进行仿真时需要给文件中的各个输入变量提供激励源，并对输入波形进行的严格定义，这种对激励源定义的文件称为 Testbench，即测试台文件。下面先讲一下 Testbench 的产生方法。

方法一：我们可以在 modelsim 内直接编写 Testbench，而且 modelsim 还提供了常用的各种模板。具体步骤如下：

⑴ 执行 File->New->Source->verilog，或者直接点击工具栏上的新建图标，会出现一个 verilog 文档编辑页面，在此文档内设计者即可编辑测试台文件。需要说明的是在 Quartus 中许多不可综合的语句在此处都可以使用，而且 testbench 只是一个激励源产生文件，只要对输入波形进行定义以及显示一些必要信息即可，切记不要编的过于复杂，以免喧宾夺主。

⑵ Modelsim 提供了很多 Testbench 模板，我们直接拿过来用可以减少工作量。在 verilog 文档编辑页面的空白处右键点 Show Language Templates 然后会出现一个加载工程，接着你会发现在刚才的文档编辑窗口左边出现了一个 Language Templates 窗口，见下图。

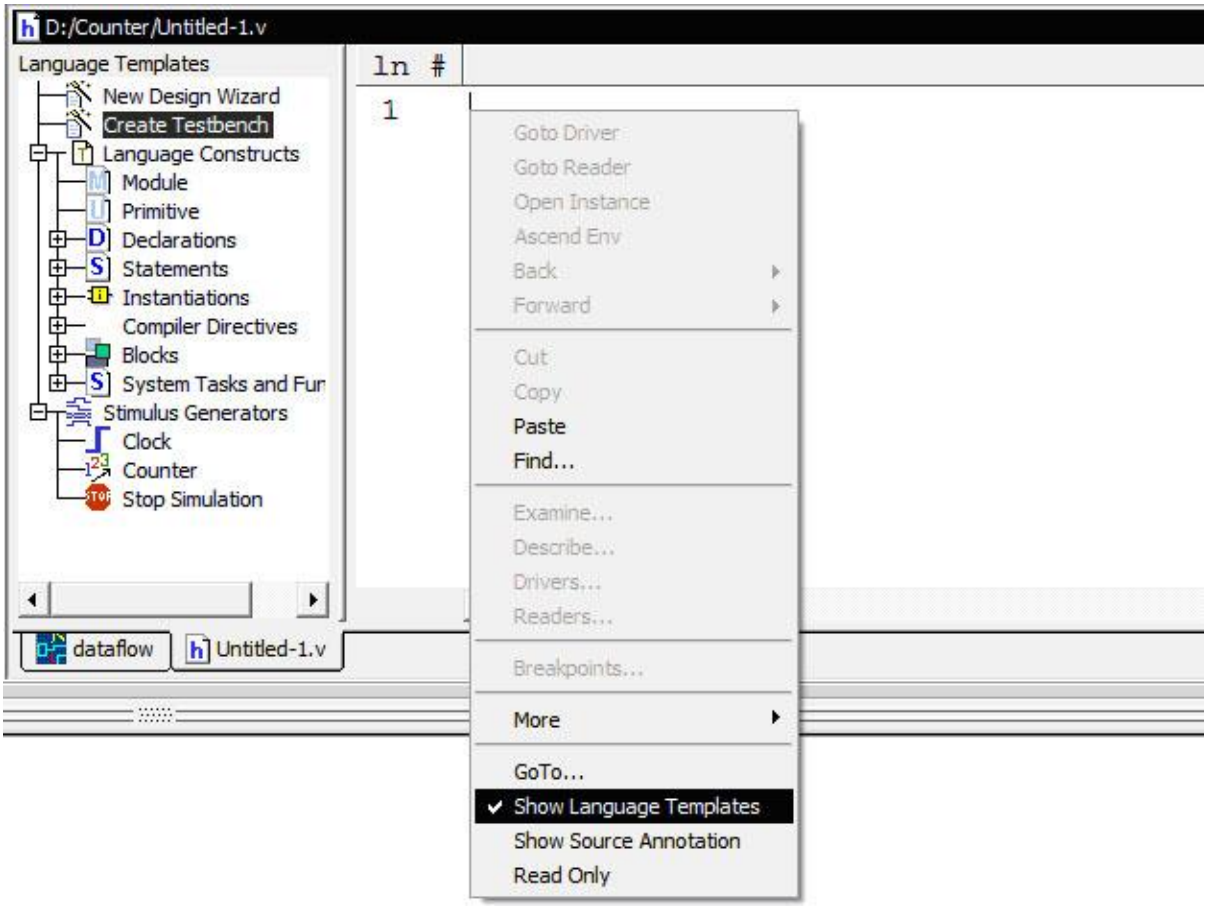


图 6 应用模板生成 Testbench 文件

双击 Creat Testbench 会出现一个创建向导，见下图。

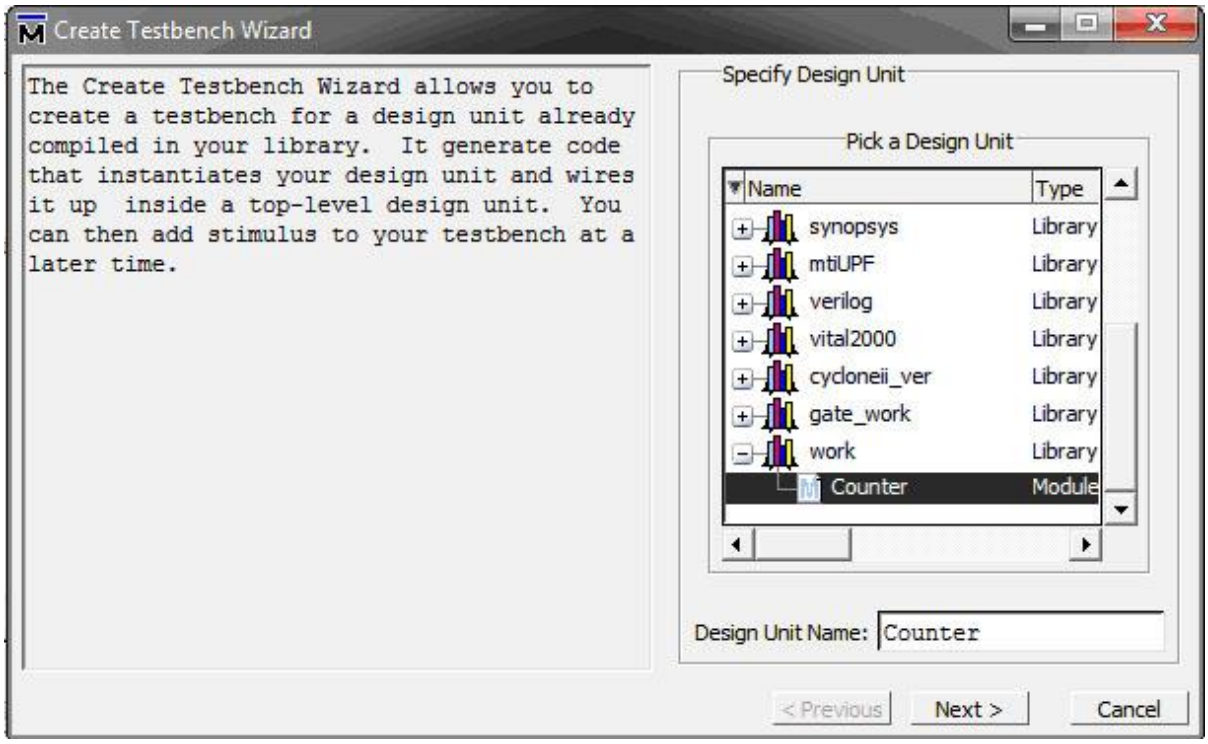


图 7 创建向导

选择 Specify Design Unit 工作库下，work 工作库下的目标文件，点 Next,出现下面对话框：

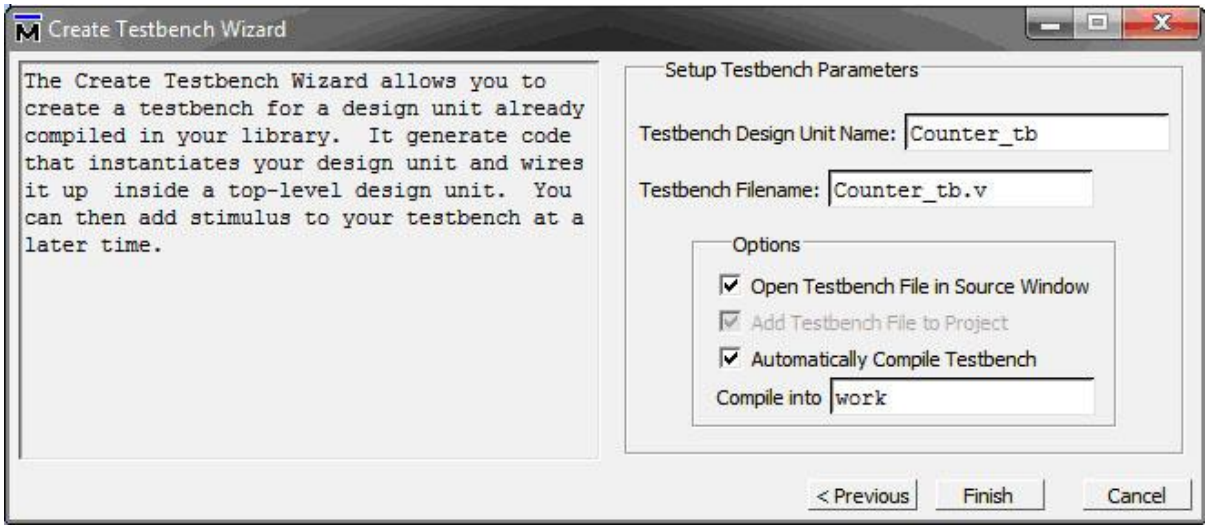


图 8 设置 Testbench 向导

可以指定 Testbench 的名称以及要编译到的库等，此处我们使用默认设置直接点 Finish。这时在 Testbench 内会出现对目标文件的各个端口的定义还有调用函数接下来，设计者可以自己往 Testbench 内添加内容了(有注释的为添加的内容)，然后保存为.v 格式即可。按照前面的方法把 Testbench 文件也编译到工作库中。

