FPGA Design with ispLEVER Tutorial

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

September 2011

Copyright

Trademarks

Trademarks

Lattice Semiconductor Corporation, L Lattice Semiconductor Corporation (logo), L (stylized), L (design), Lattice (design), LSC, CleanClock, E^2CMOS, Extreme Performance, FlashBAK, FlexiClock, flexiFlash, flexiMAC, flexiPCS, FreedomChip, GAL, GDX, Generic Array Logic, HDL Explorer, IPexpress, ISP, ispATE, ispClock, ispDOWNLOAD, ispGAL, ispGDS, ispGDX, ispGDXV, ispGDX2, ispGENERATOR, ispJTAG, ispLEVER, ispLeverCORE, ispLSI, ispMACH, ispPAC, ispTRACY, ispTURBO, ispVIRTUAL MACHINE, ispVM, ispXP, ispXPGA, ispXPLD, Lattice Diamond, LatticeCORE, LatticeEC, LatticeECP, LatticeECP-DSP, LatticeECP2, LatticeECP2M, LatticeECP3, LatticeMico, LatticeMico8, LatticeMico32, LatticeSC, LatticeSCM, LatticeXP, LatticeXP2, MACH, MachXO, MachXO2, MACO, ORCA, PAC, PAC-

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<*Italic*>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
`Courier`	Code examples. Messages, reports, and prompts from the software.
`...`	Omitted material in a line of code.
`.` `.` `.`	Omitted lines in code and report examples.
`[ ]`	Optional items in syntax descriptions. In bus specifications, the brackets are required.
`( )`	Grouped items in syntax descriptions.
`{ }`	Repeatable items in syntax descriptions.
`|`	A choice between items in syntax descriptions.

Contents

FPGA Design with ispLEVER Tutorial

Introduction

This tutorial is intended for a new user or a user who uses ispLEVER infrequently. It shows you how to use several processes, tools, and reports from the ispLEVER software suite to implement a simple (RTL) Verilog or VHDL design in a LatticeEC family device. You will prepare the design for simulation, power estimation, static timing analysis, and timing-driven placement and routing, reviewing output reports as you make progress. You will constrain the design's inputs and outputs to meet the signal type and location requirements of your system-level design. You will modify and constrain the design to leverage the architectural resources of the LatticeEC device to give you a high-performance implementation. The tutorial covers the most common procedure and software options, so you will have a base of understanding before you tackle larger or more timing-critical designs that may require more refinement and control to meet your performance and utilization objectives.

Learning Objectives

When you have completed this tutorial, you should be able to do the following:

- Use ispLEVER to create a new Verilog HDL or VHDL project, target a device, and add a Verilog HDL or VHDL source file to the project using Project Navigator.
- Generate a sysCLOCK PLL module, add it to your project, and refer to it from your source using IPexpress and Text Editor.
- Lock signals to device package pins and define a period or frequency and clock-to-out timing constraints using Design Planner.
- Implement the design using the mapping, placing, and routing processes and view the resulting reports using Project Navigator.

- Interpret the static timing analysis report and adjust the design to meet your timing objectives.
- Modify and constrain the design to meet your performance objectives.
- View the device implementation and review the relative routing congestion and programmable functional unit (PFU) utilization using Design Planner.
- Estimate power consumption using Power Calculator.
- Prepare the design for simulation.

Time to Complete This Tutorial

The time to complete this tutorial is about two and a half hours.

System Requirements

One of the following software configurations is required to complete the tutorial:

- ispLEVER
- ispLEVER Pro
- ispLEVER Starter

Accessing Online Help

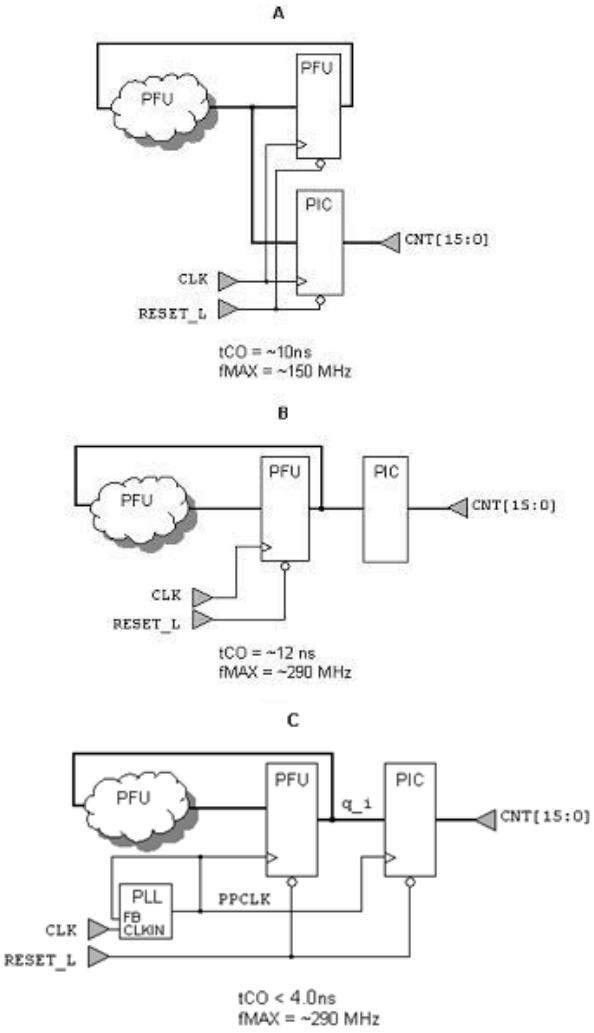
You can find online help information on any tool included in the tutorial at any time by pressing the F1 key.

About the Tutorial Design

The tutorial design begins with a simple counter described with register-transfer-level (RTL) Verilog or VHDL that you might find in any Verilog or VHDL textbook. It evolves into a more sophisticated version that uses registered outputs and a phase-locked loop to help reconcile the internal timing of the FPGA with the external specifications. The design is modified and constrained to account for the f_{MAX} switching characteristics of the LatticeEC device (programmable function units (PFUs) as compared to programmable interface cells (PICs)) and special features like the sysCLOCK PLL to leverage the feedback compensation of the internal VCO to reduce clock-to-output delay (t_{CO}).

You can perform the tutorial with either a Verilog HDL or VHDL source file. The complete tutorial design is composed of two Verilog or VHDL modules: a 16-bit up counter with an active-low asynchronous reset and a PLL. Figure 1 illustrates versions A through C of the tutorial design and the resulting timing characteristics. The counter is described with register-transfer-level (RTL) Verilog or VHDL and a structural model produced by an ispLEVER application called IPexpress, which describes the PLL module. The counter's 250-MHz clock is driven externally in versions A and B of the design. In version C, it is produced by a sysCLOCK PLL that is introduced to effectively remove from the critical path timing equation the route delay introduced by the clock tree, allowing easier timing analysis at the PC board.

Figure 1: Tutorial Design



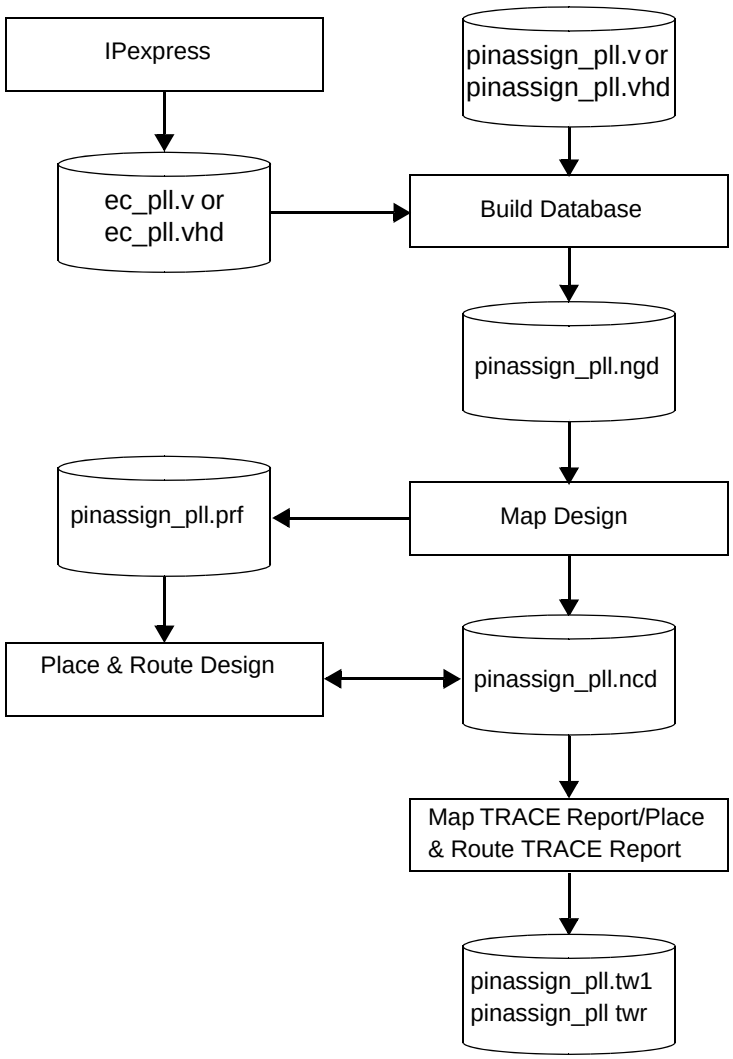
Note

In this tutorial, where the step to be performed depends on the type of source file you are using, the step is prefaced by “Verilog:” or “VHDL:.” Where a sequence of steps diverge for Verilog or VHDL, they are grouped by headings that indicate their application to Verilog or VHDL.

About the Tutorial Data Flow

A few key processes of the ispLEVER Project Navigator are used to transform the Verilog or VHDL source code of the tutorial design into Lattice Semiconductor databases that are used by the software system to implement and analyze the logic. Figure 2 illustrates the tutorial data flow through the system. You may find it helpful to refer to this diagram as you move through the tutorial tasks.

Figure 2: Tutorial Design Flow



Restore the Tutorial Files

If this tutorial has been previously run on your system, use the following procedure to restore the original tutorial files.

To restore the original tutorial files:

1. Go to the following folder:

   <*install_path*>\examples\Tutorial**fpga_design_tutor**

2. Delete the following files. They may have been modified:
   - **pinassign_pll.lpf**
   - **pinassign_pll_a.v** (or **pinassign_pll_a.vhd**)
3. Make a copy of the **pinassign_pll_a.v** (or **pinassign_pll_a.vhd**) file and rename the copy **pinassign_pll.v** (or **pinassign_pll.vhd**).

Task 1: Create a New Verilog or VHDL Project

In this task, you will create a new Verilog HDL- or VHDL-type project and provide it with a simple title using Project Navigator. To begin a new project, you give the project file (.syn) a name and declare the project type. The ispLEVER software saves an initial design file with the .syn file extension in the directory that you specify. All project files are typically created in this directory, but you can reference source files from outside of it. The project type specifies that all design sources will be of this type.

Create a New Project

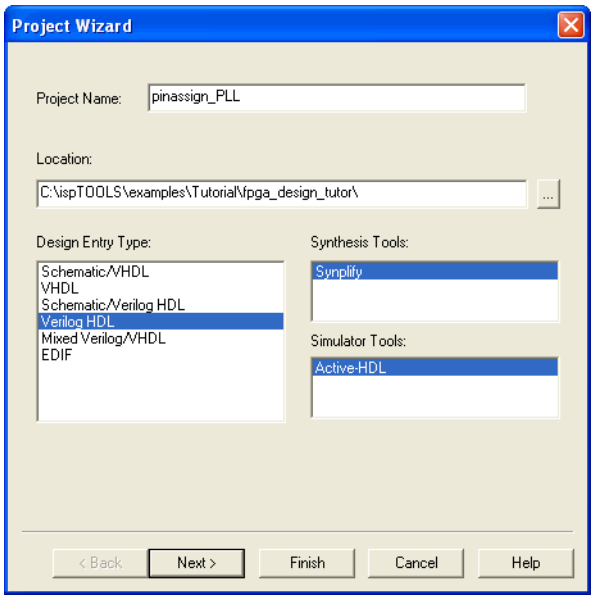
If you want to preserve the original tutorial files, save the fpga_design_tutor directory to another location on your computer before proceeding.

To create a new project:

1. Start Project Navigator if it is not already running.
2. In Project Navigator, choose **File > New Project** to open the Project Wizard dialog box.
3. In the Project Wizard dialog box, shown in Figure 3, do the following:
   a. In the Project Name box, type **pinassign_PLL**.
   b. In the Location box, specify the following directory:

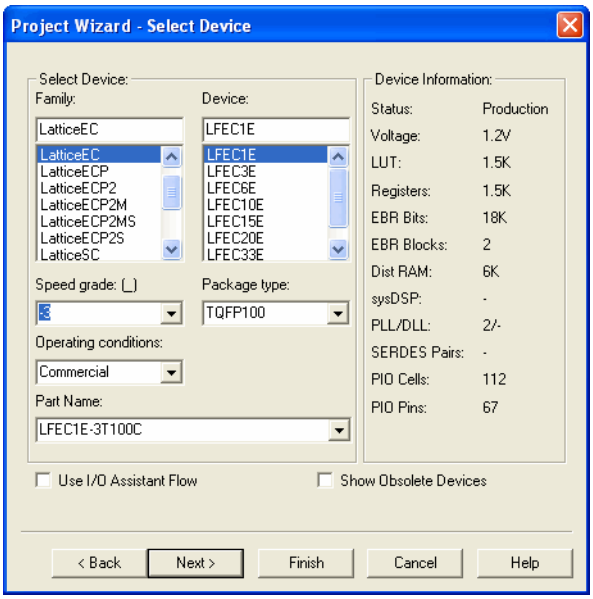
      <install_path>\examples\Tutorial**fpga_design_tutor**
   c. In the Design Entry Type box, choose **Verilog HDL** or **VHDL**.
   d. In the Synthesis Tools box, choose **Synplify**.
   e. In the Simulator Tools box, choose your preferred simulator.
   f. Click **Next**.

Figure 3: Project Wizard Dialog Box



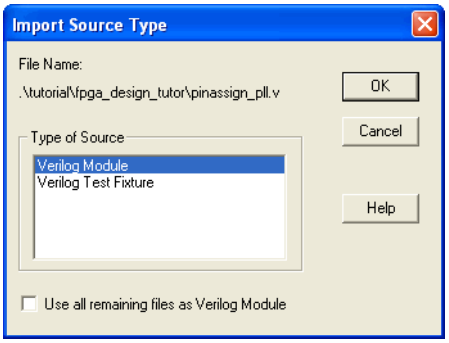
4. In the Project Wizard – Select Device dialog box, shown in Figure 4, do the following:
   a. In the Family box, choose **LatticeEC**.
   b. In the Device box, choose **LFEC1E**.
   c. In the Speed Grade box, choose **-3**.
   d. In the Package Type box, choose **TQFP100**.
   e. In the Operating Conditions box, choose **Commercial**.
   f. Click **Next** to open the Project Wizard – Add Source dialog box.

Figure 4: Project Wizard – Select a Device Dialog Box



5. In the Project Wizard – Add Source dialog box, click **Add Source** to activate the Import File dialog box.
6. Select **pinassign_pll.v** (or **pinassign_pll.vhd**) and click the **Open** button to open the Import Source Type dialog box, shown in Figure 5.

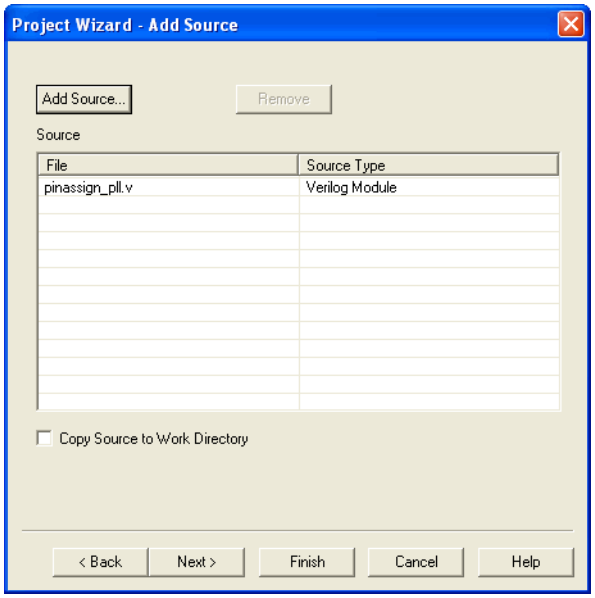
Figure 5: Import Source Type Dialog Box



7. Select **Verilog Module** (or **VHDL Module**) and click **OK**.

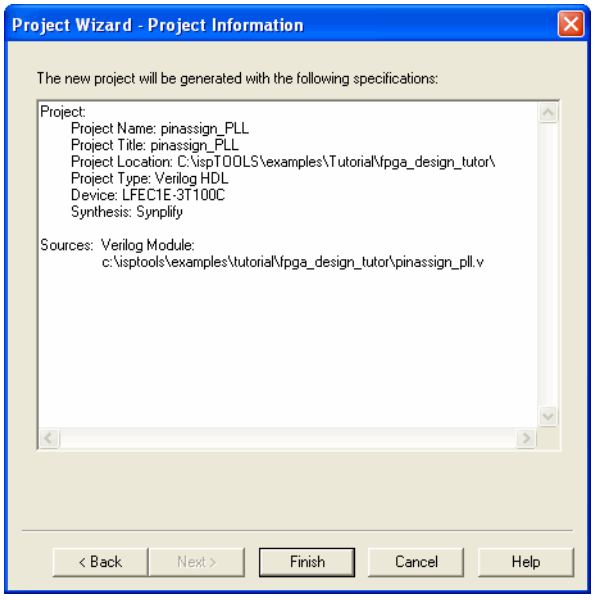
The file is added to the source file list in the Project Wizard – Add Source dialog box, as shown in Figure 6.

Figure 6: Project Wizard – Add Source Dialog Box



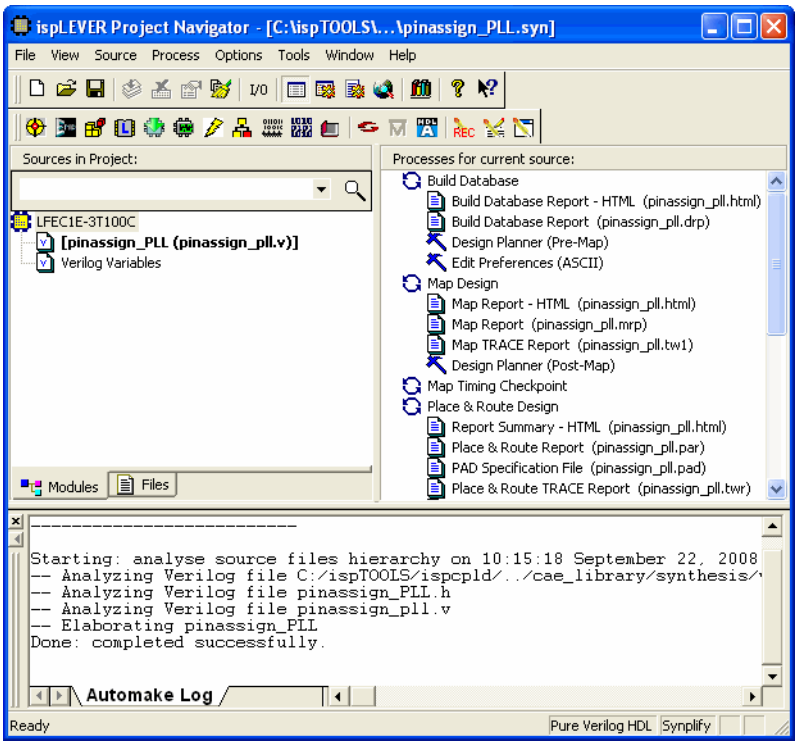
8. Click **Next**.
9. In the Project Wizard – Project Information dialog box, shown in Figure 7, click **Finish**.

