Designing with Actel

Windows and UNIX Environments

Actel Corporation, Sunnyvale, CA 94086

© 1998 Actel Corporation. All rights reserved.

Printed in the United States of America

Part Number: 5029100-2

Release: May 1999

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.

Cadence is a registered trademark of Cadence Design Systems, Inc.

Mentor Graphics is registered trademark of Mentor Graphics, 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, and ViewDraw are registered trademarks and MOTIVE and SpeedWave are 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 xi
Document Organization xi
Document Assumptions xii
Document Conventions xiii
Actel Manuals xiii
On-Line Help xv

1 Designer Series Overview 1
ACTgen Macro Builder. .1
ACTmap VHDL Synthesis 1
Silicon Expert 1
Designer 2
Schematic-Based Design Methodology 5
HDL Synthesis-Based Design Methodology 5

2 Actel Design Flows 7
Schematic-Based Design Flow Illustrated 7
Schematic-Based Design Flow Overview 8
HDL Synthesis-Based Design Flow Illustrated 10
HDL Synthesis-Based Design Flow Overview 11

3 Design Considerations. 15
Naming Conventions 15
Hierarchical Designs. 19
Multiple Sheet Designs. 19
Actel Libraries 19
Adding Power and Ground 19
Adding a Global Network 20
Combinability 22
Net Loading. 30
Logic and I/O Utilization. 31
Adding Properties 32
Adding Input and Output Pins to the Design 32

Generating a Top-Level Symbol 38
Entering Constraints for Timing Driven Place and Route 38
Estimating Pre-Layout Timing 38
Adding ACTgen Macros 39

4 Generating Macros Using ACTgen 41
ACTgen Features 41
ACTgen Main Window. 42
Generating New Macros 43
Modifying Existing Macros 44
Fan-in Control Tool 46
Generating a Macro Report. 48

5 ACTmap VHDL Synthesis Tool 51
ACTmap Features 51
ACTmap Windows. 52
Compiling VHDL 54
Optimizing a Netlist 55
Translating a Netlist 57
Defining I/Os 58
Implementing a Hierarchical Project 59
Configuration Files 61
Using ACTmap in Batch Mode 62

6 Design Implementation Using Designer 63
Importing a Netlist/Compiling a New Design 63
Opening an Existing Design 69
Importing (DCF), (PIN), and (CRT) Information 70
Changing Design Name and Family. 71
Changing Other Design Information 72
DT Edit 76
Assigning Pins. 80
PinEdit 81

DT Analyze 83
ChipEdit 84
Layout 84
Extracting Timing Information 87
Fuse 88
Exporting Files 89
Generating Reports 90
Setting Designer Preferences 94
Terminating the Designer Session 94

7 Timing Analysis using DT Analyze 95
DT Analyze 95
DT Analyze Examples 101
Batch Timer 112

8 Generating a Programming File 113
Silicon Signature 113
Generating a Programming File 113

A ChipEdit 117
ChipEdit Window 117
Placing Macros 120
Moving Macros 121
Fixing Macros 121
ChipEdit View Options .122

B Using Designer Script 125
Running Designer in Batch Mode. 125
Designer Script Language 127
Supported Commands 128
Set Command Variables .132
Script Examples 133

C Product Support 137

Actel U.S. Toll-Free Line .137
Customer Service 137
Customer Applications Center 138
Guru Automated Technical Support 138
Web Site 138
FTP Site 139
Electronic Mail 139
Worldwide Sales Offices .140

Index 141

List of Figures

Actel Schematic-Based Design Flow 7
Actel HDL-Based Design Flow 10
Combined Combinatorial Macros 23
Combined Combinatorial and Sequential Macros 23
Complex Macros Divided 24
Logic Modules Decreased 25
Logic Reduced to One Sequential Module 25
Unused Logic Removal 26
Constant Input Reduction 26
Fan-in Reduction 27
Delays Before Combining 27
Delays After Combining 28
DFM Ties-Offs to Reduce Logic Module Count 29
Buffering 30
Duplicating Logic 30
Adding ALSPRESERVE Property to a Net 32
Adding ALSPIN Property to a Net 33
System Configuration for Unused Pins 37
Tri-Stating Unused Pins 37
Critical Path Timing Example 39
ACTgen Main Window 42
Fan-in Control Dialog Box 48
ACTgen .log File 50
ACTmap Main Window 53
Designer Main Window 63
Import Netlist Dialog box 64
Setup Design Dialog Box 65
Device Selection Dialog Box (Standard) 65
Device Selection Dialog Box (SX/SXA) 66
Device Variations Dialog Box 67
Operating Conditions Dialog Box 68
DT Edit Window 76

PinEdit Window 81
Layout Dialog Box 85
Extract Dialog Box 88
Export Dialog Box 89
Timing Report Dialog Box 92
Timing Report Preferences Dialog Box 93
Program Preferences Dialog Box 94
DT Analyze Filters Dialog Box 96
DT Analyze Preferences 97
DT Analyze Window 98
Expand List Dialog Box 99
Expanded Chart Dialog Box 101
Sample Circuit 102
Maximum Register-to-Register Delay 103
Expanded Maximum Register-to-Register Delay 104
Clock-to-Output Delay .105
Clock-to-Output Delay .106
Input-to-Output Delay .107
Input-to-Output Delay .108
On-Chip Delays 110
On-Chip Data Path Delay 111
Designer Main Window .114
Fuse Dialog Box 115
ChipEdit Window 117
Macros Displayed Hierarchically 122
List Boxes Configuration Dialog Box 123

List of Tables

Global Network Attributes 20
ACT 3 Elements that Cannot Connect to HCKLBUF 21
54SX Elements that Cannot Connect to HCKLBUF 22
Junction Temp. Deltas at 1W Power Consumption 75
Constraint Results 79
Chip Window Colors and Symbols 119
Supported Script Commands 128

Introduction

The *Designing with Actel* manual contains information and procedures for using the Designer Series 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, and generate a programming file to program an Actel FPGA.

Also included in this manual is information about the Actel ACTgen Macro Builder macro generation software, which allows you to generate optimized macro blocks for your design, and the ACTmap VHDL Synthesis tool, which allows you to optimize VHDL designs for Actel devices.

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 Series Development System. Refer to “Actel Manuals” on page xiii for a list of documents.

Document Organization

The *Designing with Actel* manual is divided into the following chapters:

Chapter 1 - Designer Series Overview gives an overview of the programs contained in the Designer Series Development system software.

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

Chapter 3 - Design Considerations contains useful information and procedures about creating designs using the Actel Designer Series software.

Chapter 4 - Generating Macros Using ACTgen contains information about creating macros using the ACTgen Macro Builder software.

Chapter 5 - ACTmap VHDL Synthesis Tool contains information about using the ACTmap VHDL synthesis tool.

Chapter 6 - Design Implementation Using Designer contains information about implementing designs using the Designer Series software.

Chapter 7 - Timing Analysis using DT Analyze contains information and procedures about performing timing analysis using DT Analyze in Designer.

Chapter 8 - Generating a Programming File contains information and procedures about generating a Actel device programming file for Actel or third-party programmers.

Appendix A - Chip Edit contains information and procedures about using the ChipEdit feature of Designer, which allows you to view the location of your design's macros and edit the placement of both I/O and logic macros in your design.

Appendix B - Using Designer Script provides information about using the Actel Designer script language.

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

Document Assumptions

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

1. You have installed the 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.

Information that is meant to be input by the user is formatted as follows:

```
keyboard input
```

The contents of a file is formatted as follows:

```
file contents
```

Messages that are displayed on the screen appear as follows:

```
Screen Message
```

Actel Manuals

The Designer Series software includes printed and on-line manuals. The on-line manuals are in PDF format on the CD-ROM in the “/manuals” directory. These manuals are also installed onto your system when you install the Designer software. To view the on-line manuals, you must install Adobe® Acrobat Reader® from the CD-ROM.

The Designer Series includes the following manuals, which provide additional information on designing Actel FPGAs:

Designing with Actel. This manual describes the design flow and user interface for the Actel Designer Series software, including information about using the ACTgen Macro Builder and ACTmap VHDL Synthesis software.

Actel HDL Coding Style Guide. This guide provides preferred coding styles for the Actel architecture and information about optimizing your HDL code for Actel devices.

ACTmap VHDL Synthesis Methodology Guide. This guide contains information, optimization techniques, and procedures to assist designers in the design of Actel devices using ACTmap VHDL.

Silicon Expert User's Guide. This guide contains information and procedures to assist designers in the use of Actel's Silicon Expert tool.

DeskTOP Interface Guide. This guide contains information about using the integrated VeriBest® and Synplicity® CAE software tools with the Actel Designer Series FPGA development tools to create designs for Actel Devices.

Cadence® Interface Guide. This guide contains information and procedures to assist designers in the design of Actel devices using Cadence CAE software and the Designer Series software.

Mentor Graphics® Interface Guide. This guide contains information and procedures to assist designers in the design of Actel devices using Mentor Graphics CAE software and the Designer Series software.

MOTIVE™ Static Timing Analysis Interface Guide. This guide contains information and procedures to assist designers in the use of the MOTIVE software to perform static timing analysis on Actel designs.

Synopsys® Synthesis Methodology Guide. This guide contains preferred HDL coding styles and information and procedures to assist designers in the design of Actel devices using Synopsys CAE software and the Designer Series software.

Viewlogic Powerview® Interface Guide. This guide contains information and procedures to assist designers in the design of Actel devices using Powerview CAE software and the Designer Series software.

Viewlogic Workview Office Interface Guide. This guide contains information and procedures to assist designers in the design of Actel devices using Workview Office CAE software and the Designer Series software.

VHDL Vital Simulation Guide. This guide contains information and procedures to assist designers in simulating Actel designs using a Vital compliant VHDL simulator.

Verilog Simulation Guide. This guide contains information and procedures to assist designers in simulating Actel designs using a Verilog simulator.

Activator and APS Programming System Installation and User's Guide. This guide contains information about how to program and debug Actel devices, including information about using the Silicon Explorer diagnostic tool for system verification.

Silicon Sculptor User's Guide. This guide contains information about how to program Actel devices using the Silicon Sculptor software and device programmer.

Silicon Explorer Quick Start. This guide contains information about connecting the Silicon Explorer diagnostic tool and using it to perform system verification.

Designer Series Development System Conversion Guide UNIX® Environments. This guide describes how to convert designs created in Designer Series versions 3.0 and 3.1 for UNIX to be compatible with later versions of Designer Series.

Designer Series Development System Conversion Guide Windows Environments. This guide describes how to convert designs created in Designer Series versions 3.0 and 3.1 for Windows to be compatible with later versions of Designer Series.

Actel FPGA Data Book. This guide contains detailed specifications on Actel device families. Information such as propagation delays, device package pinout, derating factors, and power calculations are found in this guide.

Macro Library Guide. This guide provides descriptions of Actel library elements for Actel device families. Symbols, truth tables, and module count are included for all macros.

A Guide to ACTgen Macros. This Guide provides descriptions of macros that can be generated using the Actel ACTgen Macro Builder software.

On-Line Help

The Designer Series software comes with on-line help. On-line help specific to each software tool is available in Designer, ACTgen, ACTmap, Silicon Expert, Silicon Explorer, Silicon Sculptor, and APSW.

1

Designer Series Overview

The Designer Series Development System 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 created 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 Series Development System.

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 "Generating Macros Using ACTgen" on page 41 for additional information.

ACTmap VHDL Synthesis

Designed to deliver high quality results the first time out, ACTmap is a complete, streamlined VHDL synthesis tool that also optimizes gate-level descriptions. Refer to "ACTmap VHDL Synthesis Tool" on page 51 for additional information.

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.

Designer

Designer is an interactive design implementation tool that can import designs created with popular third-party schematic and HDL CAE tools. Designer features fully automatic layout, pin fixing, a chip editor, and a back annotation utility. Designer also includes DirectTime (DT) tools, including DT Edit and DT Layout for timing driven place and route, and DT Analyze for static timing analysis. These tools allow designers to define, layout (place and route), verify, and program high performance designs at high levels of utilization with improved designer productivity. Refer to “Design Implementation Using Designer” on page 63 for additional information.

Design Flow Manager

Designer uses a Design Flow Manager that graphically displays the completed steps of the design implementation process. 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 users to click any button in the Designer Main window to begin the design implementation process. Designer then prompts the user through all of the necessary steps of the flow to complete the step that was 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 “Importing a Netlist/Compiling a New Design” on page 63 for additional information.

DT Edit

DT Edit allows designers to define timing requirements for critical paths in a design for use in timing driven place and route using DT Layout. The requirements can also be used for timing verification in DT Analyze or timing reports. DT Edit must be used to perform timing driven place and route using DT Layout. Refer to “DT Edit” on page 76 for additional information.

PinEdit

PinEdit is a graphical interface that allows designers to view pin locations and manually assign, edit, and fix pin locations for a design. Manual pin assignment is optional. However, PinEdit 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. Refer to “PinEdit” on page 81 for additional information.

Layout

Layout (place and route) takes the design netlist information, PinEdit information (optional), and DT Edit information (DT Layout only), and maps the information into the selected device. Designer includes Incremental Layout, which allows designers to speed through design iterations by only re-laying out netlist information that has changed. Two layout modes are supported, Standard and DirectTime.

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.

DT Layout

For timing critical designs, DT Layout uses timing requirements entered in DT Edit to constrain Layout. DT Layout is a fully automatic timing driven place and route utility that focuses resources to meet performance requirements. DT Layout takes 2 to 4 times longer than Standard Layout. Refer to “Layout” on page 84 for additional information.

DT Analyze

DT Analyze is an interactive timing tool used for timing verification and for debugging timing problems. Timing information can be displayed in a tabular or graphical format. A timing report generator is also provided. DT Analyze is optional. Refer to “Timing Analysis using DT Analyze” on page 95 for additional information.

ChipEdit

ChipEdit 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. Refer to “ChipEdit” on page 117 for additional information.

Extract

Extract exports the necessary delay information (back annotation) to perform post-layout analysis with third-party CAE tools. Additional information external to the Designer Series software may be required for building complete analysis files. Refer to “Extracting Timing Information” on page 87 for additional information.

Fuse

Fuse generates the necessary files to program an Actel device. Fuse supports both Actel and Data I/O formats. You do not need to run the Fuse function if you are using APSW, unless you need to set a silicon signature. Refer to “Generating a Programming File” on page 113 for additional information.

Designer Script

Designer supports a script language that allows designers to group multiple design steps into a single operation. With Designer Script the FPGA design flow can be automated, making design iterations a snap. Refer to “Using Designer Script” on page 125 for additional information.

Schematic-Based Design Methodology

If you prefer designing with schematic tools, Actel offers a complete tool suite that lets you take your designs from concept to silicon. On the front end, ACTgen integrates with third-party CAE tools for schematic-entry and gate-level simulation. Once your design has been created and verified, Designer completes the design with place and route, timing analysis, and back annotation for timing verification. Refer to “Schematic-Based Design Flow Illustrated” on page 7 for an overview of the Designer Series schematic-based design flow.

HDL Synthesis-Based Design Methodology

If you prefer a high-level design methodology, the Designer Series 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, ACTmap and Designer help you complete the design with place and route, timing analysis, and timing verification. Refer to “HDL Synthesis-Based Design Flow Illustrated” on page 10 for an overview of the Designer Series synthesis-based design flow.

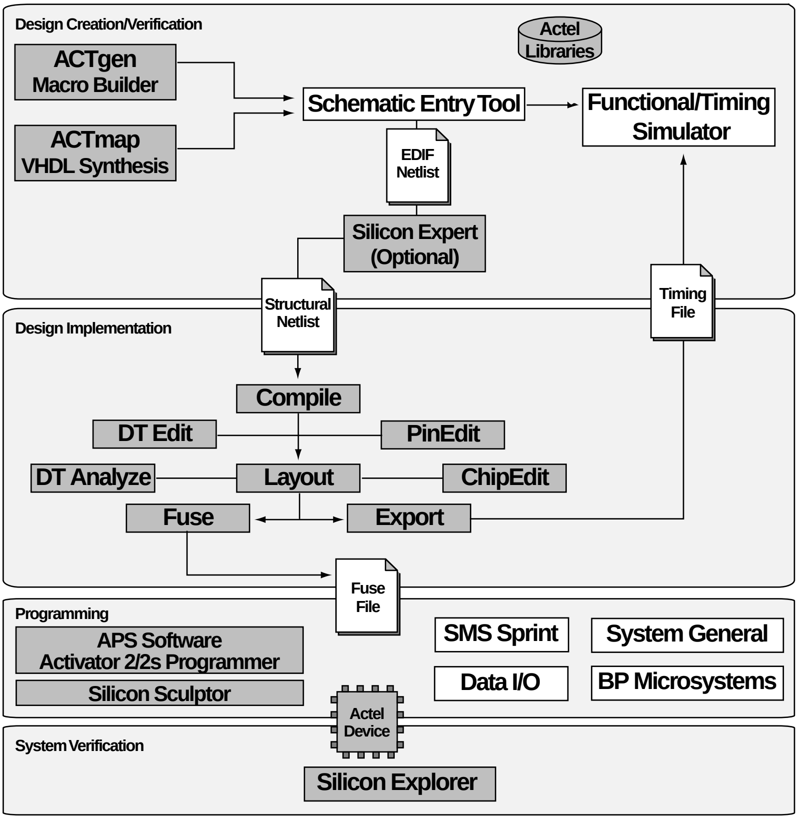
2

Actel Design Flows

The Designer Series 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 Series 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
ACTgen Macro Builder
ACTmap VHDL Synthesis
Schematic Entry Tool
Functional/Timing Simulator
EDIF Netlist
Silicon Expert (Optional)
Structural Netlist
Timing File
Design Implementation
Compile
DT Edit
PinEdit
DT Analyze
Layout
ChipEdit
Fuse
Export
Fuse File
Programming
APS Software Activator 2/2s Programmer
Silicon Sculptor
SMS Sprint
System General
Data I/O
BP Microsystems
Actel Device
System Verification
Silicon Explorer

Figure 2-1. Actel Schematic-Based Design Flow

---

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

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 *Cadence Interface Guide, Mentor Graphics Interface Guide, MOTIVE Static Timing Analysis Interface Guide, Viewlogic Powerview Interface Guide*, or *Viewlogic Workview Office 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 third-party schematic capture software. 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. Refer the documentation included with your schematic capture tool for information.

Functional Simulation

Perform a functional simulation of your design using 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.

EDIF Netlist Generation

After you have captured and verified your design, you must generate an EDIF netlist for place and route in Designer. Refer to the Actel Interface Guides and the documentation included with your schematic capture tool for information about generating an EDIF netlist.

Design Implementation

During design implementation, a design is placed and routed using Designer. Additionally, static timing analysis can be performed on a design in Designer with the DT Analyze 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 "Design Implementation Using Designer" on page 63 for information about using Designer.

Static Timing Analysis

Use the DT Analyze tool in Designer to perform static timing analysis on your design. Refer to "Timing Analysis using DT Analyze" on page 95 for information about using DT Analyze.

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

Timing Simulation

Perform a timing simulation of your design using a third-party simulation tool after placing and routing it in Designer. Timing simulation requires information extracted and back annotated from Designer. Refer to the Actel Interface Guides and the documentation included with 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 "Generating a Programming File" on page 113 and the *Activator and APS Programming System Installation and User's Guide* or *Silicon Sculptor User's Guide* for information about programming an Actel device.

System Verification

You can perform system verification on a programmed device using the Actel Silicon Explorer diagnostic tool. Refer to the *Activator and APS Programming System Installation and User's Guide* or *Silicon Explorer Quick Start* for information about using the Silicon Explorer.

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]

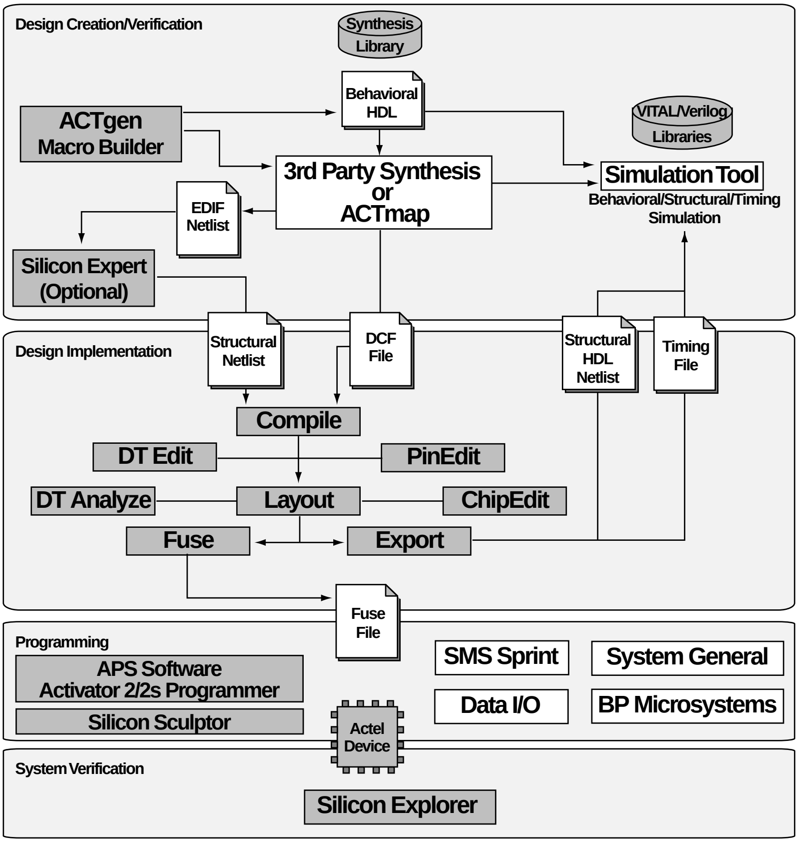

Design Creation/Verification
Synthesis Library
Behavioral HDL
ACTgen Macro Builder
VITAL/Verilog Libraries
3rd Party Synthesis or ACTmap
Simulation Tool
Behavioral/Structural/Timing Simulation
EDIF Netlist
Silicon Expert (Optional)
Design Implementation
Structural Netlist
DCF File
Structural HDL Netlist
Timing File
Compile
DT Edit
PinEdit
DT Analyze
Layout
ChipEdit
Fuse
Export
Fuse File
Programming
APS Software Activator 2/2s Programmer
Silicon Sculptor
SMS Sprint
System General
Data I/O
BP Microsystems
Actel Device
System Verification
Silicon Explorer

Figure 2-2. Actel HDL-Based Design Flow

---

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

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 *Cadence Interface Guide*, *Mentor Graphics Interface Guide*, *MOTIVE Static Timing Analysis Interface Guide*, *Synopsys Synthesis Methodology Guide*, *VHDL Vital Simulation Guide*, *Verilog Simulation Guid*e, *Viewlogic Powerview Interface Guide*, or *Viewlogic Workview Office 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 with ACTmap or third-party synthesis software. After synthesis, structural simulation of the design can be performed. Finally, an EDIF netlist is generated for use in Designer and a structural HDL netlist is generated for structural 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*, *ACTmap VHDL Synthesis Methodology Guide* or *Synopsys Synthesis Methodology Guide* for additional information about writing HDL code for Actel designs.

Behavioral Simulation

Perform a behavioral simulation of your design before synthesis. Behavioral 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, the Actel Simulation Guides, and the documentation included with your simulation tool for information performing behavioral simulation.

Synthesis

After you have created your behavioral HDL source file, you must synthesize it using ACTmap or a third-party synthesis tool before placing and routing it in Designer. Synthesis transforms the behavioral HDL file into a gate-level netlist and optimizes the design for a target technology. Refer to “ACTmap VHDL Synthesis Tool” on page 51 or the documentation included with your synthesis tool for information for information about performing design synthesis.

