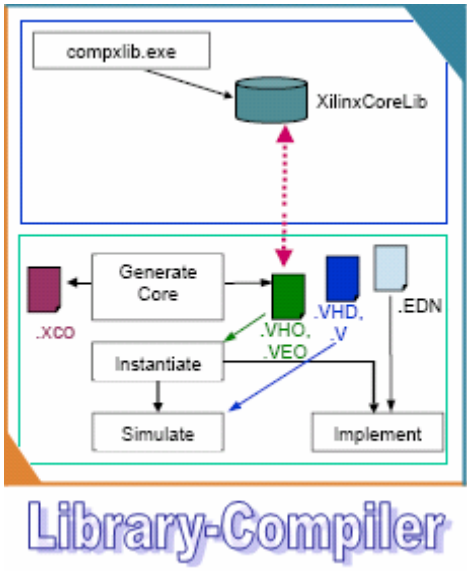
FPGA 设计全流程：Modelsim>>Synplify.Pro>>ISE

第一章　Modelsim 编译 Xilinx 库



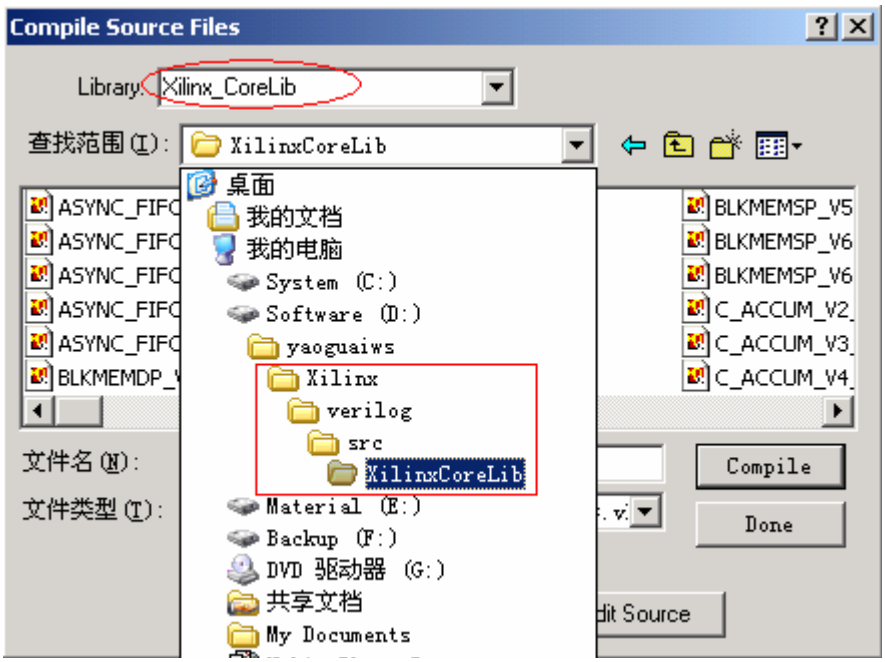
本章介绍如何编译 HDL 必须的 Xilinx 库和结构仿真。

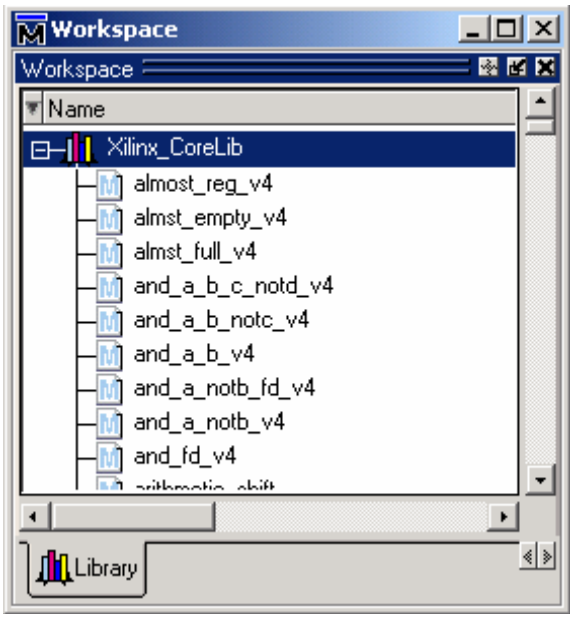
创建将被编译库的目录

在编译库之前，最好先建立一个目录（事实上必须建立一个目录），步骤如下。（假设 Modelsim 的安装目录是“ $Modeltech_6.0 ”，ISE 的安装目录是“ $Xilinx ”）

