Lattice™ Semiconductor Corporation

pDS+ Fitter User Manual

Version 3.0

Technical Support Line: 1-800-LATTICE or (408) 428-6414
pDS1100-UM Rev 3.0

Copyright

Trademarks

Lattice Semiconductor Corporation
5555 NE Moore Ct.
Hillsboro, OR 97124
(503) 681-0118

April 1996

Limited Warranty

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

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

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

Phone: 1-800-LATTICE
Fax: (408) 944-8450
E-mail: applications@latticesemi.com

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

Limitations on Warranty

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

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

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

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

Acknowledgments

The pDS+ Fitter is based on a Sequential Interactive System (SIS) developed by the Computer-Aided Design (CAD) Research Group at the University of California, Berkeley with significant enhancements and additions by Lattice Semiconductor Corporation.

Table of Contents

Preface

The pDS+™ Fitter system is used to optimize, partition, place, and route logic designs for the Lattice Semiconductor in-system programmable Large Scale Integrated (ispLSI®) devices and programmable Large Scale Integrated (pLSI®) devices.

This user manual describes the capabilities and use of the pDS+ Fitter software. It is written for design engineers who understand system and logic design and the use of design automation software. This manual contains information to guide you through compilation and device programming. It is the primary learning guide to help you use the software to design with Lattice Semiconductor ispLSI and pLSI devices.

Some technical reference material is included in this user manual to provide you with background material. However, it is assumed that you are familiar with the Lattice Semiconductor device architecture. You will need to read the *Lattice Semiconductor ISP Encyclopedia* to fully understand all the features of the Lattice Semiconductor devices.

What Is In this User Manual

This user manual contains design examples and information on the following topics:

- Using Design Attributes to specify design constraints
- Using Compiler Control Options to specify design objectives
- Running the pDS+ Fitter
- Using the pDS+ Timing Analyzer
- Understanding log and report files
- Design rules and tips

Where to Look for Information

Chapter 1, Introduction – Provides an overview of the pDS+ Fitter and its design flow.

Chapter 2, Design Attributes – Describes how Design Attributes can be used to specify design constraints.

Chapter 3, Design Compilation and Control – Provides information on using Compiler Control Options to specify design objectives and direct the compiler during design compilation.

Chapter 4, Timing Analysis – Provides information on using the Timing Analyzer and gives examples of timing reports.

Chapter 5, Design Reports – Explains the different sections that make up the post-route and pre-route report files.

Appendix A, Design Rules and Tips – Describes device-dependent design rules and user tips to optimize designs for delay, area, or routability.

Documentation Conventions

This user manual follows the documentation conventions listed in the following table:

Convention	Definition and Usage
Italics	Italicized text represents variable input. For example: *design*.1 This means you must replace *design* with the file name you have used for all the files relevant to your design. Book titles and chapter sections also appear in italics. For example: *Lattice Semiconductor ISP Encyclopedia*
`Courier Font`	Monospaced (Courier) font indicates text that the system displays. For example: `Design:          cnt4`
`Bold Courier`	Bold Courier font indicates text you type in response to system prompts. For example: **`dpm -i`** ***`file_name`*** **`-s a -l`**
|...|	Vertical bars indicate options that are mutually exclusive; you can select only one. For example: `CLK=CLK0|CLK1|CLK2`
“Quotes”	Titles of chapters are shown in quotation marks. For example: See Chapter 2, “Design Attributes.”
✍ **NOTE**	Indicates a special note.
▲ **CAUTION**	**Indicates a situation that could cause loss of data or other problems.**
❖ **TIP**	Indicates a special hint that makes using the software easier.

Related Documentation

In addition to this user manual, the following documentation is useful when using the Lattice Semiconductor pDS+ Fitter:

- *Lattice Semiconductor ISP Encyclopedia*
- *ISP Synario System User Manual*
- *ISP Synario Starter Release Notes*
- *ISP Synario System Design Tutorial*
- *ISP Daisy Chain Download Reference Manual*

Third-Party Interface Manuals

The pDS+ Fitter is design to compile designs from many different design environments. The following is a list of manuals which Lattice Semiconductor provides as support to the various third-party interfaces.

- *Altera to Lattice Semiconductor Design Conversion Application Notes*
- *Cadence and pDS+ Design and Simulation Environment User Manual*
- *Data I/O and pDS+ Design Environment User Manual*
- *Exemplar and pDS+ Design and Simulation Environment User Manual*
- *ISDATA and pDS+ Design Environment User Manual*
- *Logical Devices and pDS+ Design Environment User Manual*
- *Mentor and pDS+ Design and Simulation Environment User Manual*
- *OrCAD and pDS+ Design and Simulation Environment User Manual*
- *Synopsys and pDS+ Design and Simulation Environment User Manual*
- *VHDL and Verilog Simulation Libraries Supplement*
- *Viewlogic and pDS+ Design and Simulation Environment User Manual*
- *Viewlogic Simulation Library Supplement*
- *Viewlogic Synthesis Libraries Supplement*

Chapter 1 *Introduction*

The Lattice Semiconductor (LSC) design tool strategy for the ispLSI® and pLSI® device families is to support a wide range of design environments. The Lattice Semiconductor pLSI and ispLSI Development System Plus (pDS+™) solution combines third-party CAE tools for design entry and verification with the pDS+ Fitter (also referred to as “compiler”) to offer a complete development solution on PC, UNIX, and HP workstation platforms.

The pDS+ Fitter uses architecture-specific algorithms to synthesize a logic description into an ispLSI or pLSI device. Steps in the device fitting (design compilation) process include design optimization, automatic logic partitioning, automatic placement and routing, and timing analysis.

The pDS+ solution also supports design verification. Design verification options include both functional and timing simulation. Various combinations of graphical and text-based functional and timing simulators are supported by third-party CAE vendors.

Following design verification, the pDS+ Fitter generates a JEDEC fusemap for device programming. Lattice Semiconductor ispLSI devices can be programmed directly from a PC, UNIX, or HP workstation using an ispDOWNLOAD™ cable or ISP™ Engineering Kit, or from dedicated logic designed into the end-system. Lattice Semiconductor pLSI devices can be programmed using third-party programmers.

A diagram of the compilation process is shown in Figure 1-1 on the next page.

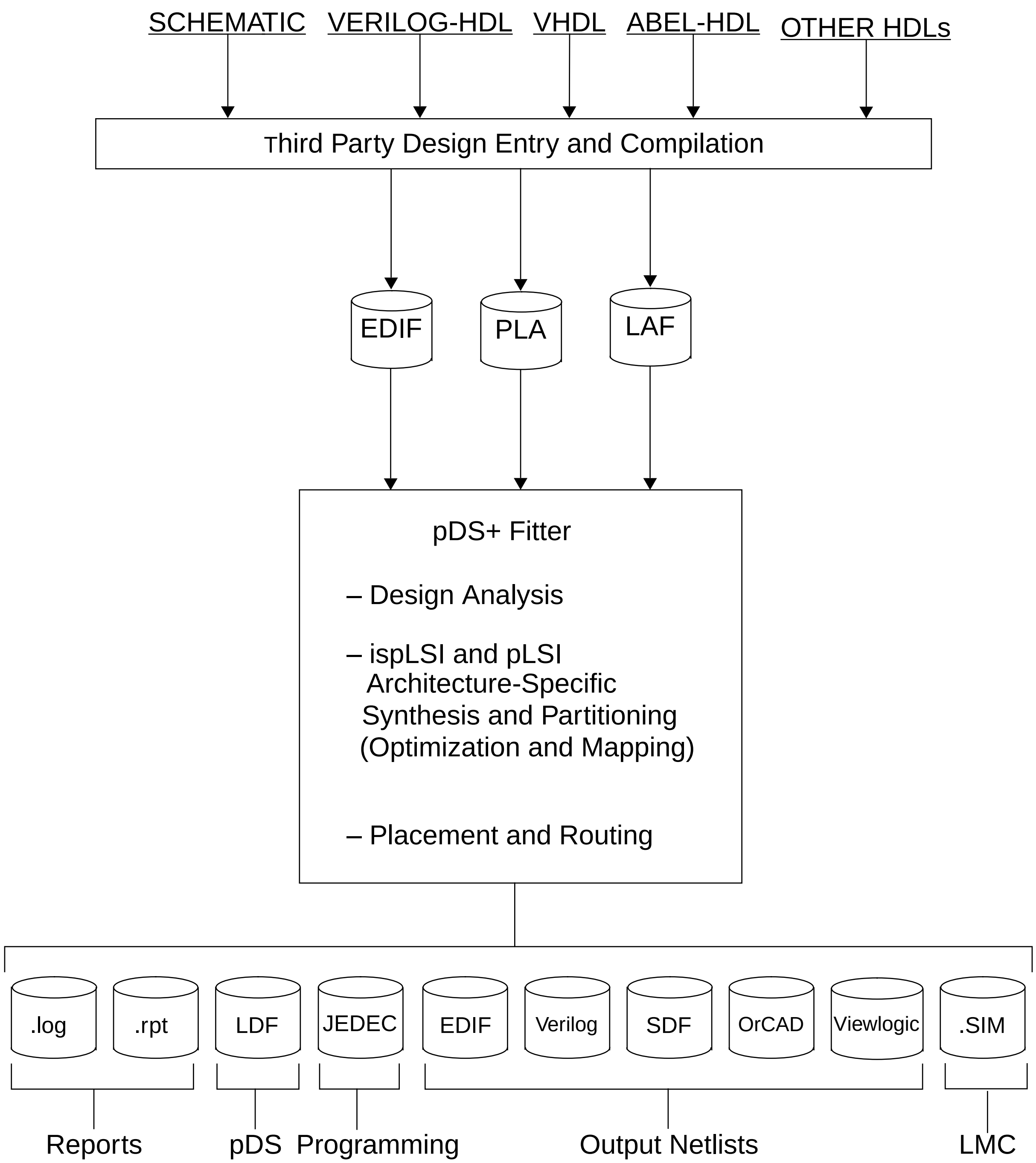


Figure 1-1. pDS+ Fitter Design Flow

Design Entry

Use a third-party schematic capture tool or a Hardware Description Language (HDL) to enter your design. Once design entry is complete, the design is ready to be implemented in an ispLSI or pLSI device.

Third-Party Design Entry Tools

The pDS+ solution supports multiple third-party CAE tools, providing designers with the capability to design in familiar CAE environments. These third-party CAE tools offer schematic capture, hardware description language (such as VHDL, Verilog, ABEL-HDL, etc.), as well as functional and timing simulators for design verification.

Input Formats

The pDS+ Fitter accepts a variety of standard inputs from third-party CAE tools, including EDIF, PLA, and LSC Advanced Format (LAF).

pDS+ Fitter Functions

During the compilation process, the pDS+ Fitter automatically performs the following functions:

- Analyzes your design for errors
- Synthesizes and partitions
- Places and routes
- Produces physical netlists
- Performs a timing analysis
- Produces a JEDEC-compatible device programming file
- Generates report and log files

Design Attributes

When you create your design, you can use Design Attributes to control how the compiler analyzes, synthesizes, partitions, places, and routes your design using the physical resources of a selected Lattice Semiconductor device. The attributes specify pin assignment, register placement, clock usage, and so on. These attributes are described in detail in Chapter 2, “Design Attributes.”

The effects of each attribute on implementation of the design, device resource utilization, and routing are described in Appendix A, “Design Rules and Tips.”

Compiler Control Options

Compiler Control Options determine how much flexibility, or lack of flexibility, the compiler has when implementing your design in the selected Lattice Semiconductor device. The Compiler Control Options specify synthesis strategy, device usage, output netlist format, part selection, and more. Compiler Control Options are described in detail in Chapter 3, “Design Compilation and Control.”

The effects of each control option on routing, device resource utilization, and compiler efficiency are described in Appendix A, “Design Rules and Tips.”

Design Analysis

The Design Analyzer checks for LSC design rule violations. It is automatically invoked whenever a design is compiled. The Design Analyzer checks the following:

- That the design is specified only with valid LSC primitives and/or their derivatives
- Pins identify primary inputs, outputs, and bidirectional I/Os
- Pins have correct assertion (input, output, or bidi)
- No dangling nets are present
- No duplicate pin names are present
- Design attributes are used correctly

Synthesis and Partitioning

The pDS+ Fitter uses Design Attributes and Compiler Control Options to synthesize and partition the design to fit into the given device without violating device architecture or design constraints.

The partitioner performs the following functions:

- Optimizes logic equations by performing the following:
  - Performs multi-level logic optimization
  - Identifies XOR logic to take advantage of physical XOR gates in the device
  - Uses XORs to reduce logic through function inversion where possible
  - Maps parallel registers into a single register
  - Optimizes unused registers and inactive logic
  - Removes unused inputs
- Clusters the partitioned functions according to common clocks, output enable signals, reset signals, and fixed pin properties.
- Groups logic to fit within Generic Logic Blocks (GLBs) and I/O Cells (IOCs).

Placement and Routing

Once the design is partitioned, the placement and routing routine performs the following functions:

- Assigns I/O pins
- Interconnects GLBs and IOCs
- Produces GLB and IOC placement information

Timing Analysis

The pDS+ Timing Analyzer enables you to obtain the following information:

- Analyzes the longest/shortest delay path between multiple source and destination nodes
- Calculates setup and hold time and boundary register for flip-flops and latches
- Determines maximum frequency for clocking a design containing one or more flip-flops and/or latches

pDS+ Fitter Output

After the compilation process, the pDS+ Fitter can create a variety of output formats for post-compilation design analysis, design verification, and device programming.

Netlist Formats

The compiler creates user-specified netlists for use with third-party simulators and Lattice Semiconductor Timing Libraries. The output netlists include the following industry-standard, third-party, and Lattice Semiconductor formats:

- EDIF – EDIF format netlist and timing information for use with any EDIF-compatible timing simulator
- LDF – LSC Design Format (LDF) post-route netlist that can be imported into Lattice Semiconductor's pLSI and ispLSI Development System (pDS®)
- ORCAD – OrCAD INF format netlist and an OrCAD delay back-annotation (DBA) file that can be imported into OrCAD's design and simulation system
- PREROUTE_LDF – LSC Design Format (LDF) pre-route netlist that can be imported into Lattice Semiconductor's pLSI and ispLSI Development System (pDS®)
- SDF – Standard Delay Format (SDF) for use with any OVI (Open Verilog International) compliant Verilog timing simulator
- SIM – SIM format netlist for timing analysis with the pDS+ Timing Analyzer and board-level simulation with Synopsys Logic Modeling Division models
- Verilog – Verilog format netlist for use with any OVI-compliant Verilog simulator
- VHDL – VHDL format netlist for use with any VITAL-compliant VHDL simulator
- Viewlogic – Viewlogic format netlist for use with any Viewlogic simulator

Fusemap Generation

Once the design has routed, the fusemap generation process reads the routed design information, converts the physical layout of the design into device programming information, and generates a fusemap in standard JEDEC format.

The JEDEC device programming file can be downloaded to an ispLSI device using an ispDOWNLOAD cable or ISP Engineering Kit, or to a pLSI device using any device programmer that accepts JEDEC fusemaps. For a list of Lattice Semiconductor-compatible programmers, and further information on device programming options from PC platforms, see the *ISP Daisy Chain Download Reference Manual*. For device programming from UNIX or HP platforms, contact your local Lattice Semiconductor sales representative.

Reports

The pDS+ Viewlogic software provides a report file that shows you how your design fit into the Lattice Semiconductor device architecture, and a timing analysis report file which contains longest and shortest path information. The pDS+ Fitter also generates a log file that lists all error messages, warnings, and informative messages that occurred during compilation.

Designing With the pDS+ Fitter

The pDS+ Fitter uses Design Attributes to specify design constraints for the selected Lattice Semiconductor device. The Compiler Control Options define the parameters of the synthesis process and the designer’s objectives when the design is implemented. The pDS+ Fitter then attempts to synthesize your design subject to the given constraints, namely design constraints and implementation objectives.

To effectively use the pDS+ Fitter, and to better achieve your design objectives, you need to be familiar with the following subjects:

- Lattice Semiconductor Device Architecture
- Design Attributes
- Compiler Control Options
- Design Rules

Constraints and design objectives describe the goals of your design implementation. You must specify to the software what you want by using a proper combination of Design Attributes and Compiler Control Options. These constraints and objectives drive the synthesis process. They can guide the process to deviate from a standard design implementation to better conform to your particular design objectives. Design-rule constraints reflect device architecture restrictions that must be met for a functional design.

Lattice Semiconductor Device Architecture

Each Lattice Semiconductor ispLSI or pLSI device contains logic resources which the pDS+ Fitter uses to partition and place and route user-specified logic in a design. How the pDS+ Fitter uses the logic resources of a device is impacted by the use of Design Attributes and Compiler Control Options. As stated previously, the pDS+ Fitter gives priority to design-rule (device architecture) constraints to meet the requirements for a functional design. Therefore, the compiler occasionally ignores a user-specified Design Attribute or Compiler Control Option to make optimum use of the logic resources of a device, or to meet device constraints.

The logic resources which you can most easily manipulate are Generic Logic Blocks (GLBs) and I/O Cells (IOCs).

Figure 1-2 shows an example of a Lattice Semiconductor ispLSI device and its logic resources.

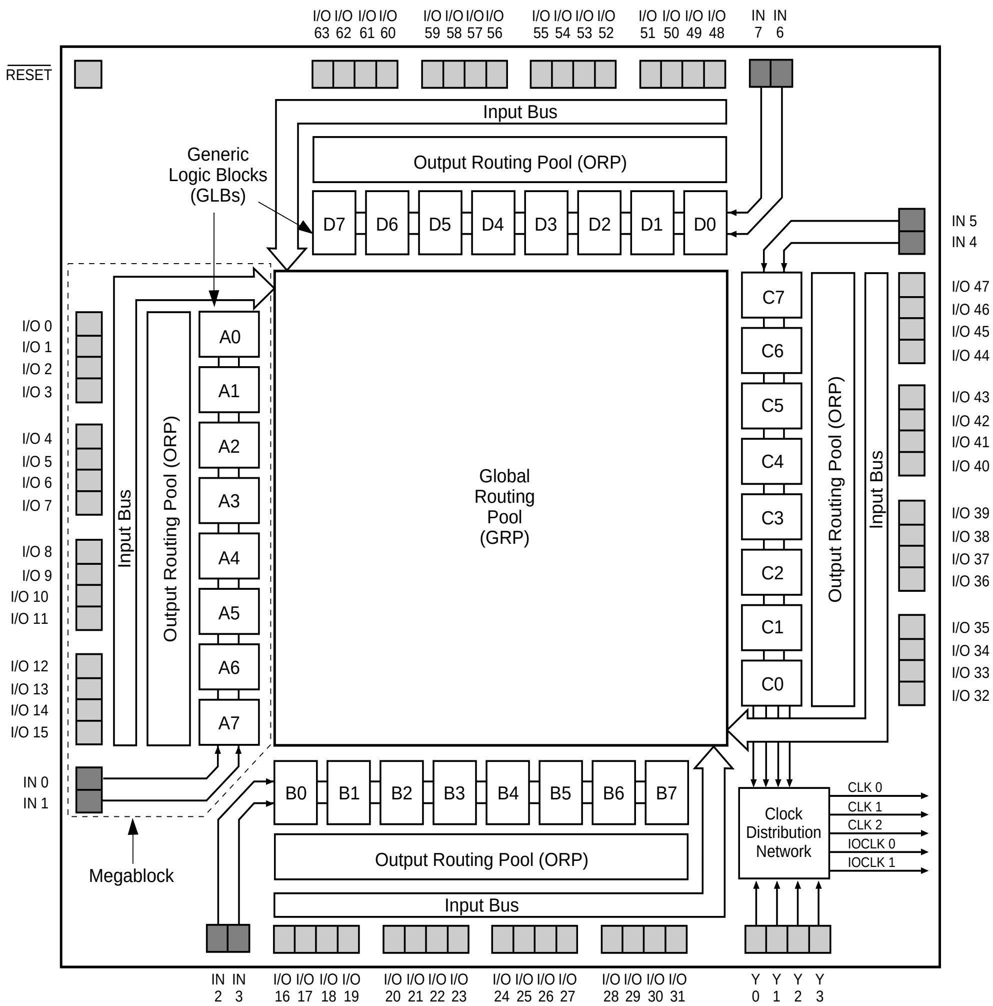


Figure 1-2. Logic Resources for an ispLSI 1032 Device

Generic Logic Blocks (GLB)

A Generic Logic Block (GLB) is the basic unit of logic for each device in the Lattice Semiconductor ispLSI and pLSI family. Each GLB contains global inputs, dedicated inputs, a programmable AND/OR/XOR array, registers, and outputs.

The pDS+ Fitter partitions the logic of your design and maps it to the GLBs of the selected device. Figure 1-3 is an example of a GLB.

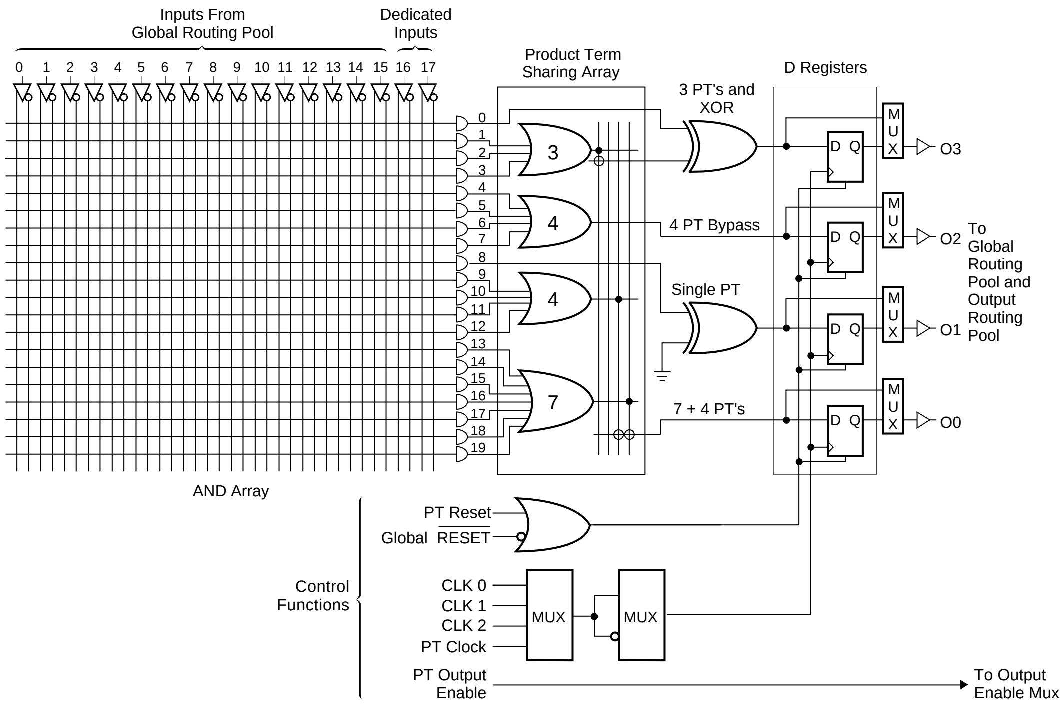


Figure 1-3. GLB Resources

I/O Cell (IOC)

Each IOC connects directly to an I/O pin and can be programmed for combinational input, registered input, latched input, direct output, 3-state output, or bidirectional I/O.

The pDS+ Fitter checks each pin for connectivity and tries to honor pins locked by the user. Figure 1-4 is an example of the logic resources available in an IOC.

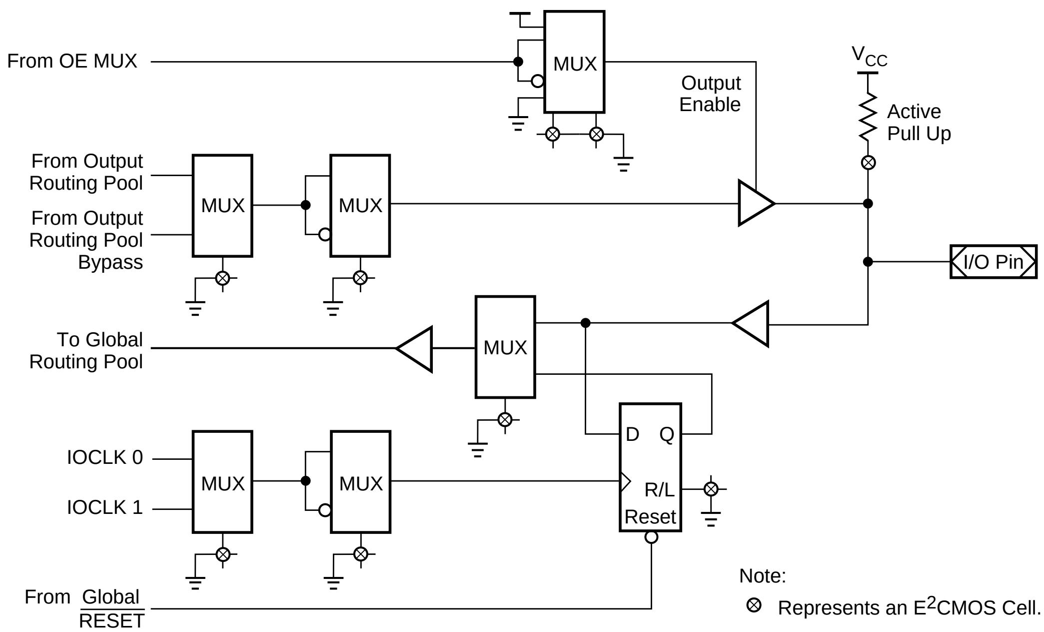


Figure 1-4. IOC Resources

Design Attributes

Design Attribute constraints represent design goals and restrictions that you want, but may not be critical to the successful operation of a design.

The pDS+ Fitter attempts to meet both design-rule and Design Attribute constraints, but gives priority to design-rule constraints, as they are required for functional designs. In cases where a Design Attribute constraint contradicts a design-rule constraint, either the Design Attribute constraint is relaxed, or the netlist is automatically modified to honor the design-rule constraint, and a warning message appears.

Compiler Control Options

Compiler Control Options define global objectives for the design implementation process. These options control usage of device resources for making trade-offs in achieving a desired level of balance among possibly conflicting objectives, such as minimum delay, maximum device resource utilization, and device routability. Such balance is obtained while observing device design rules. Design Attributes are then honored within Compiler Control Options and design rules.

Design Rules

Design rules are by-products of a systematic and automatic design implementation as well as specifics of device architecture. They identify conflicting Design Attributes and Compiler Control Options. These rules have highest priority when the pDS+ Fitter is implementing a design. Design rules range from syntactic limitations, such as maximum allowable length of design identifiers, to rules pertaining to ispLSI and pLSI architecture, such as maximum allowable number of global clocks used in a design.

Directory Structure and Path

The default directory structure for the pDS+ Fitter is listed here:

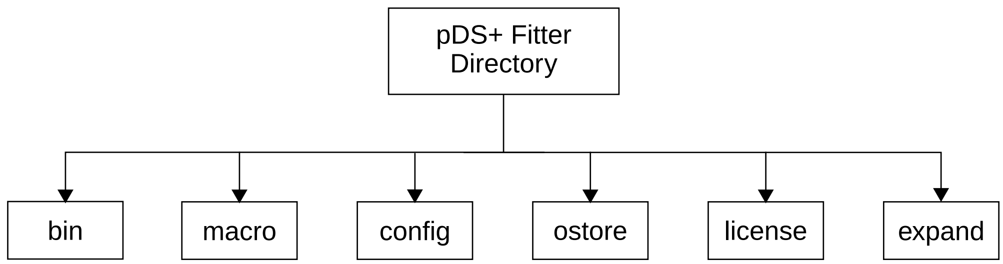


Figure 1-5. pDS+ Fitter Directory Structure

- The bin directory contains the executables for the pDS+ Fitter and utilities for the FLEX*lm* license manager for SunOS, Solaris, and HP platforms.
- The config subdirectory contains the following files:
  - pdsplus.bmf – Message file
  - *.pkg – Device package files
  - *.sdf – Device family files
  - *.tdf – Device timing files
  - library.atr – Default attributes for hard macros
  - lscpart.lst – Lattice Semiconductor device list
  - pds_lic.dat (PC platforms)
- The macro subdirectory contains *.laf files which represent hard macros.
- The ostore subdirectory contains the following directories to support the pDS+ Timing Analyzer:
  - admin – Timing Analyzer database utilities
  - etc – Contains the locator files necessary to designate a server to contain the database
  - lib – Contains device timing files
- The license subdirectory contains the following license template file for SunOS, Solaris, and HP:
  - license.tmp
- The expand subdirectory contains *.lib files for macro expansion.

Design Files

Generally, files can be divided into the following four primary categories:

- **Input files**:
  EDIF
  LAF
  PLA
- **System files**:
  .BMF – Message file
  .PKG – Device package file
  .SDF – Device family file
  .TDF – Device timing file
- **Macro Library files**:
  .LAF – Hard macro description file
- **Output files**:
  .DBA – OrCAD format delay back-annotation file
  .DPT – Detailed timing analysis report
  .EDO – EDIF format netlist file for simulation
  .IFO – OrCAD format netlist file for simulation
  .JED – JEDEC file for device programming
  .LAF – Lattice Advanced Format (LAF) file
  .LDF – Pre-route Lattice Design Format (LDF) file
  .LOG – Log file containing processing, error, and information messages
  .PDF – Post-route Lattice Design Format (LDF) file
  .PPN – Post-route pin file
  .RPT – Report file containing design parameters, design specifications, pre-route design statistics, and post-route design statistics
  .SIM – Simulation file for Viewlogic or LMC
  .SDF – SDF format netlist file for back-annotation with conditional delays
  .VHO – VHDL (generic) format netlist file for simulation
  .VLO – Verilog format netlist file for simulation
  .VSF – SDF format netlist file for back-annotation without conditional delays
  .VTO – VHDL (VITAL) format netlist file for simulation
  .VXF – Cross-reference file for VHDL output files
  .XRF – Cross-reference file for OrCAD output files

Chapter 2 *Design Attributes*

Lattice Semiconductor-specific Design Attributes affect how the compiler implements your design. Using these attributes correctly is a key factor for successful compilation of your design.

