第5章 数字电路设计与仿真

CHAPTER 5

随着数字电路设计工艺的发展和设计规模的不断扩大，EDA 工具在数字电路设计过程中扮演着越来越重要的角色。本章以 ModelSim SE 2020.4 软件为例介绍数字电路的设计与仿真。通过本章的介绍，可以从整体上了解数字电路设计的流程及 ModelSim 的使用概况，并掌握 ModelSim 的基本仿真使用方法。

ModelSim 是一款功能强大的仿真软件，可以对 VHDL、Verilog、System Verilog 和 SystemC 等格式的文件进行仿真。由于每种编程语言的语法和文件结构都不尽相同，ModelSim 对不同类文件的仿真过程也有一些差异。

5.1 数字电路设计及仿真流程

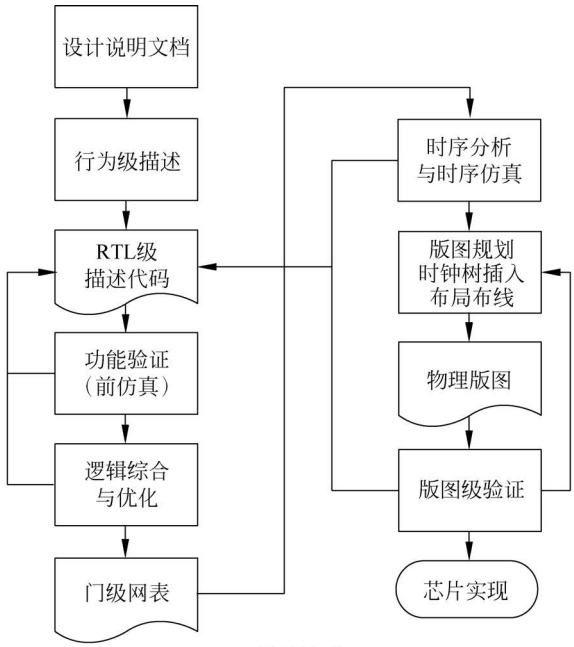
本节介绍数字电路设计的基本流程及采用 ModelSim 进行数字电路设计及仿真的基本流程，学习基本的数字电路设计和仿真方法。

5.1.1 数字电路设计流程

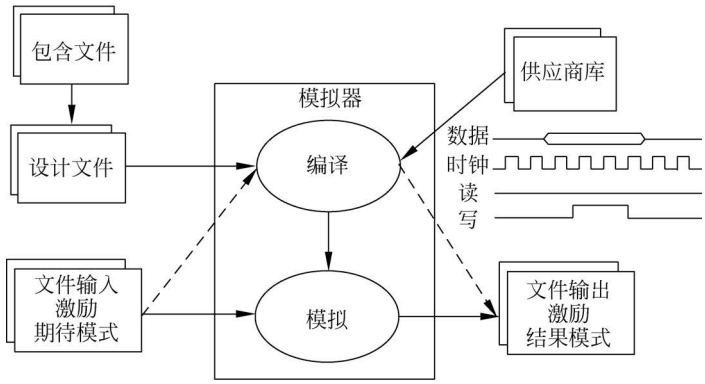
数字电路设计流程包括两大类：正向(top-down)设计流程和反向(bottom-up)设计流程。正向设计流程指的是从顶层的功能设计开始，根据顶层功能的需要，细化并完成各个子功能，直至达到底层的功能模块为止。反向设计流程正好相反，设计者最先得到的是一些底层的功能模块，使用这些底层的模块搭建出一个高级的功能，按照这种方式继续直至顶层的设计。

在数字电路设计的最初阶段，EDA 工具功能并不强大，所以两种设计流程都被采用。随着 EDA 工具的功能逐渐增强，正向设计流程得到了很好的支持并逐步成为主流的 IC 设计方法。这种方法也符合设计者的思维过程：当拿到一个设计项目时，设计者首先想到的是整体电路需要达到哪些性能指标，进而采用高级语言尝试设计的可行性，再经过 RTL 级、电路级直至物理级逐渐细化设计，最终完成整个项目。图 5-1 所示为数字电路设计的基本流程。

设计的最开始阶段一定是设计文档的编写，设计说明文档主要包含了设计要实现的具体功能和期待实现的详细性能指标，包括电路整体结构、输入/输出(I/O)接口、最低工作频率、可扩展性等参数要求。完成设计说明文档后，需要行为级描述待设计的电路。行为级描述可以采用高级语言，如 C/C++等，也可以采用 HDL 来编写。这个阶段的描述代码并不要求可综合，只需要搭建出满足设计说明的行为模型即可。



(a) 设计流程



(b) 设计文件组织

图 5-1　数字电路设计基本流程

行为级描述之后是RTL级描述。这一阶段一般采用VHDL或Verilog HDL来实现。对于比较大的设计，一般是在行为级描述时采用C/C++语言搭建模型，在RTL级描述阶段，逐一地对行为模型中的子程序进行代码转换，用HDL语言代码取代原有的C/C++语言代码，再利用仿真工具的接口，将转换成HDL代码的子程序加载到行为模型中，验证转换是否成功，并依次转换行为模型中的所有子程序，最终完成从行为级到RTL级的HDL代码描述。这样做的好处是减少了调试的工作量，一个子程序转换出现错误，只需要更改当前转换的子程序即可，避免了同时出现多个待修改子程序的杂乱局面。

RTL模型的正确与否，是通过功能验证来确定的，这一阶段也称前仿真。前仿真的最大特点就是没有加入实际电路中的延迟信息，所以，前仿真的结果与实际电路结果还是有很大差异的。不过在前仿真过程中，设计者只关心RTL模型是否能完成预期的功能，所以称为功能验证。

当RTL模型通过功能验证后，就进入逻辑综合与优化阶段。这个阶段主要由EDA工具来完成，可以给综合工具指定一些性能参数、工艺库等，使综合出来的电路符合要求。

综合生成的文件是门级网表。网表文件包含了综合之后的电路信息，其中还包括了延迟信息。将这些延迟信息反标注到 RTL 模型中，进行时序分析。主要检测的是建立时间(setup time)和保持时间(hold time)。其中建立时间的违例和较大的保持时间违例必须要修正，可以采用修正 RTL 模型或修改综合参数来完成。对于较小的保持时间违例，可以放到后续步骤中修正。对包含延迟信息的 RTL 模型进行仿真验证的过程称为时序仿真，时序仿真的结果更加逼近实际电路。

设计通过时序分析后，就可以进行版图规划与布局布线。这个阶段是把综合后的电路按一定的规则进行排布，也可以添加一些参数对版图的大小和速度等性能进行约束。布局布线的结果是生成一个物理版图，再对这个版图进行仿真验证，如果不符合要求，那就需要向上查找出错点，重新布局布线或修改 RTL 模型。如果版图验证符合要求，这个设计就可以送到工艺生产线上，进行实际芯片的生产。

当然，上述流程只是一个基本的过程，其中很多步骤都是可以展开成许多细小的步骤，也有一些步骤(如形式验证)在上述流程中并没有体现。

5.1.2　ModelSim 工程仿真流程

ModelSim 的工程仿真流程如图 5-2 所示，概括为 5 步：首先建立一个工程，然后向工程中添加设计文件，接下来编译设计文件，之后运行仿真，最后进行调试。

1. 创建工程及工程库

开始一个设计之前，先要在 ModelSim 中创建一个工程和对应的工程库。仅采用新建工程的方式直接创建默认工程库。具体按如下的步骤进行操作：

(1) 创建新工程。在 ModelSim 菜单栏中选择 File→New→Project 命令。

(2) 输入工程名称。在弹出的对话框中输入工程名(Project Name)并进行工程设置，直接采用默认库 work，输入的工程名称为 FA，其他设置保持不变，输入完毕后单击 OK 按钮完成，如图 5-3 所示。

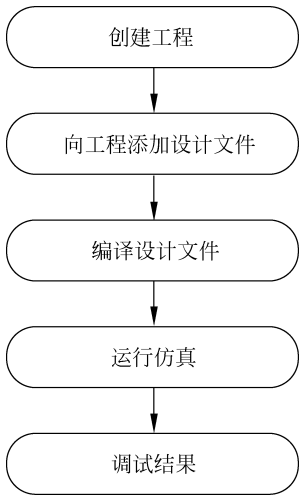


图 5-2　ModelSim 工程仿真流程

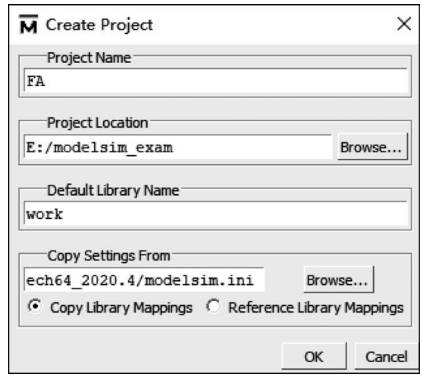


图 5-3　输入工程名称

(3) 创建工程完毕。在第二步中单击 OK 按钮后，新的工程和工程库就被创建了。创建工程前，ModelSim 的 Workspace 窗口中只有 Library 一个标签，当创建工程结束后，Workspace 窗口出现了新标签 Project。由于新建的工程中没有文件，所以显示为空白区域，如图 5-4 所示。至此，工程和工程库创建完毕，可以向工程中加载设计文件了。

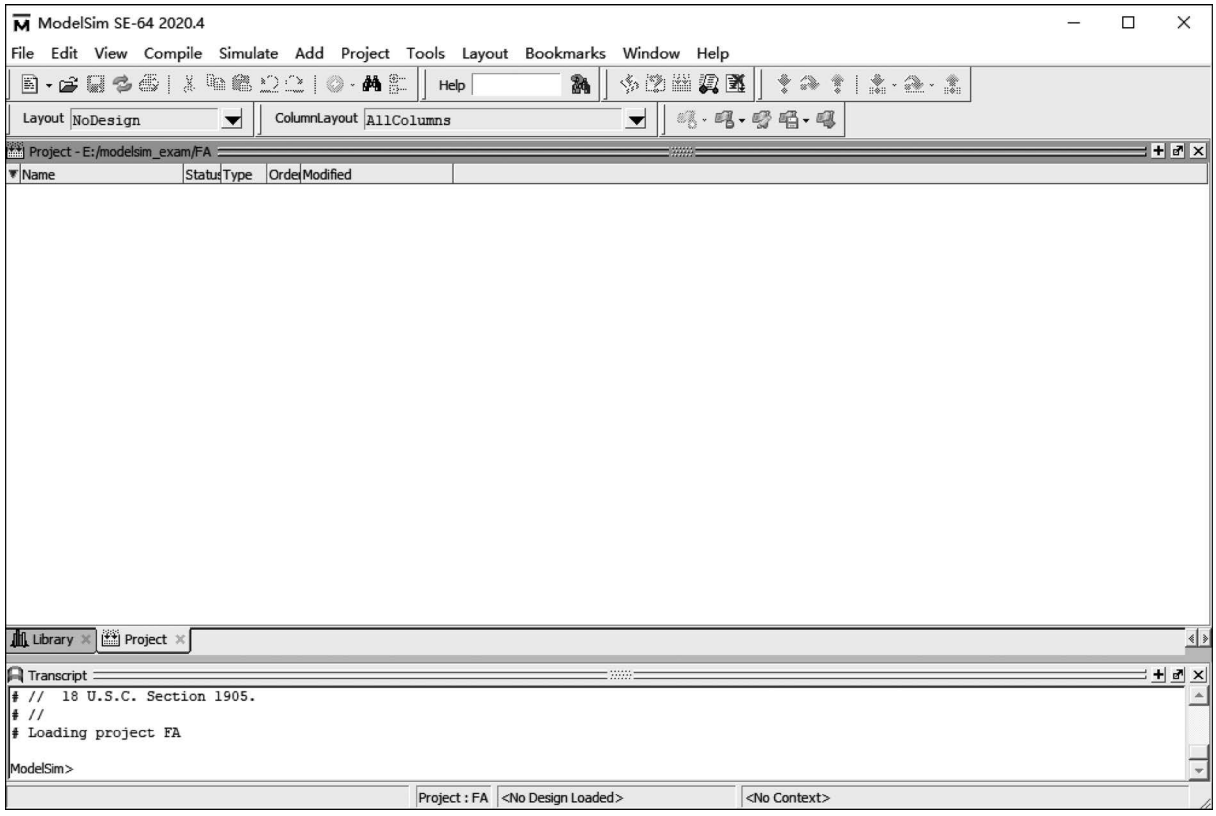


图 5-4　创建工程后的软件界面

2. 创建新文件

向工程中添加设计文件可以有两种方式：创建新文件和加载设计文件。创建新文件步骤如下：

(1) 创建新文件。在创建工程结束后，会弹出对话框，有多种添加方式可供选择。在本例中选择其中的 Create New File 选项，如图 5-5 所示。

(2) 输入文件名。在弹出的对话框中输入文件名称，在本例中输入 Fulladd，将 Add file as type 选项选为 Verilog，单击 OK 按钮完成操作，如图 5-6 所示。注意文件类型默认是 VHDL 类型，一定要选择 Verilog 类型仿真文件才能被正确编译。

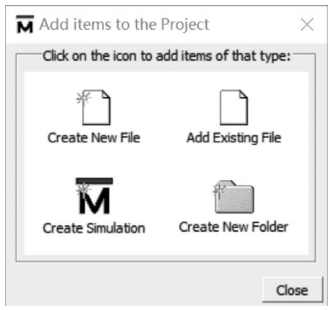


图 5-5　选择创建新文件

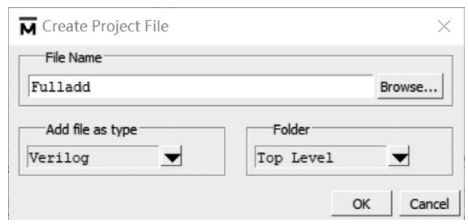


图 5-6　输入文件名及选择文件类型

(3) 新建文件结束。单击 OK 按钮后，就可以在 Project 窗口中看到新加入的文件 Fulladd.v，这时可以对该文件进行设计输入。双击文件，即可在编辑窗口看到文件内部的内容。由于是新建的文件，可

以看到内部是空白的，如图 5-7 所示。

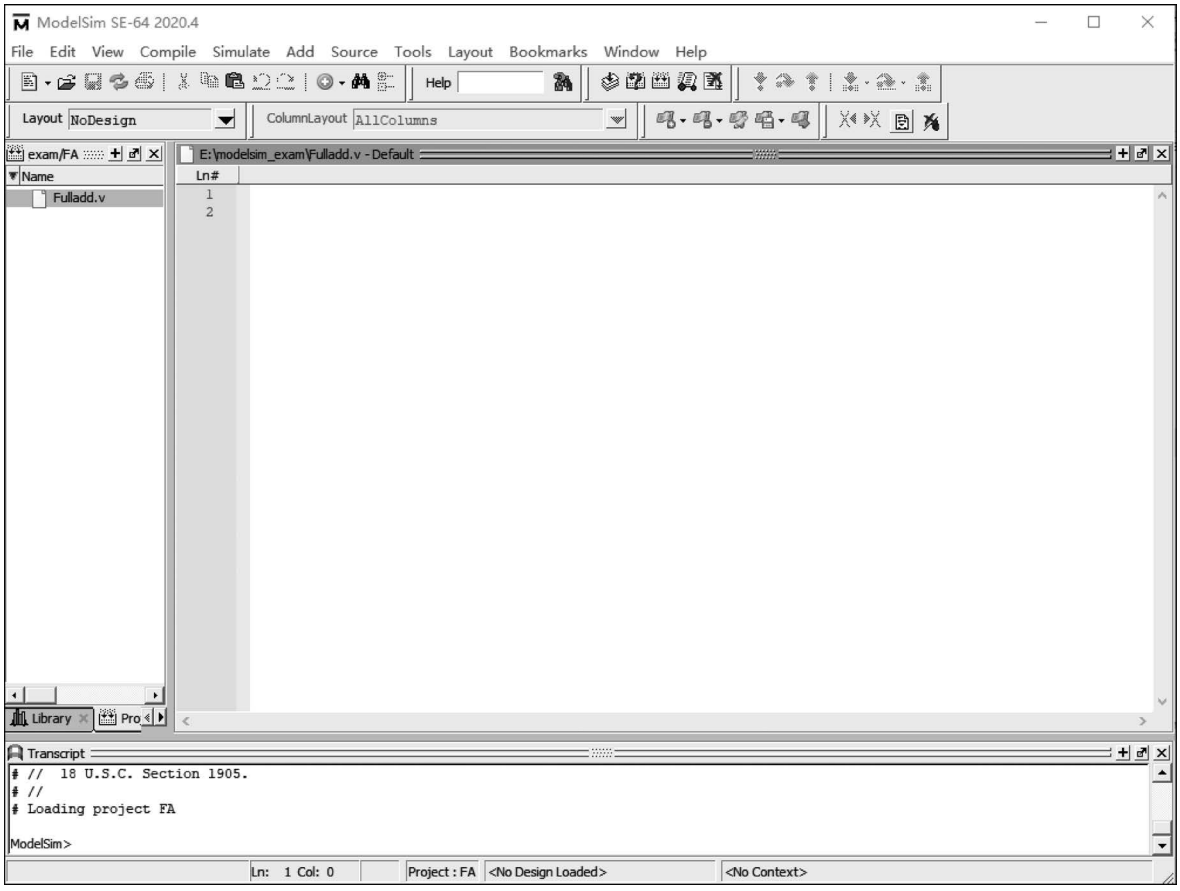


图 5-7　添加文件完成

（4）编辑文件内容。在新建的 Fulladd. v 文件中输入如图 5-8 所示的全加器设计，保存文件即可。

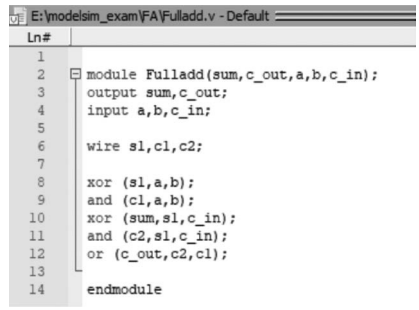


图 5-8　编辑文件内容

3. 加载设计文件

除了新建文件，还可以向工程中添加已有的设计文件。具体步骤如下：

（1）选择添加已有文件。在创建工程结束弹出的对话框中，如图 5-5 所示，选择其中的 Add Existing File 选项。

（2）选择文件路径。选择“添加已有文件”后会有如图 5-9 所示的对话框，选择添加文件的路径。ModelSim 默认的路径是安装文件夹中的 example 目录。当然，也可以手动选择其他目录添加文件。这里选择将事先准备好的工程文件夹中的 test. v 文件加载到工程中。test. v 文件中的内容见图 5-13。

（3）加载完成。单击 OK 按钮后，可以看到 Project 窗口又加入了一个 test. v 文件，如图 5-10 所示。加入的两个文件对应的 Status 栏都是“?”标志，这是设计文件还没有被编译的标志，接下来就要编译文件。

图 5-9　添加已有文件路径名

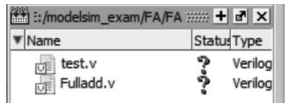


图 5-10　成功添加已有文件

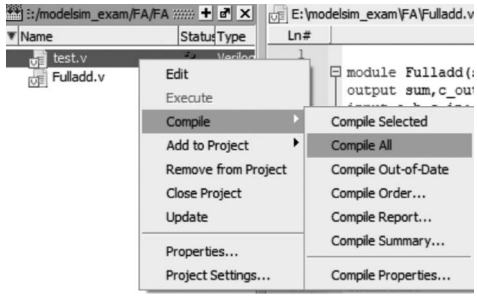


图 5-11　利用菜单选项编译

图 5-12　利用快捷工具栏编译

4. 编译源文件

编译过程是仿真器检查被编译文件是否有语法错误。没有被编译的文件是不可以进行仿真的。编译的方式先简单介绍两种：

(1) 利用菜单选项编译。在 Project 标签中选择一个文件，右击会出现快捷菜单，选择 Compile 选项，会出现一系列的编译方式。最常用的是前两个，即编译选中文件(Compile Selected)和编译所有文件(Compile All)，如图 5-11 所示。

(2) 利用快捷工具栏编译。在 ModelSim 菜单栏下方有一排快捷工具栏，其中有编译按钮，可以直接单击对文件进行编译，如图 5-12 所示。共有三个编译按钮，左侧的是编译选中的文件，中间的是编译有修改的文件，右侧的是编译所有文件。

编译通过后原有的问号会变成对号，同时在命令窗口中会出现提示：Compile of XXX was successful，如图 5-13 所示。

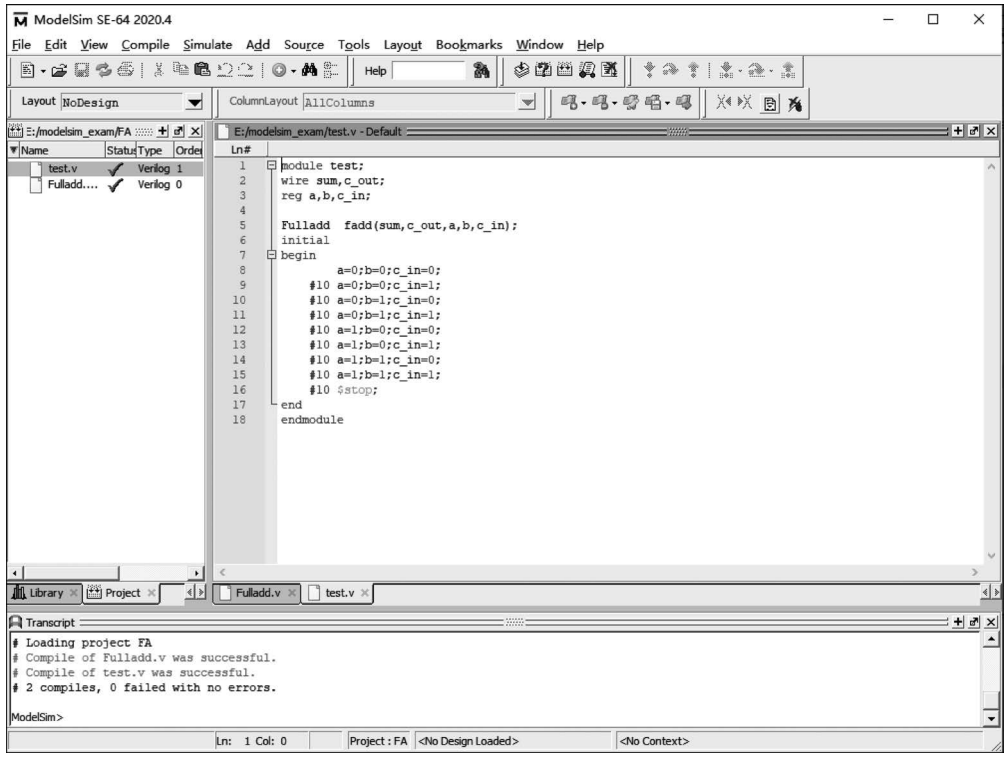


图 5-13　编译通过后的提示

5. 运行仿真

编译通过的文件就可以进行仿真。仿真的具体步骤如下：

(1) 开始仿真。仿真的方式有很多，这里采用最简单的方式：单击快捷栏的仿真按钮开始仿真，如图 5-14 所示，左侧的按钮是开始仿真，右侧的按钮是停止仿真。

(2) 选中仿真文件。开始仿真后会出现 Start Simulation 对话框，如图 5-15 所示。选中需要进行仿真的文件，在这里选中顶层模块 test。单击 Optimization Options 按钮，打开 Optimization Options 对话框，选择 Apply full visibility to all modules，单击 OK 按钮后退出，如图 5-16 所示，在 Workspace 区域会出现新的标签 sim，同时在命令窗口还会有对应的提示信息。

