第5章

基于仿真的第一个工程实例

本章导读：

本章手把手引导读者建立第一个属于自己的 FPGA 工程，包括工程新建，基本器件和工具配置，Verilog 源码创建和编辑，Verilog 语法检查以及进行 Modelsim 的仿真验证。

5.1 新建工程

本节将一起动手使用 ISE 创建一个 FPGA 工程。

首先，在硬盘中创建一个名为 project 的文件夹，注意这个文件夹所在的路径名称中不要有任何的中文和符号（下画线除外），即以数字和字母为主，例如，笔者的路径为“D:\myfpga\DK_SF_SP6\lesson”。

打开 ISE，进入主界面后，如图 5.1 所示，执行 File→New Project…命令。

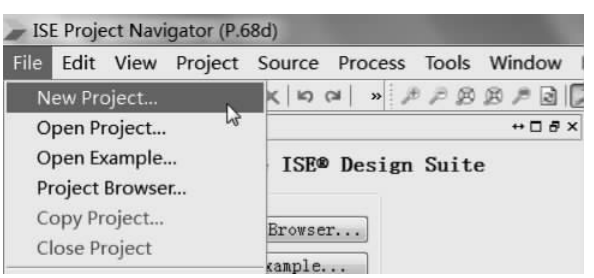

图 5.1 New Project 菜单

在弹出的 New Project Wizard 中，分别做如下的设置。

（1）Name 一栏输入工程名称，这里建议输入 sp6 作为工程名称。

（2）Location 一栏输入工程所在的文件夹路径，“D:\myfpga\DK_SF_SP6\lesson\project”即前面创建的专用于存放 FPGA 实例工程的文件夹。其中最后一级的名称在输入 Name 后自动产生，即为“D:\myfpga\DK_SF_SP6\lesson\project\sp6”，建议最后一级自动创建的文件夹由 sp6 修改为“sp6ex1”。

(3) Working Directory 一栏的路径和 Location 一栏一致即可。

(4) Top-level source type 选择 HDL,即以 HDL 方式作为设计工程的顶层模块。

设置完成如图 5.2 所示。

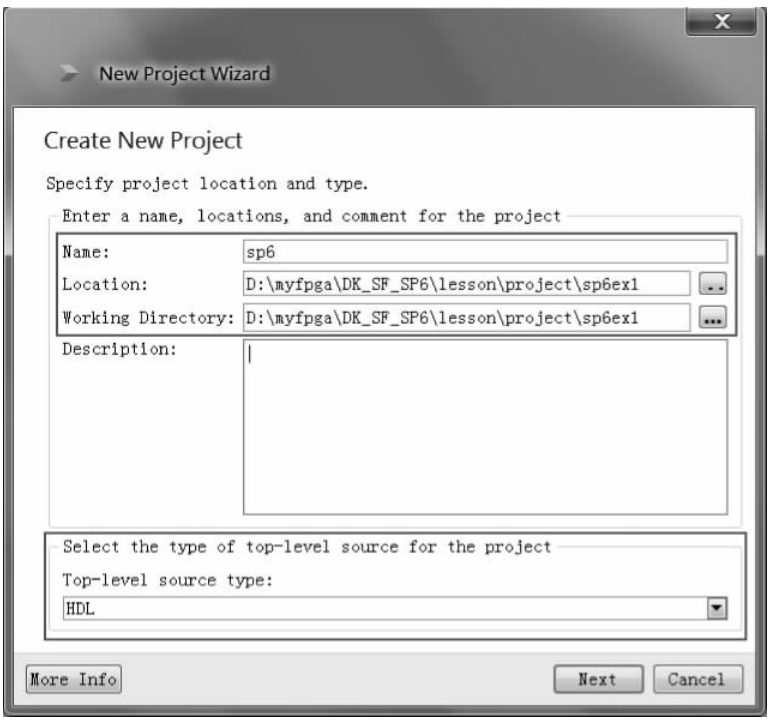

图 5.2 工程名和路径设置

完成以上设计后,单击 Next 按钮。

在下一个弹出的 Project Settings 设置界面中,需要做如图 5.3 所示的设置。这里分别设置所使用 FPGA 的系列(Family)、器件型号(Device)、封装(Package)、速度等级(Speed)、综合工具(Synthesis Tool)、仿真工具(Simulator)、优选语言(Preferred Language)等。根据所使用的 SF-SP6 开发套件,务必按照图示进行设定。

完成设定后,单击 Next 按钮。

接着弹出如图 5.4 所示的 Project Summary 界面,将前面设定的所有工程信息罗列出来,便于核对,单击 Finish 按钮完成工程创建。

此时可以看到如图 5.5 所示,在 ISE 的 Design→Implementation 窗口下,出现了新创建工程的工程名 sp6 和器件名称 xc6slx9-2tqg144。