When you assign Design Attributes to your design, the compiler takes them as suggestions, rather than as literal assignments. Although Design Attributes are generally used by the compiler as you assign them, occasionally, the compiler will not honor an assigned Design Attribute due to device constraints or resource usage conflicts.

Two common situations in which Design Attributes are rejected are when they are attached to an inactive element of your design, or when conflicting Design Attributes are specified (for example, when a particular clock line is assigned as CLK0 and IOCLK0). A warning message or error message is usually issued by the pDS+ Fitter software whenever it is necessary to ignore attributes in your design. Warning messages are also issued when Design Attributes are applied that would result in a noticeable deviation from a more optimal implementation of the design.

Use Design Attributes conservatively to take advantage of their effectiveness and to avoid any significant side effects that may result from extensive usage. Use Design Attributes in localized areas of a design with specific implementation needs in mind, such as timing or observability.

Design Attributes

The following Design Attributes control how your design is implemented into the logic resources of the target device. Each Design Attribute you add places restrictions on the compiler by giving it less freedom to use available logic resources. Apply these Design Attributes carefully to avoid overconstraining the compiler and possibly causing a routing failure.

The Design Attributes are functionally grouped as follows:

- Net Attributes
  - CLK – Assigns clock signals to specific clock lines
  - GROUP – Suggests a particular grouping of functions into GLBs
  - PRESERVE – Prevents removal of a net during logic optimization
- Path Attributes
  - SAP/EAP – Specifies the Start and End of an Asynchronous Path
  - SCP/ECP – Specifies the Start and End of a Critical Path
  - SNP/ENP – Specifies the Start and End of a No-Minimize Path
- Symbol Attributes
  - LXOR2 – Enforces direct implementation of a two-input XOR
  - OPTIMIZE – Allows optimization of a hard macro
  - PROTECT – Prevents logic optimization of a primitive
  - REGTYPE – Specifies register location (in GLB or IOC)
- Pin Attributes
  - CRIT – Assigns specific outputs to use the ORP bypass
  - LOCK – Assigns device I/O pins
  - PULLUP – Assigns specific IOCs to use the pull-up function on the device
  - SLOWSLEW – Assigns slow slew rate to specific outputs

The following shows the general syntax used in this manual for Design Attributes:

```
attribute_name[=]attribute_value
```

An equal sign (=) may or may not be needed in your design environment. The exact syntax for a Design Attribute may change slightly within the different design entry environments supported by Lattice Semiconductor including Cadence, DATA I/O, Exemplar, ISDATA, Logical Devices, Mentor, OrCAD, Synopsys, and Viewlogic.

Applying Design Attributes

Design Attributes are applied to pins, nets, or symbols in your design as shown in Table 2-1.

Table 2-1. Where to Place Attributes

Attribute Type	Attribute Placement
CLK	Net
GROUP	Net
PRESERVE	Net
SAP/EAP, SCP/ECP, SNP/ENP	Net
LXOR2	Symbol
OPTIMIZE	Symbol
PROTECT	Symbol
REGTYPE	Symbol
CRIT	External Pin
LOCK	External Pin
PULLUP	External Pin
SLOWSLEW	External Pin

Precedence of Design Attributes

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

Table 2-2 groups Design Attributes in their order of precedence when relating to the same logic. A Design Attribute with a higher precedence (for example, 1) overrides those with lower precedence (for example, 5). Design Attributes with the same level of precedence will generally not override each other, but may override each other in a design-dependent fashion if they conflict.

Table 2-2. Design Attribute Precedence

Precedence	Design Attribute
1	LOCK, LXOR2, OPTIMIZE, PRESERVE, PROTECT, PULLUP, SLOWSLEW
2	SAP/EAP
3	SNP/ENP
4	SCP/ECP
5	CLK, CRIT, GROUP, REGTYPE

Net Attributes

The following Design Attributes can be applied to the nets in your design:

- CLK
- GROUP
- PRESERVE

CLK

The CLK Design Attribute assigns device clocks to specific clock inputs of GLBs or IOCs.

Synopsis

```
CLK[=]CLK0|CLK1|CLK2|IOCLK0|IOCLK1|FASTCLK|SLOWCLK
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

A clock signal is any net connected to the clock input of a register. If you do not use a CLK attribute, the compiler automatically determines whether nets should use dedicated clock resources or the slower product term (PT) clocks.

The CLK attribute can have the following values:

- CLK0 – Assigns the signal to the dedicated clock line CLK0
- CLK1 – Assigns the signal to the dedicated clock line CLK1
- CLK2 – Assigns the signal to the dedicated clock line CLK2

Any register clocked by CLK0, CLK1, or CLK2 clock signals is automatically placed inside a GLB.

- IOCLK0 – Assigns the signal to the dedicated clock line IOCLK0
- IOCLK1 – Assigns the signal to the dedicated clock line IOCLK1

Any register clocked by IOCLK0 or IOCLK1 is automatically placed within an IOC (not within a GLB) if it satisfies the following four conditions:

- The input to the register is connected to an input pin
- The input pin only drives the register and does not drive any other gate or register
- The input pin is not locked or is locked to an IOC pin
- The register has no product term reset

> ***NOTE*** The last condition may not be required if the Compiler Control Option USE_GLOBAL_RESET ON is used. See “USE_GLOBAL_RESET” in Chapter 3 for more information.

If a register being clocked by IOCLK0 or IOCLK1 does not satisfy all of the conditions listed above, a warning message is issued by the compiler and the register may be moved to a GLB and the CLK attribute may be changed to CLK0, CLK1, or CLK2.

- FASTCLK – Assigns the signal to any of the dedicated CLK lines (CLK0, CLK1, CLK2, IOCLK0, or IOCLK1) at the discretion of the compiler. This allows more partitioning flexibility. Gated clocks, when specified as FASTCLKs, use the dedicated clock GLB and the clock distribution network where available.
- SLOWCLK – Assigns the signal to a GLB PT clock. Any register clocked by a SLOWCLK signal is automatically placed within a GLB (not within an IOC). Use this option to define a clock as a PT clock.

The CLK attribute should be attached to a net leading directly to the clock input of one or more registers. Any intervening logic gates, except simple buffers and inverters, disable the relationship between the CLK attribute and the clock signal, and any registers driven by such a signal.

Any registers with clock, reset, or data inputs driven by constants GND or VCC, whose outputs cannot be toggled, are removed and their outputs are replaced by constant GND. Any clock attribute attached to the clock inputs of these registers is ignored.

Any clock attribute applied to a clock signal which is driving both GLB and IOC registers (split clock), should completely describe the desired clock line usage. Certain combinations of CLK attributes may be acceptable within the constraints of the specified Lattice Semiconductor device when separated by commas. For example:

```
CLK CLK2, IOCLK0, FASTCLK
```

See the *Lattice Semiconductor ISP Encyclopedia* for more information on legal combinations for each Lattice Semiconductor device.

All Lattice Semiconductor devices have dedicated clock pins which can help increase the operating speed of the part. For more information on dedicated clock pins, see the *Lattice Semiconductor ISP Encyclopedia*.

Example

Figure 2-1 shows the assignment of a clock signal to the dedicated clock line CLK2.

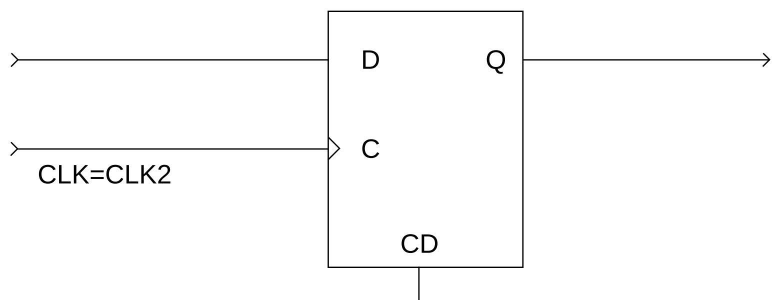


Figure 2-1. Assigning a Clock Signal to a Dedicated Clock Line

GROUP

The GROUP Design Attribute identifies GLB outputs which are to be grouped together when forming GLBs.

Synopsis

```
GROUP[=]group_name
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

Any net with the GROUP attribute is preserved in the resulting netlist. Furthermore, nets with GROUP attributes and with similar group names are grouped as GLB outputs of a single GLB where possible. Such a GLB can still be split by the placement and routing process when necessary to improve routability. The GROUP attribute is ignored if more than four nets with GROUP attributes have the same group names.

The GROUP attribute has the lowest precedence among Design Attributes, and therefore will be ignored if it conflicts with any architectural constraints or any other Design Attributes.

Use the GROUP attribute carefully in possible conjunction with other Design Attributes to guarantee a feasible grouping of logic after synthesis. Refer to the report file from the compiler to see the implementation of logic after synthesis, and to deduce the possible cause of a grouping violation.

Example

Figure 2-2 shows an example of the GROUP attribute assigned to two nets, net D and net H. These two nets will be implemented as outputs of a single GLB. However, if the Compiler Control Option MAX_GLB_IN is set to 5, then this grouping will be ignored, because it violates the specified maximum GLB input limit.

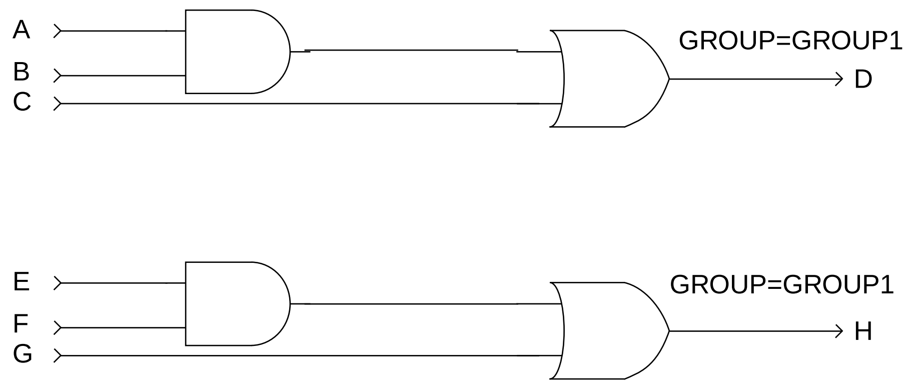


Figure 2-2. Assigning GROUP Design Attribute to Nets

PRESERVE

The PRESERVE Design Attribute identifies nets that you do not want eliminated during the logic optimization process.

Synopsis

PRESERVE

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

PRESERVE forces the net to a GLB or IOC output. This is useful for debugging purposes where specific test points need to be preserved.

Some design rules for using the PRESERVE Design Attribute include the following:

- PRESERVE assigns a net to a GLB or IOC output; this may increase delay levels, as well as the total required number of GLBs when used improperly.
- A preserved net implemented as GLB output may be duplicated by the compiler for successful routing. Duplicated nets derive their names from the preserved net name and may not be available in the user-specified form. See the SAP/EAP attribute description to avoid this duplication.
- Parallel logic is normally removed by the compiler if their outputs are not preserved. Use PRESERVE to prevent the compiler from removing parallel logic.

Example

Figure 2-3 shows an example of assigning PRESERVE to net D. In this example, the AND and OR gates can be mapped into a single GLB, but are mapped into two GLB outputs with net name D maintained as a GLB output name driven by the AND gate as a result of the PRESERVE attribute. Net D is implemented inside a GLB if it is not preserved.

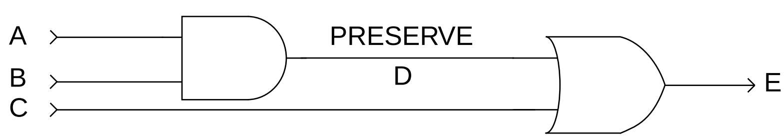


Figure 2-3. Using the PRESERVE Attribute

You can also use PRESERVE to assist the compiler in partitioning your design. For example, the logic in Figure 2-4 translates into 128 PTs in a sum-of-products form.

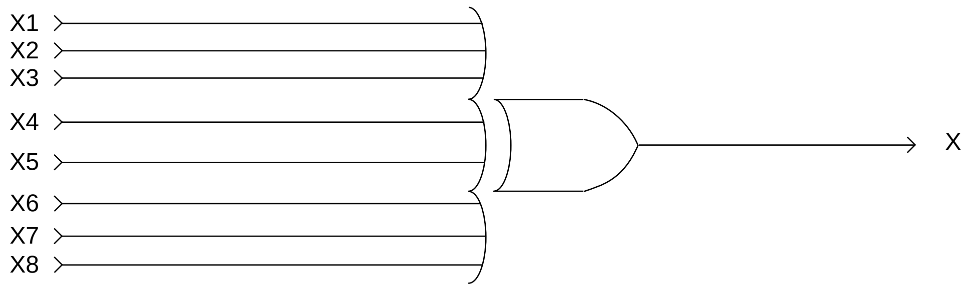


Figure 2-4. XOR Without PRESERVE Assigned

By using PRESERVE on the Y1 and Y2 nets, as shown in Figure 2-5, Y1 and Y2 are preserved and the number of PTs is reduced to eight for each first-level exclusive-or (for a total of 18 PTs from the original 128 PTs).

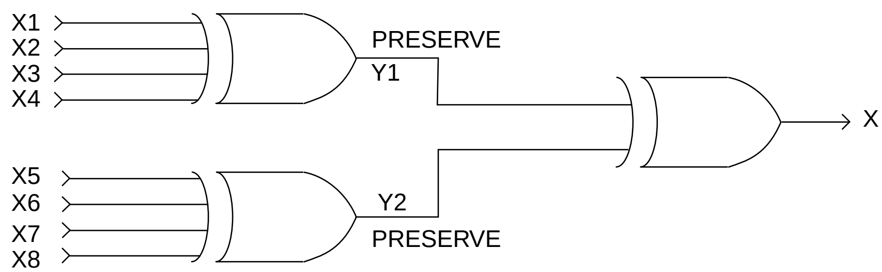


Figure 2-5. XORs with PRESERVE Assigned

Figure 2-6 is an example of parallel registers before the design is optimized.

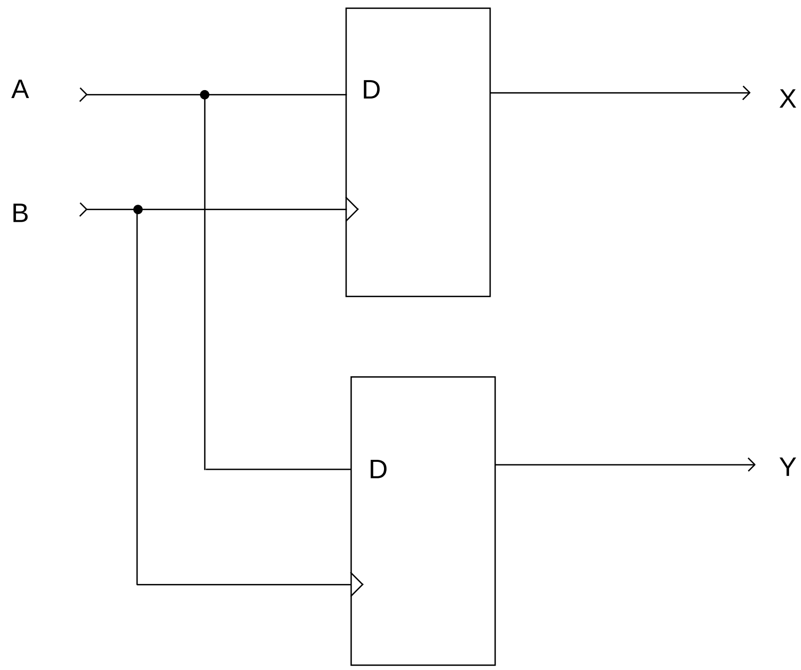


Figure 2-6. Parallel Registers without PRESERVE Attributes

Without PRESERVE attributes, a parallel register is removed during optimization, resulting in the implementation shown in Figure 2-7.

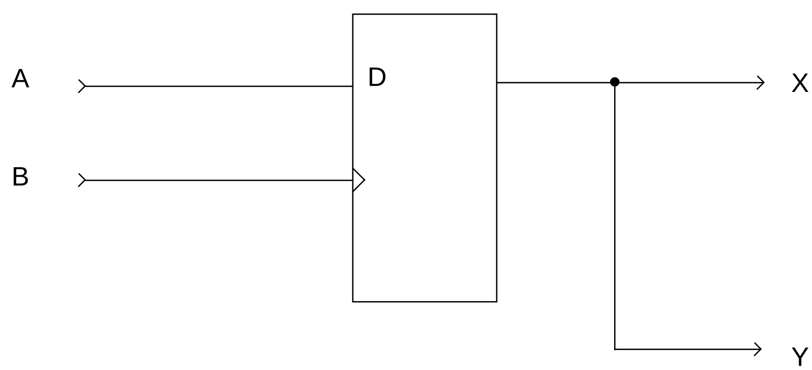


Figure 2-7. Parallel Registers Reduced

Figure 2-8 shows the parallel registers with PRESERVE attributes attached to both register outputs. This prevents the removal of parallel registers during optimization and preserves the output nets.

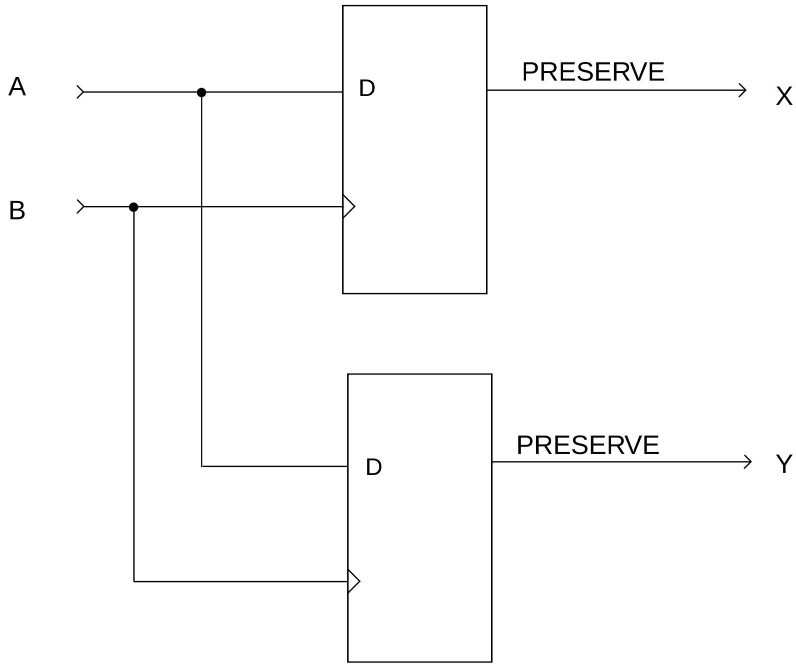


Figure 2-8. Parallel Registers with PRESERVE Attributes

Path Attributes

Path Attributes can be used to identify a set of paths as Asynchronous Paths, Critical Paths, or No-Minimize Paths. Each path identifies a single net as the starting point of the path and a corresponding single net as the ending point of the path. Any starting point path specification that does not have a corresponding ending point is ignored and a warning message appears. However, a path specification can include an ending point only, in which case any combinational path leading to the specified ending point is considered to belong to the specified path. Any path specification with duplicate starting and/or ending points for the same path is flagged as an illegal specification. Therefore, always define multiple paths in a single path specification.

When several different path specifications overlap, as far as logic optimization is concerned, Asynchronous Path specifications override any No-Minimize Path specifications; and, any Asynchronous Path and/or No-Minimize Path specifications override any Critical Path specifications involving the same logic. The net attribute PRESERVE, when used in conjunction with a path attribute, impacts the implementation of logic independently. For example, a PRESERVE attribute applied in the middle of a Critical Path creates a GLB boundary at that point, despite the fact that a more efficient implementation of logic could provide a more optimal implementation of the Critical Path.

Any path going through a register or a 3-state buffer is ignored. Any part of a path going through a hard macro is also ignored.

Any starting or ending point of a path is interpreted as a soft boundary, which allows similar gates to be merged over the boundary during the mapping process. No global optimization, however, is performed over the starting or ending points. A hard boundary can be defined by using the PRESERVE attribute in conjunction with a starting or ending point specification for the path, in which case no gates are merged over the boundaries defined by the preserved starting or ending points.

Use a buffer when there are combinational loops and the whole loop is covered by path attributes. In Figure 2-9, Path1 only relates to the net OUT and not to the path going through gates B, C, and D.

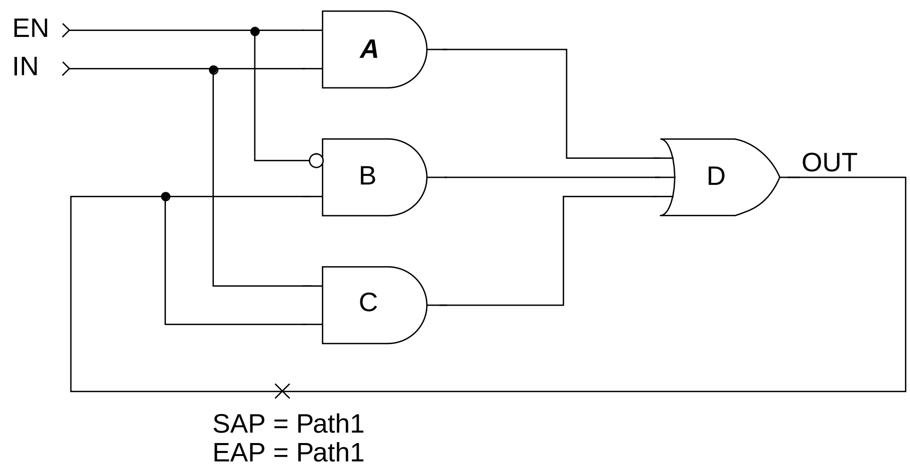


Figure 2-9. Path1 Relates to the Net OUT Only

To correctly identify the logic on the loop, modify the network as shown below in Figure 2-10. In this case, gates B, C, and D are all considered to be on Path1.

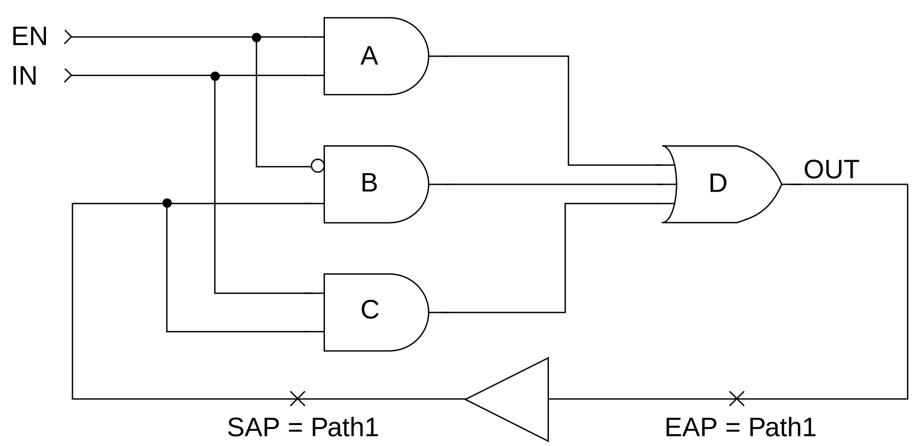


Figure 2-10. Modify the Network to Identify the Loop

SAP/EAP

The SAP Design Attribute specifies the Start of an Asynchronous Path. Each SAP attribute must have an associated EAP attribute with the same path name.

The EAP Design Attribute specifies the End of an Asynchronous Path and does not require a matching SAP attribute.

Synopsis

```
SAP[=]path1, path2, ... pathN
EAP[=]path1, path2, ... pathN
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

The compiler duplicates GLB outputs when necessary to improve routability. If a GLB output is part of an asynchronous path, its duplication may be undesirable. Asynchronous Path specifications prevent the compiler from optimizing logic and duplicating GLB outputs that are located on any path connecting the starting and ending points of the path.

If SAP and EAP are applied to the same net, that net is not duplicated by the compiler if it is implemented as a GLB output. Use the PRESERVE attribute to force the compiler to implement that net as a GLB output.

The following are some rules for using the SAP/EAP Design Attributes:

- Allowing the compiler to duplicate outputs gives the router more flexibility and can prevent routing problems. Using SAP/EAP may unnecessarily overconstrain the compiler.
- Merging similar gates at SAP/EAP boundaries that do not have the PRESERVE attribute may cause the output of the resulting gate to be duplicated by the compiler. Use PRESERVE to prevent merging similar gates and net duplication.
- If a net with SAP/EAP is driven by a signal inversion, the net may disappear due to forward or backward merging of the signal inversion over the net. Use PRESERVE to prevent merging of a signal inversion over a net with SAP/EAP attributes.

Any buffer on an asynchronous path is implemented as a single GLB level. Use caution in specifying asynchronous paths through library macros which have embedded buffers, such as input and output buffering macros.

Example

Figure 2-11 displays part of a design schematic. Figure 2-12 is a potential implementation at the end of the compilation process; the compiler duplicated the register to improve routability. Figure 2-13 specifies the register output as asynchronous and thus the compiler would not duplicate the register resulting in an implementation similar to Figure 2-14.

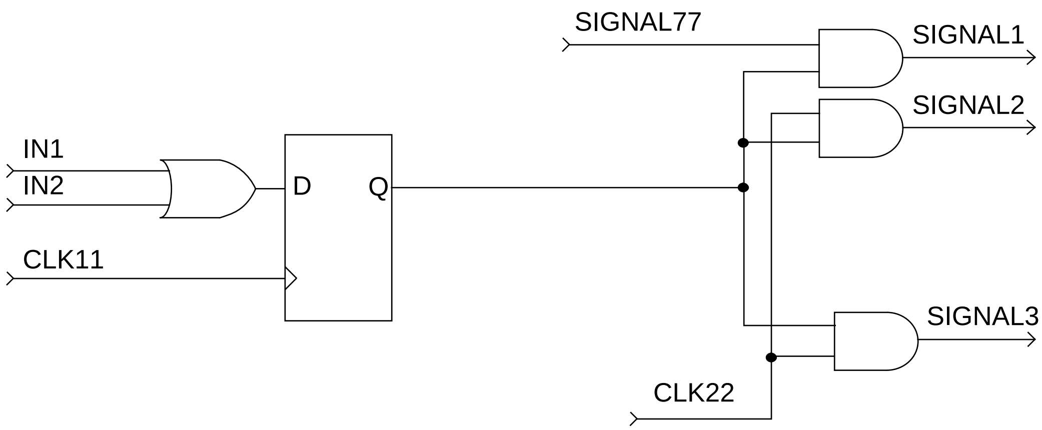


Figure 2-11. Circuit Without SAP/EAP Before Compilation

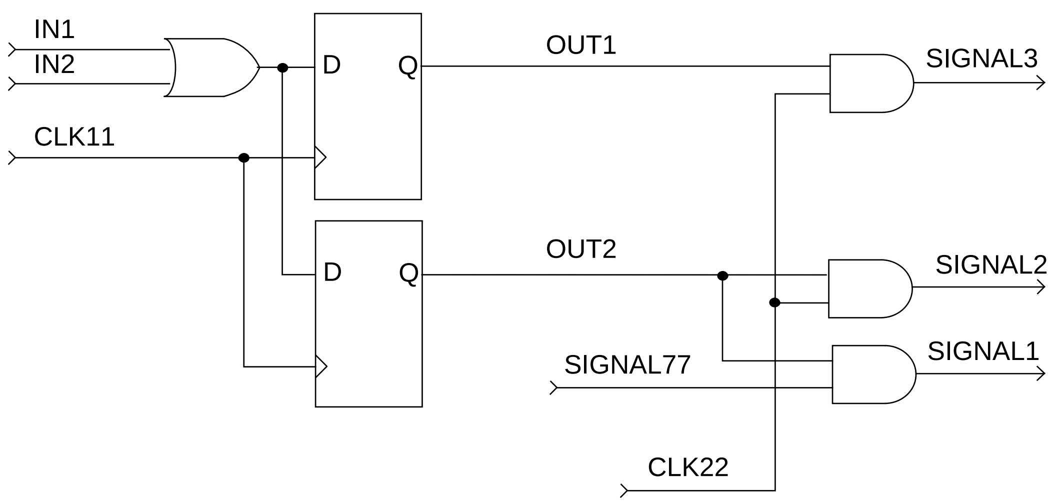


Figure 2-12. Circuit Without SAP/EAP After Compilation

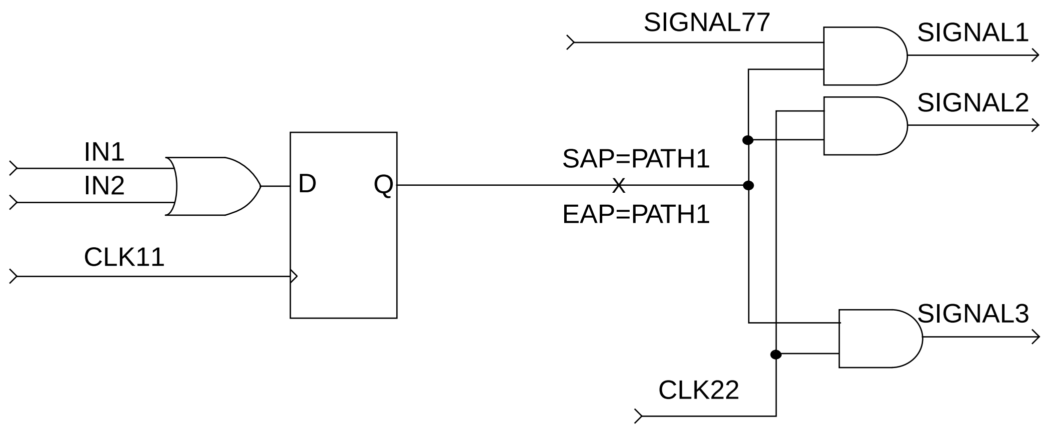


Figure 2-13. Circuit Using SAP/EAP Before Compilation

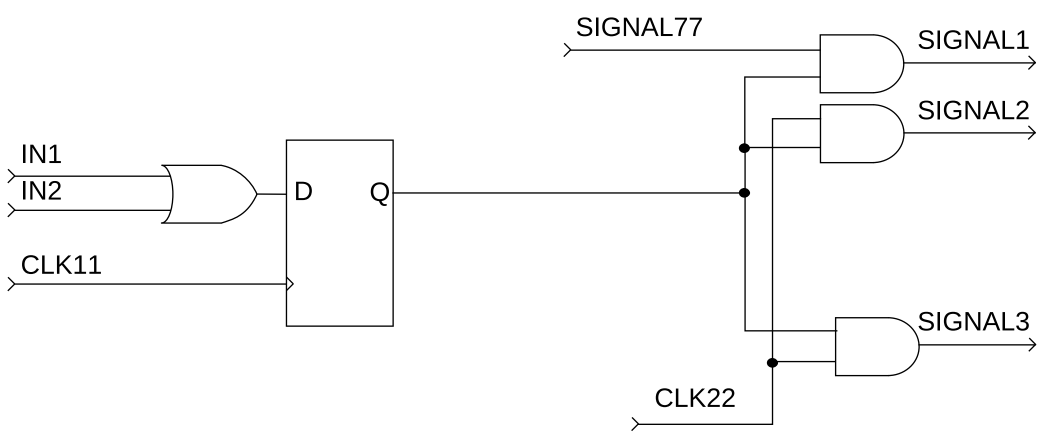


Figure 2-14. Circuit Using SAP/EAP After Compilation

SCP/ECP

The SCP Design Attribute specifies the Start of a Critical Path. Each SCP attribute must have an associated ECP attribute with the same path name.

The ECP Design Attribute specifies the End of a Critical Path and does not require a matching SCP attribute.

Synopsis

```
SCP[=]path1, path2, ..., pathN
ECP[=]path1, path2, ..., pathN
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

The SCP/ECP Design Attribute performs two functions. First, it instructs the compiler to minimize the number of GLB levels in a given path. Second, it instructs the compiler to minimize the signal path delay within each GLB level by utilizing a four-product-term bypass if possible.