- 在“ $Modeltech_6.0/ ”目录下建立一个名为 XilinxLib 的文件夹；
- 启动 Modelsim 后，从“ File ”菜单项中点击“ Change Directory ”并指定到刚刚建立的那个文件夹“ XilinxLib ”；
- 接下来要做的事情是将 Xilinx 库编译到“ XilinxLib ”文件夹中。有三个库需要被编译。它们分别是“ simprims ”,“ unisims ”和“ XilinxCoreLib ”；(所有这些库文件都在“ $Xilinx/verilog/src ”目录下)
- 点击 Modelsim 中的“ Workspace ”窗口，建立一个名为 Xilinx_CoreLib 的新库；(这个操作创建一个名为“ Xilinx_CoreLib ”的文件夹，你可以在“ Workspace ”窗口中看到它)
- 现在开始编译！在“ Compile ”菜单中点击“ Compile ”，选择

“ $Xinlinx/verilog/scr/XilinxCoreLib ”目录中所有的文件，在弹出的对话框中选中刚刚建立的“ Xilinx_CoreLib ”文件夹，再点击“ Compile ”按钮就可以了编译了；





◆ 用同样的方式编译其他两个本地库（“ simprims ”和“ unisims ”）；

第二章　调用 Xilinx CORE-Generator

当需要在设计中生成参数化和免费的 IP 内核（黑箱子）时，无论是通过原理图方式还是 HDL 方式，CORE-Generator 都是一个非常有用的程序。

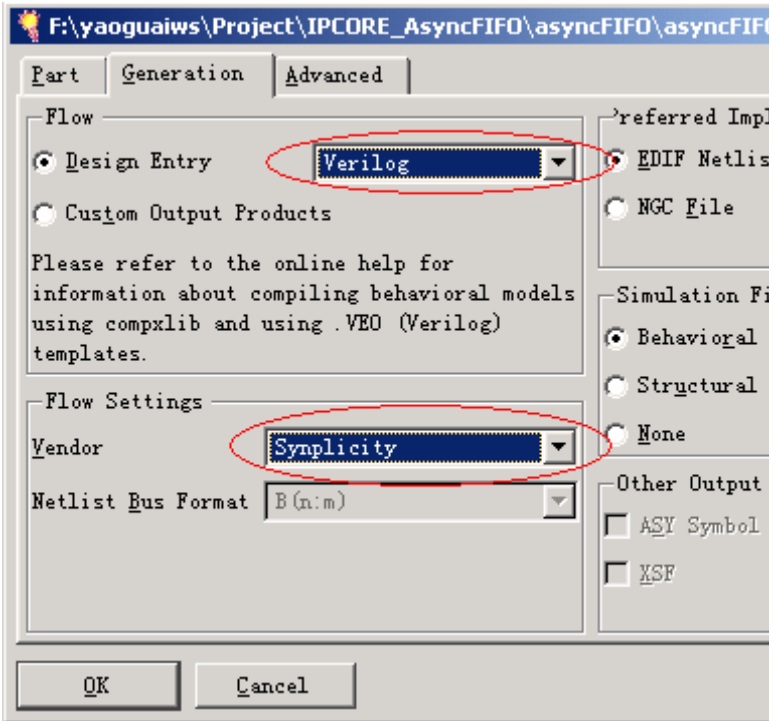
利用 CORE-Generator 创建一个 IP 核

利用 Xilinx 提供的 CORE-Generator 来生成 IP 核是非常简单的。内核是全参数化的，这就意味着你只需要在空白处填入几个数字和参数，然后程序就会自动产生一个你所需要的