如图 5.6 所示,在文件夹 project 下,自动产生了一个包含三个文件(文件夹)的新文件夹 sp6ex1。

至此,工程创建完毕。不过这只是一个开头,还要在这个工程中创建 Verilog 源代码文件。

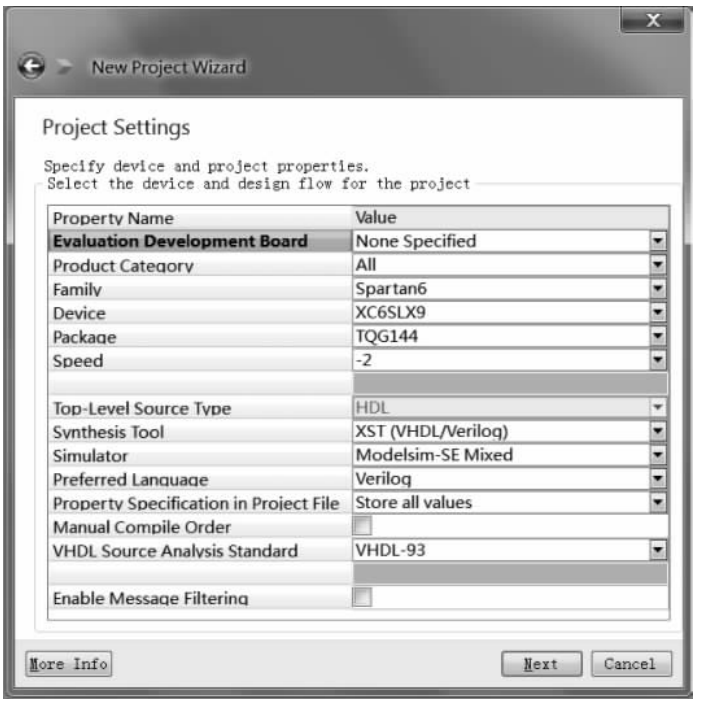

图 5.3　工程基本设置

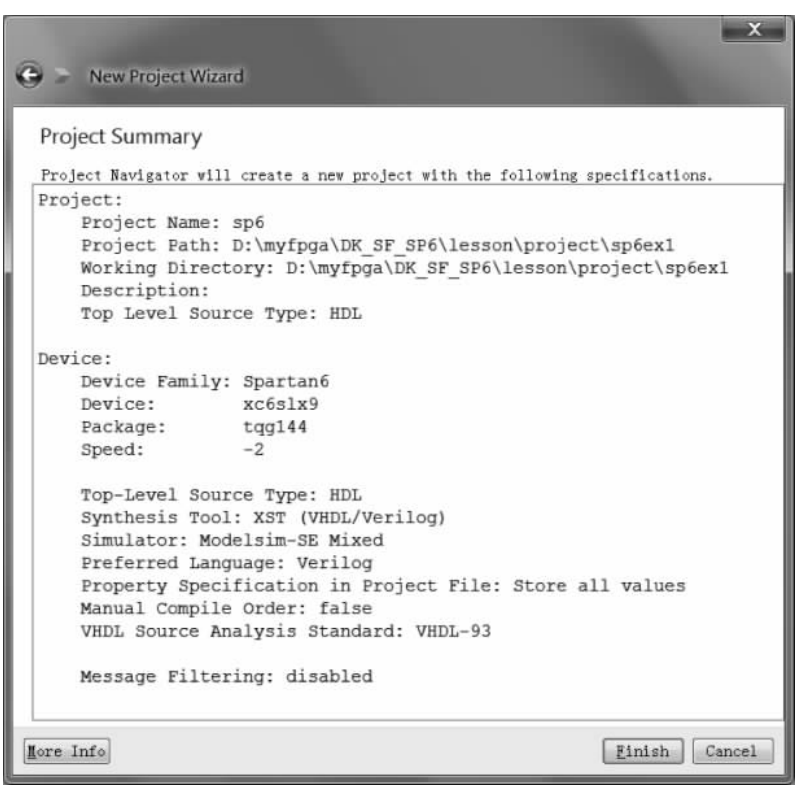

图 5.4　新建工程 Summary

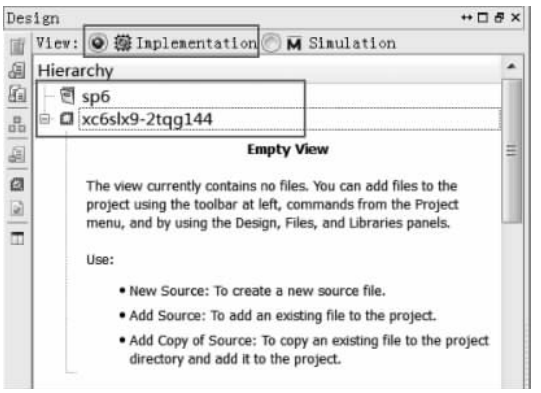

图 5.5　新建好的工程 Design 视图

图 5.6　新建工程文件夹

5.2　Verilog 源码文件创建与编辑

5.2.1　Verilog 源码文件创建

在上一个实例的基础上，如图 5.7 所示，先到工程所在文件夹中，创建一个名为 source_code 的空文件夹，将来新创建的 Verilog 源文件，都把它们存放在这里，这样做的好处是便于查看和管理。

图 5.7　新建 source_code 文件夹

接着回到 ISE 中打开 sp6ex1 工程。如图 5.8 所示，在 Design→Implementation→Hierarchy 中的任意位置右击，在弹出菜单中选择 New Source...命令。

