VHDL 上机手册(基于 Xilinx ISE & ModelSim)

1. ISE 软件的运行及ModelSim 的配置

单击“开始->程序->Xilinx ISE6->Project Navigator”，进入ISE 软件。

为了能够使用ModelSim 进行仿真，选择菜单Edit->Preferences…,选择选项卡Partner Tools，出现界面如图 1 所示。单击按钮  找出ModelSim.exe 文件，单击“确定”。需要注意的是这方面的设置与以前ISE 版本不同，在ISE4.2 中设置是这样的。但在ISE5.1 以及ISE5.2 中是指定ModelSim.exe 文件所在的目录，而ISE6.1 的设置与ISE4.2 的设置相同。单击“确定”关闭该窗口，关闭ISE（这一步非常重要，否则可能不能在ISE 中调用ModelSim 进行仿真），再重新进入ISE 既可用调用ModelSim 对设计进行仿真了。

图 1 第三方工具设置窗口

2 创建一个新工程

Step1. 单击“开始->程序->Xilinx ISE6->Project Navigator”，进入ISE 软件。

Step2. 选择File->New Project…，出现如图 2 所示的窗口。这个窗口与以前版本的差别较大，以前的版本出现的窗口中可以直接选取器件类型、封装、门数、速度等级等信息。而在ISE6.1 中需要单击“下一步”才能看到这些设置信息。在本例中，我们先选择工程存放的路径，然后输入工程名称。系统自动为每一个工程设定一个目录，目录名为工程名。再选择顶层模块类型为HDL。

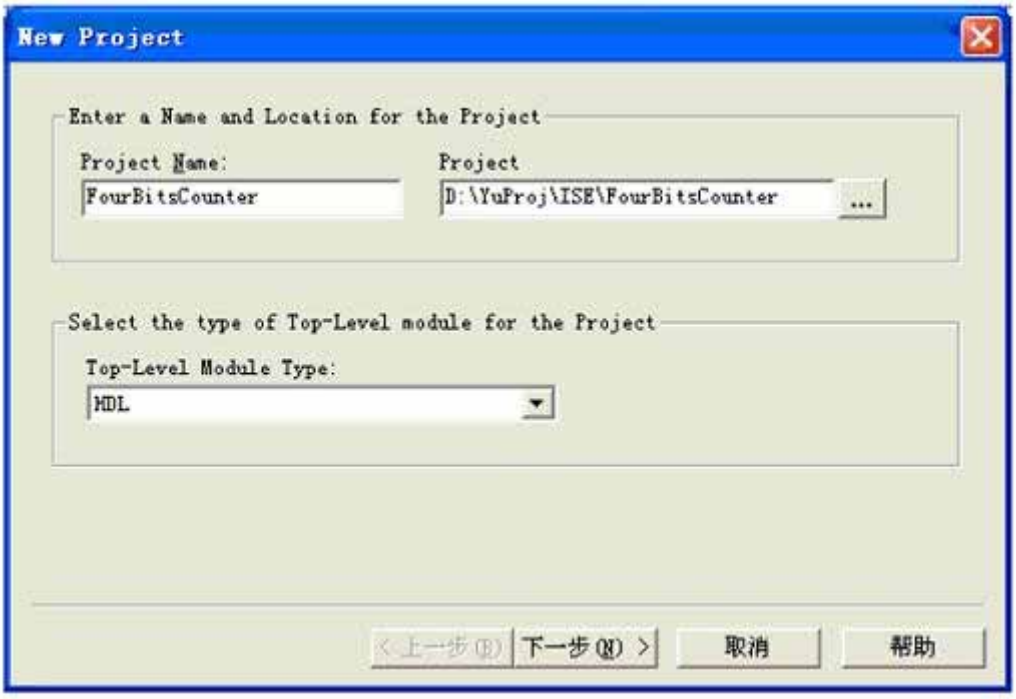

图 2 新工程项对话框

（其他几种类型说明如下：Schematic为原理图输入类型，类似于我们制作PCB 原理图时的情况，可以从库中选取器件，也可以用HDL 语言来生成器件，在后续章节会介绍原理图为设计输入的情况；EDIF为网表输入类型，EDIF 是Electronic Data Interchange Format 的缩写，是一种描述设计网表的标准的工业文件格式，可以由第三方工具生成，在ISE 中可以将其作为一种标准的输入格式。NGC 文件是一种包含了逻辑设计数据和约束的网表，所谓约束是指FPGA 设计中的一些特定的要求，例如，我们分配设计中的信号到具体的管脚时，需要一个文件来指定如何分配，这就是一种约束文件，由于NGC 网表包含了设计和约束，因此一个文件足够描述一个设计了。NGC/NGO 和EDIF 都可以在ISE 外由其他综合工具生成也可由ISE 生成。如果我们需要用ISE 作为设计输入，需要选择Schematic 或HDL作为顶层模块类型；如果已经完成的设计文件为ABEL、Verilog或VHDL，应选择HDL 为顶层模块类型；如果已经完成的设计文件为原理图，这里应该选择Schematic 作为顶层模块类型。）

Step3. 单击“下一步”，出现如图 3 所示的窗口，在该窗口中来选择设计实现时所用的器件。在包含FPGA 的PCB 板子做出来以前，我们选择不同类型的FPGA 进行测试，看看FPGA的资源是否够用，在PCB 板子做出来以后，我们在这里的选择与PCB板上的FPGA 必须一致。否则生成的下载文件无法配置到FPGA 中。此处若选择错了，也没有关系，因为后面可以随时修改这些设置。其中DeviceFamily 表示目标器件的类型；Device 表示目标器件的具体型号；Package 表示器件的封装；SpeedGrade 表示器件的速度等级。这里我们选择器件为Spartan2E，xc2s100，tq144，-6。其中xc2s100 中的 100 表示器件为 10 万门，tq144 表示器件有 144 个管脚。

图 3 设置工程所用的器件参数

Step4. 因为这里我们重新编写VHDL 源代码，而不是使用以前设计好的源代码，故再单击“下一步”，“下一步”，单击“完成”，工程创建完毕。

Step5. 这时的界面如图 4 所示，这里需要关注的是界面左上角出现的小框为我们所有的源文件的管理窗口，在其下面的窗口为我们选择不同的源文件时其所有可能操作的显示窗口；右半部分窗口为我们设计输入代码的窗口；下面的窗口为编译等信息的显示窗口。这里与以前版本不同的地方在于编译输入窗口这里将Warnings 和Errors 可以分开显示。我们可以在输入不同文件后选中不同的文件，看看进程窗口中的变化。这样，我们新建了一个工程，下一步就要在工程中输入一些设计文件来实现我们的设计。

