ISP Synario System
User Manual

June 1995

Data I/O Corporation
10525 Willows Road N.E., P.O. Box 97046
Redmond, Washington 98073-9746 USA
(206) 881-6444

Table of Contents

Before You Begin

This manual covers how to:

- Create designs for Lattice pLSI, ispLSI and GAL devices
- Fit your design
- Simulate your design

This manual assumes you are familiar with the basics of operating the Project Navigator, covered in the *Getting Started* manual.

Hardware Requirements

For optimum performance of the Lattice Fitter, you should have the following hardware:

- 386 microprocessor or better
- 16 MB or more of system memory
- 20 MB of free disk space in which to install the software

Chapter 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 fuse map creation of these devices.

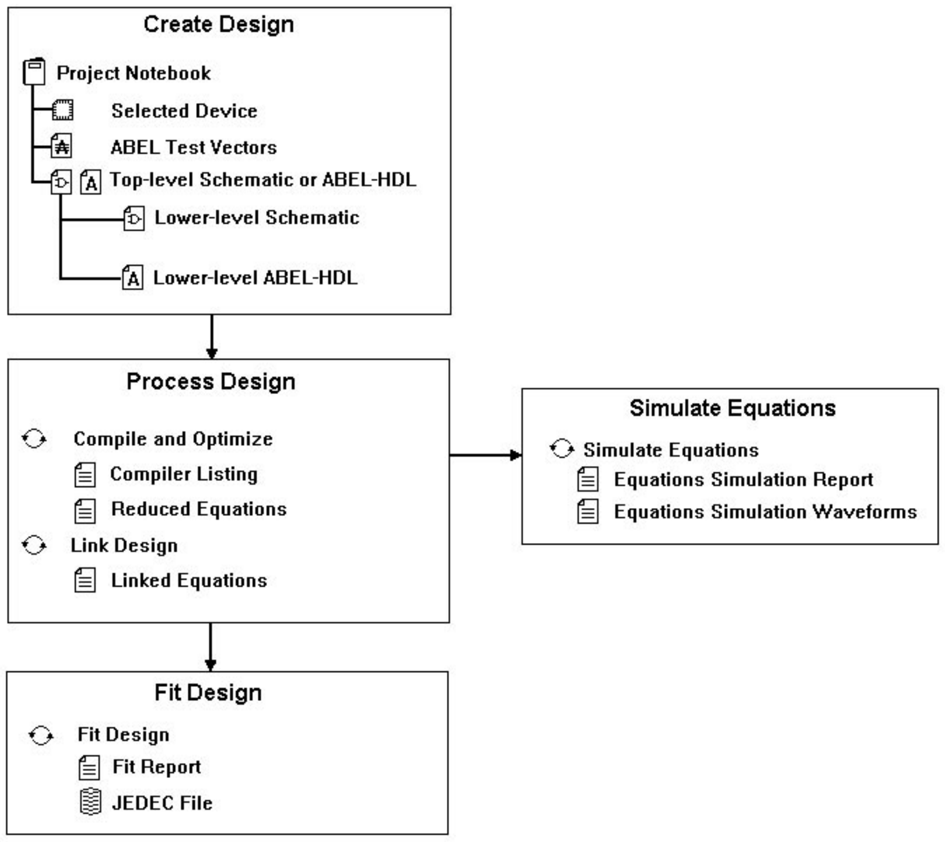
For the purposes of clarity, both pLSI and ispLSI devices will be referred to as "pLSI" in this and following chapters.