图 5-14　利用快捷工具栏仿真

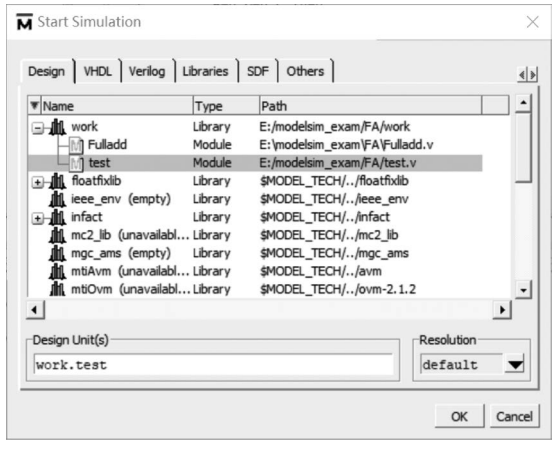


图 5-15　开始仿真窗口

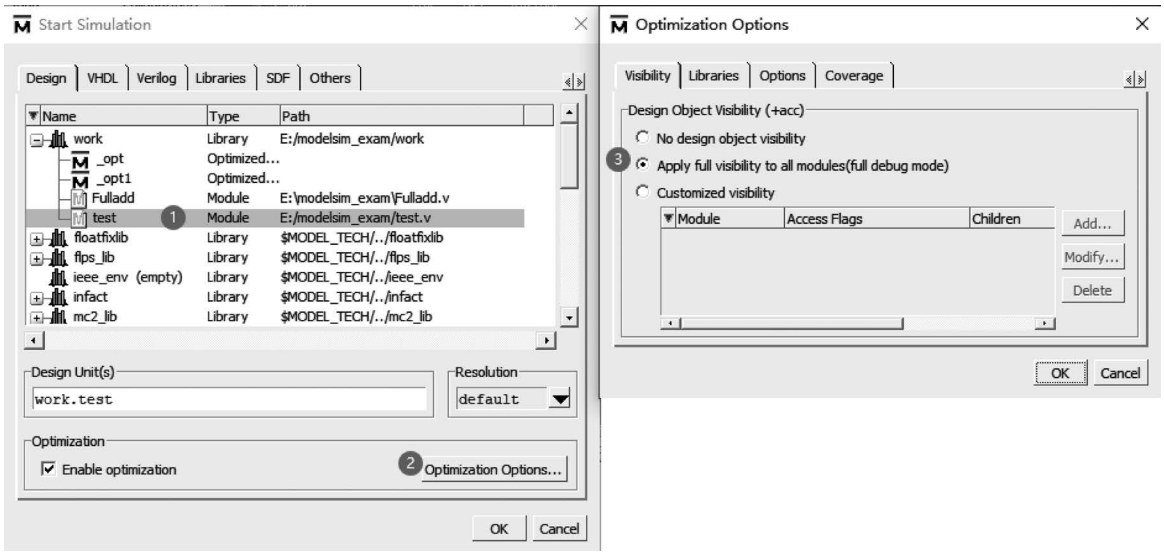


图 5-16　仿真标签及命令窗口提示

(3) 添加待观察的信号。选中 test 模块并右击,在弹出的快捷菜单中选中 Add to Wave,出现另一个新窗口 wave。这里就是观察信号变化的区域,在仿真没有运行的时候,输出的信号均为空,如图 5-17 所示。

(4) 运行仿真。快捷工具栏中也有运行仿真按钮,如图 5-18 所示。共有四个运行按钮,从左到右依次为 Run、ContinueRun、Run-All 和 Break。这里单击 Run-All 进行仿真。

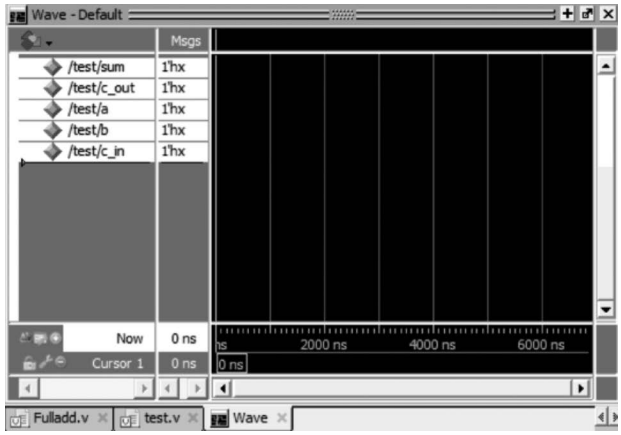


图 5-17　添加待观察信号

图 5-18　仿真工具按钮

6. 查看结果

在单击 Run-All 后,可以在波形窗口(wave 窗口)观察输入与输出信号的变化,如图 5-19 所示。如果在设计中有一些系统函数(如 display)等,在命令窗口还会看到系统函数相应的提示。在本例中命令窗口没有输出。

至此,ModelSim 的基本仿真流程就结束了,根据最后的仿真波形,可以验证程序是否正确。以光标处为例,波形的高电平处为信号 1,低电平处为信号 0,输入的信号 a 为 1,b 为 0,c_in 为 1,在全加器中可知输出的结果应该为 10,对比上方的信号,c_out 为 1,sum 为 0,结果正确。

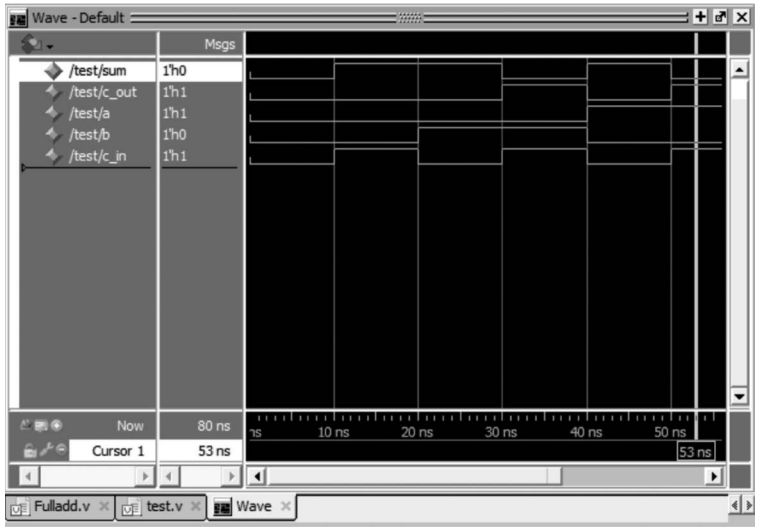


图 5-19　wave 窗口的输入与输出波形

7. 工程调试

在实际的设计中，错误是不可避免的，ModelSim 提供了丰富的错误提示类型，帮助设计者快速发现错误的位置和错误的类型。一般情况调试过程如下：

(1) 编译错误提示。此时 Status 栏会显示一个红色的叉，表示编译不通过，即源文件中有错误。此时在命令窗口中会出现红色字体的提示，告知设计者哪个文件出现了几个错误，可能包含 error，也可能包含 warning，如图 5-20 所示。

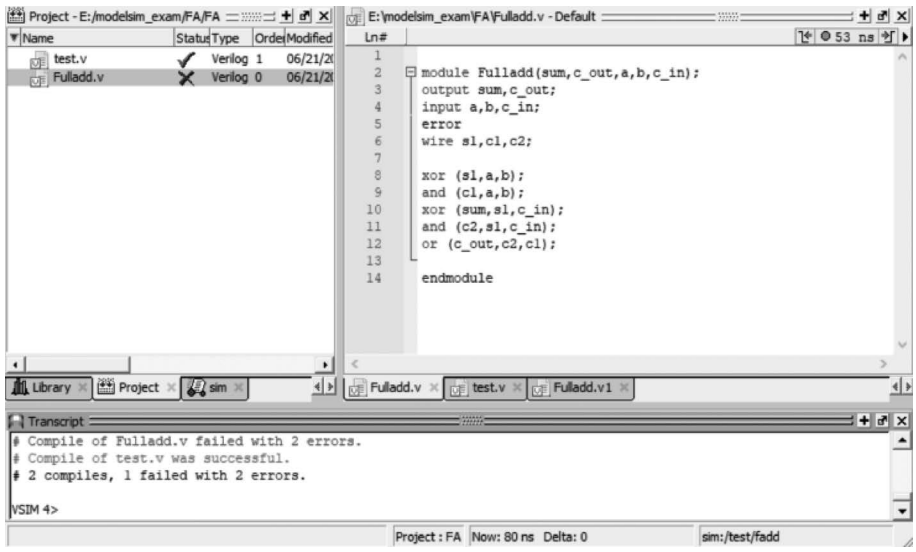


图 5-20　编译错误提示

(2) 查找错误原因和位置。双击命令窗口中的提示，会弹出一个对话框提示，其中会显示出在文件的第几行出现了哪种错误。如图 5-21 中提示，在文件的第 5 行出现了语法错误。这时文件的第 5 行会以醒目的颜色标出来，方便设计者查找。当然，同其他设计语言一样，软件指出的错误位置不一定是真正的错误，只是提供一个参考，具体的调试还需要设计者来进行。

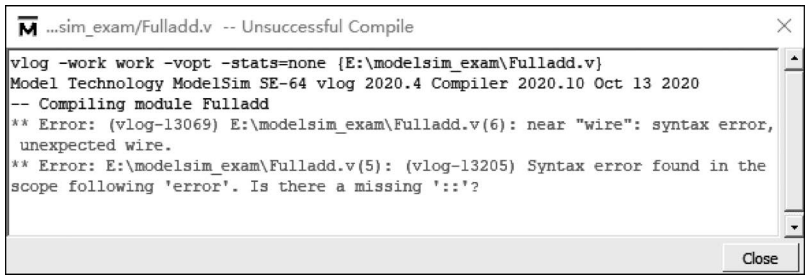


图 5-21　错误原因和位置提示

将上述的基本过程连接起来,就构成了一个简单的工程实例。从创建工程开始,经过设计文件的加载,之后对设计文件进行编译和调试,调试通过后可以按照仿真的步骤进行仿真并查看最后的输出结果。

有了输出结果并不意味着设计的完成,设计是要实现一定功能的,如果仿真器的输出结果与设计者最初的设计初衷不相符,就证明设计出现了问题,需要修改源文件。所以,一个设计并不是通过了编译就宣告成功了,需要对仿真结果进行细致分析,直至确定达到了需要完成的功能。

5.2　仿真激励及文件

仿真是工程设计过程中的一个重要步骤,其作用是验证某些设计单元是否满足设计者的最初要求,属于验证的一种手段。设计者根据最初撰写的设计文档完成了某一个或某一些功能单元,采用的语言可能有很多种,如数字电路设计中的 VHDL、Verilog 和 SystemC 等。在完成这些单元后,需要知道这些单元是否能按照文档中预期的要求来完成某些功能,所以编写另外一个文件,格式可能有很多种,但实现的是同一个功能,就是产生某些激励,把这些激励连接到设计单元的输入端或输出端,观察设计单元在这些激励的作用下会产生什么样的输出响应结果,再对这些输出响应结果进行分析,判断是否严格执行了预期的功能。如果能够很好地达到设计的初衷,这个设计单元就被认为是一个符合要求的设计单元,可以按照设计流程进行下一步的处理,如果不能达到设计要求,则需要修改设计单元并对修改后的单元重复以上操作,这就是仿真的作用。在仿真过程中使用的激励被称为测试平台(testbench)。

仿真可以分为软件仿真和硬件仿真。软件仿真指的是使用软件对设计代码进行测试,软件中只能模拟建立一个尽可能真实的环境,使整个设计仿佛工作在实际环境中一样。硬件仿真通常是使用代码生成某些中间文件,下载到 FPGA 等硬件平台中,使用实际电路测试设计的正确性。应该说,硬件仿真比软件仿真更具真实性和可信性,但付出的代价也较大,所以现在一般都是采用软硬件混合仿真的方式。

单纯使用 ModelSim 进行仿真是软件仿真,使用的设计单元是各种代码描述形式,且没有与硬件的交互。常用的硬件描述语言的仿真工具有很多,如 VCS、Verilog-XL 等都是很著名的仿真器,ModelSim 也是其中之一。一般的仿真器只能实现一种语言的仿真,例如,对 VHDL 语言的仿真或对 Verilog 语言的仿真,ModelSim 是唯一采用单一内核就可以对 VHDL 和 Verilog 两种语言进行仿真的,并且支持混合仿真。而且 ModelSim 为每个 FPGA/CPLD 厂商都提供了适应该厂商的 OEM 版本,所以在 FPGA/CPLD 方面应用得比较广泛。

硬件描述语言的软件仿真根据是否加入延迟信息可以分为功能仿真和时序仿真。功能仿真仅仅验

证设计代码是否可以完成预定功能，不考虑实际的延迟信息，所以当某一输入激励发生变化时，产生的响应会立刻出现在输出端，即输入和输出之间没有时间的延迟。时序仿真加入了信号传输需要的时间延迟，这种延迟信息一般来自厂商，例如，FPGA 厂商或 IC 设计厂商会提供一个元器件库或设计库，库中包含了该厂商对不同基本器件的延时描述，根据这些库计算当前设计在实际电路中可能出现的延迟状态，对这种延迟状态进行仿真。功能仿真和时序仿真分处于设计流程的不同阶段，功能仿真一般在设计代码完成后进行，验证功能是否正确，如果正确则表示可以尝试进行综合，在综合之后就会根据综合时的约束设置产生时序信息，这时可以进行时序仿真，验证设计是否满足时序要求，主要是 setup time（建立时间）和 hold time（保持时间）的检查。

ModelSim 可以方便地、独立地进行功能仿真，但是由于没有器件库，不能进行时序仿真，需要与第三方软件协同后进行仿真，功能仿真结果和时序仿真结果进行仿真分析的过程和方式都是一样的，本节以 ModelSim 为例介绍如何进行仿真激励设置。

5.2.1　利用波形编辑器产生激励

进行仿真分析时，需要提供设计模块和激励模块。设计模块描述设计功能，激励模块提供测试向量，测试向量又可称为激励。在 ModelSim 中激励产生的方式有两种：一种是利用 ModelSim 自带的波形编辑器生成激励；另一种是通用式的，即使用编程语言描述一系列激励。

1. 创建波形

创建波形前首先需要有编译好的设计单元，有了设计单元后，可以通过 4 种途径启动波形编辑器：分别是从库中启动波形编辑器，从结构标签 sim 中启动波形编辑器，从 Objects 窗口中启动波形编辑器和从波形窗口中启动波形编辑器。

编译通过的设计单元会映射到库中，在库中选定需要创建激励的设计单元，右击菜单，选择其中的 Create Wave，如图 5-22 所示。ModelSim 会自动识别设计单元的输入/输出端口列表，在波形编辑窗口中把这些端口一一列出。

如果存在激励文件时，仿真启动的顶层单元一般是激励文件。当没有激励文件时，仿真启动的顶层单元就是设计文件的顶层模块。启动仿真后，选中 sim 标签，在菜单栏中选择 Structure→Create Wave 命令启动波形编辑器，如图 5-23 所示。注意，Structure 菜单需要选中 sim 标签后才会出现。

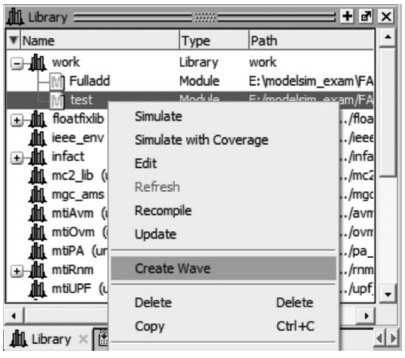


图 5-22　从库中启动波形编辑器

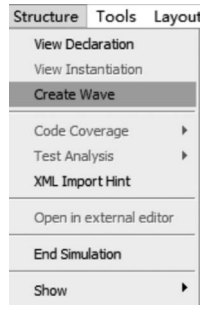


图 5-23　选中 sim 标签启动波形编辑器

从 sim 标签中创建波形，波形编辑器中的波形列表会显示选中设计单元的全部端口。有些时候不需要观察全部端口，可以利用 Objects 窗口，选中 Objects 窗口中的部分端口同样右击菜单，如图 5-24 所示，可以选择 Modify，在子菜单中有 Apply Clock 和 Apply Wave 两个选项，可以为选中的信号添加

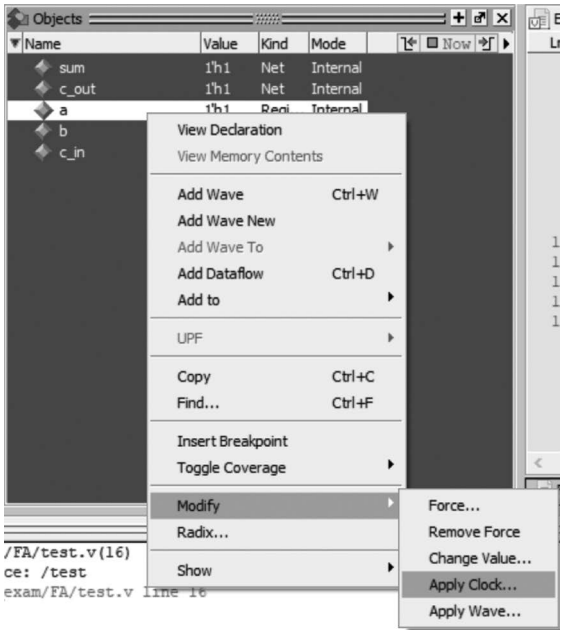


图 5-24 从 Objects 窗口启动波形编辑器

波形。由于 clock 时钟信号的特殊性，这里把 clock 信号单独做成了一个选项。

以上三种启动波形编辑器的方式略有不同。从库中启动波形编辑器时，波形编辑器的时间刻度是 ModelSim 中默认的最小刻度，即默认刻度是 1ns，这样可能会带来影响。例如，如果设计单元中指定了时间刻度是 ps 级或 ms 级，这时生成波形的刻度就显得过大或过小。另外两种方式由于是在仿真后启动的，设计单元的时间刻度都已经读入了 ModelSim 内核中，所以此时启动波形编辑器，会采用设计中需要的时间刻度。为避免引入麻烦，推荐使用后三种方式，如果一定要从库中启动波形编辑器，要注意实际的时间刻度。

另外，波形编辑器只能提供一些简单的信号，包括时钟、计数器、周期变化的随机数、恒定常量值和重复值，虽然也可以通过后面介绍的编辑波形的方式来产生一些不规则的信号波形，但是相对来说比较麻烦，如果波形不是波形编辑器中所包含的类型，建议还是采用描述语言的方式来编写激励文件。

启动波形仿真器后在波形窗口中会出现如图 5-25 所示的端口列表，所有上述波形编辑器支持的端口都会在这里列出，每一个信号前端的◆标记都是一个类似波形的符号，表示该信号处于波形编辑的状态。启动后的波形编辑器处于初始状态，由于没有进行编辑，所有的端口均显示为 No Data。

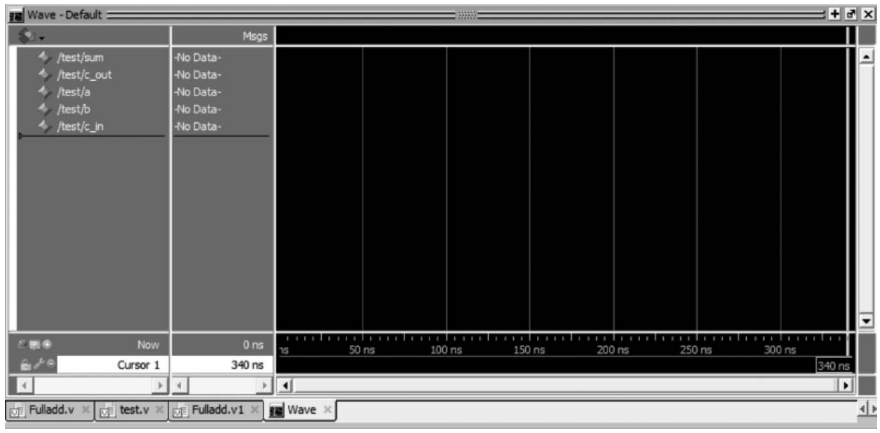


图 5-25 初始状态下的波形编辑器

如果要对波形编辑器中某一信号进行赋值，可以选中该信号(选中后信号底色变为白色)，在右键菜单中选择 Edit→Wave Editor→Create/Modify Waveform 或者选中波形窗口，在菜单栏中选择 Wave→Wave Editor→Create/Modify Waveform，如图 5-26 所示，这时会出现一个创建波形向导，如图 5-27 所示，使用该向导可以方便快捷地为信号赋值。

向导窗口的左侧是文字说明，提示可以生成哪些波形及本步进行何种操作。右侧提供了选项和设置值。Patterns 是指定该信号要赋予何种类型的波形，ModelSim 的波形编辑器只能支持五种类型的信

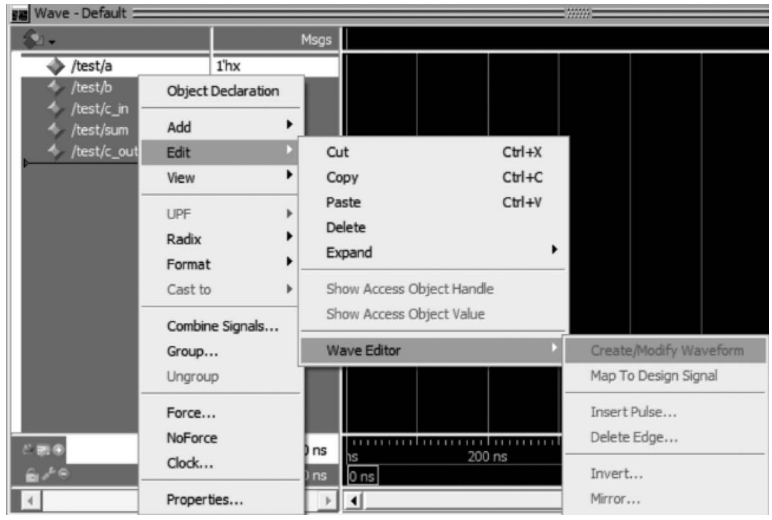


图 5-26　波形编辑

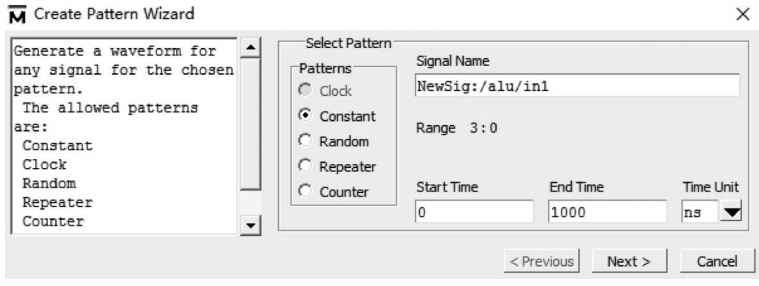


图 5-27　创建波形向导

号，可以看到可选的类型也是五种：Constant、Clock、Random、Repeater 和 Counter，可以在这五种类型中选择一种。如果信号的位数多于 1 位，则该信号不可能是时钟信号，所以此时时钟信号会变为不可选。右侧的 Signal Name 显示当前要编辑的信号名称，在信号名称的下方还显示了 Range 3：0，表示该信号是一个 4 位的信号，若信号只有 1 位，这里就没有显示。在右下方的位置可以设置该信号的起始时间（Start Time）、终止时间（End Time）和时间单位（Time Unit）。设置好需要的信号类型和起止时间，在下拉菜单中选择需要的时间单位，单击 Next 按钮，会根据选择的不同信号类型而出现不同的提示。

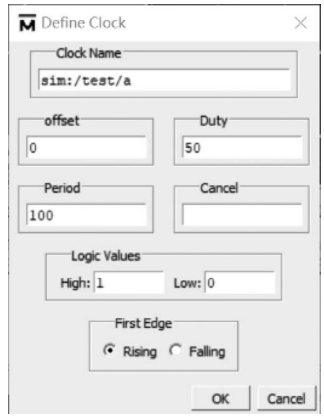


图 5-28　启动时钟波形编辑器

在波形窗口中使用右键菜单选择 Clock 命令，会出现图 5-28 所示的对话框，也是生成时钟，与直接创建窗口稍有不同。图中 Clock Name 编辑框可以输入信号名称，offset 设置时钟偏移量，Duty 设置占空比，Period 设置时钟周期，Logic Values 部分可以设置高低电平的信号值，First Edge 可以选择生成时钟的第一个边沿是上升沿还是下降沿，单击 OK 按钮设置完成。

2. 编辑波形

建立波形后，可以对生成的波形文件进行编辑。编辑波形可以通过命令行形式，也可以使用菜单操

作，由于波形文件比较直观，使用菜单操作起来方便快捷，命令行形式不是很直观，所以这里只介绍使用菜单操作波形的方式。

如果想要编辑一个建立好的波形，首先要更改鼠标的模式，可以在菜单栏中选择 Wave→Mouse Model 进行修改，如图 5-29 所示。鼠标模式有四种：Select Mode(选择模式)、Two Cursor Mode(两个光标模式)、Zoom Mode(缩放模式)和 Edit Mode(编辑模式)，编辑波形必须把鼠标模式调整至 Edit Mode。