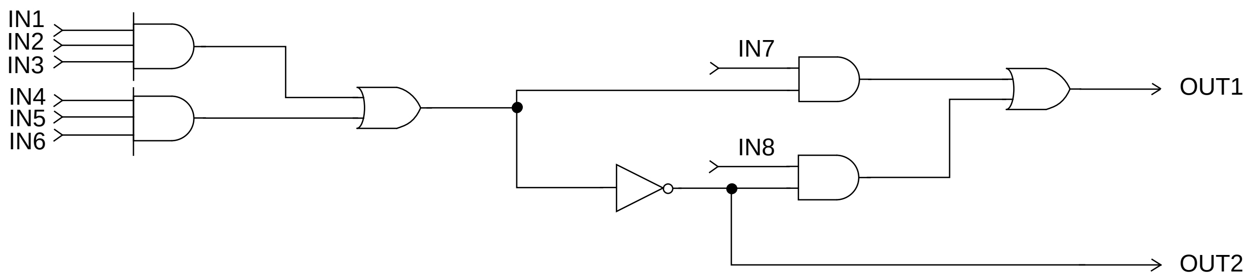
You only need to apply SCP/ECP properties to a representative subset of the related Critical Path starting and ending points. If you do not know which Critical Path beginning or ending points to mark, you can mark them all. If SCP and ECP attributes relating to the same path are applied to the same net, they are ignored.

A critical path implementation may not produce what you expected if applied to a wide-input logic gate (which is not directly mappable to the Lattice Semiconductor ispLSI and pLSI architecture). To achieve your desired implementation, replace the wide-input logic gate with several narrow-input gates and apply SCP and ECP attributes.

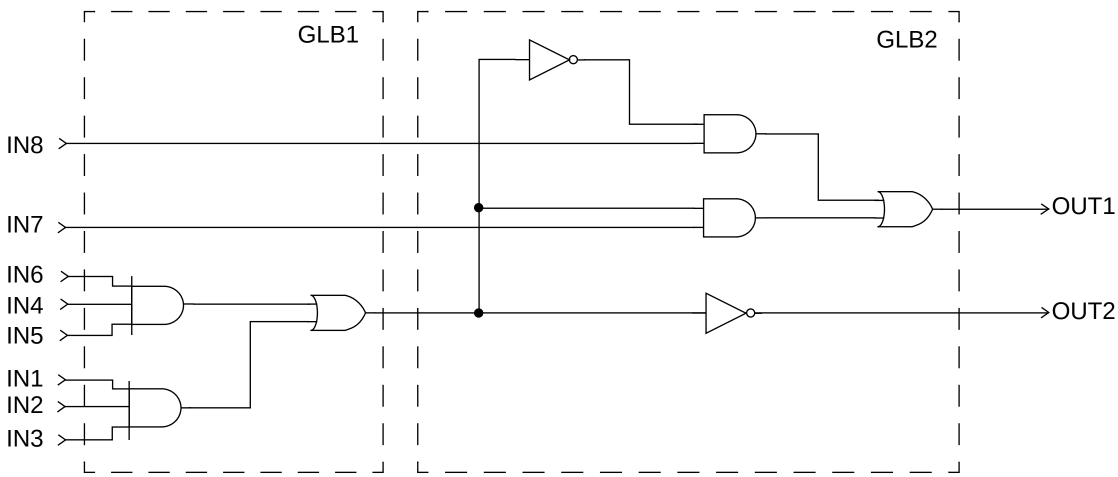
Specify critical paths with embedded registers by specifying two separate critical paths: a Critical Input Path to the register input, and a Critical Output Path from the register output.

Example

In Figure 2-15 (A), the circuit does not use SCP/ECP attributes. The resulting two-GLB level implementation is shown in Figure 2-15 (B). The compiler normally avoids a one-level GLB implementation when it results in a large number of PTs. The second circuit shown in Figure 2-16 (A) uses SCP/ECP attributes and results in a one-GLB level implementation (Figure 2-16 (B)) despite its use of more product terms and thus more GLB resources. A two-GLB level implementation uses logic resources more efficiently; a one-GLB level implementation is superior for speed-critical applications.



(A)



(B)

Figure 2-15. (A) SCP/ECP Not Used (B) Resulting Two-GLB Level Implementation

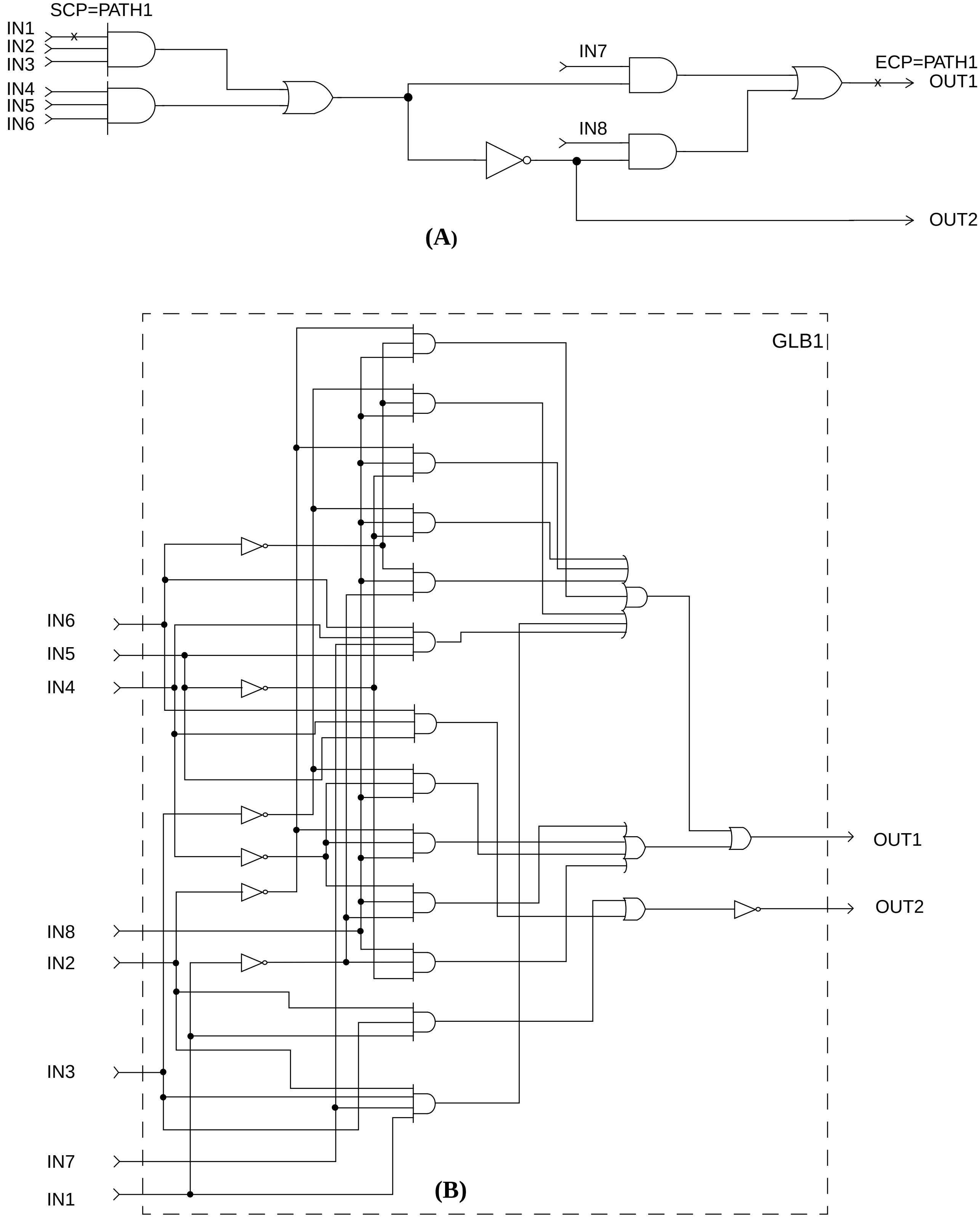


Figure 2-16. (A) SCP/ECP Used (B) Resulting One-GLB Level Implementation

SNP/ENP

The SNP Design Attribute specifies the Start of a No-Minimize Path. Each SNP attribute must have an associated ENP attribute with the same path name.

The ENP Design Attribute specifies the End of a No-Minimize Path and does not require a matching SNP attribute.

Synopsis

```
SNP[=]path1, path2, ..., pathN
ENP[=]path1, path2, ..., pathN
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

The compiler does not optimize the logic on a No-Minimize Path. However, similar gates may be merged and inactive or parallel logic may be removed, or a wide-input logic gate may be split during the mapping process.

Any buffer on a No-Minimize Path is implemented in one GLB level. Exercise caution when specifying No-Minimize Paths through library macros with embedded buffers, such as input and output buffering macros. Any inverting buffer on a No-Minimize Path is merged with the driving or driven logic when appropriate.

If SNP and ENP relating to the same path are applied to the same net, they are ignored.

Example

Figure 2-17 shows an example of a circuit without SNP/ENP attributes assigned. The implementation of this logic is displayed in Figure 2-18, which can potentially exhibit a glitch at the output node OUT. (OUT is implementing a latch function.)

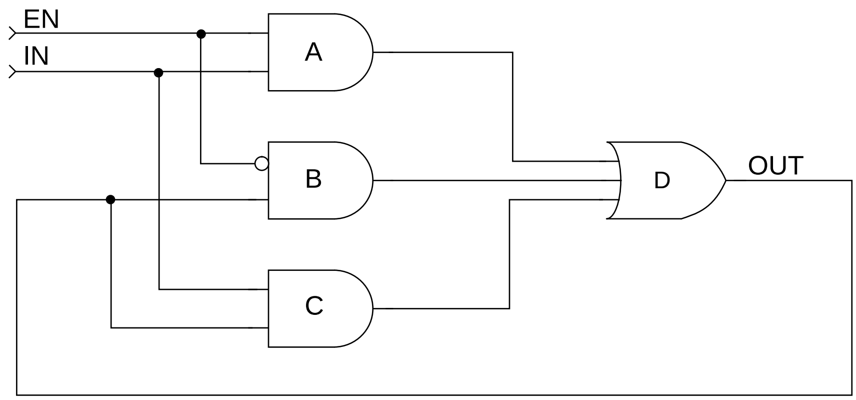


Figure 2-17. Circuit Not Using SNP/ENP Before Compilation

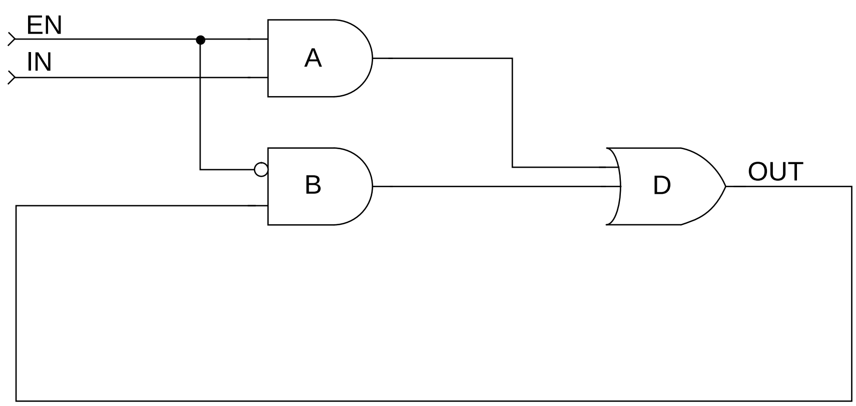


Figure 2-18. Circuit Not Using SNP/ENP After Compilation

Figure 2-19 is the same circuit with SNP and ENP attributes assigned. In the resulting implementation (Figure 2-20), the gates on the No-Minimize Path “Path1” are maintained, resulting in a glitch-free latch function.

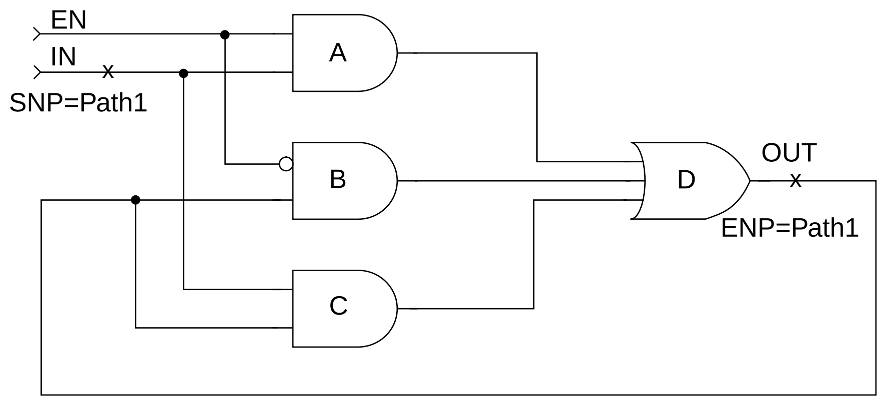


Figure 2-19. Circuit Using SNP/ENP Before Compilation

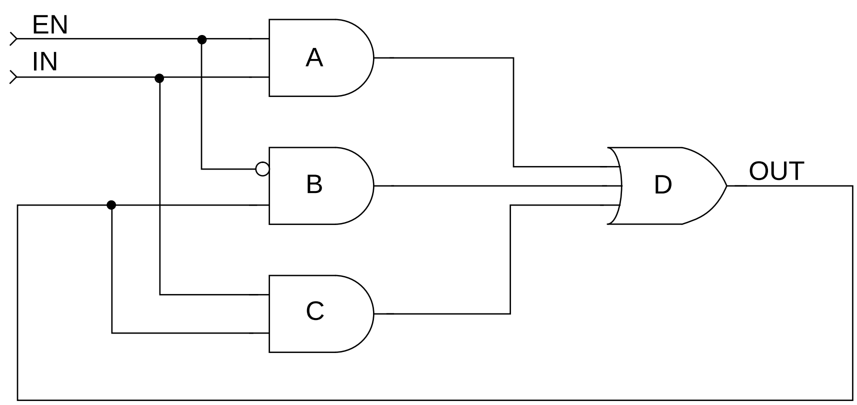


Figure 2-20. Circuit Using SNP/ENP After Compilation

Symbol Attributes

The following Design Attributes can be applied to the symbols in your design:

- LXOR2
- OPTIMIZE
- PROTECT
- REGTYPE

LXOR2

The LXOR2 Design Attribute enforces implementation of a two-input exclusive-or function using a hardware, two-input exclusive-or.

Synopsis

`LXOR2[=]`*`node_name`*

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

In schematic-based applications, this attribute may be specified through instantiation of a particular library primitive, normally named “LXOR2.” In HDL environments, the LXOR2 attribute should only be applied to a simple two-input exclusive-or function where both inputs are simple variables. This removes any ambiguity in recognizing the exclusive-or gate.

OPTIMIZE

The OPTIMIZE Design Attribute specifies whether a macro is hard or soft and can only be used with schematic packages that are supported by the LSC macro libraries.

Synopsis

```
OPTIMIZE[=]ON|OFF
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

A (soft) macro is a predefined netlist of a particular logic function. A macro may be pre-mapped to the ispLSI/pLSI architecture for optimal resource utilization or performance. Such a pre-mapped representation of a macro is referred to as a hard macro.

The default for hard macros is OPTIMIZE OFF, which instructs the compiler not to optimize them. They are treated as “black boxes” which are pre-mapped in the ispLSI and pLSI architecture. To change these hard macros to soft macros, add the OPTIMIZE ON attribute to each applicable macro instance. This tells the compiler to use the netlist of that macro and optimize it with the rest of your design. All other macros are *soft only*, including any user-created macros.

To change a “soft” hard macro back to its original non-optimizable state, change OPTIMIZE ON to OPTIMIZE OFF. This can only be done on soft macros which are also available in hard macro form. You can use OPTIMIZE ON with any of the hard macros listed in the *Macro Library Reference Manual*.

Every hard macro in the macro library has an equivalent soft macro, but there are some soft macros that *do not* have equivalent hard macros. If you attempt to specify OPTIMIZE OFF on a soft macro that does not have a corresponding hard macro, an error occurs. See the *Macro Library Reference Manual* for more details.

Because hard macros require predefined mapping of their logic into ispLSI or pLSI device resources, exercise caution when using hard macros with other Design Attributes. Certain combinations, such as applying CLK or CRIT attributes to the output of a hard macro, can make a design infeasible in the specified form. The pDS+ Fitter usually resolves this by ignoring one or more of these attributes.

During optimization, all or part of an inactive hard macro logic may be removed. This may include any unused hard macro output, as well as any inputs driven by a constant value.

Example

Figure 2-21 is an example of using OPTIMIZE ON and OPTIMIZE OFF.

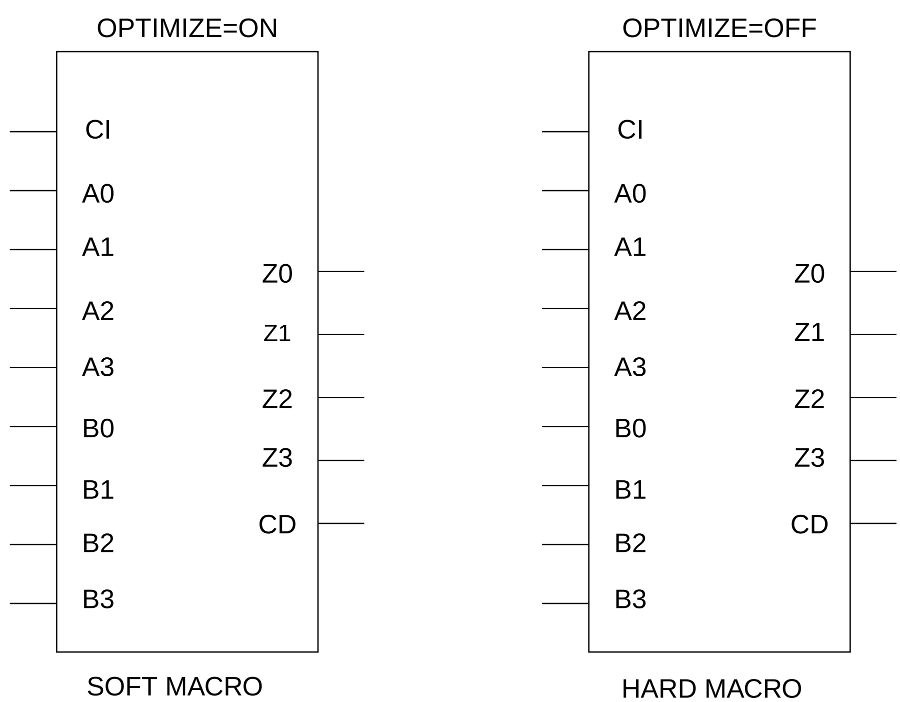


Figure 2-21. OPTIMIZE Attribute Usage

PROTECT

The PROTECT Design Attribute prevents optimization of the specified combinational primitive during logic optimization of your design.

Synopsis

```
PROTECT[=]node_name
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

Attach PROTECT to the symbol or an instance of a symbol in schematic-entry design environments. It prevents optimization of the specified combinational primitive during logic optimization of your design. The protected primitive may still be merged with similar gates or split during the mapping stage of synthesis.

Inactive or parallel logic may be removed during optimization, even though a PROTECT attribute is assigned. A protected buffer will be implemented in one GLB level. A protected inverter will be merged with the surrounding logic. Registers, LXOR2s, 3-state buffers, I/O pins, and hard macros cannot be protected.

Example

In Figure 2-22, PROTECT has similar impact to using a No-Minimize Path. The implementation will closely resemble the circuit shown in Figure 2-23.

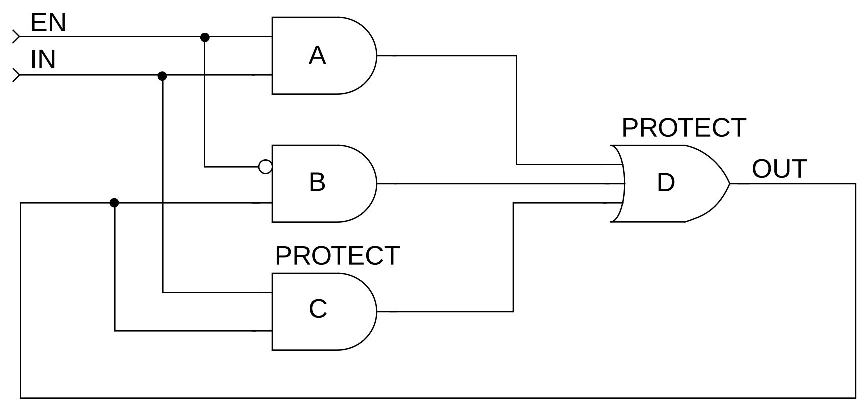


Figure 2-22. Correct Use of the PROTECT Attribute

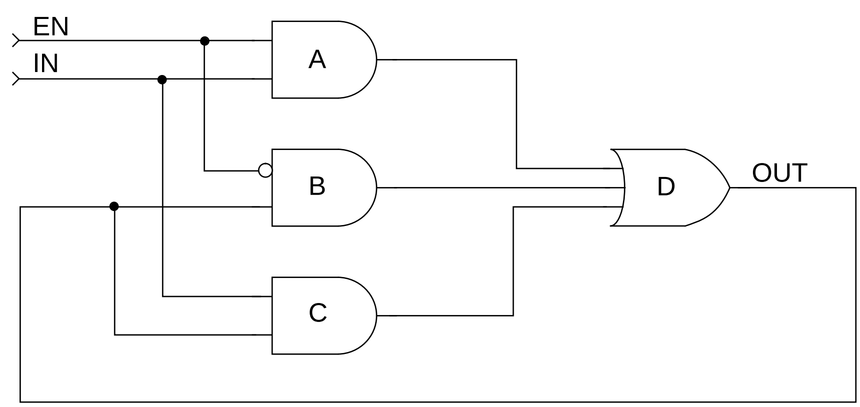


Figure 2-23. Implementation Result After Using PROTECT

REGTYPE

The REGTYPE Design Attribute allows you to place a particular register either inside a GLB or an IOC.

Synopsis

```
REGTYPE[=]node_name GLB|IOC
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

REGTYPE GLB places the register in a GLB and allows the compiler to use the following GLB clocks:

- CLK0
- CLK1
- CLK2
- SLOWCLK (PT Clock)

REGTYPE IOC places the register in an IOC and allows the compiler to use the following IOC clocks:

- IOCLK0
- IOCLK1

Specifying REGTYPE also implies which set of dedicated clocks the compiler can use for the register. REGTYPE IOC can only be used if the register satisfies the following four conditions:

- The input to the register is connected to an input pin
- The input pin only drives the register and does not drive any other gate or register
- The input pin is not locked or is locked to an IOC pin
- The register has no product term reset

NOTE The last condition may not be required if the Compiler Control Option USE_GLOBAL_RESET ON is used. See “USE_GLOBAL_RESET” in Chapter 3 for more information.

The compiler automatically places registers and assigns clock input pins for you. The compiler attempts to place as many registers into IOCs as possible. Use REGTYPE when you need to group registers into IOCs or GLBs for minimizing clock skew.

For example, if you have an 8-bit bus, you may want to place all of the bus registers in the same relative location. (All registers can be placed in GLBs or all registers can be placed in IOCs.)

If you assign a dedicated clock pin to a clock, the compiler automatically determines where to place the registers driven by the clock. In this case, the REGTYPE attribute is not required. See the CLK attribute for more information.

Example

If REGTYPE and CLK attribute values conflict, you receive a warning message, and the compiler makes the clock and register assignments. For example, the following combination of attributes (Figure 2-24) causes a warning because an IOC register cannot be clocked by a GLB clock.

```
REGTYPE=IOC
CLK=CLK2
```

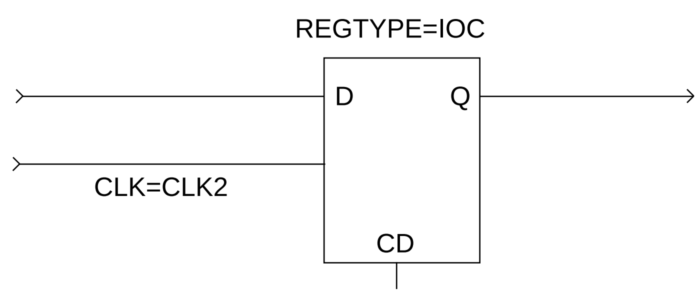


Figure 2-24. Conflicting REGTYPE and CLK Usage

Pin Attributes

The following Design Attributes are applied to the pins in your design:

- CRIT
- LOCK
- PULLUP
- SLOWSLEW

CRIT

The CRIT Design Attribute instructs the compiler to use the Output Routing Pool (ORP) bypass. Use CRIT for GLB outputs that require the least possible delay.