Features

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

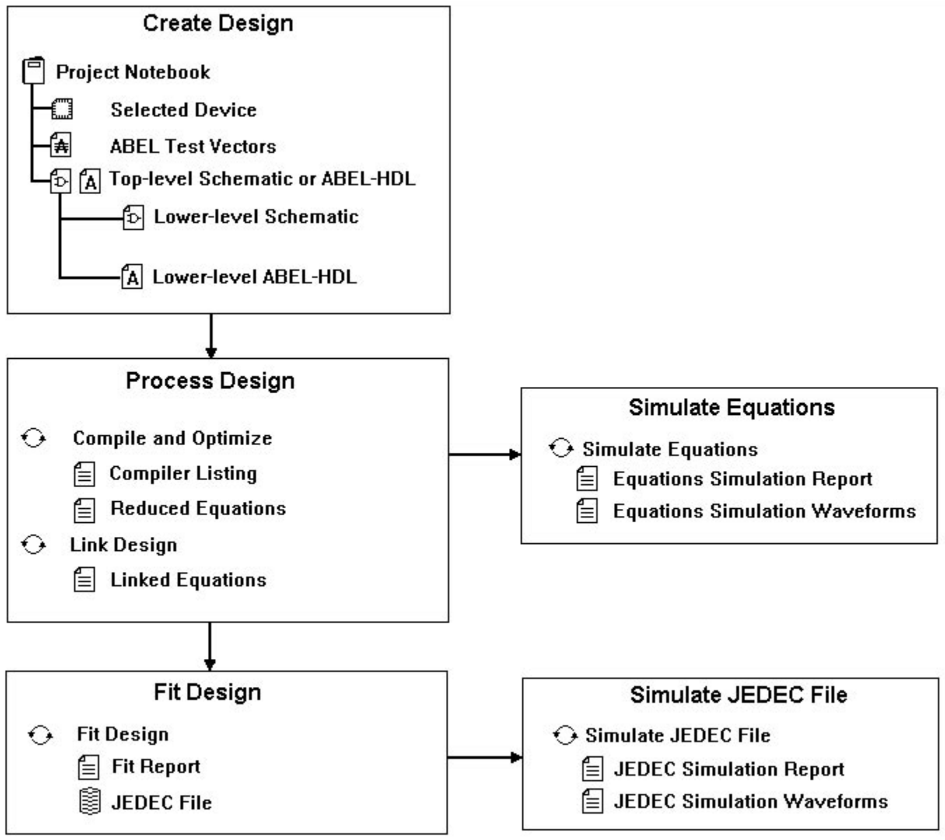
pLSI Flow

The pLSI 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 2.2 supports the following devices:

is / pLSI 1016
is / pLSI 1024
is / pLSI 2032
is / pLSI 2064
is / pLSI 2096
GAL 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 pLSI device.

Sources and Hierarchy

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

Chapter 2: Designing With Synario

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.

Design Tips

- It is generally better to keep logic broken into small pieces while entering, when designing for pLSI 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.*

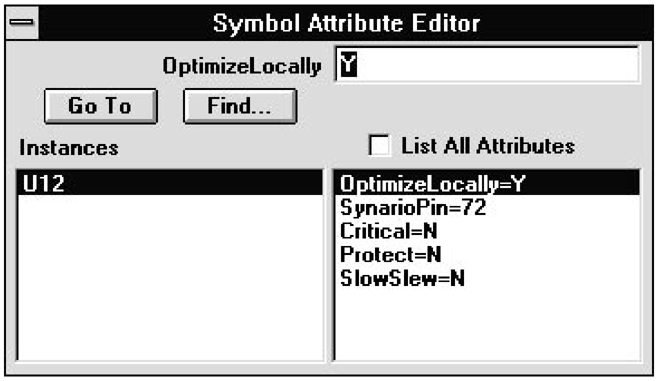
Schematic Attributes

You use attributes on symbols and nets in a schematic to control certain aspects of pLSI 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

U12

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

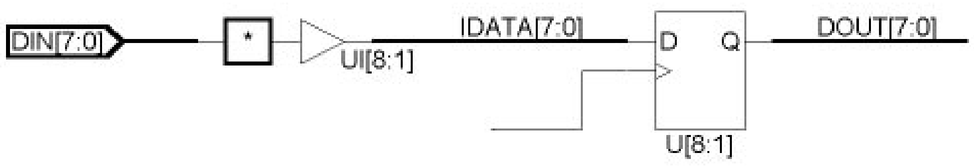
3. Select the attribute you wish to change, and enter the desired attribute value in the text entry box at the top of the attribute editor.