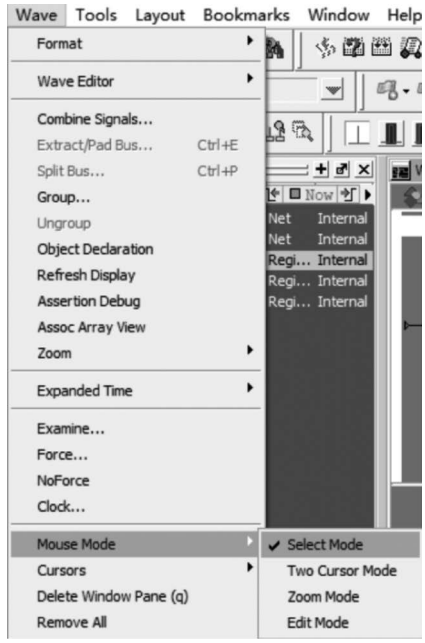


图 5-29 更改鼠标模式

再简单介绍一下选择模式和缩放模式。选择模式在观察波形的时候最常用，也是 ModelSim 的默认状态。缩放模式用来缩放波形，选中此模式后，在波形窗口中用鼠标左键划过一定的区域，这个区域就会被放大到填满整个波形窗口。在编辑波形的时候这两种鼠标模式也是经常使用的，主要用于选定信号或选择区域值，需要改变波形的数据时才切换到编辑模式。

编辑波形时用到的主要是波形窗口中的右键菜单，常用的编辑功能都集中在了 Wave 中，包含的功能如下：

(1) Create/Modify Waveform(创建/编辑波形)；
(2) Map To Design Signal(映射到设计文件)；
(3) Insert Pulse(插入脉冲)；
(4) Delete Edge(删除边沿)；
(5) Invert/Mirror(翻转/镜像)；
(6) Value(值)；
(7) Stretch Edge(扩展边沿)；
(8) Move Edge(移动边沿)；
(9) Extend All Waves(扩展全部波形)；
(10) Change Drive Type(改变驱动类型)。

3. 导出激励文件并使用

编辑好的波形文件可以导出成其他格式的文件保存，留待以后使用，可以使用命令行形式导出。使用菜单栏导出激励需要选择 Wave→Wave Edit→Export Waveform，或者选择 File→Export→Waveform，会弹出保存对话框。

5.2.2 采用描述语言生成激励

生成激励文件的另一种方式就是使用描述语言。可以使用的语言有很多，C、SystemC、Verilog、SystemVerilog 和 VHDL 都可以作为激励文件的编写语言。这里以 Verilog 语言为例，利用 ModelSim 自带的语言模板生成激励文件。语言模板支持 VHDL、Verilog、SystemC 和 SystemVerilog，在打开文件的时候 ModelSim 会自动识别该文件的类型并调用该语言的模板，如果不能确定使用的语言，ModelSim 就会把所有的语言模板以树形结构列出供选择。

激励文件也称为 testbench 文件，其与可综合 Verilog 代码所不同的是，testbench Verilog 是在计算机主机上的仿真器中执行的。testbench Verilog 的许多构造与 C 语言相似，在代码中包括复杂的语言结构和顺序语句的算法。

1. always 块和 initial 块

两种进程语句：always 块和 initial 块。always 块内的进程语句可用来模拟抽象的电路。出于模拟的目的，always 块可以包括：用以指定与不同结构之间的传播延迟等同的时序结构；或等待指定事件的时序结构。敏感列表有时可忽略。例如，用下面的代码片段来模拟时钟信号，该信号每 20 个时间单位在 0～1 间变换一次，且永远执行下去。

```
always
begin
  clk = 1;
  #20;
  clk = 0;
  #20);
end
```

initial 块内也有进程语句，但是仅在仿真之初被执行。其简单语法如下：

```
initial
begin
  进程语句;
end
```

initial 块常用于设置变量的初始值。注意，initial 块不可被综合。

2. 进程语句

进程语句应用于 initial 块、always 块、function 和 task 之中。最常用的进程语句为阻塞赋值、非阻塞赋值、if 表达式、case 表达式、循环表达式等。Verilog 支持的循环结构有 for、while、repeat 和 forever。

(1) for 循环的简单语法为：

```
for([initial_assignment]; [end_condition]; [step_assignment])
begin
  [procedural_statements;]
end
```

注意：当循环体内只有一条语句时，begin 和 end 限定词可以略去。

(2) while 循环的简单语法如下：

```
while([end_condition])
begin
  [procedural_statements;]
end
```

循环体内的语句连续重复执行，直到达到指定的终止条件([end_condition])为止。例如，上面的清理寄存器文件的操作可以使用 while 循环来描述。

(3) repeat 循环的简单语法如下：

```
repeat([number])
begin
  [procedural_statements;]
end
```

循环体内的语句被重复执行指定次数，该次数可通过[number]来指定。

(4) forever 循环，正如其名，重复执行其主体直至仿真结束位置。循环体内常包括一定的时序控制结构，以致周期性推迟执行。例如换一种方式来描述时钟信号，该信号每 10 个时间单位翻转一次，且永

远运行下去。

```
initial
begin
  clk = 1'b0;
  forever
    #10 clk = ~clk;
end
```

3. 时序控制

在 testbench 中,必须指定不同信号有效和无效或等待某事件或条件的时间。有三种时序控制结构:

- 时延控制:#[delay_time]
- 事件控制:@([event], [event], …)
- 等待语句:wait([boolean_expression])

此外还有一个编译器指令——'timescale,也与时序规范有关。

(1) 时延控制

时延控制使用#符号来指示,其后为延迟的时间数值。

如果时延控制放置在左手边,那么整条语句的执行都会被延迟。例如:

```
#10 a = 1'b0;
#5 y = a|b;
```

假设当前时间为 t,上面的语句表示:a 于 t+10 时刻得到 0 值;又过了 5 个时间单位后(即于 t+15 时刻)a|b 表达式被计算,其结果被赋给 y。

如果时延控制被放置在右手边,则表达式将会被立即运算,但是结果延迟后再赋给左手边。例如:

```
#10 a = 1'b0;
    y = #5 a|b;
```

表示 a 于 t+10 时刻得到 0 值;a|b 表达式被立即运算(即在 t+10 时刻),但其结果却在 t+15 时刻才赋给 y。

一般情况下,使用时延控制生成激励的方式来替代传播延迟的模拟。下面的格式使得代码显得更加直观。

```
a = 1'b0;                    //将 a 设定为 0
#10;                         //延时 10 个时间单位
a - 1'b1;                    //a 变为 1
#5;                          //延时 5 个时间单位
a = 1'b0;                    //a 变为 0
#20                          //延时 20 个时间单位
```

(2) 事件控制

事件控制使用@符号来指示,其后为敏感列表,用于指定所需事件。其使用方法与 always 块内的事件类似。事件,即敏感列表中的信号改变其值(信号跳变)的时刻。可加入 posedge 和 negedge 关键字以指定所需的跳变边沿(上升沿和下降沿)。在 testbench 中,直到指定事件发生,语句才可跳过延迟继续执行。事件控制的一个常见应用为:使用时钟信号来同步激励的生成。例如,下面的代码片段中,en 信号被激活持续一个时钟周期。

```
localparam delta = 1;
```

```
@(posedge clk);              //等待时钟上升沿
#delta;                      //持续 delta 个时间单位
en = 1'b1;                   //使 en 等于 1
@(posedge clk);              //等待下一个时钟上升沿
#delta;                      //持续 delta 个时间单位
en = 1'b0;                   //使 en 等于 0
```

换一种方式，可以在时钟信号的下降沿有效或解除使能 en。

```
@(negedge clk)               //等待时钟下降沿
en = 1'b1;                   //使 en 等于 1
@(negedge clk)               //等待下一个时钟下降沿
en = 1'b0;
```

(3) 等待语句

wait 语句用以等待指定条件。其简单语法如下：

```
wait[boolean_expression]
```

直到[boolean_expression]被计算为真，后面语句才可跳过延迟继续执行。可以使用 wait 语句来延迟执行，例如，等计数器数到 15 才激活某信号：

```
wait(counter == 4'b1111);    //等待计数器数到 15
…                            //继续
```

wait 语句有时很像事件控制，后者是等待某信号的跳变边沿，而前者是等待指定条件，有时可理解为电平敏感。

4. timescale 指令

编译器指令用以控制编译和预处理 verilog 代码，通过重音符号(')来指明。重音符号常位于键盘的左上角。与时间有关的指令是'timescale 指令：

```
'timescale [time_unit] / [time_precision]
```

time_unit 指定计时和延时的测量单位，time_precision 指定仿真器的精度。例如：

```
'timescale10ns/1ns
```

说明仿真单位为 10ns，精度为 1ns。当指定如下代码中的延时，即

```
#5 y = a & b;
```

表明实际上的延时为 50ns(即 5×10ns)。

也可以指定小数形式的单位延时。例如：

```
#5.12345 y = a & b;
```

说明实际延时为 51.2345ns。因为精度是 1ns，所以在仿真中就取整为 51ns。精度越小，仿真的准确性越高，但是会减慢仿真的速度。

time_unit 和 time_precision 的数字部分可以为 1、10 和 100，时间单位可以是 s(秒)、ms(毫秒)、μs(微秒)、ns(纳秒)和 ps(皮秒)。

5. 系统控制函数和任务

Verilog 有一组预定义的系统函数，以 $ 开头，执行与系统相关的操作，如仿真控制、文件读取等。下面为一些常用的函数和任务。

(1) 数据类型转换函数。$unsigned 和 $signed 函数执行无符号数类型和有符号数类型之间的

转换。

(2) 仿真时间函数。仿真时间函数返回当前的仿真时间,如 $time、$stime 和 $realtime 函数分别以 64 位整数、32 位整数和实数的形式返回时间。

(3) 仿真控制任务。有两种仿真控制任务:$finish 和 $stop。其中,$finish 任务用于终止仿真并跳出仿真器;$stop 任务用于中止仿真。在 ModelSim 中,$stop 任务则是返回到交互模式。在开发流程中,有时会停在 ModelSim 环境中,来进一步编辑或测试波形,因此代码中使用的是 $stop 任务。

(4) 显示任务。在 ModelSim 中,仿真的结果可以以波形的形式显示,也可以以文本的形式显示。四种主要的显示任务有 $display、$write、$strobe 和 $monitor,语法类似。在 ModelSim 中,文本是在控制面板显示的。

$display 任务的语法与 C 语言中的打印函数类似。其简单语法为:

```
$display([format_string], [argument], [argument], …);
```

例如:

```
$display("at %d; signal x = %b", $time, x);
```

其结果的形式如下:

```
at 5100; signal x = 00110001
```

最常用的转移符号有%d、%b、%o、%h、%c、%s 和%g,分别对应十进制、二进制、八进制、十六进制、字符、字符串和实数。

$write 任务几乎和 $display 任务等同,除了其执行之后并不跳到下一行显示,而是一直显示在当前位置。显示下一行字符\n,必须手动添加,以创建一个行中断。

Verilog 可结合 time step 的概念来塑造仿真延时。每个 time step 中可以发生很多活动。$strobe 任务与 $display 任务类似。代替立即执行的,$strobe 任务是在当前仿真的 time step 的结尾执行的。其可以规避由于竞争冒险造成的不匹配的数据显示。

$monitor 任务是非常通用的命令。鉴于 $display 任务、$write 任务、$strobe 任务是在一旦被执行的情况下才显示文本,$monitor 任务则是当其参数发生变化时即显示文本。$monitor 任务提供了简单的富有弹性的方式来跟踪仿真结果。

6. 文件 I/O 系统函数和任务

Verilog 提供一组用于访问外部数据文件的函数和任务。文件可以通过 $fopen 函数和 $fclose 函数来打开和关闭。$fopen 函数的语法为:

```
[mcd_names] = $fopen("[file_name]");
```

$fopen 函数返回一个与文件相关的 32 位的多通道描述子。这个描述子是一个 32 位的标志,代表一个文件(即一个通道)。最低位 LSB 保留,只是用以标准输出(console)。当使用该函数调用的文件被成功打开,则返回的描述子的值的某位会被置 1。例如,0…0010 表示打开第一个文件,0…0100 表示打开第二个文件,以此类推。若函数的返回值为 0,则表示文件未能被成功打开。

一旦某个文件被打开,可以向其内写入数据。可用的四种显示系统任务为 $fdisplay、$fwrite、$fstrobe 和 $fmonitor。这些任务的用法类似于先前的 $display 函数等,除了其第一个参数为描述子以外。

```
$fdisplay([mcd_name], [format_string], …);
```

下面给出一个简单的代码片段：

```
integer log_file, both_file;
localparam con_file = 32'h0000_0001;          //控制台
initialbegin
      log_file =  $fopenZ("my_log");
      if(log_file == 0)
            $display("Fail to open log file"); //写控制台
      both_file = log_file | con_file;

      //写入控制台和日志文件
       $fdisplay(both_file,"Simulation started");
      …
      //仅写入日志文件
       $fdisplay(log_file, … );
      …
      //写入
       $fdisplay(both_file,"Simulation ended");
       $fclose(log_file);
end
```

注意通过对多个描述子进行位运算来创建一个描述子，如 both_file 变量。当 both_file 被使用时，就可以同时对 console 和 log_file 进行操作。

有两个任务可以从文件中载入数据，分别为 $readmemb 和 $readmemh。这些任务假设外置文件中存储了 memory-array 的内容，然后读出这些内容存到一个变量中。$readmemb 和 $readmemh 所假设的文件格式分别为二进制和十六进制，语法格式为：

```
$readmemb("[file_name]", [mem_variable]);
$readmemh("{file_name]", [mem_variable]);
```

编写或生成激励并对待测设计单元进行仿真后，可以对仿真的结果进行分析，并根据该结果调试设计单元。在 ModelSim 中可以利用波形窗口和列表窗口对仿真波形进行分析，这两个窗口是同一信号的不同表示形式，波形相对来说更加直观。

5.3　VHDL 仿真

VHDL 的仿真过程一般可以分为四步：第一步，编译 VHDL 代码到库文件；第二步，采用 ModelSim 优化设计单元；第三步，装载设计单元；第四步，运行仿真并进行调试。

5.3.1　VHDL 文件编译

在 ModelSim 中的编译可以由两种方式进行：第一种是采用建立工程的方式，在建立工程的时候会自动地生成一个设计库，使用者可以对该库命名；第二种是直接新建库的方式，不建立工程，所有的文件编译和仿真等步骤都在库标签中进行。这两种方式无论采用哪种都是可以进行编译、优化和仿真的。本节采用建立工程的方式进行 VHDL 文件的编译，因为新建工程中实际包含了对库的操作，这样可以介绍得更加全面。

例如，建立一个工程名为 vhd_t 的工程，指定库的名称为 work，建立工程后，向工程中导入已有的 VHDL 文件，文中实例代码是 DES 加密算法的 VHDL 描述模块，实现了 64 位的 DES 加密算法。假设导入两个 VHDL 文件：testbench. vhd 和 freedes. vhd，一个设计文件，一个激励文件，在导入的时候

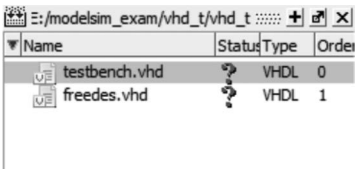


图 5-30　导入的 VHDL 文件

自动排序，激励文件的 Order 值为 0，设计文件的 Order 值为 1，如图 5-30 所示。一般来说，设计文件最好在激励文件前导入，否则编译可能会出现一些问题，可以手动调整一下 Order 值，将仿真顺序按设计文件、激励文件进行排序。

添加文件后，在 Project 标签内用右键菜单就可以直接编译所有文件，编译后的文件会自动生成到指定的库中，在本章中所有的文件都会被编译到名为 work 的库中。同样的操作可以使用命令行形式来完成。例如，想把设计文件编译到库中，就可以使用如下命令：vcom testbench. vhd freedes. vhd。

由于建立了工程，工程中所有的文件在缺省编译的情况下都会被编译到工程默认的库中，也省去了命令行中的很多参数项，只需要直接在 vcom 后加上文件名即可，多个文件名之间需要用空格隔开。例如，输入 vcom tester. vhd test circuit. vhd xcvr. vhd，可以看到命令窗口中出现的提示信息如图 5-31 所示。

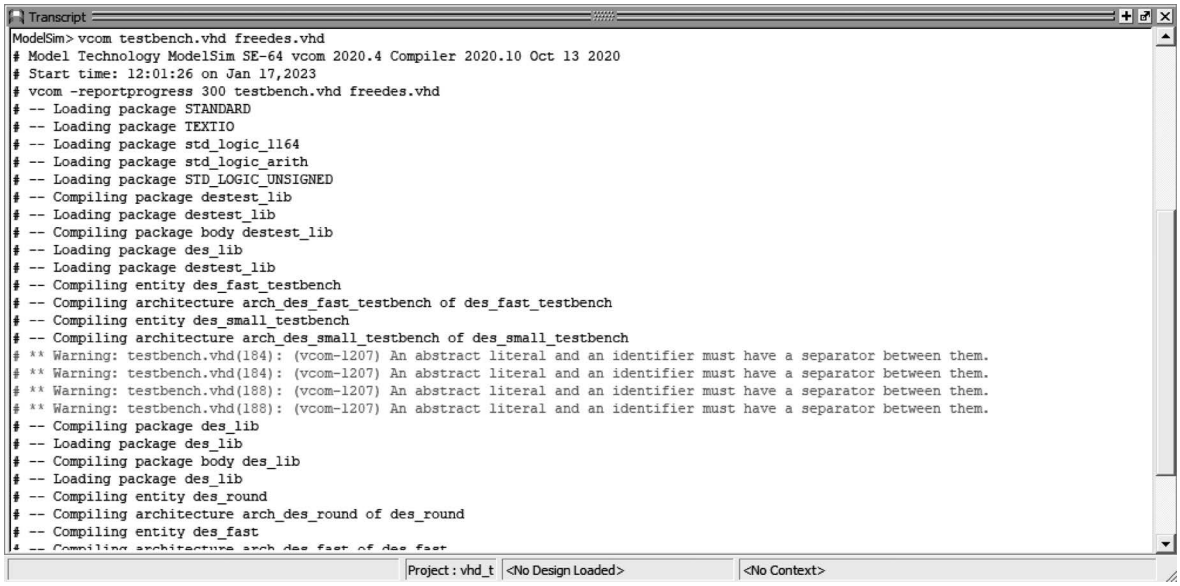


图 5-31　命令窗口提示信息

从提示信息中可以看到详细的编译过程，并且可以看到该 VHDL 文件中的所有实体(Entity)和结构体(Architecture)。了解 VHDL 的使用者都应该清楚，实体和结构体是一个 VHDL 文件必不可少的两部分，所以对于每个设计都是按先实体再结构体的顺序进行编译的。编译通过后，在 Library 标签内的 work 库中就可以看到新添加的单元，如图 5-32 所示。从库中可以看到，freedes. vhd 文件包含了三个实体和一个包，testbench. vhd 文件包含了两个实体和一个包。

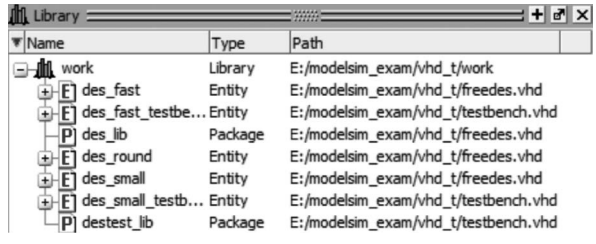


图 5-32　库中添加的单元

添加到库文件中的设计单元就可以在库标签中使用右键菜单进行仿真了，不过在工程标签中，由于采用的命令是 vcom，只是把文件编译到了库中。对于工程来说，这两个元器件还是未编译的状态，可以在命令行中输入 project compile all 来

编译所有文件。

在编译 VHDL 文件的时候，需要注意的一点就是语言版本。在 ModelSim 中提供了四种 IEEE VHDL1076 标准：VHDL-1987、VHDL-1993、VHDL-2002 和 VHDL-2008。ModelSim 默认的语言版本是 VHDL-2002。如果使用者使用了 VHDL-1987 或 VHDL-1993 版本的语言来书写代码，就需要更新代码或者为这些代码指定较早的语言版本，不过一般使用 VHDL-2008 语法中的关键字，就一定要把编译标准调整至 VHDL-2008，否则一定会报错的。

使用命令行操作可以快捷地进行版本的切换，不需要通过查看文件的属性再修改语言版本。在编译命令 vcom design_name 中间添加版本选择，例如，要按 VHDL-1987 标准进行编译，需要采用命令 vcom-87 design_name。命令中的-87 是版本号，如果需要更改不同的语言版本，就把-87 换为对应的版本代号即可，无版本号则是默认状态，即采用 VHDL-2002 标准。版本代号有四种：-87 表示 VHDL-1987 版本；-93 表示 VHDL-1993 版本；-2002 表示 VHDL-2002 版本；-2008 表示 VHDL-2008 版本。

编译的语言版本和代码的语言版本不同可能会在仿真中带来问题，下面简单说明一下各种版本之间可能出现的问题。如果使用到了混合版本，则应尽量避免下列问题：

(1) VITAL 库。如果使用 VITAL-2000 则必须采用 VHDL-1993 或 VHDL-2002 来编译；使用 VITAL-1995 则必须用 VHDL-1987 来编译，否则 ModelSim 就会报告错误信息。例如，一个典型的编译错误信息是 VITALPathDelay DefaultDelay parameter must be locally static，这表明 VITAL 需要在 VHDL-1987 版本下编译。

(2) 文件。文件的语法和用法在 VHDL-1987 版本和 VHDL-1993 版本间发生了变化。这可能会引起 ModelSim 的报警，输出信息 Using 1076-1987 syntax for file declaration。此外，即使文件声明通过，也会出现警告信息 Subprogram parameter name is declared using VHDL 1987 syntax。

(3) Package。每个 Package 的头和主体应该是同样的语言版本，如果采用了不同版本，ModelSim 就会提示混合包编译错误。

(4) Xnor。Xnor 是 VHDL-1993 的保留字。如果使用者使用 VHDL-1987 版本声明了一个 Xnor 函数，并在默认版本下编译该函数，就会出现错误信息 near“xnor”：expecting：STRING IDENTIFIER。

实际中不同版本带来的错误影响远不止上述几种，这里只是举出几种例子，应当尽量避免使用不同的语言版本，以免引入麻烦。

5.3.2　VHDL 设计优化

ModelSim 对 VHDL 语言均可以进行优化。VHDL 的优化可以有两种方式：第一种方式是通过菜单栏；第二种方式是通过命令行。下面分别介绍这两种方式。

在菜单栏中选择 Simulate→Design Optimization 即可启动设计优化，如图 5-33 所示。

单击后会出现如图 5-34 所示的优化窗口，可以看到该窗口有 5 个选项卡，分别是 Design、Libraries、Visibility、Options 和 Coverage。

在打开的时候默认选择是第一个选项卡，即 Design 选项卡。从图中可以看到，该选项卡中共有四部分的内容：第一部分是图中间的部分，这里面包含了所有的库文件，库文件中包含所有通过编译或添加到库中的设计单元，在这些设计单元中选择要优化的设计；第二部分是 Design Unit(s)(设计单元)，在设计单元中选中的设计名称会显示在这个区域中；第三部分是 Output Design Name(输出设计名称)，在这里可以为本次优化的设计指定一个输出名称，这个名称是由使用者定义的，如选中设计单元

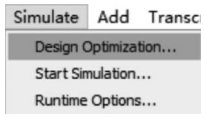


图 5-33　启动设计优化

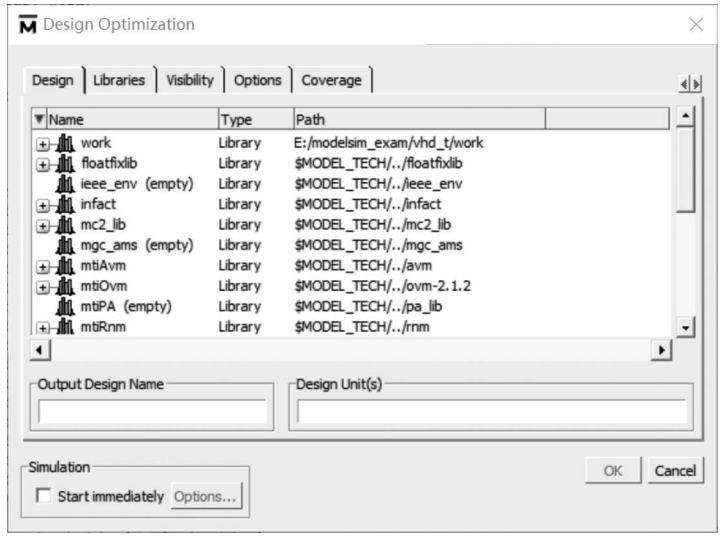


图 5-34　设计优化窗口

top,指定输出设计名称为 opt_top,优化后的输出名称就会是 opt_top;第四部分是 Simulation 选项,选中 Start immediately 后,就可以在设计优化后直接启动仿真。