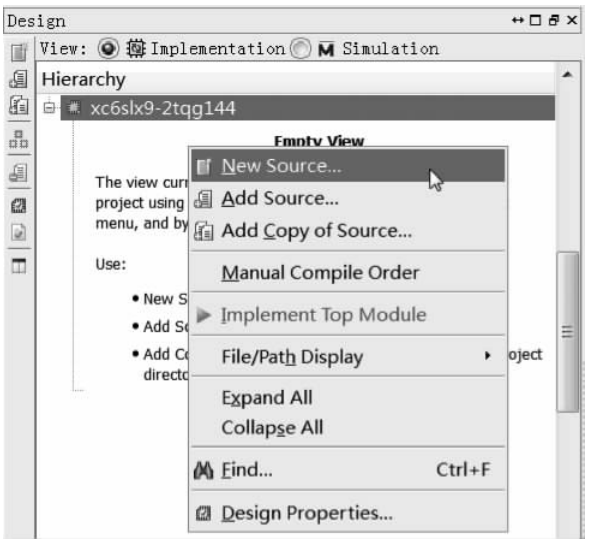

图 5.8　New Source 菜单

在 New Source Wizard 中，做如图 5.9 所示的设置。

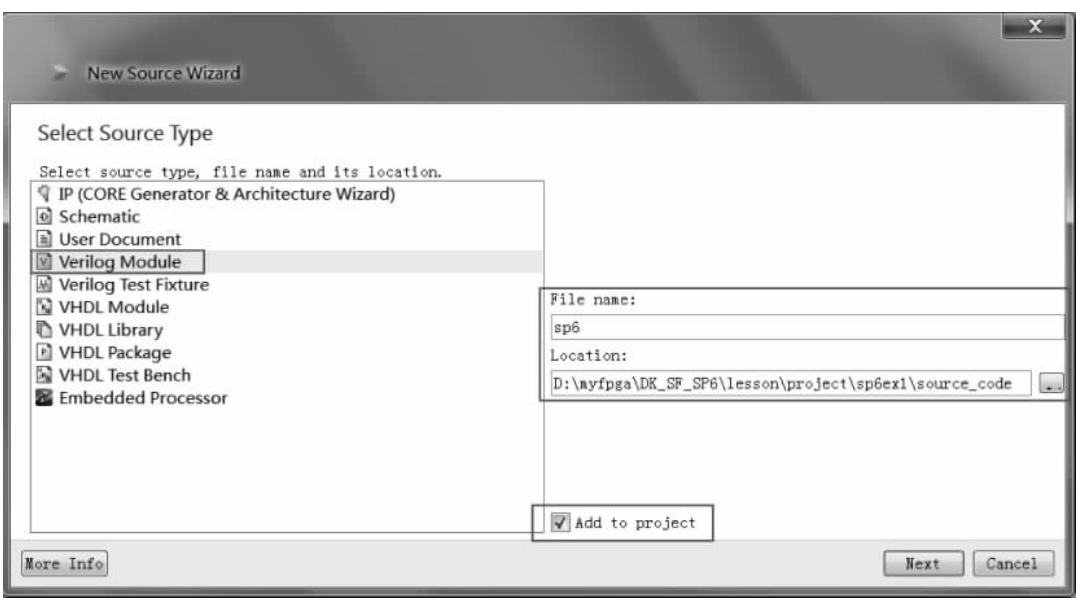

图 5.9　设置 Verilog 源文件名和路径

(1) Select Source Type 中选择新建文件类型为 Verilog Module。

(2) File name 即文件名，因为这个文件要作为工程的顶层文件，所以最好是和工程名称一致，即 sp6，如果不是工程顶层源文件，名称随意起，当然不要有中文和符号(还是要以字母、数字、下画线为主)。

(3) Location 下面输入这个新建文件所存放的路径,将其定位到新创建的 source_code 文件夹下,整个路径为“D:\myfpga\DK_SF_SP6\lesson\project\sp6ex1\source_code”。

(4) 勾选 Add to project。

完成以上设置后,单击 Next 按钮继续。

在接着弹出的 Define Module 窗口中,默认出现了 Module name 为 sp6,即上一个页面设定的名称,不需要修改。如图 5.10 所示,分别设定三个信号 ext_clk_25m(input)、ext_rst_n(input)、clk_12m5(output)作为设计文件的对外接口。这个地方也可以不用设置,创建好 Verilog 源文件后直接在文本中编辑也行。完成设置后,单击 Next 按钮。