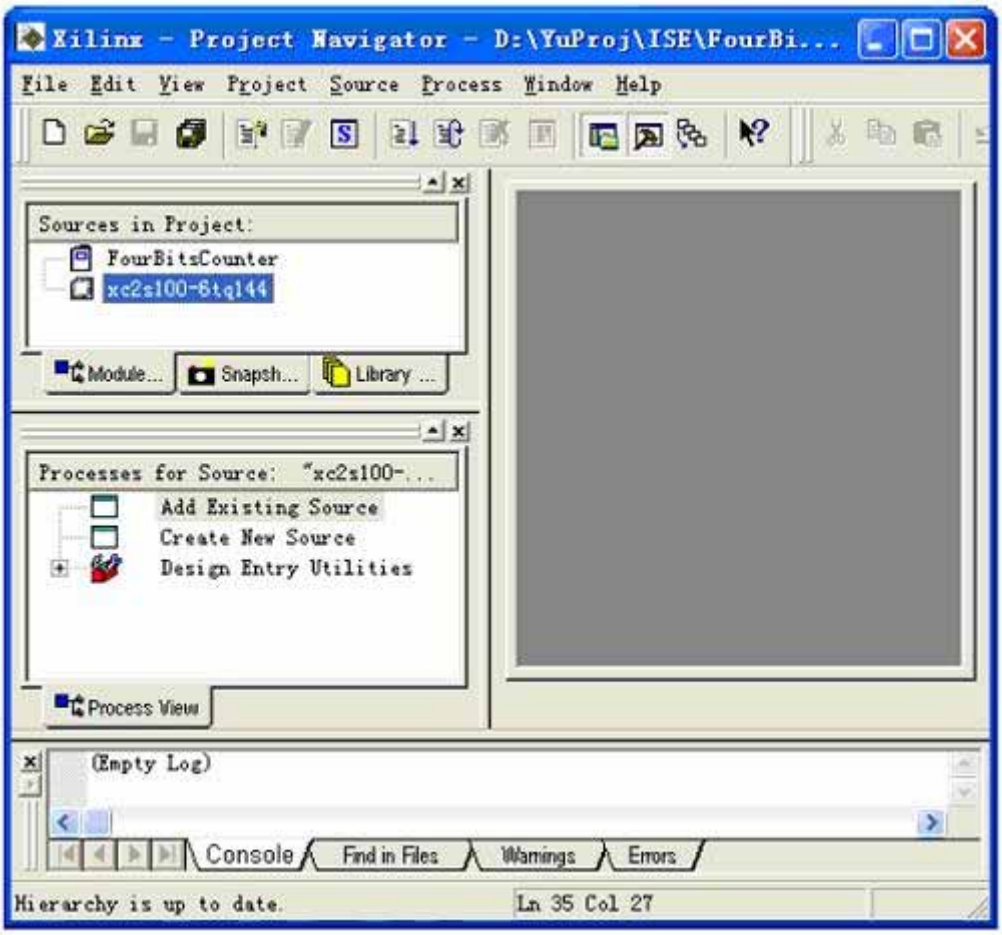

图 4 创建新工程后的ISE 界面

3 创建一个VHDL源文件框架

在本小节我们向刚刚创建的工程中添加设计文件来实现要求的功能。按照以下步骤建立一个计数器的VHDL 文件描述。注意这里仅仅新建一个有框架的文件，下一小节将向该文件中添加具体代码。

在这里我们以一个具有复位（reset）、使能（ce）、置数（load）、计数方向控制（dir）功能的计数器为基础进行设计。其方块图如图 5 所示。其中CLK 为输入计数时钟信号，系统在该信号的驱动下开始工作；RESET 为复位信号，在上升沿处，输入复位为全零；CE 为使能信号，为 1 时计数正常进行，为 0 时停止计数；LOAD 为置数信号，当在时钟上升沿该信号为 1 时，将DIN0～DIN3 分别置给COUT0～COUT3。DIR 为计数方向控制，为 1 时递增计数，为 0 时递减计数。这些功能描述只是我们的设计目标，或称为设计需求，我们在设计一个系统时，第一步就是要明确我们的设计要求。

图 5 计数器方块图

Step1. 选择 Project->New Source；（或在 Sources in Project 窗口中单击鼠标右键选择“New Source…”）出现如图 6 所示的窗口；