EDIF Netlist Generation

After you have created, synthesized, and verified your design, you must generate an EDIF netlist for place and route in Designer. This EDIF 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 netlist.

Structural HDL Netlist Generation

Generate a structural HDL netlist from your EDIF netlist for use in structural and timing simulation by either exporting it from Designer or by using the Actel “edn2vhdl” or “edn2vlog” program. 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 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

During design implementation, a design is placed and routed using Designer. Additionally, static timing analysis can be performed on a design in Designer with the DT Analyze 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 “Design Implementation Using Designer” on page 63 for information about using Designer.

Static Timing Analysis

Use the DT Analyze tool in Designer to perform static timing analysis on your design. Refer to “Timing Analysis using DT Analyze” on page 95 for information about using DT Analyze.

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

Timing Simulation

Perform a timing simulation of your design using a third-party simulation tool after placing and routing it in Designer. Timing simulation requires information extracted and back annotated from Designer. Refer to the Actel Interface Guides, the Actel Simulation Guides, and the documentation included with 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 “Generating a Programming File” on page 113 and the *Activator and APS Programming System Installation and User’s Guide* or *Silicon Sculptor User’s Guide* for information on programming an Actel device.

System Verification

You can perform system verification on a programmed device using the Actel Silicon Explorer diagnostic tool. Refer to the *Activator and APS Programming System Installation and User’s Guide* or *Silicon Explorer Quick Start* for information about using the Silicon Explorer.

3

Design Considerations

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

Naming Conventions

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

Schematic

Use only alphanumeric and underscore “_” characters for schematic net and instance names. Do not use asterisks, forward slashes, backward slashes, or spaces.

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

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

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 DT 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 for information about adding power and ground to your design.

Adding a Global Network

The Actel architecture provides global networks that allow high fan out 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	2	Yes	CLKBIBUF CLKBUF CLKINT[a] CLKBUFI[b] CLKINTI[b]	Can connect to CLK and G pins
HCLK	dedicated	ACT 3 54SX	1	No	HCLKBUF	Directly hard-wired to certain S-modules
IOCLK	dedicated	ACT 3	1	No	IOCLKBUF	Connected to all I/O modules
IOPCL	special	ACT 3	1	No	IOPCLBUF	Connected to I/O module set and reset pins
QCLK A-D	routed	3200DX 42MX	4	Yes	QCLKBUF, QCLKINT	Each QCLK drives a quadrant of the device

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

b. 54SX only.

Routed Clocks

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

CLK, CLKA, CLKB

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

QCLK

QCLK (available on 42MX and 3200DX 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 or QCLKINT macro.

Dedicated Clocks

Dedicated clocks are clock networks that are directly wired to sequential and 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 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 sequential elements that cannot be connected to HCKLBUF because they are built from combinatorial modules:

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

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

IOCLK

IOCLK is a dedicated hard-wired clock input for I/O modules. IOCLK is directly wired to each I/O module 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, 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.

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, which are described in the following sections:

- Combinatorial Module Reduction
- Sequential Remapping
- Unused Logic Removal
- Constant Input Reduction
- Fan-in Reduction

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.

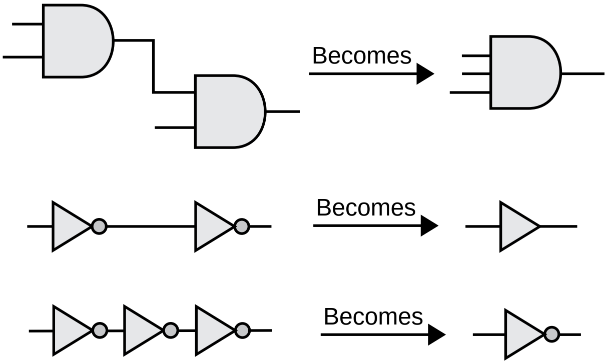


Figure 3-1. Combined Combinatorial Macros

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 devices. Instead, an automatic DirectConnect between a combinatorial module and a sequential module is used.

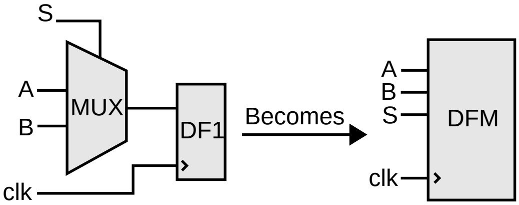


Figure 3-2. Combined Combinatorial and Sequential Macros

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 devices because the sequential modules in 54SX 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.

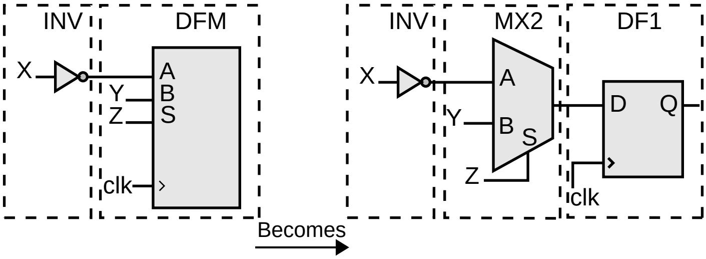


Figure 3-3. Complex Macros Divided

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.

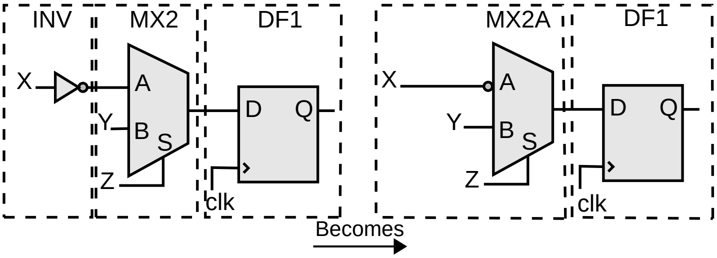

INV
MX2
DF1
X
A
Y
B
S
Z
D
Q
clk
Becomes
MX2A
DF1
X
A
Y
B
S
Z
D
Q
clk

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.

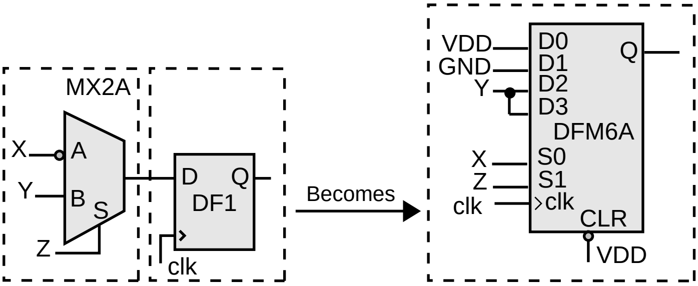

MX2A
X
A
Y
B
S
Z
D
Q
DF1
clk
Becomes
VDD
GND
Y
D0
D1
D2
D3
Q
DFM6A
X
Z
clk
S0
S1
clk
CLR
VDD

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.

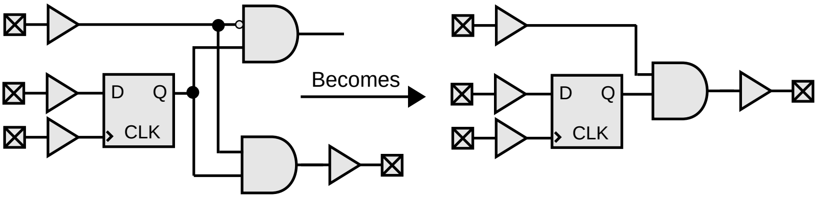


Figure 3-6. Unused Logic Removal

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.

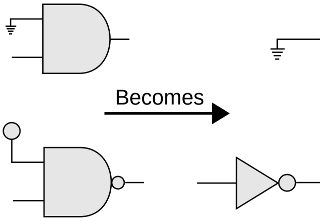


Figure 3-7. Constant Input Reduction

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 how Fan-in Reduction is done.

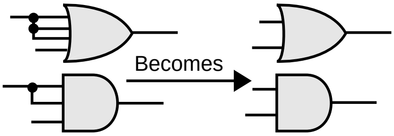


Figure 3-8. Fan-in Reduction

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 DT Analyze.

There are slight differences in the way delays are back annotated to a simulator and how they are viewed in DT Analyze. Figure 3-9 shows how delays are viewed in schematics before combining.

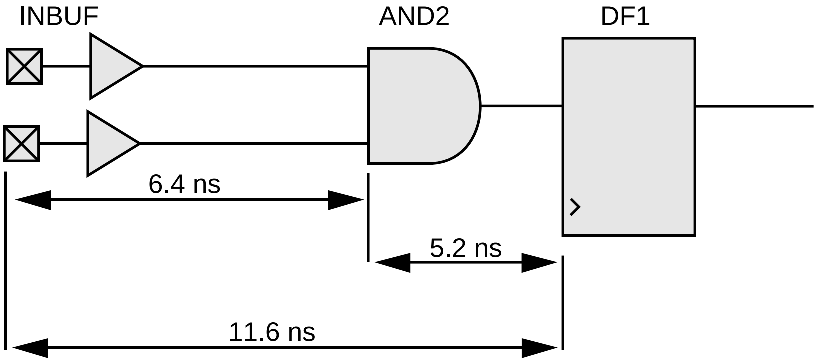


Figure 3-9. Delays Before Combining

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.” DT Analyze 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.

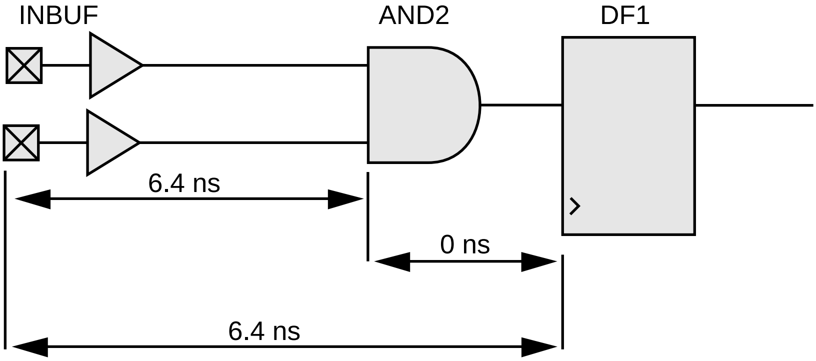


Figure 3-10. Delays After Combining

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.

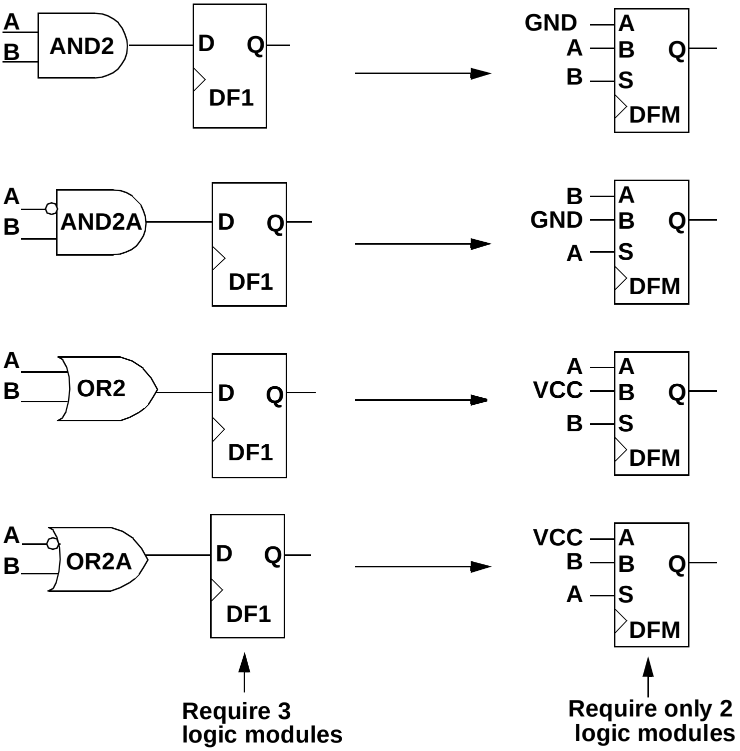

A
B
AND2
D
Q
DF1
GND
A
B
A
B
S
Q
DFM
A
B
AND2A
D
Q
DF1
B
GND
A
A
B
S
Q
DFM
A
B
OR2
D
Q
DF1
A
VCC
B
A
B
S
Q
DFM
A
B
OR2A
D
Q
DF1
VCC
B
A
A
B
S
Q
DFM
Require 3
logic modules
Require only 2
logic modules

Figure 3-11. DFM Ties-Offs to Reduce Logic Module Count

Net Loading

High fan out degrades performance and increases the potential of unrouted nets. Designer does not permit fan-outs greater than 24. 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 fan-out greater than 10, you should modify them by buffering, as shown in Figure 3-12.

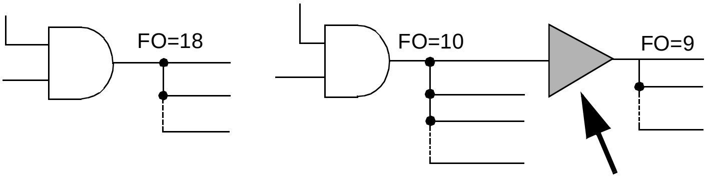


Figure 3-12. Buffering

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

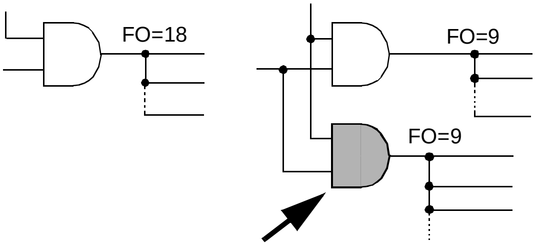


Figure 3-13. Duplicating Logic

Both buffering and duplicating logic cost additional logic modules. You can use buffering if you do not care 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 fan-out 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.

Calculate the logic module and I/O module utilization as follows:

$$\textbf{Module Utilization} = \frac{\textbf{Number of Modules Used}}{\textbf{Number of Modules Available}} \textbf{ X 100}$$

For example, an ACT 2 A1240 design using 600 logic modules and 93 I/O modules would have the following module utilization:

$$\textbf{Logic Module Utilization} = \frac{600}{684} \textbf{ X 100} = 87.7\%$$

$$\textbf{I/O Module Utilization} = \frac{93}{104} \textbf{ x 100} = 89.4\%$$

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 *FPGA Data Book and Design Guide* and *Macro Library Guide* for additional information.

Adding Properties

This section describes how to add properties to your designs.

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 “ASLPRESERVE” 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.

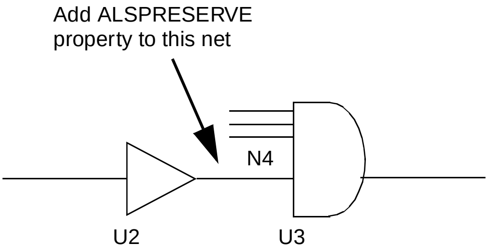


Figure 3-14. Adding ALSPRESERVE Property to a Net

Example EDIF netlist:

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

Adding Input and Output Pins to the Design

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.

Typically, pin assignments are made within Designer using PinEdit. However, many third-party CAE tools allow users to enter pin assignments directly into the design. The ASLPIN property may be added to the net connecting a port to an I/O buffer, shown in Figure 3-15. 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.

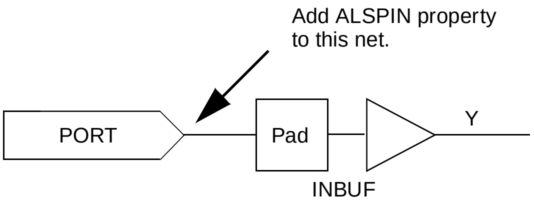


Figure 3-15. Adding ALSPIN Property to a Net

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 PinEdit:** Refer to “PinEdit” on page 81 for information about using PinEdit 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 “Importing (DCF), (PIN), and (CRT) Information” on page 70 for more information on how to import a file into Designer.

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 PinEdit.

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 placement and routing, 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 placement and routing could be compromised. Fixed manual assignments should be kept to a minimum.

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 would allow 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 due 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 *FPGA Data Book* 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 greater number of horizontal routing tracks handles these interconnections to reduce routing congestion. 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 *FPGA Data Book*.

- 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, 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 fan-out signals.

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.

If a package pin on a device is not used, Designer assigns the I/O circuitry as an output, with its input held to a logic low. Thus, the output is also a logic low. If the PCB design has a track connected to this unused pin, that track is held to a logic low. Figure 3-16 illustrates how the system configures an unused package pin.

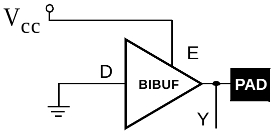


Figure 3-16. System Configuration for Unused Pins

If you want to tri-state unused I/O pins, use a TRIBUFF macro, as shown in Figure 3-17. The D and E inputs of a TRIBUFF are connected to GND. Therefore, the PAD output goes tri-state and does not load any external paths.

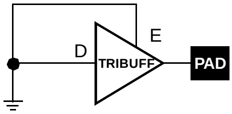


Figure 3-17. Tri-Stating Unused Pins

Using Probe Pins as I/O Pins

Layout automatically avoids using the probe pins "PRA," "PRB," "SDI," and "DCLK" 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 number of non-probe I/O pins for a device is the total number of I/O pins available minus 4. The function of the probe pins depends on the Mode Pin and Security Fuse Programming.

Using JTAG Pins as I/O Pins for 3200DX, 42MX and 54SX only

The high-capacity devices of the 3200DX, 42MX, and 54SX 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. The function of the JTAG pins depends on the J-fuse programming.

Generating a Top-Level Symbol

You can create a top-level symbol for the entire design. 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, “PRA,” “PRB,” “SDI,” “DCLK,” 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.

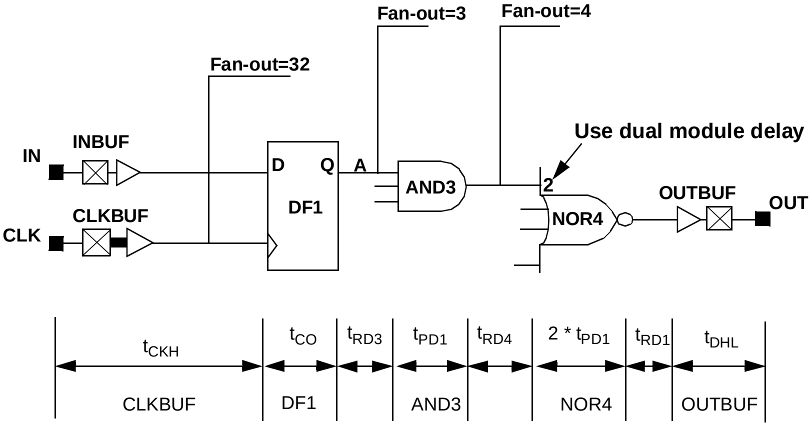
Entering Constraints for Timing Driven Place and Route

Unique timing constraints for each signal path in a design can be specified using DT Edit. It may only be necessary to specify the clock frequency for synchronous designs. The DirectTime mode typically eliminates time-consuming iterations of the design flow. Timing constraints can also be specified in a design constraint file (DCF) and imported into Designer. Refer to “DT Edit” on page 76 for information about specifying timing constraints in DT Edit, and “Importing (DCF), (PIN), and (CRT) Information” on page 70 for information about importing a DCF 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 Actel *FPGA Data Book*.

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 *FPGA Data Book*. 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.


Fan-out=3
Fan-out=4
Fan-out=32
Use dual module delay
IN
INBUF
CLK
CLKBUF
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

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

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

$$= 27.4\text{ns}$$

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. Refer to “Generating Macros Using ACTgen” on page 41 for information about using the ACTgen Macro Builder.

4

Generating Macros Using ACTgen

This chapter describes how to use the ACTgen Macro Builder, including information about generating new macros, modifying existing macros, using the Fan-In Control tool, and generating macro reports. Refer to the Actel Interface Guides for information about adding ACTgen macros to designs created with those CAE tools.

ACTgen Features

The ACTgen Macro Builder contains the following features:

Generate New Macros

ACTgen can generate datapath macros that can be inserted as symbols into schematic designs or instantiated into synthesis-based designs.

Modify Existing Macros

If you have previously generated an ACTgen macro for a design, ACTgen can modify this macro, changing the family or variations as desired.

Create HDL Behavioral Models

If you are using an HDL synthesis-based design flow, ACTgen can create VHDL or Verilog behavioral models of macros for use in behavioral simulation prior to synthesis. Once the model has been simulated, you can then instantiate the macro into your design.

Fan-In Control

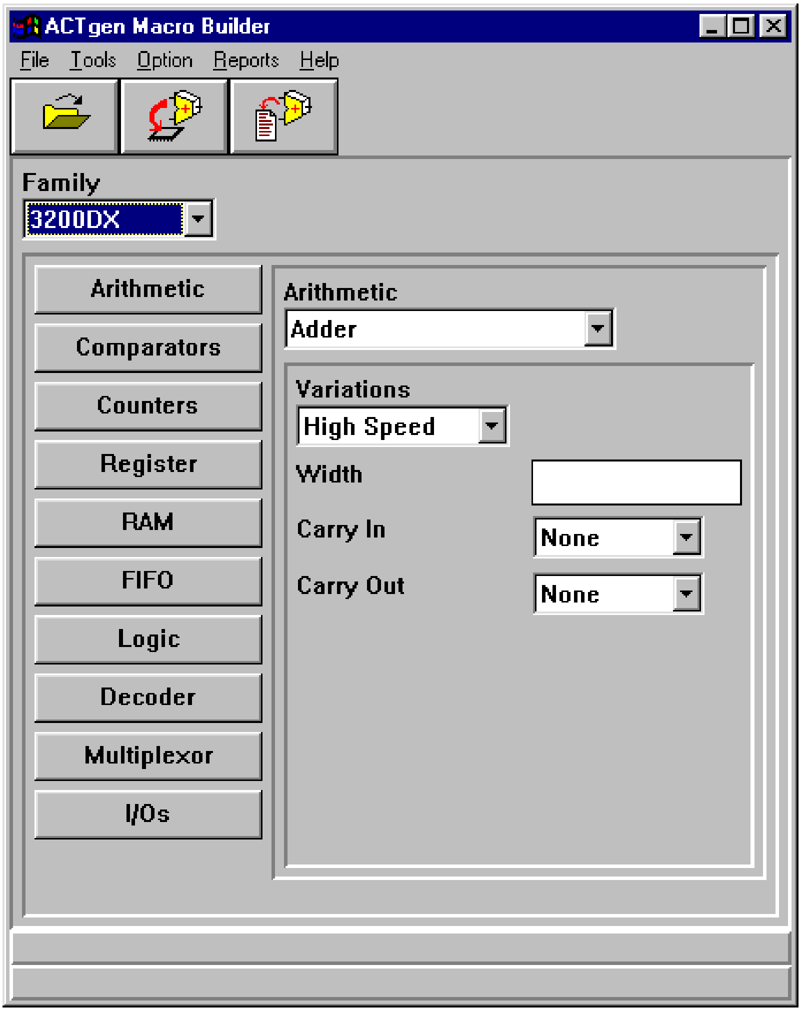
If you need to control the buffering of clocks, asynchronous presets and clears, and other control signals in a macro, you can use the fan-in control tool to specify the type of buffering to use.

Macro Reports

ACTgen stores information about a macro as it generates the macro. You can save and print this information when you generate the macro.

ACTgen Main Window

When you invoke ACTgen, the main window, shown in Figure 4-1, is displayed. From this window, the target family of Actel FPGA, macro type to generate, and the variations of that macro are selected.


