



---

# *ispDesignExpert User Manual*

---

*Version 8.0*

*Technical Support Line: 1-800-LATTICE or (408) 428-6414*  
DE-UM Rev 8.0.1

---

## Copyright

This document may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Lattice Semiconductor Corporation.

The software described in this manual is copyrighted and all rights are reserved by Lattice Semiconductor Corporation. Information in this document is subject to change without notice.

The distribution and sale of this product is intended for the use of the original purchaser only and for use only on the computer system specified. Lawful users of this product are hereby licensed only to read the programs on the disks, cassettes, or tapes from their medium into the memory of a computer solely for the purpose of executing them. Unauthorized copying, duplicating, selling, or otherwise distributing this product is a violation of the law.

## Trademarks

The following trademarks are recognized by Lattice Semiconductor Corporation:

Generic Array Logic, ISP, ispANALYZER, ispATE, ispCODE, ispDCD, ispDOWNLOAD, ispDS, ispDS+, ispEXPERT, ispGDS, ispGDX, ispHDL, ispJTAG, ispSmartFlow, ispStarter, ispSTREAM, ispSVF, ispTA, ispTEST, ispTURBO, ispVECTOR, ispVerilog, ispVHDL, ispVM, Latch-Lock, LHDL, pDS+, RFT, and Twin GLB are trademarks of Lattice Semiconductor Corporation.

E<sup>2</sup>CMOS, GAL, ispGAL, ispLSI, pDS, pLSI, Silicon Forest, and UltraMOS are registered trademarks of Lattice Semiconductor Corporation.

SPEEDSearch, Performance Analyst, and DesignDirect are trademarks of Vantis Corporation. Kooldip, MACH, MACHPRO, MACHXL, Monolithic Memories, PAL, PALASM, and Vantis are registered trademarks of Vantis Corporation.

Project Navigator is a trademark of Data I/O Corporation. ABEL-HDL is a registered trademark of Data I/O Corporation.

Microsoft, Windows, and MS-DOS are registered trademarks of Microsoft Corporation.

All other trademarks and registered trademarks are the property of their respective owners.

Lattice Semiconductor Corporation  
5555 NE Moore Ct.  
Hillsboro, OR 97124  
(503) 268-8000

December 1999

---

## Limited Warranty

Lattice Semiconductor Corporation warrants the original purchaser that the Lattice Semiconductor software shall be free from defects in material and workmanship for a period of ninety days from the date of purchase. If a defect covered by this limited warranty occurs during this 90-day warranty period, Lattice Semiconductor will repair or replace the component part at its option free of charge.

This limited warranty does not apply if the defects have been caused by negligence, accident, unreasonable or unintended use, modification, or any causes not related to defective materials or workmanship.

To receive service during the 90-day warranty period, contact Lattice Semiconductor Corporation at:

Phone: 1-800-LATTICE

Fax: (408) 944-8450

E-mail: [applications@latticesemi.com](mailto:applications@latticesemi.com)

If the Lattice Semiconductor support personnel are unable to solve your problem over the phone, we will provide you with instructions on returning your defective software to us. The cost of returning the software to the Lattice Semiconductor Service Center shall be paid by the purchaser.

## Limitations on Warranty

Any applicable implied warranties, including warranties of merchantability and fitness for a particular purpose, are hereby limited to ninety days from the date of purchase and are subject to the conditions set forth herein. In no event shall Lattice Semiconductor Corporation be liable for consequential or incidental damages resulting from the breach of any expressed or implied warranties.

Purchaser's sole remedy for any cause whatsoever, regardless of the form of action, shall be limited to the price paid to Lattice Semiconductor for the Lattice Semiconductor software.

The provisions of this limited warranty are valid in the United States only. Some states do not allow limitations on how long an implied warranty lasts, or exclusion of consequential or incidental damages, so the above limitation or exclusion may not apply to you.

This warranty provides you with specific legal rights. You may have other rights which vary from state to state.

# Table of Contents

---

<b>Preface</b> .....	<b>9</b>
What is in this Manual .....	10
Where to Look for Information .....	10
Documentation Conventions .....	11
Related Documentation .....	12
<b>Chapter 1 Introduction</b> .....	<b>13</b>
Starting the ispDesignExpert Project Navigator .....	14
Creating a New Project .....	15
Giving Your Project a Title .....	16
Importing Project Sources .....	17
Targeting the Design to a Device .....	18
Processing a Design .....	20
View Project Path .....	21
Delete the Project Files .....	22
<b>Chapter 2 Project Management</b> .....	<b>23</b>
About the Project Navigator .....	24
Project Navigator Screen .....	25
Title Bar .....	26
Menu Bar .....	26
Toolbar .....	26
Sources Window .....	26
Source Hierarchy .....	28
Processes Window .....	28
Process Flows .....	29
Describing a Project .....	29
Targeting a Device .....	29
Device Selection .....	29
Specifying Project Files .....	30
Design Hierarchy .....	30
Tips for Defining Projects .....	30
Importing an Existing Source .....	31
Where the Source File is Placed in the Project Navigator .....	32
Creating a New Source .....	32
Modifying a Source .....	33
Schematic Editor .....	33
Text Editor .....	33

Removing a Source .....	34
Processing a Design .....	34
Forcing a Process to Run .....	34
Changing the Environment and Configuration .....	34
Cleaning Up a Project .....	36
Saving a Project .....	36
What is Saved? .....	37
Tips for Saving and Naming Projects .....	37
Reserved File Names .....	37
<b>Chapter 3 Design Entry .....</b>	<b>39</b>
ABEL-HDL Design Entry .....	40
Add an ABEL-HDL Module to Your Design .....	40
The Language Editor .....	43
Schematic Design Entry .....	44
Add a Schematic to Your Design .....	44
Add Design Attributes .....	48
Create a Symbol .....	50
Add Design Control Properties (ispLSI Designs Only) .....	51
VHDL Design .....	53
Add a VHDL Module to Your Design .....	53
Create an EDIF File for ispLSI .....	56
Verilog HDL Design .....	57
Add a Verilog HDL Module to Your Design .....	57
Create an EDIF File .....	59
EDIF Design .....	61
MACH/PAL Properties in the EDIF File .....	64
Property List .....	64
Import Mechanism for MACH/PAL .....	66
Hierarchical Design .....	67
Overview of Hierarchical Design .....	67
Advantages of Hierarchical Design .....	67
Hierarchy vs. Sheets in Schematics .....	68
Approaches to Hierarchical Design .....	68
Hierarchical ABEL-HDL Design .....	69
ABEL-HDL Hierarchy Examples .....	70
Hierarchical Schematic Design .....	73
Schematic Hierarchy Examples .....	74
Hierarchical VHDL Design .....	77
Schematic/VHDL Hierarchy Example .....	78
Hierarchical Verilog HDL Design .....	80
Schematic/Verilog HDL Hierarchy Example .....	81
Hierarchical Design Considerations .....	83
Hierarchical Design Structure .....	83
Hierarchical Naming .....	84
Nets in the Hierarchy .....	85
Automatic Aliasing of Nets .....	86
Mixed Entry Design .....	87

Schematic and ABEL-HDL Mixed Entry . . . . .	87
Create an ABEL-HDL Source File. . . . .	91
Compile the ABEL-HDL Source File . . . . .	92
Schematic and VHDL Mixed Entry . . . . .	94
Create a VHDL Source File. . . . .	94
Schematic and Verilog HDL Mixed Entry . . . . .	95
ispLSI Design Attributes. . . . .	96
Assigning ispLSI Design Attributes in Project Sources . . . . .	96
In Schematics . . . . .	96
In ABEL-HDL. . . . .	98
ispLSI Attribute Processing . . . . .	99
Precedence of Design Attributes . . . . .	101
MACH Design Attributes . . . . .	102
Assigning MACH Design Attributes in Project Sources . . . . .	102
In Schematics . . . . .	102
In ABEL-HDL. . . . .	103
<b>Chapter 4 Design Implementation . . . . .</b>	<b>105</b>
Constraint Manager for ispLSI Designs . . . . .	106
Main Window . . . . .	106
Design Browser. . . . .	107
Pin Attributes Table. . . . .	107
Net Attributes Table . . . . .	108
Symbol Attributes Table . . . . .	108
Assign Attribute Values. . . . .	108
Constraint Editor for MACH Designs . . . . .	109
Main Window . . . . .	109
Location Assignment. . . . .	110
Group Assignment . . . . .	111
Pin Reservation. . . . .	112
JEDEC File Options . . . . .	112
Assigning Power Level . . . . .	113
Output Slew Rate Control . . . . .	114
Pin Locking for GAL/PAL Devices . . . . .	115
VHDL Designs . . . . .	115
Syntax. . . . .	115
Verilog HDL Designs . . . . .	116
Syntax for Synplify . . . . .	116
Syntax for LeonardoSpectrum . . . . .	117
ispLSI Compiler Properties . . . . .	118
Settings. . . . .	118
Strategy. . . . .	120
Effort . . . . .	120
Use Global Reset . . . . .	120
XOR . . . . .	120
Maximum GLB Inputs . . . . .	121
Maximum GLB Outputs. . . . .	121
Free All Pin Locks . . . . .	122

Ignore Reserved Pins	122
Use Extended Routing	122
Carry Pin Direction	123
Case Sensitive	123
Timing Analyzer	123
Parameter File	123
Interfaces Dialog Box	124
Advanced Compiler Settings Dialog Box	124
Minimize GLB Levels For All Paths	124
Use Internal Tristate IO Driver	124
BFM Packing	125
Single PT Function Packing for Routability	125
Device Options	125
Security	125
ISP	125
ISP Except Y2	125
Y1 as RESET	126
TOE_AS_IO	126
LowPower	126
UES	126
Data Type	127
Signature Size	127
MACH Global Optimization Options	128
Global Optimization	128
Pack Design	128
Spread Design	128
Advanced Options	129
Logic Synthesis	129
Boolean Logic Synthesis	129
D/T Synthesis	130
Set/Reset Don't Care	130
Node Collapsing	130
Product Term Collapsing	130
Product Term Equation Splitting	130
Utilization Options	130
Maximum % of Macrocells per Block Used	131
Maximum % of Block Inputs Used	131
Compiling/Fitting the ispLSI/GAL Design	132
Compiling/Fitting the MACH/PAL Designs	134
<b>Chapter 5 Design Verification</b>	<b>136</b>
Lattice Logic Simulator for ispLSI/GAL	137
Overview	137
Creating Test Stimulus for Lattice Logic Simulator	137
Running Functional/Timing Simulation	137
Showing the Waveforms in the Waveform Viewer	139
HDL Cross Probing for ispLSI/GAL Designs	140
Simulation Log	140

---

Running Stand-alone Lattice Logic Simulator .....	141
Equation Simulator for MACH/PAL .....	144
Overview .....	144
Creating Test Vectors for Equation Simulator .....	144
Running the Equation Simulation .....	144
The Simulator Model .....	145
Controlling the Simulation Report .....	146
Displaying the Waveforms in the Waveform Viewer .....	147
JEDEC Simulation .....	148
Simulating a JEDEC File .....	148
Viewing the Simulation Waveform .....	148
ModelSim Simulator .....	149
Generating Test Stimulus .....	149
Manually Create VHDL Test Bench or Verilog Test Fixture .....	150
Create VHDL Test Bench or Verilog Test Fixture Using the Template .....	150
Export VHDL Test Bench or Verilog Test Fixture from Waveform Editor .....	151
Performing the VHDL/Verilog Functional/Timing Simulation .....	152
<b>Chapter 6 Timing Analysis .....</b>	<b>153</b>
Timing Analysis Overview .....	154
Timing Analyzer for ispLSI .....	154
Timing Explorer .....	154
Signal Navigator .....	155
Pop-up Menus from the Signal Navigator .....	156
Timing Explorer Tables .....	157
Pop-Up Menus from the Timing Tables .....	159
Timing Path Report .....	160
Performance Analyst for MACH Designs .....	161
Analysis Types .....	161
fMAX .....	161
tSU .....	162
tPD .....	164
tCO .....	164
tOE .....	165
tCOE .....	165
Running Timing Analysis .....	166
Running Timing Analysis in Batch Mode .....	169



# ***Preface***

---

The ispDesignExpert software is used to create designs to program ispLSI<sup>®</sup>, MACH, GAL, and PAL devices from Lattice Semiconductor Corporation (LSC).

This manual describes the Project Navigator<sup>™</sup>, the Graphical User Interface (GUI) for the ispDesignExpert software. The design procedures provided in this manual are intended to help you understand how to use the ispDesignExpert software to create designs for different devices.

## What is in this Manual

This manual contains the following information:

- Overview of the software
- Discussion on project management procedure
- Introduction of the supported design entries
- Design considerations for design entries
- Applying design attributes
- Usage of the Constraint Manger and the Constraint Editor
- Design verification in different simulation environment
- Running timing analysis

## Where to Look for Information

**Chapter 1, Introduction** – Provides a brief description of all ispDesignExpert tools.

**Chapter 2, Project Management** – Covers general information about the Project Navigator and its user interface.

**Chapter 3, Design Entry** – Describes the supported design entries in the ispDesignExpert and some design rules.




**Chapter 4, Design Implementation** – Presents the method of adding compiler control options or device control options to your design.

**Chapter 5, Design Verification** – Contains descriptions for the Lattice Logic Simulator, Equation Simulator, and ModelSim Simulator.

**Chapter 6, Timing Analysis** – Illustrates how to use the Timing Analyzer and the Performance Analyst.

# Documentation Conventions

This user manual follows the typographic conventions listed here:

Convention	Definition and Usage
<i>Italics</i>	<p>Italicized text represents variable input. For example:</p> <p style="text-align: center;"><i>design.syn</i></p> <p>This means you must replace <i>project</i> with the file name you used for all the files relevant to your design.</p> <p>Valuable information may be italicized for emphasis. Book titles also appear in italics.</p> <p>The beginning of a procedure appears in italics. For example:</p> <p style="text-align: center;"><i>To import an ABEL-HDL module to your design:</i></p>
<b>Bold</b>	<p>Valuable information may be boldfaced for emphasis. Commands are shown in boldface. For example:</p> <p>1. In the Schematic Editor, select <b>File</b> ⇒ <b>Generate Symbol</b>.</p>
Courier Font	<p>Monospaced (Courier) font indicates file and directory names and text that the system displays. For example:</p> <p style="text-align: center;">In the ..\examples\ispLSI_GAL\clock directory, select ...</p>
<b>Bold Courier</b>	<p>Bold Courier font indicates text you type in response to system prompts. For example:</p> <p style="text-align: center;"><b>SET YBUS [Y0..Y6];</b></p>
...	<p>Vertical bars indicate options that are mutually exclusive; you can select only one. For example:</p> <p style="text-align: center;">INPUT OUTPUT BIDI</p>
“Quotes”	<p>Titles of chapters or sections in chapters in this manual are shown in quotation marks. For example:</p> <p style="text-align: center;">See Chapter 1, “Introduction.”</p>
 <b>NOTE</b>	Indicates a special note .
 <b>CAUTION</b>	<b>Indicates a situation that could cause loss of data or other problems.</b>
 <b>TIP</b>	Indicates a special hint that makes using the software easier.
⇒	<p>Indicates a menu option leading to a submenu option. For example:</p> <p style="text-align: center;"><b>View ⇒ Toolbar</b></p>

## Related Documentation

In addition to this manual, you might find the following reference material helpful:

- *ispDesignExpert Getting Started Manual*
- *ispDesignExpert Release Notes*
- *Design Verification Tools User Manual*
- *Schematic Entry User Manual*
- *ABEL Design Manual*
- *ABEL-HDL Reference Manual*
- *ispLSI Macro Library Reference Manual*
- *5K/8K Macro Library Supplement*
- *ispEXPERT Compiler User Manual*
- *ISP Daisy Chain Download User Manual*
- *ispDOWNLOAD Cable Reference Manual*
- *VHDL and Verilog Simulation User Manual*
- *Synplicity Synplify User Guide*

These manuals provide technical specifications for the ispDesignExpert software. They give helpful information on device use and design development. They are located in the manuals directory on the CD-ROM. The ispDesignExpert Help menu also provides access to the manuals.

# Chapter 1 *Introduction*

---

The ispDesignExpert software provides support for the ispLSI, GAL, MACH, and PAL device families under the ispDesignExpert Project Navigator. The software contains all executable, libraries, and device support lists necessary to configure the Project Navigator for design entry, timing simulation, and fuse map creation for these devices.

The Project Navigator is the main interface for ispDesignExpert. Using the Project Navigator you can create a project that represents your design, run processes on the sources in your project, compile your design, simulate your design, and create JEDEC files that can be used to physically implement your design into a target device. Furthermore, you can access other ispDesignExpert tools from the Project Navigator.

## Starting the ispDesignExpert Project Navigator

To start the Project Navigator, choose the **Start** menu. In the **Start** menu, select the menu item **Programs** ⇒ **Lattice Semiconductor** ⇒ **ispDesignExpert System**. The Project Navigator window appears (Figure 1-1).

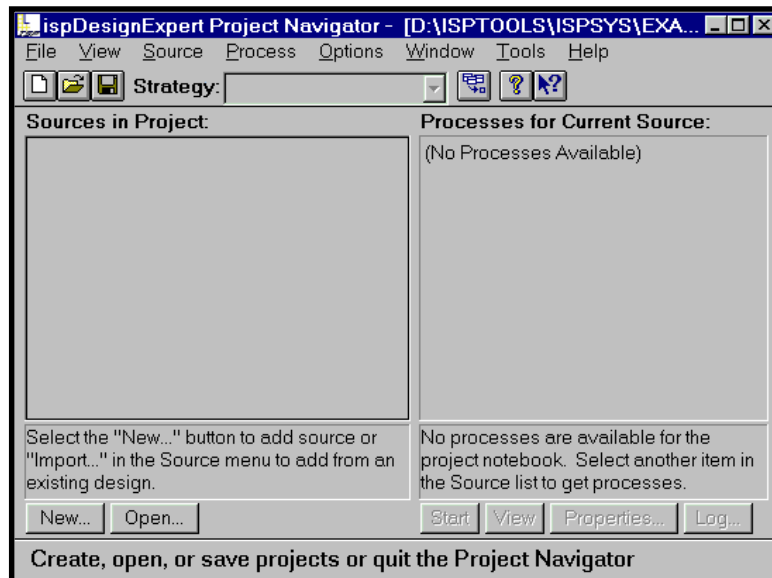


Figure 1-1. ispDesignExpert Project Navigator with an Open Project

The ispDesignExpert is a design entry tool. You must create or open a project to begin working. Only basic menus are available until you open or create a project. Once you do, all the menu options display.

## Creating a New Project

After starting the Project Navigator, you can create a new project. Design projects are made of one or more sources which can be ABEL-HDL files, VHDL files, Verilog HDL files, schematics, test vector files, waveform stimulus, VHDL test bench files, Verilog test fixture files, or documentation files.

A new project can be created by selecting existing ABEL-HDL, VHDL, Verilog HDL files, or Schematics and using the Project Navigator menus to import the existing source(s); or, by creating new sources from within the Project Navigator. You need to specify the type of the project by choosing Schematic/ABEL, Schematic/VHDL, Schematic/Verilog HDL, or EDIF from the Project Type field.

To create a new project:

1. Choose **File** ⇒ **New Project** to open the Create New Project dialog box.
2. Go to the directory in which you want to place your project files and then type a name for your project. The default project name is *untitled.syn*. In this example, go to the `<drive>:\<isptools_path>\<ispsys_path>\examples\test` directory.
3. Create a new folder called `test`. Then, go into this folder and name the project file `test.syn`. Specify the type of the new project by choosing Schematic/ABEL, Schematic/VHDL, Schematic/Verilog HDL, or EDIF from the Project Type list (Figure 1-2).

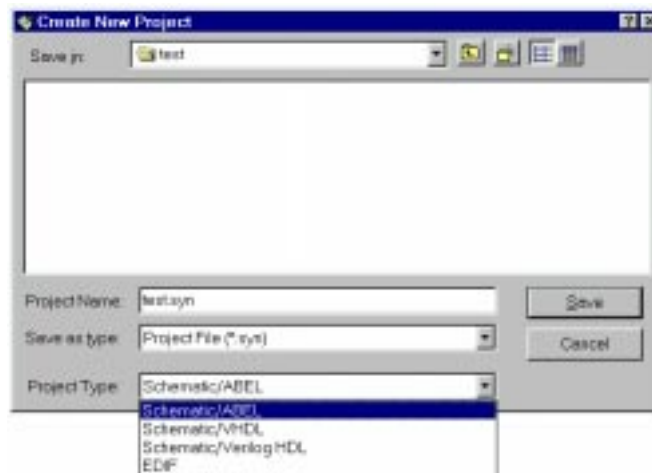


Figure 1-2. Create New Project Dialog Box

4. Click **Save**. The untitled project appears in the Sources window of the Project Navigator.

## Giving Your Project a Title

The default title for a new project is “Untitled.” You can create a title for the project with many characters. The title can contain spaces and any other keyboard character except tabs and returns.

*To give your project a title:*

1. In the Sources window, double click the title of the project `Untitled` to open the Project Properties dialog box.

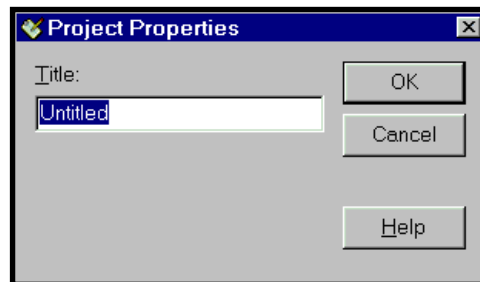


Figure 1-3. Project Properties Dialog Box

2. Type the name `Clock` for the title of your project and then click **OK**. The new project title appears in the Sources window.

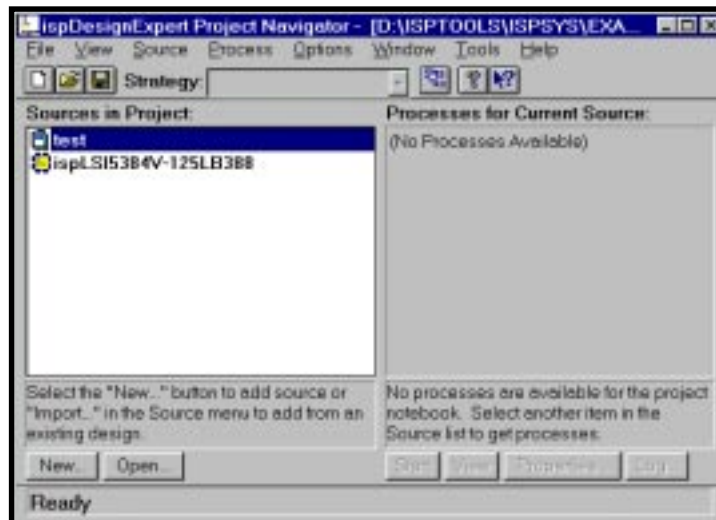


Figure 1-4. Project Navigator Window with Project Title Entered



## Importing Project Sources

You define the logic in a design by importing sources into the project. These sources can include several types, such as ABEL-HDL descriptions, VHDL descriptions, Schematics, EDIF netlists, or simulation files.

To import existing sources into your project:

1. Choose **Source** ⇒ **Import** to open the Import File dialog box (Figure 1-5).



Figure 1-5. Import File Dialog Box

2. In the `..\examples\ispLSI_GAL\clock` directory, select all of the files (`clock.wdl`, `clocktop.abv`, `clocktop.sch`, `control.sch`, `hours.abl`, `minutes.abl`, `presclr.abl`, `secctr.abl`, and `sseg.abl`). These will be the source files for your new project (Figure 1-6).



Figure 1-6. Import File Dialog Box - Select Existing Source Files

3. When you are finished selecting these sources, click **OK**. The selected sources appear in the Project Navigator Sources window (Figure 1-7).

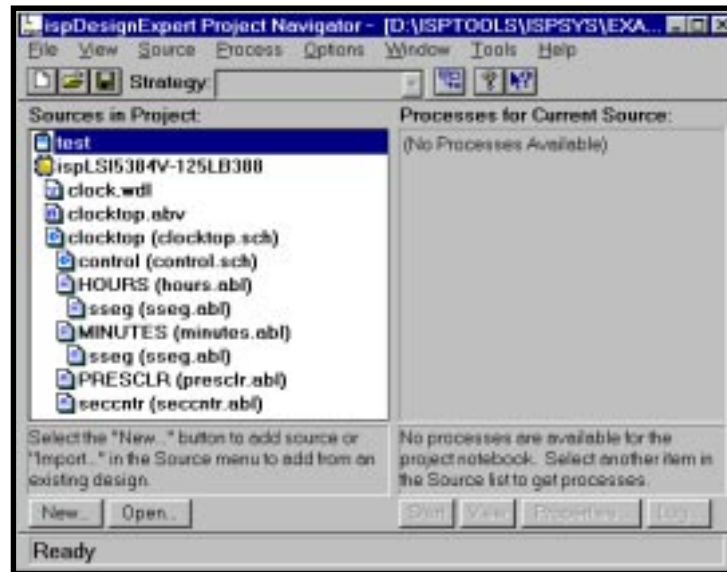


Figure 1-7. Project Navigator with Sources Imported

## Targeting the Design to a Device

The target device is the device in which you intend to implement your design. When you open a new project, the default target device is **ispLSI5384V-125LB388**. If you do not want to use the default device, you can target the design for a specific supported device.

*To target a specific device:*

1. In the Sources window, double click the default device icon to open the Device Selector dialog box (Figure 1-8).

The Device Selector contains all the supported devices and their options.

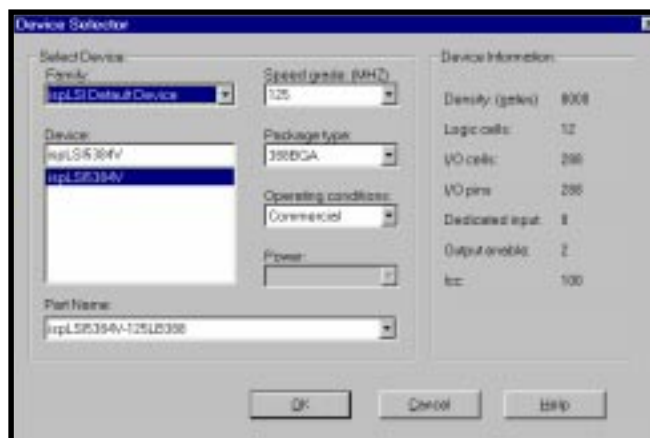


Figure 1-8. Device Selector Dialog Box

2. Under Select Device, select the ispLSI 1K device family and ispLSI1032E-100LJ84 within that family. Accept the default options.



Figure 1-9. Device Selector Dialog Box with Target Device Selected

- When you are finished, click **OK**. The Confirm change dialog box appears (Figure 1-10).

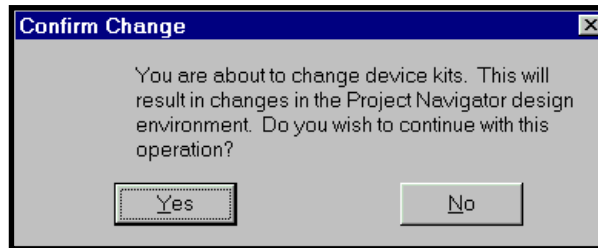


Figure 1-10. Confirm Change Dialog Box

- Click **Yes**. The specified target device appears in the Sources window. Also, all the processes for that device are shown in the Processes window (Figure 1-11).

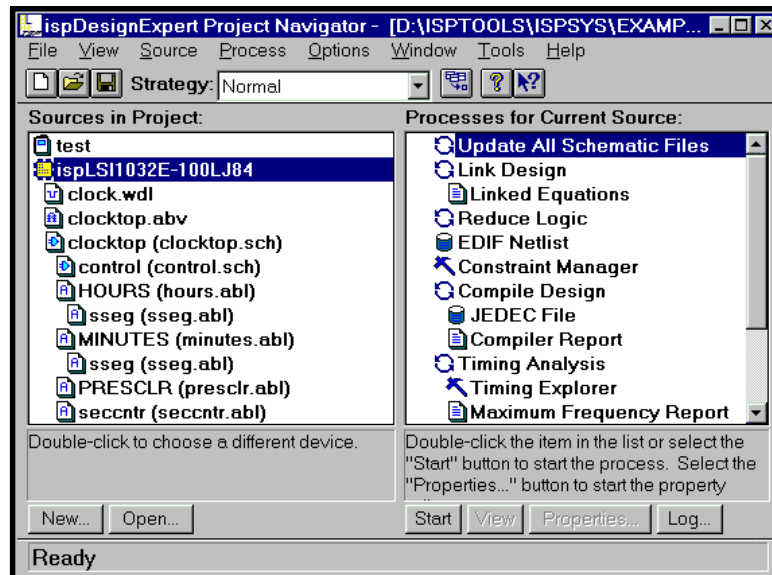


Figure 1-11. Project Navigator with Target Device Selected

## Processing a Design

A process is a specific task in the overall processing of a source or project. Typical processing tasks include creating a netlist, compiling the design, reducing and synthesizing the logic, fitting the design, and analyzing the timing. You can run a process on a single source, or on the entire project. You can view the available processes for a source by selecting the source, and the processes for that source appear in the Processes window.

In general, you start a process by selecting a source and then choosing Start from the Process menu.

*To fit the design:*

1. In the Sources window, select the Target Device.
2. In the Processes window, select the Compile Design or Fit Design process (Figure 1-12).

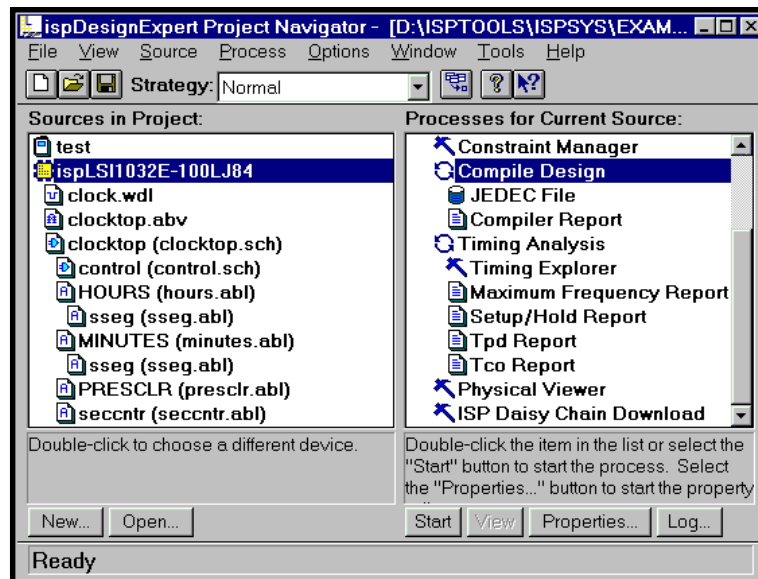


Figure 1-12. Project Navigator - Compile Design Process

3. Start the processing in one of these ways:
  - Choose the **Process** ⇒ **Start** menu item.
  - Click the **Start** button at the bottom of the Project Navigator window.
  - Double click the process label (Compile Design or Fit Design).
4. The Project Navigator processes the design up to the specific step highlighted in the Processes window. In this case, it is Compile Design (Figure 1-13).

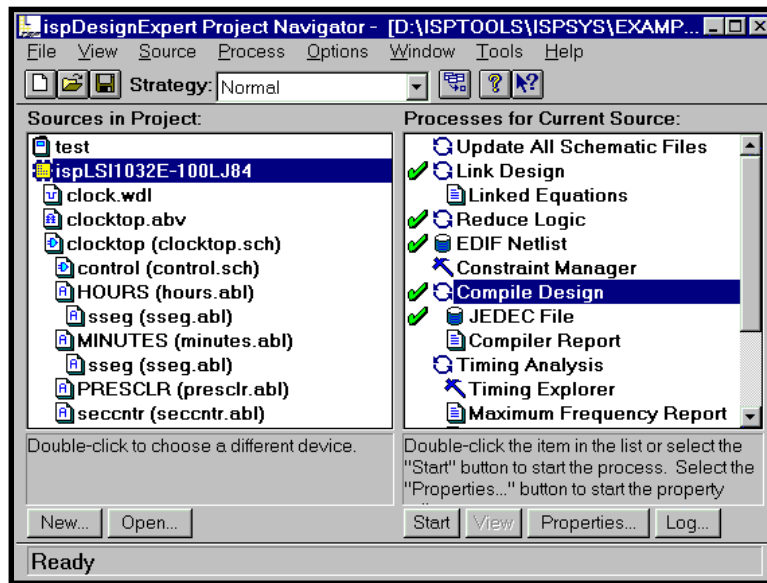



Figure 1-13. Project Navigator - Complete the Compile Design Process

 **NOTE** Yellow exclamation beside the process points indicate that warnings were generated. Red Xs indicate that errors were encountered. The warning or error is described in the auto-make log file displayed in the Report Viewer. Green check marks indicate the process completed successfully.

## View Project Path

You can use the **File** ⇒ **Full Project Path** menu item of the Project Navigator to view your Project Path from the prompt Full Project Path dialog box (Figure 1-14).



Figure 1-14. Full Project Path Dialog Box

## Delete the Project Files

After you have completed this quick tutorial, you can delete the project files from your computer.

*To delete the project files:*

1. Choose the **File** ⇒ **Close Project** menu item.
2. Using the Windows Explorer or similar tool, go to the `..\examples` directory and delete the `test` folder.

## Chapter 2 *Project Management*

---

The Project Navigator is the primary interface for the ispDesignExpert and provides an integrated environment for managing the project elements and processes.

This chapter contains a detailed explanation of how to use the Project Navigator to interact with the software; from creating design files through downloading the design. The design procedures are described with examples of the menus and dialog boxes needed to perform these tasks.

This chapters covers information on the following topics:

- About the Project Manager
- Describing a Project
- Changing the Environment and Configuration
- Cleaning up a Project
- Saving a Project

## About the Project Navigator

Using the Project Navigator you can select all the source components for a design, as well as specification documents and test files, and assemble them into one project file. The Project Navigator also helps you keep track of all the processing steps necessary to move the design from the conceptual stage through to implementation of a programmable device. When you switch the target device, the Project Navigator automatically changes the design flow and processes to one that is appropriate for the new target device.

The Project Navigator also associates all the tools needed for a particular design step. For example, for HDL source files the Project Navigator associates the Text Editor and HDL synthesis tools, for schematic sources the Project Navigator associates the Schematic Editor, Symbol Editor, Hierarchy Navigator, Library Manager tools and schematic compiling tools, and for waveform stimulus source files the Project Navigator associates Waveform Editor, Waveform Viewer, and Lattice Logic Simulator. Furthermore, the Project Navigator keeps track of preferences for you, automatically setting options that work for most systems until you want to modify the options yourself.

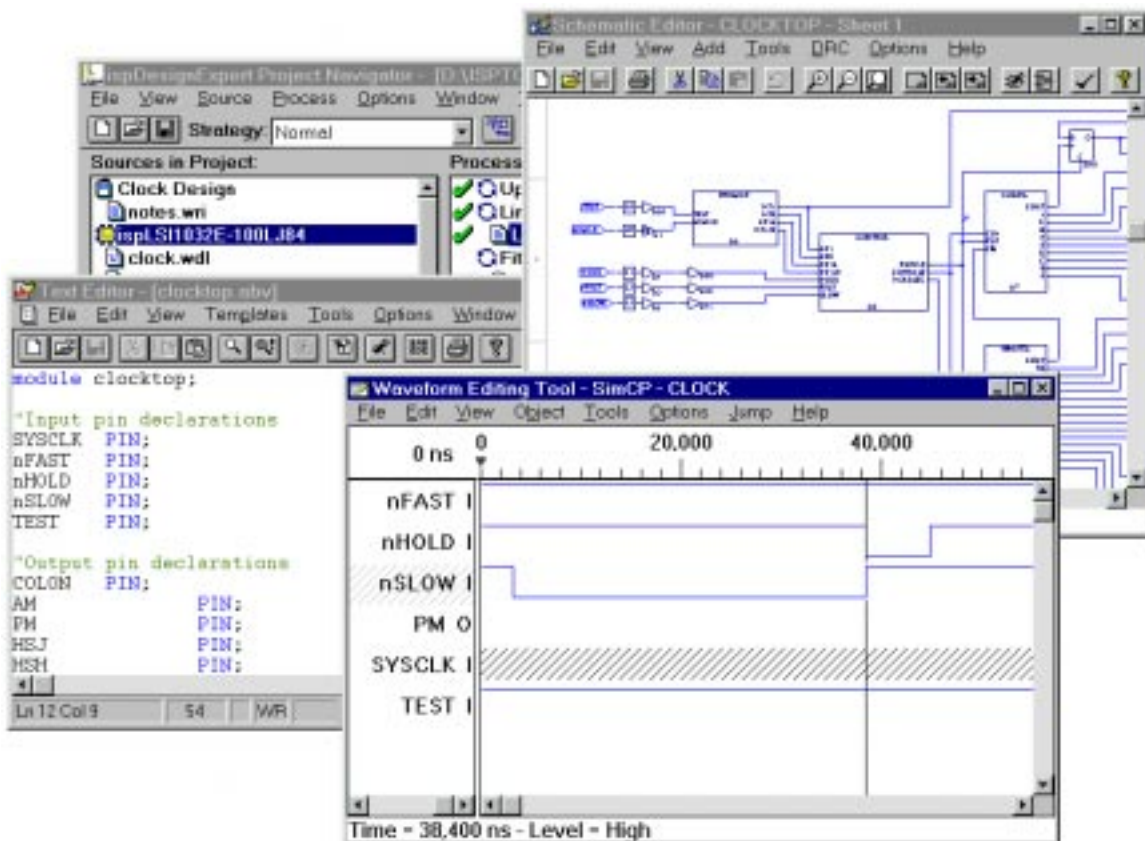


Figure 2-1. Project Navigator and Other Tools



## Project Navigator Screen

Once the ispDesignExpert software is invoked, the Project Navigator is displayed on the screen as shown in Figure 2-2.

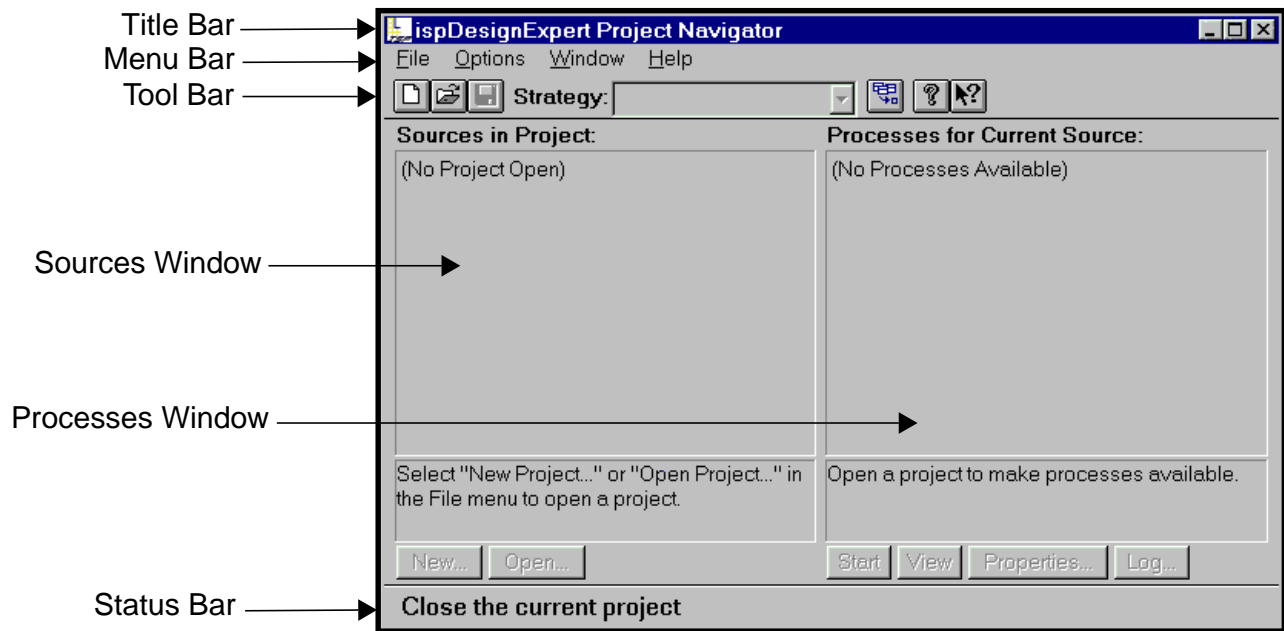


Figure 2-2. Project Navigator Screen

Your files are listed in the Sources Window, while the Processes connected to your files are shown in the Processes Window. If no project is open, the Sources and Processes windows would be empty as shown in Figure 2-2.