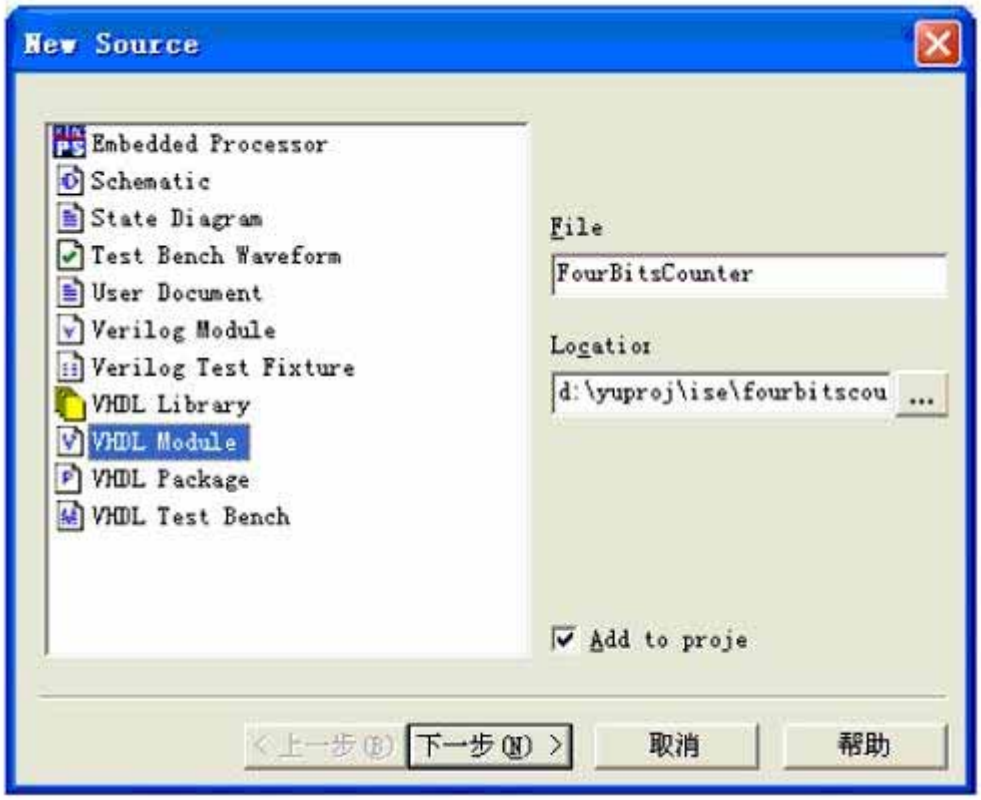

图 6 源程序的类型选择

Step2. 选择 VHDL Module（VHDL 模块）作为新建源文件的类型；

Step3. 在文件名中键入“FourBitsCounter”；

Step4. 单击“下一步”；

Step5. 单击“下一步”；

Step6. 单击“完成”，完成这个新源程序的创建。新源程序文件 FourBitsCounter.vhd 将会显示在 HDL 编辑窗口中，它包括 Library，Use，Entity，Architecture 等语句。

4 利用计数器模板向导生成设计

设计文件建立之后，我们就可以向其中填写代码了。我们可以直接书写HDL 代码，也可以利用ISE 的语言模板（ISE Language Template）工具来辅助我们书写HDL 代码。在这里我们使用语言模板，选择其中的计数器描述来完成本源程序的设计。

Step1. 选择Edit->Language Templates 打开语言模板，或者通过单击按钮 来打开语言模板，如图 7 所示；

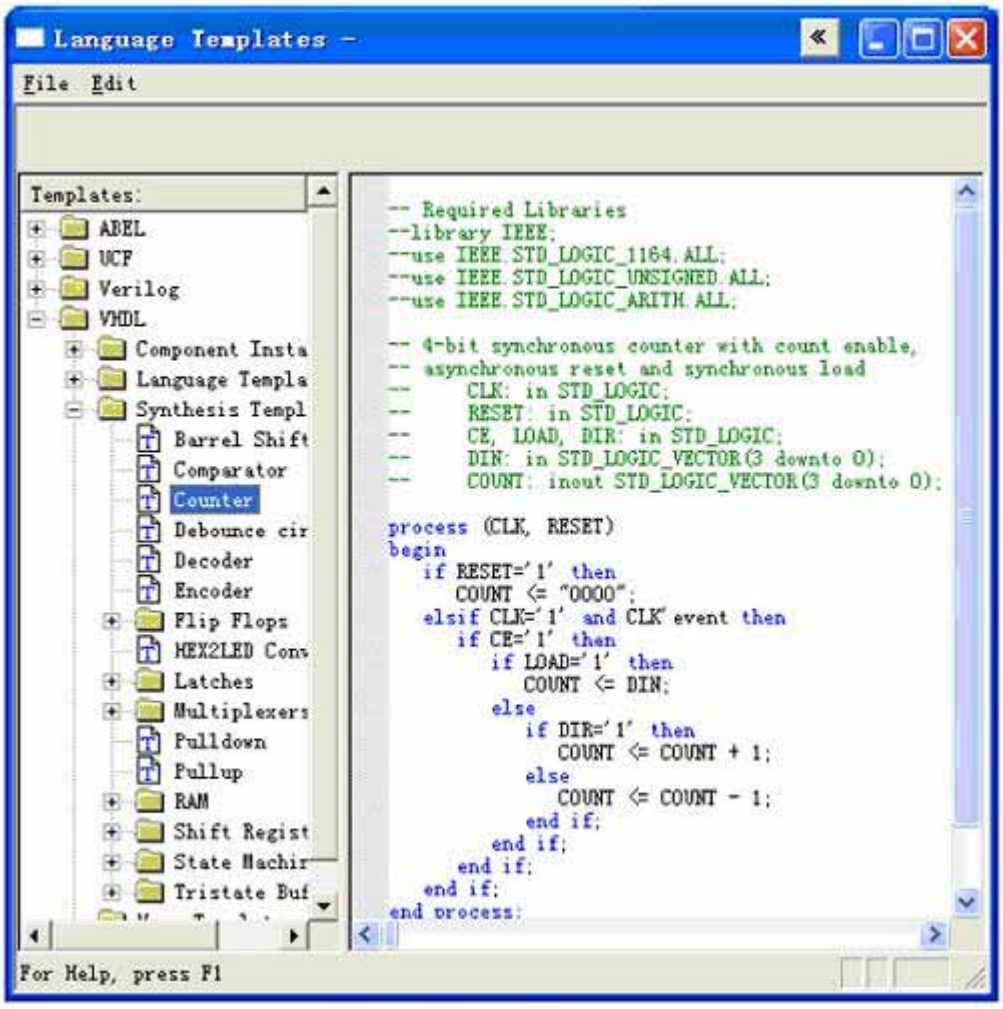

图 7 计数器语言模板

Step2. 在Language Templates 中通过单击“＋”来展开VHDL 下的综合模板（Synthesis Templates）；

Step3. 从VHDL 综合模板中选择计数器模板（Counter Template），并把它粘贴到源程序

counter.vhd 的begin 和end 之间。（或在Counter Template 上单击右键选择“Use incounter.vhd”，建议直接复制过去）；

Step4. 关闭Language Templates 窗口；

Step5. 将带有注释符号“--”的计数器端口定义语句剪切并粘贴到计数器的实体（entity）描述中。这些语句如下所列：

```
-- CLK: in STD_LOGIC;
-- RESET: in STD_LOGIC;
-- CE, LOAD, DIR: in STD_LOGIC;
-- DIN: in STD_LOGIC_VECTOR(3 downto 0);
-- COUNT: inout STD_LOGIC_VECTOR(3 downto 0);
```

