ISP Synario System User Manual

096-0211-001

October 1996 096-0211-001

Synario Design Automation, a division of Data I/O, has made every attempt to ensure that the information in this document is accurate and complete. Synario Design Automation assumes no liability for errors, or for any incidental, consequential, indirect or special damages, including, without limitation, loss of use, loss or alteration of data, delays, or lost profits or savings, arising from the use of this document or the product which it accompanies.

Synario Design Automation
10525 Willows Road N.E., P.O. Box 97046
Redmond, Washington 98073-9746 USA
(206) 881-6444

Acknowledgments:

Synario and Data I/O are a registered trademarks of Data I/O Corporation.

Data I/O Corporation acknowledges the trademarks of other organizations for their respective products or services mentioned in this document.

Table of Contents

Before You Begin

1. Introduction 1-1

2. Designing With ISP Synario 2-1

Before You Begin

This manual covers how to:

- Install ISP Synario System
- Create designs for Lattice ispLSI, pLSI, ispGDS, and GAL devices
- Fit your design
- Simulate your design

This manual assumes you are familiar with the basics of operating the Synario Project Navigator. If you are not familiar with the Project Navigator, after installing the ISP Synario System, refer to the *ISP Synario System Design Tutorial*.

Minimum Requirements

For optimum performance of the ISP Synario System, you should have the following hardware:

- 486 microprocessor or better
- 16 MB or more of system memory
- At least 30 MB of free disk space in which to install the software

Installation

Installing the ISP Synario System software is a four step process:

- Install ISP Synario Entry/Simulation
- Install the ISP Fitter
- Install Win32s (for Windows 3.1 installations only)
- Install ISP Daisy Chain Download (optional, for downloading an ISP file to device)
- Install the Acrobat Reader (optional, for viewing the online manuals)

Note about Windows 3.1 and Windows 95

The following instructions specify that you enter text into the **RUN** text box.

In **Windows 3.1**, you can access the **Run** text box in the Windows Program Manager by doing the following:

1. Click on the **File** menu
2. From the File menu, click on the **Run** menu pick.

In **Windows 95**, you can access the **Run** text box from the **Start** menu by doing the following:

1. Click on the **Start** button.
2. From the Start menu, click on the **Run** menu pick.

Install ISP Synario Entry/Simulation

Use the instructions below to install the ISP Synario Entry / Simulation:

1. Insert the CD ROM which contains the ISP Synario programs into your CD ROM drive.

Note: If you are running Windows 95, when you insert your ISP Synario CD into your PC's CD-ROM drive, a dialog box automatically appears which displays choices you have. Click on Install to start the installation program and skip to step 3 below.

2. In the **Run** text box (from the Windows 3.1 Program Manager File menu or the Windows 95 Start menu), enter

   drive:\CDSETUP

   where *drive* is the disk drive letter where you inserted the ISP Synario CD ROM.

3. Select the icon for product installations.

4. Follow the instructions on the screen.

This completes ISP Synario Entry / Simulation installation for Windows 3.1+ and Windows 95.

If you have trouble with installation, refer to the README.WRI file located in the root directory of the Synario CD and to the installation program's online help.

Install the ISP Fitter

Use the instructions below to install the ISP Fitter:

1. In the **Run** text box (from the Windows 3.1 Program Manager File menu or the Windows 95 Start menu), enter

   drive:\pdsplus\SETUP

   where *drive* is the disk drive letter where you inserted the
   ISP Synario CD ROM.

2. Follow the instructions on the screen. Accept all the defaults when prompted, unless you already have a full version of the pDS+ Fitter installed. If this is the case, specify a different installation directory, such as **c:\synplus**.

This completes ISP Fitter installation for Windows 3.1+ and Windows 95.

If you have trouble with installation, refer to the *ISP Synario System Release Notes* and to the installation program's online help.

Install ISP Daisy Chain Download

Use the instructions below to install the ISP Daisy Chain Download program:

1. In the **Run** text box (from the Windows 3.1 Program Manager File menu or the Windows 95 Start menu), enter

   drive:\pdsplus\SETUP

   where *drive* is the disk drive letter where you inserted the
   ISP Synario CD ROM.

2. Follow the instructions on the screen. Accept all the defaults when prompted, unless you already have a full version of the pDS+ Fitter installed. If this is the case, specify a different installation directory, such as **c:\synplus**.

This completes Daisy Chain Download installation for Windows 3.1+ and Windows 95.

Install Win32s

If you have Windows 3.1+ (not Windows 95) and if you do not have Win32s version 1.25 or greater installed on your PC, or do not know what version you have, do the following:

Use the instructions below to install Win32s:

1. In the **Run** text box (from the Windows 3.1 Program Manager File menu), enter

   drive:\alliance\win32s\disk1\setup

   where *drive* is the disk drive letter where you inserted the ISP Synario CD ROM.

2. Follow the instructions on the screen.

The Win32s install will restart Windows.

Note: Do not install Win32s on Windows 95 or Windows NT.

This completes Win32s installation for Windows 3.1.

If you have trouble with installation, refer to the files **install.txt** and **readme.txt** located in the \alliance\win32s directory on the ISP Synario CD.

Install the Acrobat Reader

If you do not have a .PDF viewer installed, you need to install one to view the online manuals located on the ISP Synario CD.

Use the instructions below to install the Acrobat Reader:

1. In the **Run** text box (from the Windows 3.1 Program Manager File menu or the Windows 95 Start menu), enter

   drive:\alliance\acroread\acroread

   where *drive* is the disk drive letter where you inserted the
   ISP Synario CD ROM.

2. Follow the instructions on the screen.

3. (Optional) After the Acrobat Reader is installed, follow the directions below for installing Acrobat Plug-Ins.

   The "PLUG_INS" folder is the standard place to "install" Acrobat Plug-Ins. For Acrobat Reader (2.1 or later), this is located by default in c:\acroread\plug_ins.

   To install Acrobat Plug-Ins from the CD-ROM:

   a) Exit the Acrobat Read if it is open.

   b) Using File Manager or Explorer, select the CD-ROM drive.

   c) Click on the subdirectory "alliance".

   d) Click on the subdirectory "acroread".

   e) Copy the "plug_in.exe" file to the "PLUG_INS" folder (default c:\acroread\plug_ins):

   f) Double click on the program "plug_in.exe" in the PLUG_INS folder (do not double click on the file on the CD-ROM).

   To un-install Acrobat Plug-Ins: Drag the particular Acrobat plug-in you want to un-install from the "PLUG_INS" folder to the "OPTIONAL" folder.

   Note: Not all plug-ins are designed to work with Acrobat Reader 2.1 and Acrobat Reader 2.0 does not support plug-ins at all.

This completes the installation.

1. Introduction

Overview

The ISP Synario System provides support for the Lattice pLSI, ispLSI and GAL families under the Synario Project Navigator. The kit contains all executables, libraries and device support lists necessary to configure the Project Navigator for design entry, simulation and fusemap creation of these devices.