Figure 7: Project Wizard – Project Information Dialog Box



The new project is created, and the source file is referenced in the Sources in Project list, as shown in Figure 8. Click on the part name to see the contents of the Processes for Current Source window.

Figure 8: Project Navigator Window Showing New Project



10. The Sources Window provides two tabs to organize the design modules and files related to the project. The Modules tab displays each unique module and related filename used in the design hierarchy. A text entry box and magnifying glass icon allows you to search the module list.

    In the Sources Window, click the **Files** tab. Project comments and source files related to the project appear. Folders such as Documents, Stimulus Files and Input Files help organize files. Project Navigator automatically compiles all HDL source files of the project to create the module hierarchy and an ordered file list for simulation and synthesis tools. If syntax faults occur or the top of design is ambiguous you may be prompted to specify the top. File order can be adjusted in those cases where automatic ordering does not work correctly.

11. In the Sources in Project window, double-click the project title, **pinassign_PLL**, to open the Project Properties dialog box.

    The default title for a new project is the project name that you initiated in step 3. You can create another title for the project with as many characters as you want. The title can contain spaces and any other keyboard characters, except tabs and returns.

12. In the Title text box, type **Pin assign and PLL sample** and click **OK**.

Note

You can target a design to another Lattice Semiconductor device later, if you want. For example, you might discover that you can target a smaller, less expensive device package if your design will fit, or you might decide to migrate an existing project to a new device family. To select a new device, choose **Source > Select New Device**.

View Project Source File

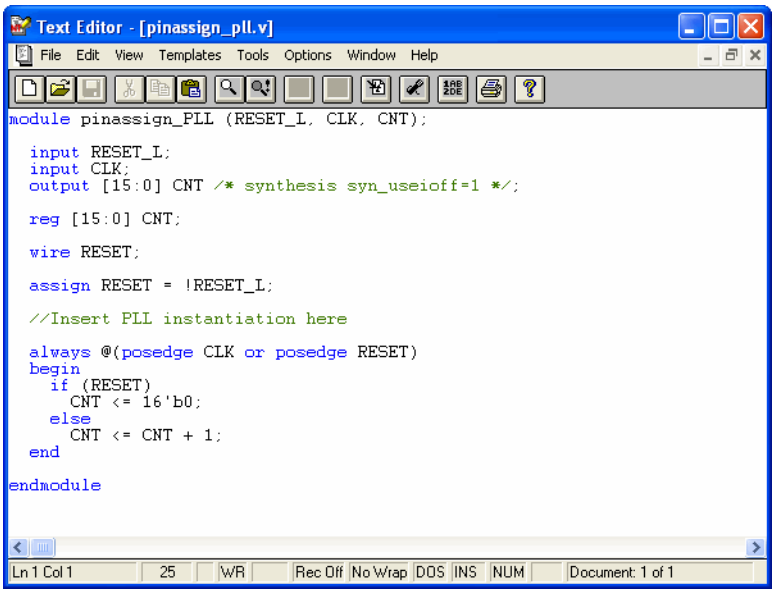
You "describe" a project by specifying the source files that represent the design. You can either import an existing source or create a new one. The source files for the project appear in hierarchical and alphabetical order within the Sources in Project window, Modules tab, by module name. If your source file contains more than one module, it will appear in the list as many times as there are module instantiations, using the style:

module_name (*source_filename*)

To view a source file of the project:

1. Click the **Modules** tab in the Sources in Project window.
2. Double-click the **pinassign_PLL** module to open the source file in Text Editor, as shown in Figure 9.

Figure 9: Source File in Text Editor



3. Choose **File > Exit** to exit Text Editor.

Adjust Tool and Environment Options

Timing checkpoints occur after both the mapping and the placement and routing stages of the process flow to flag excessive delay conditions. In this tutorial, you will set checkpoints that will report when a check fails but will not block forward progress.

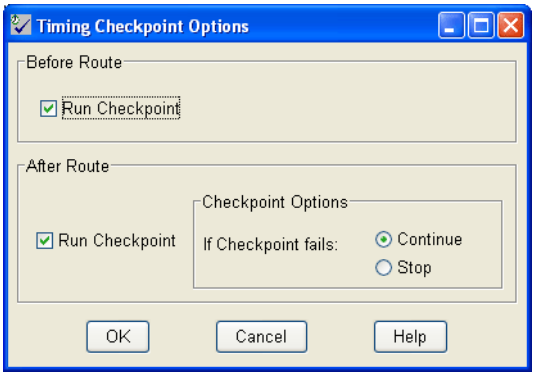
To adjust tool and environment options:

1. Choose **Tools > Timing Checkpoint Options** to bring up the Timing Checkpoint Options dialog box, shown in Figure 10.
2. In this dialog box, select the following:
   a. In the Before Route section, select **Run Checkpoint**.

b. In the After Route section, select **Run Checkpoint** and select **Continue** for the “If Checkpoint fails” option.

c. Click **OK**.

Figure 10: Timing Checkpoint Options Dialog Box



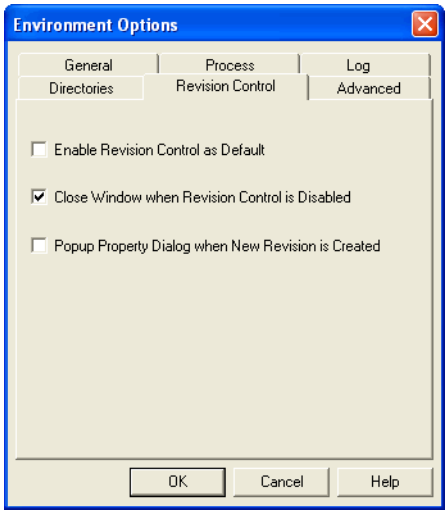
3. Choose **Options > Environment** to open the Environment Options dialog box, shown in Figure 11. In this dialog box, do the following:

   a. Click the **Log** tab and turn off **Using Web Browser**.

   b. Click the **Revision Control** tab and select **Close Window when Revision Control is Disabled**.

   c. Click **OK**.

Figure 11: Environment Options



4. If the Revision Control window is open in Project Navigator, right-click inside it and choose **Turn Off** from the pop-up menu.

   The revision control window closes. This tutorial will not use project revision control.

To re-open the Revision Control window, click on the Revision Control icon  in the Project Navigator toolbar.

Task 2: Assign Location and Timing Preferences

In this task, you lock signals to device package pins and define period or frequency and clock-to-out timing constraints. You use Design Planner, a graphical interface to the logical preference file (<*project_name*>.lpf), to create timing and location constraints for ispLEVER. The logical preference file is interpreted by the design mapper (Map Design process) to produce a physical preference file (<*project_name*>.prf) for the place-and-route system. Now that the logic design has been captured, you can create constraints that will guide the placement and routing process to locate signals at specific package pin locations and implement the logic and routing to meet your design's performance objectives.

Assign Pin Location Preferences

To assign location preferences to the pins with the Design Planner:

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. In the Processes for Current Source window, double-click the **Design Planner (Pre-Map)** tool to activate Design Planner.

   The Build Database process synthesizes the Verilog HDL or VHDL source and translates the synthesis output in EDIF to an NGD Logical Design database.
3. Click **OK** to close the ispLEVER Process warning message box that appears.

   Information and warning messages appear in the Automake Log tab of the output panel of Project Navigator. You can ignore these warnings in this tutorial.

   After a few moments, the Design Planner Control opens.
4. In the Design Planner Control window, choose **View > Default Window Layout**.

   Spreadsheet View and Package View now appear.
5. If it is not already selected, select the **Port Attributes** tab in the Spreadsheet View.

   The Spreadsheet View displays the Port Attributes tab for assignments related to location and buffer configuration. Preferences such as pin location, I/O type, and slew rate appear as columns in the sheet. It is common to use the Logic Signal Connections section of the device data sheet as a reference to type in pin locations.

6. Click anywhere on Spreadsheet View and verify that the **View > Show Default Value** option is selected.

   Note

   The menu items of the Spreadsheet View are context-sensitive and enable functions only if they apply to the window or pane that is selected. If a menu item is unexpectedly unavailable, click on the pane to which the function applies.

   Package View window provides a graphical representation of the 100-pin TQFP device package.

   In the next steps, you will drag and drop external signals to pin locations in Package View.

7. Double-click the title bar of the Package View to maximize the view.
8. Click anywhere in the right pane of the Package View window and choose **View > Top View**.

   The Package View window adjusts to display the device package as viewed from above.

9. Click on the + next to Device and next to Design Signals to expand the lists, and then expand the list of System pins and User pins.

   a. Under User pins, expand the PIO list.

   b. Find pin **99** in the expanded tree.

   c. Highlight pin **99** and right-click.

   d. Select **Locate Device Pin**.

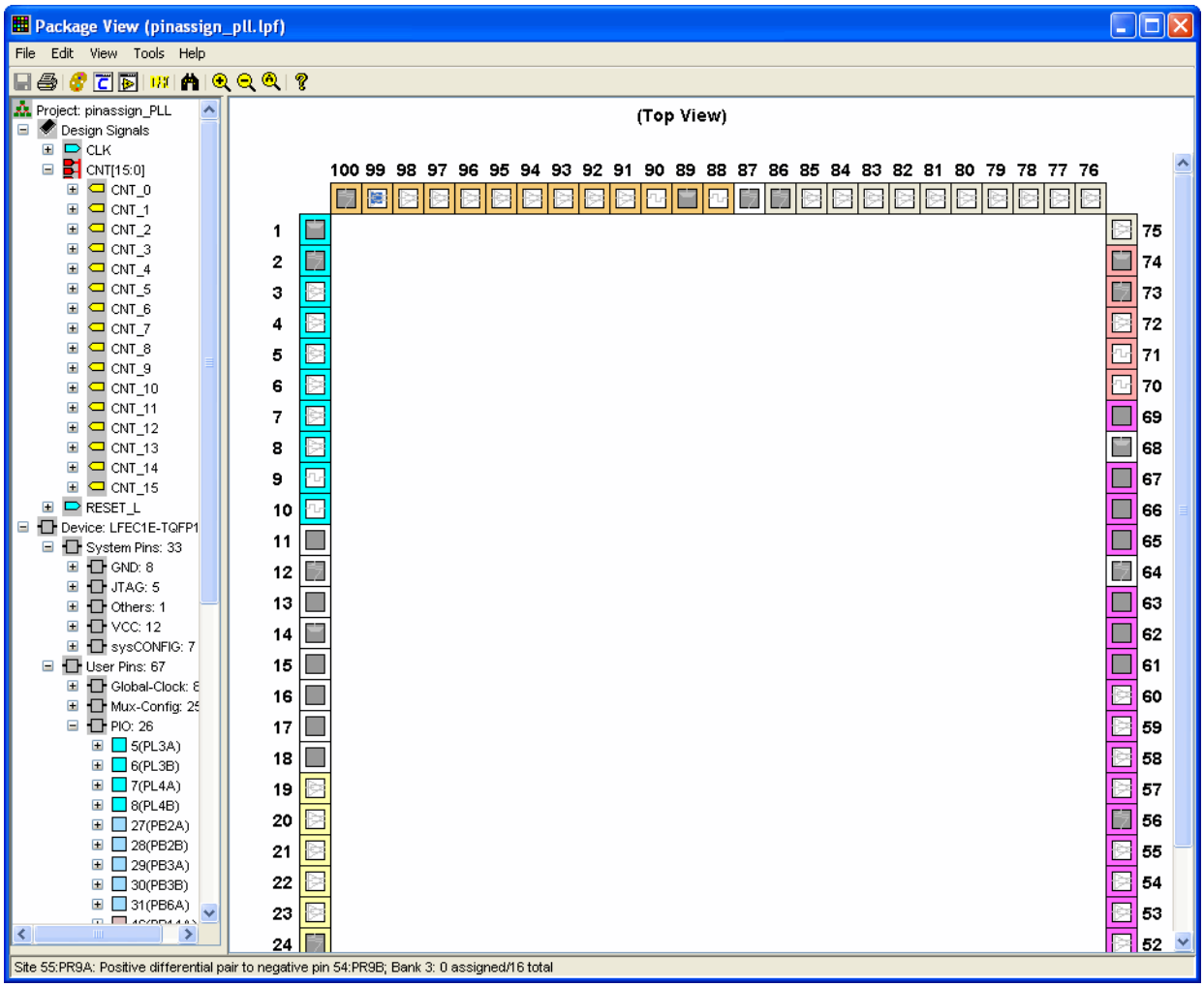
   The Package View expands the view and highlights pin 99.

   Note

   You can display the name of each pad by holding the mouse pointer over the pad.

10. Expand the **CNT[15:0]** bus from the Design Signals list, as shown in Figure 12.

Figure 12: List of Signals in Bus



11. Drag and drop the CNT[15:0] bus onto device pin 99.

    Note

    Make sure that you place the arrow connected to the bus squarely onto the device pin before releasing the mouse button.

    Design Planner fills the top left I/O bank (Bank 0) and a portion of the top right bank (Bank 1). The colored outlines of the pads represent banks. You can inspect specific assignments by holding the mouse pointer over a pad. A pop-up display provides details about the PIO the signal assigned to it.

12. Drag and drop the following input signals onto the pins specified:

    - RESET_L, pin 3 (global reset)
    - CLK, pin 19 (one of two reference clock pins for the LFEC1, 100-pin TQFP sysCLOCK PLL, which will be added later in the tutorial)

    As you assign pins, the Package View pin location is colored yellow to indicate output type ports, and the pin number appears next to the user

signal in the left pane of the Package View pane. The blue color represents inputs.

Spreadsheet View is also updated with the assignments that you have made. The Pin column of the Port Attributes tab displays the pin numbers assigned to each user signal.

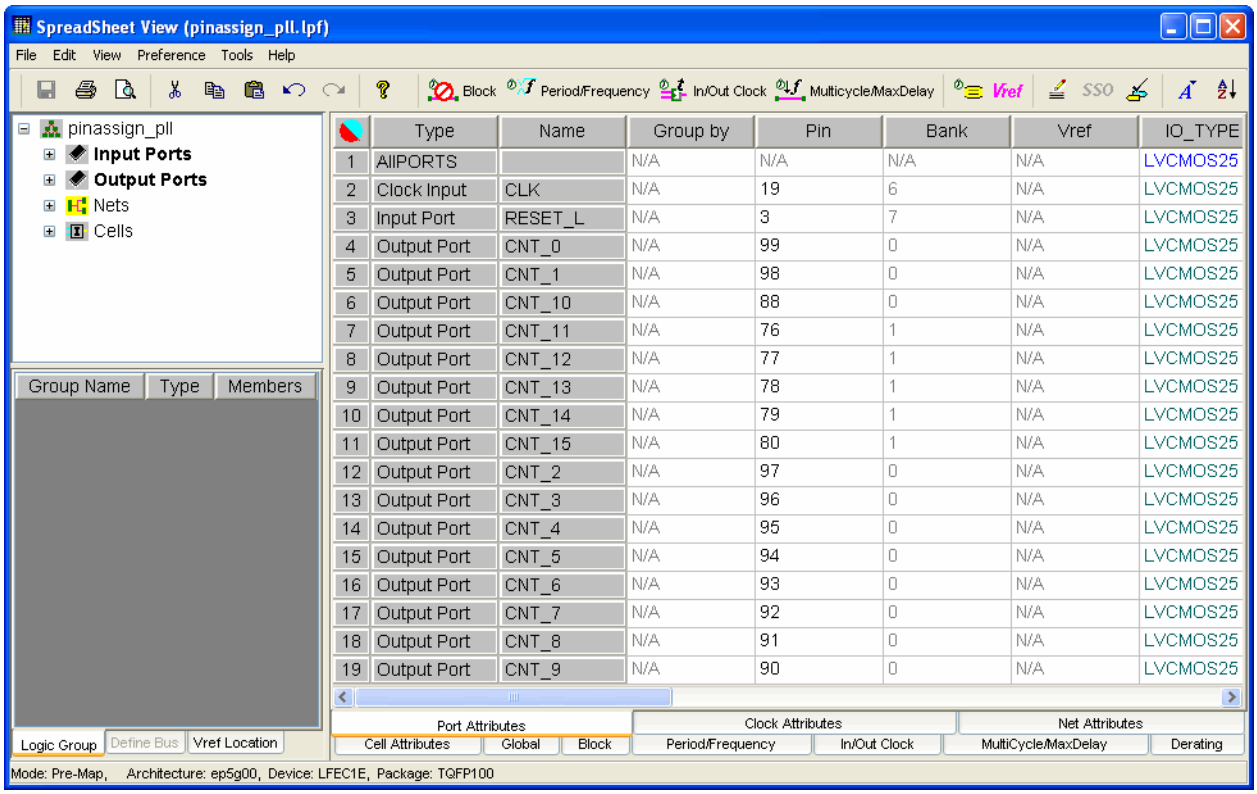
13. Click on Spreadsheet View and choose **Tools > PIOS DRC**.
14. Click **OK** in the Design Planner - Information pop-up message box that contains the following line:

    ```
    PIO DRC checks: No errors detected.
    ```

    The results of the PIO design rule check appear in the Design Planner Control window. They help to ensure that the assignments that you have made are legal before you submit the design to the mapping, placement, and routing tools later.
15. In Spreadsheet View, choose **File > Save** to update pinassign_pll.lpf.

    Figure 13 shows the pin assignments in the Spreadsheet View.

Figure 13: Pin Assignments in the Spreadsheet View



Assign Timing Preferences

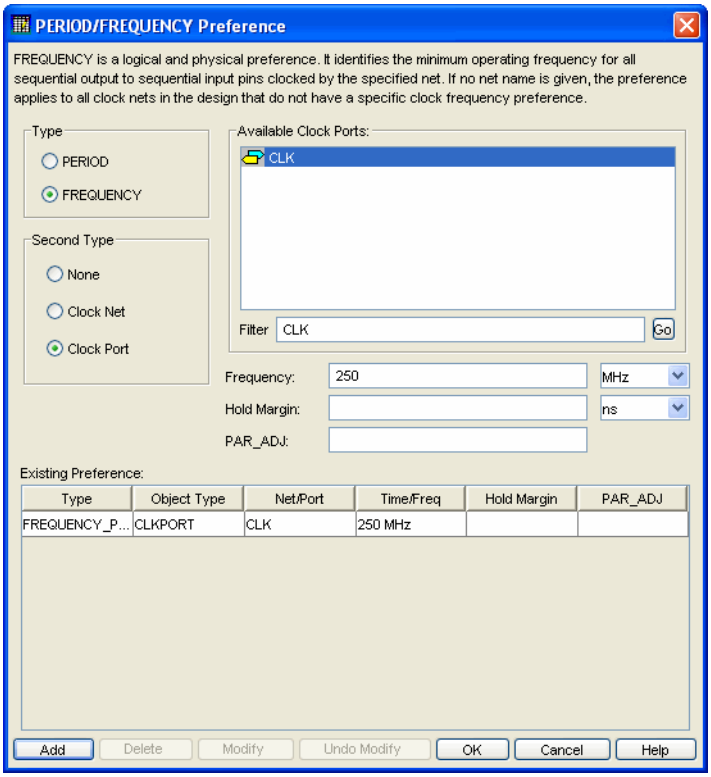
Timing constraints are typically set with specialized dialog boxes that are accessed from Design Planner. In the following steps, you will set the objectives for the operating frequency and the clock-to-out delay of the counter.