## Title Bar

The Title Bar displays the ispDesignExpert Project Navigator name and the current project name (if any).

## Menu Bar

The Menu Bar contains menu topics related to the functions used to create the design. Each menu item has one character underlined. This character is used to execute the command from the keyboard. Execute commands from the keyboard by pressing the **Alt** key and the letter underscored in the menu (called the hot key). For example, to execute **File** ⇒ **Open Project**, press and hold the **Alt** key and press **F** and **O**. The ellipsis (...) following some of the menu items in the pull-down menus indicate that a dialog box appears when this menu item is selected.

## Toolbar














The toolbar contains functions found in the menu bar. The toolbar icons provide a quick and easy way to access the most commonly used features of the ispDesignExpert. The toolbar icons and their equivalent menu bar functions are discussed in the online help. You can use **View** ⇒ **Toolbar** to display or hide the toolbar.

## Sources Window

The Sources window (on the left side of the Project Navigator) shows all the design files associated with a project, listed in their logical, hierarchical order. Each object in the list is identified with an icon. For example, at the top of the Sources window is the Project Notebook; it is denoted with the **Project** icon. (To see all the objects in the project, use the scroll bar at the right edge of the Sources window to move up and down in the list.)

There are several kinds of design sources in ispDesignExpert, including schematics, ABEL-HDL modules, VHDL modules, Verilog HDL modules, EDIF netlists, and couple test stimulus files for simulation. Listed below (Table 2-1) are the acceptable sources for a project. When you begin a new project, there will be no sources except for the Project Notebook.

Table 2-1. Acceptable ispDesignExpert Project Source Files

Source Type	Icon	File Extension
Project Notebook		.syn
Target Device		None
Document Source		.wri, .doc, .txt, .xls, .hlp, .prp, .par (or any extension not recognized by the Project Navigator)
Schematic Source		.sch
ABEL-HDL		.abl
ABEL-HDL Test Vector		.abv
VHDL		.vhd
Verilog HDL		.v
EDIF Netlist		.ed*
Waveform Stimulus		.wdl
VHDL test bench		.vhd
Verilog HDL test fixture		.tf
Undefined or incorrect source reference		Any

## Source Hierarchy

One source file in a project is the top-level source for the design, which can be an HDL module or schematic. The top-level source defines the inputs and outputs that will be mapped into the device, and references the logic descriptions contained in lower-level sources. The referencing of another source is called **instantiation**. Lower-level sources can also instantiate sources to build as many levels of logic as necessary to describe your design.



### **NOTE**





If you build a project with a single source, that source is automatically the top-level source.

## Processes Window

The Processes window (on the right side of the Project Navigator) shows all the processing tasks that apply to whatever object or file is highlighted in the Sources window (on the left side). A processing task includes: netlisting, compiling, logic reduction, logic synthesis, placement and routing, functional and timing simulation – in other words, any step along the way from design entry to implemented Lattice Semiconductor devices.

The table below (Table 2-2) lists the ispDesignExpert process types and shows its corresponding icons.

Table 2-2. ispDesignExpert Process Types

Process Type	Icon
Process	
Report	
Output File	
Tools	

## Process Flows

One of the most powerful features of the ispDesignExpert is that the Project Navigator is context-sensitive and automatically adjusts the processes for you depending on what you want to do.

The steps in the Processes window are context-sensitive in two ways. First, the process flow changes depending on what kind of source file is highlighted in the Sources window (source-level flow). Second, the processing for a given file changes depending on the target device you have chosen (project-level flow).

## Describing a Project

You describe a project by targeting a particular device implementation, and by specifying the project files that will represent the design.

### Targeting a Device

The Project Navigator lets you target a design to a specific device at any time during the design process. If you do not know the specific device, you can target the ispLSI Default Device, ispLSI5384V-125LB388.

### Device Selection

Because the Project Navigator is context-sensitive, when you choose the ispLSI Default Device, processes allowed for the ispLSI devices are shown. If you choose a MACH, PAL, or GAL device family, the processes change to reflect your selection.

*To target a specific device or device family:*

1. In the Sources window, double-click the Target Device icon to open the Device Selector dialog box (Figure 2-3). The Device Selector dialog box contains all available devices and their options.

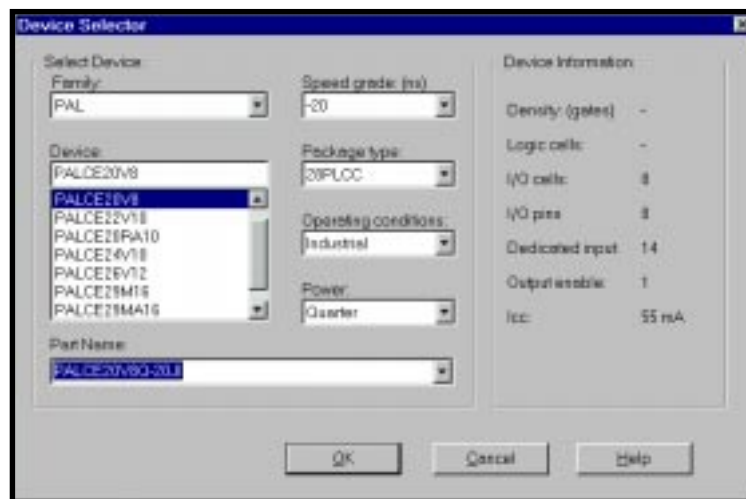


Figure 2-3. Device Selector Dialog Box

2. Under **Select Device**, select a device family and a specific device within that family. Then choose the options you want for that device.
3. When you are finished, click **OK**. The **Confirm Change** dialog box appears.
4. Click **Yes**. The specified target device appears in the **Sources** window.

## Specifying Project Files

You define a project by importing existing sources into the project. You can also create new sources or modify existing sources using an editor.

## Design Hierarchy

A single module design is a flat design in which there is only one source describing the entire design, such as a single schematic. You can also have a test file, such as ABEL-HDL test vectors, in a flat design because all processes, such as functional simulation, in the flat design involve the entire design.

When designs can be broken into multiple levels, this is called hierarchical design. *ispDesignExpert* supports full hierarchical design, allowing you to create a design that is divided into multiple levels, either to clarify its function or permit the reuse of functional blocks.

## Tips for Defining Projects

Use the following guidelines when saving and naming source files and your project:

- Understand and use the different methods for hierarchical design.
- Avoid using ABEL-HDL or EDIF reserved words for module and signal names in any of your source files.
- Avoid saving a project that has the same base file name as one of its sources. If a source and project have the same base name, you may have problems with the Project Navigator's Auto-make feature.
- Each source must have a unique name in the project. You cannot have two different sources with the same name. You can use the same source many times in a design by instantiating the source, but two different sources with the same name can cause problems with the hierarchy. For example, do not use an ABEL-HDL source called `compare` and a schematic source also called `compare`.
- The file name and module name of an EDIF file should be the same.

## Importing an Existing Source

To import an existing source:

1. Choose **Source** ⇒ **Import** to open the Import File dialog box (Figure 2-4).



Figure 2-4. Import File Dialog Box

2. Find the source file you want to import. You can change the type of file that is displayed in the Files of type field.
3. When you are finished selecting the source, click **Open**. ispDesignExpert imports the selected sources into the project and displays them in the Sources window.

Depending on the source type you are importing, you may be prompted to provide additional information in the Import Source Type dialog box. For example, if you choose a `.vhd` file, you are prompted to choose to import it as a VHDL Module or a VHDL Test Bench in the Import Source Type dialog box.

### ❖ TIP

You can import the lower-level sources before or after importing upper-level sources. However, if you import a source that has links to lower-level sources and the lower-level sources are not already part of your project, you will get undefined sources in your project until you import the missing lower-level sources. Also, import test files that are to be associated with other sources after importing or creating the other source.

## Where the Source File is Placed in the Project Navigator

After you import a source into the Project Navigator, it appears in the Sources window. However, where the source appears in the window depends on the following:

- If the imported source is a documentation file or a file type not recognized as a logic description or test file, the source appears between the Notebook icon and Targeted Device icon.
- If the source is a logic description, the source is placed in alphabetical order for each level of hierarchy following the project notebook and the targeted device. For example, if the source is called `multiplx` and the top-level source, a schematic called `myChip`, contains a functional block called `multiplx`, the source is placed underneath `myChip` in the Sources window.
- If the source is an ABEL test vector file, the source is placed beneath the Targeted Device icon.

## Creating a New Source

To create a new source:

1. Choose **Source** ⇒ **New** to open the New Source dialog box.

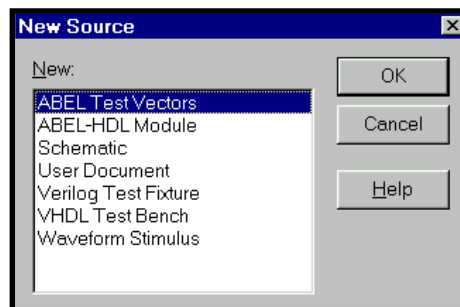


Figure 2-5. New Source Dialog Box

2. In the dialog box, select the type of source you want to create and then click **OK**.  
The Project Navigator starts an editor that you can use to enter the information for your new source. For ABEL-HDL, VHDL, Verilog HDL, ABEL test vector, VHDL test bench, and Verilog test fixture sources, the Text Editor is started. For schematic sources, the Schematic Editor is started. For waveform stimulus sources, the Waveform Editor is started.
3. In the editor, create a source.



## Modifying a Source

You can edit any of the sources that make up your project by double clicking on them to open the corresponding editor.



### **NOTE**

In order for Windows file associations to work properly with ispDesignExpert, in the Project Navigator choose **Options** ⇒ **Environment**. Then under Window Settings, select Use File Associations.

## Schematic Editor

Use the ispDesignExpert Schematic Editor to edit a schematic source. You can open the Schematic Editor in these ways:

- In the Project Navigator, choose **Window** ⇒ **Schematic Editor**.
- Double-click the schematic source name.
- Select the schematic source and choose **Source** ⇒ **Open**.
- Select the schematic source and then click **Open** at the bottom of the Project Navigator.

The Symbol Editor, the Hierarchy Navigator, and/or the Hierarchy Browser work in conjunction with the Schematic Editor. You can open the Symbol Editor from the Schematic Editor toolbar, or on the Project Navigator **Window** menu.

## Text Editor

The Text Editor provides several macros and templates to help you enter and edit test stimulus and behavioral modules written in ABEL-HDL, VHDL, or Verilog HDL. You can open the Text Editor in these ways:

- In the Project Navigator, choose **Window** ⇒ **Text Editor**.
- Double-click the text source name.
- Select the text source and choose **Source** ⇒ **Open**.
- Select the text source and click **Open** at the bottom of the Project Navigator.

Also, you can use any ASCII editor to edit behavioral modules, ABEL test vectors, VHDL test benches, and Verilog test fixtures. Then, you can import them into your project using drag-and-drop from the Windows Explorer, or by choosing **Source** ⇒ **Import**.

## Removing a Source

Sometimes you may want to remove a source from a project. To remove a source from a project you must use the ispDesignExpert **Remove** command. You cannot remove a source using the Window's **Delete** command.

1. In the Sources window, select the source that you want to remove.
2. Choose **Source** ⇒ **Remove**. ispDesignExpert removes the source from the project.

## Processing a Design

A process is a specific task in the overall processing of a source or project. Typical processing tasks include netlisting, compiling, logic reduction, logic synthesis, fitting the design, and simulation. To view the available processes for a source, select the source. Then ispDesignExpert displays the processes for that source in the Processes window.

## Forcing a Process to Run

If the process is up-to-date (indicated by a check mark to the left of the process), it will not run again. You can, however, force a process to run by doing the following.

- Choose **Process** ⇒ **Force** to start the highlighted process and run all the steps, even if the process is up-to-date. When the process is finished running, the Project Navigator displays the selected file, if applicable. This command allows you to temporarily override the Process Force settings in **Options** ⇒ **Environment** to start the highlighted process.
- Choose **Process** ⇒ **Force One Level** to start the highlighted process.

## Changing the Environment and Configuration

You can set many environment variables and change settings for the Project Navigator and programs started from within the Project Navigator. You can even add menus to access other Windows programs.

Changes to the environment can be made by:

- Editing `.ini` files used by the Project Navigator and other programs.
- Choosing **Options** ⇒ **Environment** from the Project Navigator menus. The Environment Options dialog box appears.

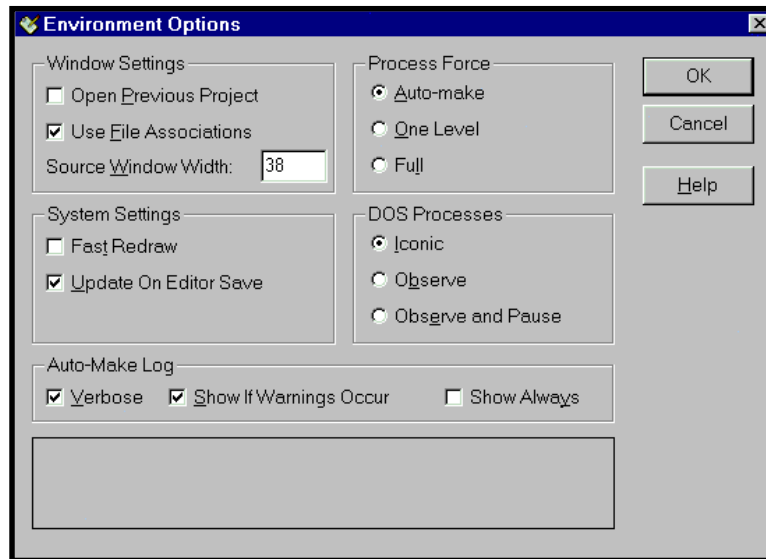


Figure 2-6. Environment Options Dialog Box

There are three options in the Window Settings area.

- If the “Open Previous Project” option is checked, the project that was opened when the Project Navigator was last running is opened when you start the ispDesignExpert. If the project no longer exists or cannot be found, this option is ignored and the Project Navigator opens without loading a project.
- If the “Use File Associations” option is enabled (checked), document sources use Windows file associations for non-Project Navigator sources.
- The Source Window Width, in number of characters, of the Sources window. The Project Navigator uses this value and the average character width of the currently-selected font to determine the width of the Sources window. The recommended available range is from 20 to 60 characters.

Three mutually-exclusive options for Process Force are “Auto-make,” “One Level,” and “Full.”

- If you choose “Auto-make” option, the Project Navigator will use the auto-make instructions when a process is started.
- If you choose “One Level” option, the Project Navigator always runs the last step for a process, regardless of the auto-make state.
- If you enable “Full” option, the Project Navigator always runs all steps for a process, regardless of the auto-make state.

Two options are set in the System Settings area.

- If “Fast Redraw” is enabled, the Project Navigator display is altered to speed up screen redrawing. For example, 3-D controls are displayed as 2-D. If this option is disabled, normal screen drawing is performed.
- The “Update on Editor Save” option specifies whether the Project Navigator should update the Sources window hierarchy when you save a source file from the Text Editor or the Schematic Editor.

In the DOS Processes area, “Iconic,” “Observe,” and “Observe and Pause” options are set.

- The “Iconic” option is enabled to run all DOS processes in an iconic DOS window.
- The “Observe” option enables you to run all DOS processes in a visible window, which is useful for debugging. If this option is not checked, DOS engines are run minimized (as icons).
- The “Observe and Pause” option allows you to run all DOS processes in a visible window, and pause after each process. This allows you to see program output before the DOS window is dismissed.

Three options “Verbose,” “Show if Warnings Occur,” and “Show Always” are set in the Auto-make Log area. This area specifies what type of information to record in the auto-make log and when to display the log.

- The “Verbose” option is set to generate a verbose version of the log, containing more detail on each processes’s progress.
- The “Show if Warnings Occur” option is set to show the auto-make log when Warnings are encountered.
- The “Show Always” option enables you to show the auto-make log whenever an auto-make process is started.

Refer to the Project Navigator online help for more information.

## Cleaning Up a Project

Before you archive the project directory, you may want to delete non-critical or intermediate files created during processing of a project. This procedure avoids archiving unnecessary files.

- To delete only intermediate files for the current project, choose **File** ⇒ **Clean Up**.
- To delete both intermediate and report files for the current project, choose **File** ⇒ **Clean Up All**.

## Saving a Project

To save a new project, choose **File** ⇒ **Save As**. The Project Navigator prompts you for a file name for the project.

## What is Saved?

Saving a project saves a project file ( `.syn` extension) with the following information:

- The title of the project
- The sources in the project
- Matching Symbol files (these are `.sym` files with the same name as their relevant modules in the design source; matching symbols are used as functional blocks in schematics to represent a lower-level module)
- Constraint files

ispDesignExpert also tells the Schematic Editor and the Text Editor to Save at the same time you save a project.

When you choose **Save As** to save a project to another directory, ispDesignExpert copies all of the project files to that directory.

## Tips for Saving and Naming Projects

Use the following guidelines when saving and naming source files and projects:

- Do not save more than one project in the same directory.
- Avoid saving a project that has the same base file name as one of its sources. If a source and project have the same base name, you may have problems with the Project Navigator's Auto-make feature. For instance, avoid calling your project `myFile.syn` if it contains a source named `myFile.abl`.

## Reserved File Names

ispDesignExpert reserves several filename extensions for its own use. You should avoid using the following extensions when naming your own files:

<code>_ln</code>	Hierarchy Navigator log file
<code>_sc</code>	Schematic Editor log file
<code>_sy</code>	Symbol Editor log file
<code>_wt</code>	Waveform Editing Viewer log file
<code>_wv</code>	Waveform Viewer log file
<code>.asc</code>	ASCII schematic file
<code>.asy</code>	ASCII symbol file
<code>.bin</code>	Binary waveform file
<code>.ed*</code>	EDIF netlist

---

<b>.err</b>	Error OUTPUT file
<b>.his</b>	Waveform Viewer history file
<b>.nam</b>	Binary waveform name file
<b>.pin</b>	Netlist file for generic netlist by pin
<b>.sch</b>	Schematic Editor files
<b>.sym</b>	Symbol Editor file
<b>.tre</b>	Hierarchy Navigator file
<b>.vci</b>	Constraint file
<b>.vct</b>	Temporary copy of the constraint file
<b>.vco</b>	Constraint output from the Fitter
<b>.vtr</b>	Hierarchy Navigator temporary file
<b>.wav</b>	Waveform Viewer waveforms and trigger information
<b>.wdl</b>	Waveform Editing Tool database
<b>.wet</b>	Waveform Editing Tool database

## Chapter 3 *Design Entry*

---

The chapter describes the supported design entry of the ispDesignExpert. It contains information on:

- ABEL-HDL Design
- Schematic Design
- VHDL Design
- Verilog HDL Design
- EDIF Design
- Hierarchical Design
- Mixed Entry Design
- Design Attributes

## ABEL-HDL Design Entry

This section describes using ABEL-HDL as an entry for your design that supports the ispLSI, GAL, MACH, and PAL devices.

### Add an ABEL-HDL Module to Your Design

To add an ABEL-HDL module to a design, you can either import a .ab1 file or create a new ABEL-HDL module file in the ispDesignExpert Text Editor using ABEL-HDL syntax.

*To import an ABEL-HDL module to your design:*

1. Choose **Source** ⇒ **Import** from the menu bar. The Import File dialog box appears (Figure 3-1). The project type Schematic/ABEL is shown in the title bar of the dialog box letting you double check your project type before importing the source files.

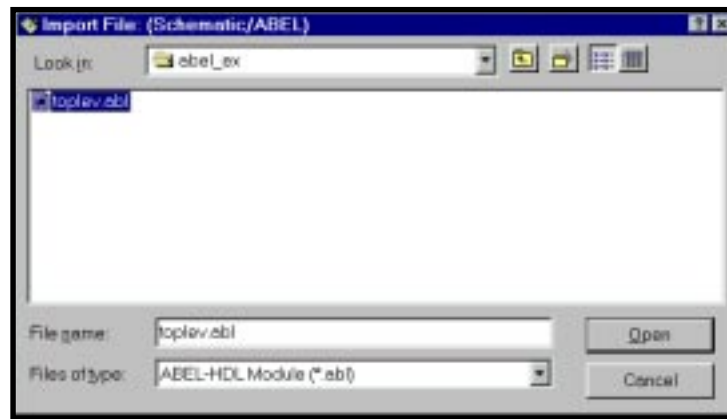


Figure 3-1. Import File Dialog Box

2. In the Import File dialog box, select the desired drive and path.
3. Choose ABEL-HDL Module (\*.ab1) from the Files of type field and highlight the \*.ab1 file you want to import from the File name field.
4. Click **OK**. The selected ABEL-HDL file appears in the Sources in Project list of the Project Navigator as shown in Figure 3-2.



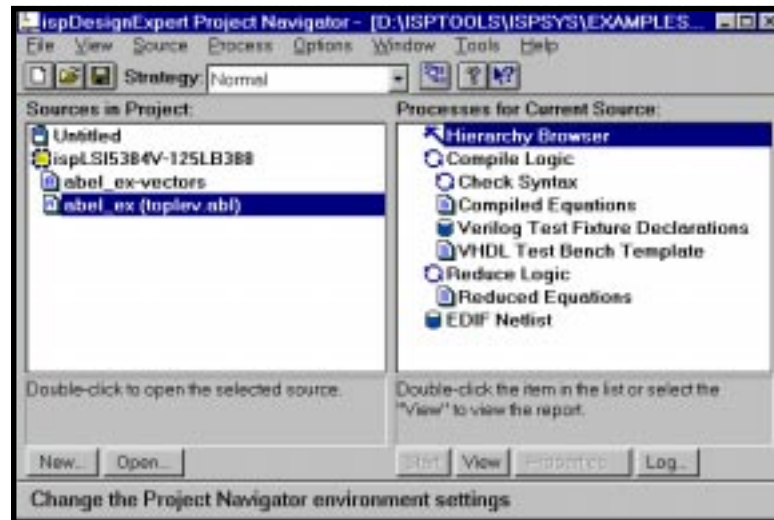


Figure 3-2. Project Navigator Window with \*.abl File Imported

Highlight the **ABEL-HDL** icon and note the processes associated with it. Use the Text Editor or just double-click the \*.abl icon from the Sources list to view the syntax of the ABEL-HDL module (Figure 3-3). The Text Editor is available from the **Window** ⇒ **Text Editor** menu item in the Project Navigator.

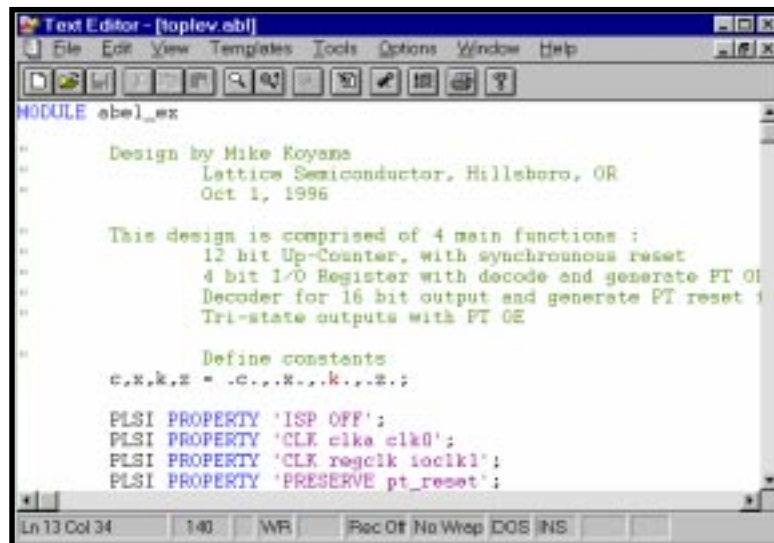


Figure 3-3. Text Editor with Sample ABEL-HDL File

5. With the ABEL-HDL file highlighted, the Compile Logic item in the Process list should also be highlighted. Click the **Properties** button. The Properties dialog box (Figure 3-4) appears.

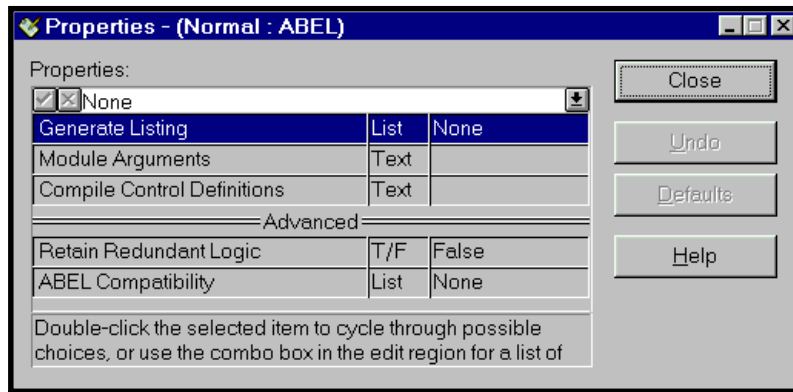


Figure 3-4. Properties - (Normal: ABEL) Dialog Box

- The properties you see are the defaults for the ABEL-HDL module.

When any of the properties are changed, the **Defaults** button becomes active so you can revert to the default settings if you desire. The **Undo** button allows you to undo the latest change you made.

To create a new ABEL-HDL module:

- Choose **Source** ⇒ **New** from the menu bar. The New Source dialog box appears (Figure 3-5).

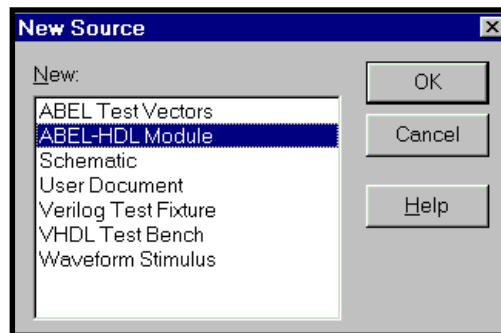


Figure 3-5. New Source Dialog Box

- In the New Source dialog box, choose ABEL-HDL Module and click **OK**. The Text Editor window appears together with the New ABEL-HDL Source dialog box (Figure 3-6).

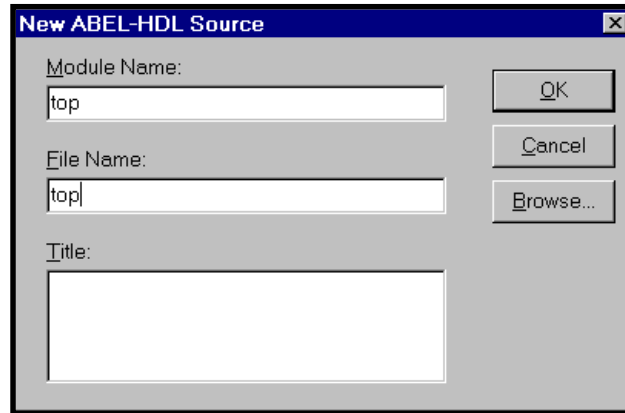


Figure 3-6. New ABEL-HDL Source Dialog Box

3. In the New ABEL-HDL Source dialog box, enter relevant contents into text fields.
4. Click **OK** or press **Enter**. The new ABEL-HDL file appears in the Text Editor window.
5. Use the items in the **Edit** menu to cut, copy, paste, find or replace text.
6. You can also add design attributes to the ABEL-HDL file. Refer to [ABEL-HDL Reference Manual](#) for more information.

## The Language Editor

The Language Editor—the Text Editor—can be used to create or modify HDL modules, ABEL-HDL test vectors, VHDL test benches, and Verilog HDL test fixture files.

You have several ways to open the Language Editor:

- From the Project Navigator Sources window, double-click on an HDL file, for example *design.abl*. Or, select the HDL source from the Sources window of the Project Navigator, click the **Open** button at the bottom or choose the **Source** ⇒ **Open** menu item.
- From the Project Navigator, choose **Window** ⇒ **Text Editor**.

## Schematic Design Entry

This section describes using a schematic as an entry for your design that supports ispLSI, GAL, MACH, and PAL devices.

### Add a Schematic to Your Design

To add a schematic to your design, you can either import a `.sch` file or create a new schematic file in the Schematic Editor window.

*To import a schematic to your design:*

1. Choose **Source** ⇒ **Import** from the menu bar. The Import File dialog box (Figure 3-7) appears. The project type Schematic/ABEL is shown in the title bar of the dialog box letting you double check your project type before importing the source files.



Figure 3-7. Import File Dialog Box

2. In the Import File dialog box, select the desired drive and path.
3. Choose Schematic (`*.sch`) from the Files of type field and highlight the `*.sch` file you want to import from the File name field.
4. Click **OK**. The selected schematic file appears in the Sources of Project list of the Project Navigator. The processes associated with the `.sch` source vary from different devices and different project type you have chosen.
5. Highlight the **Schematic** icon and note the processes associated with it. Use the Schematic Editor or just double-click the `*.sch` icon from the Sources list to view the schematic (Figure 3-8). The Schematic Editor is available from the **Windows** ⇒ **Schematic Editor** menu item in the Project Navigator.

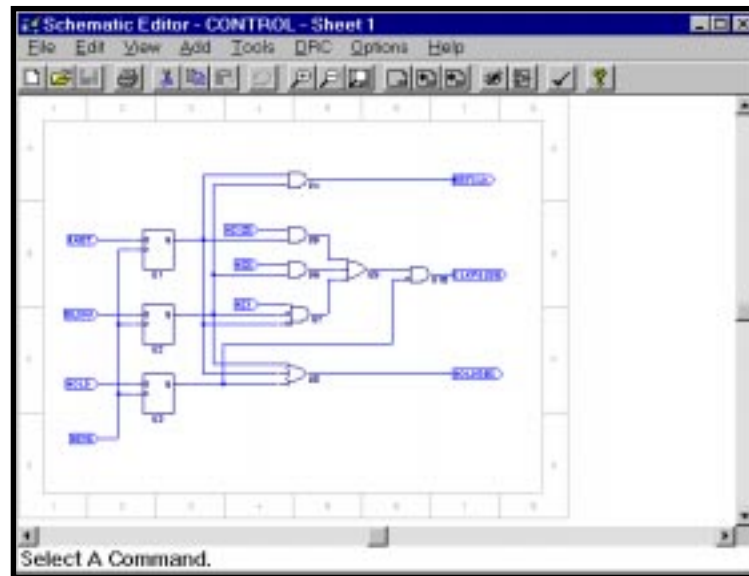


Figure 3-8. Schematic Editor with Sample Schematic File

*To add a blank schematic sheet source to your project:*

1. Choose **Source** ⇒ **New** from the menu bar of the ispDesignExpert Project Navigator window. The New Source dialog box appears.
2. In the New Source dialog box, choose Schematic and click **OK** or press **Enter**.
3. The New Schematic dialog box appears with a blank schematic asking you to enter the name for a new schematic. Enter the file name in the Schematic File Name field. The default file extension shown in Save as type field is `.sch`. Click **OK** or press **Enter**.



**NOTE**

To avoid naming conflicts problems, use different names for the source file and the project file.

To add symbols or macros to your schematic:

1. Choose **Add** ⇒ **Symbol** from the Schematic Editor. The Symbol Libraries dialog box (Figure 3-9) appears.

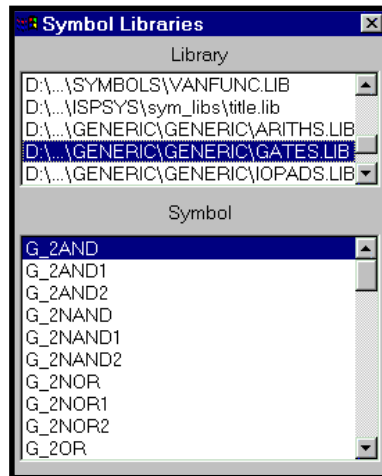


Figure 3-9. Symbol Libraries Window



**TIP**

Use the Zoom features under the **View** pull-down menu or on the Schematic Editor Toolbar for viewing the opened \* .sch file.

2. In the Symbol Libraries dialog box, select <drive>:\..\\*.lib from the library list, then select the target symbol or macro.
3. Move the pointer back over to the Schematic Editor; notice that the symbol you selected is attached to the pointer. Place the symbol by clicking on the schematic.
4. Move the cursor back to the Symbol Libraries dialog box and select another symbol. Place the symbol in its proper position on the schematic.
5. From the Schematic Editor menu bar, select **Add** ⇒ **Wire**. Click on the output pin of a gate to start the wire. Each successive click will bend the wire (a double-click will end the wire if it is not connected). Connect the wire to the input of a gate.
6. Repeat the above procedure to add other symbols or macros from the Symbol Library. Figure 3-10 is an example of a simple schematic.

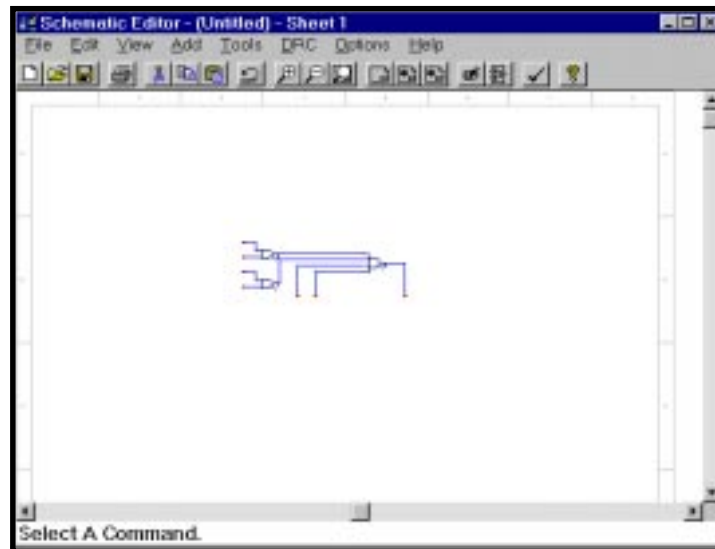


Figure 3-10. Building the Schematic

7. Choose **File** ⇒ **Save** from the menu bar of the Schematic Editor window to save your design.

If you cannot find a symbol that you just generated from the Symbol Libraries of the Schematic Editor, select **Options** ⇒ **ispLSI/GAL/MACH/PAL Schematic Configuration** from the Project Navigator. In the Symbol Paths tab of the Schematic Environment dialog box, add the path of the newly generated symbol in the Paths field. Then you will be able to add the symbol to your schematic from the Symbol Libraries.

To complete your design you need to add net names and I/O Markers. When adding net names, you will use a feature of the ispDesignExpert that allows you to add the net name and the net simultaneously. I/O Markers are special symbols required to indicate which signals represent pins. The markers assume the name of the net they are attached to and are different from I/O Pad symbols.

To add net names or I/O markers:

1. Select **Add** ⇒ **Net Name** from the Schematic Editor menu bar. The status bar at the bottom of the window will prompt you to enter the net name. Type the net name and press **Enter**. The Net Name will be attached to your cursor.
2. Move the cursor to the net where you want to add a name, click and hold on the unconnected end of the net (i.e. the red box at the left end of the net), drag to the left and release. This will place the net name and create a net simultaneously. The net name should now be attached to the end of the net.
3. Repeat this procedure to add net names to other nets in the schematic.
4. Select **Add** ⇒ **I/O Marker** from the Schematic Editor menu bar. The I/O Markers dialog box appears. Choose **Input**.
5. Move the cursor to the end of an input net (between the end of the net and the net name) and click. An input marker appears with the net name inside of it. Move to the next input and click again. Repeat until all inputs have I/O Markers.



**TIP**

To add all the input markers at once, click and hold the cursor, and drag it to select all the input net names. This procedure works for output pins as well.

6. Choose **Output** from the I/O Markers dialog box and click on the end of the output net. Save your schematic.

### Add Design Attributes

Attributes can be added to either symbols or nets. In the ispDesignExpert, pin attributes are actually added to the I/O Pad symbols, not the I/O Markers. I/O Pad symbols are only necessary if you want to add attributes to pins. Otherwise, you only need I/O Markers.

To add design attributes to a symbol:

1. From the Schematic Editor menu bar, select **Edit** ⇒ **Attribute** ⇒ **Symbol Attribute**. The Symbol Attribute Editor dialog box appears. On the schematic, click on a symbol or an I/O Pad attached to a net. A list of related attributes appears in the dialog box.
2. Click List All Attributes to display all of the available symbol design attributes.
3. Select the attribute you need to add or edit and replace \* with proper values as shown in Figure 3-11 in the text box. Click **Go To** to add the value of the selected attribute to the I/O Pad or symbol in the schematic. Close the dialog box.



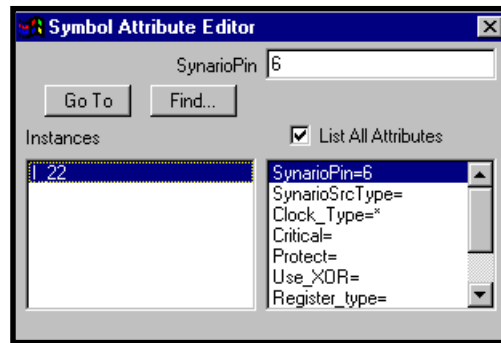


Figure 3-11. Symbol Attribute Editor Dialog Box (ispLSI Devices)

4. The steps are similar for adding a Net attribute. On the schematic, click on the net you want to edit. In the Net Attribute dialog box (Figure 3-12), enter the relevant contents of the attribute. Close the dialog box.

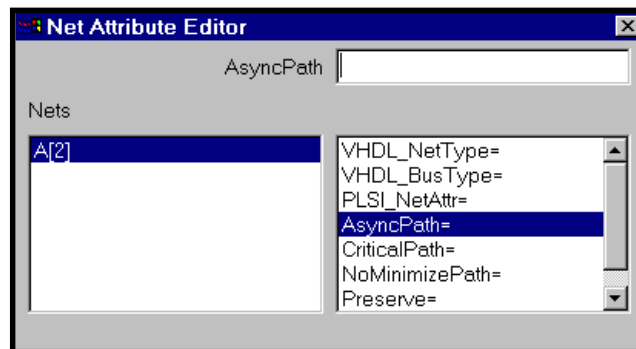


Figure 3-12. Net Attribute Dialog Box (ispLSI Devices)

5. Check your schematic for errors by using **DRC** ⇒ **Consistency Check**. An Error Report window pops up to show the error message. If no error is found, the message “No errors detected” will display in the Error Report window.
6. From the Schematic Editor menu bar, choose **File** ⇒ **Save** to save your design. Select **File** ⇒ **Exit** to close the Schematic Editor window.

## Create a Symbol

A useful feature of the ispDesignExpert software is to quickly create a symbol for a schematic. By using this, you create a reusable macro that can be placed on a higher level schematic sheet.

*To create a symbol:*

1. Open the schematic file by double-clicking on the schematic source \*.sch in the ispDesignExpert Project Navigator.
2. In the Schematic Editor menu bar, select **File** ⇒ **Matching Symbol**.
3. Select **File** ⇒ **Exit** to close the schematic.
4. The symbol is created and added to your symbol list. It can be found in the Local symbol library.

Or (to any design that contains .naf file)

1. In the Schematic Editor, select **File** ⇒ **Generate Symbol**.
2. The Select File dialog box appears prompting you to choose a .naf file. When you import, create, or save a design source (\*.abl, \*.sch, \*.vhd, or \*.v) file in a project, .naf file(s) containing port information of module(s) in the design source will automatically be generated and saved under the current project directory.



### NOTE

If you cannot find a desired .naf file when creating a symbol, open the corresponding source in the Text Editor (if it is an HDL source) or the Schematic Editor (if it is a schematic source). Make a modification to that source, which will not change its original functionality, for example add a space at the end of an HDL file, or add a wire to a schematic file and then remove the wire. Save the modified source file. Then you will be able to find the relevant .naf file.



### NOTE

The base name of a .naf file is the same as its relevant module.

3. Click **Open**. A notice will appear telling you “Symbol has been generated.” The symbol has been created and added to your symbol list. It can be found in the Local symbol library.

Refer to [page 87](#) for more details on the Local symbol library.

## Add Design Control Properties (ispLSI Designs Only)

A subset of Compilation Properties, Device Control Options are device dependent and define the objective for the design implementation process. To set the properties from a schematic design flow, use the following procedures.

1. In the Project Navigator, select the top-level .sch file in the Sources in Project list.
2. For an ispLSI 1000, 2000, and 3000 device, select the Compile Schematic icon in the Processes for Current Source list. For an ispLSI 5000V, 6000, or 8000 device, select EDIF Netlist icon in the Processes for Current Source list.
3. Select the Properties button at the bottom of the Project Navigator window. The Properties dialog box appears as shown in Figure 3-13, Figure 3-14, Figure 3-15, and Figure 3-16.