第二个选项卡是 Libraries 标签,如图 5-35 所示。这里可以设置搜索库,可以指定一个库来搜索实例化的 VHDL 设计单元。Search Libraries 和 Search Libraries First 的功能基本一致,唯一不同的是,Search Libraries First 中指定的库会被指定在用户库之前被搜索。注意名称后面的-L 和-Lf,这些都是优化指令 vopt 与设计名称 designname 之间的参数。

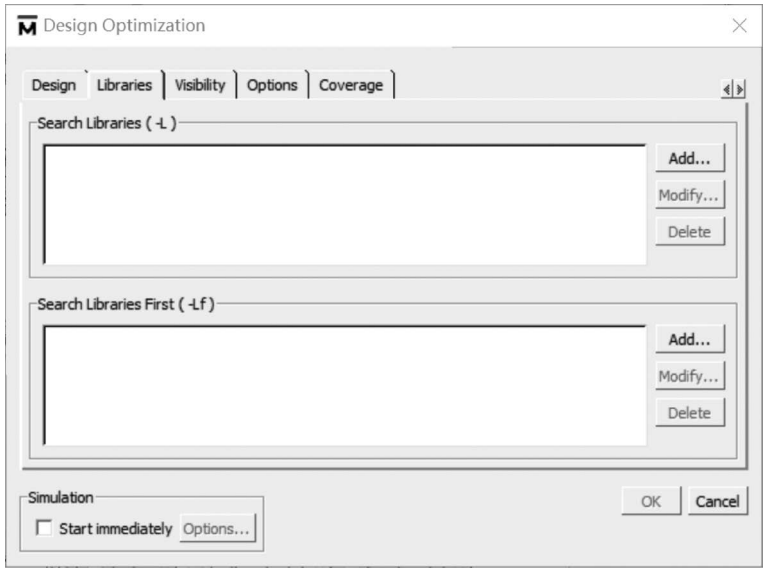


图 5-35　Libraries 标签

第三个选项卡是 Visibility(可见度)标签,如图 5-36 所示。可以通过对该选项卡进行设定来选择性地激活对设计文件的访问,可以使用此功能来保护设计文件。可提供的三个选项,第一个选项是 No design object visibility,即没有设计对象是可见的,选择此选项后优化命令适用于所有可能的优化并且

不关心调试的透明度。许多的 nets、ports 和 registers 在用户界面和其他各种图形界面中是不可见的。此外，许多这类对象没有 PLI 的访问句柄，潜在地影响 PLI 的应用。

第二个选项是 Apply full visibility to all modules(full debug mode)，这个选项正好与第一个选项相反，会访问所有设计对象，但是这可能会大大降低仿真器的性能。

可见度标签中的第三个选项是 Customized visibility，即自定义可见度。选中此选项后单击右侧的 Add 按钮，可弹出如图 5-37 所示的窗口，在此窗口中可以设置所有的库和设计单元的可见度。在 Selected Module(s)中可以输入一个或多个想要添加访问标志的模块，可以手动地输入模块名称或者在上方窗口的库中选择需要访问的模块，注意只有 Module 格式的才能被选择，即库中的图标是一个大写的 M 形式的才是可选的模块，库中的 Package 或 Entity 等格式都不可以被选中。如果想添加全部的模块，只需勾选后面的 Apply to all modules 选项即可。在 Selected Module(s)下方的 Recursive(递归)选项的作用是为选中模块的子区域或子模块添加标志。最下方的一个部分是 Access Visibility Specifications(访问可见度说明)。

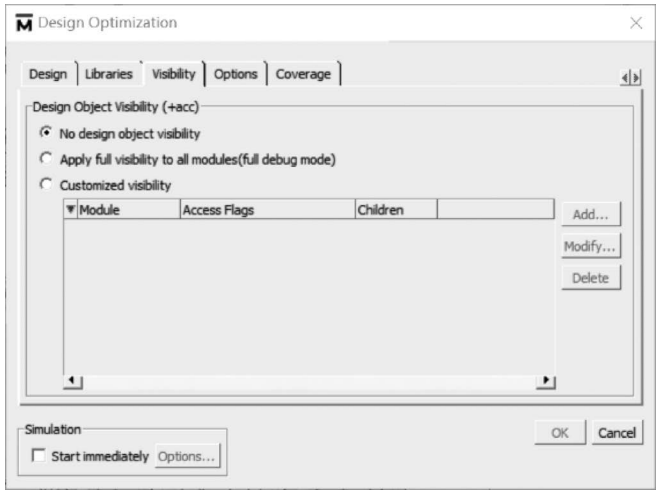


图 5-36　Visibility 标签

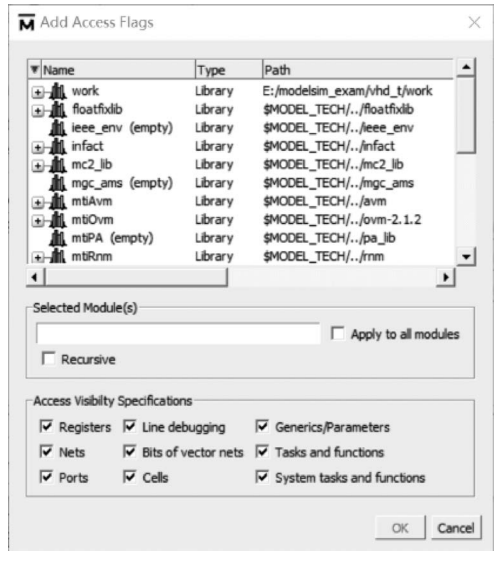


图 5-37　设置可见度

设计优化中的第四个标签是 Options 标签，如图 5-38 所示。顾名思义，这里有很多的选项设置，按功能的不同划分成了不同的区域。Optimization Level 区域用来指定设计的优化等级，这个选项只对 VHDL 和 SystemC 设计有效，可以根据需要指定禁用优化、启用部分优化、最大优化等。Optimized Code Generation 区域用来指定优化代码的产生，其中 EnableHazard Checking 用来启用冒险检测，这是针对 Verilog 模块的；Keep delta delays 用来保持 delta 延迟，即在优化时不去除 delta 延迟，这是针对 Verilog 编译器的；Disable Timing Checks in Specify Blocks 是用来禁止在指定的块中进行时序检测任务，这也是针对 Verilog 编译器的。Verilog Delay Select 区域用来选择延迟，默认为 default 状态，即典型状态，可提供三种不同的延迟，即最小延迟、典型延迟和最大延迟，根据此处的选择来调用器件库中元器件的延迟值。Command Files 区域用来添加命令文件，其文件格式应该是 text 格式，内部包含命令参数.可以单击右侧的 Add 按钮进行添加，单击 Modify 和 Delete 按钮进行修改和删除。Other Vopt Options 区域可以附加 vopt 命令，即手动输入 ModelSim 可识别的优化指令，用来实现在 Design Optimization 窗口中没有定义的选项。

图 5-38　Options 标签

设计优化窗口的主要选项都已介绍完毕，在使用中可以根据不同情况来选择适当的设置。在最简单的情况下，只需要选中 Design 标签中的设计单元，指定输出设计单元的名称，就可以添加一个设计优化文件。直接在菜单栏中进行设计优化和在 Project 标签中添加设计优化文件有所不同。

5.3.3　VHDL 设计仿真

编译成功并建立需要的优化后，就可以对被编译的文件进行仿真了。仿真开始的方式有很多，可以选择快捷工具栏中的仿真按钮开始仿真；可以通过菜单栏选择 Simulate→Start Simulation 开始仿真；可以通过命令行形式输入 vsim 指令开始仿真，以上的三种方式都会弹出 Start Simulation 窗口，如图 5-39 所示。同样是多标签选项的形式，共有 Design、VHDL、Verilog、Libraries、SDF 和 Others 6 个标签，每个标签中提供不同的选项设置。

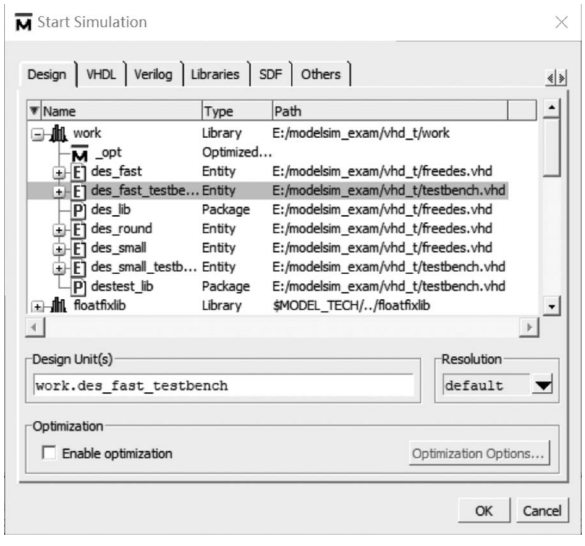


图 5-39　开始仿真窗口

首先介绍 Design 标签。该标签内居中的部分是 ModelSim 中包含的全部库，可展开看到库中包含的设计单元，这些库和单元是为仿真提供选择的，可以选择需要进行仿真的设计单元开始仿真，被选中的仿真单元的名称会出现在下方的 Design Unit(s)位置。ModelSim 支持同时对多个文件进行仿真，可以利用 Ctrl+Shift 键来选择多个文件，被选中的全部文件名都会出现在 Design Unit(s)区域，本节选择 work 中的 des_fast_testbench 进

行仿真。在 Design Unit(s)区域的右侧是 Resolution 选项，这里可以选择仿真的时间刻度(时间单位)。时间刻度的概念类似于长度度量单位的米，在 ModelSim 进行仿真的时候，有一个最小的时间单位，这个单位是可以指定的。如最小单位是 10ns，在仿真器工作的时候都是按 10ns 为单位进行仿真，对 10ns 单位以下发生的信号变化不予考虑或不予显示，当测试文档中有类似于“#1 a=1'b1;”的句子时，ModelSim 就不会考虑句中的延迟。Resolution 选项一般都是设置在默认的状态，这时会根据仿真器中指定的最小时间刻度来进行仿真，如果设计文件中没有指定，则按 1ns 来进行仿真。最下方的区域是 Optimization 区域，可以在仿真开始的时候激活优化。在 5.3.2 节优化窗口中也有选项，可以在优化设计后立刻开始仿真，两者功能是相同的。选中 Enable optimization 后，右侧的 Optimization Options 按钮会变为可选，单击就会弹出优化设置的相关选项，这里出现的优化设置选项的功能与 5.3.2 节中介绍的完全相同，只是没有了 Design 标签(因为在 Start Simulation 中已经指定了设计单元)。

第二个标签是 VHDL 标签，如图 5-40 所示。其中有 VITAL、TEXTIO Files 和 Other Options 三个区域。①VITAL 区域内有三个选项：Disable timing checks 的功能是对 VITAL 中生成的模块禁用时序检测；Use VITAL 2.2b SDF mapping 的功能是采用 VITAL 2.2b 库进行 SDF 文件的标注，默认情况下是采用 VITAL95 库；Disable glitch generation 的功能是禁用毛刺生成。②TEXTIO Files 区域可以选择输入或输出的 TEXTIO 文件。③Other Options 中有两个选项：Treat non-existent VHDL Files opened for read as empty 的功能是当需要打开一个库中不存在的 VHDL 文件时，把该文件作为一个空文件读入；Do not share file descriptors for VHDL files opened for write or append that have identical names 的功能是关闭文件描述符的共享，ModelSim 默认情况下向所有打开的 VHDL 分享文件描述符。

第三个标签是 Verilog 标签，如图 5-41 所示。图中的 Pulse Options 区域用来设置脉冲选项，可以选择 Disable pulse error and warning messages 来禁用路径脉冲的 error 和 warning 信息。下方的两个选项中：Rejection Limit(抑制限度)用来设置抑制限度，这个限度值采用路径延迟的百分比形式；Error Limit(误差限度)用来设置误差限度，也是采用路径延迟的百分比形式。Other Options 区域提供两个附加选项：Enable hazard checking 选中后可激活冒险检测；Disable timing checks in specify blocks 选

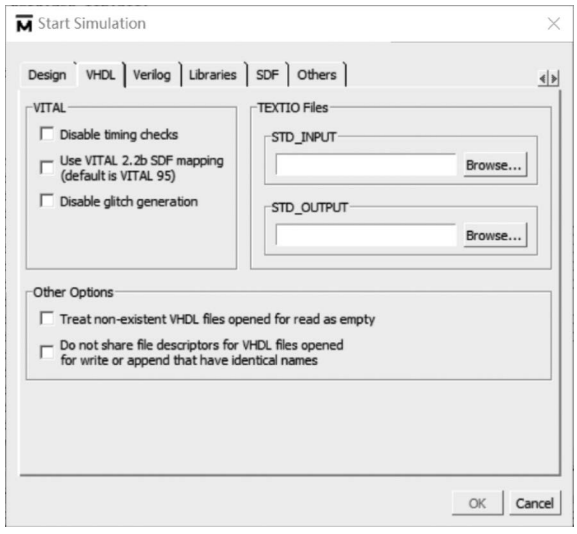


图 5-40　VHDL 标签

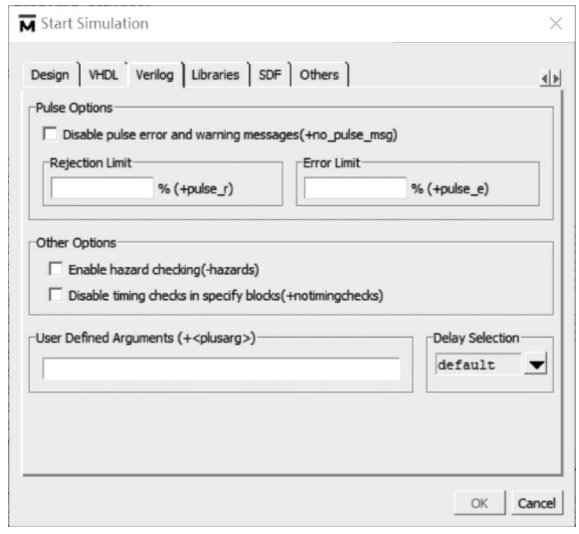


图 5-41　Verilog 标签

中后可以禁用指定块中的时序检测。User Defined Arguments 区域由使用者自行设定参数值，数值必须以“+”开头，否则 ModelSim 就会报错。Delay Selection 选项用来设置延迟信息，可以从下拉列表中选择 default(缺省延迟)、min(最小延迟)、type(典型延迟)或者 max(最大延迟)。

有关这三种(最大、最小和典型)延迟的解释在各类硬件语言书中都可以找到，这里只做一个简单的解释。在硬件描述语言中，会为各类基础器件建立一个器件模型，这个模型就要用到这三类延迟，这些延迟反映了集成电路制造工艺过程中带来的影响。真实的器件延迟总是在最大值和最小值之间变化，而典型值则是所有器件的一个平均期待水平。举例来说，一个最基本的与非门，假定由工艺决定该与非门的最小延迟是 0.1ns，最大延迟是 0.3ns，典型值是 0.15ns，就表明实际信号从输入到输出要经过 0.1～0.3ns 的延迟，而不是立即输出。0.15ns 的典型值表示在该工艺条件下生产一批该与非门，有一定百分比的器件延迟在 0.15ns 以下，这个百分比可由工艺厂商制定。在实际用 VHDL 描述该与非门时，就需要对应设定这三种延迟信息，以求最好地反映实际硬件电路。这些延迟会在时序仿真的时候使用到，而且这些值的大小会直接影响时序电路的工作状态，更加详细的内容可以查阅有关时序分析的书籍。

第四个标签是 Libraries 标签。这个标签的内容和功能与优化设计窗口中的 Libraries 标签完全一致，这里不再赘述。

第五个标签是 SDF 标签。其内容如图 5-42 所示。SDF 是 Standard Delay Format(标准延迟格式)的缩写，内部包含了各种延迟信息，也是用于时序仿真的重要文件。SDF Files 区域用来添加 SDF 文件，选择 Add 进行添加；选择 Modify 进行修改；选择 Delete 删除添加的文件。SDF Options 设置 SDF 文件的 warning(警告)和 error(错误)信息：第一个 Disable SDF warnings 是禁用 SDF 警告；第二个 Reduce SDF errors to warnings 是把所有的 SDF 错误信息转变成警告信息。Multi-Source delay 可以控制多个目标对同一端口的驱动，如果有多个控制信号同时控制同一个端口或互联，且每个信号的延迟值不同，可以设置此选项统一延迟，此功能下拉列表中可供选择的有三个选项，max、min 和 latest：max 即选择所有信号中延迟最大的值作为统一值；min 即选择所有信号中延迟最小的值作为统一值；latest 则是选择所有信号中最后的延迟作为统一值。

添加 SDF 文件的窗口如图 5-43 所示。SDF File 区域可以输入需要添加的 SDF 文件名，或单击 Browse

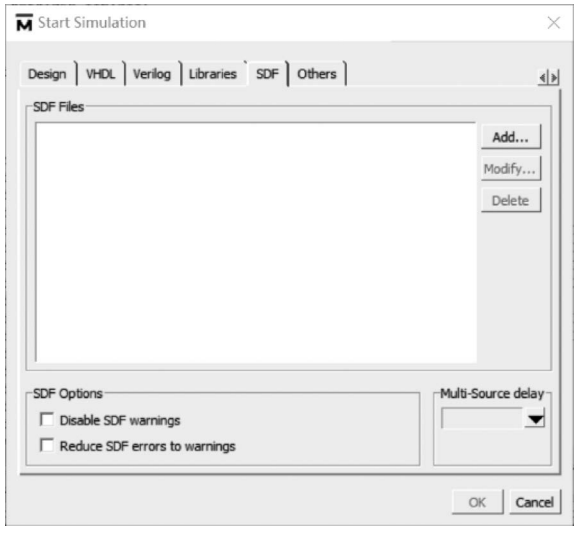


图 5-42　SDF 标签

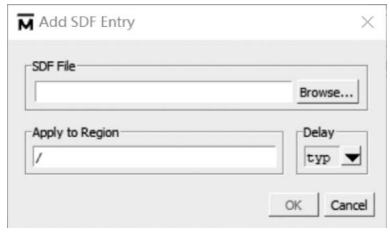


图 5-43　添加 SDF 文件

浏览选择。Apply to Region 用来指定一个设计区域，把 SDF 标签中的所有选项都应用到这个区域中。Delay 也是用来选择最小、最大、典型三种延迟类型的。

第六个标签是 Others 标签，这里包含一些杂项，如图 5-44 所示。Generics/Parameters 区域用来指定参数值，单击 Add 按钮会出现对话框，如图 5-45 所示：在对话框中 Name 区域输入参数的名称，在 Value 区域输入对应的数值，单击 OK 按钮后该参数就会出现在 Generics/Parameters 区域；对话框中 Override Instance-specific Values 是覆盖选项，选中此选项后，如果设计文件中有相同的参数就会被此处的参数值覆盖。Coverage 区域用来指定代码覆盖检测，选中此选项开始仿真，就会自动出现代码覆盖率检测的窗口。Profiler 区域用来激活内存分析。WLF File 区域用来指定 WLF 文件(波形文件)的存储路径和存储名称，默认的路径是工作路径，即 C:\modeltech64_2020.4\examples，默认的名称是 vsim.wlf。Assertions 区域用来设置断言选项，可以勾选或取消三个选项来启动或禁用 PSL、SVA 和断言调试窗口。断言文件的名称可在 Assert File 区域指定。和其他窗口一样，如果需要设置选项卡中没有的功能，可以在 Other Vsim Options 区域输入标准格式的 vsim 命令。

图 5-44　Others 标签

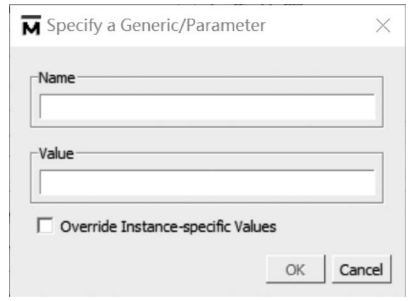


图 5-45　添加指定参数值

Start Simulation 窗口的选项基本如上所述，进行不同文件的仿真需要设置不同的选项，设置完毕后，单击 OK 按钮即可开始仿真。开始仿真后，多标签区域会增加很多新的标签。在仿真之前，多标签区域一般只有 Project 和 Library 两个标签。

开始仿真后，在多标签区域一般会增加 sim 标签、Files 标签和 Memories 标签。除了多标签区域会增加标签，在 MDI 窗口中也会新出现一个 Object 窗口，在多标签区域中的 sim 标签选中一个设计单元，在 Object 窗口中就会出现该单元包含的输入/输出端口，如图 5-46 所示。

另一种开始仿真的方式，是在库中选择需要仿真的单元，使用右键菜单中的 Simulate 或 Simulate with coverage 命令开始仿真，选中命令后会跳过 Start Simulation 窗口，直接出现 sim 标签和 Object 窗口。如果不需要进行选项设置，也可以采用这种方式快速地开始仿真。

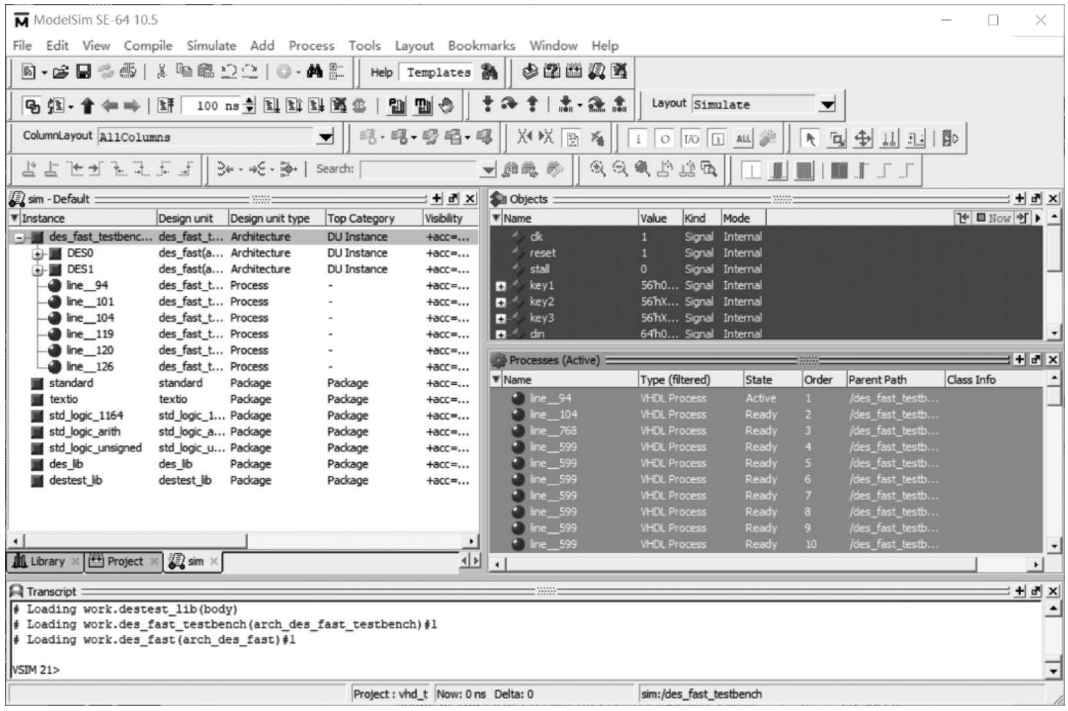


图 5-46　仿真开始后的工作区和对象窗口

在 sim 标签中可以看到本次仿真的模块 des_fast_testbench 内部调用实例化模块和内部包含的寄存器、线网等信息。选中 sim 标签中的模块，在右键菜单中选择 Add Wave 命令，可将选中模块的所有信号全部添加到波形窗口中；或者选择 Add to→Wave 命令，在三个选项中选择需要显示的信号即可。如果只需要选择模块中的一个或几个信号，可以在右侧的 Object 窗口中采用同样方式在右键菜单中选择 Add→Add to→Wave 命令添加选中的信号，如图 5-47 所示。