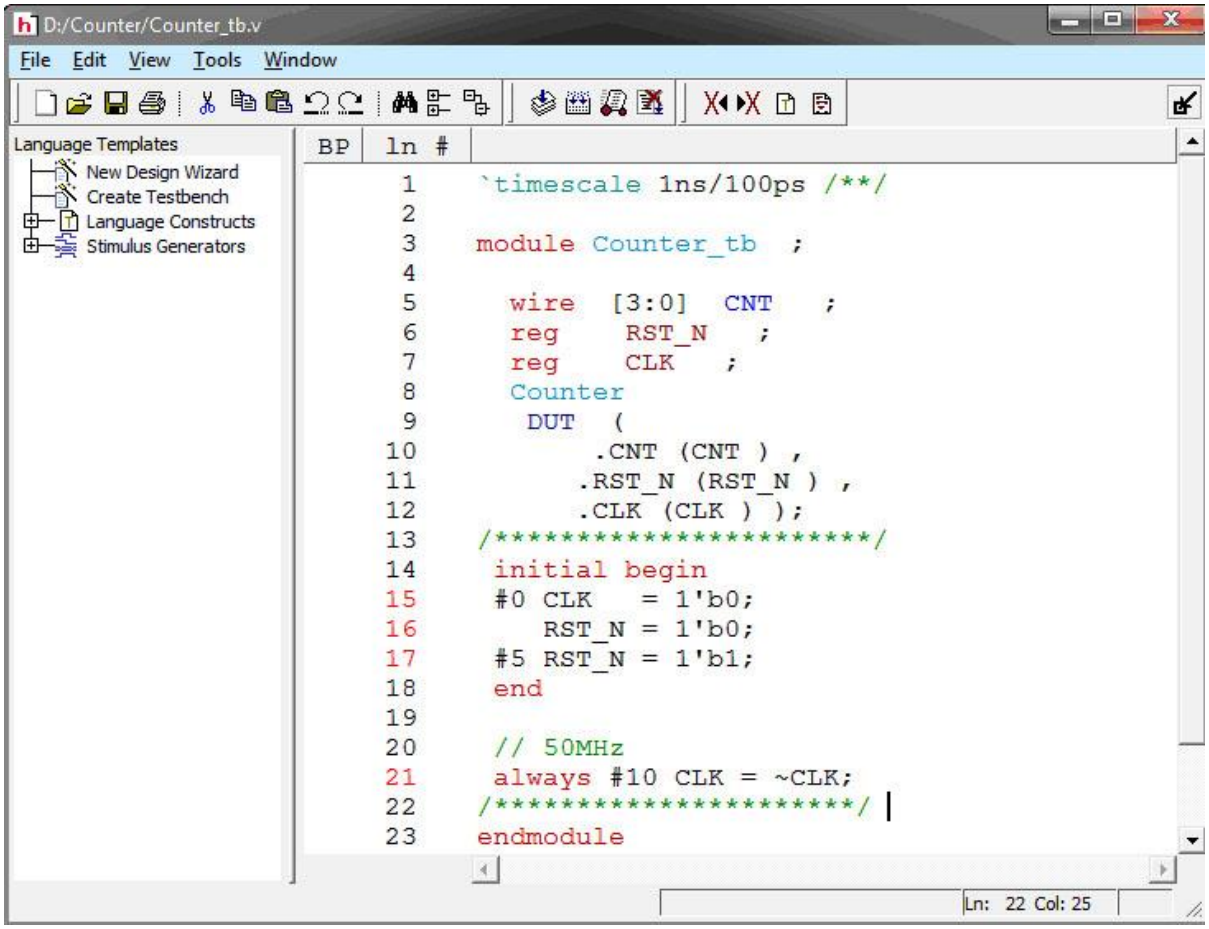


图 9 生成及修改后的 Testbench 文件

方法二：在 Quartus II 内编写并编译 Testbench ，之后将 Testbench 和目标文件放在同一个文件夹下，按照前面的方法把 Testbench 文件和目标文件都编译到工作库中之后。

PS：如果在工作库中没有该文件(在 Testbench 文件没有端口的情况下)，则在 Simulate——>Start Simulate 卡片中去掉优化选项，如下图所示。之后再重新编译，即可在工作库中找到该文件。

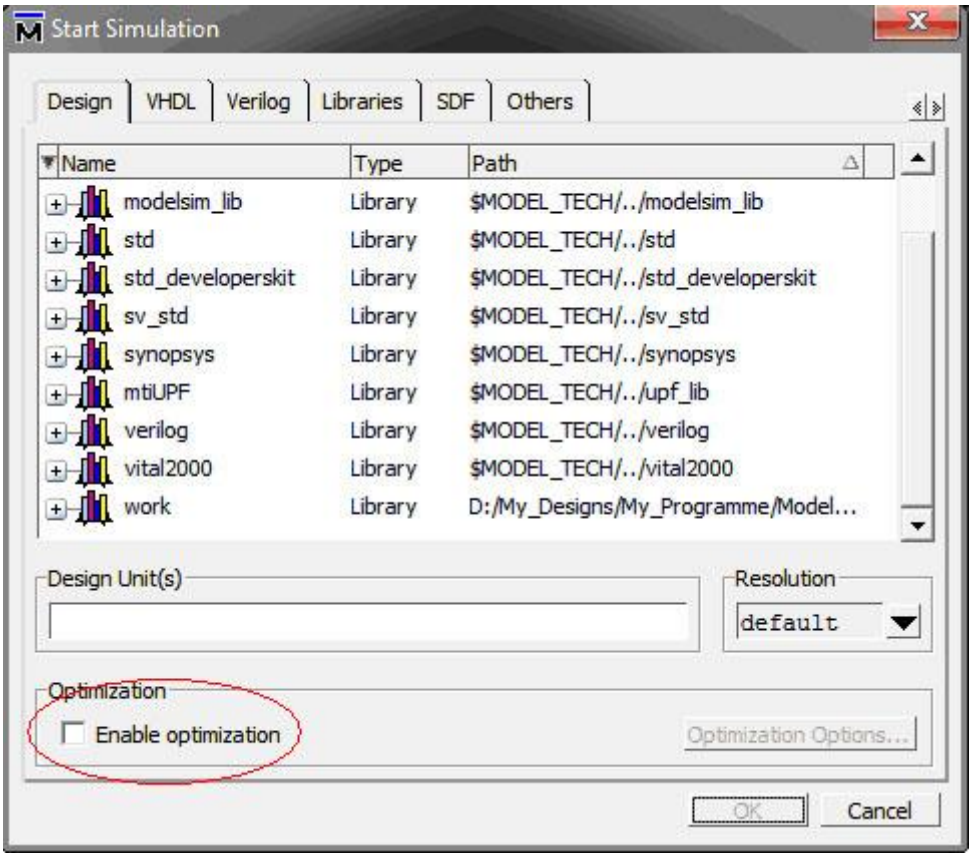


图 10 去掉优化选项

3.3 执行仿真

因为仿真分为前仿真和后仿真，下面分别说明如何操作。

⑴ 前仿真

前仿真，相对来说是比较简单的。在上一步我们已经把需要的文件编译到工作库内了，现在我们只需点 simulate->Start Simulation 或快捷按钮 会出现 start simulate 对话框。点击 Design 标签选择 Work 库下的 Testbench 文件，然后点 OK 即可，也可以直接双击 Testbench 文件 Counter_tb.v，此时会出现下面的界面。