ACTgen Macro Builder
File
Tools
Option
Reports
Help
Arithmetic
Comparators
Counters
Register
RAM
FIFO
Logic
Decoder
Multiplexor
I/Os

Family: 3200DX

Arithmetic: Adder

Variations: High Speed

Width:

Carry In: None

Carry Out: None

Figure 4-1. ACTgen Main Window

Generating New Macros

Use the following procedure to generate a new macro with ACTgen.

1. **Invoke ACTgen.**

   PC

   Select ACTgen Macro Builder from the Designer menu under Programs in the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   actgen
   ```

   The ACTgen Main window, shown in Figure 4-1, is displayed.

2. **Specify the Family.** Select the family of the Actel device the macro is to be used for in the Family pull-down menu.
3. **Select the macro type to generate.** Click one of the macro type buttons, specifying Arithmetic, Comparators, Counters, Register, RAM, FIFO, Logic, Decoder, Multiplexer, or I/Os.
4. **Specify the specific macro type.** Each type of macro may have several specific types available in the pull down menu above the Variations area.
5. **Set macro variations.** The Variations area displays options that are available for the specific Macro type.
6. **(Optional) Set Fan-in Control.** If you wish to alter fan-in for control signals, specify the specific macro type and set macro variations first. Choose the Fan-in Control command from the Options menu. The Fan-In Control dialog box, shown in Figure 4-2, is displayed. Set the desired buffering type and value for each signal. Refer to “Fan-in Control Tool” on page 46 for additional information.
7. **Generate the macro.** Select the Generate command from the File menu or click the Generate Macro Button on the toolbar. The Generate Macro dialog box is displayed.

8. **Select the output format from the Netlist/CAE Formats menu.** Choose the appropriate output format.

   Note: The Cadence (Concept), Mentor Graphics, or Viewlogic format will not be in the Netlist/CAE Formats menu if these libraries were not installed when Designer was installed.

9. **(Optional) Generate a VHDL or Verilog behavioral model for the macro.** If your design is an HDL design and you want to perform a behavioral simulation before synthesizing the design, check the appropriate option check box to generate a behavioral model for your macro.

   Note: These options may not be available if the behavioral models are not available for the given selection.

10. **Enter the name of your macro.** Type the path and name of your design in the File Name Box or use the Browse button.

11. **Click OK.** The ACTgen Macro Builder generates the new macro with the specified options and displays information about the macro in the ACTgen Report window.

    If you specified the netlist type as Viewlogic, Cadence, or Mentor Graphics, in addition to generating the macro, the ACTgen Macro Builder generates a symbol for the macro and displays the process on screen.

12. **(Optonal) Save your report.** In the ACTgen Macro Builder, save your report in *.log format by choosing Save As in the Reports menu, or click the Save button on the toolbar. Name the file and click OK.

Modifying Existing Macros

Use the following procedure to modify an existing macro.

1. **Invoke ACTgen.**

   PC

   Select ACTgen Macro Builder from the Designer menu under Programs in the Start menu.

UNIX

Type the following command at the prompt:

```
actgen
```

The ACTgen Main window, shown in Figure 4-1, is displayed.

2. **Open the macro to modify.** Choose the Open command from the File menu. The Open Macro dialog box appears. Type the name of the macro you want to open or use the Browse button.
3. **Modify the macro.** Change the family, macro type, specific macro type, and Variations as desired.
4. **(Optional) Set Fan-in Control.** If you wish to alter fan-in for control signals, specify the specific macro type and set macro variations first. Choose the Fan-in Control command from the Options menu. The Fan-In Control dialog box, shown in Figure 4-2, is displayed. Set the desired buffering type and value for each signal. Refer to “Fan-in Control Tool” on page 46 for additional information.
5. **Re-generate the macro.** Select the Generate command from the File menu. The Generate Macro dialog box is displayed.
6. **Select the output format from the Netlist/CAE Formats menu.** Choose the appropriate output format.

   Note: The Cadence (Concept), Mentor Graphics, or Viewlogic format will not be in the Netlist/CAE Formats menu if these libraries were not installed when Designer was installed.

7. **(Optional) Generate a VHDL or Verilog behavioral model for the macro.** If your design is an HDL design and you want to perform a behavioral simulation before synthesizing the design, check the appropriate option check box to generate a behavioral model for your macro.

   Note: These options may not be available if the behavioral models are not available for the given selection.

8. **Enter the name of your macro.** Type the path and name of your design in the File Name Box or use the Browse button.

9. **Click OK.** The ACTgen Macro Builder generates the new macro with the specified options and displays information about the macro in the ACTgen Report window.

   If you specified the netlist type as Viewlogic, Cadence, or Mentor Graphics, in addition to generating the macro, the ACTgen Macro Builder generates a symbol for the macro and displays the process on screen.

10. **(Optional) Save your report.** In the ACTgen Macro Builder, save your report in *.log format by choosing Save As in the Reports menu, or click the Save button on the toolbar. Name the file and click OK.

Fan-in Control Tool

The Fan-in Control tool gives advanced users the ability to control the buffering of clocks, asynchronous presets and clears, and other control signals. This tool is optional because default buffering values are provided for all signals. The tool supports two types of buffering control, automatic and no buffering, which provide maximum buffering flexibility.

Automatic Buffering

Automatic buffering automatically inserts buffers as required, and provides ease of use for fanning out heavily loaded signals. Automatic buffering is the default buffering type for most signals. ACTgen automatically inserts buffers/inverters for this option and provides a single input for the signal. The value defined for automatic buffering indicates the maximum loading on the network for the given control signal. ACTgen also balances the loading as required. Automatic buffering can indirectly define input loading to a macro.

No Buffering

No buffering restricts ACTgen from inserting buffers. This allows designers to manually use global clock resources for control signals. This also provides the ability to enhance performance of control signals by performing a logic function and correcting for fan-in by duplicating logic external to the macro.

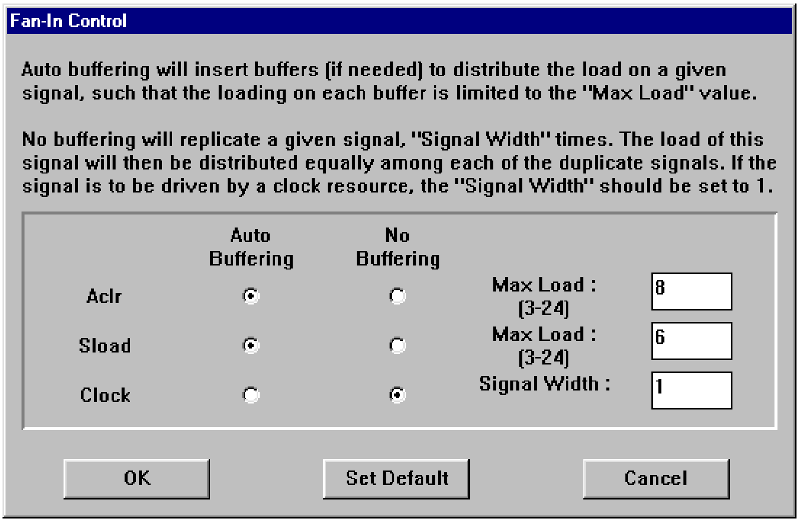
Fan-in Control Tool Guidelines

The Fan-in Control tool has the following limitations.

- The Fan-in Control tool has been designed to be a slave to the primary macro definition screen. Therefore, you should define exceptions to default values only after you have made all primary screen selections. Changing the main screen may affect the defined fan-in values. Information on modified fan-in will be provided in the Report window and should always be verified for correctness.
- The ability to perform no buffering on some control signals is limited to a single polarity because of hardware limitations. For example, ACT 2, 1200XL, ACT 3, 3200DX, 42MX, and 54SX limit asynchronous clears to Active Low only. Choosing Active High for this signal causes the No Buffering option to be unavailable. When this situation occurs, go back to the primary screen and change the active level for the given signal if no buffering is a must.
- Some control signals, such as the Count Enable signal are not included in the Fan-in Control tool because fan-out is corrected internally using AND and OR logic functions.

Using Fan-In Control

The Fan-in Control dialog box, shown in Figure 4-2, consists of three information/control columns. The first column defines the signal name. The second column specifies automatic or no buffering for the signal. The third column displays assigned buffering values where the actual value is entered. The description for this column changes depending on the type of buffering selected. A signal width value of one (1) causes all loads to be driven by a single input.


Fan-In Control

Auto buffering will insert buffers (if needed) to distribute the load on a given signal, such that the loading on each buffer is limited to the "Max Load" value.

No buffering will replicate a given signal, "Signal Width" times. The load of this signal will then be distributed equally among each of the duplicate signals. If the signal is to be driven by a clock resource, the "Signal Width" should be set to 1.

	Auto Buffering	No Buffering		
Aclr	●	○	Max Load : [3-24]	8
Sload	●	○	Max Load : [3-24]	6
Clock	○	●	Signal Width :	1

Figure 4-2. Fan-in Control Dialog Box

Generating a Macro Report

As ACTgen generates a macro it writes information to the ACTgen Report window. The report contains information defining the macro, and is divided into the following sections:

- **Macro Parameter.** This section lists the options selected to build the macro.
- **Fan-in Control Information.** This section defines the type of buffering for each control signal and the values used to distribute the total load.
- **Compile Report.** This section describes the number of sequential (flip-flop) resources, the total number of logic modules, and any I/O resources.
- **Timer Report.** This section describes the maximum delay from input I/O to output I/O, maximum delay from input I/O to internal

registers, maximum delay from internal registers to output I/O, and maximum delays for each clock network.

- **Output Format Information.** This section contains information specific to the selected output format. It shows the path for the "viewdraw.ini" file if Viewlogic is selected, or the path for the output netlist file.

To generate a .log file:

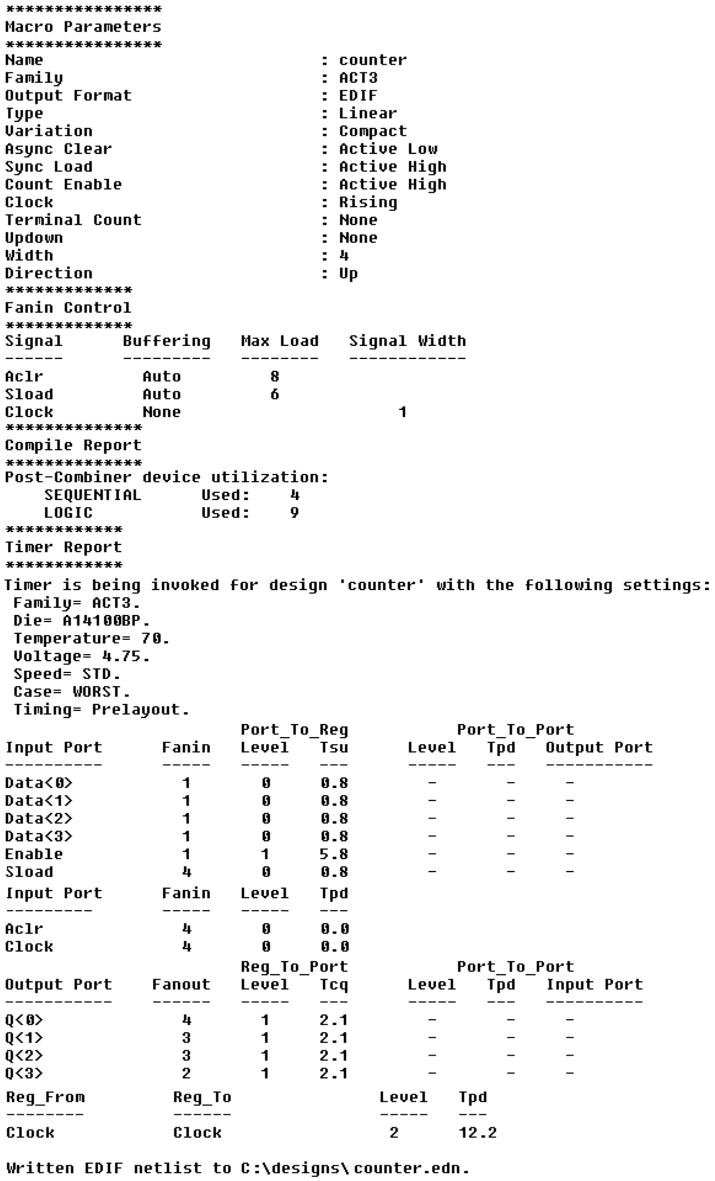
1. **Select Save As from the Reports menu or the toolbar.** The Save As dialog box is displayed.
2. **Enter the name of your report.** Type the path and name of your report in the File Name Box.
3. **Click OK.** The ACTgen Macro Builder generates the .log file. An example .log file is shown in Figure 4-3.

To print a .log file:

1. **Select Print from the Reports menu.** The report is queued to your local printer.
2. **Click OK.**

To disable/enable reports:

1. **Select Disable from the Reports menu.** This turns off the Compile and Timing reports, which provides the user with much faster macro Generation Capability.
2. **Select Enable from the Reports.** This turns the Timing and Compile reports on again.



```
****************
Macro Parameters
****************
Name                            : counter
Family                          : ACT3
Output Format                   : EDIF
Type                            : Linear
Variation                       : Compact
Async Clear                     : Active Low
Sync Load                       : Active High
Count Enable                    : Active High
Clock                           : Rising
Terminal Count                  : None
Updown                          : None
Width                           : 4
Direction                       : Up
*************
Fanin Control
*************
Signal      Buffering   Max Load   Signal Width
------      ---------   --------   ------------
Aclr          Auto         8
Sload         Auto         6
Clock         None                        1
**************
Compile Report
**************
Post-Combiner device utilization:
    SEQUENTIAL      Used:    4
    LOGIC           Used:    9
************
Timer Report
************
Timer is being invoked for design 'counter' with the following settings:
 Family= ACT3.
 Die= A14100BP.
 Temperature= 70.
 Voltage= 4.75.
 Speed= STD.
 Case= WORST.
 Timing= Prelayout.
                        Port_To_Reg           Port_To_Port
Input Port      Fanin   Level   Tsu      Level   Tpd   Output Port
----------      -----   -----   ---      -----   ---   -----------
Data<0>           1       0     0.8        -       -     -
Data<1>           1       0     0.8        -       -     -
Data<2>           1       0     0.8        -       -     -
Data<3>           1       0     0.8        -       -     -
Enable            1       1     5.8        -       -     -
Sload             4       0     0.8        -       -     -
Input Port      Fanin   Level   Tpd
---------       -----   -----   ---
Aclr              4       0     0.0
Clock             4       0     0.0
                        Reg_To_Port           Port_To_Port
Output Port    Fanout   Level   Tcq      Level   Tpd   Input Port
-----------    ------   -----   ---      -----   ---   ----------
Q<0>              4       1     2.1        -       -     -
Q<1>              3       1     2.1        -       -     -
Q<2>              3       1     2.1        -       -     -
Q<3>              2       1     2.1        -       -     -

Reg_From         Reg_To                Level   Tpd
--------         ------                -----   ---
Clock            Clock                  2      12.2

Written EDIF netlist to C:\designs\counter.edn.
```

Figure 4-3. ACTgen .log File

5

ACTmap VHDL Synthesis Tool

This chapter describes how to use the ACTmap VHDL synthesis tool to synthesize VHDL designs and optimize netlists for Actel devices. This includes information about implementing a hierarchical project and generating a netlist suitable for use in Designer. Refer to the *ACTmap VHDL Synthesis Methodology Guide* for additional information about using ACTmap VHDL.

ACTmap Features

ACTmap contains the following features:

Compile VHDL

ACTmap can compile VHDL and target designs for the Actel architecture. This includes incorporating ACTgen macros that have been instantiated into the VHDL and compiling hierarchical designs. Refer to “Compiling VHDL” on page 54 for information about compiling VHDL files and “Implementing a Hierarchical Project” on page 59 for information about instantiating macros and creating a project file.

Optimize Netlists

ACTmap can optimize a netlist for optimal performance using Actel devices. Refer to “Optimizing a Netlist” on page 55 for information.

Translate Netlists

ACTmap acts as a netlist translator by generating VHDL, Verilog, EDIF, or ADL netlists for Designer or for netlist symbol files. Refer to “Translating a Netlist” on page 57 for information.

Generate Symbols

ACTmap can generate symbols from VHDL for use in the Viewlogic schematic capture tool. Refer to “Adding ACTmap Blocks” in the *Viewlogic Workview Office Interface Guide* for information.

Automatic I/O Insertion

ACTmap automatically inserts global I/Os and buffers in all Actel family devices. During insertion, ACTmap inserts CLKBUF macros in all dangling clock network input ports. In addition, it inserts INBUF macros in all other dangling input ports, and OUTBUF macros in all dangling output ports.

Silicon Expert can also be used on netlists for further I/O insertion.

Sequential Remapping

For almost all ACT 3 flip-flops, and for some ACT 2 flip-flops, ACTmap performs pre-optimized, sequential remapping. The sequential remapping enhances the optimizer performance to take advantage of combinatorial and sequential combining features. It divides sequential library elements into smaller and more basic elements that may generate better results during optimization. Sequential remapping applies to both HDL synthesis and optimization.

State Machine Encoding Algorithms

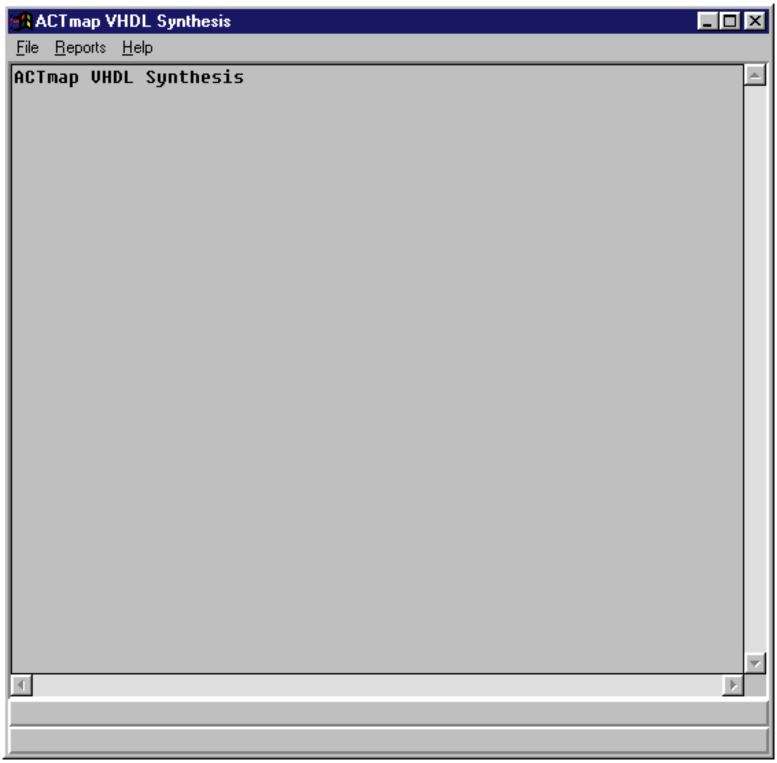
You can select between six state-machine encoding algorithms. ACTmap uses the algorithms to synthesize state machines in ACTmap VHDL source files to netlists. The following encoding algorithms are available; Compact, OneHot, Gray, Johnson, Sequential and User.

ACTmap Windows

ACTmap is divided into four windows. Each window performs a different function. This section describes the windows. Refer to the ACTmap on-line help for additional information.

ACTmap Main Window

The ACTmap Main window, shown in Figure 5-1, displays report-type information when an action is performed in ACTmap.



```
ACTmap VHDL Synthesis
```

Figure 5-1. ACTmap Main Window

VHDL Compiler Window

The VHDL Compiler window allows designers to read in a VHDL file and compile it for a specific Actel family architecture.

Netlist Optimizer Window

The Netlist Optimizer window allows designers read in an EDIF, ADL, or Verilog netlist and optimize it for a specific Actel family architecture. The optimization output file is an EDIF netlist.

Netlist Translate Window

The Netlist Translate window allows designers read in an EDIF netlist generated during optimization, merge this netlist with any existing EDIF, ADL, or Verilog files, and generate the output in the selected file format.

Compiling VHDL

ACTmap can compile behavioral, RTL, or structural VHDL design files for a specified Actel family architecture. Use the following procedure to compile a VHDL file.

1. **Invoke ACTmap.**

   PC

   Select ACTmap VHDL Synthesizer from the Designer menu under Programs in the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   actmapw
   ```

   The ACTmap Main window, shown in Figure 5-1, is displayed.
2. **Open the VHDL Compiler window.** Choose the VHDL Compiler command from the File menu.
3. **Select the VHDL file to compile.** Type the full path name of the design in the Source Design box or click the Browse button.
4. **Select a Map Style.** Area optimizes for the smallest number of logic blocks. Speed optimizes for the fastest path through a device.
5. **Specify a target family in the Family pull-down menu.**
6. **Specify a State Encoding type in the State Encoding pull-down menu.** The State Encoding type specifies the Finite State Machine description used to translate an ACTmap VHDL source file into a netlist.

7. **Select the optimization Mode.** Use the Block mode if you do not want to add any I/Os. Use the Chip mode to add basic I/Os, Tristate I/Os, and Bidirectional I/Os to a design block.

   Note: If I/Os already exist in the design, ACTmap does not add additional I/Os.

8. **Specify the Max Fanout.** Specify a number in the Max Fanout box to set the maximum fanout limit during netlist mapping. The fanout range is from 2 to 24. The default option is 16 for all devices. If you choose an integer greater than 24, ACTmap generates an error message.

9. **Select the Flatten option.** If the option is set to On, the netlist is compiled without hierarchy. If the option is set to Off, ACTmap preserves the hierarchy during compilation.

10. **Compile the design.** Click the Run button. ACTmap automatically enters a file name in the Target Design text box. The name is the same as the source design, but with an .edn suffix. Compiled designs can immediately be translated to the desired output format by selecting the Translate button, or can be further optimized by selecting the Optimize button.

Optimizing a Netlist

ACTmap can optimize EDIF, ADL, or Verilog netlist for a specific Actel family architecture. Use the following procedures to optimize a netlist.

1. **Invoke ACTmap.**

   PC

   Select ACTmap VHDL Synthesizer from the Designer menu under Programs in the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   actmapw
   ```

   The ACTmap Main window, shown in Figure 5-1, is displayed.

2. **Open the Netlist Optimizer window.** Choose the New Design command from the File menu and open an ADL or EDIF netlist through the Read Design dialog box. Go to step 4.

   or

   Choose the Netlist Optimizer command from the File menu.

   The Netlist Optimizer window can also be accessed through the VHDL Compiler and Netlist Translate windows.

3. **Select the netlist to optimize.** Type the full path name of the netlist in the Source Design box or click the Browse button.

4. **(Optional) Enter a path and file name in the Target Design box.** If you leave this blank, ACTmap automatically generates an .edo file with the same name as the Source Design when the netlist is optimized.

5. **Select a Map Style.** Area optimizes for the smallest number of logic blocks. Speed optimizes for the fastest path through a device.

6. **Specify a target family in the Family pull-down menu.**

7. **Specify the format of the source netlist in the Input Format pull-down menu.**

8. **Select the optimization Mode.** Use the Block mode if you do not want to add any I/Os. Use the Chip mode to add basic I/Os, Tristate I/Os, and Bidirectional I/Os to a design block.

   Note: If I/Os already exist in the design, ACTmap does not add additional I/Os.

9. **Specify the Max Fanout.** Specify a number in the Max Fanout box to set the maximum fanout limit during netlist mapping. The fanout range is from 2 to 24. The default option is 10 for ACT 1 and 40MX, and 16 for all other devices. If you choose an integer greater than 24, ACTmap generates an error message.

10. **Optimize the netlist.** Click the Run button. ACTmap optimizes the netlist for the target family using the options set in the Netlist Optimizer window.

Translating a Netlist

ACTmap acts as a netlist translator by generating the VHDL, Verilog, EDIF, or ADL netlists for the Designer Series or the Viewlogic tools. ACTmap merges an optimized EDIF netlist with existing netlist files or ADL files, and generates the desired output file format. Use the following procedure o translate a netlist.

1. **Invoke ACTmap.**

   PC

   Select ACTmap VHDL Synthesizer from the Designer menu under Programs in the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   actmapw
   ```

   The ACTmap Main window, shown in Figure 5-1, is displayed.