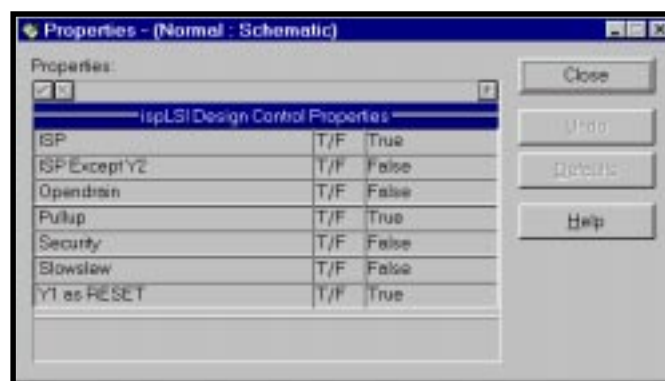


Figure 3-13. Compile Schematic Properties Dialog Box

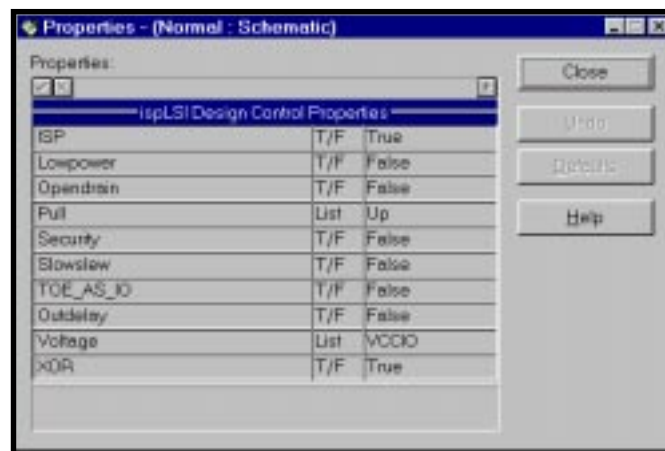


Figure 3-14. ispLSI 5000V EDIF Netlist Properties Dialog Box

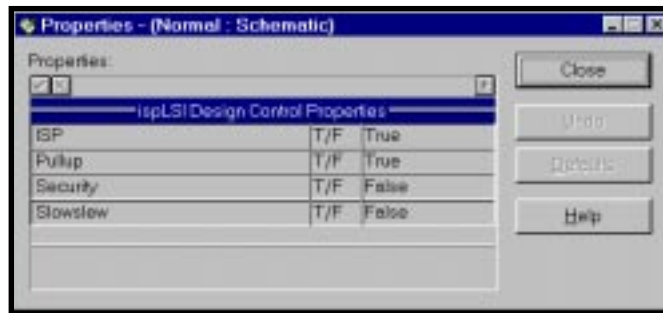


Figure 3-15. ispLSI 6000 EDIF Netlist Properties Dialog Box

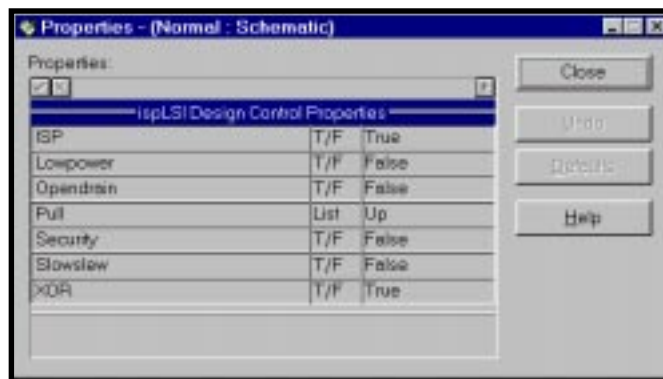


Figure 3-16. ispLSI 8000 EDIF Netlist Properties Dialog Box

4. Set the appropriate Design Control Property depending on your target device. Check the [ispEXPERT Compiler User Manual](#) for device dependencies.
5. Click **Close**.

## VHDL Design

This section describes using VHDL as an entry for your design that supports the ispLSI, GAL, MACH, and PAL devices.

### Add a VHDL Module to Your Design

To add a VHDL module to a design, you can either import a .vhd file or create a new VHDL module file in the ispDesignExpert Text Editor.

*To import a VHDL module to your design:*

1. Choose **Source** ⇒ **Import** from the menu bar. The Import File dialog box appears (Figure 3-17). The project type Schematic/VHDL is shown in the title bar of the dialog box letting you double check your project type before importing the source files.

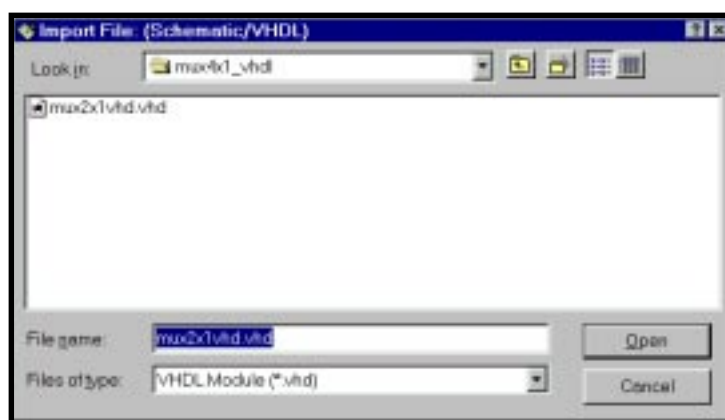


Figure 3-17. Import File Dialog Box

2. Choose the desired VHDL module (\* .vhd) and click **OK**.
3. The Import Source Type dialog box appears (Figure 3-18). Choose the type of source you want to import into your project, either VHDL Module or VHDL Test Bench, from the Type of Source field. Click **OK**.

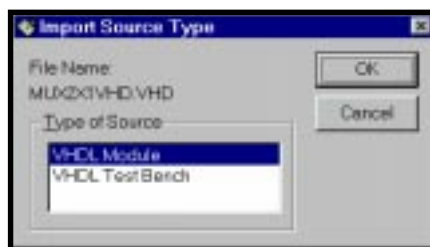


Figure 3-18. Import Source Type Dialog Box

**NOTE**

If you have not chosen Schematic/VHDL as the project type, the .vhd file will be imported as a VHDL test bench without prompting the Import Source Type dialog box.

- The selected VHDL file appears in the Sources in Project list of the Project Navigator as shown in Figure 3-19.

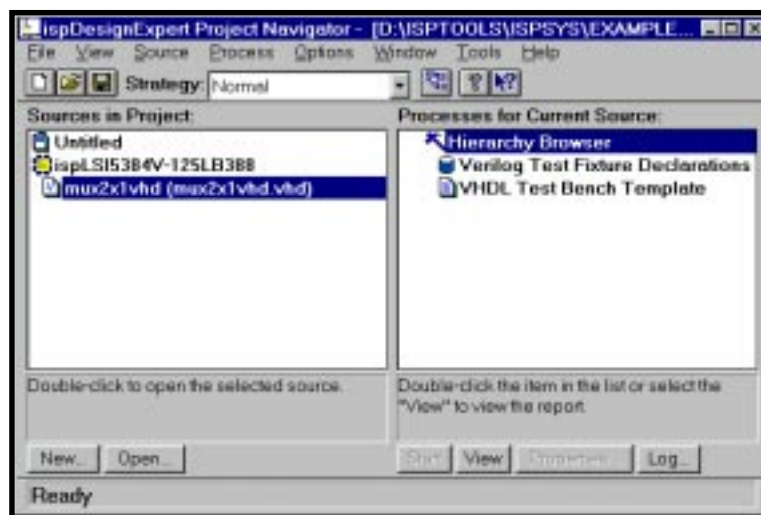


Figure 3-19. Project Navigator Window with a VHDL Module File Imported

Highlight the **VHDL** icon and use the text editor to view the syntax of the VHDL module (Figure 3-20).

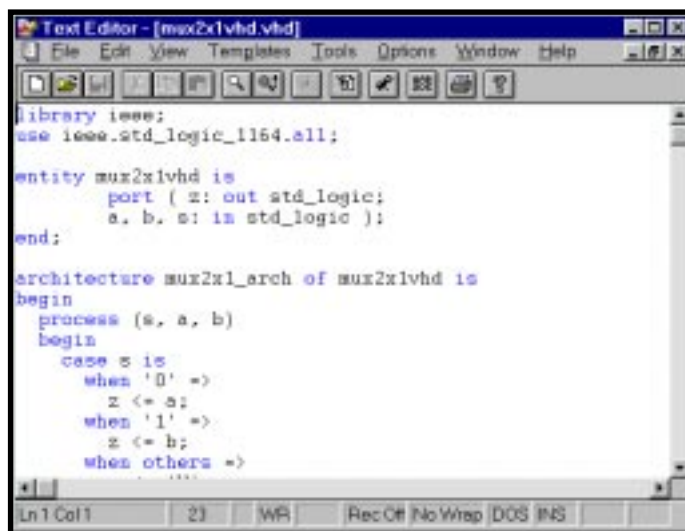


Figure 3-20. Text Editor with Sample VHDL Module File

To create a new VHDL module:

1. Choose **Source** ⇒ **New** from the menu bar. The New Source dialog box appears (Figure 3-21). The project type Schematic/VHDL is shown in the title bar of the dialog box letting you double check your project type before creating the source files.

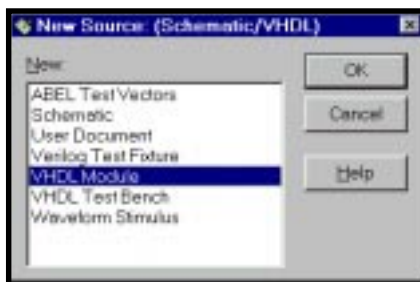


Figure 3-21. New Source Dialog Box

2. In the New Source dialog box, choose VHDL Module and click **OK**. The Text Editor window appears together with the New VHDL Source dialog box (Figure 3-22).

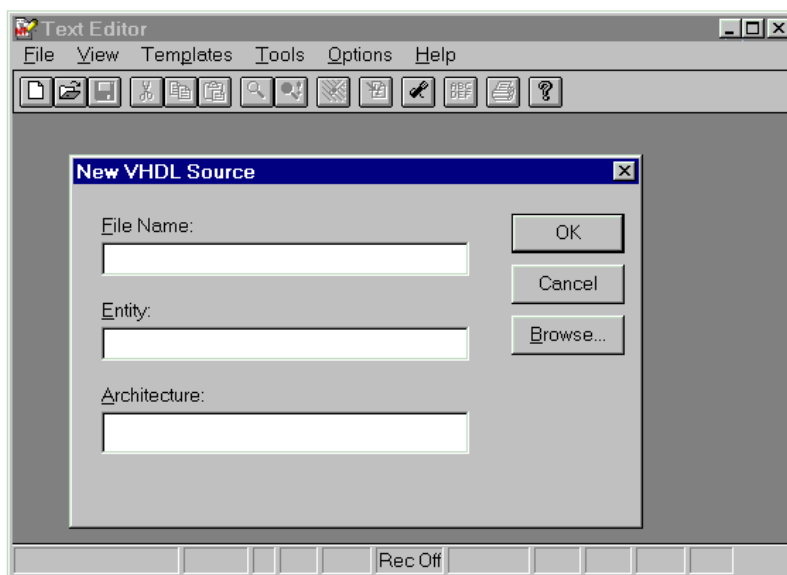


Figure 3-22. Text Editor Window with New VHDL Source Dialog Box

3. In the New VHDL Source dialog box, enter relevant contents into the text fields.
4. Click **OK**. The new VHDL file appears in the Text Editor window.
5. Use the items in the **Edit** menu to cut, copy, paste, or replace text.

## Create an EDIF File for ispLSI

Before fitting your VHDL design into an ispLSI part, you will need to create a \*.edf file for input to the ispDesignExpert Compiler. The synthesis of the VHDL file will be finalized in this process.

*For an ispLSI design, to create a .edf file before fitting your design:*

1. In the Project Navigator, select the target device icon to display the associated processes in the Processes window as shown in Figure 3-23.

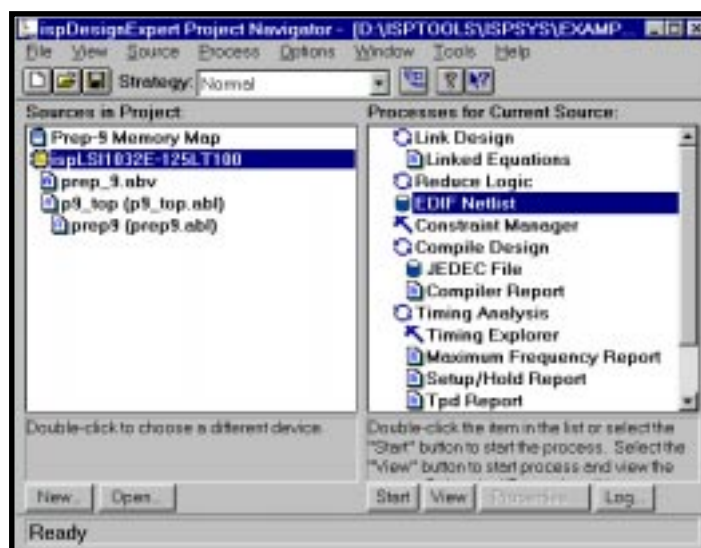


Figure 3-23. VHDL Design Flow

2. Click the Start button or double-click on the EDIF Netlist process to start creating the .edf file for the ispDesignExpert Compiler. The VHDL file will be synthesized in this process as well.

For a GAL, MACH or PAL devices, the EDIF netlists will be created when you run the Fit Design process.

For more information about ispLSI Design Attributes, Device Control Options, EDIF Property Files, or Parameter Files, see the [ispEXPERT Compiler User Manual](#).



## Verilog HDL Design

This section describes using Verilog HDL as an entry for your design that supports the ispLSI, GAL, MACH, and PAL devices.

### Add a Verilog HDL Module to Your Design

To add a Verilog HDL module to a design, you can either import a `.v` file or create a new Verilog HDL module file in the Text Editor.

*To import a Verilog HDL module to your design:*

1. Choose **Source** ⇒ **Import** from the menu bar. The Import File dialog box appears. The project type Schematic/Verilog HDL is shown in the title bar of the dialog box letting you double check your project type before importing the source files.
2. In the Import File dialog box, select the desired `.v` file. Click **OK**.
3. The selected Verilog HDL file appears in the Sources in Project list of the Project Navigator as shown in Figure 3-24.

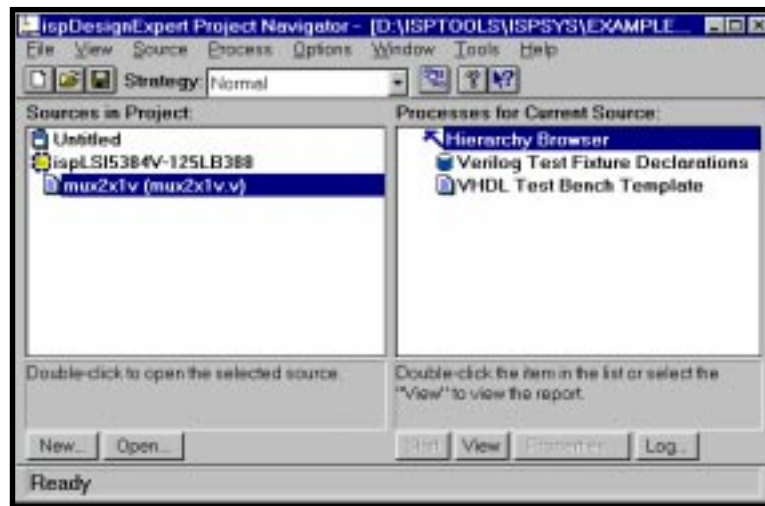


Figure 3-24. Project Navigator Window with `*.v` File Imported

Highlight the **Verilog HDL** icon and use the text editor to view the syntax of the Verilog HDL module (Figure 3-25).

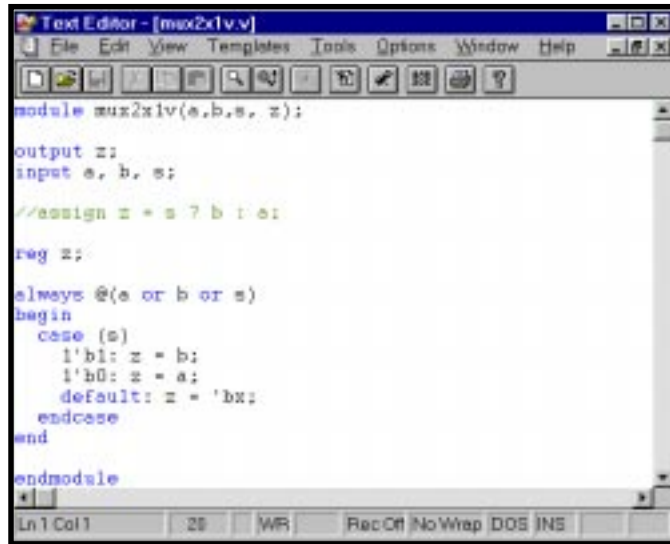


Figure 3-25. Text Editor with Sample Verilog HDL File

To create a new Verilog HDL module:

1. Choose **Source** ⇒ **New** from the menu bar. The New Source dialog box appears (Figure 3-26). The project type Schematic/Verilog HDL is shown in the title bar of the dialog box letting you double check your project type before creating the source files.

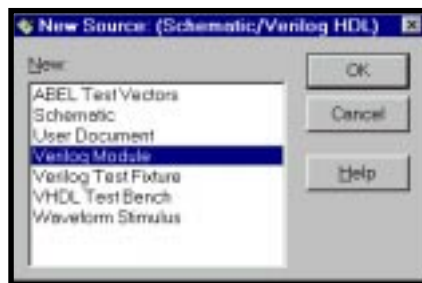


Figure 3-26. New Source Dialog Box

2. In the New Source dialog box, choose Verilog Module and click **OK**. The Text Editor window appears together with the New Verilog Module dialog box (Figure 3-27).

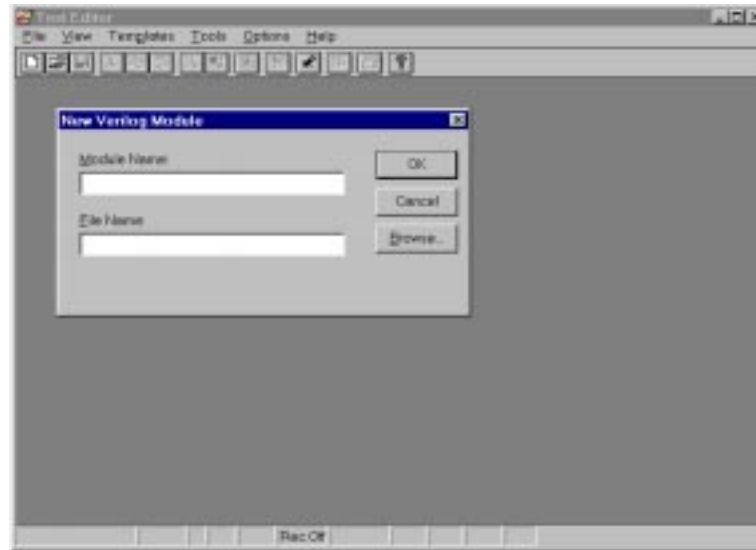


Figure 3-27. Text Editor Window with New Verilog Module Dialog Box

3. In the New Verilog Source dialog box, enter relevant contents into text fields.
4. Click **OK**. The new Verilog HDL file appears in the Text Editor window.
5. Use the items in the **Edit** menu to cut, copy, paste, find or replace text.

## Create an EDIF File

Before fitting your Verilog HDL design into an ispLSI part, you will need to create a \* .edf file for input to the ispEXPERT Compiler. The synthesis of the Verilog HDL file will be finalized in this process.

*To a .edf file before fitting your design to an ispLSI part:*

1. In the Project Navigator, select the target device icon to display the associated processes in the Processes window as shown in Figure 3-28.

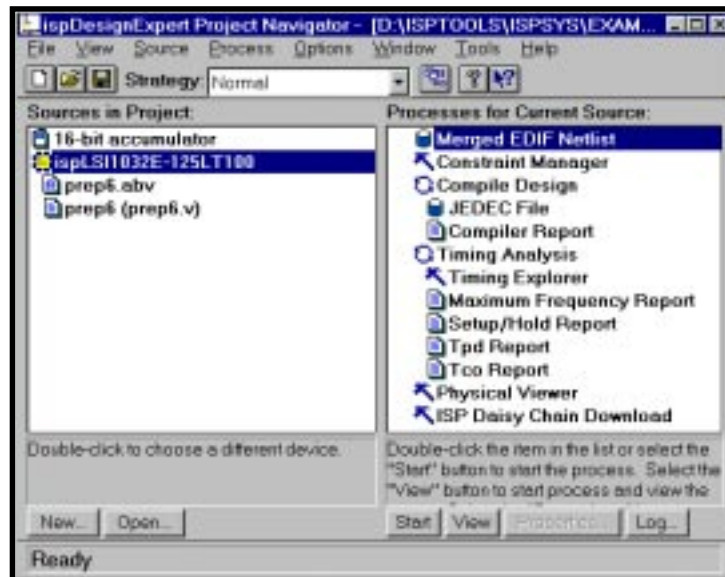


Figure 3-28. Verilog HDL Design Flow

2. Click the Start button or double-click on the Merged EDIF Netlist process to start creating the .edf file for the ispEXPERT Compiler. The Verilog HDL file will be synthesized in this process as well.

For a GAL, MACH or PAL devices, the EDIF netlists will be created when you run the Fit Design process.

For more information about ispLSI Design Attributes, Device Control Options, EDIF Property Files, or Parameter Files, see the [ispEXPERT Compiler User Manual](#).

## EDIF Design

You can import an EDIF design netlist description into the ispDesignExpert from third-party synthesis or schematic tools for the ispLSI, GAL, MACH, or PAL designs.

*To import an EDIF netlist into your design:*

1. In the Project Navigator, choose **Source** ⇒ **Import** to open the Import File dialog box.
2. Choose EDIF Netlist (\*.ed\*) from the Files of Type field of the Import File dialog box (Figure 3-29). And then select the EDIF file you want to import.

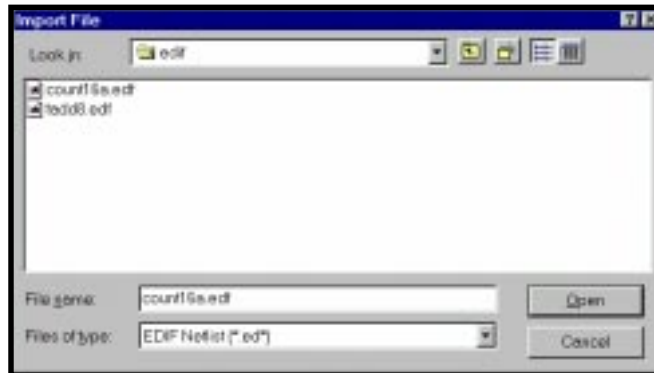


Figure 3-29. Import File Dialog Box

3. Click **Open**. If you have selected an ispLSI or GAL device, the EDIF Reader Settings dialog box appears (Figure 3-30). If you have selected a MACH or PAL device, the Import EDIF dialog box displays (Figure 3-31).

Both dialog boxes are used to specify the vendor whose tool was used to prepare the design file and to specify settings to be used when the EDIF design is read into the project.



Figure 3-30. EDIF Reader Settings Dialog Box

---

Vendor field	From the pull-down menu, select the vendor of the third-party software used to create the design. If you select Altera, the format expands to show the Altera-specific fields. If you turn on the Load-vendor-specific defaults check box before selecting a vendor, the defaults for the vendor you select display. If you do not wish to display the default settings when you change vendors, deselect this check box. You may make changes after selecting a vendor.
VCC and GND	<p>You can specify whether VCC or GND symbols are represented as Nets or Cells when an EDIF design file is read. In the VCC GND Representation area, check whether VCC and GND are to be read as Nets or Cells. The default representation is a net.</p> <p>You can also specify VCC and GND names. The default names are VCC and GND. In the logic evaluation phase for the signals connected to the “PRN” pin and “CLRn” pin, the conversion program must know the name of the power and ground signals.</p>
Bus Reconstruction	<p>The ispDesignExpert supports EDIF files that contain arrays for ispLSI designs. A <i>design.ary</i> file is created automatically by the EDIF Reader and is used during VHDL and Verilog output generation after the design is compiled. The vectors or buses are automatically reconstructed and included for VHDL or Verilog post-route outputs based on the information provided by the array file.</p> <p>Use the Array Index Ordering radio buttons to specify the index range.</p> <p>Use the Least Significant Bit radio buttons to specify whether the leftmost bit or the rightmost bit is to be the least significant bit (LSB).</p>
Ground Floating Output Pins	Select this option to ground all floating output pins.

If you choose Altera in the Vendor field, the dialog box expands to display the Altera Options section. Refer to the [ispEXPERT Compiler User Manual](#) for more information on this section.

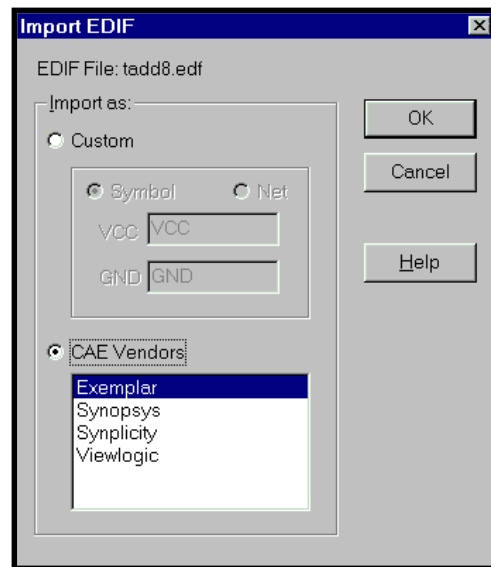


Figure 3-31. Import EDIF Dialog Box

Custom field

The default setting for power and ground in ispDesignExpert are the VCC and GND symbols. Check the **Custom** radio button if you know that the EDIF generated by other tools use different conventions. After you check the **Custom** radio button, the whole Custom field becomes active. Select either Symbol or Net representation. Then type the new names for VCC and GND.

CAE Vendors

If you generate the EDIF file from the supported third-party design kit, you can then select CAE Vendors, and then choose from the list the vendor that generated the EDIF file.

4. Click **OK**. The software adds the selected EDIF file (.ed\*) to the project sources (Figure 3-32).

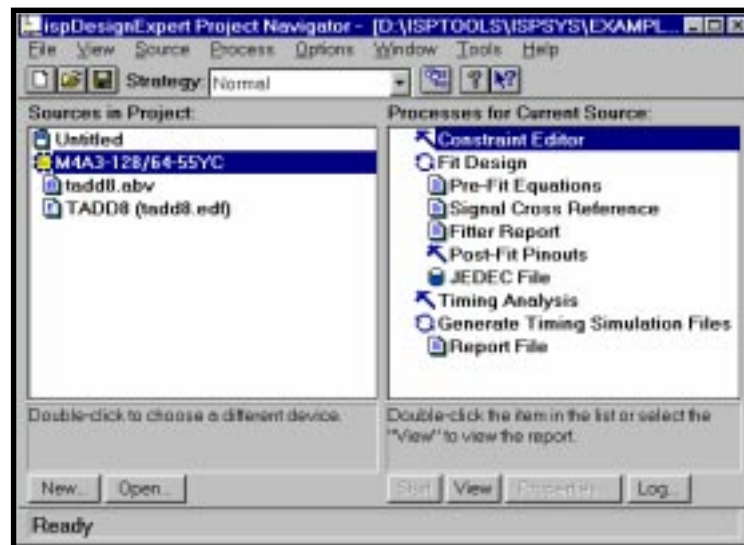


Figure 3-32. Project Navigator after Importing EDIF Netlist

## MACH/PAL Properties in the EDIF File

### Property List

If you have chosen a MACH or PAL device for your EDIF design, the ispDesignExpert takes design specific constraints as properties from an EDIF netlist. The following is the list of properties which are supported by the fitter:

EDIF property handling for CPLD:

#### 1. PIN LOCATION PROPERTY:

NAME: LOC  
VALUE: {PIN#}

Example:

LOC=P20  
SCOPE: IO PORT, net connect to the IO port.

#### 2. GROUPING:

NAME: GROUPING  
VALUE: GROUP NAME.  
NAME: LOC  
VALUE: LOCATION NAME.

Example: Use the following command to assign signal locations in your design. In this case, you have a list of internal node: a, b, c, d, e, f, and g, and you want to assign them into a group "mg", and the location of this group need to be assign to Block "A", Segment "2".



```
On net a, grouping = mg
On net a, loc = "A, 2"
On net b, grouping = mg
On net b, loc = "A, 2"
On net c, grouping = mg
On net c, loc = "A, 2"
On net d, grouping = mg
On net d, loc = "A, 2"
On net e, grouping = mg
On net e, loc = "A, 2"
On net f, grouping = mg
On net f, loc = "A, 2"
On net g, grouping = mg
On net g, loc = "A, 2"
```

### 3. OUTPUT SLEW PROPERTY:

```
NAME: SLEW
VALUE: {Fast, Slow}
```

Example: To set port A to high slew, put the following property on the net or on the Port:

```
SLEW = Fast.
SCOPE: OUTPUT PORT/NET.
```

### 4. Signal Optimization PROPERTY:

```
NAME: OPT
VALUE: {KEEP, COLLAPSE}
SCOPE:
On any net of the design.
```

## Import Mechanism for MACH/PAL

By default, properties in EDIF files are ignored. If you need EDIF properties to be imported to ispDesignExpert, do the following:

1. Choose **Tools** ⇒ **Import Source Constraint Option**.

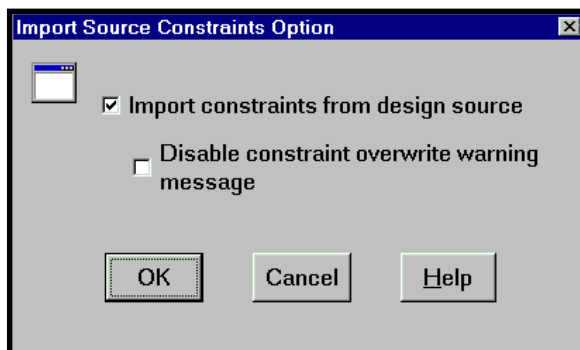


Figure 3-33. Import Source Constraints Option Dialog Box

The Import Source Constraints Option dialog box (Figure 3-33) lets you import constraints, such as Location (pin/code) Assignments, Group Assignments, and Output Slew Rate, from source files (ABEL, schematic, VHDL, Verilog HDL, or EDIF).

2. Check Import constraints from design source and click **OK**.

When the “Import constraints from design source” option is selected, ispDesignExpert displays a confirmation dialog box prior to implementing the function. This confirmation dialog box will appear every time you run the Fit Design process, unless you check the “Disable constraint overwrite warning message” option.

On the warning message dialog box, if you click **Yes**, the constraints from the source files are written into the project constraint file.



### **NOTE**

Constraints from source files and existing constraints in project constraint file are not merged; existing constraints are overridden by the new constraints.

Existing constraints (only Location Assignments, Group Assignments, and Output Slew Rate are affected) in the project are cleared, regardless if there are constraints in the source file. If there are constraints in the source file, then the new constraints are written into the project constraint file. If there are no constraints in the source file then no constraints will be written into the file.

# Hierarchical Design

A design with more than one level is called hierarchical. A single-level design is referred to as being flat. Converting a section of circuitry to a block makes a flat design hierarchical. This is commonly referred to as “hierarchical” design.

## Overview of Hierarchical Design

ispDesignExpert supports full hierarchical design. Hierarchical structuring permits a design to be broken into multiple levels, either to clarify its function or permit the reuse of functional blocks. For instance, a large, complex design does not have to be created as a single module. By using a hierarchical design, each component or piece of a complex design can be created as a separate module.

A design is hierarchical when it is broken up into functional blocks, or modules. For example, you could create a top-level schematic describing an integrated circuit. In the schematic, you could place a Block symbol (a Block symbol represents a functional block) that provides a specific function of the chip. You can then elaborate the underlying logic for the Block symbol as a separate schematic or as a separate HDL module.

The module represented by the Block symbol is said to be at one level below the schematic in which the symbol appears. Or, the schematic is at one level above the Block's module. Regardless of how you refer to the levels, any design with more than one level is called a hierarchical design.

## Advantages of Hierarchical Design

The most obvious advantage of hierarchical design is that it encourages modularity. A careful choice of the circuitry you select to be a module will give you a Block symbol that can be reused.

Another advantage of hierarchical design is the way it lets you organize your design into useful levels of abstraction and detail. For example, you can begin a project by drawing a top-level schematic that consists of nothing but Block symbols and their interconnections. This schematic shows how the project is organized but does not display the details of the modules (Block symbols).

You then draw the schematic for each Block symbol. These schematics can also contain Block symbols for which you have not yet drawn schematics. This process of decomposition can be repeated as often as required until all components of the design have been fully described as schematics.

Breaking the schematic into modules adds a level of abstraction that lets you focus on the functions (and their interaction) rather than on the device that implements them. At the same time, you are free to view or modify an individual module.

Although there are many ways of “breaking apart” a complex design, some may be better than others. In general:

- Each module should have a clearly defined purpose or function and a well-defined interface.
- Look for functions or component groupings that can be reused in other projects.
- The way in which a design is divided into modules should clarify the structure of the project, not obscure it.

## Hierarchy vs. Sheets in Schematics

A hierarchical design is not the same as creating a schematic with multiple sheets. In a schematic, you can add as many sheets as desired to extend beyond the original sheet. However, regardless of how many sheets you add, all the components of the design are still at a single level; all sheets are still contained in the same module.

## Approaches to Hierarchical Design

Hierarchical designs consist of one top-level module. This module can be of any format, such as ABEL-HDL, VHDL, Verilog HDL, schematic, or EDIF netlist. Lower-level modules can be of any supported sources also, and are represented in the top-level module by a functional block or other “place-holders.”

Following are some rules you need to follow when creating a hierarchical design in ispDesignExpert.

- The top-level source can be of any format, such as ABEL-HDL, VHDL, Verilog HDL, schematic, or EDIF netlist.
- For hierarchical Schematic/ABEL designs:
  - If the upper-level source is a schematic file, the lower-level source can be either a schematic or an ABEL-HDL file;
  - If the upper-level source is an ABEL-HDL file, the lower-level source can be either a schematic or an ABEL-HDL file.
- For hierarchical Schematic/VHDL designs:
  - If the upper-level source is a schematic file, the lower-level source can be either a VHDL file or a schematic file;
  - If the upper-level source is a VHDL file, the lower-level source can only be a VHDL file.
- For hierarchical Schematic/Verilog HDL designs:
  - If the upper-level source is a schematic file, the lower-level source can be either a Verilog HDL file or a schematic file;
  - If the upper-level source is a Verilog HDL file, the lower-level source can only be a Verilog HDL file.
- For EDIF designs:
  - Hierarchical EDIF design is not allowed.

You can create the top-level module first, or create it after creating the lower-level modules. For example, in the Schematic Editor you can create schematic project components in any order and then combine them into a complete design. You can draw a schematic first and then create a Block symbol for it, or specify the Block first and then create the schematic for it later.

## Hierarchical ABEL-HDL Design

The following steps outline how to specify a lower-level block symbol in an ABEL-HDL design. Figure 3-34 shows an ABEL module (Add) instantiated in another ABEL-HDL module (Top). However, you can follow the same procedures to instantiate a lower-level schematic Block symbol in an ABEL module as well.

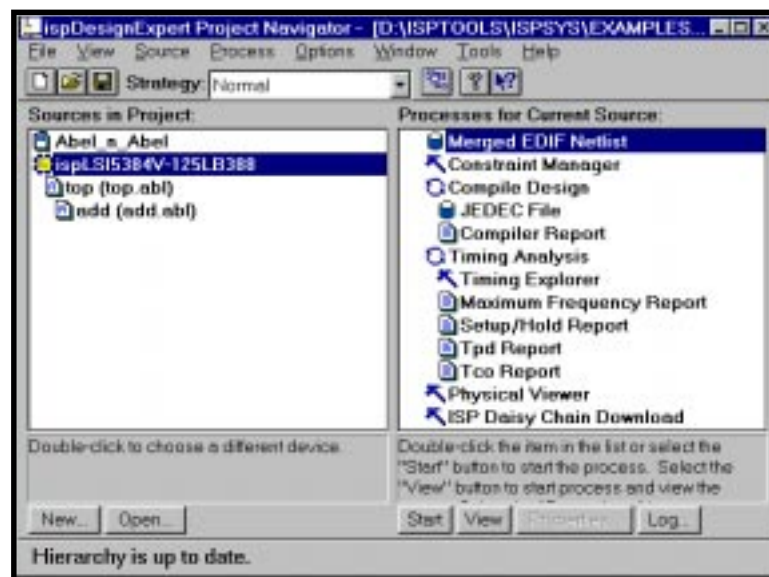


Figure 3-34. Hierarchical ABEL-HDL Design

*To instantiate a lower-level module in an ABEL-HDL module:*

1. In a Text Editor, open your ABEL-HDL file (**File** ⇒ **Open**) or create a new ABEL-HDL file (**File** ⇒ **New**).
2. In the ABEL-HDL file, use the **interface** and **functional\_block** keywords to instantiate lower-level files.

You can place multiple instances of the same interface in the same design by using the **functional\_block** statement.

The interface must have the same names as the corresponding nets (schematics) or pin names (ABEL-HDL) in the lower-level module.

## ABEL-HDL Hierarchy Examples

Figure 3-35 below shows an upper-level ABEL-HDL module (`top.abl`) that references either a lower-level ABEL-HDL module (`add.abl`). Following that, Figure 3-36 shows a lower-level module implemented as an ABEL-HDL block, while Figure 3-37 shows the lower-level module implemented as a schematic block (`add.sch`). Both `add.abl` and `add.sch` can be instantiated in the upper-level source `top.abl`.

```
MODULE top

"inputs
AIN,BIN,CARRYIN pin;

"outputs
CARRYOUT,SUMOUT pin;

add INTERFACE(A,B,CI -> SUM,CO);

my_add functional_block add;

EQUATIONS
my_add.A = AIN;
my_add.B = BIN;
my_add.CI = CARRYIN;
SUMOUT = my_add.SUM;
CARRYOUT = my_add.CO;

END
```

Figure 3-35. Top-level ABEL-HDL Module (`top.abl`)

```

MODULE add

"inputs
A,B,CI    pin;

"outputs
CO,SUM    pin;

EQUATIONS
SUM =      A&B&CI
          +!A&!B&CI
          +!A&B&!CI
          +A&!B&!CI;

CO =       A&B
          +A&CI
          +B&CI;

END

```

Figure 3-36. Lower-level ABEL-HDL Module (add.abl)

Figure 3-37 shows the schematics for the lower-level ABEL-HDL module Add. It can also be instantiated by the `top.abl` design.

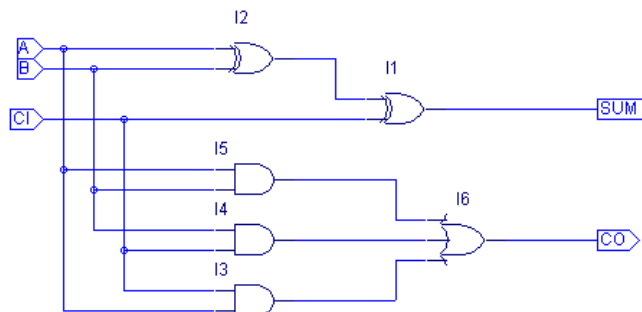


Figure 3-37. Lower-level Schematic Block (add.sch)

**NOTE**

If you are in a lower-level schematic, you can choose **Add ⇒ New Block Symbol** and then click **Use Data From This Block** on the dialog box to automatically create a functional block symbol for the current schematic.

The name of the lower-level schematic must match the block name (schematic) or the interface name (ABEL-HDL) in the upper-level module. This associates the lower-level module with the symbol representing it. For example, the schematic in Figure 3-37 must be named `add.sch`.

The net name in the lower-level schematic correspond to the pin names (schematics) or pin names (ABEL-HDL) in the upper-level module.



## Hierarchical Schematic Design

The following steps outline how to specify a lower-level block symbol in a schematic. Figure 3-38 shows a schematic block (`add.sch`) instantiated in another schematic (`top.sch`). However, you can follow the same procedures to instantiate a lower-level ABEL-HDL block symbol in a schematic.