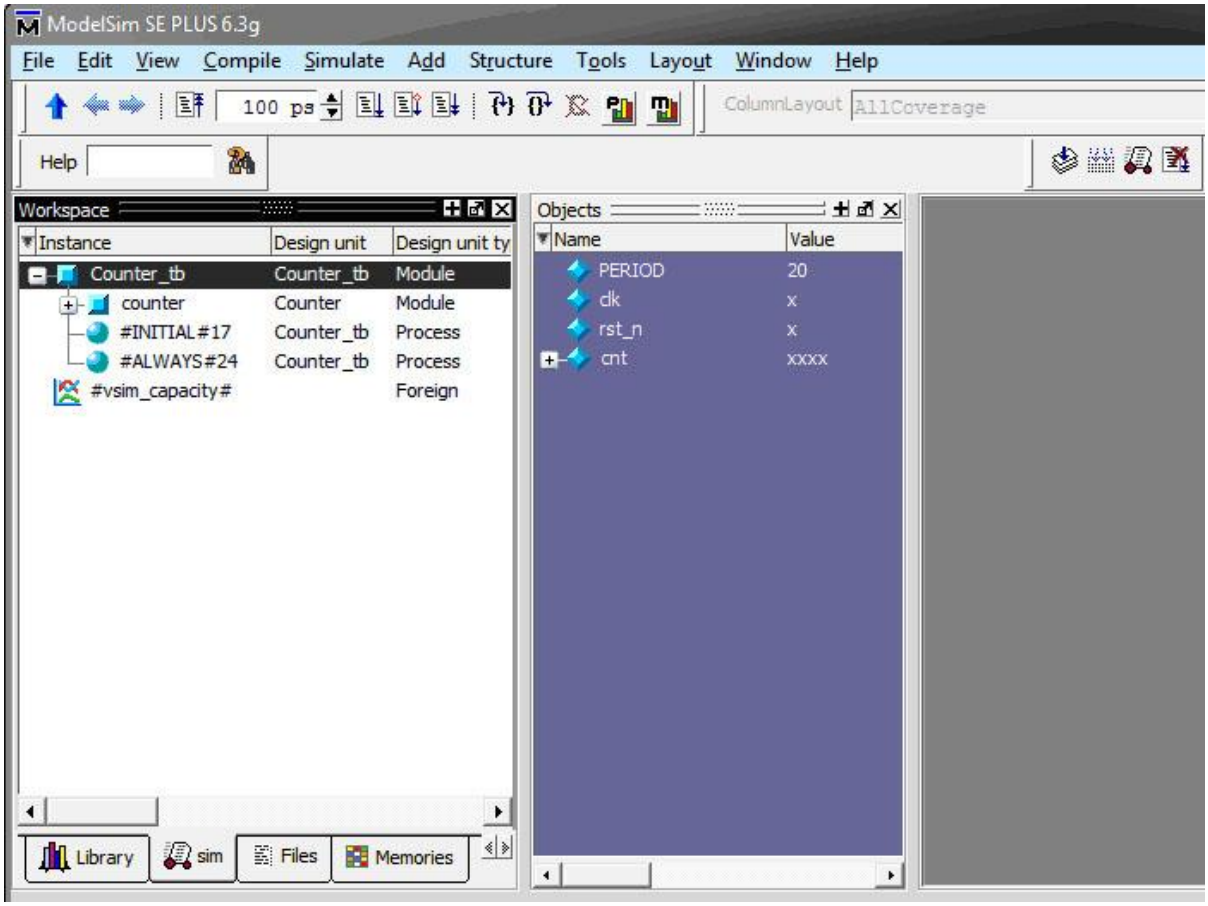


图 11 start simulate

在主界面中会多出来一个Objects窗口，里面显示 Testbench 里定义的所有信号引脚，在Workspace里也会多出来一个Sim标签。右键点击Counter_tb.v，选择Add->Add to Wave，如下图所示。然后将出现Wave窗口，现在就可以仿真了，见下图。

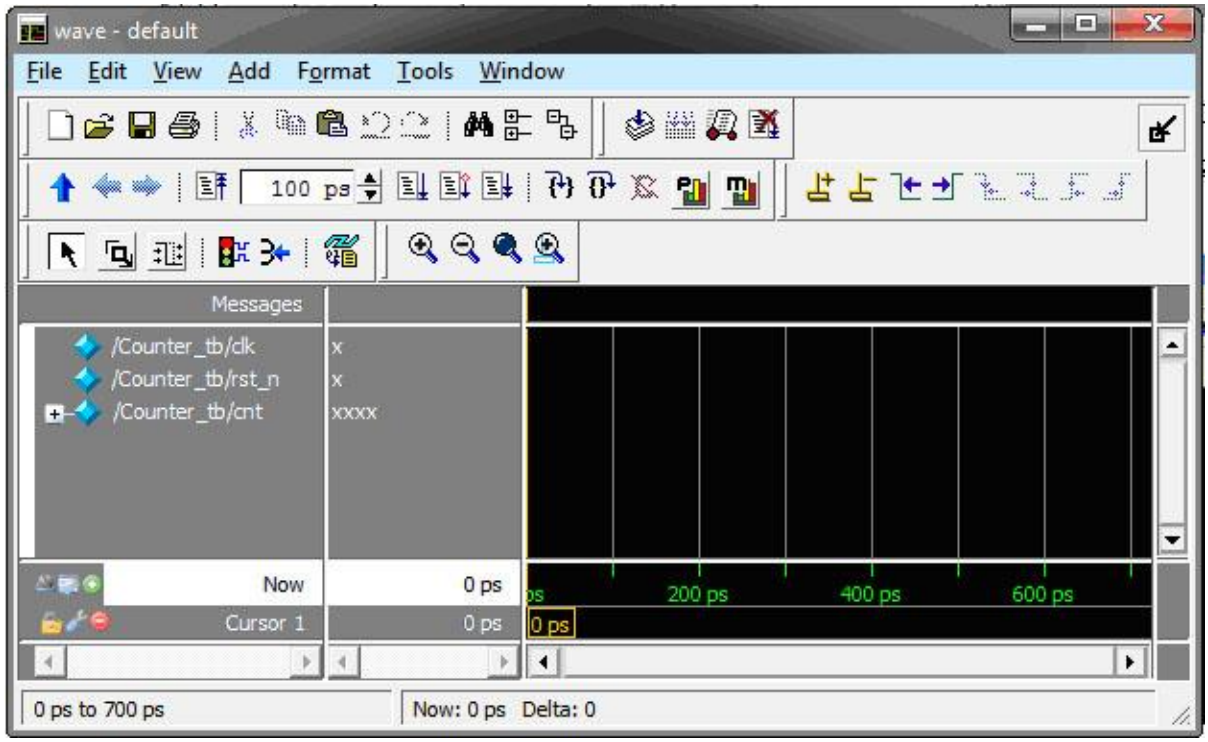


图 12 Wave 窗口

窗口里面已经出现了待仿真的各个信号,点 将开始执行仿真到 100ns,继续点仿真波形也将继续延伸，见下图。

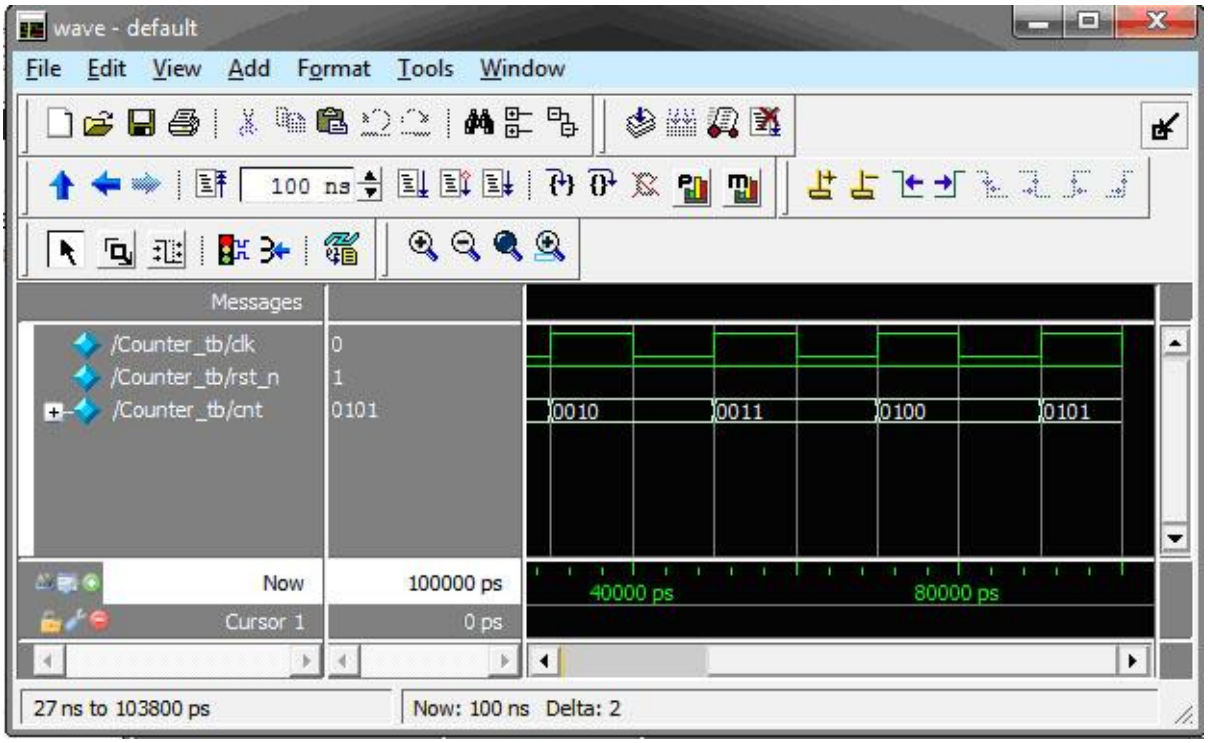


图 13 仿真波形

若点 ，则仿真一直执行，直到点 才停止仿真。

也可以在命令行输入命令： run @1000

则执行仿真到 1000ns,后面的 1000 也可以是别的数值，设计者可以修改。在下一次运行该命令时将接着当前的波形继续往后仿真。 至此，前仿真步骤完成。

⑵ 后仿真

这里是采用的 Cyclone ii 做的一个 counter 的例子。

后仿真与前仿真的步骤大体相同，只不过中间需要添加仿真库(和所选器件及所有 IP Core 相关)、网表和延时文件的步骤。

后仿真的前提是 quartus 已经对要仿真的目标文件进行编译，并生成 modelsim 仿真所需要的.vo 文件(网表文件)和.sdo 文件（时延文件），具体操作过程又有两种方法，一种是通过 Quartus 调用 Modelsim，Quartus 在编译之后自动把仿真需要的.vo 文件以及需要的仿真库加到 modelsim 中，操作简单；一种是手动将需要的文件和库加入 modelsim 进行仿真，这种方法可以增加主观能动性，充分发挥 modelsim 的强大仿真功能。

① 通过 Quartus 调用 Modelsim

使用这种方法时首先要对 Quartus 进行设置。 先运行 Quartus，打开要仿真的工程，点菜单栏的 Assignments，点 EDA Tool settings，选中左边 Category 中的 Simulation.，在右边的 Tool name 中选 ModelSim(Verilog),选中下面的 Run Gate Level Simulation automatically after complication.见下图。

