Actel Quick Start Guide

Libero v6.1

Actel

Actel Corporation, Mountain View, CA 94043

Printed in the United States of America

Part Number: 5029123-7

Release: October 2004

Trademarks

Table of Contents

Introduction

The *Actel Quick Start Guide for Libero* contains information for using the Libero IDE software to create designs for, and program, Actel devices.

This manual includes information about the Libero IDE software, which allows you to generate and/or import a netlist generated from a third-party CAE tool, perform pre-synthesis simulation, synthesize your design, place-and-route the design, run physical synthesis, perform static timing analysis, extract timing information, estimate power consumption, and generate a programming file to program an Actel FPGA.

This manual also refers to other Actel documents that contain additional information, including CAE software interface guides and simulation guides with specific information about using CAE tools with the Libero IDE.

Document Organization

The *Actel Quick Start Guide for Libero* is divided into the following chapters:

Chapter 1 - Libero IDE Overview gives an overview of the programs contained in the Designer software.

Chapter 2 - Actel Design Flows illustrates and describes the design flow for creating Actel designs using the Designer software and third-party CAE tools.

Chapter 3 - Quick Start Tutorial illustrates a basic VHDL design for the APA Evaluation Board targeted at the Actel ProASIC3E family.

Chapter 4 - Design Considerations contains useful information and procedures about creating designs using the Actel Designer software.

Appendix A - PDC Constraints in Libero IDE lists some of the most common I/O and logic constraints for ProASIC3E

Appendix B - Synopsys Design Constraints (SDC) in Libero IDE lists some of the common SDC (timing) constraints for ProASIC3E.

Appendix C - Product Support provides information about contacting Actel for customer and technical support.

Document Assumptions

The information in this manual is based on the following assumptions:

1. You have installed the Libero IDE software.
2. You are familiar with PCs and Windows operating environments.

3. You are familiar with FPGA architecture and FPGA design software.
4. You are familiar with Libero IDE software and have prior experience with the ProASIC or ProASIC^{PLUS} families.

Document Conventions

The <act_fam> variable represents an Actel device family. To reference an actual family, substitute the name of the Actel device when you see this variable.

The <vhd_fam> variable represents Compiled VHDL libraries. To reference an actual compiled library, substitute the name of the library when you see this variable. Compiled VHDL libraries must begin with an alpha character.

Your Comments

Actel Corporation strives to produce the highest quality online help and printed documentation. We want to help you learn about our products, so you can get your work done quickly. We welcome your feedback about this guide and our online help. Please send your comments to documentation@actel.com.

Online Help

The Designer software comes with online help. Online help specific to each software tool is available in Libero IDE, Designer, ACTgen, Silicon Explorer II, and Silicon Sculptor.

Actel Manuals

Libero IDE includes printed and online manuals. The online manuals are in PDF format and available from Libero's Start Menu and on the CD-ROM.

The complete list of Libero manuals is available on the Actel website at http://www.actel.com.

1 Libero IDE Overview

Actel's Libero IDE offers the latest and best-in-class tools from leading EDA vendors such as Mentor Graphics, SynaptiCAD, Synplicity, and our latest partner - Magma Design Automation. These tools, combined with custom developed tools from Actel, are all integrated into a single FPGA development package. The Libero IDE flow includes a powerful design manager that guides you through the design process, keeps track of your design files, and manages file exchanges between the various tools. Libero IDE includes Actel's Designer software, which offers premier backend tools for physical implementation, including a comprehensive floorplanning capability via Actel's ChipPlanner feature. Designer is available as a standalone product, for those who want to use their own design and verification tools. Libero IDE features include:

- Powerful project and design flow management
- Schematic and HDL design capability
- Combining of schematic and VHDL or schematic and Verilog design flows
- Automatic core generation (with the ACTgen core generator)
- VHDL or Verilog behavioral, post-synthesis and post-layout simulation capability
- VHDL / Verilog Synthesis
- Physical implementation, floorplanning, and place-and-route
- Timing analysis and constraints entry
- Power analysis
- Real-time internal probing of the programmed device using Silicon Explorer II
- Cross-probing links between tools
- ChainBuilder enables programming of multiple devices on a chain through a common header Programming software for ProASIC and ProASIC^{PLUS} devices

Libero IDE Family Support

Libero IDE provides full design flow support for all devices, including the new ProASIC3/E family. An experienced Libero IDE user that is comfortable designing with Actel's ProASIC^{PLUS} family will notice differences in the ProASIC3/E design process.

Importing Files

Libero supports new constraint files for the ProASIC3/E families. ProASIC3/E families utilize Physical Design Constraint (PDC) files for physical constraints and Synopsys Design Constraint (SDC) files for timing constraints. GCF constraint files that have been used for ProASIC and

ProASIC^PLUS families are not supported in ProASIC3/E. Instead, timing constraints for ProASIC3/E devices must be imported via the Timer user interface or as an auxiliary file in the SDC format. All GCF physical constraints (physical placement constraints and global resources constraints) for ProASIC3/E files must be imported as a source file in the PDC format. As before, all netlist formats (EDIF, Verilog, VHDL) are supported.

Compile Options and Compile Report for ProASIC3/E

A new Compile Options graphical user interface is available when the you click Compile in Designer (in Libero IDE). You can change the global routing resources and fanout settings for your ProASIC3/E device with this new interface. Compile Options provide global control over your design and you can use PDC files for port, instance, and net specific constraints.

The Compile report has been redesigned to show detailed information when compiling a design. Detailed information is divided into the following sections:

- Information about the design, device selection, and the compile options
- Compile information and a list of warnings or errors
- A netlist optimization report that displays information about optimized macros
- A device utilization report that displays the number of core logic tiles and I/Os that are used in the design
- A net information report displays information about global clock nets, quadrant clock nets, local clock nets, high fanout nets, and the nets that are candidates for local clock assignment

I/Os

I/O attribute support for ProASIC3/E is significantly improved over previous versions. ProASIC3/E devices offer a wide variety of I/O options, and the software enables you to easily select a large number of attributes for any I/O or I/O combination you wish to use in the design. In addition, you can select and change attributes without having to rerun compile and layout.

You can access ProASIC3/E I/O macros three different ways:

1. Explicitly instantiate user specified (non-default) I/O macros in the top-level design
2. Use generic I/O macros and select from the I/O Attribute Editor within the MultiView Navigator to specify the desired attributes
3. Any combination of the above

Any I/O pair can be used as a PECL I/O. For non-default I/O macros, please refer to the ProASIC3/E Macro Library Guide for detailed information.

Timer and Timing Simulation with ProASIC3/E

Timing simulation is performed after the design has completed place-and-route and the timing information is based on the delays in the placed-and-routed design. Timing simulation for ProASIC^{PLUS} devices uses the pre-optimized netlist, which is the original netlist used by the Designer software. ProASIC3/E devices use an Actel-flattened netlist for timing simulation, which simplifies the debug process.

ACTgen

ACTgen is a graphical core generation tool that creates optimized logic elements that can be easily included in your schematic or synthesis design. Architecture-specific rules control the generation of cores, so the quality of output is "correct by construction," and no logic verification is required. Refer to the ACTgen online help for additional information on how to use ACTgen. Refer to the *ACTgen Cores Reference Guide* for a complete description of all the cores available in ACTgen. The ACTgen tool and user interface has been redesigned to enable easy use of the advanced architectural features in ProASIC3/E.

PLL Core Wizard

ProASIC3/E devices offer flexible clock conditioning capabilities for the PLLs. PLL configuration for ProASIC3/E is significantly different that previous versions, offering a higher degree of programmability from the user interface.

RAM/FIFO User Interface

In ProASIC3/E families, SRAM is organized as 4,608 bits. The embedded memory in the ProASIC3/E families has a vastly improved and highly flexible architecture. The ACTgen user interface for ProASIC3/E RAM/FIFO accommodates these changes. With the new ACTgen tool, only two RAM cores (RAM4K9 and RAM512X18) and one FIFO core (FIFO4K18) are used for all memory configurations, and it now automatically cascades RAM blocks to create wider and deeper memories while choosing the most efficient aspect ratio. ACTgen also supports the generation of memories that have different read and write aspect ratios.

Note: The asynchronous memory feature is not available in the ProASIC3/E family. Therefore, designs incorporating asynchronous RAM or FIFO must be modified.

Designer

Designer is an interactive design implementation tool that can import designs created with Libero or third-party schematic and HDL CAE tools. Designer features fully automatic layout, pin fixing, a

chip editor, netlist viewer, and a back-annotation utility. Designer also includes Timer (which provides tools for timing driven place-and-route and static timing analysis) and a power analysis tool that enables designers to estimate the power consumption of their design. These tools enable designers to define, layout (place-and-route), verify, and program high performance designs at high levels of utilization with improved productivity.

Design Flow Manager

Designer uses a Design Flow Manager that displays the completed steps of the design implementation process. The Design Flow Manager also keeps track of information required to begin each step of the design's current status and design source changes.

Designer also uses demand-driven options that allow you to click any button in the Designer Main window to begin the design implementation process. Designer then prompts you through all of the necessary steps of the flow to complete the step that you selected.

Compile

Compile reads in a netlist and compiles the design into an Actel database (ADB) file. Compile contains a variety of functions, including the Combiner and the Design Rule Checking functions, that perform legality checking and basic netlist optimization. Compile also checks for netlist errors (bad connections and fan-out problems), removes unused logic (gobbling), and combines functions to reduce logic count and improve performance (combining or logic collapsing). In addition, Compile verifies that the design fits into the selected device. Refer to the Designer or Libero online help for additional information.

Axcelerator Only: Compile can also combine I/Os and Registers (where possible) on a per-I/O or design basis, depending on your requirements.

Layout

Layout (place-and-route) takes the design netlist information, pin information (optional), and Timing information (Timing Driven Layout only), and maps the information into the selected device. The Incremental Layout option enables designers to speed through design iterations. Designer supports two layout modes: Standard and Timing-Driven.

Standard Layout

Standard layout is available for non-performance-critical applications. Standard layout maximizes the average performance for all paths and treats each part of a design equally for performance optimization, using net weighting (or criticality) to influence the results. Standard layout is the faster of the two modes.

Timing Driven Layout

For timing critical designs, Timing Driven Layout uses timing requirements entered in Timer to constrain Layout. Timing Driven Layout is a fully automatic place-and-route utility that focuses resources to meet performance requirements.

Axcelerator Only: There are five different placement effort levels available for Axcelerator devices (marked from 1-5, five is the highest). The higher the effort level, the better chance you have of meeting your routability and performance goals. However, there is a corresponding increase in run time for the higher effort levels.

Back-annotate

Back-annotate lets you generate a post-layout timing file (*.sdf) and a netlist file. For Axcelerator, Back-annotate generates a Post-Compile optimized netlist.

Fuse (or Bitstream)

Fuse generates the necessary files to program an Actel device. Refer to the Designer or Libero online help for additional information.

User Tools

The user tools are NetlistViewer, PinEditor, ChipPlanner/ChipEditor, Timer, and SmartPower. They enable you to view and modify pin locations and macro placement, and to evaluate the timing and the power consumption for your design. Tool availability varies depending on device family; not all tools are available for all devices. See the tool descriptions below for a list of the supported families.

MultiView Navigator

The MultiView Navigator interface supports Axcelerator and all Flash families. This interface integrates the NetlistViewer, PinEditor, ChipPlanner and I/O Attribute Editor tools.

The tools for all other device families do not open in MultiView Navigator, they open in separate windows.

NetlistViewer

The NetlistViewer displays your netlist in a hierarchical manner (though only if you import a hierarchical, or non-flattened, netlist), providing you with a logical view of your design. The NetlistViewer can be used alone to explore each level of the hierarchy and to trace signals. Used with PinEditor, ChipEditor, SmartPower, or Timer, the NetlistViewer assists you in meeting area, power, and timing goals by helping you with critical path identification.

Axcelerator Only: When you use the NetlistViewer with Axcelerator devices, you can choose to display the **Pre-Optimized Netlist** or **Optimized Netlist** using the **View** menu.

PinEditor

PinEditor is a graphical interface that enables designers to view pin locations and manually assign, edit, and fix pin locations for a design. Manual pin assignment is optional. Use PinEditor to fix pins that are automatically assigned by Designer to maintain pin locations once a design is ready to be used to program a device. PinEditor is also used to customize I/O attributes and assign I/O bank standards (for Axcelerator devices). Refer to the Designer online help for additional information.

ChipEditor/ChipPlanner

ChipEditor/ChipPlanner is a graphical interface that allows designers to view a design's core placement and to edit the placement of *both* I/O and logic macros. ChipEditor/ChipPlanner also enables you to see the nets between selected macros. The ChipPlanner (available for ProASIC, ProASIC^{PLUS}, ProASIC3/E and Axcelerator devices) supports floorplanning, an optional methodology that you can use to improve the performance and routability of your design. The objective in floorplanning is to assign logic to specific regions on the chip in order to enhance performance and routability.

When floorplanning, you analyze your design to see if certain logic can be clustered within regions. This is especially helpful for hierarchical designs with plenty of local connectivity within a block. If your timing analysis indicates several paths with negative slack, try clustering the logic included in these paths into their own regions. This forces the placement of logic within the path closer together and may improve timing.

Refer to the Designer online help for additional information.

I/O Attribute Editor

I/O Attribute Editor displays I/O attributes in a spreadsheet format. Use the I/O Attribute Editor to view, sort, select, and edit standard and device-specific I/O attributes.

Timer

Timer is an optional interactive tool used for timing verification and for entering timing constraints. Timing information can be displayed in a tabular or graphical format. A report generator is also provided. Refer to the Timer online help (in Designer) for additional information.

SmartPower (Axcelerator, ProASIC, ProASIC^{PLUS}, and ProASIC3/E devices only)

SmartPower is Actel's state of the art power analysis tool. Power analysis is a convenient and thorough method of analyzing, debugging and validating the power performance of a design. This is achieved by breaking the design down into a nets, blocks, and gates, and then calculating the power

requirements of the component parts. SmartPower includes a report generator that summarizes your power consumption information. Refer to the SmartPower online help for more information.

Libero IDE Maintenance

After your first installation of Libero, unless you completely remove the original installation, Setup always goes into maintenance mode. Maintenance mode (also accessible from the Start - Programs > Libero IDE menu) enables the following three options:

- Modify: Select new program components to add or select currently installed components to remove.
- Repair: Reinstall all program components installed by the previous setup.
- Remove: Remove all installed components.