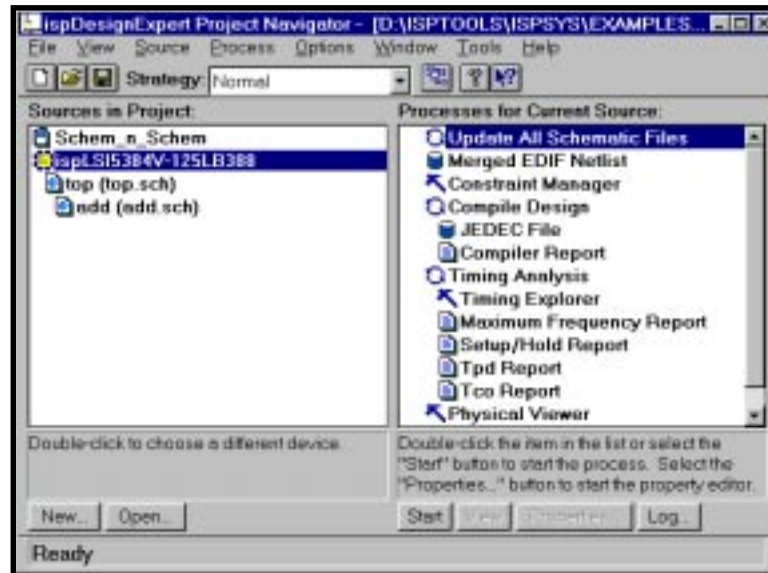


Figure 3-38. Hierarchical Schematic Design

*To instantiate a lower-level schematic block in a schematic:*

1. In the Project Navigator, choose **Window** ⇒ **Schematic Editor** to open the Schematic Editor. The Schematic Editor opens with a new sheet. This schematic is going to be the top-level schematic.  
Next, you will place a Block symbol (a functional block) in the schematic that will represent a more-detailed schematic or other source at the next-lower level.
2. To add a new Block symbol to the schematic, choose **Add** ⇒ **New Block Symbol** to open the New Block Symbol dialog box from the Schematic Editor.
3. Type a Block name, and input and output pin names in the relevant fields of the New Block Symbol dialog box. The first character of each pin name must be alphabetic. Separate pin names with a comma (Figure 3-39).

The symbol pins must have the same names as the corresponding lower-level I/O markers (schematics) or pin names (ABEL-HDL) in the lower-level module. For example, a wire connected to a pin named **A** on the symbol is also connected to the net named **A** in the lower-level module. The **DRC** ⇒ **Consistency Check** command in the Schematic Editor and the **DRC** ⇒ **Check Circuit** command in the Hierarchy Navigator flag an error if a Block symbol has a pin without a corresponding net in the related schematic.

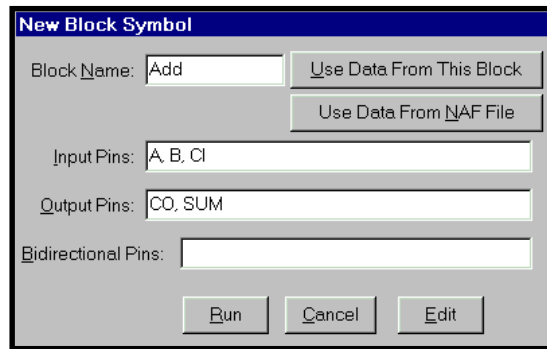


Figure 3-39. New Block Symbol Dialog Box

4. When you are finished, click **Run**. The new symbol attaches to the cursor and is ready for placement.
5. Click in the schematic sheet to place the new Block symbol (Figure 3-40). To end the symbol entry process, right-click anywhere in the sheet.

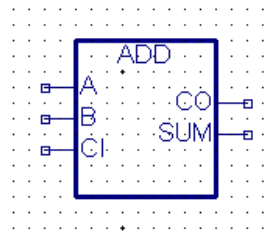


Figure 3-40. New Block Symbol - Add

### Schematic Hierarchy Examples

Figure 3-41 shows an example of how the new symbol corresponds to an underlying schematic. In this figure, pin A on the Block symbol corresponds to the net in the schematic, which is also named A. The other pins, B, CI (Carry In), CO (Carry Out) and SUM, also correspond to named nets in the schematic.

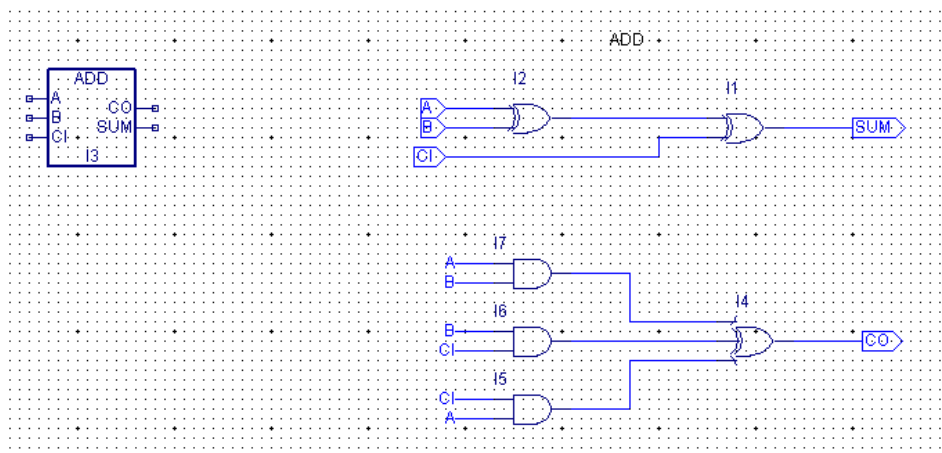


Figure 3-41. A Block Symbol and its Underlying Schematic

Figure 3-42 shows one top-level schematic and different ways to implement the lower-level modules.

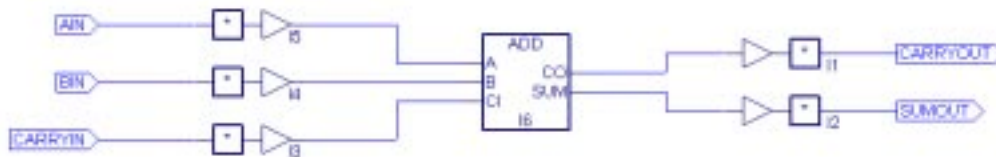


Figure 3-42. Top-level Schematic for Top



**NOTE**

If you are in a lower-level schematic, you can click **Use Data From This Block** button in the New Block Symbol dialog box to automatically create a functional block symbol for the current schematic.

The name of the lower-level schematic must match the block name (schematic) or the interface name (ABEL-HDL) in the upper-level module. This associates the lower-level module with the symbol representing it. For example, the schematic in Figure 3-42 must be named `add.sch`.

The nets in the lower-level schematic correspond to the pin names (schematics) or pin names (ABEL-HDL) in the upper-level module.

The symbol should be a Block symbol. If the symbol used is in a module directory, it can also be a Cell symbol.

Figure 3-43 presents the lower-level ABEL-HDL module for the Add block symbol. It can also be instantiated by the `top.sch` design.

```
MODULE add

"inputs
A,B,CI    pin;

"outputs
CO,SUM    pin;

EQUATIONS
SUM =      A&B&CI
           +!A&!B&CI
           +!A&B&!CI
           +A&!B&!CI;

CO =       A&B
           +A&CI
           +B&CI;

END
```

Figure 3-43. Lower-level ABEL-HDL Module for Add Block Symbol

**NOTE**

It is best to create the lowest-level sources first and then import or create the higher-level sources.

## Hierarchical VHDL Design

The following steps outline how to specify a lower-level VHDL module (`mux2x1vhd.vhd`) and a lower-level schematic (`mux2x1.sch`) in an upper-level schematic (`mux4x1.sch`). Figure 3-44 shows a schematic block and a VHDL module instantiated in another schematic.

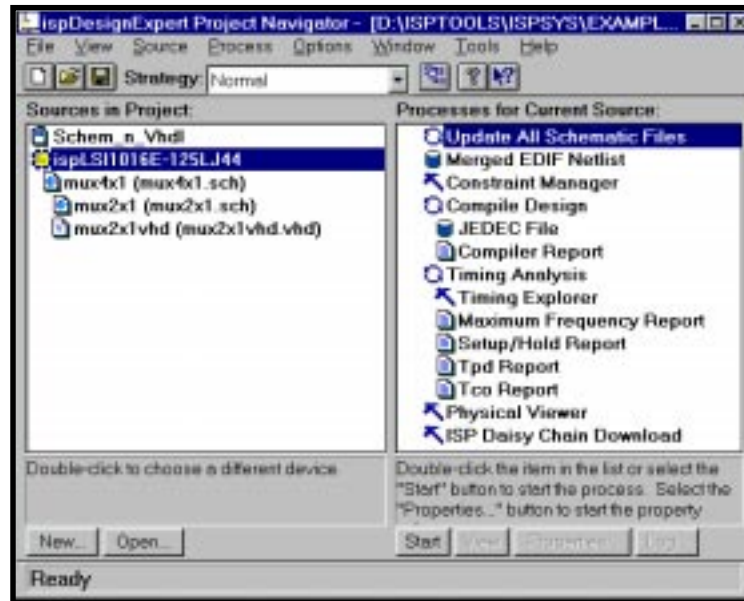


Figure 3-44. Hierarchical Schematic/VHDL Design

*To instantiate a lower-level schematic and a lower-level VHDL module in a schematic:*

1. In the Project Navigator, choose **Window** ⇒ **Schematic Editor** to open the Schematic Editor. The Schematic Editor opens with a new sheet. This schematic is going to be the top-level schematic.  
Next you will place a Block symbol (a functional block) in the schematic that will represent a more-detailed schematic or VHDL module at the next-lower level.
2. To add a block symbol that represents the lower-level schematic (`mux2x1.sch`) in this schematic, choose **Add** ⇒ **Symbol** to open the Symbol Libraries dialog box. Select (Local) from the library list, then select the target symbol `mux2x1`. (If `mux2x1` is not available from the Local symbol library, generate it upon the sub-level module `mux2x1.sch`. Refer to [page 50](#) for more details on how to create a symbol.)
3. Move the pointer back over to the Schematic Editor. Notice that the symbol you selected is attached to the pointer. Place the symbol by clicking on the schematic.
4. Move the cursor back to the Symbol Libraries dialog box and select another target symbol `mux2x1vhd`. Place the symbol in its proper position on the schematic.
5. From the Schematic Editor menu bar, select relevant menu items to add wires, net names, and I/O markers for the schematic.

**NOTE**

You must generate a symbol file for the lower-level schematic or the lower-level VHDL module before you create the upper-level schematic. Refer to the [Schematic Entry User Manual](#) for details on how to generate a symbol.

### Schematic/VHDL Hierarchy Example

Figure 3-45 shows the upper-level schematic `mux4x1.sch` that references a lower-level schematic and a lower-level VHDL module. Following that, Figure 3-46 shows the lower-level schematic `mux2x1.sch`, and Figure 3-47 shows the lower-level VHDL module `mux2x1vhd.vhd`.

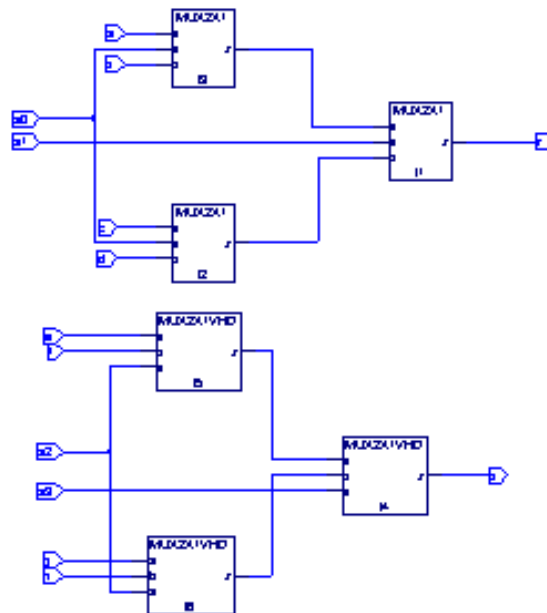


Figure 3-45. Top-level Schematic (mux4x1.sch)

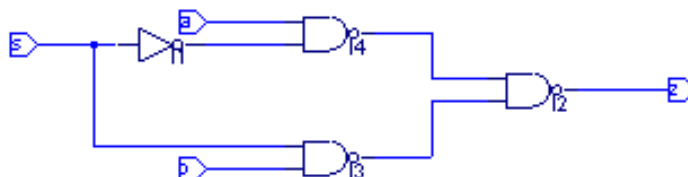


Figure 3-46. Lower-level Schematic (mux2x1.sch)

```
library ieee;
use ieee.std_logic_1164.all;

entity mux2x1vhd is
    port ( z: out std_logic;
          a, b, s: in std_logic );
end;

architecture mux2x1_arch of mux2x1vhd is
begin
    process (s, a, b)
    begin
        case s is
            when '0' =>
                z <= a;
            when '1' =>
                z <= b;
            when others =>
                z <= 'X';
        end case;
    end process;
end mux2x1_arch;
```

Figure 3-47. Lower-level VHDL Module (mux2x1vhd.vhd)

## Hierarchical Verilog HDL Design

The following steps outline how to specify a lower-level VHDL module (`mux2x1vhd.vhd`) and a lower-level schematic (`mux2x1.sch`) in an upper-level schematic (`mux4x1.sch`). Figure 3-44 shows a schematic block and a VHDL module instantiated in another schematic.

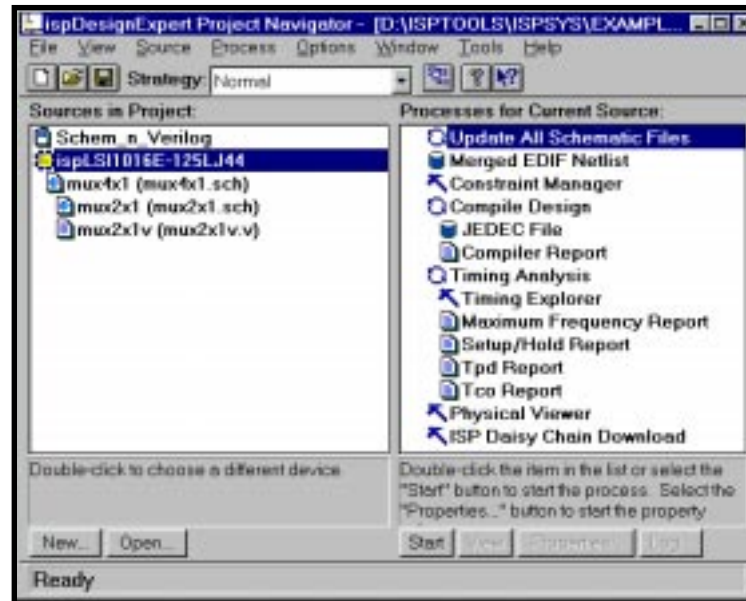


Figure 3-48. Hierarchical Schematic/Verilog HDL Design

*To instantiate a lower-level schematic and a lower-level VHDL module in a schematic:*

1. In the Project Navigator, choose **Window** ⇒ **Schematic Editor** to open the Schematic Editor. The Schematic Editor opens with a new sheet. This schematic is going to be the top-level schematic.  
Next you will place a Block symbol (a functional block) in the schematic that will represent a more-detailed schematic or Verilog HDL module at the next-lower level.
2. To add a block symbol that represents the lower-level schematic (`mux2x1.sch`) in this schematic, choose **Add** ⇒ **Symbol** to open the Symbol Libraries dialog box. Select (Local) from the library list, then select the target symbol `mux2x1`. (If `mux2x1` is not available from the Local symbol library, generate it upon the sub-level module `mux2x1.sch`. Refer to [page 50](#) for more details on how to create a symbol.)
3. Move the pointer back over to the Schematic Editor. Notice that the symbol you selected is attached to the pointer. Place the symbol by clicking on the schematic.
4. Move the cursor back to the Symbol Libraries dialog box and select another target symbol `mux2x1v`. Place the symbol in its proper position on the schematic.
5. From the Schematic Editor menu bar, select relevant menu items to add wires, net names, and I/O markers for the schematic.





**NOTE**

You must generate a symbol file for the lower-level schematic or the lower-level Verilog HDL module before you create the upper-level schematic. Refer to the [Schematic Entry User Manual](#) for details on how to generate a symbol.

**Schematic/Verilog HDL Hierarchy Example**

Figure 3-49 shows the upper-level schematic `mux4x1.sch` that references a lower-level schematic and a lower-level Verilog HDL module. Following that, Figure 3-50 shows the lower-level schematic `mux2x1.sch`, and Figure 3-51 shows the lower-level Verilog HDL module `mux2x1v.v`.

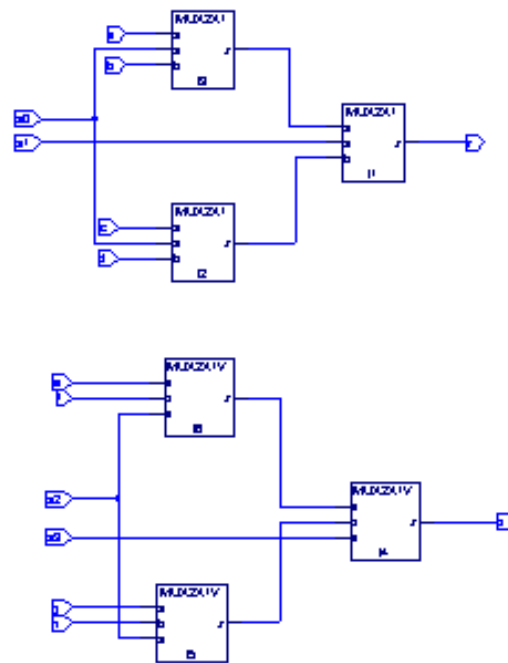


Figure 3-49. Top-level Schematic (mux4x1.sch)

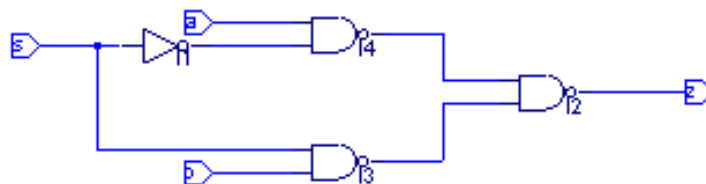


Figure 3-50. Lower-level Schematic (mux2x1.sch)

```
module mux2x1v(a,b,s, z);  
  
output z;  
input a, b, s;  
  
reg z;  
  
always @(a or b or s)  
begin  
    case (s)  
        1'b1: z = b;  
        1'b0: z = a;  
        default: z = 'bx;  
    endcase  
end  
  
endmodule
```

Figure 3-51. Lower-level Verilog HDL Module (mux2x1v.v)

## Hierarchical Design Considerations

Apply the following considerations when using hierarchical design techniques. We take a hierarchical schematic design as an example. The rules implied in the following example are also applicable to other types of hierarchical designs.

### Hierarchical Design Structure

When a symbol is placed in a schematic, the component or subcircuit the symbol represents is added to the circuit. For example, when you place a latch symbol you are actually including the OR gate, inverter, and two AND gates from the latch's schematic.

Figure 3-52 shows (on the left) a 4-bit register (REG4) constructed from four latch symbols (`latch.sym`). The right side of the figure shows the underlying components. The four latch symbols represent a total of eight AND gates, four OR gates, and four inverters.

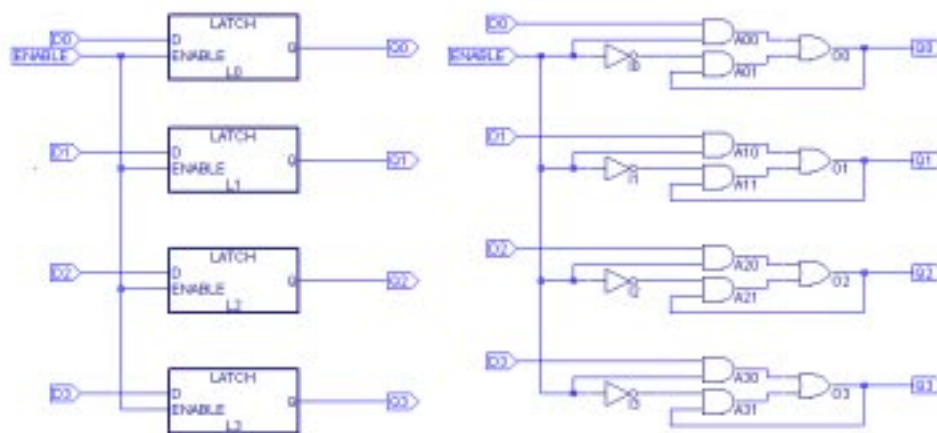


Figure 3-52. Circuit REG4 and its Equivalent Circuit

This hierarchical building process could be repeated by using the Schematic Editor's **File** ⇒ **Matching Symbol** command or **File** ⇒ **Generate Symbol** command (if the corresponding `.naf` file has been generated) to create a symbol for schematic `reg4`, then placing the `reg4` symbol in a higher-level schematic. If you created a schematic for a 16-bit register, `reg16`, by placing four copies of symbol `reg4`, you would be defining a circuit with a total of 64 gates. But instead of having to view 64 gates on a single level, you can work with symbols that represent gates, at the appropriate level of detail.

## Hierarchical Naming

In the `latch` schematic (Figure 3-52), the inverter has the instance name `I1`. In schematic `reg4`, four copies of the symbol `latch` are placed and assigned instance names `L1` through `L4`. Schematic `reg4` therefore contains four copies of inverter `I1`.

The Hierarchy Navigator distinguishes among these otherwise identical inverters by combining the inverter's instance name with the instance name of the latch containing it. The four inverters are therefore named (in the Hierarchy Navigator):

```
L1.I1
L2.I1
L3.I1
L4.I1
```

If we created a 16-bit register by combining four `reg4` symbols, the resulting schematic would represent a new hierarchical level, containing four copies of `reg4` (named `R1` through `R4`). Each copy of `reg4` contains the four inverters as named above. The Hierarchy Navigator would then name the 16 inverters by combining the instance names of the four `reg4` symbols with each of the four instance names of the inverters as follows:

```
R1.L1.I1, R1.L2.I1, R1.L3.I1, R1.L4.I1
R2.L1.I1, R2.L2.I1, R2.L3.I1, R2.L4.I1
R3.L1.I1, R3.L2.I1, R3.L3.I1, R3.L4.I1
R4.L1.I1, R4.L2.I1, R4.L3.I1, R4.L4.I1
```

When you view an individual latch schematic in the Schematic Editor, you see the instance names of the gates, without the hierarchical context. When the schematic becomes part of a larger design and is viewed in the Hierarchy Navigator, the instance names include the hierarchical path (as shown above) to assure their uniqueness.

If you select the top-level schematic source in the Project Navigator, double-click the Hierarchy Browser process associated. The Hierarchy Browser will be opened together with the Hierarchy Navigator (if the source is a schematic) or with the HDL Viewer (if it is an HDL file). All the instances will be listed in the Hierarchy Browser (Figure 3-53).

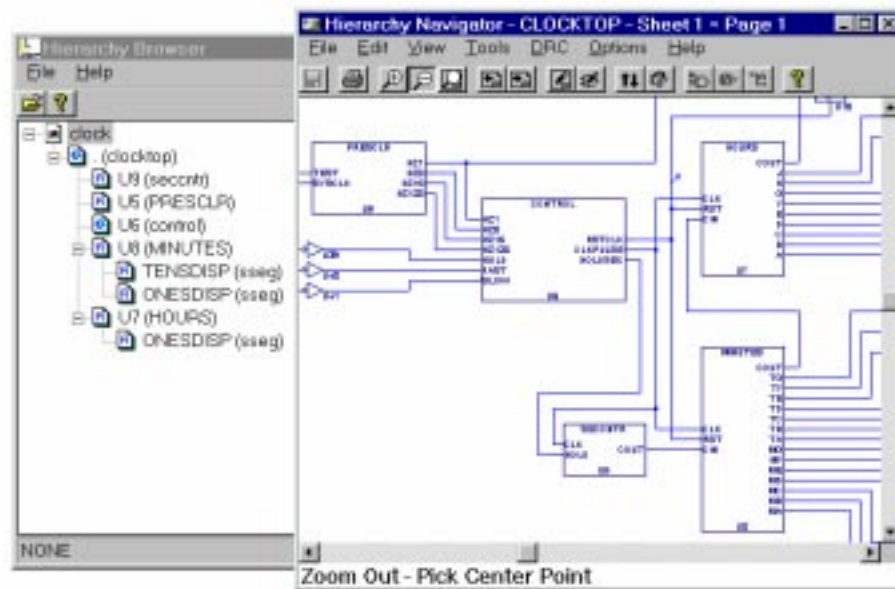


Figure 3-53. Hierarchy Browser

### Nets in the Hierarchy

The schematic definition for the latch circuit contains both local and external nets. The output of the inverter is connected to the AND gate with a local net. Two other local nets connect the outputs of the AND gates to the inputs of the OR gate. Assume these nets have been named N1, N2, and N3. When 16 copies of this circuit are combined in `reg16`, 16 copies of these local nets are created.

The 16 local nets named N1 are individual nets, not branches of the same net, so the Hierarchy Navigator creates a unique name for each. The local net name (N1) is prefixed with the instance name of the schematic where the net is defined. A dash separates the net and instance names. The 16 N1s then become:

`R1.L1-N1, R1.L2-N1... R4.L3-N1, R4.L4-N1`

The `latch` schematic contains three external nets, `D`, `ENABLE`, and `Q`. The symbol pins on the latch connect these nets to the hierarchical level mentioned above.

**Automatic Aliasing of Nets**

When a design is loaded into the Hierarchy Navigator, nets take the name of the highest (top-level) net in the design. That is, the name of top-level net propagates downward through the hierarchy to override the local name. By forcing all nets to the same name, this aliasing feature greatly speeds signal tracing in a multi-level design.

In the preceding example, the net name `D` from the latch is overridden by the higher-level external reference to become `D1, D2, D3....`. This override becomes the reference at all levels of the hierarchy. If, in the suggested 16-bit register, the `D0, D1, D2...` inputs were connected to wires named and marked `Bit0, Bit1, ... Bit15`, these new names take precedence and the `D0, D1, D2...` names would no longer be accessible at any level of the hierarchy.

## Mixed Entry Design

The ispDesignExpert supports mixed entry mode – Schematic and ABEL-HDL; Schematic and VHDL; and Schematic and Verilog HDL; They are mutually exclusive, so you must choose one of the three types when you begin a new project.

### Schematic and ABEL-HDL Mixed Entry

You can use the ispDesignExpert software to create an ABEL-HDL file or a schematic and then connect it to the top-level schematic or the ABEL-HDL file.

*To add a schematic to a project:*

1. With the device selected (Figure 3-54), select **Source** ⇒ **New** from the Project Navigator menu bar. In the dialog box, select Schematic and click **OK**. Enter the file name in the File Name text box. Click **OK**. You are now in the Schematic Editor.

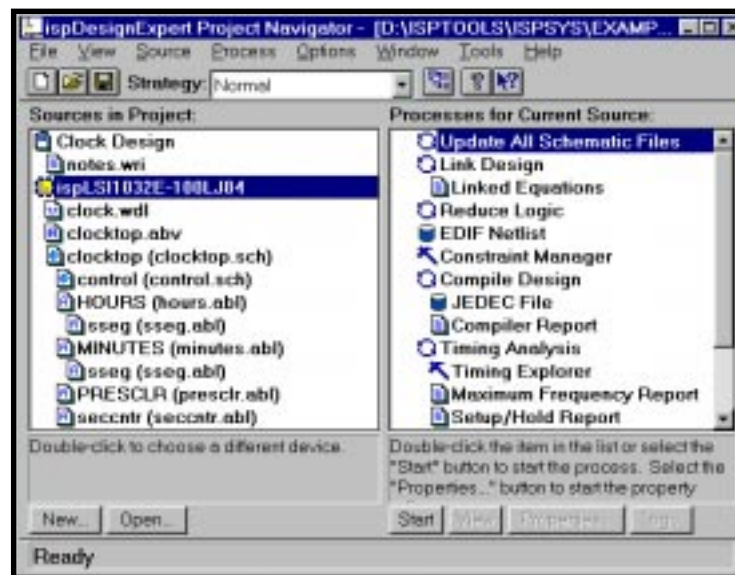


Figure 3-54. ispDesignExpert Project Navigator

2. Add the symbol for the schematic you created earlier. Select **Add** ⇒ **Symbol**. The Symbol Libraries dialog box appears with the Local library selected. The symbols display in the Symbols text box (Figure 3-55). Select the symbol you need to add and place it on your schematic.

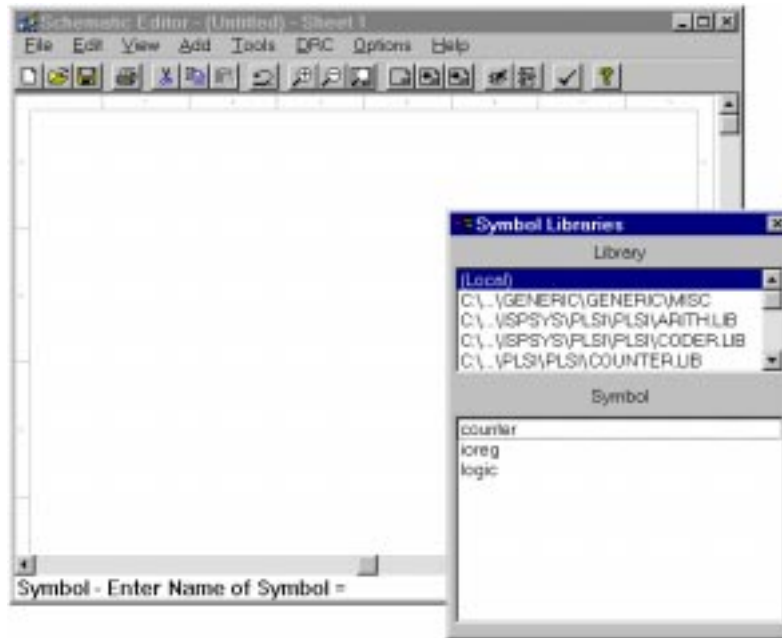


Figure 3-55. Schematic Editor with Symbol Libraries Dialog Box

The next step is to create a top-level symbol for your ABEL-HDL design file. A symbol can be created for any lower-level design module as long as you know its interface. In this way, the actual ABEL-HDL file for your design can be completed later.

3. In the Schematic Editor, select **Add** ⇒ **New Block Symbol**. In the New Block Symbol dialog box (Figure 3-56) that appears, type a name for the top-level symbol in the Block Name text field, relevant input pins in the Input Pins text field, relevant output pins in the Output Pins text field, and relevant bidirectional pins in the Bidirectional Pins text field.

If you want to create a symbol that represents the current schematic, click on the **Use Data From This Block** button. The edit fields are automatically filled with the names of those nets that are labeled with I/O markers. If you have already labeled all nets with I/O markers, you won't have to enter anything manually.

If you want to create a symbol from an existing design source including ABEL-HDL, VHDL, Verilog HDL, or schematic, click on the **Use Data From NAF File** button. The Select File dialog box opens. Select the `.naf` file whose base name is the same as that of the symbol you want to create, the edit fields are automatically filled with the names of those nets defined in the `.naf` file.

When you import, create, or save a design source (`*.abl`, `*.sch`, `*.vhd`, or `*.v`) file in a project, `.naf` file(s) containing all the information of module ports in the design source with the same name as modules in the `.abl`, `.sch`, `.vhd`, or `.v` file will automatically be generated and saved under the current project directory.

Refer to [page 50](#) for more information on how to create a symbol.

4. Click **Run**.



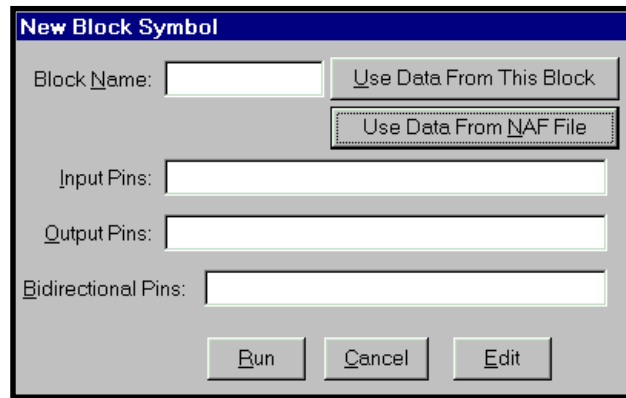


Figure 3-56. New Block Symbol Dialog Box

5. A symbol is added to your Local library and the symbol is attached to your cursor. Place the symbol in the proper position of the symbol you have already added. Click with the right mouse button to place the symbol and to display the Symbol Libraries dialog box (Figure 3-55) again. Close the dialog box.
6. Complete the top-level schematic by adding the necessary wires, net names, and I/O Markers to finish the design. Refer to the instructions for adding nets and symbols from the [Schematic Entry User Manual](#) if you need assistance. When you finish, perform a consistency check and create a matching symbol. Save your design and exit the Schematic Editor.

To verify the correctness and consistency of your top-level design, you can move through the design levels using the Hierarchy Browser or the Hierarchy Navigator feature. A number of editing functions are available through the Navigator.

7. In the Sources in Project list of the ispDesignExpert Project Navigator window, highlight the top-level schematic \*.sch. In the Processes list, double-click on Hierarchy Browser or Navigate Hierarchy. The Hierarchy Browser and the Hierarchy Navigator window appears with your top-level design.

You can traverse the design hierarchy by clicking on the hierarchy tree shown in the Hierarchy Browser. If a relevant module of the selected instance is defined in a schematic source file, it will be opened in the Hierarchy Navigator. If a relevant module of the selected instance is defined in an HDL source file, it will be opened in the HDL Viewer.

8. Select **View** ⇒ **Push/Pop** from the Hierarchy Navigator. The cursor becomes cross hairs. Click on the desired symbol. The Hierarchy Navigator will open the sheet for that symbol at the next level. There will be no sheet for a primitive cell if you click on a symbol that is a primitive cell. If the symbol you choose is generated from an HDL module, the Text Editor will pop up for with that HDL file.
9. Select **File** ⇒ **Exit** to close the Hierarchy Navigator window. You will be asked to save changes you made, if any. Select **File** ⇒ **Exit** to close the Hierarchy Browser window.

To add symbols or macros to your schematic:

1. Choose **Add** ⇒ **Symbol** from the Schematic Editor. The Symbol Libraries dialog box (Figure 3-57) appears.

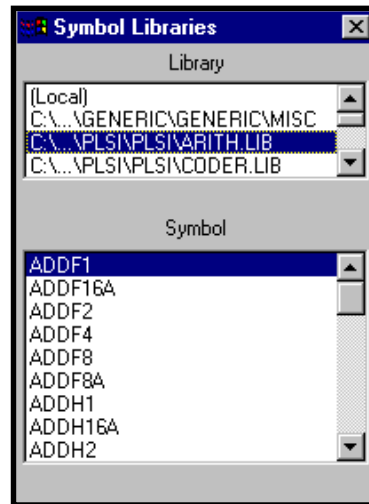


Figure 3-57. Symbol Libraries Window with ispLSI Libraries



**TIP**

Use the Zoom features under the **View** pull-down menu or on the Schematic Editor Toolbar for viewing the opened \*.sch file.

2. In the Symbol Libraries dialog box, select <drive>:\..\\*.lib from the library list, then select the target symbol or macro.
3. Move the pointer back over to the Schematic Editor; notice that the symbol you selected is attached to the pointer. Place the symbol by clicking on the schematic.
4. Move the cursor back to the Symbol Libraries dialog box and select another symbol. Place the symbol in its proper position on the schematic.
5. From the Schematic Editor menu bar, select **Add** ⇒ **Wire**. Click on the output pin of a gate to start the wire. Each successive click will bend the wire (a double-click will end the wire if it is not connected). Connect the wire to the input of a gate.
6. Repeat the above procedure to add other symbols or macros from the Symbol Library.

If necessary, add design attributes and design control properties to the schematic.

## Create an ABEL-HDL Source File

Now you need to create an ABEL-HDL source file and link it to the symbol on the top-level schematic.

7. To create the source, highlight the top source, then select **Source** ⇒ **New**. In the New Source dialog box, choose ABEL-HDL Module and click **OK**. The Text Editor window appears with New ABEL-HDL Source dialog box (Figure 3-58).

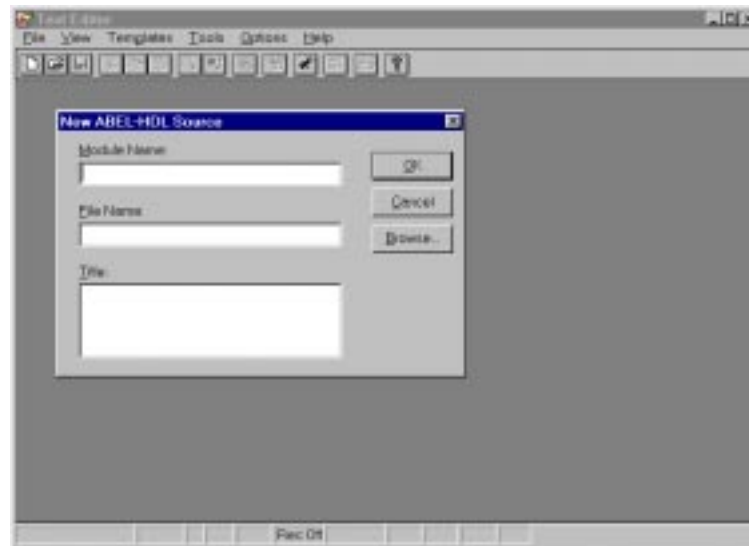
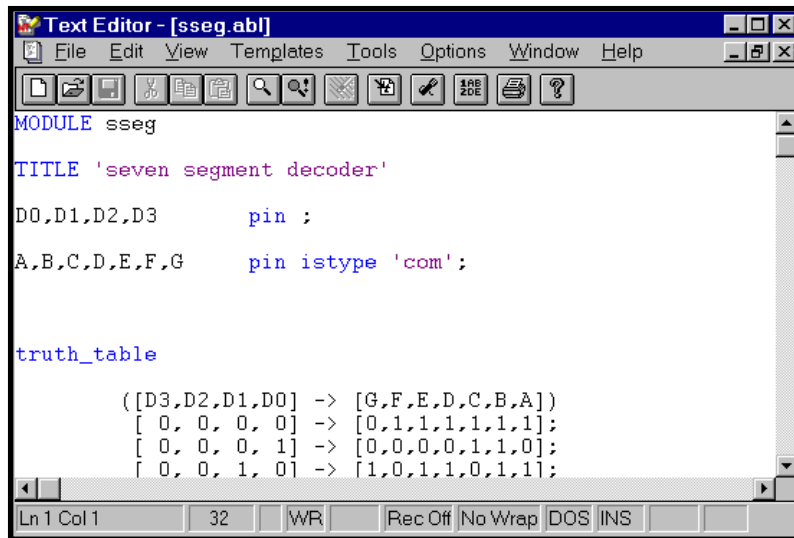


Figure 3-58. Text Editor Window with New ABEL-HDL Source Dialog Box

In order for the file to be linked to the symbol, the Module Name must match the symbol name. The File Name does not need to match the symbol name. Fill in the text fields with relevant contents.

Click **OK**. You will now be in the Text Editor and will see the ABEL-HDL framework already started for you.

8. Enter your design in the Text Editor window. Make sure that you enter it between the TITLE statement and the END statement.



```

Text Editor - [sseg.abl]
File Edit View Templates Tools Options Window Help
MODULE sseg
TITLE 'seven segment decoder'
D0,D1,D2,D3      pin ;
A,B,C,D,E,F,G    pin istype 'com';

truth_table
    ([D3,D2,D1,D0] -> [G,F,E,D,C,B,A])
    [ 0, 0, 0, 0] -> [0,1,1,1,1,1,1];
    [ 0, 0, 0, 1] -> [0,0,0,0,1,1,0];
    [ 0, 0, 1, 0] -> [1,0,1,1,0,1,1];
  
```

Figure 3-59. ispDesignExpert Text Editor with Code Entered

9. Select **File** ⇒ **Save**, then **File** ⇒ **Exit**. Notice in the Project Navigator that the icon next to the source has changed. This means you have an ABEL-HDL file associated with this source and it is linked correctly.

If you have chosen an ispLSI or GAL device, to verify the correctness of your design or to check the hierarchical structure of the whole project, you can use the Hierarchy Browser. Choose the upper-level source in Sources window of the Project Navigator. In the Processes window, double-click the Hierarchy Browser. The Hierarchy Browser will be opened together with the Hierarchy Navigator if the source you selected is a schematic or the HDL Viewer if the source you selected is an HDL file. All the instances included in the top-level design are listed in the Hierarchy Browser upon the hierarchical level.

