Actel® Quick Start Guide

v5.0

Windows® and UNIX® Environments

For more information about Actel's products, call 888-99-ACTEL
or visit our Web site at http://www.actel.com

Actel Corporation • 955 East Arques Avenue • Sunnyvale, CA USA 94086
U.S. Toll Free Line: 888-99-ACTEL • Customer Service: 408-739-1010 • Customer Service FAX: 408-522-8044
Customer Applications Center: 800-262-1060 • Customer Applications FAX: 408-739-1540

Actel Europe Ltd. • Dunlop House, Riverside Way • Camberley, Surrey GU15 3YL• United Kingdom
Tel: +44 (0)1276.401450 • Fax: +44 (0)1276.401490

Actel Japan • EXOS Ebisu Bldg. 4F • 1-24-14 Ebisu Shibuya-ku • Toyko 150 • Japan
Tel: +81 (0)334-457-671 Fax: +81 (0)334-457-668

5029123-5

Actel® Quick Start Guide

Windows® and UNIX® Environments

Actel Corporation, Sunnyvale, CA 94086

© 2003 Actel Corporation. All rights reserved.

Printed in the United States of America

Part Number: 5029123-5

Release: July 2003

No part of this document may be copied or reproduced in any form or by any means without prior written consent of Actel.

Actel makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability or fitness for a particular purpose. Information in this document is subject to change without notice. Actel assumes no responsibility for any errors that may appear in this document.
This document contains confidential proprietary information that is not to be disclosed to any unauthorized person without prior written consent of Actel Corporation.

Trademarks
Actel and the Actel logotype are registered trademarks of
Actel Corporation.

Adobe and Acrobat Reader are registered trademarks of
Adobe Systems, Inc.

Mentor Graphics, Precision RTL, Exemplar Spectrum, and Leonoardo Spectrum are registered trademarks of Mentor Graphics, Inc.

WaveFormerLite is a registered trademark of SynaptiCAD, Inc.

Synplify is a registered trademark of Synplicity, Inc.

Sun and Sun Workstation, SunOS, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc

Synopsys is a registered trademark of Synopsys, Inc.

Verilog is a registered trademark of Open Verilog International.

Viewlogic, ViewSim, ViewDraw and SpeedWave are trademarks or registered trademarks of Viewlogic Systems, Inc.

Windows is a registered trademark and Windows NT is a trademark of
Microsoft Corporation in the U.S. and other countries.

UNIX is a registered trademark of X/Open Company Limited.

All other products or brand names mentioned are trademarks or registered trademarks of their respective holders.

Table of Contents

Introduction 7
Document Organization 7
Document Assumptions 8
Document Conventions 8
Your Comments 8
Online Help 9
Actel Manuals 9

1 Designer Software Overview 11
Designer. 11
ACTgen Macro Builder 15
Silicon Expert 15
Designer Maintenance 15
Schematic-Based Design Methodology. 15
HDL Synthesis-Based Design Methodology 16

2 Actel Design Flows 17
Schematic-Based Design Flow Illustrated 17
Schematic-Based Design Flow Overview 18
HDL Synthesis-Based Design Flow Illustrated. 20
HDL Synthesis-Based Design Flow Overview 21

3 Design Considerations 23
Naming Conventions 23
Hierarchical Designs 27
Multiple Sheet Designs 27
Actel Libraries. 27
Adding Power and Ground. 27
Adding a Global Network 28
Using Flash Global Routing Resources. 32
Combinability 32
Net Loading. 41
Logic and I/O Utilization 42

Adding Properties 42
ALSPIN 47
Generating a Top-Level Symbol 52
Entering Constraints for Timing Driven Place-and-Route 52
Estimating Pre-Layout Timing 52
Adding ACTgen Macros 54

4 Flash Design Considerations 55
Design Optimization 55
Design Integrity 55
Design Hints 56
Implementing Memories 57

5 Quick Start Tutorial 59
Step 1 – Create a New Project 59
Step 2 – Perform Pre-synthesis Simulation 65
Step 3 – Synthesize the Design in Synplify 74
Step 4 – Perform Post-Synthesis Simulation 76
Step 5 – Implement the Design with Designer 77
Step 6 – Perform a Timing Simulation with Back-Annotated Timing . . 83
Step 7 – Generate the Programming File 84
Step 8 – Program the Device 85

A Product Support 95
Actel U.S. Toll-Free Line 95
Customer Service 95
Actel Customer Technical Support Center 95
Guru Automated Technical Support 96
Web Site 96
Contacting the Customer Technical Support Center 96
Worldwide Sales Offices 98

Index 99

Introduction

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

This manual includes information about the Designer software, which allows you to import a netlist generated from a third-party CAE tool, place-and-route the design, 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 Designer Development System.

Document Organization

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

Chapter 1 - Designer Software 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 - Design Considerations contains useful information and procedures about creating designs using the Actel Designer software.

Chapter 4 - Flash Design Considerations contains information and procedures to assist you in creating Actel designs with ProASIC devices.

Chapter 5 - Quick Start Tutorial illustrates a basic VHDL design for the APA Evaluation Board targeted at the Actel ProASIC$^{\text{PLUS}}$ family.

Appendix A- 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 Designer Series software.
2. You are familiar with UNIX workstations and UNIX operating systems, or with PCs and Windows operating environments.
3. You are familiar with FPGA architecture and FPGA design software.

Document Conventions

The following conventions are used throughout this manual.

The contents of a file is formatted as follows:

```
file contents
```

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. Available families include the act1, act2 (for ACT 2 and 1200XL devices), ACT3, 3200DX, 40MX, 42MX, and 54SX, 54SX-A, eX, and Axcelerator.

The <vhd_fam> variable represents Compiled VHDL libraries. To reference an actual compiled library, substitute the name of the library (act1, act2 (for ACT 2 and 1200XL devices), ACT3, A3200DX, A40MX, A42MX, A54SX, A54SX-A, eX, A500K/APA, and Axcelerator) 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, ACTmap, Silicon Expert, Silicon Explorer II, and Silicon Sculptor.

Actel Manuals

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

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

1

Designer Software Overview

The Designer Software is an integrated suite of user-friendly tools for PC and Workstation environments that takes your design idea to working silicon quickly and easily. These tools are designed to satisfy the demands of today's design engineers to accelerate the system logic design process. This chapter describes the components and features of the Designer Development System.

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 so you don't have to.

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. Fuse supports both Actel and Data I/O formats. 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 the 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. PinEditor should be used 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 macro 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 Flash 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

The 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 and Flash 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. Please refer to the SmartPower online help (in Designer) for more information.

ACTgen Macro Builder

ACTgen is a graphical macro 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 macros, 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 Macros Reference Guide* for a complete description of all the macros available in ACTgen.

Silicon Expert

Silicon Expert can be used during design creation to add I/Os to a design, balance buffer trees, or generate a netlist report. Silicon Expert can also be used after design creation to translate a structural netlist from one format to another. Refer to the *Silicon Expert User's Guide* for additional information. Not available for Flash and Axcelerator devices.

Designer Maintenance

After your first installation of the Designer, unless you completely remove the original installation, Setup always goes into maintenance mode. Maintenance mode (also accessible from the Start -> Programs -> Designer 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.

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. Refer to “Schematic-Based Design Flow Illustrated” on page 17 for an overview of the Designer schematic-based design flow.

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, Silicon Expert and Designer help you complete the design with place-and-route, timing analysis, and timing verification.

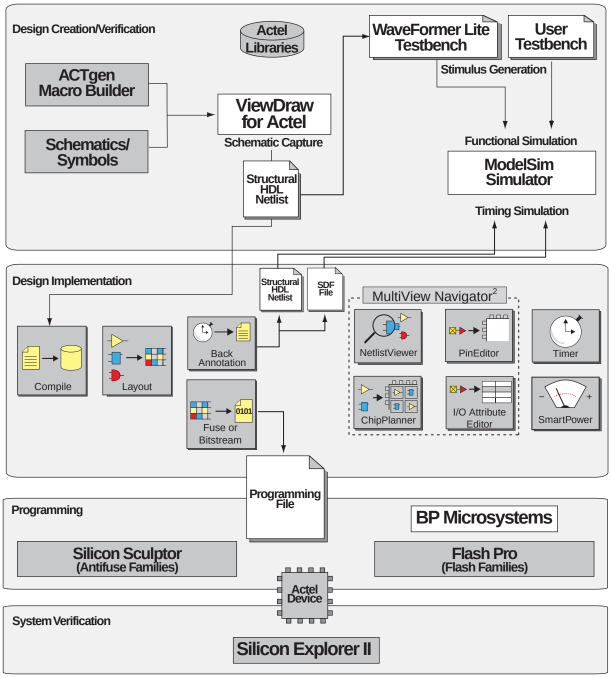
2

Actel Design Flows

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

Schematic-Based Design Flow Illustrated

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



Design Creation/Verification
Actel Libraries
WaveFormer Lite Testbench
User Testbench
Stimulus Generation
ACTgen Macro Builder
Schematics/ Symbols
ViewDraw for Actel
Schematic Capture
Functional Simulation
ModelSim Simulator
Structural HDL Netlist
Timing Simulation
Design Implementation
Structural HDL Netlist
SDF File
MultiView Navigator[2]
Compile
Layout
Back Annotation
NetlistViewer
PinEditor
Timer
0101
Fuse or Bitstream
ChipPlanner
I/O Attribute Editor
SmartPower
Programming File
Programming
BP Microsystems
Silicon Sculptor (Antifuse Families)
Flash Pro (Flash Families)
Actel Device
System Verification
Silicon Explorer II

Figure 2-1. Actel Schematic-Based Design Flow

---

[1] Actel-specific utilities/tools are denoted by the grey boxes in Figure 2-1.

[2] The MultiView Navigator interface and ChipPlanner tool support only Flash and Axcelerator devices. For all other familes, Designer displays the 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[1], 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. 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.

---

[1] *Visit our website at http://www.actel.com/products for more information on the Libero IDE.*

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 and Flash 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.

System Verification

You can perform system verification on a programmed device using the Actel Silicon Explorer diagnostic tool. Refer to the *Silicon Explorer User's Guide* for more information.

HDL Synthesis-Based Design Flow Illustrated

Figure 2-2 shows the HDL synthesis based design flow for an Actel device using Designer Series and third-party HDL synthesis software.[1]

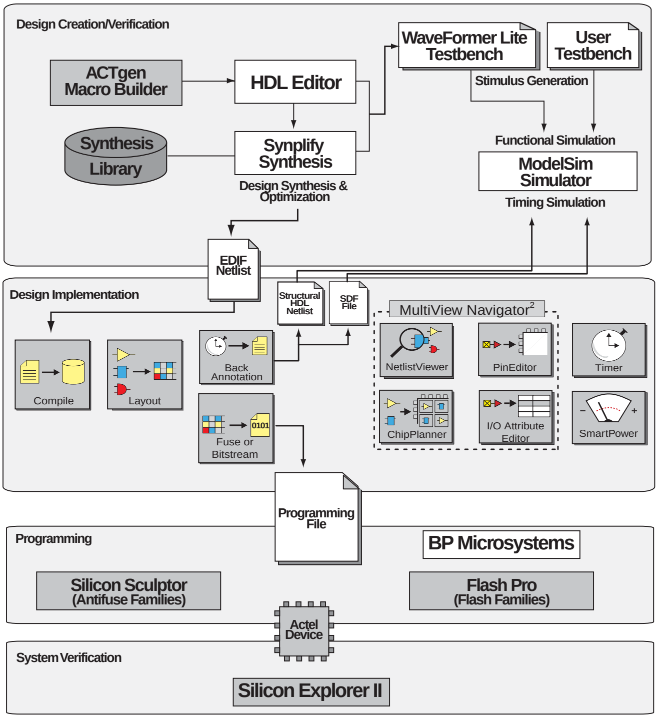

Design Creation/Verification
ACTgen Macro Builder
HDL Editor
WaveFormer Lite Testbench
User Testbench
Stimulus Generation
Synthesis Library
Synplify Synthesis
Design Synthesis & Optimization
Functional Simulation
ModelSim Simulator
Timing Simulation
EDIF Netlist
Design Implementation
Structural HDL Netlist
SDF File
MultiView Navigator2
Compile
Layout
Back Annotation
Fuse or Bitstream
0101
NetlistViewer
PinEditor
Timer
ChipPlanner
I/O Attribute Editor
SmartPower
Programming File
Programming
BP Microsystems
Silicon Sculptor (Antifuse Families)
Flash Pro (Flash Families)
Actel Device
System Verification
Silicon Explorer II

Figure 2-2. Actel HDL-Based Design Flow

---

1. *Actel-specific utilities/tools are denoted by the grey boxes in Figure 2-2.*
2. *The MultiView Navigator interface and ChipPlanner tool support only Flash and Axcelerator devices. For all other familes, Designer displays the NetlistViewer, ChipEditor, PinEditor, I/O Attribute Editor, Timer, and SmartPower.*

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 (Funtional) 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 19 for more information.