2. **Open the Netlist Translate window.** Choose the Translate Netlist command from the File menu.
3. **Select the netlist to translate.** Type the full path name of the netlist in the Source Design box or click the Browse button.
4. **Specify a target family in the Family pull-down menu.**
5. **Specify the format of the source netlist in the Input Format pull-down menu.**
6. **Specify the format of the target netlist in the Output Format pull-down menu.**
7. **Translate the netlist.** Click the Run button. ACTmap automatically enters a file name in the Target Design text box. The name is the same as the source design, but with an .opt (ADL or Designer), .edt (EDIF), .vho (VHDL), or .vlo (Verilog) suffix.

Defining I/Os

Defining I/Os is an optional procedure that allows designers to define which signals to use during clock buffer insertion. This command is only available in the VHDL Compiler and Netlist Optimizer window, and only if the Chip optimization mode is selected. ACTmap names and inserts I/O macros differently for each Actel device family. Use the following procedure to define clock buffers:

1. **Invoke ACTmap.**

   PC

   Select ACTmap VHDL Synthesizer from the Designer menu under Programs in the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   actmapw
   ```

   The ACTmap Main window, shown in Figure 5-1, is displayed.

2. **Open the VHDL Compiler window or Netlist Optimizer window.** Choose the VHDL Compiler command or the Netlist Optimizer command from the File menu.
3. **Open the Define I/Os dialog box.** Choose the Define I/Os command from the Options menu. The options available depend on the device family that is selected. ACT 1 and 40MX display one clock, “CLK.” ACT 2/1200XL, 3200DX, and 42MX display two clocks, “CLKA” and “CLKB.” ACT 3 and 54SX display “CLKA,” “CLKB,” and “HCLK.”
4. **Enter the clock port name(s).**
5. **Click OK.** CLKBUF macros are inserted when the Run button is clicked.

Implementing a Hierarchical Project

ACTmap can compile all of the VHDL files that make up a design into a single project using a project file, including ACTgen macros that are written out in VHDL. The files are then compiled together, preserving hierarchy and resolving fanout. Follow the steps below to create a project using the project file:

1. **Write the VHDL descriptions of your design.** Multiple files may be used to describe different blocks and levels of hierarchy.

   ACTgen macros that are included should be saved in VHDL format and instantiated into the VHDL. Each macro needs a component declaration and a port map statement to properly instantiate them. Refer to “Generating Macros Using ACTgen” on page 41 for information about creating macros with ACTgen and the *ACTmap VHDL Synthesis Methodology Guide* for additional information about writing VHDL for ACTmap.

2. **Create a project file.** Use a text editor to enter the names of all the files used to describe the design including VHDL files generated by ACTgen. The order that the files are listed in the project file is not important.

3. **Save the project file.** Use the top-level design name as the file name prefix and “.prj” as the file name suffix. Make sure the “.prj” file is in the same directory as the VHDL design files. ACTmap only searches the directory where “.prj” file is located for the VHDL design files. Below is an example project file for a design whose top level design name is top:

   ```
   top.vhd
   fsm.vhd
   counter.vhd
   ```

4. **Invoke ACTmap.**

   PC

   Select ACTmap VHDL Synthesizer from the Designer menu under Programs in the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   actmapw
   ```

   The ACTmap Main window, shown in Figure 5-1, is displayed.

5. **Open the VHDL Compiler window.** Choose the VHDL Compiler command from the File menu.
6. **Select the project file as the Source Design.** Type the path and name of the project file In the Source Design box, or click the Browse button.
7. **Select Chip as the optimization Mode.** CLKBUF macros are added to the clk ports, INBUF macros are added to the input ports, and OUTBUF macros are added to the output ports.
8. **Specify a target family in the Family pull-down menu.**
9. **Specify No as the Flatten option.** With the option set to Off, ACTmap preserves the hierarchy during compilation.
10. **Compile the design.** Click the Run button. ACTmap automatically enters a file name in the Target Design text box. The name is the same as the source design, but with an .edn suffix. All the design files listed in the top.prj file are compiled. ACTmap resolves the fanout between blocks and hierarchical compilation.

Configuration Files

A configuration file is a file that sets all options in any or all ACTmap windows. Configuration files allow designers to set the same compilation, optimization, and translation options for all designs. Once a configuration file is created it can be open and used every time a design is opened in ACTmap. This section describes the procedure for creating and opening a configuration file.

To create a configuration file:

1. **Invoke ACTmap.**

   PC

   Select ACTmap VHDL Synthesizer from the Designer menu under Programs in the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   actmapw
   ```

   The ACTmap Main window, shown in Figure 5-1, is displayed.

2. **Set the desired options in all of the ACTmap windows.** Close each window after setting the options.

3. **Save the configuration file.** Choose the Save All Configurations command from the File menu in the ACTmap Main window. The Save Configuration dialog box is displayed. Enter the name of your configuration file. Click OK.

4. **Specify which ACTmap window options to save.** In the Save All Configurations dialog box select one, multiple, or all windows. Click OK. ACTmap saves configuration files with an “.ami” extension.

To open a configuration file:

1. **Invoke ACTmap.**

   PC

   Select ACTmap VHDL Synthesizer from the Designer menu under Programs in the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   actmapw
   ```

   The ACTmap Main window, shown in Figure 5-1, is displayed.

2. **Open a configuration file.** Choose the Open Configuration command from the File menu. The Open Configuration dialog box is displayed. Type the full path name of the configuration file or browse to it and select it. Configuration files have an “.ami” extension. Click OK.

   ACTmap reads in the configuration file settings and displays both the name of the configuration file in the title bar and settings in the ACTmap window(s).

Using ACTmap in Batch Mode

ACTmap can be used in batch mode. Refer to *ACTmap VHDL Synthesis Methodology Guide* for information.

6

Design Implementation Using Designer

This chapter describes how to use the Designer place and route software to optimize and implement designs to program Actel devices.

Importing a Netlist/Compiling a New Design

You must import a netlist and compile it into an ADB file before working on a new design. The following steps describe the procedure.

1. **Invoke Designer.** The Designer Main window is displayed.

   PC

   Choose Designer from the Designer group in the Programs menu under the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   designer
   ```

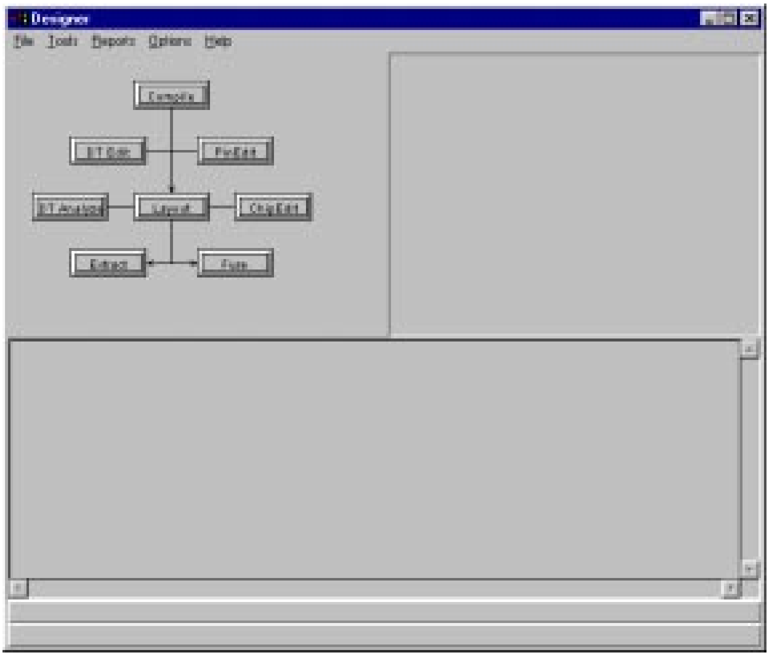
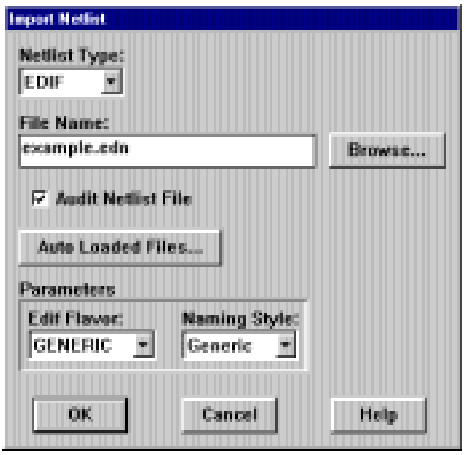

Designer
File Tools Reports Options Help
Compile
DT Edit
PinEdit
DT Analyze
Layout
ChipEdit
Extract
Fuse

Figure 6-1. Designer Main Window

2. **Open the Import Netlist dialog box.** Choose the Import Netlist File command from the File menu. You can also click the Compile button or another button in the Main window. Click New when asked if you would like to import a netlist or open an existing design. The Import Netlist dialog box is displayed, as shown in Figure 6-2.



Import Netlist

Netlist Type: EDIF

File Name: example.edn

☑ Audit Netlist File

Parameters

Edif Flavor: GENERIC

Naming Style: Generic

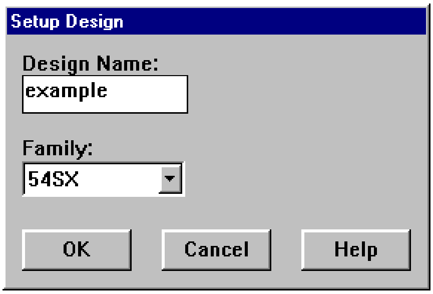
Figure 6-2. Import Netlist Dialog box

3. **Specify netlist options.** Specify the type of netlist to import in the Netlist Type pull-down menu. Select your netlist by typing the full path name or clicking the Browse button. Select Edif Flavor and Naming Style (if necessary).

   The default EDIF flavor is “GENERIC.” If your EDIF is generated from a specific CAE environment, choose the appropriate flavor to maintain CAE specific properties within the netlist. The default Naming Style is “Generic.” If the naming style is set to VHDL or Verilog, the names of nets, instances, and ports from an EDIF or ADL netlist are modified as necessary to conform to VHDL or Verilog naming conventions.

4. **(Optional) Import auxiliary files.** If you are using auxiliary files (DCF, PIN, CRT), click the Auto Loaded Files button to select the files. Refer to “Importing (DCF), (PIN), and (CRT) Information” on page 70 for additional information.

5. **Setup the design.** Click OK in the Import Netlist dialog box. The Design Setup dialog box is displayed. Specify the design name and family. Click OK.



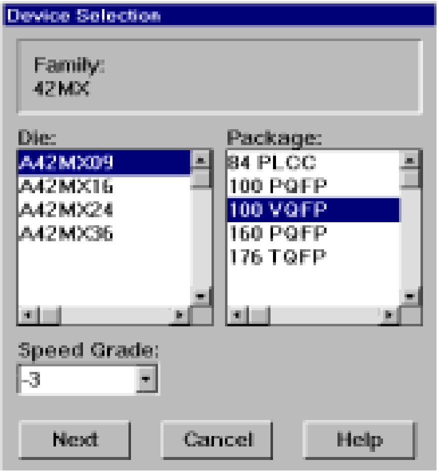
Design Name: example

Family: 54SX

Figure 6-3. Setup Design Dialog Box

6. **Select device and package.** In the Device Selection dialog box, specify Die Package, and Speed Grade. You must specify Die and Package to continue. Refer to “Device, Package, and Speed Grade” on page 72 for additional information. Click Next.

   Note: If you import a netlist using the Import Netlist File command, you must click a button in the Designer Main window to display the Device Selection dialog box.



Family: 42MX

Die:
A42MX09
A42MX16
A42MX24
A42MX36

Package:
84 PLCC
100 PQFP
100 VQFP
160 PQFP
176 TQFP

Speed Grade: -3

Figure 6-4. Device Selection Dialog Box (Standard)

7. **(Optional) Upgrade your compatible SX device to an SXA device without re-compiling.** Select 54SXA from the Change To drop-down menu in the Device Selection dialog box.

   If you selected an SX family device, you will not see the dialog box displayed above. An SX family device may be upgraded to a compatible SXA family device without re-compiling. If you chose an SX device in the Setup Design dialog box (Figure 6-3), you will see the SX/SXA Device Selection dialog box displayed below in Figure 6-5.

   Note: Devices must be compatible for this feature to operate properly. Even though the layout information is preserved when you switch from a (compatible) die of SX to that of SXA family, you will need to regenerate the fuse files.If the dies are different, then the layout will be lost.

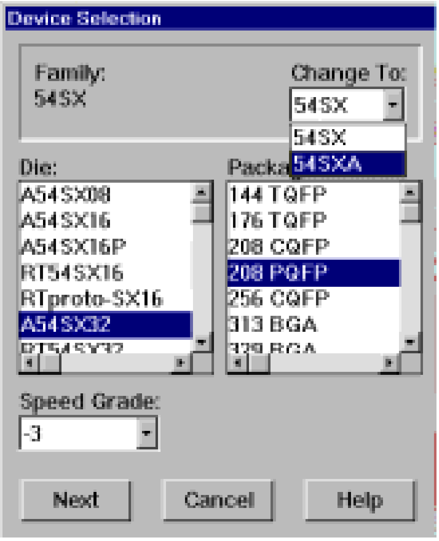
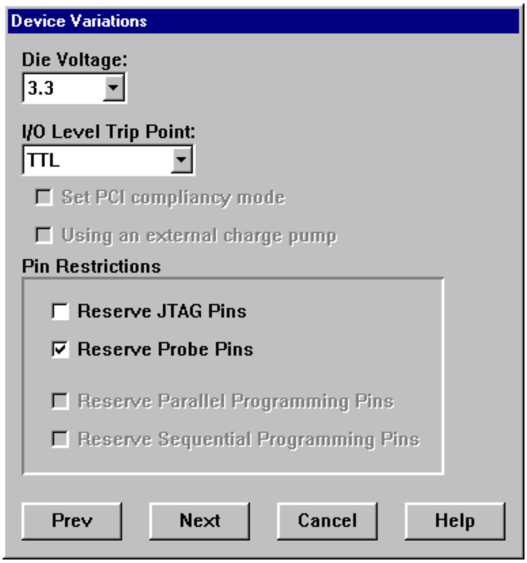

Family: 54SX
Change To: 54SX
Die: A54SX32
Package: 208 PQFP
Speed Grade: -3
Device Selection
54SX
54SXA
A54SX08
A54SX16
A54SX16P
RT54SX16
RTproto-SX16
A54SX32
RT54SX32
144 TQFP
176 TQFP
208 CQFP
208 PQFP
256 CQFP
313 BGA
329 BGA
Next
Cancel
Help

Figure 6-5. Device Selection Dialog Box (SX/SXA)

8. **Set device variations.** In the Device Variations dialog box, specify Die Voltage, options, and Pin Restrictions. Refer to “Device Variations” on page 73 for additional information. Click Next.



Device Variations

Die Voltage: 3.3

I/O Level Trip Point: TTL

☐ Set PCI compliancy mode

☐ Using an external charge pump

Pin Restrictions

☐ Reserve JTAG Pins

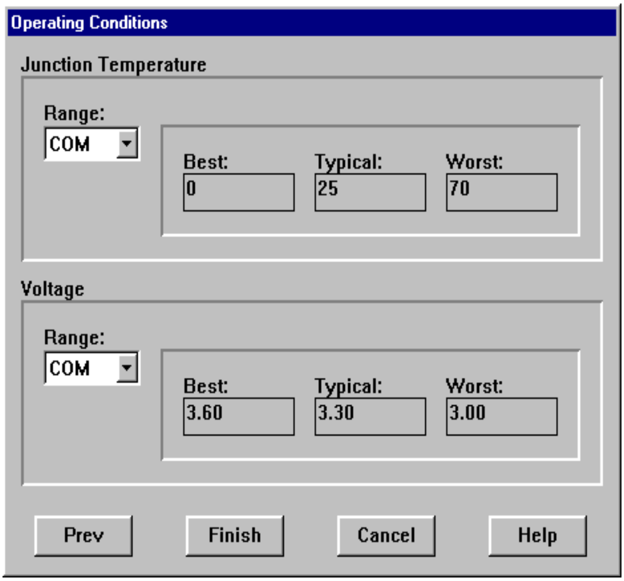
☑ Reserve Probe Pins

☐ Reserve Parallel Programming Pins

☐ Reserve Sequential Programming Pins

Figure 6-6. Device Variations Dialog Box

9. **Set operating conditions.** In the The Operating Conditions dialog box, specify Junction Temperature Range and Voltage Range. Refer to “Operating Conditions” on page 74 for additional information. Click Finish. Designer compiles the design.



Operating Conditions

Junction Temperature

Range: COM

Best: 0

Typical: 25

Worst: 70

Voltage

Range: COM

Best: 3.60

Typical: 3.30

Worst: 3.00

Figure 6-7. Operating Conditions Dialog Box

During compile, the message window in the Main window displays information about your design, including warnings and errors. Designer issues warnings when your design violates recommended Actel design rules. Actel recommends that you address all warnings, if possible, by modifying your design before continuing.

If the design fails to compile due to errors in your input files (netlist, constraints, etc.), you must modify the design to remove the errors. You must then re-import and re-compile the files. Refer to the Designer on-line help for a list of compile warning and error codes.

After the design is compiled, you can run Layout to place and route the design or you can use the other Designer features, (DT Edit, PinEdit, DT Analyze, or ChipEdit) to perform additional optimization prior to place and route.

Opening an Existing Design

Once you have imported a netlist and compiled a design, you can save the design as an ADB file. You can then open the ADB file, skipping the compile step, and perform optimization on the design, including updating netlist and auxiliary file information.

Designer can open unmodified designs and designs with netlists that have been modified since the last time it was opened in Designer. If you have modified your netlist since the ADB file was created, Designer detects this and prompts you to re-import it. Designer can also audit auxiliary files when the netlist is changed and prompts you to re-import these files as well. Changes can even be made to the netlist when the ADB file is open. Designer notifies you that the netlist has changed and automatically re-imports the netlist.

Designer can also open designs created in previous versions of Designer. Refer to "Designs Created in Previous Versions of Designer" on page 69 for additional information.

Designs Created in Previous Versions of Designer

Designer can directly open designs created using software versions 1.22 and later, except for versions 3.0 and 3.1.

Designs created in versions 3.0 or 3.1 must be converted to be compatible with the current version of Designer. Refer to the *Designer Series Development System Conversion Guide* for information about converting these designs.

Designs created in versions other than 3.0 or 3.1 can be opened directly into Designer using the procedure described in the next section. The design information contained in existing files is read in to function with the new software. This conversion occurs automatically when opening an ".als" or ".def" file.

All existing die, package, pin assignment, and placement routing information is read and maintained. Designs created in previous versions of software may need library conversions when loaded into the Designer environment. If your design requires this conversion, you will be prompted to allow the software to update the design to the new library before you attempt to launch any of the Designer features.

To open an existing design:

1. **Invoke Designer.** The Designer Main window is displayed (see Figure 6-1 on page 63).

   PC

   Choose Designer from the Designer group in the Programs menu under the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   designer
   ```

2. **Open the design.** Choose the Open command from the File Menu. The Open dialog box is displayed. Type the full path name of the ADB file (DEF file if your design was created in Designer prior to Designer 3.0) you want to open or browse to the file and select it.

   The software checks time stamps on the netlist. If the netlist has not been modified, The ADB file is opened. If the netlist has been modified, Designer prompts you to re-import the updated netlist. The ADB file is opened and the new netlist is merged with existing design information.

Importing (DCF), (PIN), and (CRT) Information

Designer contains an option to import additional design information. Most design methodologies do not require this additional step. However, if you are using synthesis tools and want to import a delay constraint file (DCF), criticality (CRT), or I/O pin assignment (PIN) file, they can be imported when you import a netlist or after the netlist/ ADB has been opened.

To import files when netlist is imported:

1. **Import the netlist.** Refer to “Importing a Netlist/Compiling a New Design” on page 63.
2. **Open the Auto Loaded Files dialog box.** Click the Auto Loaded Files button.

3. **Select the auxiliary file(s) to import.** Select your auxiliary file(s) by typing the full path name in the box under the type of file(s) you want to import or the Browse button. Check the Audit File box to force an import of the auxiliary file, regardless of changes to the netlist file. Click OK.

To import files after the netlist/ADB file has been opened:

1. **Choose the Import Auxiliary File command from the File menu.** The Import Auxiliary File dialog box is displayed.
2. **Select an auxiliary file to import.** Specify the type of file to import in the File Type pull-down menu and type the full path name of the auxiliary file or use the Browse button.

Changing Design Name and Family

Design name and family are set when a netlist is initially imported and a new design is compiled. However, you can change this information for existing designs. If you change the family, Designer notifies you that you must re-import the netlist and automatically prompts you when you select the next Designer function. Use the following procedure to change the name of a design and the targeted Actel family for the design.

1. **Choose the Setup Design command from the Options menu in the Designer Main window.** The Setup Design dialog box is displayed (see Figure 6-3 on page 65).
2. **Specify the design name and family.** Click OK. Refer to the Actel *FPGA Data Book* for Actel Family specifications.

Changing Other Design Information

Device and package information, device variations, and operating conditions are set when a netlist is initially imported and a new design is compiled. However, you can change this information for existing designs using the Device Setup Wizard.

To change design information for existing designs:

Choose the Device Setup Wizard command from the Options menu in the Designer Main window. Designer prompts you with a number of dialog boxes. Use the Next button to move to the next dialog box after you change information or if you do not want to change any information in that dialog box. The Device Setup Wizard is made up of the Device Selection, Device Variations, and Operating Conditions dialog boxes.

Refer to the Actel *FPGA Data Book* or call your local Actel Sales Representative for information about device, package, speed grade, variations, and operating conditions.

Device, Package, and Speed Grade

Use the Device Selection dialog box (see Figure 6-4 on page 65) to specify or change the device and package type and the speed grade based on your design needs.

Select a device. Available packages are then displayed. Select a package. Specify a speed grade in the Speed Grade pull-down menu.

Devices that are no longer available from the Device Selection dialog box can be selected using Designer Script. Because these parts may no longer be available, do not use these devices unless you have the parts. Refer to “Using Designer Script” on page 125 for information about using Designer Script.

Compatible Die Change

When the device is changed, some design information can be preserved depending on the type of change.

Changing Die Revisions

If the die is changed from one technology to another, all information except timing is preserved. An example is changing an A1020A (1.2um) to an A1020B (1.0um) while keeping the package the same.

Device Change Only

Constraint and pin information is preserved, when possible. An example is changing an A1240A in a PL84 package to an A1280A in a PL84 package.

Repackager Function

When the package is changed (for the same device), the Repackager automatically attempts to preserve the existing pin and Layout information by mapping external pin names based on the physical bonding diagrams. This will always work when going from a smaller package to a larger package (or one of the same size). When changing to a smaller package, the Repackager will determine if any of the currently assigned I/Os are not bonded-out on the smaller package. If some of the I/Os are not bonded-out, then the layout is invalidated and the unassigned pins identified.

Speed Grade

Use the Speed Grade pull-down menu to select an available speed grade for the selected device and package. If a desired speed grade is unavailable, then it may not be supported for that device.

Device Variations

Use the Device Variations dialog box (see Figure 6-6 on page 67) to specify or change die variations based on your design needs.