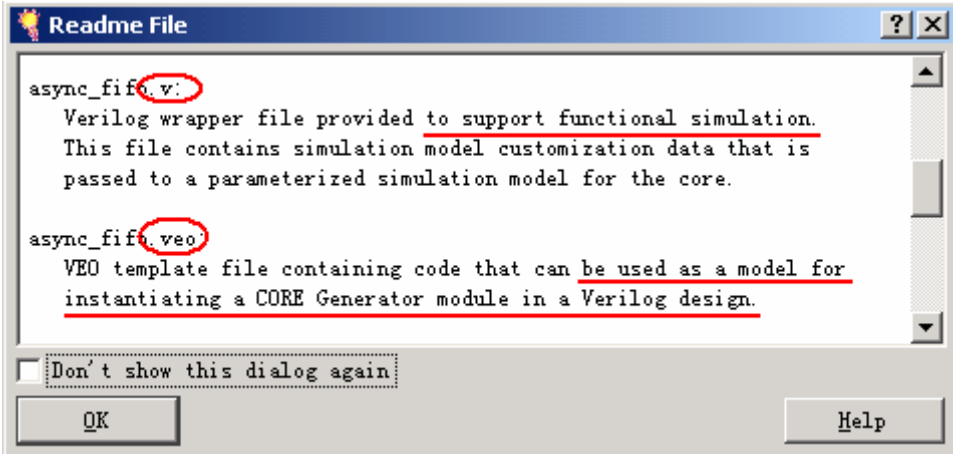
内核。（有些内核是全免费的，有些则没有这么慷慨）

利用 CORE-Generator 来生成 IP 核的步骤如下：

- 在“ 程序 ”中找到“ Xilinx ”项，然后在“ Accessories ”中启动单独存在的“ CORE-Generator ”；
- 在“ Part ”标签栏中选择恰当的 FPGA 模型；

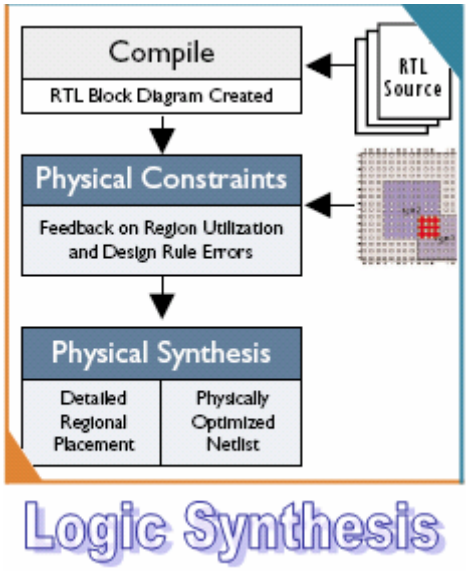


- 从“ Generation ”标签栏中选择正确的设计流；（完成后按“ OK ”按钮）
- 定制你的参数化内核；
- 在内核生成的同时，会弹出一个“ Readme File ”的信息框来通知一些重要的信息；



- “ *.v ”文件是用来作仿真和综合用的，而“ *.veo ”文件是用来作综合实例用的。（调用意味着把相应的文件加入 Synplify.Pro 工程中，而实例指的是可以拷贝这个文件中的某些线到 HDL 设计的顶层模块中去。**退出！**）

第三章　　使用 Synplify.Pro 综合 HDL 和内核



综合是将设计好的 HDL 代码，图形代码和原理图转变成逻辑单元的技术。同与硬件执行和物理布线非常接近的物理综合相比，逻辑综合是更高层次的综合技术。

利用 Synplify.Pro 进行逻辑综合

Synplify.Pro 对于大容量低价格的 Xilinx Spartan 系列 FPGA 而言，有着非常好的综合能力。

具体步骤如下：

- 首先创建一个工程；
- 往工程中加入 HDL 文件（我的演示文件有三个文件，CORE-Generator 生成的“async_fifo.v”和“dcm4clk”和一个 Verilog 顶层文件“top.v”）。在 Synplify.Pro 环境中设置“Implementation Option”；（如果读者非常熟练的话，可以省略这步）
- 往 CORE-Generator 生成的两个 Verilog 文件中插入 Synplify.Pro 能够识别的指示这些指示告诉综合器如何处理这两个特殊的文件；