.

3

Design Considerations

This chapter contains information and procedures to assist you in creating Actel designs. For more information on creating designs for Flash devices, please refer to “Flash Design Considerations” on page 55.

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

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

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 3-1.

Table 3-1. Global Network Attributes

Input Pad Name	Type	Family	#	Internal Drive Option	Macro	Note
CLK	routed	ACT 1 40MX	1	Yes (CLKBIBUF only)	CLKBUF; CLKBIBUF	Can adjust skew with clock balancing
CLKA CLKB	routed	ACT 2; 1200XL; ACT 3; 3200DX; 42MX; 54SX; 54SX-A; RTSX-S; eX	2	Yes[a]	CLKBIBUF; CLKBIBUFI; CLKBUF; CLKINT; CLKBUFI[b]; CLKINTI[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; 54SX; 54SX-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]
IOCLK	dedicated	ACT 3	1	No	IOCLKBUF	Connected to all I/O module registers
IOPCL	special	ACT 3	1	No	IOPCLBUF	Connected to I/O module set and reset pins
QCLK A-D	routed	3200DX[d]; 42MX; 54SX72A; RTSX72S	4	Yes	QCLKBUF ; QCLKBUFI[b]; QCLKBIBUFI[b];QCLKBIBUF[b]; QCLKINT; QCLKINTI[b]	Each QCLK drives a quadrant of the device

a. The Internal Drive Option for ACT 2, 1200XL, ACT 3, 3200DX, 42MX, 54SX and 54SX-A can only be utilized by selecting the CLKINT macro to drive an internal clock network.

b. 54SX72A 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 3-2 lists the ACT 3 sequential elements and Table 3-3 lists the 54SX/54SX-A/eX sequential elements that cannot be connected to HCKLBUF because they are built from combinatorial modules.

---

[1] *Axcelerator only.*

Table 3-2. 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 3-3. 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. IOPCL[1], 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 ProASIC$^{\underline{\text{PLUS}}}$ (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. Each Flash device includes four dedicated global lines that all tiles in the device can access. 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 3-1.

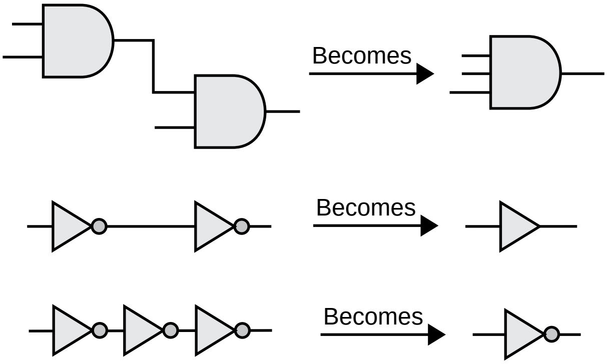


Figure 3-1. Combined Combinatorial Macros

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 3-2. Combinatorial Module Reduction is not available for ACT 1 or 40MX devices because these families do not have sequential modules. 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.

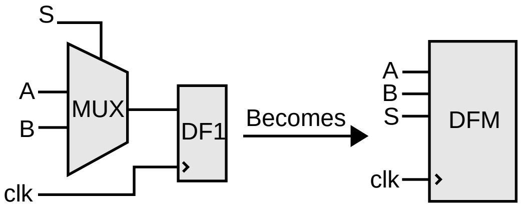


Figure 3-2. Combined Combinatorial and Sequential Macros

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 ACT 1 or 40MX devices because these families do not have sequential modules. Sequential Remapping is not available for 54SX/54SX-A or eX devices because the sequential modules in 54SX/54SX-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 3-3.

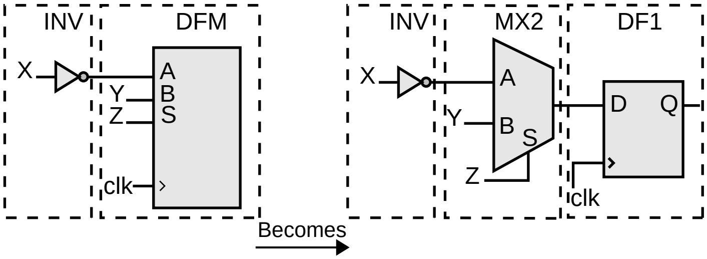

INV
DFM
X
A
B
S
Y
Z
clk
Becomes
INV
MX2
DF1
X
A
Y
B
S
Z
D
Q
clk

Figure 3-3. 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 3-4.

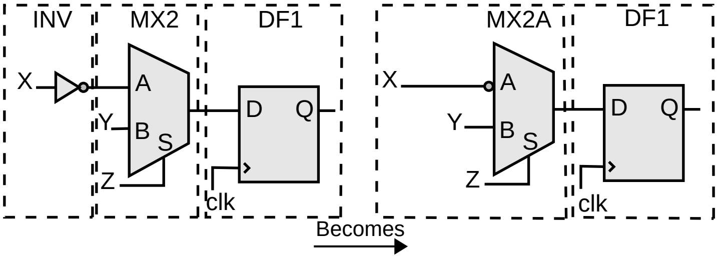

INV
MX2
DF1
X
A
Y
B
S
Z
D
Q
clk
Becomes
MX2A
DF1
X
A
Y
B
S
Z
D
Q
clk

Figure 3-4. 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 3-5.

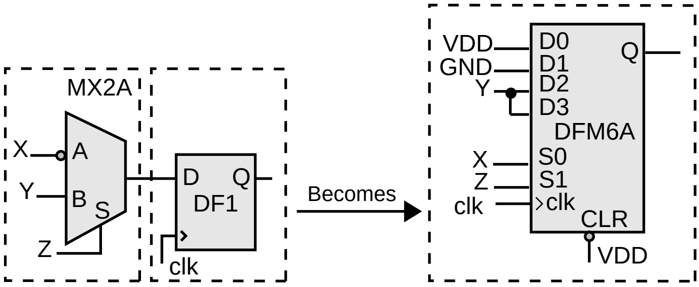


Figure 3-5. Logic Reduced to One Sequential Module

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 3-6.

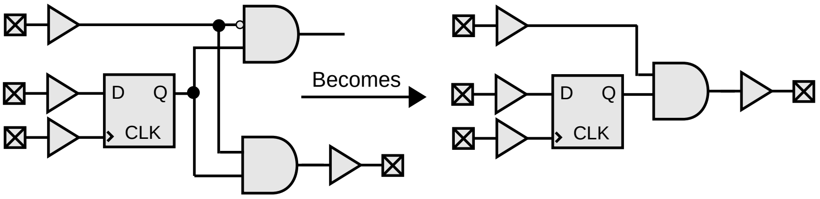


Figure 3-6. Unused Logic Removal

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 3-7 illustrates how Constant Input Reduction is achieved.

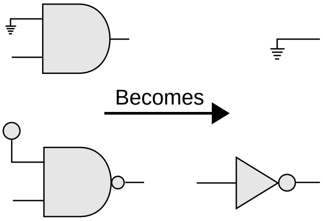


Figure 3-7. Constant Input Reduction

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 3-8 illustrates Fan-in Reduction.

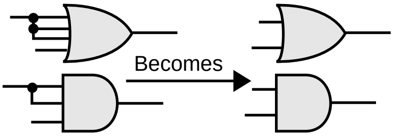


Figure 3-8. Fan-in Reduction

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 3-9 shows how delays are viewed in schematics before combining.

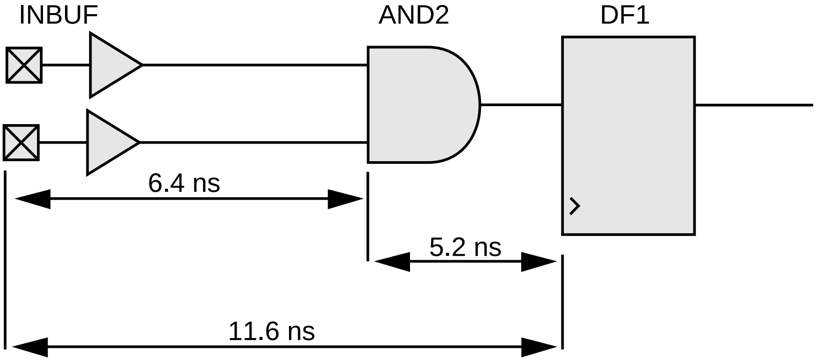


Figure 3-9. Delays Before Combining

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 3-10 illustrates this process.

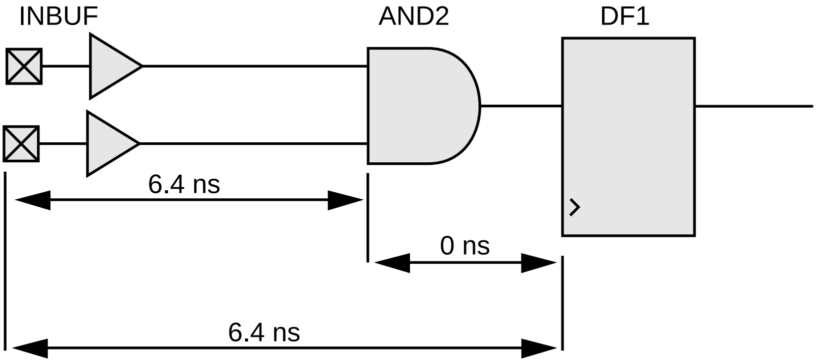


Figure 3-10. Delays After Combining

Combinability with ACT 1 and 40MX Devices

The ACT 1 and 40MX architectures do 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 3-11 illustrates how to tie the “A,” “B,” and “Select” inputs to implement a given gate/flip-flop combination in ACT 1 or 40MX using one “DFM.” This lowers the module count and the propagation delay.

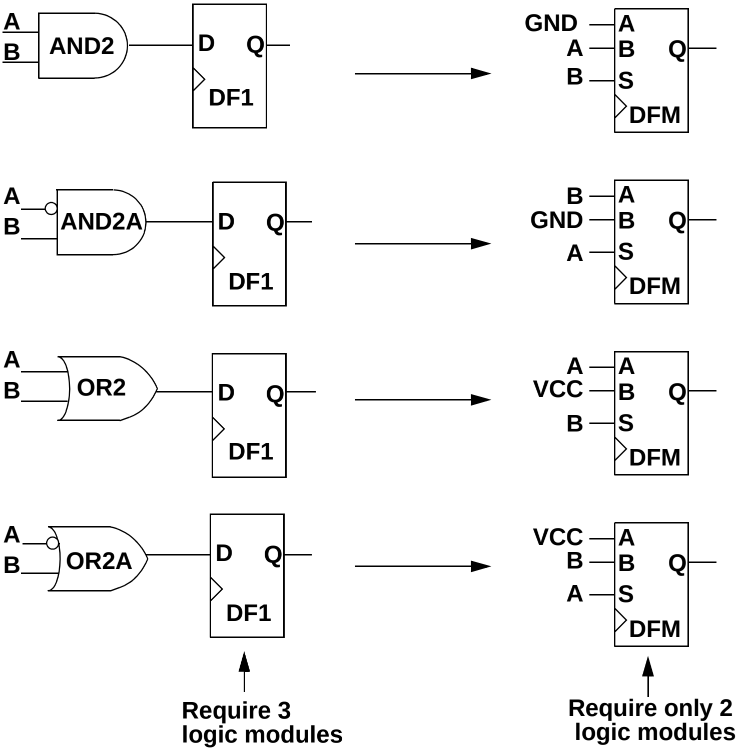

A
B
AND2
D
Q
DF1
GND
A
B
A
B
Q
S
DFM
A
B
AND2A
D
Q
DF1
B
GND
A
A
B
Q
S
DFM
A
B
OR2
D
Q
DF1
A
VCC
B
A
B
Q
S
DFM
A
B
OR2A
D
Q
DF1
VCC
B
A
A
B
Q
S
DFM
Require 3
logic modules
Require only 2
logic modules

Figure 3-11. 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/guru 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 3-12.

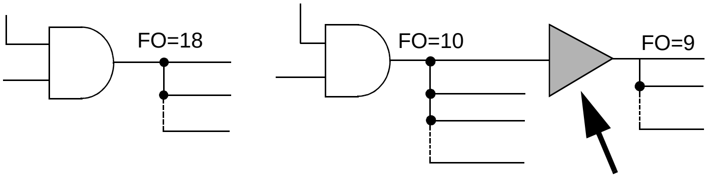


Figure 3-12. Buffering

An alternative method to buffering is duplicating logic to eliminate the buffer delay, shown in Figure 3-13.