To a MACH or PAL design, if the source you selected is an HDL file, double-click the associated Hierarchy Browser process from the Project Navigator. The Hierarchy Browser appears together with the HDL Viewer for you to verify the design; if the source you selected is a schematic file, double-click the Navigate Hierarchy process from the Project Navigator. The Hierarchy Navigator displays the underlying logic in the schematic file.

### Compile the ABEL-HDL Source File

10. In the Sources in Project field of the Project Navigator, select the source (\*.abl). In Processes for Current Source, double-click on Reduce Logic. The Compiler will take care of running the Compile Logic process before it executes the Reduce Logic process. When the process is finished, your Navigator should match the one in Figure 3-60.

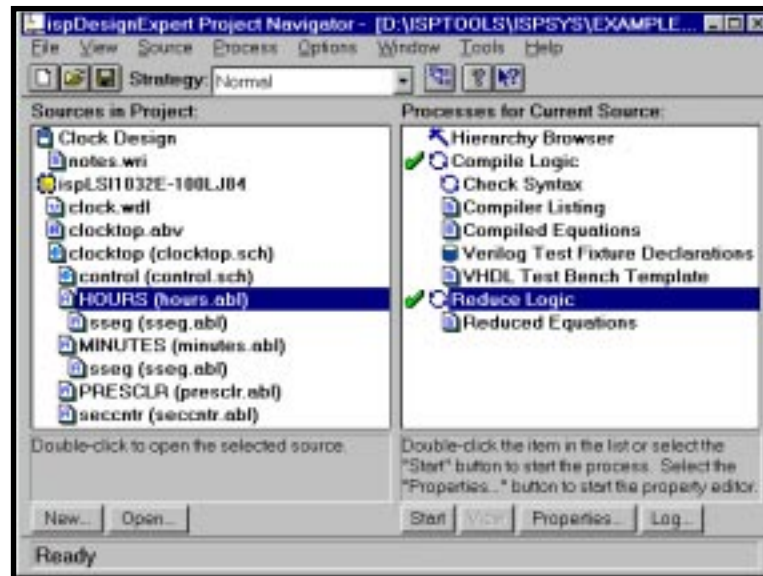


Figure 3-60. Project Navigator after ABEL-HDL Compiling

## Schematic and VHDL Mixed Entry

You can use the ispDesignExpert software to create a VHDL file and then connect it to the schematic from the previous steps for a top-level schematic. Make sure schematics are not instantiated by VHDL sources.

### Create a VHDL Source File

You have previously created a top-level schematic. Now you need to create a VHDL source file to link it to the symbol of the top-level schematic.

*To add a VHDL source file to your project:*

1. Highlight the top source in the Sources window. Then select **Source** ⇒ **New**. In the New Source dialog box, check the project type from the title bar of the dialog box. Choose VHDL Module and click **OK**. The Text Editor window appears with New VHDL Source dialog box.
2. In order for the file to be linked to the symbol, the Module Name must match the symbol name. The File Name does not need to match the symbol name. Fill in the text fields with relevant contents.
3. Click **OK**. You will now be in the Text Editor and will see the VHDL framework already started for you.
4. Enter your design in the Text Editor window.
5. Select **File** ⇒ **Save**, then **File** ⇒ **Exit**. Notice in the Project Navigator that the icon next to the source has changed. This means you have a VHDL file associated with this source and it is linked correctly.

Now you have a sub-level VHDL source file for your mixed Schematic and VHDL design. The next step is to create a symbol for the VHDL source you just created.

*To add a symbol to your top-level schematic:*

6. Open the top-level schematic in the Schematic Editor. Select **Add** ⇒ **New Block Symbol**. The New Block Symbol dialog box appears (Figure 3-61).

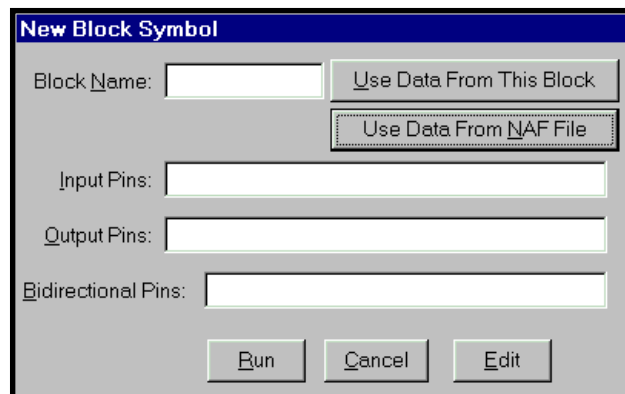


Figure 3-61. New Block Symbol Dialog Box

7. In the New Block Symbol dialog box, type a name for the top-level symbol in the Block Name text field, relevant input pins in the Input Pins text field, relevant output pins in the Output Pins text field, and relevant bidirectional pins in the Bidirectional Pins text field.

If you want to create a symbol that represents the current schematic, click on the **Use Data From This Block** button. The edit fields are automatically filled with the names of those nets that are labeled with I/O markers. If you have already labeled all nets with I/O markers, you won't have to enter anything manually.

If you want to create a symbol from an existing design source including ABEL-HDL, VHDL, Verilog HDL, or schematic, click on the **Use Data From NAF File** button. The Select File dialog box opens. Select the `.naf` file whose base name is the same as that of the symbol you want to create, the edit fields are automatically filled with the names of those nets defined in the `.naf` file.

When you import, create, or save a design source (`*.abl`, `*.sch`, `*.vhd`, or `*.v`) file in a project, `.naf` file(s) containing all the information of module ports in the design source with the same name as modules in the `.abl`, `.sch`, `.vhd`, or `.v` file will automatically be generated and saved under the current project directory.

Refer to [page 50](#) for more information on how to create a symbol.

8. Click **Run**.

Follow [step 5 to 9 on page 89](#) to complete adding the symbol to your design.

## Schematic and Verilog HDL Mixed Entry

You can use the ispDesignExpert software to create a Verilog HDL file and then connect it to the schematic from the previous steps for a top-level schematic. Make sure schematics are not instantiated by Verilog HDL sources. This complete design will be simulated and compiled into an ispLSI device.

Add a Verilog HDL source file to your project in the same way as adding a VHDL source file.

## ispLSI Design Attributes

ispLSI Design Attributes can only be applied to ispLSI designs in ispDesignExpert. They will affect how the ispDesignExpert implements your ispLSI design. When you assign Design Attributes, the compiler uses them as local design assignments. Design Attributes can be set in a schematic, an ABEL-HDL file, or in an EDIF property file.

### Assigning ispLSI Design Attributes in Project Sources

#### In Schematics

Attributes in schematics are used to describe the characteristics or properties belonging to, or associated with, a symbol, pin, or net. Attributes only apply to describing characteristics in schematics.

There are two types of attributes used in the Schematic Editor: symbol and net. Symbol attributes describe features related to a whole symbol. Symbol attributes usually apply only to the symbol on which they appear. Net attributes describe characteristics associated with nets.

Every attribute consists of four components: name, value, modifier, and window. You can assign either symbol or net attributes to a schematic via the Symbol Attribute Editor or Net Attribute Editor dialog box of the Schematic Editor. With the schematic source opened in the Schematic Editor, choose the **Edit ⇒ Attribute ⇒ Symbol Attribute** or **Edit ⇒ Attribute ⇒ Net Attribute** menu item. The Symbol Attribute Editor or Net Attribute Editor dialog box appears prompting you to assign attributes. For example, if you want to add a symbol attribute to a symbol:

1. With the schematic source opened in the Schematic Editor, select **Edit ⇒ Attribute ⇒ Symbol Attribute**. The Symbol Attribute Editor dialog box displays. Select the target symbol from the schematic source.
2. In the Symbol Attribute Editor dialog box, check the List All Attributes option to show all the available symbol attributes. Choose the desired symbol attribute and set the value to the attribute.
3. Click **Go To** or press Enter in the Symbol Attribute Editor dialog box.



## In Lower-level Schematics

Properties controlling design placement and routing are effective in any level of a hierarchical design. You may place a property on any net or symbol in a hierarchical design, and it will be automatically linked and merged into the final netlist to be passed on to the ispEXPERT Compiler. Be cautious when using properties in lower-level modules that are instantiated more than once. Project Navigator does not interpret unique names given in certain properties, such as the user defined path name used in the Asynchronous Path property. If a module uses a property such as this and is instantiated more than once, the netlist file will reflect more than one path using the same name, and this will cause a fatal error during fitter processing. In cases such as this, it is best to modify the lower-level source to bring the required nodes up to the level of instantiation, so that truly unique names can be given for each instance. Consider the following schematics:

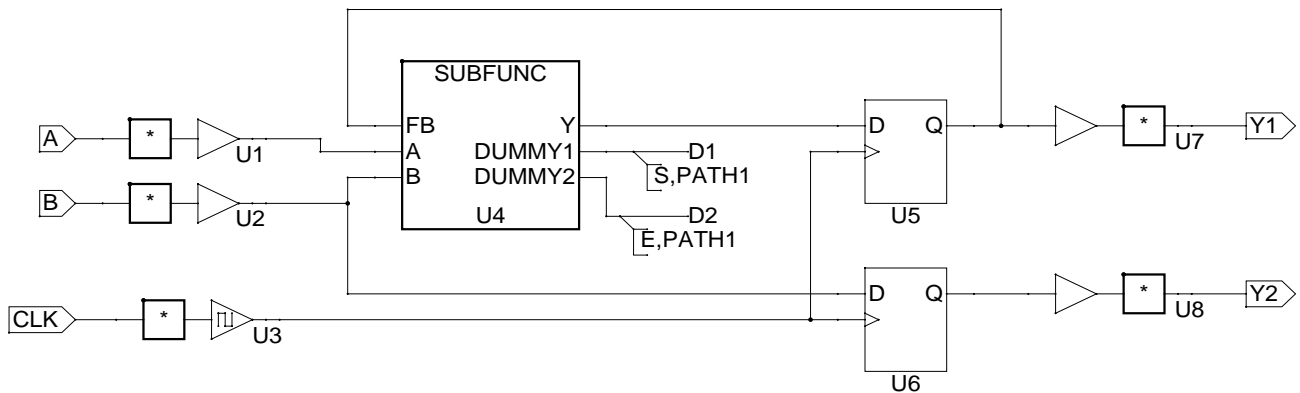


Figure 3-62. Sample Case

This first schematic shows a top-level design with a subfunction called SUBFUNC. There are two DUMMY pins on the symbol from which nets are drawn and the desired attributes (in this case, the beginning and the end of a No-Minimize path) are attached. The schematic for the lower-level function is shown below.

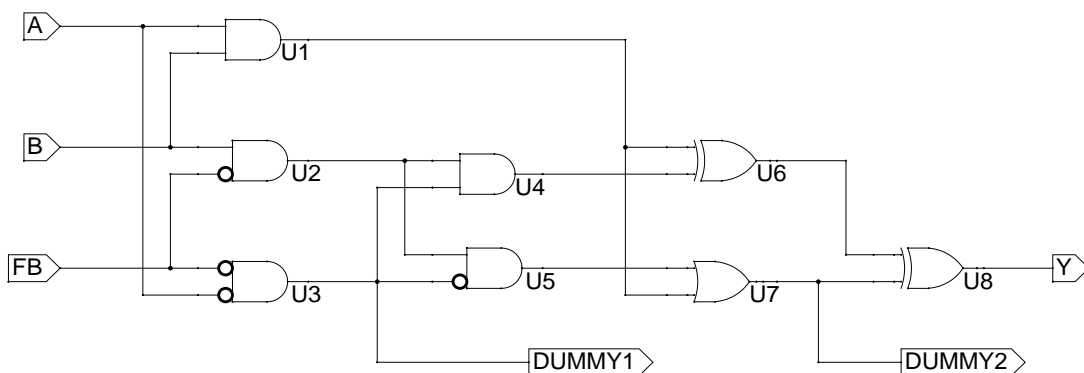


Figure 3-63. Sample Case

The second schematic shows where the paths for no-minimization are taken from. Note that when the design is processed by the ispEXPERT Compiler, the unused nets (DUMMY1 and DUMMY2) will be removed since they do not reach output pins. The properties remain attached and active for the intended nodes.

By attaching all necessary design control properties at the top-level using this technique, it is possible to call a subfunction or lower-level module more than once while maintaining unique path names and attributes for each instance of the subfunction.

## In ABEL-HDL

The same information holds true for ABEL-HDL designs as well. Properties may be assigned at any level of a hierarchical ABEL-HDL design, though properties requiring unique names should not be used in lower-level modules that are instanced more than once. If necessary, lower-level nodes may be brought up to the instancing module for property assignment. Since the properties are used at the upper-level, each module would be assured of having properties containing truly unique names.

### Syntax

```
PLSI PROPERTY `<property_name>[<property_values>]';
```

where

*property\_name*                    the name of the property

*property\_values*                the numerals or strings assigned to the constraint

### Example

```
...
"Inputs
  IN1, IN2, IN3, IN4        pin 15,16,17,18;
  CLKA, CLKB                pin 35,33;

"Properties
PLSI PROPERTY 'CLK CLKA CLK1';
...
```

## ispLSI Attribute Processing

The ispDesignExpert makes use of schematic attributes by embedding them within the file compiled from the schematic as “PLSI PROPERTY” statements. Attributes available for nets are all similar, though the attributes available for a particular symbol are dependent upon the symbol type. The attributes are shown in the table below. The attributes available for a particular symbol are dependent upon the symbol type.

Attribute	Purpose	Syntax	Applies To	Supported Devices
BFM	Places the logic of a specific net into one bigfastmegablock	<i>signal_name</i> <BigFastMegaBlockIndex>	Net	8000
Clock_type	Assigns ispLSI clock type	CLK0, CLK1, CLK2, CLK3, IOCLK0, IOCLK1, FASTCLK, SLOWCLK	Net	All
Group	Marks net to be included in a group	<i>Identifier</i>	Net	1000, 2000, 3000, 6000
Preserve	Marks net to be preserved	Y or N	Net	All
XOR	Defines user-defined XOR gate	ON or OFF	Net	5000V, 8000
SAP/EAP	Marks begin or end of Asynchronous path	<i>S,identifier</i> or <i>E,identifier</i>	Net	All
SCP/ECP	Marks begin or end of Critical path	<i>S,identifier</i> or <i>E,identifier</i>	Net	All
SLP/ELP	Marks begin or end of LowPower path	<i>S,identifier</i> or <i>E,identifier</i>	Net	5000V, 8000
SNP/ENP	Marks begin or end of No Minimize path	<i>S,identifier</i> or <i>E,identifier</i>	Net	All
STP/ETP	Marks begin or end of Turbo path	<i>S,identifier</i> or <i>E,identifier</i>	Net	5000V, 8000
Use_XOR	Tags this gate to be hard XOR in GLB	Y or N	G_XOR or XOR2 symbol	All
Optimize	Changes hard macros to soft macros	ON or OFF	ispLSI hard macros	1000, 2000, 3000, 6000
Protect	Marks primitive (symbol) to be preserved	Y or N	Any symbol	All

Attribute	Purpose	Syntax	Applies To	Supported Devices
Register_Type	Assigns register type	GLB or IOC	Any Register	All
Reserve_Pin	Prevents package pins from being assigned during the Compiler's pin-locking process	<i>pin_number</i>	I/O symbols	1000, 2000, 3000, 5000V, 8000
Critical	Marks output as critical	Y or N	Any output symbol	1000, 2000, 3000, 6000
LOCK	Assigns design signals to specific package pins	<i>pin_number</i>	I/O symbols	All
LOCK_BFM	Places I/O pin(s) into a particular BFM with BFM_index or any BFM without BFM_index	Y or 0 to < <i>maxBigFastMegaBlock-1</i> >	I/O symbols	8000
LOCK_GRP	place I/O pin(s) into a particular GRP with GRP_index or any GRP without GRP_index	Y or 0 to < <i>maxNumGRP-1</i> >	I/O symbols	8000
OpenDrain	Marks output to use open-drain feature	Y or N	Any output symbol	2000E, 2000V, 2000VE, 5000V, 8000
Outdelay	Makes the output buffer slower by 0.5 ns for output or bidirectional pins	Y or N	Any output symbol	5000V
Pull	Marks I/O symbols to have a pullup or datahold attached	UP or OFF or HOLD	I/O symbols	All
PullUp	Marks output to have a pullup attached	Y or N	I/O symbols	1000, 2000, 3000, 6000
SlowSlew	Marks output to use a slow slew rate	Y or N	Any output symbol	All

Attribute	Purpose	Syntax	Applies To	Supported Devices
Voltage	Marks I/O symbols to be programed to VCC or VCCIO	VCC or VCCIO	I/O symbols	5000V

Two other attributes available in the Schematic Editor control pin assignment and schematic logic optimization.

Attribute	Purpose	Syntax	Applies To
SynarioPin	Tags I/O pad with pin number	<i>pin_number</i> or <i>pin_name</i>	I/O symbols
SynarioSrcType	Marks a symbol as representative of a module not in the current project.	External (or blank)	Block symbols

## Precedence of Design Attributes

When several Design Attributes are used in a design, they are all honored as long as they do not conflict or overlap. If they conflict, one or more of the Design Attributes will be ignored, depending on the design. If they overlap, one Design Attribute can override other Design Attributes.

Table 3-1 groups Design Attributes in their order of precedence when relating to the same logic. A Design Attribute with a higher precedence (for example, 1) overrides those with lower precedence (for example, 5). A Design Attribute with the same level of precedence will generally not override another unless they conflict.

Table 3-1. Design Attribute Precedence

Precedence	Design Attribute
1	BFM, LOCK, LOCK_BFM, LOCK_GRP, LXOR2, OPENDRAIN, OPTIMIZE, OUTDELAY, PRESERVE, PROTECT, PULL, RESERVE_PIN, SLOWSLEW, VOLTAGE, XOR
2	SAP/EAP
3	SNP/ENP
4	SCP/ECP, SLP/ELP, STP/ETP
5	CLK, CRIT, GROUP, REGTYPE

# MACH Design Attributes

MACH Design Attributes, applied to MACH designs only in ispDesignExpert, affect how ispDesignExpert implements your MACH designs. After you import these attributes, the Fitter will use as local design assignments in the design implementation process. Design Attributes can be set in a schematic or an ABEL-HDL file.

## Assigning MACH Design Attributes in Project Sources

### In Schematics

Attributes in schematics are used to describe the characteristics or properties belonging to, or associated with, a symbol, pin, or net. Attributes only apply to describing characteristics in schematics.

There are two types of attributes used in the Schematic Editor: symbol and net. Symbol attributes describe features related to a whole symbol. Symbol attributes usually apply only to the symbol on which they appear. Net attributes describe characteristics associated with nets.

Every attribute consists of four components: name, value, modifier, and window. You can assign either symbol or net attributes to a schematic via the Symbol Attribute Editor or Net Attribute Editor dialog box of the Schematic Editor. With the schematic source opened in the Schematic Editor, choose the **Edit ⇒ Attribute ⇒ Symbol Attribute** or **Edit ⇒ Attribute ⇒ Net Attribute** menu item. The Symbol Attribute Editor or Net Attribute Editor dialog box appears prompting you to assign attributes. For example, if you want to add a symbol attribute to a symbol:

1. With the schematic source opened in the Schematic Editor, select **Edit ⇒ Attribute ⇒ Symbol Attribute**. The Symbol Attribute Editor dialog box displays. Select the target symbol from the schematic source.
2. In the Symbol Attribute Editor dialog box, check the List All Attributes option to show all the available symbol attributes. Choose the desired symbol attribute and set the value to the attribute.
3. Click **Go To** or press **Enter** in the Symbol Attribute Editor dialog box.

**In ABEL-HDL**

ABEL-HDL source files have their own syntax for describing characteristics:

```
MACH_LOCATION (PinName, PinType, PinNo, BlockNo, SegmentNo);
```

**Example**

```
module cntbuf2
title 'Counter and Bidirectional Buffer'

Library 'mach';

"MACH_LOCATION(PinName, PinType, PinNo, BlockNo, SegmentNo);
"PinType:          Input, Output, Node
"PinNo,BlockNo,SegmentNo:* = Any pin, block or segment
"                  - = Not applicable(for BlockNo or SegmentNo)
MACH_LOCATION(Clk,input,11,-,-);
MACH_LOCATION(Clr,input,* ,B,-);

"MACH_GROUP(GroupNo, BlockNo, SegmentNo, NoOfSignals,
Signal_list);
"PinNo,BlockNo,SegmentNo:* = Any block or segment
"                  - = Not applicable(for BlockNo or SegmentNo)
MACH_GROUP(GroupA,* ,-,4,Q0:Q1:Q2:Q3);
MACH_GROUP(GrpA,B,-,2,A0:A1);
MACH_GROUP(GrpB,* ,-,2,B0:B1);

"MACH_SLEW_SLOW(NoOfSignals, Signal_list); - Default unlisted
"signals to FAST Slew rate
"MACH_SLEW_FAST(NoOfSignals, Signal_list); - Default unlisted
"signals to SLOW Slew rate
MACH_SLEW(SLOW,2,Q0:Q1);

"MACH_RESERVE(PinType, PinNo, Output_state);
"PinType:          Input, Output, Bidir
"Output_state:    - = Not applicable (Output_state)
MACH_RESERVE(Output,8,out_high);
MACH_RESERVE(input,13,-);
```

```

" Inputs
  Clk,Clr pin;
  Dir, OE          pin 5,4;

" Outputs
  Q3,Q2,Q1,Q0      pin ;
  Q3,Q2,Q1,Q0      ISTYPE 'reg,buffer';

  A1,A0,B1,B0      pin ;
  A1,A0,B1,B0      ISTYPE 'com';

" Special Constants
  C, X, Z ,z = .C. , .X., .Z.,.Z.;

" Set assignments
  A      = [A1,A0];
  B      = [B1,B0];
  Count = [Q3..Q0];

equations
  Count.OE = !OE;
  Count.Clk = Clk;

  Count := (Count.fb + 1) & !Clr;

@include 'counter.inc';

equations
  A.OE = Dir;
  B.OE = !Dir;

  A = B.pin; "Combinatorial signal can only have pin feedback
  B = A.pin; "for this device.

@include 'buffer.inc';

end

```



## Chapter 4 *Design Implementation*

---

After you create a design, you can use compiler options and device options to control how the compiler analyzes, synthesizes, partitions, places, and routes your design using the physical resources of a selected target device. These options represent design goals and restrictions that you want but that may not be critical to the successful operation of a design. For an ispLSI design, you can use the Constraint Manager to set or edit the constraints. If it is a MACH design, you can use the Constraint Editor to finish the assignment and modification of the constraints.

Once you finish setting the design, the ispDesignExpert software uses the compiler control options, the design attributes, and pin assignments to fit your design into the device according to your design constraints and the device architecture.

This chapter covers the following information:

- Constraint Manager for ispLSI
- Constraint Editor for MACH
- Pin Locking for GAL/PAL Devices
- Compiler Properties
- Compiling/Fitting the Design

## Constraint Manager for ispLSI Designs

The ispLSI Constraint Manager lets you set pin, symbol, and net attributes for the signals, the pins, and the symbols in your ispLSI designs. You can also read in an existing Property File or save the attributes as a Property File.

Besides the Constraint Manager, you can still assign attributes directly to a source file. For information on how to assign attributes in a source file such as a schematic or an ABEL-HDL file, refer to [Chapter 3, “Design Entry.”](#)

The Constraint Manager consists of the Design Browser, a Pin Attributes Table, a Net Attribute Table, and a Symbol Attributes Table.

If your EDIF design contains attributes, or if you read in an EDIF Property File when you created your project, or if you set pin attributes using the Compiler Graphic User Interface (GUI), those values display in the attribute tables. If you make changes to the attribute tables, the new values are reflected in your project. You can create a Property File by selecting the **File** ⇒ **Save Property As** menu item.

You can open an existing Property File by selecting the **File** ⇒ **Open Property** menu item. The values from the Property File being opened are reflected in the property tables. You can save changes to the open file using **File** ⇒ **Save Property**.

### Main Window

Use the main Constraint Manager window to set attribute values for signals, the pins, and the symbols.

*To invoke the Constraint Manager:*

1. Select the target ispLSI device from the Sources window of the Project Navigator.
2. In the Processes window, double click Constraint Manager. The main window of the Constraint Manager appears (Figure 4-1).

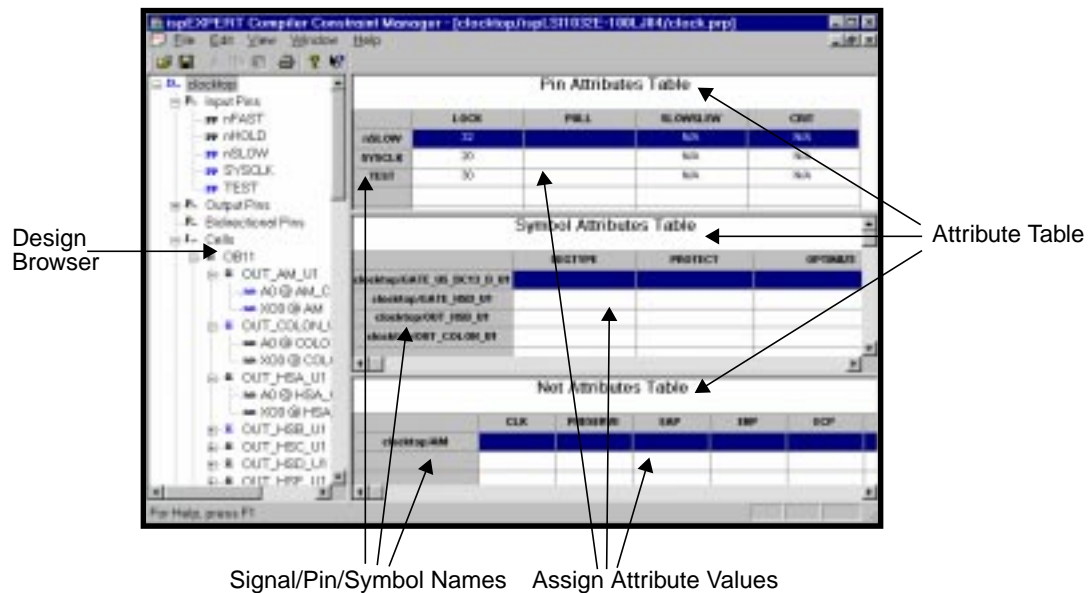


Figure 4-1. Constraint Manager Main Window

## Design Browser

The Design Browser shows all the Input Pins, Output Pins, Bidirectional Pins, Nets, and Cells. Click on the + to expand the list to see signal, pin, or symbol names.

To add a signal, pin, or symbol to an Attribute Table, double click on the signal, pin, or symbol name. To add all the signals, pins, or symbols in a category to an Attribute Table, double click on the category name (Input Pins, Output Pins, Bidirectional Pins, Nets, and Cells).

## Pin Attributes Table

The Pin Attributes Table allows you to set attributes for input pins, output pins, and bidirectional pins in your ispLSI design. The Pin Attributes Table displays when you click on a pin, or input, output, or bidirectional pin category in the Design Browser.

Depending on the device used in the design, you can set the following pin attributes:

- CRIT
- LOCK
- LOCK\_BFM
- LOCK\_GRP
- OPENDRAIN
- PULL
- SLEW
- SLEW\_RATE
- VOLTAGE

The Reserved Pin attribute must be set from the Assign Pin Attributes dialog box of the ispEXPERT Compiler.

### **Net Attributes Table**

The Net Attributes Table lets you set net and path attributes for the nets in your ispLSI design. The Net Attributes Table displays when you click on a signal or nets category in the Design Browser.

Depending on the device used in the design, you can set the following net and path attributes:

- CLK
- GROUP
- PRESERVE
- EAP
- ECP
- ELP
- ENP
- ETP
- XOR

### **Symbol Attributes Table**

The Symbol Attributes Table lets you set attributes for the symbols in your ispLSI design. The Symbol Attributes Table displays when you click on a symbol or cells category in the Design Browser.

Depending on the device used in the design, you can set the following symbol attributes:

- OPTIMIZE
- PROTECT
- REGTYPE

### **Assign Attribute Values**

The table cells are used to assign the attribute values. Click the right mouse button on a cell to display a list of the attribute values that can be used for that attribute. You can set multiple cells by holding the Control key down when you select them. You can select all cells in a column by clicking on the attribute name in the column heading.

If no values are listed, selected Edit from the right button menu to enter a value. Then type in the values.

If an attribute is not supported for a particular pin or signal or symbol type (such as input pins), N/A displays in the cell.

## Constraint Editor for MACH Designs

The MACH Constraint Editor lets you select pin and node assignments, group assignments, pin reservation, power level settings, output slew-rate and JEDEC file options (such as User Signature and Zero-Hold Time bits). This window reads the constraint file and displays the constraint settings. Modifications to the constraint file are made via the function dialogs, such as Location Assignment, Group Assignment, Pin Reservation, JEDEC File Options, Assign Power Level and Output slew rate Control.

Note that the Constraint Editor implements a simple error checking, to ensure that the user assignments or constraints are applicable to the selected device, and that there are no conflicting assignments. If the user constraints do not apply to the selected device, or are conflicting with the selected device, Constraint Editor will display these constraints in red (user can choose the color via the **Option** ⇒ **Color** command). For instance, if you change the device type after specifying some pin assignments, the Constraint Editor will display the non-applicable pin assignments in red. You can delete these constraints in the “Loc” function dialog boxes or via the Clear Project Assignments dialog box.

The features of the Constraint Editor and all its function dialog boxes are available only if they are applicable to the selected device.

### Main Window

Use the main window to set attribute values for signals and pins.

*To invoke the Constraint Editor:*

1. Select the target MACH device from the Sources window of the Project Navigator.
2. In the Processes window, double click Constraint Editor. The main window of the Constraint Editor appears (Figure 4-2).

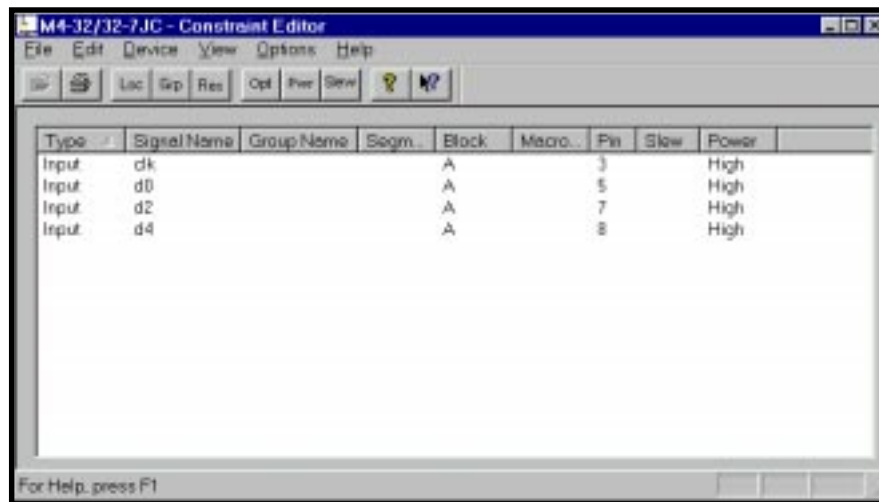


Figure 4-2. Constraint Editor Main Window

Note that the design signal labels are only listed exclusively in the Location and Group Assignments dialog boxes. For instance, if signal “ld\_new\_alarm\_time” is assigned to a pin via the Location Assignment dialog box, then this signal will not be listed in the Group Assignment dialog, and vice versa. This feature prevents conflicting assignments from being implemented.

## Location Assignment

Use the Location Assignment dialog box to assign input and output pins, and buried nodes.

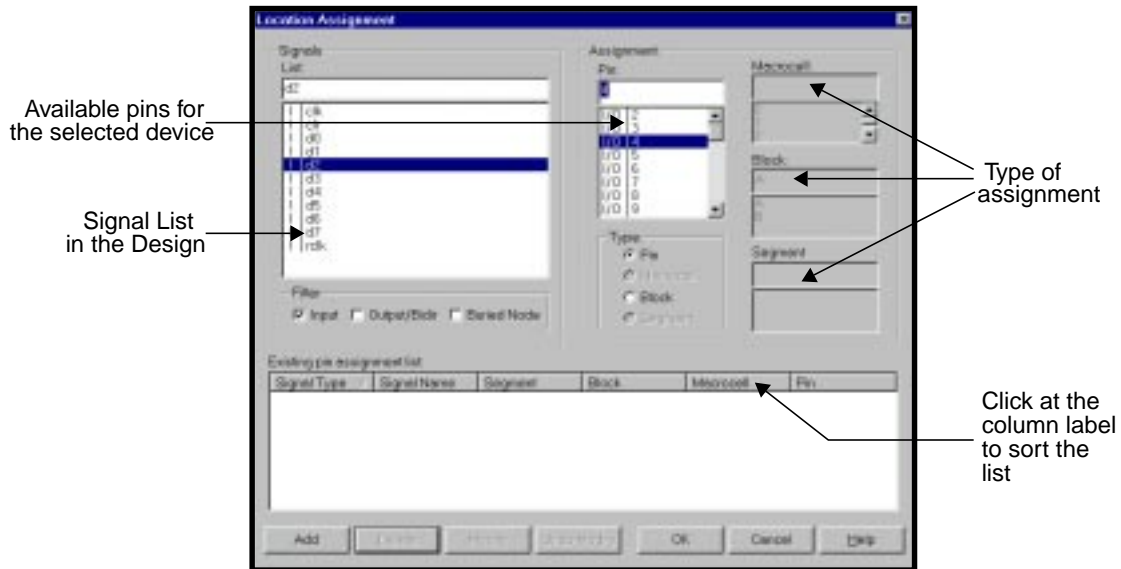


Figure 4-3. Location Assignment Dialog Box

*To make a location assignment:*

1. Select a signal from the Signal List. The signal list can be controlled using the Filter check boxes.
2. Select the assignment type (Pin, Macrocell, Block, or Segment).
3. Click **Add**.

You can make modifications to the existing list with the **Delete**, **Modify** and **Undo Modify** buttons. The **Cancel** button discards all new edits, and returns to the previous assignment list (list prior to opening the dialog box).

## Group Assignment

Use the Group Assignment dialog box to assign a group to a Block or Segment.

- For MACH 1, 2, and 4 devices:

### **Block=Any:**

If this option is selected, all the signals in a group are grouped together in a single Block or Segment, but it could be any block or segment in the device.

- For MACH 5 devices

**Block=Any, Segment=Any:** If this option is selected, all the signals in a group are grouped together in a single Block of the “Any” Segment. Note that if you specify “Any” Segment, by default, you can only select “Any” for Block.

**Block=Any, Segment=Number:** If this option is selected, all the signals in a group are grouped in a single Block in the specified Segment.

The **Clear** button clears the current selected signal list and group name.

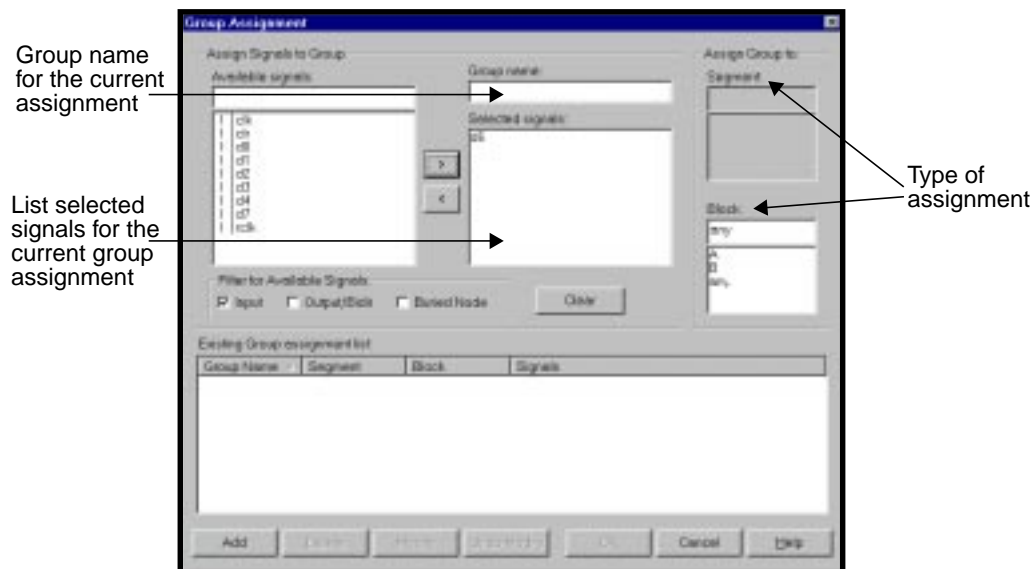


Figure 4-4. Group Assignment Dialog Box

## Pin Reservation

Pin Reservation constraints are “soft constraints.” If the design failed to fit, the Fitter ignores the pin reservation constraints. The software issues a warning message to the ERROR or LOG file when the constraint is discarded.

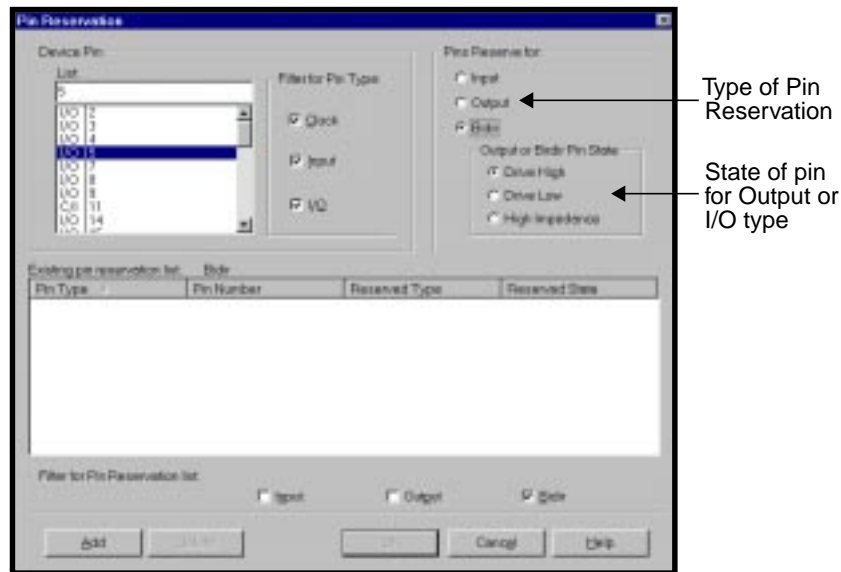


Figure 4-5. Pin Reservation Dialog Box

## JEDEC File Options

Depending on the device selected, you can select these special features:

**Zero Hold-Time:** This is a MACH 4 feature that sets the Zero-Hold Time (ZHT) fuse. This fuse controls the time delay associated with loading into all I/O cell registers or latches. When turned ON, ZHT increases the data path setup delays to input storage elements, matching equivalent delays in the clock path. When turned OFF, setup time to the input storage element is minimized.

**User Signature:** This feature allows you to enter a design identifier such as revision number, design codes, etc. The design identifier is programmed into the device and can be read from the device, regardless of the state of the device’s security bit.

**Pull-up:** All MACH 4 and MACH 5 devices have a Bus-Friendly input structure that weakly holds an input at its last driven state. When you select this option, ispDesignExpert disables the Bus-Friendly structure on all I/O pins and enables the pull-up resistor structure.



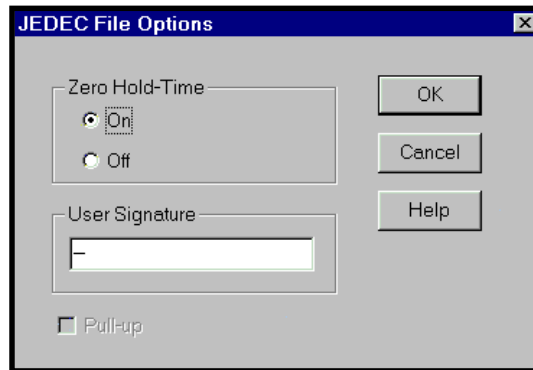


Figure 4-6. JEDEC File Options Dialog Box

### Assigning Power Level

This dialog box lets you set the power level for the macrocell or block, depending on the device selected. Power level for the unassigned macrocell signals or blocks are set to the “Default Power Level” setting.