Synopsis

```
CRIT[=]pin_name
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

You can place CRIT only on output or bidirectional pins; CRIT attributes placed on nets are ignored. In the pLSI 1000 and 3000 device families, only two of four GLB outputs can use the ORP bypass. You may restrict routing and device resource utilization if you specify too many CRIT outputs in these device families.

Certain combinations of the CRIT and LOCK and/or CRIT and CLK attributes may cause a conflict, result in a warning message, and then one or more of those attributes may be ignored.

Example

In Figure 2-25, the CRIT and LOCK attributes require the two registers to be placed in the same GLB, but they require two different global clocks and cannot be placed into the same GLB. This is illegal. Remove one or more attributes for possible implementation. In this case, the CRIT attribute is ignored by the compiler.

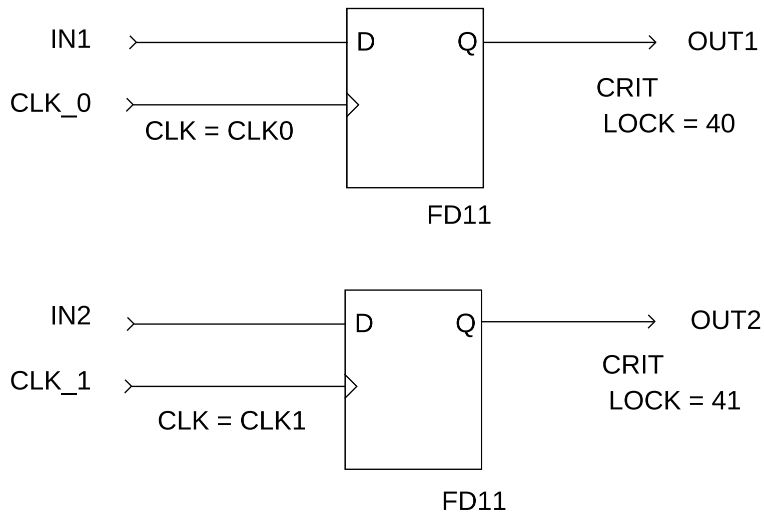


Figure 2-25. Conflicting Use of CRIT and LOCK Attributes in the ispLSI1032-80LJ84

LOCK

The LOCK Design Attribute assigns package pins to design I/O ports.

Synopsis

`LOCK[=]`*`pin_name`*`|`*`pin_number`*

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

The LOCK Design Attribute can be used to assign design I/O ports to specific package pins. Any redundant input pins which are not driving logic after logic optimization are removed along with their associated LOCK attributes.

Example

Figure 2-26 is an example of the correct use of the LOCK attribute.

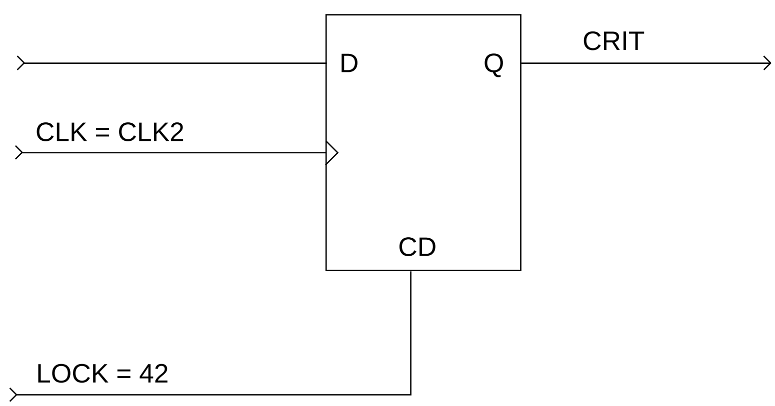


Figure 2-26. Correct Use of the LOCK Attribute

Certain combinations of LOCK attributes may result in a non-optimal design implementation. Figure 2-27 (A) is an example of improper use of the LOCK attribute; a single AND gate is locked to two different megablocks at the input and output sides. In this case, the output Z should be fed through a second GLB before it can connect to an IOC pin locked to a megablock other than that of pin A. Figure 2-27 (B) shows the resulting implementation.

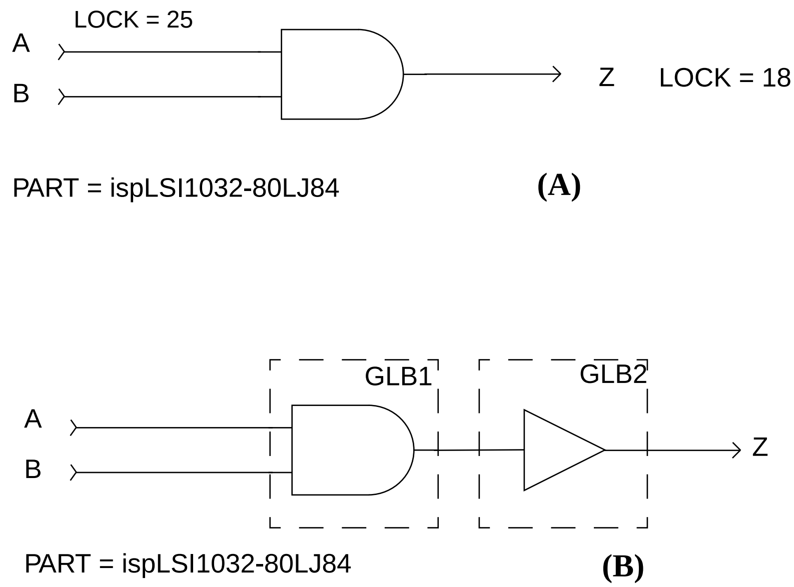


Figure 2-27. (A) Improper Use of the LOCK Attribute
(B) The Resulting Implementation

PULLUP

The PULLUP Design Attribute identifies an external pin to use the device pull-up feature.

Synopsis

```
PULLUP[=]pin_name
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

External pins can be pulled up individually by the PULLUP Design Attribute only if the Compiler Control Device Option PULLUP is set to OFF. Compiler Control Device Option PULLUP set to ON (default) activates the pull-up on all pins, regardless of Design Attribute PULLUP settings on ***individual*** pins.

SLOWSLEW

The SLOWSLEW Design Attribute identifies an external output pin to use slow slew rate.

Synopsis

```
SLOWSLEW[=]pin_name
```

See the appropriate “Third Party Vendor and pDS+ Design Environment User Manual” for the proper syntax for your design environment.

Description

The SLOWSLEW attribute can only be attached to output or bidirectional pins. Slew rate control is device dependent and is not available on all parts. Check the *Lattice Semiconductor ISP Encyclopedia* for availability.

Chapter 3 *Design Compilation and Control*

You can invoke the pDS+ Fitter (also referred to as “compiler”) from the command line of your system through the Design Process Manager (DPM). When you execute the pDS+ Fitter through DPM, all relevant files and parameters are passed to the appropriate pDS+ Fitter modules. Once a design routes, DPM merges the various report files into a single report file (*design_name*.rpt) and a log file (*design_name*.log) which contains messages, warnings, or errors issued by the pDS+ Fitter. If you interrupt the compilation process (for example, by pressing Control-C) these files may not be created, or some intermediate files may not be removed or may be incomplete.

Compiler Control Options influence control of the pDS+ Fitter and compilation of your design. These options are device-dependent, and determine how your design is partitioned, placed, and routed into the physical resources of the target device.

The Compiler Control Options perform the following functions:

- Determine the flexibility given to the compiler to complete your design
- Define the global objectives for design implementation
- Describe the allocation of physical device resources in support of specific Device Options
- Produce a physical netlist of your design

Usually you try to optimize your design for speed, for resource utilization, or for both. You can begin compiling with strict parameters that optimize your design goals. If the parameters are too restrictive for the available device resources, a compilation failure may occur. If this happens, try sets of parameters that gradually decrease the restrictions until you are able to successfully compile your design. If the compiler encounters an error, or if it determines that your design is not routable for any reason, the compile process ends. Check for problems in the report and log files.

The following sections describe the Compiler Control Options, DPM command-line options, and the Compiler Parameter File.

Compiler Control Options

Compiler Control Options can be specified from the command-line, through a Compiler parameter (.par) file, or in a design source file. If you specify a Compiler Control Option in more than one place, the precedence for implementation is command-line, Parameter File, and design description.

Table 3-1. Compiler Control Options

Compiler Control Option		**Description**
Parameter File Syntax	**Command Line Syntax**	
`CARRY_PIN_DIRECTION ON|OFF`	`-c`	Maintains user-specified pin direction in any simulation output. Default is OFF.
`CASE_SENSITIVE ON|OFF`	`-C`	Enables the compiler to treat identifiers, such as pin names and net names, as case-sensitive or case-insensitive. Default is OFF.
`EFFORT 1|2|3`	`-e [1..3]`	Provides a number of different optimization options. The larger the value, the longer the run-time. Default value is 2.
`EXTENDED_ROUTE ON|OFF`	`-q`	Instructs the router to use a complete routing cycle in an attempt to route a design (ON), or to question the user if routing time is very long (-q or OFF). Default is ON.
`IGNORE_FIXED_PIN ON|OFF`	`-l`	Instructs the compiler to ignore the pin locking attributes in your design. Default is OFF.
`NA`	`-if`	Specifies an input netlist format from the command line only. The values for input netlist formats are edif, laf, pla, or viewlogic. The default format is laf.
`ISP ON|OFF`	`NA`	Informs the compiler that you want to use the ISP pins on an ispLSI device. Default is OFF.

Table 3-1. Compiler Control Options (Continued)

Compiler Control Option		Description
Parameter File Syntax	**Command Line Syntax**	
`ISP_EXCEPT_Y2 ON|OFF`	`NA`	Allows the compiler to use the Y2 clock input for routing. Valid only for the ispLSI1016 and ispLSI2032 devices. Default is OFF.
`MAX_GLB_IN 2|..|24`	`-m [2..24]`	Specifies the maximum number of GLB inputs the compiler is allowed to use for each GLB. Input values are 2 to 18 for the 1000 and 2000 device families, and 2 to 24 for the 3000 device family. The default value is 16 for 1000 and 2000 device families, and 24 for the 3000 device family.
`MAX_GLB_OUT 1|2|3|4`	`-n [1..4]`	Specifies the maximum number of GLB outputs the compiler is allowed to used for each GLB. Default value is 4.
`OUTPUT_FORM` *format*	`-of` *format*	Specifies the output netlist format to be generated for post-route simulation. Values are edif, ldf, orcad, preroute_ldf, sim, verilog, vhdl, and viewlogic. There is no default for this option.
`PARAM_FILE` *file_name*	`-r` *file_name*	Specifies the name of a Parameter File for the compiler to use for compilation specifications. The Parameter File name *must* be different than the design name and have a .par extension.
`PART` *part_name*	`-p` *part_name*	Specifies the part name of the target device. The default part name is ispLSI1032-90LT100.
`PIN_FILE file_name`	`-y` *file_name*	Directs the compiler to back-annotate your design so design pins are fixed to specific package pins according to the pin assignments in the pin file.

Table 3-1. Compiler Control Options (Continued)

<table>
<tr><th colspan="2">Compiler Control Option</th><th rowspan="2">Description</th></tr>
<tr><th>Parameter File Syntax</th><th>Command Line Syntax</th></tr>
<tr><td>`PULLUP ON|OFF`</td><td>`NA`</td><td>Specifies global use of I/O pin pull-up resistors to the compiler. Default is ON.</td></tr>
<tr><td>`SECURITY ON|OFF`</td><td>`NA`</td><td>Influences the device security cell programming. However, this option does not guarantee the security cell is set or cleared, because device programmer options also affect the security cell. Default is OFF.</td></tr>
<tr><td>`STRATEGY AREA|DELAY`</td><td>`-s [a|d]`</td><td>Allows you to specify one of two optimization strategies: AREA or DELAY. Default is AREA.</td></tr>
<tr><td>`TIMING_ANALYZER VERBOSE|OFF`</td><td>`-ta [e|off]`</td><td>Generates a detailed timing analysis report, or turns off the Timing Analyzer. Default is ON.</td></tr>
<tr><td>`TIMING_FILE` *`file_name`*</td><td>`-t` *`file_name`*</td><td>Directs the compiler to generate a Viewlogic timing simulation wir file (*file_name*.1) with the OUTPUT_FORM VIEWLOGIC (-of viewlogic) option.</td></tr>
<tr><td>`USE_GLOBAL_RESET ON|OFF`</td><td>`-z`</td><td>Allows the compiler to move a global register reset signal to the global reset pin. Default is OFF.</td></tr>
<tr><td>`Y1_AS_RESET ON|OFF`</td><td>`NA`</td><td>The Y1/$\overline{\text{RESET}}$ pin is a global reset input when set to ON. The Y1/$\overline{\text{RESET}}$ pin is the Y1 clock input if set to OFF. This option applies only to ispLSI/pLSI 1016 devices and ispLSI/pLSI 2032 devices Default is ON.</td></tr>
</table>

Design Process Manager

The **dpm** command to invoke the Design Process Manager which manages the individual functions of the compilation process. Only one active compilation job can be running in a working directory at any given time. Running more than one compilation job from the same directory may result in a system error.

Synopsis

```
dpm [-c] [-e 1..3] [-i file_name]
    [-if edif|laf|pla|viewlogic]
    [-l] [-m 2..24] [-n 1..4]
    [-of edif|ldf|orcad|preroute_ldf|sim|verilog|vhdl|
     viewlogic] [-p part_name] [-q] [-r file_name]
    [-s a|d] [-t file_name] [-ta e|off] [-y file_name]
    [-z] [-C] [design_name]
```

Description

The following are examples and descriptions of the **dpm** command-line options. The command-line options for the **dpm** command are case sensitive.

-c **CARRY_PIN_DIRECTION**

CARRY_PIN_DIRECTION allows you to carry user-specified pin directions to any simulation output.

-e [1..3] **EFFORT**

EFFORT allows you to specify one of three optimization effort levels. The level values are 1 to 3, with a default of 2.

-i *file_name* **Input File**

Specifies an input file name in complete form.

Examples:

```
dpm -if edif -i my_design.edn
dpm -if laf -i my_design.laf
```

-if *format* **Input Netlist Format**

Specifies an input netlist format. The values for input netlist formats are edif, laf, pla, or viewlogic. The default format is laf.

Example:

```
dpm -if pla -i adder.pla
```

-l **IGNORE_FIXED_PIN**

IGNORE_FIXED_PIN causes the compiler to ignore the pin locking attributes (LOCK) in your design.

-m [2..24] **MAX_GLB_IN [2..24]**

MAX_GLB_IN limits the number of GLB inputs available to the compiler. MAX_GLB_IN input values are 2 to 18 for the 1000 and 2000 device families, and 2 to 24 for the 3000 device family. Default is 16 for 1000 and 2000 device families. Default is 24 for the 3000 device family.

-n [1..4] **MAX_GLB_OUT [1..4]**

MAX_GLB_OUT limits the number of GLB outputs available to the compiler. MAX_GLB_OUT output values are 1 to 4, with a default of 4.

-of *format* **OUTPUT_FORM**

Specifies an output netlist format. The values for output netlist formats are edif, ldf, orcad, preroute_ldf, sim, verilog, vhdl, and viewlogic. Several output formats can be specified by repeating **-of** *format* on the command line. There is no default for this option.

Example:

```
dpm -if laf -i adder.laf -of verilog -of edif
```

-p *part_name* **PART**

PART specifies the part number of the target device. The default part number is ispLSI1032E-90LT100. For a complete list of valid part numbers, see the *ISP Synario Starter Release Notes*.

-q

EXTENDED_ROUTE OFF

EXTENDED_ROUTE OFF instructs the router to question the user if routing time is very long. You can decide either to continue, or stop and relax some design constraints before trying to route again.

-r ***file_name***

PARAM_FILE

Directs the Design Process Manager to use the specified Compiler Parameter File for compilation specifications. The Parameter File name should have a .par extension and *must* be different than the design name. See “Compiler Parameter File” on page 71. for more information.

Example:

```
dpm -if edif -i adder.edn -r my_file.par
```

-s [a|d]

STRATEGY

STRATEGY allows you to specify one of the following two optimization strategies:

- AREA (a) – Minimum area (default)
- DELAY (d) – Minimum delay

-t ***file_name***

TIMING_FILE

When you compile your design, you can direct the pDS+ Fitter to generate a Viewlogic timing simulation wir file (timing.1) with the OUTPUT_FORM option. The TIMING_FILE option allows you to replace the default name “timing.1” with a different file name. You can save several different versions of the wir file using different TIMING_FILE file names, and simulate the wir files at a later time with a Viewlogic simulator.

If you specify the name of your output file to be the same as your design, you will receive an error message from the Design Analysis program if you run the compiler. To avoid receiving an error message, change the value of TIMING_FILE to a name other than your project name before you run the compiler.

Example:

```
dpm -if laf -i input.laf -of viewlogic -t my_file
```

The resulting file name is my_file.1. If you specify TIMING_FILE without a file name, the timing.1 file is not changed.

-ta [e|off] **TIMING_ANALYZER**

TIMING_ANALYZER allows you to generate an extended (detailed) timing analysis report, or turn off the Timing Analyzer.

- If you specify `-ta e`, DPM generates an extended (detailed) timing analysis report (*design_name*.dpt). If you do not specify `-ta e`, DPM generates a short timing analysis report which is appended to the compiler report (.rpt) file.
- If you specify `-ta off`, this turns off the Timing Analyzer.

-y *file_name* **PIN_FILE**

PIN_FILE allows you to receive pin assignments from a pin file.

Example:

```
dpm -if pla -i adder.pla -y adder
dpm -if pla -i adder.pla -l -y adder
```

In the second example, pin assignments in the netlist source file are ignored due to the **-l** option. The pin assignments from the pin file are then applied.

-z **USE_GLOBAL_RESET**

USE_GLOBAL_RESET allows the compiler to move a global register reset signal to the global reset pin.

-C **CASE_SENSITIVE**

CASE_SENSITIVE allows you to treat identifier names as case sensitive.

design_name **Viewlogic Design**

Specifies a design file name with no extension for Viewlogic designs. You must be in a Viewlogic working directory when using this option.

Example:

```
dpm -if viewlogic my_design
```

Compiler Parameter File

The Compiler Parameter File contains alternate sets of Compiler Control Options and Device Control Options that can be used to run different iterations of your design. You can create this simple text file using an ASCII text editor.

When a Parameter File is used, all relevant files and parameters are passed to the appropriate pDS+ Fitter software modules by the Design Process Manager (DPM). Once a design is routed, DPM merges the various report files into a report file, *design_name*.rpt, and a log file, *design_name*.log, containing messages, warnings, or errors issued by the DPM processes.

The following rules apply to Parameter Files:

- Each statement consists of an option name followed by the option value.
- Each statement in the file is on a separate line.
- Each set of parameters is terminated by an **END** statement.
- You can have an unlimited number of parameter sets in a file.
- The # symbol at the beginning of a line specifies comments.
- Unspecified options are replaced with default values or values from a design source file.
- The part number specified in the design source file is assumed, unless a part number is explicitly defined in each parameter set.
- The statements can use any mixture of uppercase and lowercase characters.
- The compiler will stop after running a successful set of parameters or if an error occurs. Any output generated corresponds to this successful run or the latest run.
- When two different part names relating to different packages are used in one Parameter File, any pin lockings can be ignored by specifying the IGNORE_FIXED_PIN ON option.
- There is no “=” between any attribute name and value.
- The Parameter File name should have a .par extension and *must* be different than the design name.

NOTE Do not use PARAM_FILE within a Compiler Parameter File. This could cause a loop in the program, which is not allowed. PARAM_FILE can be used in a design description source file.

Compiler Parameter File Example

The following is an example of a Compiler Parameter File.

```
MAX_GLB_IN 12
PART pLSI1016-60LJ44
END

# This begins the 2nd set of parameters
# Part number defaults to the original part specified
MAX_GLB_OUT 4
STRATEGY DELAY
OUTPUT_FORM VIEWLOGIC
TIMING_FILE your_design
TIMING_ANALYZER VERBOSE
END

# The final two examples use default values for unspecified parameters.
# Part number defaults to the original part specified
MAX_GLB_IN 14
MAX_GLB_OUT 3
END

# New part specified
PART ispLSI1032-80LJ84
IGNORE_FIXED_PIN ON
TIMING_FILE my_design
TIMING_ANALYZER VERBOSE
END
```

NOTE If you are using a Compiler Parameter File, the compiler stops at the first successful set of parameters or if an error occurs.
It is usually a good idea to place the most restrictive sets of parameters at the beginning of the file.

Parameter File Options

The following are examples and descriptions of the syntax and values for Compiler Control Options that can be used in a Parameter File, or in a design description source file.

CARRY_PIN_DIRECTION

The CARRY_PIN_DIRECTION option maintains user-specified pin directions in any simulation output. Default is OFF.

Synopsis

```
CARRY_PIN_DIRECTION ON|OFF
```

Description

- CARRY_PIN_DIRECTION ON attempts to maintain user-specified pin directions for 3-state outputs into any simulation output netlist. The 3-state outputs can be connected to external output pins or bidirectional pins.
- CARRY_PIN_DIRECTION OFF converts 3-state outputs to external output pins in any simulation output netlist.

CASE_SENSITIVE

The CASE_SENSITIVE option enables the compiler to treat identifiers, such as pin names and net names, as case-sensitive or case-insensitive. The default is OFF.

Synopsis

```
CASE_SENSITIVE ON|OFF
```

Description

- CASE_SENSITIVE ON results in consideration of identifiers as case-sensitive.
- CASE_SENSITIVE OFF results in consideration of identifiers as case-insensitive.

EFFORT

The EFFORT option provides a number of different optimization options.

Synopsis

```
EFFORT 1|2|3
```

Description

EFFORT has a range of 1 to 3. The larger the effort, the larger the run-time and memory requirement. While a higher effort level usually leads to better results, this is not guaranteed. Different effort levels should be tried to find the best result for a specific design. The default value is 2.

EXTENDED_ROUTE

The EXTENDED_ROUTE option instructs the router to use a complete routing cycle in an attempt to route a design, or question the user if routing time is very long. The default value is ON.

Synopsis

```
EXTENDED_ROUTE ON|OFF
```

Description

- EXTENDED_ROUTE ON instructs the router to continue until the design is fully routed, or until routing fails.
- EXTENDED_ROUTE OFF instructs the router to question the user if routing time is very long. You can decide either to continue, or stop and relax some design constraints before trying to route again.

IGNORE_FIXED_PIN

The IGNORE_FIXED_PIN option instructs the compiler to either ignore or honor the pin locking attributes in your design.

Synopsis

```
IGNORE_FIXED_PIN ON|OFF
```

Description

- IGNORE_FIXED_PIN ON causes the compiler to ignore the pin locking attributes (LOCK) on your design. This allows the compiler to place I/Os anywhere on the device, without restriction.
- IGNORE_FIXED_PIN OFF causes the compiler to honor any LOCK attributes on your design. The default value is OFF.

MAX_GLB_IN

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

Synopsis

```
MAX_GLB_IN 2|..|24
```

Description

MAX_GLB_IN has a range of 2 to 18 for the 1000 and 2000 device families, and 2 to 24 for the 3000 device family. The default value is 16 for 1000 and 2000 device families, and 24 for the 3000 device family.

Specifying MAX_GLB_IN 18 results in the maximum use of device resources in the 1000 and 2000 device families. This usually results in the minimum-level implementation of any wide logic. However, it also causes an increase in placement and routing difficulties for the design. Placement and routing may then take more time or may fail. Specifying MAX_GLB_IN 12 to 14 often produces results similar to specifying MAX_GLB_IN 18, but requires less time to route.

Specifying MAX_GLB_IN 24 results in the maximum use of device resources in the 3000 device family. This takes advantage of the large number of inputs in the device for a minimum-level implementation of logic. Exercise caution when lots of wide-input logic exists in a design and a large MAX_GLB_IN value is used. A large MAX_GLB_IN value can easily consume inputs common between GLBs in a Twin GLB (3000 device family), requiring some GLB outputs to remain unused. This can result in an unfittable or unroutable design.

MAX_GLB_IN does not limit inputs to GLBs that accommodate clock logic, hard macros, or protected logic.

✍ ***NOTE***	Use smaller values for MAX_GLB_IN to increase routability. However, this may also increase the levels of delay and decrease resource utilization.

MAX_GLB_OUT

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

Synopsis

```
MAX_GLB_OUT 1|2|3|4
```

Description

MAX_GLB_OUT has a range of 1 to 4, with a default value of 4. Specifying MAX_GLB_OUT 4 results in the maximum use of device resources. However, it also causes the placement and routing program to work harder and take more time, and may result in an unroutable design. Use a value of 2 or 3 to improve routability.

OUTPUT_FORM

The OUTPUT_FORM option specifies the output netlist format to be generated for post-route simulation.

Synopsis

```
OUTPUT_FORM EDIF|LDF|ORCAD|PREROUTE_LDF|SIM|VERILOG|
            VHDL|VIEWLOGIC
```

Description

The possible values are EDIF, LDF, ORCAD, PREROUTE_LDF, SIM, VERILOG, VHDL, and VIEWLOGIC. There is no default value for this option.

- EDIF – Generates an EDIF 2 0 0 format netlist (*design*.edo) file.
- LDF – Generates a *design*.pdf file, representing the post-route design if the design routes. The LSC Design Format (LDF) file can be imported in to the Lattice Semiconductor pDS software package. The pDS software allows you to manually partition the logic in a device to possibly improve routability. This is useful if a design requires significant manual intervention.
- ORCAD – Generates an OrCAD INF format netlist (*design*.ifo) file and an OrCAD delay back-annotation (DBA) file that can be used for post-route simulation with OrCAD's VST simulator.
- PREROUTE_LDF – Generates a *design*.ldf file, representing the pre-route partitioned logic design. The LSC Design Format (LDF) file can be imported in to the Lattice Semiconductor pDS software package. The pDS software allows you to manually partition the logic in a device to possibly improve routability. This is useful if a design requires significant manual intervention.
- SIM – Generates a *design*.sim file for timing analysis with the pDS+ Timing Analyzer and board-level simulation with Synopsys Logic Modeling Division models.
- VERILOG – Generates an SDF format netlist (*design*.sdf) file as well as a Verilog format netlist (*design*.vlo) file for use with any Verilog compatible or SDF-compatible simulator.
- VHDL – Generates two generic VHDL format netlist files for maximum delay (*design*.vho) and minimum delay (*design*.vhn), and a VITAL VHDL format netlist (*design*.vto) file for use with any VHDL-compatible simulator.
- VIEWLOGIC – Generates both a Viewlogic wir file (timing.1) to use with Viewlogic simulators and a *design*.sim file for board-level simulation. In a Viewlogic design environment, the *design*.sim file is placed in your current working directory, and the timing.1 file is placed in your wir directory. The Viewlogic **edifneti** and **edifneto** utilities must be present for this output option to work correctly. This option can be used with TIMING_FILE to replace the "timing.1" default with a different file name. See TIMING_FILE for more information.

> **NOTE** Setting OUTPUT_FORM to VIEWLOGIC with TIMING_FILE results in the generation of both the *file_name*.1 and *design_name*.sim files. Setting OUTPUT_FORM to SIM results in the generation of the design_name.sim file only. See TIMING_FILE for more information.

The OUTPUT_FORM option can specify several output netlist formats by specifying the formats on the same line separated by commas. For example:

```
OUTPUT_FORM LDF, SIM
```

PARAM_FILE

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

Synopsis

PARAM_FILE *file_name*

Description

The Compiler Parameter File contains alternate sets of Compiler Control Options and Device Control Options that can be used to run different iterations of your design. You can create this file using an ASCII text editor. The Parameter File name should have a .par extension and *must* be different than the design name. See “Compiler Parameter File” on page 71 for more information.

> **NOTE** Do not use PARAM_FILE within a Compiler Parameter File. This could cause a loop in the program, which is not allowed. PARAM_FILE can be used in a design description source file.

PART

The PART option specifies the part number of the target device. The default part number is ispLSI1032E-90LT100.

Synopsis

PART *part_number*

Description

When using the PART option, you must enter the part number exactly as shown in the Part Number column of the tables provided in the *ISP Synario Starter Release Notes*.

Example

```
PART pLSI1032-80LJ84
PART ispLSI1032-80LJ84
```

PIN_FILE

The PIN_FILE option directs the compiler to read a pin file with pin assignments. Design pins are then fixed to specific package pins according to the pin assignments in the pin file.

Synopsis

PIN_FILE *file_name*

Description

Any pin not specified in the pin file remains unaltered as assigned in the input netlist. To ignore pin lockings in the input netlist before making pin assignments according to the pin file, use IGNORE_FIXED_PIN ON.

A pin file consists of any number of lines, each line conforming to the following syntax:

<*pin_name*> <*pin_direction*> <*pin_number*>

- **pin_name** is the external pin name.
- **pin_direction** is IN, OUT, BIDI, or SYS. Lines with *pin_direction* identified as SYS are ignored.
- **pin_number** is the package pin number.

STRATEGY

The STRATEGY option allows you to specify one of the following two optimization strategies:

- AREA – Maximum resource utilization
- DELAY – Maximum performance

Synopsis

```
STRATEGY AREA|DELAY
```

Description

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

- For logic level considerations, DELAY offers the least number of logic levels.
- AREA optimizes for device utilization and consequently may use more logic levels.

Use STRATEGY AREA during the first attempt in implementing a design. This option provides a good compromise between resource utilization and routability, as well as global optimization of a design. Use STRATEGY DELAY and other Design Attributes to refine the implementation of a design.

STRATEGY AREA normally leads to more routable designs, especially if used with moderate values for MAX_GLB_IN and MAX_GLB_OUT attributes. In this case, the compiler may insert feed-through buffers to resolve any user conflicts or to remove any routing congestion. Use SCP/ECP and SAP/EAP attributes or STRATEGY DELAY to control the behavior of the compiler.

While STRATEGY AREA usually leads to better resource utilization, and STRATEGY DELAY usually leads to better levels of delay, these methods are not exact, are highly design-dependent, and can potentially lead to unexpected results. Try different alternatives to refine the design implementation, such as using different global optimization strategies (AREA and DELAY) or making local refinements by trying different Design Attributes (SCP/ECP, etc.).

TIMING_FILE

When you compile your design, you can direct the pDS+ Fitter to generate a Viewlogic timing simulation wir file (timing.1) with the OUTPUT_FORM option.

Synopsis

```
OUTPUT_FORM VIEWLOGIC
TIMING_FILE file_name
```

Description

The TIMING_FILE option allows you to replace the default name “timing.1” with a different file name. You can save several different versions of the wir file using different TIMING_FILE file names, and simulate the wir files at a later time with a Viewlogic simulator.

If you specify the name of your output file to be the same as the name of your design, you will receive an error message from the Design Analysis program when you run the compiler. To avoid receiving an error message, change the TIMING_FILE value to a name other than your project name before you run the compiler.

Example

```
OUTPUT_FORM VIEWLOGIC
TIMING_FILE my_design
```

The resulting file name in this example is my_design.1.

USE_GLOBAL_RESET

The USE_GLOBAL_RESET option makes the global reset pin available for use by the compiler. Default is OFF.