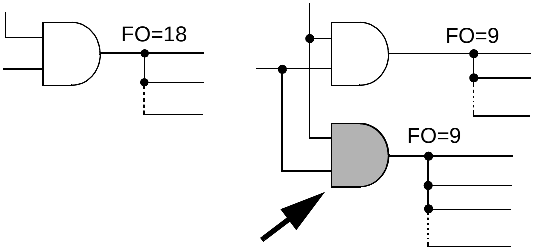


Figure 3-13. Duplicating Logic

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* and *Flash 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 Flash 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 3-14. 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.

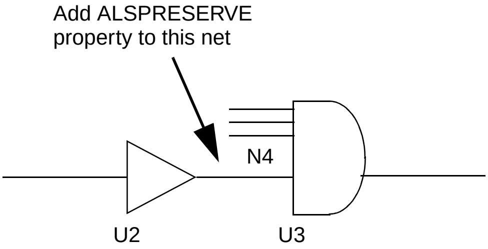


Figure 3-14. Adding ALSPRESERVE Property to a Net

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 3-15, 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 3-15). 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.

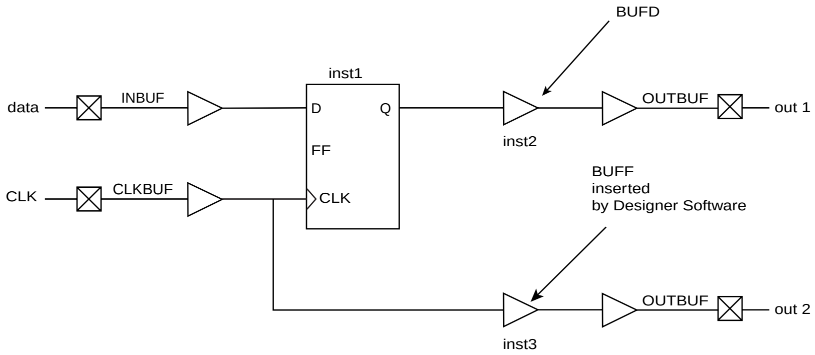

BUFD
inst1
data
INBUF
D
Q
OUTBUF
out 1
inst2
FF
BUFF
inserted
by Designer Software
CLK
CLKBUF
CLK
OUTBUF
out 2
inst3

Figure 3-15. Phase Relationship Control - Data and System Clock using BUFD

In Figure 3-16, 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.

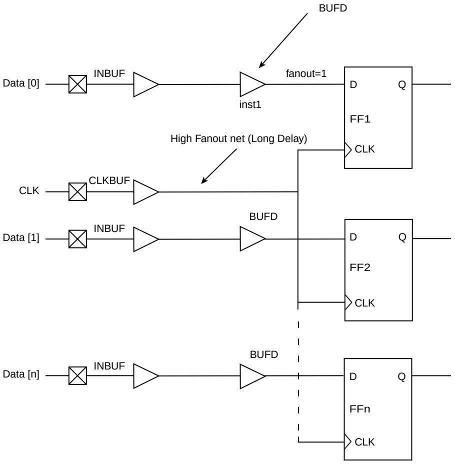

BUFD
Data [0]
INBUF
fanout=1
inst1
D
Q
FF1
High Fanout net (Long Delay)
CLK
CLK
CLKBUF
BUFD
Data [1]
INBUF
D
Q
FF2
CLK
BUFD
Data [n]
INBUF
D
Q
FFn
CLK

Figure 3-16. 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: Flash devices do not supprot 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 3-17. 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.

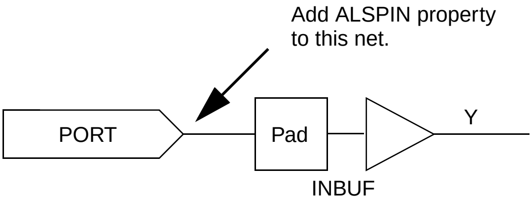


Figure 3-17. Adding ALSPIN Property to a Net

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 Compatability

- 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 3200DX, 42MX, 54SX and 54SX-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, 54SX-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.

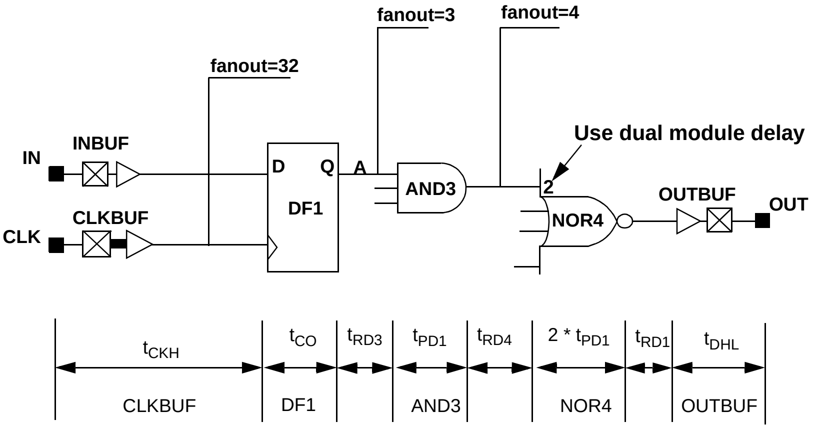
Entering Constraints for Timing Driven Place-and-Route

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 3-18 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.


fanout=3
fanout=4
fanout=32
Use dual module delay
IN
INBUF
CLKBUF
CLK
D
Q
A
DF1
AND3
2
NOR4
OUTBUF
OUT
t_{CKH}
t_{CO}
t_{RD3}
t_{PD1}
t_{RD4}
2 * t_{PD1}
t_{RD1}
t_{DHL}
CLKBUF
DF1
AND3
NOR4
OUTBUF

t_{CRT}(CLK to OUT) = t_{CKH}(CLKBUF)+t_{CO}(DF1)+t_{RD3}+t_{PD1}(AND3)+t_{RD4}+2 * t_{PD1}(NOR4)+t_{RD1}+t_{DHL}(OUTBUF)

= 5.8 + 3.0 + 1.8 + 3.0 + 2.3 + 2 * 3.0 + 0.9 + 4.6

= 27.4ns

Figure 3-18. Critical Path Timing Example

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

Adding ACTgen Macros

With the ACTgen Macro Builder, you can create macros 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 the ACTgen Macro Builder to to configure the PLLs, RAM, FIFO, etc., for Flash and Axcelerator devices.

4

Flash Design Considerations

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

Design Optimization

Use Designer report files and iterating tasks to optimize the design. When iterating only to optimize the design after a previously successful layout, all subsequent processing usually succeeds. If the design has only minor changes you can recycle the previous placement in an ECO mode and run placement refinement.

Optimizing Inverters

If you run Designer with Netlist Optimization enabled (default mode), inverters and buffers that do not limit fanout are removed from the design netlist. The removal of inverters is based on the use of internal inverted pins.

Timing Optimization

Timing optimization is performed by iterating the design analysis and layout cycle on a design until it meets timing and layout requirements. Refer to the Timer online help for additional information.

Design Integrity

Designer offers a combination of safeguards that protect the integrity of a design, and the flexibility to override the safeguards when necessary. Designer keeps track of a design project and ensures that the design flow is followed to maintain the integrity of the project database and to keep it synchronized with the design state.

By default the system performs tasks in the order shown in the task list, using the results from each step as input for the next step. Additionally, only tasks required to produce the stated outputs are performed and you are prompted for required inputs for each task.

Although Designer performs tasks in a specific order, the system can perform a subset of these operations to suit the current design state unless an operation violates one of the safeguards.

For example, if a design is placed and routed but post-layout delays violate timing requirements, Designer can be directed to only optimize place-and-route and recalculate the delays. File checking or any other task previously done are not repeated unless any of the input files have also changed between runs.

Designer processes a design incrementally by default. It keeps track of successfully completed tasks and does not repeat them. If data for a task are missing, the system either automatically supplies it or prompts the user to supply the needed information.

Design Hints

The design netlist is the main input for Designer. When creating a design, device usage and Designer features can be maximized by using the following guidelines:

- Use the automatic global resource assignments specified by Designer and, if appropriate, add constraints.
- Use the automatic mapping that Designer specifies between package pins and a design's I / O signals and, if appropriate, add constraints.
- Avoid over-constraining a design so that Designer can apply its algorithms to the best advantage for place-and-route.
- Implement tristates, memories, and hierarchical blocks with Designer-supported elements.
- Avoid asynchronous designs. They are difficult to verify, are less robust, and typically include problems such as hazard and race conditions.
- When gating clocks, ensure that the gate signals are stable when the clock is active. Otherwise, glitches might be propagated through the circuit.
- Use a minimum of logic before the first flip-flop or after the last flip-flop on signals that are to be driven on or off chip to provide as much time as possible to drive the signal off the chip and onto the next chip.
- Ensure that process variations do not adversely affect reliability, always design with worst-case processes in mind.

Implementing Memories

All Flash devices support embedded memories. Synchronous or asynchronous two port RAMs and FIFOs can be generated by ACTgen and imported into the synthesis and simulation environments. Designer also supports distributed memories. Although this implementation of memories is much less efficient in terms of gate utilization, it may be appropriate for smaller memories in a design.

5

Quick Start Tutorial

This tutorial illustrates a basic VHDL design for the APA Evaluation Board. The design is targeted at the Actel ProASIC$^{\underline{PLUS}}$ family. To show the design in its simplest form, a simple andgate design is created in Actel's Libero IDE v5.0. 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 a Timing Simulation with Back-Annotated Timing

Step 7 – Generate the Programming File

Step 8 – Program the Device

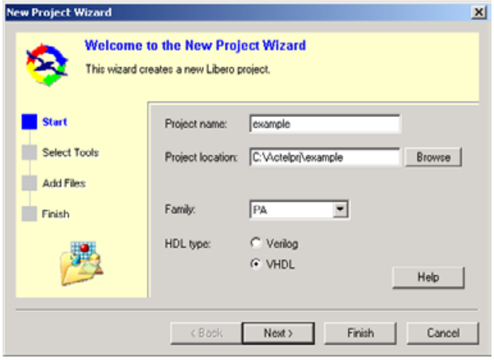
Step 1 – Create a New Project

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 2-1.


New Project Wizard
Start
Select Tools
Add Files
Finish
Browse
Help
< Back
Next >
Finish
Cancel

Welcome to the New Project Wizard

This wizard creates a new Libero project.

Project name: example

Project location: C:\Actelprj\example

Family: PA

HDL type: ○ Verilog ● VHDL

Figure 2-1. New Project Wizard in Libero IDE

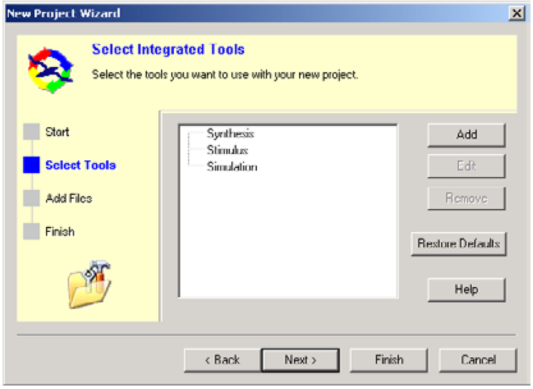
3. Enter your **Project name**. For this tutorial, name your project *example*.

4. In the **Project location** field, click **Browse** to navigate to C:\Actelprj.

5. Select your project **Family** from the drop-down list. For this tutorial, select **PA** (ProASIC^{PLUS}).

6. Select your **HDL type**. For this example, select **VHDL**.

7. Click **Next** to **Select Integrated Tools** in the New Project Wizard (Figure 2-2).


New Project Wizard
Start
Select Tools
Add Files
Finish
Synthesis
Stimulus
Simulation
Add
Edit
Remove
Restore Defaults
Help
< Back
Next >
Finish
Cancel

Select Integrated Tools

Select the tools you want to use with your new project.

Figure 2-2. Selected Integrated Tools in New Project Wizard (Libero IDE)

8. 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 2-3).