- |                   |  |
|-------------------|--|
| For MACH 1 and 2: | Power level can be assigned to individual macrocell signal, so the signal labels will be listed in the “Default Power Level” list. |
| For MACH 4:       | Power level can be assigned to the block number (for example A, B, C, D, etc.)   |
| For MACH 5LV:     | Power level can be assigned to the block number in the segment (such as 1A, 1B, 2A, etc.) The number indicates the segment number. |

Note that the “Default Power Level” is set to HIGH to reduce timing delays. Any changes to the power level will affect the design timing.

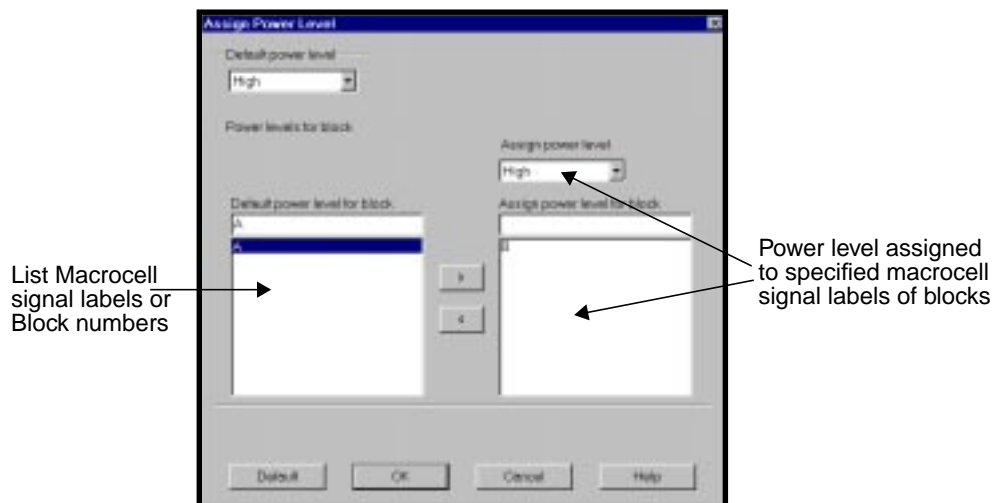


Figure 4-7. Assigning Power Level Dialog Box

## Output Slew Rate Control

This dialog box lets you set the output slew rate of the device. By default, the output slew rate is set to FAST. If the output is set to SLOW slew rate, it will delay the design timing.

The **Default** button discards the current changes and return back to the previous settings before the dialog box is opened.

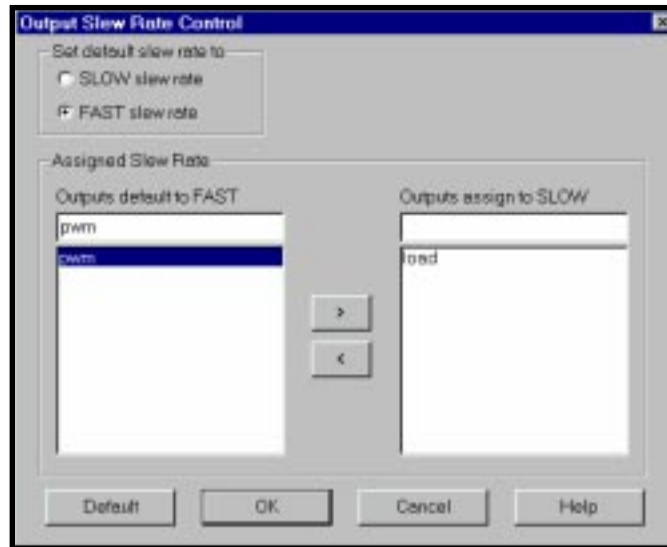


Figure 4-8. Output Slew Rate Control Dialog Box

## Pin Locking for GAL/PAL Devices

For GAL/PAL devices, if you have a VHDL or Verilog HDL module as the top-level source in your project, you can only specify pin locking information in the design sources (.vhd or .v).

### VHDL Designs

If you have a VHDL module as the top-level source in your project, modify the VHDL file adding pin locking information before fitting your design to the target device.

#### Syntax

```
attribute loc : string;
attribute loc of <signal_name> : signal is "P<pin_number>";
```

#### Example

The following example shows how to lock an input signal a to pin 5.

```
library ieee;
use ieee.std_logic_1164.all;

entity mux2x1vhd is
port ( z: out std_logic;
      a, b, s: in std_logic );

attribute loc: string;
attribute loc of a : signal is "P5";

end;

architecture mux2x1_arch of mux2x1vhd is
begin
  process (s, a, b)
  begin
    case s is
      when '0' =>
        z <= a;
      when '1' =>
        z <= b;
      when others s=>
        z <= 'X';
    end case;
  end process;
end mux2x1_arch;
```

## Verilog HDL Designs

If you have a Verilog HDL module as the top-level source in your project, modify the Verilog HDL file adding pin locking information before fitting your design to the target device. However, there are two synthesis tools supported in the ispDesignExpert software, Synplicity Synplify and Exemplar LeonardoSpectrum. You need to vary pin locking syntax in a Verilog HDL file with the synthesis tool you are going to use.

### Syntax for Synplify

```
input <signal_name> /*synthesis loc="P<pin_number>"*/;
output <signal_name> /*synthesis loc="P<pin_number>"*/;
```

### Example

The following example shows how to lock an input signal a to pin 5.

```
module mux2x1v(a,b,s, z);

output z;
input a /* synthesis loc="P5" */, b, s;

reg z;

always @(a or b or s)
begin
    case (s)
        1'b1: z = b;
        1'b0: z = a;
        default: z = 'bx;
    endcase
end

endmodule
```

## Syntax for LeonardoSpectrum

```
//exemplar attribute <signal_name> loc P<pin_number>
```

### Example

The following example shows how to lock an output signal z to pin 25.

```
module mux2x1v(a,b,s, z);

output z;
input a, b, s;

reg z;

//exemplar attribute z loc P25

always @(a or b or s)
begin
    case (s)
        1'b1: z = b;
        1'b0: z = a;
        default: z = 'bx;
    endcase
end

endmodule
```

## ispLSI Compiler Properties

The Compiler Properties dialog box, for ispLSI devices only, lets you set options that can control the compiler process of your design, specify global design control properties, and User Electronic Signature. Refer to the [ispEXPERT Compiler User Manual](#) for more details on design compilation options.

You can invoke the Compiler Properties dialog box in these ways:

- Highlight the device icon in the Sources window of the Project Navigator and select the Compile Design process in the Processes window. Choose the **Process** ⇒ **Properties** menu item or click the **Properties** button at the bottom of the Project Navigator.
- With an ispLSI project opened in the Project Navigator, choose **Tools** ⇒ **Compiler Properties**.



### NOTE

Invoke the ispEXPERT Compiler Graphic User Interface if you need to specify a post-route pin file (.ppn). See the [ispEXPERT Compiler User Manual](#) for more information.

There are three tabs in the Compiler Properties dialog box.

## Settings

The Settings tab (Figure 4-9, Figure 4-10, and Figure 4-11) lets you set options that can control how the compiler processes your design. Not all options are available for all devices. In addition, the values for some of options will vary upon the different device you choose.

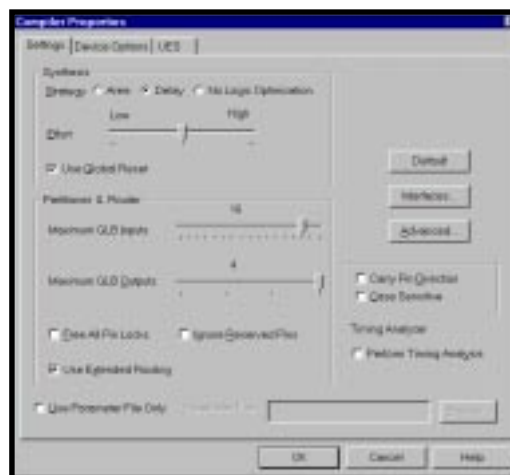


Figure 4-9. Compiler Properties Dialog Box - Settings (for ispLSI1000, 2000, and 3000 devices)

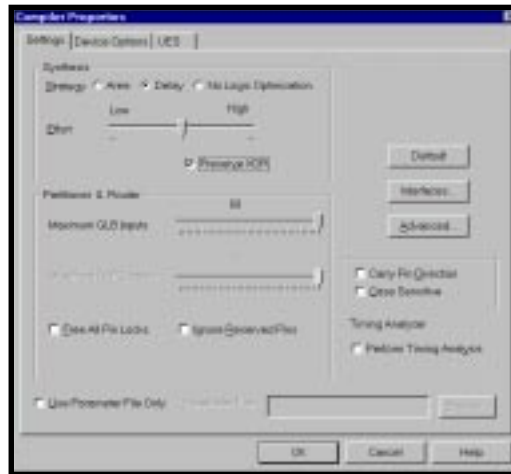


Figure 4-10. Compiler Properties Dialog Box - Settings (for ispLSI 5000 and 8000 devices)

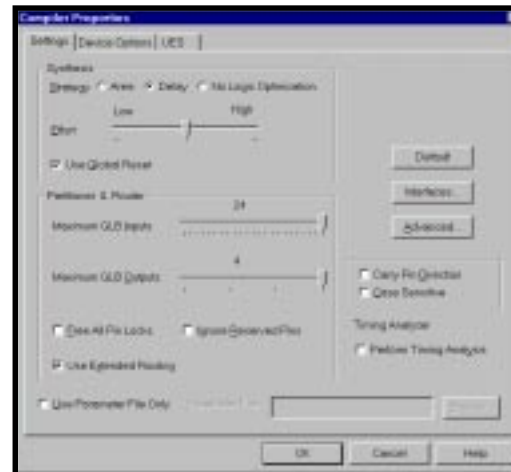


Figure 4-11. Compiler Properties Dialog Box - Settings (for ispLSI 6000 devices)

## Strategy

The options of the Strategy option are:

- Area                               Maximum resource utilization
- Delay                               Maximum performance
- No Logic Optimization       Minimal change to the design

The following points are important to remember when you use the Strategy option:

- For logic level considerations, DELAY offers the least number of logic levels (default).
- AREA optimizes for device utilization and consequently may use more logic levels.
- NO LOGIC OPTIMIZATION bypasses the synthesis and optimization stage and maps your design directly into the ispLSI architecture.

## Effort

The Effort option provides three different optimization levels:

- Low
- Medium
- High

The higher the effort, the longer the runtime and the larger the memory requirement. While a higher effort level usually leads to better results, this is not guaranteed. Try different effort levels to find the best result for a specific design. The default value is Medium.

## Use Global Reset

The User Global Reset option makes the global reset pin available for use by the compiler. The default is ON.

For ispLSI 5000V and 8000 device families, Use Global Reset is always ON. If a registered preset or reset is not found in an ispLSI 5000V or 8000 device design, the fitter will automatically assign global reset.

## XOR

The XOR option preserves user-defined XOR gates on primary output nodes during the logic optimization process.

The XOR compiler option, when applied globally, preserves all user-defined XOR gates on primary output nodes. To preserve other user-defined gates, apply a local attribute. When the global attribute is turned off (default), all user-defined XOR gates are expanded unless a local XOR attribute has been specified.

This option is only supported on ispLSI 5000V and 8000 devices.



### Maximum GLB Inputs

The Maximum GLB Input option specifies the maximum number of GLB inputs the compiler is allowed to use for each GLB. This applies to every GLB in your design.

The values and default for each device family are shown in the following table:

Device Family	Range	Default Value
ispLSI1000 and 2000	2 to 18	16
ispLSI 3000 and 6000	2 to 24	24
ispLSI 5000V	34 to 68	68
ispLSI 8000	22 to 44	42

### Maximum GLB Outputs

The Maximum GLB Outputs option specifies the maximum number of GLB outputs the compiler is allowed to use for each GLB. This applies to every GLB in your design.

The values and default for each device family are shown in the following table:

Device Family	Range	Default Value
ispLSI 1000, 2000, 3000, and 6000	1 to 4	4
ispLSI 5000V	32	32
ispLSI 8000	20	20

## Free All Pin Locks

The Free All Pin Locks option instructs ispDesignExpert to either ignore or honor the pin locking attributes in your design. This option is the same as the IGNORE\_FIXED\_PIN parameter file option.

- Turn on Free All Pin Locks so the compiler will ignore the pin locking attributes in your design. This allows the compiler to place I/Os anywhere on the device, without restriction.
- Turn off Free All Pin Locks so the compiler will honor any pin locking attributes on your design. The default is Off.

The pin assignments from the source design are read into the project when it is created; they are not read again at any other time unless you specifically update the project. These assignments may be edited in the Assign Pin Locations window of the ispEXPERT Compiler. These assignments shown in the Assign Pin Locations window are the ones used when you compile. If you want to compile with all pins free, set this option to On.

## Ignore Reserved Pins

The Ignore Reserved Pins option instructs the ispDesignExpert to either ignore or honor the reserved pin assignments in your design.

- Turn on Ignore Reserved Pins so the compiler will ignore the reserved pin assignments in your design. This allows the compiler to place I/Os anywhere on the device, without restriction.
- Turn off Ignored Reserved Pins so the compiler will honor any pin assignments in your design. The default is Off.

## Use Extended Routing

The Use Extended Routing option instructs the ispDesignExpert to use a complete routing cycle in an attempt to route a design, or question the user if routing time is very long. The default is On.

- Turn on Use Extended Routing to instruct the ispDesignExpert to continue until the design is fully routed or until routing fails.
- Turn off Use Extended Routing to instruct the ispDesignExpert to question the user if routing time is very long. You can decide to continue or stop and relax some design constraints before trying to compile again.

## Carry Pin Direction

The Carry Pin Direction option maintains user-specified pin directions in any simulation output. The default is Off.

- Carry Pin Direction On attempts to maintain user-specified pin directions for 3-state outputs into any simulation output netlist. The 3-state outputs can be connected to external output pins or bidirectional pins.
- Carry Pin Direction Off converts 3-state outputs to external output pins in any simulation output netlist.

## Case Sensitive

The Case Sensitive option enables ispDesignExpert to treat identifiers, such as pin names and net names, as case-sensitive or case-insensitive. The default is Off.

- Turning on Case Sensitive results in consideration of identifiers as case-sensitive.
- Turning off Case Sensitive results in consideration of identifiers as case-insensitive.

## Timing Analyzer

The Timing Analyzer option allows you to run the Timing Analyzer during the compiler step or turn off the Timing Analyzer.

- If you specify TIMING\_ANALYZER Off, you turn off the Timing Analysis during the compilation and a timing report is not generated.
- If you specify TIMING\_ANALYZER On, the Timing Analyzer runs during the compilation. This generates the Clock Frequency Table, the Setup and Hold Table, the Tco Table, the Tpd Table. This option does not let you select the Path Analysis information and report types that you can set when you set the Timing Analyzer Settings and run Timing Analysis from the **Tools** menu of the ispEXPERT Compiler as a separate procedure.

## Parameter File

The Parameter File option specifies the name of an optional Parameter File for the compiler to use for compilation specifications.

The Parameter File contains alternate sets of Compiler Options and Device Options that can be used to run different iterations of your design. You can create this file using an ASCII text editor. The Parameter File name should have a `.par` extension and must be different from the design name.

When a Parameter File is used, all relevant files and parameters are passed to the appropriate ispDesignExpert software modules. Once a design is routed, ispDesignExpert merges the various report files into a report file, `design.rpt`, and a log file, `design.log`, containing messages, warnings, or errors issued by ispDesignExpert.

## Interfaces Dialog Box

Click on the **Interfaces** button in the Settings tab. The Interfaces dialog box appears (Figure 4-12). You can specify the output netlist format that is to be generated for the post-route simulation.

The possible values are EDIF, VHDL, Verilog, and LMC. No default value is set for this option.



Figure 4-12. Interfaces Dialog Box

## Advanced Compiler Settings Dialog Box

Click on the **Advanced** button in the Settings tab. The Advanced Compiler Settings dialog box displays (Figure 4-13). This dialog box is used to specify additional compiler settings.

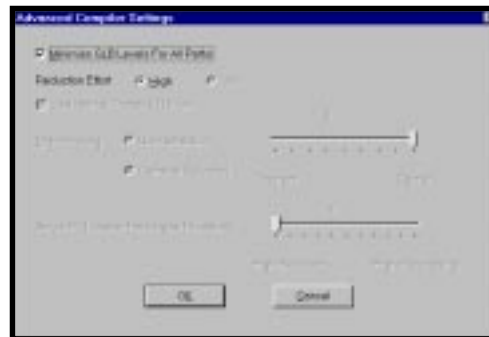


Figure 4-13. Advanced Compiler Settings Dialog Box

### Minimize GLB Levels For All Paths

The Minimize GLB Levels For All Paths advanced option instructs the compiler to reduce the GLB levels on all paths in your design. Default is High.

### Use Internal Tristate IO Driver

The Use Internal Tristate IO Driver advanced option controls whether the I/O driver from the GRP is used for the internal tristate bus. This option is only valid for ispLSI 8000 devices. Default is Off.

## BFM Packing

The BFM Packing advanced option controls the number of BFMs into which your design is placed in ispLSI 8000 devices.

## Single PT Function Packing for Routability

The Single PT Function Packing for Routability advanced option specifies how single PTs are mapped to macrocells in ispLSI 8000 devices. Default is 0.

## Device Options

The Device Options tab (Figure 4-14) is for you to define objectives for the design implementation. Options that are not applicable on that device are grayed out.

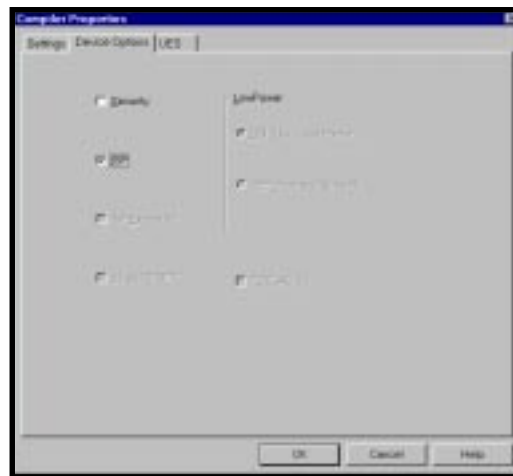


Figure 4-14. Compiler Properties Dialog Box - Device Options

## Security

The Security option influences the device security cell programming. However, this option does not guarantee that the security cell is set or cleared, because device programmer options also affect the security cell. The default is Off.

## ISP

The In-System Programming (ISP) option informs the software that you want to use the ISP pins on an ispLSI device. Default is On.

## ISP Except Y2

The ISP\_Except\_Y2 option allows the software to use the Y2 clock input for routing, which increases resource utilization. This option is device dependent. Default is Off.

**Y1 as RESET**

The Y1\_AS\_RESET option determines how the Y1/RESET pin is used. Default is On.

The Y1/RESET pin is a global reset input if Y1\_AS\_RESET is turned on. The Y1/RESET pin is the Y1 clock input if Y1\_AS\_RESET is turned off. This option is device dependent.

You cannot lock a signal to the Y1/RESET input pin. If Y1\_AS\_RESET is turned on, the Global Reset signal is automatically connected to the Y1/RESET pin, and you will get an error if you try to lock a signal to this pin.

**TOE\_AS\_IO**

The TOE\_AS\_IO option determines the use of the TOE/IO shared pin.

The TOE/IO shared pin in the ispLSI 5000V device can be defined either as a TOE (Test Output Enable) or a regular IO pin. The default setting of Off indicates the pin will be a TEST OE pin.

You cannot lock to pin IO0 unless you set TOE\_AS\_IO to On.

**LowPower**

The LowPower option controls whether a lower power mode (ON) or a faster speed mode (OFF) is used for the ispLSI 5000V and 8000 devices.

The LowPower Device Options allow you to globally select a lower power mode or a faster (higher power) speed mode. Default for the ispLSI 8000 device family is ON. Default for the ispLSI 5000V device family is OFF.

**UES**

The UES tab (Figure 4-15) lets you choose the data type and specify the User Electronic Signature.

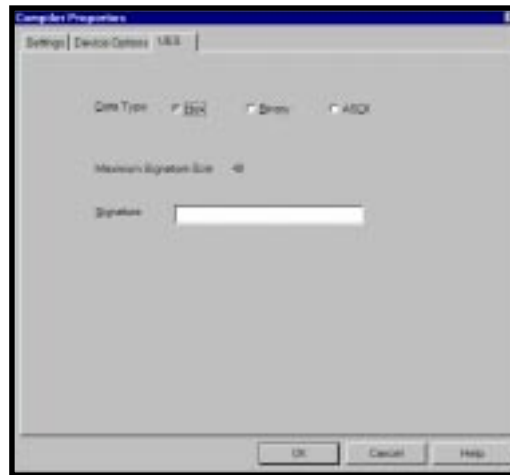


Figure 4-15. Compiler Properties Dialog Box - UES

### Data Type

Check one of the radio buttons to indicate that your UES should be binary, hexadecimal, or ASCII format.

### Signature Size

The Maximum Signature Size field tells how many characters can be used for signature; this varies depending on the format of signature and the device type. Once you enter the maximum number of characters, no additional characters are accepted in this field.

## MACH Global Optimization Options

If you have a MACH design, you can set the optimization options for the Fitter in the Global Optimization Options dialog box.

With a MACH project opened in the Project Navigator, choose **Tools** ⇒ **Global Project Optimization** to invoke the Global Optimization dialog box.

There are three tabs in this dialog box.

### Global Optimization

The Global Optimization tab (Figure 4-16) offers you optimization options. Use the **Defaults** button to reset the selections to their default settings. Use the **Apply** button to apply the options.

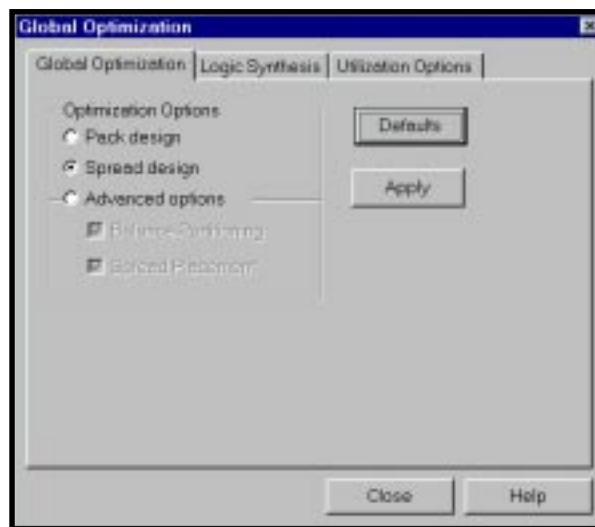


Figure 4-16. Global Optimization Dialog Box - Global Optimization

### Pack Design

If this radio button is checked, the Fitter packs the design pack by pack. This option packs the design for maximum density and speed. It sets both the Balance Partitioning and Spread Placement features to FALSE.

### Spread Design

If this radio button is checked, the Fitter spreads the design throughout the device. This option allows the design to be partitioned and placed in a spread out method. This method of placement enhances the upgrade capability of the design, by spreading the resources throughout the device. However, the disadvantage is that more resources may be used because the design is partitioned evenly through all the blocks. This option sets Balance Partitioning and Spread Placement to TRUE.



## Advanced Options

This option lets you choose the Balance Partitioning and Spread Placement algorithms.

- **Balance Partitioning:** When this option is enabled, the design is partitioned evenly among all the blocks in the device. So each block should have about the same amount of resources used. When this option is disabled, the design is partitioned block by block, filling up one block at a time. Some blocks may be filled up completely, while others may be unused.
- **Spread Placement:** When this option is enabled, the design signals are placed evenly or spread-out among macrocells in the block. Spreading out the placement allows the user to make minor changes to the existing output and node signals in the block. When this option is disabled, the design signals are assigned to the first available macrocell in the block. This placement method will leave all unused macrocells at the end of the block, making it easier to add new outputs or nodes to this block.

## Logic Synthesis

The Logic Synthesis tab (Figure 4-17) lets you select the synthesis or optimization options. Use the **Defaults** button to reset the selections to their default settings. Use the **Apply** button to apply the options.



Figure 4-17. Global Optimization Dialog Box - Logic Synthesis

## Boolean Logic Synthesis

This option enables or disables boolean logic reduction. Its default state is Enabled.

## D/T Synthesis

This option enables or disables D and T-type synthesis. When enabled, the Venus optimizer synthesizes the registered signals to use the D or T-type flip-flop with the least number of product term implementation. When disabled, the register type specified is left untouched. Its default state is Enabled.

## Set/Reset Don't Care

This option allows the signals that have defined Set and Reset functions to fit into the same block as signals which don't. If this option is disabled, signals without set or reset function will not be partitioned into the same block as signals that contain set or reset functions, and hence may inhibit design fitting. Its default state is Enabled.

## Node Collapsing

This option enables or disables node collapsing. If the user has specified specific nets or nodes to "Keep", these nets or nodes will not be collapsed regardless of whether Node Collapsing option is enabled. Its default state is Enabled.

- Collapse All Nodes: Collapses all nodes up to the set Product Term limit without regard for the logic.
- Collapse Selective Nodes: Collapses all nodes up to the set Product Term limit, but considers common product terms and groups them into nodes.

## Product Term Collapsing

This option sets the product term limit for the Node Collapsing algorithm. Normally, this value should equal the product term limit for Equation Splitting. The default value is its Family default.

## Product Term Equation Splitting

This option sets the product term limit for the equation splitting algorithm. Normally, this value should equal the product term limit for Node Collapsing. The default value is its Family default.

## Utilization Options

The Utilization Options tab (Figure 4-18) lets you specify the utilization limits for macrocells, block inputs, segment inputs, and inter-segment lines. These limits are "Soft Constraint", which means that the Fitter will ignore these constraints if they inhibit design fitting. Warnings will be specified if the constraints are ignored. Use the **Default** button to reset the selections to their default settings. Use the **Apply** button to apply the options.

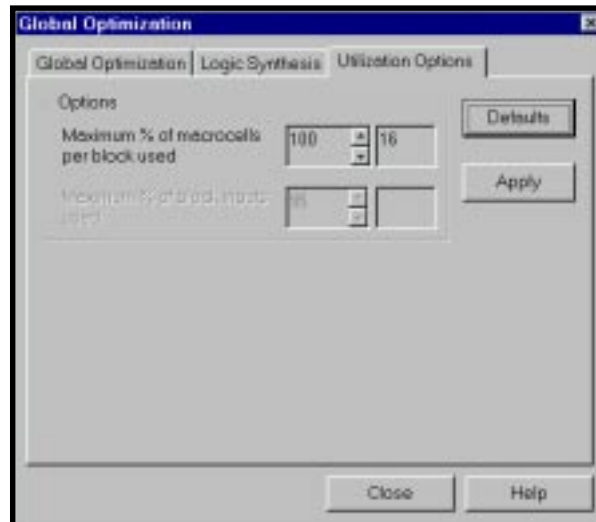


Figure 4-18. Global Optimization Dialog Box - Utilization Options

**Maximum % of Macrocells per Block Used**

This option sets the limit for the number macrocells to be used in a block.

**Maximum % of Block Inputs Used**

This option sets the limit for the number of inputs to be used going into a block.

## Compiling/Fitting the ispLSI/GAL Design

Once you completed an ispLSI or GAL design, you may want to fit the design into a target device. Since you selected a device earlier in your design, the remaining steps are straightforward.

1. Select the device in the Sources in Project field of the Project Navigator window and observe the related processes.
2. If you have chosen an ispLSI device, the ispDesignExpert software has several user controls that can be accessed from the Project Navigator; highlight Compile Design and click the **Properties** button at the bottom of the Navigator window. The Compiler Properties dialog box (Figure 4-9, Figure 4-10, Figure 4-11) with compiler settings, device options, and UES in it appears. See the [ispEXPERT Compiler User Manual](#) or online help for explanations.

If you have chosen a GAL device, you can also access some design properties; highlight the Fit Design process and click the **Properties** button to open the Fit Design Properties dialog box (Figure 4-19) with design settings. Refer to the online help for more information on these properties.

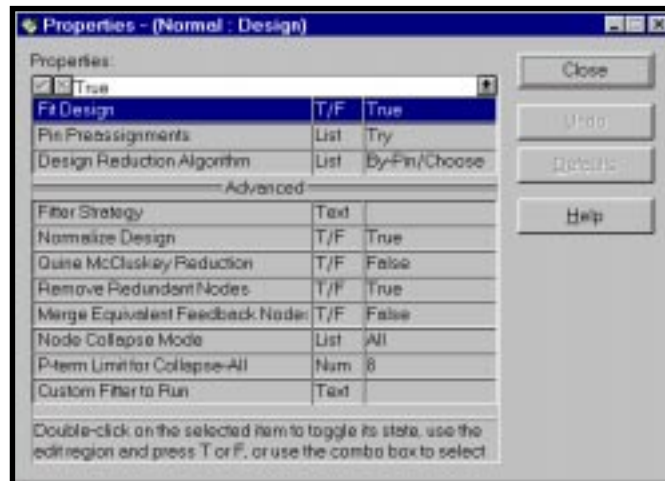


Figure 4-19. Fit Design Properties Dialog Box for GAL Designs

3. Double click the Compile/Fit Design process or click the **Start** button. The ispDesignExpert Process dialog box appears.

The ispDesignExpert finishes compiling the source, then links the source files together. Finally, the software partitions and fits the design into the target device. Note that check marks have been added to Compile/Fit Design process that have been successfully completed (Figure 4-20).

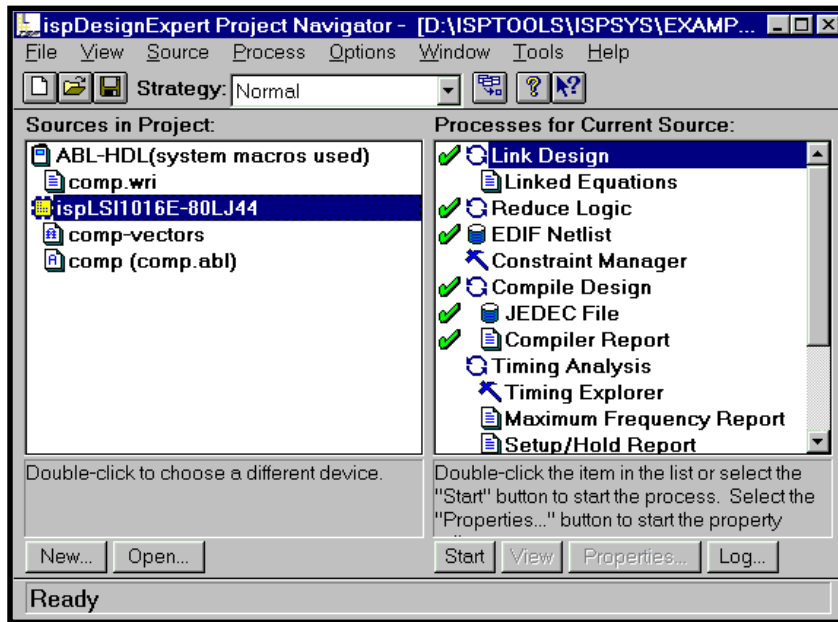



Figure 4-20. ispDesignExpert Project Navigator after the Compile/ Fit Design Process of ispLSI/GAL Designs

 **NOTE** Yellow exclamation besides the process points indicate that warnings were generated. Red Xs indicate that errors were encountered. The warning or error is described in the auto-make log file displayed in the Report Viewer. Green check marks indicate the process completed successfully.

4. Double click on Compiler Report in the Processes window to see the statistics related to the fitting of your design (Figure 4-21).

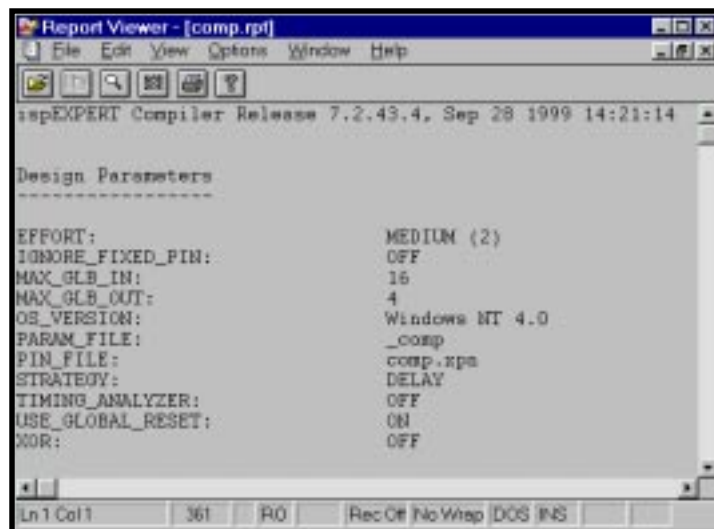


Figure 4-21. Compiler Report

## Compiling/Fitting the MACH/PAL Designs

If you have completed your MACH/PAL design, you may want to fit the design into a target device. Since you selected a device earlier in your design, you can do the following.

1. Select the device in the Sources in Project field of the Project Navigator window and observe the related processes.
2. If you have a MACH device, no properties can be set for the Fit Design process. If you have chosen a PAL device, the ispDesignExpert software has several user controls that can be accessed from the Navigator; highlight Fit Design and click the **Properties** button at the bottom of the Navigator window. The Properties dialog box (Figure 4-22) with design settings in it appears. See the online help for explanations.

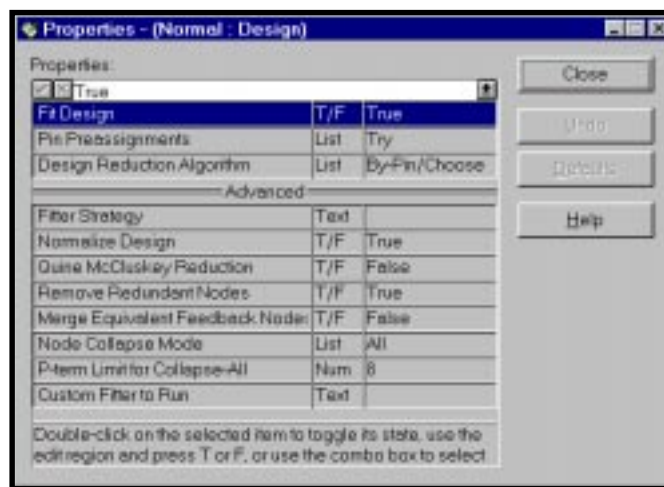


Figure 4-22. Fit Design Properties Dialog Box for PAL Designs

3. Double click the Fit Design process or click the **Start** button. The ispDesignExpert Process dialog box appears.

The ispDesignExpert finishes compiling the source, then links the source files together. Finally, the software partitions and fits the design into the target device. Note that check marks have been added to Compile/Fit Design process that have been successfully completed (Figure 4-23).

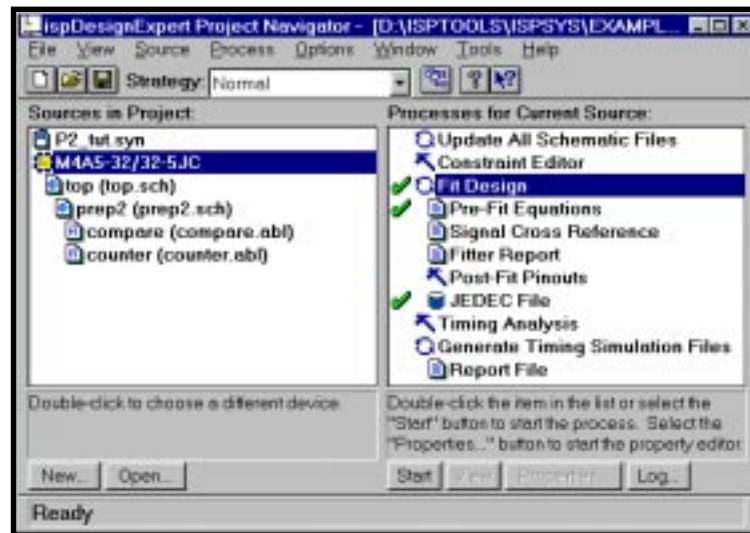



Figure 4-23. ispDesignExpert Project Navigator after the Fit Design Process of MACH/PAL Designs

 **NOTE** Yellow exclamation besides the process points indicate that warnings were generated. Red Xs indicate that errors were encountered. The warning or error is described in the auto-make log file displayed in the Report Viewer. Green check marks indicate the process completed successfully.

4. Double click on the Fitter Report process from the Processes window to see the statistics related to the fitting of your design (Figure 4-24).



Figure 4-24. Fitter Report

## Chapter 5 *Design Verification*

---

The ispDesignExpert provides you a full simulation environment. The Lattice Logic Simulator is a flexible tool that helps you verify the operation and logic functionality of your ispLSI or GAL design. The Equation Simulator enables you to verify a MACH or PAL design. The ispDesignExpert also has an integration interface with ModelSim, providing you easy access to VHDL and Verilog HDL simulation.

This chapter covers the following information:

- Lattice Logic Simulator for ispLSI/GAL
- Equation Simulator for MACH/PAL
- JEDEC Simulation for PAL/GAL
- ModelSim Simulation



# Lattice Logic Simulator for ispLSI/GAL

## Overview

Lattice Logic Simulator is the Lattice proprietary gate-level simulation tool so that you can run both Functional Simulation and Timing Simulation. Functional Simulation, pre-route design verification, occurs before the design has been fitted and routed. It helps you find logical or coding error in early design cycle. Timing simulation, post-route simulation, confirms that your design is compatible with the timing and propagation delays that exist in a specific device.

## Creating Test Stimulus for Lattice Logic Simulator

Before simulation, you must create a stimulus file that specifies the input waveforms. There are two ways to create a stimulus file:

- Create a graphic waveform file

The Waveform Editing Tool (WET) lets you graphically create input stimulus waveforms for your design by drawing them directly on the screen. The stimuli can be edited graphically or by modifying values in dialog boxes. The WET then converts the waveforms into a stimulus file that the simulator recognizes. Waveform files (in Waveform Description Language format) are also useful as input to automatic test equipment or as documentation of the circuit's expected behavior.

If you associate the waveform stimulus file (.wdl) with the selected device in your design, both the functional and timing simulation processes are supported. However, if you associate the waveform stimulus file with a design module, only functional simulation is available.

- Create a test vector file

You can create a test vector file in a text editor using proper keywords. Test vectors are sets of input stimulus values and corresponding expected outputs that can be used with both functional and timing simulators. Test vectors can be specified either in a top-level ABEL-HDL source or in a separate ABEL-HDL test vector format file called a .abv file. The .abv file is considered a text document and is kept above the device level in the Sources window. Whether the test vectors are part of a top-level ABEL-HDL source or are in a separate file, they will be compiled and passed to the simulator.

For more details, refer to the [Design Verification Tools User Manual](#).

## Running Functional/Timing Simulation

You can simulate an ispLSI or a GAL design using the Functional/Timing Simulation process in the Project Navigator. This process launches the Lattice Logic Simulator to verify your design operation.

To run Functional/Timing Simulation:

1. Select the ABEL-HDL test vector file (.abv) or the waveform stimulus file (.wdl) of an ispLSI or a GAL design from the Sources window of the ispDesignExpert Project Navigator (Figure 5-1).

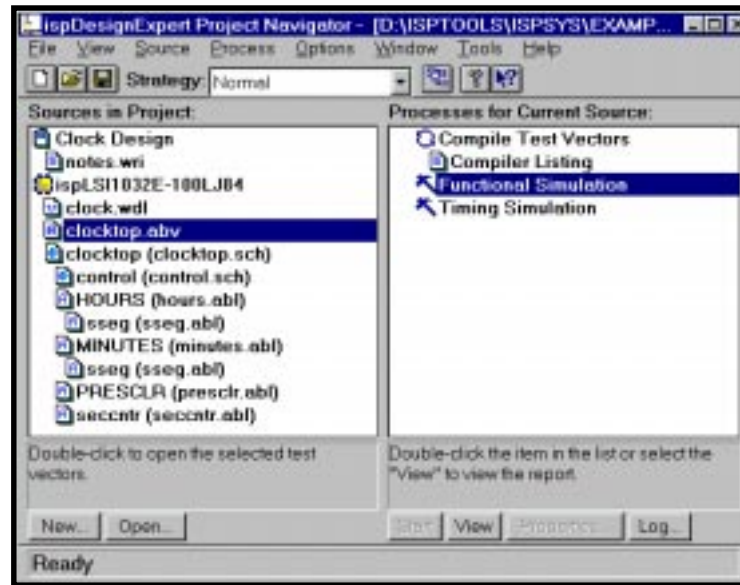


Figure 5-1. Running the Lattice Logic Simulator

2. Select the Functional/Timing Simulation process to launch the Simulator.



**NOTE**

If the test stimulus file is associated with the selected device, both Functional and Timing Simulation processes are supported. However, if you associate the test stimulus file with a design module/schematic, only Functional Simulation is available.