如图 5.11 所示,最后弹出 Summary 页面,单击 Finish 按钮完成创建。

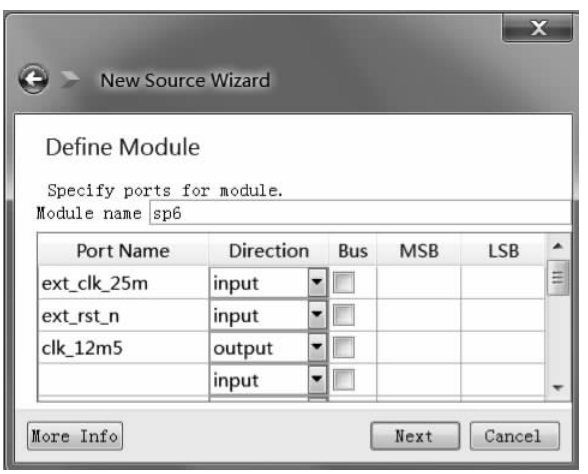

图 5.10 定义模块端口

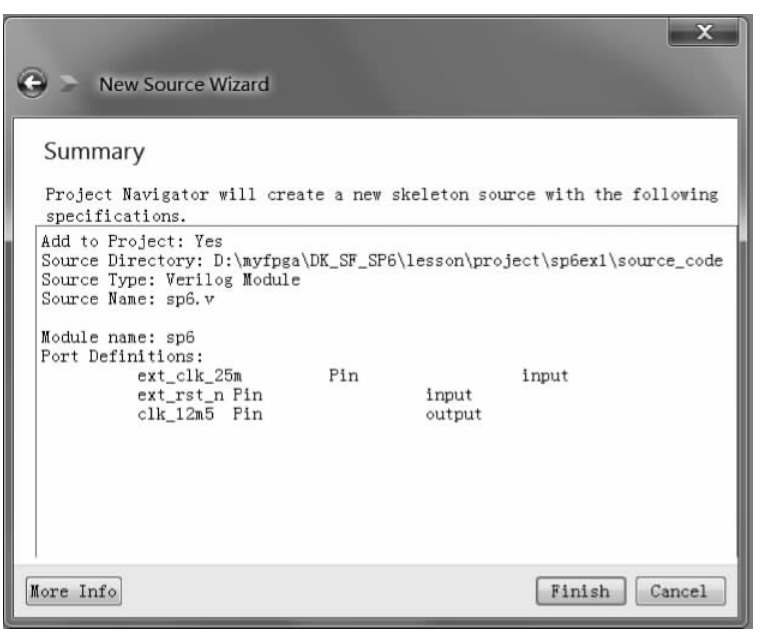

图 5.11 新建工程 Summary 页面

此时，如图 5.12 所示，弹出了 Notepad++打开的 sp6.v 文件，即刚刚创建好的 Verilog 工程源码文件。

sp6.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:   14:53:31 05/06/2015
7  // Design Name:
8  // Module Name:   sp6
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21 module sp6(
22     input ext_clk_25m,
23     input ext_rst_n,
24     output clk_12m5
25     );
26
27
28 endmodule
29
```

图 5.12　sp6.v 工程源码

如图 5.13 所示，在 source_code 文件夹下也出现了 sp6.v 文件。

图 5.13　文件夹下的 sp6.v 文件

5.2.2　Verilog 源码文件编辑

在 sp6.v 源文件中，输入一段对时钟二分频的代码，编辑好后的 sp6.v 源文件代码如下。