2

Actel Design Flows

The Libero IDE software integrates with third party schematic and HDL CAE tools to implement, simulate, and program Actel devices. This chapter illustrates and describes the design flow for creating Actel designs using the Libero IDE software.

Design Flow Illustrated

Figure 2-1 shows the design flow for an Actel device using Libero IDE, Designer, and third-party schematic capture software.[1]

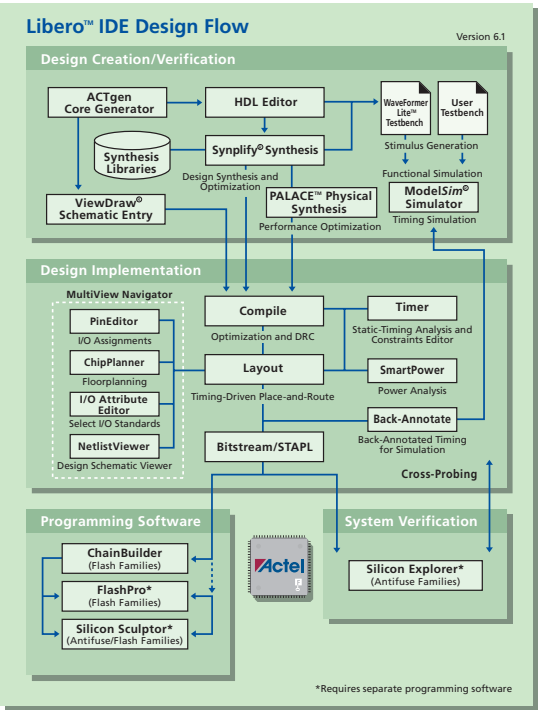


Figure 2-1. Actel Libero Design Flow

1. The MultiView Navigator interface supports only ProASIC3/E, ProASIC^PLUS, ProASIC, and Axcelerator devices. For all other families, Designer displays NetlistViewer, ChipEditor, PinEditor, I/O Attribute Editor, Timer, and SmartPower.

Schematic-Based Design Flow Overview

The Actel schematic-based design flow has four main steps; design creation/verification, design implementation, programming, and system verification. These steps are described in the following sections.

Third-party software users can also refer to the Actel *Mentor Graphics Interface Guide, Mentor Graphics eProduct Designer Interface Guide* for information about using these tools with Actel software and devices.

Design Creation/Verification

During design creation/verification, a schematic representation of a design is captured using Actel schematic capture software, such as the Libero Integrated Design Environment, or capture software from a third-party. After design capture, a pre-layout (functional) simulation can be performed with third-party simulation software. Finally, an EDIF netlist is generated for use in Designer.

Schematic Capture

Enter your schematic using a third-party schematic capture tool, such as Libero IDE. Refer to the additional documentation included with your schematic capture tool for information.

Netlist Generation

After you have captured and verified your design, you may place-and-route in Designer. You may place-and-route with an EDIF, Verilog, or VHDL netlist. Refer to your the manuals included in your simulation software for more information on creating an EDIF, Verilog, or VHDL netlist.

Functional Simulation

Perform a functional simulation of your design using Libero IDE or a third-party simulation tool before generating an EDIF netlist for place-and-route. Functional simulation verifies that the logic of the design is correct. Unit delays are used for all gates during functional simulation. Refer to the Actel Interface Guides and the documentation included with your simulation tool for information about performing functional simulation.

Design Implementation

During design implementation, a design is placed-and-routed using Designer. Additionally, static timing analysis can be performed in Designer with the Timer tool. After place-and-route, post-layout (timing) simulation is performed using third-party simulation software.

Place-and-Route

Use Designer to place-and-route your design. Refer to the Designer online help for more information.

Static Timing Analysis

Use the Timer tool in Designer to perform static timing analysis on your design. Refer to the Timer online help for more information.

You can also perform static timing analysis using third-party software. Refer to the documentation included with your static timing analysis tool for information.

Power Analysis (Optional)

Axcelerator, ProASIC, ProASIC^{PLUS}, and ProASIC3/E Families Only: The SmartPower tool is a state of the art power estimation tool. Enter your clock and data frequencies and use SmartPower to estimate both the static and dynamic power of your design, and to identify your most power-hungry blocks, nets, and gates. The tool accepts switching activity from simulation in VCD (value change dump) and SAIF (switching activities interface and probabilities file) formats.

Timing Simulation

Perform a timing simulation of your design using a third-party simulation tool after you place-and-route in Designer. Timing simulation requires that data be extracted and back-annotated from Designer. Refer to the Actel Interface Guides and the documentation included with the simulation tool for information about performing timing simulation.

Programming

Program a device with programming software and hardware from Actel or a supported third-party programming system. Refer to the Designer online help and the FlashPro User's Guide or Silicon Sculptor User's Guide for more information.

Schematic-Based Design Methodology

If you prefer designing with schematic tools, Actel offers a complete tool suite that lets you take your designs from concept to silicon. On the front end, ACTgen integrates with third-party CAE tools for schematic-entry and gate-level simulation. Once your design has been created and verified, Designer completes the design with place-and-route, timing analysis, and back annotation for timing verification.

HDL Synthesis-Based Design Flow Overview

The Actel HDL synthesis-based design flow has four main steps: design creation/verification, design implementation, programming, and system verification. These steps are described in the following sections.

Third-party software users can also refer to the Actel *Mentor Graphics Interface Guide, VHDL Vital Simulation Guide, Verilog Simulation Guid*e, and the *Mentor Graphics eProduct Designer Interface Guide* for information about using these tools with Actel software and devices.

Design Creation/Verification

During design creation/verification, a design is captured in an RTL-level (behavioral) HDL source file. After capturing the design, behavioral simulation of the HDL file can be performed with third-party simulation software to verify that the HDL code is correct. The code is then synthesized into a structural HDL netlist. After synthesis, structural simulation of the design can be performed. Finally, an EDIF netlist is generated for use in Designer and configured with the structural netlist for back-annotation simulation with third-party simulation software.

HDL Design Source Entry

Enter your design source using a text editor or a context-sensitive HDL editor. Your HDL design source can contain RTL-level constructs as well as instantiations of structural elements, such as ACTgen macros. Refer to the *Actel HDL Coding Style Guide* for additional information about writing HDL code for Actel designs.

Functional Simulation

Perform a functional simulation of your design before synthesis. Functional simulation verifies the functionality of your HDL code. Typically, unit delays are used and a standard HDL test bench can be used to drive simulation. Refer to the Actel Interface Guides and the documentation included with your simulation tool for information performing Functional simulation.

Synthesis

After you have created your functional HDL source file, you must synthesize it using a third-party synthesis tool before you place-and-route it in Designer. Synthesis transforms the behavioral HDL file into a gate-level netlist and optimizes the design for a target technology. Refer to the documentation included with your synthesis tool for information about performing design synthesis.

Netlist Generation

After you have created, synthesized, and verified your design, you can place-and-route in Designer using an EDIF, Verilog, or VHDL netlist. This netlist is also used to generate a structural HDL

netlist. Refer to the documentation included with your synthesis tool for information about generating an EDIF, Verilog, or VHDL netlist.

Post-Synthesis (Functional) Simulation

Generate a structural HDL netlist from your EDIF/Verilog/VHDL netlist for use in structural and timing simulation by exporting it from Designer. Refer to the documentation included with your synthesis tool for information about generating a structural netlist.

Structural Simulation

Perform a structural simulation with a third-party simulation tool before placing and routing it. Structural simulation verifies the functionality of your post-synthesis structural HDL netlist. Unit delays included in the compiled Actel libraries are used for every gate. Refer to the Actel Interface Guides, the Actel Simulation Guides, and the documentation included with your simulation tool for information about performing structural simulation.

Design Implementation

Design Implementation for synthesis based designs is identical to that of the schematic based designs. Refer to "Design Implementation" on page 16 for more information.

HDL Synthesis-Based Design Methodology

If you prefer a high-level design methodology, Designer allows you to move from design description to a programmed part. To get you through the design phase, Actel supports Verilog and VHDL synthesis tools, as well as behavioral simulation. Once the design is synthesized, Designer helps you complete the design with place-and-route, timing analysis, and timing verification.

.

3 Quick Start Tutorial

The design is targeted at the Actel ProASIC3E family. To show the design in its simplest form, a simple counter design is created in Actel's Libero IDE. The steps involved are:

Step 1 – Create a New Project

Step 2 – Perform Pre-synthesis Simulation

Step 3 – Synthesize the Design in Synplify

Step 4 – Perform Post-Synthesis Simulation

Step 5 – Implement the Design with Designer

Step 6 – Perform Timing Simulation with Back-Annotated Timing

Step 1 – Create a New Project

This step uses the Libero IDE HDL Editor to enter an Actel VHDL design.

To create the VHDL project:

1. Double-click the **Libero IDE** icon on your desktop to start the program.
2. From the **File** menu, select **New Project**. This displays the **New Project Wizard**, as shown in Figure 3-1.

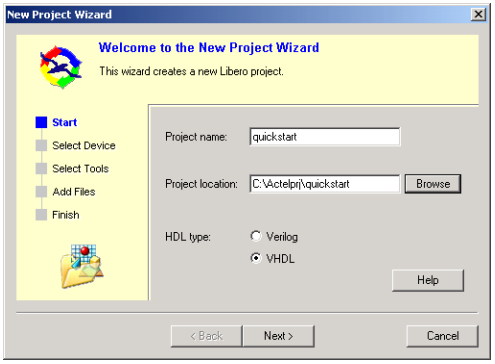


Figure 3-1. New Project Wizard in Libero IDE

3. Enter your **Project name**. For this tutorial, name your project *quickstart*. Select your HDL type.
4. If necessary, in the **Project** location field, click **Browse** to navigate to **C:\Actelprj**. Click **Next** to continue.

5. Select your project **Family**, **Die**, and **Package**. For this tutorial, select **ProASIC3E**, the **A3PE600** die, and **208 PQFP** for your package (Figure 3-2).

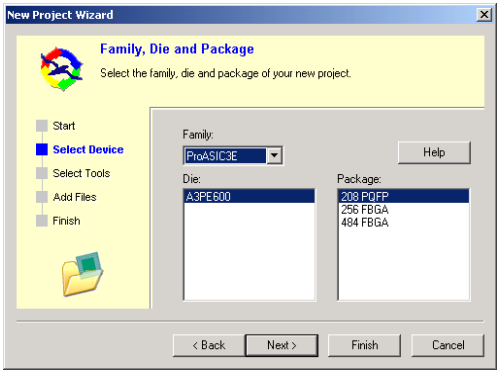


Figure 3-2. Select Family, Die, and Package

6. Click **Next** to select **Integrated Tools** in the **New Project Wizard** (Figure 3-3).

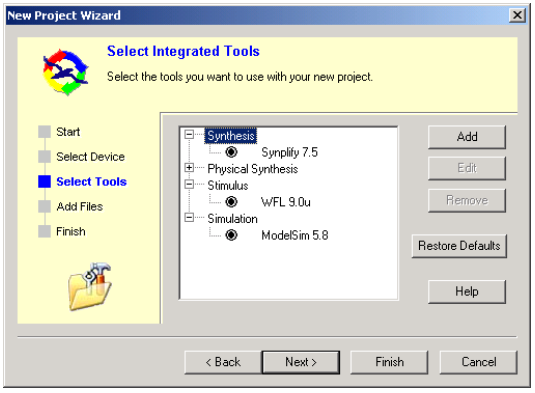


Figure 3-3. Selected Integrated Tools in New Project Wizard (Libero IDE)

7. Click the **Restore Defaults** button to use the default tools included with Libero IDE. Click the **Add** button to add a different Synthesis, Simulation, or Stimulus tool. If you wish to **Add** a tool, Libero IDE opens the **Add Profile** dialog box (Figure 3-4).

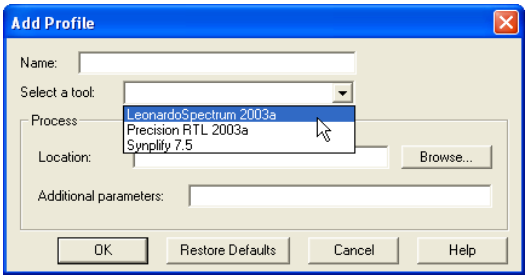


Figure 3-4. Add Profile Dialog Box in Libero IDE

Name, **Select** (from the list of Libero IDE supported tools), and **Browse** to the **Location** of your tool. Click **OK** to return to the New Project Wizard.

After you have selected your tools, click **Next** to continue.

8. **Add Files** in the **New Project Wizard** to add existing Project Design Files, including any ACTgen cores or Block Symbol, Schematic, VHDL Package, HDL, Implementation, and Stimulus files (Figure 3-5).

Figure 3-5. Add Files in the New Project Wizard (Libero IDE)

Select the file type and click **Add File**. **Browse** to your file, and click **Add**. Add as many files as you wish in this way. Click **Next** to continue.

9. Review your project information. Click **Finish** to close the Wizard and create your new project (Figure 3-6). Click **Back** to return to any step of the Wizard and correct information in your project.

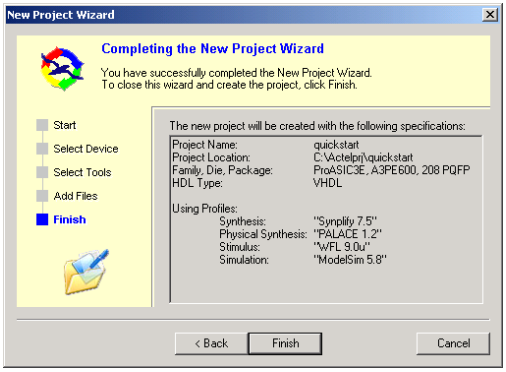


Figure 3-6. Summary in New Project Wizard (Libero IDE)

Your Libero project exists, but you must add code or source to the project, such as a schematic, an ACTgen core, or a VHDL entity or package file, before you can run synthesis.