3. The Simulator Control Panel window appears (Figure 5-2).

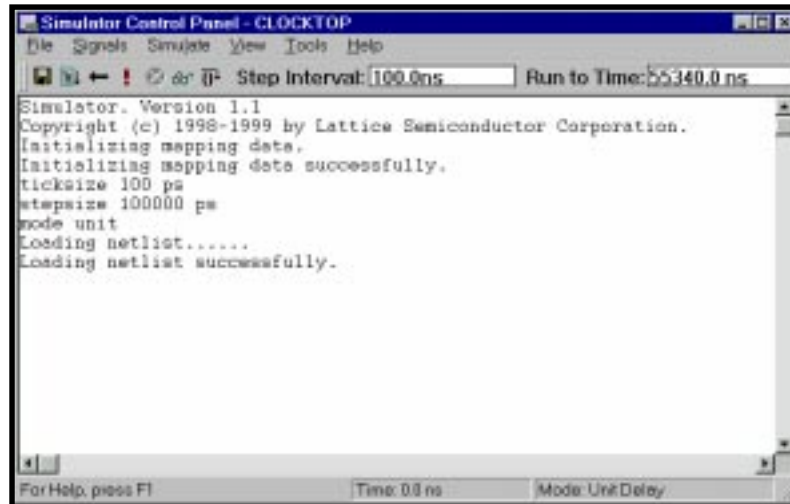


Figure 5-2. Simulator Control Panel

4. Click the **Run** icon from the toolbar or select the **Simulate** ⇒ **Run** menu item to start the simulation.

**NOTE**

If you keep the **View** ⇒ **Show Waveforms** menu item checked in the Simulator Control Panel, the Waveform Viewer is invoked in the course of simulation to show the simulation results in waveforms.

### Showing the Waveforms in the Waveform Viewer

5. When the simulation ends, the waveforms of the signals display in the Waveform Viewer. You can click the **Waveform Viewer** icon from the toolbar or select **Tools** ⇒ **Waveform Viewer** from the Simulator Control Panel window to open the Waveform Viewer if you did not check the **View** ⇒ **Show Waveforms** menu item before running the simulation. Select **Edit** ⇒ **Show** in the Waveform Viewer window; the Show Waveforms dialog box appears (Figure 5-3).

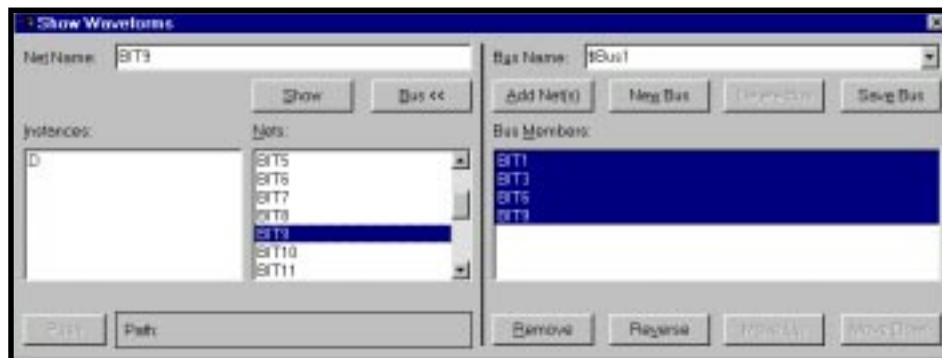


Figure 5-3. Show Waveforms Dialog Box

6. Select the signals in the Nets field, then click the **Show** button to add the selected waveforms in the display area of the Waveform Window. You can also select desired signals to form buses and add them in the view. Refer to the [Design Verification Tools User Manual](#) for more information on how to create multi-signal “buses.”

## HDL Cross Probing for ispLSI/GAL Designs

For an ispLSI/GAL design, you can use the cross probing function between the Waveform Viewer/Editor and the Hierarchy Navigator/HDL viewer to probe an item after invoking the Hierarchy Browser.

*To probe an item in an HDL source or a schematic:*

1. Select the top-level source from the Sources window of the Project Navigator.
2. Double click on the **Hierarchy Browser** process in the Processes window. The Hierarchy Browser will be opened together with the Hierarchy Navigator or the HDL Viewer.
3. Run the Functional Simulation or the Timing Simulation process associated with the top-level source. Select **Tools** ⇒ **Probe Item** then click on a net in the Hierarchy Navigator if your top-level source is a schematic, or just click on a port signal name in the HDL Viewer if your top-level source is an HDL module, the corresponding waveform will be shown in the Waveform Viewer.



### **NOTE**

Refer to the [Design Verification Tools User Manual](#) for detailed information on other Waveform Viewer commands and functions.

## Simulation Log

The Simulator logs errors and status information in one of the three files, depending on the nature of the information:

<code>automake.log</code>	logs processing information
<code>*.err</code>	Logs logic errors
<code>*.slg</code>	Logs simulation status information

If a simulation error occurs, it is recorded in, and may be viewed by selecting the **View** ⇒ **Simulator Log** menu item in the Simulator Control Panel window. Simulation errors do not automatically cause the Report Viewer to appear.

## Running Stand-alone Lattice Logic Simulator

The Lattice Logic Simulator can be invoked in its stand-alone mode by selecting the **Tools** ⇒ **Lattice Logic Simulator** menu item from the ispDesignExpert Project Navigator. The stand-alone simulator enables you to simulate the design file or stimulus file outside the current project. Even if you have previously opened the Simulator for a project, you can change to the stand-alone mode.

To run the stand-alone simulator:

1. Select **Tools** ⇒ **Lattice Logic Simulator** from the Project Navigator. The Simulator Control Panel window appears (Figure 5-4) in its stand-alone mode.

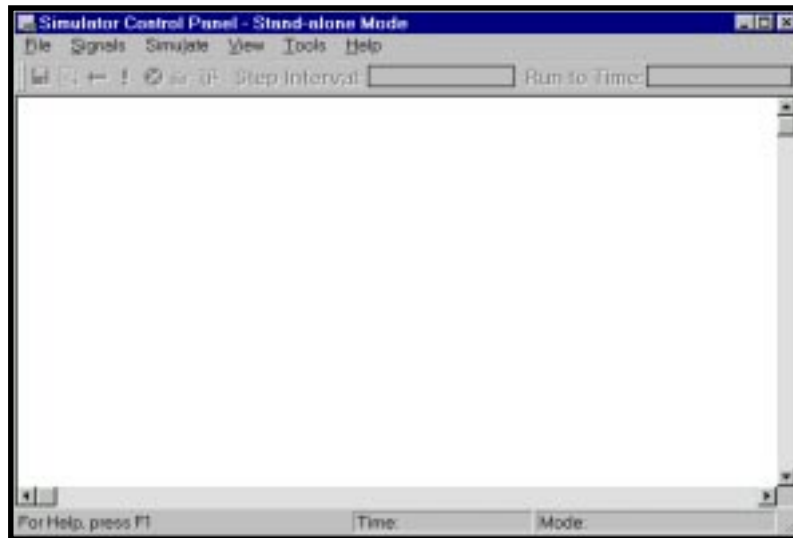


Figure 5-4. Simulator Control Panel - Stand-alone Mode

2. Select **File** ⇒ **Open Design** from the Simulator Control Panel window. The Open Design dialog box displays (Figure 5-5).

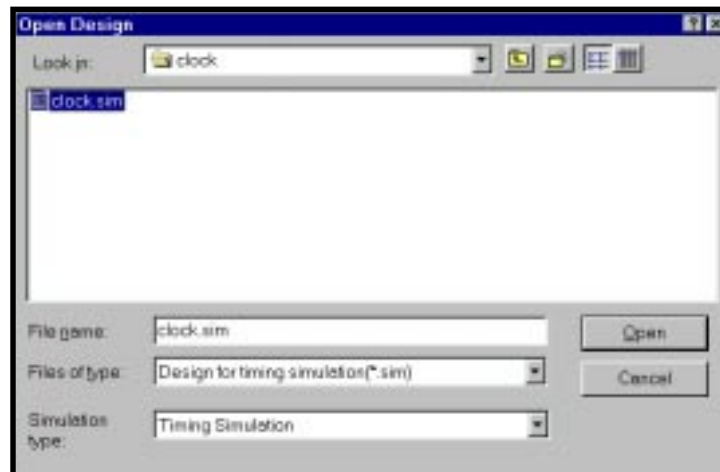


Figure 5-5. Open Design Dialog Box

- In the Files of type field, select the type of design you need to open. Either a design for functional simulation (\*.edn or \*.edf) or a design for timing simulation (\*.sim). The Simulation type field enables you to choose either Functional Simulation or Timing Simulation. Choose a desired \*.edn, \*.edf, or \*.sim file from the project directory.

**NOTE**

A design with other file extensions can be loaded in the simulator. But only design with correct format can be used to perform simulation successfully.

For example, if you choose *design.s1* with .sim format to do timing simulation, a warning message will be issued:

The selected file extension does not match the simulation type. Make sure that the .edf(.edn) file is used for functional simulation and .sim file is used for timing simulation. Press OK to continue.

Click **OK** to continue. You can load the file in the simulator successfully. And you should be able to run timing simulation successfully after loading a stimulus file.

If you choose to load *design.s2* design that is not .sim format, you will get an error message when running the simulation. You cannot run simulation successfully though you have loaded the design.

You will get a message stating Loading design file “\*.sim” (\*.edf or \*.edn) successfully in the Simulator Control Panel window. But the toolbar is still gray at this moment.

- Choose **File** ⇒ **Open Stimulus**. The Open Stimulus dialog box appears prompting you to select a stimulus file (\*.wdl, \*.abv, or \*.abl) (Figure 5-6).

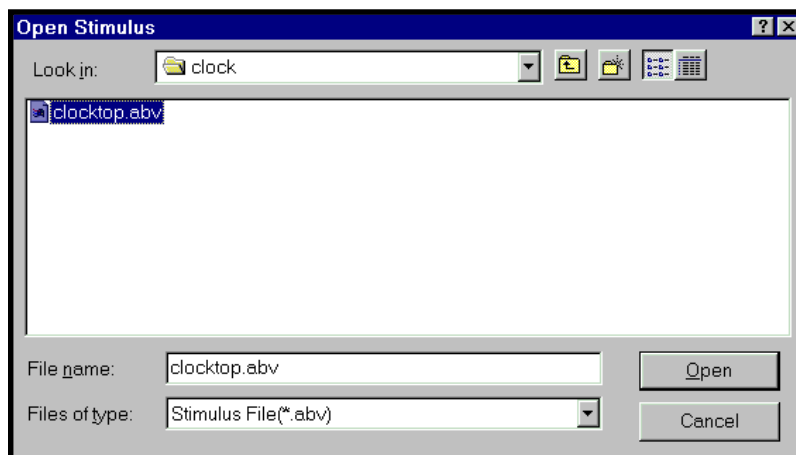


Figure 5-6. Open Stimulus Dialog Box

**NOTE**

The stimulus file you choose must be in the same directory as the netlist file.

A message “Loading netlist successfully” appears in the Simulator Control Panel window. The toolbar is activated at this moment.

5. Click the **Run** icon from the toolbar or select **Simulate** ⇒ **Run** to start the simulation.

**NOTE**

If you keep the **View** ⇒ **Show Waveforms** menu item checked, the Waveform Viewer is invoked in the course of simulation to show the simulation results in waveforms.

Refer to [“Showing the Waveforms in the Waveform Viewer” on 139](#) for details on viewing the waveforms.

**NOTE**

Refer to the [Design Verification Tools User Manual](#) for detailed information on other Waveform Viewer commands and functions.

# Equation Simulator for MACH/PAL

## Overview

Equation Simulator is a design verification tool enabling you to verify your ABEL-HDL or schematic designs with a MACH or PAL device. Equation simulation is similar to Functional simulation, uses design test vectors, that you supply, to simulate the design logic or equations independent of any device. The more comprehensive and detailed your test vectors are, the more useful your simulation results will be.

Equation simulation can be conducted before you select a device. It only tests the equations in your design as specified by test stimulus (ABEL-HDL test vectors).

## Creating Test Vectors for Equation Simulator

You need to create test stimulus with ABEL test vectors before you do equation simulation for a MACH or PAL design. You can specify the test vectors in the way described on [page 137](#).

When using test vectors, you specify the required input pattern and the expected outputs at the device pins and buried nodes. The simulator applies inputs from the test vectors to the simulated circuit and compares the simulated output with the output specified in the test vectors. If there is any difference, an error is indicated.

## Running the Equation Simulation

Equation Simulation, similar to Functional Simulation, tests your design without using device-specific information. Equation Simulation can be conducted before you select a device. However, it only tests the equations in your MACH or PAL design as specified by test stimulus.

To make the simulation process available, you have to select the test vector file in the Sources window. The simulation process appears in the Processes window.

*To invoke the Equation Simulation process:*

1. Select the ABEL-HDL test vector file (.abv) of a MACH or PAL design from the Sources in Project window in the ispDesignExpert Project Navigator (Figure 5-7).



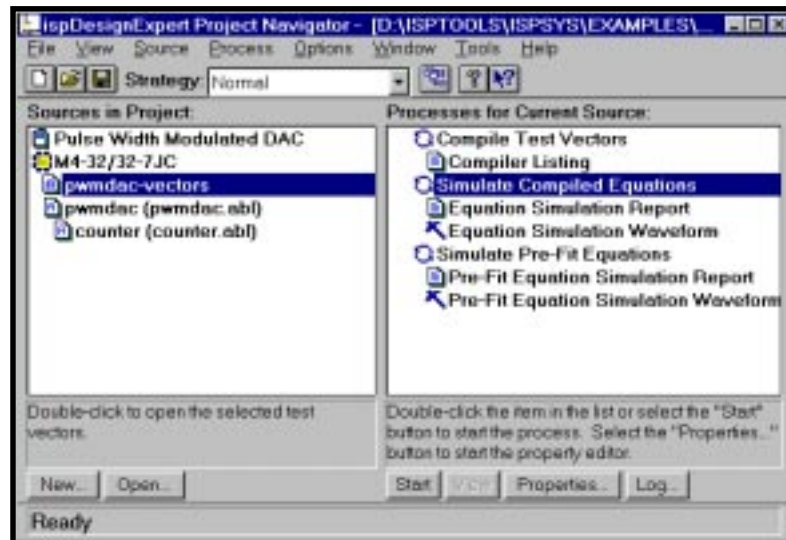


Figure 5-7. Running the Equation Simulator

2. Select the Simulate Compiled Equations process to start the simulation.

## The Simulator Model

The Equation Simulator uses the Equation file to build a model of the design. This method includes macrocells, sum-terms, and product terms. Select the Report Type Macro-cell property to display the model.

*To view the simulation model:*

1. In the Sources window, select the test vector (.abv) file.
2. In the Processes window, select the associated Simulate Pre-Fit Equations process. Click the Properties button at the bottom of the Project Navigator to open the Properties dialog box.
3. In the Properties dialog box (Figure 5-8), select Report Type and set the List property to Macro-cell. Then close the dialog box.

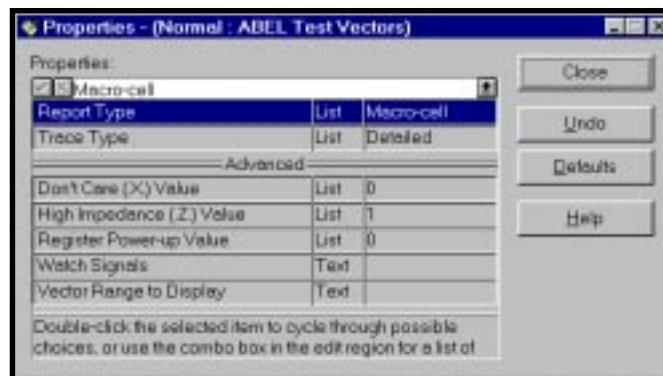


Figure 5-8. Properties Dialog Box - Report Type: Macro-cell

When you select the Macro-cell List property, the simulation results will be displayed for all dot extensions associated with I/O macrocells. Note that this display option is detailed, and should be used in conjunction with the “Signal” option to reduce the size of the output report.

Macro-cell reports provide the same information as Tabular, with the addition of internal device information such as OR-gate outputs, and the final outputs.

The Macro-cell report option produces large files if all pins and nodes are traced for all vectors. To generate a smaller file, use the Watch Signals and Vector Range to Display properties (to specify the pins or nodes for a limited number of vectors). If you do not specify which signals to watch, the first I/O pin in the device is traced.

## Controlling the Simulation Report

Report and trace types and break points allow you to control the amount of information the simulation provides. Simulation can provide simple error messages (indicating that the actual outputs differ from the outputs you predicted in your test vectors), or detailed information about the states of internal registers and product terms of a device during simulation. If you simulate a design with Report Type set to None (Figure 5-9), you can determine if any errors exist. If there are errors, you can use the more detailed Report and Trace types to increase the amount of information provided until you have enough information to solve the problem.

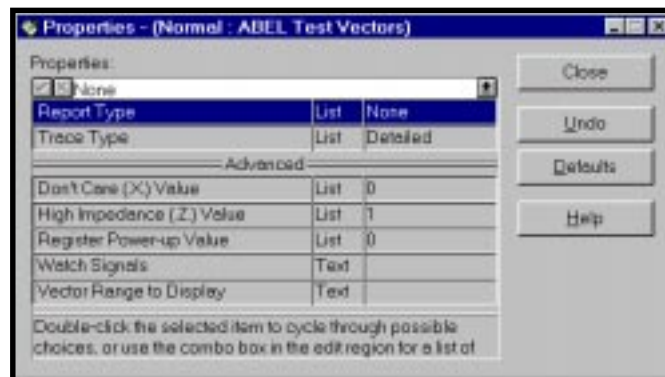


Figure 5-9. Properties Dialog Box - Report Type: None



### **NOTE**

In order to see the clock waveform in the Waveform Viewer, you need to use Trace Type: Detailed, which is the default. Otherwise, you will not see the clock edges displayed.

For more details on controlling the simulation report, refer to the [Design Verification Tools User Manual](#).

## Displaying the Waveforms in the Waveform Viewer

The Waveform Viewer displays the results of logic simulation. The nets whose waveforms are to be displayed can be interactively chosen from the schematic. Query functions can be used to trace signals to their source on the schematic. Trigger functions can be used to locate the occurrence of a specific logic event. Delays between events can be measured with markers.

The Waveform Viewer is typically used in conjunction with a simulator. You must run simulator before you can run the Waveform Viewer; without simulation information, the Waveform Viewer has no data to display. Therefore, you “open” the Waveform Viewer by running the simulation.

*To open the Waveform Viewer:*

1. In the Sources window, select the .abv file. The associated processes appear in the Processes window of the Project Navigator.
2. In the Processes window, double click the last process to run the simulation. After the simulation runs, the Waveform Viewer opens (Figure 5-10) displaying the simulation waveforms.

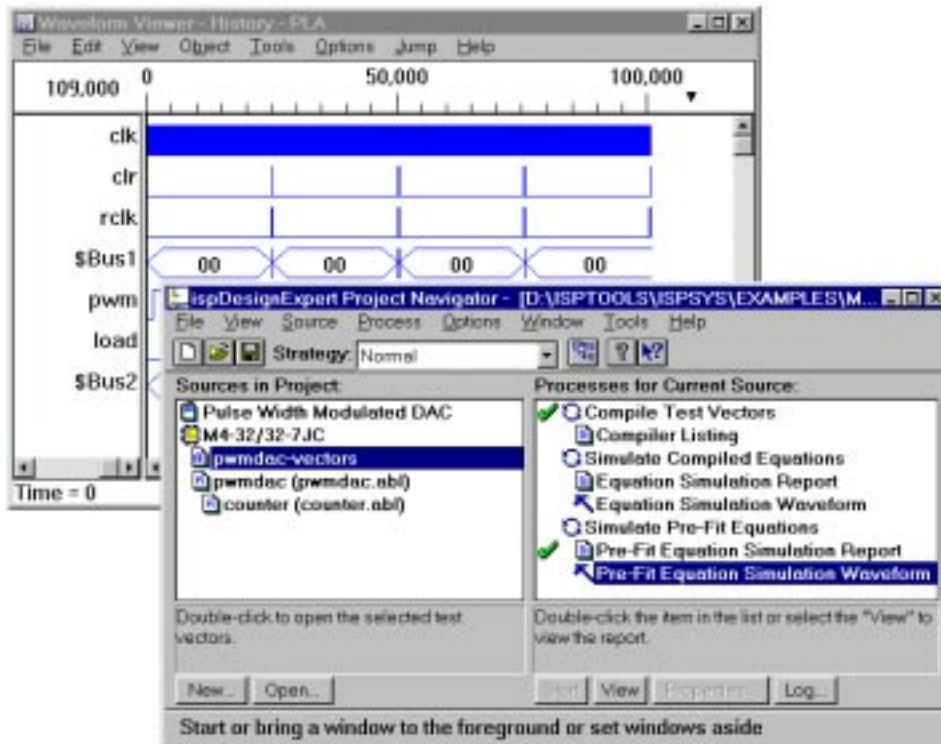


Figure 5-10. Opening the Waveform Viewer



### **NOTE**

Refer to the [Design Verification Tools User Manual](#) for detailed information on other Waveform Viewer commands and functions.

## JEDEC Simulation

JEDEC simulation simulates your GAL or PAL design JEDEC file as a final indication that the information to be programmed into your GAL or PAL is correct.

### Simulating a JEDEC File

For GAL or PAL devices, you can simulate the JEDEC fuse file by using the ABEL-HDL test vector process Simulate JEDEC File. This process simulates the JEDEC fuse map as compared to an internal model of the target device itself.

To simulate the JEDEC file:

1. Select the ABEL-HDL test vector `.abv` in the Sources window of the Project Navigator.
2. Double click the **Simulate JEDEC File** process from the Processes window or click the **Start** button to run the process (Figure 5-11).

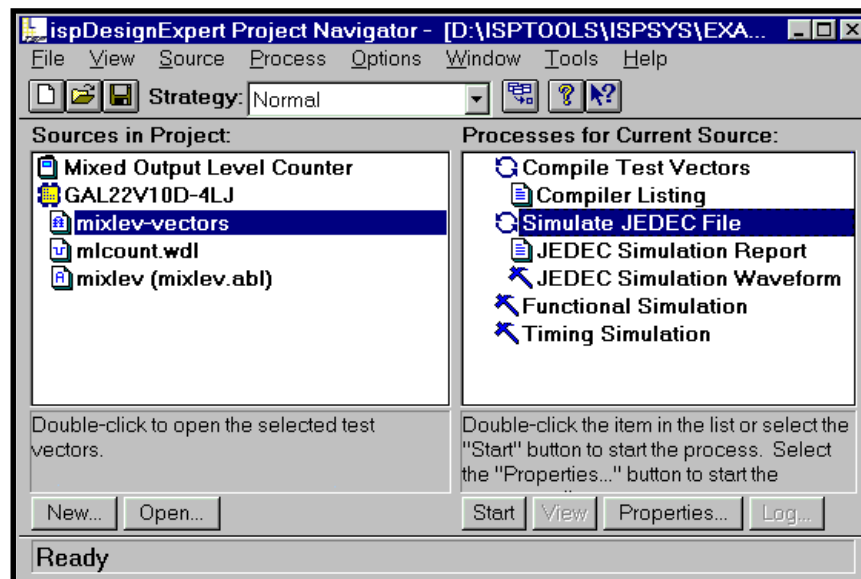


Figure 5-11. Running JEDEC Simulation

### Viewing the Simulation Waveform

Like all the other simulation, you can also view the simulation result via the Waveform Viewer.



#### NOTE

Refer to the [Design Verification Tools User Manual](#) for detailed information on other Waveform Viewer commands and functions.

## ModelSim Simulator

The ispDesignExpert software has an interface integration with ModelTech ModelSim Simulation Tool providing you a VHDL and Verilog Simulator (Figure 5-12).

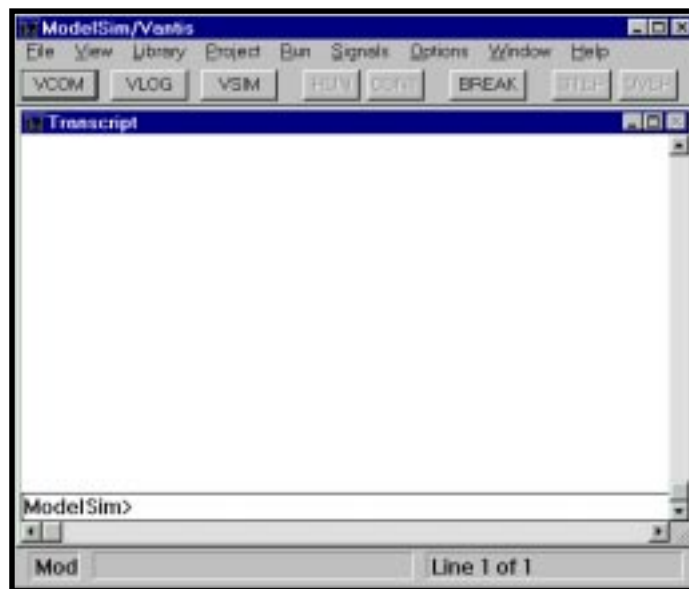


Figure 5-12. ModelSim Simulator

### Generating Test Stimulus

You can specify the test stimulus before performing the VHDL/Verilog simulation in these ways:

- manually create VHDL test bench or Verilog test fixture
- create VHDL test bench or Verilog test fixture using the template
- export a VHDL test bench or Verilog test fixture from the Waveform Editor if you have a source .wdl

Following are the rules you need to comply with when creating a VHDL test bench file:

- the entity name of the test bench is the same as the file name of the test bench;
- the instance name of the top design module is `UUT`.

While creating a Verilog test fixture file, you need to following these rules:

- the name of the test module must be the same as the file name of the test fixture;
- the instance name of the top design module is `d`.

## Manually Create VHDL Test Bench or Verilog Test Fixture

You can manually create a VHDL test bench or a Verilog test fixture in the Text Editor. Refer to other third-party manuals for VHDL test bench and Verilog test fixture syntax rules.

## Create VHDL Test Bench or Verilog Test Fixture Using the Template

Once you have an ABEL HDL, a schematic, a VHDL, or a Verilog HDL source in your ispLSI, GAL, MACH, or PAL project, you can use the VHDL Test Bench Template process associated with the selected source in the Project Navigator to generate a template file (\* .vht) for the test bench. In order to use the test bench, you must edit or rename it with the extension .vhd. Then you can add relevant contents to the .vhd file and save it as the VHDL test bench for your project.

**▲ CAUTION** The VHDL Test Bench Template (.vht) is an intermediate file that can be deleted when using the File ⇒ Clean Up All command of the Project Navigator. Do not save your modifications to the .vht file.

And, you can use the Verilog Test Fixture Declarations process associated with the selected source in the Project Navigator to generate a Verilog test fixture declarations file (\* .tfi) that should be included in the test fixture. By using the Text Fixture Declarations include file in your simulation test fixtures, you ensure your design and test fixture stay synchronized. Add relevant contents to the .tfi file and then you have a Verilog test fixture for your project.

**▲ CAUTION** The Verilog HDL Test Fixture Declarations (.tfi) is an intermediate file that can be deleted when using the File ⇒ Clean Up All command of the Project Navigator. Do not save your modifications to the .tfi file.

## Export VHDL Test Bench or Verilog Test Fixture from Waveform Editor

If you have a source `.wdl` file in your project, you can export the VHDL test bench or the Verilog test fixture from the Waveform Editor.

To export VHDL test bench or Verilog test fixture from Waveform Editor (WET):

1. Select the source `.wdl` file from the Sources window of the Project Navigator and open it in the Waveform Editor.



### NOTE

If you import a `.wdl` file to your project for the first time and want to view the waveform of the `.wdl` file in the Waveform Editor, select **Edit** ⇒ **Show** from WET. In the prompt Show Patterns dialog box, choose the desired signals and click the **Show** button. The waveform of the desired signals are thus shown in WET.

2. Choose **File** ⇒ **Export** from the Waveform Editor. The WET Export dialog box appears (Figure 5-13).



Figure 5-13. WET Export Dialog Box

- Input File – Enables you to choose an input `.wdl` file.
- Output File – Enables you to have an output file default name of which is the same as the input file chosen. The file extension varies with the Export Style you choose.
- Export Style – Specifies the style of the file you need to export, either a Verilog test fixture (`.tf`) or a VHDL test bench (`.vht` or `.tb`).



### NOTE

The exported `.vht` or `.tb` file needs to be renamed to `.vhd` before it is to be used as a VHDL test bench in your project.

## Performing the VHDL/Verilog Functional/Timing Simulation

You can simulate a VHDL or Verilog HDL design using the VHDL/Verilog Functional/Timing Simulation process in the Project Navigator.

*To run VHDL/Verilog Functional/Timing Simulation:*

1. Choose the VHDL test bench (.vhd) or the Verilog test fixture (.tf) from the Sources window of the Project Navigator.

**NOTE**

If the test stimulus file (.vhd or .tf) is associated with the selected device, both VHDL/Verilog Functional and Timing Simulation processes are supported. However, if you associate the test stimulus file (.vhd or .tf) with a design module/schematic, only VHDL/Verilog Functional Simulation is available.

2. Select the VHDL/Verilog Functional/Timing process from the Processes window. The ModelSim Simulator prompts with simulation messages showing in the Transcript window. When the simulation is done, you will see #-End in the Transcript window.



## Chapter 6 *Timing Analysis*

---

The ispDesignExpert has two built-in static Timing Analyzers that are device dependent. For ispLSI devices, the Timing Analyzer can be accessed to provide accurate pin-to-pin timing formation for your design. For MACH and PAL devices, you can use the Performance Analyst to analyze the permanence of your design, after it had been optimized and implemented by the Fitter. The Timing Analyzer calculates maximum clock frequency, calculates chip boundary setup and hold requirements, calculates Tpd and Tco path delays, calculates GLB boundary delays, and performs path enumeration. Generally speaking, Timing Analysis can be performed after you compile or fit a design.

This chapter covers information on the following topics:

- Timing Analysis Overview
- Timing Analyzer for ispLSI
- Performance Analyst for MACH

## Timing Analysis Overview

The static timing analyzer enables you to evaluate the performance of the design after successful compilation. The analyzer traces all the signal paths and their delays, determines critical (timing) paths, and evaluates maximum frequency of the design and setup/hold requirements.

## Timing Analyzer for ispLSI

The Timing Analyzer enables you to evaluate the performance of the ispLSI or GAL designs after successful compilation. The Timing Analyzer performs the following functions:

- Determines maximum frequency for clocking a design containing two or more flip-flops and/or latches. It also lists the clock periods between all the internal register pairs and the frequencies, along with the names of the signals that drive the clock inputs of those registers. The frequency is provided only for those sets of registers that are driven by the same reference clock; otherwise the field is left blank. The clock signal could be a primary input, register Q output, or a module I/O. It also lists the number of GLB levels for each path.
- Calculates setup and hold time for boundary registers.
- Calculates Tpd and Tco path delays.
- Calculates GLB boundary delays.
- Performs path enumeration by calculating path delays from all the source nodes to all the primary output nodes. The delays listed are in descending order and the source nodes are primary inputs, register Q outputs, or module I/Os. To obtain path delays for the remaining destination nodes that include register inputs and module I/Os, use the ispEXPERT Compiler Design Manager menus.

Once you compile the ispLSI or GAL design and run Timing Analysis either automatically or manually, the Timing Analyzer can be run. The Timing Analyzer uses the `.sim` file generated by the ispDesignExpert software as input. The timing information for the device part in the `.sim` file must exist to perform timing analysis.

## Timing Explorer

The Timing Explorer provides an interactive method for viewing and querying timing information for the design. The Timing Explorer is accessed in the Processes window of the Project Navigator. Information is calculated in response to your query, and the turn-around time is fast. Only the data you requested displays. This section describes methods to use to obtain the timing information you need.

To start the Timing Explorer:

1. With the device selected, double-click the Timing Explorer process from the Project Navigator.
2. The Timing Explorer is invoked with the timing reports generated. If you have not run the Compiler, ispDesignExpert will automatically fit the device first.

The Timing Explorer consists of the Signal Navigator and several tables. Refer to the remainder of this section for details on adding data to your Timing Explorer tables.

## Signal Navigator

The Signal Navigator (Figure 6-1) lists design signals in a tree format and groups them into four categories—Inputs, Outputs, Bidirectional, and Registers. You can traverse the design in fan-in or fan-out mode (click the right mouse button to select the mode). You can expand the signal tree until the selected signal is a boundary signal or starts a loop.

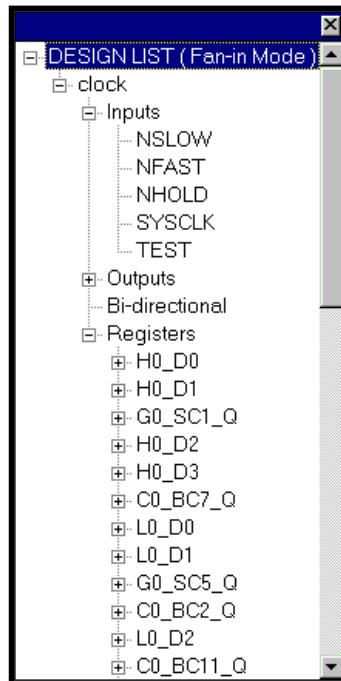


Figure 6-1. Signal Navigator

## Pop-up Menus from the Signal Navigator

When you click the right mouse button within the Signal Navigator, a number of commands are available, depending on where the cursor is when you click the right mouse button.

From the Signal Navigator window, you can select the following commands:

- **Fan-In Mode** – Changes the display to fan-in mode.
- **Fan-Out Mode** – Changes the display to fan-out mode.
- **Hide** – Removes the Signal Navigator from the screen.

From the design name in the Signal Navigator tree, you can select the following commands:

- **Timing Matrix Table** – Displays the Timing Matrix Table for the design.
- **Longest Timing Path** – Calculates and highlights the design's longest timing path in the Timing Matrix Table.
- **Shortest Timing Path** – Calculates and highlights the design's shortest timing path in the Timing Matrix Table.
- **Frequency** – Calculates and highlights the maximum design frequency in the Frequency Table.

From a signal category or signal name in the Signal Navigator tree, you can select the following commands:

- **Add Source Timing Tag** – Adds the highlighted signal to the sources in the Timing Matrix Table. You need to select **Show Timing Data** to calculate values for this source.
- **Add Destination Timing Tag** – Adds the highlighted signal to the destinations in the Timing Matrix Table. You need to select **Show Timing Data** to calculate values for this destination.
- **Frequency** – Adds the highlighted signal to the frequency table and shows the values for that signal.
- **Tco Path** – Displays the Tco Path Table with the Tco values for that signal.
- **Setup and Hold** – Displays the Setup and Hold Table with the Setup and Hold values for that signal.
- **Tpd Path** – Displays the Tpd Table with the Tpd values for that signal.

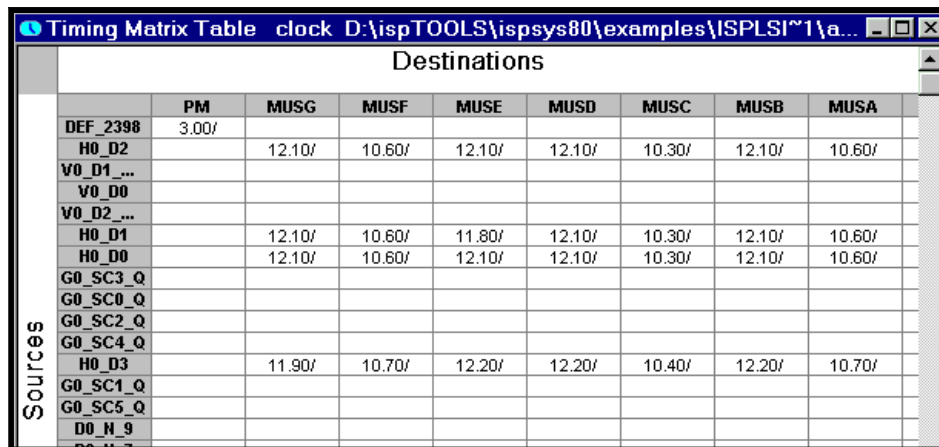
## Timing Explorer Tables

The following tables are available in the Timing Explorer. They are displayed when you request that timing information from the Compiler **Results** menu, from the Physical Viewer, or from within the Timing Explorer.

- Timing Matrix Table (Figure 6-2)
- Clock Frequency Table (Figure 6-3)
- Setup and Hold Table (Figure 6-4)
- Tco Table (Figure 6-5)
- Tpd Table (Figure 6-6)

You may wish to tile or cascade the tables for easier access. You can adjust the column widths by moving the cursor to the line at the right side of the column heading and dragging it to change the column size.

Once the tables have been created, you can move between the tables or redisplay a closed table using the **View** menu or the **Timing Matrix Table**, **Frequency Table**, **Setup and Hold Table**, **Tco Path Table**, and **Tpd Table** icons from the tool bar. If you select these icons before requesting a table through a popup menu, the table displays but it does not contain any data. You can also close a table or the Signal Navigator by deselecting an icon or a menu item on the **View** menu.



		Destinations							
		PM	MUSG	MUSF	MUSE	MUSD	MUSC	MUSB	MUSA
	DEF_2398	3.00/							
	HO_D2		12.10/	10.60/	12.10/	12.10/	10.30/	12.10/	10.60/
	V0_D1_...								
	V0_D0								
	V0_D2_...								
	HO_D1		12.10/	10.60/	11.80/	12.10/	10.30/	12.10/	10.60/
	HO_D0		12.10/	10.60/	12.10/	12.10/	10.30/	12.10/	10.60/
	G0_SC3_Q								
	G0_SC0_Q								
	G0_SC2_Q								
	G0_SC4_Q								
	HO_D3		11.90/	10.70/	12.20/	12.20/	10.40/	12.20/	10.70/
	G0_SC1_Q								
	G0_SC5_Q								
	DO_H_9								
	DO_H_7								

Figure 6-2. Timing Matrix Table

When paths in the Timing Matrix table are preceded by 'D:', the delay is to the data input of that register. When paths are preceded by 'CLK:', the delay is to the CLK of that register.

Frequency Table

Source ...	Source ...	Destina...	Destina...	Clock P...	Frequat...	GLB Le...
SYSCLK	US_BC3...	SYSCLK	US_BC3...	19.10	53	2
SYSCLK	US_BC3...	SYSCLK	US_BC1...	18.80	53	2
SYSCLK	US_BC2...	SYSCLK	US_BC1...	18.80	53	2
SYSCLK	US_BC1...	SYSCLK	US_BC1...	18.80	53	2
SYSCLK	US_BC8...	SYSCLK	US_BC1...	18.70	53	2
SYSCLK	US_BC0...	SYSCLK	US_BC1...	18.70	53	2
SYSCLK	US_BC1...	SYSCLK	US_BC1...	18.60	54	2
SYSCLK	US_BC5...	SYSCLK	US_BC1...	18.60	54	2
SYSCLK	N_59_Q...	SYSCLK	US_BC1...	18.60	54	2
SYSCLK	US_BC7...	SYSCLK	US_BC1...	18.60	54	2
SYSCLK	US_BC4...	SYSCLK	US_BC1...	18.60	54	2
SYSCLK	US_BC9...	SYSCLK	US_BC1...	18.50	54	2
SYSCLK	US_BC1...	SYSCLK	US_BC1...	18.50	54	2
SYSCLK	US_BC1...	SYSCLK	US_BC1...	18.40	54	2
SYSCLK	US_BC8...	SYSCLK	US_BC1...	18.40	54	2

Figure 6-3. Clock Frequency Table

Setup and Hold Table

Registe...	Data	Refer...	Setup (ns)	Hold (ns)
US_BC3...	TEST	SYSCLK	7.50	-0.90
US_BC1...	TEST	SYSCLK	7.50	-0.90
US_BC0...	TEST	SYSCLK	7.20	-0.90
N_61_Q...	TEST	SYSCLK	7.00	-0.40
US_BC7...	TEST	SYSCLK	7.50	-0.90
US_BC8...	TEST	SYSCLK	7.50	-0.90
US_BC6...	TEST	SYSCLK	7.50	-0.90
US_BC1...	TEST	SYSCLK	7.20	-0.90
US_BC2...	TEST	SYSCLK	7.20	-0.90
US_BC1...	TEST	SYSCLK	7.50	-0.90
N_59_Q...	TEST	SYSCLK	7.20	0.00
US_BC9...	TEST	SYSCLK	7.50	-0.90
US_BC1...	TEST	SYSCLK	6.80	0.00
US_BC1...	TEST	SYSCLK	7.20	0.00
US_BC1...	TEST	SYSCLK	7.50	0.00

Figure 6-4. Setup and Hold Table

Tco Table

Registe...	Source ...	Destina...	Delay(ns)
US_BC1...	SYSCLK	COLON	14.50
N_61_Q...	SYSCLK	COLON	14.70
US_BC1...	SYSCLK	COLON	14.60
N_60_Q...	SYSCLK	COLON	14.70

Figure 6-5. Tco Table

Tpd Table

Source ...	Destina...	Delay (ns)
DATAIN2	DATAOUT	10.80
DATAIN1	DATAOUT	10.80

Figure 6-6. Tpd Table

## Pop-Up Menus from the Timing Tables

When you click the right mouse button from the signal name (row or column headers) in the Timing Matrix Table, the following commands are available:

- **Sort** – Rearranges the table so the values in the column with the cursor are arranged from lowest to highest or in alphabetical order, as appropriate.
- **Show Timing Data** – Shows the timing data for the entire row or column.
- **Add Signal** – When you select the command, a dialog box displays so you can enter the name of a signal you want to add to the table.
- **Remove Signal** – When you select this command, the signal name where the cursor is located is removed from the table.