```
//对外部输入时钟做二分频
module sp6(
            input ext_clk_25m,     //外部输入 25MHz 时钟信号
            input ext_rst_n,       //外部输入复位信号,低电平有效
            output reg clk_12m5    //二分频时钟信号
        );
always @(posedge ext_clk_25m or negedge ext_rst_n)
    if(!ext_rst_n) clk_12m5 <= 1'b0;
    else clk_12m5 <= ~clk_12m5;
endmodule
```

这段代码的功能如下。

(1) 输入复位信号 ext_rst_n 为低电平时,即复位状态。无论输入时钟 ext_clk_25m 是否运行,输出信号 clk_12m5 始终保持低电平。

(2) 输入复位信号 ext_rst_n 为高电平时,即退出复位。每个 ext_clk_25m 时钟信号的上升沿,信号 clk_12m5 的输出值翻转。

如图 5.14 所示,这便是前面的代码将要实现的功能。

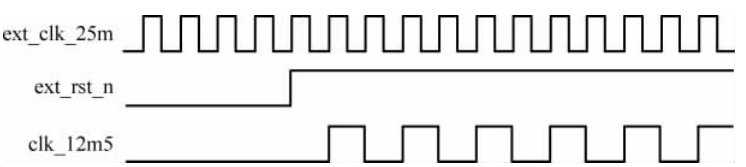

图 5.14 源码实现功能波形

5.3 Verilog 语法检查

至此,已经创建并且编辑好 Verilog 源码文件。现在要到 ISE 中对这个源码进行语法检查。如图 5.15 所示,在 Hierarchy 下,需要先选中 sp6.v 这个源代码文件。接着在 Processes 中单击 Synthesize-XST 前面的+号,展开综合选项。

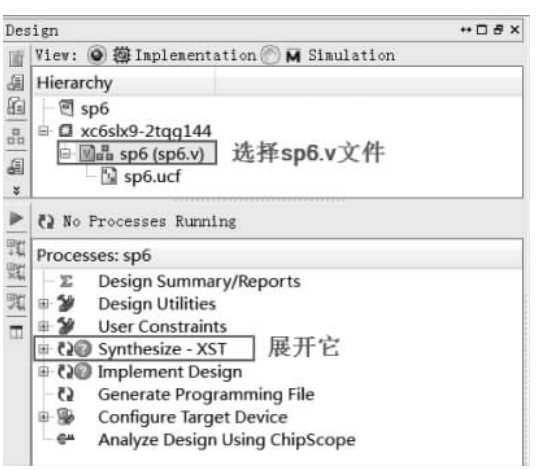

图 5.15 Design 视窗

如图 5.16 所示，展开 Synthesize-XST 后，看到了 4 个选项，双击 Check Syntax 这个项目，即“语法检查”功能。

数秒后，如图 5.17 所示，可以看到 Check Syntax 选项的前面出现了绿色的勾号，说明语法检查完成，并且通过。

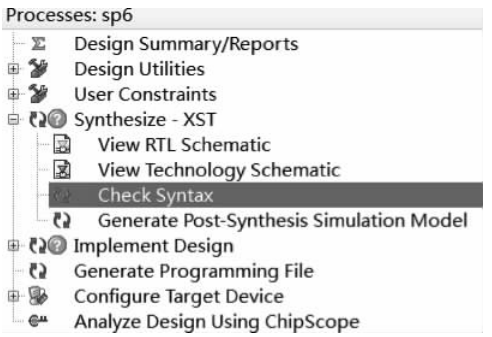

图 5.16　Processes 视窗

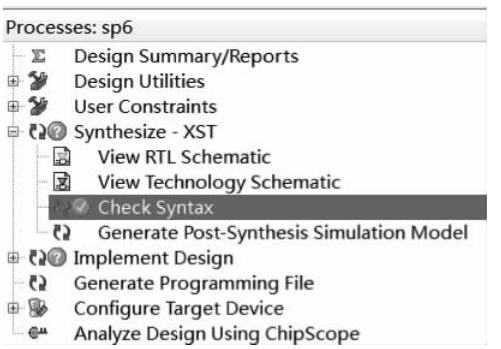