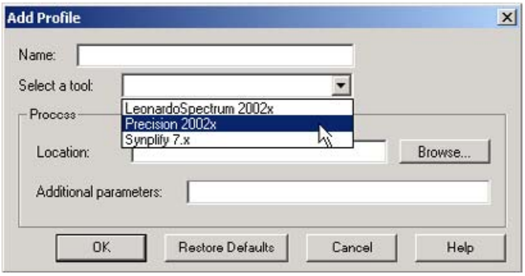
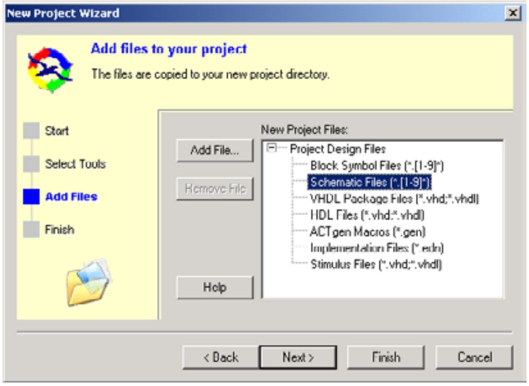

Add Profile
Name:
Select a tool:
LeonardoSpectrum 2002x
Precision 2002x
Synplify 7.x
Process
Location:
Browse...
Additional parameters:
OK
Restore Defaults
Cancel
Help

Figure 2-3. 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.

9. **Add Files** in the New Project Wizard to add existing Project Design Files, including any ACTgen macros or Block Symbol, Schematic, VHDL Package, HDL, Implementation, and Stimulus files (Figure 2-4).



Add files to your project

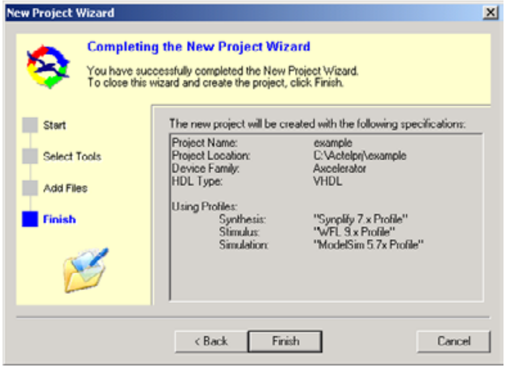
The files are copied to your new project directory.

New Project Wizard
Start
Select Tools
Add Files
Finish
Add File...
Remove File
Help
New Project Files:
Project Design Files
Block Symbol Files (*.[1-9]*)
Schematic Files (*.[1-9]*)
VHDL Package Files (*.vhd;*.vhdl)
HDL Files (*.vhd;*.vhdl)
ACTgen Macros (*.gen)
Implementation Files (*.edn)
Stimulus Files (*.vhd;*.vhdl)
< Back
Next >
Finish
Cancel

Figure 2-4. 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.

10. Review your project information. Click **Finish** to close the Wizard and create your new project (Figure 2-5). Click Back to return to any step of the Wizard and correct information in your project.



Completing the New Project Wizard

You have successfully completed the New Project Wizard.
To close this wizard and create the project, click Finish.

The new project will be created with the following specifications:

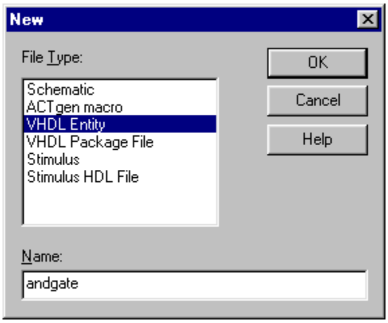
```
Project Name:           example
Project Location:       C:\Actelprj\example
Device Family:          Axcelerator
HDL Type:               VHDL

Using Profiles:
        Synthesis:      "Synplify 7.x Profile"
        Stimulus:       "WFL 9.x Profile"
        Simulation:     "ModelSim 5.7x Profile"
```

Figure 2-5. 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 macro, 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 2-6.



File Type:
- Schematic
- ACTgen macro
- VHDL Entity
- VHDL Package File
- Stimulus
- Stimulus HDL File

Name: andgate

Figure 2-6. New File Dialog Box

2. Select **VHDL Entity** in the File Type field, enter **andgate** 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.

```
-- AND Gate Tutorial for APA Evaluation Board
LIBRARY ieee;
USE ieee.std_logic_1164. ALL;


ENTITY andgate is
port (A, B : in std_logic;-- Data Inputs
OUTPUT : out std_logic); -- Output= A AND B
end andgate;


architecture behaviour of andgate is


begin


OUTPUT <= A AND B;


end behaviour;
```

3. From the **File** menu, click **Save**. The design file “andgate” appears in the **Design Hierarchy**. Libero IDE lists “andgate.vhd” under HDL files in the **File Manager**, as shown in Figure 2-7.

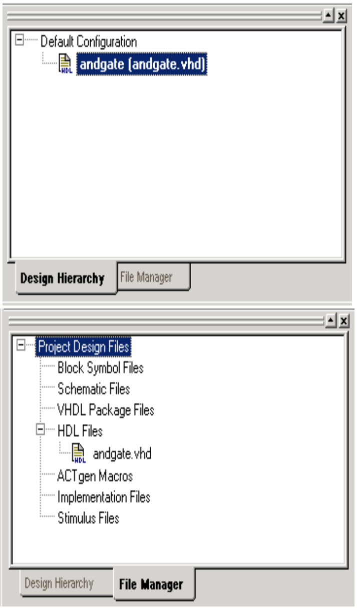

Default Configuration
andgate (andgate.vhd)
Design Hierarchy
File Manager
Project Design Files
Block Symbol Files
Schematic Files
VHDL Package Files
HDL Files
andgate.vhd
ACTgen Macros
Implementation Files
Stimulus Files
Design Hierarchy
File Manager

Figure 2-7. 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 **andgate.vhd** and select **Check HDL**. This checks the syntax of the **andgate.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

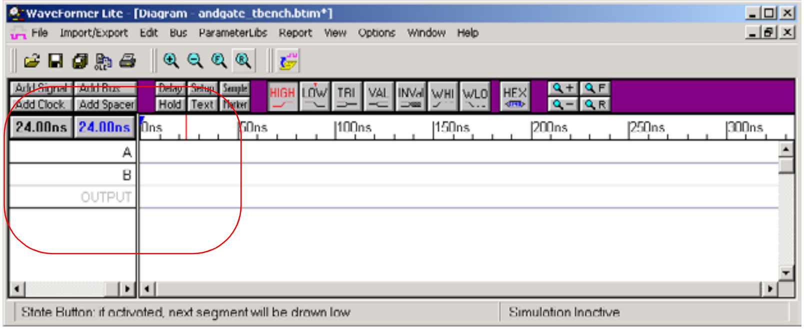
WaveFormer Lite generates VHDL testbenches from drawn waveforms. There are three basic steps for creating testbenches using WaveFormer Lite and the Actel Libero IDE software:

1. Import Signal Information
2. Drawing Waveforms
3. Export the Testbench

Import Signal Information

To launch WaveFormer Lite and import signal information:

1. Right-click the andgate.vhd file in the **Design Hierarchy** tab and select **Create Stimulus**. WaveFormer Lite launches with the port signals in the Diagram window, as shown in Figure 2-8.


WaveFormer Lite - [Diagram - andgate_tbench.btim*]
File
Import/Export
Edit
Bus
ParameterLibs
Report
View
Options
Window
Help
Add Signal
Add Bus
Add Clock
Add Spacer
Delay
Setup
Sample
Hold
Text
Marker
HIGH
LOW
TRI
VAL
INVal
WHI
WLO
HEX
24.00ns
24.00ns
0ns
50ns
100ns
150ns
200ns
250ns
300ns
A
B
OUTPUT

State Button: if activated, next segment will be drawn low

Simulation Inactive

Figure 2-8. WaveFormer Lite Timing Diagram Window

The andgate design contains the following signals:

- A Input signal
- B Input signal
- OUTPUT 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 2-9.

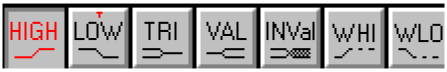


Figure 2-9. State Buttons

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. To activate a state button, click on 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:

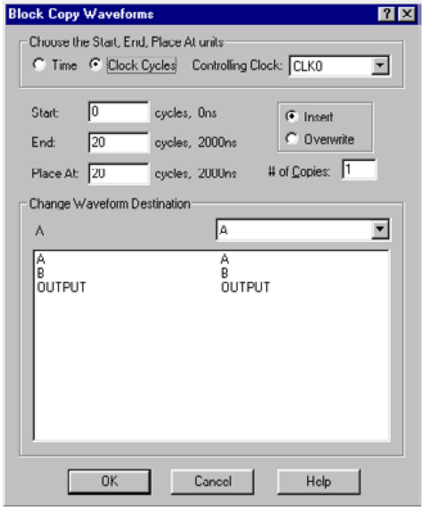
1. Select the **High** state and place the mouse cursor inside the **Diagram** window at the same vertical row as the signal name.
2. Click the mouse button. This draws a waveform from the end of the signal to the mouse cursor. The red state button on the button bar determines the type of waveform drawn. 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. This opens the **Block Copy Waveforms** dialog box with the selected signals displayed in the **Change Waveform Destination** list box.



Choose the Start, End, Place At units
○ Time ● Clock Cycles Controlling Clock: CLK0

Start: 0 cycles, 0ns
End: 20 cycles, 2000ns
Place At: 20 cycles, 2000ns
● Insert
○ Overwrite
# of Copies: 1

Change Waveform Destination
A A

A	A
B	B
OUTPUT	OUTPUT

Figure 2-10. Block Copy Waveforms Dialog Box

3. In the dialog, 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:

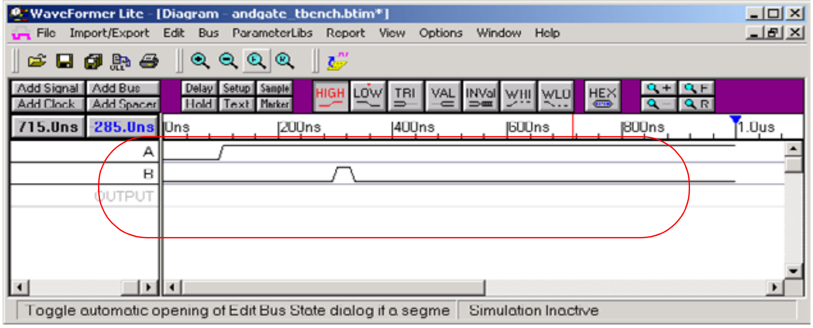
In this step, a design stimulus file is created using WaveFormer Lite. Following the instructions in the previous sections, define values for the A input signal (A), the B input signal (B), and the output signal (OUTPUT).

1. Following the instructions on the previous pages, create waveforms for A and B, as described below:

Table 2-1.

A –	low 0nS – 100 ns
	high 100 nS – 1 us
B –	low 0nS – 300 ns
	high 300 nS – 330 ns
	low 330 nS – 1 us

This creates the waveform shown in Figure 2-11.


WaveFormer Lite - [Diagram - andgate_tbench.btim*]
File
Import/Export
Edit
Bus
ParameterLibs
Report
View
Options
Window
Help
Add Signal
Add Bus
Add Clock
Add Spacer
Delay
Setup
Sample
Hold
Text
Marker
HIGH
LOW
TRI
VAL
INVal
WHI
WLO
HEX
715.0ns
285.0ns
0ns
200ns
400ns
600ns
800ns
1.0us
A
B
OUTPUT

Toggle automatic opening of Edit Bus State dialog if a segme

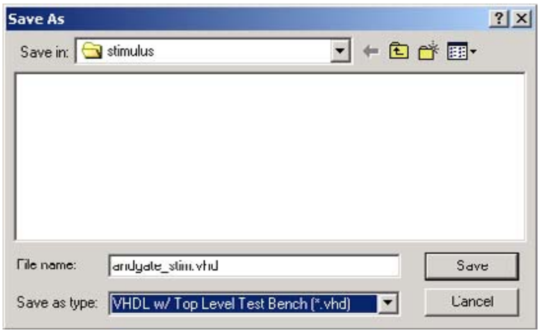
Simulation Inactive

Figure 2-11. WaveForm Timing diagram

2. After successfully creating the waveforms, select **Save As** from the **File** menu. In the **Save As** dialog box, enter andgate_stim.btim as the file name and click **Save**.

3. After saving the timing diagram file, select **Export Timing Diagram As** from the **Export** menu.

4. Select **VHDL w/ Top Level Testbench** in **Files of Type** and enter andgate_stim.vhd for the file name, as shown in Figure 2-12.


Save As
?
Save
Cancel

Save in: stimulus

File name: andgate_stim.vhd

Save as type: VHDL w/ Top Level Test Bench (*.vhd)

Figure 2-12. Export VHDL Testbench Save As Dialog Box

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

5. 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 andgate_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. Select a stimulus file. Right-click andgate in the **Design Hierarchy** tab and choose **Select a Stimulus File**, as shown in Figure 2-13.