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

You can access any applicable design-level pLSI properties described in the Lattice pDS+ 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 the Synario Lattice Starter Kit 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 pLSI designs through the use of the special properties, described in "Designing for pLSI and 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;
...
```

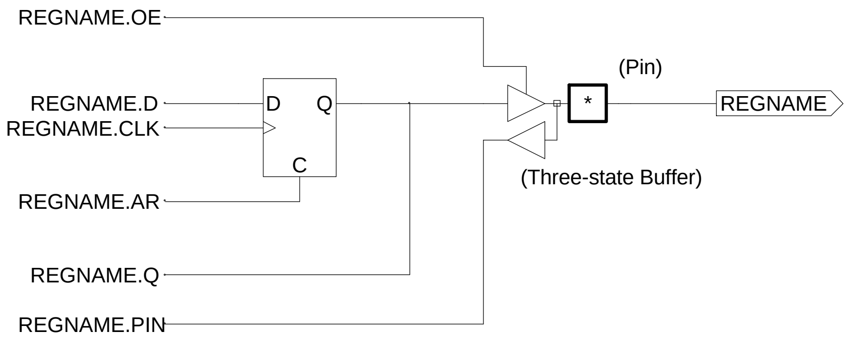
The CLK input feeds the register's .CLK input, the logic equation feeds the register's .D input, and the combinational output's equation uses the .Q feedback from the register. This is a common example of ABEL-HDL detailed syntax. Pin-to-pin syntax can be used, but may cause polarity discrepancies or 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 pLSI 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 pLSI architecture aspects involve the use of special properties described in "Accessing pLSI Features."

ABEL-HDL Property Statements for pLSI 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.

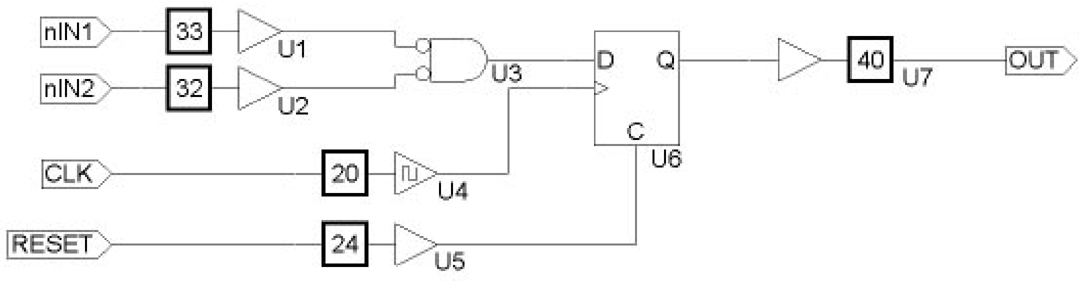
Assigning Pins

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 Lattice pDS+ fitter later merges into the JEDEC file.

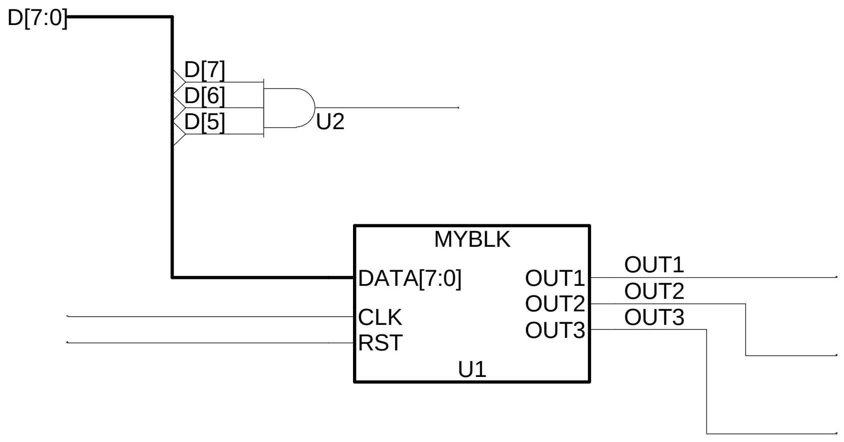
To create project-level ABEL-HDL test vectors:

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.

Chapter 3: Designing for pLSI and 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:

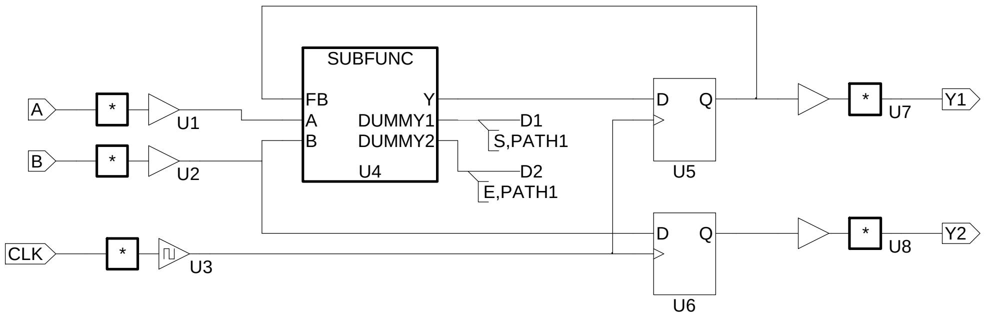
```
...
"Nodes
  N1   node;