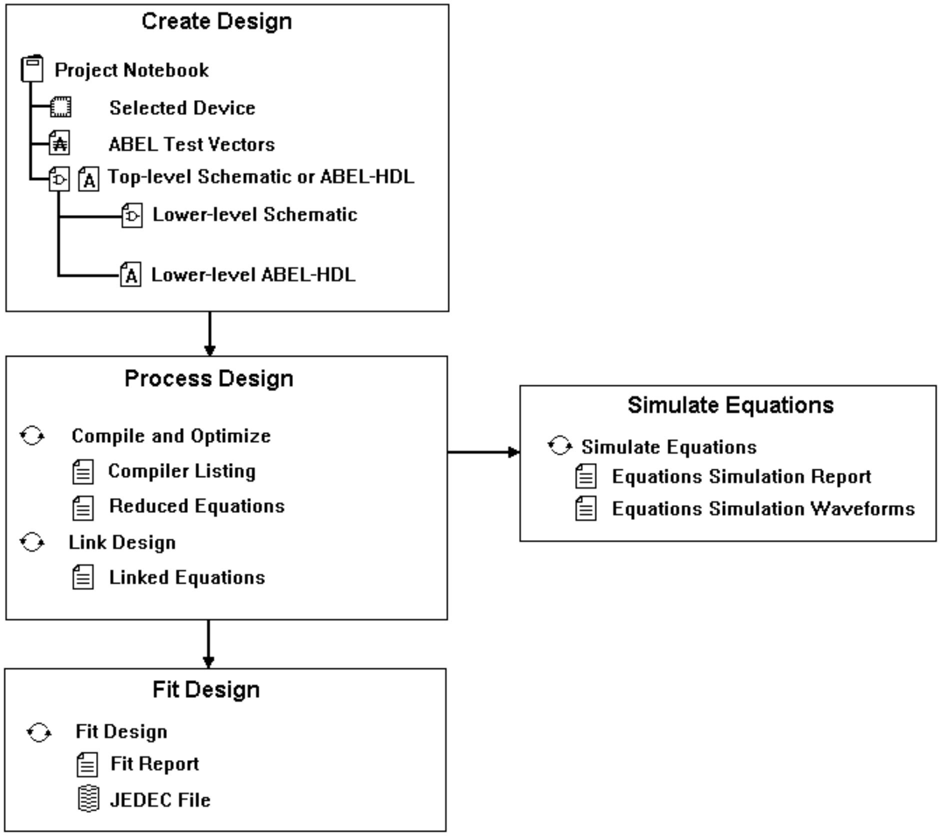
For the purposes of clarity, both pLSI and ispLSI devices will be referred to as "ispLSI" in this and following chapters.

Features

- Schematic entry
- ABEL-HDL entry
- Fit control properties
- Source reduction properties
- ABEL-HDL test vectors equation pre-fit simulation

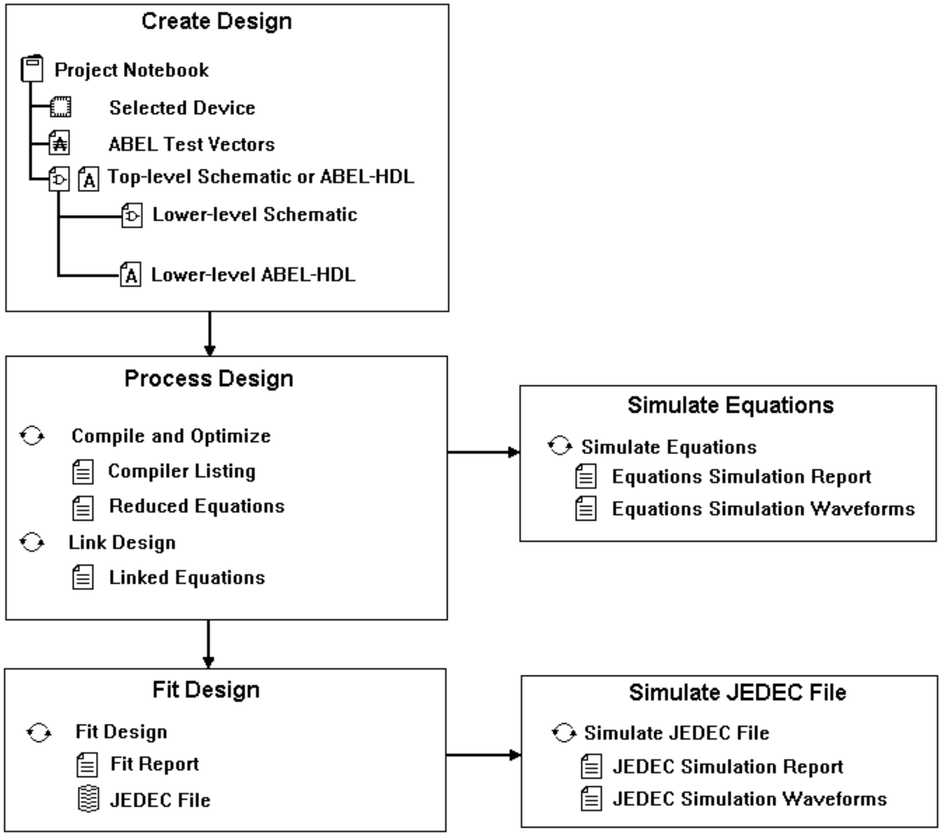
ispLSI Flow

The ispLSI project development flow available from the Project Navigator is illustrated in the figure below.



GAL Flow

The GAL project development flow available from the Project Navigator is illustrated in the figure below.



Device Support

The ISP Synario System version 3.0 supports the following devices:

ispLSI 1016/E

ispLSI 1024

ispLSI 1032/E

ispLSI 1048/C/E

ispLSI 2032/V/LV

ispLSI 2064

ispLSI 2096

ispLSI 2128

GAL Devices

ispGDS Devices

The device list is further broken out by package type and speed grade. Double-click on the device source in the Project Navigator to bring up the Choose Device dialog box and select any ispLSI device.

Refer to the *ISP Synario System Release Notes* for a complete list of supported devices.

Sources and Hierarchy

Sources and hierarchy are not limited. You can use ABEL-HDL modules or schematics at any level in the design.

2. Designing With ISP Synario

Start ISP Synario

After installing ISP Synario Design System, do the following to start the Project Navigator. (Refer to the beginning of this manual for instructions on how to install the ISP Synario System,.)

For Windows version 3.1+ Installations

1. Select the "ISP Synario" icon from the ISP Synario program group created during installation.

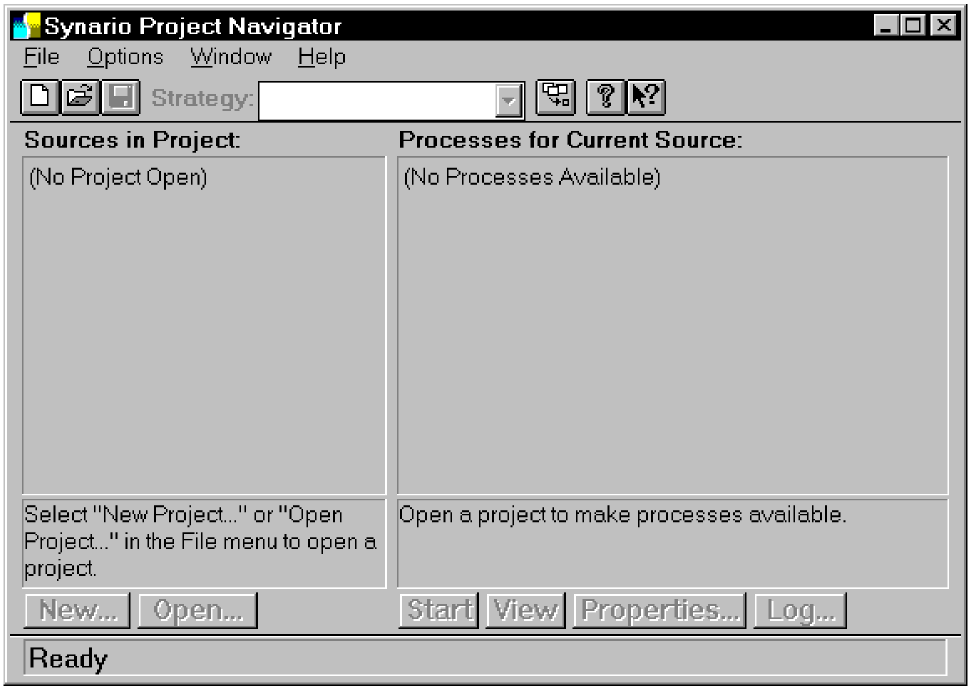
For Windows 95 Installations

1. Click on the **Start** button, and then select the following from the Start menus:

   a) **Programs**
   b) **ISP Synario 3.0 (or the current version of ISP Synario)**
   c) **ISP Synario**

After the program loads, the Project Navigator window appears.

Figure 2-1: ISP Synario Project Navigator



(No Project Open)

(No Processes Available)

Select "New Project..." or "Open Project..." in the File menu to open a project.