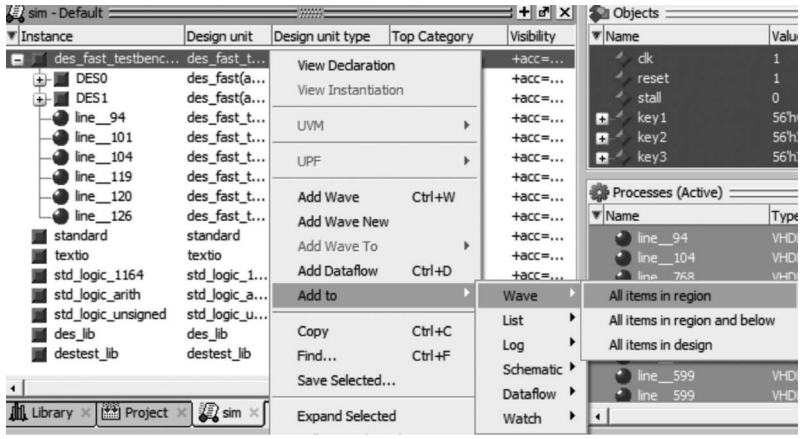


图 5-47　向波形窗口中添加信号

添加信号后，波形窗口会出现在 MDI 区域(在启动仿真时波形窗口并不出现)，如图 5-48 所示。但此时只是将信号添加到了波形窗口中，仿真并没有开始运行，所以在波形显示区域并没有输入/输出的信号波形。

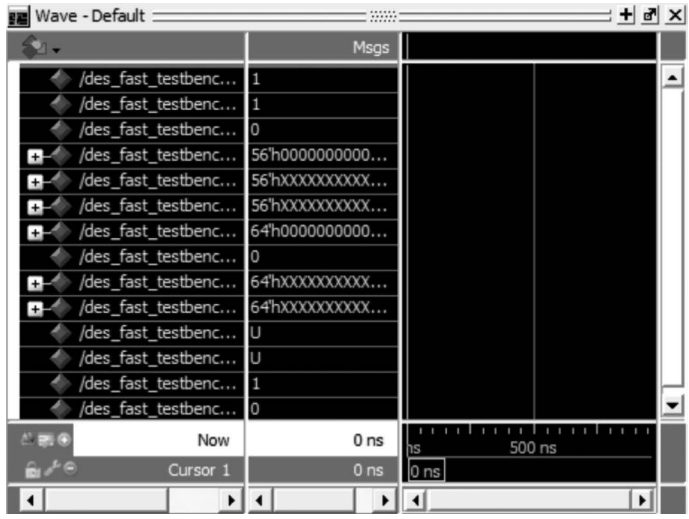


图 5-48　波形窗口显示

添加信号后，在命令窗口中输入 run -all 命令可以运行仿真。此条命令的功能是运行全部的仿真，即从测试平台中 0 时刻开始直至有系统任务或函数中断仿真为止。由于测试平台中没有使用具有停止和中断功能的系统函数，所以仿真会一直持续下去。但是仿真运行的时候波形是不会更新的，依然是一片空白。此时选择快捷工具栏中的中断仿真按钮或在菜单栏中选择 Simulate→Break 命令可以中断仿真。中断仿真后，在波形窗口中会出现刚才运行的全部仿真波形，如图 5-49 所示。

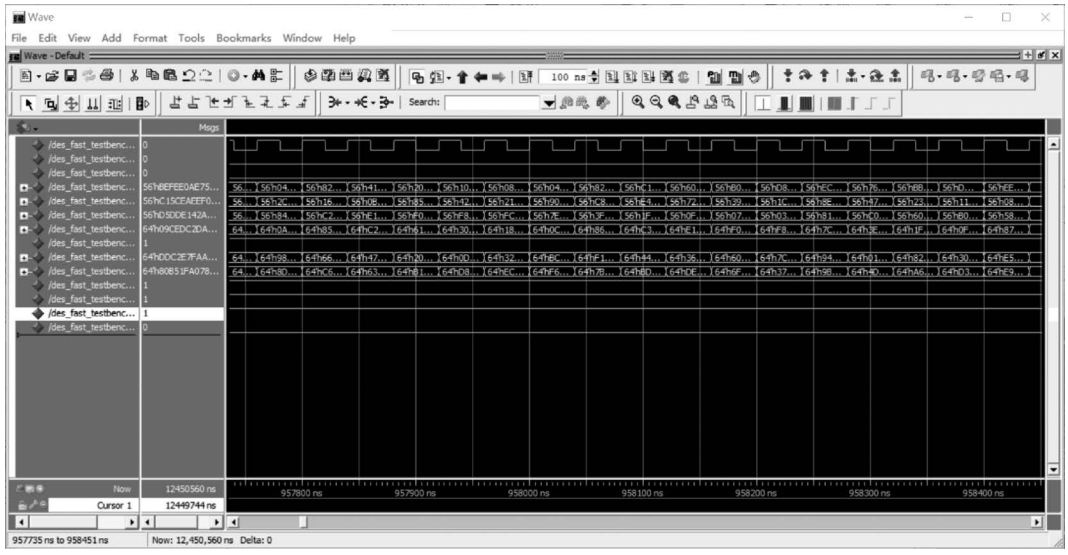


图 5-49　仿真波形窗口

5.4　Verilog 仿真

Verilog 和 VHDL 同属于硬件描述语言，故在编译、优化、仿真的过程中，所进行的操作也十分类似。本节介绍对于 Verilog 设计的仿真，与 VHDL 相同的部分会简单略过，重点在于不同的操作步骤。

本节中以一个32位浮点乘法器说明Verilog文件的仿真过程。Verilog文件的仿真过程与VHDL文件的仿真过程是基本相同的,只是语言上有各自的特点,所以仿真时用到的功能会有局部的区别。为了使Verilog文件的仿真实例不成为VHDL仿真实例的翻版,在VHDL文件仿真时只用到了波形窗口,且未作任何设置。在本节的Verilog文件仿真实例中不再使用波形窗口,使用列表窗口观察信号值,虽然两者的形式不同,但都是观察仿真结果的方式。

5.4.1 Verilog文件编译

同VHDL一样,Verilog文件的编译也有两种方式,即基于建立工程的仿真和基于不建立工程的仿真。仿真的过程也与VHDL相似,只有一点不同:VHDL中仿真(编译)的命令是vcom,而Verilog语言中编译的命令是vlog。例如,对Verilog文件的编译可采用如下形式:

```
vlog design_name
```

Verilog文件中有一类比较特殊,这就是SystemVerilog。SystemVerilog语言可以说是Verilog的一种发展,但与Verilog语言又有很大区别。对于这两种语言的区别和联系,可以查阅相关文献进行了解。在ModelSim的使用中,默认的语言标准是针对Verilog语言的,如果要编译SystemVerilog语言,则需要在选项中进行设置,或采用命令行参数的形式进行编译。设置SystemVerilog工程编译选项如图5-50所示。这个Project Compiler Settings是在建立工程的情况下,在Project标签内使用右键菜单,选择Compile→Compile Properties命令后出现。对于不建立工程的情况,可以在菜单栏中选择Compile→Compile Options,会出现一个名为CompilerOptions的窗口,名称不同,但是Verilog标签内的选项都是相同的,在Verilog标签中把语言版本从Default选为Use SystemVerilog,就可以对

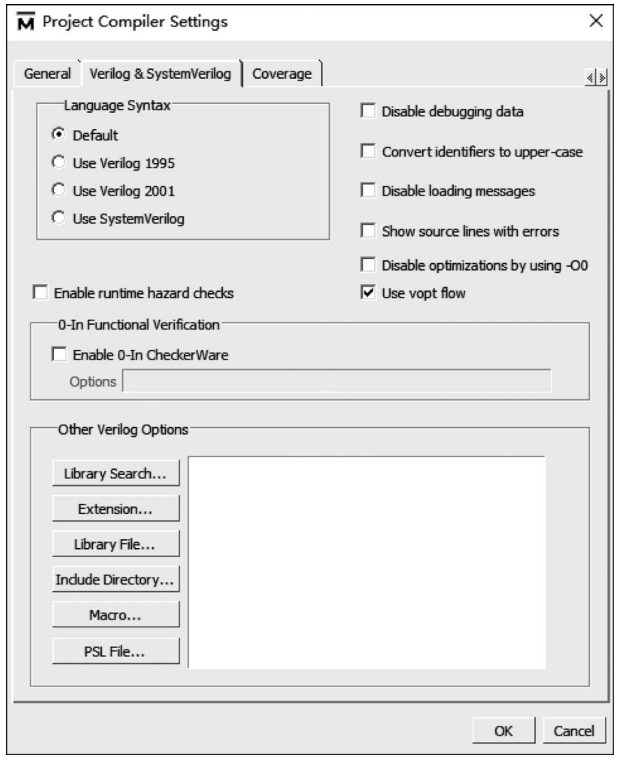


图5-50 激活SystemVerilog的编译

SystemVerilog 文件进行编译了。

采用命令行方式也可以对 SystemVerilog 文件进行编译。ModelSim 中提供了如下两种方式：

```
vlog design_name.sv
vlog -sv design_name.v
```

第一种方式中使用.sv 后缀的文件名，ModelSim 在编译的时候会根据文件名自动激活 SystemVerilog 语法标准。第二种方式中使用.v 后缀的文件名，此时需要添加命令参数把编译器设置为 SystemVerilog 语法标准，即采用“-sv”来设置命令参数。

ModelSim 对于 Verilog 的编译命令参数有一部分是和其他仿真软件相同的，这样就可以很方便地完成两种软件间的切换和衔接。

在本节仿真的过程依然采用建立工程的方式，建立工程的名称为 Verilog_t，默认的库依然指定为 work，这样所有编译的单元都会添加到 work 库中。将全部文件加入工程中进行编译，通过编译的工程界面如图 5-51 所示。

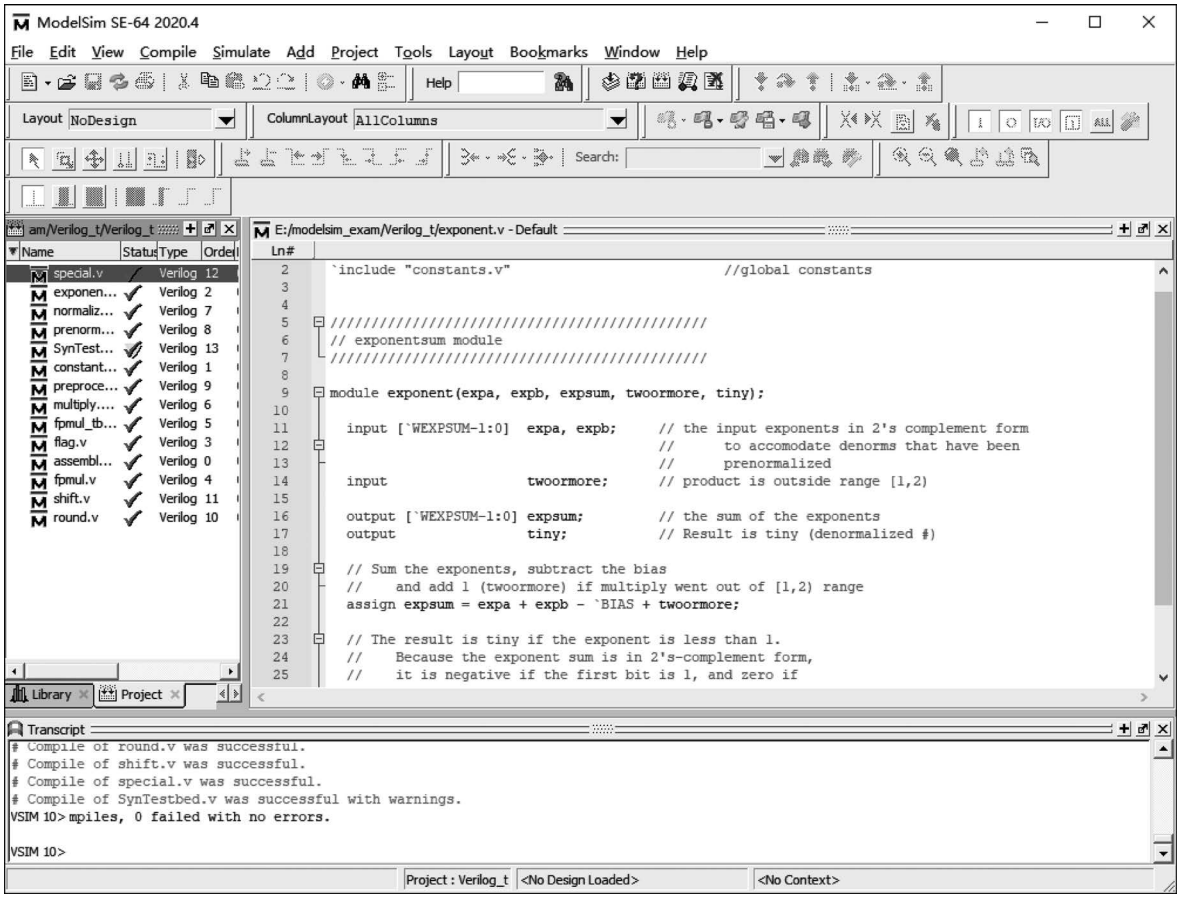


图 5-51　通过编译的工程界面

5.4.2　Verilog 设计优化

Verilog 语言的优化也有两种方式：第一种是通过菜单栏；第二种是通过命令行。通过菜单栏的方式与 VHDL 相同，也是在菜单栏中选择 Simulate→Design Optimization 命令来进行设置。具体的选项

设置和选项功能在 5.4.1 节中均已详细介绍，这里就不再重复了。采用命令行的方式与 VHDL 类似，也是使用 vopt 命令，命令格式如下：

```
vopt lib_name.unit_name -o output_name
```

同样也可以在编译时输入 vopt 命令，命令格式如下：

```
vlog -work lib_name -vopt file_name
```

与 VHDL 唯一的不同就在于指令是以 vlog 开头而不是 vcom。

5.4.3 Verilog 设计仿真

Verilog 文件的仿真和 VHDL 一样，可以选择快捷工具栏中的仿真按钮开始仿真；可以通过菜单栏选择 Simulate→Start Simulation 开始仿真；可以通过命令行形式输入 vsim 指令开始仿真。这三种方式都会弹出 Start Simulation 窗口，该窗口在 5.3.3 节中已经详细介绍，可以参考其中的内容。

在 5.3.3 节中，只是介绍了开始仿真的部分，对仿真的中间过程并没有介绍，包括仿真时间的设置、重新仿真、sim 标签中的快捷菜单等，此节将进行介绍。按照仿真进行的顺序，首先介绍 sim 标签中的快捷菜单。

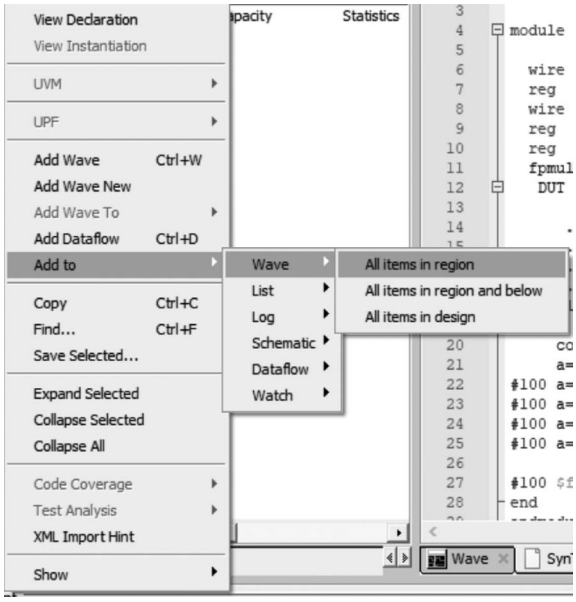


图 5-52 sim 标签中的右键快捷菜单

在 ModelSim 的任意位置右击都会出现快捷菜单，且不同区域出现的菜单不尽相同。在 sim 标签中的快捷菜单主要是为了仿真使用，其包含的命令如图 5-52 所示。

右键快捷菜单中包含的命令功能如下：

(1) View Declaration(显示声明语句)。选中 sim 标签中一个线网、寄存器或设计模块单元的时候，该选项变为可选选项，使用此命令可以查看被选中目标在源代码中第一次被声明的位置。如果源文件处于打开状态，则会直接跳转到声明语句所在的行；如果源文件未打开，则会打开该文件并显示声明语句。

(2) View Instantiation(显示实例化语句)。选中设计模块时变为可选选项，但是选中的设计模块不能是顶层模块。该命令的功能是显示被选中的模块在何处被实例化。由于除了模块之外的 Verilog 类型都没有被实例化，所以只有选中模块才能使用此命令。又因为顶层模块在此工程中不会被调用，所以顶层模块也不能使用此命令。

(3) Add Wave(添加到波形)。把选中的信号添加到波形窗口，如果选中的是一个模块，则把整个模块中包含的可见信息(如端口、局部信号等)都添加到波形窗口。

(4) Add Wave New(添加波形到新窗口)。把选中的信号添加到一个新的窗口中，一般仿真打开时，都会出现一个默认的波形窗口。而使用此命令，可以新建一个窗口并把信号添加进去。

(5) Add Wave To(添加波形到)。当出现多个波形窗口的时候此选项变为可选，可以把选中的信号指定添加到某一个波形窗口中。

(6) Add Dataflow(添加到数据流)。把所选的模块或信号添加到数据流窗口。

(7) Add to(添加到)。此为多选项菜单,可以把选中的信号或模块添加到 Wave(波形)窗口、List(列表)窗口、Log(日志)窗口、Schematic(原理图)窗口、Dataflow(数据流)窗口、Watch(观察)窗口。

(8) Copy(复制)。此命令的功能是复制选中信号的路径名称。前面的内容介绍过,添加信号的命令形式为 add wave sim:/module_name/single_name,在 sim 标签中复制选中的信号或模块,复制的就是 add wave 后需要添加的信号路径。

(9) Find(查找)。选中此命令会出现对话框。此框在 sim 标签页的下方,在 Find 区域输入想要查找的名称,在 Search For 区域用下拉菜单选择查找的类型,根据选中目标不同可以选择 Instance(实例)、Design Unit(设计单元)、Entity/Module(实体/模块)或 Architecture(结构体)。

(10) Expand/Collapse Selected(展开/合并所选)。模块中一般包含一些输入与输出信号,故模块可以有展开和合并两种形式,展开时可以显示内部的信号,合并时只是显示模块名。这两个命令可以展开或合并选中的模块。

(11) Collapse All(合并所有)。功能和 Expand/Collapse Selected 相似,只是此命令会合并 sim 标签中所有可展开的项。

(12) Code Coverage(代码覆盖)。只有在进行代码覆盖率仿真时此命令才是可选命令。该命令具有三个子命令:Code Coverage Report 命令的功能是输出代码覆盖率报告;Clear Code Coverage Data 命令的功能是清除已有代码覆盖率的数据;Enable Recursive Coverage Sums 命令的功能是显示每个设计对象或文件的覆盖率数据和图标,默认是已选的。

(13) Test Analysis(测试分析)。此选项在 UCDB 文件被选中时变为可选,可以选择一些覆盖率的分析情况。

(14) XML Import Hint(XML 导入提示)。显示 XML 的导入路径层次名称和行数等信息。

(15) Show(显示)。显示 sim 标签(Structure 窗口)中可以显示的信息,勾选的类型将被显示,如果要隐藏某些类型,可以取消选择。另外,还可以根据个人需要,在 Change Filter 选项中选择此子菜单要显示的选项。

在仿真标签使用右键菜单选择 Add to 命令,在 VHDL 实例中选择的是添加到波形窗口,本例中选择添加到列表,即选择 Add to→List。添加信号后的列表窗口如图 5-53 所示。由于没有进行仿真,所以初始时间在 0s,所有的数据都是未知状态。

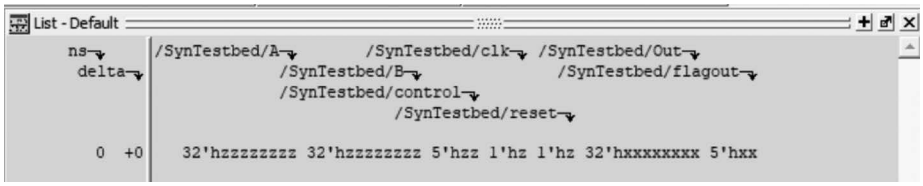


图 5-53　列表窗口

设计中采用'timescale 命令自定义了时间刻度,所以在仿真选项中不需要设置。仿真刻度设定是 1ns/10ps,即最小的刻度是 10ps,所以仿真时间会调整到 ps 级别。在 Runtime Options 窗口中设定的运行命令运行的是 100 个时间单位,这个时间单位是以仿真器给出的单位为准,在本例中就是运行 100ns。具体如图 5-54 所示,在输入数字的框中也可以输入单位。例如,在框中输入 100ns,执行 run 命令的时候仿真器会优先按照用户指定的时间运行,即运行 100ns。

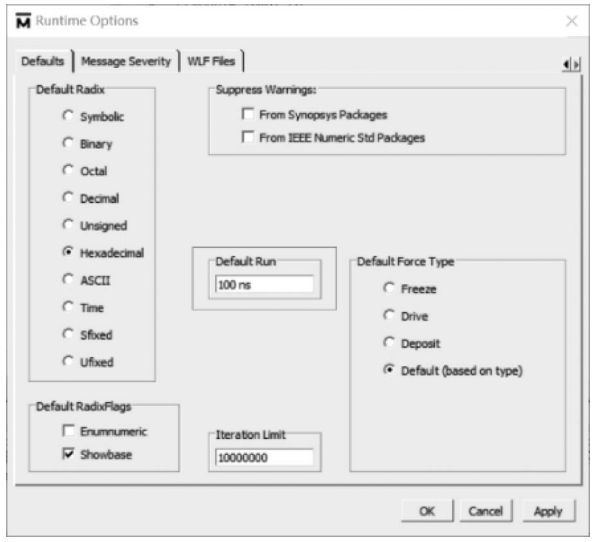


图 5-54　默认运行时间

在 ModelSim 的命令行中输入 run -all,执行此命令后,测试程序中所有测试向量都会被输入,列表窗口的数据会更新,如图 5-55 所示。

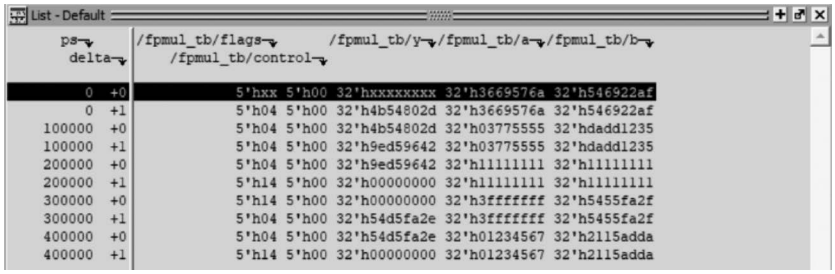


图 5-55　列表窗口数据

在列表窗口的顶端用箭头形式指示出了每个信号的路径和在列表窗口中的位置,这里的数据是以十六进制显示的,默认情况下是以 32 位二进制形式表示,可以在 Runtime Options 窗口中设置显示数据的基数。基数的设置最好在仿真开始的时候进行,因为列表窗口不同于波形窗口,波形窗口可以任意修改仿真波形的表示方式。

无论从十六进制修改成二进制还是从二进制修改成十六进制,都不会有任何问题。但是在列表窗口中,只能由多位向少位转变。以本例来说,在仿真最初将基数设置为十六进制,仿真后的数据如果改成二进制表示,在列表中就无法显示正常的数据,只显示一连串的“*”号。这时只能通过右键菜单中查看细节命令才能看到数据。但是如果初始设置为二进制,仿真后修改为十六进制,所有数据都能正常显示。这是一个显示上的问题,使用的时候需要留意。

在 VHDL 的仿真曾经提到,仿真时如果没有设置中断或停止命令,仿真会一直运行,直到使用 ModelSim 的 Break 命令,即使用中断仿真命令时才能停止仿真。但是采用这种中断的方式时,中断的时间和位置是不能确定的,如果需要在某一个确定的位置暂停仿真,就不可能采用这种形式,而是要采用设置中断点的方式。

中断点可以在源文件窗口中设置,图 5-56 中第 22 行就是设置好的中断点。中断点的添加很简单,

需要在哪一行添加中断点，就在哪一行的最前端单击，在该行的行号后方就会出现一个实心的红色圆点，表示该行已经被设置成中断点了。如果要取消中断点，只需再单击一次，红色实心点会变成灰色的空心点，表示该点取消。