This task provides a preview of the many logical and physical preferences available for constraining FPGA mapping, placement, and routing. The FPGA Preference Language is explained in detail in the online help.

To assign timing preferences with the Design Planner:

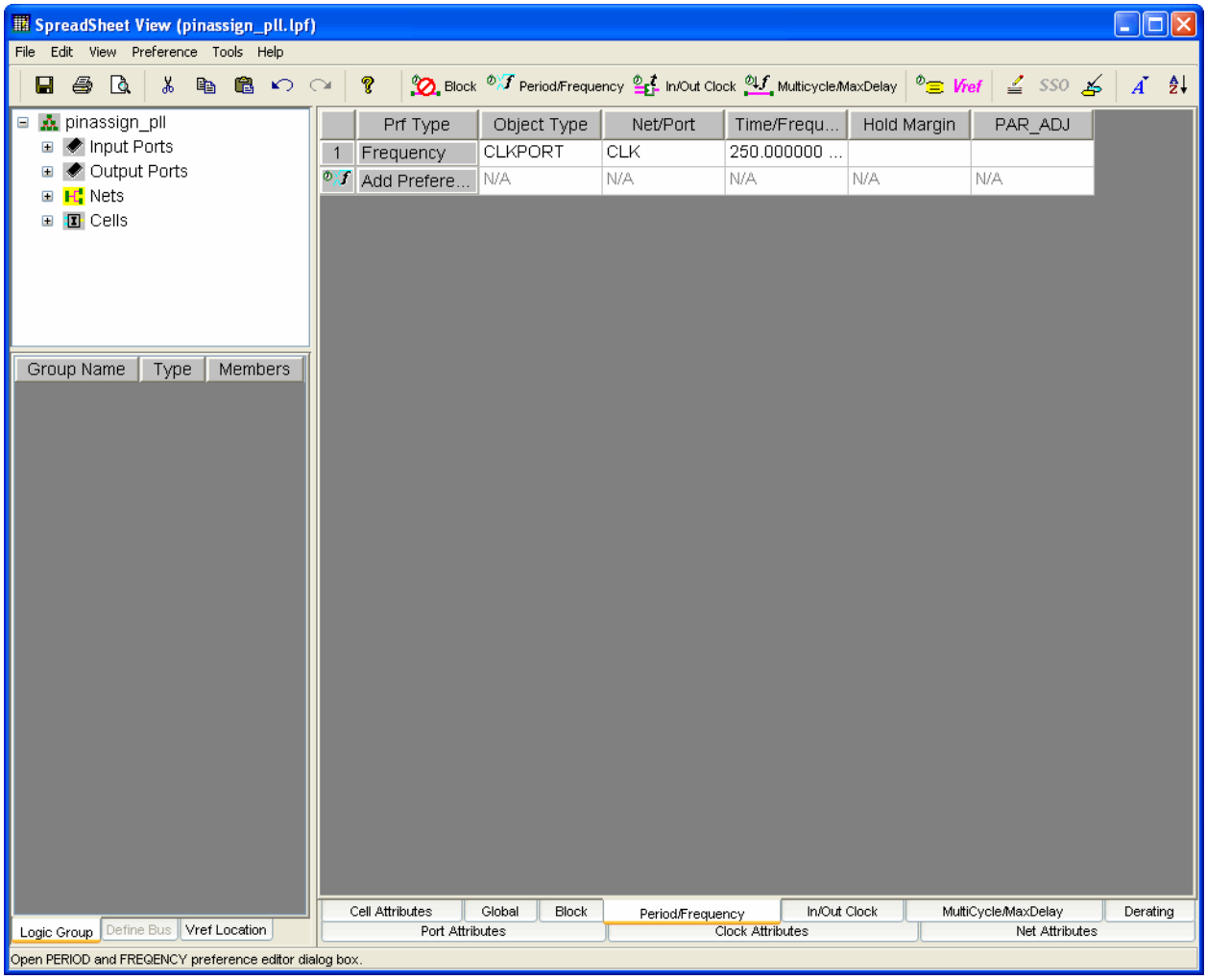
1. Maximize the Spreadsheet View by double-clicking on the title bar, and choose **Preference > Period/Frequency** to open the PERIOD/ FREQUENCY Preference dialog box, shown in Figure 14.
   a. In the Type field of the dialog box, select **FREQUENCY**.
   b. In the Second Type field, select **Clock Port**.
      A list of clock ports appears in the Available Clock Ports list.
   c. In the Frequency box, type **250**.
   d. Select **CLK** in the Available Clock Ports box.
   e. Click **Add**.
   f. Click **OK** to close the dialog box.

Figure 14: PERIOD/FREQUENCY Preference Dialog Box



The new preference appears in the Period/Frequency tab of the Spreadsheet View, as shown in Figure 15.

Figure 15: Frequency Preference Set in Spreadsheet View



The FREQUENCY preference identifies the minimum operating frequency for all sequential-output-to-sequential-input pins clocked by the specified net.

2. Choose **Preference > Input_setup/Clock_to_out** to open the INPUT_SETUP / CLOCK_TO_OUT Preference dialog box, shown in Figure 16.

   a. In the Type field of the dialog box, select **CLOCK_TO_OUT**.

      A list of signal names appears as the Clock Out Ports list.

   b. In the Second Type field, choose **All Ports**.

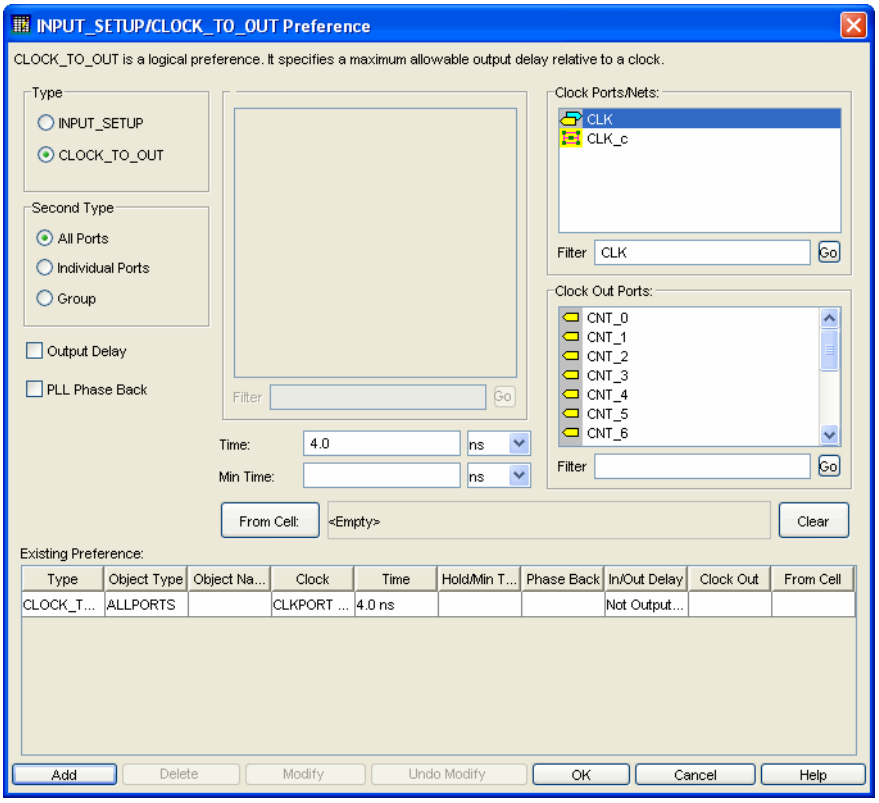
   c. In the Time box, enter **4.0** ns.

   d. In the Clock Ports/Nets box, select **CLK**.

   e. Click **Add**.

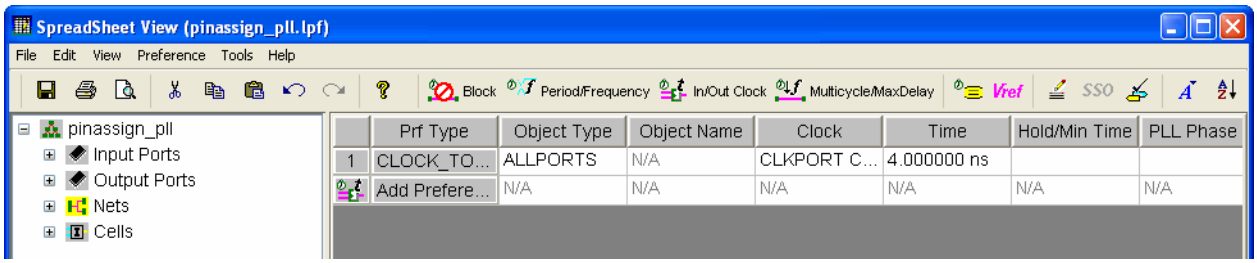
f. Click **OK** to close the dialog box.

Figure 16: INPUT_SETUP/CLOCK_TO_OUT Preference Dialog Box



The new preference appears in the In/Out Clock tab of the Spreadsheet View, as shown in Figure 17.

Figure 17: Clock-to-Out Preference Set in In/Out Clock Tab of Spreadsheet View



The CLOCK_TO_OUT preference specifies a maximum allowable output delay relative to a clock. For example, the delay from CLK to the CNT[15:0] counter output bus must be 4.0 ns or less.

3. Choose **Preference > Block Preference** to open the Block Preference dialog box, shown in Figure 18 on page 18.

   a. In the Type field of the dialog box, select **Block Nets**.

      A list of signal names appears in the Available Nets list.

b. In the Filter box, type **R***, and click **Go**.

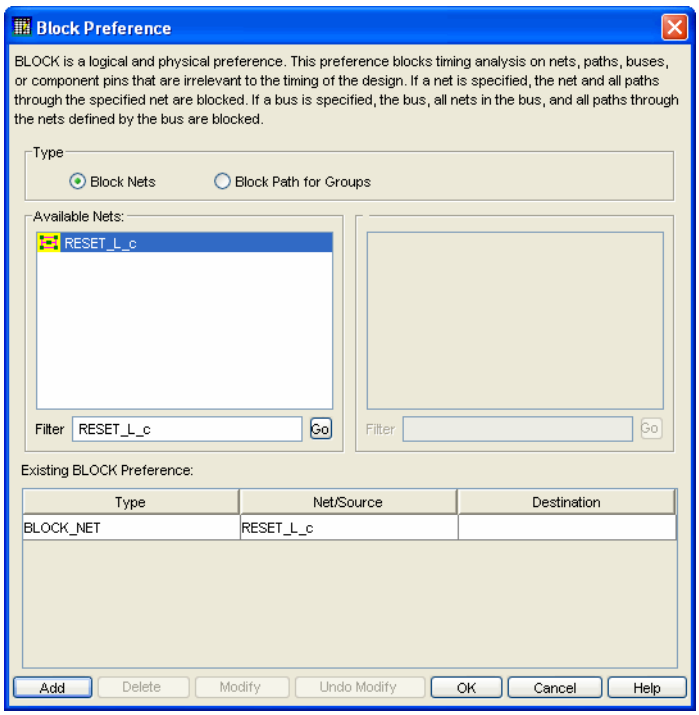
   Those nets that begin with “R” appear in the Available Nets list.

c. Select **RESET_L_c**.

d. Click **Add**.

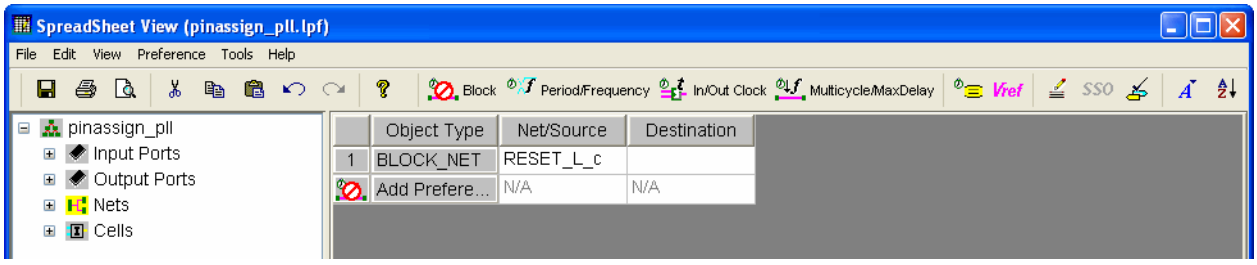
e. Click **OK** to close the dialog box.

Figure 18: Block Preference Dialog Box



The new preference appears in the Block tab of the Spreadsheet View, as shown in Figure 19.

Figure 19: Block Preference Set in Block Tab of Spreadsheet View



The BLOCK preference blocks timing analysis on nets, paths, buses, or component pins that are irrelevant to the timing of the design. In this example, RESET_L_c drives the global set/reset (GSR) line that serves

as the counter's asynchronous reset. You can safely ignore it during timing analysis.

Note

For more information on setting preferences, select the Setting Preferences topic in the online help.

Note

Notice the <*port_name*>_c net names created by the Synplify synthesis process for the clock tree and GSR nets. These internal signals are driven through buffers inferred by logic synthesis. In some cases, timing and location preferences refer to these internal names instead of the external port name associated with an I/O buffer pad. You may need to inspect the Map report or the output EDIF netlist of the Build Database process to understand what nets your synthesis tool produced.

4. Choose **File > Save** to update pinassign_pll.lpf.
5. Choose **File > Exit** in the Design Planner Control window to exit the Design Planner.

Task 3: Design Synthesis and Mapping

In this task, you will review the report created by the mapping process, which converts a logical design represented as a network of device-independent components (for example, gates and flip-flops) produced by logic synthesis and the Build Database process into a network of device-specific components (for example, PFU/PFF, IOLOGIC, or EBR) that will be eventually implemented by the placement and routing process.

After mapping is complete, you can perform static timing analysis to confirm that the current implementation, accounting only for logic element delays, will meet the timing constraints that you specified earlier.

View the Mapping Results

To view the mapping results:

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. Double-click **Map Report (pinassign_pll.mrp)**.
3. After a few moments, an ispLEVER Process warning message box might appear. If it does, click **OK** to close it.

   The pinassign_pll.mrp tab appears in the output panel.

   Note

   Click on a tab to view it. If you do not see a tab, drag the vertical splitter bar to the right.

4. Review the major sections of the report:

- Design Information: Command line, device, and software version
- Design Summary: Number of physical elements mapped
- IO (PIO) Attributes: Detail about the configuration of I/Os

Note

To view the results of logic synthesis and the translation process from EDIF 2 0 0 to the pre-map NGD database, see the Automake Log tab of the output panel.

View the Static Timing Analysis Report

To view the post-map static timing analysis report:

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. Double-click **Map TRACE Report**.

   After a few moments, the pinassign_pll.tw1 tab appears in the output panel.
3. Review the major sections of the report:
   - Report Information: Command line, device, and preference file
   - Preference header: Preference type, value, and score and error count
   - Logical Details: Detail of one or more signal paths in terms of the logical components used in the input design
   - Physical Path Details: Detail of one or more signal paths in terms of the physical (mapped) components
   - Report Summary: An index of preference values and actual results based on the analysis
   - Timing Summary: Number of errors, score, and coverage

The report indicates that the FREQUENCY preference passes, but the CLOCK_TO_OUT of 4.0 ns failed. You will attempt to improve these delays later in the tutorial.

Task 4: Place, Route, and Post-Route Timing

In this task, you will implement the design translated from the mapping phase in a completely placed and routed design. On the basis of the location and timing preferences that you assigned earlier, the place and route program (PAR) will attempt to satisfy the constraints that you declared in earlier tasks.

After placement and routing is complete, you can perform static timing analysis to confirm that the current implementation, accounting for both logic element and routing delays, will meet the timing constraints that you specified earlier.

Place and Route the Design

To place and route the design:

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. In the Processes for Current Source window, right-click the **Place & Route Design** process and choose **Properties** to open the Properties dialog box.

   Note

   You can place a design without routing it by using a command on the command line. In the online Help, see Design Flow User Guide > Running FPGA Tools from the Command Line > Command Line Tool Usage > Running PAR from the Command Line for details.

3. Select the **Placement Effort Level** row and press **F1** to bring up the online Help topic for that property.

   PAR properties like Placement Effort and Routing Passes can influence the run time, resource utilization, and operating performance of your design. These options, along with the location and timing constraints that you place on your design, are the major factors that influence results.

   Note

   For methods and advice to help meet your performance objectives, open the online Help and see Design Flow User Guide > Design implementation > Applying Design Constraints > Achieving Timing Closure.

4. Close the online Help.
5. In the Properties dialog box, shown in Figure 20, specify the following values by selecting the property and then entering the value in the box next to the **X** or using the pull-down menu next to the **X**:
   - Placement Effort Level: **3**
   - Routing Passes: **3**
6. Click **Close**.
7. Double-click **Place & Route Design**.
8. If an ispLEVER Process warning message box appears, click **OK** to close it.

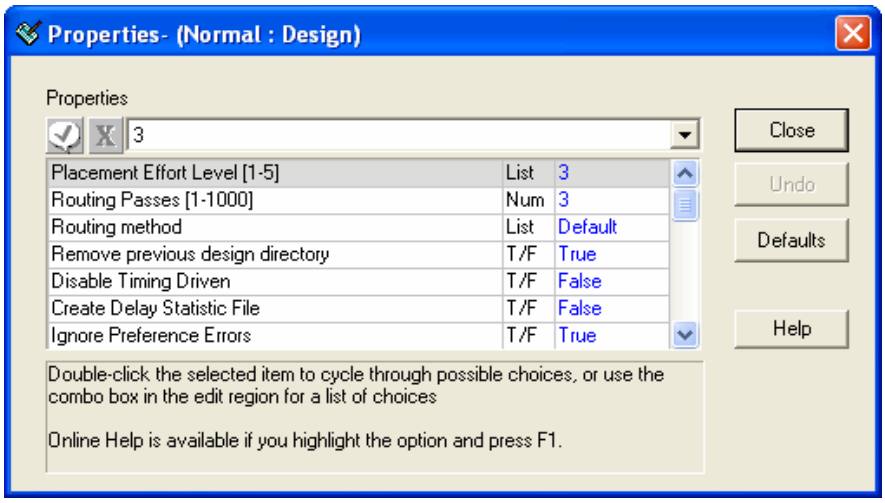
   The Automake Log tab of the output panel displays the run results.

9. Double-click **Report Summary - HTML**.

   A summary of several reports appears in your Web browser as pinassign_pll.html, as shown in Figure 21. The table of contents allows you to quickly traverse the major sections.

   - The Map Report includes the results of the design mapping. You examined the ASCII version of this report in “Task 3: Design Synthesis and Mapping” on page 19.

Figure 20: Properties Dialog Box



- The Place & Route Report provides the details of the implementation, including the number of routed and unrouted connections, number of device resources utilized, and which iteration of PAR produced the best results.
- The PAD Specification File gives details about the configuration of the device I/O.
- The Place & Route TRACE Report shows static timing analysis results based on the routed design and the timing constraints that you specified.