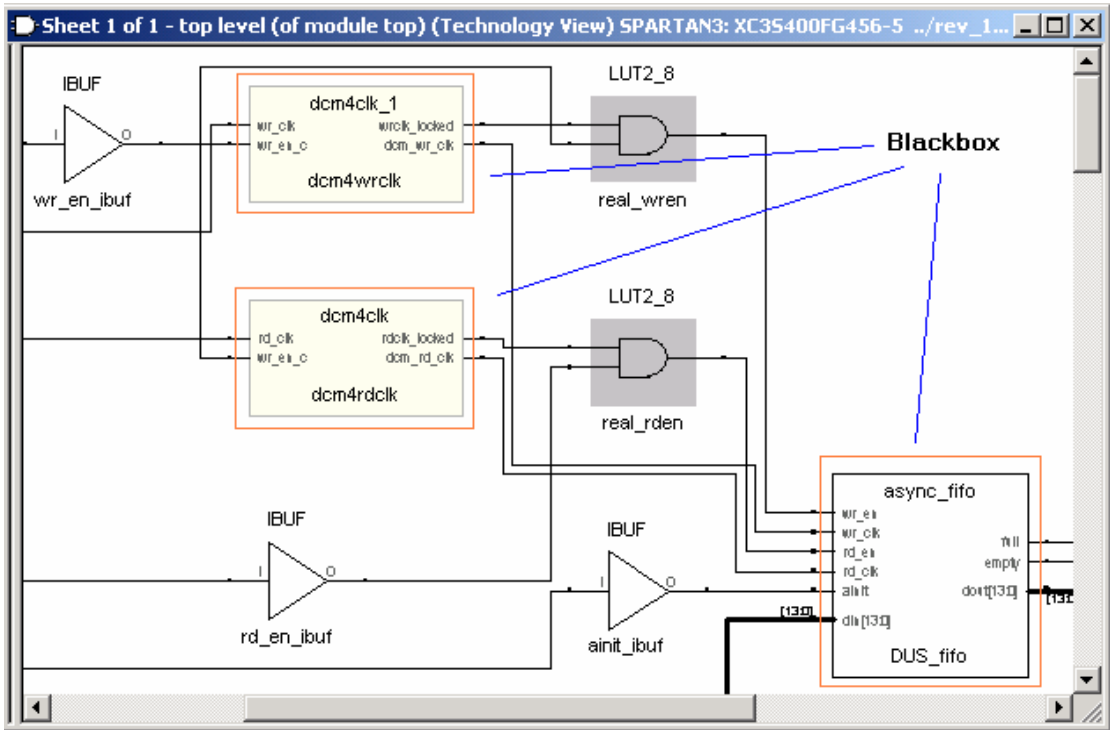
ISE\dcm4clk.v (verilog)

```
00021 `timescale 1ns / 1ps
00022
00023 module dcm4clk(CLKIN_IN,
00024                RST_IN,
00025                CLKIN_IBUFG_OUT,
00026                CLK0_OUT,
00027                LOCKED_OUT)/* synthesis syn_black_box */;
00028
00029    input CLKIN_IN /* synthesis syn_isclock=1 */;
00030    input RST_IN;
00031   output CLKIN_IBUFG_OUT;
00032   output CLK0_OUT;
00033   output LOCKED_OUT;
00034
```