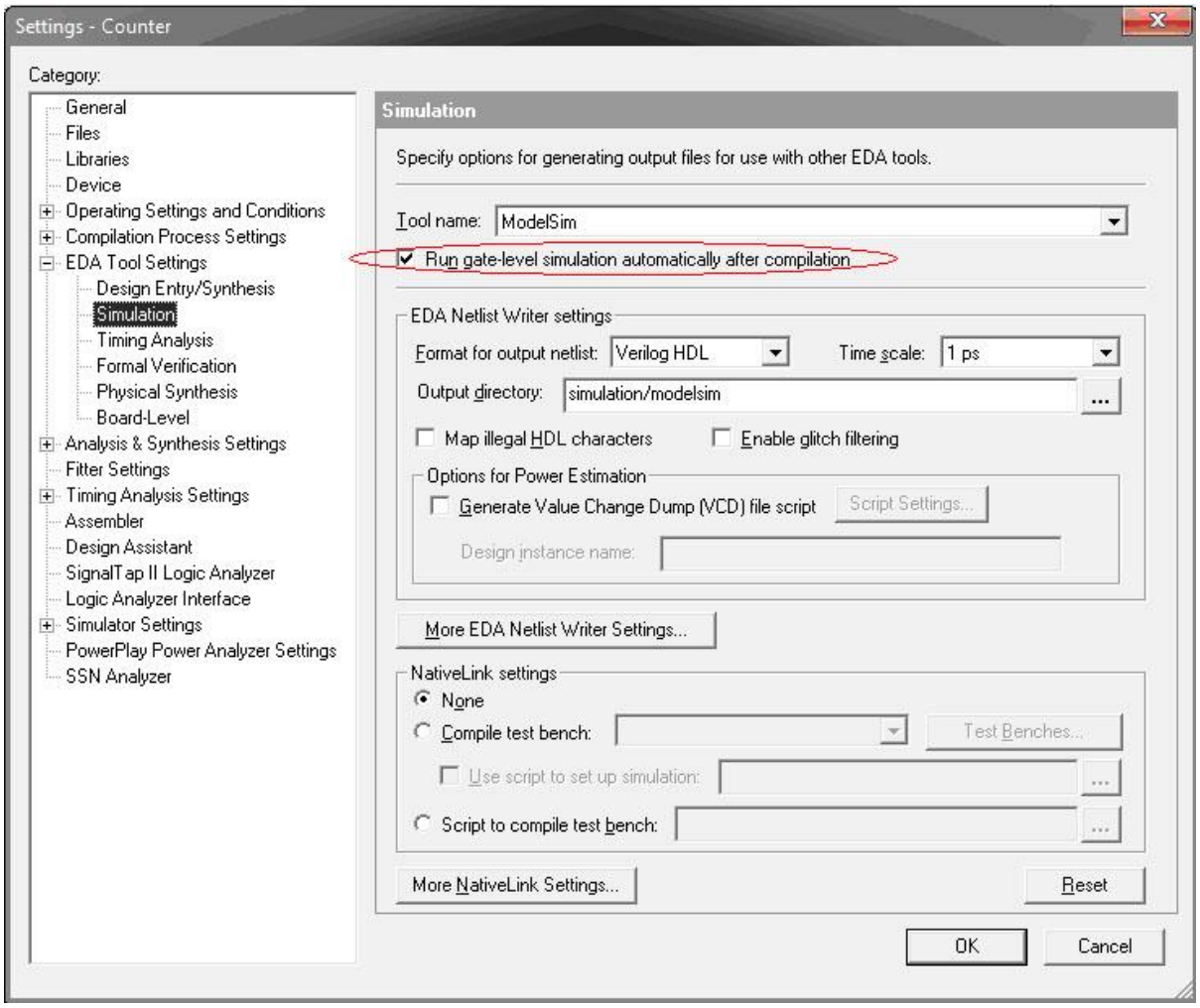


图 14 对 Quartus 进行设置

Quartus 中的工程准备好之后点击 start complication 按钮，此时 modelsim 会自动启动，而 quartus 处于等待状态（前提是系统环境变量中用户变量中 PATH 要设置好 modelsim 安装路径,如：D:\Modeltech_6.3\win32）。在打开的 modelsim 的 Workspace 窗口中你会发现多了工作库和资源库，而且 work 库中出现了需要仿真的文件。Modelsim 自动将 quartus 生成的.vo 文件编译到 work 库，并建立相应的资源库。如图所示。

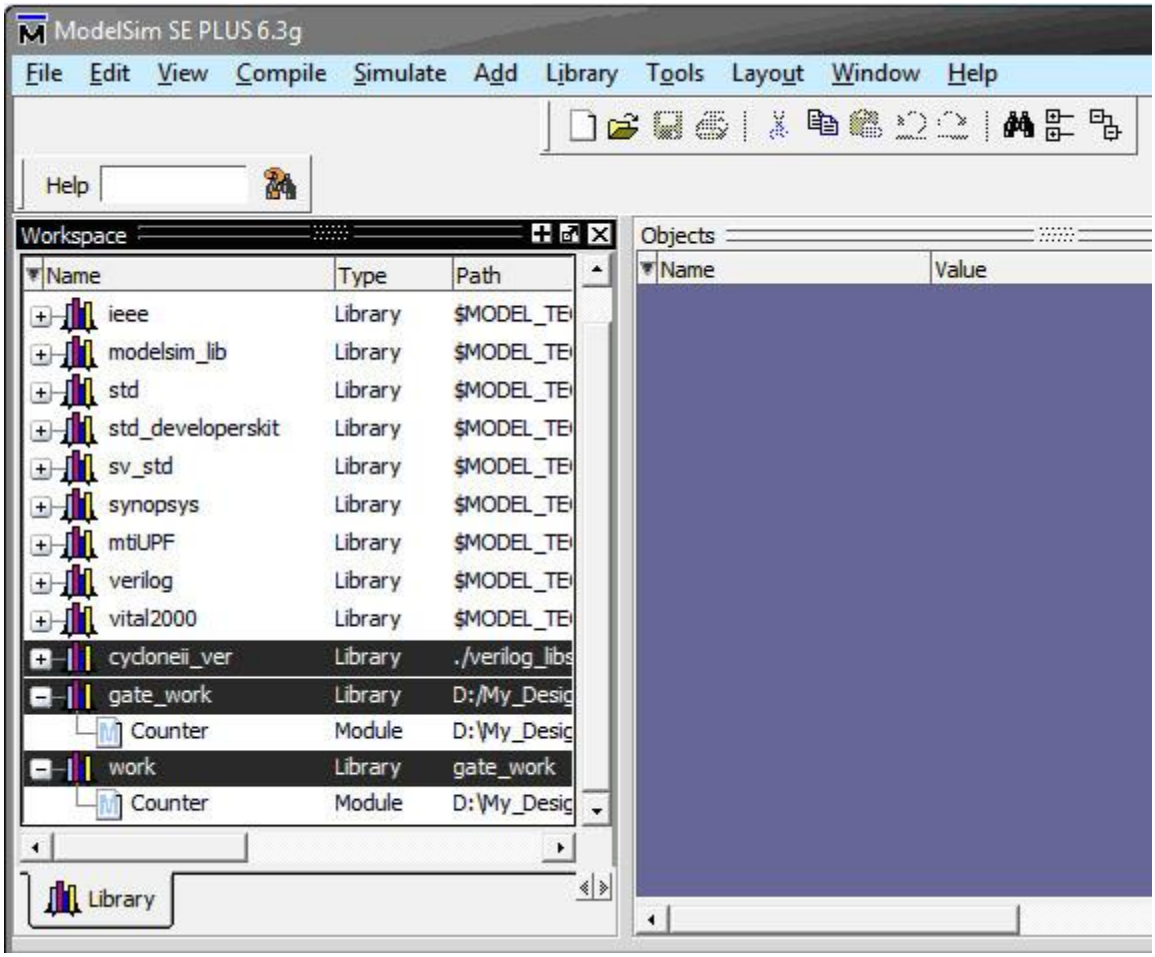


图 15 Quartus II 编译之后自启动 modelsim

观察库，可以发现，多了 verilog_libs 库、gate_work 库和 work 库，但是在"工程文件夹\simulation\modelsim"下，只有 verilog_libs 和 gate_work 文件夹，为什么库里面却多了一个 work 库呢？而且 gate_work 库和 work 库文件内容相同！

gate_work 库是 Quartus II 编译之后自动生成的，而 work 库是 modelsim 默认库。仔细观察二者路径，二者路径相同，均为 gate_work 文件夹，可知 modelsim 将 gate_work 库映射到 work 库。因此，在后续的工作中操作 gate_work 库或者 work 库都能得到正确结果。

编写测试台程序 Counter_tb.v ，最好放在生成的.vo 文件所在的目录，以方便在需要手动仿真时使用。点 Compile 在出现的对话框中选中 Counter_tb.v 文件，然后点 Compile 按钮,编译结束后点 Done，这时在 Work 库中会出现测试台文件。如下图所示。