设置中断点后，当仿真器运行到中断点时就会自动跳出仿真。例如，图 5-56 中第 22 行为测试模块添加了中断点，在命令行中输入 run -all 命令执行全部仿真，会得到以下的输出信息：

```
run -all
# Break in Module fpmul_tb at E:/modelsim_exam/v_t/fpuml_tb.v line 22
```

该信息提示，当前的仿真在第 22 行被中断。需要注意的是，中断点设置的代码行不会在仿真中执行，也就是说，在此例中第 22 行的代码是不被执行的。

如果需要在执行完全部测试向量后停止仿真，可以采用设置中断点来实现，也可以使用系统任务的方式来实现。常用的系统函数(命令)有 $stop 和 $finish。$stop 命令表示运行到此行停止仿真，与中断点功能相同。例如，在源文件的第 27 行中添加 $stop 命令(需要取消第 22 行的中断点，否则不会执行到第 27 行)，如图 5-57 所示。

```
18  initial
19  begin
20      control=5'b00000;
21      a=32'h3669_576a;b=32'h5469_22af;
22  #100 a=32'h0377_5555;b=32'hdadd_1235;
23  #100 a=32'h1111_1111;b=32'h1111_1111;
24  #100 a=32'h3fff_ffff;b=32'h5455_fa2f;
25  #100 a=32'h0123_4567;b=32'h2115_adda;
26
27  #100 $finish;
28  end
29  endmodule
```

图 5-56　中断点设置

```
18  initial
19  begin
20      control=5'b00000;
21      a=32'h3669_576a;b=32'h5469_22af;
22  #100 a=32'h0377_5555;b=32'hdadd_1235;
23  #100 a=32'h1111_1111;b=32'h1111_1111;
24  #100 a=32'h3fff_ffff;b=32'h5455_fa2f;
25  #100 a=32'h0123_4567;b=32'h2115_adda;
26
27  #100 $stop;
```

图 5-57　添加 stop 函数(命令)

此时运行 run -all 命令会在命令行中得到如下输出：

```
run -all
# ** Note: $stop          : E:/modelsim_exam/v_t/fpuml_tb.v(27)
# Time: 500 ns            Iteration: 0 Instance: /fpmul_tb
# Break in Module fpmul_tb at E:/modelsim_exam/v_t/fpuml_tb.v line 27
```

与中断点一样，在此行之前的所有测试向量都被执行。

另一个系统任务 $finish 并不常使用，因为使用此命令会关闭仿真器。例如，在程序中添加了 $finish 语句，执行到此条语句时仿真器会弹出对话框，询问是否关闭仿真，如图 5-58 所示。

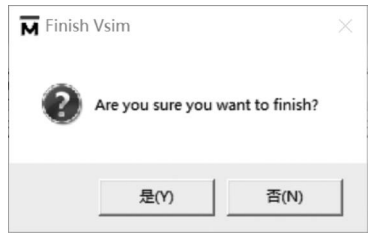


图 5-58　Finish 对话框

选择“否”会出现如下提示：

```
run -all
# * * Note: $finish       : E:/modelsim_exam/v_t/fpuml_tb.v(28)
# Time: 600 ns            Iteration: 0 Instance: /fpmul_tb
# 1
# Break in Module fpmul_tb at E:/modelsim_exam/v_t/fpuml_tb.v line 28
```

此时功能与中断的功能相同。但是如果选择了“是”，ModelSim 就会自动关闭，所有仿真的波形和数据都不会被保存。

5.4.4　单元库

ModelSim 通过了 ASIC 委员会制定的 Verilog 测试集并由此获得了通过测试的库，即获得了被该

委员会认可的库。使用到的测试集是专门为了确保 Verilog 的时序正确性和功能正确性而设计的，是完成全 ASIC 设计的重要支持。许多 ASIC 厂商和 FPGA 厂商的 Verilog 单元库与 ModelSim 的单元库并不冲突。

单元模块通常包含 Verilog 的“特定块”，这些块用来描述单元中的路径延迟和时序约束。在 ModelSim 模块中，源引脚(input 或 inout)到目的引脚(output 或 inout)的延迟称为模块路径延迟(module path delay)，在 Verilog 中，路径延迟用关键字 specify 和 endspecify 表示。在这两个关键字之间的部分构成一个 specify 块。时序约束是指对于各条路径上数据的传输和变化做一个时间上的约束，使整个系统能够正常地工作。

Verilog 模型可以包含两种延迟：分布式延迟和路径延迟。在 Primitive、UDP 和连续赋值语句中定义的延迟是分布式延迟，而端口到端口的延迟被定为路径延迟。这两个延迟相互作用，直接影响最终观测到的实际延迟。大多数的 Verilog 单元库中仅仅使用到路径延迟，而分布式延迟则被设置成零。分布式延迟的例子如下：

```
module or2(y, a, b)
input a, b;
output y;
or (y, a, b)
specify
     (a => y) = 4;
     (b => y) = 4;
endspecify
endmodule
```

上面这个代码是一个二输入或门的例子，这个或门的分布式延迟被定义为 0，实际从模块端口得到的延迟是从路径延迟中获得的，路径延迟已经说过，是在 specify 和 endspecify 中定义的。这个例子不是一个独立的实例，大多数的单元都是采用这种结构进行建模的。当单元中需要指定两种延迟时，这两种延迟中比较大的一种延迟被使用到各条路径中，这是一种默认的准则。另外，在 ModelSim 中，编译器的延迟模式参数要优先于延迟在这个代码指令中的模式。

单元库中包含的延迟模型主要有以下几种：

(1) Distributed delay mode(分布式延迟模型)。在分布式延迟模型中，路径延迟信息是被忽略的，重点关注分布式延迟。可以使用编译参数“+delay_mode_distributed”或者使用编译指令“'delay_mode_distributed”调用这种延迟模型。

(2) Path delay mode(路径延迟模型)。在路径延迟模型中，分布式延迟在所有的模块中都被设置成 0。可以使用编译参数“+delay_mode_path”或者使用编译指令“'delay_mode_path”调用这种延迟模型。

(3) Unit delay mode(单位延迟模型)。在单位延迟模型中，所有非零的分布式延迟被设置成一个时间单位，这个时间单位会在设计文件的“'timescale”中定义，或在仿真时设置。可以使用编译参数“+delay_mode-unit”或者使用编译指令“'delay_mode_unit”，调用这种延迟模型。

(4) Zero delay mode(零延迟模型)。在零延迟模型中所有的分布式延迟被设为 0，而且所有的路径延迟和时序约束都被忽略。可以使用编译参数“+delay_mode_zero”或者使用编译指令“'delay_mode zero”调用这种延迟模型。

5.5　针对不同器件的时序仿真

时序仿真，是利用 SDF 文件对原有设计进行时序标注，继而进行仿真的方式。后仿真从一定程度上可以反映设计的时序性能，更加接近设计的实际工作情况。本章前面所做的仿真称为功能仿真，主要是验证功能是否符合设计要求。

ModelSim 本身并不能生成后仿真需要的 SDF 文件，但是由于 ModelSim 对多数 FPGA 厂商的支持，使其可以利用其他 FPGA 工具生成的 SDF 文件进行时序仿真。本节中以主流的 Altera 公司和 Xilinx 公司的工具为例，介绍 ModelSim 如何对这两个公司的器件进行时序仿真。

5.5.1　ModelSim 对 Altera 器件的时序仿真

Altera 公司的 FPGA/CPLD 器件占据了大量的市场，很多学校和公司都使用 Altera 公司的产品进行设计和开发，如何利用 ModelSim 对 Altera 提供的器件进行后仿真也是很多初学者面临的问题，由于采用的设置方式不同，加之对两种软件提供的功能不是非常了解，往往会出现不能进行后仿真的情况。在本节中会详细地介绍如何使用 ModelSim 对 Altera 器件进行后仿真。

Quartus 与 ModelSim 进行后仿真的流程有两种：一种是直接使用 Quartus 调用 ModelSim 进行时序仿真；另一种是使用 Quartus 生成 ModelSim 进行后仿真需要的文件，再使用 ModelSim 进行时序仿真。这两种仿真的实际效果都是一样的，只是采用步骤和设置有所不同。

使用 Quartus 调用 ModelSim 时，需要设置的是 ModelSim 的路径。因为此时 Quartus 需要按指定的路径调用 ModelSim 软件。采用这种方法的时候还要注意 ModelSim 的 license 问题，当 license 达到上限时是无法启动 ModelSim 的。例如，PC 的 license 仅能支持一个 ModelSim，如果打开 ModelSim 的时候再使用 Quartus 调用 ModelSim 就会产生错误。使用第一种方法进行后仿真的流程可以归纳为以下几步：

(1) 在 Quartus 中创建工程并按向导进行设置。

(2) 指定 ModelSim 仿真的 Testbench。

(3) 在 Quartus 中执行综合、布局布线、时序分析等步骤。

(4) 生成网表文件和 SDF 文件后，调用 ModelSim。

(5) ModelSim 自动完成仿真功能。

第二种方法是分段操作的：先使用 Quartus 软件；再使用 ModelSim 进行时序仿真。这时 ModelSim 不是被调用的，而是设计者自行启动的，这里就会有一个问题：在第一种方法中，Quartus 调用 ModelSim 的时候会把需要的库文件同时加载到 ModelSim 中，但是在第二种方法中这些库文件是没有的，需要设计者指定，如果没有库文件的支持，整个设计显然是无法仿真的。使用第二种方法进行后仿真的流程可以归纳为以下几步：

(1) 在 Quartus 中创建工程并按向导进行设置。

(2) 在 Quartus 中执行综合、布局布线、时序分析等步骤。

(3) 生成网表文件和 SDF 文件后，退出 Quartus。

(4) 启动 ModelSim，编译对应器件的库文件。

(5) 把 Quartus 生成的后仿真文件添加到工程中。

(6) 编译添加的文件，进行仿真。

以上两种方式有两步的描述都是相同的，但是设置上有所不同，随后的综合、布局布线等操作的显示也有不同。

1. 在 Quartus 中创建工程并按向导进行设置

这里以 Quartus Prime Lite 22.1 为例介绍采用 ModelSim 的时序仿真。启动 Quartus Prime Lite 22.1，选择菜单栏中的 File→New Project Wizard，打开一个新的工程向导，如图 5-59 所示。

执行该命令后，会出现工程向导对话框。这个工程向导有很多步骤，第一步出现的是介绍，如图 5-60 所示。这个对话框中介绍了本工程向导如何建立一个新的工程，用项目编号的形式给出了包含的步骤：工程名和目录、顶层设计名称、工程文件和库、目标器件设定、EDA 工具的选择。第一步是介绍页面，不需要选择，可以单击 Next 按钮进入下一步，如果不想在以后新建工程的过程中看到这个页面，可以把左下角的 Don't show me this introduction again 勾选上。

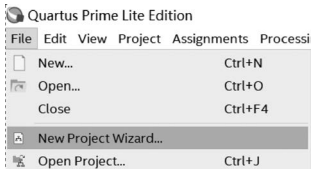


图 5-59 新建工程向导

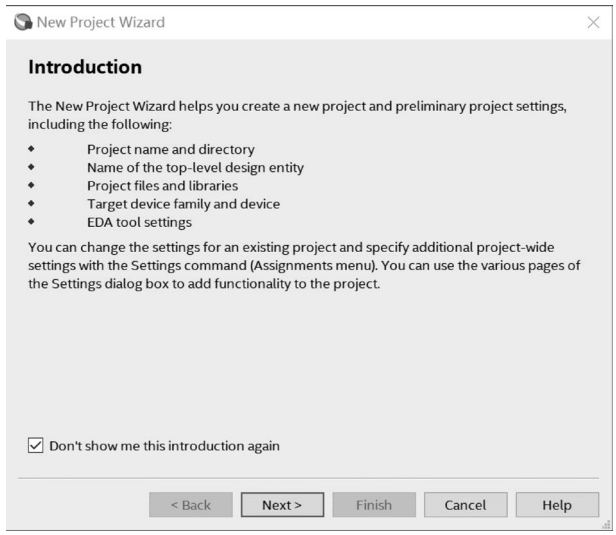


图 5-60 工程向导-介绍

进入向导的下一步是设置工程的目录、工程名和顶层模块，如图 5-61 所示。第一栏中指定当前工程使用的目标文件夹，这里在默认目录下建立一个名为 mywork 的文件夹，用来存放本节的例子，即目标路径为 C:\intelFPGA_lite\。第二栏中指定工程的名称，在此栏中输入的工程名会被 ModelSim 默认为顶层模块名，如图 5-61 中所示，在第二栏中输入工程名 fpadd，在第三栏中就会同步显示 fpadd。这里就需要注意：Quartus 对工程名没有特别的要求，但是对第三栏中的顶层设计单元是有要求的，图中也说得很详细，这个顶层设计单元的名称必须要和设计文件中的顶层单元名称相同，否则在 Quartus 进行分析的时候就会提示找不到设计单元。为了避免错误，一般都是采用和顶层设计相同的工程名称，即如图 5-61 所示的样式。如果比较熟练，可以指定不同的工程名和顶层设计单元名称。设置好这三个参数后，单击 Next 按钮进入下一步设置。

设置工程名后需要指定添加的文件，如图 5-62 所示。首先要单击图中 File name 空白栏后的按钮浏览需要添加的文件，单击此按钮后会自动打开前一步中设置的工程目录；然后将此工程用到的源文件复制到该目录中，同时选中这些文件，选中的文件名就会显示在 File name 一栏；再单击 Add 按钮后这些文件就会添加到工程中，图中中间区域所示就是添加后的文件；添加文件并配置库文件后单击 Next 按钮进入下一步。

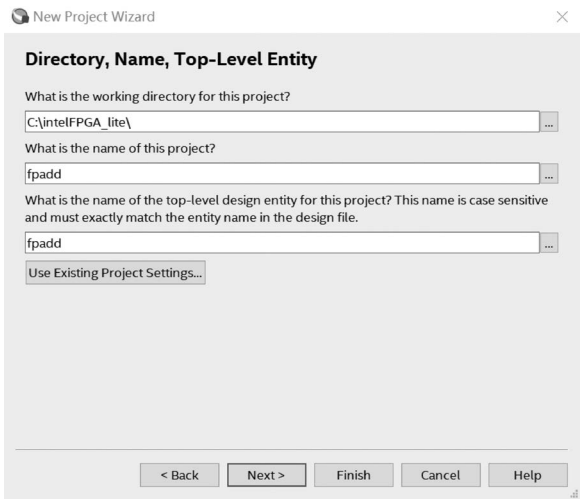


图 5-61　工程向导-目录、工程名和顶层模块

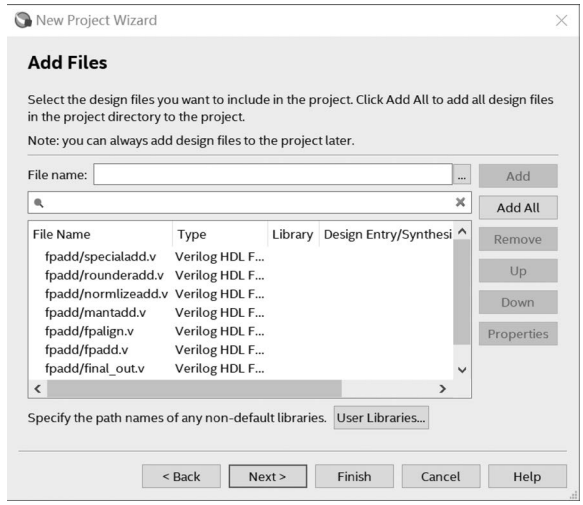


图 5-62　工程向导-添加文件

添加文件后要指定使用的 FPGA 器件，如图 5-63 所示。如图中标示的位置，选择器件族 Cyclone 10 LP，选择 Auto device selected by the Fitter，让 Quartus 根据需要选择器件。要记住选择的器件族，后仿真的时候需要用到，选择好后单击 Next 按钮进入下一步。

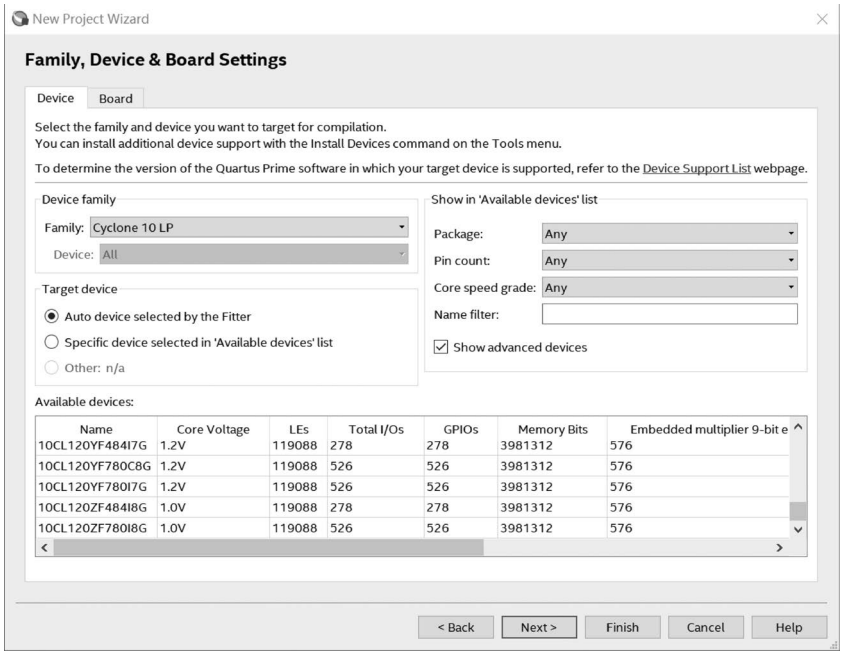


图 5-63　工程向导-选择器件

这一步中需要指定 EDA 工具，可以使用的 EDA 工具分别是：综合工具、仿真工具、形式验证工具和板级工具。本节中使用到的是仿真工具(如图 5-64 所示)，在 Simulation 行的 Tool Name 下拉菜单中选择工具 ModelSim，根据文件的需要选择文件形式，Format 一栏为 Verilog HDL。特别要注意，在第一种流程中要勾选选项 Run gate-level simulation automatically after compilation，这个选项会在编译后直接运行门级仿真，即运行时序仿真。设置好 EDA 工具后单击 Next 按钮进入最后一步。

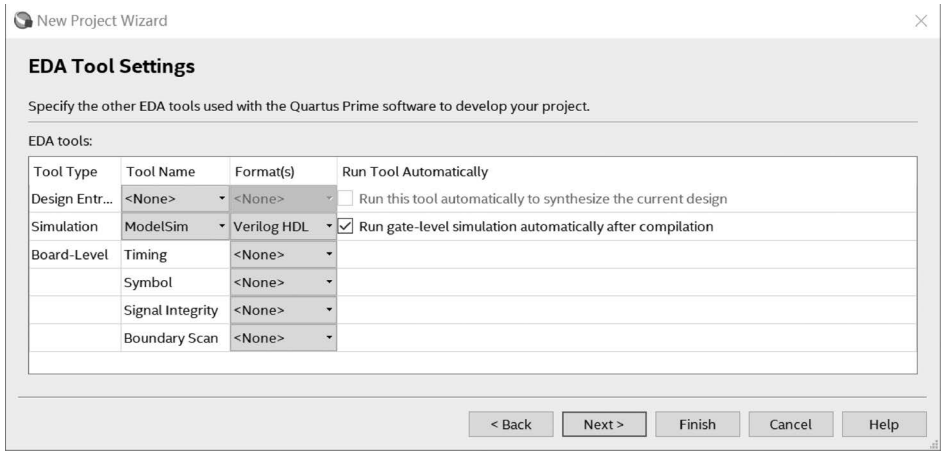


图 5-64　工程向导-指定仿真工具

最后一步是前面所有设置的一个摘要，如图 5-65 所示。此界面可以查看前面设置的所有项目，如果与预想不同，可以单击 Back 按钮返回上一步进行修改。确认无误后单击 Finish 按钮结束工程向导。

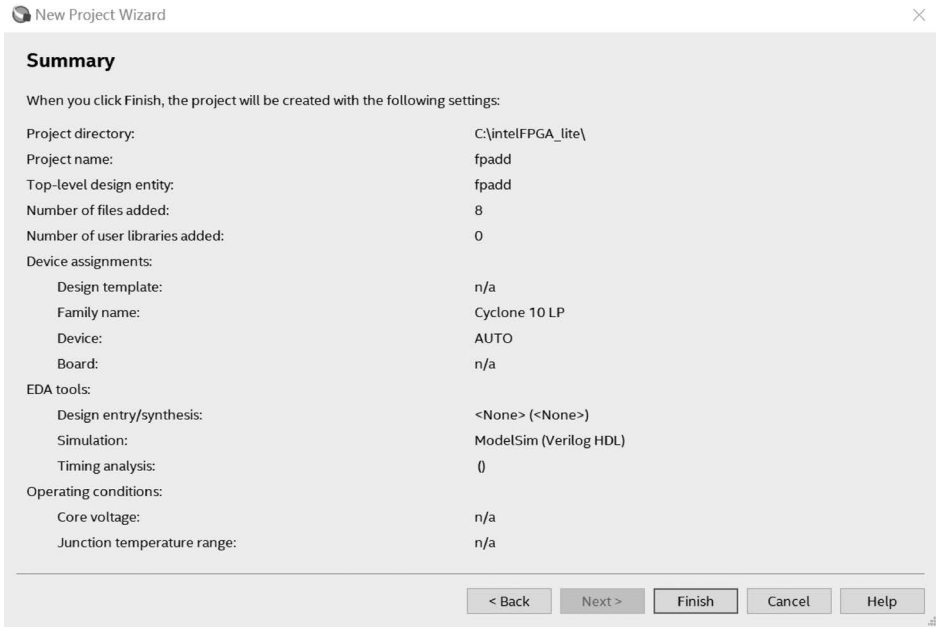


图 5-65　工程向导-确定设置

单击 Finish 按钮结束向导后，在工程区域的显示会发生变化，如图 5-66 所示。在层次标签中显示的是顶层设计单元的名称 fpadd，名称上方的 Cyclone 10 LP：AUTO 表示自动选择 Cyclone 10 LP 器件族的器件，如果在工程设置中指定了器件，AUTO 则会被选中的器件型号代替。在文件标签中显示所有添加的 8 个文件，这 8 个文件被划分在器件设计文件夹中。

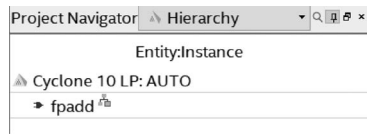


图 5-66　工程区域的显示

2. 指定 ModelSim 仿真的 Test bench

若想进行自动仿真，需要为设计提供一个 Test bench，否则进入 ModelSim 后会陷入等待状态，而不能充分发挥 Quartus 的功能。由

于工程向导中设置的仿真选项很简单，这里需要再次启动详细的设置，选择菜单栏中的 Assignments→Settings 命令。运行该命令后会显示如图 5-67 所示的窗口，可以详细地设置各项参数，其中包括 EDA Tool Settings 下的 Simulation 选项，在左侧选中此项后，右侧的区域会显示详细的参数设置。上方标示的区域是设置仿真工具和启动门级仿真选项，如果在工程设置中没有修改仿真工具，可以在此处进行修改。在 EDA Netlist Writer settings 中指定输出的网表格式是 Verilog HDL，仿真的时间刻度也改为 1ns，输出的目标文件夹是 simulation/modelsim。下方标示的 NativeLink settings 区域是指定 Test Benches 的位置，默认选项是 None，选中第二项 Compile test bench 可以进行指定。单击 Test Benches 按钮后会出现图 5-68 所示的窗口，此窗口中加入 tb_fpadd.v 测试文件。