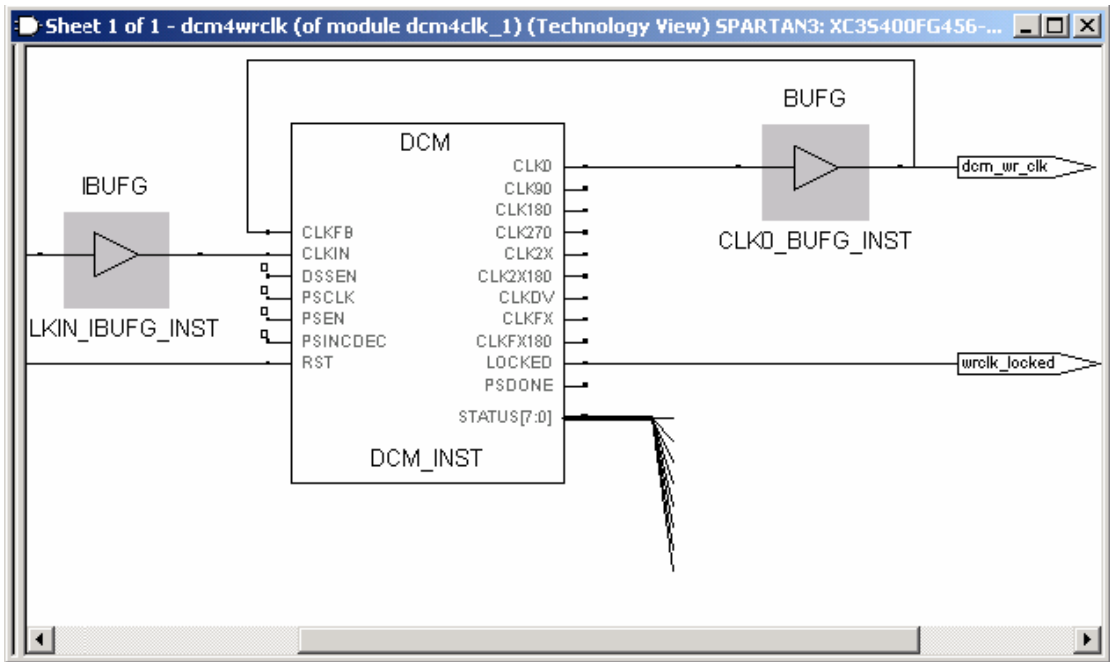
Line 20 Col 40

- 插入“/*synthesis syn_black_box*/”指示通知 Synplify.Pro 把模块当作黑箱子来处理，同时指示“/*synthesis syn_isclock=1*/”表示这个作为时钟输入端的端口不能被综合器识别，因为它除了端口名外没有下层结构；
- 将工程保存在合适的地方，然后综合这个工程；
- 在综合完成后，选择“Technology View”按钮来观察层次结构；（你可以发现内核

文件已经被综合成黑箱子了）



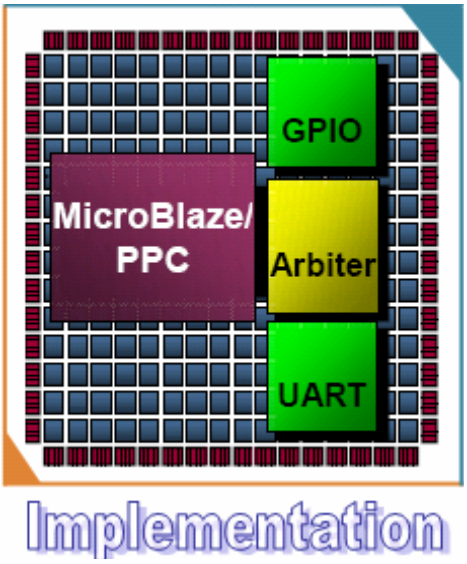
◆ 继续深入了解“ dcm4clk1 ”模块的结果；



◆ 不管你相信与否！ Synplify.Pro 已经生成了你所希望的东西。（拥有专用 Clock-Input-Buffer，IBUG 连接的 DCM 结构，并且有一个从 Global-Clock-Buffer，BUFG 的反馈结构“ CLKFB ”）

第四章　综合后的项目执行

执行是将生成的位文件下载到 FPGA 的最后一个步骤。



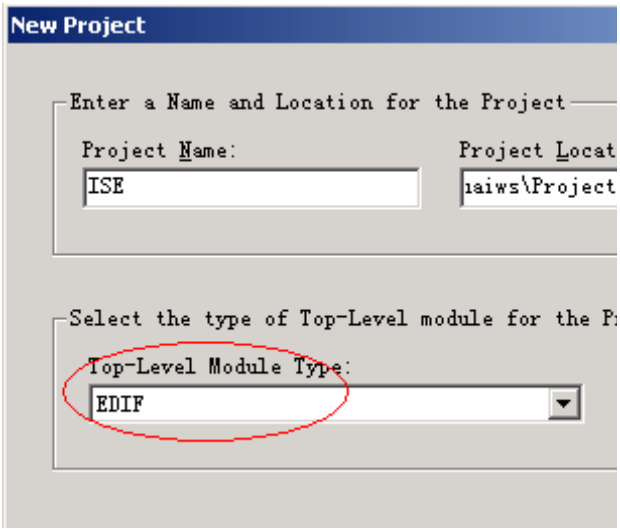
同时创建不同的时序模型（post-translate 模型，post-map 模型和 post-PAR 模型）和时序报告。