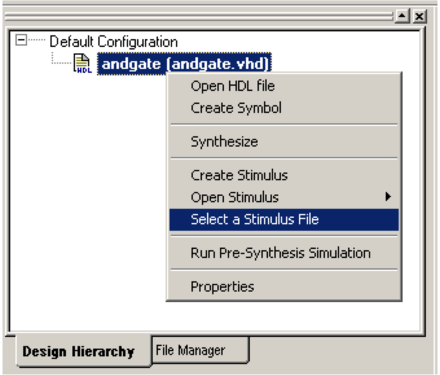
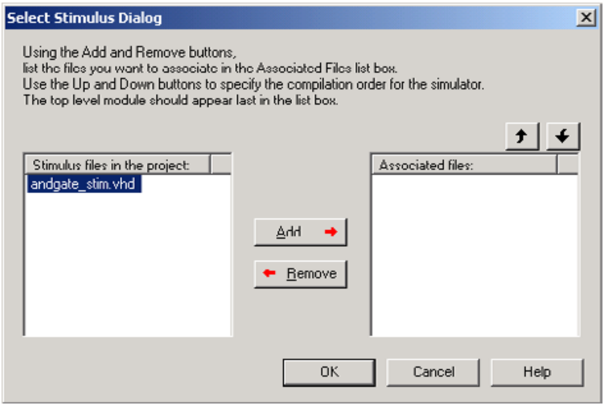

Default Configuration
andgate (andgate.vhd)
Open HDL file
Create Symbol
Synthesize
Create Stimulus
Open Stimulus
Select a Stimulus File
Run Pre-Synthesis Simulation
Properties
Design Hierarchy
File Manager

Figure 2-13. Selecting a Stimulus File

The **Select Stimulus** dialog box appears (Figure 2-14).



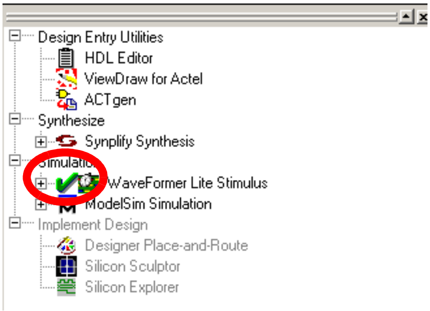
Using the Add and Remove buttons,
list the files you want to associate in the Associated Files list box.
Use the Up and Down buttons to specify the compilation order for the simulator.
The top level module should appear last in the list box.

Select Stimulus Dialog
Stimulus files in the project:
andgate_stim.vhd
Associated files:
Add
Remove
OK
Cancel
Help

Figure 2-14. Select Stimulus Dialog Box

2. Select andgate_stim.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 2-15.



Design Entry Utilities
HDL Editor
ViewDraw for Actel
ACTgen
Synthesize
Synplify Synthesis
Simulation
WaveFormer Lite Stimulus
ModelSim Simulation
Implement Design
Designer Place-and-Route
Silicon Sculptor
Silicon Explorer

Figure 2-15. Check Mark in Waveformer Lite

4. Double-click the **ModelSim** simulation icon in the **Process** window, or right-click andgate in the Design Hierarchy tab and select **Run Pre-synthesis Simulation**, as shown in Figure 2-16.

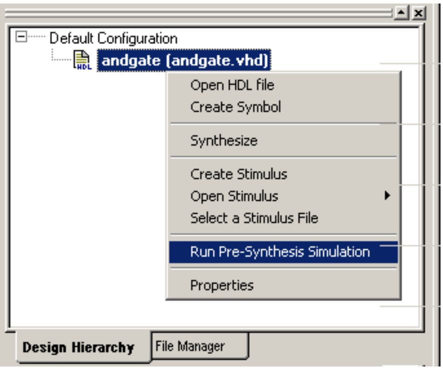
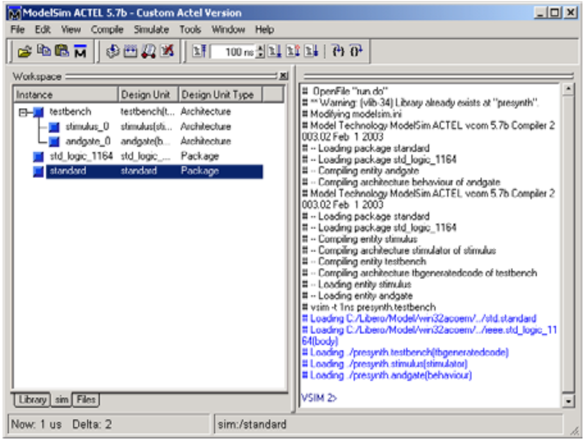

Default Configuration
andgate (andgate.vhd)
Open HDL file
Create Symbol
Synthesize
Create Stimulus
Open Stimulus
Select a Stimulus File
Run Pre-Synthesis Simulation
Properties
Design Hierarchy
File Manager

Figure 2-16. Run Pre-Synthesis Simulation

The ModelSim VHDL simulator opens and compiles the source files, as shown in Figure 2-17.


ModelSim ACTEL 5.7b - Custom Actel Version
File
Edit
View
Compile
Simulate
Tools
Window
Help
100 ns
Workspace
Library
sim
Files

Instance	Design Unit	Design Unit Type
testbench	testbench(t...	Architecture
stimulus_0	stimulus(sti...	Architecture
andgate_0	andgate(b...	Architecture
std_logic_1164	std_logic_...	Package
standard	standard	Package

```
# OpenFile "run.do"
# ** Warning: (vlib-34) Library already exists at "presynth".
# Modifying modelsim.ini
# Model Technology ModelSim ACTEL vcom 5.7b Compiler 2
003.02 Feb  1 2003
# -- Loading package standard
# -- Loading package std_logic_1164
# -- Compiling entity andgate
# -- Compiling architecture behaviour of andgate
# Model Technology ModelSim ACTEL vcom 5.7b Compiler 2
003.02 Feb  1 2003
# -- Loading package standard
# -- Loading package std_logic_1164
# -- Compiling entity stimulus
# -- Compiling architecture stimulator of stimulus
# -- Compiling entity testbench
# -- Compiling architecture tbgeneratedcode of testbench
# -- Loading entity stimulus
# -- Loading entity andgate
# vsim -t 1ns presynth.testbench
# Loading C:/Libero/Model/win32acoem/../std.standard
# Loading C:/Libero/Model/win32acoem/../ieee.std_logic_11
64(body)
# Loading ../presynth.testbench(tbgeneratedcode)
# Loading ../presynth.stimulus(stimulator)
# Loading ../presynth.andgate(behaviour)

VSIM 2>
```

Now: 1 us   Delta: 2   sim:/standard

Figure 2-17. 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 2-18, opens to display the simulation results. Scroll in the **Wave** window to verify that the design functions properly.

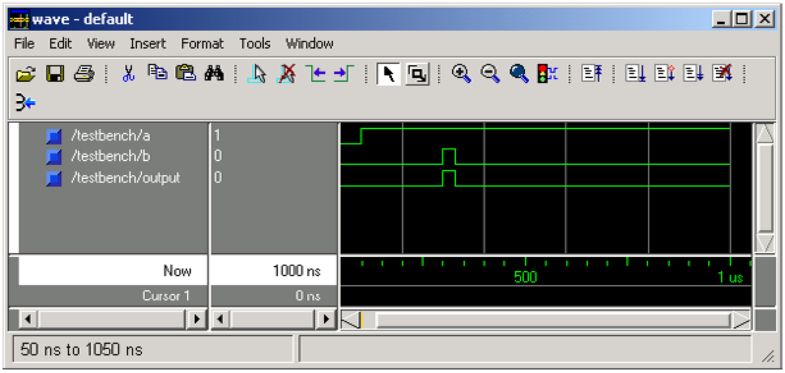

wave - default
File
Edit
View
Insert
Format
Tools
Window
/testbench/a
/testbench/b
/testbench/output
1
0
0
Now
1000 ns
Cursor 1
0 ns
500
1 us
50 ns to 1050 ns

Figure 2-18. ModelSim Wave Window

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

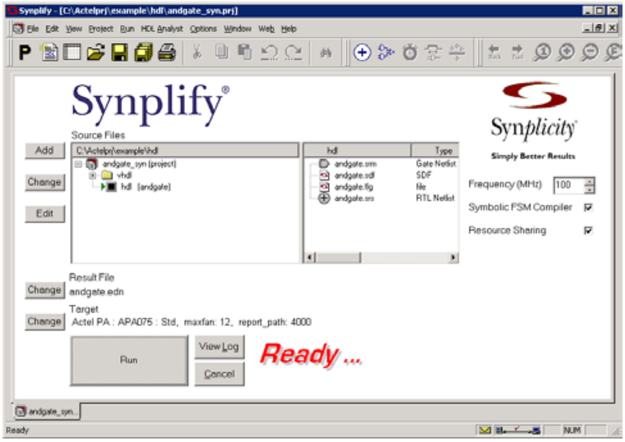
Step 3 – Synthesize the Design in Synplify

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 andgate.vhd file in the Design

Hierarchy and select **Synthesize**. This launches the Synplify synthesis tool with the appropriate design files, as shown in Figure 2-19.


Synplify - [C:\Actelprj\example\hdl\andgate_syn.prj]
File Edit View Project Run HDL Analyst Options Window Web Help
Synplify
Synplicity
Simply Better Results
Add
Change
Edit
Run
View Log
Cancel
Ready ...

Source Files

C:\Actelprj\example\hdl
- andgate_syn (project)
  - vhdl
  - hdl (andgate)

hdl	Type
andgate.srm	Gate Netlist
andgate.sdf	SDF
andgate.flg	file
andgate.srs	RTL Netlist

Frequency (MHz) 100

Symbolic FSM Compiler ☑

Resource Sharing ☑

Result File

andgate.edn

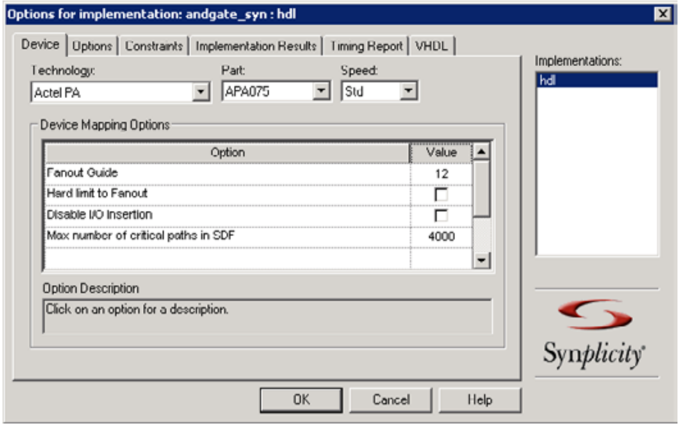
Target

Actel PA : APA075 : Std, maxfan: 12, report_path: 4000

Ready

Figure 2-19. Synplify

2. From the **Project** menu, select **Implementation Options**. This displays the **Options for Implementation** dialog box, as shown in Figure 2-20.


Options for implementation: andgate_syn : hdl
Device
Options
Constraints
Implementation Results
Timing Report
VHDL
Implementations:
Synplicity
OK
Cancel
Help

Technology: Actel PA

Part: APA075

Speed: Std

Device Mapping Options

Option	Value
Fanout Guide	12
Hard limit to Fanout	☐
Disable I/O Insertion	☐
Max number of critical paths in SDF	4000

Option Description

Click on an option for a description.

Implementations: hdl

Figure 2-20. Options for Implementation Dialog Box

3. Set the following in the dialog box:
    - **Technology:** Actel PA (Set by Libero IDE)
    - **Part:** APA075
    - **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 andgate.edn. The resulting andgate.edn file is then automatically translated by Libero into a VHDL netlist called andgate.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 andgate.prj in Synplify.

Step 4 – Perform Post-Synthesis Simulation

The next step is simulating the VHDL netlist of the andgate using the VHDL testbench created in “To create a stimulus file and generate a VHDL testbench:” on page 69.

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

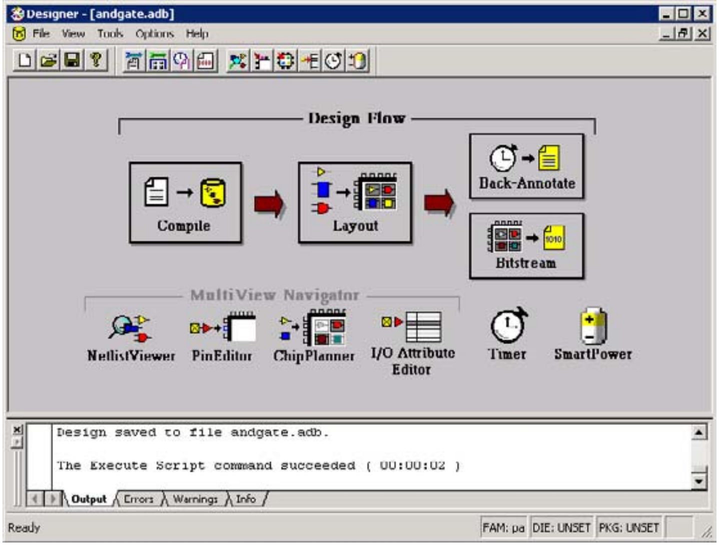
    Once the compilation completes, the simulator runs for 1000 ns and a Wave window opens to display the simulation results.