When you click the right mouse button on a cell in the Timing Matrix Table, the following commands are available:

- **Show Timing Data** – Shows the timing data for that cell.
- **Display Timing Path** – Cascades to show Longest Path, Shortest Path, and Both Paths. Choose one of these options to have the path display in the Connectivity window of the Physical Viewer. The Physical Viewer tool bar icon changes to show you are in Timing Mode.
- **Report Timing Path** – In a new window, displays the timing path as a text report.

When you click the right mouse button on the signal name in the Frequency Table, the **Display Timing Path** and **Report Timing Path** commands are available.

When you click the right mouse button on a cell in the Source Register or Destination Register columns in the Frequency Table, the following additional commands are available:

- **Setup and Hold Table** – Displays the setup and hold values of the selected register in the Setup and Hold Table.
- **Tco Table** – Displays the Tco data of the selected register in the Tco Table.

From a cell in the Register column of the Setup and Hold Table, you can access the Tco Table, and it will contain a value for the highlighted register. From a cell in the Register column of the Tco Table, you can access the Setup and Hold Table, and it will contain a value for the highlighted register.

## Timing Path Report

When you select the **Report Timing Path** command, the Timing Path Information Window (Figure 6-7) displays. The report information varies, depending on the table you were in when you requested the report. The boundary report displays when you access the Timing Path Report. Click the **Detailed Report** button to switch to the detailed report. When the detailed report displays, click the **Boundary Report** button to switch back to the boundary report. Click the **Display Path** button to display the path in the Connectivity window of the Physical Viewer.

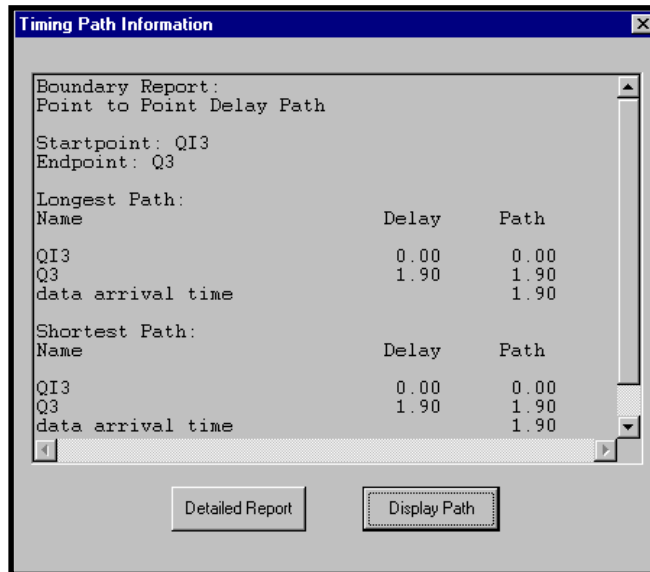


Figure 6-7. Timing Path Information Window



## Performance Analyst for MACH Designs

The Performance Analyst is a static timing analysis tool that enables a user to quickly determine the performance of MACH designs implemented in ispDesignExpert.

Worst case signal delays are reported in a graphical spreadsheet format that can be filtered by the user to verify the speed of critical paths and identify performance bottlenecks.

The Performance Analyst traces each logical path in the design and calculates the path delays using the device's timing model and worst case AC specs supplied in the device data sheet.

The Performance Analyst performs six distinct analysis types: fMAX, tSU, tPD, tCO, tOE, and tCOE. Timing threshold filter, source and destination filter, and path filter can be used to independently fine-tune each analysis.

### Analysis Types

There are six types of analysis you can perform using the Performance Analyst. The first type, fMAX, is an internal register-to-register delay analysis. fMAX measures the maximum clock operating frequency, limited by worst-case register-to-register delay. The remaining five types are external pin-to-pin delay analysis. The following describes these types.

#### fMAX

**Maximum Clock Operating Frequency** – The fMAX path trace analysis reports the worst-case fMAX (maximum clock operating frequency) for each clock in the design. fMAX is equal to reciprocal of the worst case register-to-register delay (Figure 6-8).

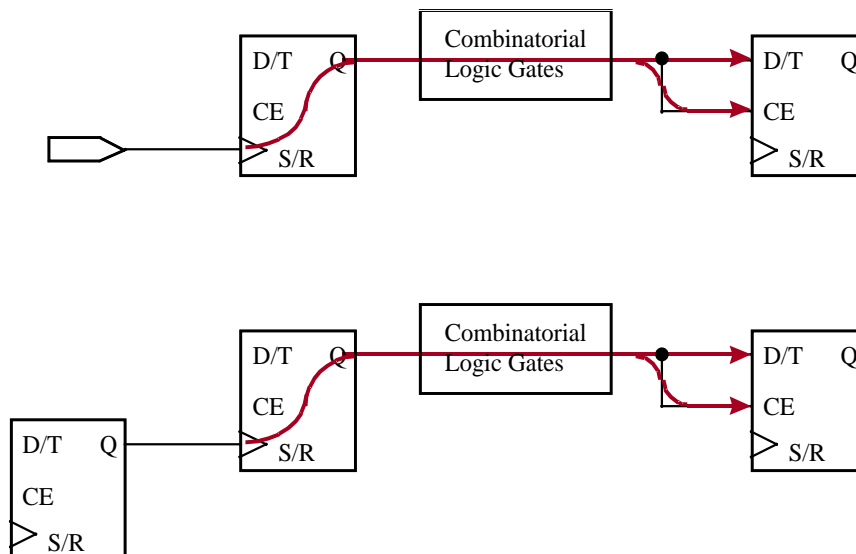


Figure 6-8. Default Register-to-Register Delay Path Tracing

The Performance Analyst reports all register-to-register delays in a spreadsheet format with clock sources displayed. You can specify which clocks the Performance Analyst reports in the spreadsheet, and whether tracing is enabled through all tracing paths. When there are no register paths in the design, the Start button is disabled and the spreadsheet is empty.

The Performance Analyst does not attempt to report external fMAX because it cannot make assumptions about the arrival time of signals driving MACH device inputs, and the tSU of devices driven by MACH device outputs. Therefore it is up to you to determine the external fMAX based on the operating requirements of the system.

### Default fMAX Path Trace

The default fMAX path starts at the source register clock input and traces through the clock-to-output path of the register, through any number of levels of combinatorial logic (through internal feedback only), to the D, T, or CE inputs of the destination register, including destination register setup time (tSU).

The Performance Analyst assumes that the same clock signal and the same edge of the clock signal clock the source and destination registers. However, delays are calculated when the source and destination registers are clocked by two clock signals or by different edge of the same clock signal. In the first case, the delay obtained is actually the setup time of the destination register through the clock-to-output path of the source register. In the second case, the actually fMAX will be half of what is calculated by the tool.

### tSU

**Setup Time** – The tSU path trace analysis reports setup and hold time for data and clock enable signals with respect to a clock edge, or the register recovery time from asynchronous S/R inputs. You can specify whether tracing is checked at the register's D/T, CE or S/R inputs.

### Default tSU Path Trace

This data path starts at an input pin and then traces through any number of levels of combinatorial logic to the D, T or CE inputs of a register. The internal tSU of the register is added to the delay path. The value of the internal tSU is dependent on the register being clocked by a global clock or product term clock.

The global clock net clock delay is modeled as 0ns for MACH CPLDs, so it is not used in the tSU calculation. The tSU of the register is adjusted for the skew between the global clock and data path.

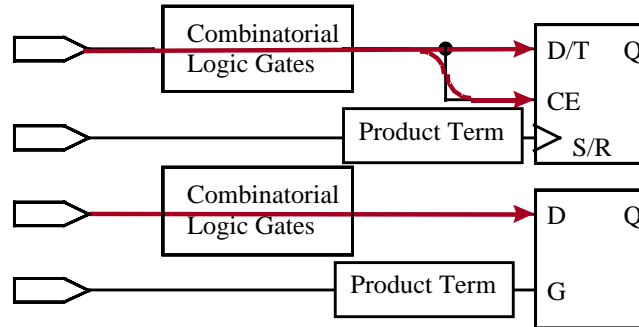


Figure 6-9. Default tSU Path Tracing

### Path Endpoints for tSU

$$t_{SU} = (\text{longest\_data\_path\_delay}) - (\text{shortest\_clock\_path\_delay}) + (\text{internal\_setup\_time}).$$

$$t_{HD} = (\text{longest\_clock\_path\_delay}) - (\text{shortest\_data\_path\_delay}) + (\text{internal\_hold\_time}).$$

For simplicity, in the Timing Analysis spreadsheet tHD will be shown as a “0” if the calculation is negative, regardless of its value. However, the exact hold time can be observed on the Expanded Delay Path window, which is opened by double clicking on the spreadsheet cell.

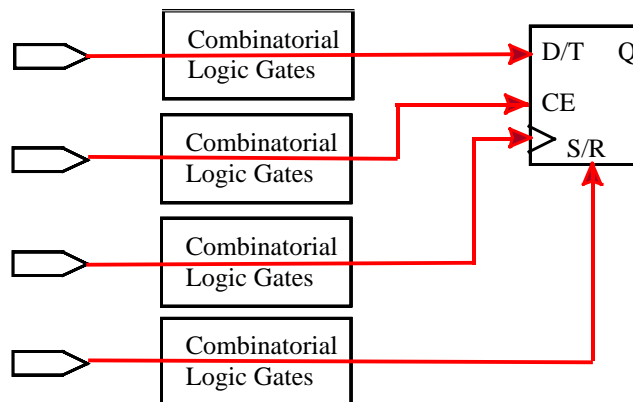


Figure 6-10. Tracing tSU at D/T, CE, CLOCK, and S/R

**Register D/T Inputs**

Reports  $t_{SU}$  /  $t_{HD}$  at Register data-input (D/T).

**Register CE Inputs**

Reports  $t_{SU}$  /  $t_{HD}$  at Register Clock Enable (CE).

**Register Asynch S/R Inputs**

Reports Recovery-time at Register Set/Reset.

**tPD**

**Propagation Delay Time** – The tPD path trace analysis reports input pin to output pin delay of combinatorial signals. You can specify whether reporting is enabled for paths traced through asynchronous register inputs and transparent latches.

**Default tPD Path Trace**

This path starts at an input pin and traces through any number of levels of combinatorial logic, through the data path of the output buffer, to the output pin.



Figure 6-11. Default tPD Path Tracing

**tCO**

**Clocked Output-to-Pin Time** – The tCO path trace analysis reports clock-to-out delay starting from the primary input, going through the clock of flip-flops or gate of latches, and ending at the primary output. You can specify whether reporting is enabled for paths traced through asynchronous register inputs, ripple clocks, or data-input of transparent latch.

**Default tCO Path Trace**

This path starts at an input pin and traces through any number of levels of combinatorial logic to the clock pin of a register. Tracing continues through the clock-to-output path of the register and through any number of levels of combinatorial logic, through the data path of the output buffer to the output pin. Only a single register clock-to-output delay exists in this path.

When tracing input latch gate to output delays, the path starts at the pin, traces through the gate-to-output path of the latch and through any number of levels of combinatorial logic, through the data path of the output buffer to the output pin. Only a single latch gate-to-output delay exists in this path.

Paths are not reported if they trace through asynchronous register set/reset inputs, ripple clocks, output enable paths, or transparent input latches.

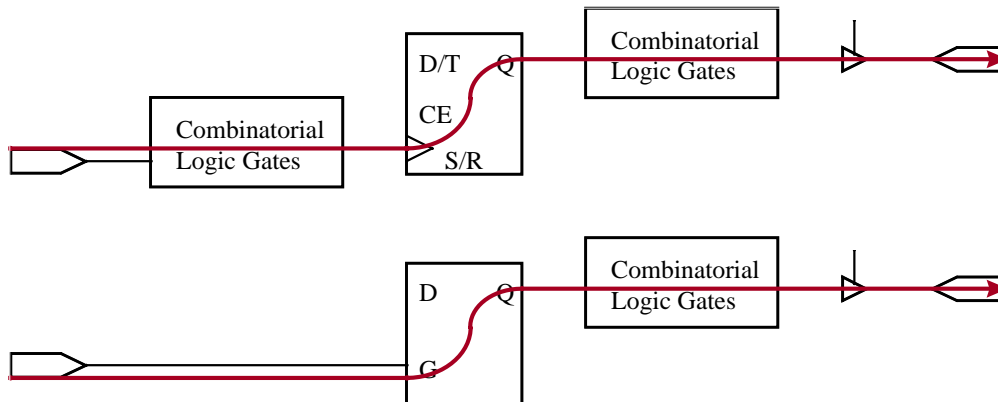


Figure 6-12. Default tCO and tGO Path Tracing

## tOE

**Output Enable Path Delay** – The tOE path trace analysis reports the input pin-to-output enable path delay starting from the primary input, through the Enable of output buffers, ending at the primary output. You can specify whether reporting is enabled for paths passing through the asynchronous register inputs or data-input of transparent latches.

### Default tOE Path Trace

This path starts from the primary input pin and traces any number of levels of combinatorial logic, through the Enable of output buffers, to the primary output.



Figure 6-13. Default tOE Path Tracing

## tCOE

**Clock to Output Enable Time** – This path trace analysis reports the input clock –to-output enable path delay starting from the primary input, going through the clock of flip-flops or gate of latches, going through the Enable of output buffer, and ending at the primary output. You can specify whether reporting is enabled for paths traced through asynchronous register inputs, ripple clocks, or data-input of transparent latch.

## Default tCOE Path Tracing

This path starts from primary input pin and traces through the Register Clock, through any number of levels of combinatorial logic, to the Enable of output buffers.



Figure 6-14. Default tCOE Path Tracing

## Running Timing Analysis

You must target a certain MACH device for your design before you run timing analysis. If you have not already run the Fitter, ispDesignExpert will fit the device automatically when you start the Timing Analysis process.

*To start the Performance Analyst:*

1. In the Sources window, select the target device. Note that you must have a specific MACH device chosen.
2. Double click the Timing Analysis process in the Processes window.

The ispDesignExpert Process dialog box appears momentarily indicating the progress. Then the Performance Analyst window is displayed as shown in Figure 6-15.

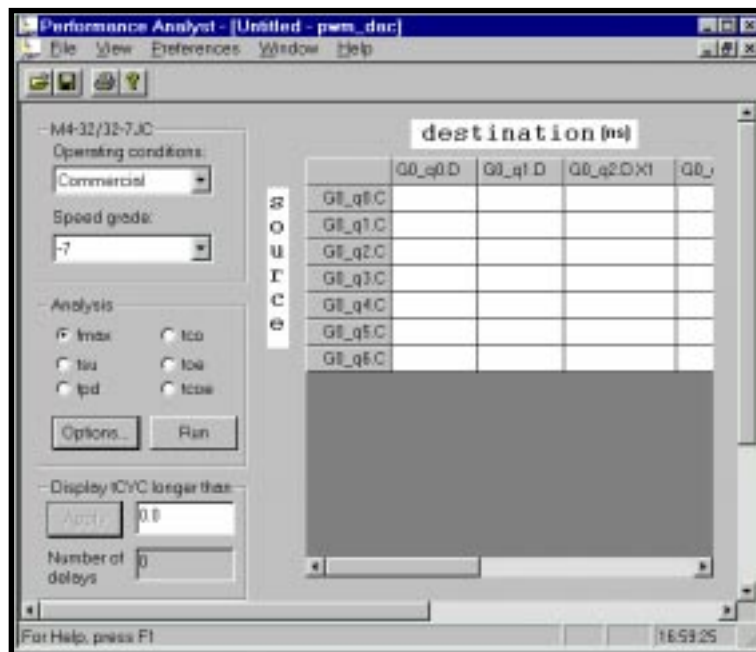


Figure 6-15. Performance Analyst Window

To set options and filters:

1. In the Analysis file, select the path analysis report that you want to run. These choices determine the path tracing rules that the Performance Analyst will follow during timing analysis.
2. Click **Options** to open the Options dialog box (Figure 6-16).

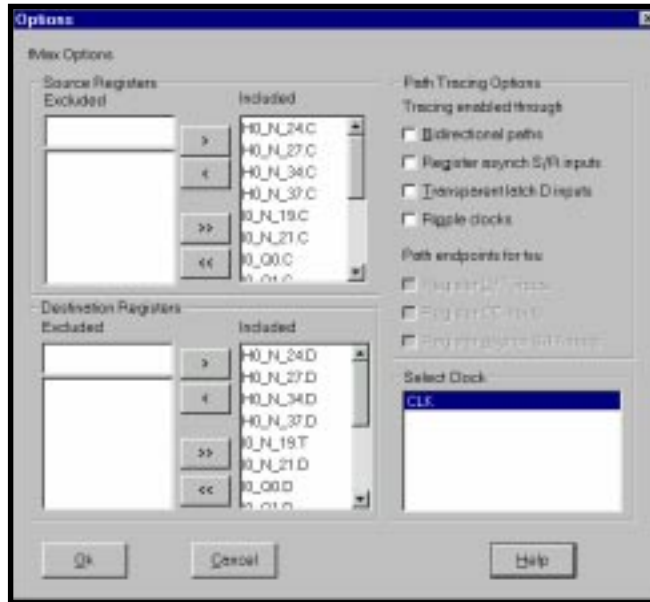


Figure 6-16. Options Dialog Box



**NOTE**

The Options dialog box will vary slightly depending on the analysis type you have chosen.

3. In the Source Registers and Destination Registers fields, exclude the items from the list that you do not want to include in the timing analysis.
4. In the Path Tracing Options field, select the tracing path and endpoint options you want to use.



**NOTE**

Endpoint options are only enabled for tSU.

5. Click **OK** to close the Options dialog box.

To run timing analysis and evaluating the results:

1. In the Analysis field of the Performance Analyst, click **Run**. The Performance Analyst calculates the delay paths according to the options you have chosen and displays them in the table (Figure 6-17).

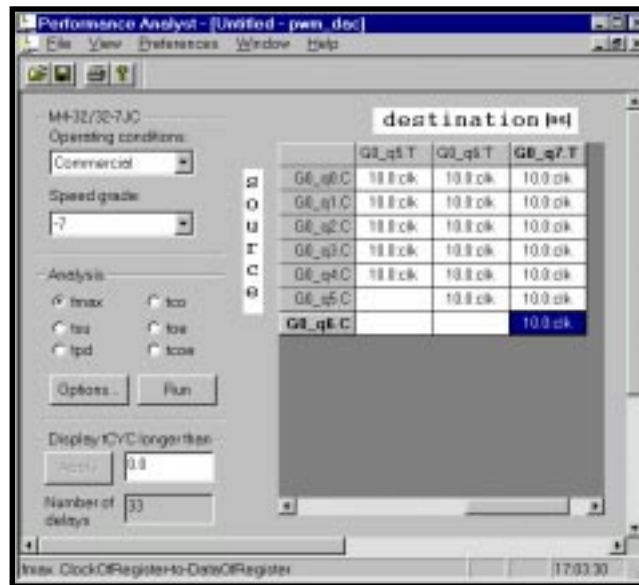


Figure 6-17. Performance Analyst Calculating the Delay Paths

2. You can analyze individual timing components used to calculate the timing path by double clicking on the field to open the Expanded Path dialog box.

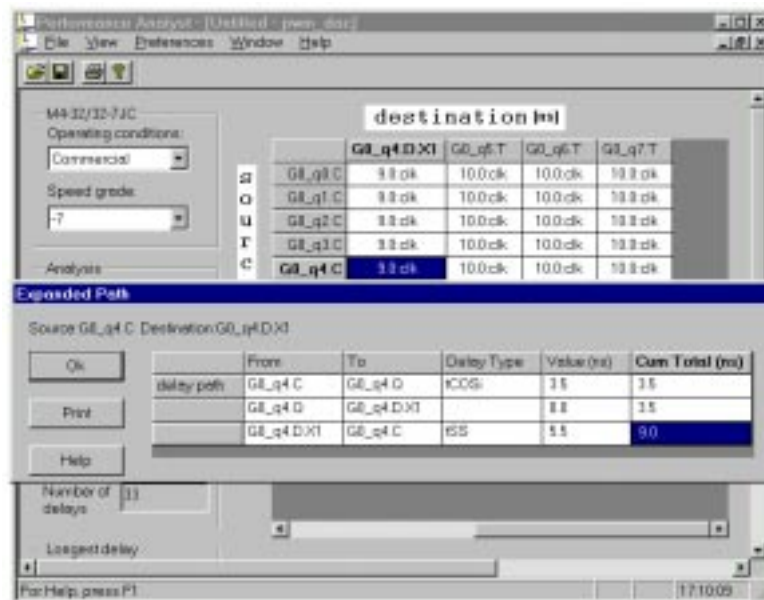


Figure 6-18. Expanded Paths Dialog Box

3. Click **Ok** to close the Expanded Paths dialog box.
4. You can print an entire analysis report by clicking the Print icon in the Toolbar or by selecting **File** ⇒ **Print**. Also, you can print the timing results of individual paths by clicking **Print** button in the Expanded Path dialog box.



## Running Timing Analysis in Batch Mode

There may be times when you want more precision and flexibility while running timing analysis than is available with the Performance Analyst graphic user interface. For example, on the Options dialog you can select Bidirectional path tracing as either “on” or “off.” However, this selection applies to all Bidirectional I/O’s in design. There is no way to select an individual one or a partial set.

Or in another example, six fixed groups cover the total path traces allowed by the user interface. There is no way to report a path between two arbitrary points in a design.

In addition to using the graphical user interface to run timing analysis, you can run the Performance Analyst in “batch mode.” This feature is called the Batch Timer. The Batch Timer executes a user-predefined command file and puts the result into a log file.

Refer to the [Design Verification Tools User Manual](#) for details of Batch Commands and Batch Timer usage.

# Index

---

## Symbols

.abl [27](#), [40](#), [92](#)  
.abv [27](#), [137](#)  
    location of [137](#)  
.doc [27](#)  
.ed\* [27](#)  
.edf [56](#), [59](#), [142](#)  
.edn [142](#)  
.err [140](#)  
.hlp [27](#)  
.ini [34](#)  
.log [140](#)  
.naf [50](#), [88](#), [95](#)  
.par [27](#)  
.ppn [118](#)  
.prp [27](#)  
.sch [27](#), [44](#), [50](#)  
.sim [142](#)  
.slg [140](#)  
.sym [37](#)  
.syn [27](#), [37](#)  
.tb [151](#)  
.tf [27](#), [151](#)  
.tfl [150](#)  
.txt [27](#)  
.v [27](#), [57](#)  
.vhd [27](#), [53](#)  
.vht [151](#)  
.wdl [27](#), [137](#)  
    association of [137](#)  
.wri [27](#)  
.xls [27](#)

## A

ABEL-HDL modules  
    add to a design [40](#)  
    compile [92](#)  
    create [42](#), [91](#)  
    import [40](#)  
Assigning properties to nodes  
    in ABEL-HDL [98](#)  
    in lower-level schematics [97](#)  
Attributes  
    ispLSI design [96](#)  
    assigning in ABEL-HDL [98](#)

    assigning in schematic [97](#)  
    precedence of [101](#)  
    processing [99](#)  
MACH design [102](#)  
    assigning in ABEL-HDL [103](#)  
    assigning in schematic [102](#)  
    syntax in ABEL-HDL [103](#)  
processing [99](#)  
table of [99](#)

## B

Balance partitioning (advanced)  
    MACH global optimization option [129](#)  
BFM packing (advanced)  
    ispLSI compiler property [125](#)  
Boolean logic reduction  
    MACH global optimization option [129](#)  
Boolean logic synthesis  
    MACH global optimization option [129](#)  
Boundary Report [160](#)

## C

Calculate frequency  
    in Timing Viewer [156](#)  
Carry pin direction  
    ispLSI compiler property [123](#)  
Case sensitive  
    ispLSI compiler property [123](#)  
Clock Frequency Table  
    Timing Explorer [157](#)  
Collapse all nodes (for speed)  
    MACH global optimization option [130](#)  
Collapse selective nodes (for area)  
    MACH global optimization option [130](#)  
Compiler properties  
    ispLSI [118](#)  
        BFM packing (advanced) [125](#)  
        Carry pin direction [123](#)  
        Case sensitive [123](#)  
        Effort [120](#)  
        Free all pin locks [122](#)  
        Ignore reserved pins [122](#)  
        invoke dialog box [118](#)  
        Maximum GLB inputs [121](#)  
        Maximum GLB outputs [121](#)

- Minimize GLB levels for all paths
  - (advanced) [124](#)
- Parameter file [123](#)
- Settings [118](#)
- Single PT function packing for routability
  - (advanced) [125](#)
- Strategy [120](#)
- Timing Analyzer [123](#)
- Use extended routing [122](#)
- Use global reset [120](#)
- Use internal tristate IO driver
  - (advanced) [124](#)
- Constraint Editor
  - Assigning Power Level [113](#)
  - for MACH [109](#)
  - Group Assignment [111](#)
  - invoke [109](#)
  - JEDEC File Options [112](#)
  - Location Assignment [110](#)
  - make [110](#)
  - main window [109](#)
  - Output Slew Rate Control [114](#)
  - Pin Reservation [112](#)
- Constraint Manager
  - Assign Attribute Values [108](#)
  - Design Browser [107](#)
  - for ispLSI [106](#)
  - invoke [106](#)
  - main window [106](#)
  - Net Attributes Table [108](#)
  - Pin Attributes Table [107](#)
  - Symbol Attributes Table [108](#)
- Constraints
  - MACH/PAL [66](#)

## D

- D/T synthesis
  - MACH global optimization option [130](#)
- Design
  - hierarchy [30, 67](#)
  - ispLSI/GAL
    - compiling/fitting [132](#)
  - MACH/PAL
    - compiling/fitting [134](#)
  - procedures [23](#)
  - processing [20, 34](#)
  - targeting a device [18](#)
- Design attributes
  - add to a symbol [48](#)
  - ispLSI [96](#)
    - assigning in ABEL-HDL [98](#)
    - assigning in schematic [97](#)

- precedence of [101](#)
- processing [99](#)
- MACH [102](#)
  - assigning in ABEL-HDL [103](#)
  - assigning in schematic [102](#)
  - syntax in ABEL-HDL [103](#)
- Design control properties
  - ispLSI [51](#)
- Design entry
  - ABEL-HDL [40](#)
  - EDIF [61](#)
  - mixed Schematic and ABEL-HDL [87](#)
  - mixed Schematic and Verilog HDL [95](#)
  - mixed Schematic and VHDL [94](#)
  - Schematic [44](#)
  - Verilog HDL [57](#)
  - VHDL [53](#)
- Detailed Report [160](#)
- Device
  - targeting [18](#)
- Device options
  - ispLSI [125](#)
  - ISP [125](#)
  - ISP except Y2 [125](#)
  - Lowpower [126](#)
  - Security [125](#)
  - TOE\_AS\_IO [126](#)
  - Y1 as reset [126](#)
- Device selection [29](#)
  - dialog box [29](#)

## E

- EDIF
  - create for ispLSI [56, 59](#)
  - import mechanism for MACH/PAL [66](#)
  - import netlist [61](#)
  - property files for MACH/PAL [64](#)
- Effort
  - ispLSI compiler property [120](#)
- Environment and configuration
  - change [34](#)
- Equation simulation
  - run [144](#)
- Equation Simulator
  - for MACH/PAL [144](#)
  - simulation model [145](#)
  - view [145](#)
  - simulation report
    - control [146](#)

**F**

File names  
 reserved [37](#)

fMAX  
 analysis type  
 Performance Analyst [161](#)

Force a process to run [34](#)

Free all pin locks  
 ispLSI compiler property [122](#)

Frequency  
 calculation from Timing Viewer [156](#)

Frequency Table  
 pop-up menus from [159](#)

Functional simulation  
 ispLSI/GAL  
 run [137](#)

Verilog HDL  
 perform [152](#)

VHDL  
 perform [152](#)

**G**

Global optimization options  
 MACH [128](#)  
 Balance partitioning (advanced) [129](#)  
 Boolean logic reduction [129](#)  
 Boolean logic synthesis [129](#)  
 Collapse all nodes (for speed) [130](#)  
 Collapse selective nodes (for area) [130](#)  
 D/T synthesis [130](#)  
 Maximum % macrocells per block  
 used [131](#)  
 Maximum % of block inputs used [131](#)  
 Pack design [128](#)  
 Product term collapsing [130](#)  
 Product term equation splitting [130](#)  
 Set/Reset don't care [130](#)  
 Spread design [128](#)  
 Spread placement (advanced) [129](#)

**H**

HDL cross probing  
 for ispLSI [140](#)

HDL Viewer  
 use [92](#)

Hierarchical design [67](#)  
 ABEL-HDL [69](#)  
 example [70](#)  
 abstract [67](#)  
 advantages of [67](#)  
 approach [68](#)  
 Block symbols [67](#)

considerations [83](#)  
 naming [84](#)  
 nets [85](#)  
 aliasing [86](#)  
 overview [67](#)  
 rules of creating [68](#)

Schematic [73](#)  
 example [74](#)

Schematic/Verilog HDL  
 example [81](#)

Schematic/VHDL  
 example [78](#)  
 structure [83](#)  
 techniques [68](#)

Verilog HDL [80](#)

VHDL  
 example [77](#)

Hierarchy  
 modular design [67](#)  
 vs. sheets in schematics [68](#)

Hierarchy Browser  
 use [92](#)

Hierarchy Navigator  
 use [92](#)

**I**

Icons  
 Frequency Table [157](#)  
 processes [28](#)  
 project [26](#)  
 schematic [44](#)  
 Setup and Hold Table [157](#)  
 sources [27](#)  
 Tco Path Table [157](#)  
 Timing Matrix Table [157](#)  
 Tpd Table [157](#)

Ignore reserved pins  
 ispLSI compiler property [122](#)

INI files  
 edit [34](#)

Instantiation [28](#)

ISP  
 ispLSI device option [125](#)

ISP except Y2  
 ispLSI device option [125](#)

ispLSI compiler properties  
 BFM packing (advanced) [125](#)  
 Carry pin direction [123](#)  
 Case sensitive [123](#)  
 Effort [120](#)  
 Free all pin locks [122](#)  
 Ignore reserved pins [122](#)

Maximum GLB inputs [121](#)  
 Maximum GLB outputs [121](#)  
 Minimize GLB levels for all paths  
     (advanced) [124](#)  
 Parameter file [123](#)  
 Single PT function packing for routability  
     (advanced) [125](#)  
 Strategy [120](#)  
 Timing Analyzer [123](#)  
 Use extended routing [122](#)  
 Use global reset [120](#)  
 Use internal tristate IO driver  
     (advanced) [124](#)  
 XOR [120](#)  
 ispLSI device options [125](#)  
   ISP [125](#)  
   ISP except Y2 [125](#)  
   Lowpower [126](#)  
   Security [125](#)  
   TOE\_AS\_IO [126](#)  
   Y1 as reset [126](#)

**J**

JEDEC simulation  
   GAL/PAL  
     run [148](#)

**L**

Lattice Logic Simulator  
   for ispLSI/GAL [137](#)  
   invoke [141](#)  
   stand-alone mode  
     run [141](#)

Log  
   simulation [140](#)

Lowpower  
   ispLSI device option [126](#)

**M**

MACH global optimization options [128](#)  
   Balance partitioning (advanced) [129](#)  
   Boolean logic reduction [129](#)  
   Boolean logic synthesis [129](#)  
   Collapse all nodes (for speed) [130](#)  
   Collapse selective nodes (for area) [130](#)  
   D/T synthesis [130](#)  
   Maximum % macrocells per block  
     used [131](#)  
   Maximum % of block inputs used [131](#)  
   Pack design [128](#)  
   Product term collapsing [130](#)  
   Product term equation splitting [130](#)

  Set/Reset don't care [130](#)  
   Spread design [128](#)  
   Spread placement (advanced) [129](#)  
 Macro  
   add to schematic [46](#)  
 Maximum % of block inputs used  
   MACH global optimization option [131](#)  
 Maximum % of macrocells per block used  
   MACH global optimization option [131](#)  
 Maximum GLB inputs  
   ispLSI compiler property [121](#)  
 Maximum GLB outputs  
   ispLSI compiler property [121](#)  
 Menus  
   pop-up  
     in Signal Navigator [156](#)  
 Minimize GLB levels for all paths (advanced)  
   ispLSI compiler property [124](#)  
 ModelSim Simulator [149](#)

**N**

Net names  
   add in schematic [48](#)  
 Nodes  
   assigning properties to [97](#)

**O**

Open Design  
   File menu function  
     in Simulator Control Panel [141](#)  
 Open Stimulus  
   File menu function  
     in Simulator Control Panel [142](#)  
 Optimization  
   schematic attributes for [101](#)

**P**

Pack design  
   MACH global optimization option [128](#)  
 Parameter file  
   ispLSI compiler property [123](#)  
 Performance Analyst  
   analysis types  
     fMAX [161](#)  
     tCO [164](#)  
     tCOE [165](#)  
     tOE [165](#)  
     tPD [164](#)  
     tSU [162](#)  
   evaluate results [168](#)  
   for MACH/PAL [161](#)  
   run [168](#)

- start [166](#)
- Pin locking
  - for GAL/PAL [115](#)
  - in Verilog HDL designs [116](#)
    - example [116, 117](#)
    - syntax for LeonardoSpectrum [117](#)
    - syntax for Synplify [116](#)
  - in VHDL designs [115](#)
    - example [115](#)
    - syntax [115](#)
- Pop-up menus
  - Signal Navigator [156](#)
  - Timing Tables [159](#)
- Probe
  - in HDL [140](#)
  - in schematic [140](#)
- Process flow [29](#)
- Process types
  - output file [28](#)
  - process [28](#)
  - report [28](#)
  - tools [28](#)
- Processes window
  - Project Navigator [28](#)
- Product term collapsing
  - MACH global optimization option [130](#)
- Product term equation splitting
  - MACH global optimization option [130](#)
- Project
  - clean up [36](#)
  - create [15](#)
  - describe [29](#)
  - giving a title [16](#)
  - icon [26](#)
  - import sources [17](#)
  - removing files [22](#)
  - save [36](#)
  - tips for defining [30](#)
  - tips for naming [37](#)
  - tips for saving [37](#)
  - view path [21](#)
- Project Navigator
  - features [24](#)
  - menu bar [26](#)
  - processes window [28](#)
  - screen [25](#)
  - sources window [26](#)
  - title bar [26](#)
  - toolbar [26](#)
  - tools associated [24](#)
- Properties
  - ispLSI compiler [118](#)

- BFM packing (advanced) [125](#)
- Carry pin direction [123](#)
- Case sensitive [123](#)
- Effort [120](#)
- Free all pin locks [122](#)
- Ignore reserved pins [122](#)
- invoke dialog box [118](#)
- Maximum GLB inputs [121](#)
- Maximum GLB outputs [121](#)
- Minimize GLB levels for all paths
  - (advanced) [124](#)
- Parameter file [123](#)
- Settings [118](#)
- Single PT function packing for routability
  - (advanced) [125](#)
- Strategy [120](#)
- Timing Analyzer [123](#)
- Use extended routing [122](#)
- Use global reset [120](#)
- Use internal tristate IO driver
  - (advanced) [124](#)

## R

- Reserved file names [37](#)
- Run
  - Simulate menu function
    - in Simulator Control Panel [139, 143](#)

## S

- Schematic Editor
  - open [33](#)
- Schematics
  - add a blank sheet to a design [45](#)
  - add to a design [44](#)
  - add to an ABEL-HDL file [87](#)
  - import [44](#)
- Security
  - ispLSI device option [125](#)
- Set and Hold Table
  - Timing Explorer [157](#)
- Set/Reset don't care
  - MACH global optimization option [130](#)
- Show
  - Edit menu function
    - in Waveform Viewer [139](#)
  - Show Waveforms dialog box
    - in Waveform Viewer [139](#)
- Show Waveforms [139](#)
  - View menu function
    - in Simulator Control Panel [139](#)
- Show waveforms [147, 148](#)
- Signal Navigator [155](#)

- pop-up menus from [156](#)
- Simulation log [140](#)
- Simulator Control Panel
  - File menu functions
    - Open Design [141](#)
    - Open Stimulus [142](#)
  - Simulate menu function
    - Run [139](#), [143](#)
  - View menu functions
    - Show Waveforms [139](#)
- window [138](#)
- Single PT function packing for routability (advanced)
  - ispLSI compiler property [125](#)
- Source
  - ABEL-HDL test vector [27](#)
- Sources
  - ABEL-HDL [27](#)
  - being placed in the Project Navigator [32](#)
  - create [32](#)
  - document [27](#)
  - EDIF netlist [27](#)
  - hierarchy [28](#)
  - import [31](#)
  - modify [33](#)
  - project notebook [27](#)
  - remove [34](#)
  - schematic [27](#)
  - target device [27](#)
  - undefined [27](#)
  - Verilog HDL [27](#)
  - Verilog HDL test fixture [27](#)
  - VHDL [27](#)
  - VHDL test bench [27](#)
  - waveform stimulus [27](#)
- Sources window
  - Project Navigator [26](#)
- Spread design
  - MACH global optimization option [128](#)
- Spread placement (advanced)
  - MACH global optimization option [129](#)
- Starting ispDesignExpert [14](#)
- Strategy
  - ispLSI compiler property [120](#)
- Symbol
  - add to schematic [46](#)
  - add to top-level schematic [94](#)
  - create [50](#)
- Symbol Libraries [46](#)

## T

- tCO
  - analysis type
    - Performance Analyst [164](#)
- Tco Table
  - Timing Explorer [157](#)
- tCOE
  - analysis type
    - Performance Analyst [165](#)
- Test stimulus
  - create for Lattice Logic Simulator [137](#)
  - generate for ModelSim Simulator [149](#)
- Test vector
  - create for Equation Simulator [144](#)
- Text Editor
  - open [33](#), [43](#)
- Timing Analyzer
  - for ispLSI/GAL [154](#)
  - functions [154](#)
  - ispLSI compiler property [123](#)
- Timing Explorer
  - accessing [154](#)
  - adjust table columns [157](#)
  - Clock Frequency Table [157](#)
  - Setup and Hold Table [157](#)
  - Signal Navigator [155](#)
  - Tco Table [157](#)
  - Timing Matrix Table [157](#)
  - Tpd Table [157](#)
- Timing Matrix Table
  - pop-up menus from [159](#)
  - Timing Explorer [157](#)
- Timing Path Report [160](#)
- Timing Paths
  - displaying [160](#)
- Timing simulation
  - ispLSI/GAL
    - run [137](#)
  - Verilog HDL
    - perform [152](#)
  - VHDL
    - perform [152](#)
- tOE
  - analysis type
    - Performance Analyst [165](#)
- TOE\_AS\_IO
  - ispLSI device option [126](#)
- tPD
  - analysis type
    - Performance Analyst [164](#)
- Tpd Table
  - Timing Explorer [157](#)

**tSU**

- analysis type
- Performance Analyst [162](#)

**U****UES**

- Data type [127](#)
- Signature size [127](#)
- Use extended routing
  - ispLSI compiler property [122](#)
- Use global reset
  - ispLSI compiler property [120](#)
- Use internal tristate IO driver (advanced)
  - ispLSI compiler property [124](#)

**V****Verilog HDL modules**

- add to a design [57](#)
- create [58](#)
- import [57](#)

**VHDL modules**

- add to a design [53](#), [94](#)
- create [55](#), [94](#)
- import [53](#)

**W****Waveform Viewer**

- Edit menu function
- Show [139](#)

**Waveforms**

- show [139](#), [147](#), [148](#)

**X****XOR**

- ispLSI compiler property [120](#)

**Y****Y1 as reset**

- ispLSI device option [126](#)