ISE，唯一可以用来执行的工具

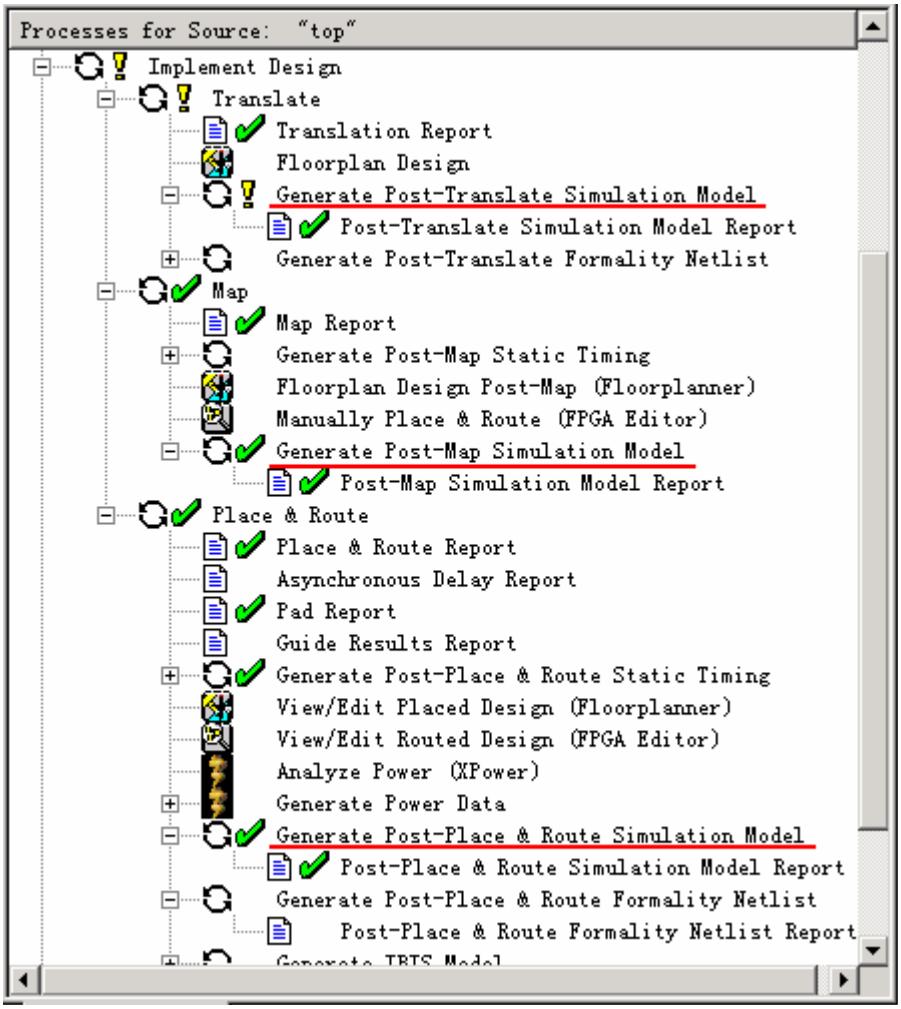
ISE 控制着设计流的各个方面。通过 Project Navigator 界面，可以进入所有不同的设计实体和实际执行工具。同时也可以访问于工程有关的文件和文档。Project Navigator 包含一个平坦的目录结构；