2. Scroll in the Wave window to verify that the andgate 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.

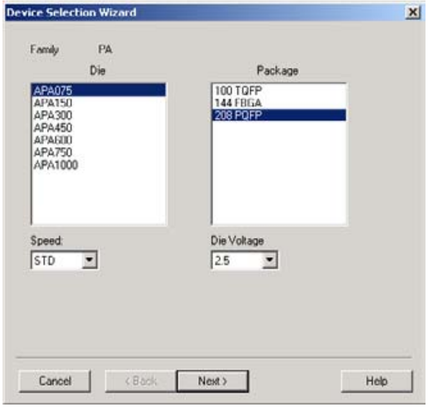
1. Double-click Designer **Place-and-Route** in the Libero IDE Process window, or right-click andgate.edn in the Design Hierarchy and select ***Run Designer.*** Designer opens and reads in the design file.


Designer - [andgate.adb]
File
View
Tools
Options
Help
Design Flow
Compile
Layout
Back-Annotate
Bitstream
MultiView Navigator
NetlistViewer
PinEditor
ChipPlanner
I/O Attribute Editor
Timer
SmartPower
Output
Errors
Warnings
Info
Ready
FAM: pa
DIE: UNSET
PKG: UNSET

```
Design saved to file andgate.adb.

The Execute Script command succeeded ( 00:00:02 )
```

Figure 2-21. Designer

2. Compile the design. Click **Compile** in the **Design Flow** window. This starts the DSW (**Device Selection Wizard**) shown in Figure 2-22.


Device Selection Wizard
Die
Package
APA150
APA300
APA450
APA600
APA750
APA1000
100 TQFP
144 FBGA
Cancel
< Back
Next >
Help

Family PA

Die APA075

Package 208 PQFP

Speed: STD

Die Voltage 2.5

Figure 2-22. Device Selection Wizard

Select **APA075** in the Die field and select **208 PQFP** in the package field. Accept the default speed grade and die voltage and click **Next**.

Complete the remaining fields and click **Finish**. Click the **Compile** icon.

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.

3. Pin Editing

Once the design compiles successfully, use the **PinEditor** tool to drag and drop the placement of pins and fix pin locations for subsequent place-and-route runs.

Click the **PinEditor** user tool. This opens the **MultiView Navigator** interface (Figure 2-23). See the Libero or Designer online help for more information on PinEditor or MultiView Navigator.

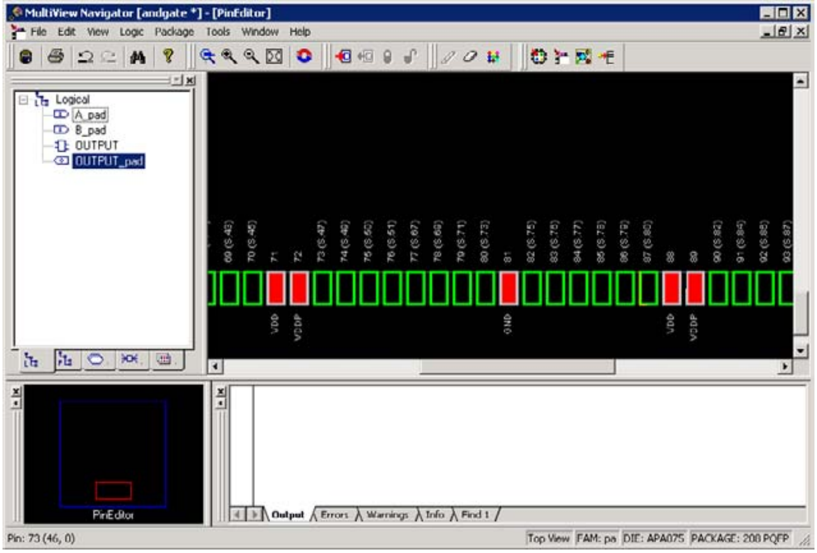

MultiView Navigator [andgate *] - [PinEditor]
File
Edit
View
Logic
Package
Tools
Window
Help
Logical
A_pad
B_pad
OUTPUT
OUTPUT_pad
71
72
81
88
89
VDD
VDDP
GND
VDD
VDDP
PinEditor
Output
Errors
Warnings
Info
Find 1
Pin: 73 (46, 0)
Top View
FAM: pa
DIE: APA075
PACKAGE: 208 PQFP

Figure 2-23. PinEditor Window in MultiView Navigator

Assign A to pin 55, B to pin 63, and the output to pin 87, as shown in Table 2-2.

Table 2-2.Pin Assignments

Signal	Direction	PIN
A	Input	55 (SW1)
B	Input	63 (SW1)
OUTPUT	Output	87 (LED DS1)

Click and drag each pin to the proper location, as shown in Figure 2-24.

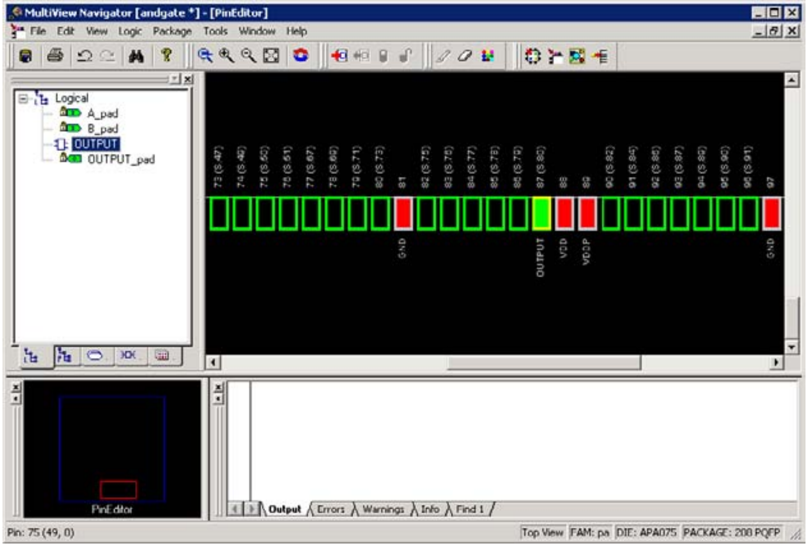

MultiView Navigator [andgate *] - [PinEditor]
File
Edit
View
Logic
Package
Tools
Window
Help
Logical
A_pad
B_pad
OUTPUT
OUTPUT_pad
73 (S 47)
74 (S 48)
75 (S 50)
76 (S 51)
77 (S 67)
78 (S 69)
79 (S 71)
80 (S 73)
81
82 (S 75)
83 (S 76)
84 (S 77)
85 (S 78)
86 (S 79)
87 (S 80)
88
89
90 (S 82)
91 (S 84)
92 (S 86)
93 (S 87)
94 (S 89)
95 (S 90)
96 (S 91)
97
GND
OUTPUT
VDD
VDDP
GND
PinEditor
Output
Errors
Warnings
Info
Find 1
Pin: 75 (49, 0)
Top View
FAM: pa
DIE: APA075
PACKAGE: 208 PQFP

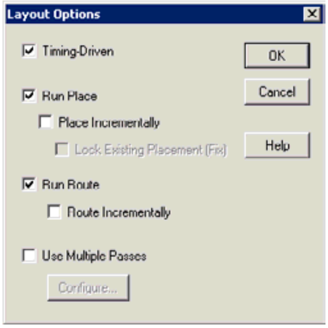
Figure 2-24. PinEditor with Pins Assigned

Once a pin number is assigned to all of the signals, select **Commit** from the **File** menu and close the PinEditor window.

4. (Optional) After successfully compiling the design, use the Designer Tools to view pre-layout static timing analysis with **Timer**, set time constraints in Timer, analyzes static and dynamic power with **SmartPower**, and use **ChipEditor** 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. For this tutorial, you do not need to make any changes to the design after you fix your pins in PinEditor.

5. Layout the Design. From **Designer**, click **Layout**. This opens the **Layout Options** dialog box shown in Figure 2-25.



☑ Timing-Driven

☑ Run Place

- ☐ Place Incrementally
  - ☐ Lock Existing Placement (Fix)

☑ Run Route

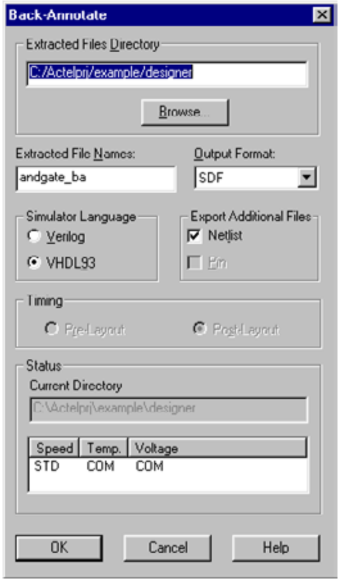
- ☐ Route Incrementally

☐ Use Multiple Passes

Figure 2-25. 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.

6. Back-Annotate the design. From **Designer**, click **Back-Annotate** in the **Design Flow** window. This opens the **Back-Annotate** dialog box shown in Figure 2-26.



Extracted Files Directory: C:/Actelprj/example/designer

Extracted File Names: andgate_ba

Output Format: SDF

Simulator Language: ○ Verilog ● VHDL93

Export Additional Files: ☑ Netlist ☐ Pin

Timing: ○ Pre-Layout ● Post-Layout

Status

Current Directory: C:\Actelprj\example\designer

Speed	Temp.	Voltage
STD	COM	COM

Figure 2-26. Back-Annotate Dialog Box

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

7. 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 andgate.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 a 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 andgate file in the Design Hierarchy and select **Run Post-Layout Simulation**.

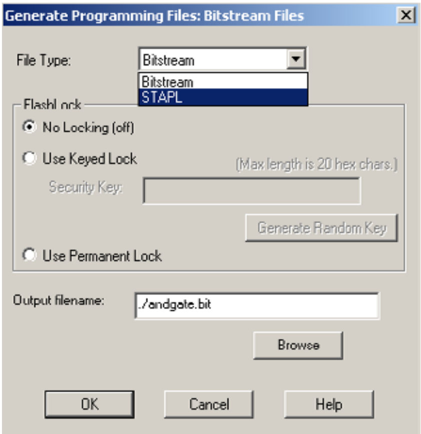
   This launches the ModelSim Simulator that compiles the back annotated VHDL netlist file and testbench. Once the compilation completes, the simulator runs for 1000 ns and a Wave window opens to display the simulation results.

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

Step 7 – Generate the Programming File

This step generates the necessary file for programming the ProASIC$^{\underline{PLUS}}$ APA Evaluation Board. The APA Evaluation Board is available from Actel. Visit the Actel website (http://www.actel.com) for more information.

1. Right-click **andgate** in the **Design Hierarchy** tab to open **Designer**.
2. Generate a Bitstream. Click **Bitstream** in the **Design Flow** window or from the **File** menu, select **Export > Programming files**.This opens the **Generate Programming Files: Bitstream Files** dialog box, as shown in Figure 2-27.



Generate Programming Files: Bitstream Files
File Type:
Bitstream
Bitstream
STAPL
FlashLock
● No Locking (off)
○ Use Keyed Lock
(Max length is 20 hex chars.)
Security Key:
Generate Random Key
○ Use Permanent Lock
Output filename:
./andgate.bit
Browse
OK
Cancel
Help

Figure 2-27. Bitstream Files Dialog Box

3. Select **STAPL** from the **File Type** drop-down list box.
4. Select an **Output File Name**. Enter the andgate file name or select the andgate file by clicking the **Browse** button. The **Bitstream** file dialog box appears. Click **OK**.

   On successful completion, the Bitstream icon turns green. The programming file is saved to the Libero IDE; it appears in the File Manager under **Implementation Files**.

   Note: The STAPL file header contains the security key.

Step 8 – Program the Device

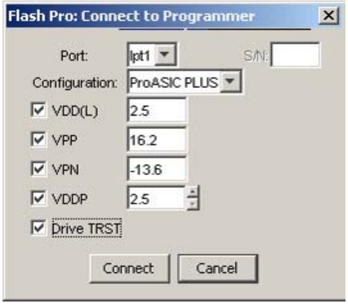
After generating the programming file, program the device using Actel's FlashPro Lite programmer.

Initial Setup

Before performing any action with the FlashPro programmer, it must be properly set up. Please properly connect the FlashPro ribbon cable with the programming header and turn on the switch.