Synopsis

```
USE_GLOBAL_RESET ON|OFF
```

Description

If USE_GLOBAL_RESET is set to ON, and all registers and latches in the design are driven by a common direct (no-logic) reset line, that line is moved to a global reset pin and is cleared from the generated output netlist. This can eliminate a high-fanout net and improve routability of the design. However, this may require inversion of the reset signal outside the device, depending on the polarity of the reset signal and the global reset pin.

By using this option in conjunction with the REGTYPE IOC Design Attribute, some registers and latches which are normally assigned to a GLB because of the presence of a product term reset can be moved to an IOC by the compiler. See “REGTYPE” in Chapter 2 for more information.

Device Control Options

Device Control Options are a subset of Compiler Control Options that you can use in your design for the final device description to relay availability and allocation of resources in the physical device.

ISP

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

Synopsis

```
ISP ON|OFF
```

Description

If you are using the ISP capability, set this option to ON. The ISP option requires four input pins. Setting this option to OFF makes the four ISP pins (SCLK, SDI, SDO, and MODE) available for routing as dedicated input pins, and the router can then assign signals to these pins.

This option is ignored if you specify a non-ISP device. You can remove the ISP option to improve resource availability. See the *Lattice Semiconductor ISP Encyclopedia* for device-specific ISP pin numbers.

NOTE See ISP_EXCEPT_Y2 on the following page for information concerning the use of ISP pins on the ispLSI 1016 and the ispLSI 2032 devices.

ISP_EXCEPT_Y2

The ISP_EXCEPT_Y2 option allows the software to use the Y2 clock input for routing, which increases resource utilization.

Synopsis

```
ISP_EXCEPT_Y2 ON|OFF
```

Description

This option is valid only for the ispLSI 1016 and the ispLSI 2032 devices, and is ignored if you choose any other device. The default value is OFF.

If ISP_EXCEPT_Y2 OFF and ISP ON are specified, you cannot use the Y2 input as a clock input pin in your design; it is used only for ISP. If ISP_EXCEPT_Y2 ON is specified, the compiler may use the Y2 pin as a clock input pin; you will need to externally multiplex the ISP SCLK signal and the Y2 clock input.

Example

Figure 3-1 shows a typical Y2 clock multiplexing scheme for an ispLSI1016 device.

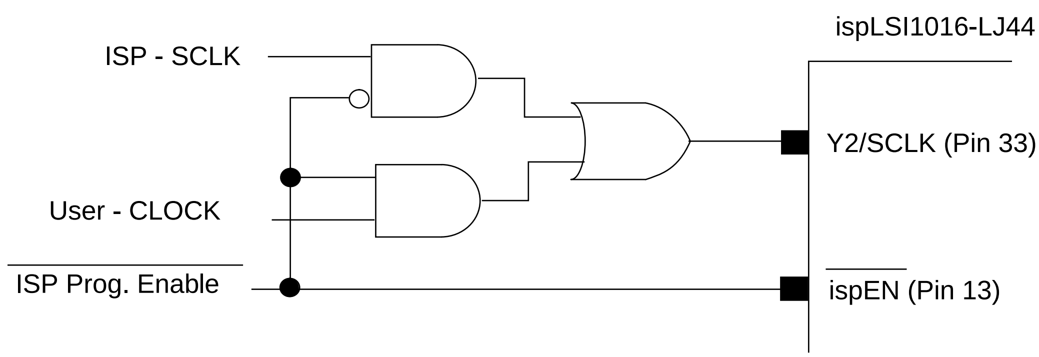


Figure 3-1. Typical Y2/SCLK Multiplexer

NOTE For the ispLSI1016-LJ44 and ispLSI2032-LJ44 devices, when the ISP mode is enabled (ispEN, pin 13, is LOW), pin 33 is defined as SCLK, which is used to shift in-programming information. When the ISP mode is disabled (pin 13 is HIGH), pin 33 can be used as a clock input to your design if ISP_EXCEPT_Y2 ON is specified.

PULLUP

The PULLUP option specifies the use of I/O pin pull-up resistors to the compiler. The default value is ON.

Synopsis

```
PULLUP ON|OFF
```

Description

PULLUP ON places active weak pull-ups on all I/O pins. PULLUP OFF places pull-ups only on unused I/O pins. The PULLUP parameter has no effect on routing or resource utilization. If PULLUP OFF is specified, individual pins can be pulled up by using the PULLUP Design Attribute.

> ✍ ***NOTE*** The PULLUP on the ispEN, RESET, and TOE pins are always on.

SECURITY

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

Synopsis

```
SECURITY ON|OFF
```

Description

SECURITY ON inserts a “1” in the G field of the JEDEC file to inform the device programmer that the device is to be secured. Most device programmers require that another option be set in the programmer software to secure the device if the G field is set to “1.”

With the device security cell programmed ON, you can reprogram the device, but you cannot read its contents. The security cell cannot be cleared except by erasing the entire device.

SECURITY OFF inserts a “0” in the G field of the JEDEC file, which prevents the programmer from securing the device. However, many programmers have the capability to manually secure the device, even if this value is set to OFF.

Y1_AS_RESET

The Y1_AS_RESET option determines how the Y1/$\overline{\text{RESET}}$ pin is used on ispLSI/pLSI 1016 devices and ispLSI/pLSI 2032 devices. The default value is ON.

Synopsis

```
Y1_AS_RESET ON|OFF
```

Description

The Y1/$\overline{\text{RESET}}$ pin is a global reset input if Y1_AS_RESET ON. The Y1/$\overline{\text{RESET}}$ pin is the Y1 clock input if Y1_AS_RESET OFF. This option applies only to ispLSI/pLSI 1016 devices and ispLSI/pLSI 2032 devices and is ignored for all other devices.

You cannot lock a signal to the Y1/$\overline{\text{RESET}}$ input pin. If Y1_AS_RESET ON, the Global Reset signal is automatically connected to the Y1/$\overline{\text{RESET}}$ pin, and you will get an error if you try to lock a signal to this pin.

> ✍ ***NOTE*** If Y1_AS_RESET is set to OFF in ispLSI/pLSI 1016 or ispLSI/pLSI 2032 devices, no registers can be globally reset. Specify any required reset signal as PT reset.

Timing Analyzer Control Option

The Timing Analyzer Control Option is a subset of the Compiler Control Options that enables you to control the pDS+ Timing Analyzer.

TIMING_ANALYZER

The TIMING_ANALYZER option allows you to generate a detailed (expanded) timing analysis report, or turn off the Timing Analyzer. The default is ON, and the default report is a short timing analysis report which is appended to the report (.rpt) file generated by the compiler.

Synopsis

```
TIMING_ANALYZER VERBOSE|OFF
```

Description

The Timing Analyzer uses the .sim file generated by the pDS+ Fitter as input. The timing information for the device part in the sim file must exist in order to perform timing analysis.

- TIMING_ANALYZER VERBOSE generates a detailed (expanded) timing analysis report (*design_name*.dpt) in addition to a short timing report which is appended to the compiler report (.rpt) file.
- TIMING_ANALYZER OFF turns off the timing analyzer.

Chapter 4 *Timing Analysis*

The pDS+ Fitter has a built-in static timing analyzer which provides accurate pin-to-pin timing information for your design. The Timing Analyzer identifies critical paths, including longest and shortest paths, calculates chip boundary setup and hold times, and calculates maximum clock frequency.

This chapter contains the following information:

- An overview of timing analysis
- Features and reports of the pDS+ Timing Analyzer

Timing Analysis Overview

Timing analysis enables you to analyze the performance of your design after compilation. Timing analysis traces all signal paths, determines critical (timing) paths, and finds paths that limit the performance of your design.

Path Analysis

Paths normally start at primary input ports, bidirectional ports, or output pins of registers, and end at primary output ports, bidirectional ports, data pins of registers, and clock pins of the registers. There are four types of timing paths:

- Primary input to register – A path begins at the primary input port or bidi port and ends at the data pin or clock pin of a register.
- Register to Register – A path begins at an output pin of a register and ends at a data pin or clock pin of a register.
- Register to primary output – A path begins at an output pin of a register and ends at primary output ports or bidi ports.
- Primary input to primary output – A path begins at a primary input port or bidi port and ends at a primary output port or bidi port.

The Timing Analyzer calculates the delay of a path by tracing from the starting point of the path to its ending point, cumulatively adding delays along the way. The longest path is the path which has the largest delay from start point to end point. The shortest path is the path which has the smallest delay from start point to end point. The Timing Analyzer also considers the maximum delay of registers for longest path and minimum delay of registers for shortest path calculations.

- Maximum Delay – Maximum clock to output delay of register.
- Minimum (Register) Delay – Minimum clock to output delay of register.

> ✍ ***NOTE*** Paths starting from power (Vcc) or ground (GND) connections are not considered since these connections do not propagate transitions.

Longest and Shortest Path Example

The example in Figure 4-1 illustrates longest and shortest path calculation. For this example, the following information is assumed:

- The inverter gate g1 has a delay of 20 ns.
- The AND gate g2 has a delay of 40 ns.
- The AND gate g3 has a delay of 30 ns.
- The OR gate g4 has a delay of 30 ns.

Based on these assumption the calculations for longest path and shortest path from starting point A to ending point F are the following:

- The longest path from A to F:

  `A->g1->D->g2->H->g4->F = 90`
- The shortest path from A to F:

  `A->g3->E->g4->F = 60`

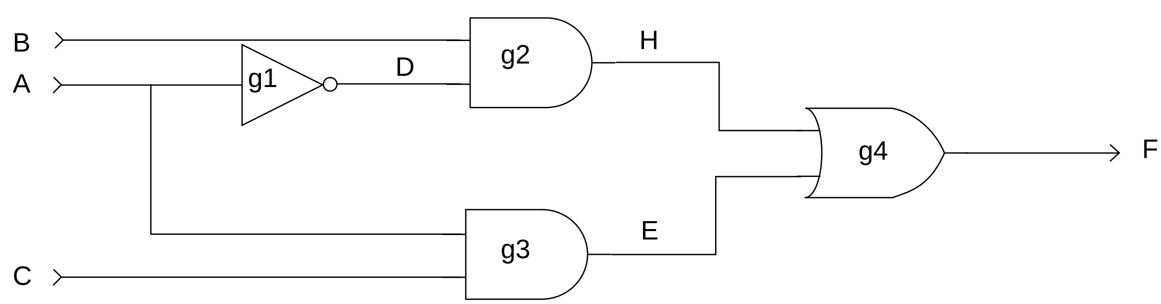


Figure 4-1. Longest Path and Shortest Path Example

Path Analysis Example

Figure 4-2 is an example of a small design. All the design objects are pin, port, net and instance. Each instance has input pins and output pins. Each net has a driver pin (port) and driven pins (ports). A pin is the connection point for the net and the instance.

For the example in Figure 4-2, the following information is assumed:

- The longest path and the shortest path delay from IN1 to D1 are both 100.
- The longest path and the shortest path delay from IN2 to D1 are both 40.
- The longest and the shortest path delay from CLK to C1 are both 20.
- The longest and the shortest path delay from CLK to C2 are both 20.
- The longest and the shortest path delay from IN3 to D2 are both 50.
- The longest and the shortest path delay from Q1 to D2 are both 90.
- The longest and the shortest path delay from Q2 to OUT1 are both 30.
- The longest and the shortest path delay from IN3 to OUT2 are both 80.

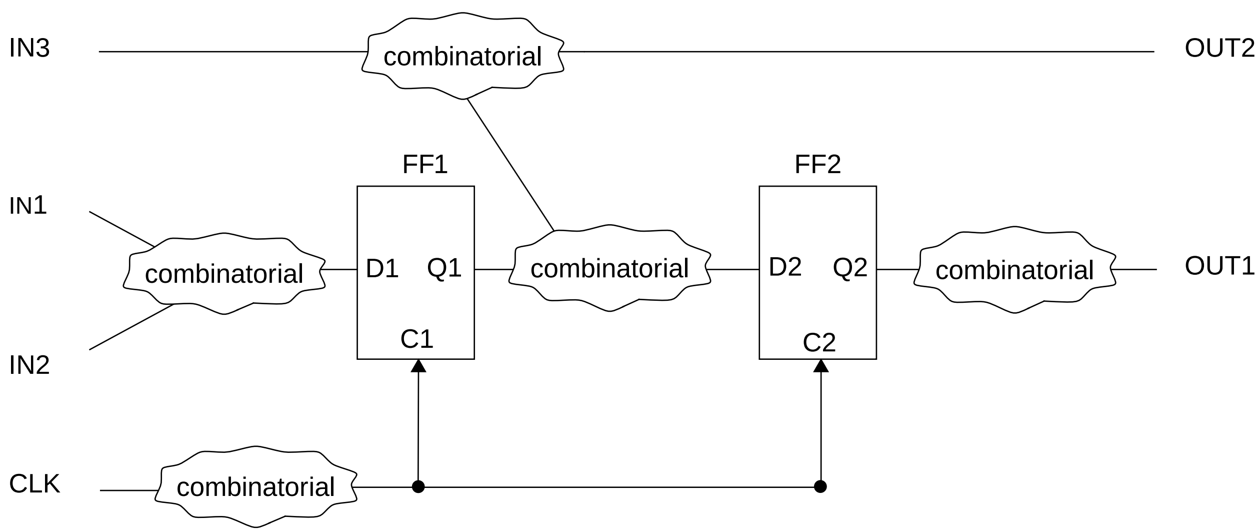


Figure 4-2. Path Analysis Example

For the above example, the timing analysis results would be the following:

```
Longest Paths:

 Startpoint           Endpoint           Path Delay
  IN1                   D1                  100
  Q1                    D2                   90
  IN3                   OUT2                 80
  Q2                    OUT1                 30
  CLK                   C1                   20
  CLK                    C2                   20

Shortest Paths:

 Startpoint           Endpoint           Path Delay
  CLK                   C1                   20
  CLK                   C2                   20
  Q2                    OUT1                 30
  IN2                   D1                   40
  IN3                   D2                   50
  IN3                    OUT2                 80
```

The example in Figure 4-2 is a very simple design; however, in a more complex design, loops may exist. If loops exist in a design, all related loops are broken and then the longest path delay calculation is done. Shortest path delay calculation can be done without breaking loops in the design.

Setup and Hold Time Evaluation

Timing analysis determines the setup and hold times for registers from the ports of a design by examining all destination registers and storing the longest and shortest delay paths between all ports connected to the D or CLK inputs of flip-flops and latches.

Setup Time

Setup time is the length of time a data signal must be stable before the active edge of a CLK (see Figure 4-3). Setup time for primary input ports is calculated as follows:

```
setup time = (longest data path delay) - (shortest clock
              path delay) + (setup time of register)
```

Because of differences in the longest data path and the shortest clock path, it is possible for setup time to be negative.

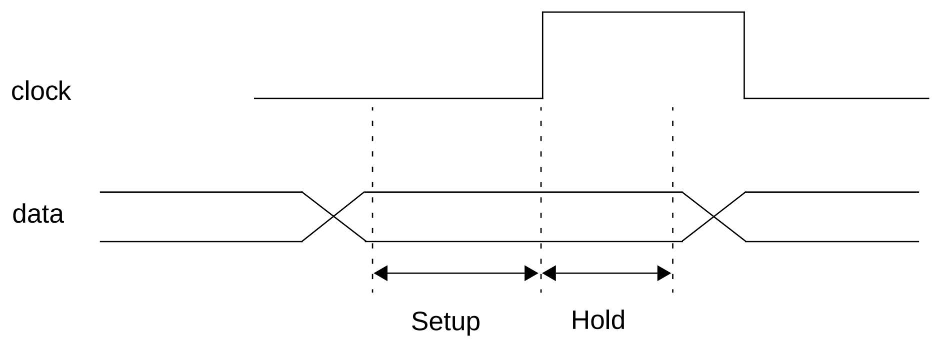


Figure 4-3. Setup and Hold Time

Hold Time

Hold time is the length of time a data signal must remain stable after the active edge of a CLK (see Figure 4-3). Hold time for primary input ports is calculated as follows:

```
hold time = (clock path delay) - (shortest data path
             delay) + (hold time of register)
```

Because of differences in the longest clock path and the shortest data path, it is possible for hold time to be negative.

Setup and Hold Time Example

Figure 4-4 is an example to illustrate the setup and hold time calculation for primary input ports. For this example, the setup time and hold time of each register is 10.

Setup and hold time for register FF1 is calculated as follows:

- The setup time of port IN1 to port clk is: 100 - 20 + 10 = 90
- The hold time of port IN1 to port clk is: 20 - 100 + 10 = -70
- The setup time of port IN2 to port clk is: 40-20+10 = 10
- The hold time of port IN2 to clk is: 20 - 40 + 10 = -10

Because there is no port that drives the data pin of register FF2 directly, there is no setup time and hold time requirement for FF2 from primary input ports.

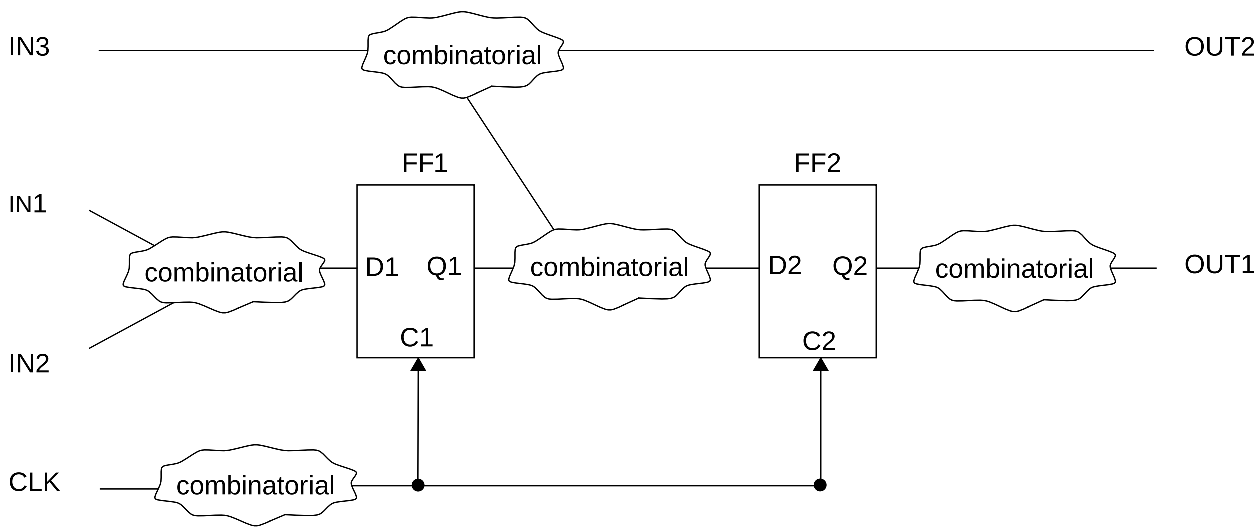


Figure 4-4. Setup and Hold Time Example

pDS+ Timing Analyzer

The pDS+ Timing Analyzer enables you to obtain the following information.

- Analyzes the longest/shortest delay path between multiple source and destination nodes
- Calculates setup and hold time and boundary register for flip-flops and latches
- Determines maximum frequency for clocking a design containing one or more flip-flops and/or latches

Frequency Calculation

The performance of a circuit is usually measured by frequency. The larger the frequency, the faster the circuit.

The Timing Analyzer first examines the D inputs of all destination registers looking for the Q outputs of source registers. If it finds a Q output, the Timing Analyzer adds the propagation delay through the source register and the setup time for the register to the delay path time, and stores the resulting value. After the Timing Analyzer has examined all registers, the maximum delay time of all values calculated will be the minimum clock period value. This value, in turn, gives the maximum allowable clock frequency.

NOTE An accurate frequency calculation by the Timing Analyzer assumes the registers in the design are controlled by a single clock. Frequency calculation may be skewed if the registers in the design are controlled by a gated clock.

No frequency calculation is done by the Timing Analyzer if one of the following two conditions exists:

- A design has no register.
- A design has registers, but it has only one level register.

Frequency Calculation Example

In Figure 4-5, the setup time, hold time, and delay from clock pin to output pin of each register is 10.

In this example, the longest path delay form Q1 to D2 is 90, the delay from C1 to Q1 of FF1 is 10, and the setup time of FF2 is 10. The calculation for minimum clock period would then be 10 + 90 + 10 = 110.

The frequency is then calculated as follows:

```
frequency = 1/minimum clock period
          = 1/110
```

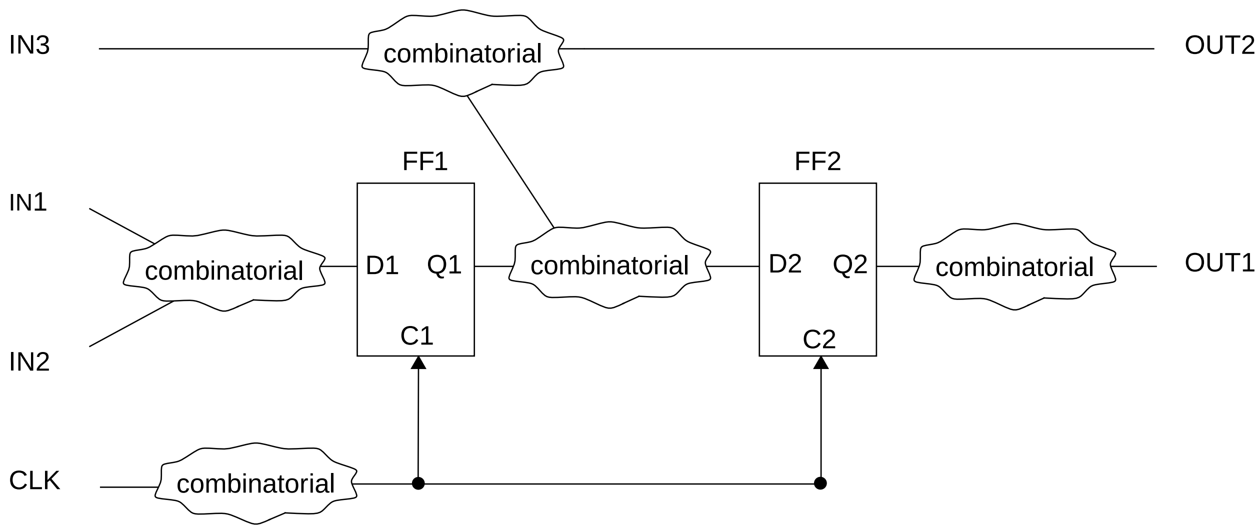


Figure 4-5. Frequency Calculation Example

Timing Analyzer Options

The Timing Analyzer uses the .sim file generated by the pDS+ Fitter as input. The timing information for the device part in the sim file must exist to perform timing analysis. The Timing Analyzer can be controlled by using command-line or parameter file options.

Command Line Syntax

The DPM Timing Analyzer option, `-ta e|off`, allows you to generate an extended (detailed) timing analysis report, or turn off the pDS+ Timing Analyzer.

- If you specify `-ta e`, the compiler generates an extended (detailed) timing analysis report (*design_name*.dpt). If you do not specify `-ta e`, the compiler generates a short timing analysis report which is appended to the compiler report (.rpt) file.
- If you specify `-ta off`, the compiler does not run the Timing Analyzer.

Parameter File Syntax

The Timing Analyzer options can also be specified in a Parameter File. The parameter file option, TIMING_ANALYZER VERBOSE|OFF, allows you to generate an extended (detailed) timing analysis report, or turn off the pDS+ Timing Analyzer.

- If you specify TIMING_ANALYZER VERBOSE, the compiler generates a detailed (expanded) timing report in addition to a short timing report which is appended to the report file (.rpt) generated by the compiler.
- If you specify TIMING_ANALYZER OFF, the compiler does not run the timing analyzer.

Timing Analyzer Report Files

Both the short report file and the detailed report file from the Timing Analyzer contain the following three parts:

- Frequency calculation
- Path analysis
- Setup and hold time evaluation

In addition to these three parts, each report file contains the timing information for flip-flops and latches when applicable.

The difference between a short report file and a detailed report file is:

- A short report file is appended to the report (*design_name*.rpt) file generated by the compiler and contains the starting point, ending point, and delay value for a long or short path.
- A detailed report (*design_name*.dpt) file is separate and contains the starting point, intermediate point, and accumulated delay value from the starting point to the intermediate point and the ending point for a long or short path.

Short Report Example

A short report is automatically generated by the timing analyzer when the compiler is invoked and appended to the output report (*design_name*.rpt) file from the compiler.

- The starting point for a path can be one of the following:
  - Primary input port. The timing analyzer reports the port name followed by “[in].”
  - Bidirectional port. The timing analyzer reports the bidi port name followed by “[bidi].”
  - The output of a flip-flop or latch. The timing analyzer reports the instance name followed by a “/,” followed by the output pin name, followed by “[inst_primitive name].”
- The ending point for a path can be one of the following:
  - Primary output port. The timing analyzer reports the port name followed by “[out].”
  - Bidi port. The timing analyzer reports the bidi port name followed by “[bidi].”
  - The data pin or clock pin of a flip-flop or latch. The timing analyzer reports the instance name followed by “/,” followed by the pin name, followed by “[inst_primitive name].”
- The path delay is the longest path or the shortest path delay value from the starting point to the ending point.
- For setup and hold information, the first column is the name of flip-flip or latch, the second column is the port name which drives the data pin of the flip-flop or latch, the third column is the port name which drives the clock pin of the flip-flop or latch. the fourth column is the setup value, and the fifth column is the hold value.

An example of a short report starts on the following page.

```
Timing Analysis - Short Report
------------------------------

Maximum Operating Frequency:    87 MHz

The following path determines the frequency:

  Startpoint: __0_$1N23/Q0
              (edge-triggered flip-flop)
  Endpoint: __0__LAF_G/D0
              (edge-triggered flip-flop)

         Point
  Name/pin_name [type]                      Delay      Path
  ---------------------------------------------------------
  __0_$1N23/Q0 [PGDFFR]                     0.00       0.00
  __0_$1N23_grp/A0 [PRBUFO1]                0.00       0.00
  __0_$1N23_grp/Z0 [PRBUFO1]                1.50       1.50
  __0_D6_P_LC_I11/A0 [PGBUFI]               0.00       1.50
  __0_D6_P_LC_I11/Z0 [PGBUFI]               1.00       2.50
  __0_D6_P_LC_P13/A0 [PGANDD2]              0.00       2.50
  __0_D6_P_LC_P13/Z0 [PGANDD2]              2.50       5.00
  __0_D6_P_LC_P13_xa/A0 [PGBUFXA]           0.00       5.00
  __0_D6_P_LC_P13_xa/Z0 [PGBUFXA]           1.50       6.50
  __0_D6_P_LC_X0MO/A0 [PGBUFXI]             0.00       6.50
  __0_D6_P_LC_X0MO/Z0 [PGBUFXI]             1.50       8.00
  __0_D6_P_LC_X0O/A1 [PGXOR2]               0.00       8.00
  __0_D6_P_LC_X0O/Z0 [PGXOR2]               0.50       8.50
  __0__LAF_G/D0 [PGDFFR]                    0.00       8.50
  ---------------------------------------------------------

The clock period is 11.50.
clock period = path delay + clock-to-output delay + setup time
  path delay: 8.50
  clock-to-output delay: 1.00
  setup time: 2.00


Information for flip-flop:

  Data-to-output delay:            0.00
  Clock-to-output delay:           1.00
  User reset-to-output delay:      2.50
  Global reset-to-output delay:    2.50
  Setup time:                      2.00
  Hold time:                       3.50
  Pulse-width time:                5.00
```

```
Longest Paths:

     Startpoint                     Endpoint                  Path Delay
  Name/pin_name [type]         Name/pin_name [type]
  ------------------------------------------------------------------------
  A [in]                         __0_$1N21/D0 [PGDFFR]          10.50
  C [in]                         __0_$1N23/D0 [PGDFFR]          10.50
  __0_$1N23/Q0 [PGDFFR]          __0__LAF_G/D0 [PGDFFR]         8.50
  D [in]                         __0_$1N23/CLK [PGDFFR]         6.50
  E [in]                         __0__LAF_G/CLK [PGDFFR]        6.50
  __0__LAF_G/Q0 [PGDFFR]         G [out]                        5.50
  B [in]                         __0_$1N21/CLK [PGDFFR]         5.50


Shortest Paths:

     Startpoint                     Endpoint                  Path Delay
  Name/pin_name [type]         Name/pin_name [type]
  ------------------------------------------------------------------------
  D [in]                         __0_$1N23/CLK [PGDFFR]         4.50
  E [in]                         __0__LAF_G/CLK [PGDFFR]        4.50
  __0__LAF_G/Q0 [PGDFFR]         G [out]                        5.50
  B [in]                         __0_$1N21/CLK [PGDFFR]         5.50
  __0_$1N23/Q0 [PGDFFR]          __0__LAF_G/D0 [PGDFFR]         8.50
  A [in]                         __0_$1N21/D0 [PGDFFR]          10.50
  C [in]                         __0_$1N23/D0 [PGDFFR]          10.50


Required Setup and Hold:

  Inst.Name                  Data         Clock        Setup        Hold
  ------------------------------------------------------------------------
  __0_$1N23                  C            D            8.00         -0.50
  __0_$1N21                  A            B            7.00         -1.50
```

Detailed Report Example

A detailed report is generated by using the `-ta e` option when the compiler is invoked. The output is a separate detailed report file (*design_name*.dpt).