Add HDL to Your Project

1. From the **File** menu, click **New**. This opens the **New** dialog box, as shown in Figure 3-7.

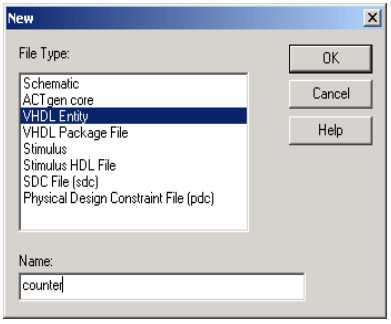


Figure 3-7. New File Dialog Box

2. Select **VHDL Entity** in the File Type field, enter **counter** in the **Name** field and click **OK**. The HDL Editor opens. Enter the following VHDL file, or if this document is open in an electronic form, cut and paste it from here.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
entity counter is
  port(Clock : in std_logic;
       Q : out std_logic_vector(1 downto 0);
       Aclr : in std_logic);
end counter;
architecture behavioral of counter is
    signal Qaux : UNSIGNED(1 downto 0);
begin
  process(Clock, Aclr)
  begin
    if (Aclr = '1') then
      Qaux <= (others => '0');
    elsif (Clock'event and Clock = '1') then
      Qaux <= Qaux + 1;
    end if;
  end process;
  Q <= std_logic_vector(Qaux);
end behavioral;
```

3. From the **File** menu, click **Save**. The design file counter appears in the **Design Hierarchy**. Libero IDE lists **counter.vhd** under HDL files in the File Manager, as shown in Figure 3-8.

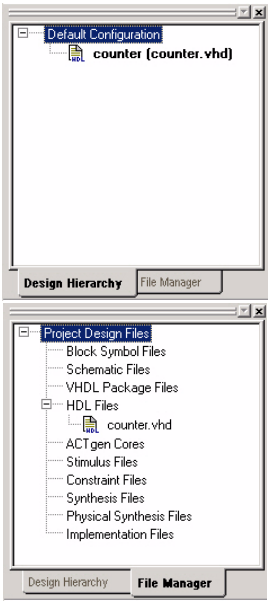


Figure 3-8. Design Hierarchy and File Manager Tabs

4. Check the HDL in the file before you continue. In the **Design Hierarchy** or **File Manager** tab, right-click **counter.vhd** and select **Check HDL** file. This checks the syntax of the counter.vhd file. Before moving to the next section, please modify the code if you find any errors.

Step 2 – Perform Pre-synthesis Simulation

The next step is simulating the RTL description of the design. First, use WaveFormer Lite to create a stimulus for the design and then generate a testbench for the design.

Creating Stimulus Using WaveFormer Lite

WaveFormer Lite generates VHDL testbenches from drawn waveforms. There are three basic steps for creating testbenches using WaveFormer Lite and the Actel Libero IDE software:

1. Import Signal Information
2. Drawing Waveforms

3. Export the Testbench

Import Signal Information

To launch WaveFormer Lite and import signal information:

Right-click the **counter.vhd** file in the **Design Hierarchy** tab and select **Create Stimulus > HDL Source files (source files)**. WaveFormer Lite launches with the port signals in the **Diagram** window, as shown in Figure 3-9.

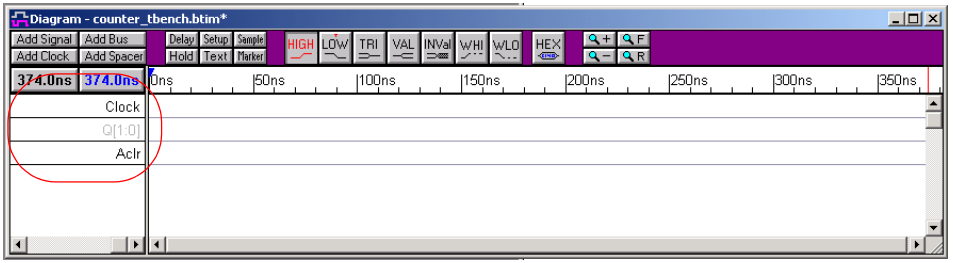


Figure 3-9. WaveFormer Lite Timing Diagram Window

The counter design contains the following signals:

- Clock Input signal
- Q[1:0] Output signal

Drawing Waveforms

The state buttons are the buttons with the waveforms drawn on their face: HIGH, LOW, TRIstate, VALid, INValid, WHI weak high, and WLO weak low, as shown in Figure 3-10.

Figure 3-10. State Buttons

When a state button is activated, it is pushed in and colored red. The active state is the type of waveform that is drawn next. Click a state button to activate it.

The state buttons automatically toggle between the two most recently activated states. The state with the small red "T" above the name will be the toggle state. The initial activated state is HIGH and the initial toggle state is LOW.

Signal edges are automatically aligned to the closest edge grid when signals are drawn using the mouse. Control the edge grid from the **Options > Grid Settings** menu item.

To draw a waveform:

1. Select the **High** state and place the mouse cursor inside the Diagram window at the same vertical row as the signal name.
2. Click the mouse button. This draws a waveform from the end of the signal to the mouse cursor. The red state button on the button bar determines the type of waveform drawn. The cursor shape also mirrors the red state button.
3. Move the mouse to the right and click again to draw another segment.

To copy waveforms:

It is possible to copy and paste sections of waveforms onto (overwrite) or into (insert) any signal in the diagram. To copy and paste waveform sections:

1. Select the names of the required signals. If no signals are selected, the **Block Copy** command selects all the signals in the diagram.
2. Select the **Edit > Block Copy Waveforms** menu option (choose to Select all waveforms, if necessary). This opens the **Block Copy Waveforms** dialog box with the selected signals displayed in the **Change Waveform Destination** list box.

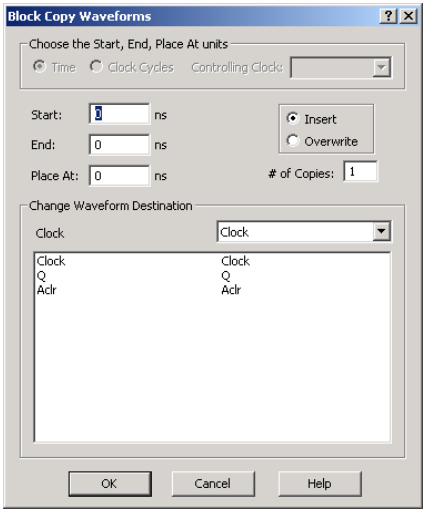


Figure 3-11. Block Copy Waveforms Dialog Box

3. Enter the values that define the copy and paste.

   Select either **Time** or **Clock** cycle for the base units of the dialog. Remember:

   - When copying only signals (no clocks), time is the default base unit of the dialog.
   - When copying part of a clock, it is best to choose a clock cycles base unit and choose the copied clock as the reference clock.
   - If you select time when copying clocks, the (end_time - start_time) must equal an integral number of clock periods, and the place_at time must be at the same clock period offset as the start_time.
   - **Start** and **End** define the times of the block copy.
   - **Place At** is the time at which the block will be pasted.
   - The **Insert** and **Overwrite** radio buttons determine whether the paste block is inserted into the existing waveforms or overwrites those waveforms.
   - The list box at the bottom of the dialog determines which signal the copied waveforms will be pasted into.

To change this mapping:

- Select a line in the list box.

  This places the destination signal in the drop-down list box on top of the list box.

- Select another signal from the drop-down list box.

  Each destination signal can be used only once per copy.

- Click **OK** to complete the copy and paste operation.

Export the Testbench

In this step you create a stimulus file for the design and generate a testbench using WaveFormer Lite. After exporting the testbench, perform a pre-synthesis simulation using ModelSim.

To create a stimulus file and generate a VHDL testbench:

In this step, a design stimulus file is created using WaveFormer Lite. Following the instructions in the previous sections and define a 20 MHz value for the Clock.

1. Right-click the **Clock** signal to select it. Select **Signal(s) <-> Clock(s)** to create the clock signal (Figure 3-12).

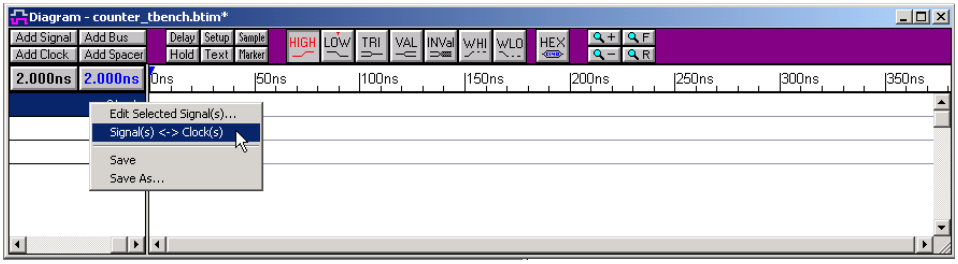


Figure 3-12. Create Clock Signal in WaveFormer Lite

2. Double-click any clock segment to edit the clock parameters (Figure 3-13). Set a frequency of **20 MHz**.

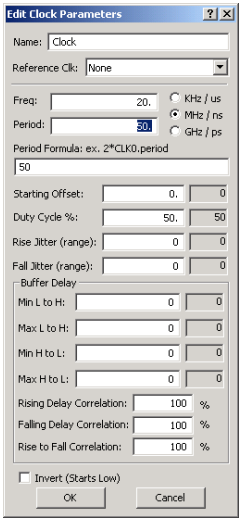


Figure 3-13. Edit Clock Parameters in WaveFormer Lite

This creates the waveform shown in Figure 3-14.

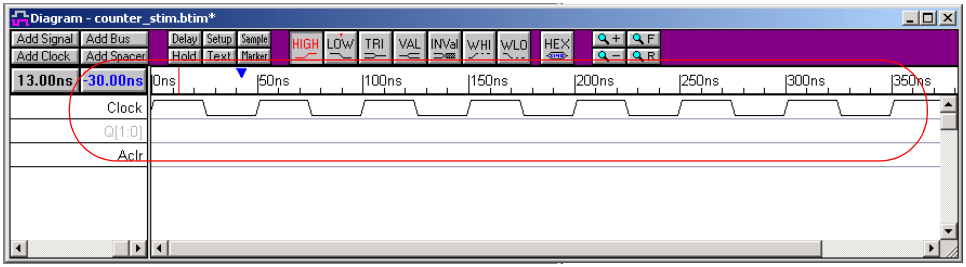


Figure 3-14. WaveForm Timing diagram

3. Click the **HIGH** state button before you start to draw the Aclr waveform (if you have not selected any other state buttons since you drew the clock waveform, you may continue).
4. Click and drag to in the **Aclr** signal to draw a high segment from **0** to **50** ns. When you release the mouse button, WaveFormer Lite switches your state to **LOW**.
5. Click and drag in the **Aclr** signal to draw a low segment from **50** ns to **350** ns (Figure 3-15).

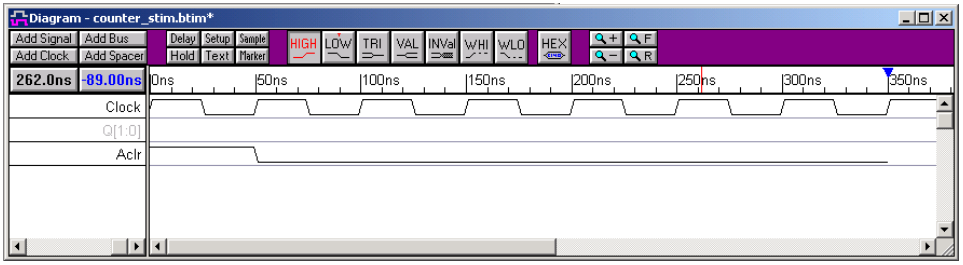


Figure 3-15. counter_stim.btim File with Clock and Aclr Signals

6. After creating the waveforms, select **Save As** from the **File** menu. In the **Save As** dialog box, enter **counter_stim.btim** as the file name and click **Save**.
7. After saving the timing diagram file, select **Export Timing Diagram As** from the **Export** menu.

8. Select **VHDL w/ Top Level Testbench** in **Files of Type** and enter **counter_tbench.vhd** for the file name, as shown in Figure 3-16.

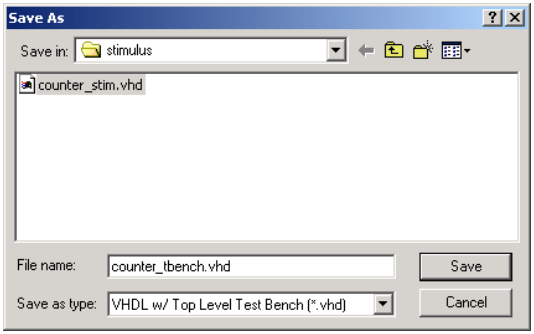


Figure 3-16. Export VHDL Testbench Save As Dialog Box

The WaveFormer Lite Report window displays the VHDL testbench with a component declaration and instantiation inside.

9. Exit WaveFormer Lite (**File > Exit**). The File Manager displays the stimulus files. The design is ready to simulate under ModelSim.

Alternatively, you can create a testbench using the HDL editor

To create a testbench using the HDL editor:

1. From the **File** menu, select **New**. This opens the New File dialog box.
2. Select **Stimulus HDL file** from the **File Type** list, enter **counter_stim** for the name, and click **OK**. The file opens in the HDL Editor.
3. Create the VHDL testbench and save it.

Pre-Synthesis Simulation

Once you generate a testbench, use ModelSim to perform a pre-synthesis simulation.

To perform a pre-synthesis simulation:

1. Right-click counter in the **Design Hierarchy** tab and choose **Select Stimulus**, as shown in Figure 3-17.

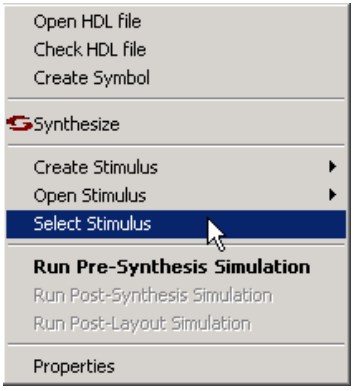


Figure 3-17. Selecting a Stimulus File

The Select Stimulus dialog box appears (Figure 3-18).

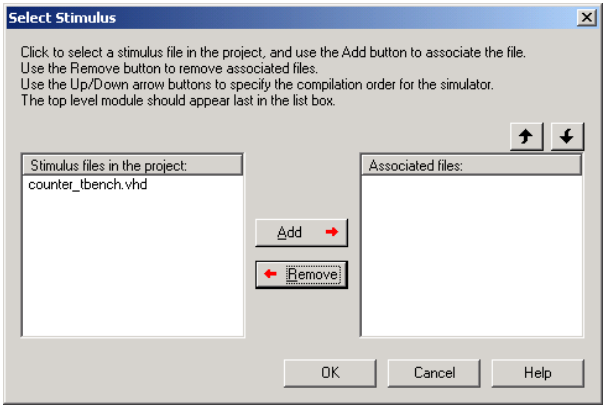


Figure 3-18. Select Stimulus Dialog Box

2. Select **counter_tbench.vhd** in the **Project List** box and click **Add** to add the file to the **Associated Files** list.

3. Click **OK**. A check mark appears next to WaveFormer Lite in the Process window to notify you that there is a testbench file associated, as shown in Figure 3-19.

   Stimulus icons in the Design flow window turn green to notify you that there is a testbench file associated with them.

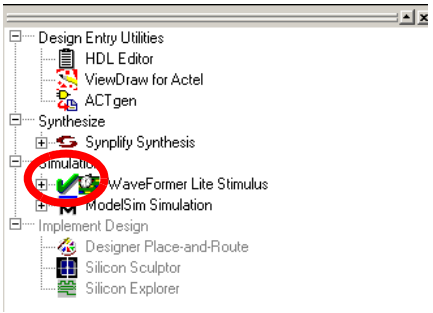


Figure 3-19. Check Mark in Waveformer Lite

4. Double-click the **Model*Sim*** simulation icon in the **Process** window, or right-click counter in the **Design Hierarchy** tab and select **Run Pre-synthesis Simulation**, as shown in Figure 3-20.

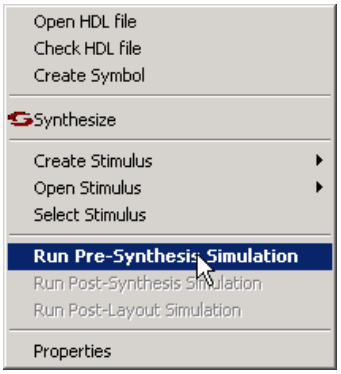


Figure 3-20. Run Pre-Synthesis Simulation

The ModelSim VHDL simulator opens and compiles the source files, as shown in Figure 3-21.

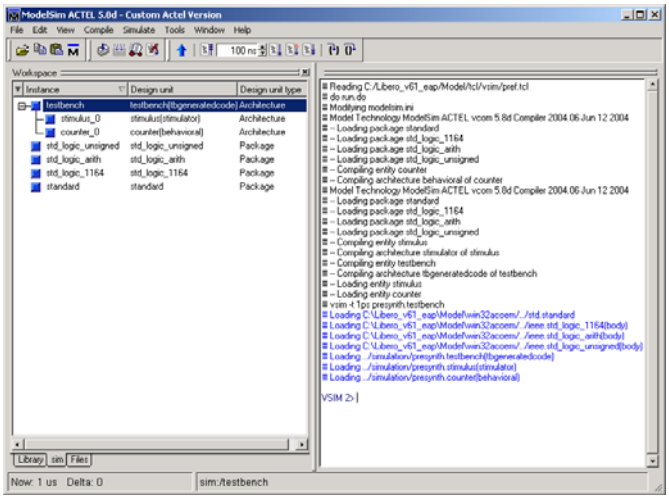


Figure 3-21. ModelSim Main Window

Once the compilation completes, the simulator simulates for the default time period of 1000 ns and a **Wave** window, shown in Figure 3-22, opens to display the simulation results. Scroll in the **Wave** window to verify that the design functions properly.

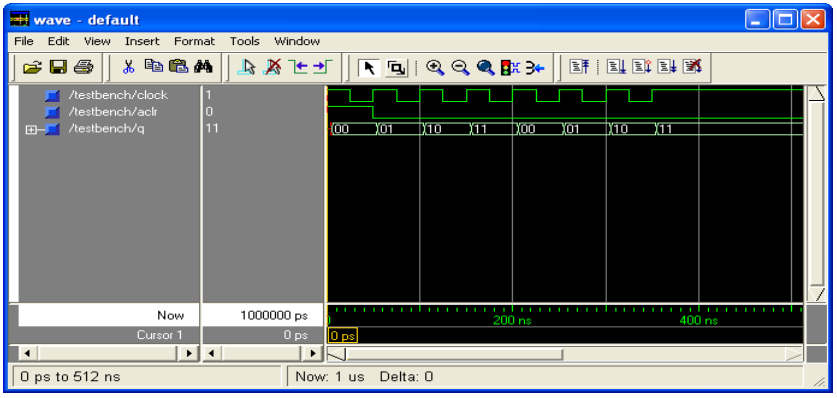


Figure 3-22. ModelSim Wave Window

5. In the ModelSim window, select **File** > **Quit** to close the window.

Step 3 – Synthesize the Design in Synplify

The next step is generating an EDIF netlist by synthesizing the design in Synplify. For HDL designs, Libero IDE launches and loads Synplicity's Synplify synthesizer with the appropriate design files.

To create an EDIF netlist for the design using Synplify:

1. In the Libero IDE, double-click the **Synplify Synthesis** icon in the Libero IDE process window or right-click the **counter.vhd** file in the Design Hierarchy and select **Synthesize**. This launches the Synplify synthesis tool with the appropriate design files, as shown in Figure 3-23.

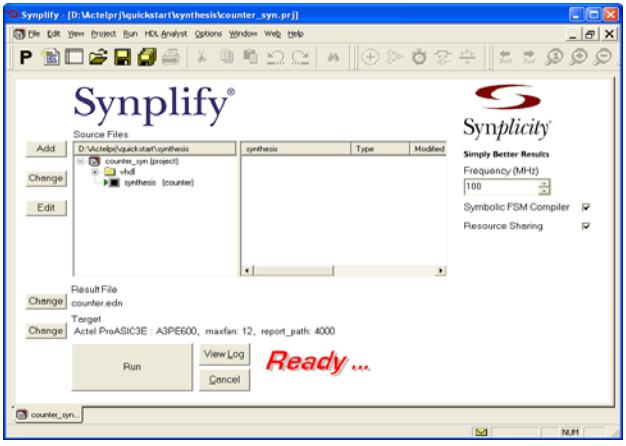


Figure 3-23. Synplify

2. From the **Project** menu, select **Implementation Options**. This displays the **Options for Implementation** dialog box, as shown in Figure 3-24.

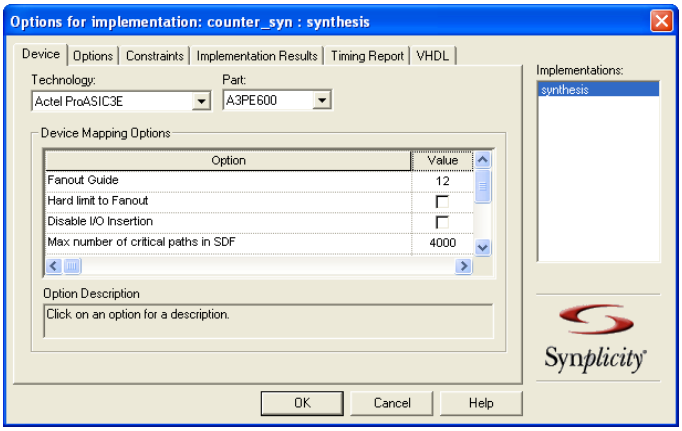


Figure 3-24. Options for Implementation Dialog Box

3. Set the following in the dialog box:
   - **Technology:** Actel ProASIC3E (Set by Libero IDE)
   - **Part:** A3PE600
   - **Fanout Guide:** 12 (Default)
   - **Hard Limit to Fanout:** Off (Default). This refers to the fanout limit.

   Accept the default values for each of the other tabs in the Options for Implementation dialog box and click **OK**.

4. In the Synplify main window, click **Run**. Synplify compiles and synthesizes the design into a netlist called **counter.edn**. The resulting **counter.edn** file is then automatically translated by Libero into a VHDL netlist called counter.vhd.

   The resulting EDIF and VHDL files are displayed under **Implementation Files** in the **File Manager**.

   Note: If any errors appear after you click the **Run** button, edit the file using the Synplify editor. To edit the file, double-click the file name in the Synplify window. Any changes made here are saved to the original design file in Libero IDE.

5. **Save** and close Synplify. From the **File** menu, click **Exit** to close Synplify. Click **Yes** to save any settings made to the counter.prj in Synplify.

Step 4 – Perform Post-Synthesis Simulation

The next step is simulating the VHDL netlist of the counter using the VHDL testbench created in "To create a stimulus file and generate a VHDL testbench:" on page 29.

1. Click the **ModelSim Simulation** icon in the Libero IDE Process window, or right-click the **counter.edn file** in the **Design Hierarchy** and select **Run Post-Synthesis Simulation**. This launches the ModelSim Simulator that compiles the source file and testbench.

   Once the compilation completes, the simulator runs for 1000 ns and a Wave window opens to display the simulation results.