To setup FlashPro Lite:

1. From the **File** menu, click **Connect**. The **FlashPro: Connect to Programmer** dialog box displays, as shown in Figure 2-28.


Flash Pro: Connect to Programmer

Port: lpt1

S/N:

Configuration: ProASIC PLUS

☑ VDD(L) 2.5

☑ VPP 16.2

☑ VPN -13.6

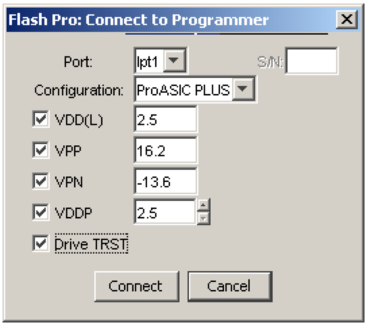
☑ VDDP 2.5

☑ Drive TRST

Figure 2-28. FlashPro: Connect to Programmer Dialog Box

2. In the **Port** list, select the port the FlashPro programmer is connected to.
3. In the **Configuration** list, select ProASIC$^{\underline{\text{PLUS}}}$. FlashPro Lite does not support ProASIC devices.

4. Disable voltages from the programmer if they are available on the board.



Flash Pro: Connect to Programmer

Port: lpt1

S/N:

Configuration: ProASIC PLUS

☑ VDD(L) 2.5

☑ VPP 16.2

☑ VPN -13.6

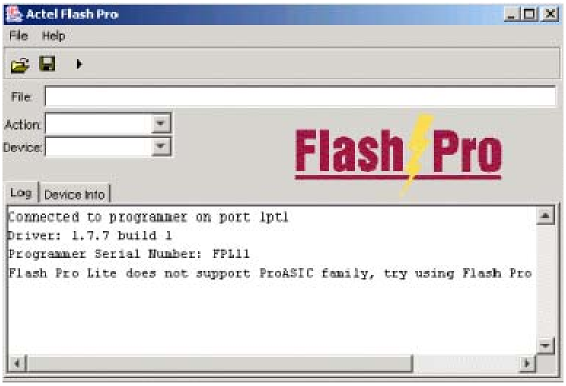
☑ VDDP 2.5

☑ Drive TRST

Figure 2-29. Connect to Programmer Dialog Box for ProASIC Devices

Note: To power-up the device from the board power supply, please deselect **VDD(L)** and **VDDP**. **VPP** and **VPN** are required during programming only and are supplied by the FlashPro programmer. Programming of ProASIC devices requires that **VDD(L)** is at 0 volts during programming. The board power supply design must allow for this if it is used to power-up the device during programming. ProASIC$^{\text{PLUS}}$ devices do not have this requirement.

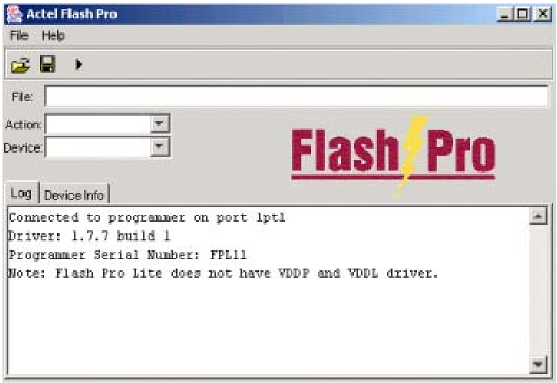
FlashPro Lite only supports ProASIC$^{\text{PLUS}}$ devices as shown in the FlashPro Lite Log window in Figure 2-30. Use FlashPro to program ProASIC devices.


Actel Flash Pro
File Help
File:
Action:
Device:
Flash Pro
Log
Device Info

```
Connected to programmer on port lpt1
Driver: 1.7.7 build 1
Programmer Serial Number: FPL11
Flash Pro Lite does not support ProASIC family, try using Flash Pro
```

Figure 2-30. FlashPro Lite Log Window

5. Click **Connect**. A successful connect, or any errors, appear in the Log window, as shown in Figure 2-31.


Actel Flash Pro
File Help
File:
Action:
Device:
Flash Pro
Log
Device Info

```
Connected to programmer on port lpt1
Driver: 1.7.7 build 1
Programmer Serial Number: FPL11
Note: Flash Pro lite does not have VDDP and VDDL driver.
```

Figure 2-31. FlashPro Lite Successful Connection

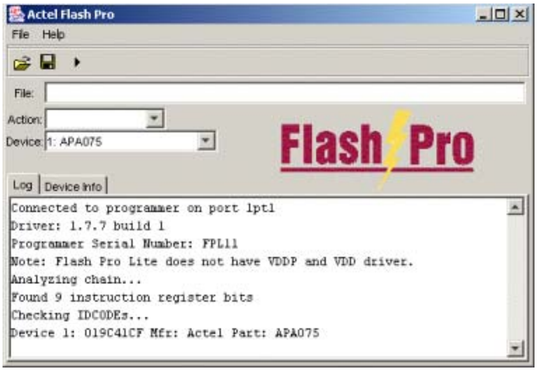
Note: FlashPro Lite does not have VDDP and VDD drivers.

Analyze Chain and Device Selection

Double-click the FlashPro icon to start the program. Then use FlashPro to analyze your chain and select your device.

To analyze the chain and select the device:

1. From the File menu, click **Analyze Chain**. Chain details appear in the **Log** window, as shown in Figure 2-32. If any failures appear, refer to the error and troubleshooting section of the *FlashPro User's Guide* at: http://www.actel.com/documents/flashproUG.pdf



```
Connected to programmer on port lpt1
Driver: 1.7.7 build 1
Programmer Serial Number: FPL11
Note: Flash Pro Lite does not have VDDP and VDD driver.
Analyzing chain...
Found 9 instruction register bits
Checking IDCODEs...
Device 1: 019C41CF Mfr: Actel Part: APA075
```

Figure 2-32. FlashPro: Analyzing Chain

2. Select the **APA 075** device from the **Device** list. If only one device is present in the chain, performing **Analyze Chain** selects that device automatically from the Device list.

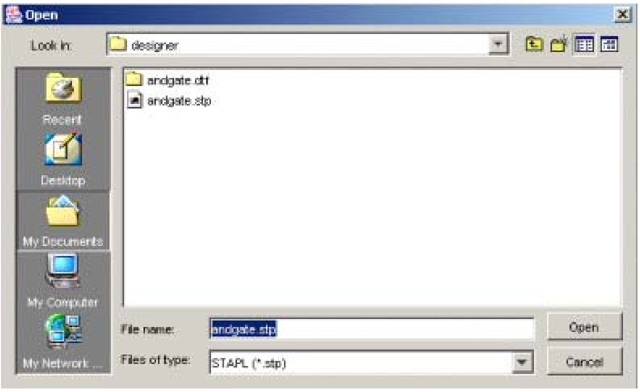
Loading the STAPL File

FlashPro Lite programmer uses a STAPL (*.stp) file to program the device.

To load the STAPL file:

1. Click the **Open File** button in the toolbar, or from the **File** menu, click **Open STAPL file**.

The **Open** dialog box appears, as shown in Figure 2-33.


Open
Recent
Desktop
My Documents
My Computer
My Network ...
andgate.dtf
andgate.stp
Open
Cancel

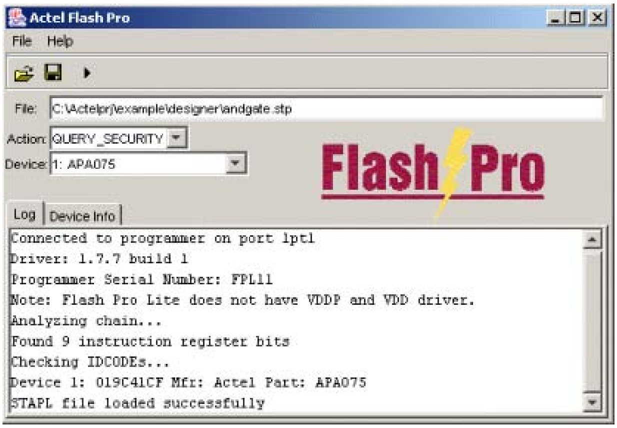
Look in: designer

File name: andgate.stp

Files of type: STAPL (*.stp)

Figure 2-33. Open Dialog Box

2. Browse to the root folder of the project, select the STAPL file, and click **Open**. The FlashPro software loads the file. The FlashPro **Log** window displays a message indicating that the software has successfully loaded, as shown in Figure 2-34.


Actel Flash Pro
File
Help
Flash Pro
Log
Device Info

File: C:\Actelprj\example\designer\andgate.stp

Action: QUERY_SECURITY

Device: 1: APA075

```
Connected to programmer on port lpt1
Driver: 1.7.7 build 1
Programmer Serial Number: FPL11
Note: Flash Pro Lite does not have VDDP and VDD driver.
Analyzing chain...
Found 9 instruction register bits
Checking IDCODEs...
Device 1: 019C41CF Mfr: Actel Part: APA075
STAPL file loaded successfully
```

Figure 2-34. STAPL File Loaded Successfully

Selecting an Action

After loading the STAPL file, select an action from the Action list. See Table 2-3 for a definition of each action.

Table 2-3. Action Options

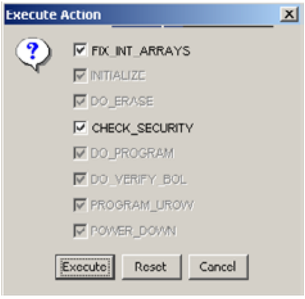
Option	Action
QUERY_SECURITY	Checks for security feature. If the device is programmed with the security feature, then this command exists with Read inhibit:1 Write inhibit:1. If the security feature is not present, the values are Read inhibit:0 Write inhibit 0.
ERASE	Erases the device.
READ_IDCODE	Reads the device ID code.
VERIFY	Verifies whether the device was programmed with the loaded STAPL file. If the wrong STAPL file is loaded, an Exit 11 result appears in the log window. A successful operation results in Exit 0. This command resembles the checksum command of antifuse product's programming.
PROGRAM	Programs the device.
DEVICE_INFO	Displays the serial number of the device, the Design Name that is programmed into the device, and the checksum that is programmed into the device.

Programming the Device

To program the device:

1. In the **Action** list, select **PROGRAM**.
2. In the **Device** list, select the **APA 075** device.
3. Click the **Execute** button in the toolbar.

The **Execute Action** dialog box appears, as shown in Figure 2-35.

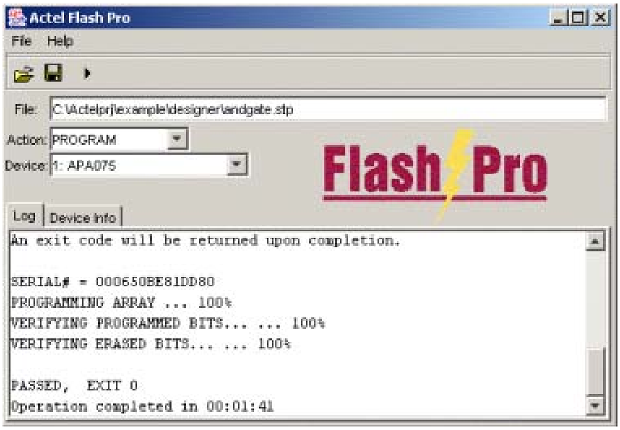


☑ FIX_INT_ARRAYS
☑ INITIALIZE
☑ DO_ERASE
☑ CHECK_SECURITY
☑ DO_PROGRAM
☑ DO_VERIFY_BOL
☑ PROGRAM_UROW
☑ POWER_DOWN

Figure 2-35. Execute Action Dialog Box

All the steps of the programming sequence are listed. Optional steps appear in bold. Grayed out options are required for programming and cannot be changed.

4. Make the required selections and click **Execute** to start programming. The progress of the programming action displays in the **Log** window. The message **Exit 0** indicates that the device has been programmed successfully, as shown in Figure 2-36.



File: C:\Actelprj\example\designer\andgate.stp
Action: PROGRAM
Device: 1: APA075

```
An exit code will be returned upon completion.

SERIAL# = 000650BE81DD80
PROGRAMMING ARRAY ... 100%
VERIFYING PROGRAMMED BITS... ... 100%
VERIFYING ERASED BITS... ... 100%

PASSED,  EXIT 0
Operation completed in 00:01:41
```

Figure 2-36. Successfully Programmed Device

Note: Do not interrupt the programming sequence, it may damage the device or programmer.

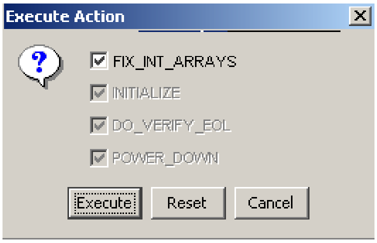
If you encounter any failures, please refer to the troubleshooting section of the *FlashPro User's Guide*.