- For a detailed report, the first column lists all the points in a path from starting point to ending point, the second column is the pin-to-pin delay value, and the third column is the partial path delay value from the starting point to the current point. Currently, the Timing Analyzer assumes that all the starting points are available at time 0.
- The starting point for a path can be one of the following:
  - Primary input port. The timing analyzer reports the port name followed by "[in]."
  - Bidirectional port. The timing analyzer reports the bidi port name followed by "[bidi]."
  - The output of a flip-flop or latch. The timing analyzer reports the instance name followed by a "/," followed by the output pin name, followed by "[inst_primitive name]."
- The ending point for a path can be one of the following:
  - Primary output port. The timing analyzer reports the port name followed by "[out]."
  - Bidi port. The timing analyzer reports the bidi port name followed by "[bidi]."
  - The data pin or clock pin of a flip-flop or latch. The timing analyzer reports the instance name followed by "/," followed by the pin name, followed by "[inst_primitive name]."
- For setup and hold information, the first column is the name of flip-flip or latch, the second column is the port name which drives the data pin of the flip-flop or latch, the third column is the port name which drives the clock pin of the flip-flop or latch. the fourth column is the setup value and the fifth column is the hold value.

An example of a detailed report starts on the following page.

```
Timing Analysis - Detailed Report
---------------------------------

Maximum Operating Frequency:    87 MHz

The following path determines the frequency:

  Startpoint: __0_$1N23/Q0
              (edge-triggered flip-flop)
  Endpoint: __0__LAF_G/D0
              (edge-triggered flip-flop)

         Point
  Name/pin_name [type]                      Delay      Path
  ---------------------------------------------------------
  __0_$1N23/Q0 [PGDFFR]                     0.00       0.00
  __0_$1N23_grp/A0 [PRBUFO1]                0.00       0.00
  __0_$1N23_grp/Z0 [PRBUFO1]                1.50       1.50
  __0_D6_P_LC_I11/A0 [PGBUFI]               0.00       1.50
  __0_D6_P_LC_I11/Z0 [PGBUFI]               1.00       2.50
  __0_D6_P_LC_P13/A0 [PGANDD2]              0.00       2.50
  __0_D6_P_LC_P13/Z0 [PGANDD2]              2.50       5.00
  __0_D6_P_LC_P13_xa/A0 [PGBUFXA]           0.00       5.00
  __0_D6_P_LC_P13_xa/Z0 [PGBUFXA]           1.50       6.50
  __0_D6_P_LC_X0MO/A0 [PGBUFXI]             0.00       6.50
  __0_D6_P_LC_X0MO/Z0 [PGBUFXI]             1.50       8.00
  __0_D6_P_LC_X0O/A1 [PGXOR2]               0.00       8.00
  __0_D6_P_LC_X0O/Z0 [PGXOR2]               0.50       8.50
  __0__LAF_G/D0 [PGDFFR]                    0.00       8.50
  ---------------------------------------------------------

The clock period is 11.50.
clock period = path delay + clock-to-output delay + setup time
  path delay: 8.50
  clock-to-output delay: 1.00
  setup time: 2.00


Information for flip-flop:

  Data-to-output delay:            0.00
  Clock-to-output delay:           1.00
  User reset-to-output delay:      2.50
  Global reset-to-output delay:    2.50
  Setup time:                      2.00
  Hold time:                       3.50
  Pulse-width time:                5.00
```

```
Longest Paths:

  Startpoint: A (input port)
  Endpoint: __0_$1N21/D0
            (edge-triggered flip-flop)

         Point
  Name/pin_name [type]                        Delay      Path
  ---------------------------------------------------------
  A [in]                                      0.00       0.00
  __0_IO11_P_IO_IBUFO/XI0 [PXIN]              0.00       0.00
  __0_IO11_P_IO_IBUFO/Z0 [PXIN]               1.00       1.00
  __0__PIN_A/A0 [PXBUFI]                      0.00       1.00
  __0__PIN_A/Z0 [PXBUFI]                      1.00       2.00
  __0__PIN_A_grp/A0 [PRBUFO1]                 0.00       2.00
  __0__PIN_A_grp/Z0 [PRBUFO1]                 1.50       3.50
  __0_A6_P_LC_I11/A0 [PGBUFI]                 0.00       3.50
  __0_A6_P_LC_I11/Z0 [PGBUFI]                 1.00       4.50
  __0_A6_P_LC_P13/A0 [PGANDD1]                0.00       4.50
  __0_A6_P_LC_P13/Z0 [PGANDD1]                2.50       7.00
  __0_A6_P_LC_P13_xa/A0 [PGBUFXA]             0.00       7.00
  __0_A6_P_LC_P13_xa/Z0 [PGBUFXA]             1.50       8.50
  __0_A6_P_LC_X0MO/A0 [PGBUFXI]               0.00       8.50
  __0_A6_P_LC_X0MO/Z0 [PGBUFXI]               1.50      10.00
  __0_A6_P_LC_X0O/A1 [PGXOR2]                 0.00      10.00
  __0_A6_P_LC_X0O/Z0 [PGXOR2]                 0.50      10.50
  __0_$1N21/D0 [PGDFFR]                       0.00      10.50
  data arrival time                                     10.50
  ---------------------------------------------------------

  Startpoint: C (input port)
  Endpoint: __0_$1N23/D0
            (edge-triggered flip-flop)

         Point
  Name/pin_name [type]                        Delay      Path
  ---------------------------------------------------------
  C [in]                                      0.00       0.00
  __0_IO43_P_IO_IBUFO/XI0 [PXIN]              0.00       0.00
  __0_IO43_P_IO_IBUFO/Z0 [PXIN]               1.00       1.00
  __0__PIN_C/A0 [PXBUFI]                      0.00       1.00
  __0__PIN_C/Z0 [PXBUFI]                      1.00       2.00
  __0__PIN_C_grp/A0 [PRBUFO1]                 0.00       2.00
  __0__PIN_C_grp/Z0 [PRBUFO1]                 1.50       3.50
  __0_A1_P_LC_I11/A0 [PGBUFI]                 0.00       3.50
  __0_A1_P_LC_I11/Z0 [PGBUFI]                 1.00       4.50
  __0_A1_P_LC_P13/A0 [PGANDD1]                0.00       4.50
  __0_A1_P_LC_P13/Z0 [PGANDD1]                2.50       7.00
  __0_A1_P_LC_P13_xa/A0 [PGBUFXA]             0.00       7.00
  __0_A1_P_LC_P13_xa/Z0 [PGBUFXA]             1.50       8.50
  __0_A1_P_LC_X0MO/A0 [PGBUFXI]               0.00       8.50
  __0_A1_P_LC_X0MO/Z0 [PGBUFXI]               1.50      10.00
  __0_A1_P_LC_X0O/A1 [PGXOR2]                 0.00      10.00
  __0_A1_P_LC_X0O/Z0 [PGXOR2]                 0.50      10.50
  __0_$1N23/D0 [PGDFFR]                       0.00      10.50
  data arrival time                                     10.50
  ---------------------------------------------------------
```

```
Startpoint: __0_$1N23/Q0
            (edge-triggered flip-flop)
Endpoint: __0__LAF_G/D0
          (edge-triggered flip-flop)

       Point
Name/pin_name [type]                    Delay      Path
---------------------------------------------------------
__0_$1N23/Q0 [PGDFFR]                   0.00       0.00
__0_$1N23_grp/A0 [PRBUFO1]              0.00       0.00
__0_$1N23_grp/Z0 [PRBUFO1]              1.50       1.50
__0_D6_P_LC_I11/A0 [PGBUFI]             0.00       1.50
__0_D6_P_LC_I11/Z0 [PGBUFI]             1.00       2.50
__0_D6_P_LC_P13/A0 [PGANDD2]            0.00       2.50
__0_D6_P_LC_P13/Z0 [PGANDD2]            2.50       5.00
__0_D6_P_LC_P13_xa/A0 [PGBUFXA]         0.00       5.00
__0_D6_P_LC_P13_xa/Z0 [PGBUFXA]         1.50       6.50
__0_D6_P_LC_X0MO/A0 [PGBUFXI]           0.00       6.50
__0_D6_P_LC_X0MO/Z0 [PGBUFXI]           1.50       8.00
__0_D6_P_LC_X0O/A1 [PGXOR2]             0.00       8.00
__0_D6_P_LC_X0O/Z0 [PGXOR2]             0.50       8.50
__0__LAF_G/D0 [PGDFFR]                  0.00       8.50
data arrival time                                  8.50
---------------------------------------------------------

Startpoint: D (input port)
Endpoint: __0_$1N23/CLK
          (edge-triggered flip-flop)

       Point
Name/pin_name [type]                    Delay      Path
---------------------------------------------------------
D [in]                                  0.00       0.00
__0__PIN_D/XI0 [PXINK]                  0.00       0.00
__0__PIN_D/Z0 [PXINK]                   1.00       1.00
__0__PIN_D_clk1/XI0 [PKIN1]             0.00       1.00
__0__PIN_D_clk1/Z0 [PKIN1]              4.50       5.50
__0_A1_P_LC_CLK/A0 [PGBUFK]             0.00       5.50
__0_A1_P_LC_CLK/Z0 [PGBUFK]             1.00       6.50
__0_$1N23/CLK [PGDFFR]                  0.00       6.50
data arrival time                                  6.50
---------------------------------------------------------
```

```
Startpoint: E (input port)
Endpoint: __0__LAF_G/CLK
          (edge-triggered flip-flop)

       Point
Name/pin_name [type]                    Delay      Path
--------------------------------------------------------
E [in]                                  0.00       0.00
__0__PIN_E/XI0 [PXINK]                  0.00       0.00
__0__PIN_E/Z0 [PXINK]                   1.00       1.00
__0__PIN_E_clk2/XI0 [PKIN2]             0.00       1.00
__0__PIN_E_clk2/Z0 [PKIN2]              4.50       5.50
__0_D6_P_LC_CLK/A0 [PGBUFK]             0.00       5.50
__0_D6_P_LC_CLK/Z0 [PGBUFK]             1.00       6.50
__0__LAF_G/CLK [PGDFFR]                 0.00       6.50
data arrival time                                  6.50
--------------------------------------------------------

Startpoint: __0__LAF_G/Q0
            (edge-triggered flip-flop)
Endpoint: G (output port)

       Point
Name/pin_name [type]                    Delay      Path
--------------------------------------------------------
__0__LAF_G/Q0 [PGDFFR]                  0.00       0.00
__0__LAF_G_iomux/A0 [PRBUFX1]           0.00       0.00
__0__LAF_G_iomux/Z0 [PRBUFX1]           2.50       2.50
__0_IO48_P_IO_OBUFI/A0 [PXBUFO]         0.00       2.50
__0_IO48_P_IO_OBUFI/Z0 [PXBUFO]         0.50       3.00
__0_G/A0 [PXOUT]                        0.00       3.00
__0_G/XO0 [PXOUT]                       2.50       5.50
G [out]                                 0.00       5.50
data arrival time                                  5.50
--------------------------------------------------------
```

```
  Startpoint: B (input port)
  Endpoint: __0_$1N21/CLK
            (edge-triggered flip-flop)

         Point
  Name/pin_name [type]                      Delay      Path
  ---------------------------------------------------------
  B [in]                                    0.00       0.00
  __0__PIN_B/XI0 [PXINK]                    0.00       0.00
  __0__PIN_B/Z0 [PXINK]                     1.00       1.00
  __0__PIN_B_clk0/XI0 [PKIN0]               0.00       1.00
  __0__PIN_B_clk0/Z0 [PKIN0]                3.50       4.50
  __0_A6_P_LC_CLK/A0 [PGBUFK]               0.00       4.50
  __0_A6_P_LC_CLK/Z0 [PGBUFK]               1.00       5.50
  __0_$1N21/CLK [PGDFFR]                    0.00       5.50
  data arrival time                                    5.50
  ---------------------------------------------------------


Shortest Paths:

  Startpoint: D (input port)
  Endpoint: __0_$1N23/CLK
            (edge-triggered flip-flop)

         Point
  Name/pin_name [type]                      Delay      Path
  ---------------------------------------------------------
  D [in]                                    0.00       0.00
  __0__PIN_D/XI0 [PXINK]                    0.00       0.00
  __0__PIN_D/Z0 [PXINK]                     1.00       1.00
  __0__PIN_D_clk1/XI0 [PKIN1]               0.00       1.00
  __0__PIN_D_clk1/Z0 [PKIN1]                2.50       3.50
  __0_A1_P_LC_CLK/A0 [PGBUFK]               0.00       3.50
  __0_A1_P_LC_CLK/Z0 [PGBUFK]               1.00       4.50
  __0_$1N23/CLK [PGDFFR]                    0.00       4.50
  data arrival time                                    4.50
  ---------------------------------------------------------
```

```
Startpoint: E (input port)
Endpoint: __0__LAF_G/CLK
          (edge-triggered flip-flop)

       Point
Name/pin_name [type]                     Delay      Path
---------------------------------------------------------
E [in]                                   0.00       0.00
__0__PIN_E/XI0 [PXINK]                   0.00       0.00
__0__PIN_E/Z0 [PXINK]                    1.00       1.00
__0__PIN_E_clk2/XI0 [PKIN2]              0.00       1.00
__0__PIN_E_clk2/Z0 [PKIN2]               2.50       3.50
__0_D6_P_LC_CLK/A0 [PGBUFK]              0.00       3.50
__0_D6_P_LC_CLK/Z0 [PGBUFK]              1.00       4.50
__0__LAF_G/CLK [PGDFFR]                  0.00       4.50
data arrival time                                   4.50
---------------------------------------------------------

Startpoint: __0__LAF_G/Q0
            (edge-triggered flip-flop)
Endpoint: G (output port)

       Point
Name/pin_name [type]                     Delay      Path
---------------------------------------------------------
__0__LAF_G/Q0 [PGDFFR]                   0.00       0.00
__0__LAF_G_iomux/A0 [PRBUFX1]            0.00       0.00
__0__LAF_G_iomux/Z0 [PRBUFX1]            2.50       2.50
__0_IO48_P_IO_OBUFI/A0 [PXBUFO]          0.00       2.50
__0_IO48_P_IO_OBUFI/Z0 [PXBUFO]          0.50       3.00
__0_G/A0 [PXOUT]                         0.00       3.00
__0_G/XO0 [PXOUT]                        2.50       5.50
G [out]                                  0.00       5.50
data arrival time                                   5.50
---------------------------------------------------------
```

```
Startpoint: B (input port)
Endpoint: __0_$1N21/CLK
          (edge-triggered flip-flop)

       Point
Name/pin_name [type]                        Delay      Path
---------------------------------------------------------
B [in]                                      0.00       0.00
__0__PIN_B/XI0 [PXINK]                      0.00       0.00
__0__PIN_B/Z0 [PXINK]                       1.00       1.00
__0__PIN_B_clk0/XI0 [PKIN0]                 0.00       1.00
__0__PIN_B_clk0/Z0 [PKIN0]                  3.50       4.50
__0_A6_P_LC_CLK/A0 [PGBUFK]                 0.00       4.50
__0_A6_P_LC_CLK/Z0 [PGBUFK]                 1.00       5.50
__0_$1N21/CLK [PGDFFR]                      0.00       5.50
data arrival time                                      5.50
---------------------------------------------------------

Startpoint: __0_$1N23/Q0
            (edge-triggered flip-flop)
Endpoint: __0__LAF_G/D0
          (edge-triggered flip-flop)

       Point
Name/pin_name [type]                        Delay      Path
---------------------------------------------------------
__0_$1N23/Q0 [PGDFFR]                       0.00       0.00
__0_$1N23_grp/A0 [PRBUFO1]                  0.00       0.00
__0_$1N23_grp/Z0 [PRBUFO1]                  1.50       1.50
__0_D6_P_LC_I11/A0 [PGBUFI]                 0.00       1.50
__0_D6_P_LC_I11/Z0 [PGBUFI]                 1.00       2.50
__0_D6_P_LC_P13/A0 [PGANDD2]                0.00       2.50
__0_D6_P_LC_P13/Z0 [PGANDD2]                2.50       5.00
__0_D6_P_LC_P13_xa/A0 [PGBUFXA]             0.00       5.00
__0_D6_P_LC_P13_xa/Z0 [PGBUFXA]             1.50       6.50
__0_D6_P_LC_X0MO/A0 [PGBUFXI]               0.00       6.50
__0_D6_P_LC_X0MO/Z0 [PGBUFXI]               1.50       8.00
__0_D6_P_LC_X0O/A1 [PGXOR2]                 0.00       8.00
__0_D6_P_LC_X0O/Z0 [PGXOR2]                 0.50       8.50
__0__LAF_G/D0 [PGDFFR]                      0.00       8.50
data arrival time                                      8.50
---------------------------------------------------------
```

```
Startpoint: A (input port)
Endpoint: __0_$1N21/D0
          (edge-triggered flip-flop)

       Point
Name/pin_name [type]                    Delay      Path
---------------------------------------------------------
A [in]                                  0.00       0.00
__0_IO11_P_IO_IBUFO/XI0 [PXIN]          0.00       0.00
__0_IO11_P_IO_IBUFO/Z0 [PXIN]           1.00       1.00
__0__PIN_A/A0 [PXBUFI]                  0.00       1.00
__0__PIN_A/Z0 [PXBUFI]                  1.00       2.00
__0__PIN_A_grp/A0 [PRBUFO1]             0.00       2.00
__0__PIN_A_grp/Z0 [PRBUFO1]             1.50       3.50
__0_A6_P_LC_I11/A0 [PGBUFI]             0.00       3.50
__0_A6_P_LC_I11/Z0 [PGBUFI]             1.00       4.50
__0_A6_P_LC_P13/A0 [PGANDD1]            0.00       4.50
__0_A6_P_LC_P13/Z0 [PGANDD1]            2.50       7.00
__0_A6_P_LC_P13_xa/A0 [PGBUFXA]         0.00       7.00
__0_A6_P_LC_P13_xa/Z0 [PGBUFXA]         1.50       8.50
__0_A6_P_LC_X0MO/A0 [PGBUFXI]           0.00       8.50
__0_A6_P_LC_X0MO/Z0 [PGBUFXI]           1.50      10.00
__0_A6_P_LC_X0O/A1 [PGXOR2]             0.00      10.00
__0_A6_P_LC_X0O/Z0 [PGXOR2]             0.50      10.50
__0_$1N21/D0 [PGDFFR]                   0.00      10.50
data arrival time                                 10.50
---------------------------------------------------------
```

```
  Startpoint: C (input port)
  Endpoint: __0_$1N23/D0
            (edge-triggered flip-flop)

         Point
  Name/pin_name [type]                     Delay      Path
  ---------------------------------------------------------
  C [in]                                   0.00       0.00
  __0_IO43_P_IO_IBUFO/XI0 [PXIN]           0.00       0.00
  __0_IO43_P_IO_IBUFO/Z0 [PXIN]            1.00       1.00
  __0__PIN_C/A0 [PXBUFI]                   0.00       1.00
  __0__PIN_C/Z0 [PXBUFI]                   1.00       2.00
  __0__PIN_C_grp/A0 [PRBUFO1]              0.00       2.00
  __0__PIN_C_grp/Z0 [PRBUFO1]              1.50       3.50
  __0_A1_P_LC_I11/A0 [PGBUFI]              0.00       3.50
  __0_A1_P_LC_I11/Z0 [PGBUFI]              1.00       4.50
  __0_A1_P_LC_P13/A0 [PGANDD1]             0.00       4.50
  __0_A1_P_LC_P13/Z0 [PGANDD1]             2.50       7.00
  __0_A1_P_LC_P13_xa/A0 [PGBUFXA]          0.00       7.00
  __0_A1_P_LC_P13_xa/Z0 [PGBUFXA]          1.50       8.50
  __0_A1_P_LC_X0MO/A0 [PGBUFXI]            0.00       8.50
  __0_A1_P_LC_X0MO/Z0 [PGBUFXI]            1.50      10.00
  __0_A1_P_LC_X0O/A1 [PGXOR2]              0.00      10.00
  __0_A1_P_LC_X0O/Z0 [PGXOR2]              0.50      10.50
  __0_$1N23/D0 [PGDFFR]                    0.00      10.50
  data arrival time                                  10.50
  ---------------------------------------------------------


Required Setup and Hold:

  Inst.Name                 Data         Clock        Setup        Hold
  ------------------------------------------------------------------------
  __0_$1N23                 C            D            8.00         -0.50
  __0_$1N21                 A            B            7.00         -1.50
```

Chapter 5 *Design Reports*

The pDS+ Fitter software provides a Report File that tells you how your design fit into the Lattice Semiconductor device architecture.

The Design Process Manager generates report summaries which are combined and saved in a file called *design_name*.rpt. The report file provides information about the environment under which a design is being processed, a description of the design which is to be processed, and a description of the processed design.

When a design successfully routes, the following report summaries are presented in the *design_name*.rpt file:

- Design Parameters – Describes the running environment.
- Design Specification – Summarizes the inputs and attributes in the design as specified by the designer.
- Pre-Route Design Statistics – Summarizes resource usage in a completed design.
- Post-Route Design Implementation – Summarizes partitioned design statistics and timing analysis at the end of successful routing.

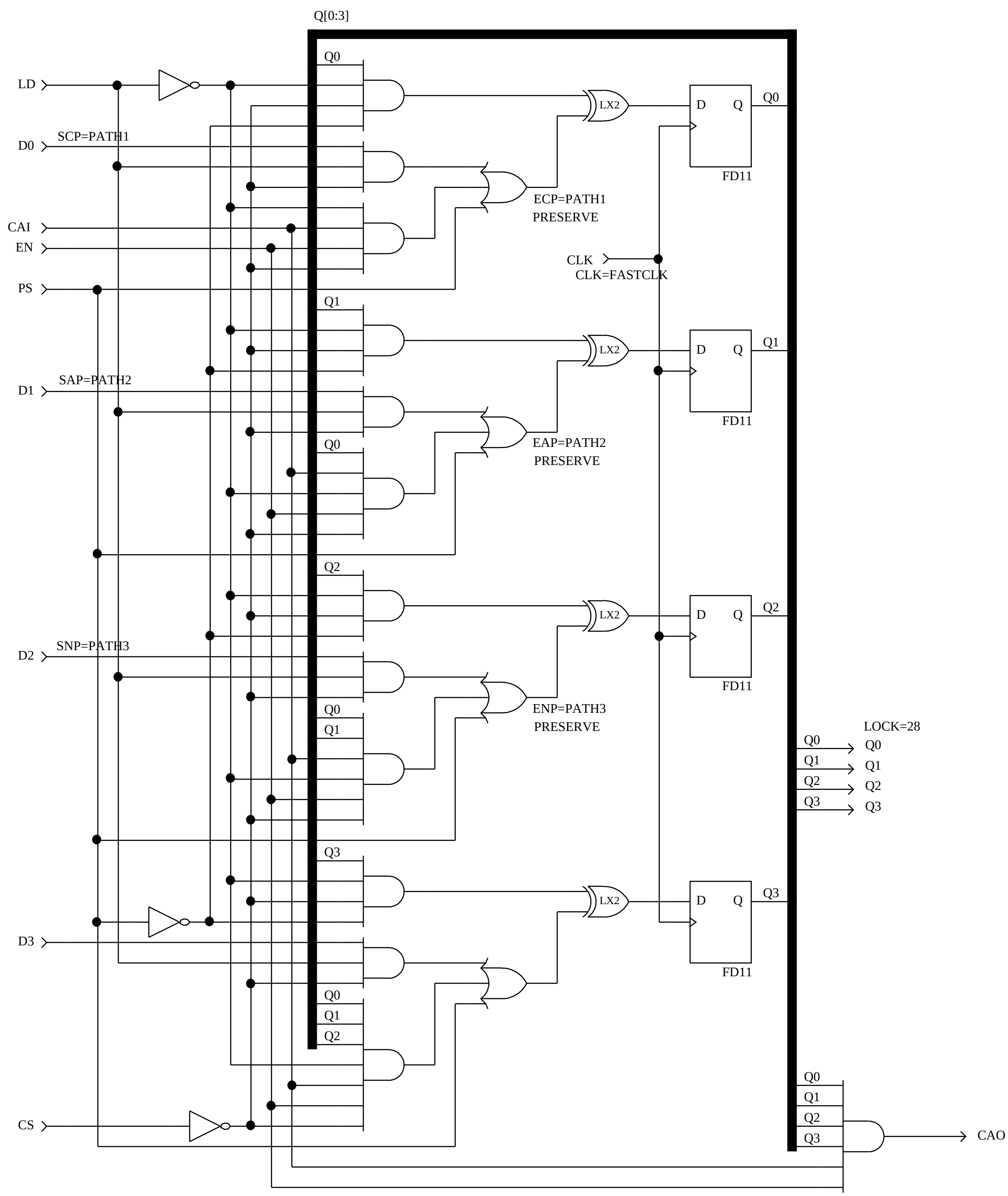
If a design does not successfully route, the following report summaries are presented in the *design_name*.rpt file:

- Design Parameters – Describes the running environment.
- Design Specification – Summarizes the inputs and attributes in the design as specified by the designer.
- Pre-Route Design Statistics – Summarizes resource usage in completed design. Indicates the cause of the unsuccessful routing.
- Pre-Route Design Implementation – Summarizes partitioned design statistics before routing.

Example Design

This chapter describes a design whose *design_name*.rpt file is used as an example throughout, then explains each section of the *design_name*.rpt file.

Throughout this section, a report file for a 4-bit counter is used as an example. The schematic for the 4-bit counter appears below.



Design Parameters

The Design Parameters Report describes the running environment, such as command-line or Parameter File options.

The following is an example of a Design Parameters Report.

```
Design Parameters
-----------------

EFFORT:                         2
EXTENDED_ROUTE:                 ON
IGNORE_FIXED_PIN:               OFF
MAX_GLB_IN:                     16
MAX_GLB_OUT:                    4
OUTPUT_FORM:                    VIEWLOGIC
PARAM_FILE:                     cnt4
STRATEGY:                       AREA
TIMING_FILE:                    CNT4
```

Design Specification

The Design Specification Report summarizes the inputs into the program. This report contains the following information:

- Design – The name of the design.
- Part – The part name of the target device for this design.
- IOC Statistics – Number of critical pins (Critical Pins), unlocked pins (Free Pins), and locked pins (Locked Pins).
- Pin Statistics – A listing of each input, output, and bidi pins by pin name, pin number, and the pin attribute assigned (if any).
- Symbol Attributes – A list of attributed symbol instances in the design, grouped by symbol attribute name.
- Net Attributes – A list of attributed nets, grouped by net attribute name.
- Path Descriptions – The path name, type (Critical, Asynchronous or No-Minimize), starting point, and ending point. For more information about paths see Chapter 2, “Design Attributes.”

The following is an example of a Design Specification Report.

```
Design Specification
--------------------

Design:                         cnt4
Part:                           pLSI1032-80LJ84

Number of Critical Pins:        0
Number of Free Pins:            14
Number of Locked Pins:          1


Input Pins

    Pin Name                Pin Attribute

        CAI
        CLK
        CS
        D0
        D1
        D2
        D3
        EN
        LD
        PS
```

Pin Statistics

```
Output Pins

    Pin Name                Pin Attribute

        CAO
        Q0                      LOCK 28                    Pin Statistics
        Q1
        Q2
        Q3


Clock Nets

    Net Name                Clock Specification            Pin Statistics

        CLK                     CLK0


Preserved Nets

    Net Name

        $1N365                                             Pin Statistics
        $1N419


Asynchronous Paths                                         Path Description

    Path Name               Starting Net            Ending Net

        PATH2                   D1                      $1N365


Critical Paths                                             Path Description

    Path Name               Starting Net            Ending Net

        PATH1                   D0                      $1N312


No-Minimize Paths                                          Path Description

    Path Name               Starting Net            Ending Net

        PATH3                   D2                      $1N419
```

Pre-Route Design Statistics

The Pre-Route Design Statistics Report is generated after partitioning of the design is complete. This report contains the following information:

- Total number of GLBs used in the design.
- Total number of IOCs used in the design.
- Total number of internal nets in the design.
- Break down of IOCs into types.
- Utilization of physical resources in the device.
- Distribution and average number of fanouts per net.
- Distribution and average number of inputs per GLB.
- Distribution and average number of outputs per GLB.
- List of output enable nets and their fanouts.

The following is an example of the Pre-Route Design Statistics Report.

```
Pre-Route Design Statistics
---------------------------

Number of GLBs:                 2
Number of I/Os:                 14
Number of Nets:                 16
```

Total Number of GLBs, IOCs, and Nets

```
Number of Free Inputs:          9
Number of Free Outputs:         4
Number of Free Three-States:    0
Number of Free Bidi's:          0

Number of Locked Input IOCs:    0
Number of Locked DIs:           0
Number of Locked Outputs:       1
Number of Locked Three-States:  0
Number of Locked Bidi's:        0

Number of CRIT Outputs:         0
Number of Global OEs:           0
Number of External Clocks:      1
```

Breakdown of IOCs into types

```
GLB Utilization (Out of 32):    6%
I/O Utilization (Out of 72):    19%
Net Utilization (Out of 200):   8%
```

Utilization of physical resources in the device

```
Nets with Fanout of  1:         7
Nets with Fanout of  2:         5
Nets with Fanout of  3:         4

Average Fanout per Net:         1.81
```

Distribution and average number of fanouts per net

```
GLBs with 11 Input(s):          1
GLBs with 13 Input(s):          1

Average Inputs per GLB:         12.00
```

Distribution and average number of inputs per GLB

```
GLBs with  3 Output(s):         1
GLBs with  4 Output(s):         1

Average Outputs per GLB:        3.50
```

Distribution and average number of outputs per GLB

Post-Route Design Implementation

The Post-Route Design Implementation Report summarizes resource usage in the design after successfully routing. The report provides information in several sections: GLB equations, pin and clock assignments, summary statistics, and timing analysis.