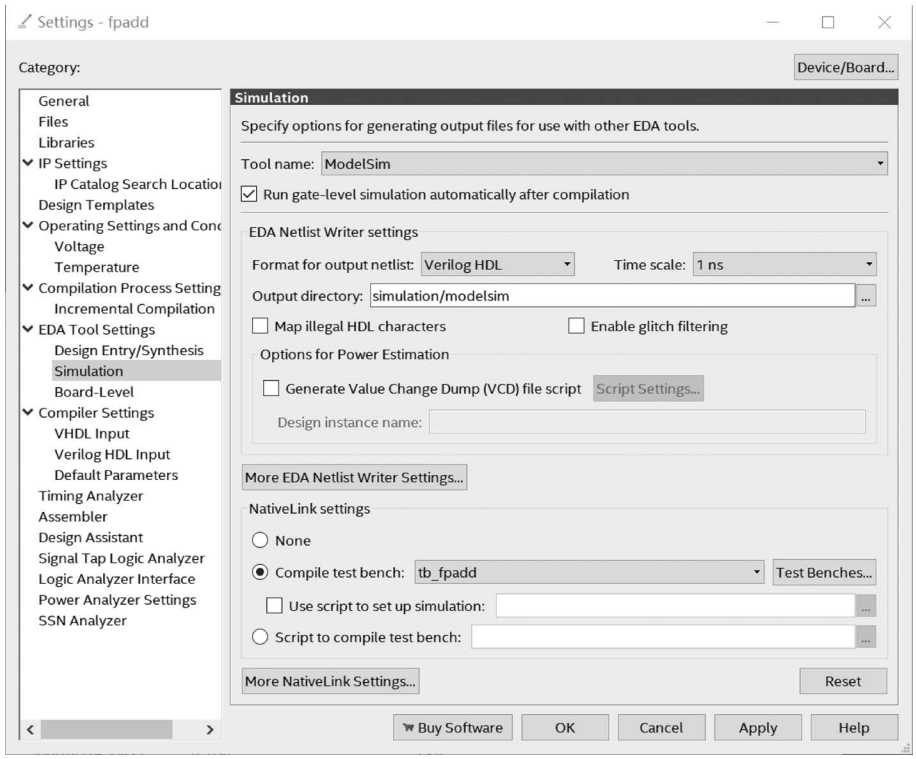


图 5-67　设置仿真参数

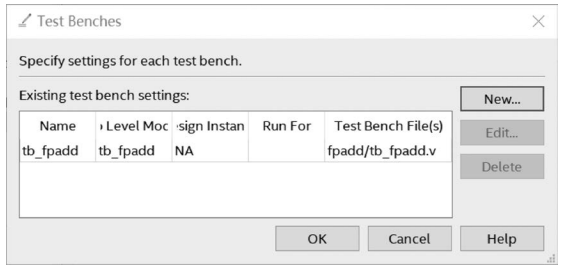


图 5-68　设置仿真测试文件

单击 New 按钮后会出现如图 5-69 所示的窗口，这是设置的关键窗口。窗口分为两部分，第一部分有三栏需要填写，要在 Test bench name 一栏中填写测试平台名称 tb_fpadd.v，在 Top level module in test bench 一栏中填写测试平台的顶层模块名称 tb_fpadd，勾选上 Use test bench to perform VHDL

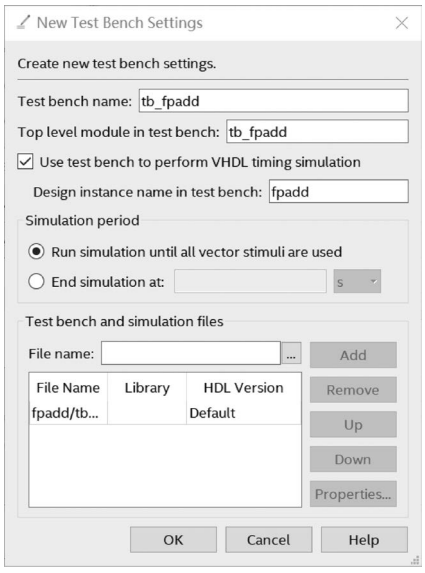


图 5-69　添加仿真测试文件

timing simulation，可以进行时序仿真；在第二部分里可以指定运行仿真时的终止条件，选择第一个选项 Run simulation until all vector stimuli are used，即所有的仿真向量运行结束时终止仿真，第二个选项是指定一个具体的时间，当仿真运行指定时间后会终止仿真，可以在使用的时候根据需要设置，设置这两处后还要在下方的区域添加测试文件，添加方式与工程中添加文件的方式相同。设置好所述的几项之后单击 OK 按钮会出现图 5-67 所示的显示，指定 Testbench 的步骤就结束了。

设置了测试平台，还需要指定 ModelSim 的安装路径，否则 Quartus 无法调用 ModelSim，选择菜单栏中的 Tools→Options 选项，会出现图 5-70 所示的对话框，在左侧选中 EDA Tool Options，在右侧区域就会显示 EDA 工具，选择其中的 ModelSim 行，把路径信息设置为 C:\modeltech64_2020.4\win64，即 ModelSim.exe 所在的文件夹，单击 OK 按钮确认设置。

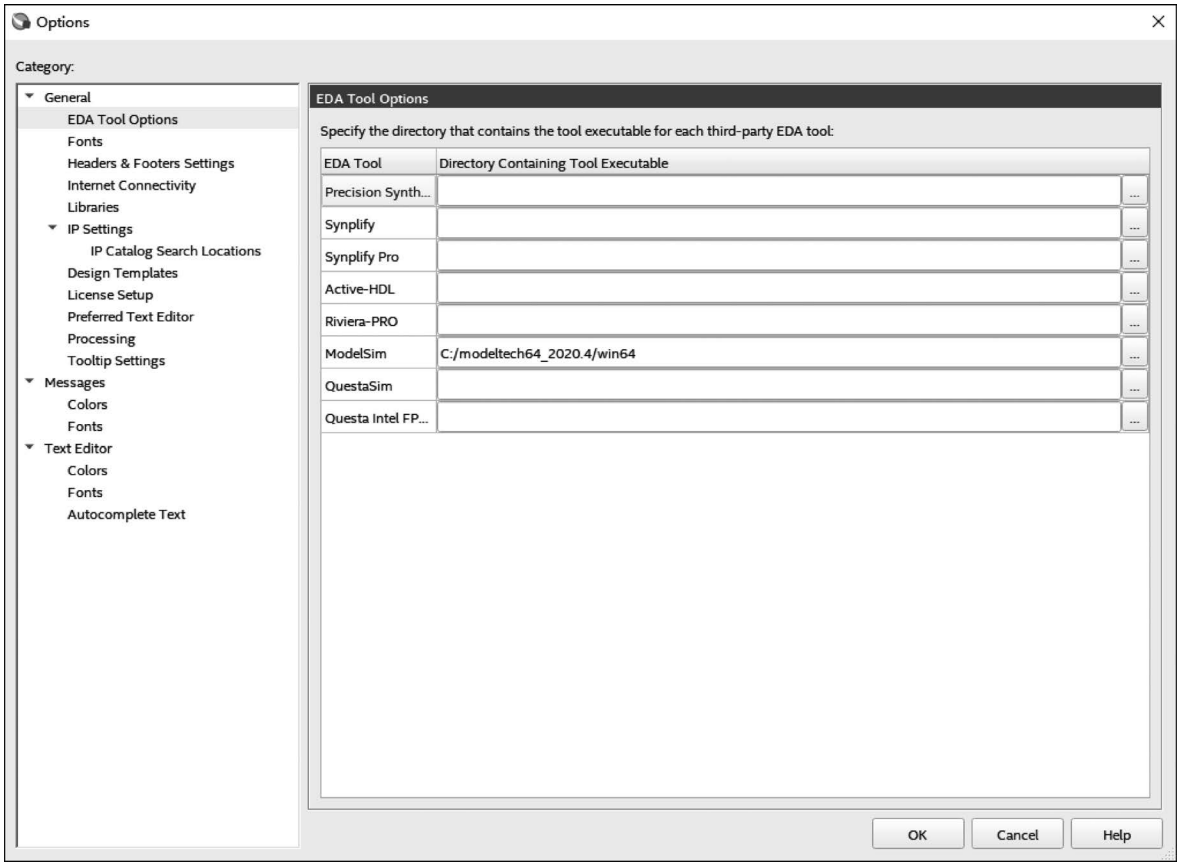


图 5-70　配置 ModelSim 路径

3. 在 Quartus 中执行综合、布局布线、时序分析等步骤

启动 Quartus 的各项功能有不同的用处，每项功能都可以做大篇幅的解释，这里采用一种比较简单的方式。在工具栏中单击 Start Compilation 按钮后会自动地完成后仿真需要的全部步骤（在菜单中也可以分别进行调用），如图 5-71 所示。

同时，在 Quartus 的状态区会显示要进行的操作状态，如图 5-72 所示。在该图中显示了五个步骤：Analysis&Synthesis、Fitter、Assembler、Timing Analysis、EDA Netlist Writer，这里不去关心前四个步骤，这些步骤是为最后的仿真做准备的。与 EDA 工具有关的是最后两个操作的运行状态。EDA Netlist Writer 是生成 EDA 工具需要的网表文件，生成的文件后缀为“.v0”格式，同时生成的文件还有一个后缀名为“.sdo”的 SDF 文件，这两个文件都是时序仿真的必要文件。执行此步操作之后，Quartus 会自动调用指定的 EDA 软件进行门级仿真，这里指定的是 ModelSim，会启动 ModelSim 进行仿真。

图 5-71　启动编译

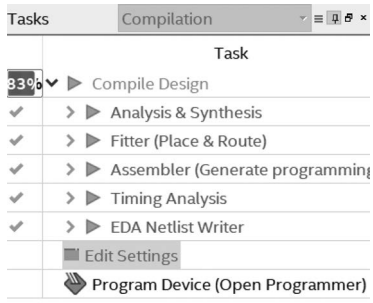


图 5-72　编译过程需要进行的操作及运行状态

在进行分析和综合后，工程区的显示会发生变化，如图 5-73 所示。层级标签中会显示整个设计的设计层次，可以单击前面的箭头来展开设计层次；设计单元标签中会显示全部的设计单元，还有该单元对应的 HDL 类型。

随着 Quartus 中操作的进行，任务区各条命令会显示其完成状态和运行时间，在如图 5-74 所示的 Compile Design 功能完成 83%的时候，完成 ModelSim 的功能仿真如图 5-75 所示。退出 ModelSim 软件，完成整个编译过程。

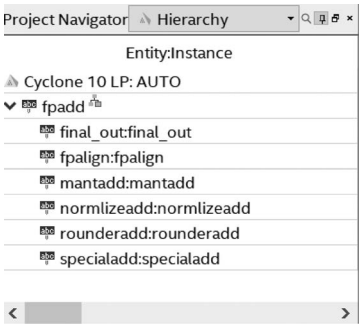


图 5-73　工程区显示的变化

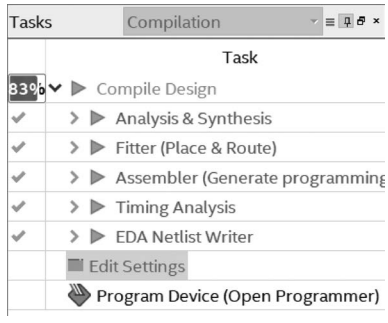


图 5-74　编译过程进度

4. ModelSim 自动完成仿真功能

在第一种流程中，启动了 ModelSim 后的工作完全由上一步中 fpadd_run_msim_gate_verilog.do 来执行，所有的编译、信号添加、仿真等原本需要手工操作的命令都会在这个.do 文件中。该.do 文件中包含的信息如图 5-76 所示。

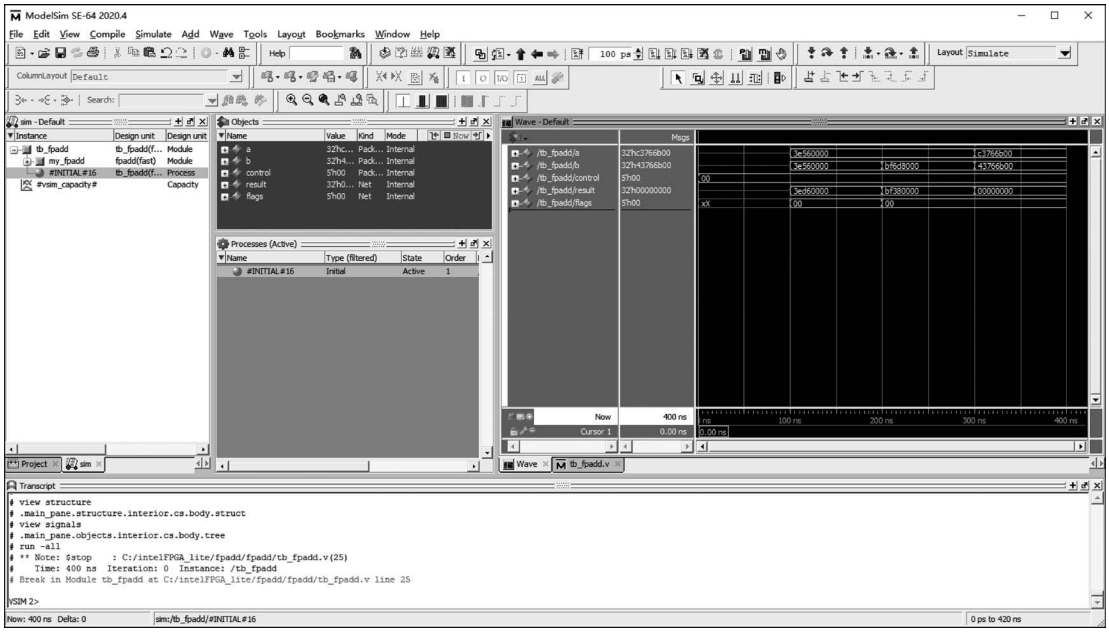


图 5-75　调用 ModelSim 完成功能仿真

```
transcript on
if ![file isdirectory verilog_libs] {
    file mkdir verilog_libs
}

vlib verilog_libs/altera_ver
vmap altera_ver ./verilog_libs/altera_ver
vlog -vlog01compat -work altera_ver {c:/intelfpga_lite/22.1std/quartus/eda/sim_lib/altera_primitives.v}

vlib verilog_libs/cyclone10lp_ver
vmap cyclone10lp_ver ./verilog_libs/cyclone10lp_ver
vlog -vlog01compat -work cyclone10lp_ver {c:/intelfpga_lite/22.1std/quartus/eda/sim_lib/cyclone10lp_atoms.v}

if {[file exists gate_work]} {
    vdel -lib gate_work -all
}
vlib gate_work
vmap work gate_work

vlog -vlog01compat -work work +incdir+. {fpadd.vo}

vlog -vlog01compat -work work +incdir+C:/intelFPGA_lite/fpadd {C:/intelFPGA_lite/fpadd/tb_fpadd.v}

vsim -t lps -L altera_ver -L cyclone10lp_ver -L gate_work -L work -voptargs="+acc"  tb_fpadd

add wave *
view structure
view signals
run -all
```

图 5-76　生成的.do 仿真文件

整个的运行过程除了少数 TCL 语句之外，全都是 ModelSim 中使用到的命令行形式。首先建立了一个 Altera 的库文件，用来支持对 Cyclone 10 LP 器件族的仿真，然后建立了工程库，把预先设置好的测试平台和在 Quartus 中刚刚得到的设计优化文件进行编译和仿真，接着添加波形，运行仿真，最后等待操作。简而言之，就是按照.do 文件中的命令依次执行，直至运行到 run -all 命令，然后停止。得到的波形如图 5-77 所示。

如果对仿真波形有深入认识，就会明白时序仿真与逻辑仿真的不同。前面的逻辑仿真波形中，如果输入信号发生了变化，输出的波形就会立即发生变化，因为之前进行的仿真都是功能仿真，没有时序的

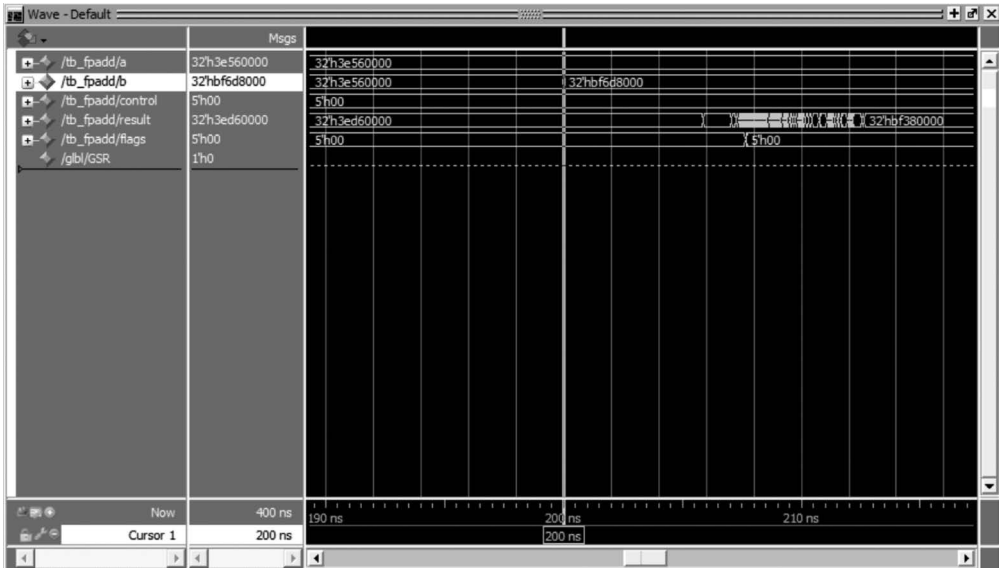


图 5-77　仿真波形(显示)

信息,只是按照程序的代码依次执行。而在时序仿真中,每一条代码程序都要耗费一定的时间,这样从输入信号变化到输出信号就需要一定的时间。在图中标示出了两个光标,一个是输入信号发生变化的位置,对应的时间点是 200ns; 另一个是输出信号发生变化的位置,对应的时间点是 212.654ns,从输入信号到输出结果相差了 12.654ns,这就是本例中浮点加法器的实际工作时间。当然,实际的器件中工作时间可能与这个时间还不相同,但是比较接近的。

如果注意观察会发现,输入信号发生变化后,输出信号 result 会发生多次的波动,在图中表现就是一段浓密的信号,如果把图示中光标的波形放大到如图 5-78 所示的波形,会看出细节上变化,这也是时序仿真特有的。因为本例是一个组合逻辑器件,注定内部会发生一系列的变化,只有当信号稳定后的输

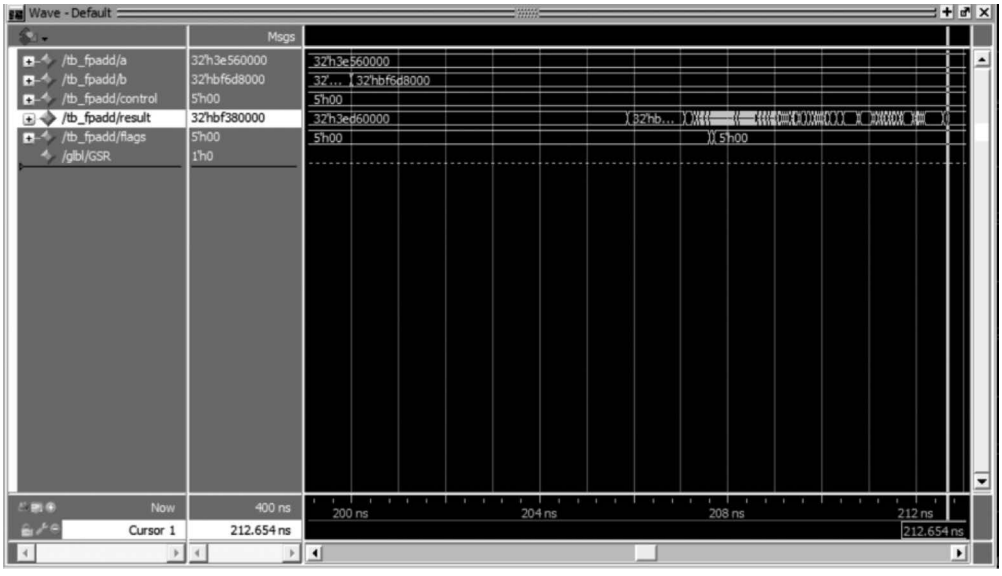


图 5-78　放大的细节波形

出才是最终的输出，而中间状态只是运算过程中的副产品。观察仿真波形后就可以关闭 ModelSim 了，关闭 ModelSim 后 Quartus 会继续接管控制权。

5.5.2 ModelSim 对 Xilinx 器件的时序仿真

当使用 Xilinx 器件的时候，就必须使用 Xilinx 公司提供的软件进行编译和后仿真。Xilinx 器件的使用范围也很广泛，本节中会对 Xilinx 器件的后仿真进行介绍。

本节中使用的版本是 Vivado 2022.2，该版本 ISE 打开后的整体界面如图 5-79 所示。该界面与 Quartus 的界面相似，由于还没有建立工程，所以左侧区域中没有层次显示和进程信息等，建立工程后即可出现。

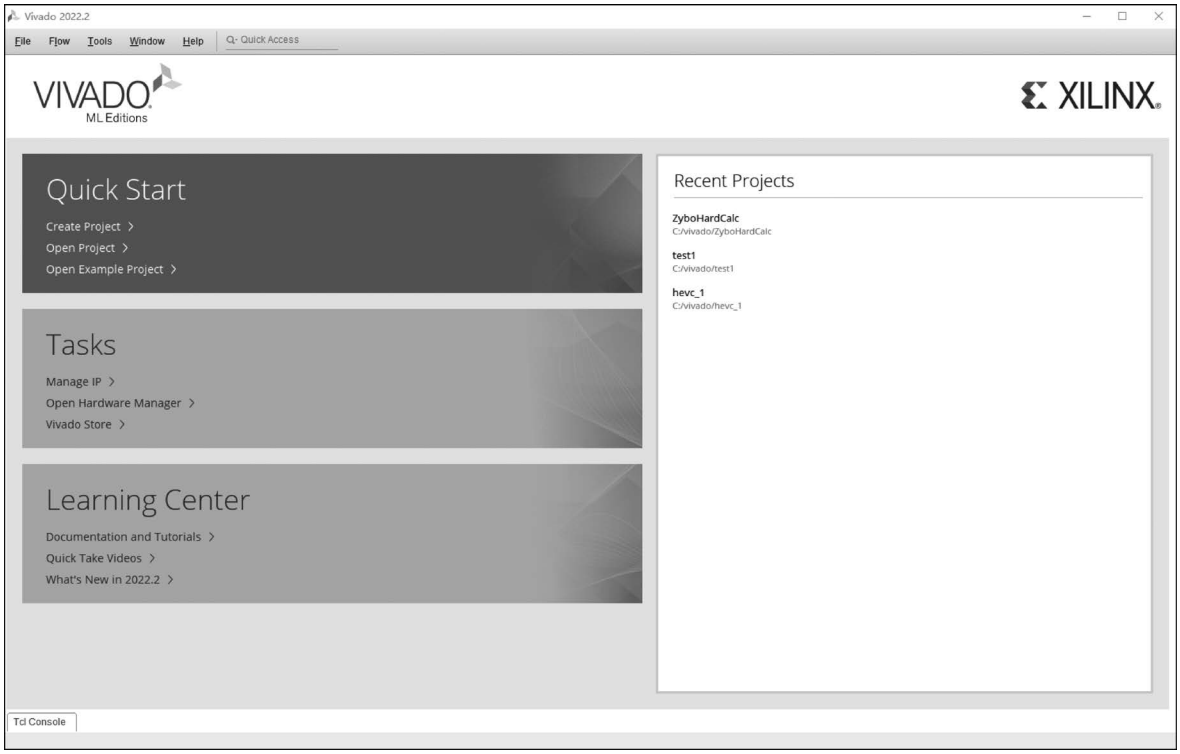


图 5-79 Vivado 2022.2 界面

Vivado 的时序仿真流程和 Altera 的流程相同，也可以分为两种：第一种是先使用 Vivado 生成后仿真需要的文件，再启动 ModelSim 进行仿真；第二种是使用 Vivado 启动 ModelSim 进行仿真。

第一种流程与 Altera 的流程相似，这里归纳为 6 个步骤。

(1) 在 Vivado 中创建工程并完成设置。

(2) 在 Vivado 中执行编译区的综合、布线等功能。

(3) 生成布线后仿真模型，退出 Vivado。

(4) 启动 ModelSim，编译 Xilinx 库文件。

(5) 把 Vivado 生成的后仿真文件添加到工程。

(6) 编译添加的文件，进行仿真。

下面在实例中详细讲解。

1. 在 ISE 中创建工程并完成设置

启动 ISE 后，在菜单栏中选择 File→NewProject，打开新的工程，如果之前从未建立过工程，会在初始界面的左侧看到工程选项。选择图 5-79 主界面左上角的 New Project 一样可以建立新工程，生成图 5-80 所示的工程向导界面。