2. Scroll in the Wave window to verify that the counter works correctly. Use the zoom buttons to zoom in and out as necessary.

Step 5 – Implement the Design with Designer

After creating and testing the design, the next phase is implementing the Design using the Actel Designer software.

1. Double-click Designer **Place-and-Route** in the Libero IDE Process Window, or right-click **counter.edn** in the **Design Hierarchy** and select **Run Designer.** Designer reads in the design file and opens the Device Selection Wizard.

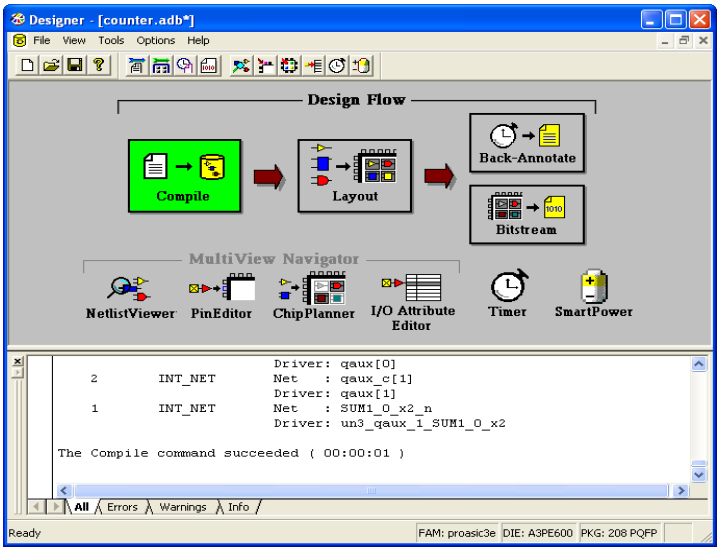
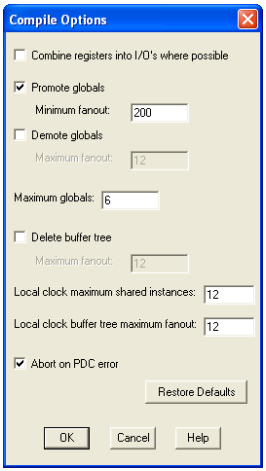


Figure 3-25. Designer

Select **A3PE600** in the Die field and select **208 PQFP** in the Package field. Accept the default Speed grade and Die Voltage and click **Next**.

Use the default I/O settings and click **Finish**.

Click the **Compile** icon. Leave the default Compile settings and click **OK**.



Designer compiles the design and shows the utilization of the selected device. Also, note that the **Compile** icon in Designer turns green, indicating that the compile has successfully completed.

2. Once the design compiles successfully, use the I/O Attribute Editor tool to assign the pin for subsequent place-and-route runs. Click the I/O Attribute Editor to open the tool (it opens in

the MultiView Navigator user interface, as in Figure 3-26). See the Libero or Designer online help for more information on the I/O Attribute Editor or MultiView Navigator.

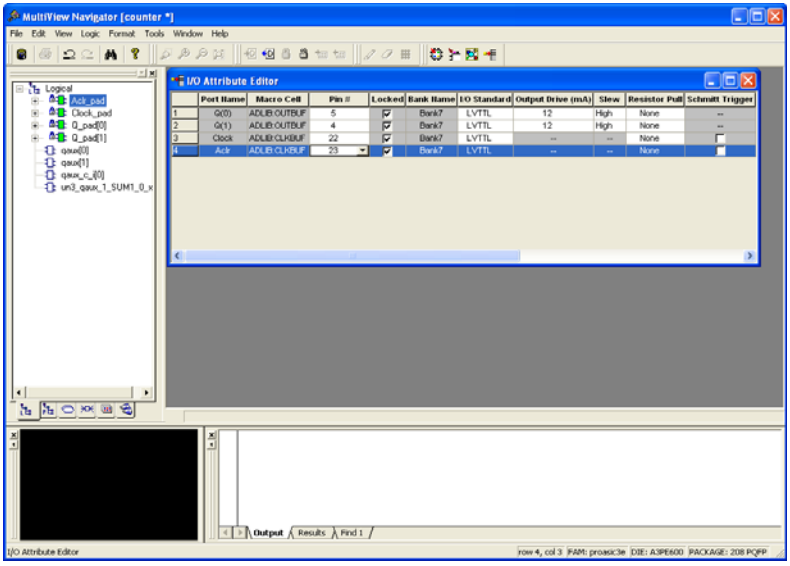


Figure 3-26. I/O Attribute Editor in MultiView Navigator

Assign a pin number to all of the signals, then select **Commit** from the **File** menu and close the I/O Attribute Editor.

3. (Optional) After successfully compiling the design, use the Designer Tools to view pre-layout static timing analysis with **Timer**, set timing constraints in Timer, analyze static and dynamic power with **SmartPower**, and use **ChipPlanner** to assign modules. Click the appropriate icon to access these tools.

   For more information on these functions, refer to the Designer or Libero online help.

4. In **Designer,** click **Layout**. This opens the Layout Options dialog box shown in Figure 3-27.

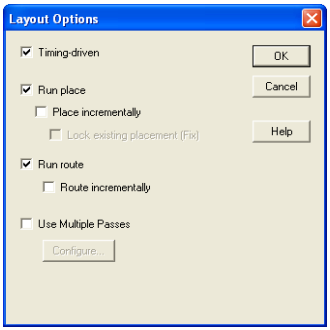


Figure 3-27. Layout Options Dialog Box

Click **OK** to accept the default layout options. This runs the place-and-route on the design. The Layout icon turns green to indicate that the layout has successfully completed.

5. From Designer, click **Back-Annotate** in the **Design Flow** window. This opens the **Back-Annotate** dialog box shown in Figure 3-28.

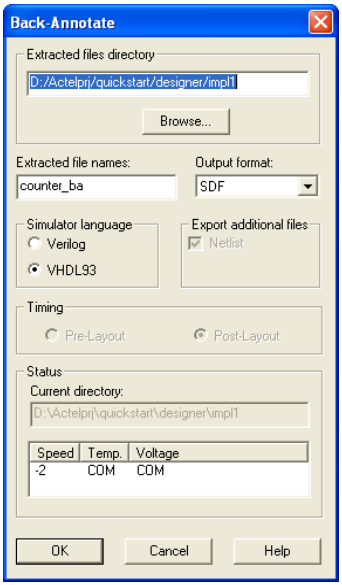


Figure 3-28. Back-Annotate Dialog Box

Accept the default settings and click **OK**. The **Back-Annotate** icon turns green.

6. Save and close Designer. From the **File** menu, click **Exit**. Click **Yes** to save the design before closing designer. Designer saves all the design information in a *.adb file.

   The file counter.adb appears under the Designer Files of the File Manager. To reopen the file, right-click the file and select **Open in Designer**.

Step 6 – Perform Timing Simulation with Back-Annotated Timing

After completing the place-and-route and back annotation of the design, perform a timing simulation with the ModelSim HDL simulator.

To perform a timing simulation:

1. Click the **ModelSim Simulation** icon in the Libero IDE Process window, or right-click the counter file in the Design Hierarchy and select **Run Post-Layout Simulation**.

   This launches the ModelSim Simulator that compiles the back annotated VHDL netlist file and testbench. Once the compilation completes, the simulator runs for 1000 ns and a Wave window opens to display the simulation results.

2. Scroll in the Wave window to verify that the counter works correctly. Use the zoom buttons to zoom in and out as necessary.

4 Design Considerations

This chapter contains information and procedures to assist you in creating Actel designs.

Naming Conventions

This section lists schematic, Verilog, and VHDL naming conventions that should be followed to avoid potential design flow problems.

Schematic

Use only alphanumeric and underscore "_" characters for schematic net and instance names. Do not use asterisks, forward slashes, backward slashes, spaces, or special characters ($, #, @, !, ( ), *, &, %, etc.). Refer to the Libero online help for more information on schematic naming conventions.

VHDL

If simulation is to be completed using a VHDL simulator, it is important to create schematics or write HDL code that complies with the VHDL naming conventions. The following naming conventions apply to VHDL designs:

- VHDL is not case sensitive.
- Two dashes "--" are used to begin comment lines.
- Names can use alphanumeric characters and the underscore "_" character.
- Names must begin with an alphabetic letter.
- Do not use two underscores in a row, or use an underscore as the last character in the name.
- Spaces are not allowed within names.
- Object names must be unique. For example, you cannot have a signal named A and a bus named A(7 downto 0).

VHDL Keywords

The following is a list of the VHDL reserved keywords:

abs	downto	library	postponed	subtype
access	else	linkage	procedure	then
after	elsif	literal	process	to

alias	end	loop	pure	transport
all	entity	map	range	type
and	exit	mod	record	unaffected
architecture	file	nand	register	units
array	for	new	reject	until
assert	function	next	rem	use
attribute	generate	nor	report	variable
begin	generic	not	return	wait
block	group	null	rol	when
body	guarded	of	ror	while
buffer	if	on	select	with
bus	impure	open	severity	xnor
case	in	or	shared	xor
component	inertial	others	signal	
configuration	inout	out	sla	
constant	is	package	sra	
disconnect	label	port	srl	

Verilog

If simulation is to be completed using a Verilog simulator, it is important to create schematics or write HDL code that complies with the Verilog naming conventions. The following naming conventions apply to Verilog HDL designs:

- Verilog is case sensitive.
- Two slashes "//" are used to begin single line comments. A slash and asterisk "/*" are used to begin a multiple line comment and an asterisk and slash "*/" are used to end a multiple line comment.
- Names can use alphanumeric characters, the underscore "_" character, and the dollar "$" character.

- Names must begin with an alphabetic letter or the underscore.
- Spaces are not allowed within names.

Verilog Keywords

The following is a list of Verilog reserved keywords:

always	endfunction	macromodule	realtime	tran
and	endmodule	medium	reg	tranif0
assign	endprimitive	module	release	tranif1
attribute	endspecify	nand	repeat	tri
begin	endtable	negedge	rnmos	tri0
buf	endtask	nmos	rpmos	tri1
bufif0	event	nor	rtran	triand
bufif1	for	not	rtranif0	trior
case	force	notif0	rtranif1	trireg
casex	forever	notif1	scalared	unsigned
casez	fork	or	signed	vectored
cmos	function	output	small	wait
const	highz0	parameter	specify	wand
deassign	highz1	pmos	specparam	weak0
default	if	posedge	strength	weak1
defparam	ifnone	primitive	strong0	while
disable	initial	pull0	strong1	wire
edge	inout	pull1	supply0	wor
else	input	pulldown	supply1	xnor
end	integer	pullup	table	xor

endattribute	join	remos	task
endcase	large	real	time

Hierarchical Designs

Multiple-level or hierarchical designs are created by creating symbolic representations of blocks and adding them to other levels. The Designer software reads and writes hierarchical netlists.

Multiple Sheet Designs

The Designer software supports multiple sheet designs. Each sheet in the design is considered as part of a schematic, and it is not considered as a level of hierarchy. Most schematic capture tools have page connectors to connect the sheets. Refer to the documentation provided with your schematic capture tool for additional information.

Actel Libraries

Actel provides libraries to support schematic- and synthesis-based designs. Logic can be described in behavioral languages (VHDL or Verilog). Structured logic such as counters, adders, and comparators can be automatically built using the ACTgen Macro Builder. Functions can be exclusively behavioral, schematic, or a combination of the two. Timing definition is supported by the timing driven layout tools. I/O definitions can be described in the design source or in Designer.

Adding Power and Ground

Actel provides special VCC and GND macros to connect nets to power and ground. Add the VCC and GND macros to your design and connect them by nets to the functional logic making up the design. It is important to use the symbols provided by Actel in order to prevent design flow issues. Do not add power and ground to designs by naming nets or adding third-party power and ground symbols. Refer to the Actel Interface Guides or the Libero IDE online help for information about adding power and ground to your design.

Adding a Global Network

The Actel architecture provides global networks that allow high fanout drive for flip-flops and latches with minimal skew. The available global networks are shown in Table 4-1.

Table 4-1. Global Network Attributes

Input Pad Name	Type	Family	#	Internal Drive Option	Macro	Note
CLK	routed	MX	1	Yes (CLKBIBUF only)	CLKBUF; CLKBIBUF	Can adjust skew with clock balancing
CLKA CLKB	routed	DX; MX; SX; SX-A; RTSX-S; eX	2	Yes[a]	CLKBIBUF; CLKBIBUFI; CLKBUF; CLKINT; CLKBUFI[b]; CLKINTI<Superscript>b	Can connect to CLK and G (gate) pins
CLKA CLKB CLKC CLKD	routed	Axcelerator	4	yes	CLKBUF; CLKINT	Can connect to CLK and G (gate) pins
HCLK	dedicated	ACT 3; SX; SX-A; RTSX-S; eX	1	No	HCLKBUF	Directly hard-wired to certain S-modules[c]
HCLKA HCLKB HCLKC HCLKD	dedicated	Axcelerator	4	No	HCLKBUF; HCLKINT	Directly hard-wired to certain S-modules[c]
QCLK A-D	routed	DX[d]; MX; SX72A; RTSX72S	4	Yes	QCLKBUF; QCLKBUFI<Superscript>b; QCLKBIBUFI<Superscript>b; QCLKBIBUF<Superscript>b; QCLKINT; QCLKINTI<Superscript>b	Each QCLK drives a quadrant of the device

a. The Internal Drive Option for DX, MX, SX and SX-A can only be utilized by selecting the CLKINT macro to drive an internal clock network.

b. SX72A only.

c. HCLK is hardwired to all S-modules. Only the sequential macros made of C-modules cannot be driven by HCLK.

d. Not available in 32140DX, 3265DX

Routed Clocks

Routed clocks are clock networks with unlimited fanout and offer clock speeds that are independent of the number of logic modules being driven. Routed clocks can be used as RESET and PRESET networks to drive the reset and preset pins of internal logic modules. They can also be connected to most logic module inputs.

CLK, CLKA, CLKB

CLK, CLKA, CLKB, CLKC[1], and CLKD[1] are routed global clocks that can be used by selecting the CLKBUF, CLKBIBUF, CLKBIBUFI, CLKINT, or CLKINTI macro.

QCLK

QCLK (available on 42MX, 24MX36, 42MX36, 3200DX, and 54SX72A devices) is a routed quadrant or global clock that can drive from one to four quadrants on a device. In addition to global RESET and PRESET networks, QCLK can be used as a quadrant RESET and PRESET network. *QCLK* can be used by selecting the QCLK, QCLKINT, QCLKINTI, QCLBUF, QCLBUFI, QCLBIBUF, QCLBIBUFI macro.

Note: QCLK is not available for Axcelerator.

Dedicated Clocks

Dedicated clocks are clock networks that are directly wired to either sequential or I/O modules. They contain no programming elements in the path from the I/O pad driver to the input of sequential or I/O modules. These clocks provide sub-nanosecond skew and guaranteed performance.

HCLK

HCLK (HCLKA, HCLKB, HCLKC, and HCLKD for Axcelerator devices) is a dedicated hard-wired clock input for sequential modules. HCLK is directly wired to each sequential module and offers clock speeds independent of the number of sequential modules being driven. HCLK can be used by selecting the HCLKBUF macro. Table 4-2 lists the ACT 3 sequential elements and Table

1. Axcelerator only.

4-3 lists the SX/SX-A/eX sequential elements that cannot be connected to HCKLBUF because they are built from combinatorial modules.

Table 4-2. ACT 3 Sequential Elements that Cannot Connect to HCKLBUF

DFP1	DFP1B	DFP1D	DFPCA	DFPC	DLC1
DLC1A	DLC1F	DLC1G	DLE2C	DLE3B	DLE3C
DLP2C	DLP1A	DLP1B	DLP1C	DFP1A	

Table 4-3. 54SX/54SX-A/eX Sequential Elements that Cannot Connect to HCKLBUF

DLC1	DLC1A	DLC1F	DLC1G	DLE2C	DLE3B
DLE3C	DLP2C	DLP1A	DLP1B	DLP1C	DFP1A

IOCLK[1]

IOCLK is a dedicated hard-wired clock input for I/O modules. IOCLK is directly wired to each I/O module register and offers speeds independent of the number of I/O modules being driven. IOCLK can be used by selecting the IOCLKBUF macro.

Special Clocks

Special clocks are special hard-wired networks for I/O modules that can only drive preset/clear pins of I/O modules. IOPCL1, the only special clock, is a special network directly wired to the preset and clear inputs of all I/O registers. IOPCL functions as an I/O when no I/O preset or clear macros are used. IOPCL can be used by selecting the IOPCLBUF macro.

PLL