Open a project to make processes available.

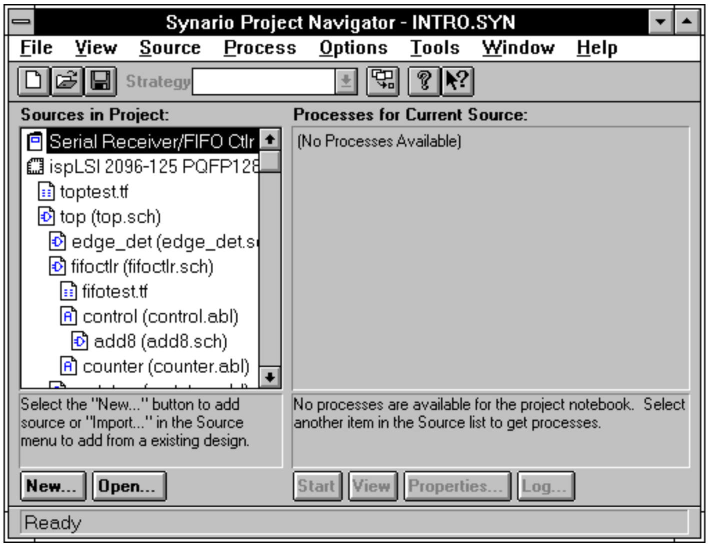
Ready

How the ISP Synario Project Navigator Works

ISP Synario employs the concept of a project. A project is a design. Each project has its own directory in which all source files, intermediate data files, and resulting files are stored. The picture shown below shows an example of what the Project Navigator might look like with a programmable IC project (called **intro.syn**) opened.

The Sources in Project Window (Sources window) shows all the design files associated with a project.

The Processes for Current Source Window (Processes window) shows the available processes for the selected source.



Synario Project Navigator - INTRO.SYN
File View Source Process Options Tools Window Help
Strategy
Sources in Project:
Processes for Current Source:
New...
Open...
Start
View
Properties...
Log...

Serial Receiver/FIFO Ctlr
ispLSI 2096-125 PQFP128
toptest.tf
top (top.sch)
edge_det (edge_det.s
fifoctlr (fifoctlr.sch)
fifotest.tf
control (control.abl)
add8 (add8.sch)
counter (counter.abl)

(No Processes Available)

Select the "New..." button to add source or "Import..." in the Source menu to add from a existing design.

No processes are available for the project notebook. Select another item in the Source list to get processes.

Ready

This picture shows the Project Navigator after a programmable IC project is opened. If no project is open, the Sources and Processes windows would be empty.

The Sources Window

The Sources window (on the left side of the Project Navigator) shows all the design files associated with a project. 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 engineering-notebook icon. In **intro**, the Project Notebook is labeled as "Serial Receiver / FIFO Ctlr." (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 Synario, including schematics, ABEL-HDL modules, and test stimulus for simulation.

The Notebook Icon (▤)

In the Project Navigator Sources window, projects are organized by collecting all of the files into a **Project Notebook** (represented by the notebook icon (▤)).

The Project Notebook lists the schematics and behavioral sources that create the logic of your design, testing files, and the device specification. The Project Notebook can also include any other design documents you want to keep with the design, such as design specifications, meeting notes or other supplementary files. Each project is stored in its own directory to simplify archiving.

Tip To rename the project, double-click the project icon (▤) in the Source window.

Project Sources

Your design can be represented in various ways. Those representations are called sources. A source is any element in the design such as a HDL file, schematic, or simulation file. Sources are displayed in the Project Navigator Source's Window.

They include not only the description of the circuits, as represented by schematics, state diagrams, and hardware description languages, but also include waveforms for simulation, stimulation test fixtures, links that connect represent connections to other projects, and documentation of the design. All those pieces are part of the whole design, which includes not only the circuits but the things you need to do to assure yourself that those circuits work as they ought to.

The design description (logic) for a project is contained within the following types of sources:

- Schematics
- ABEL-HDL

One source file in a project is the top-level source for the design. 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.*

You might have sources other than schematic and ABEL-HDL modules. These sources might include simulation test fixtures, documentation files, or other files related to Windows applications.

Creating a New Project

A new project may be created by either dragging and dropping existing ABEL-HDL files or Schematics, using the Project Navigator menus to import existing sources, or by creating new sources from within the Project Navigator.

For more detailed information on the ISP Synario Project Navigator and projects, refer to the *ISP Synario Design System Tutorial*.

Selecting a Device

To target your design to a specific device family:

1. Double-click on the **device icon** (▣) in the Project Navigator Sources window.
2. Click on a device from the available device selections, and then click on **OK**.

Using Examples, Help, and Online Documentation

Examples

Opening Examples is made a bit easier through **Open Examples** from the File menu.

For a description of an example, refer to the online documentation (accessible through the Help menu). You can also open the example in the Project Navigator and double-click on any documentation included with the project.

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 may be made:

- By choosing **Options: Environment** from the Project Navigator menus.
- Editing .INI files used by the Project Navigator and other programs

Refer to the Project Navigator online help for information. (The help for these topics can be accessed from the **Project Navigator Help Map** by clicking on the Start icon on the map and choosing from the selections.)

Design Tips for ISP Devices

- It is generally better to keep logic broken into small pieces while entering, when designing for ispLSI devices. Extra nodes can be collapsed away later if necessary; the final optimization pass will determine whether groups of nodes can be collapsed down efficiently without exceeding the product term limit. Use the **Fit Design** property **Node Collapse Type** to **None** for optimization of modules with large logic functions, such as adders or byte-wide comparators. This property keeps the fitter from removing nodes which help break the circuit up into smaller, more easily-fit pieces.
- Use the @CARRY directive in flat ABEL-HDL designs to break up large logic functions (such as comparators, truth tables, or adders) into smaller pieces linked with combinational summing (carry) nodes.
- Keep the Link process property node collapse range set at 10 or less for hierarchical designs to keep the product terms in the merged design within the routing limitations of the Lattice fitter.
- Avoid using an "interface" statement on the module declaration line of lower-level ABEL modules, as this can cause the upper-level schematic not to connect to it due to schematic netlister pin-name re-ordering.

Schematic Entry Considerations

The Generic Library

Schematics are built from symbols in the Device-Independent (Generic) symbol library. This library contains symbols for functions that are fairly universal in programmable device design description, such as gates, registers, multiplexers, and input/output pad symbols. Because each schematic module is compiled into a BLIF file, this device kit includes BLIF models for each of the generic symbols, which may be linked in with the schematic BLIF during processing.

Symbol Connections

When you enter a schematic, you can select symbols from the Generic or a device-specific library and place and interconnect them as desired. You can connect symbols by

- Drawing a wire from one symbol's pin to the other.
- Implying a connection by giving nets on each symbol the same net name.

Hierarchical Design

Bottom-up Design

You use net names and I/O markers to connect a schematic to an upper-level schematic. Connections between schematics are called "port" connections in Synario. When you have placed all port connections in the schematic, you can automatically create a block symbol for it by selecting **Matching Symbol** from the Schematic Editor's **File** menu. In this way, one may enter a design from the lowest level schematic first, working your way up to the top level. This is "bottom-up" design.

Top-down Design

Similarly, you can enter your design from the top-down by entering the top-level schematic first, and creating block symbols for functions needed but not yet described.

To create a block symbol:

1. Select **New Block Symbol** from the **Add** menu, and enter the symbol name, input pins, output pins and bi-directional pins defining the symbol desired. When you click OK, the described symbol is attached to the cursor.
2. Move and place the new symbol.