图 5.17　语法检查成功

与此同时，在 ISE 最下方的 Console 窗口中，打印了语法检查结果的报告。如图 5.18 所示，这里看到语法检查没有发现任何的 errors 和 warnings。

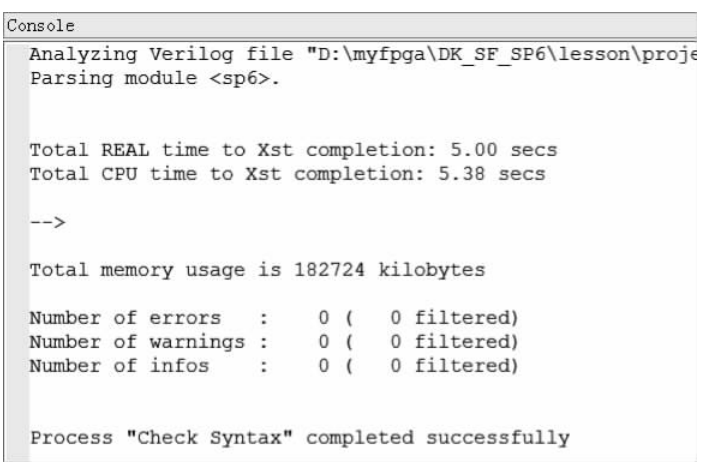

图 5.18　语法检查成功信息

当然，如果有语法错误时，会是什么情况呢？大家不妨自己动手试试，随便将源代码中的某个“；”去掉，在重新进行 Check Syntax 后，就可以出现如图 5.19 和图 5.20 所示的提示和报告。

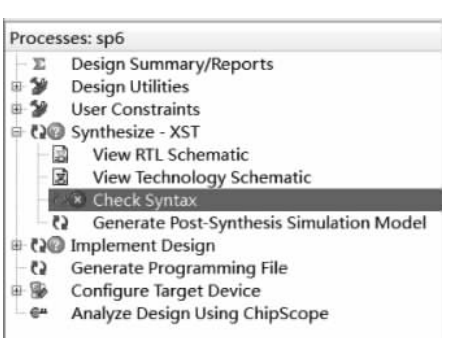

图 5.19　语法检查失败

语法检查有一点非常好，就是在 ERROR 中会明确定位具体出错的位置，即某一行代码附近（注意是附近，不一定就能准确定位到错误的那一行，可能是它的上下行）有错误，可以根据这个提示查找错误。另外必须提醒注意的是，并不是有

图 5.20 语法检查失败信息

几个 ERROR 就表示有几个实际的 ERROR，可能多个 ERROR 对应的是一个实际的 ERROR。

5.4 Modelsim 仿真验证

5.4.1 ISE 基本设置

既然语法检查通过了，那么接下来不妨小试牛刀，让仿真工具 Modelsim 来输出波形验证设计结果和预想的是否一致。

在用 Modelsim 仿真前，在 ISE 中需要确认几个设置。

首先，如图 5.21 所示，执行 Project→Design Properties…命令。

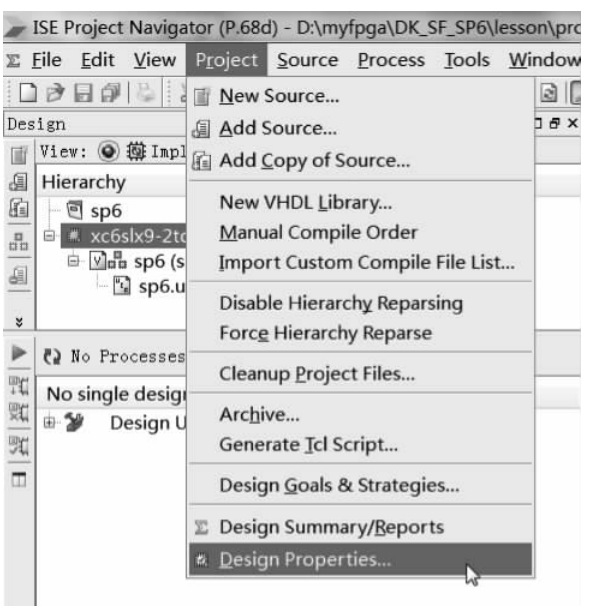

图 5.21 Design Properties 菜单

如图 5.22 所示，确认这里 Simulator 的选择为 Modelsim-SE Mixed，这在新建工程时已经设定好了，为保万无一失，还是确认一下。

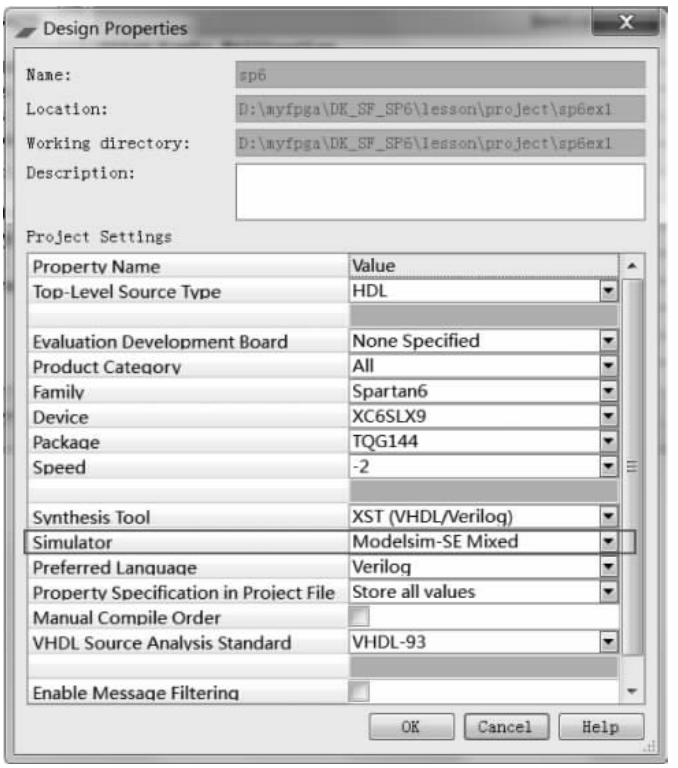

图 5.22　仿真工具设置

其次，如图 5.23 所示，从上到下依次执行以下三步操作。

图 5.23　操作步骤

(1) 切换 View 模式到 Simulation 模式。

(2) 选中 sp6.v 源文件。

(3) 右击 Processes 下的 Simulate Behavioral Model,单击弹出菜单中的 Process Properties 选项。