View the Static Timing Analysis Report

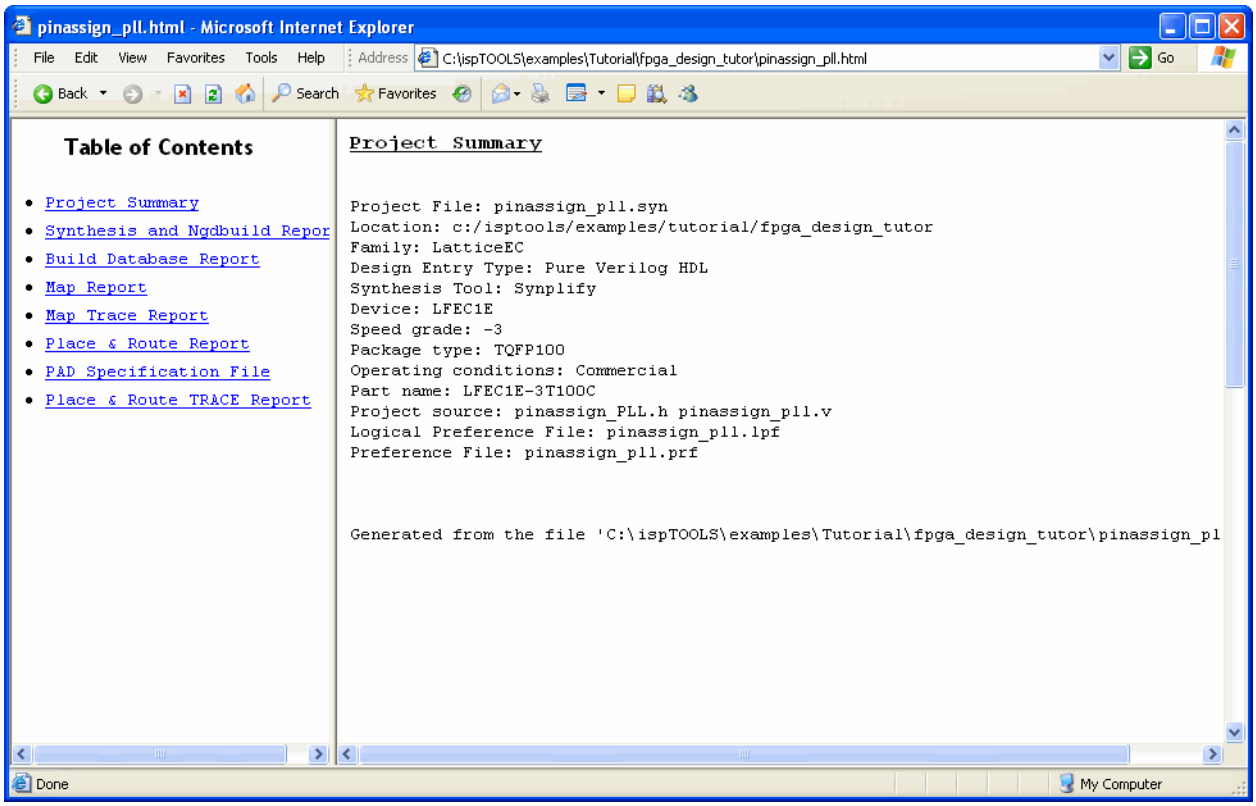
To view the post-place-and-route static timing analysis report:

1. In the Table of Contents panel of the HTML Place and Route Report, click **Place & Route TRACE Report**.

   The Lattice TRACE Report appears in the right panel and the Table of Contents expands.

2. Click **Setup** to expand the Table of Contents further.
3. Click **Report Summary** and review the operating frequency reported in the Actual column of the FREQUENCY PORT “CLK” 250.000000 MHz preference. See Figure 22.

Figure 21: HTML Place and Route Report



The report indicates that after placement and routing, the actual operating results are far below the objectives. Next, you will inspect the device implementation to understand why.

Note

The actual values of your report may vary.

Figure 22: Report Summary

Report Summary

Preference	Constraint	Actual	Levels
FREQUENCY PORT "CLK" 250.000000 MHz ;	250.000 MHz	170.242 MHz	6 *
CLOCK_TO_OUT ALLPORTS 4.000000 ns CLKPORT "CLK" ;	4.000 ns	8.377 ns	2 *

4. Close the HTML Place & Route Report.

Note

If you want to exit the tutorial at this point and resume it later, choose **File > Save** in Project Navigator to save the pinassign_pll.syn file. To resume the tutorial, re-load this file by choosing **File > Open Project**.

Task 5: Viewing the Device Implementation

In this task, you will view the placed and routed design in Design Planner, which provides a graphical view of the implementation. Design Planner, along with location and timing constraints, is used as part of a design floorplanning strategy to meet your performance objectives or accommodate design reuse. The tool is flexible enough to enable you to view pre-routed or partially routed designs.

Note

As you view the device implementation, it is helpful to refer to the Map Report section of the HTML Place & Route Report and the *LatticeECP/EC Family Data Sheet* for details on the physical elements of your design.

View the Device Implementation After Placement and Routing

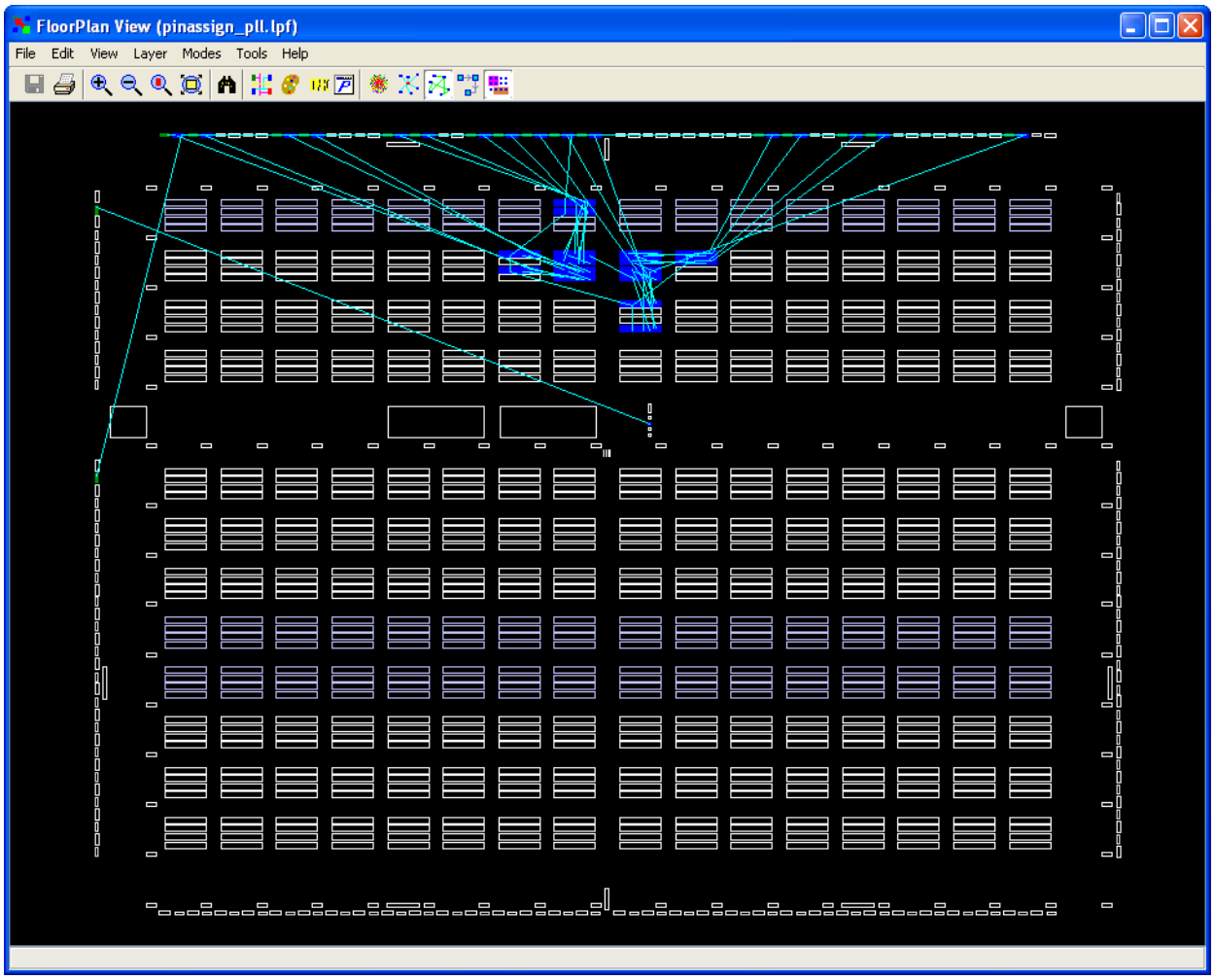
To view the device implementation after placement and routing:

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. Double-click **Design Planner (Post-PAR)**.
3. In the Design Planner Control, choose **View > Floorplan View**.
4. Maximize the view by double-clicking the Floorplan View title bar, and then choose **View > Zoom to Fit**.
5. Choose **Layer > Ratsnests**.

The LatticeEC PIC, PFU, PFF, and EBR elements are shown as small boxes in the display, similar to the simplified block diagram of the data sheet. Routed connections appear as cyan “flywires.” You will recognize the relative signal placement along the top edge of the chip die that was based on the pin assignments that you made in the Spreadsheet View earlier.

In the next procedure, you will examine the critical path of the counter design.

Figure 23: Ratsnest



Find the Critical Path

To trace signal paths related to timing constraints:

1. In Floorplan View, choose **Tools > Path Tracer**.

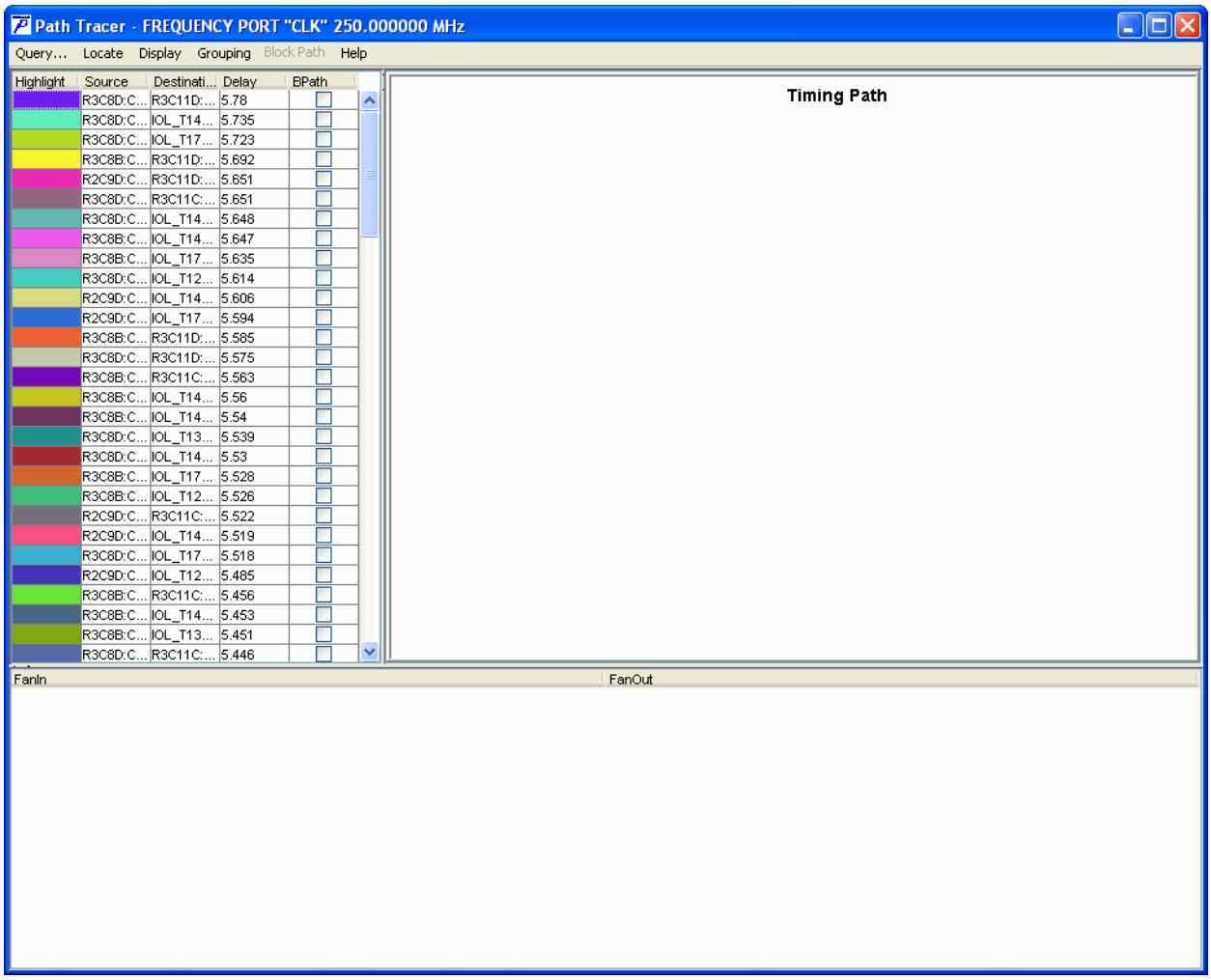
   During the next steps, it is helpful to tile the Path Tracer window next to the Floorplan View.

2. Click **Query** to open the Timing Query dialog box.

3. In the Preference list, select **FREQUENCY PORT "CLK" 250.000000 MHz;**, and click **Query**.

   All paths related to the CLK timing constraint are displayed in the query result pane, as shown in Figure 24 on page 26. Each row represents a path between the source pin and the destination pin reported in terms of a PFU/PFF location (R*n*C*n*, where *n* is a row or column number). The delay in nanoseconds is shown for each path in the Delay column. The highlighting enables you to control how connections and related slices appear in the Floorplan View.

Figure 24: Paths Related to CLK Constraint in Path Tracer Window



4. Click on the **Delay** column header to sort the delays from the longest to the shortest.
5. Locate the longest delay, select the row, and click **Display**.

   A schematic of the timing path appears with slice, I/O logic, and routing elements annotated with the component delays of the entire path, as shown in Figure 25 on page 27. The first and last components in the schematic represent the first and last registers of the critical path, and all components between represent combinatorial logic. This path is also reported in the static timing analysis results of the Place & Route TRACE Report that you examined earlier. See the pinassign_pll.twr tab in the output panel of Project Navigator to review it.
6. Click **Locate**.

   The display in the Floorplan View is updated to show the logical interconnect related to the path, as shown in Figure 26.

   The schematic helps explain why the 250-MHz preference is failing. Notice the first and last components of the path:

Figure 25: Schematic of Timing Path in Path Tracer Window

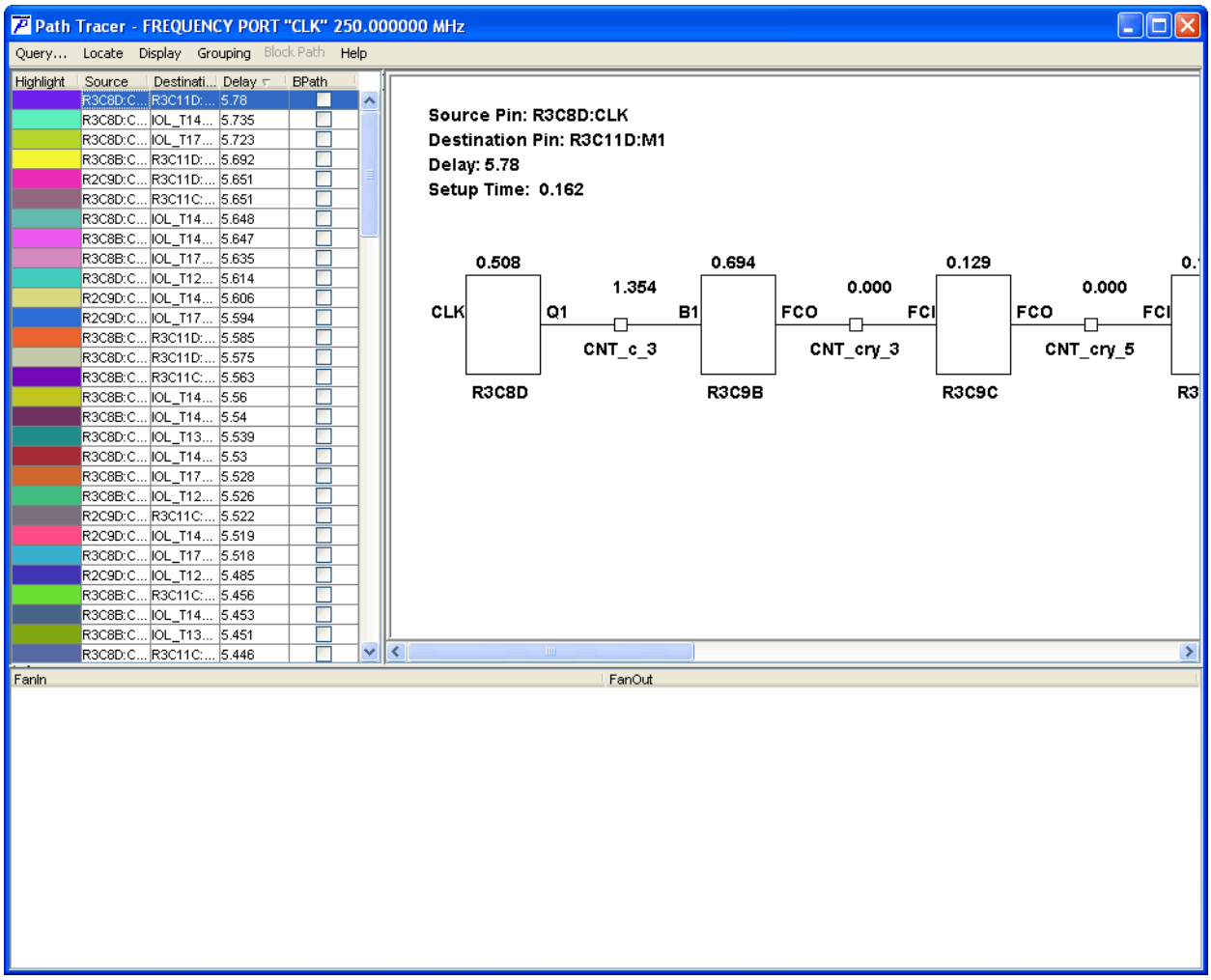
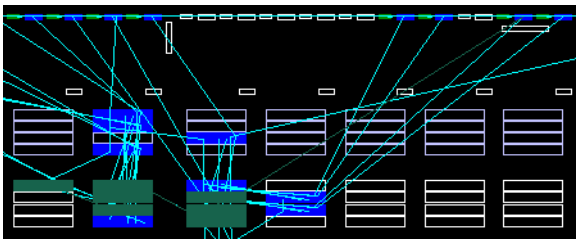


Figure 26: Logical Interconnect Related to Critical Path



- The first component is a slice programmed into a mode featuring logic and a flip-flop.
- The last component is a PIO programmed into a mode also featuring a registered output block.

The block diagram shown in Figure 27 clarifies the path trace results.