Note: *If you save a schematic that has symbols for which an underlying description does not exist, the Project Navigator automatically adds that function's name to the source list, and gives the icon for that source a question mark indicating that this function is not yet described.*

Schematic Attributes

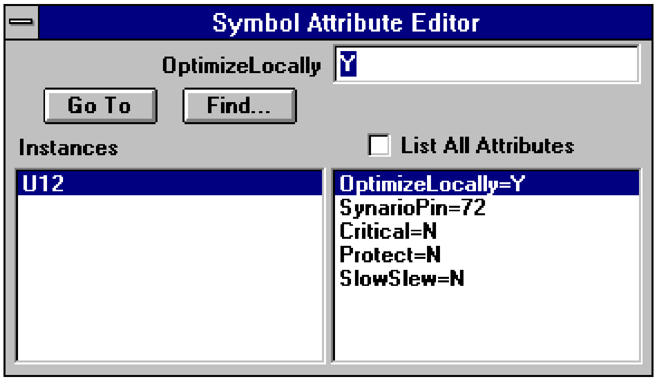
You use attributes on symbols and nets in a schematic to control certain aspects of ispLSI design processing and resource utilization.

To access symbol or net attributes:

1. Select **Attribute** from the **Add** menu. The dialog box lists the attributes available for the currently-selected item.

2. Click on the symbol or net whose attributes you wish to edit. A list of available attributes and their current settings appears.

   For example, an Input Pin symbol (G_INPUT) has the attribute dialog shown below:



OptimizeLocally Y

□ List All Attributes

U12

OptimizeLocally=Y
SynarioPin=72
Critical=N
Protect=N
SlowSlew=N

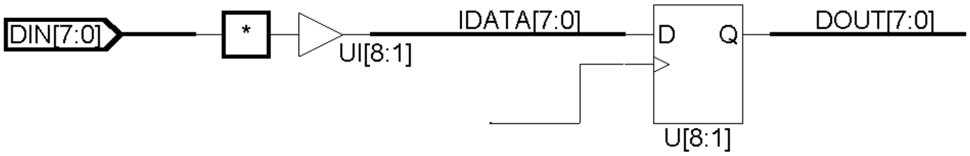
3. Select the attribute you wish to change, and enter the desired attribute value in the text entry box at the top of the attribute editor.

   The symbol attribute highlighted in this case, **OptimizeLocally**, is set to Y for the symbol **U1**.

You can access any applicable design-level ispLSI properties described in the ISP Synario Fitter using attributes in the schematic. For example, to ensure that a given register is placed in the IOC rather than GLB, you set the REGISTER_TYPE attribute to IOC for the register.

Iterated instances

A given symbol may be given an "iterated" instance name, making it in a sense more than one symbol. This is useful for large designs using datapaths or large signal groups. See the figure below.



In this example, the input buffer and the register both are iterated eight times. The input buffer's instance numbers are UI[8], UI[7], UI[6], etc. The data bus elements are connected up to the appropriate iteration of each symbol, such that each signal within the bus connects to a unique input buffer or register. Non-bus signals connected to iterated symbol instances are considered to be global to all iterations. For instance, a CLK input on the register's CLK pin would go to all of the CLK inputs to all of the iterated registers.

Schematic Naming Conventions

- Use only alphanumeric characters or underlines for net names (a-z,A-Z,_).
- VCC and GND are reserved net names, and may be used only to tie to HIGH or LOW power levels within a design.
- While ISP Synario is predominantly case-sensitive for net names, do not rely on case-sensitivity to make net names unique.

On Nodes and Nets

The schematic compiler automatically creates a BLIF node for each net that is encountered. This is necessary to allow properties and design control to have effect over a BLIF netlist from a schematic. However, it may also cause excess resources to be used unless processing and fitting optimization properties are set to remove excess or "redundant" nodes.

ABEL-HDL Entry Considerations

Most design control is achieved through use of language elements such as dot extensions and ABEL-HDL operators. Further design control may be accomplished for ispLSI designs through the use of the special properties, described in "Designing for ispLSI Devices."

You should use detailed syntax wherever possible in an ABEL-HDL design, to make the design less ambiguous to the fitter and easier to link and merge with other modules if similar techniques are used everywhere. For example, consider the following source file fragment:

```
...
IN1..IN4, CLK  PIN ;
OUT_F          PIN ISTYPE 'REG_D';
OUT_J          PIN ISTYPE 'COM';
...
Equations
   OUT_F.CLK = CLK;
   OUT_F.D = IN1 & IN2 & IN3;
   OUT_J = OUT_F.Q & IN4;
...
```

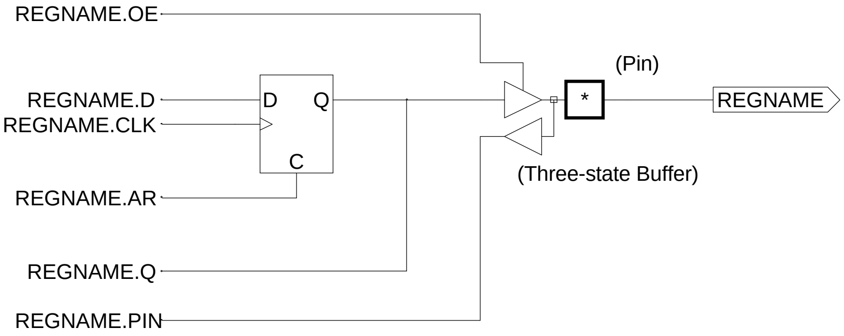
The CLK input feeds the register's .CLK input, the logic equation feeds the register's .D input, and the combinational output's equation uses the .Q feedback from the register. This is a common example of ABEL-HDL detailed syntax. Pin-to-pin syntax can be used, but may cause polarity discrepancies or other errors during design fitting.

Note: *The register's equation feeds the register's .D input, and the other equation refers to the .Q feedback from the register. This is the basic essential of ABEL-HDL "detailed" syntax. This is the recommended form to be used in ABEL-HDL designs targeted to ispLSI devices. Pin-to-pin syntax may also be used, but polarity issues may occur during linking due to ambiguous feedback definition and signal attribution.*

Note: *Istypes Keep and Collapse are used to keep nodes intact during the collapsing by the linker/optimizer.*

Use Detailed Syntax to Access device Features

Using dot extensions with ABEL-HDL detailed syntax to access specific features and signal paths in a Lattice device. Typical dot extensions are illustrated below.



These dot extensions allow you access to any desired point of a register or other hard feature available within the Lattice device. Defining more specific features for routing or particular ispLSI architecture aspects involve the use of special properties described in "Accessing ispLSI Features."

ABEL-HDL Property Statements for ispLSI design

ABEL-HDL property statements attach to pins or nodes in an ABEL-HDL design, and are only supported in a top-level ABEL-HDL module. Properties statements used in lower-level ABEL-HDL modules are ignored.

The PLSI properties have following syntax:

```
PLSI PROPERTY 'property_name node_name options';
```

where

property_name The name of the property.

node_name The name of the node to which the property is to be attached.

options Property-specific options you can set.

Optimize Locally In Schematics

You can reduce the work the linker and fitter have to do (and thus minimize processing time) by setting the attribute "optimize locally" to "Y" in the schematic. This allows the symbol BLIF description to be read into the individual schematic's PLA file and optimized along with the rest of the logic in that source before final linking and optimization of the entire design. With a large design, this could reduce processing time, and can provide a more optimal fit.

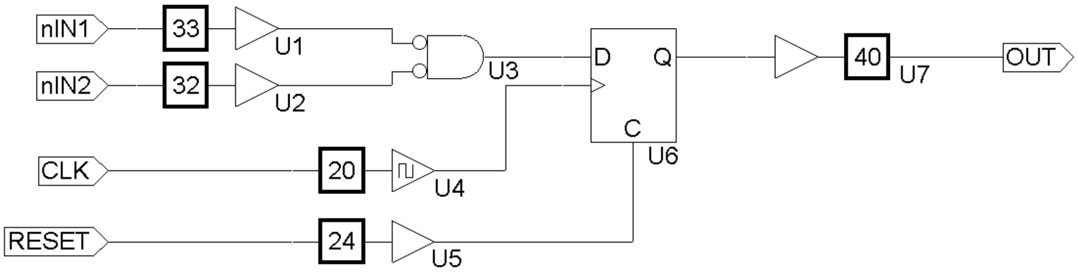
Assigning Pins

In Schematics

To assign a pin in a schematic:

1. Set the I/O symbol's SynarioPin attribute to the desired pin number (replacing the asterisk (*)).
2. Set the fit property **Ignore Fixed Pins** to **False**.