如图 5.24 所示,弹出 Process Properties 设置窗口,在右边的 Compiled Library Directory 后面填入之前编译库时设置的已编译库的路径,单击旁边的"…"按钮来选定也行。前面将库路径设置为"C:\Xilinx\Xilinx_lib",因此这里也这么设置。其他选项默认即可,单击 OK 按钮完成设置。

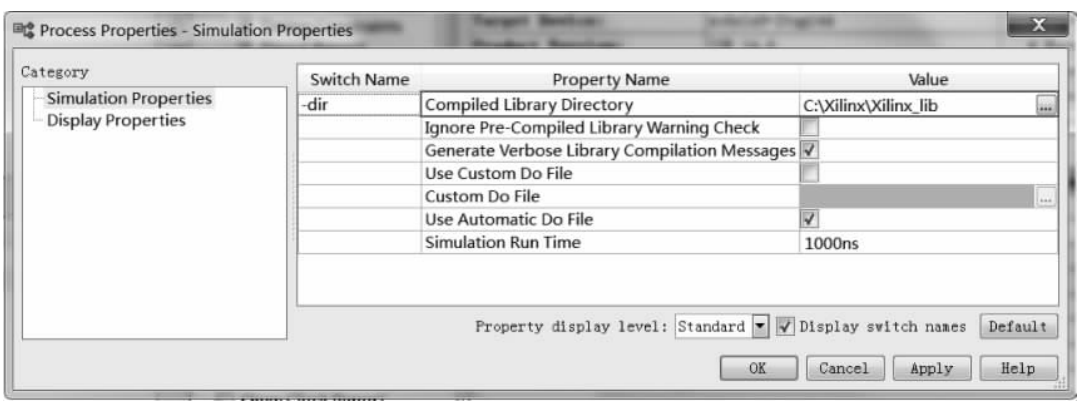

图 5.24 设置 Compiled Library Directory

5.4.2 测试脚本创建与编辑

回到 Simulation→Hierarchy 窗口中,在任意空白处单击右键,在弹出的菜单中选择 New Source,如图 5.25 所示。

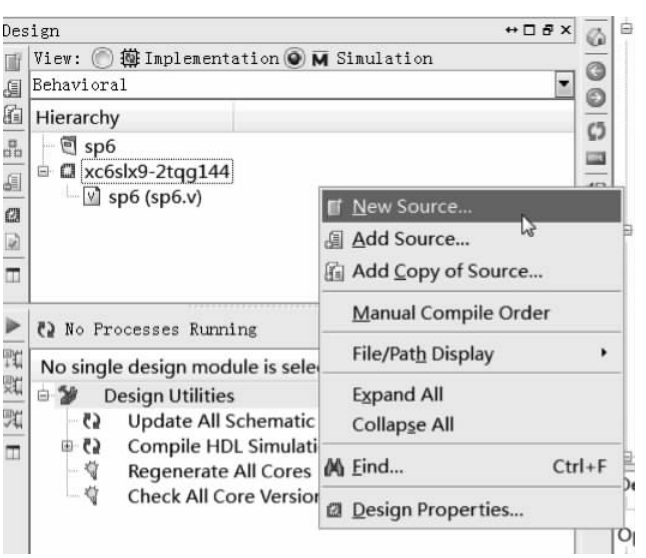

图 5.25 New Source 菜单

如图 5.26 所示，选择新建源文件类型是 Verilog Test Fixture，在工程路径下新建一个名为 testbench 的文件夹，同时将新建文件的路径定位到这个文件夹下面。和前面新建 source_code 文件夹的初衷一样，新建 testbench 文件夹的目的也只是为了便于文件的分类和管理。还有 Add to project 通常必须勾选上（默认即勾选）。单击 Next 按钮到下一步。

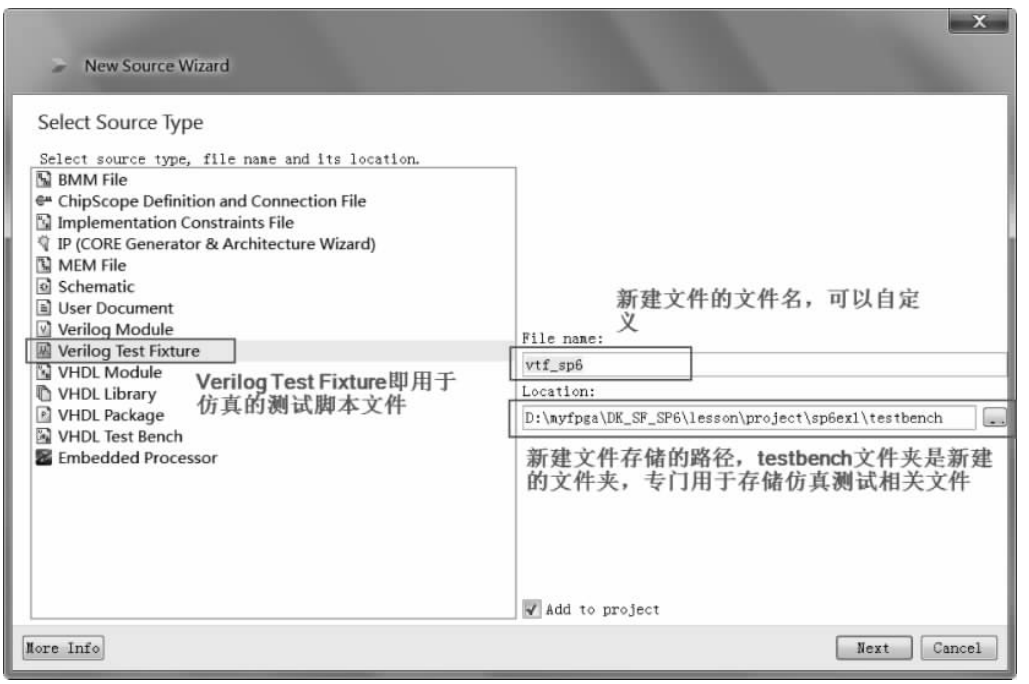

图 5.26　新建测试脚本文件名和路径设置

如图 5.27 所示，这里 Associate Source 是选择测试脚本对应的设计源文件，这里只有一个设计源文件。选中 sp6.v，然后单击 Next 按钮。

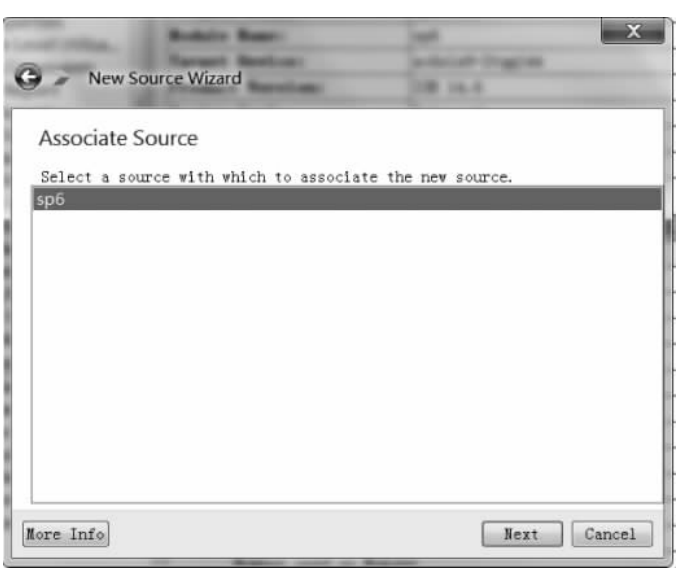

图 5.27　新建测试脚本的相关源文件设置

如图 5.28 所示，罗列出前面设置的报告，没问题就单击 Finish 按钮完成文件创建。

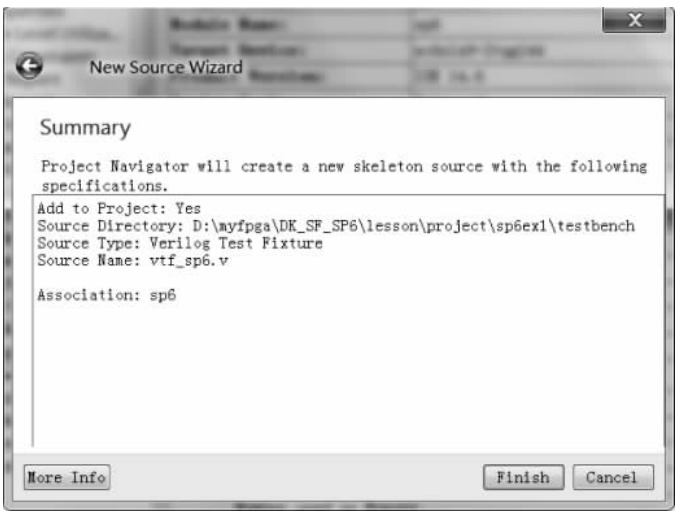

图 5.28　新建测试脚本 Summary 页面

如图 5.29 所示，测试脚本文件在 Notepad++中打开了。这里的测试脚本只是一个基本的模板，它把设计文件 sp6 的接口在这个模块里面例化申明了，还需要动手添加复位和时钟的激励。