Verifying the Correct Programming

To verify the device is programmed with the correct STAPL file:

1. Load the STAPL file.
2. In the **Action** list, click **Verify**.
3. Click the **Execute** button in the toolbar.

   The **Execute Action** dialog box appears, as shown in Figure 2-37.



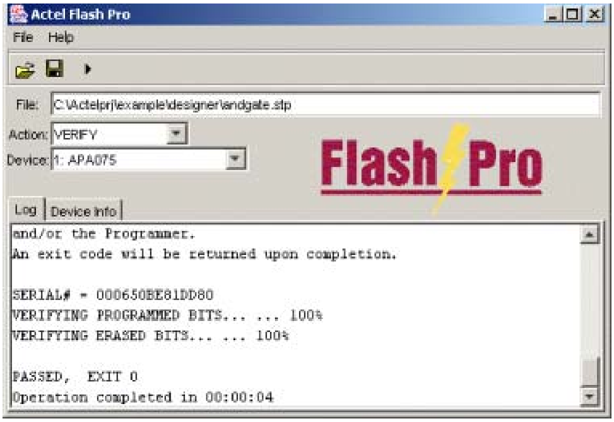
☑ FIX_INT_ARRAYS
☑ INITIALIZE
☑ DO_VERIFY_EOL
☑ POWER_DOWN

Figure 2-37. Execute Action Dialog Box

The default settings appear in the **Execute Action** dialog box.

4. Click **Execute** to start the verification process. A successful verification results in Exit 0, as shown in Figure 2-38. If the STAPL file is different from the file used for programming, **Exit 11** appears in the Log window.

Note: Do not interrupt the programming sequence, it may damage the device.



File: C:\Actelprj\example\designer\andgate.stp

Action: VERIFY

Device: 1: APA075

```
and/or the Programmer.
An exit code will be returned upon completion.

SERIAL# = 000650BE81DD80
VERIFYING PROGRAMMED BITS... ... 100%
VERIFYING ERASED BITS... ... 100%

PASSED,  EXIT 0
Operation completed in 00:00:04
```

Figure 2-38. Successful Verification

Saving Your Log File

All FlashPro results are displayed in the Log window. Save these results into a file.

A

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.

Actel U.S. Toll-Free Line

Use the Actel toll-free line to contact Actel for sales information, technical support, requests for literature, Customer Service, investor information, and using the Action Facts service.

The Actel toll-free line is (888) 99-ACTEL.

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 (408) 522-4480.
From Southeast and Southwest U.S.A., call (408) 522-4480.
From South Central U.S.A., call (408) 522-4434.
From Northwest U.S.A., call (408) 522-4434.
From Canada, call (408) 522-4480.
From Europe, call (408) 522-4252 or +44 (0) 1276 401500.
From Japan, call (408) 522-4743.
From the rest of the world, call (408) 522-4743.
Fax, from anywhere in the world (408) 522-8044.

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.

Guru Automated Technical Support

Guru is a web-based automated technical support system accessible through the Actel home page (http://www.actel.com/guru/). Guru provides answers to technical questions about Actel products. Many answers include diagrams, illustrations, and links to other resources on the Actel web site.

Web Site

Actel has a World Wide Web home page where you can browse a variety of technical and non-technical information. The URL is http://www.actel.com. For information on IP products, please visit http://www.actel.com/products/ip.

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:

Electronic Mail

You can communicate your technical questions to our e-mail address and receive answers back by e-mail, fax, or phone. Also, if you have design problems, you can e-mail your design files to receive assistance. We constantly monitor the e-mail 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 e-mail address is **tech@actel.com**.

Telephone

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:

(408) 522-4460
(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. Please see our list of Worldwide Sales Offices.

Worldwide Sales Offices

Headquarters

Actel Corporation
955 East Arques Avenue
Sunnyvale, California 94086
Toll Free: 888.99.ACTEL

Tel: 408.739.1010
Fax: 408.739.1540

US Sales Offices

California

Bay Area
Tel: 408.328.2200
Fax: 408.328.2358

Irvine
Tel: 949.727.0470
Fax: 949.727.0476

Newbury Park
Tel: 805.375.5769
Fax: 805.375.5749

Colorado

Tel: 303.420.4335
Fax: 303.420.4336

Florida

Tel: 407.977.6846
Fax: 407.977.6847

Georgia

Tel: 770.277.4980
Fax: 770.277.5896

Illinois

Tel: 847.259.1501
Fax: 847.259.1575

Massachusetts

Tel: 978.244.3800
Fax: 978.244.3820

Minnesota

Tel: 651.917.9116
Fax: 651.917.9114

New Jersey

Tel: 609.517.0304

North Carolina

Tel: 919.654.4529
Fax: 919.674.0055

Pennsylvania

Tel: 215.830.1458
Fax: 215.706.0680

Texas

Tel: 972.235.8944
Fax: 972.235.965

International Sales Offices

Canada

235 Stafford Rd. West,
Suite 106
Nepean, Ontario K2H 9C1

Tel: 613.726.7575
Fax: 613.726.8666

France

Actel Europe S.A.R.L.
361 Avenue General de Gaulle
92147 Clamart Cedex

Tel: +33 (0)1.40.83.11.00
Fax: +33 (0)1.40.94.11.04

Germany

Lohweg 27
85375 Neufahrn

Tel: +49 (0)8165.9584.0
Fax: +49 (0)8165.9584.1

Italy

Via de Garibaldini, No. 5
20019 Settimo Milanese,
Milano, Italy

Hong Kong

39th Floor
One Pacific Place
88 Queensway
Admiralty, Hong Kong

Tel: 852-22735712

Japan

EXOS Ebisu Building 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150

Tel: +81 (0)3.3445.7671
Fax: +81 (0)3.3445.7668

Korea

30th Floor, ASEM Tower,
159-1 Samsung-dong,
Kangam-ku,
Seoul, Korea

Tel: +82.2.6001.3382
Fax: +82.2.6001.3030

United Kingdom

Dunlop House,
Riverside Way
Camberley,
Surrey GU15 3YL

Tel: +44 (0)1276.401452
Fax: +44 (0)1276.401490

Index

<act_fam> variable 8
<vhd_fam> variable 8

A

Actel 8
  Device Families 8
  Libraries 27
  web site 96
  web-based technical support 96
Actel Manuals 9
ACTgen Macro Builder 54
Adding
  Global Networks 28
  Ground 27
  Pins 47
  Power 27
  Properties 42
Adding ACTgen Macros 54
ALSPIN 47
Assumptions 8
Asynchronous designs 56
Automatic
  Fan-In Reduction 32
  Logic Reduction 32
  Module Reduction 32
  Pin Assignment 48
Automatic mapping 56

B

Back Annotation
  Effects of Combiner 37
Back-Annotated Timing 83
Buffering 32, 41

C

Calculating Module Utilization 42
Capturing a Design
  HDL-Based 21
  Schematic-Based 18
Checker
  checker log 32
ChipEdit 80
ChipEditor 13
CLK 30
CLKA 30
CLKB 30
Clock
  Dedicated 30
  Routed 30
Combinatorial Module Reduction 33
Combiner 32
  Back Annotation Effects 37
Combining Modules 32
  40MX 39
  ACT 1 39
Compile 11
Constant Input Reduction 36
Contacting Actel
  customer service 95
  electronic mail 96
  telephone 97
  toll-free 95
  web-based technical support 96
Conventions 8
  <act_fam> variable 8
  <vhd_fam> variable 8
Customer service 95

D

DCLK 51
Dedicated Clocks 30
  HCLK 30

- IOCLK 31
- Delay
  - Estimating 52
- Design
  - netlist 56
- design
  - implementation 77
  - layout 81
  - practices 56
- Design Creation/Verification 18, 21
  - EDIF Netlist Generation 18, 22
  - Functional Simulation 18, 21
  - HDL Source Entry 21
  - Schematic Capture 18
  - Structural Netlist Generation 22
  - Structural Simulation 22
  - Synthesis 22
- Design Flow
  - Design Creation/Verification 18, 21
  - Design Implementation 19
  - Schematic-Based 17–20
  - Synthesis-Based 20–22
- Design Implementation 19
  - Place-and-Route 19
  - Power Analysis 19
  - Timing Simulation 19
- Design Layout 19
- Design Optimization, ProASIC 55
- Design Synthesis 22
- design synthesis 74
- Designer 77
  - Overview 11–16
  - Place-and-Route 19
  - Timing Analysis 19
- Designer Maintenance 15
- Designs
  - Hierarchical 27
  - Multiple Sheet 27
- Device
  - Families 8
  - Programming 19
  - Verification 20, 22
- device
  - programming 85
- device selection 88
- Document Assumptions 8
- Document Conventions 8
- Document Organization 7
- Duplicating Logic 41

E

- EDIF Netlist Generation
  - Schematic-Based 18
  - Synthesis-Based 22
- Electronic mail 96, 97
- Embedded memories 57
- Estimating Delays 52

F

- Fan Out 41
- Fan-In Reduction 32, 37
- FlashPro Lite
  - analyze chain 88
  - programming the device 90
  - saving the log file 93
  - setting up 85
  - verifying the correct programming 92
- Functional Simulation 18, 21
- Fuse 12

G

- Gate-Level Netlist 22

Gating clocks 56
Generating 52
  EDIF Netlist 18, 22
  Gate-Level Netlist 22
  Structural Netlist 22
Global Network 28
GND 27
Ground 27

H
HCLK 30
HDL Source Entry 21
Hierarchical
  Designs 27
Hierarchical blocks 56
High Fan Out 41
High fanout signals 32

I
I/O
  Clock 31
  Module Utilization 42
IOCLK 31
IOPCL 31

J
JTAG Pins 52

L
Layout 12
  Standard 12
  Timing Driven 12
Libraries 27
Logic Module Utilization 42
Logic Reduction 32, 36

M
Maintenance Mode 15
Manual Pin Assignment 48
Memories
  embedded 57
Module Reduction 32
  40MX 39
  ACT 1 39
  Combinatorial 33
Multiple Sheet Designs 27

N
Naming Conventions
  Schematic 23
  Verilog 25
  VHDL 23
Net
  Loading 41
Netlist Generation
  EDIF 18, 22
  Gate-Level 22
  Structural 22
Network 28
new project
  creation 59

O
Online Help 9

P
Pin
  Adding 47
  Assignment 47–52
  Automatic Assignment 48
  DCLK 51
  JTAG 52

- PRA 51
- PRB 51
- Probe 51
- SDI 51
- TCK 52
- TDI 52
- TDO 52
- TMS 52
- Unused I/O 51

pin editing 78
PinEdit 80
PinEditor 13
Place-and-Route 19
post-synthesis simulation 76
Power 27
Power Analysis 19
PRA 51
PRB 51
Preserving Macros 42
pre-synthesis simulation
- drawing waveforms 67
- exporting the testbench 69
- importing signal information 66
- performing 65

ProASIC
- global routing resources 32

Probe Pins 51
Product Support 95–98
Product support
- customer service 95
- electronic mail 96, 97
- oll-free line 95
- technical support 96
- web site 96

Programming a Device 19
programming file
- generation 84

Property, ALSPRESERVE 42

R

Remapping 34
Routed Clocks 30
- CLK 30
- CLKA 30
- CLKB 30
- QCLK 30

S

Schematic Capture 18
Schematic-Based Design Flow 17–20
- Design Creation/Verification 18
- Design Implementation 19
- Programming 19
- System Verification 20

SDI 51
Sequential Remapping 34
Simulation
- Functional 18, 21
- Schematic-Based 18, 19
- Structural 22
- Synthesis-Based 21, 22
- Timing 19

SmartPower 19
Special Clocks 31
- IOPCL 31

stapl file
- loading 88

Static Timing Analysis 19
stimulus
- creating using WaveFormer Lite 66

Structural Netlist Generation 22
Structural Simulation 22

Symbol, Top Level 52
Synplify 74
Synthesis 22
Synthesis-Based Design Flow 20–22
  Design Creation/Verification 21
  System Verification 22
System Verification 20, 22
  Silicon Explorer II 20, 22

T

TCK 52
TDI 52
TDO 52
test bench
  exporting 69
Timer 14, 80
Timer Tool 19
Timing 12
Timing Analysis 19
Timing Constraint 52
Timing Simulation 19
timing simulation 83
TMS 52
Toll-free line 95
Top Level Symbol 52

U

Unit Delays 18, 21
Unused Logic Removal 36
Utilization 42

V

variables
  <act_fam> 8
  <vhd_fam> 8
VCC 27

W

WaveFormer Lite 66
waveforms 67
Web-based technical support 96