Step6. 去掉上述语句中的注释符号；

Step7. 去掉上述最后一个端口定义语句后的分号；此时的程序如图 8 所示。

Step8. 选择File->Save，保存counter.vhd 源程序。

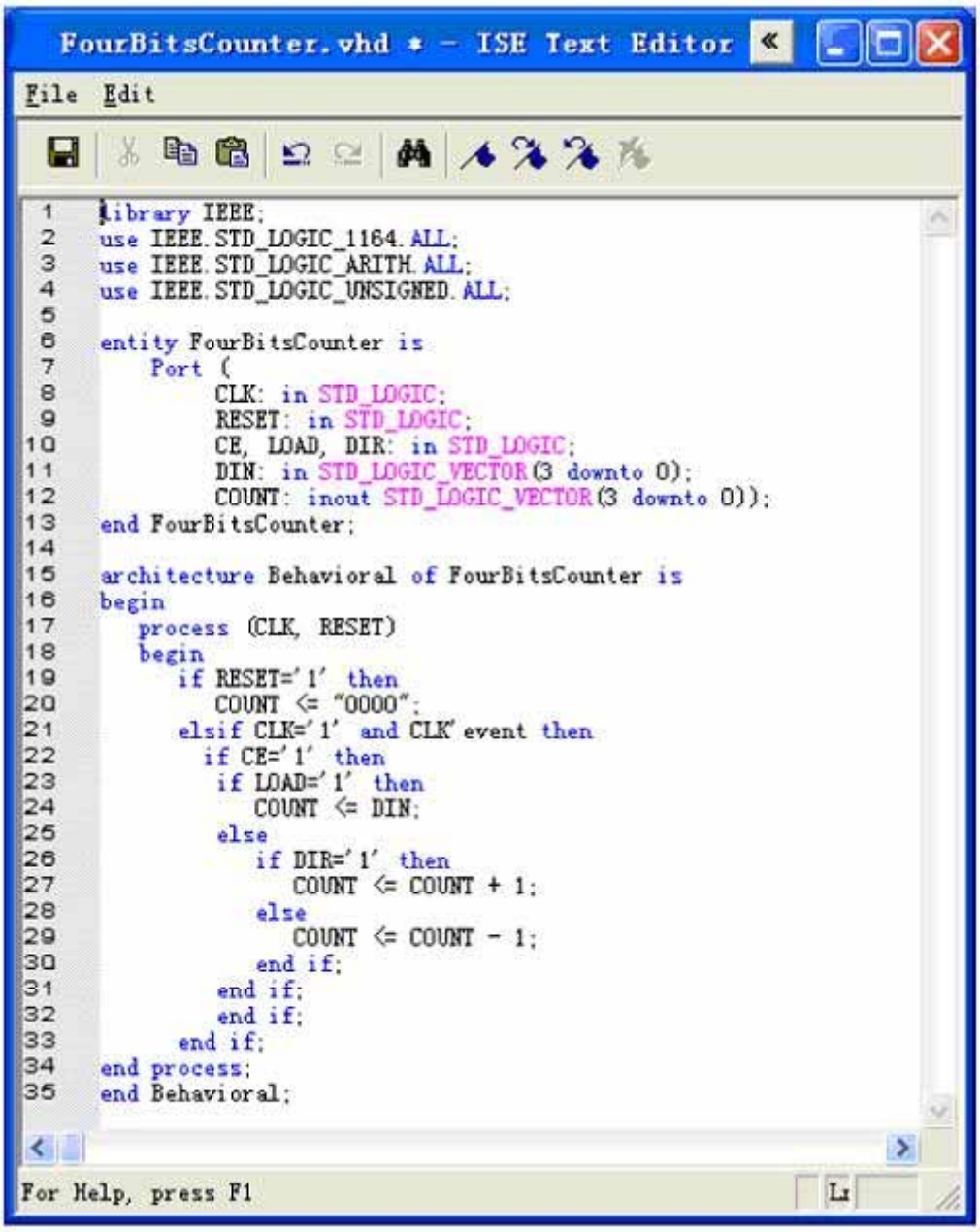

图 8 修改后的计数器描述文件

5 仿真

我们可以通过设置计数器模块的输入来观察仿真输出，以测试我们编写的VHDL 源文件是否满足逻辑功能要求。我们建立的testbench 波形将被用于与仿真软件ModelSim 的连接，用来验证所设计的计数器的功能和延时是否达到要求。

6 创建Testbench波形源文件

在仿真前，首先创建一个Testbench 波形源文件，与以前版本不同的是，该文件不是在HDL Bencher（ISE 集成的一个工具，用于设置输入波形）中打开，而是在ISE 中打开，这也是ISE6.1 不同于以前版本的地方。具体步骤如下：

Step1. 打开上一节所建立的工程；

Step2. 选择Project->New Source…,（或通过在Sources in Project 中单击右键选择“NewSource…”），出现如图 9 所示的窗口；

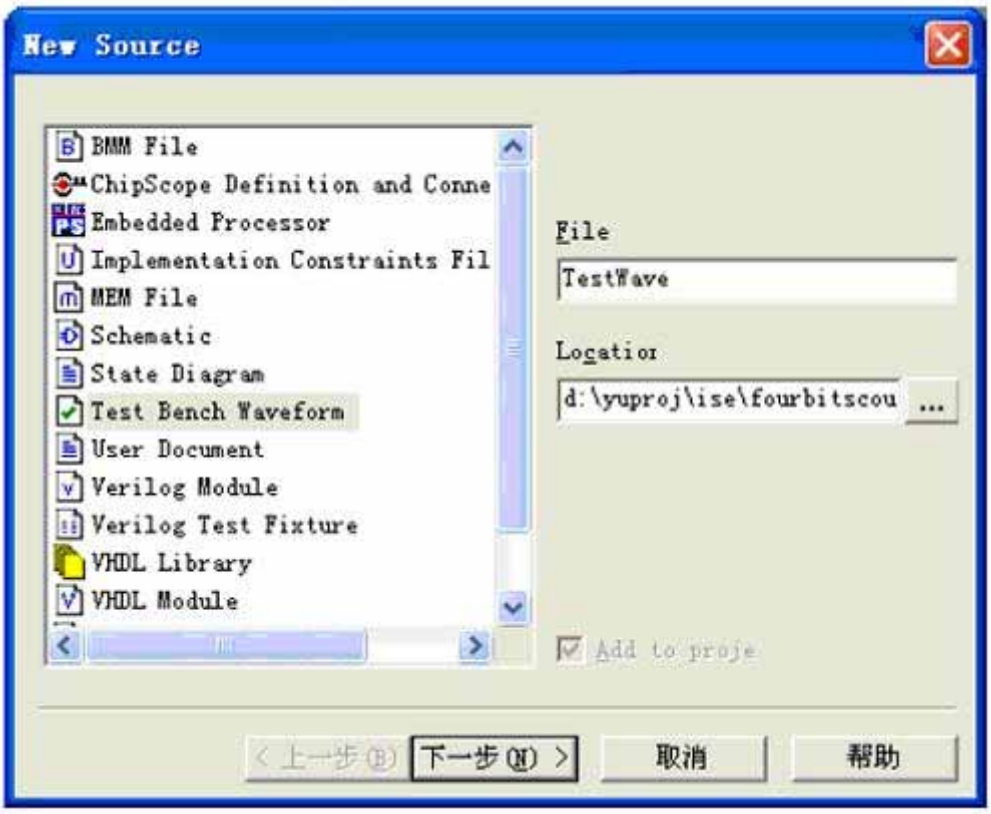

图 9 创建波形源文件

Step3. 选择文件类型为 Test Bench Waveform；

Step4. 键入文件名“TestWave”，如图 9 中所示；

Step5. 单击“下一步”，在本步骤中可以将波形文件与 VHDL 文件进行关联。

Step6. 单击“下一步”；

Step7. 单击“完成”；

Step8. 此时，HDL Bencher 程序自动启动，如图 10 所示，我们可以选择哪一个信号是时钟信号并可以输入所需的时序需求；在这里我们采用系统的默认值，单击“OK”按钮；