```
23  //////////////////////////////////////////////////////////////////////////////////
24
25  module vtf_sp6;
26
27      // Inputs
28      reg ext_clk_25m;
29      reg ext_rst_n;
30
31      // Outputs
32      wire clk_12m5;
33
34      // Instantiate the Unit Under Test (UUT)
35      sp6 uut (
36          .ext_clk_25m(ext_clk_25m),
37          .ext_rst_n(ext_rst_n),
38          .clk_12m5(clk_12m5)
39      );
40
41      initial begin
42          // Initialize Inputs
43          ext_clk_25m = 0;
44          ext_rst_n = 0;
45
46          // Wait 100 ns for global reset to finish
47          #100;
48
49          // Add stimulus here
50
51      end
52
53  endmodule
```

图 5.29　测试脚本代码

有读者可能是第一次接触测试仿真，关于仿真的基本概念请查看《FPGA 设计实战演练（逻辑篇）-ch9 -设计仿真. pdf》。

vtf_sp6. v 文件需要做一些编辑，增加复位和时钟信号，修改后代码如下。

```
module vtf_sp6;
    //Inputs
    reg ext_clk_25m;
    reg ext_rst_n;
    //Outputs
    wire clk_12m5;
    //Instantiate the Unit Under Test (UUT)
    sp6 uut (
        .ext_clk_25m(ext_clk_25m),
        .ext_rst_n(ext_rst_n),
        .clk_12m5(clk_12m5)
    );
    initial begin
        //Initialize Inputs
        ext_clk_25m = 0;
        ext_rst_n = 0;
        //Wait 100 ns for global reset to finish
        #100;
        ext_rst_n = 1;
        //Add stimulus here
        #2000;
        $stop;
    end
    always #20 ext_clk_25m = ~ ext_clk_25m;     //产生 25MHz 时钟源
endmodule
```

5.4.3 调用 Modelsim 仿真

保存修改后的代码，然后回到 ISE 中。

此时，如图 5.30 所示，vtf_sp6. v 已经成了这个仿真 Hierarchy 的顶层了，它下面是设计文件 sp6. v。选中 vtf_sp6. v 文件，然后双击 Simulate Behavioral Model，随后 Modelsim 将被调用，启动仿真。

弹出 Modelsim 后，如图 5.31 所示，可以打开 Wave 查看，同时单击 Zoom Full 按钮，Modelsim 的使用并不难，大家要多动手，所有的菜单按钮都简单易懂，有些地方右键菜单也有很多功能，如果这里一一介绍恐怕需要一本书，所以大家自动动手，贵在尝试，很快就会上手。

效果如图 5.32 所示。

通过这个简单的工程，读者就可以掌握使用 ISE 进行工程创建、设计文本创建和编辑、测试脚本创建和编辑、使用 Modelsim 进行仿真等基本的技能了。当然，这只是刚刚领进门，让读者熟悉一下工具的一些基本操作。这个例程就到这里，就不往下进行板级的实验了。

图 5.30 启动仿真

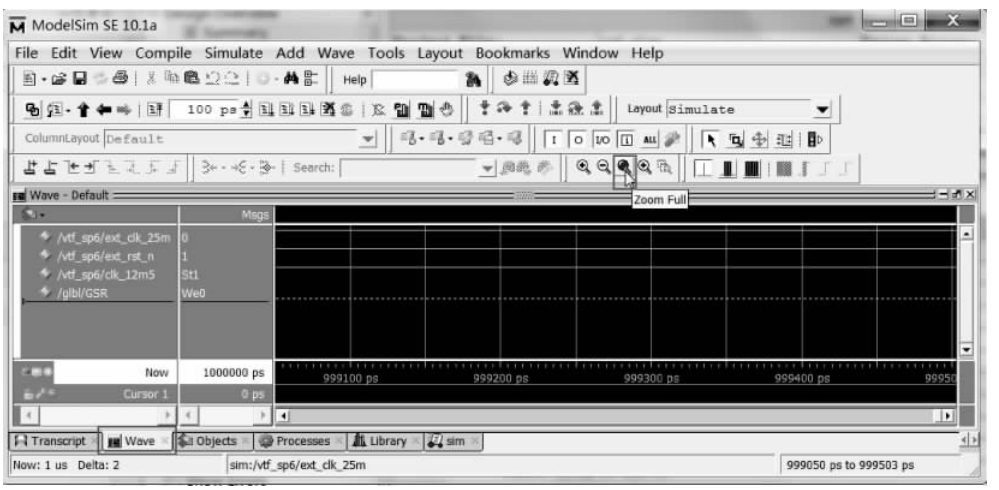

图 5.31 仿真界面

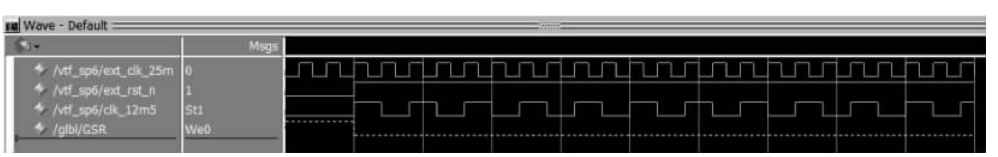

图 5.32 仿真波形