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

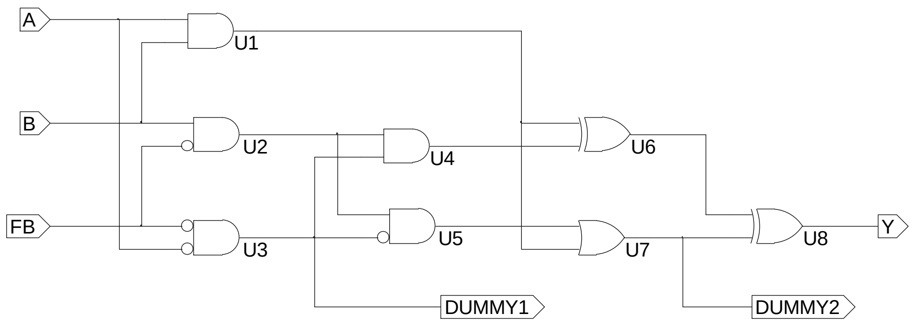
Assigning Properties to Nodes

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 pDS+ 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 pDS+ 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 Lattice pDS+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 Synario Lattice Starter Kit makes use of the schematic attributes by embedding them within the BLIF file compiled from the schematic as "pLSI PROPERTY" statements. In the Synario Lattice Starter Kit, the 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 pLSI 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	Not used by the Synario Lattice Starter Kit.	n/a	All symbols

Accessing pLSI features

Setting a Critical Path

Use Critical Paths to improve performance through the logic path with speed requirements through the GLB. 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 PATH_1 N1';  "marks beginning of critical path
PLSI PROPERTY 'ECP PATH_1 N2';  "marks end of critical path
PLSI PROPERTY 'PRESERVE N1';  " preserves nodes in fitter
reduction
PLSI PROPERTY 'PRESERVE N2';
...
```

This syntax assures that the Lattice pDS+ 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. 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 PATH_1 N1';
PLSI PROPERTY 'ENP PATH_1 N2';
PLSI PROPERTY 'PRESERVE N1';
PLSI PROPERTY 'PRESERVE N2';
...
```

This causes the Lattice pDS+ fitter to leave the logic in between N1 and N2 unminimized.

Setting an Asynchronous Path

Defining an Asynchronous Path prevents the partitioner from duplicating a GLB output 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 PATH_1 N1';
PLSI PROPERTY 'EAP PATH_1 N2';
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 DPM for better routing, thereby avoiding a possible race condition.

Bypassing the Output Routing Pool

You can bypass the Output Routing pool with two outputs from each GLB using the CRITICAL attribute in schematics and the CRIT property in ABEL-HDL. See the pDS+ fitter 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 bypass the Output Routing Pool. You can place this property only on output pins.

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 Lattice pDS+ fitter to allow those signals to bypass the output routing pool.

Attaching pullups to individual output pins

You can attach pullups to individual output pins by using the PullUp attribute in schematics and the PULLUP property in ABEL-HDL. See the pDS+ fitter 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 bypass the Output Routing Pool. 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 Pullups, telling the Lattice pDS+ fitter to attach weak pullup 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 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 pDS+. 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 pDS+ 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 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 Lattice pDS+ 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 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.

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 Lattice pDS+ 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 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 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 only)

This allows Lattice pDS+ fitter to use the Y2 clock input for routing, which increases resource utilization. This option is valid only for the ispLSI 1016 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 Pullups on all I/O pins

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

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 pin 35 is used on the pLSI/ispLSI 1016. If it is set to ON, pin 35 is used as the global reset input. Otherwise, pin 35 is the Y1 clock input. This option applies only to the pLSI and ispLSI 1016 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';
```

Chapter 4: Controlling pLSI Processing and Fitting

Note: *The pDS+ 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 Lattice pDS+ fitter. The error will appear in the pLSI 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 pLSI Features in Schematics

You can access design-level pLSI properties described in the Lattice pDS+ 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 Lattice 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 pDS+.

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

Design Normalization

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

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

Fit Strategy

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

Fit Effort

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

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

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

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 Lattice pDS+ 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

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

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.

Generate SIM file

May be left to default, or set explicitly to Yes. This will cause a <project>.SIM file to be created during processing.

Generate Verilog Timing Model

May be left to default, or set explicitly to Yes. This will cause a <project>.VLO file and <project>.SDF to be created during processing. These files are essential for Verilog Timing Simulation.

Pin File Name

Allows you to specify the name of a separate Pin Assignment file for pDS+ 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 Lattice pDS+ 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 manual for more information on parameter files.*

Chapter 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 Lattice pDS+ABEL 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