GLB Equations

The GLB equations portion of the Post-Route Implementation Report contains the following information:

- GLB information for each GLB, including:
  - GLB type and GLB name
  - Number and names of GLB inputs
  - Number and names of GLB outputs
  - Number of used product terms (PTs)
- GLB output information for each GLB, including:
  - Number of inputs relating to each GLB output, their sources and their names
  - Number and names of fanouts on the output of each GLB
  - Number of GLB levels leading to the output of each GLB
  - A Boolean equation for each GLB output, representing the implementation inside the GLB
- Post-Route GLBs
  - GLB input and GLB output locations in the order of specified inputs and outputs
  - Number of GLB levels for a registered GLB output relates to the maximum number of GLB levels of combinatorial logic generating data input, clock input, and reset input of the register.

The GLB equations portion uses the following symbols:

- & indicates ANDing of signals.
- # indicates ORing of PTs.
- ! indicates negation (NOT) at the input of a GLB.
- $ means the XOR gate is hardwired in the GLB.
- .D indicates the D input of a flip-flop.
- .C indicates the clock input of a flip-flop.
- .R indicates PT reset of the flip-flop inside a GLB.
- _ Names beginning with an underscore usually indicate an internal name for a net.
- ( ) Any parenthesized terms in GLB equations represent an implementation through a PT sharing array.

- GLB inputs may be represented as a parenthesized triplet of names. The first entry represents the source of the signal, the second entry is the signal name, and the last entry is post-route input location.
- GLB outputs may be represented as a parenthesized pair of names. The first entry is the signal name, and the second entry is post-route location.

The following dot extensions are used after GLB, IOC, or DI names to refer to specific input or output connections.

- .On (or On) where n=0 . . 3, represents the specific GLB output as signal source.
- .O represents the IOC/DI as signal source.
- .OE represents the specific GLB enable output as signal source.
- .Im (or Im) where m=0 . . 17, or m=0 . . 23, represents the specific GLB input as signal destination.
- .IR represents connection to the IOC through ORP as signal destination.
- .ID represents connection to the IOC through ORP bypass as signal destination.
- .OEe where e=0 or 1, represents the IOC enable input as signal destination.

GLB Equations Example

The following is an example of the GLB equations portion of the Post-Route Report.

```
Post-Route Design Implementation
--------------------------------

Number of GLBs:         2
Number of IOCs:         14
Number of DIs:          0
Number of GLB Levels:   2

                                                          GLB input name triplet
GLB glb0, A6

    13 Input(s)
        (glb0.O2, QI0, I10), (glb1.O1, QI1, I9), (glb1.O0, QI2, I8),
        (glb0.O0, QI3, I16), (CAI.O, _PIN_CAI, I15), (CS.O, _PIN_CS, I13),
        (D0.O, _PIN_D0, I6), (D1.O, _PIN_D1, I4), (D2.O, _PIN_D2, I2),
        (D3.O, _PIN_D3, I11), (EN.O, _PIN_EN, I5), (LD.O, _PIN_LD, I0),
        (PS.O, _PIN_PS, I1)
    4 Output(s)
        (QI3, O0), (QI0, O2), ($1N419, O1), ($1N365, O3)    GLB output name pair
    14 Product Term(s)
```

```
Output QI3

    10 Input(s)
        QI0, QI1, QI2, QI3, _PIN_CAI, _PIN_CS, _PIN_D3, _PIN_EN,
        _PIN_LD, _PIN_PS
    3 Fanout(s)
        glb0.I16, glb1.I7, Q3.IR
```

.IR indicates connection to the IOC through the ORP

```
    4 Product Term(s)
    1 GLB Level(s)

    QI3.D = (_PIN_PS
        # _PIN_D3 & _PIN_LD & !_PIN_CS
        # QI0 & QI1 & QI2 & _PIN_CAI & _PIN_EN & !_PIN_CS & !_PIN_LD)
        $ QI3 & !_PIN_CS & !_PIN_LD & !_PIN_PS
    QI3.C = _PIN_CLK

Output QI0

    7 Input(s)
        QI0, _PIN_CAI, _PIN_CS, _PIN_D0, _PIN_EN, _PIN_LD, _PIN_PS
    3 Fanout(s)
        glb0.I10, glb1.I5, Q0.IR
    4 Product Term(s)
    1 GLB Level(s)

    QI0.D = (_PIN_PS
        # _PIN_D0 & _PIN_LD & !_PIN_CS
        # _PIN_CAI & _PIN_EN & !_PIN_CS & !_PIN_LD)
        $ QI0 & !_PIN_CS & !_PIN_LD & !_PIN_PS
```

Product Term sharing

```
    QI0.C = _PIN_CLK

Output $1N419

    8 Input(s)
        QI0, QI1, _PIN_CAI, _PIN_CS, _PIN_D2, _PIN_EN, _PIN_LD, _PIN_PS
    1 Fanout(s)
        glb1.I6
```

.Im indicates the specific GLB input as the signal destination, in this case, I6

```
    3 Product Term(s)
    1 GLB Level(s)

    $1N419 = (_PIN_PS
        # _PIN_D2 & _PIN_LD & !_PIN_CS
        # QI0 & QI1 & _PIN_CAI & _PIN_EN & !_PIN_CS & !_PIN_LD)

Output $1N365

    7 Input(s)
        QI0, _PIN_CAI, _PIN_CS, _PIN_D1, _PIN_EN, _PIN_LD, _PIN_PS
    1 Fanout(s)
        glb1.I4
    3 Product Term(s)
    1 GLB Level(s)

    $1N365 = (_PIN_PS
        # _PIN_D1 & _PIN_LD & !_PIN_CS
        # QI0 & _PIN_CAI & _PIN_EN & !_PIN_CS & !_PIN_LD)
```

```
GLB glb1, D6

    11 Input(s)
        (glb0.O2, QI0, I5), (glb1.O1, QI1, I17), (glb1.O0, QI2, I16),
        (glb0.O0, QI3, I7), (glb0.O3, $1N365, I4), (glb0.O1, $1N419, I6),
        (CAI.O, _PIN_CAI, I0), (CS.O, _PIN_CS, I2), (EN.O, _PIN_EN, I10),
        (LD.O, _PIN_LD, I15), (PS.O, _PIN_PS, I14)
    3 Output(s)
        (QI2, O0), (QI1, O1), (_LAF_CAO, O2)
    5 Product Term(s)

    Output QI2

        5 Input(s)
            QI2, $1N419, _PIN_CS, _PIN_LD, _PIN_PS
        3 Fanout(s)
            glb0.I8, glb1.I16, Q2.IR
        2 Product Term(s)
        2 GLB Level(s)

        QI2.D = (QI2 & !_PIN_CS & !_PIN_LD & !_PIN_PS)
            $ $1N419
        QI2.C = _PIN_CLK

    Output QI1

        5 Input(s)
            QI1, $1N365, _PIN_CS, _PIN_LD, _PIN_PS
        3 Fanout(s)
            glb0.I9, glb1.I17, Q1.IR
        2 Product Term(s)
        2 GLB Level(s)

        QI1.D = (QI1 & !_PIN_CS & !_PIN_LD & !_PIN_PS)
            $ $1N365
        QI1.C = _PIN_CLK
    Output _LAF_CAO

        6 Input(s)
            QI0, QI1, QI2, QI3, _PIN_CAI, _PIN_EN
        1 Fanout(s)
            CAO.IR
        1 Product Term(s)
        1 GLB Level(s)

        _LAF_CAO = QI0 & QI1 & QI2 & QI3 & _PIN_CAI & _PIN_EN
```

Pin and Clock Information

The Pin and Clock portion of the Post-Route Implementation Report contains the following information:

- I/O (Input/Output/BIDI/Registered/Combinational) type, I/O name and location of each pin. Each I/O is followed by a description of the I/O source or destinations and connections.
- A list of the global clocks and their types.
- .E represents the enable input of a 3-state or bidirectional IOC
- .C represents the clock input of a registered input or bidirectional IOC.
- .G represents the enable input of a latched input or bidirectional IOC.

NOTE If you are using an ispLSI or pLSI 1032, the IOCLK0 and IOCLK1 will be used interchangeably in the clock assignment section of the report file.

Pin and Clock Example

The following is an example of the Pin and Clock portion of the Post-Route Implementation Report.

```
Input CAI, IO31

    Output _PIN_CAI
        2 Fanout(s)
            glb0.I15, glb1.I0

Output CAO, IO50

    Input (glb1.O2, _LAF_CAO)

    CAO = _LAF_CAO

Clock Input CLK, Y0

    Output _PIN_CLK
        2 Fanout(s)
            glb0.CLK0, glb1.CLK0

Input CS, IO61

    Output _PIN_CS
        2 Fanout(s)
            glb0.I13, glb1.I2
```

```
Input D0, IO54

    Output _PIN_D0
        1 Fanout(s)
            glb0.I6

Input D1, IO20

    Output _PIN_D1
        1 Fanout(s)
            glb0.I4

Input D2, IO34

    Output _PIN_D2
        1 Fanout(s)
            glb0.I2

Input D3, IO59

    Output _PIN_D3
        1 Fanout(s)
            glb0.I11

Input EN, IO21

    Output _PIN_EN
        2 Fanout(s)
            glb0.I5, glb1.I10

Input LD, IO0

    Output _PIN_LD
        2 Fanout(s)
            glb0.I0, glb1.I15

Input PS, IO1

    Output _PIN_PS
        2 Fanout(s)
            glb0.I1, glb1.I14


Output Q0, IO2

    Input (glb0.O2, QI0)

    Q0 = QI0

Output Q1, IO49

    Input (glb1.O1, QI1)

    Q1 = QI1
```

```
Output Q2, IO48

    Input (glb1.O0, QI2)

    Q2 = QI2

Output Q3, IO4

    Input (glb0.O0, QI3)

    Q3 = QI3

Clock Assignments

    Net Name                    Clock Assignment

    _PIN_CLK                    External CLK0
```

Summary Statistics

The Summary Statistics section of the Post-Route Implementation Report consists of three tables: GLB and GLB Output Statistics, Maximum Level Trace, and Pin Assignments.

GLB and GLB Output Statistics Table

The GLB and GLB Output Statistics table contains the following information:

- Name, Location, and Output Names for each GLB.
- Input, Output, and Product Term statistics for each GLB.
- Inputs, Fanouts, Product Terms, and Levels for each GLB output.

The following is an example of the GLB and GLB Output table from the Post-Route Implementation Report.

```
GLB and GLB Output Statistics

    GLB Name, Location      GLB Statistics          GLB Output Statistics
    GLB Output Name         Ins, Outs, PTs          Ins, FOs, PTs, Levels

        glb0, A6                13,  4, 14
            $1N365                                      7,  1,  3,  1
            $1N419                                      8,  1,  3,  1
            QI0                                         7,  3,  4,  1
            QI3                                        10,  3,  4,  1

        glb1, D6                11,  3,  5
            QI1                                         5,  3,  2,  2
            QI2                                         5,  3,  2,  2
            _LAF_CAO                                    6,  1,  1,  1
```

Maximum Level Trace Table

The Maximum Level Trace table contains the following information:

- Number of GLB levels for the related GLB output name, GLB name, and number of inputs in the transitive fan-in of the related GLB output.
- GLB Output Name for each GLB on the specified critical path.

This table provides a trace-back for GLB outputs which have their GLB levels equal to the maximum GLB levels in the design. The trace related to different GLB outputs are separated by blank lines. For each GLB output the paths relating to the maximum level are reported in each section. This information can be used to identify performance bottlenecks in the design. GLB levels may be inaccurate if the design contains combinational loops.

The following is an example of the Maximum Level Trace table from the Post-Route Implementation Report.

```
Maximum Level Trace

    GLB Level, Name, Ins    GLB Output Name

        2, glb1, 9              QI2
        1, glb0                  $1N419

        2, glb1, 8              QI1
        1, glb0                  $1N365
```

Pin Assignments Table

The Pin Assignments table contains the following information:

- Name of each pin.
- Pin assignment for each pin.
- Type of pin and any pin Design Attribute assigned (except LOCK Design Attribute).

The following is an example of the Pin Assignments table from the Post-Route Implementation Report.

```
Pin Assignments

    Pin Name                Pin Assignment          Pin Type, Pin Attribute

        Q2                      3                       Output
        Q1                      4                       Output
        CAO                     5                       Output
        D0                      9                       Input
        D3                      14                      Input
        CS                      16                      Input
        CLK                     20                      Clock Input
        LD                      26                      Input
        PS                      27                      Input
        Q0                      28                      Output
        Q3                      30                      Output
        D1                      49                      Input
        EN                      50                      Input
        CAI                     60                      Input
        D2                      70                      Input
```

Timing Analysis

The Timing Analysis section contains the following information:

- Frequency calculation
- Longest and shortest paths
- Setup and hold time evaluation and boundary register

In addition to these three parts, each report file contains the timing information for flip-flops and latches when applicable. The following is an example of a Short Report which is appended to the .rpt file.

```
Timing Analysis - Short Report
------------------------------

Maximum Operating Frequency:    47 MHz

The following path determines the frequency:

  Startpoint: __0_QI1/Q0
              (edge-triggered flip-flop)
  Endpoint: __0_QI2/D0
              (edge-triggered flip-flop)

         Point
  Name/pin_name [type]                     Delay      Path
  ---------------------------------------------------------
  __0_QI1/Q0 [PGDFFR]                      0.00       0.00
  __0_QI1_grp/A0 [PRBUFO1]                 0.00       0.00
  __0_QI1_grp/Z0 [PRBUFO1]                 1.50       1.50
  __0_A6_P_LC_I9/A0 [PGBUFI]               0.00       1.50
  __0_A6_P_LC_I9/Z0 [PGBUFI]               1.00       2.50
  __0_A6_P_LC_P10/A3 [PGAND6]              0.00       2.50
  __0_A6_P_LC_P10/Z0 [PGAND6]              2.00       4.50
  __0_A6_P_LC_F4/A2 [PGORF53]              0.00       4.50
  __0_A6_P_LC_F4/Z0 [PGORF53]              3.00       7.50
  __0_A6_P_LC_G2/A0 [PGORG1]               0.00       7.50
  __0_A6_P_LC_G2/Z0 [PGORG1]               1.00       8.50
  __0_A6_P_LC_X1O/A0 [PGXOR2]              0.00       8.50
  __0_A6_P_LC_X1O/Z0 [PGXOR2]              0.50       9.00
  __0_$1N419/A0 [PGBUFXO]                  0.00       9.00
  __0_$1N419/Z0 [PGBUFXO]                  1.00      10.00
  __0_$1N419_grp/A0 [PRBUFO1]              0.00      10.00
  __0_$1N419_grp/Z0 [PRBUFO1]              1.50      11.50
  __0_D6_P_LC_I6/A0 [PGBUFI]               0.00      11.50
  __0_D6_P_LC_I6/Z0 [PGBUFI]               1.00      12.50
  __0_D6_P_LC_P13/A0 [PGANDD1]             0.00      12.50
  __0_D6_P_LC_P13/Z0 [PGANDD1]             2.50      15.00
  __0_D6_P_LC_P13_xa/A0 [PGBUFXA]          0.00      15.00
  __0_D6_P_LC_P13_xa/Z0 [PGBUFXA]          1.50      16.50
  __0_D6_P_LC_X0MO/A0 [PGBUFXI]            0.00      16.50
  __0_D6_P_LC_X0MO/Z0 [PGBUFXI]            1.50      18.00
  __0_D6_P_LC_X0O/A1 [PGXOR2]              0.00      18.00
  __0_D6_P_LC_X0O/Z0 [PGXOR2]              0.50      18.50
  __0_QI2/D0 [PGDFFR]                      0.00      18.50
  ---------------------------------------------------------
```

```
The clock period is 21.50.
clock period = path delay + clock-to-output delay + setup time
  path delay:              18.50
  clock-to-output delay:   1.00
  setup time:              2.00


Information for flip-flop:

  Data-to-output delay:           0.00
  Clock-to-output delay:          1.00
  User reset-to-output delay:     2.50
  Global reset-to-output delay:   2.50
  Setup time:                     2.00
  Hold time:                      3.50
  Pulse-width time:               5.00


Longest Paths:

      Startpoint                     Endpoint                  Path Delay
  Name/pin_name [type]          Name/pin_name [type]
  -----------------------------------------------------------------------
  PS [in]                         __0_QI2/D0 [PGDFFR]            21.20
  PS [in]                         __0_QI1/D0 [PGDFFR]            20.70
  EN [in]                         CAO [out]                      17.20
  PS [in]                         __0_QI0/D0 [PGDFFR]            11.20
  PS [in]                         __0_QI3/D0 [PGDFFR]            11.20
  __0_QI0/Q0 [PGDFFR]             Q0 [out]                       5.50
  __0_QI1/Q0 [PGDFFR]             Q1 [out]                       5.50
  __0_QI2/Q0 [PGDFFR]             Q2 [out]                       5.50
  __0_QI3/Q0 [PGDFFR]             Q3 [out]                       5.50
  CLK [in]                        __0_QI0/CLK [PGDFFR]           5.50
  CLK [in]                        __0_QI1/CLK [PGDFFR]           5.50
  CLK [in]                        __0_QI2/CLK [PGDFFR]           5.50
  CLK [in]                        __0_QI3/CLK [PGDFFR]           5.50
```

```
Shortest Paths:

    Startpoint                     Endpoint                  Path Delay
  Name/pin_name [type]         Name/pin_name [type]
  ------------------------------------------------------------------------
  __0_QI0/Q0 [PGDFFR]            Q0 [out]                       5.50
  __0_QI1/Q0 [PGDFFR]            Q1 [out]                       5.50
  __0_QI2/Q0 [PGDFFR]            Q2 [out]                       5.50
  __0_QI3/Q0 [PGDFFR]            Q3 [out]                       5.50
  CLK [in]                       __0_QI0/CLK [PGDFFR]           5.50
  CLK [in]                       __0_QI1/CLK [PGDFFR]           5.50
  CLK [in]                       __0_QI2/CLK [PGDFFR]           5.50
  CLK [in]                       __0_QI3/CLK [PGDFFR]           5.50
  __0_QI0/Q0 [PGDFFR]            __0_QI0/D0 [PGDFFR]            7.50
  __0_QI3/Q0 [PGDFFR]            __0_QI3/D0 [PGDFFR]            7.50
  __0_QI1/Q0 [PGDFFR]            __0_QI1/D0 [PGDFFR]            8.00
  __0_QI2/Q0 [PGDFFR]            __0_QI2/D0 [PGDFFR]            8.00
  __0_QI1/Q0 [PGDFFR]            CAO [out]                      14.00


Required Setup and Hold:

  Inst.Name                 Data         Clock        Setup        Hold
  ------------------------------------------------------------------------
  __0_QI0                   PS           CLK          7.70         -1.70
  __0_QI0                   LD           CLK          7.70         -1.70
  __0_QI0                   EN           CLK          7.70         -2.20
  __0_QI0                   D0           CLK          7.50         -2.00
  __0_QI0                   CS           CLK          7.70         -1.70
  __0_QI0                   CAI          CLK          7.70         -2.20
  __0_QI3                   PS           CLK          7.70         -1.70
  __0_QI3                   LD           CLK          7.70         -1.70
  __0_QI3                   EN           CLK          7.70         -2.20
  __0_QI3                   D3           CLK          7.50         -2.00
  __0_QI3                   CS           CLK          7.70         -1.70
  __0_QI3                   CAI          CLK          7.70         -2.20
  __0_QI1                   PS           CLK          17.20        -2.20
  __0_QI1                   LD           CLK          17.20        -2.20
  __0_QI1                   EN           CLK          17.20        -11.70
  __0_QI1                   D1           CLK          17.00        -11.50
  __0_QI1                   CS           CLK          17.20        -2.20
  __0_QI1                   CAI          CLK          17.20        -11.70
  __0_QI2                   PS           CLK          17.70        -2.20
  __0_QI2                   LD           CLK          17.20        -2.20
  __0_QI2                   EN           CLK          17.20        -11.70
  __0_QI2                   D2           CLK          17.00        -11.50
  __0_QI2                   CS           CLK          17.20        -2.20
  __0_QI2                   CAI          CLK          17.20        -11.70
```

Pre-Route Design Implementation

The Pre-Route Design Implementation Report is only generated if routing fails. It contains partitioned design statistics at the time of the routing failure. The cause of the routing failure is indicated after the list of output enable nets and before the Pre-Route Design Implementation header.

The Pre-Route Design Implementation Report is similar to the Post-Route Design Implementation Report, except the GLB, IOC, and the Input/Output location information is not included. The following is a portion of a Pre-Route Design Implementation Report.

```
Design Parameters
-----------------

EXTENDED_ROUTE:                 ON
IGNORE_FIXED_PIN:               OFF
MAX_GLB_IN:                     16
MAX_GLB_OUT:                    4
OUTPUT_FORM:                    VIEWLOGIC, LDF
PARAM_FILE:                     None
STRATEGY:                       AREA
TIMING_FILE:                    TIMING


Design Specification
--------------------

Design:                         btc_50
Part:                           pLSI1032-80LJ84


Number of CRIT Pins:            3
Number of Free Pins:            36
Number of Locked Pins:          2


Input Pins

    Pin Name                Pin Number              Pin Attribute

        CD
        CLK
        D0
        D1
        D2
        D3
        EN
        INCLK0
        INCLK1
        LD
        OE0
        OE1
```

```
Output Pins

    Pin Name                Pin Number              Pin Attribute

        Q0_0
        Q0_1
        Q2_0
        Q2_1
        Q2_10
        Q2_11
        Q2_12
        Q2_13
        Q2_14
        Q2_15
        Q2_2
        Q2_3
        Q2_4
        Q2_5
        Q2_6
        Q2_7
        Q2_8
        Q2_9
        Q3
        Q4
        Q5
        Q7_0                    26                      CRIT, LOCK
        Q7_1                    27                      CRIT, LOCK
        Q7_2                                            CRIT
        Q7_3
        Q7_4


Hardmacro Instances

    Instance Name           Hardmacro Name

        $1I2                    cdd18
        $1I452                  cdd24



Protected Gates

    Instance Name           Gate Name

        $1I561                  OR2


Preserved Nets

    Net Name

        $1N475
```

```
Pre-Route Design Statistics
---------------------------

Number of GLBs:                 7
Number of Nets:                 26

Number of Inputs:               11
Number of Outputs:              3
Number of Three-States:         23
Number of Bidi's:               0

Number of Locked Input IOCs:    0
Number of Locked DIs:           0
Number of Locked Outputs:       0
Number of Locked Three-States:  2
Number of Locked Bidi's:        0

Number of CRIT Outputs:         3
Number of Global OEs:           0
Number of External Clocks:      1


GLB Utilization (Out of 32):    21%
I/O Utilization (Out of 72):    51%
Net Utilization (Out of 200):   13%


Nets with Fanout of  1:         10
Nets with Fanout of  2:         2
Nets with Fanout of  3:         5
Nets with Fanout of  4:         3
Nets with Fanout of  5:         4
Nets with Fanout of  6:         1
Nets with Fanout of 18:         1

Average Fanout per Net:         3.27


GLBs with  3 Input(s):          1
GLBs with  4 Input(s):          1
GLBs with  8 Input(s):          1
GLBs with  9 Input(s):          1
GLBs with 10 Input(s):          1
GLBs with 12 Input(s):          1
GLBs with 13 Input(s):          1

Average Inputs per GLB:         8.43


GLBs with  1 Output(s):         2
GLBs with  2 Output(s):         1
GLBs with  3 Output(s):         3
GLBs with  4 Output(s):         1

Average Outputs per GLB:        2.43
```

```
Output Enable Nets

    Net Name                Net Fanout

        $1N34                   23
```

List of Output Enable Nets and their fanouts

```
Placement and routing completed unsuccessfully due to 1 unconnected nets

IOC Q7_2 cannot be placed
IOC Q7_2 has unconnected OE
```

Cause of routing failure

```
Pre-Route Design Implementation
-------------------------------
```

Header

```
Number of GLBs:         9
Number of IOCs:         37
Number of DIs:          0
Number of GLB Levels:   2


GLB glb0_part1

    8 Input(s)
        $1N482, $1N518, _LAF_Q3, _LAF_Q4, _LAF_Q5, _PIN_CD, _PIN_EN, _PIN_LD

    1 Output(s)
        _LAF_Q4

    5 Product Term(s)

    Output _LAF_Q4

        8 Input(s)
            (glb3.O2, $1N482), (glb0_part3.O2, $1N518), (glb0_part2.O1,
            _LAF_Q3), (glb0_part1.O0, _LAF_Q4), (glb1.O0, _LAF_Q5), (CD.O,
            _PIN_CD), (EN.O, _PIN_EN), (LD.O, _PIN_LD)
        5 Fanout(s)
            glb1.I3, glb0_part1.I3, glb0_part2.I2, glb0_part3.I2, Q4.IR
        5 Product Term(s)
        1 GLB Level(s)

        _LAF_Q4.D = (_LAF_Q4 & _PIN_EN & !_PIN_LD & !_PIN_CD & !$1N518 &
            !_LAF_Q3
            # _LAF_Q5 & _PIN_EN & !_PIN_LD & !_PIN_CD & !$1N518 & !_LAF_Q3
            # _LAF_Q4 & _LAF_Q5 & _PIN_EN & !_PIN_LD & !_PIN_CD
            # $1N482 & _PIN_LD & !_PIN_CD)
            $ _LAF_Q4 & !_PIN_LD & !_PIN_CD
        _LAF_Q4.C = _BUF_1116
```

```
GLB glb0_part2

    8 Input(s)
        $1N518, _LAF_Q3, _LAF_Q4, _LAF_Q5, $1N565, _PIN_CD, _PIN_EN, _PIN_LD

    1 Output(s)
        _LAF_Q3

    6 Product Term(s)

    Output _LAF_Q3

        8 Input(s)
            (glb0_part3.O2, $1N518), (glb0_part2.O1, _LAF_Q3),
            (glb0_part1.O0, _LAF_Q4), (glb1.O0, _LAF_Q5), (glb5.O0, $1N565),
            (CD.O, _PIN_CD), (EN.O, _PIN_EN), (LD.O, _PIN_LD)
        5 Fanout(s)
            glb1.I2, glb0_part1.I2, glb0_part2.I1, glb0_part3.I1, Q3.IR
        6 Product Term(s)
        2 GLB Level(s)

        _LAF_Q3.D = (_LAF_Q3 & _PIN_EN & !_PIN_LD & !_PIN_CD & !$1N518
            # _LAF_Q5 & _PIN_EN & !_PIN_LD & !_PIN_CD & !$1N518 & !_LAF_Q4
            # _LAF_Q4 & _PIN_EN & !_PIN_LD & !_PIN_CD & !$1N518 & !_LAF_Q5
            # _LAF_Q3 & _LAF_Q5 & _PIN_EN & !_PIN_LD & !_PIN_CD
            # $1N565 & _PIN_LD & !_PIN_CD)
            $ _LAF_Q3 & !_PIN_LD & !_PIN_CD
        _LAF_Q3.C = _BUF_1116
```

Appendix A *Design Rules and Tips*

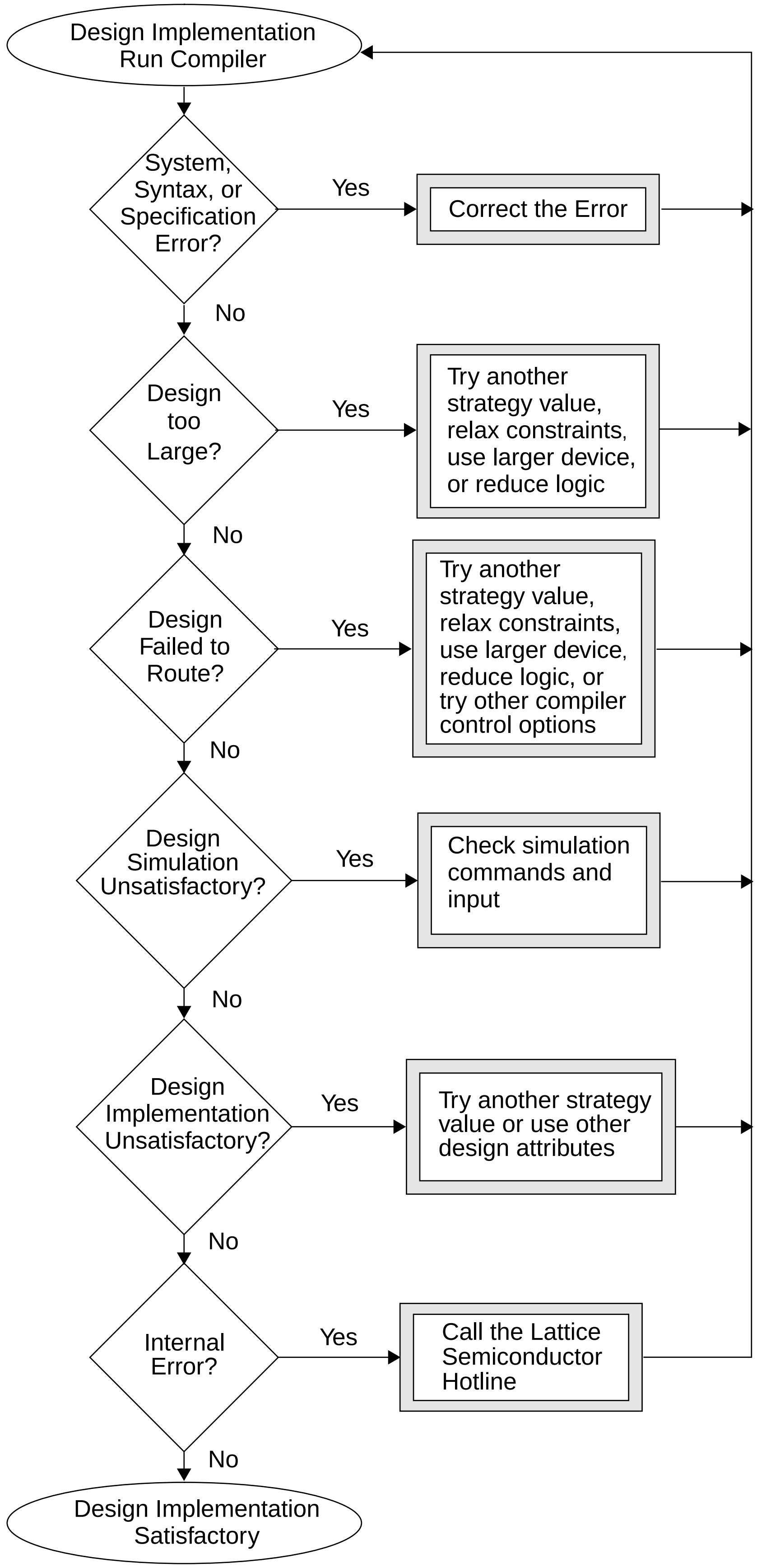
During the design process (design entry through routing), you may experience problems implementing your design the way you want in the Lattice Semiconductor ispLSI and pLSI devices. This appendix is designed to help you complete a design which meets your objectives by identifying common design errors and problems, design rules, and design tips.