According to the data sheet, the clock-to-Q delay of PFU/PFF elements is about half that of the clock-to-output of PIO/IOLOGIC elements. In a later

Figure 27: Path Trace Results

task, you will adjust the design to exclusively use the faster PFU/PFF register elements for the counter.

Note

The slice locations of your floorplan and path tracer may vary from the figures shown.

7. Close Path Tracer, and minimize Floorplan View.

Examine Programming of Design Planner Elements

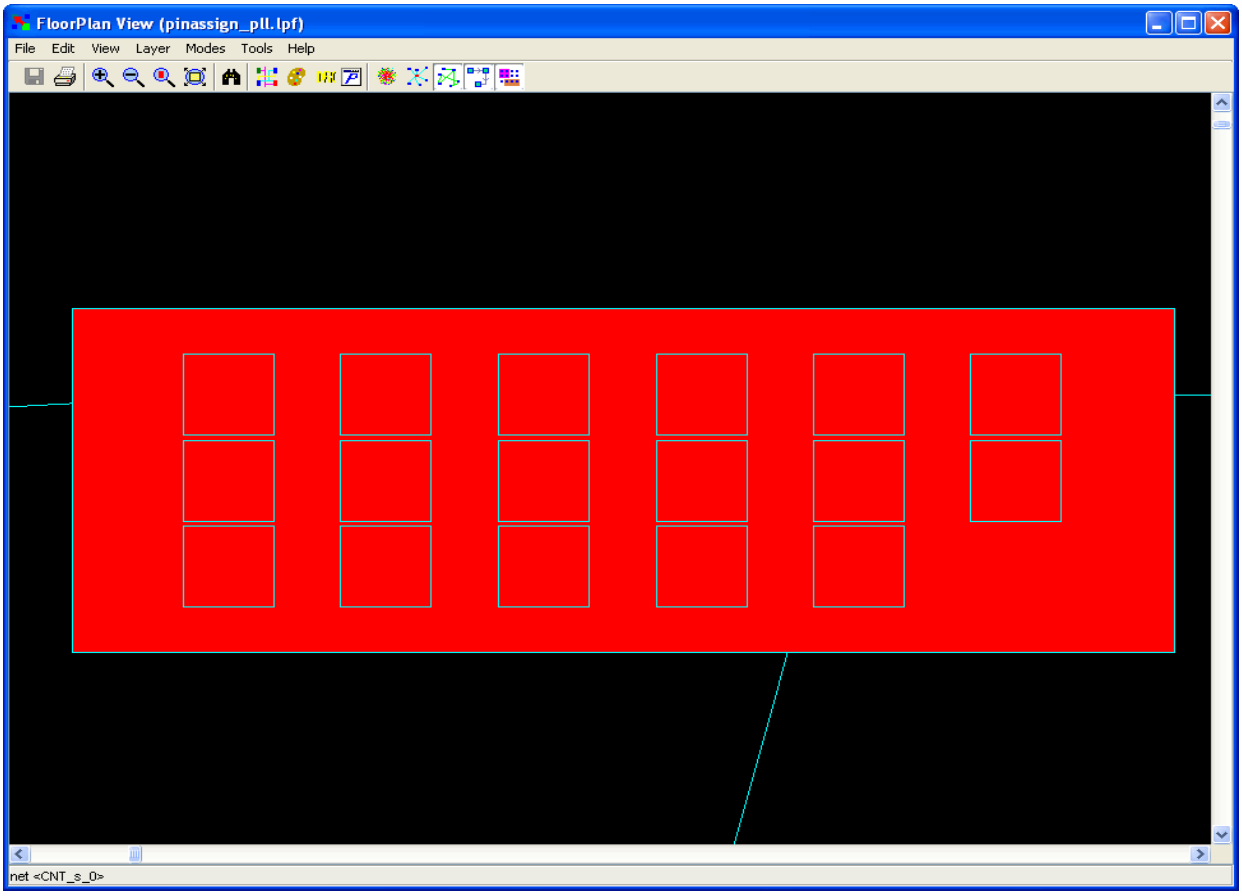
To examine the programming of Design Planner elements:

1. In Design Planner Control, choose **View > Post-Mapped View**.
2. In the right pane of Post-Mapped View, expand the Instance list and the net list.

   From Post-Mapped View, you can see how specific registers of the counter were implemented by the placement and routing. Notice the physical elements related to CNT[0] of the counter. Look for CNT_0(PIO)@ and CNT_0_MGIOL(IOLOGIC)@. A PIO is equivalent to a sysIO buffer of the LatticeEC Programmable I/O Cell (PIC), and an IOLOGIC represents the register element of the PIC. This is another hint to indicate why the frequency preference failed. PIC registers do not provide fast carry chain routing resources, so additional logic and routing is required to emulate the counter behavior.
3. Go to Floorplan View and choose **Tools > Customization**. In the Customization dialog box, select **Activates window when component is located and** click **OK**.
4. In Post-Mapped View, select **CNT_0_MGIOL(IOLOGIC)@IOL_*scl***, where:
   - ***s*** = [T|B|L|R] indicates the side of the chip die
   - ***c*** is the column number
   - ***l*** = [A|B] indicates a specific pad of an LVDS pair
5. Click the right mouse button, and choose **Locate > Floorplan View**.

The Floorplan View highlights the IOLOGIC block and changes the zoom level, as shown in Figure 28.

Figure 28: Highlighted IOLOGIC Block



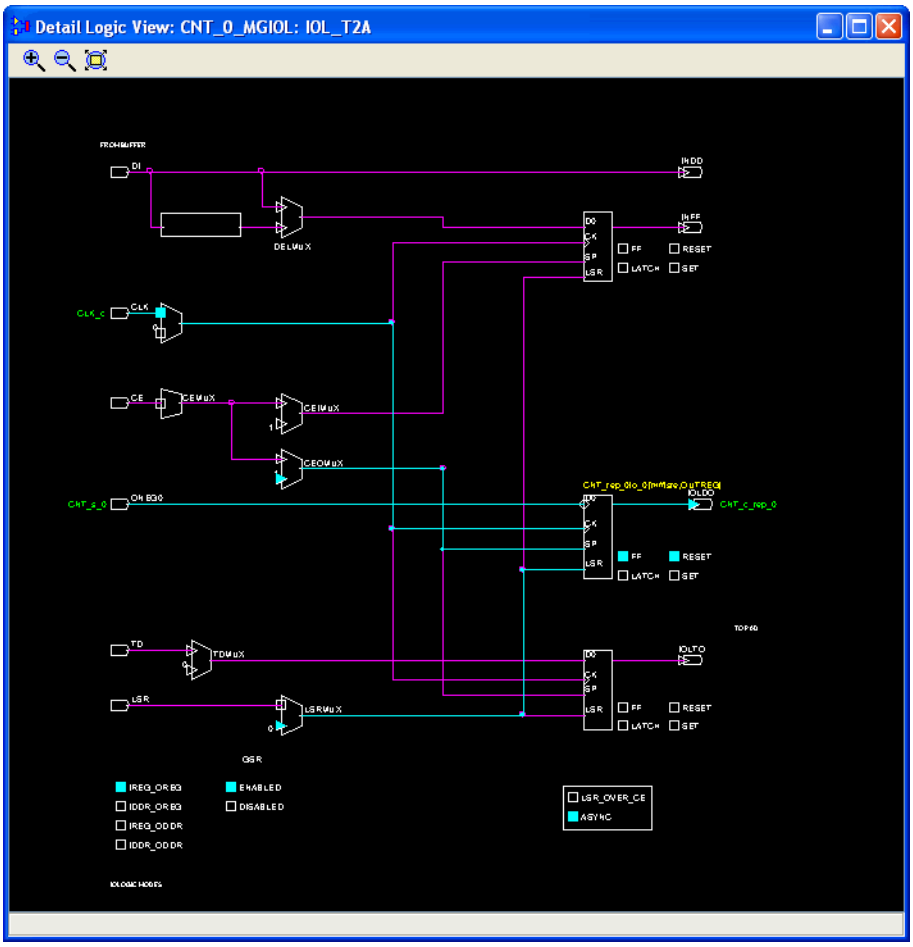
6. Right-click on the highlighted block, and choose **Detailed View**.

   A detailed schematic view of the IOLOGIC block configuration appears, as shown in Figure 29 on page 30.

   Details like the logic mode, GSR setting, reset, and configuration of the register elements appear as blue highlighted boxes, and the data path through the block is annotated in green.

7. Close the Detail Logic View.

Figure 29: Detailed Schematic View of IOLOGIC Block



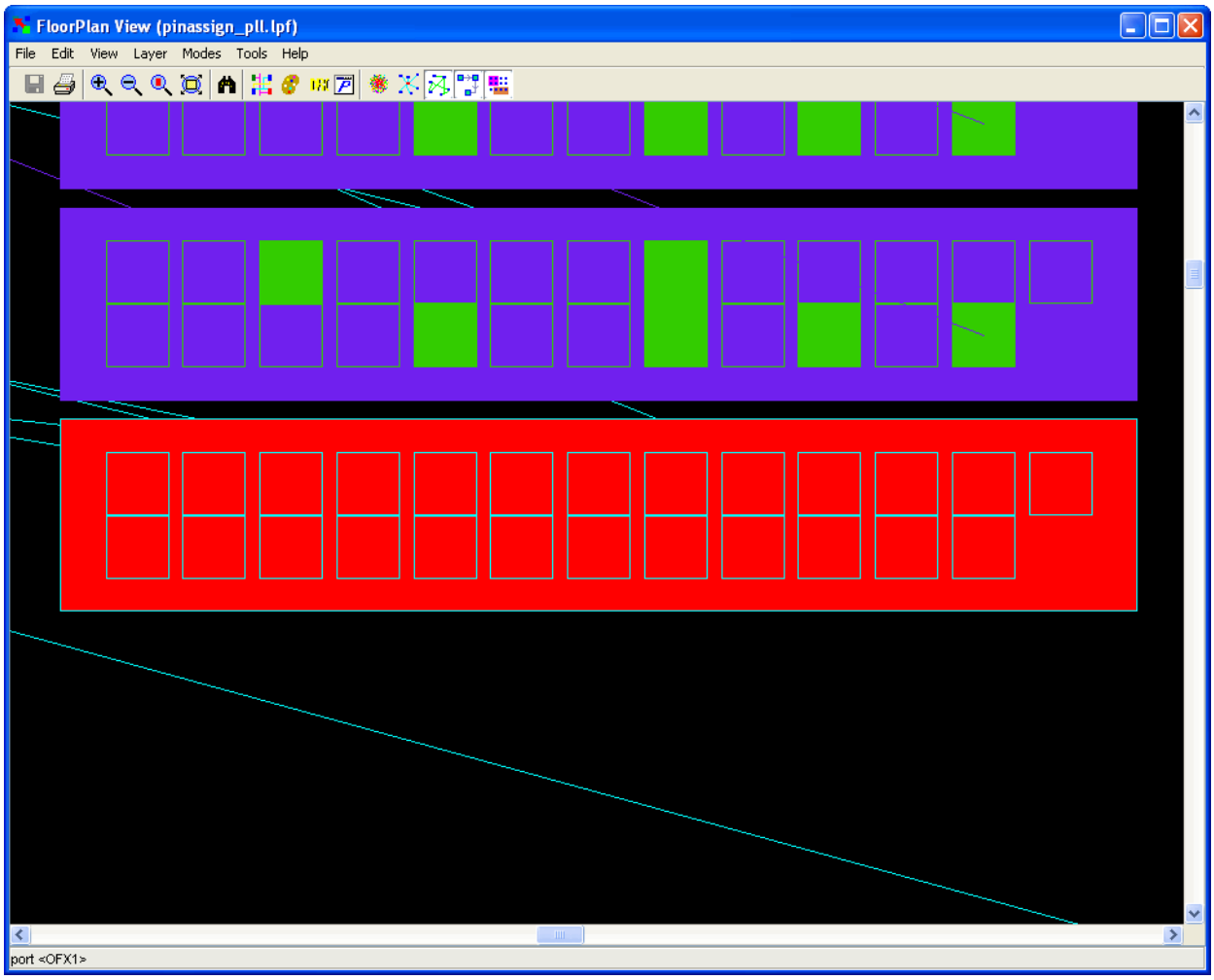
Examine the Counter Implementation in Slices

To examine the counter implementation in slices:

1. In Post-Mapped View, select **SLICE_0(FSLICE)@*rcs***, where:
   - ***r*** is the row number.
   - ***c*** is the column number.
   - ***s*** = [A|B|C|D] to indicate a specific slice of a PFU.
2. Right-click and choose **Locate > Floorplan View**.

   Floorplan View highlights the SLICE block and changes the zoom level, as shown in Figure 30.

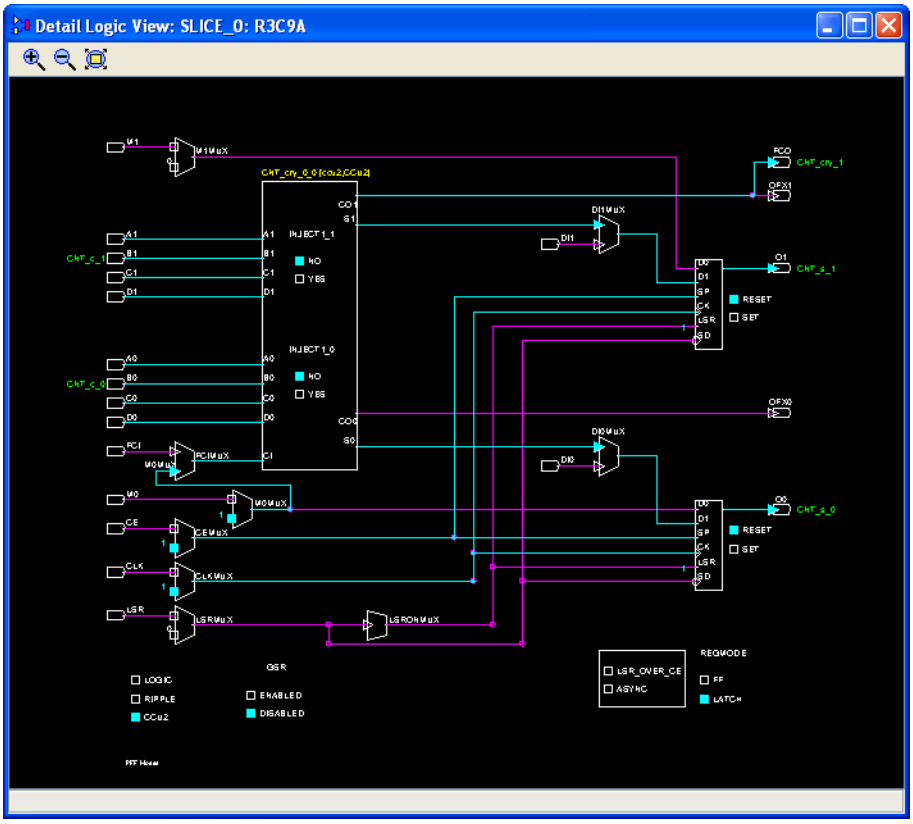
Figure 30: Highlighted SLICE Block



3. Right-click the highlighted block, and choose **Detailed View**.

A detailed schematic view of the SLICE diagram appears, as shown in Figure 31.

Figure 31: Detailed Schematic View of SLICE Block



The SLICE_0 LUTs are configured as up counters (CNTUP), with ripple mode (RIPPLE), and fast carry out (FCO). SLICE_0 through SLICE_7 are configured in the same manner.

4. Close the Detail Logic View.
5. In Post-Mapped View, select **SLICE_8(FSLICE)@*rcs***.
6. Right-click and choose **Locate > Floorplan View**.

   Floorplan View highlights the SLICE diagram and changes the zoom level.
7. Right-mouse click on the highlighted block and choose **Detailed View**.

   A detailed schematic view of the SLICE block configuration appears.

   SLICE_8 LUTs are configured as logic and flip-flops (REGMODE=FF). SLICE_8 through SLICE_15 are configured in the same manner.
8. Close the Detail Logic View.

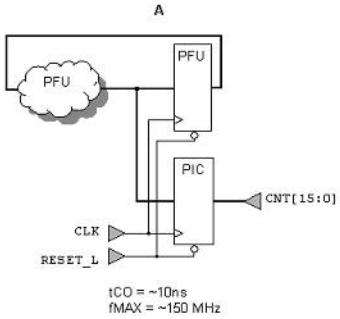
9. From the Design Planner Control window, choose **File > Exit**.

Note

You can often gather what logic is associated with what slice from the slice details of the Post-Mapped View.

Figure 32 illustrates the implementation as a block diagram of the logical implementation and the approximate timing results.

Figure 32: Logic Implemented as Block Diagram



In the next task, you will direct the system to implement the counter registers in PFU/PFF-based elements to take advantage of fast carry chain resources.

Task 6: Improving PAR Results – Part 1

This task demonstrates how synthesis can dramatically influence the operating frequency of the design. In the previous task, you discovered that the counter speed was constrained because of the use of PIC registers instead of PFU/PFF slices. In this task, you will adjust a compiler directive to target the counter logic to PFU/PFF slices exclusively.

Timing closure is a large subject and is covered in more detail by the Lattice Semiconductor technical documentation cited at the end of the tutorial.

If you are using a Verilog HDL source file, see “Improving PAR Results for Verilog” on page 34. If you are using a VHDL source file, see “Improving PAR Results for VHDL” on page 34.

Note

For more information on RTL code style for best results, see the “Strategies for Timing Closure” chapter in the *FPGA Design Guide*.

Improving PAR Results for Verilog

To edit the Verilog source and add a directive to control placement:

1. In Project Navigator, double-click the **pinassign_PLL (pinassign_pll.v)** module to open the Verilog file in Text Editor.

   Note

   There is a completed source file in the following project subdirectory:

   <*install_path*>\examples\tutorial\fpga_design_tutor**pinassign_PLL_b.v**

2. Modify the output declaration at line 5 from:

   ```
   output [15:0] CNT /* synthesis syn_useioff=1 */;
   ```

   to:

   ```
   output [15:0] CNT /* synthesis syn_useioff=0 */;
   ```

   Note

   Verilog is case-sensitive, so you must enter the code exactly as just given.

3. In the Text Editor, choose **File > Save**.
4. In the Text Editor, choose **File > Exit**.

For more information on Synplify Synthesis Directives, see the *Synplify for Lattice Reference Manual* in the online Help.

Continue with the tutorial by going to “Viewing the TRACE Reports” on page 35.

Improving PAR Results for VHDL

To edit the VHDL source and add a directive to control placement:

1. In the Project Manager, double-click the **pinassign_PLL (pinassign_pll.vhd)** module to open the VHDL file in the Text Editor.

   Note

   There is a completed source file in the following project subdirectory:

   <*install_path*>\examples\tutorial\fpga_design_tutor**pinassign_PLL_b.vhd**

2. Modify the output declaration at line 12 from:

   ```
   attribute syn_useioff of CNT : signal is true;
   ```

   to:

   ```
   attribute syn_useioff of CNT : signal is false;
   ```

3. Modify the output declaration at line 21 from:

   ```
   attribute OUTFF of CNT : signal is true;
   ```

to:

```
attribute OUTFF of CNT : signal is false;
```

4. Choose **File > Save**.
5. Choose **File > Exit**.

For more information on Synplify Synthesis Directives, see the *Synplify for Lattice Reference Manual* in the online Help.