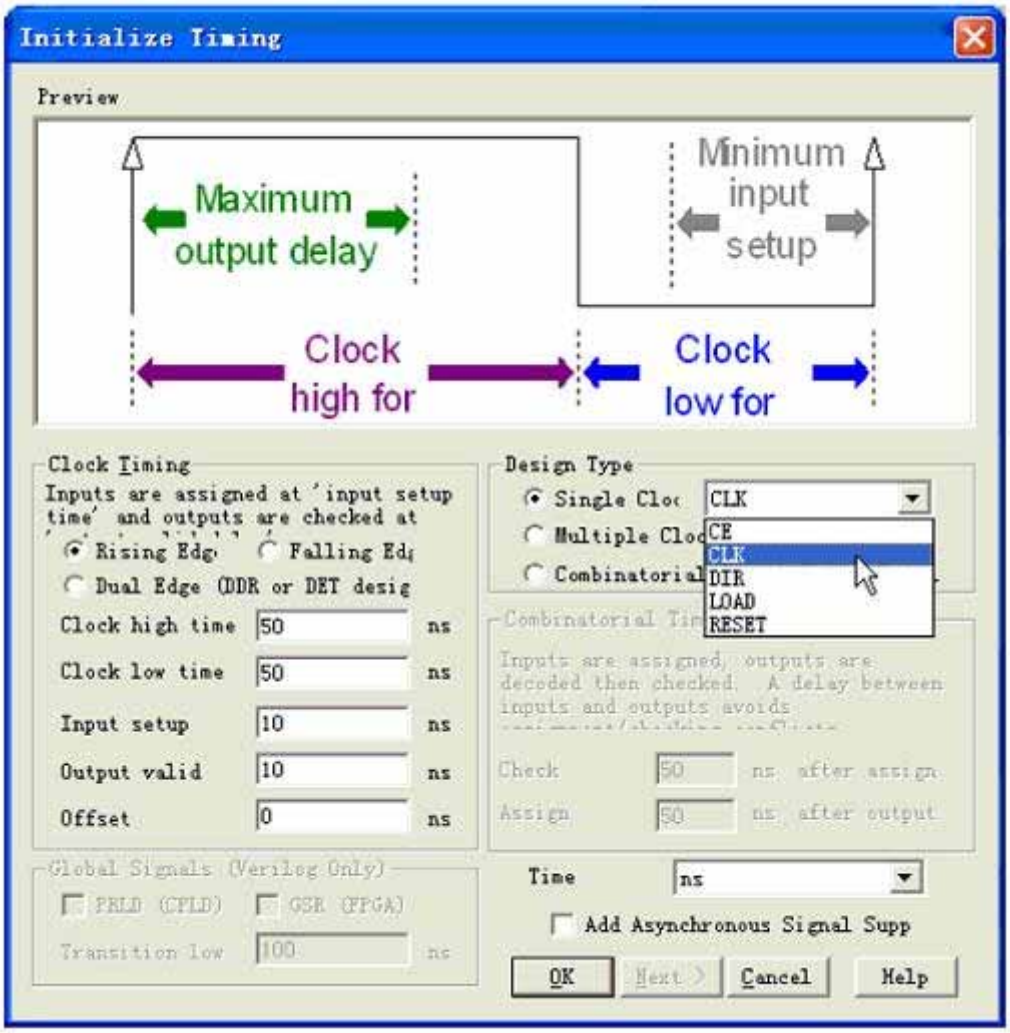

图 10 仿真时间参数的设置

Step9. 这时出现了如图 11 所示的波形；

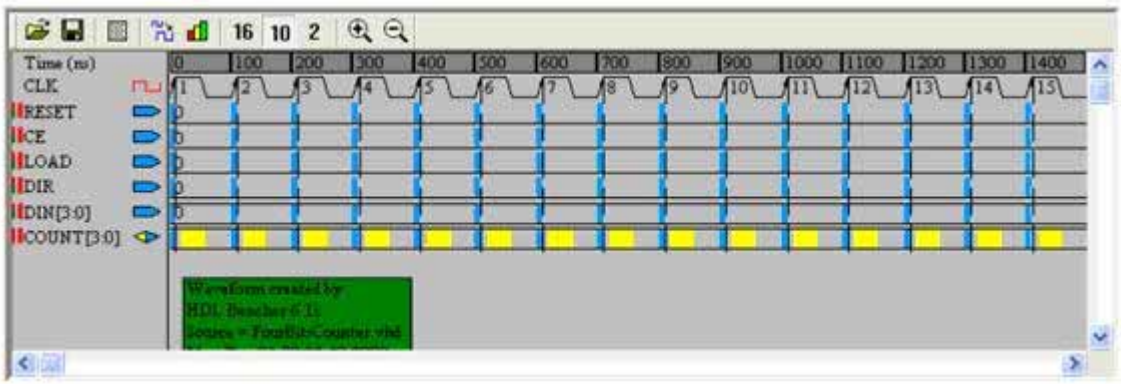

图 11 新建的波形文件

7 设置输入仿真波形

我们可以打开刚刚建立的波形文件，来初始化输入波形，步骤如下：

Step1. 单击波形图中的蓝色方块来设置波形电平的高低，并将仿真时间线（图中的垂直的蓝色线）拉到第 10 个时钟周期处，设置后的波形如图 12 所示；

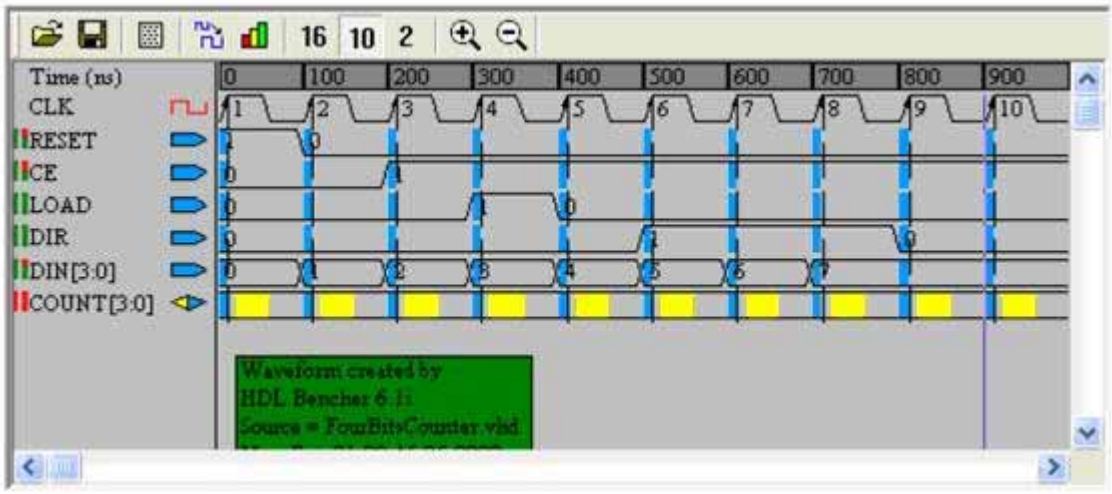

图 12 HDL Bencher 中输入波形的设置

Step2. 单击图 12 中工具栏上的图标 ，将波形文件保存。

Step3. 查看代码覆盖率统计，单击图 12 中工具栏上的图标 ，显示出代码覆盖率统计，统计结果如图 13 所示。因为还没有输出，所有输出的统计均为零。代码覆盖率是一种测试术语，它可以表示运行完当前仿真时，所运行的代码占所有代码的比例。其中的Assign 为赋值情况代码占所有代码的比例，Toggle 为上升下降沿代码占所有代码的比例。因此，代码覆盖率越高越好。

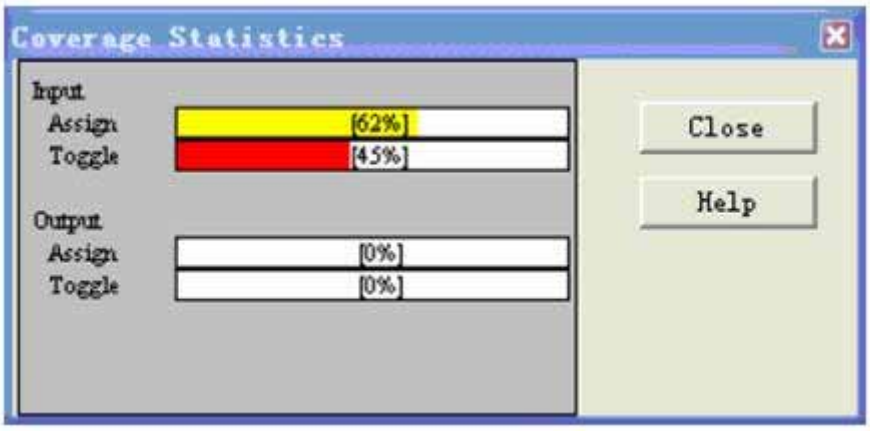

图 13 代码覆盖率统计结果

8 调用ModelSim 进行仿真简介

其实在上一节中产生预定输出时，已经使用了ModelSim，只是我们在界面上看不出来而已。这一小节我们在ISE 中调用ModelSim 进行仿真，这里讨论的仿真仍然是基于波形文件的，因此不涉及ModelSim 中过多的知识。在ModelSim 中可以进行的仿真有Simulate Behavioral Model（仿真行为模型）、Simulate Pose-Translate VHDL Model(转换后仿真)、Simulate Post-Map VHDL Model(映射后仿真)以及Simulate Post-Place&RouteVHDL Model(布局布线后仿真)。其实， 转换（ Translate ）、映射（Map） 以及布局布线（Place&Route）是FPGA 及CPLD 设计实现时的不同阶段。