Design Problems

If your design failed the compilation process, you can use the Report (.rpt) and Log (.log) files to determine the cause and take corrective action. In general, a compilation failure is caused by the following conditions:

- There is a syntax or system error.
- The design is too large for the chosen device.
- The design is too complex for the chosen device with specified constraints and objectives to route successfully.
- Timing simulation results do not meet your design objectives or expectations.
- The implementation of your design does not meet your design objectives or expectations.
- During the compilation process, you receive an internal error message.

The diagram on the following page shows how to correct these design problems.


Design Implementation
Run Compiler
System, Syntax, or Specification Error?
Yes
Correct the Error
No
Design too Large?
Yes
Try another strategy value, relax constraints, use larger device, or reduce logic
No
Design Failed to Route?
Yes
Try another strategy value, relax constraints, use larger device, reduce logic, or try other compiler control options
No
Design Simulation Unsatisfactory?
Yes
Check simulation commands and input
No
Design Implementation Unsatisfactory?
Yes
Try another strategy value or use other design attributes
No
Internal Error?
Yes
Call the Lattice Semiconductor Hotline
No
Design Implementation Satisfactory

System, Syntax, and Specification Errors

The most common problems occur during installation of the software (system errors), or during design entry (syntax or specification errors).

System Errors

System errors are usually caused by corrupted files, incorrect file protections, incorrectly set environment variables, or use of unsupported or unauthorized part names. The PDSPLUS environment variable must point to the directory where the pDS+ Fitter is installed. The path variable should be set to point to the pDS+ Fitter executables directory.

Reserved File Names

A number of files are generated and maintained by the Design Process Manager (DPM). These files cannot be used as data files in the directory in which DPM is being run. If this happens, any such file may be removed, overwritten or a system error occurs. Examples of such file names are *design*.par, and *design*.ppn. It is best to avoid using these names as your file names, or, separate the DPM run directory from your data files directory.

Syntax Errors

Syntax errors are usually caused by incorrect spelling or unsupported options. Syntactic problems should be corrected before your design can be properly processed by the pDS+ Fitter. Use the information provided by the Log report to identify these errors and make corrections as required.

Valid Characters

Design documentation information can be added to your design as comments without affecting the compiler. When adding comments to your design source file, you can use all of the following standard alphanumeric characters and symbols except the semicolon and new-line characters:

- a-z
- A-Z
- 0-9
- @ # $ % ^ & * ~ _ - + = < > / | \ ( ) { } [ ] “ ” ‘ ’: , . ? !
- space, tab

Identifier names in your design must begin with an alphabetic character, and are restricted to the following characters:

- a-z
- A-Z
- 0-9
- @ # $ % & * ~ _ - + / | \ ‘ ’ <> [] !

White space characters, space, and horizontal tabs can be used as separators.

Valid Identifiers and Text

Identifiers are case insensitive, unless the CASE_SENSITIVE option is set to ON. You can use either uppercase or lowercase characters in any combination, except where specifically noted. However, all identifiers are modified to uppercase characters in output files, such as the .log and .rpt files, if the CASE_SENSITIVE option is set to OFF.

The following list gives the maximum number of characters you can use:

- Component names – 31
- Component pin names – 31
- Net names – 255
- External pin names – 31
- Path names – 31
- Design documentation comments – 255

Specification Errors and Problems

Specification errors and problems occur when names are misspelled or keywords or reserved prefixes are used.

Attribute and Option Names and Values

Design Attributes and Compiler Control Options names and values must be entered in the correct format shown in Chapters 2 and 3. Improper spelling and misuse of the names and values are flagged by the compiler as errors or warnings. Use the log (.log) and report (.rpt) files from the compiler to verify what you input versus what the program processed.

Keywords

Keywords are reserved identifiers that cannot be used to name designs, pins, nodes, constants, sets, macros, or signals. Keywords are case insensitive. The following is a list of the keywords:

ALLOC	PART	RST
AUTHOR	PROJECT	TEST_OE
DESCRIPTION	PROJECTNAME	XRESET
DESIGN	RESET	XTEST_OE

Any keyword used in a design is converted to an internal name in the implemented design, and may not be accessible in its original form to the user. Avoid using keywords as user-specified names in a design.

When a keyword is used as an identifier in a design, the pDS+ Fitter prepends the keyword with the character sequence “_KWD_” before processing the design.

Reserved Prefixes

The pDS+ Fitter maintains user-specified names wherever possible. However, the logic synthesis process sometimes deletes some nets and introduces new nets and net names. These new net names relate to logic elements in the synthesized circuit.

To minimize the chance of new net names conflicting with user-specified net names, the compiler prepends the new net names with special character sequences called “reserved prefixes.” These reserved prefixes cannot be used at the beginning of any user-specified names in a design. The following is a list of the reserved prefixes:

AND	_INV_	_PLA_
BUF	_KWD_	_PIN_
DEF	_LAF_	_RES_
GND	_NC_	_VCC_
_HM_	_OR_	

If reserved prefixes are used at the beginning of a user-specified name or identifier, they are prepended by the “_RES_” character sequence before the design is processed.

Duplicate Names

Lattice Semiconductor recommends that you use unique names throughout your design, instead of repeating names. If a name is repeated in different contexts, the pDS+ Fitter may remove or modify the name before generating the correct output. For example, a particular name may not be used as an external pin name as well as an internal net name (or instance name in schematic-entry systems). Reviewing the log and report files is the best method of identifying and then correcting this situation.

Optimizing Your Design

In general, optimizing a design for speed (delay) usually implies using more logic resources. Similarly, optimizing a design for resource utilization area usually implies lower speed. These objectives are often contradictory; there is a trade-off between area optimization and delay optimization.

The time required to compile very large or dense designs can be significant. In some cases you may need to maximize the compiler's efficiency to minimize the time required to complete your design.

The next two sections provide guidelines for design optimization.

Resizing your Design

If the report files indicate that your design is too large for the intended device, you have the following four options:

- Choose a larger device
- Reduce your design size (remove some logic)
- Optimize your design for device resource utilization
- Relax design constraints

The next section, "Optimizing for Resource Utilization," explains how to choose Compiler Control Options and Design Attributes to achieve higher logic density.

Optimizing for Resource Utilization

The primary Compiler Control Options and Design Attributes that affect logic density, and hence chip area utilization, are the following:

- CRIT
- EFFORT
- LOCK
- LXOR2
- MAX_GLB_IN
- MAX_GLB_OUT
- PRESERVE
- PROTECT
- SCP/ECP, SAP/EAP, and SNP/ENP
- STRATEGY

If your primary consideration is placing the largest amount of logic into a particular device, do the following:

1. Remove all PRESERVE attributes to allow the compiler to optimize your design better.
2. Use the USE_GLOBAL_RESET control option to allow the compiler to use the a global reset pin instead of an I/O pin.
3. Remove all CRIT attributes to allow the compiler to use the Output Routing Pool.
4. Try different levels of EFFORT.
5. Remove all pin specifications (LOCK) to allow the compiler to choose pin locations.
6. Remove all LXOR2 attributes to allow the compiler to decide on XOR usage when appropriate.
7. Increase MAX_GLB_IN to allow the compiler to use GLB resources more extensively.
8. Increase MAX_GLB_OUT to allow the compiler to use GLB resources more extensively.
9. Remove all PROTECT attributes to allow the compiler to optimize your design better.
10. Remove all path restrictions (SCP/ECP, SAP/EAP, or SNP/ENP) to allow the compiler to use any possible mapping scheme.
11. Use hard macros in your design wherever possible.
12. Use STRATEGY AREA.

Although the above guidelines provide a good starting point for achieving a denser implementation of your design in an ispLSI or pLSI part, the methods employed by the pDS+ Fitter do not always produce optimum results, and unexpected results may occur at times. When you encounter unexpected results, you should try different Design Attributes and Compiler Control Options.

Caution should always be used when trying extreme values for Compiler Control Options, or extensive use of one or more Design Attributes. This may lead to a denser implementation of your design, but may result in an unsuccessful routing.

Improving Routability

Device routing, device resource utilization, and compiler efficiency are controlled by Design Attributes and Compiler Control Options. Choosing optimal values for routability is a trial and error process due to the complex nature of the compilation process and its dependency on the characteristics of the design. The following sections focus on how to choose Design Attributes and Compiler Control Options to achieve the best overall results.

Optimizing for Routability

If you determine that your design is not too large for the intended device, yet your design does not pass through the compiler because of routing or resource limitations, then your design is probably overconstrained and you need to relax some attributes or parameters to achieve a routable design.

Table A-1 lists Design Attributes and Compiler Control Options in order of their effectiveness in improving routability and resource utilization; the most difficult ones for the compiler are listed first. You can use this table as a guideline to determine which attributes have the most impact on the compile process. However, these are only guidelines. A thorough knowledge of the device architecture and of your design is your best tool for determining the best combination of Design Attributes and Compiler Control Options.

Table A-1. Design Attributes and Compiler Control Options

Design Attribute or Compiler Control Option	**How to Improve Routability and Device Resource Utilization**
PART	This parameter determines device type and the resources available to your design. Choose a larger device if your design is unroutable due to lack of resources.
STRATEGY	The pDS+ Fitter has two unique fitting methods that it can use to fit your design. Each method is optimized for a specific type of implementation objective. You can determine which method works best for your design through the use of the STRATEGY attribute. STRATEGY AREA tends to produce more routable designs.
EFFORT	Try different EFFORT levels for best routability.
LOCK	LOCK assigns I/O pins to signal names. However, LOCK restricts optimal utilization of device resources. Remove LOCK attributes wherever possible for better routability.
CRIT	The CRIT attribute restricts output routing. Some combinations of CRIT and LOCK, or CRIT and CLK, can result in an infeasible design. Remove unnecessary CRIT properties to improve routing.
SCP/ECP	Critical Path attributes restrict routing and can decrease resource utilization. Remove unnecessary SCP/ECP attributes to improve routing and resource utilization.
SAP/EAP	Asynchronous Path attributes prevent the router from duplicating GLB outputs, thus decreasing routability. Remove unnecessary SAP/EAP attributes to improve routing.
SNP/ENP	No-Minimize Path attributes restrict optimization of your design. Remove unnecessary SNP/ENP attributes to increase resource utilization.
PRESERVE	The PRESERVE attribute restricts optimization of your design. Remove unnecessary PRESERVE attributes to increase resource utilization.
PROTECT	The PROTECT attribute restricts optimization of your design. Remove unnecessary PROTECT attributes to increase resource utilization.
GROUP	The GROUP attribute restricts optimization of your design. Remove unnecessary GROUP attributes to increase resource utilization.

Table A-1. Design Attributes and Compiler Control Options (Continued)

Design Attribute or Compiler Control Option	**How to Improve Routability and Device Resource Utilization**
CLK	Remove unnecessary CLK properties to improve resource utilization.
MAX_GLB_IN MAX_GLB_OUT	Limiting the usable GLB inputs and/or outputs increases routability at the expense of resource utilization. For example, for a 1000 family device, a value of 12 or below for MAX_GLB_IN, and a value of 3 or below for MAX_GLB_OUT usually lead to more routable designs.
ISP	The ISP option requires four I/O pins. Use ISP OFF to improve resource utilization and routability.
ISP_EXCEPT_Y2	This option allows the Y2 pin to be used as a clock input and can increase clock resource utilization (applies only to the ispLSI 1016 and ispLSI 2032).
Y1_AS_RESET	This option uses the Y1 pin as a reset input and decreases the available clock resources (applies only to the ispLSI and pLSI 1016 and 2032).
OPTIMIZE	Use OPTIMIZE OFF (default) to select hard macros, which are optimized for speed or resource utilization. Hard macros require less time to compile. Use OPTIMIZE ON to select soft macros, which may provide better routing for some designs.
REGTYPE	REGTYPE restricts the placement of registers by specifying where to place a particular register, either inside a GLB or inside an IOC.
USE_GLOBAL_RESET	USE_GLOBAL_RESET ON can improve routability of your design if all your registers and IOC latches are driven by a direct (no-logic) reset signal.
PULLUP	The PULLUP option has no effect on routing or utilization.
SECURITY	The SECURITY option has no effect on routing or utilization.
SLOWSLEW	The SLOWSLEW option has no effect on routing or utilization.

Design Simulation

Lattice Semiconductor recommends that you simulate your design before and after implementation by the pDS+ Fitter. The following are some points to remember when simulating your design:

- Signals representing power and ground lines (VCC and GND nets) should be properly asserted if the simulator does not understand these signals as special nets before simulation can begin.
- !XRESET should be toggled to low to globally reset all registers. If you are using an ispLSI/pLSI 1016 or 2032 with Y1_AS_RESET set to OFF, no global reset is available, and registers can only be reset if the Product Term reset is defined for them.
- XTEST_OE should be set to high if an ispLSI/pLSI 3192 or 3256 is used.
- If you use any keyword as a user-specified signal name, or if you use a reserved prefix as part of a user-specified signal name, the name is changed by the pDS+ Fitter and is not available to the simulator in its original form.
- Pin names are retained in a timing simulation netlist. However, internal names may not be accessible to a timing simulation netlist. A PRESERVEd net name is available in a timing simulation netlist if it is not inactive, and if it is not an internal name similar to an external pin name. However, the compiler may duplicate the PRESERVEd net, thereby modifying its name. Use SAP/EAP to prevent duplication of a particular net.

Improving a Working Design

This section provides guidelines for making changes to a design that has already been compiled successfully. Even minor changes can produce a very different layout after recompiling. Therefore, you need to understand the implications of your changes, especially if your original design was difficult to compile.

The following guidelines are recommended:

- Make a copy of your working design before experimenting with changes. If you recompile a working design without making changes, it will have the same physical layout as before.
- Do not try to keep all pin assignments. Locking the assigned pin numbers before recompiling severely restricts the compiler and may cause your design to be unroutable. Assigning only a few pins provides a guideline for the compiler and, depending on the amount of changes you made, results in a very similar layout.
- Use the PRESERVE attribute with caution. Removing PRESERVE restrictions can free some device resources, but can also produce a very different layout.
- Apply the CRIT attribute carefully. Because only two of the four outputs of a GLB in the pLSI 1000 and pLSI 3000 device families can have CRIT attributes (to specify the ORP bypass), you could cause the GLB logic to be specially grouped. If device resources are very limited, this could cause your design to become unroutable.
- Use moderation in making any changes to the Compiler Control Options (MAX_GLB_IN, MAX_GLB_OUT, and so on.) These changes have a global effect and are likely to cause a very different layout of your design.
- To modify the SECURITY, PULLUP, or SLOWSLEW attributes, you must recompile your design. If you have made no other changes, your design will work exactly as before; the only differences will be in the JEDEC device programming file.

Improving a working design requires using the Design Attributes and Compiler Control Options to specify your design needs. By default, the compiler implements a globally optimized design.

The report file represents implementation of logic, and the log file may include some warnings that can normally be ignored. Designers who want to fine-tune their design can use the report and log files to identify logic which must be implemented in certain ways to meet their specific needs. These requirements need to be identified and specified for the compiler before a desirable implementation can be achieved.

Optimizing for Speed

The primary compiler properties that affect speed (input to output propagation delays) are the following:

- CRIT
- EFFORT
- PRESERVE
- SCP/ECP
- STRATEGY
- USE_GLOBAL_RESET

If your primary consideration is achieving the fastest possible design, do the following:

1. Specify **`STRATEGY DELAY`** to reduce the number of logic levels globally.
2. Remove **`PRESERVE`** attributes wherever possible to allow the compiler to remove nets during optimization, instead of forcing the nets to a GLB or IOC output.
3. Specify **`USE_GLOBAL_RESET`** to move a common reset line to a global reset pin.
4. Specify **`CRIT`** to use the ORP bypass on the outputs that need to take out fast signals.
5. Specify **`SCP/ECP`** to mark all appropriate paths as critical.
6. Try different levels of **`EFFORT`**.

Other Design Attributes can also be used to achieve faster speed. Larger values of MAX_GLB_IN normally results in a wider logic and less GLB levels. No-minimize paths (SNP/ENP), along with the PRESERVE attribute, can also be used to closely duplicate implementation of a piece of logic in a certain way to meet specific design requirements. Asynchronous paths (SAP/EAP) can be used to correct timing problems caused by signal skew.

Design Run-Time and Memory Requirements

The pDS+ Fitter typically requires a reasonable amount of run-time and memory. This requirement is highly design dependent. To improve run-time and memory requirements of your design use a lower **`EFFORT`** level.

Design Rules

The following sections describe design rules that you should observe in your design to make it conform better to the Lattice Semiconductor device architecture. The pDS+ Fitter conforms to these rules by modifying the user netlist or relaxing the constraints automatically. Warnings may be issued if the compiler changes your design significantly to conform to these design rules.

If the resulting netlist does not meet your requirements, use Design Attributes and Compiler Control Options to direct the compiler toward your implementation objectives. Sometimes the netlist cannot be mapped or routed if it is too complex or overconstrained. A thorough knowledge of the design rules and device architecture is needed to concisely direct the compiler.

I/O Pin Designations

- Minimize the use of locked pins on initial design implementations. This provides the partitioner and router maximum freedom in partitioning and routing the design.
- Only lock non-registered inputs to the dedicated input pins to improve usage of IOC registers.
- Do not lock two signals to the same pin.
- Do not lock data I/O pins to clock signals.
- Do not lock outputs using the same output enable (OE) signal to different megablocks. This forces duplication on the OE signal and uses OE resources.
- Do not lock two or more external inputs to the same GLB if they are from IOCs that are a multiple-of-16 apart. This forces the addition of a logic level. For example, pin 26 (I/O0) and pin 45 (I/O16) cannot supply signals to the same GLB in a pLSI 1032-LJ84 device.
- Do not lock two or more external outputs supplied by the same GLB to IOCs that are a multiple-of-4 apart. For example, pin 26 (IO0) and pin 34 (IO8) cannot receive signals from the same GLB in a pLSI 1032-LJ84 device.
- Do not lock signals to the ISP pins if you are using the ISP option. Some dedicated inputs become unavailable for routing if the ISP option is enabled.
- For an ispLSI 1016 and ispLSI 2032 device, selecting the ISP option causes some pins and dedicated inputs, including Y2, to become unavailable for use by the compiler.
- I/O pins that lead to logic that was eliminated during logic optimization are removed from the design.

Global Reset Signal

- The external reset input automatically connects to every register in the device; do not connect internal nets to the $\overline{\text{RESET}}$ pin.

Output Enable Signals

- Do not use more than one OE signal per megablock for pLSI 1000 and 2000 parts. Do not use more than one OE signal per two megablocks for a pLSI 3000 part.
- The OE signal must reside in the same megablock as the IOCs to which it is connected.

Generic Logic Blocks and Megablocks

- Connect a maximum of two dedicated inputs to a single Generic Logic Block (GLB).
- Do not lock dedicated inputs in two different megablocks if they are inputs to the same GLB.
- Do not use a locked dedicated input and a locked output from different megablocks for the same GLB.
- Do not use a locked dedicated input to generate an OE in one megablock to enable IOCs in a different megablock.
- An IOC and its OE must be in the same megablock.
- A GLB can have no more than 18 inputs, two of them dedicated input pins, in the pLSI 1000 or pLSI 2000 device families. (The pLSI 3000 device family does not have dedicated input pins.)
- A GLB can have no more than four outputs; two of them can use the ORP bypass in the pLSI 1000 or pLSI 3000 device families.
- A dedicated input cannot drive more than eight GLBs.

Nets

- Each net in your design must have only one source.
- Do not connect internal nets to the VCC and GND pins.
- Do not use VDD or VSS as power lines. These nets are not recognized as constants by the compiler.
- Internal 3-state nets and buffers are not supported.

Clock Usage

- Lock clock signals only to a clock pin if they do not drive other logic.
- Do not lock clock signals to I/O pins.
- Do not lock a signal to the Y3 or Y4 pin if the signal is used as a data line.
- Do not lock a signal to the Y0 or Y1 (or Y2 for the pLSI 3000 device family) pin if the signal clocks IOCs. (Y1 can connect to an IOC clocks in a pLSI 1016 part.)
- Do not use a signal from an IOC as a fast clock, unless it first passes through the dedicated clock GLB.
- Only one GLB per design can generate internal fast clock signals in 1000 devices.
- A design can have a maximum of two GLB and two IOC clocks from the clock GLB where available.
- The clock GLB can only use locked dedicated inputs that belong to the same megablock as the clock GLB.
- A design can have a maximum of five global clocks in the pLSI 1000 and pLSI 3000 device families, and three global clocks in the pLSI 2000 device family. This limitation includes three GLB clocks and two I/O clocks where available.
- A design can have a maximum of five external global clocks (Y0 to Y4) in the pLSI 3000 device family, a maximum of four external global clocks in the pLSI 1000 device family (except pLSI/ ispLSI 1016), and a maximum of three external global clocks in the pLSI 2000 device family (and pLSI/ispLSI 1016).
- All pLSI 1000 devices except the pLSI/ispLSI 1016 have only one external global clock, Y2, that supplies both GLBs and IOCs. In the pLSI 1016 and ispLSI 1016, both Y1 and Y2 can supply both GLBs and IOCs, provided that Y1 and Y2 are not set to perform other functions.
- If you specify Y1 as a clock signal, you get an error if Y1_AS_RESET is ON. (Y1_AS_RESET applies only to the pLSI or ispLSI 1016 and 2032.)
- Pin Y1 can drive IOCs only on the pLSI 1016; other devices must use Y2 or Y3 to clock IOCs where IOC registers are available.
- Do not use internally-generated clock signals as data lines.

Glossary

386 Enhanced Mode	A mode Microsoft Windows runs to access the extended-memory capabilities of the 80386 processor. Windows uses memory more effectively.
Application Window	The window containing the work area and menu bar for an application. The application window has its name at the top of the window.
Array	The area occupied by the rows of modules and the routing interconnects.
Asynchronous	Data that is not synchronous with a clock signal. The next I/O may start operation before the current one is finished. The output responds immediately to a change in the input signal.
Attribute	Design constraint data specified during logic entry.
Back Annotation	The process of translating data generated by the pDS+ system to the CAE design environment. Post route timing delay information is back annotated to the CAE simulator.
Boolean	The “mathematics of logic” developed by George Boole in the nineteenth century, based upon the rules and operations of logical functions rather than numbers. AND, NOT, and OR are the primary operations of Boolean logic.
Browse	A button on some screens that opens a dialog box that list files or directories from which you choose.
Cascade In (CI)	Input to a counter that is used to connect the output from a previous stage.
Cascade Out (CO)	Output from a counter that is used to connect the input to a subsequent stage.
Cell	An elementary unit of storage for data.
Clock Distribution Network	Interconnection location of the clock signals.

Clock GLB	A GLB that can be used to generate global gated clocks to drive GLB or IOC registers.
Command	A word or series of words used to carry out a directive. A command is either typed at a prompt or selected from a menu.
Compiler	The pDS+ Fitter uses architecture-specific methods to synthesize a logic description into a ispLSI or pLSI device. The compiler determines if the logic can fit into the assigned GLBs and IOCs. It maps logic to the cells and provides input to the fusemap generation process. The compiler updates the netlist used by the place and route process.
Configure	Process for determining placement and routing for a design.
Control Option	A compiler control option that can change the normal operation of the software when specified differently from its default value. This results in a different implementation of the design.
Conventions	Rules that govern design entry for names and notations.
Critical Net	A network whose signal propagation delay is part of the critical path in the design.
Dedicated Clock Input Signals	Signals from the dedicated clock input pins that go through the clock distribution network to the global clocks.
Dedicated Input (DI)	Inputs that bypass the Global Routing Pool (GRP) and go directly to the GLBs. These signals are megablock-specific.
E^{2}CMOS®	Electronically Erasable CMOS logic.
EDIF	Electronic Design Interchange Format.
Fanout	The number of destination inputs driven by a source signal.
Fusemap	A design file that contains a list of E^{2}PROM fuse addresses used by the programming hardware to program the device.
Fusemap Generation	Fusemap generation (process) creates the programming file that is used to program a device, and is the last step in the design process.
Generic Logic Block (GLB)	The basic logic element in the ispLSI and pLSI architecture.
GLB Clocks	Clock signals used to drive GLB registers.
GLB-Generated Clocks	Clock signals generated from a Clock GLB that go through the Clock Distribution Network to the global clocks.

Global Clocks	The clocks used to drive either GLB or IOC registers globally.
Global OE	Output enable signal from the GOE pin that can be used to enable any or all IOCs in the device.
Global Reset	A signal that resets all the registers in the device.
Global Routing Pool (GRP)	Interconnection location of the internal logic. The GRP provides complete interconnectivity with fixed and predictable delays.
Hard Macro	A GLB-level macro that is predefined and cannot be edited.
Input/Output Cell (IOC)	Each I/O Cell is directly connected to an I/O pin and can be programmed for combinatorial input, registered input, latched input, direct output, 3-state output, or bi-directional I/O.
Input/Output (IOC) Clocks	Two clock signals, IOCLK0 and IOCLK1, that are used for clocking all of the IOC registers in the device.
ispLSI	An acronym for in-system programmable Large Scale Integration. Allows programming of a device on a PC board, and requires no external programmer.
JEDEC File	File in the format prescribed by the Joint Electronic Device Engineering Council.
LSC Advanced Format (LAF)	A design file in ASCII format used for an intermediate design representation.
LSC Design File (LDF)	A design file in ASCII format used to enter a design into pDS.
LSC Internal Format (LIF)	The binary file format for a design used by pDS and pDS+.
Macros	Predefined, reusable logic blocks that reduce the amount of time necessary to enter the equivalent Boolean equation.
Megablock (MB)	A megablock consists of a group of eight Generic Logic Blocks (GLBs), Output Routing Pools (ORPs), and I/O Cells (IOCs) coupled together. The various members of the ispLSI/pLSI families are created by combining several megablocks on a single device.
Megacell (MC)	One of two halves of a megablock in an ispLSI/pLSI part.
Naming	Conventions that define the signal (net) names, syntax entries, and pin numbers.

Net	A logic signal path between logic elements containing the source, signal, and destination.
Netlist	A tabular format report that contains the net name, source and destination, GLB locations, and fanout data.
Notation	Conventions that define the style and format for syntax entries.
Output Enable (OE)	Logic signal that enables the output of an IOC.
Output Routing Pool (ORP)	The Output Routing Pool connects the Generic Logic Blocks (GLBs) output to the I/O Cells (IOCs).
Partitioning	Dividing a design into functional blocks. These blocks can be a few components or multiple circuits with numerous components. The design is organized to meet the capabilities of the targeted device.
pDS	The pLSI and ispLSI Development System software package that is used to implement designs in ispLSI and pLSI devices through Boolean equation entry and manual partitioning.
pDS+	The pLSI and ispLSI Development System Plus software package that is used to implement designs in ispLSI and pLSI devices through automatic partitioning.
pLSI	An acronym for programmable Large Scale Integration. Allows programming of a device to meet specific design criteria using an external programmer.
Product Term (PT)	A term generated by one of the twenty AND gates within a GLB. The inputs to the GLB are ANDed to produce the product term which can be used as a logic element, a product term clock (PT clock), a product term reset (PT reset), or a product term output enable (PTOE).
Pull-ups	Allow the holding of floating inputs at a known state. They are useful in debugging a design and reducing noise interference.
Report Files (rpt)	A method for supplying information to users covering Design Analysis, GLB Resources, External Pins, and Routing.
Router	An automated tool that uses the device files and design files to place design components, GLBs, and IOCs, and then route the interconnections. The Router reads design constraint data specified during logic entry from the design netlist.
Soft Macro	Predefined blocks of logic consisting of macros and primitives which can be edited. Mapping, placement and routing is not predetermined for soft macros.

Index

O

P

R

S

T

U