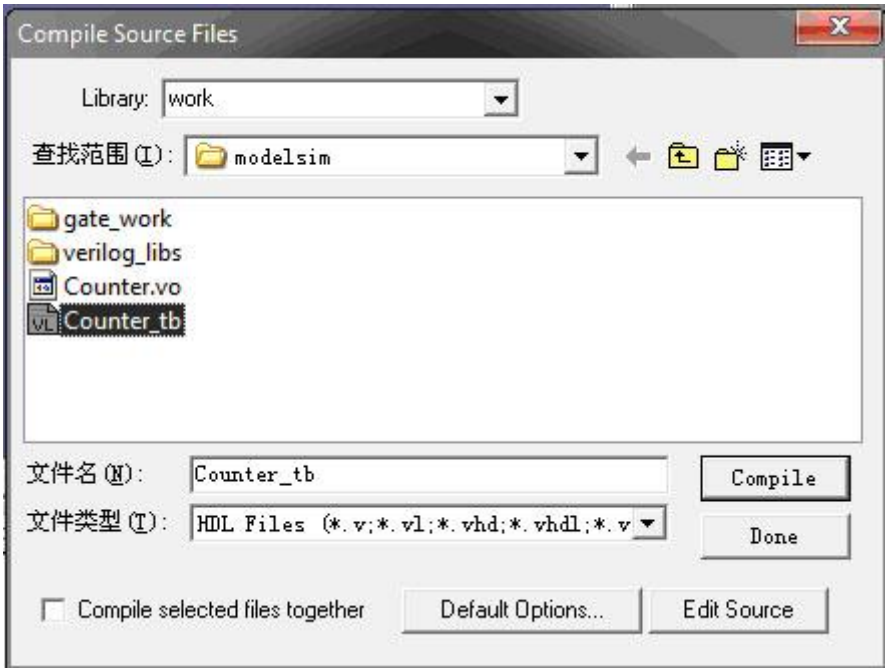


图 16 编译测试文件

点击 simulate->Start Simulation 或快捷按钮 会出现 start simulate 对话框。点击 Design 标签选择 Work 库下的 Counter_tb.v 文件，然后点击 Libraries 标签在 Search Library 中点击 Add 按钮，选择仿真所需要的资源库（如果不知道需要选择哪个库，可以先直接点 Compile 看出现的错误提示中说需要的库名，然后再重复上述步骤）见下图。

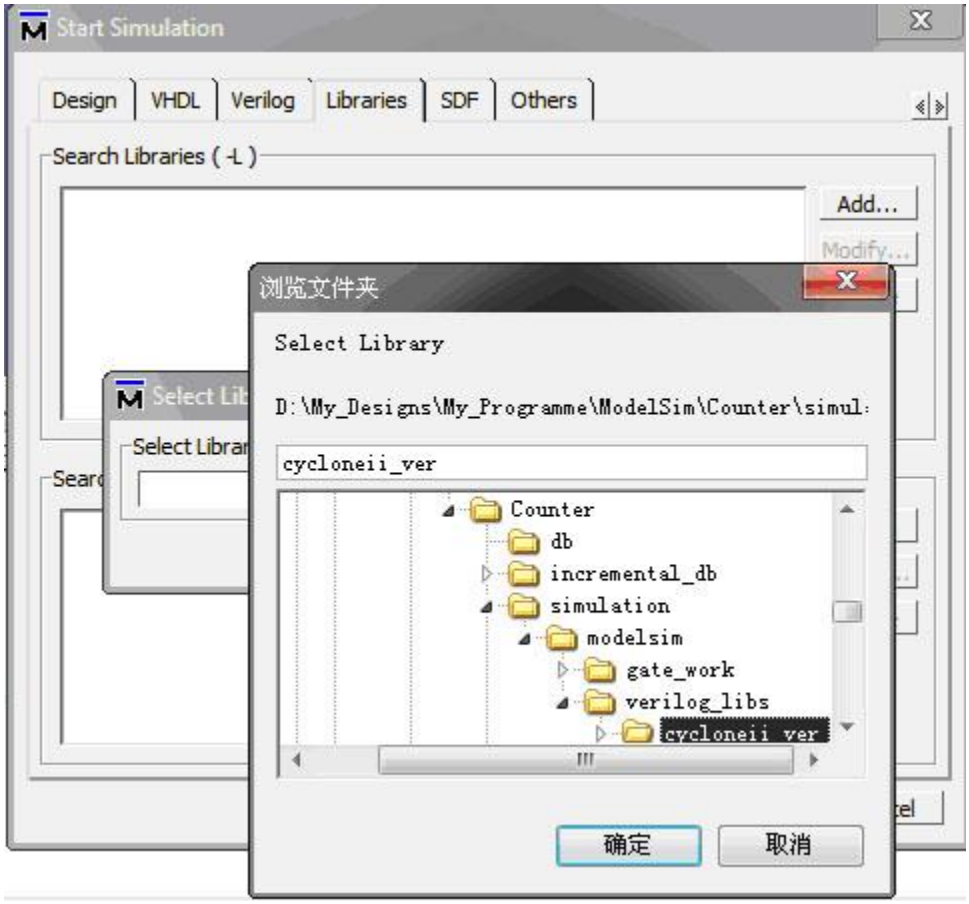


图 17 选择仿真所需要的资源库

再点 start simulate 对话框的 SDF 标签。在出现的对话框的 SDF File 框内加入.sdo 时延文件路径。在 Apply To Region 框内有一个“/”， 在“/”的前面输入测试台文件名，即“Counter_tb”,在它的后面输入测试台程序中调用被测试程序时给被测试程序起的名称，本例中为“DUT”，见下图。然后点 OK。后面观察波形与前仿真步骤相同。

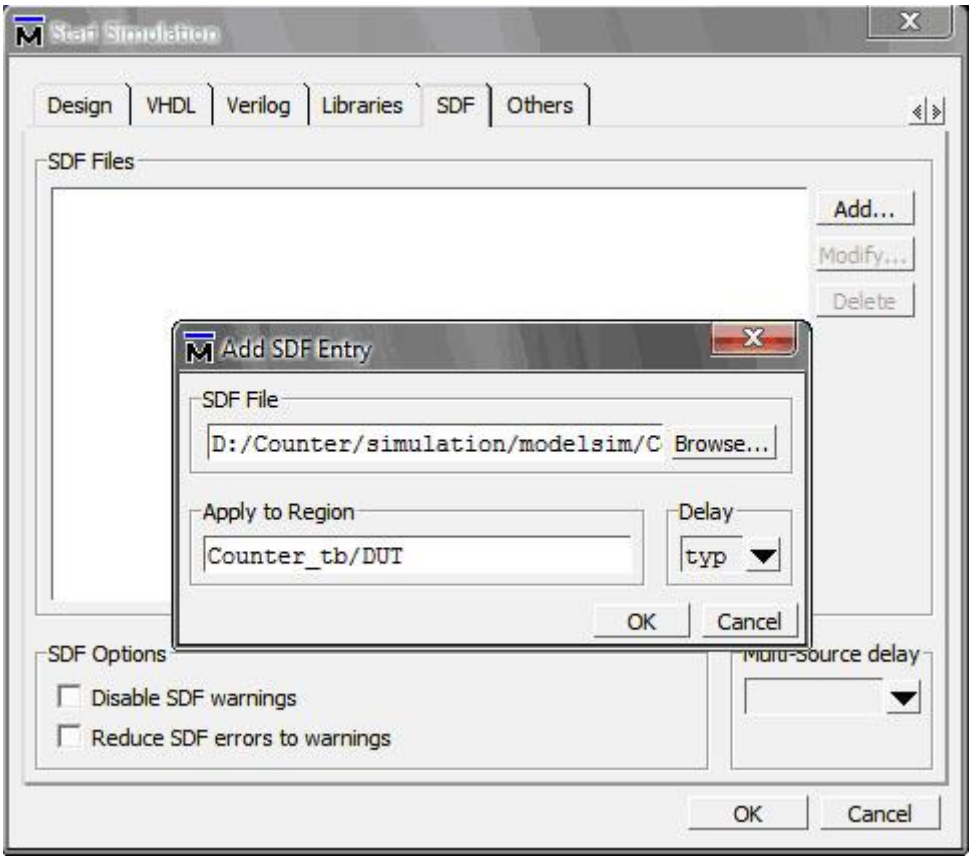


图 18 添加.sdo 文件

自动仿真和手动仿真的区别：

这种方法比较简单，因为 Quartus II 调用 Modelsim ，所以除了生成自动生成了 modelsim 仿真所需要的.vo 文件(网表文件)和.sdo 文件(时延文件)外,还生成了 gate_work 文件夹、verilog_libs 文件夹；gate_work 文件夹(可以叫工作库，也可以叫编译库)下存放了已编译的文件，verilog_libs 文件夹下存放了仿真所需要的资源库，上例是 cycloneii_ver 库(文件夹)。而手动仿真则需要自己添加这些文件和库。具体如下。

② 手动仿真

手动仿真需要自己添加文件和编译库，但可以充分发挥 modelsim 强大的仿真功能。操作时也要先对 quartus 进行设置,设置与前面相同只是不要选中 Run Gate Level Simulation automatically after complication。然后启动 modelsim，将当前路径改到"工程文件夹\simulation\modelsim"下。如下图所示。