PLLs (Phase-Locked Loops) are dedicated clock resources that are available for the ProASIC3/E, ProASIC^{PLUS} (two PLLs), and Axcelerator families. Each Axcelerator device has eight PLLs; four PLLs may be used on the HCLK network and four PLLs may be used on the routed clock network.

Use PLLs to reduce the skew on the clock network. PLLs can accept feedback from internal and external clock sources, and used to multiply, divide, and phase-shift clocks internally. You can also cascade your PLLs to obtain a specific clock frequency.

1. IOCLK and IOPCL are only available in the DX/42MX family.

Using Flash Global Routing Resources

Use the automatic global resource assignments specified by Designer. These lines exhibit very low skew. When Designer imports the netlist and device data, it can automatically assign unused global resources to the nets with the highest fanouts (for example, clock and reset signals).

If high fanout signals mapped by Designer to global resources are buffered, Designer automatically removes these buffers. Flash devices provide automatic buffering for global resources.

Signals can also be assigned to the global resources by using Designer global primitives in netlists, and by specifying constraint files for Designer.

The checker contains a description of the highest fanout nets in a design and the results of automatic global assignments. If your constraints appear not to have been honored, refer to this report file.

Combinability

The functionality of some combinations of gates and flip-flops can be combined to fit into a single logic module instead of a logic module to implement each gate or flip-flop. This ability is called combinability. Designer has an automatic utility called the Combiner to perform this function, as well as to reduce the logic in other ways.

The Combiner reduces the number of logic modules, logic levels, and fan-ins in a design by remapping, removing, and combining certain hard macros, including the deletion of buffers on clock networks. It also simplifies a design netlist using features of the Actel architecture.

The Combiner is integrated into the Compile function in Designer. It improves the density, speed, and routability of a design by performing the following functions:

- Combinatorial Module Reduction
- Sequential Remapping
- Unused Logic Removal
- Constant Input Reduction
- Fan-in Reduction
- I/O-Register Combining in Axcelerator (see the Axcelerator datasheet at http://www.actel.com/documents/axds.pdf for more information)
- I/O-FIFO Combining in Axcelerator (see the Axcelerator datasheet for more information)

Combinatorial Module Reduction

Combinatorial Module Reduction reduces the number of combinatorial modules and logic levels, giving a design more density and speed. It does this in one of the following two ways:

1. It combines two or more combinatorial macros into a single macro. For example, two 2-input AND gates, two inverters, and three inverters are combined into a single 3-input AND gate, a buffer, and an inverter respectively, shown in Figure 4-1.

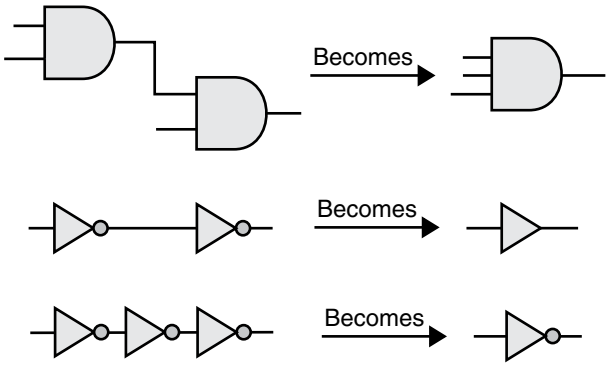


Figure 4-1. Combined Combinatorial Macros

2. It combines a combinatorial macro and a sequential macro into a single sequential macro. For example, a combinatorial macro “MUX” and a sequential macro “DF1” are combined into a single sequential macro “DFM,” shown in Figure 4-2. Module Reduction is not available for 54SX/54SX-A or eX devices. Instead, an automatic DirectConnect between a combinatorial module and a sequential module is used.

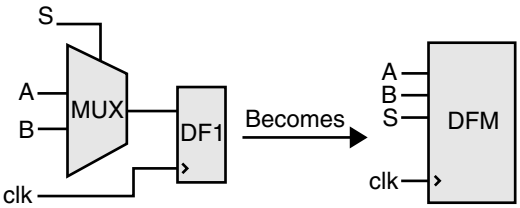


Figure 4-2. Combined Combinatorial and Sequential Macros

Sequential Remapping

Sequential Remapping attempts to achieve better results in the density, speed and routability of a design. If Sequential Remapping cannot reduce the total number of combinatorial macros in the

design, the Combiner does not use it. Sequential Remapping is not available for MX devices because these families do not have sequential modules. Sequential Remapping is not available for SX/SX-A or eX devices because the sequential modules in SX/SX-A and eX do not have a combinatorial component. Instead, optimal performance is achieved by an automatic DirectConnect between a combinatorial module and a sequential module whenever possible. Sequential Remapping performs the following three steps in order:

1. **It divides complex sequential library macros into basic combinatorial and sequential macros.** For example, a D-type flip-flop with 2-input multiplexed data "DFM" remaps into a 2 to 1 multiplexor "MX2" and a D-type flip-flop "DF1." The total number of logic modules has temporarily increased from one combinatorial and one sequential module to two combinatorial and one sequential module, shown in Figure 4-3.

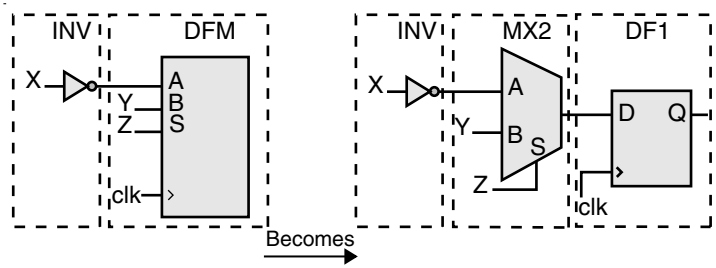


Figure 4-3. Complex Macros Divided

2. **It implements the new combinatorial macro by combining the basic combinatorial macro from step 1 with its previous combinatorial macro in a design.** For example, "MX2" and an inverter "INV" are combined into "MX2A." The total number of logic modules is decreased to one combinatorial and one sequential module, shown in Figure 4-4.

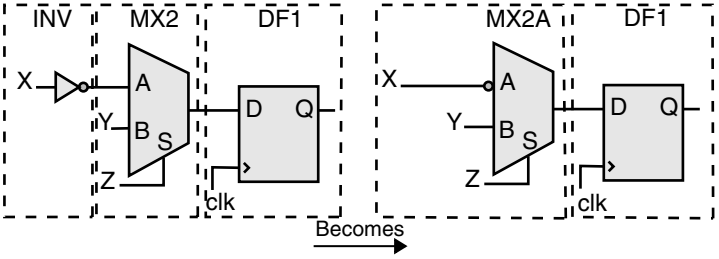


Figure 4-4. Logic Modules Decreased

3. **It combines the new combinatorial macro from step 2 with the basic sequential macro from step 1**. For example, "MX2A" and "DF1" are combined into a D type flip-flop with 4 input

multiplexed data, active low clear, and active high clock "DFM6A." The total number of logic modules is further decreased to one sequential module, shown in Figure 4-5.

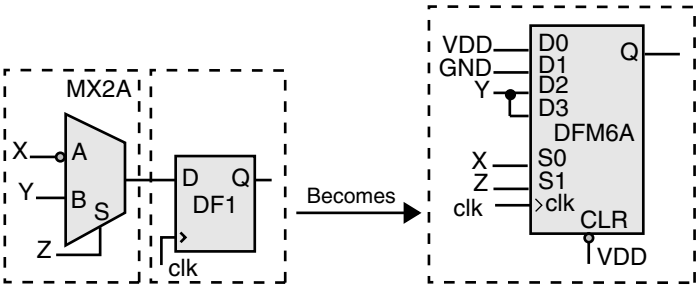


Figure 4-5. Logic Reduced to One Sequential Module

Unused Logic Removal

Unused Logic Removal removes all logic macros that are not driving any other logic macro input or do not propagate to an output pad. The removal of such macros does not affect the functionality of the circuit. An example of Unused Logic Removal is shown in Figure 4-6.

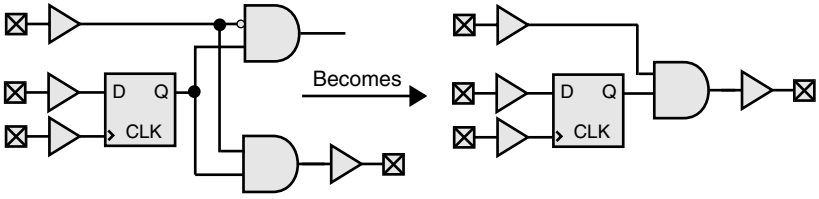


Figure 4-6. Unused Logic Removal

Constant Input Reduction

Every Actel hard macro can be mapped or configured in many different ways to implement logic functions. However, when macro inputs are tied to power or ground, the number of configurations available to the place-and-route software is decreased. Constant Input Reduction reconfigures macros that have unused inputs connected to power or ground into logically equivalent functions with the power or ground connections eliminated. Constant Input Reduction only reconfigures

combinatorial macros. Sequential macros that have inputs connected to power or ground are not affected. Figure 4-7 illustrates how Constant Input Reduction is achieved.

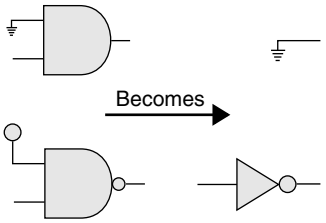


Figure 4-7. Constant Input Reduction

Fan-in Reduction

Fan-in Reduction reduces the number of inputs of a combinatorial macro that has inputs tied together by mapping it to a logically equivalent macro with fewer inputs. This function substantially improves the routability of a design. Fan-in Reduction does not reduce the number of logic modules and only reconfigures combinatorial macros. Sequential macros that have inputs tied together are not affected. Figure 4-8 illustrates Fan-in Reduction.

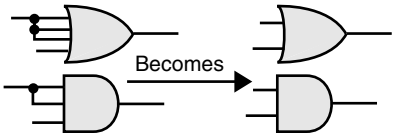


Figure 4-8. Fan-in Reduction

Back-Annotation Effects

The Combiner does not change the functionality of a design. Rather, it improves the density, speed and routability of a design. Changes resulting from logic removal or combining are not visually back annotated to the schematic, but the timing is adjusted accordingly. The removal or combining of logic does not adversely affect either back annotation to a simulator or timing analysis performed by Timer.

There are slight differences in the way delays are back annotated to a simulator and how they are viewed in Timer. Figure 4-9 shows how delays are viewed in schematics before combining.

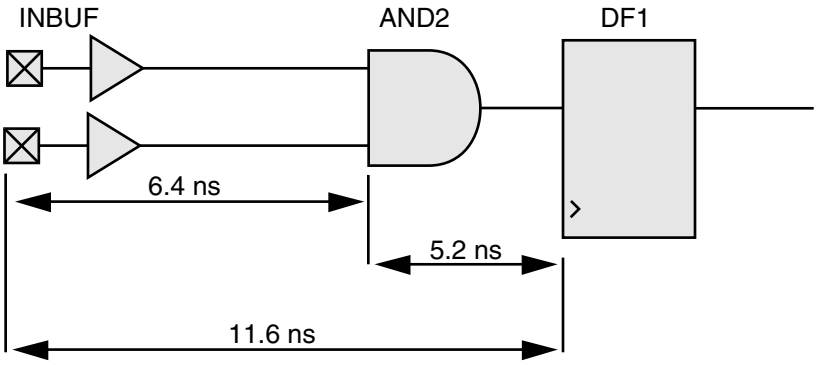


Figure 4-9. Delays Before Combining

A delay of 11.6ns (=6.4+5.2) represents the macro delay of "INBUF" and the wire delay between "INBUF" and "AND2" (6.4ns) plus the macro delay of "AND2" and the wire delay between "AND2" and "DF1" (5.2ns).

When "AND2" is combined with "DF1," the macro delay of "AND2" and the wire delay between "AND2" and "DF1" (5.2ns) no longer exists. However, "AND2" still exists on the schematic. Therefore, for back annotation purposes, Designer back annotates a delay of 0.0ns for the macro delay of "AND2" and the wire delay between "AND2" and "DF1." Timer also shows a 0.0ns delay for the macro delay of "AND2" and the wire delay between "AND2" and "DF1." The resulting delay

is 6.4ns (=6.4+0.0) instead of 11.6ns (=6.4+5.2) because of the zero delay for the macro delay of "AND2" and the wire delay between "AND2" to "DF1." Figure 4-10 illustrates this process.

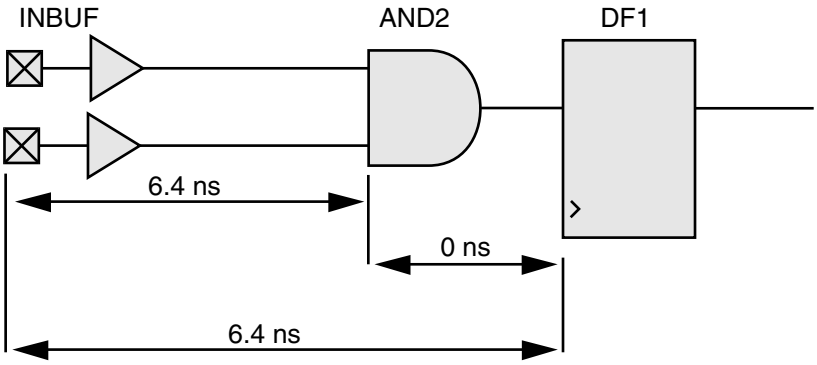


Figure 4-10. Delays After Combining

Combinability with ACT 1 and 40MX Devices

The MX architecture does not use sequential modules to implement flip-flops and latches. Therefore, combinatorial macros with flip-flops and latches cannot be combined. However, DFM type flip-flops with their inputs tied to GND and/or VCC can be used to implement a logic function with fewer modules and shorter propagation delay. Figure 4-11 illustrates how to tie the "A," "B," and "Select" inputs to implement a given gate/flip-flop combination in MX using one "DFM." This lowers the module count and the propagation delay.

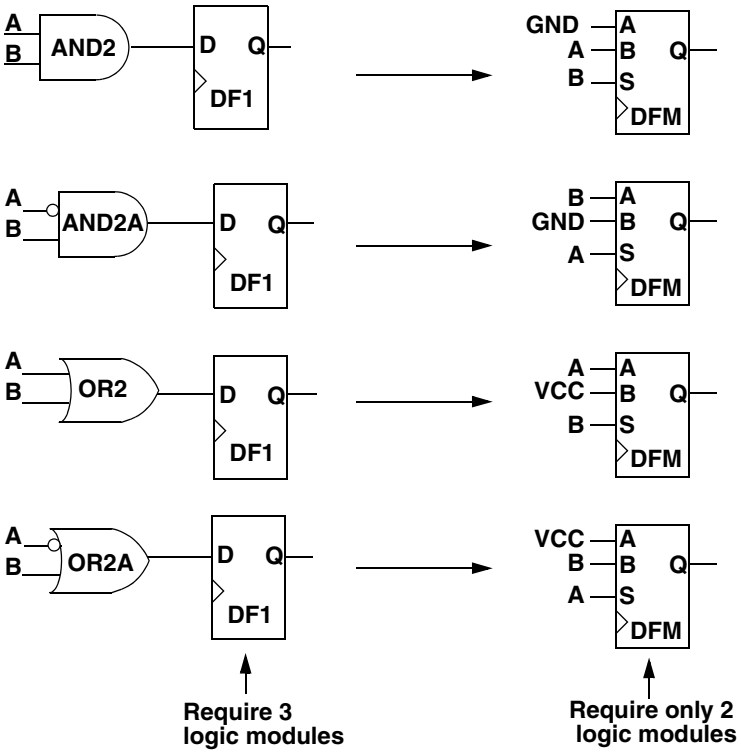


Figure 4-11. DFM Ties-Offs to Reduce Logic Module Count

Net Loading

High fanout degrades performance and increases the potential of unrouted nets. Due to physical limitations, Designer does not permit fanouts greater than 24 (except for the Axcelerator family - the fanout limit for Axcelerator devices is 32; check http://www.actel.com for device-specific information). In addition, the software warns you if any nets have an excess of 10 loads. If these nets are not speed-critical and there are a modest number of them, you can ignore these warnings. However, if critical nets in the design have a fanout greater than 10, you should modify them by buffering, as shown in Figure 4-12.

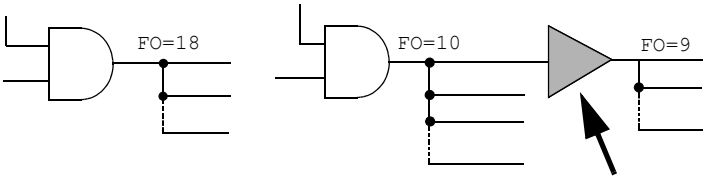


Figure 4-12. Buffering

An alternative method to buffering is duplicating logic to eliminate the buffer delay, shown in Figure 4-13.

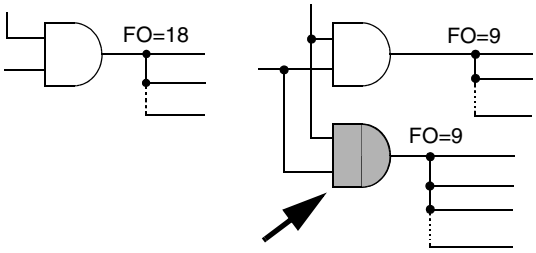


Figure 4-13. Duplicating Logic

Both buffering and duplicating logic cost additional logic modules. You can use buffering if you are not concerned about the skew or if you want to minimize the amount of logic resources used, because buffering generally takes less logic than logic duplication. However, Actel recommends that you duplicate logic to reduce fanout because it does not add an additional delay in the form of a buffer and it reduces skew.

Logic and I/O Utilization

The Designer software calculates the logic module and I/O module utilization. You can view module utilization by generating a Status report.

To improve the chances of optimal routing, Actel recommends that the total logic module utilization be between 50% and 85%.

- Lower utilization, especially with high I/O usage, causes excessively long delays and, possibly, unrouted nets.
- High utilization should be avoided if there is significant use of high pin count macros (for example, “MX4” and “DFM6A”).

If the design requires additional resources, you can use a larger device by changing the device and package selection within Designer. The specialized I/O macros take advantage of architectural enhancements with the later families. Use these macros to improve performance.

Refer to the Actel datasheet for your device and the *Antifuse, ProASIC/ProASIC*^{PLUS}, and *ProASIC3/E Macro Library* guides for additional information. Also, refer to the Actel website at http://www.actel.com for more information on routability.

Adding Properties

This section describes how to add properties to your designs.

Note: ALSPIN and ALSPRESERVE are not available for ProASIC3/E, ProASIC^{PLUS}, or ProASIC devices.

ALSPRESERVE

When Designer compiles a design, one of the functions of the Combiner is to combine (optimize) combinatorial functions into sequential macros whenever possible. Combining logic does not affect the functionality of the circuit. To prevent such combining, an "ALSPRESERVE" property may be added to the net connecting the macros that would otherwise be combined, as shown in Figure 4-14. The "ALSPRESERVE" property does not need to have a value assigned to it. Most schematic capture tools pass the property to the netlist. Refer to the documentation provided with your schematic capture tools for information about adding properties to nets.

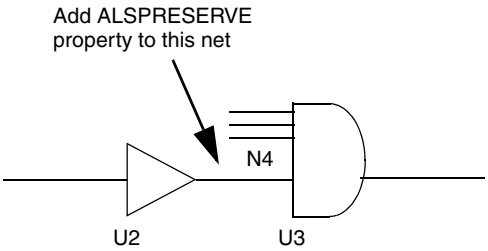


Figure 4-14. Adding ALSPRESERVE Property to a Net

Example EDIF netlist:

```
(net N4 (joined (portRef Y (instanceRefU2))
    (portRef D (instanceRef U3)))
        (property ALSPRESERVE (boolean (true))))
```

BUFD and INVD Delay Macros

The suggested uses of BUFD and INVD are as follows:

- Maintain the phase relationship between clock and data input signals when sending signals derived from them these signals off-chip. This application enables them to meet external setup and hold requirements.
- Maintain the phase relationship of clock and data input signals in designs with high-fanout clock signals.

Note: Our Flash devices do not support these delay macros.

Maintaining Phase Relationships

As shown in Figure 4-15, a certain phase relationship exists between the clock and data signals (generally, the data signal lags the clock signal). Actel recommends you employ BUFD to maintain this phase relationship. All Actel antifuse device families have a restriction that a CLKBUF cannot be connected directly to an OUTBUF, so Designer software automatically inserts a BUFF module (inst3 in Figure 4-15). This introduces a delay that may cause the data signal to arrive ahead of the sys_clk signal going off-chip. To prevent this, you can insert a BUFD macro in your netlist. The BUFD, unlike a regular BUFF, is assigned the ALSPRESERVE property. This ensures that the BUFD is not optimized away during the compilation step and hence can offset the delay caused by automatic insertion of the BUFF in the sys_clk path. This technique maintains the phase relationship of the data andclock inputs when going off-chip and the external setup and hold requirements can be met.

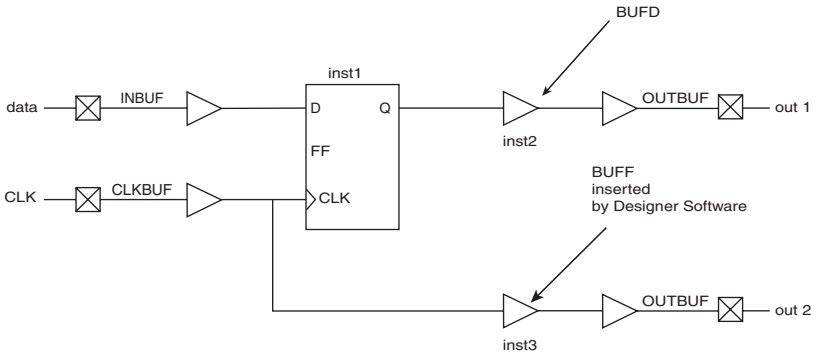


Figure 4-15. Phase Relationship Control - Data and System Clock using BUFD

In Figure 4-16, the clock signal coming from the CLKBUF has a high fanout. This means increased capacitive loading and hence increased delay. In contrast, the data signal has a fanout of only one.

The clock network's high fanout may allow the data input to (undesirably) lead the clock signal. Again, introducing one or more BUFD macros (inst1) in the data signal path provides enough delay to offset the delay of the high-fanout clock net.

If inversion of the signal is required in addition to delay, the INVD macro is available. Again, the ALSPRESERVE property is assigned to this macro to avoid elimination during optimization steps.

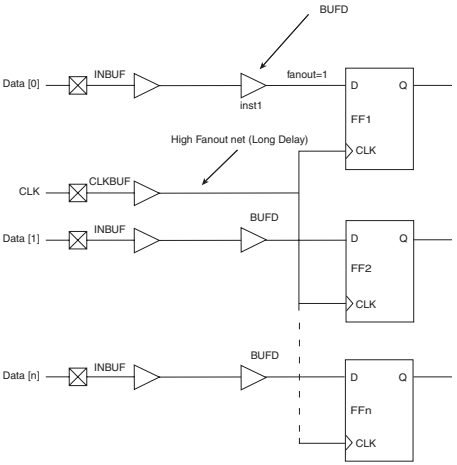


Figure 4-16. Phase Relationship Control - Designs with High Clock Fanouts

Usage Model - HDL Design Flow

None of the synthesis tools that support Actel technologies will infer the INVD or BUFD macros in the synthesized netlist. Simulation library models of INVD and BUFD, similar to their INV and BUFF counterparts, are available from Actel in both VHDL and Verilog. The manual process for buffer insertion is as follows:

1. Import the synthesized netlist into Designer. Run compile and layout.
2. Invoke Timer from Designer to check the timing/phase relationships for the signals of interest.
3. Using a text editor, manually modify the netlist and insert BUFD and/or INVD macros into the desired paths.
4. Import the modified netlist into Designer. Again run compile and layout.
5. Invoke Timer again to review the delays for the signals of step 2.

If necessary, repeat steps 3 through 5 until the design timing requirements are met.

Usage Models - Schematic Capture Design Flow

INVDs and BUFDs are available for the following Actel-supported schematic capture tools:

- ConceptHDL PE13.5 and PE13.6
- Mentor Graphics c.4 and D.1
- ePD 1.1/2.0

To avoid inadvertent instantiation of INVD or BUFD where INV or BUFF is intended, the graphical symbols for INVD and BUFD have warning labels "Special Use Only."

The flow for buffer insertion is as follows:

1. Generate an EDIF netlist with the schematic tool.
2. Import the EDIF netlist into Designer. Run compile and layout.
3. Invoke Timer from Designer to check the timing/phase relationships for the signals of interest.
4. Edit the schematics to insert BUFD and/or INVD macros into the desired paths.
5. Check and save your schematics. Generate an EDIF netlist from the modified schematics.
6. Import the modified netlist into Designer. Again run compile and layout.
7. Invoke Timer again to review the delays for the signals of step 3.
8. If necessary, repeat steps 4 through 7 until the design timing requirements are met.

Delay Values

Rising (R) and falling (F) delays for INVD and BUFD in different Actel families can be found in the relevant datasheets. These values are obtained using the fastest speed grade for the device using typical temperature, voltage, and process corners as well as optimized placement. Actual delay numbers may vary with the device, speed grade, and actual placement. Please visit our website at http://www.actel.com/documents/BUFD.pdf for more information.

ALSPIN

Actel provides special input, output, tri-state, and bi-directional macros for adding I/Os to your design. Add the I/O macros to your design and connect them by nets to both port symbols, and the functional logic making up the design. Most third-party CAE tools have special symbols to represent the ports. Refer to the Actel Interface Guides for information about adding I/Os to your design.

Note: ProASIC and ProASIC^{PLUS} devices do not support the ALSPIN property.

The ALSPIN property works only in pure schematic based designs in Designer. If your design is not purely schematic-based, use the ALSPIN property with the Synplify SCOPE tool.

Typically, pin assignments are made within Designer using PinEditor. However, many third-party CAE tools allow users to enter pin assignments directly into the design. The ALSPIN property may be added to the net connecting a port to an I/O buffer, shown in Figure 4-17. Assign the corresponding pin number as the value of the property. Most schematic capture tools pass the property to the netlist. Refer to the documentation provided with your schematic capture tools for more information about adding pin assignments.

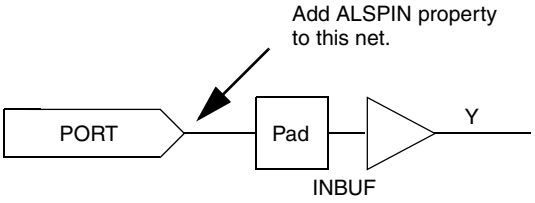


Figure 4-17. Adding ALSPIN Property to a Net

Example EDIF netlist:

```
(net N5 (joined (portRef PAD (instanceRefU3))
    (portRef Y(instanceRef U3)))
        (property ALSPIN (string "B2"))
        (property ALSFIX (boolean (true)))))))
```

Selecting I/O Placements

The design flow for Actel FPGAs allows the following different methods of selecting I/O placements:

Automatic pin assignment. No manual assignment is made for an I/O signal. After you compile the design, run Layout and Designer automatically selects the best pin location depending on characteristics of the design, such as timing constraints, device type and package, and overall design topology.

Manual pin assignment. Manual pin assignments allow maximum control over the design flow, since the I/O assignments remain fixed regardless of design changes. You can use any of the following three methods to perform manual pin assignment, but Actel recommends that you use only one method.

- **Use PinEditor:** Refer to the PinEditor online help for information about using PinEditor for manual pin assignment.
- **Assign pins from within the design schematic:** Designer allows you to assign and fix pins to I/O

signals in your design schematic. For more information on how to assign pins from within the design schematic, refer to the Actel Interface Guides. You cannot back annotate pin assignments from Designer to your design schematic.

- **Import the <design>.pin file into Designer:** Designer can import pin files to assign pins. Refer to the Designer online help for more information on how to import a file into Designer.

  Note: For Axcelerator, the PDC file replaces the PIN file.

  For Flash devices, the GCF file replaces the PIN file.

How you use the three methods above depends on the requirements of the project. You can get the final pin assignment by generating a pin report and/or printing the graphical view of the device pin assignment in PinEditor.

Suggested I/O Assignment Method

Because both methods (automatic or manual) of determining I/O pin assignments for Actel devices can be used in combination, there are many methods to create quality pin placements. The best method for a design depends on factors such as scheduling constraints, design changes, and performance specifications.

To design effectively, use automatic pin assignment for 100% of the I/O signals. Designer optimizes the place-and-route, especially if it is given the flexibility of automatically assigning all I/O placements. Designer selects, evaluates, and optimizes many different I/O configurations specific to the device architecture. If as few as 10% of the pin assignments are inefficiently assigned manually, the quality of the place-and-route could be compromised. Fixed manual assignments should be kept to a minimum.

Note: Automatic pin assignment may not be suitable for Axcelerator because of I/O bank assignments. Please see the PinEditor online help for more information on I/O bank assignments.

Fast I/O Assignment Procedure

The I/O assignment process does not have to be a gating item to the overall system schedule. The optimal design flow allows the software to reposition the I/O every design iteration. However, as long as I/O assignments change, PCB layout cannot take place, and the board may take several weeks to manufacture. To save time, as long as the number and function of the I/Os are finalized, use the following procedure to have Designer automatically assign all of the pins before the details of the design are completed and before the design is completely tested and debugged.

1. **Enter the design as completely as possible, ignoring small details to be modified later.** It is important to include all of the major functions that will be in the FPGA circuit. The objective is to obtain a netlist that approximates the final design topology. The circuit does not need to be

functionally correct at this point but must have the same major functional blocks (adders, counters, and so on) and approximately the same number of logic modules as the final design.

2. **Layout the design in Designer. Ignore any warnings from the Compile program that may be attributed to the unfinished state of the design.** Run Layout in standard mode with incremental placement turned off.

   At this point, Layout has automatically assigned I/Os according to the design topology. This is done before all of the functional bugs have been discovered and resolved. Minor functional changes that may be made to the design will have little effect on the quality and effectiveness of the pin placements.

3. **Layout the PCB.** Complete the PCB layout knowing that the pin assignments for the Actel device will not require modifications in the future.

   Layout does not modify any I/O placements that have been fixed. This method can be used even if the I/Os are manually assigned. The results of automatic I/O assignment by Layout can be used as a template for manual I/O assignment. The automatic placements can be used as a guide and small modifications can be made as required.

Assigning I/Os Manually

If any or all of the I/O placements must be assigned manually, a broader knowledge of Actel FPGA architecture is required. Refer to the Actel datasheets for specific devices for information about different architectures of the Actel families. The structure of the logic and I/O modules, routing tracks, antifuses, and other architectural details are discussed. This information can be used to assign I/O signals manually with the specific details of the device in mind.

During the process of assigning I/Os for the device, the following guidelines can help to increase routability and performance:

- Try to force signal paths, especially large data buses, to flow horizontally across the die, since there are more horizontal (than vertical) routing resources. In most cases, a large data bus requires many interconnections as it traverses the circuit.

  The horizontal and vertical orientation of the die is the same as shown in the package pin assignment and mechanical detail drawings in the Actel datasheets.

I/O Standard Compatibility

- Count the number of levels of logic between I/Os to determine placement. For example, I/Os that are separated by one gate should be placed closer than I/Os that are separated by a long shift register. In general, the Actel device architecture allows signals to be routed across two vertical rows and over one-third of the columns of the device without using a long routing track. For example, consider an A1225 device that has 13 rows and 46 columns. Two I/O signals should not be placed on opposite sides of the device unless there are at least two horizontal or three vertical

levels of logic between them.

- Use the top and bottom I/O pins for slow and/or local signals. The fact that there are fewer vertical routing resources should not harm the performance of these signals.
- Use the global clock buffers. Each of these pins is connected to a dedicated, low-skew distribution network that is optimized for high fanout signals.
- For Axcelerator, be sure make the assignments to I/O banks with the appropriate I/O technology.

Unused I/O Pins

Every I/O bond pad on the die is connected to a multipurpose circuit capable of becoming an input, output, or tri-state I/O. The circuit is always connected to the bond pad, and Designer configures it according to the design's specification. Refer to http://www.actel.com/custsup/search.html and search the keyword "unused" for more information.

Note: This option is not available for Axcelerator devices.

Using Probe Pins as I/O Pins

Layout automatically avoids using the probe pins unless you have fixed pin assignments to one or more of these pins, or the number of pins in the design exceeds the number of non-probe I/O pins. The function of the probe pins depends on the JTAG mode Pin and Security Fuse Configuration. The availability and functionality of pins varies by device family. Please refer to the family-specific datasheets for precise data specifications (Axcelerator probe pins are dedicated). To use probe pins and JTAG pins as I/Os:

- Avoid using PRA or PRB as Input or Bidirectional pins, and
- Avoid using TDI/SDI, TCK/DCLK as Input or Bidirectional (using any of these pins in this way disables probing in the ACT1/40MX families)

Using JTAG Pins as I/O Pins

The high-capacity devices of the DX, MX, SX and SX-A families have four JTAG pins: "TDI," "TMS," "TCK" and "TDO." Layout automatically avoids using these pins as I/O pins. If you are not using the JTAG function, you can use these pins as I/O pins except TMS. In addition, SX-A and eX families have a JTAG reset pin TRST. This pin can also be used as a user I/O if it is not used for the JTAG function. For Axcelerator, the JTAG pins are dedicated and cannot be used as I/O pins.

Generating a Top-Level Symbol

You can create a top-level symbol for the entire design in your schematic entry tool. The pin names on the symbol must match the underlying port names. The top-level symbol must only contain pins for I/O buffers. Do not add power, ground, probe, or other special pins to the symbol. Doing so causes errors during netlist generation. Remember to update the symbol if pins change on the schematic.

Entering Constraints for Timing Driven Place-and-Route

Unique timing constraints for each signal path in a design can be specified using the Timer tool. It may only be necessary to specify the clock frequency for synchronous designs. The Timer mode typically eliminates time-consuming iterations of the design flow. Timing constraints can also be specified in a design constraint file and imported into Designer. Refer to the Timer online help for information about specifying timing constraints in Timer, and the Designer online help for information about importing a constraints file.

Estimating Pre-Layout Timing

It is often necessary to estimate the timing of a design to be implemented prior to using the design tools. With some knowledge of the critical path, you can perform this analysis using the timing information in the device-specific Actel datasheet. If you are having trouble selecting the appropriate device, contact your local sales representative.

Figure 4-18 illustrates the procedure using an ACT 2 A1225XL-1 design example with a critical path from the CLK pad to the OUT pad. The delays used come from the timing tables for the A1225XL-1 in the Actel datasheets. These values include a statistical net delay which is associated

with each macro output. The OUTBUF macro assumes a load of 35 pF. The analysis is based on worst-case commercial conditions.

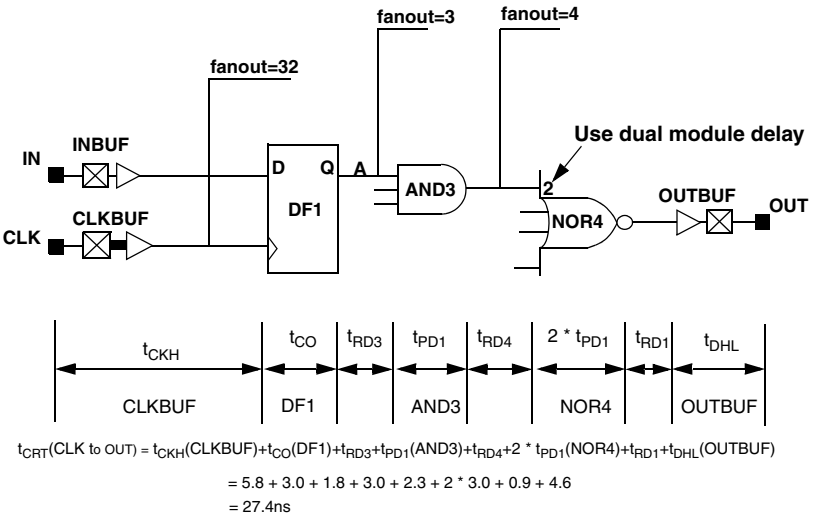


Figure 4-18. Critical Path Timing Example

Note: The "NOR4" macro has a significantly higher delay because it is a two-module macro.

Adding ACTgen Cores

With ACTgen you can create cores using a simple graphical interface. For many schematic capture tools, a symbol and netlist description of the macro can be created directly from ACTgen. Other tools require a separate utility that uses third-party netlist readers to create a symbol and netlist description. Once the ACTgen symbol has been added to a schematic, the macro is added to the design's netlist during netlisting. Use ACTgen to configure the PLLs, RAM, FIFO, etc., for ProASIC, ProASIC^{PLUS}, ProASIC3/E, and Axcelerator devices.

A

PDC Constraints in Libero IDE

ProASIC3/E only supports PDC constraints for placement. Some GCF commands may not have equivalent PDC commands, although other similar constraints can be used. The following is only a short list of the constraints affected. Consult the online help included with the software for information on individual constraints.

I/O Constraints

The most common placement constraints are I/O related. The pin assignment constraint can use a GCF constraint or a TCL command for ProASIC^{PLUS}. If the 'pin_assign' TCL command is used to assign an I/O to the pin location, then the same TCL command can be used to assign the I/O for ProASIC3/E. However, if GCF constraints are used to assign the I/O, please refer to the recommendations below.

set_io

This statement is used to either assign package pins to I/O ports or locate I/O ports on a specified side of a device. This is a hard constraint and cannot be overruled by the placer.

GCF Constraint: set_io { package_pin | pad_location} netName/portName;

PDC: set_io <portname> -pinname <pinname> -fixed yes

set_empty_io

This statement is used to specify a location in which no I/O pin can be placed.

GCF Constraint: set_empty_io { package_pin | pad_location};

There is currently no equivalent PDC constraint for this GCF constraint. However, if the set_io constraint is used to assign all the I/Os, this constraint is not required for the ProASIC3/E families.

set_initial_io

This statement is used to initially assign package pins to I/O ports or place I/O ports at specific locations of a device. The placer can reassign or relocate the cells.

GCF Constraint: set_initial_io { package_pin | pad_location} io_port_name [, io_port_name , …];

PDC: set_io <portname> -pinname <pinname> -fixed no

Logic Placement Constraints

Do not convert the logic placement constraints from ProASIC^{PLUS} (APA) to ProASIC3/E devices. Use the MVN GUI or use PDC constraints to create new placement constraints.

In ProASIC family, floorplanning can be done using GCF constraints or the MultiView Navigator (MVN) tool to improve timing and placement. The following sections describe the various GCF constraints for the logic placement and equivalent PDC constraints that will help to create new placement constraints for ProASIC3/3E family devices.

set_location

This statement is used to locate a cell instance at specified x,y coordinates, defined as a region. The first variation shown specifies the exact location for cell placement (lower left corner of cell to be placed at this tile). Placer cannot relocate the cell instance during place-and-route. The second variation provides a bounding box within which the cell instance is to be placed.

Variation 1 - Specify the Exact Location