要实现一个设计，首先要进行编译或转换（Translate），转换是将HDL 描述转换为RTL 描述，转换后仿真可以认为是RTL 级仿真，而且仅仅是逻辑仿真，在仿真中不包含任何的器件、时延等信息，仅仅用于验证设计转换为RTL 级描述后是否满足功能要求；下面就是综合，在该阶段，设计文件按照约束文件与Xilinx 的原型库联系起来，映射（Map）则是将当前设计映射到具体器件的特定逻辑单元以及特定的工艺，所谓特定的逻辑单元是FPGA 中的基本的逻辑块，所谓工艺是FPGA 的制作工艺，因此，映射后仿真是将设计实现到具体器件具体逻辑单元具体工艺后进行的逻辑仿真，类似于我们制作PCB 时画完原理图后进行的仿真，此时的仿真已经考虑到了器件延时，由于没有布线，因此，连线的长度等信息就不能知道了，故此时的仿真是仅仅考虑到逻辑单元延时的仿真，而没有考虑到连线的电容、电阻、长度等信息。在亚微米（0.35 微米）以上的工艺中，连线的延时可以不太重视，而在深亚微米工艺中，连线的影响就不可小看了，为了保证深亚微米设计的成功，需要在布局布线前对设计进行时序仿真，这时候修改错误对设计进度的影响要小很多。最后是布局布线后仿真，为进行这个仿真，首先要进行布局布线，类似于我们对PCB 的布线，之后要进行参数提取，提取出互连线的长度、电阻、电容等信息，然后就可以根据这些信息进行仿真了，这时候的仿真中包括了器件本身的延时和互连线的延时等等部分，这种仿真也最近似实际情况。

也许读者会疑惑，有了映射后仿真为什么还需要转换后仿真呢？这是因为许多EDA 工具只能认识RTL 描述，而人们习惯使用高级的HDL描述，这就需要转换，如果转换后的RTL 描述是错误的，那么后续的过程还有什么意义呢，故还是需要进行转换后仿真的，尽管一般转换阶段不会发生什么错误。