Die Voltage

Use the Die Voltage pull-down menu to set 5.0, 3.3/5.0, or 3.3 Volts. To set the die core to 5.0 Volts and the I/Os to 3.3 Volts, select 3.3/5.0.

PCI Compliance

To utilize the high-current drive buffers in the Multiplex I/Os and meet PCI specifications, check the Set PCI compliancy mode box.

Pin Restrictions

To avoid using the JTAG pins “TDI,” “TMS,” “TCK,” and “TDO” during layout, check the Reserve JTAG Pins box. To avoid using the Probe pins “PRA,” “PRB,” “SDI,” and “DCLK” during layout, check the Reserve Probe Pins box.

Operating Conditions

Use the Operating Conditions dialog box (see Figure 6-7 on page 68) to define the voltage and temperature ranges a device encounters in a working system. It supports standard industry temperature and voltage ranges, including commercial (COM), industrial (IND), and military (MIL). In addition, you can set fully custom ranges by specifying Custom in the pull-down menu. The operating condition range entered in the Operating Conditions dialog box is used by DT Analyze, the timing report, and the back annotation function. These tools provide the capability to analyze worst, typical, and best case timing. The operating conditions used are shown in the following table.

Table 6-1. Operating Conditions

Timing	Process	Temperature	Voltage
Best Case	Best	Best	Best
Typical Case	Typical	Typical	Typical
Worst Case	Worst	Worst	Worst

The temperature range represents the junction temperature of the device, which is a function of ambient temperature, air flow, and power consumption. Because Actel devices are CMOS, power consumption must be calculated for each design. For most low power applications (e.g. 250mW), the default conditions should be adequate. Junction temperature can be calculated from values in the Actel *FPGA Data Book* or by using Table 6-2 for example values. Performance should decrease about 2.5% for every 10 degrees C that the temperature value is increased.

Temperature Range

Use the pull-down Range box to select COM, IND, MIL or Custom. If you select Custom, then the Best, Typical, and Worst fields become editable. You can then modify the range to the desired value (integer) such that Best ≤ Typical ≤ Worst.

Voltage Range

Use the pull-down Range box to select COM, IND, MIL or Custom. If you select Custom, then the Best, Typical, and Worst fields become editable. You can then modify the range to the desired value (real) such that Best ≥ Typical ≥ Worst.

Junction Temperature Table

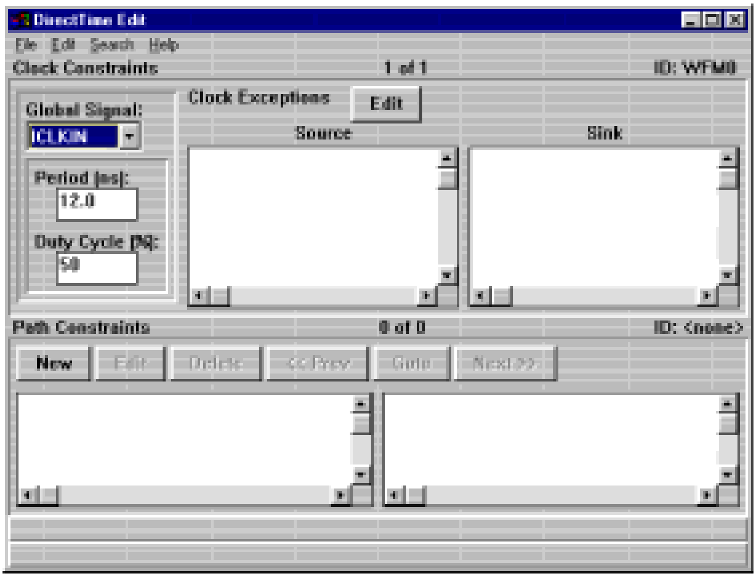
The following information defines the delta increase in temperature the junction temperature will be over the ambient temperature depending on package type and air flow. Actual junction temperature is simply the expected ambient temperature plus the value shown in the following table. You can calculate junction deltas for additional wattages by simply multiplying the actual wattage (in watts) times the value in the table.

Table 6-2. Junction Temp. Deltas at 1W Power Consumption

Package Type	**Still Air (in degrees C)**	**300 ft/min (in degrees C)**
44 PLCC	52	40
68 PLCC	45	35
84 PLCC	44	38
100 PQFP	55	47
144 PQFP	35	26
160 PQFP	33	26
208 PQFP	33	26
208 RQFP	17	13
80 VQFP	68	55
100 VQFP	43	35
176 TQFP	32	25
225 BGA	25	21
313 BGA	23	18
84 CPGA	33	20
100 CPGA	35	17
132/133 CPGA	30	15
175/176 CPGA	25	14
207 CPGA	22	13
256 CPGA	15	8
84 CQFP	40	30
132 CQFP	55	30
172 CQFP	25	15
196 CQFP	36	24
265 CQFP	30	18

DT Edit

DT Edit allows you to define clock constraints and exceptions and path constraints for your design. These constraints are used to generate timing reports, back annotate timing information, and for use in DT Layout. Although information entered in DT Edit is not used during Standard Layout, Actel recommends that you use DT Edit to enter constraints for use in generating timing reports and back annotation. If you use DT Layout you must enter constraints in DT Edit. The DT Edit window is shown in Figure 6-8. Refer to the Designer online help for additional information about using DT Edit.


DirectTime Edit
File Edit Search Help
Clock Constraints
1 of 1
ID: WFM0
Clock Exceptions
Edit
Source
Sink
Path Constraints
0 of 0
ID: <none>
New
Edit
Delete
<< Prev
Goto
Next >>

Global Signal: ICLKIN

Period [ns]: 12.0

Duty Cycle [%]: 50

Figure 6-8. DT Edit Window

Clock Constraints and Exceptions

The Clock Constraints section allows you to assign values to each clock network in your design. Values can be defined by period. Clock Exceptions indicate that the defined period is not applicable for the defined pins (which can either be the start or end of a network). Exceptions are assumed to be don’t care paths (or path length infinite) unless assigned otherwise under the Path Constraint section.

To assign clock constraints and exceptions:

1. **Invoke DT Edit.** Click the DT Edit button in the Designer Main window.
2. **Select the clock of interest in the Global Signal pull-down menu.**
3. **Define the Period and Duty Cycle.**
4. **Click the Clock Exceptions EDIT button.**
5. **Define source or sink points.**
6. **Select nodes from the left column.**
7. **Select ADD to define these nodes as exceptions.**
8. **Click OK.**

Path Constraints

Path Constraints section allows you to define constraints for specific paths in your design. Path constraints can be defined in terms of either network starting points or ending points.

To assign path constraints:

1. **Invoke DT Edit.** Click the DT Edit button in the Designer Main window.
2. **Click the New button.**
3. **Configure the arrow direction to define the master and slave directions.**
4. **Select masters points of interest.** Check boxes should be configured first. Networks can be defined in terms of either pins or nets. A name filter is also provided at the bottom of the list. Type in the filter box (U1/* for example) and press Enter to activate the filter. Click the SA button to select all pins or nets.
5. **Select corresponding slaves.** Check boxes should be configured first. A name filter is also provided at the bottom of the list. Type in the filter box (U1/* for example) and press Enter to activate the filter. Click the SA button to select all pins or nets.
6. **(Optional) Use the “Excepted Only” checkboxes to assign timing constraints to paths that have been designated as**

clock exceptions. Unless this option is selected, timing constraints will not be allowed on paths that are listed as clock exceptions.

7. **Define the delay.** Enter the delay in the Delay box.
8. **Click OK.**

Conflicts

The Path Constraint Editor allows you to enter multiple values for the same path. The tools consider the tightest constraint to be the one of interest.

DT Edit Guidelines

Delay constraints control the DT Layout engine. You can define these constraints using DT Edit or by using an external DCF file, which must be imported. The DT Layout engine considers the defined delays when allocating silicon resources with the goal of meeting or beating all constraints if possible. The DT Layout engine does this by automatically considering the performance criticality of one function versus another when allocating device resources. Because resources are limited, use the following guidelines to ensure the defined constraints meet the needs of the design without negatively impacting device resources.

Set Sufficient Constraints

All constraints for the design should be defined to ensure correct operation of the DT Layout engine. DT Layout considers networks that have not been defined to be don’t care paths, and will have a low priority for resource allocation. If these undefined paths are actually critical, they may fail to meet performance demands. Incrementally adding constraints may then cause incremental performance problems in other undefined networks.

Avoid Unnecessary Constraints

Non-critical paths should also be described to free high performance device resources. Not defining a path is one mechanism for doing this. However, it is difficult to avoid defining some don’t care paths, so Designer provides clock exceptions and global stop sets to enhance this capability.

Avoid Overconstraining

The DT Layout engine is designed to achieve the delay constraint defined (less than or equal). Defining a constraint shorter than the actual requirement for margin can have a negative impact on the performance of the device. The defined network or other networks may not meet requirements because of competition for device resources.

For example, consider a design that has a 40MHz clock (25ns period). When the constraint is set at 23.0ns, the device runs out of overall resources, making the 23.0ns goal impossible. In fact, the results can actually be worse than what was originally required because of the allocation method employed by the DT Layout engine, as shown for the 22.0ns case in Table 6-3 below.

Table 6-3. Constraint Results

Defined Constraint	Post DTL Result
25.0ns	24.8ns
24.8ns	23.9ns
23.0ns	24.5ns
22.0ns	25.2ns

Limitations

DT Layout cannot currently handle minimum delay or skew requirements.

Delay Constraint Definitions

A description of delay constraint methodology requires the following terms:

DTL terminals. DT Layout (DTL) terminals define the starting and ending points for a signal path. They are always I/Os or sequential elements. No intermediate combinatorial element is currently supported as a terminal.

Signal Path. The signal path describes a consecutive sequence of logic macros and nets, the first net being driven by a start terminal, and the last net driving a macro input pin of the end terminal.

Network. A network can consist of 1 or more start terminals and 1 or more end terminals. All signal paths connecting any start terminal to any end terminal are included in the network.

Only one delay value can be assigned to each defined network. Networks can be defined implicitly by a common clock (synchronous network) or explicitly by a defined set of terminals. Network and Paths are used interchangeably.

Path Delay. The path delay defines the sum of all the individual delays of the nets and the logic macros in the signal path.

Delay Constraint. A delay constraint defines a fixed amount of time required for a signal to propagate from all starting terminals to all ending terminals for a network.

Don't Care Path. A signal path in which the delay is considered to be infinite.

Global Stop. A defined intermediate point in a network that forces all paths through the defined point to be don't care paths regardless of any constraint assignment.

Clock Exception. A terminal in a synchronous network that should be excluded from the specified clock period. The exception can remain undefined (don't care) or can be assigned a unique value in the Path Constraint Editor.

Assigning Pins

There are two methods for I/O signal placement. You can let Designer automatically assign I/O locations during Layout, or you can manually assign I/O locations prior to Layout.

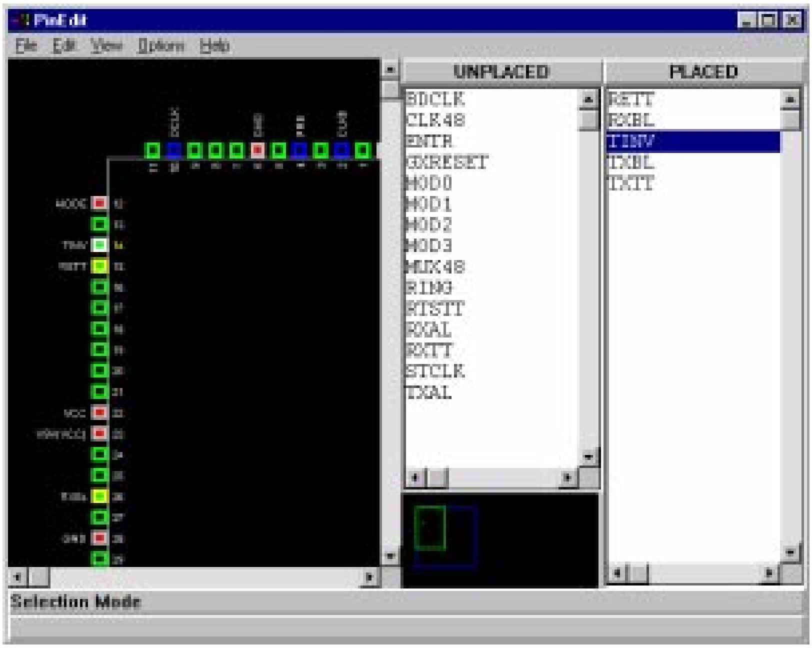
Actel recommends that you let Designer to automatically assign I/O locations during Layout. Layout is designed to place the I/Os for optimum routability and performance. Refer to “Layout” on page 84 for information about automatically assigning I/O locations during place and route.

If you must manually assign the I/O locations, you can assign I/O locations in your design schematic, in a pin file that you import into Designer, or using PinEdit. Refer to documentation included with your

CAE tools for information about assigning I/O signal placement in a schematic or in pin file. Refer to “PinEdit” on page 81 for information about using PinEdit to manually assign I/O locations.

PinEdit

PinEdit allows you to position I/O signals on device pins using list boxes and a graphical representation of the device. The PinEdit window is shown in Figure 6-9.



UNPLACED

BDCLK
CLK48
ENTR
GXRESET
MOD0
MOD1
MOD2
MOD3
MUX48
RING
RTSTT
RXAL
RXTT
STCLK
TXAL

PLACED

RETT
RXBL
TINV
TXBL
TXTT

Figure 6-9. PinEdit Window

The Pin window on the left displays a graphic representation of the pins on the device. When you select an assigned pin in the Pin window, the selected pin is highlighted in the PLACED list box. The small window under the UNPLACED list box is a position window that allows you to change what part of the device you are viewing.

The PLACED and UNPLACED list boxes display a list of placed or unplaced I/O signals in the design. Both fixed and unfixed pins are displayed in the PLACED list box.

Use the Configure List Boxes dialog box under the Options menu to specify how the pin information displayed in PinEdit.

Assigning I/O Signals to Pins

Use the following procedure to assign unplaced I/O signal to pins using Pinedit.

1. **Invoke PinEdit.** Click the PinEdit button in the Designer Main window.
2. **Assign the I/O signal name to a pin.** Select the signal name in the UNPLACED list box that you want to place, drag the signal name to the pin location in the Pin window where you want to place it, then release the mouse button.

To unplace a signal:

Select the signal name in the PLACED list box or Pin window that you want to unplace, drag the signal name to the UNPLACED list box, then release the mouse button.

Fixing Pins

Pins assigned in a design schematic or a pin file are automatically fixed and are not moved during Layout. Pins placed by Designer are unfixed and should be manually fixed at some point in the design cycle. Unfixed pins may be moved during Layout. Remember, you can move unfixed pins around as the design is iterated. Fixing pins invalidates a completed Layout. Use the following procedure to fix pins using Pinedit.

1. **Invoke PinEdit.** Click the PinEdit button in the Designer Main window.
2. **Select the pin(s) to be fixed.** Click the pin to be fixed in the PLACED list box or Pin window. To select multiple pins, hold the Shift button and click multiple pins. To select all pins, choose the Select All command from the Edit menu.
3. **Fix the pin(s).** Choose the Fix command from the Edit menu.

To unfix a pin:

1. **Select the pin(s) to be unfixed.** Click the pin to be unfixed in the PLACED list box or Pin window. To select multiple pins, hold the Shift button and click multiple pins. To select all pins, choose the Select All command from the Edit menu.
2. **Unfix the pin(s).** Choose the Unfix command from the Edit menu.

Printing Pin Assignments

Designer has two methods of printing pin assignments, a graphical printout of the device and a text list of pinouts. These printouts list the I/O signal locations on the device, sorted by I/O signal names or by package number.

To print a graphical printout of the device:

1. **Invoke PinEdit.** Click the PinEdit button in the Designer Main window.
2. **Choose the Print command from the File menu.**

To print a text list of pinouts:

Choose the Pin command from the Report menu in the Designer Main window.

Committing Pin Assignments

After you have placed and fixed pins, you must commit the changes to your design. You can commit changes by selecting the Commit command from the File menu or by exiting PinEdit and clicking Yes when asked if you would like to commit changed made in PinEdit.

DT Analyze

Use DT Analyze to perform timing analysis. Refer to “Timing Analysis using DT Analyze” on page 95 for information about using DT Analyze.

ChipEdit

Use ChipEdit to view and edit the placement of both I/O and logic macros. Refer to “ChipEdit” on page 117 for information about using ChipEdit.

Layout

After compiling and optionally using other Designer features, the design is ready for layout. Layout takes the netlist information, PinEdit information, and DT Edit information and maps this information into the selected Actel device. Layout assigns physical locations to unassigned I/O and logic modules (placement), routing tracks to nets (routing), and calculates detailed delays for all paths (extract).

Designer supports two modes of layout, Standard and DirectTime. The physical result of each approach is similar, but the tools and algorithms are quite different. In either mode, the incremental placement option allows you to save the performance of a successfully placed and routed design even if you change the netlist.

Standard Layout

Standard layout maximizes the average performance for all paths. Standard layout treats each part of a design equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results.

Standard layout does not consider delay constraints that have been set for a design during place and route. However, can a delay report based on delay constraints entered in DT Edit can still be generated for the design. This is helpful to determine if DirectTime Layout is required.

DirectTime Layout

The primary goal of DirectTime layout is to meet delay constraints set in DT Edit or in a DCF file. The secondary goal is to produce high performance for the rest of the design. Delay constraint driven design is more precise and typically results in higher performance.

Incremental Placement

In either Standard or DirectTime mode, the Incremental Placement option allows you to preserve the timing of a successfully placed and routed design, even if you change part of the netlist. Incremental placement has no effect the first time you run layout. During design iteration, incremental placement preserves the placement information for any unchanged macros in a modified netlist. As a result, the timing relationships for unchanged macros will approximate their initial values, decreasing the execution time to perform Layout.

By forcing Designer to retain the placement information for a portion of the design, some flexibility for optimal design layout may be lost. Therefore, do not use incremental placement to place your design in pieces. You should only use it if you have successfully run Layout and you have minor changes to your design. Incremental placement requires prior completion of Layout. Do not use incremental placement if the previous Layout failed to meet performance goals.

Incremental placement has two levels, FIX and ON. The FIX setting treats all unchanged macros as fixed placements. This is the strongest level of control, but it may be too restrictive for the new placement to successfully complete. The default ON setting treats unchanged macro locations as placement hints, but alters their locations as needed to successfully complete placement.

To layout your design.

1. **Click the Layout button.** The Layout dialog box is displayed.

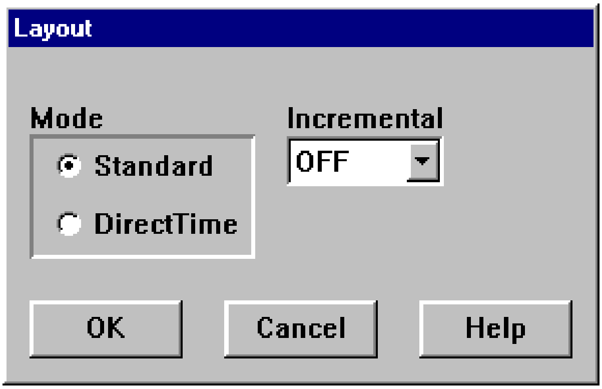


Figure 6-10. Layout Dialog Box

2. **Select Mode.** Select Standard or DirectTime layout Mode.
3. **Select Incremental Placement option.** Specify OFF, ON or FIX in the Incremental pull-down menu.

Layout Failures

If Layout fails at any stage, Designer provides information that can help you determine and correct the problem. This section describes some failures and methods to fix the failures.

Failures During Initialization (DirectTime only)

Layout can fail during initialization if the assigned constraints are impossible (i.e. no routing path on the device can meet the assigned constraint). You must change the circuit or relax the constraints to proceed with DirectTime Layout. Ways to do this include:

- Change the Speed Grade to increase minimum delay.
- Modify the design to reduce the number of logic levels in these paths.
- Relax over-conservative delay constraints. If the constraints from DT Edit or a DCF file are unnecessarily tight, change them to more realistic values that still satisfy your timing requirements.

Failures During Constructive Placement

A constructive placement error is very rare. If you get this type of error, check the following cases and make modifications as necessary:

- ACT 1 and 40MX designs (especially A1010 devices) with many two- and three-logic module HARD macros are difficult to place and route. Replace them with logically equivalent implementations along with the “preserve” property. See “ALSPRESERVE” on page 32 for more information.
- ACT 2 or ACT 3 designs that utilize more than 95% of the sequential modules and use many two- and three-logic HARD macros are difficult to place and route. You can usually use 100% of the modules, but excessive use of registers may make placement difficult.
- Designs with many manually-placed logic macros are difficult to place and route. Unplace the pre-placed macros before running Layout.

- Designs with high fanout networks may be difficult to route. Consider using Silicon Expert to add buffers to high fanout networks in your netlist. This maximizes the efficiency of your routing resources.

Failures During Placement Optimization

If you run into an error during DirectTime Layout, it may be due to one or more of the following:

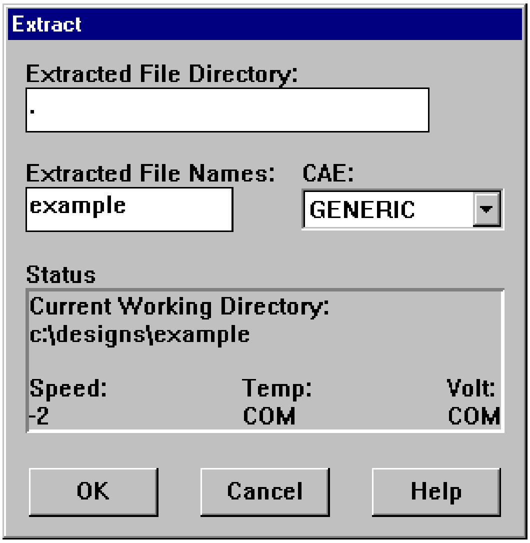
- Fixed pin placements may not allow Layout to succeed. Unplace or unfix I/O pins.
- Designs with very high I/O utilization (above 90%) and very low logic utilization (below 60%) put excessive demand on the long vertical or horizontal routing resources of the device. Try using a smaller device. Often the same design utilizing 50% of an A1280 will successfully place and route using 100% of an A1240.
- If Layout completes but some delay constraints are not satisfied, verify that all of your constraints are necessary and sufficient. If there are constraints for “don’t care” or “false logic” paths, re-enter these constraints with appropriate exclusions, exceptions, and stop sets. Then re-run Layout in non-incremental mode.

Extracting Timing Information

Extract allows you to extract (back annotate) post-layout timing data to your CAE simulator for timing simulation. You can derate the back annotation for different operating conditions and device speed grades.

To create a back annotation file:

1. **Extract back annotation file.** Click the Extract button in the Designer Main window. The Extract dialog box is displayed.



Extract

Extracted File Directory: .

Extracted File Names: example

CAE: GENERIC

Status

Current Working Directory:
c:\designs\example

Speed: -2
Temp: COM
Volt: COM

Figure 6-11. Extract Dialog Box

2. **Save the file.** Set the directory, file name, and CAE type. Click OK.

The Extract program creates the files necessary for back annotation to the CAE file output type that you choose in the dialog box.

Refer to Actel Interface Guides or the documentation included with your simulation tool for information about selecting the correct CAE output format and using the back annotation files.