Be sure to use appropriate I/O symbols on all pins. Assigning pins this way allows you to fix pin numbers to specific pin names. See the figure below.



Note the use of input buffers, output buffers, and the clock buffer. The pins are assigned using the schematic attribute SynarioPin.

Inputs are shown as nets given the input pin's desired name (and Input I/O markers) tied to a G_INPUT symbol's PAD connection. The BUFFER connection (output) of the G_INPUT symbol is tied to the logic of the schematic. The output of the schematic's logic is tied to a G_OUTPUT symbol, and a net is tied to that symbol's PAD and is given the desired output pin name. Note the use of the clock buffer G_CLKBUF as well. The pins are assigned using the schematic attribute SynarioPin described above.

The schematic compiler automatically creates a BLIF node for each net that is encountered. Nodes which do not have PLSI properties attached may be reduced out if necessary by the design process flow, if set up to remove and collapse excess nodes. Nodes which have PLSI properties attached to them (such as CriticalPath, AsynchronousPath, and/or PRESERVE) are automatically flagged as "keep" in order to remain intact through all Project Navigator processes.

In ABEL-HDL

To assign pins in ABEL, use the PIN keyword. See your *ABEL-HDL Reference* for more information.

Test Vectors In Schematics

The Project Navigator allows you to place ABEL-HDL test vectors in a separate file to be compiled into a .TMV file that the ISP Synario Fitter later merges into the JEDEC file.

To create project-level ABEL-HDL test vectors:

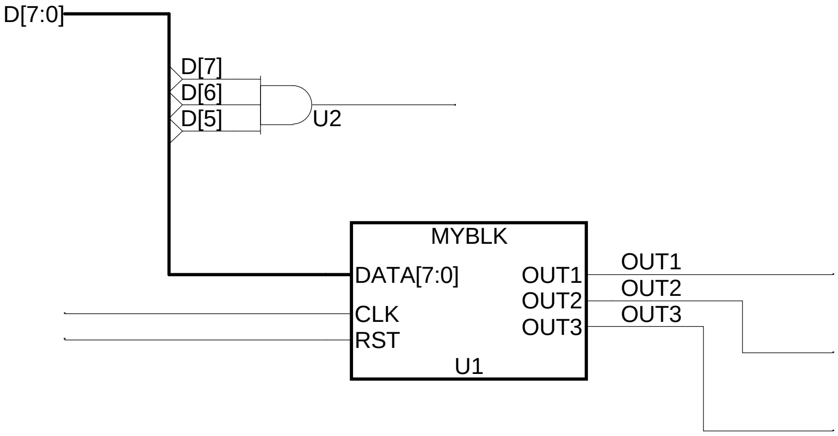
1. Create an ASCII document by using the Synario menu option SOURCE/NEW... and selecting ABEL Test Vectors. Enter your test vectors.

   Be sure to associate the .ABV file with the device, rather than any of the sources in the project. It should appear above the device type in the Synario source list.

2. During design fitting, the .ABV file is automatically compiled and merged into the output JEDEC file.

Assigning Buses In Schematics

Buses are drawn as wires, but are given compound names using "bus" format. An example of a bus name is DATA[7:0]. When a net is given a bus name, it becomes thicker to indicate that it is a bus. The bus may be connected to a symbol with a bus pin, or may be split into its individual components through the user of bus taps. Each bus tap must be given a name, in order for the schematic netlister to know which tap is to connect to which bus element. For example, elements of the bus example above would be DATA[7], DATA[6], DATA[5], ... DATA[0]. See the figure below.



In this schematic, the bus D[7:0] is connected to the bus pin DATA[7:0]. The order of connection is based upon the positional order of the bus names (from right to left). In this case, D[7] connects to DATA[7], D[6] connects to DATA[6], etc. Bus taps are also used to connect D[7:5] to an AND gate. Each bus tap must be given the desired net name in order for a connection to be made.

Instantiating an ABEL-HDL Module

In Schematics

You can instantiate an ABEL-HDL module in a schematic the same way as any other schematic symbol.

To instantiate an ABEL-HDL module in a schematic:

Use the "Create New Block Symbol" selection from the "Add" menu to create the ABEL-HDL symbol containing the module's name, and input and output pin names.

Note: *Bus names can be used in the symbol, but will connect to non-bus names in the ABEL-HDL module, using the name followed by the bus element index enclosed in underscores. For example, if the function MYBLK is an ABEL-HDL module, then its pin statements would look like this:*

```
DATA_7_,DATA_6_,DATA_5_,DATA_4_ pin; "Connects to bus signals DATA[7:4]
DATA_3_,DATA_2_,DATA_1_,DATA_0_ pin; "Connects to bus signals DATA[3:0]
CLK, RST          pin;
OUT1, OUT2, OUT3  pin istype 'reg_d';
```

The schematic compiler will create the bus connections to the ABEL-HDL module provided the compile property "Suppress Bus Naming" is set to Y. Otherwise, buses are netlisted using the square brackets, and may fail to link with other modules in the project.

In ABEL-HDL

You instantiate lower-level ABEL-HDL modules using the Functional_Block and Interface keywords. See your *ABEL-HDL Reference* for more information.

3. Designing for ispLSI devices

Techniques for Design

Preserving Nodes

In order for properties to "stick" to a node, that node must be kept throughout processing. It is possible to set up the processing flow (between source and final fitting run) to minimize logic and remove extra, redundant nodes in the design. This keeps a design from using too many device resources. However, those nodes that require a property to be kept (such as "start critical path" and "end critical path") must not be minimized away.

Also, you must preserve any internal net with the following ABEL-HDL properties: SAP/EAP, SCP/ECP, or SNP/ENP. You can only preserve internal nodes, not on I/O pins. Note that preserved nodes should also be assigned ISTYPE 'KEEP' in order to retain them from collapse by the ABEL-HDL linker and optimizer.

Unassigned nodes are collapsed if the collapse level is ALL.

In Schematics

In schematics, redundant nodes are kept if the symbols in the net path are not optimized locally.

To preserve a node:

1. For each symbol on a net path with nodes you want to keep, set the **Optimize_locally** attribute to N.
2. Set the net's PRESERVE attribute to Y.

Note: *The attribute will appear attached to the net as Y.*

In ABEL-HDL

To preserve a node:

Use the pin and node attribute Istype 'keep' to preserve nodes during processing by the linker / optimizer.

To collapse a node:

1. Use Istype 'collapse' on the node declaration.
2. Set the collapse level to ALL or EXPLICIT in the module reduction or design fitting properties.