Viewing the TRACE Reports

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. Double-click **Place & Route TRACE Report**.

   After a few moments, the pinassign_pll.twr tab appears in the output panel.

   Note

   If you do not see this tab, drag the vertical splitter bar to the right.

3. In the Report Summary of the pinassign_pll.twr TRACE report, review the operating frequency reported in the Actual column of the FREQUENCY PORT “CLK” 250 MHz preference, shown in Figure 33.

The 250-MHz objective for operating frequency has been met, but the clock-to-output delay (t_{CO}) has worsened now that the clock-to-output path must traverse a PFU and a PIC to the external pad.

Figure 33: Report Summary of pinassign_pll.twr TRACE Report

```
Report Summary
--------------
------------------------------------------------------------------------------
Preference                                |   Constraint|       Actual|Levels
------------------------------------------------------------------------------
                                          |             |             |
FREQUENCY PORT "CLK" 250.000000 MHz ;     |  250.000 MHz|  295.421 MHz|   8
                                          |             |             |
CLOCK_TO_OUT ALLPORTS 4.000000 ns         |             |             |
CLKPORT "CLK" ;                           |     4.000 ns|     9.761 ns|   2 *
                                          |             |             |
------------------------------------------------------------------------------
```

The TRACE reports shown in Figure 34 on page 36 illustrate the worst-case path as scored by the static timing analyzer before and after the design

changes of this task. Notice the Delay summary of each preference and the utilization ratio of logic versus route resources.

Note

The actual values of your report may vary. To see how the device implementation has changed, use the procedure in “Task 5: Viewing the Device Implementation” on page 24 to examine how the counter registers are now implemented in PFU/PFF SLICE elements instead of IOLOGIC.

Figure 34: TRACE Reports Showing Worst-Case Path

Before

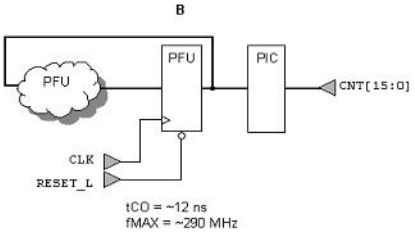
```
================================================================================
Preference: FREQUENCY NET "CLK " 250.000000 MHz ;
            272 items scored, 262 timing errors detected.
--------------------------------------------------------------------------------

Error:  The following path exceeds requirements by 2.144ns

 Logical Details:  Cell type  Pin type   Cell/ASIC name  (clock net +/-)

   Source:         FF         Q          CNT7_6_F2 (from CLK_c +)
   Destination:    FF         Unknown    CNT15_14_F2_repio (to CLK_c +)

   Delay:               6.248ns  (44.8% logic, 55.2% route), 6 logic levels.
```

After

```
================================================================================
Preference: FREQUENCY PORT "CLK" 250.000000 MHz ;
            72 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------

Passed:  The following path meets requirements by 0.615ns

 Logical Details:  Cell type  Pin type       Cell/ASIC name  (clock net +/-)

   Source:         FF         Q              CNT_0 (from CLK_c +)
   Destination:    FF         Unknown        CNT_15 (to CLK_c +)
                   FF                        CNT_14

   Delay:               2.776ns  (75.0% logic, 25.0% route), 8 logic levels.
```

Figure 35 illustrates the implementation as a block diagram of the logical implementation and the approximate timing results.

Figure 35: Logic Implemented as Block Diagram



In the next task, you will adjust the source design to use a form of pipelining to reduce clock-to-output delay (t_{CO}) and add a sysCLOCK PLL to effectively remove from the critical path timing equation the route delay introduced by the clock tree.

Task 7: Generate a Module Using IPexpress

In this task, you will configure and generate the source files for a sysCLOCK PLL module using IPexpress, then you will reference the new module from the top-level design. Using IPexpress, you can select a module from the module tree and specify parameters within the dialog box. When you click Generate, the software uses the parameters that you specified and produces the required output files. After you have declared the module and copied the boundary description to your HDL source file, the software automatically includes it in your design. In this module, you will create a sysCLOCK phase-locked loop (PLL), which is a device that provides the ability to multiply, divide, or phase shift clocks.

The purpose of the tutorial PLL is to effectively remove from the critical path timing equation the route delay introduced by the clock tree by feeding the global clock signal back into the PLL. The LatticeEC clock tree provides a dedicated routing path for this sort of application, which ensures that the delay related to the feedback compensation result is the same from design to design.

Generate a sysCLOCK PLL Module

To generate a sysCLOCK PLL module:

1. From Project Navigator, choose **Tools > IPexpress**, or click the button on the Project Navigator toolbar.

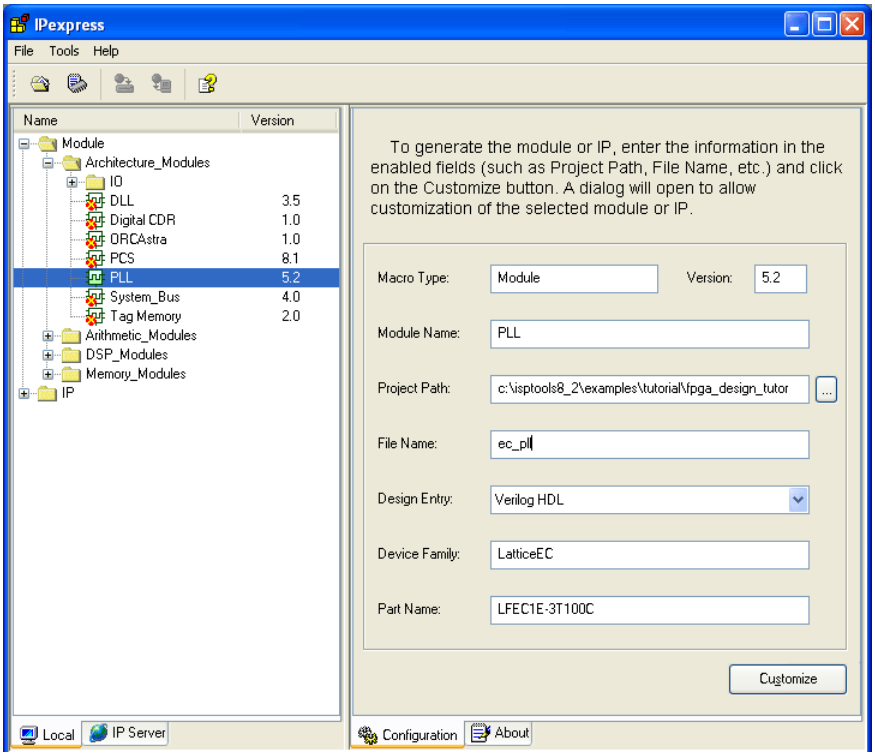
   The IPexpress window appears.

2. In the left pane, expand the **Module** folder.
3. Expand the **Architecture_Modules** folder and select **PLL**.

4. In the right pane, type **ec_pll** in the File Name box.

   The IPexpress window should now resemble that shown in Figure 36. For VHDL designs, the Design Entry field will display “VHDL.”

Figure 36: IPexpress Window



5. Click **Customize**.
6. In the Lattice FPGA Module -- PLL dialog box, click the **Configuration** tab.

7. Select or type the following parameters in the Configuration tab to configure the PLL to produce a 250-MHz clock:

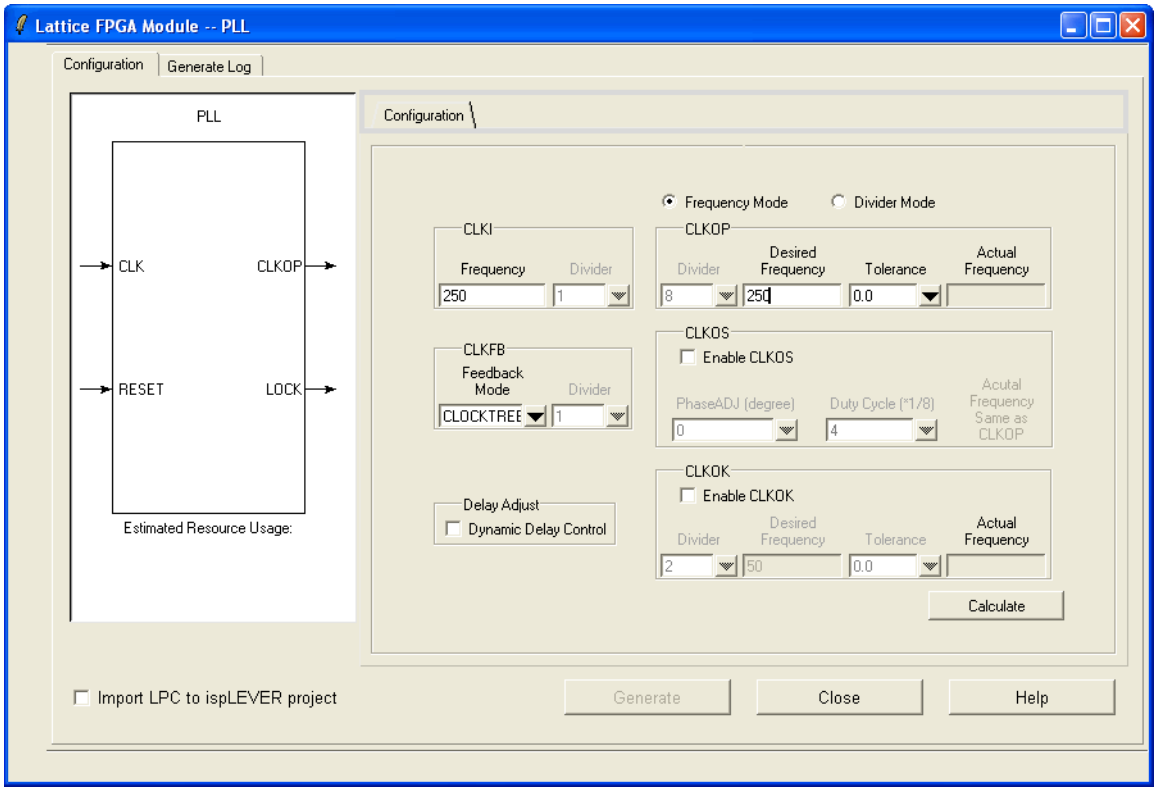
   a. Select **Frequency Mode**.

   b. In the CLKI Frequency box, type **250**.

   c. In the CLKOP Desired Frequency box, type **250**.

   The Configuration tab should look like the illustration shown in Figure 37.

Figure 37: IPexpress Configuration Tab



8. Click **Calculate**.

9. Click **Generate**.

Note

For more information on how to determine appropriate divider values to meet your frequency requirements, see Technical Note TN1049, “LatticeECP/EC sysCLOCK PLL Design and Usage Guide” at www.latticesemi.com.

A message appears in the Generate Log tab, confirming that the module has been generated successfully. IPexpress creates the following files in the project folder:

- ec_pll.lpc – Parameter file, where the file extension corresponds to the type of module generated. This file is loaded into IPexpress so that modifications can be made to the module.
- ec_pll.srp – Report file

- ec_pll.v or .vhd – HDL file for both synthesis and simulation
- ec_pll_generate.log – IPexpress log file
- ec_pll_tmpl.v or .vhd – Component or instantiation template for HDL netlist
- msg_file.log – Log message file

10. Click **Close** to close the Generate Log tab.
11. Choose **File > Exit** to close IPexpress.

Add a PLL Instance

In this procedure, you refer to the PLL module from your source, using Text Editor. When IPexpress builds the Verilog HDL or VHDL module, it produces output files for generating your design's database and places them inside the project directory. Before you generate the database for your Verilog HDL or VHDL design, you must add one or more module instances to your source code and connect the interface to your top-level design signals.

The PLL module instance is wired into the design so that it uses the external CLK signal to produce a new clock, PPCLK, to drive the counter design. Feedback is taken from PPCLK, which effectively removes the clock-tree delay from the critical-path timing equation, given the natural matching behavior of the PLL.

An alternative to importing HDL source files generated by IPexpress into the project is to import the module's .lpc file. After the file is imported, you can double-click the .lpc file to cause IPexpress to automatically open and load the module's configuration.

To edit the HDL source to "pipeline" the counter results with I/O registers and add a PLL instance:

1. In Project Navigator, double-click the **pinassign_PLL (pinassign_pll.v)** module to open the HDL file in Text Editor.
2. Modify the source code as shown in Figure 38 (Verilog) or Figure 39 (VHDL).

Note

There are completed source files at:

<*install_path*>\examples\tutorial\fpga_design_tutor**pinassign_pll_c.v**

<*install_path*>\examples\tutorial\fpga_design_tutor**pinassign_pll_c.vhd**

Figure 38: Verilog Source Code

```
module pinassign_PLL (RESET_L, CLK, CNT);

  input RESET_L;
  input CLK;
  output [15:0] CNT; /* synthesis syn_useioff=1 */

  reg [15:0] q_i;
  reg [15:0] CNT;

  wire RESET;
  wire PPCLK;

  assign RESET = !RESET_L;

  //Insert PLL instantiation here
  ec_pll ec_pll_inst(
    .CLK(CLK),
    .RESET(RESET),
    .CLKOP(PPCLK),
    .LOCK());

  always @(posedge PPCLK or posedge RESET)
  begin
    if (RESET)
      q_i <= 16'b0;
    else
      q_i <= q_i + 1;
  end

  always @(posedge PPCLK or posedge RESET)
  begin
    if (RESET)
      CNT <= 16'b0;
    else
      CNT <= q_i;
  end

endmodule
```

Figure 39: VHDL Source Code

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

entity pinassign_PLL is
  port(
    CNT : out std_logic_vector(15 downto 0);
    CLK : in std_logic;
    RESET_L : in std_logic
  );
  attribute syn_useioff : boolean;
  attribute syn_useioff of CNT : signal is true;

end pinassign_PLL;
```

Figure 39: VHDL Source Code (Continued)

```
architecture rtl of pinassign_PLL is
  signal RESET: std_logic;
  signal PPCLK: std_logic;
   signal q_i: unsigned(15 downto 0);

   attribute OUTFF : boolean;
   attribute OUTFF of CNT : signal is true;

   -- parameterized module component declaration
   component ec_pll
      port (
         CLK: in std_logic;
         RESET: in std_logic;
         CLKOP: out std_logic;
         LOCK: out std_logic);
   end component;

begin

  RESET <= not(RESET_L);

  --Insert PLL instantiation here
  ec_pll_inst: ec_pll
    port map(
      CLK=>CLK,
      RESET=>RESET,
      CLKOP=>PPCLK,
      LOCK=>open);

  process (PPCLK, RESET)
  begin
    if RESET = '1' THEN
      q_i <= (others=>'0');
    elsif (PPCLK'event and PPCLK='1') then
      q_i <= q_i + 1;
    end if;
  end process;

process (PPCLK, RESET, q_i)
  begin
    if RESET = '1' THEN
      CNT <= (others=>'0');
    elsif (PPCLK'event and PPCLK='1') then
      CNT <= std_logic_vector(q_i);
    end if;
  end process;

END rtl;
```

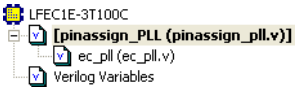
3. Choose **File > Save**.
4. Choose **File > Exit**.
5. In Project Navigator, choose **Source > Import** to open the Import File dialog box.
6. In the dialog box, do the following:
   a. Select **ec_pll.v** (or **ec_pll.vhd**), which is added to the File Name box.
   b. Click **Open**.

   The Import Source Type dialog box appears.
7. In the Import Source Type dialog box, select **Verilog Module** (or **VHDL Module**) and click **OK**.

   The file is added to the source file list.

   The new module appears in the Sources in Project window as **ec_pll (ec_pll.v)** (or as **ec_pll (ec_pll.vhd)**), as shown in Figure 40. The project is now complete.

Figure 40: New PLL Module in Project Navigator



Examine Timing Results with sysCLOCK PLL

In this procedure, you examine the timing changes resulting from the addition of the sysCLOCK PLL module.

To examine the timing results:

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. Double-click **Place & Route TRACE Report**.

   After a few moments, the updated pinassign_pll.twr appears in the output panel.
3. Review the operating frequency reported in the Actual column of the FREQUENCY NET “PPCLK” 250 MHz preference of the pinassign_pll.twr TRACE report, shown in Figure 41.

   The 250-MHz objective for operating frequency has been met, and the clock-to-output delay (t_{CO}) has improved but still does not meet the objective of 4.0 ns.

Note

The actual values of your report may vary.

Figure 41: Report Summary of pinassign_pll.twr TRACE Report

```
Report Summary
--------------
-------------------------------------------------------------------------------
Preference                                |   Constraint|       Actual|Levels
-------------------------------------------------------------------------------
                                          |             |             |
FREQUENCY NET "PPCLK" 250.000000 MHz ;    |  250.000 MHz|  295.421 MHz|   8
                                          |             |             |
FREQUENCY NET "CLK_c" 250.000000 MHz ;    |            -|            -|   0
                                          |             |             |
FREQUENCY PORT "CLK" 250.000000 MHz ;     |            -|            -|   0
                                          |             |             |
CLOCK_TO_OUT ALLPORTS 4.000000 ns         |             |             |
CLKPORT "CLK" ;                           |     4.000 ns|     5.028 ns|   2 *
                                          |             |             |
-------------------------------------------------------------------------------
```

In the next task, you will modify the design again to compensate for the clock-to-output delay (t_{CO}) delay.

Task 8: Improving PAR Results – Part 2

This task demonstrates how to use the ispLEVER software controls and constraints that may improve the maximum operating frequency of your design. Timing closure is a broad subject and is covered in more detail by the Lattice Semiconductor technical documentation resources cited in “Recommended Reference Materials” on page 64. In this task, you will attempt to improve placement and routing (PAR) performance results by using the delay compensation feature of the sysCLOCK PLL and effort level and number of iterations taken by the PAR program.

Review the Constraint Details section of the Place and Route TRACE Report in the pinassign_pll.twr tab of the output panel. It is shown in Figure 42. It provides information on how to improve the clock-to-output path delay. The clock path delay can be influenced by the delay adjustment feature of the PLL to offset the 0.250 ns by which the path exceeds the constraint. The sysCLOCK PLL delay adjustment feature enables you to adjust the phase of the clock in increments of 250 ps. Therefore, your adjustment will be –5 to account for the 1.028 ns by which the path exceeds the requirement (–5 x 0.250 ns = –1.250 ns).

Figure 42: Preferences Failing Timing Constraints in Place and Route TRACE Report

```
================================================================================
Preference: CLOCK_TO_OUT ALLPORTS 4.000000 ns CLKPORT "CLK" ;
            16 items scored, 16 timing errors detected.
--------------------------------------------------------------------------------


Error:  The following path exceeds requirements by 1.028ns

 Logical Details:  Cell type  Pin type       Cell/ASIC name  (clock net +/-)

   Source:         FF         Q             CNT_0__Qio  (from PPCLK +)
   Destination:    Port       Pad           CNT_0

   Data Path Delay:     4.250ns  (100.0% logic, 0.0% route), 2 logic levels.

   Clock Path Delay:    3.289ns  (23.7% logic, 76.3% route), 2 logic levels.