Fuse

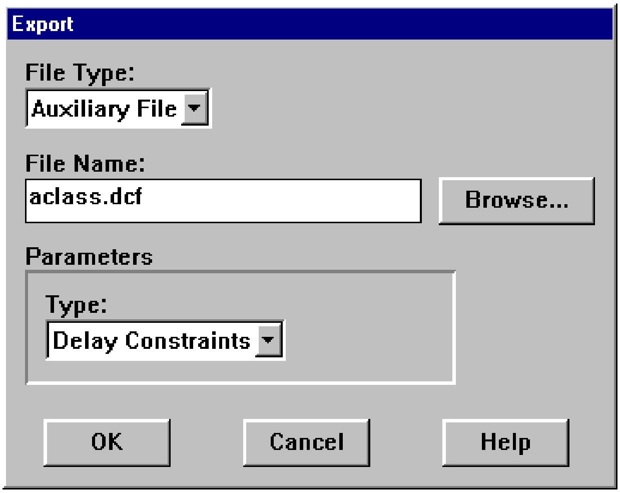
Fuse allows you to generate a programming file. You do not need to run Fuse if you are using APSW, unless you need to set a silicon signature. Refer to “Generating a Programming File” on page 113 for information about using Fuse to set a silicon signature and to generate a programming file.

Exporting Files

Designer lets you export auxiliary files (.afl, .cob, .crt, .dcf, .loc, .pin, .prb, seg) fuse files (.afm, .dio, .fus), log files (.log) netlist files (.adl, .edn, structural VHDL, and structural Verilog), script files (.dsf), and timing files (.stf and .sdf) from your design.

To export a file:

1. **Select the Export command from the File menu in the Designer Main window.** The Export dialog box is displayed.



Export

File Type: Auxiliary File

File Name: aclass.dcf

Parameters

Type: Delay Constraints

Figure 6-12. Export Dialog Box

2. **Select file type to export.** Specify the type of file you want to export from the File Type pull-down menu.
3. **Specify file name.**
4. **Specify parameters.** Select the parameters (if necessary) in the Parameters section of the dialog box.
5. **Click OK.**

Generating Reports

Designer has the capability to generate several types of reports that give designers in depth information about a design. All of the reports are available from the Report menu in the Designer Main window. This section describes the reports that can be generated.

Pin Report

The pin report allows you to create a text list of the I/O signal locations on a device. You can generate a pin report sorted by I/O signal names or by package number.

To generate a pin report:

1. **Choose the Pin command from the Report menu in the Designer Main window.** The Pin Report dialog box is displayed.
2. **Specify the type of report to generate.** Select Number or Name from the List By pull-down menu, then click OK.

Status Report

The status report allows you to create a report containing device and design information, such as die, package, percentage of the logic and I/O modules used, etc.

To generate a status report:

Choose the Status command from the Report menu in the Designer Main window.

Flip Flop Report

The flip flop report allows you to create a report that lists the number and type of flip-flops (sequential or CC, which are flip-flops made of 2 combinatorial macros) used in a design. There are two types of reports that can be generated, Summary or Extended.

A Summary report displays whether the flip-flop is a sequential, I/O sequential, or CC flip-flop, the macro implementation of the flip-flop, and the number of times the implementation of the flip-flop is used in the design.

An Extended report displays whether the flip-flop is a sequential, I/O sequential, or CC flip-flop, the macro implementation of the flip-flop, and individually lists the names of the macros in the design.

To generate a flip flop report:

1. **Select the Flip flops command from the Reports menu.** The Flip flop Report dialog box appears.
2. **Specify the type of report to generate.** Select Summary or Extended from the Type pull-down menu, then click OK.

Timing Report

The timing report allows you to quickly determine if any timing problems exist in your design. The timing report lists the following information about your design:

- maximum delay from input I/O to output I/O
- maximum delay from input I/O to internal registers
- maximum delay from internal registers to output I/O
- maximum delays for each clock network
- maximum delays for interactions between clock networks

To generate a timing report:

1. **Select the Timing command from the Report menu in the Designer Main window.** The Timing Report dialog box is displayed.

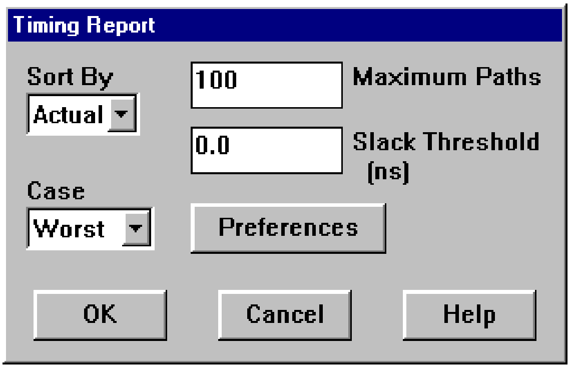


Figure 6-13. Timing Report Dialog Box

2. **Specify the sort method in the Sort By pull-down menu.** Specify “Actual” to list the actual delays between starting and ending points. For example, if a signal takes 20ns to get from point A to point B, the timing report lists 20ns for that path.

   Specify “Slack” to list the difference between the actual delay and a timing constraint previously specified in a DCF file or in DT Edit. For example, if you specified a timing constraint of 15ns, the timing report lists 5ns for that path.

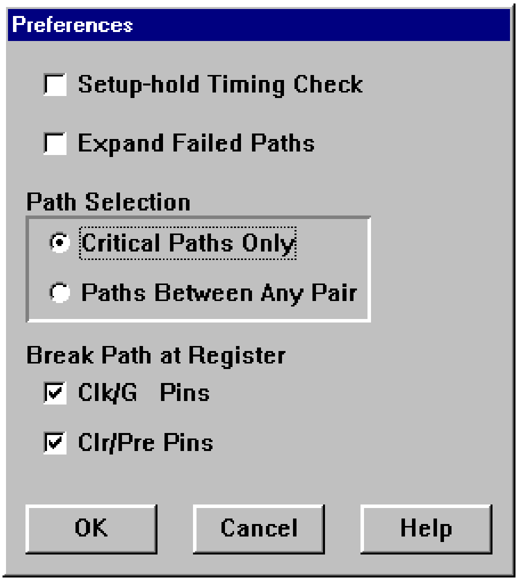
   For either mode, if you enter a constraint, and the actual delay exceeds this constraint, the path is automatically expanded.

3. **Specify the Slack Threshold.** If you select “Slack” as the sort method, you can limit the number of delays displayed based upon a slack threshold. For example, if you only want to see delays that have a slack less than 5ns, enter 5 in the Slack Threshold box.

4. **Specify the number of paths to display in the Maximum Paths box.**

5. **Select the report mode.** Specify Worst, Typical, or Best from the Case pull-down menu.

6. **(Optional) Set preferences.** Click the Preferences button The Preferences dialog box is displayed. This dialog box allows you to configure the timing report to calculate external setup and hold information for device inputs in addition to the standard information. Click OK when finished.



Preferences

☐ Setup-hold Timing Check

☐ Expand Failed Paths

Path Selection

● Critical Paths Only

○ Paths Between Any Pair

Break Path at Register

☑ Clk/G Pins

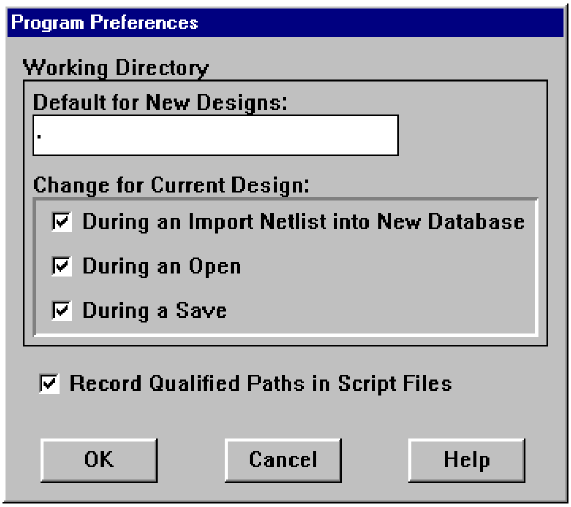
☑ Clr/Pre Pins

Figure 6-14. Timing Report Preferences Dialog Box

7. **Click OK in the Timing Report dialog box.** A timing report is generated. All delays are sorted from longest delay (or worst slack) to shortest delay.

Setting Designer Preferences

You can set the default working directory for Designer. Whenever you execute a command or function such as Open or Import, Designer uses the directory you specified as the default directory. Choose the Preferences command from the File menu to open the Program Preferences dialog box.


Program Preferences

Working Directory

Default for New Designs: .

Change for Current Design:

☑ During an Import Netlist into New Database

☑ During an Open

☑ During a Save

☑ Record Qualified Paths in Script Files

Figure 6-15. Program Preferences Dialog Box

Terminating the Designer Session

To end a Designer session, choose the Exit command from the File menu. If the information has not been saved to disk, you are asked if your want to save the design before exiting. If you choose YES, the <design_name>.adb file is updated with information entered the current session. If you choose NO, the information is not saved and the <design_name>.adb file remains unchanged.

7

Timing Analysis using DT Analyze

This chapter contains information, procedures, and examples for using DT Analyze to perform static timing analysis on an Actel design.

DT Analyze

DT Analyze lets you analyze static timing delays for almost any path or group of paths in your design. This interactive tool is designed for precise static timing analysis. For dynamic timing analysis of your design, use the Extract function to back annotate post-layout delays to your CAE simulator.

DT Analyze works with either pre-layout estimated delays or post-layout calculated delays. Used in conjunction with DT Edit and DT Layout, it displays delay constraint goals versus actual results for any specific path or clock constraint.

Once you initiate the DT Analyze, enter the desired information in the Filter Box and click OK. A spreadsheet list defining path and delay information is presented. Post-layout delays are assumed if the Layout function has been completed (Layout button displayed green).

Configure the spreadsheet using the Filter commands from the Options menu. You can configure the spreadsheet in terms of pins or nets, longest or shortest paths, sorting by actual delay or slack, and filtering on names (starting or ending points) to move to the points of interest.

Expanding Paths

To expand a path, select the path, then select the List or Chart command from the Expand menu. The Chart and List are connected, and selecting individual points on either will highlight both.

List

The List command presents a second spreadsheet defining all delay components for the selected path.

Chart

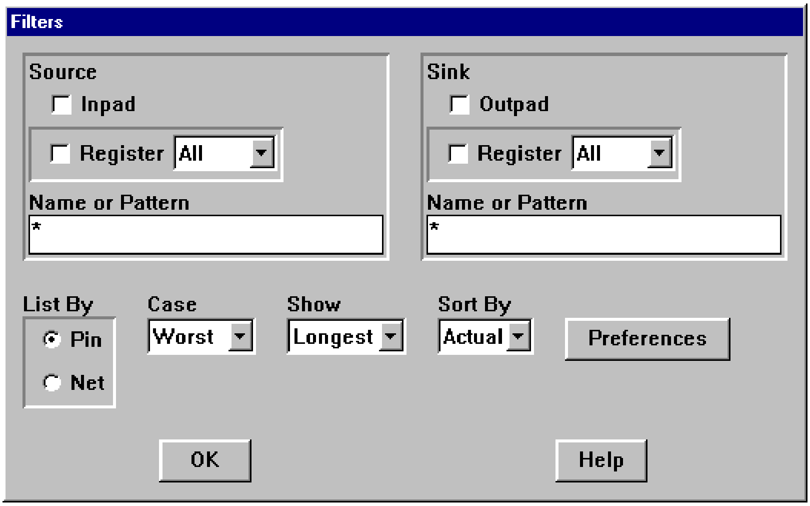
The Chart command graphically displays the entire network for the defined end point along with an expanded spreadsheet containing all paths. Chart is a useful debug tool for determining the magnitude of a particular timing problem and options for correcting the problem.

Using DT Analyze

DT Analyze provides options and commands to sort and display static timing data according to your specific needs. With the menu commands, you can select case display preferences, and you can sort which paths to analyze. Clicking the Preferences button allows you to select the amount of detail to display for each path.

To set up and use DT Analyze for static timing analysis:

1. **Invoke DT Analyze.** Click the DT Analyze button in the Designer Main window. the DT Analyze Filters dialog box is displayed.



Filters

Source
☐ Inpad
☐ Register All
Name or Pattern
*

Sink
☐ Outpad
☐ Register All
Name or Pattern
*

List By
● Pin
○ Net

Case Worst

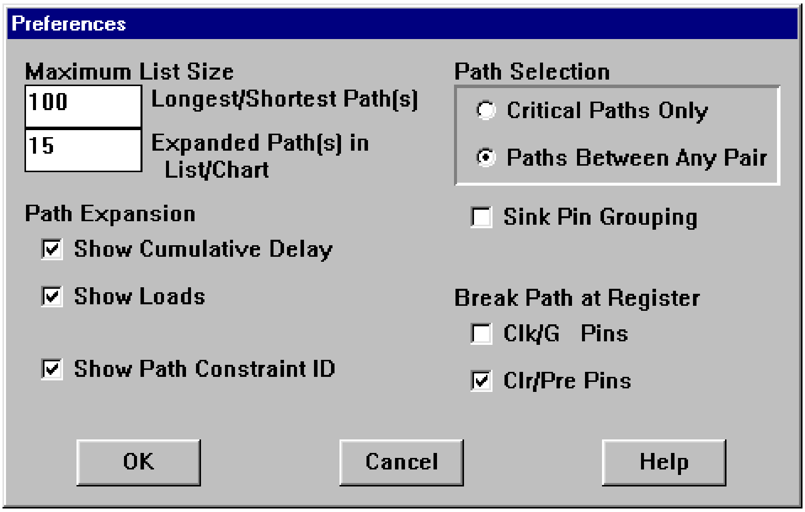
Show Longest

Sort By Actual

Figure 7-1. DT Analyze Filters Dialog Box

2. **Choose which paths to analyze.** In the Filters dialog box you can enter specific names or patterns for the start and end terminals, or you can use the default filter types: inpad (all of the input pad pins), output (all of the output pad pins), or register (all input pins on the flip-flops and latches). You must select a Source and Sink or you will generate a blank report in the DT Analyze window.

3. **(Optional) Set analysis preferences.** Click the Preferences button in the Filters dialog box to access the DT Analyze Preferences dialog box. This dialog box allows you to set the number of paths to display and choose whether or not to show cumulative delay and output loading for each path. Click OK when finished.



Preferences

Maximum List Size
100 Longest/Shortest Path(s)
15 Expanded Path(s) in List/Chart

Path Expansion
☑ Show Cumulative Delay
☑ Show Loads
☑ Show Path Constraint ID

Path Selection
○ Critical Paths Only
● Paths Between Any Pair

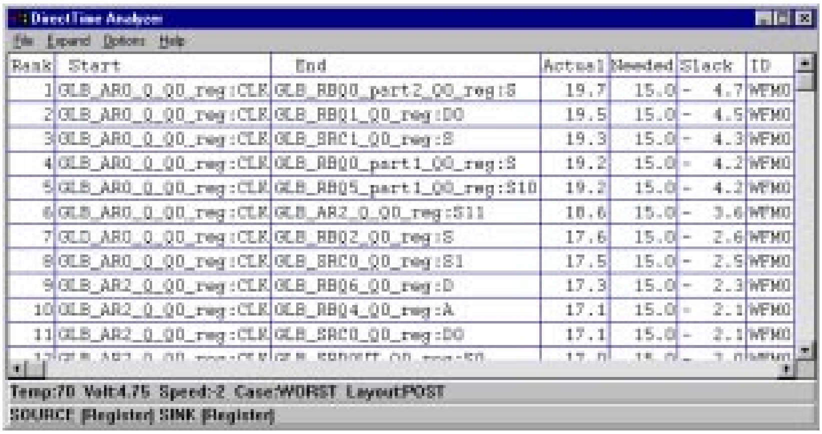
☐ Sink Pin Grouping

Break Path at Register
☐ Clk/G Pins
☑ Clr/Pre Pins

Figure 7-2. DT Analyze Preferences

Break Path at Register is used to prevent paths, that pass through either a clock, ground, clear, or preset pin, from being displayed in DT Analyze.

4. **View DT Analyze results.** After you set the filter options, click OK to display the DT Analyze window. Figure 7-3 shows the report you would generate if you had specified timing constraints previously. If you had not set timing constraints, the “Needed,” “Slack,” and “ID” columns would not appear on the report.



Rank	Start	End	Actual	Needed	Slack	ID
1	GLB_AR0_0_Q0_reg:CLK	GLB_RB00_part2_Q0_reg:S	19.7	15.0	- 4.7	WFM0
2	GLB_AR0_0_Q0_reg:CLK	GLB_RB01_Q0_reg:D0	19.5	15.0	- 4.5	WFM0
3	GLB_AR0_0_Q0_reg:CLK	GLB_SRC1_Q0_reg:S	19.3	15.0	- 4.3	WFM0
4	GLB_AR0_0_Q0_reg:CLK	GLB_RB00_part1_Q0_reg:S	19.2	15.0	- 4.2	WFM0
5	GLB_AR0_0_Q0_reg:CLK	GLB_RB05_part1_Q0_reg:S10	19.2	15.0	- 4.2	WFM0
6	GLB_AR0_0_Q0_reg:CLK	GLB_AR2_0_Q0_reg:S11	18.6	15.0	- 3.6	WFM0
7	GLD_AR0_0_Q0_reg:CLK	GLB_RB02_Q0_reg:S	17.6	15.0	- 2.6	WFM0
8	GLB_AR0_0_Q0_reg:CLK	GLB_SRC0_Q0_reg:S1	17.5	15.0	- 2.5	WFM0
9	GLB_AR2_0_Q0_reg:CLK	GLB_RB06_Q0_reg:D	17.3	15.0	- 2.3	WFM0
10	GLB_AR2_0_Q0_reg:CLK	GLB_RB04_Q0_reg:A	17.1	15.0	- 2.1	WFM0
11	GLB_AR2_0_Q0_reg:CLK	GLB_SRC0_Q0_reg:D0	17.1	15.0	- 2.1	WFM0

Temp:70 Volt:4.75 Speed:-2 Case:WORST Layout:POST

SOURCE [Register] SINK [Register]

Figure 7-3. DT Analyze Window

The DT Analyze window shows the following information for each path:

- **Rank.** The position of the path relative to others in terms of delay. The ranking order depends on the “Slack” or “Actual” selection in the “sort by” option in the Filters dialog box.

  Actual: ranking based on actual-delay with longest/shortest.

  - Longest: use max

  - Shortest: use min

  - Slack: ranking based on slack.

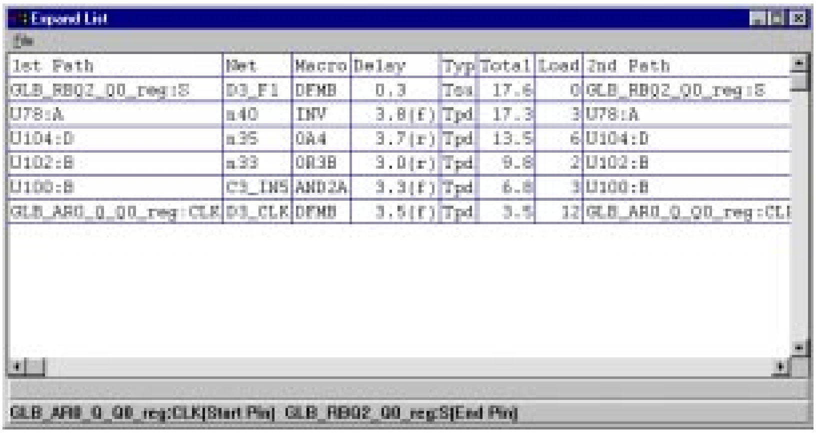
- **Start terminal (Start).** The first pin or net of the path.
- **End terminal (End).** The last pin or net of the path.
- **Actual delay (Actual).** The actual delay calculated by DT Analyze.

- **Required delay (Needed).** The delay constraint or goal for the path from DT Edit or DCF file. This information is displayed only if you previously have set timing constraints.
- **Difference between required delay and actual delay (slack).** If this number is negative, then the actual delay was longer than the required delay. If this number is positive, then the actual delay was less than the required delay (thus meeting the timing requirement).
- **Constraint name (ID).** The name of the constraint for this path, defined in DT Edit or DCF file.

5. **Expand path details.** Each path has one or more logic macros that contribute to its total delay. To view the expanded path details follow these steps:

A. Highlight the path in the DT Analyze window by clicking anywhere in the path's row. Follow either step B or step C depending on whether you want to view path details only (step B) or path details *and* a graphical representation of the path (step C).

B. Select the List command from the Expand menu to view the expanded path's details, as shown in Figure 7-4.



1st Path	Net	Macro	Delay	Typ	Total	Load	2nd Path
GLB_RBQ2_Q0_reg:S	D3_F1	DFMB	0.3	Tsu	17.6	0	GLB_RBQ2_Q0_reg:S
U78:A	n40	INV	3.8(f)	Tpd	17.3	3	U78:A
U104:D	n35	OA4	3.7(r)	Tpd	13.5	6	U104:D
U102:B	n33	OR3B	3.0(r)	Tpd	9.8	2	U102:B
U100:B	C3_INS	AND2A	3.3(f)	Tpd	6.8	3	U100:B
GLB_AR0_0_Q0_reg:CLK	D3_CLK	DFMB	3.5(f)	Tpd	3.5	12	GLB_AR0_0_Q0_reg:CL

GLB_AR0_0_Q0_reg:CLK[Start Pin] GLB_RBQ2_Q0_reg:S[End Pin]

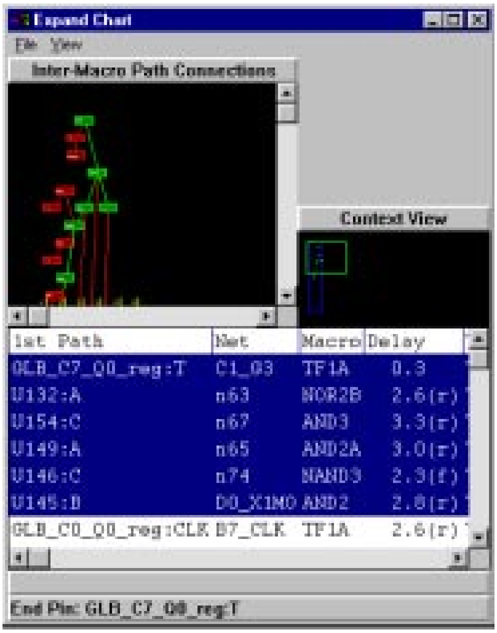
Figure 7-4. Expand List Dialog Box

The Expand List dialog box lists the following information:

- **Pin name (1st Path).** The input pin to the logic macro in the first path.
- **Net name (Net).** The name of the net driven by the logic macro output.
- **Macro type (Macro).** The type of logic macro from the Actel macro library.
- **Actual delay (Delay).** The actual delay calculated by DT Analyze. This column also indicates low-to-high (r) and high-to-low (f) logic transitions. A delay of 0.0ns indicates that the logic combining function in Compile has combined this logical function into the next macro in the path. Logic combination increases the speed and the density of the circuit.
- **Delay type (Typ).** The delay types are propagation delay (Tpd), setup time (Tsu), hold time (Thd), and arrival time (Arr).
- **Cumulative delay (Total).** The cumulative delay for the path up to that logic macro.
- **Output load (Load).** The number of loads that are driven by the logic macro output.

For each path (rising and falling) between the starting and ending points, these columns of information are repeated for the 2nd path, 3rd path, 4th path, etc.

C. Select the Chart command from the Expand menu to view the path details (as described in step B above) and a graphical representation of the path, shown in Figure 7-5.