Example:

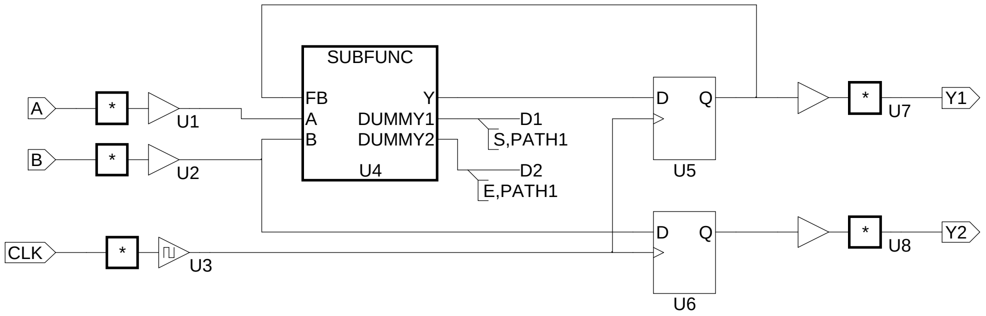
```
...
"Nodes
  N1   node;

PLSI PROPERTY 'PRESERVE N1';
...
```

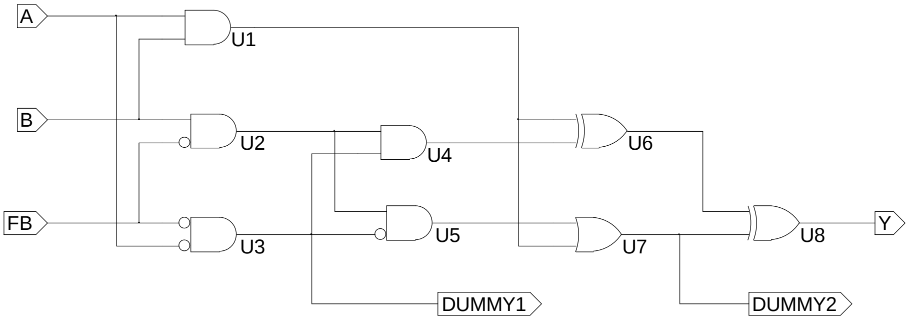
Assigning Properties to Nodes

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 .TT2 to be passed on to the ISP Synario fitter. Care must be taken, however, when using properties in lower level modules that are instanced 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 instanced more than once, the .TT2 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 schematics shown below.



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.



The second schematic shows where the paths for no-minimization are taken from. Note that when the design is processed by the ISP Synario Fitter, the unused nets (DUMMY1 and DUMMY2) will be removed since they don't 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.

Attribute Processing

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

Attribute	Purpose	Syntax	Applies To
Clock_type	Assigns ispLSI clock type	CLK0, CLK1, CLK2, IOCLK0, IOCLK1, FASTCLK, SLOWCLK	G_CLKBUF symbol
Critical	Marks output as critical	Y or N, T or F, 1 or 0	Any output symbol
Register_Type	Assigns register type	GLB or IOC	Any Register
AsynchPath	Marks begin or end of Asynchronous path	S,*identifier* or E,*identifier*	Net
CriticalPath	Marks begin or end of Critical path	S,*identifier* or E,*identifier*	Net
NoMinimizePath	Marks begin or end of No Minimize path	S,*identifier* or E,*identifier*	Net
Preserve	Marks net to be preserved	Y or N, T or F, 1 or 0	Net
Protect	Marks primitive (symbol) to be preserved	Y or N, T or F, 1 or 0	Any symbol
Group	Marks net to be included in a group	*identifier*	Net
SlowSlew	Marks output to use a slow slew rate	Y or N, T or F, 1 or 0	Any output symbol
Pullup	Marks output to have a pullup attached	Y or N, T or F, 1 or 0	Any output symbol
Use_XOR	Tags this gate to be hard XOR in GLB	Y or N, T or F, 1 or 0	G_XOR symbol

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	*number*	I/O symbols
OptimizeLocally	Tags symbol to be optimized prior to final merge.	Y or N, T or F, 1 or 0	All symbols
SynarioSrcType	Marks a symbol as representative of a module not in the current project..	External (or blank)	Block symbols

Accessing ispLSI features

Setting a Critical Path

Use Critical Paths to improve performance through the logic path with speed requirements through the GLB by using the four p-term bypass, if possible, and by minimizing the number of GLB levels for the specified path. Critical paths restrict routing and can decrease resource utilization. The ending attribute or statement can terminate multiple critical paths.

In Schematics

To set a critical path:

1. Mark the net for the beginning of the critical path with the CriticalPath attribute, with the text

   ```
   S,identifier
   ```

   where *identifier* is a unique name for the critical path.

2. Mark the end of the critical path with the same attribute, CriticalPath, using the text

   ```
   E,identifier [,identifier2,identifier3...]
   ```

***Note**: A critical path can have more than one starting point. In this case, their path names are added to the first path name, separated by commas. Note that the attribute will appear attached to the net with the text of the attribute's value, such as "S,CPATH1."*

In ABEL-HDL

To set a critical path:

1. Mark the beginning of a critical path by placing an "SCP" property on the desired node, along with a unique path name.
2. Mark the end of a critical path by placing an "ECP" property on the desired node, along with the same path name.

Example:

```
...
"Nodes
   N1, N2          node istype 'com,keep';  " preserves nodes in
ABEL reduction

"Properties
PLSI PROPERTY 'SCP N1 PATH_1';  "marks beginning of critical path
PLSI PROPERTY 'ECP N2 PATH_1';  "marks end of critical path
PLSI PROPERTY 'PRESERVE N1';  " preserves nodes in fitter
reduction
PLSI PROPERTY 'PRESERVE N2';
...
```

This syntax assures that the ISP Synario Fitter uses the fastest available route from N1 to N2, keeping the number of passes back through the GLB as low as possible.

Setting a No-minimize Path

No-minimize paths restrict the optimization of logic for the specified path. One ending attribute or statement can terminate multiple no-minimize paths.

Note: *A no-minimize path can have more than one starting point.*

In Schematics

To set a no-minimize path:

1. Mark the beginning of the no-minimize path with the "NoMinimizePath" attribute, using the text

   `S,`*`identifier`* `[,`*`identifier2`*`,`*`identifier3`*`...]`
2. Mark the end of a no-minimize path using the same attribute, with the text

   `E,`*`identifier`*

In ABEL-HDL

To set a no-minimize path:

1. Mark the beginning of a no-minimize path by placing an "SNP" property on the desired nodes, along with a unique path name.
2. Mark the end of a no-minimize path by placing an "ENP" property on the desired node, along with the same path name.

Example:

```
"Nodes
N1, N2       node istype 'com,keep';

"Properties
PLSI PROPERTY 'SNP N1 PATH_1';
PLSI PROPERTY 'ENP N2 PATH_1';
PLSI PROPERTY 'PRESERVE N1';
PLSI PROPERTY 'PRESERVE N2';
...
```

This causes the ISP Synario Fitter to leave the logic in between N1 and N2 unminimized.

Setting an Asynchronous Path

Defining an Asynchronous Path prevents the fitter from duplicating GLB outputs for logic along the specified path which may cause a race condition. An Asynchronous Path can have more than one starting point.

In Schematics

To set an asynchronous path:

1. Mark the beginning of the Asynchronous path with the "AsynchPath" attribute, using the text

   `S,`*`identifier`*`[,`*`identifier2`*`,`*`identifier3`*`...]`
2. Mark the end of an Asynchronous Path with the same attribute, with the text

   `E,`*`identifier`*

   Note that the attribute will appear attached to the net with the text of the attribute's value, such as "S,APATH1."

In ABEL-HDL

To set an asynchronous path:

1. Mark the beginning of an asynchronous path by placing an "SAP" property on the desired node, and assigning a path name.
2. Mark the end of the asynchronous path by placing an "EAP" property on the desired node and assigning the same path name.

Example:

```
"Nodes
N1 node istype 'reg,keep';
N2, N3, N4   node istype 'com,keep';

"Properties
PLSI PROPERTY 'SAP N1 PATH_1';
PLSI PROPERTY 'EAP N2 PATH_1';
PLSI PROPERTY 'PRESERVE N1';
PLSI PROPERTY 'PRESERVE N2';

Equations
  N2 = N1 & IN1 & IN2;
  N3 = N1 & FB1 & IN2;
  N4 = N1 & FB2 & IN1;
...
```

This code assures that the logic fed by N1 will not be duplicated by the fitter for better routing, thereby avoiding a possible race condition.

Bypassing the Output Routing Pool

You can bypass the Output Routing pool to reduce logic to pin delay using the CRITICAL attribute in schematics and the CRIT property in ABEL-HDL on the desired pins. See the pDS+ Fitter User Manual for more information.

In Schematics

Use the "CRITICAL" attribute on the desired output pin. This attribute is available on the G_OUTPUT, G_BIDIR, and G_TRI symbols.

In ABEL-HDL

Use the CRIT property to instruct the software to use Output Routing Pool (ORP) bypass. You can place this property only on output pins. Use this property for GLB outputs that require the least possible delay.