 Constraint Details:

      3.289ns delay CLK to CNT_0_MGIOL less
      2.511ns feedback compensation
      4.250ns delay CNT_0_MGIOL to CNT_0 (totaling 5.028ns) exceeds
      4.000ns offset CLK to CNT_0 by 1.028n
```

Add a Delay Adjustment Factor

To add a delay adjustment factor to the sysCLOCK PLL:

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. Double-click the **Design Planner (Pre-Map)** tool.

   The Spreadsheet View, Package View, and Design Planner Control windows appear.
3. In Spreadsheet View, select the **PLL Attributes** tab.

   A single row appears with the sysCLOCK PLL ec_pll_inst added to your design earlier.
4. Double-click the **FDEL** cell and select **-5** from the list.
5. Choose **File > Save** to update pinassign_pll.lpf.
6. Choose **File > Exit** in the Design Planner Control window to exit the Spreadsheet View.
7. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
8. Double-click **Place & Route TRACE Report**.

   After a few moments, the updated pinassign_pll.twr appears in the output panel.

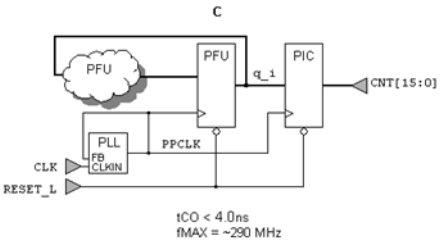
Now both preferences pass the timing constraints, as shown in Figure 43.

Figure 43: Preferences Passing Timing Constraints in Place and Route TRACE Report

```
================================================================================
Preference: CLOCK_TO_OUT ALLPORTS 4.000000 ns CLKPORT "CLK" ;
            16 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------


Passed:  The following path meets requirements by 0.222ns

 Logical Details:  Cell type  Pin type       Cell/ASIC name  (clock net +/-)

   Source:         FF         Q             CNT_0__Qio  (from PPCLK +)
   Destination:    Port       Pad           CNT_0

   Data Path Delay:     4.250ns  (100.0% logic, 0.0% route), 2 logic levels.

   Clock Path Delay:    2.039ns  (-23.1% logic, 123.1% route), 2 logic levels.

 Constraint Details:

      2.039ns delay CLK to CNT_0_MGIOL less
      2.511ns feedback compensation
      4.250ns delay CNT_0_MGIOL to CNT_0 (totaling 3.778ns) meets
      4.000ns offset CLK to CNT_0 by 0.222ns
```

Figure 44 illustrates the implementation as a block diagram of the logical implementation and the approximate timing results.

Figure 44: Logic Implemented as Block Diagram



Often the first and easiest approach to improving results is to raise the placement effort and the number of routing passes performed by PAR.

Perform Placement and Routing

To place and route the design:

1. Make note of the operating frequency reported in the Report Summary of the pinassign_pll.twr TRACE report so you can compare the results of this task to it. Look at the Actual column of the FREQUENCY NET “PPCLK” 250 MHz preference.
2. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
3. Select the **Place & Route Design** process.
4. Click the right mouse button, and choose **Properties** to open the Properties dialog box.
5. Click **Defaults**.

   Placement effort and routing passes are updated to the default values:

   - Placement Effort Level: **5**
   - Routing Passes: **6**
6. Click **Close**.
7. In Project Navigator, select the **Place & Route Design** process.
8. Click the right mouse button, and choose **Force One Level**.

   Force One Level executes only this particular stage of the process flow. It is a convenience as you experiment with different process property settings or user constraints. If Project Navigator requires input files to be updated, other processes may be executed as well. The Automake Log tab of the output panel displays the run results.
9. In Project Navigator, double-click **Place & Route TRACE Report**.

   After a moment, the updated pinassign_pll.twr appears in the output panel.

10. Review the operating frequency reported in the Actual column of the FREQUENCY NET “PPCLK” 250 MHz preference in the Report Summary of the pinassign_pll.twr TRACE report.

Guide Component Placement

Guiding placement of components is part of a floorplanning strategy that may improve (or in some cases degrade) the results of your design. Location constraints guide the placement phase of the placement and routing and, depending on the organization of your logic, can reduce the delay of critical paths. Location constraints can be declared within HDL source files or as part of an ASCII preference file, <*project*>.lpf, produced by Design Planner or the EPIC Device Editor. In this experiment, all PFU/PFF slices of your design are associated with a physical group (or PGroup), which PAR will attempt to place in proximity with each other. This sort of packing may shorten the overall routing distances and improve maximum frequency.

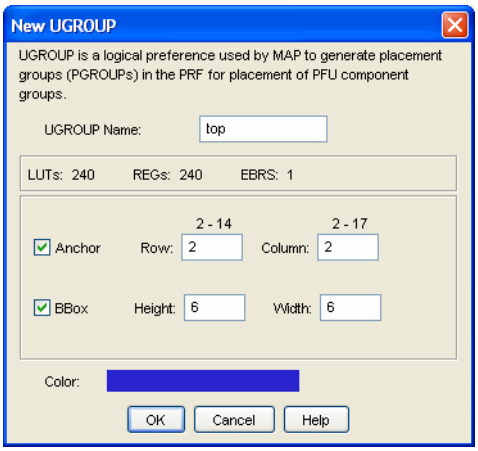
To guide component placement of the pinassign_PLL module:

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. Double-click **Design Planner (Post-PAR)** to open Design Planner.
3. Maximize the Post-Mapped View window by double-clicking on the window pane title bar (if you do not see the Post-Mapped View, choose **View > Post-Mapped View** in the Design Planner Control window).
4. Expand the Instance tree, and select all slices in the netlist (right) pane of the Post-Mapped View window by clicking on the first slice (SLICE_0) in the list, then shift-clicking the last slice (SLICE_8).
5. Right-click anywhere on the list and choose **New UGROUP** to activate the New UGROUP dialog box, as shown in Figure 45.
6. In the New UGROUP dialog box, do the following:
   a. Type **top** into the UGROUP Name entry box.
   b. Select **BBox**, and enter **6** into both the Height and the Width boxes.
   c. Select **Anchor** so you can specify the upper left row and column of a rectangle that will anchor the UGROUP in the device resource area. Accept the defaults in the Row and Column boxes.
   d. Click **OK**.
7. From the Design Planner Control window, choose **File > Save**.
8. Choose **File > Exit**.
9. To view the results of the UGROUP constraint along with all the other constraints that you created in earlier tasks, double-click the **Edit Preferences (ASCII)** process tool.

   The project’s preference file, pinassign_pll.lpf, appears in Text Editor as UGROUP “top”:

```
UGROUP "top" BBOX 6 6
        BLKNAME q_i_cry_0_4
        BLKNAME q_i_0
        BLKNAME q_i_cry_0_10
```

Figure 45: New PGROUP Dialog Box




```
            BLKNAME q_i_1
                .
                .
                .
            BLKNAME q_i_15
            BLKNAME RESET_L_c_i
            BLKNAME q_i_cry_0_0
            BLKNAME q_i_cry_0_2;
    LOCATE UGROUP "top" SITE "R2C2D" ;
```

10. Choose **File > Exit**.
11. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
12. Double-click the **Place & Route Design** process.

    PAR is executed with the new PGROUP preference. The Automake Log tab of the output panel displays the run results.
13. In Project Navigator, double-click **Place & Route TRACE Report**.

    After a few moments, the updated pinassign_pll.twr appears in the output panel.
14. Review the operating frequency reported in the Actual column of the FREQUENCY NET “PPCLK” 250 MHz preference of the pinassign_pll.twr TRACE report. Given the relative size of this design example, more routing iterations and grouping will not improve the design beyond what has already been accomplished through the RTL code style and use of the internal PLL.

Note

To see how the PGroup location constraint influenced the placement and routing, open the Post-PAR Design Floorplan and notice how the slices are now closer together.

Task 9: Examining Device Utilization

In this task, you will view the resource utilization and critical paths of the placed and routed design in Design Planner. Design Planner illustrates connection and utilization density with colored graphics and allows you to trace any signal path.

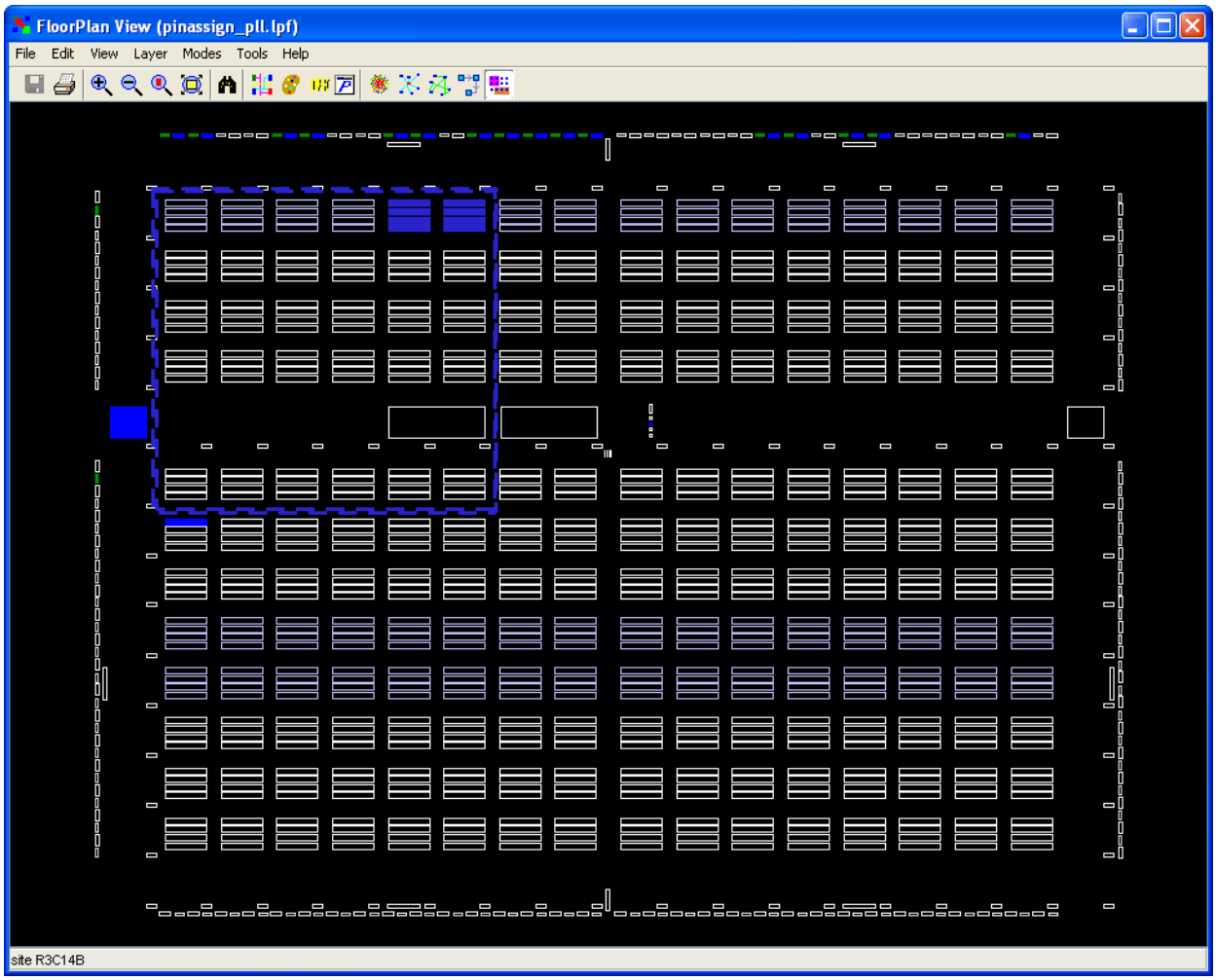
View the Device Implementation

To view the device implementation after placement and routing:

1. In Project Navigator, select the **LFEC1E-3T100C** device in the Sources in Project window.
2. Double-click **Design Planner (Post-PAR)** to open Design Planner.
3. If Floorplan View and Post-Mapped View do not appear, choose **View > Floorplan View** and **Post-Mapped View** from the Design Planner Control window.
4. Click anywhere within Floorplan View and choose **View > Zoom to Fit**.

   The device architecture fills the window, as shown in Figure 46. The LatticeEC PIC, PFU, PFF, and EBR elements that are shown as small boxes in the display similar to the simplified block diagram of the data sheet.

Figure 46: Device Architecture in Floorplan View



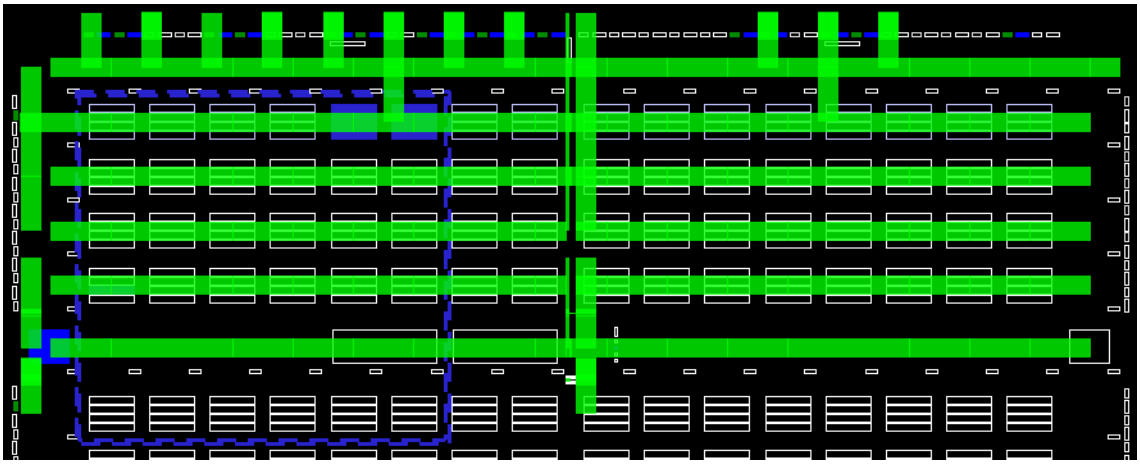
View the Routing Congestion

To view the routing congestion:

1. From the Floorplan View toolbar, choose **Layer > Congestion**.

   Horizontal and vertical colored bars appear in the display to illustrate channel congestion of short PFU connections, as shown in Figure 47. By default, green indicates low congestion, and red indicates high congestion.

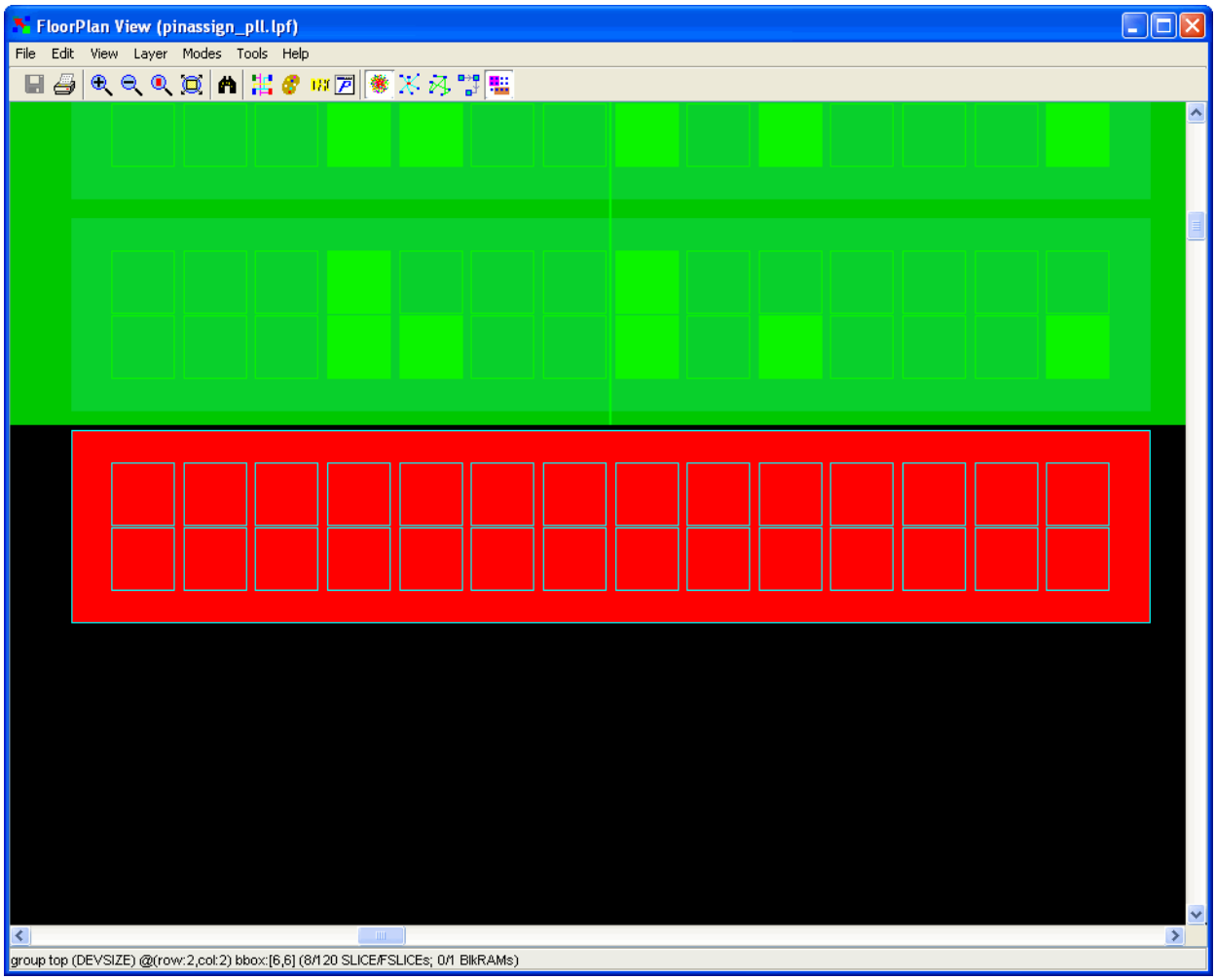
Figure 47: Channel Congestion of Short PFU Connections



2. In Post-Mapped View, open the Instance tree and select **SLICE_0(FSLICE)@R*n*C*n***, where *n* is a row or column number from the Instance tree in the right pane.
3. Click the right mouse button and choose **Locate > Floorplan View**.

   The Floorplan View highlights the PFU/PFF slice and changes the zoom level, as shown in Figure 48.

Figure 48: PFU/PFF Slice in Floorplan View



4. Choose **File > Exit** in the Design Planner Control window.

Note

If you want to exit the tutorial at this point and resume it later, choose File > Save in Project Navigator to save the pinassign_pll.syn file. To resume the tutorial, re-load this file by choosing **File > Open**.

Task 10: Estimating Power Consumption

In this task, you will review the report created by Power Calculator, which uses your estimates of activity factors and frequency to calculate power consumption. The tool is flexible enough to use early estimates of device utilization and operational behavior with pre-route or post-route design databases.

Estimate Power for Routed Design

To use Power Calculator to estimate power consumption for a routed design:

1. In Project Navigator, choose **Tools > Power Calculator**.

   After a few moments, the Power Calculator application appears.

2. Choose **File > New** to open the Power Calculator – New Project dialog box.

3. In the dialog box, check that:

   - Project Name shows **pinassign_pll**.
   - Project Directory shows the correct path (<*install_path*>\examples\tutorial\fpga_design_tutor).
   - NCD File shows <*install_path*>\examples\tutorial\fpga_design_tutor\ **pinassign_pll.ncd**.

4. Click **Finish**.

   The main Power Calculator window appears with the Power Summary tab selected, as shown in Figure 49.

   The Power Summary tab shows information about the device, its operating environment, and summaries of the design's current and power requirements. The Power Calculator window offers several other tabs to show how different parts of the device are affecting the power requirements.

5. Choose **Edit > Frequency Settings**.

6. Select **Use TWR**.

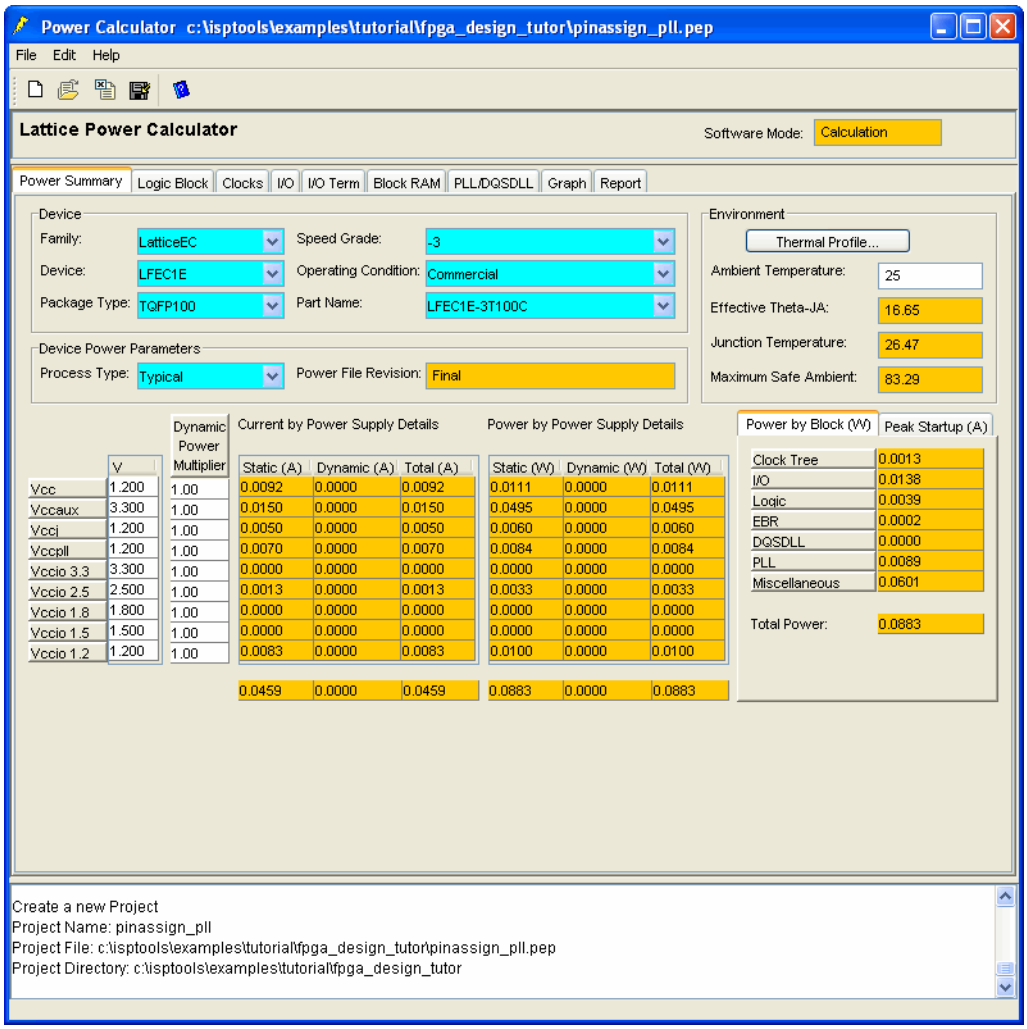
7. Click the Browse button and select **pinassign_pll.twr** from the list. Click **Open**.

8. Click **OK**.

   The Power Calculator loads the preference file to obtain frequency constraints of the design.

9. Choose **Edit > Activity Factor Settings**.

Figure 49: Power Summary Tab in Power Calculator Window



10. In the Activity Factor Default box, type **31.25**.

    The activity factor (AF) percentage for the 16-bit counter is taken from the number of signal transitions per rising clock edge. Considering each counter bit from LSB to MSB:

    AF = 1 (LSB) + 1/2 + 1/4 + … + 1/215 (MSB) = ~2

    AF/Counter nodes = 2/16 = 0.125

    AF (%) = 0.125 x 250 MHz = 31.25 MHz

11. Click **OK**.
12. Click the **Logic Block** tab.

    The COMBINATORIAL row indicates the unclocked logic of the design, in this case, the asynchronous reset. The PPCLK row illustrates the synchronous logic of the design clocked by PPCLK. You can see that the logic slice and ripple-carry logic cells reflect the utilization by the 16-bit counter logic.

13. Click the **Block Ram** tab.

    No rows are included in this section since the tutorial design does not use embedded block RAM.

14. Click the **I/O** tab.

    The PPCLK row indicates the synchronous IOLOGIC of the counter's output register clocked by PPCLK. The I/O Register cells of single data rate (sdr) indicate the single-ended I/O standard of the design's LVCMOS sysI/O buffers (16 outputs and 2 inputs).

15. Enter the following AC characteristic value in the **I/O** tab:
    - COMBINATORIAL, Frequency (MHz): **250**
16. Click the **Graph** tab.

    In a few moments, the graphs appear, showing how total power changes with changing V_{CC}, temperature, and PPCLK frequency.

17. Click the **Report** tab.

    The Report tab summarizes the data in a printable form.

18. Choose **File > Save**.

You can quickly change your analysis to consider a different device by adjusting the Family, Device, or Part Name fields.

Estimate Power for a Different Temperature

One of the critical factors in reducing power consumption is to minimize the operating temperature of the device. Power Calculator offers different models to estimate the temperature for a given device when it is mounted on a board.

To estimate the power consumption for a different operating temperature:

1. Click the **Power Summary** tab.
2. Click **Thermal Profile** in the Environment section.

3. In the Thermal Profile dialog box, under Heat Sink Selection, choose **Medium Profile Heat Sink**.

   The Effective Theta-JA value decreases.

4. Click **OK**.

   Values in the Power Summary tab are automatically updated to show the effects of the reduced temperature.

Estimate Power for a Different Device

To estimate the power consumption for a different device:

1. From the Device list on the Power Summary tab, choose the 6.1K LUT, **LFEC6E**.

   The values in the Power Summary tab increase to reflect the larger device's power consumption.

2. Choose **File > Exit**.

Task 11: Simulating the Design

In this task, you prepare for simulation by using utilities for creating a Verilog test fixture or a VHDL test bench, and run the simulator by using the scripts created by Project Navigator. Although this task is one of the last in the tutorial, you can perform a functional simulation on all or a portion of the module hierarchy at any time.

In the first portion of this task, you will create a Verilog test fixture or a VHDL test bench to drive the unit under test (UUT), module pinassign_PLL.

See "Create a Verilog Test Fixture" on page 57 if you are using a Verilog HDL source file, or see "Create a VHDL Test Bench" on page 60 if you are using a VHDL source file.

Create a Verilog Test Fixture

To create a Verilog test fixture:

1. In Project Navigator, select the **pinassign_PLL (pinassign_pll.v)** module.

   Several Verilog source-related processes appear in the Processes for Current Source window.

2. Double-click the **Verilog Test Fixture Declarations** process output file.

   A Verilog test fixture declarations file is created in the project directory. This file includes the port interface details of the pinassign_PLL module that you will refer to later in a Verilog test fixture.

3. Choose **Source > New** to open the New Source dialog box.

4. Select **Verilog Test Fixture** and click **OK**.

5. In the Associate Verilog Test Fixture dialog box, select the target device, **LFEC1E-3T100C**, from the Associate With list, and click **OK**.

   Text Editor opens, and the New File dialog box appears.

   The associate function enables you to determine which module of your design you want a test fixture to drive. It accommodates a bottom-up functional verification approach as you add more and more modules to your design. Associating the test fixture with the target device indicates that the "top" module and those of the hierarchy below it are to be simulated for both functional and timing simulation.

6. Type **pinassign_PLL_tf** into the File Name box of the New File dialog box and click **OK**.

   The new, empty test fixture file appears in Text Editor and is added to the source list of Project Navigator as a Verilog test-fixture-type file.

7. In Text Editor, choose **Templates > Insert** to open the Insert Template dialog box, shown in Figure 50.

   A variety of Verilog HDL source code templates are provided to ease design entry.

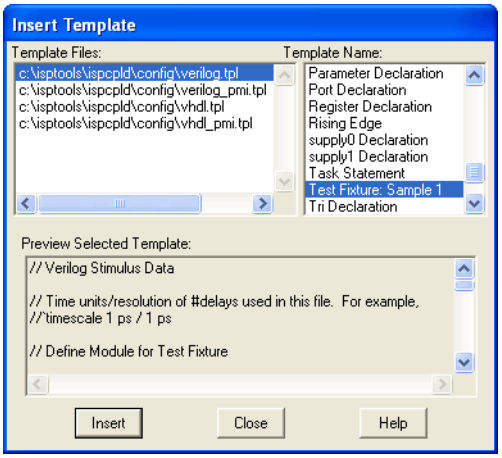
8. Select the following in the dialog box:

   a. Template Files: <isptools>\ispcpld\config**verilog.tpl**

   b. Template Name: **Test Fixture: Sample 1**

9. Click **Insert**.

Figure 50: Insert Template Dialog Box



   A template of Verilog code appears in Text Editor.

10. In the Insert Template dialog box, click **Close**.

A simple test fixture module called “t” is provided here with commented sample stimulus statements.

Note

A completed test fixture file is provided in the following project subdirectory. You can copy this file into your project directory, or you can view this file for reference while you complete this portion of the task.

<*install_path*>\examples\tutorial\fpga_design_tutor\pinassign_pll_lab.tf

11. Modify the timescale directive at line 4 from:

    ```
    //`timescale 1 ps / 1 ps
    ```

    to:

    ```
    `timescale 1 ns / 1 ns
    ```

12. Add the following compiler directive to line 5:

    ```
    `define auto_init;
    ```

13. Modify the reference to the test fixture declaration file at line 11 from:

    ```
    //`include "p2_top.tfi"
    ```

    to:

    ```
    `include "pinassign_PLL.tfi"
    ```

14. Modify the code for the clock stimulus at line 24 from:

    ```
    //always #(clock_period / 2) clk = ~clk;
    ```

    to:

    ```
    always #5 CLK = ~CLK;
    ```

15. Modify the code for the reset stimulus beginning at line 33 from:

    ```
    //my_reset[0:1] = 2 'h 1; #22500;
    //my_reset[0:1] = 2 'h 2; #15500;
    //my_reset[0:1] = 2 'h 3; #28500;
    ```

    to:

    ```
    #87 RESET_L = 0;
    #95 RESET_L = 1;
    ```

    Note

    Verilog is case-sensitive, so you must enter the code exactly as just given.

16. Choose **File > Save**.
17. Choose **File > Exit**.

Continue with the tutorial by going to “Start Functional Simulation” on page 61.

Create a VHDL Test Bench

To create a VHDL test bench:

1. In Project Navigator, select the **pinassign_PLL (pinassign_pll.vhd)** module.

   Several VHDL source-related processes appear in the Processes for Current Source window.

2. Double-click the **VHDL Test Bench Template** process output file.

   A VHDL test bench template called pinassign_PLL.vht appears in the output panel. This file includes a design unit that instantiates the pinassign_PLL component.

3. Choose **Source > New** to open the New Source dialog box.

4. Select **VHDL Test Bench** and click **OK**.

5. In the Associate VHDL Test Bench dialog box, select the target device, **LFEC1E-3T100C**, from the Associate With list, and click **OK**.

   Text Editor opens, and the New File dialog box appears.

   The associate function enables you to determine which module of your design you want a test bench to drive. It accommodates a bottom-up functional verification approach as you add more and more modules to your design. Associating the test bench with the target device indicates that the “top” module and those of the hierarchy below it are to be simulated for both functional and timing simulation.

6. Type **pinassign_PLL_tb** in the File Name box of the New File dialog box and click **OK**.

   The new file appears in the Text Editor, although it has no contents, and is added to the source list of Project Navigator as a VHDL test-bench-type file.

7. Choose **File > Open**.

8. Type **pinassign_pll.vht** in the File Name box, and click **Open**.

   The test bench template appears.

9. Choose **Edit > Select All** and **Edit > Copy**.

10. Choose **Window > pinassign_pll_tb.vhd**.

11. Choose **Edit > Paste**.

12. Modify the SIGNAL declarations beginning at line 38.