1st Path	Net	Macro	Delay
GLB_C7_Q0_reg:T	C1_Q3	TF1A	0.3
U132:A	n63	NOR2B	2.6(r)
U154:C	n67	AND3	3.3(r)
U149:A	n65	AND2A	3.0(r)
U146:C	n74	NAND3	2.3(f)
U145:B	D0_X1M0	AND2	2.8(r)
GLB_C0_Q0_reg:CLK	B7_CLK	TF1A	2.6(r)

End Pin: GLB_C7_Q0_reg:T

Figure 7-5. Expanded Chart Dialog Box

DT Analyze Examples

This section provides examples of how to use the DT Analyze to determine timing data for your designs. These examples show how to calculate the following data:

- Maximum register-to-register delay
- Maximum operating frequency
- Clock-to-output delay
- Input-to-output delay
- On-chip data delay

These examples use the sample circuit shown in Figure 7-6.

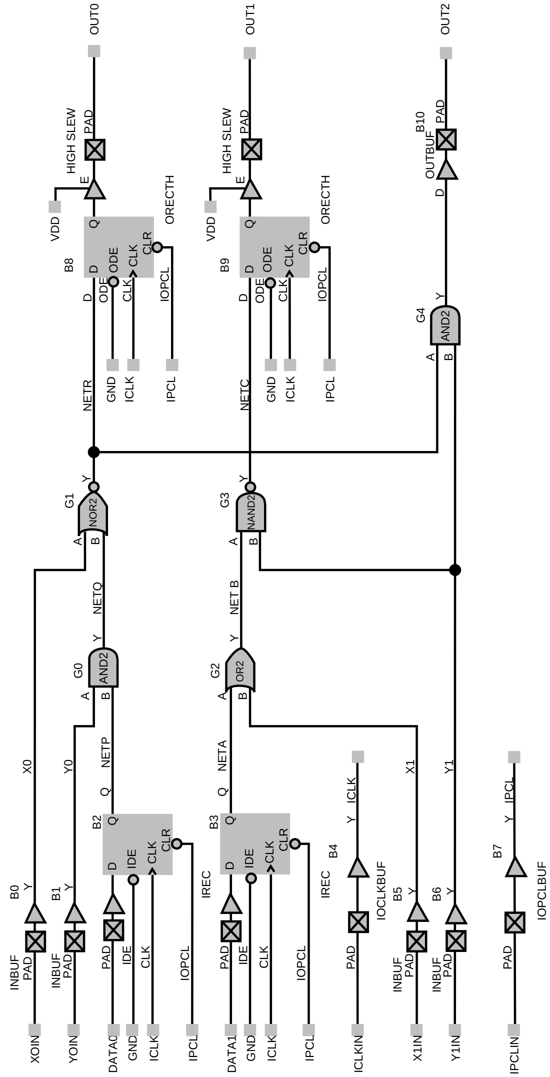
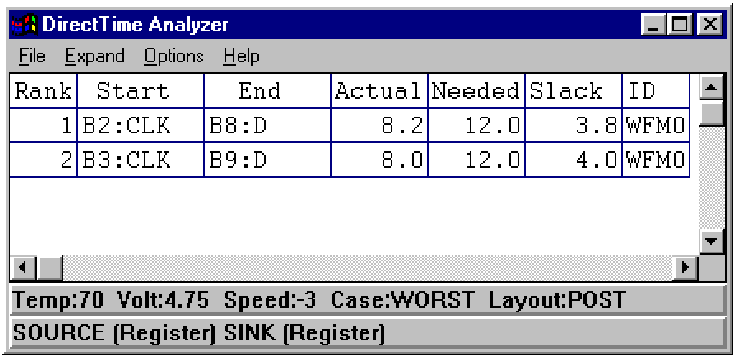

OUT0
OUT1
OUT2
HIGH SLEW
PAD
E
VDD
B8
B9
B10
OUTBUF
D
Q
ODE
CLK
CLR
ORECTH
IOPCL
GND
ICLK
IPCL
NETR
NETC
G4
AND2
A
B
Y
G1
NOR2
G3
NAND2
NETQ
NET B
G0
G2
OR2
NETP
NETA
X0
Y0
X1
Y1
B2
B3
IDE
IREC
B4
IOCLKBUF
B5
B6
B7
IOPCLBUF
INBUF
B0
B1
XOIN
YOIN
DATA0
DATA1
ICLKIN
X1IN
Y1IN
IPCLIN

Figure 7-6. Sample Circuit

Example 1—Calculating Maximum Register-to-Register Delay

To determine maximum register-to-register delay:

1. **Invoke DT Analyze.** Click the DT Analyze button in the Designer Main window. the DT Analyze Filters dialog box is displayed (see Figure 7-1).
2. **Click the Register boxes under both Source and Sink.**
3. **Specify the Source and Sink clock nets.** Use the pull-down menus next to the Register boxes to choose the active clock nets. Choose “All” for both to find delays for all register-to-register paths.
4. **Select display order.** Select Longest or Shortest in the Show pull-down menu and Actual in the Sort By pull-down menu to specify whether the register delays are displayed in order from longest to shortest or shortest to longest. Otherwise, it depends on Slack.
5. **(Optional) Set analysis preferences.** Click the Preferences button in the Filters dialog box to access the DT Analyze Preferences dialog box. This dialog box allows you to set the number of paths to display and choose whether or not to show cumulative delay and output loading for each path. Click OK when finished.
6. **Click OK.** DT Analyze displays the results, as shown in Figure 7-7.



Rank	Start	End	Actual	Needed	Slack	ID
1	B2:CLK	B8:D	8.2	12.0	3.8	WFM0
2	B3:CLK	B9:D	8.0	12.0	4.0	WFM0

Temp:70 Volt:4.75 Speed:-3 Case:WORST Layout:POST

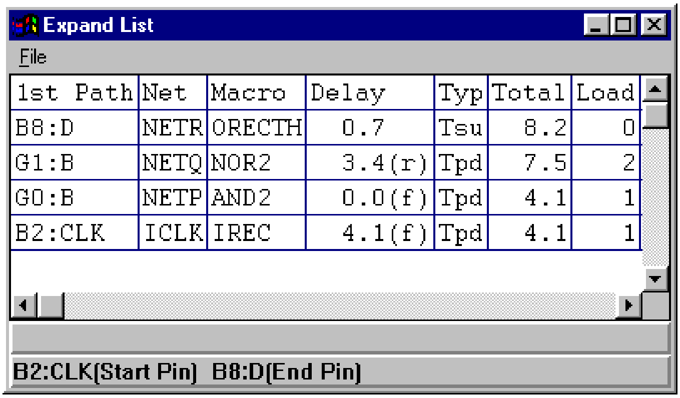
SOURCE (Register) SINK (Register)

Figure 7-7. Maximum Register-to-Register Delay

The result is two paths shown in descending order. The total delay for the first path (shown as Rank 1) is 8.2ns. The starting point is the clock input of B2 (B2:CLK). The end pin (B8:D) is the data input of B8.

You can view more path details for any path in DT Analyze by selecting the List command from the Expand menu, as described in the previous section. The expanded path details show the delay, delay type, and loading for each path component.

Figure 7-8 shows the Expand List dialog box for the longest register-to-register path (Rank = 1) for the sample circuit. The delay is separated into its individual propagation delay and setup time components.



1st Path	Net	Macro	Delay	Typ	Total	Load
B8:D	NETR	ORECTH	0.7	Tsu	8.2	0
G1:B	NETQ	NOR2	3.4(r)	Tpd	7.5	2
G0:B	NETP	AND2	0.0(f)	Tpd	4.1	1
B2:CLK	ICLK	IREC	4.1(f)	Tpd	4.1	1

B2:CLK(Start Pin) B8:D(End Pin)

Figure 7-8. Expanded Maximum Register-to-Register Delay

Example 2—Calculating Maximum Operating Frequency

In a synchronous design, the maximum operating frequency is the inverse of the longest register-to-register delay (1/n). Thus, to calculate maximum operating frequency, calculate maximum register-to-register delay (as described above) and invert. The register-to-register delay is the delay from the clock (gate) of a flip-flop (latch) to the data input of a flip-flop (latch), including all combinatorial gate delays between the registers and the required setup time on the ending register. In this example, the maximum frequency is 1/8.2ns = 122MHz. You must use worst case operating conditions to determine a safe maximum operating frequency.

Example 3—Calculating Clock-to-Output Delay

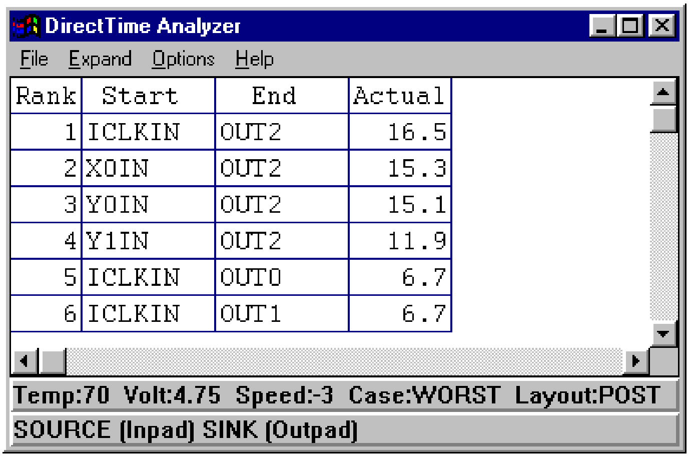
In a synchronous system, the clock-to-output delay is important to determine the setup conditions of the next device on the printed circuit board.

Note: Clock-to-output delay = register clock input to output pad delay.

To determine the clock-to-output delay:

For the sample circuit shown in Figure 7-10, you can determine the clock-to-output delay using the following commands:

1. **Invoke DT Analyze.** Click the DT Analyze button in the Designer Main window. The DT Analyze Filters dialog box is displayed.
2. **Click the Inpad box under Source and the Outpad box under Sink.** This specifies the clock-to-output delays and the combinational input-to-output delays.
3. **Select display order.** Select Longest or Shortest in the Show pull-down menu and Actual in the Sort By pull-down menu to specify whether the register delays are displayed in order from longest to shortest or shortest to longest. Otherwise, it depends on Slack.
4. **(Optional) Set analysis preferences.** Click the Preferences button in the Filters dialog box. The DT Analyze Preferences dialog box lets you set the number of paths to display and choose whether or not to show cumulative delay and output loading for each path. Click OK.
5. **Click OK.** DT Analyzer displays the results, as shown in Figure 7-9.



Rank	Start	End	Actual
1	ICLKIN	OUT2	16.5
2	X0IN	OUT2	15.3
3	Y0IN	OUT2	15.1
4	Y1IN	OUT2	11.9
5	ICLKIN	OUT0	6.7
6	ICLKIN	OUT1	6.7

Temp:70 Volt:4.75 Speed:-3 Case:WORST Layout:POST

SOURCE (Inpad) SINK (Outpad)

Figure 7-9. Clock-to-Output Delay

The longest delay, 16.5ns, is from the ICLKIN input signal to the OUT2 output signal. This path is highlighted in Figure 7-10.

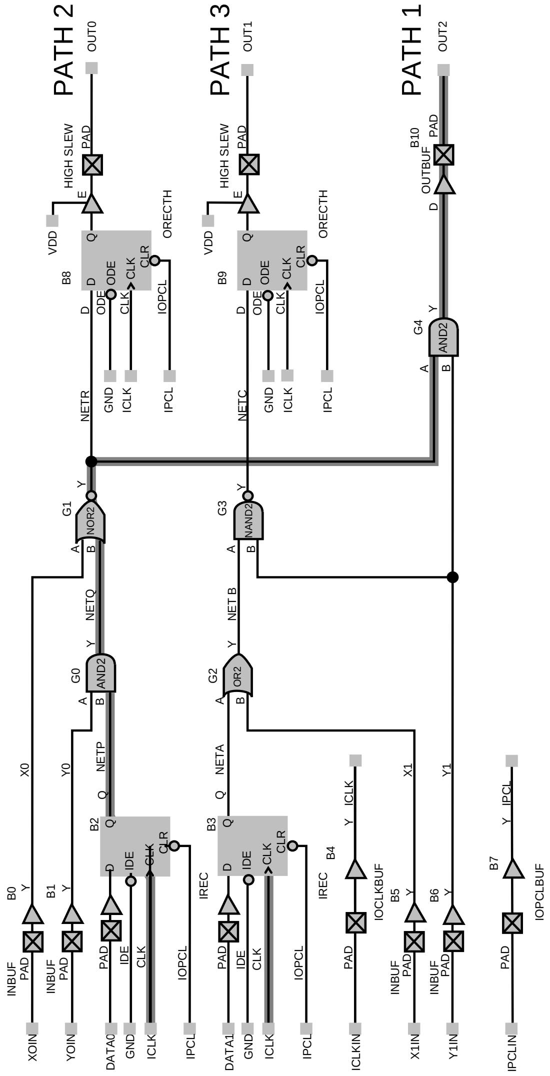

PATH 2
OUT0
PATH 3
OUT1
PATH 1
OUT2
HIGH SLEW
PAD
E
VDD
B8
D
Q
ODE
CLK
CLR
ORECTH
IOPCL
NETR
GND
ICLK
IPCL
B9
NETC
B10
OUTBUF
G4
AND2
A
B
Y
G1
NOR2
G3
NAND2
NETQ
NET B
G0
AND2
G2
OR2
X0
Y0
NETP
NETA
X1
Y1
B0
B1
B2
B3
B4
B5
B6
B7
INBUF
IDE
IREC
IOCLKBUF
IOPCLBUF
XOIN
YOIN
DATA0
DATA1
ICLKIN
X1IN
Y1IN
IPCLIN

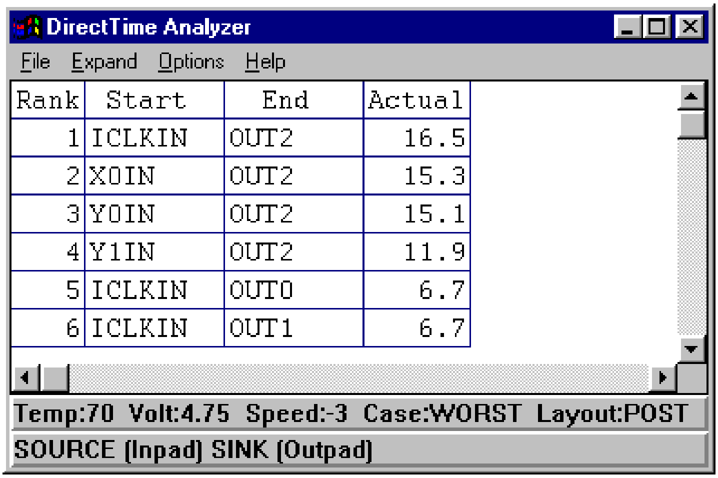
Figure 7-10. Clock-to-Output Delay

Example 4—Calculating Input-to-Output Delay

Input-to-output delay is the delay of combinatorial paths from input pads to output pads. For the sample circuit shown in Figure 7-12, the combinatorial path is from the data inputs (X0IN, X1IN, Y0IN, Y1IN) to the output signal, OUT2.

To determine input-to-output delay:

1. **Invoke DT Analyze.** Click the DT Analyze button in the Designer Main window. The DT Analyze Filters dialog box is displayed (see Figure 7-1).
2. **Click the Inpad box under Source and the Outpad box under Sink.** This displays the clock pad to output delays.
3. **Click the List By Net radio button.**
4. **Select display order.** Select Longest or Shortest in the Show pull-down menu and Actual in the Sort By pull-down menu to specify whether the register delays are displayed in order from longest to shortest or shortest to longest. Otherwise, it depends on Slack.
5. **(Optional) Set analysis preferences.** Click the Preferences button in the Filters dialog box. The DT Analyze Preferences dialog box allows you to set the number of paths to display and choose whether or not to show cumulative delay and output loading for each path. Click OK when finished.
6. **Click OK.** DT Analyze displays the results, as shown in Figure 7-11.



Rank	Start	End	Actual
1	ICLKIN	OUT2	16.5
2	X0IN	OUT2	15.3
3	Y0IN	OUT2	15.1
4	Y1IN	OUT2	11.9
5	ICLKIN	OUT0	6.7
6	ICLKIN	OUT1	6.7

Temp:70 Volt:4.75 Speed:-3 Case:WORST Layout:POST

SOURCE (Inpad) SINK (Outpad)

Figure 7-11. Input-to-Output Delay

The third longest delay, 15.1ns, is from the YOIN input signal to the OUT2 output signal. This path is highlighted in Figure 7-12.

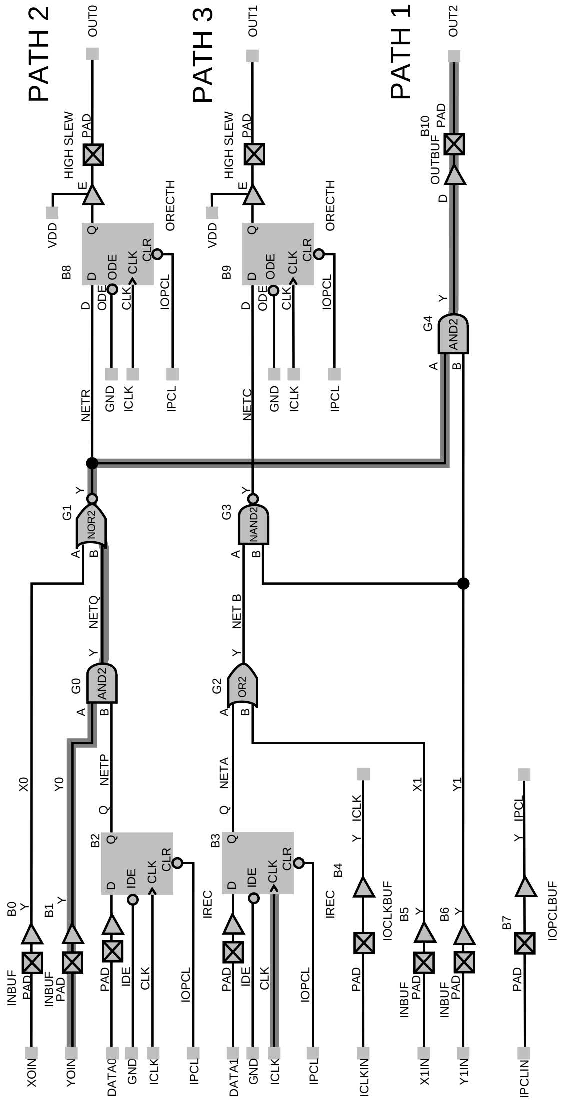

PATH 2
PATH 3
PATH 1
OUT0
OUT1
OUT2
HIGH SLEW
PAD
E
VDD
Q
B8
D
ODE
CLK
CLR
ORECTH
IOPCL
B9
B10
OUTBUF
G4
AND2
A
B
Y
NETR
NETC
GND
ICLK
IPCL
G1
NOR2
G3
NAND2
NETQ
NET B
G0
AND2
G2
OR2
X0
Y0
NETP
NETA
X1
Y1
B0
B1
B2
B3
B4
B5
B6
B7
INBUF
IDE
IREC
IOCLKBUF
IOPCLBUF
XOIN
YOIN
DATA0
DATA1
ICLKIN
X1IN
Y1IN
IPCLIN

Figure 7-12. Input-to-Output Delay

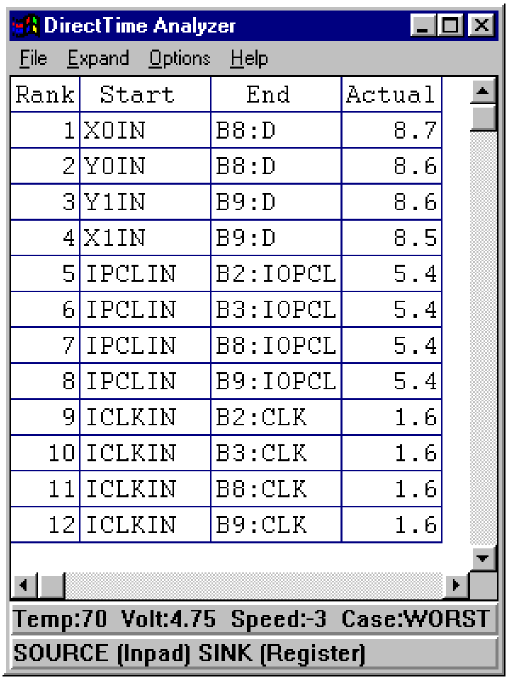
Example 5—Calculating On-Chip Delays

The on-chip delay is the time for signals to arrive on-chip from the input pads. For the sample circuit shown in Figure 7-14, the longest on-chip data paths are from the data inputs (X0IN, Y0IN, X1IN, Y1IN) to the B8 and B9 flip-flops.

To determine On-Chip delays:

1. **Invoke DT Analyze.** Click the DT Analyze button in the Designer Main window. The DT Analyze Filters dialog box is displayed (see Figure 7-1).
2. **Click the Inpad box under Source and Register under Sink.**
3. **Select display order.** Select Longest or Shortest in the Show pull-down menu and Actual in the Sort By pull-down menu to specify whether the register delays are displayed in order from longest to shortest or shortest to longest. Otherwise, it depends on Slack.
4. **(Optional) Set analysis preferences.** Click the Preferences button in the Filters dialog box. the DT Analyze Preferences dialog box. allows you to set the number of paths to display and choose whether or not to show cumulative delay and output loading for each path. Click OK when finished.

5. **Click OK.** DT Analyze displays the results, as shown in Figure 7-13.


DirectTime Analyzer
File Expand Options Help

Rank	Start	End	Actual
1	X0IN	B8:D	8.7
2	Y0IN	B8:D	8.6
3	Y1IN	B9:D	8.6
4	X1IN	B9:D	8.5
5	IPCLIN	B2:IOPCL	5.4
6	IPCLIN	B3:IOPCL	5.4
7	IPCLIN	B8:IOPCL	5.4
8	IPCLIN	B9:IOPCL	5.4
9	ICLKIN	B2:CLK	1.6
10	ICLKIN	B3:CLK	1.6
11	ICLKIN	B8:CLK	1.6
12	ICLKIN	B9:CLK	1.6

Temp:70 Volt:4.75 Speed:-3 Case:WORST

SOURCE (Inpad) SINK (Register)

Figure 7-13. On-Chip Delays

The longest delay, 8.7ns, is from the XOIN input signal to the B8 flip-flop data input.

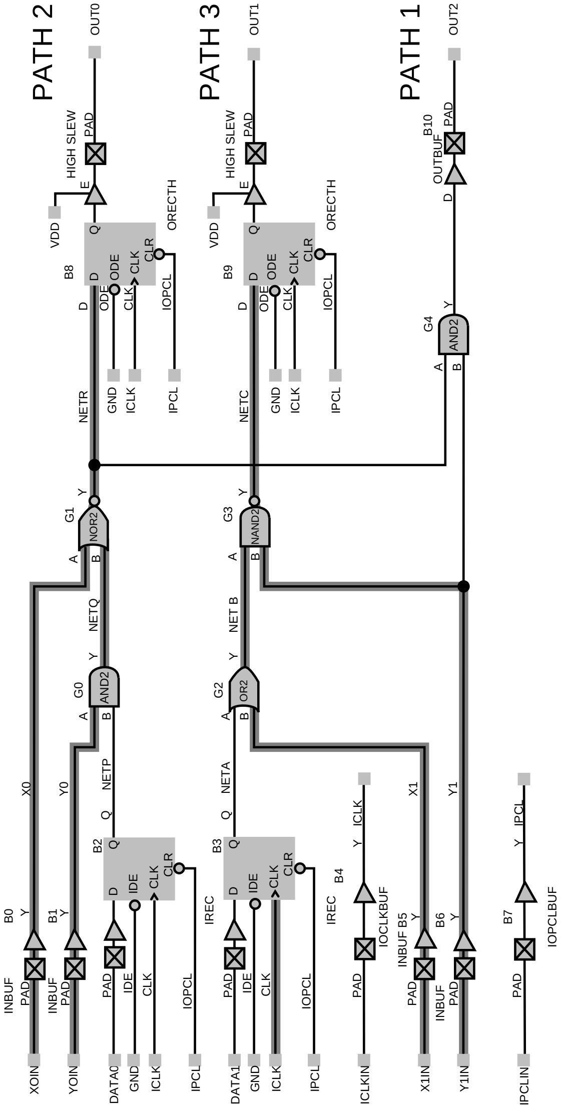

PATH 2
OUT0
HIGH SLEW
PAD
E
VDD
Q
B8
D
ODE
CLK
CLR
ORECTH
IOPCL
NETR
GND
ICLK
IPCL
PATH 3
OUT1
B9
NETC
PATH 1
OUT2
B10
OUTBUF
G4
AND2
A
B
Y
G1
NOR2
G3
NAND2
NETQ
NET B
G0
G2
OR2
X0
Y0
NETP
NETA
X1
Y1
B2
B3
IDE
IREC
B4
IOCLKBUF
INBUF B5
B6
B7
IOPCLBUF
B0
B1
INBUF
XOIN
YOIN
DATA0
DATA1
ICLKIN
X1IN
Y1IN
IPCLIN

Figure 7-14. On-Chip Data Path Delay

Batch Timer

The Batch Timer lets you analyze static timing delays using command files. It is designed to find timing data for any path or group of paths in your design. Since it is run through command files instead of interactive commands, it is well suited for complex analysis of large numbers of paths. For dynamic timing analysis of your design, use the Extract function to back annotate post-layout delays to your CAE simulator. Refer to the Designer on-line help for additional information about the Batch Timer.