Example:

```
...
"Outputs
   OUT1, OUT2, OUT3 pin  istype 'com';

"Properties
PLSI PROPERTY 'CRIT OUT2, OUT3';
...
```

This assigns the outputs "OUT2" and "OUT3" as "CRITICAL", telling the ISP Synario Fitter to allow those signals to bypass the output routing pool.

Attaching pull ups to individual output pins

You can attach pull ups to individual output pins by using the PullUp attribute in schematics and the PULLUP property in ABEL-HDL. See the PDS+ Fitter User Manual for more information.

In Schematics

Use the "PullUp" attribute on the desired output pin. This attribute is available on the G_OUTPUT, G_BIDIR, and G_TRI symbols.

In ABEL-HDL

Use the PULLUP property to instruct the software to attach a weak pullup to the desired output pin. You can place this property only on output pins.

Example:

```
...
"Outputs
   OUT1, OUT2, OUT3 pin  istype 'com';
"Properties
PLSI PROPERTY ‘PULLUP OUT2’;
PLSI PROPERTY ‘PULLUP OUT3’;
...
```

This assigns the outputs "OUT2" and "OUT3" to have Pull ups, telling the ISP Synario Fitter to attach weak pull up resistors to those outputs during fitting.

Protecting Primitives

You can protect combinational logic paths from being restructured by using the Protect attribute in schematics and the PROTECT property in ABEL-HDL. This has a similar effect to using the No-Minimize Path attribute or property. See the PDS+ Fitter User Manual for more information.

In Schematics

Use the Protect attribute on the desired symbol, setting the value to Y. This attribute is available on all symbols.

In ABEL-HDL

Use the PROTECT property to instruct the software to keep the symbol desired from being minimized away during processing by the ISP Synario Fitter. The property is attached to the node fed by the primitive desired for protection.

Example:

```
...
"Nodes
   N1, N2, N3      node istype 'com';
"Properties
PLSI PROPERTY ‘PROTECT N2’;
PLSI PROPERTY ‘PROTECT N3’;
...
```

This assigns the nodes "N2" and "N3” as being protected from reduction.

Grouping Signals into a GLB

Signals may be grouped into a common GLB, to aid the fitter in determining a fit. This is commonly used in multiple-bit-wide logic designs, such as counters or shift registers, in which each bit is dependent upon other bits within the group. Groups are formed using the Group attribute in schematics and the GROUP property in ABEL-HDL. See the PDS+ Fitter User Manual for more information.

In Schematics

Use the "Group" attribute on the desired nets, setting the value to an identifier naming the group. The name is arbitrary. Any net using the same group name will be considered part of that group.

In ABEL-HDL

Use the GROUP property to instruct the software to group signals together. It may be attached to nodes our outputs.

Example:

```
...
"Outputs
   OUT1, OUT2, OUT3 pin  istype 'com';

"Properties
PLSI PROPERTY 'GROUP OUT1 MYGRP';
PLSI PROPERTY ‘GROUP OUT2 MYGRP’;
PLSI PROPERTY ‘GROUP OUT3 MYGRP’;
...
```

This code assigns the outputs to be grouped into a group named “MYGRP” during ISP Synario Fitter processing.

Forcing a Slow Slew Rate Output

You can cause an output to have a slow slew rate by using the SlowSlew attribute in schematics and the SLOWSLEW property in ABEL-HDL. See the PDS+ Fitter User Manual for more information.

In Schematics

Use the "Slow Slew" attribute on the desired output pin. This attribute is available on the G_OUTPUT, G_BIDIR, and G_TRI symbols.

In ABEL-HDL

Use the SLOWSLEW property to instruct the software to give the associated output a slow slew rate. You can place this property only on output pins. This feature is not available on all devices. Check the Lattice Semiconductor Databook for availability.

Example:

```
...
"Outputs
   OUT1, OUT2, OUT3 pin  istype 'com';

"Properties
PLSI PROPERTY 'SLOWSLEW OUT1';
...
```

This code assigns the output "OUT1" as having a slow slew rate.

Setting the Use of a Hard XOR Gate

This feature is available only on the G_XOR symbol. This option forces the ISP Synario Fitter to use the symbol as a hard XOR gate, rather than trying to construct one from logic. Note that you must also have the XOR output node assigned as PRESERVE, and that the source must not be optimized.

In Schematics

Set the attribute USE_XOR to Y.

In ABEL-HDL

Assign the LXOR2 property to the node fed by the desired XOR function.

Example:

```
"Nodes
  N1     node istype 'com';
"Properties
  PLSI PROPERTY 'LXOR2 N1';
Equations
  N1 = IN1 $ IN2;
```

Placing a Register in the GLB or IOC

You can only use place a register in the IOC if the register's D-input is fed directly from an input (G_INPUT) or bidir (G_BIDIR) pin.

In Schematics

Set the register symbol's REGISTER_TYPE attribute to either IOC or GLB as desired.

In ABEL-HDL

Assign a register to the IOC or GLB using the REGTYPE property:

```
PLSI PROPERTY 'REGTYPE pin_name IOC';
```

or

```
PLSI PROPERTY 'REGTYPE pin_name GLB';
```

Setting the Clock Type

You can assign specified clock nets to the clock inputs of GLB and IOC registers. If you do not assign a CLK property, the software automatically determines whether nets should use the dedicated clock routing or the slower product term (signal) clock routing.

In Schematics

To set the clock type:

1. To assign the clock signal to a particular type, use a G_CLKBUF symbol on the Clock net.
2. Assign the symbol's CLOCK-TYPE attribute to the desired type, either CLK0, CLK1, CLK2, IOCLK0, IOCLK1, FASTCLK or SLOWCLK.

See the PDS+ Fitter User Manual for more information.

In ABEL-HDL

To set the clock type:

Use the CLK property to assign specified clock nets to the clock inputs of GLB and IOC registers. You can set the clock type to CLK0, CLK1, CLK2, IOCLK0, IOCLK1, FASTCLK and SLOWCLK.

Example:

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

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

This assigns the input "CLKA" to the routing resource "CLK1."

See the PDS+ Fitter User Manual for more information.

Reserving the ISP pins

In-System Programming specifies how you wish to use the ISP pins. If you do not use the in-system programming capability, the four dedicated ISP pins (SCLK, SDI, SDO and MODE) are available as general input pins for use by the router. This option is ignored if a non-isp device is targeted.

In Schematics

In the Project Navigator, set the Compile Schematic property **In-System Programming** to True.

In ABEL-HDL

To use the in-system programming capability, set the ISP property to ON.

Example:

```
PLSI PROPERTY 'ISP ON';
```

Using the Y2 Clock Input for Routing (ispLSI 1016 and ispLSI 2032 only)

This allows ISP Synario Fitter to use the Y2 clock input for routing, which increases resource utilization. This option is valid only for the ispLSI 1016 and ispLSI 2032, and is ignored if you choose any other device.

In Schematics

In the Project Navigator, set the Compile Schematic property **In-System Programming Except Y2** to True.

In ABEL-HDL

Set the ISP_EXCEPT_Y2 property to ON.

Example:

```
PLSI PROPERTY 'ISP_EXCEPT_Y2 ON';
```

Assigning Pull ups on all I/O pins

The Pullup property specifies how you wish to use the I/O pin on pull up resistors. If set to ON, active pull ups are placed on all I/O pins. Set to OFF, pull ups are placed only on the unused I/O pins. This option has no effect on routing or resource utilization. Default for the Pullup property is ON.

In Schematics

In the Project Navigator, set the Compile Schematic property **Pullups** to True.

In ABEL-HDL

Set the Pullup property ON:

```
PLSI property 'PULLUP ON';
```

Enabling the Security Fuse

In Schematics

In the Project Navigator, set the Compile Schematic property **Enable Security Fuse** to True.

In ABEL-HDL

Set the SECURITY property to ON, and a "1" will be placed in the security fuse location of the created JEDEC file. Note that it is up to the programmer to determine whether or not the device security bit actually gets programmed by this bit.