```
SIGNAL CLK :  std_logic:='1';
SIGNAL RESET_L :  std_logic:='1';
```

13. Add the following code at line 48:

```
RESET_L <=   '0' after 87 ns,
             '1' after 95 ns;
```

```
CLK <= not(CLK) after 5 ns;
```

Note

A completed test bench file is provided in the following project subdirectory. You can copy this file into your project directory, or you can view this file for reference while you complete this portion of the task.

<*install_path*>\examples\tutorial\fpga_design_tutor\pinassign_pll_tf.vhd

14. Choose **File > Save**.
15. Choose **File > Exit**.

Start Functional Simulation

In this portion of the task, you will start the default simulator using the Project Navigator process for functional simulation.

To start the simulator from Project Navigator:

1. Select **pinassign_pll_tf.v** (or **pinassign_pll_tb.vhd**).

   Several simulation-related processes appear in the process window.

2. Double-click the **Verilog** (or **VHDL**) **Functional Simulation** tool process.

   The default simulator is started using a script file generated by Project Navigator.

3. Exit the simulator.

Note

For more information, see the “Design Simulation” section of the “Design Flow User Guide” in the online help.

Start Timing Simulation

Now you will start the simulator to perform the timing simulation.

To start the default simulator from Project Navigator for timing simulation:

1. Select **pinassign_pll_tf.v** (or **pinassign_pll_tb.vhd**).

   Several simulation-related processes appear in the Processes for Current Source window.

2. Double-click the **Verilog** (or **VHDL**) **Post-Route Timing Simulation** tool process.

   The default simulator is started using a script file generated by Project Navigator.

3. Exit the simulator.

Summary

You have completed the “LatticeEC FPGA Design with ispLEVER Tutorial.” In this tutorial, you have learned how to do the following:

- Use ispLEVER to create a new Schematic/Verilog HDL or VHDL project and target a device.
- Import source files into Project Navigator.
- Generate a sysCLOCK PLL module from the Module/IP Manager.
- Synthesize and build the .ncd physical design database.
- Map, place, and route the design and use Design Planner to view the results.
- Run static timing analysis and view the results.
- Calculate power-consumption estimates of the pre-route design by using Power Calculator.
- Create a Verilog test fixture or VHDL test bench and simulate the design.

Glossary

Following are the terms and concepts that you should understand to use this tutorial effectively.

bank. A bank is an organization of LatticeECP/EC sysIO buffers arranged around the periphery of the device.

clock tree or **clock spine**. A clock tree or a clock spine is a clock distribution network in the LatticeECP/EC device architecture used for high-fanout signals.

clock-to-out delay (t_{CO}). Clock-to-out delay (t_{CO}) is the time it takes for a signal to go from the primary input pin through the clock of the flip-flops or gate of latches to the primary output pin.

critical path. The critical path is the path through a circuit that determines the maximum speed at which the circuit can operate. The critical path is a signal in a section of combinatorial logic that limits the speed of the logic. Storage elements begin and end a critical path, which may include I/O pads.

embedded block RAM (EBR). An embedded block RAM (EBR) is a large dedicated fast memory block in the LatticeECP/EC device architecture.

F_{MAX}. F_{MAX} is the symbol for the maximum operating frequency expressed in hertz. Clock frequency is the number of complete clock cycles that occur per unit of time. For example, a 50-MHz clock signal cycles 50 million times per second. The maximum frequency is a function of the longest combinatorial delay between registered elements.

IOLOGIC. An IOLOGIC is a representation of PIO logic within a PIC block that is generated by the ispLEVER Design Planner.

lookup table (LUT). A lookup table is an architectural element within a LatticeECP/EC PFU for implementing combinational logic or memory.

Manhattan routing. Manhattan routing is a representation of the physical routing interconnect among slices in which the routing is rectilinear.

.lpf file. The logical preference file (.lpf) is a post-synthesis FPGA constraint file that stores logical preferences that have been defined in the pre-map stage and post-map stage. This file is automatically generated when you create a new project in Project Navigator, and it stores logical preferences only.

.ncd file. An .ncd file is a binary format FPGA post-map physical design database file generated by the Map Design process of Project Navigator. The .ncd file includes mapped and potentially placement and route information.

.ngd file. An .ngd file is a binary format FPGA pre-map logical design database file generated by the Build Database process in Project Navigator. The .ngd file represents the logical design information of the ASCII EDIF netlist.

programmable function without RAM/ROM (PFF). A programmable function without RAM/ROM (PFF) is a PFU that does not contain a RAM.

phase-locked loop (PLL). A phase-locked loop is an electronic circuit with a voltage- or current-driven oscillator that is constantly adjusted to match in phase, so it locks on the frequency of an input signal. In addition to stabilizing a particular communications channel (keeping it set to a particular frequency), a PLL can be used to generate a signal, modulate or demodulate a signal, reconstitute a signal with less noise, or multiply or divide a frequency. PLLs are more commonly used for digital data transmission but can also be designed for analog information. The LatticeECP/EC device architecture contains up to four analog PLLs per device.

programmable function unit (PFU). A programmable function unit (PFU) is a block within the LatticeECP/EC device that implements combinational logic, memory, and registers. The core of the architecture consists of PFU blocks that can be programmed to perform logic, arithmetic, distributed RAM, and distributed ROM functions.

programmable I/O cell (PIO). A programmable I/O cell (PIO) is an architectural element within an FPGA that handles actual input/output functions. It includes the I/O registers, tristate registers, and control multiplexers of a PIC.

programmable interface cell (PIC). A programmable interface cell (PIC) is a cell within a LatticeECP/EC device architecture that contains a group of two PIOs. The edges of the architecture consist of PIC blocks that contain two PIOs connected to sysIO buffers. PICs provide high-speed I/O registers and buffering to support a variety of signal interface standards.

slice. A slice is an architectural element within an FPGA consisting of two LUT4 lookup tables that feed two registers (programmed to be in FF or latch mode), and some associated logic that allows the LUTs to be combined to perform functions such as LUT5, LUT6, LUT7, and LUT8. It also includes control logic to perform set/reset functions (programmable as synchronous/ asynchronous), clock select, chip-select, and wider RAM/ROM functions. The registers in the slice can be configured for positive or negative and edge or level clocks. There are four interconnected slices per PFU block. Each slice in a PFU is capable of four modes of operation: logic, ripple, RAM, and ROM. Each slice in the PFF is capable of all modes except RAM.

voltage-controlled oscillator (VCO). A voltage-controlled oscillator (VCO) is an oscillator whose frequency is regulated by its input voltage.

Recommended Reference Materials

The following reference materials are recommended for this tutorial:

- *LatticeECP & LatticeEC FPGA Handbook*
- *LatticeECP/EC Family Data Sheet*
- TN1049, *LatticeECP/EC sysCLOCK PLL Design and Usage Guide*
- TN1052, *Estimating Power Using the Power Calculator for LatticeECP/EC Devices*
- Lattice ispLEVER Help
- *FPGA Design Guide*