9 调用ModelSim 进行行为仿真（Simulate Behavioral Model）

如上所述，行为仿真验证所设计的模块的功能。未涉及到设计实现中的时延等问题。具体步骤如下：

Step1. 在如图 14 所示的Sources in Project 窗口中，选中TestWave 文件；

图 14 Sources in Project 窗口

Step2. 在如图 7-15 所示的Processes for Source :"TestWave"窗口中，通过单击"＋"展开它；

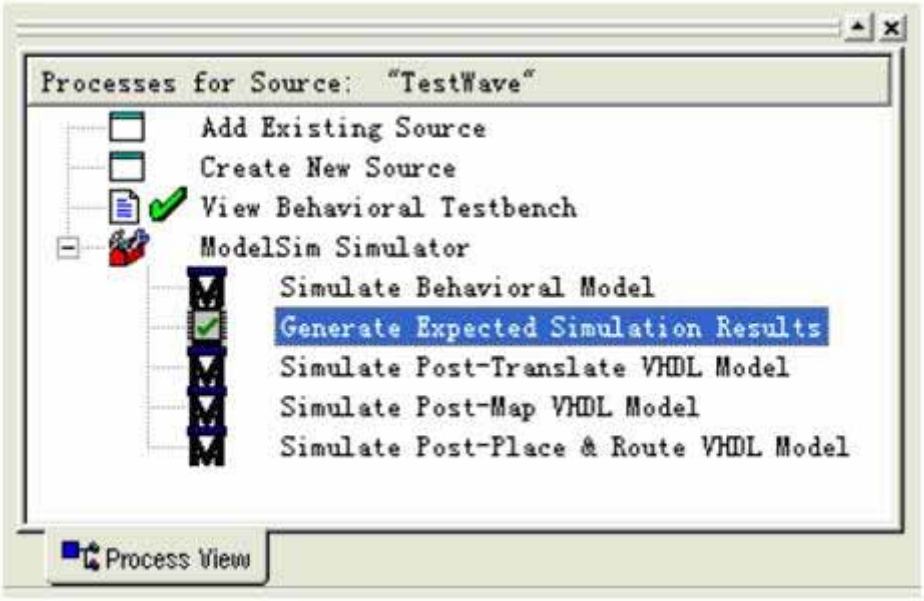

图 15 Processes for Current Source 窗口

Step3. 双击Simulate Behavioral VHDL Model，ModelSim 会自动运行，仿真结果出现在ModelSim 的波形窗口（Wave Windows）中，如图 16 所示，从双击命令到所有窗口的出现都是在ISE 自动创建的仿真宏文件（.fdo）的控制下来完成了，用户可以在工程存放的路径下看到该文件counter_tbw.fdo；用记事本打开该文件可以看到其内容。

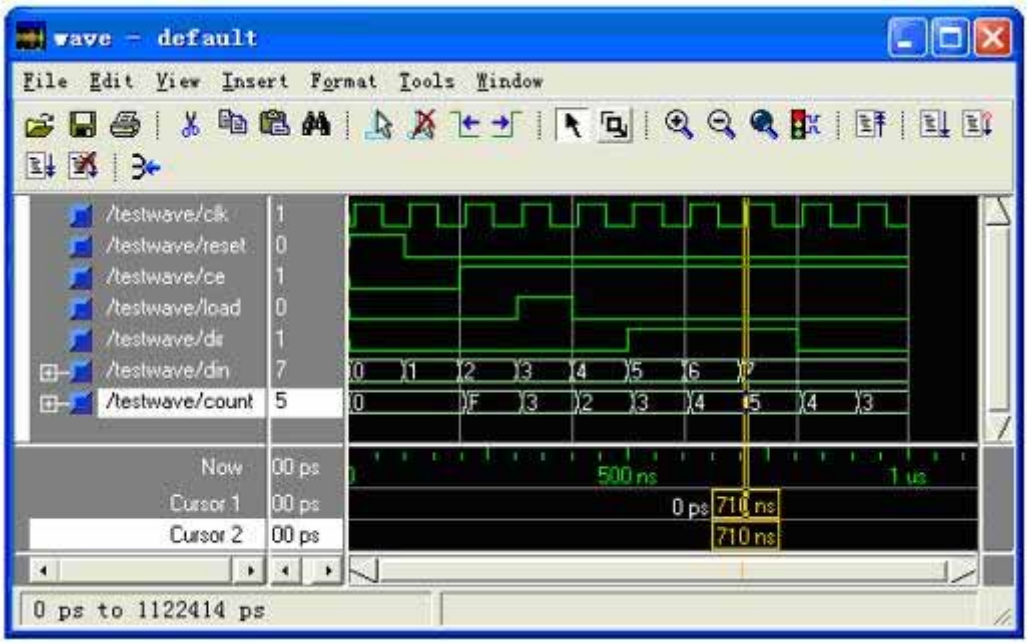

图 16　ModelSim 行为仿真结果

Step4. 打开波形窗口，单击按钮  ，可以将所有波形在屏幕中显示，仿真结果如图 16 所示。可以看到，时钟上升沿和计数值改变的时刻之间相差为零（图中两根竖线之间的间距为零）。

Step5. 关闭ModelSim 主窗口，确认退出ModelSim。

10　转换后仿真（Simulate Pose-Translate VHDL Model）

如上所述，转换后仿真是将设计转换为RTL 级描述后进行的仿真。在其中不包含实现器件的信息。具体仿真步骤与行为仿真相同。只是在第三步，双击Simulate Pose-Translate VHDLModel 就可以了。仿真波形图如图 17 所示。可以看到，时钟上升沿和计数值改变的时刻之间相差为零（图中两根竖线之间的间距为零）。