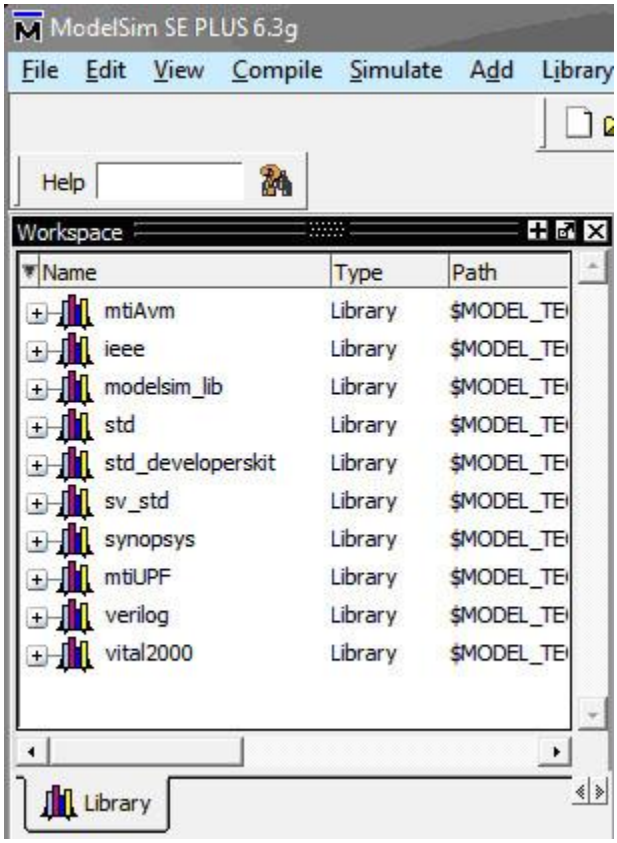


图 19 启动 modelsim

相比①中，这里少了一些库(实际是 verilog_libs 库、gate_work 库和 work 库)，因此下面要添加一个库。新建一个库，此处默认库名为 work，此时，"工程文件夹\simulation\modelsim"文件夹下出现了一个 work 文件夹，work 库下面没有目标文件和测试文件，即 work 文件夹下没有任何文件，建库的目的就是将编译的文件都放在该库里，即放在该文件夹下。编译之前，还需要添加仿真所需要的资源库 cycloneii_atoms(用到 EP2C8)，将 D:\altera\90\quartus\eda\sim_lib 目录下的 cycloneii_atoms 文件复制到.vo 所在的目录，即"工程文件夹\simulation\modelsim"下。

如果按照①中的方法编写 testbench 并同样放在.vo 所在的目录，这时点 Compile 下的 Compile 或点 将会出现下面的对话框，将所选文件进行编译。

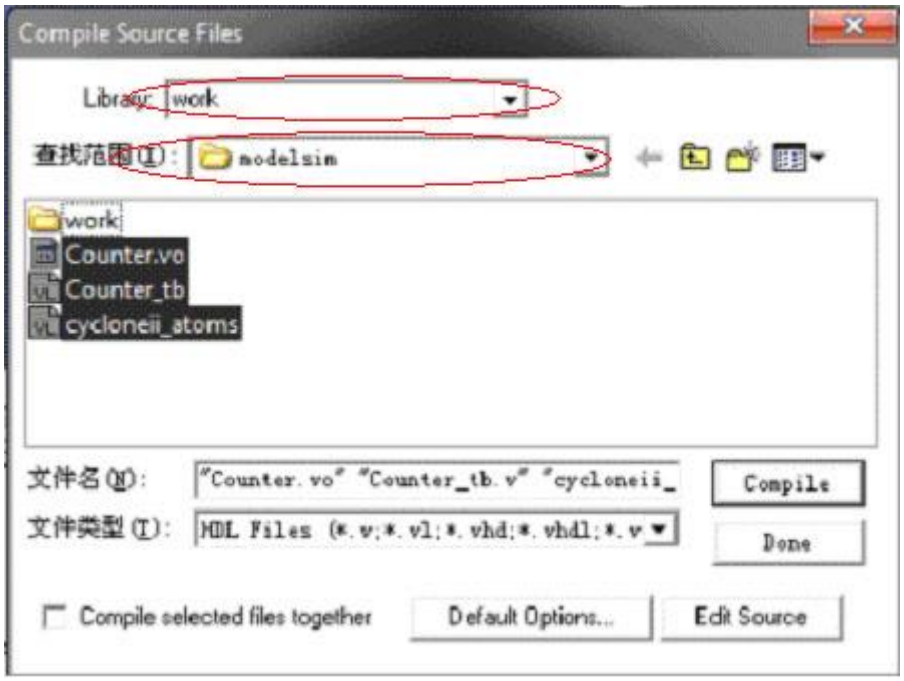


图 20 编译所需文件和资源库

编译完成之后，work 工作库下多了很多文件，同样 work 文件夹下也多了很多文件(夹)，建库的目的可见一斑，其中有 Counter_tb 测试文件和 counter 目标文件。点击 simulate->Start Simulation 或快捷按钮 会出现 start simulate 对话框。这里和①相比只有 Libraries 标签在 Search Library 时不一样，其余 2 项都一样。Libraries 标签在 Search Library 的设置如下图。

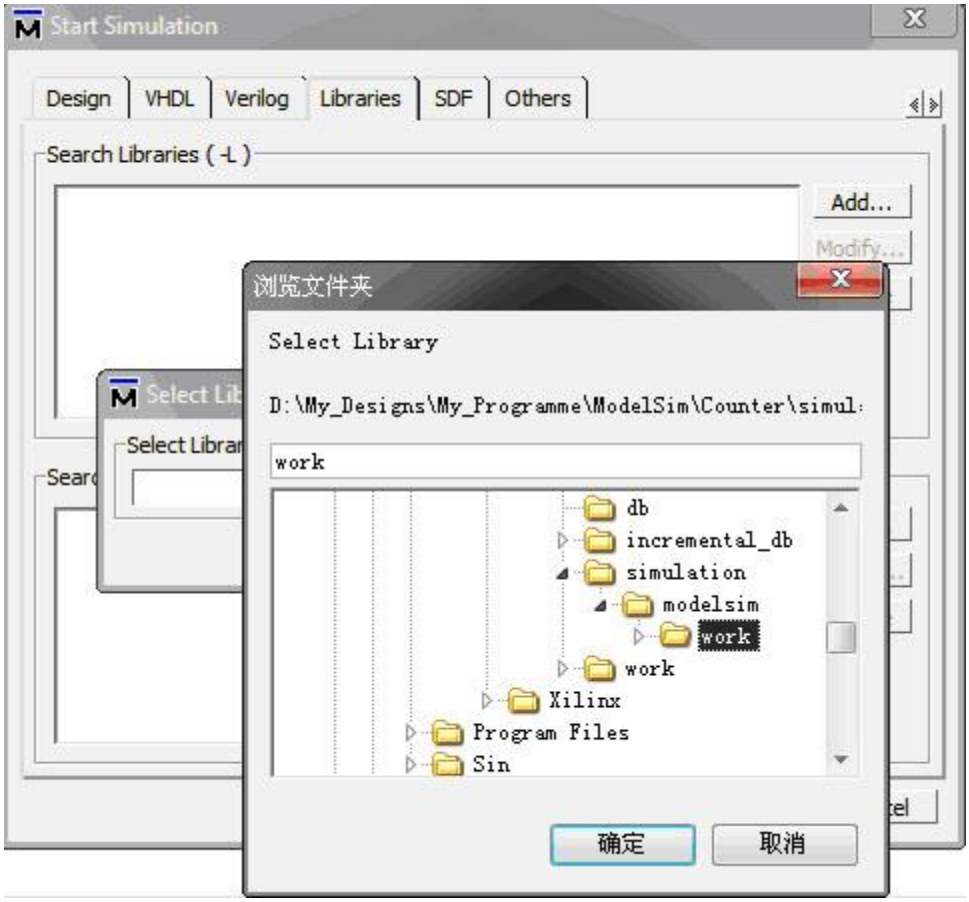


图 21 选择仿真所需要的资源库

后面的观察波形步骤跟前面一样。

四、观察波形的一些方法和技巧。

4.1 手动创建输入波形(待定)

对于复杂的设计文件，最好是自己编写 testbench 文件，这样可以精确定义各信号以及各个信号之间的依赖关系等，提高仿真效率。

对于一些简单的设计文件，也可以在波形窗口自己创建输入波形进行仿真。具体方法是鼠标右击 work 库里的目标仿真文件 counter.v，然后点 create wave，弹出 wave default 窗口。如下图所示。