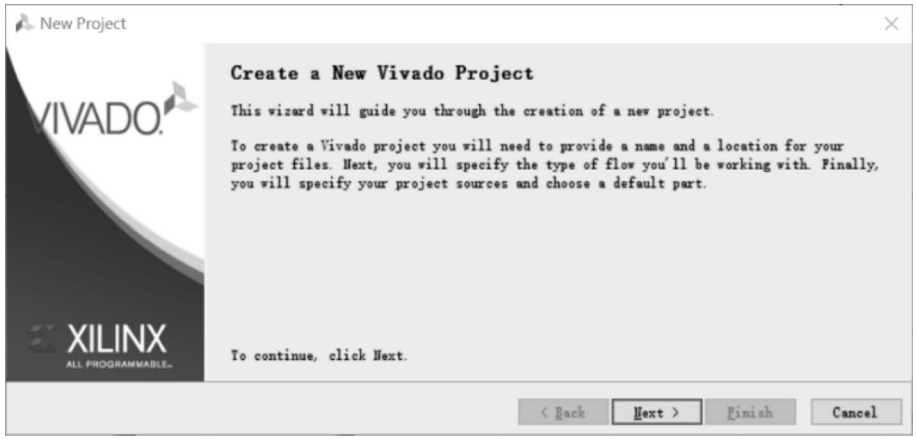


图 5-80　Vivado 工程向导界面

选中新建工程命令后会出现新建工程向导，同 Quartus 一样，也分为多个步骤，第一步的窗口如图 5-81 所示，要设置工程的名称和工程路径，并指定顶层设计的类型。在 Project name 中填写 fpadd，在 Project location 中指定目录为 E:/vivado，单击 Next 按钮进入下一步。下一步是创建工程类型，出现的窗口如图 5-82 所示。选择 RTL Project 即可。

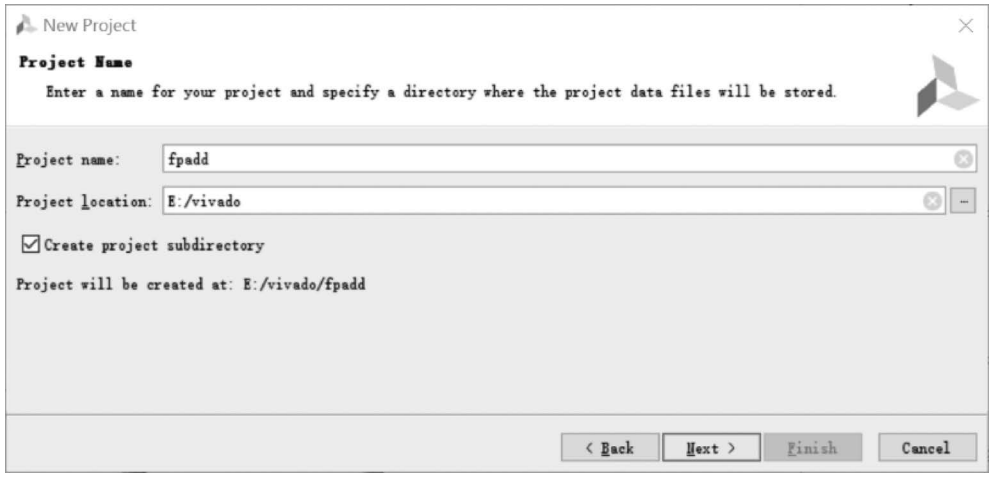


图 5-81　指定工程名称

设置工程信息后需要添加指定工程原文件，如图 5-83 所示。首先要单击图中 Add Files 空白栏后的按钮浏览需要添加的文件，单击 OK 按钮后这些文件就会添加到工程中，图中中间区域所示就是添加后的文件。添加文件后单击 Next 按钮进入下一步。

下一步为添加工程约束文件，如图 5-84 所示。首先要单击 Create File 按钮，在 Create Constraints Files 对话框中输入 File Name：为 fadd 后单击 OK 按钮退出。系统将生成的约束文件添加到工程中，图中中间区域所示就是添加后的文件。添加文件后单击 Next 按钮进入下一步。

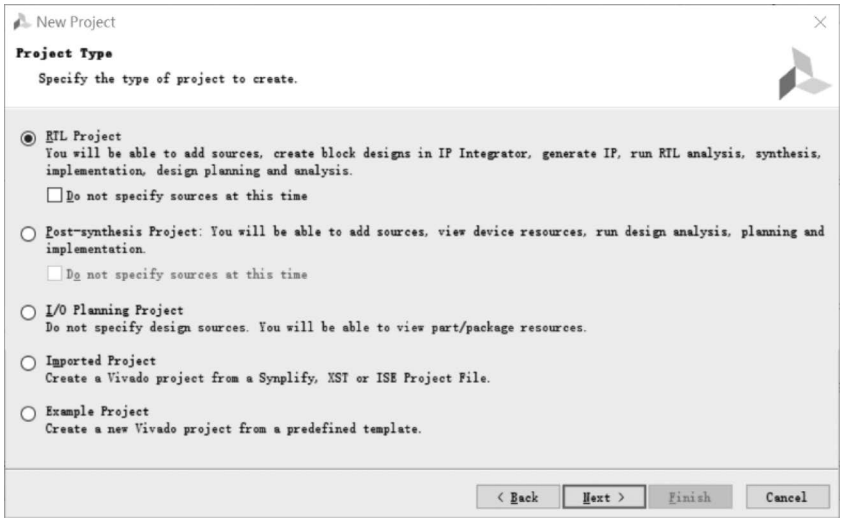


图 5-82　选择工程类型

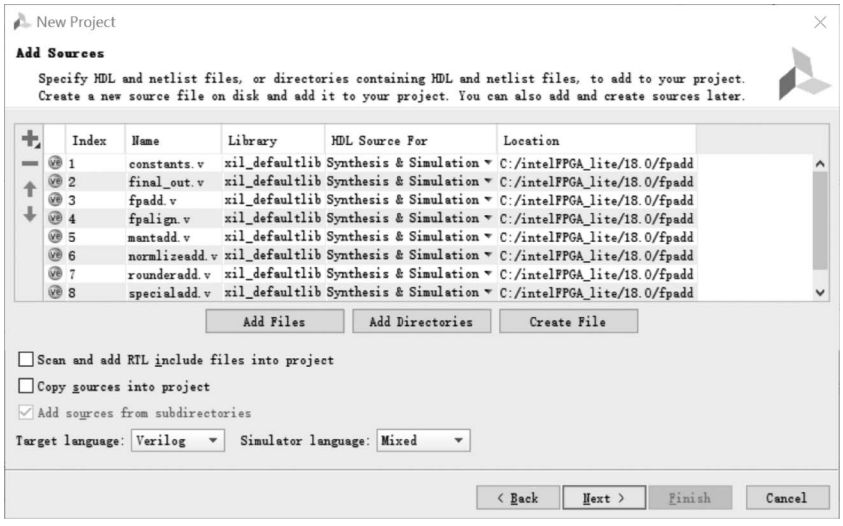


图 5-83　添加工程文件

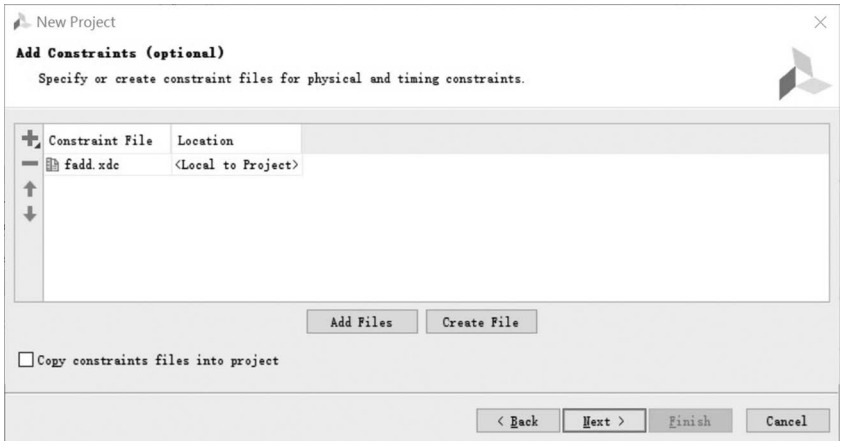


图 5-84　选择约束文件

下一步是器件选择和工具的指定，出现的窗口如图 5-85 所示。在窗口的上半部分是器件的选择，还可以选择不同的开发板。由于没有固定的器件可供使用，这里随意选择即可，本例中选择的是 xcku025-ffva1156-2-e，单击 Next 按钮进入下一步。

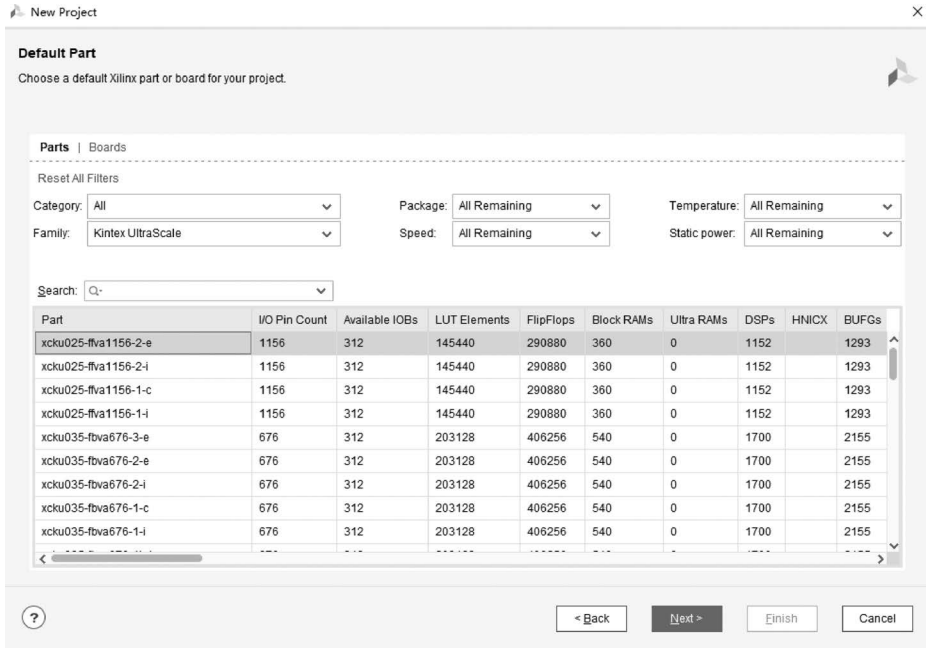


图 5-85　选择器件

最后一步是工程设置的概要，供使用者确认，如图 5-86 所示。如果有问题，则可以单击 Cancel 按钮取消操作；如果没有问题，则单击 Finish 按钮完成工程。

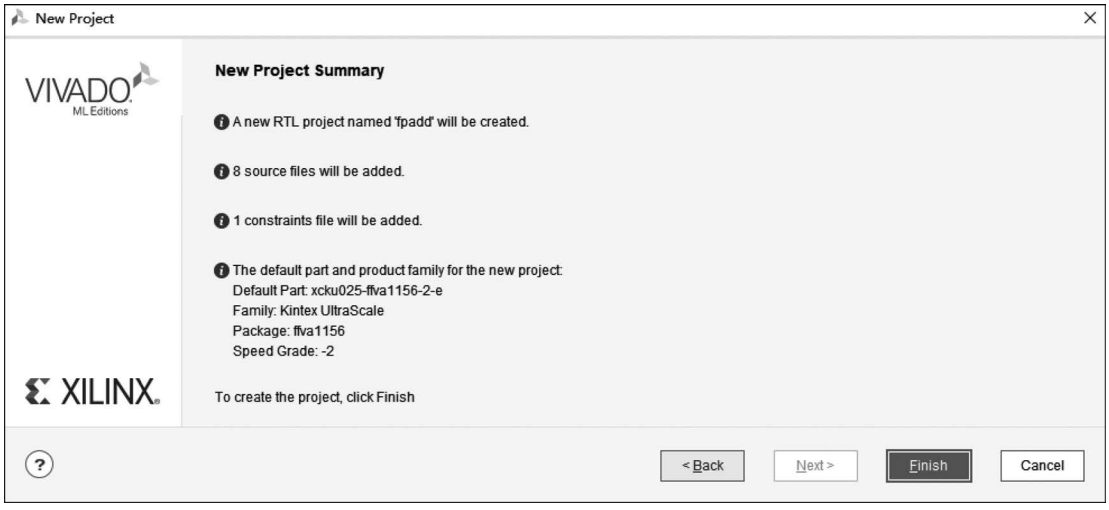


图 5-86　工程设置的概要

完成工程后在 Vivado Project Manager 窗口会出现图 5-87 所示的信息，在 Hierarchy 选项卡区域中显示的是工程设计文件和仿真文件信息；层次显示区域会出现本设计中包含的模块，这里显示的是 fpadd(fpadd.v)，括号内显示的是文件名，括号外显示的是该文件中包含的模块名，共包含 6 个模块；在

Simulation Sources 单击右键选择 Add Sources 加入仿真测试文件 tb_fpadd 作为仿真源文件，如图 5-87 所示。

2. 在 Vivado 中执行编译区的综合、布线等功能

在 Vivado 中执行编译区的综合、布线等功能可以通过如图 5-88 所示的 Flow Navigator 窗口进行。如果要执行综合则选择 Run Synthesis，要完成实现则选择 Run Implementation。在完成了综合、转译、布局等过程后，在 Vivado 窗口会出现图 5-89 所示的信息，包括编译、综合、DRC 检查、系统资源利用情况、时序报告、电源报告信息。

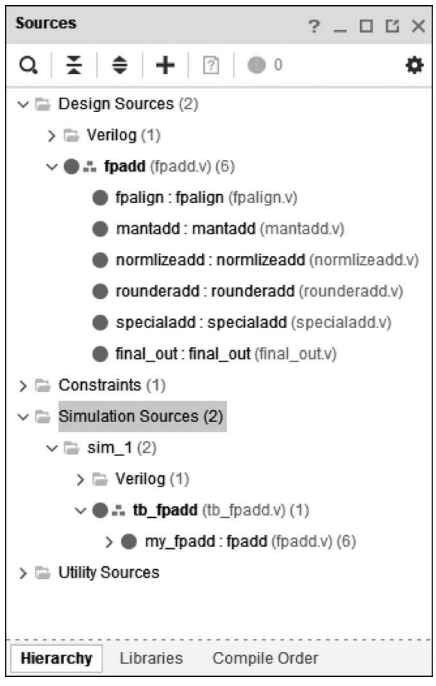


图 5-87　工程相关信息

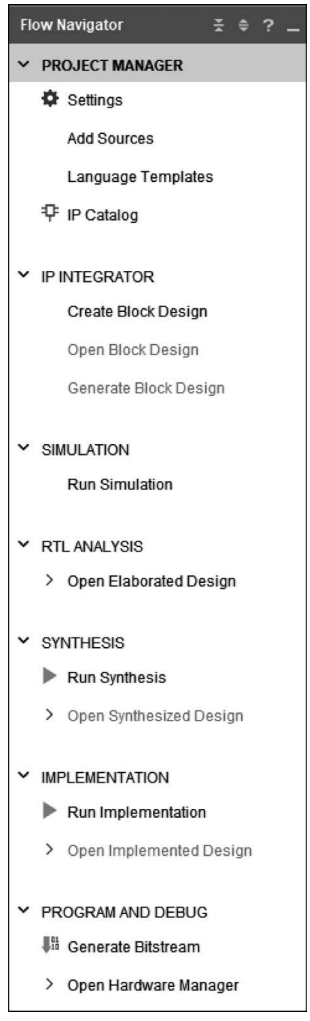


图 5-88　Flow Navigator 窗口界面

3. 设置 ModelSim 仿真环境并执行仿真

通过菜单 Tools→Compile Simulation Libraries 进行仿真模型器件库的编译，如图 5-90 设置仿真模型为 ModelSim 及其所在路径，系统将自动生成执行仿真的命令。单击 Compile 按钮进行仿真模型器件库的编译。

编译结束后可以在 E:\vivado\fpadd\fpadd.cache\compile_simlib\modelsim 目录中生成用于 ModelSim 进行仿真的器件库，如图 5-91 所示。

图 5-89　工程完成综合和实现后的窗口界面

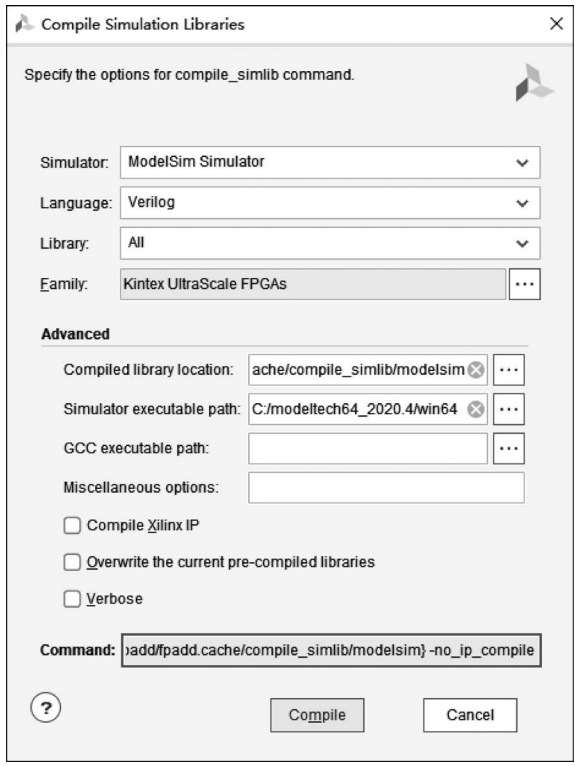


图 5-90　设置编译 ModelSim 仿真界面

图 5-91　编译生成 ModelSim 使用的器件库

在 Vivado 中设置 ModelSim(即第三方仿真工具)的安装路径。在 Vivado 菜单中选择 Tools→Options,选择 Simulation 选项卡,将滚动条拉到底部,在 Target Simulator 栏中选择 ModelSim 工具的安装路径,如图 5-92 所示。

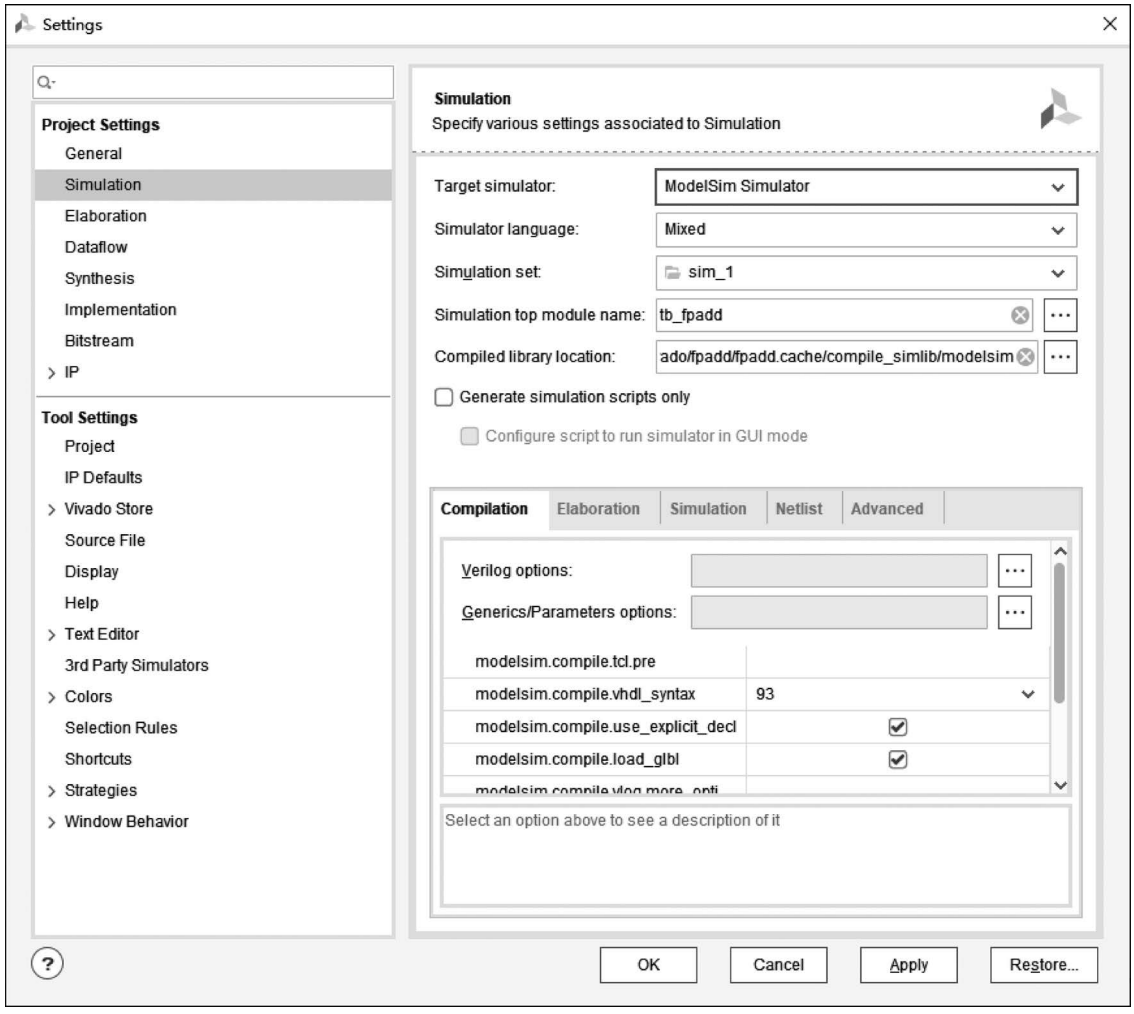


图 5-92　设置使用 ModelSim 进行仿真并设置安装路径

设置好仿真参数后，如果设计文件和仿真文件也准备好，那么就可以开始对设计的功能进行仿真了。选择 Flow→Run Simulation→Run Post-Implementation Timing Simulation 类型或单击流程向导中的 Run Simulation→Run Post-Implementation Timing Simulation 进行仿真，如图 5-93 所示。

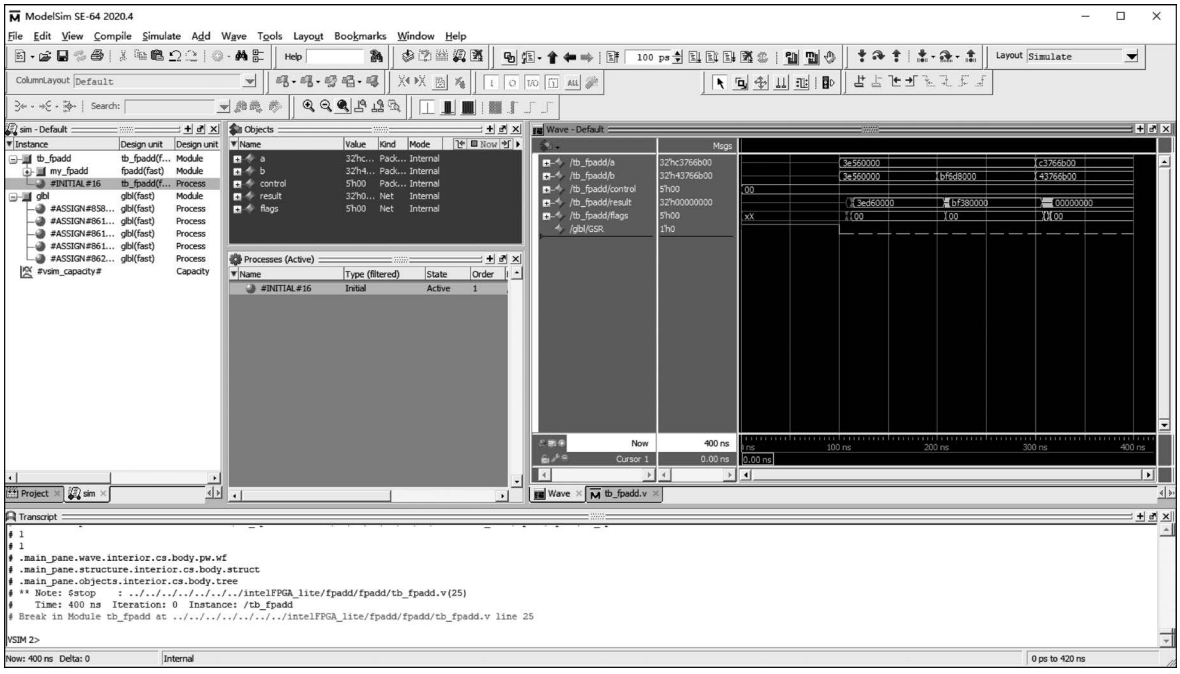


图 5-93　使用 ModelSim 进行时序仿真结果