```
set_location (x,y) hier_inst_name;
```

The equivalent PDC command to locate a cell instance at specified x,y coordinates is also set_location.

```
set_location <clustername> -fixed yes x y
```

where cluster name is the cluster corresponding to the region. Please check the Designer online help for the cluster name of the ProASIC3 device.

Variation 2 - Specify the Region

```
set_location (x bl ,y bl x tr ,y tr ) hier_inst_name/*;
```

where x , y (required) are the (x, y) tile coordinates that specify the empty cell location and x bl , y bl x tr , y tr (required) are the x, y tile coordinates for the bottom left and top right corner of the region. To assign an instance or instances to a region, two PDC commands must be used. The define_region command defines a rectangular region and the assign_region command constrains a set of macros to a specified region.

```
define_region -name region_name -type inclusive x1 y1 x2 y2
assign_region region hier_inst_name
```

set_initial_location

This statement is used to initially locate a cell instance at specified x, y coordinates. The placer can relocate the cell instance during place-and-route.

```
set_initial_location ( x, y ) hier_inst_name ;
```

Where x , y (required) are the x, y tile coordinates for the location of a specified cell instance and "hier_inst_name" (required) is the hierarchical path to a cell instance. The equivalent PDC command to locate a cell instance at specified x,y coordinates is also set_location.

```
set_location <clustername> -fixed no x y
```

set_empty_location

This statement is used to specify a location in which no cell should be placed.

```
set_empty_location ( x ,y);
set_empty_location (xbl ,ybl xtr ,ytr );
```

Where x , y (required) are the (x, y) tile coordinates that specify the empty cell location and x bl , y bl x tr, y tr (required) are the x, y tile coordinates for the bottom left and top right corner of the region. There is currently no equivalent PDC constraint to specify the empty location for one cell. However, there is an equivalent PDC command to define an empty region.

```
define_region -name region_name -type empty x1 y1 x2 y2
```

assign_local_clock

GCF constraint allows a single spine or a rectangle of spine region to be specified which may encompass more than one spine region. use_global B1, T3 <net_name>;

The equivalent PDC command is assign_local_clock. To assign a net to a local user clock region, use the following command:

```
assign_local_clock -net <net name> -type <clock type> <clock region>
```

Where <net name> is the name of the net assigned to the local user clock region. <clock type > is 'chip' for a clock region driven by a local clock driver located in the middle of the chip. <clock type > is 'quadrant' for a clock region driven by a local clock driver located in the top or the bottom of the chip.

<clock region> is a single spine defined as T# (Top spine) or B# (Bottom spine). Individual spines are defined from left to right starting at 1. The maximum spine number is a function of the die. <clock region> is a multi-spine rectangle defined as <upper left spine name>:<lower right spine name>. The spine names are identical to the one defined above. For example:

```
assign_local_clock -net localReset -type chip T1
assign_local_clock -net localReset -type quadrant T1:T5
assign_local_clock -net localReset -type chip T1:B6
```

assign_global_clock

GCF statement is used to classify nets as global nets.
set_global hier_net_name [, hier_net_name …];

In PDC for ProASIC3/E, to promote a given net of the design to a global clock network, use the following command:

```
assign_global_clock -net <net name>
```

Where <net name> is the name of the net being promoted to a global clock network and

<quadrant clock region> is an optional parameter (clock type is quadrant) defining which quadrant the net should be assigned to. Quadrant clock regions are defined as UL (upper left), UR (upper right), LL (lower left) and LR (lower right).

unassign_global_clock

In GCF file, use set_noglobal to avoid automatic promotion of nets to global nets.

```
set_noglobal hier_net_name [ , hier_net_name … ];
```

If the net was previously assigned to a global resource, this statement will demote it from the global resource.

In PDC, to demote a given global net of the design to a regular net, use the following command:

```
unassign_global_clock -net <net name>
```

Where <net name> is the name of the global net being demoted to a regular net.

set_net_region

This GCF constraint allows all the connected instances, drivers, and all driven instances to be put in the specified rectangle. The I/Os are excluded from the region. For example:

```
set_net_region (x1, y2 x2, y2) <net_name_wildcard>;
```

Two PDC constraints are required to place the elements in a particular area. The assign_net_macros command assigns macros connected to a net to a region.

```
define_region -name region_name -type inclusive x1 y1 x2 y2
assign_net_macros region_name net1 ?netN?…
```

In Designer software for the ProASIC3/E families, there is a new Compile option that can be used in addition to PDC constraints to control global promotion and demotion.

B

Synopsys Design Constraints (SDC) in Libero IDE

ProASIC3/E only supports SDC file(s) for timing constraints. The constraints listed in this appendix comprise a partial list. Please refer to the Libero or Designer online help for more information on constraints and constraint files.

create_clock

A GCF clock constraint on a net needs to be converted to a clock constraint on one of the eligible ports. For example:

GCF Constraint: create_clock -period 2.000000 U1/gla_clk_int

SDC: create_clock -period 2.000000 U2/Core:GLA

The clock constraint on net "U1/gla_clk_int" has been translated to port "U2/Core:GLA".

The GCF clock constraint on a port can be used in SDC. No conversion is required.

set_max_path_delay

The set_max_path_delay GCF constraint can be translated into a max_delay with the "-from" and "-to" options. For example:

GCF Constraint: set_max_path_delay 4.000000 s_reg_5.CLK s_reg_5.Q U15.A U15.PAD SDC: set_max_delay 4.000000 -from [get_pins {s_reg_5:CLK}] -to [get_pins {U15:PAD}]

set_input_to_register_delay

A set_input_to_register_delay GCF constraint needs to be translated into a max_delay constraint. For example:

GCF Constraint: set_input_to_register_delay 3.000000 -from dat*

SDC: set_max_delay 3.000000 -from [get_ports {dat*}] -to [all_registers]

set_register_to_output_delay

A set_register_to_output_delay GCF constraint needs to be translated into a max_delay constraint. GCF Constraint: set_register_to_output_delay 2.000000 -to out SDC: set_max_delay 2.000000 -from [all_registers] -to [get_ports {out}]

C

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0)1276.401500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650. 318.8044**

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Actel Technical Support

Visit the Actel Customer Support website (www.actelcom/.custsup/search.html) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/contact/offices/index.html.

Index

E

F

G

H

R

S

T

U

V

W

For more information about Actel's products, visit our website at http://www.actel.com

Actel Corporation *• 2061 Stierlin Court • Mountain View, CA 94043 USA*
Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

Actel Europe Ltd*. • Dunlop House, Riverside Way • Camberley, Surrey GU15 3YL • United Kingdom*
Phone +44 (0)1276 401 452 • Fax +44 (0)1276 401 490

Actel Japan *• EXOS Ebisu Bldg. 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
Phone +81.03.3445.7671 Fax +81.03.3445.7668

Actel Hong Kong *• 39th Floor, One Pacific Place • 88 Queensway, Admiralty Hong Kong*
Phone +852.227.35712 Fax +852.227.35999

5029123-7