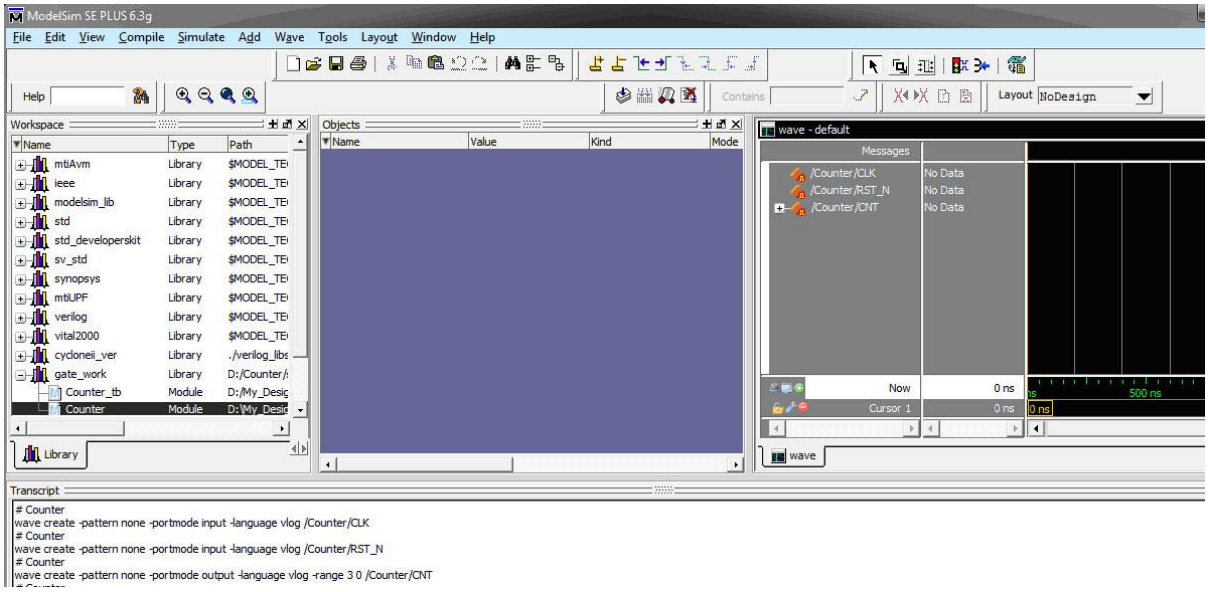


图 22 Add to Wave

在 wave 窗口中选中要创建波形的信号，如此例中的 CLK，然后右键点击，选择 Create/Modify/Wave 项出现下面的窗口：

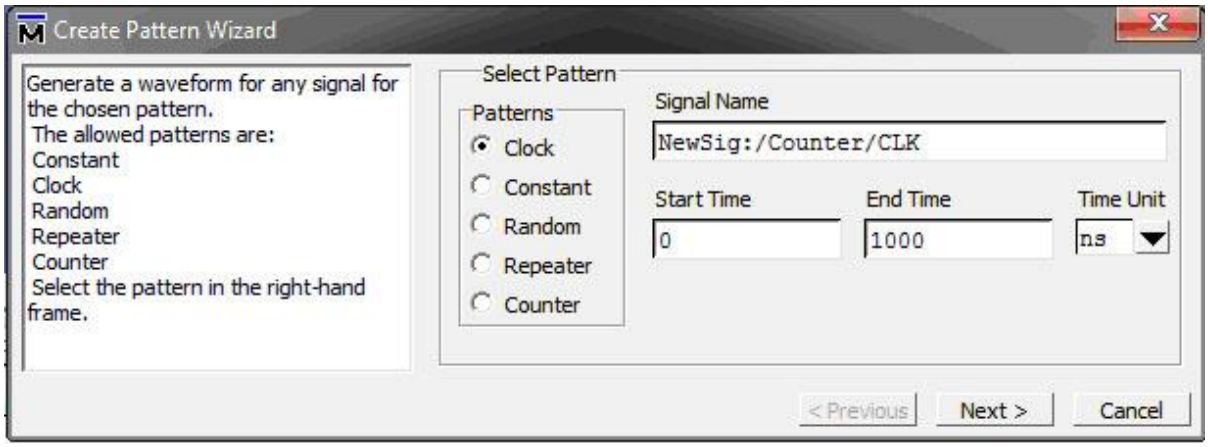


图 23 设置输入波形

在 Patterns 中选择输入波形的类型，然后分别在右边的窗口中设定起始时间、终止时间以及单位，再点 Next 出现下面的窗口，我们把初始值的 HiZ 改为 0，然后修改时钟周期和占空比，然后点 Finish。

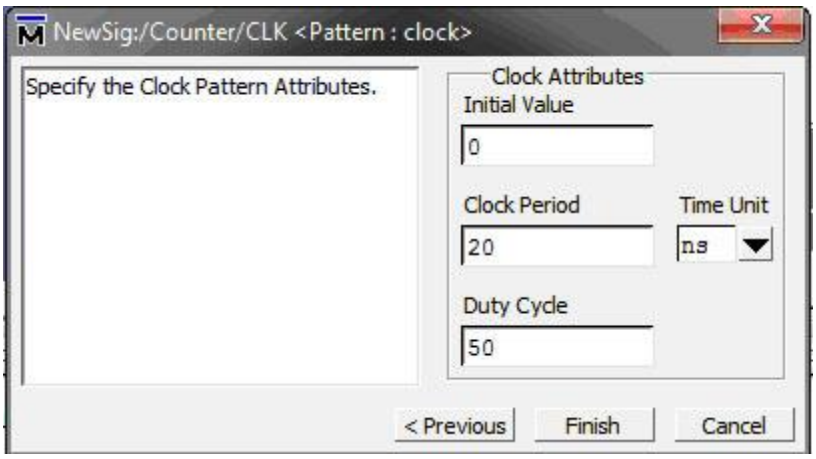


图 24 设置输入波形

接着继续添加其他输入波形，出现下面的结果。前面出现的红点表示该波形是可编辑的。后面的操作与用 testbench 文本仿真的方法相同 。

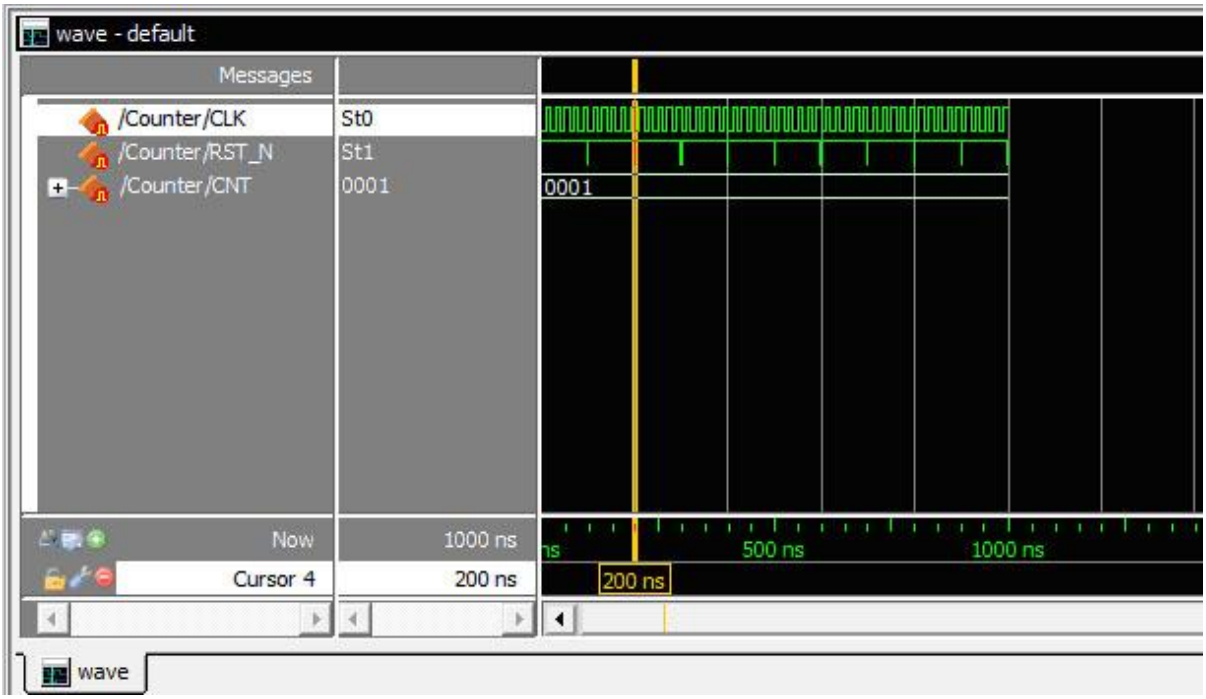


图 25 仿真波形

4.2 观察特定信号波形

如果设计者只想查看指定信号的波形，可以先选中 objects 窗口中要观察的信号，然后点右键选择 Add to Wave->Selected signals,见下图，那么在 Wave 窗口中只添加选中的信号。

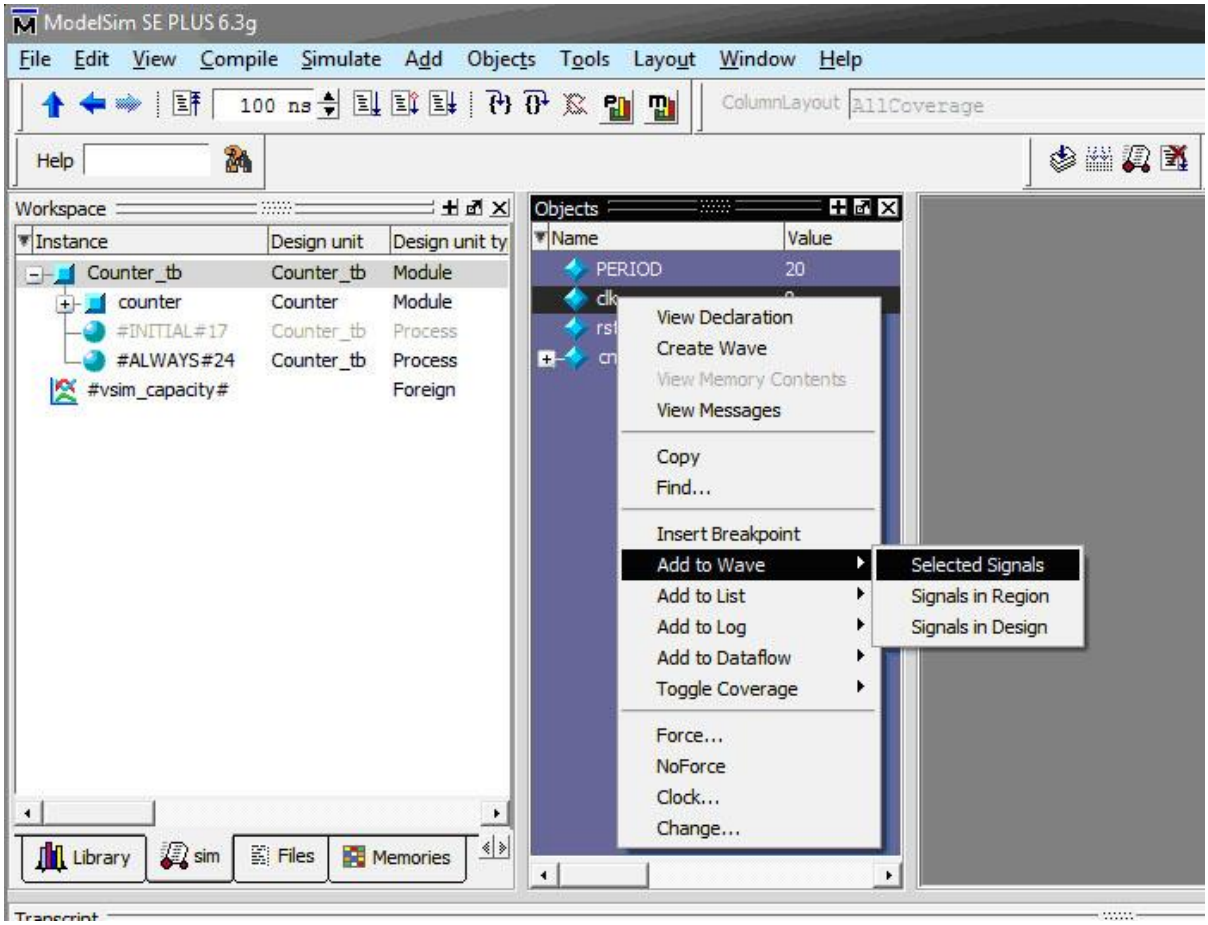


图 26 查看特定信号波形

4.3 保存和导入波形文件(待定)