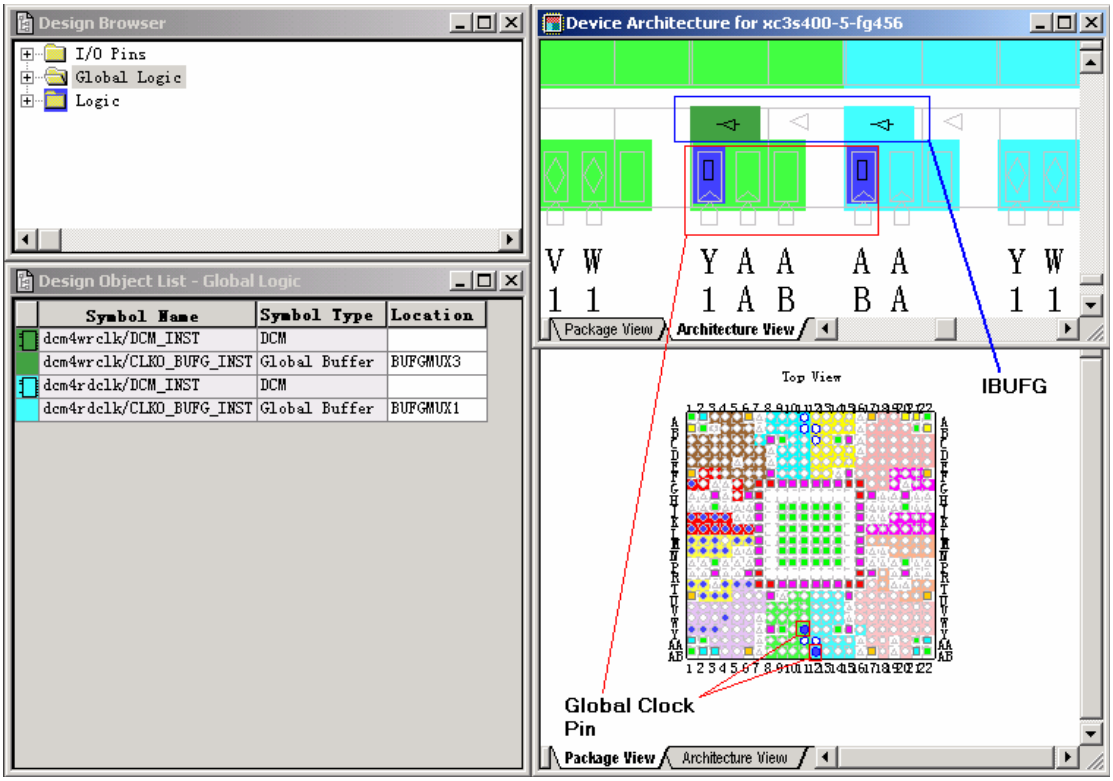
在演示项目中，ISE 的一些贫乏的功能不得不让道给其他的第三方软件，例如 ModelSim.Pro 和 Synplify.Pro，因此 ISE 一般仅仅被用作执行工具。

- 启动 ISE，用“ EDIF ”作为文件输入；(“ EDIF ”文件由 Synplify.Pro 软件生成，作为终端设计文件，可以被大多数的 FPGA 开发环境识别，例如 ISE，Quartus，ispLevel。)

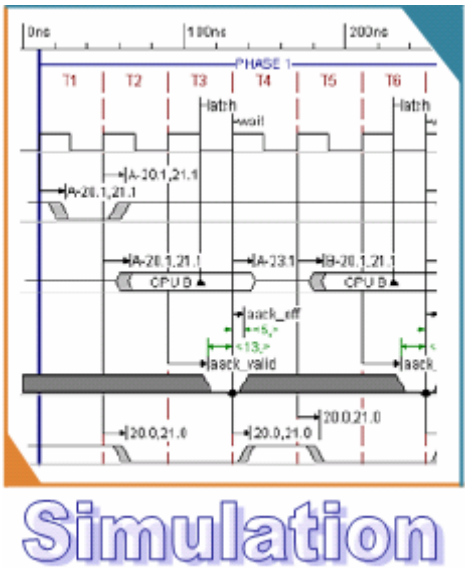


- 在建立了 ISE 工程后，可以加入其他两个文件，一个是与内核相关的“ *.xco ”文件，另一个是与 DCM 结构有关的“ *.xaw ”；
- 现在可以生成需要仿真的所有的模块；(点击下划红线的选项)
- 如果想仿真 post-PAR 模块，最好首先定义引脚，特别是专用的外部时钟引脚；
- 启动“ map ”程序中的“ Floorplaner ”选项来定义引脚；
- “ DCMs ”和“ IBUFGs ”应该被放在正确的位置。