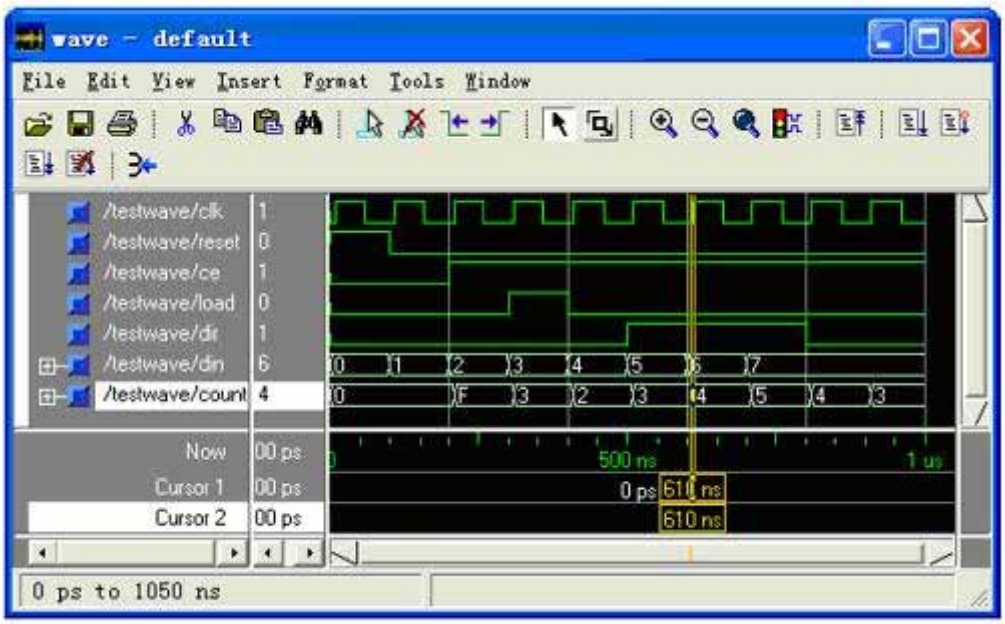

图 17　ModelSim 转换后仿真结果

11 调用ModelSim 进行映射后仿真（Simulate Post-Map VHDL Model）

映射后仿真是设计映射到具体工艺和器件后进行的仿真，在其中包含了器件本身的延时信息。具体仿真步骤与行为仿真相同。只是在第三步，双击Simulate Post-Map VHDL Model就可以了。仿真波形图如图 18 所示。可以看到，时钟上升沿和计数值改变的时刻之间相差6794ps（图中两根竖线之间的间距），说明了器件的延时为 6794ps。

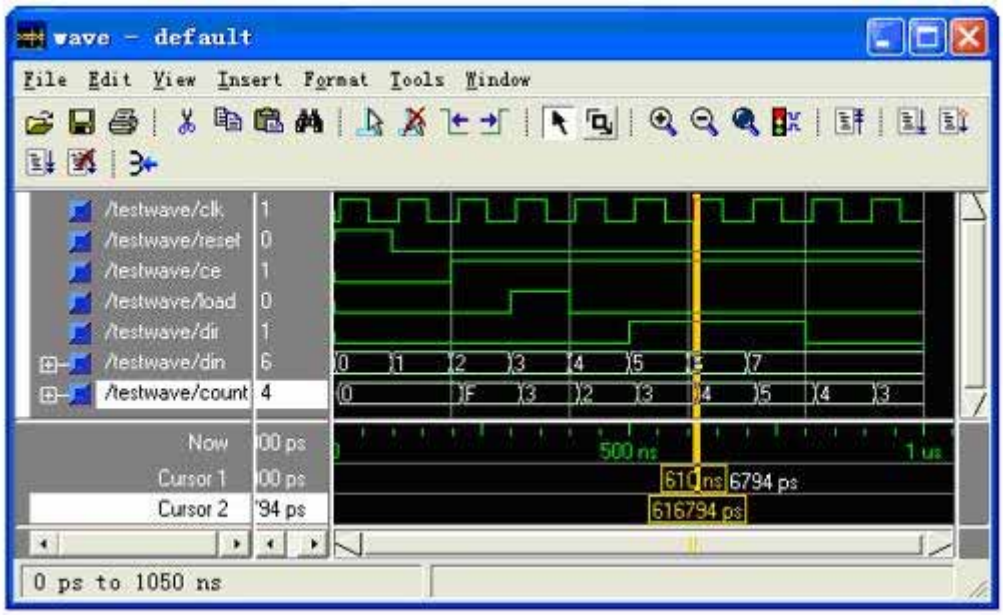

图 18　ModelSim 映射后仿真结果

12 布局布线后的仿真（ Simulate Post-Place&Route VHDL Model）

布局布线后仿真利用了从布局布线中提取出的一些信息，其中包括了目标器件及互连线的时延、电阻、电容等信息。具体仿真步骤与行为仿真相同。只是在第三步，双击Simulate

Post-Place&Route VHDL Model 就可以了。仿真波形图如图 19 所示。可以看到，时钟上升沿和计数值改变的时刻之间相差 8296ps（图中两根竖线之间的间距），说明了器件的延时加上互连线延时为 6794ps。

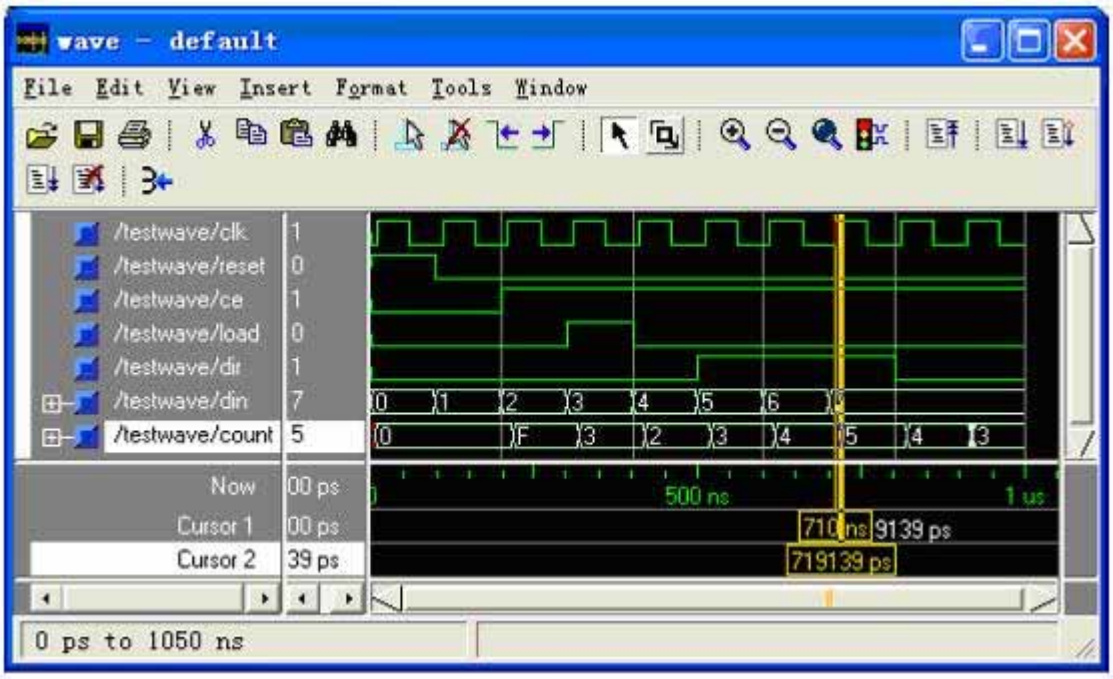

图 19　ModelSim 布局布线后仿真结果