如果要保存波形窗口当前信号的分配,可以点 File->Save,在出现的对话框中设置保存路径及文件名，保存的格式为.do 文件。

如果是想导出自己创建的波形（在文章最后有详细的解释）可以选择 File->Export Waveform 在出现的对话框中选择 EVCD File 并进行相关设置即可。

如果导入设计的波形选择 File->Import ECVD 即可。

4.4 Dataflow 窗口观察信号波形

在主界面中点 View->Dataflow 可以看到会出现 dataflow 窗口，在 objects 窗口中拖一个信号到该窗口中，你会发现在 dataflow 窗口中出现你刚才选中信号所在的模块，如果双击模块的某一引脚，会出现与该引脚相连的别的模块或者引线，见下图。

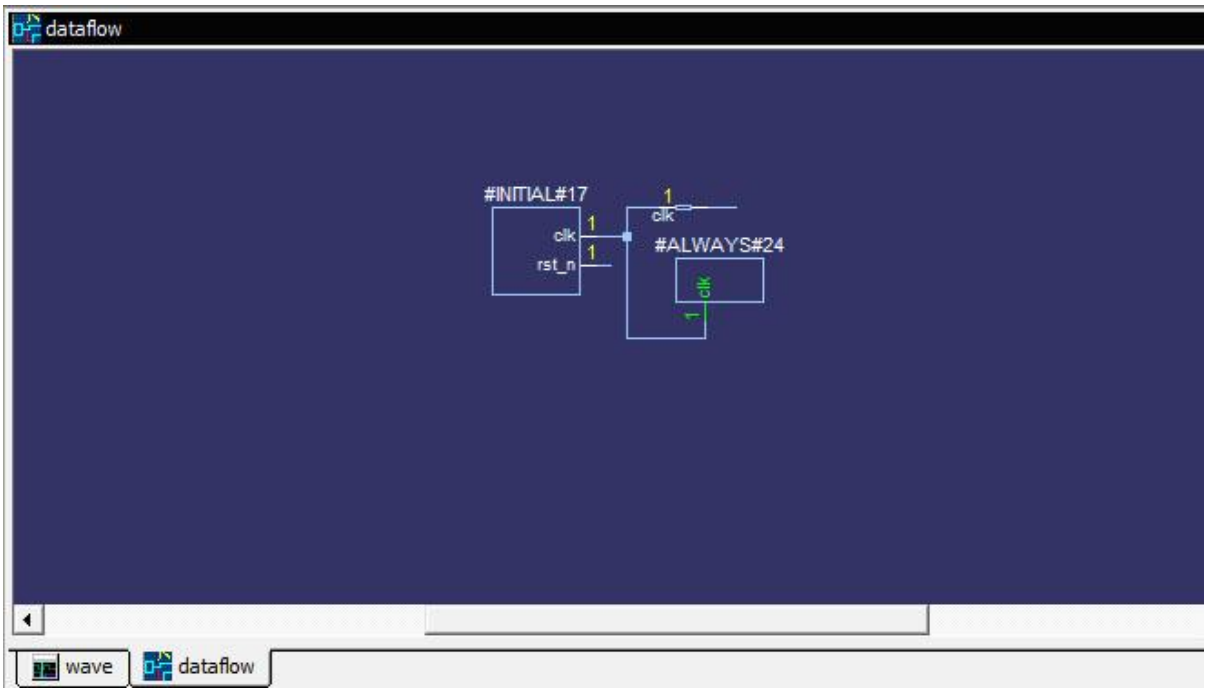


图 27 Dataflow 窗口

在 dataflow 窗口中点 View->Show Wave，会在 dataflow 窗口中出现一个 wave 窗口，双击上面窗口中的某一模块，则在下面的 wave 窗口中出现与该模块相连的所有信号，如果已经执行过仿真，在 wave 窗口中还会出现对应的波形，见下图。

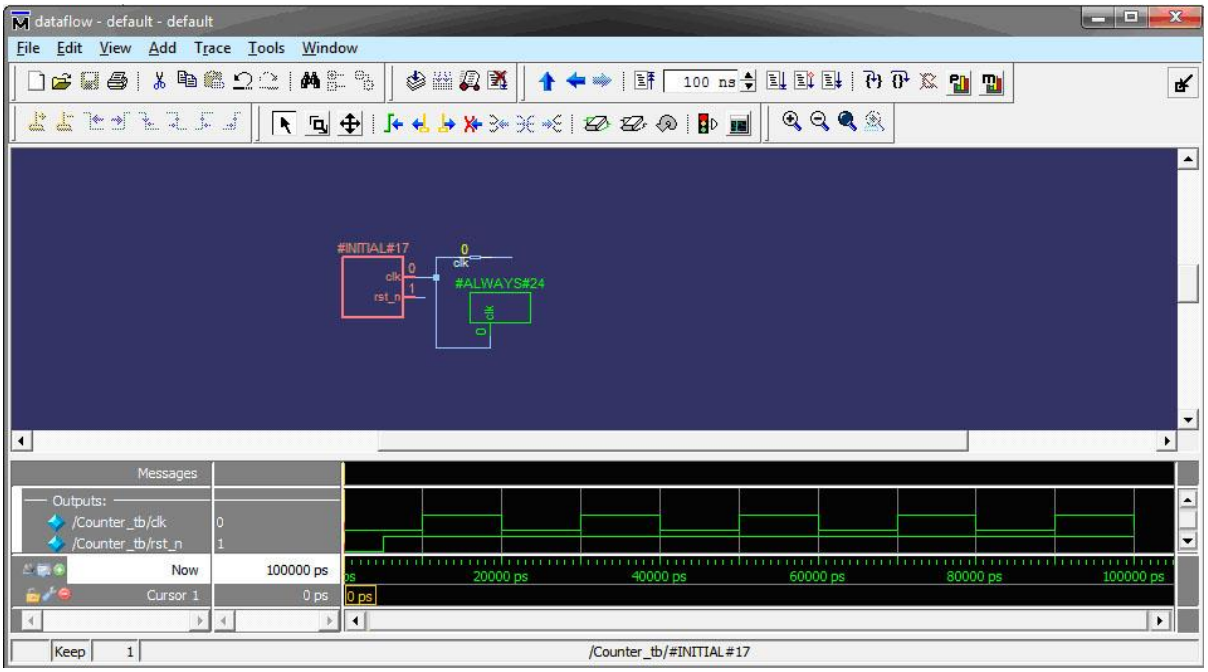


图 28 Dataflow 窗口观察仿真波形

在波形窗口中拖动游标，上面模块的引脚信号的值也会随着游标当前位置的改变而改变。

如果要在 modelsim 中修改原设计文件，在文档页面点击右键，取消 Read Only，即可修改，修改后继续仿真。如果想结束仿真可以点 Simulate->End Simulation,或直接在命令行输入 quit -sim，此时 quartus 也会显示结束所有编译过程。

五、 一些说明

关于在 testbench 里使用`timescale 的问题

`timescale 是编译器指令，用来定义时延精度和时延单位，命令格式为

`timescale time_unit/time_precision

其中 time_unit 定义时延单位，即后面模块中出现的时延数值的单位，time_precision 定义时延精度。例如

`timescale 1ns/100ps 表示时延单位为 1ns，时延精度为 100ps。

如果后面有语句 #5.22 a=1;

此时时延值 5.22ns 应该对应为 5.2ns，因为精度为 0.1ns。

如果设计中多个模块带有自身的`timescale，编译时模拟器总是定义在所有模块的最小时延精度上，并且所有模块中的时延都自动地换算为到最小试验精度上。

附录：

Counter 源代码：

```
`timescale 1ns/100ps
module Counter (
input                CLK,
input                RST_N,
output [3:0] CNT
);
reg [3:0] cnt;
assign CNT = cnt;
always@(posedge CLK, negedge RST_N) begin
if (!RST_N)
        cnt <= #5 4'h0;
else
        cnt <= #0 cnt + 1'b1;
end
endmodule
```

Counter_tb 源代码：

```
`timescale 1ns/100ps
module Counter_tb    ;
    wire    [3:0]    CNT      ;
    reg        RST_N      ;
    reg        CLK      ;
```

```
    Counter
      DUT    (
               .CNT (CNT ) ,
              .RST_N (RST_N ) ,
              .CLK (CLK ) );
//http://wenku.baidu.com/view/cd93f34ecf84b9d528ea7a95.html
  initial begin
  #0 CLK      = 1'b0;
        RST_N = 1'b0;
  #5 RST_N = 1'b1;
  end
  // 50MHz
  always #10 CLK = ~CLK;
```