Processes for Source: "top"
Implement Design
Translate
Translation Report
Floorplan Design
Generate Post-Translate Simulation Model
Post-Translate Simulation Model Report
Generate Post-Translate Formality Netlist
Map
Map Report
Generate Post-Map Static Timing
Floorplan Design Post-Map (Floorplanner)
Manually Place & Route (FPGA Editor)
Generate Post-Map Simulation Model
Post-Map Simulation Model Report
Place & Route
Place & Route Report
Asynchronous Delay Report
Pad Report
Guide Results Report
Generate Post-Place & Route Static Timing
View/Edit Placed Design (Floorplanner)
View/Edit Routed Design (FPGA Editor)
Analyze Power (XPower)
Generate Power Data
Generate Post-Place & Route Simulation Model
Post-Place & Route Simulation Model Report
Generate Post-Place & Route Formality Netlist
Post-Place & Route Formality Netlist Report


Design Browser
I/O Pins
Global Logic
Logic
Design Object List - Global Logic
Symbol Name
Symbol Type
Location
dcm4wrclk/DCM_INST
DCM
dcm4wrclk/CLKO_BUFG_INST
Global Buffer
BUFGMUX3
dcm4rdclk/DCM_INST
DCM
dcm4rdclk/CLKO_BUFG_INST
Global Buffer
BUFGMUX1
Device Architecture for xc3s400-5-fg456
Package View
Architecture View
Top View
IBUFG
Global Clock
Pin

第五章　不同类型结构的仿真



仿真是用来验证设计的时序和功能是否正确的调试方法之一。

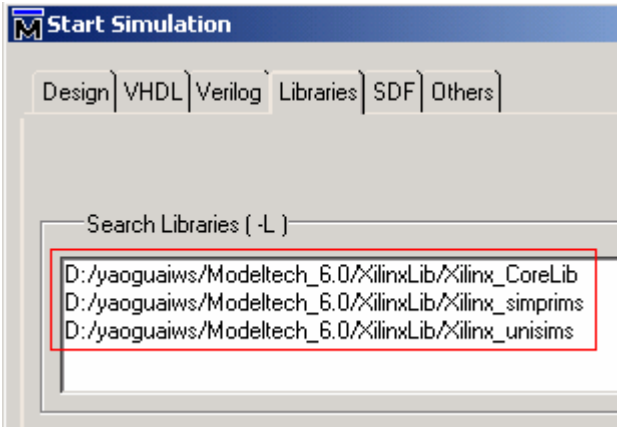
在验证调试电路和观察波形的过程中，应该进行四个不同类型的仿真。

不同的仿真类型针对的不同的平台。功能仿真用来验证设计的功能是否正确；post-translate 仿真用来验证设计的基于原语延时；post-map 仿真用来仿真基于原语延时和网络延时；最后，post-PAR 仿真在 post-map 仿真的基础上加入了输入输出和布线延时。

我不会给出演示设计的全部详细的仿真过程，但是给出了重点和重要的步骤。只给出了 post-PAR 仿真过程，列出了对于所有其他三种仿真需要的不同文件。（实际上，不同的文件是不同的参考时序模型：<DesignName>_translate.v 是 post-translate 模型，<DesignName>_map.v 是 post-map 模型。）

- 对于 post-PAR 仿真，需要四种类型的文件，“ glb1.v ”是用来作 FPGA 全局复位的（从“ $Xilinx/verilog/src ”目录中拷贝），“ <DesignName>_timesim.v ”用来作 post-PAR 仿真（必须命名为<DesignName>.v），<TestBenchName.v>用来作仿真用和 <DesignName>_timesim.sdf 用来作时序后注。

- post-map 仿真跟上述类似，post-translate 没有“ *.sdf ”文件，功能仿真除了没有“ *.sdf ”文件外还没有“ glb1.v ”文件；
- 通过点击“ Simulation ”菜单下的“ Start Simulation ”命令把前面讨论过的三个 Xilinx 库文件加入到当前仿真库中；



- 在“Design”栏中选择“glbl”和“<TestbenchName>”，仿真设计；
- 在“Transcript”窗口中输入“add wave*”命令，你就可以到在波形窗口中出现了信号。