Property Y1_AS_RESET

This property determines how the Y1/RESET is used on the pLSI/ispLSI 1016 and pLSI/ispLSI 2032. If it is set to ON, the Y1/RESET pin is used as the global reset input. Otherwise, the Y1/RESET pin is the Y1 clock input. This option applies only to the pLSI and ispLSI 1016 and 2032 devices. The default is ON.

In Schematics

In the Project Navigator, set the **Compile Schematic** property **Y1 as Reset** to True.

In ABEL-HDL

Include a property statement setting Y1 as Reset to ON.

Example:

```
PLSI PROPERTY 'Y1_AS_RESET ON';
```

4. Controlling ispLSI Processing and Fitting

Note: *The ISP Synario Fitter automatically removes any unconnected inputs from the circuit, alerting you with warnings about not having any fanout. Unconnected outputs are flagged as errors in the fitter. The error will appear in the Processing Log.*

Compiling and Reducing Schematics

Schematic modules are compiled into BLIF files, which are then optimized before linking with the rest of the design. Symbols in the schematic with the **Optimize_Locally** attribute set to Y are linked into the schematic BLIF for optimization.

Note: *If all symbols in a schematic have **Optimize Locally** set to **False**, then the resulting BLIF after optimization contains no logic, and viewing the reduced equations for the schematic will show no equations at all.*

The schematic compile process has a properties to control the schematic netlist and pLSI design. These properties apply to the top-level schematic only. These properties are shown under the feature they control in Chapter 2.

Optimizing Processing

For Schematics

Use schematic properties and attributes coupled with optimization and fitting properties to control how a design fits. Break up designs into manageable chunks of logic with cells of no more than four to six product terms each. This strategy will make it easier for the fitter to find a fit for a given equation. Large counters, for example, should be broken up into groups of outputs with look-ahead carry summing nodes between counter stages.

For ABEL-HDL Modules

Do not use "interface" statements in lower-level ABEL-HDL modules referenced from schematics. The netlister and compiler may re-order the pin numbers, causing problems in linking the upper-level schematic with the instantiated ABEL-HDL module.

Fitting Strategies

Setting ispLSI Features in Schematics

You can access design-level ispLSI properties described in the ISP Synario Fitter. In schematics, you use attributes; in ABEL-HDL, you use dot extensions or property statements. See Chapter 2.

Reducing Linking and Fitting Times

You can reduce the work the linker and fitter have to do (and possibly minimize processing time) by setting the Optimize Locally attribute to Y in the schematic. This setting allows the symbol BLIF description to be read into the individual schematic PLA file and optimized with the rest of the logic in that source before final linking and optimization of the entire design. With a large design, this can reduce the processing time and provide a more optimal fit.

Pin Assignment

To assign a pin in a schematic, edit the I/O symbol's **SynarioPin** attribute, replacing the "*" with the desired pin number. The fit property **Ignore Fixed Pins** must be set to "False" to allow the fitter to pay attention to the assigned pins.

Fit Properties

The fitter properties are broken up into the pre-fit reduction properties list, and the Fitter Strategies list. The reduction properties apply to the post-link design and determine how it is optimized into the final .TT2 file for processing by the fitter.

The Fitter Strategies list determines the processing options that are sent to the fitter when it is run on the project. The list is as follows:

Design Normalization

For GAL designs, may be set to TRUE or False. For ispLSI designs, may be set to Basic Normalization, Synthesize Synchronous Reset / Preset, or Off. This tells the pre-fit flip flop transformation process whether to perform standard flip flop transformation and feedback normalization, or to merge Synchronous Reset or Preset product terms into the Register's D input, or to pass the design through unchanged.

Extended Route

For ispLSI designs. By default, this does not affect the fitter command line. May be set to "No" explicitly, forcing the fitter to stop and query you to allow it to continue if it senses that the processing may take a long time.

Fit Strategy

For ispLSI designs. May be left to default, or set to Area or Delay. This tells the fitter what is most important to optimize toward during the fitting process.

Fit Effort

For ispLSI designs. May be left to default, or set to 1, 2 or 3. Allows the fitter greater or lesser time and memory for processing. A fit effort of 3 takes more time and memory, but usually leads to better fit results.

Maximum GLB Inputs

For ispLSI designs. A numeric value specifying to the fitter the maximum number of GLB inputs the fitter is allowed to use for each GLB.

Maximum GLB Outputs

For ispLSI designs. A numeric value specifying to the fitter the maximum number of GLB outputs the fitter is allowed to use for each GLB.

Ignore Fixed Pins

For ispLSI designs. Tells the fitter to ignore the pin assignments made in the top-level source. It is set to FALSE by default.

Use Schematic properties and attributes coupled with optimization and fitting properties to control how a design fits. It is best to break up designs into manageable pieces of logic, trying for cells with no more than four to six product terms each, so the fitter can more easily find a fit for a given equation. Large counters, for example, should be broken up into groups of outputs with look-ahead carry summing nodes between counter stages.

The fitter automatically removes any unconnected inputs from the circuit, alerting you with warnings about not having any fanout. Unconnected outputs will be flagged as errors in the pLSI Processing Log.

Use Global Reset

For ispLSI designs. May be left to default, or set explicitly to Yes. This will make the global reset pin available for use by the fitter. If this is set to Yes, and all registers and latches within the design use a common, direct reset, then that reset will be moved by the fitter on to the global reset pin. Note that the global reset pin is active low.

Use Case Sensitivity

For ispLSI designs. May be left to default, or set explicitly to Yes. This will cause the fitter to be case-sensitive when it considers pin, node and net names in the input .TT2 file.

Pin File Name

Allows you to specify the name of a separate Pin Assignment file for the ISP Synario fitter to read during processing. Information on the format of the .PIN file is in your PDS+ Fitter User Manual.

Parameter File Name

This property specifies the name of a parameter file (with the extension of .PAR implied). The parameter file is read by the ISP Synario Fitter during processing.

Note: *Since fitter strategies specified in the Project Navigator override parameters set in the parameter file, set all other properties to their default values if you are using the parameter file. See the PDS+ Fitter User Manual for more information on parameter files.*

5. Simulating Your Design

You can simulate your design using the Equations or JEDEC simulators. The Equations simulator simulates your design using the linked and optimized equations that are fed to the device fitter. The JEDEC simulator simulates your GAL designs JEDEC file as a final indication that the information to be programmed into your GAL is correct.

Simulating Pre-fit Equations

You can simulate the linked and optimized form of the project by using the ABEL Test Vectors process **Simulate Reduced Logic.** This process simulates the Equations that are given as input to the ISP Synario fitter. Refer to the *Equation and JEDEC Simulators User Manual* for more information.

To simulate the reduced equations:

1. Create ABEL-HDL test vectors.

   You can put the test vectors either in a top-level ABEL-HDL source or in a separate ABEL-HDL test vector format file called an ".ABV" file. The .ABV file is considered a "text document" and is kept above the device level in the Sources window.

2. Run the **Simulate Reduced Logic** process. 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.

Simulation Log

The Equation and JEDEC Simulators log errors and status information in one of three files, depending on the nature of the information:

synario.log or abel.log	Logs processing errors
err.err	Logs logic errors
***.st2**	Logs simulation status information

If a functional simulation error occurs, it is recorded in, and may be viewed by double-clicking the Simulation Results report. Functional simulation errors do not automatically cause the Report Viewer to appear.

Simulating a JEDEC file

For GAL devices, you can simulate the JEDEC fuse file by using the ABEL Test Vector process **Simulate JEDEC File.** This process simulates the JEDEC fuse map as compared to an internal model of the target device itself. Refer to the *Equation and JEDEC Simulators User Manual* for more information.

To simulate the JEDEC file:

1. Create ABEL-HDL test vectors.

   You can put the test vectors either in a top-level ABEL-HDL source or in a separate ABEL-HDL test vector format file called an ".ABV" file. The .ABV file is considered a "text document" and is kept above the device level in the Sources window.

2. Run the **Simulate JEDEC File** process. 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.

Index