8

Generating a Programming File

This chapter describes how to use the Designer Fuse feature to prepare a programming file for your design to program an Actel device. This includes information about setting a silicon signature and preparing programming files for Silicon Sculptor programmers and Data I/O programmers.

Silicon Signature

You can specify a unique silicon signature to program into the device when you generate a programming file. This signature is stored in the design database, the programming file, and programmed into the device permanently during programming. With Designer tools, you can use the silicon signature to identify and track Actel designs and devices. Go to “Generating a Programming File” on page 114 for the procedure.

Generating a Programming File

Designer can generate a programming file for APSW/Activator programmers, Silicon Sculptor programmers, and Data I/O programmers.

APSW/Activator Programmers

The Designer Series includes APSW software to program devices with Activator 2 and 2S programmers. If you use APSW, you do not need to use the Fuse command to create a programming file unless you are specifying a silicon signature. Refer to the *Activator and APS Programming System Installation and User’s Guide* for information about programming a device using APSW software.

Silicon Sculptor Programmers

Silicon Sculptor programmers use AFM files to program Actel devices. Refer to the *Silicon Sculptor User’s Guide* for information about programming a device using Silicon Sculptor programmers.

Data I/O Programmers

Data I/O Software/Algorithm Systems program Actel devices on the Unisite, Pinsite, and 3900 programmers. They require a DIO file to program Actel devices. Refer to the Data I/O documentation for information about programming devices with Data I/O programmers.

Generating a Programming File

Use the following procedure to generate a programming file and to optionally enter a silicon signature.

1. **Invoke Designer.** The Designer Main window is displayed.

   PC

   Choose Designer from the Designer group in the Programs menu under the Start menu.

   UNIX

   Type the following command at the prompt:

   ```
   designer
   ```

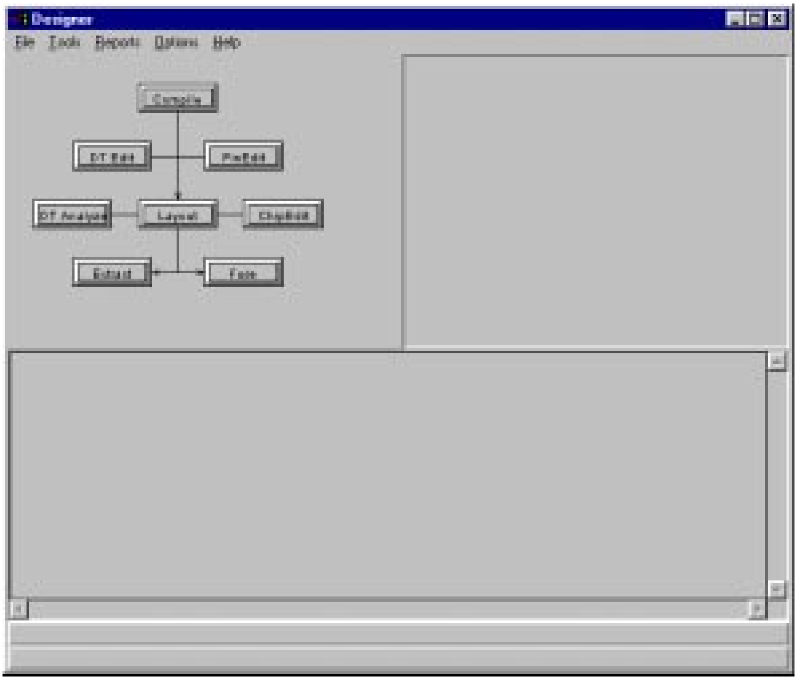
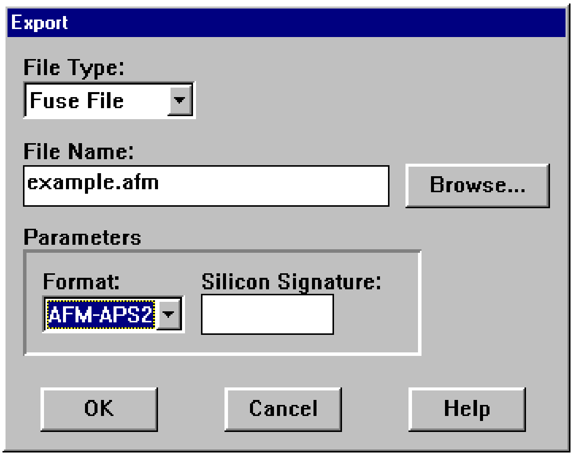

Designer
File
Tools
Reports
Options
Help
Compile
DT Edit
PinEdit
DT Analyze
Layout
ChipEdit
Extract
Fuse

Figure 8-1. Designer Main Window

2. **Open your design.** Choose the Open command from the File menu, select the ADB file to open in the Open dialog box, then click OK.

3. **Open the Export dialog box.** Click the Fuse button in the Designer Main window. The Fuse dialog box is displayed.



Export

File Type: Fuse File

File Name: example.afm

Parameters

Format: AFM-APS2

Silicon Signature:

Figure 8-2. Fuse Dialog Box

4. **Specify Fuse in the File Type pull-down menu.**
5. **(Optional) Name the file.** Designer automatically names the file based on the <design_name>.adb file. You can change the name by entering it in the File Name box.

   Note: Do not add a file extension or suffix to the file name. Designer automatically adds the extension to the programming file name when you specify the programming format.

6. **Specify programming format.** Select the appropriate file type in the Format pull-down menu. Select “AFM-APS2” if you are using APSW/Activator programmers or Silicon Sculptor programmers. Select “DATA I/O” if you are using Data I/O programmers.
7. **(Optional) Enter a silicon signature.** Enter a 5-digit hexadecimal value in the Silicon Signature box to identify the design. Valid characters are “0” through “9,” and “a” through “f.” You must create a programming file to set the silicon signature if you are using APSW to program.
8. **Save the programming file.** Click OK when finished.

A

ChipEdit

This appendix contains information and procedures about using ChipEdit to view and manually place I/O and logic macros in a design.

ChipEdit Window

ChipEdit allows you to view and position I/O and logic modules on a device using list boxes and a graphical representation of the device. ChipEdit contains two windows and two list boxes. The ChipEdit window is shown in Figure A-1.

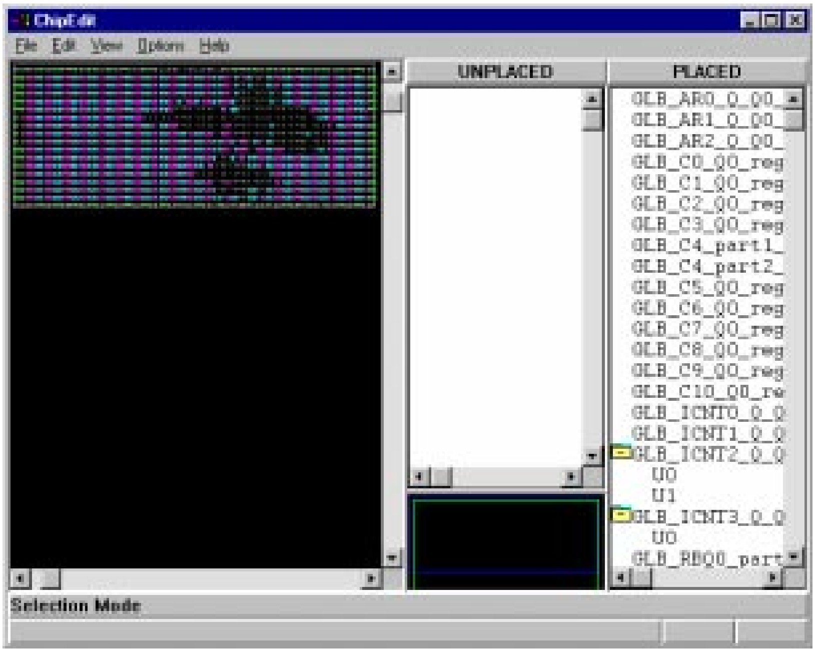

ChipEdit
File Edit View Options Help
UNPLACED
PLACED
GLB_AR0_0_Q0_
GLB_AR1_0_Q0_
GLB_AR2_0_Q0_
GLB_C0_Q0_reg
GLB_C1_Q0_reg
GLB_C2_Q0_reg
GLB_C3_Q0_reg
GLB_C4_part1_
GLB_C4_part2_
GLB_C5_Q0_reg
GLB_C6_Q0_reg
GLB_C7_Q0_reg
GLB_C8_Q0_reg
GLB_C9_Q0_reg
GLB_C10_Q0_re
GLB_ICNT0_0_Q
GLB_ICNT1_0_Q
GLB_ICNT2_0_Q
U0
U1
GLB_ICNT3_0_Q
U0
GLB_RBQ0_part
Selection Mode

Figure A-1. ChipEdit Window

The Chip window on the left displays a graphic representation of the logic modules on the device. When you select an assigned macro in the Chip window, the selected macro is highlighted in the PLACED list box. The small window under the UNPLACED list box is a position window that allows you to change the area of the device that is displayed in the Chip window.

The PLACED and UNPLACED list boxes display a list of placed and unplaced macros in the design. Both fixed and unfixed macros are displayed in the PLACED list box.

Use the Configure List Boxes dialog box under the Options menu to specify how the pin information displayed in PinEdit.

View Menu

The following commands allow you to change what is displayed in the Chip window.

Fit

The Fit command displays the entire design in the Chip window.

Zoom Area

The Zoom Area command allows you to select what area of the design is displayed in the Chip window.

Zoom In

The Zoom In command enlarges what is displayed in the Chip window by 200%. The Zoom In command can be chosen multiple times to zoom in as much as necessary.

Zoom Out

The Zoom Out command reduces what is displayed in the Chip window by 200%. The Zoom Out command can be chosen multiple times to zoom out as much as necessary.

Redraw

The Redraw command redraws what is displayed in the Chip window at the current zoom percentage.

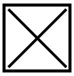
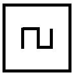
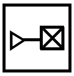
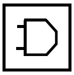
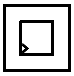
Placed List Box

The Place List Box command toggles the PLACED list box to be displayed or not displayed.

Chip Window Colors and Symbols

The Chip window displays a graphical representation of a design using colors and symbols. Table A-1 describes the colors and symbols used to represent a design in the Chip window.

Table A-1. Chip Window Colors and Symbols

Color/Symbol[a]	**Definition**
White Border	A white borders denotes a selected object.
Black Background	A black background denotes an unused or unplaced module.
Yellow	Yellow denotes fixed logic modules. If the module is selected, the symbol appears yellow. If the module is unselected, the border appears yellow.
Green	Green denotes I/O modules.
Red	Red denotes clock modules.
	Reserved modules that are not user-definable are gray cross-out symbols on a black background.
	Clock modules are red. Unused/unplaced modules are red symbols on a black background. Used/placed modules are black symbols on a red background.
	Input/Output modules are green. Unused/unplaced modules are green symbols on a black background. Used/placed modules are black symbols on a green background.
	Combinatorial modules are blue. Unused/unplaced modules are blue symbols on a black background. Used/placed modules are black symbols on a blue background.
	Sequential modules are magenta. Unused/unplaced modules are magenta symbols on a black background. Used/placed modules are magenta symbols on a black background.

[a] Macros that use more than one module appear as one with a gray background and black symbols.

Working with Multiple ChipEdit Windows

Working with multiple ChipEdit windows is useful if your design is large and you want to move macros to different locations. You can zoom the windows and place, unplace, and move macros between windows. To place, unplace, and move macros using multiple ChipEdit windows, invoke multiple ChipEdit windows before continuing. Macros can be moved between ChipEdit windows.

Placing Macros

Use the following procedure to place macros in your design using ChipEdit.

1. **Invoke ChipEdit.** Click the ChipEdit button in the Designer Main window.
2. **Select the macro(s) to be placed.** Click the macro to be placed in the UNPLACED list box. To select multiple macros, hold the Shift button and click multiple macro names. To select all unplaced macros, choose the Select All Unplaced command from the Edit menu.
3. **Place the macro(s).** If you have selected a single macro to place, drag the macro to the logic module in the Chip window you want to place it in, then release the mouse button.

   If you have selected multiple macros, choose the Place command from the Edit menu. ChipEdit then prompts you to place the macros one at a time.

To unplace placed macros:

4. **Select the macro(s) to be unplaced.** Click the macro to be unplaced in the PLACED list box or the Chip window. To select multiple macros, hold the shift button and click multiple macros. To select all placed macros, choose the Select All Placed command from the Edit menu.
5. **Unplace the macro(s).** If you have selected a single macro to unplace, drag the macro to the UNPLACED list box, then release the mouse button. If you have selected multiple macros, choose the Unplace command from the Edit menu.

Moving Macros

Macros can be moved after they have been placed, either during Layout, or using ChipEdit. Use the following procedure to move macros using ChipEdit.

1. **Invoke ChipEdit.** Click the ChipEdit button in the Designer Main window.
2. **Move the macro.** Click the macro to be moved in the PLACED list box or the Chip window and drag it to the logic module you want to move it to.

Fixing Macros

Macros are not fixed in Designer unless you use ChipEdit to fix them. If you have placed or moved macros in ChipEdit and you do not want Designer to move them during Layout, you should fix the macros in using the following procedure.

1. **Invoke ChipEdit.** Click the ChipEdit button in the Designer Main window.
2. **Select the macro(s) to be fixed.** Click the macro to be fixed in the PLACED list box or the Chip window. To select multiple macros, hold the Shift button and click multiple macros. To select all placed macros, choose the Select All Placed command from the Edit menu.
3. **Fix the macro(s).** Choose the Fix command from the Edit menu.

To unfix macros:

1. **Select the macro(s) to be unfixed.** Click the macro to be unfixed in the PLACED list box or the Chip window. To select multiple macros, hold the shift button and click multiple macros. To select all placed macros, choose the Select All Placed command from the Edit menu.
2. **Unfix the macro(s).** Choose the Unfix command from the Edit menu.

ChipEdit View Options

The PLACED and UNPLACED list boxes can display macro instance names flat or hierarchically. When macro instance names are displayed hierarchically, expanded levels are preceded by a plus sign (+) and collapsed levels are preceded by a minus (-) sign. Clicking the plus sign expands the hierarchy of a macro. Clicking the minus sign collapses the hierarchy of a macro. Macros are displayed hierarchically by default. Figure A-2 is an example of hierarchically displayed macros.

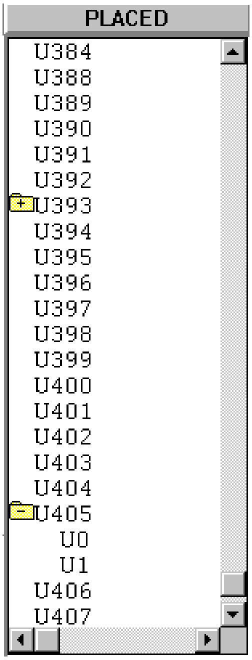
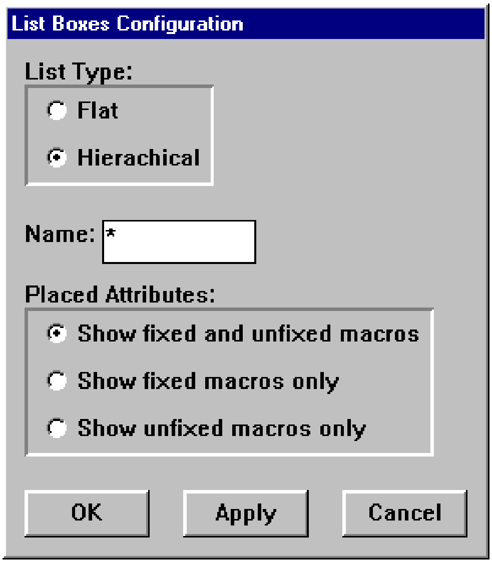

PLACED
U384
U388
U389
U390
U391
U392
+ U393
U394
U395
U396
U397
U398
U399
U400
U401
U402
U403
U404
- U405
U0
U1
U406
U407

Figure A-2. Macros Displayed Hierarchically

ChipEdit can also display certain types of macros (fixed, unfixed, or both). Both fixed and unfixed macros are displayed by default.

To change how macros are displayed:

1. **Choose the Configure List Box command from the Options menu.** The List Boxes Configuration dialog box is displayed.



List Boxes Configuration

List Type:

○ Flat

● Hierarchical

Name: *

Placed Attributes:

● Show fixed and unfixed macros

○ Show fixed macros only

○ Show unfixed macros only

Figure A-3. List Boxes Configuration Dialog Box

2. **Specify the List Type.** Select Flat or Hierarchical.
3. **Select a Placed Attributes option.**
4. **Click OK.**

B

Using Designer Script

This appendix contains information and procedures about using the Designer Script Language to implement designs using Designer. This includes information about how to write scripts and available script commands.

Running Designer in Batch Mode

You can run Designer in Batch mode with a Designer Script File (DSF) or through the Designer interface, with or without arguments. DSF files can run Designer in Background mode, Foreground mode, or Query mode.

Background Mode

In Background mode, Designer is invoked in Batch mode, the DSF is executed, then Designer is exited.

To run Designer in Background mode:

Type the following command at the prompt:

```
designer script:script_file.dsf script_mode:batch
```

Foreground Mode

In Foreground mode, Designer is invoked in Batch mode, the DSF is executed, then Designer is left running.

To run Designer in Foreground mode:

Type the following command at the prompt:

```
designer script:vl.dsf script_mode:startup
```

Query Mode

In Query mode, Designer is invoked in Batch mode, the DSF is executed, then Designer is left running.

To run Designer in Query mode:

Type the following command at the prompt:

```
designer script:qy.dsf script_mode:query
```

Query Mode allows the invocation of certain key query boxes. The following is a list of the commands that may be called.

```
setup_design
set_device
import_netlist
compile
operatopm_cond
dt_edit
pinedit
dt_analyze
```

Executing Scripts with Arguments

To executes a DSF with an argument, the DSF file name and the argument must be in quotes. for example:

```
designer script:"script_file.dsf cntroll"
```

Batch mode argument passing is only supported on UNIX. To use an argument with a DSF on the PC, you must use the Execute Script command from the File menu in the Designer interface.

Designer Interface

You can execute DSF files in Designer by selecting the Execute Script command from the File menu and then completing the information in the Execute Script dialog box, including the DSF file name and the arguments to be passed to the script.

Designer Script Language

The Designer Script Language supports all of the commands currently available from the Designer interface in a C-Language-like syntax. A log file containing these commands is generated as part of the Designer program. Every command returns a true/false value that can be used as part of an if statement. The overall commands allow for function definitions, variables, arithmetic, and if statements.

- **Function Definition may be defined including parameters.** Example:

```
test_program(print_value)
{
   print(“value = ”, print_value);
}
```

- **Variables are implied when they are used.** The two variable types are strings and integers. For example:

```
test_variable(i, j, k)
{
   l = 5;
   m = “hello world”;
   print(“i = ”, i, “j = ”, j, “k = ”, k, “m = ”, m, “n = ”,
n, “\n”);
 }
main()
{
   test_variable(5, 6, 7);
   test_variable(“five”, “six”, “seven”);
}
```

- The script supports the arithmetic operators ‘+,-,*, / and %’ of integers, and ‘+’ of strings.
- **Standard if statements are supported.** Comparison operators are ‘<. <=, ==, !=, >=, and >’.

Supported Commands

Table B-1 lists the available Actel script commands and provides brief descriptions of what they do. Refer to “Script Examples” on page 133 for examples of how to use these commands. Bold type indicates the argument that can be inserted in the command.

Table B-1. Supported Script Commands

Commands	Description
arg_count();	The argument count for the script file.
arg_value(i);	The argument value for the argument “i.”
batch_timer(inputfile, outputfile);	Run the timer, taking commands from the file **inputfile**, and writing output to the file **outputfile**.
chipedit();	Run the Chip Editor. Note: This only works in query mode.
close();	Close the current Actel database.
compile();	Perform the compile step.
constraints_exist();	Return a 1 if constraints exist; 0 otherwise.
dt_analyze();	Run the Interactive Timer. Note: This only works in query mode.
dt_edit();	Run the Delay Constraint Editor. Note: This only works in query mode.
export(fname, ftype);	The filetype **ftype** may be adl, afl, edif, crt, dcf, stf, loc, pin, log, diagnostic, design_script, session_script, stf_pre_layout, afm, fus, dio, sdf, sdf_pre_layout, verilog, vhdl, cob, sdf1.0, sdf1.0_pre_layout.

Table B-1. Supported Script Commands (Continued)

Commands	**Description**
extract();	Create files for the back annotation programs.
file_exists(fname);	Return a 1 if the file exists; 0 otherwise.
fix_all_placed_iopins();	Fixes all placed I/O.
fuse();	Create a fuse file for programming.
gen_pin_report(fname, format);	Generate a pin Report, writing to the file **fname**. The **format** parameter is optional and determines the pin report format. Legal values are Name and Number.
gen_status_report(fname);	Generate a status Report, writing to the file **fname**.
gen_timing_report(fname);	Generate a timing Report, writing to the file **fname**.
get(varname);	Get the value of the variable **varname**.
get_environment_var(name);	Return the environment variable's value.
import_aux_file(fname, ftype);	Read in the file **fname**. The value of **ftype** may be crt, dcf, loc, or pin.
import_netlist(fname, ftype);	Read in the design netlist in the file **fname**, of type **ftype**. The type ftype may be either edif, adl, or verilog.

Table B-1. Supported Script Commands (Continued)

Commands	Description
layout();	Place and Route the design, according to the previously defined states. Returns: 1 Success -92 Impossible constraint -91 DCF error -100 Place failed -73 Route failed
new_design ();	Create a new Actel database.
open(fname);	Open the actel database **fname**, or convert the previous als version of a design. Legal file types for this command are: .adb Designer Actel Database Files .als Designer files .def pre-Designer files
pinedit();	Run the Pin Editor. Note this only works in query mode.
print(value);	Print the variable **value** to the screen.
save();	Save the database.
save_as(fname);	Save the database in the file **fname**.
set(varname, varval);	Set the variable **varname**, to the value **varval**. See "Set Command Variables" on page 132 for details.
setup_design(design, fam);	Set up the design by the name of **design**, using the family **fam**.

Table B-1. Supported Script Commands (Continued)

Commands	Description
set_device("variable1=value1, variable2=value2, ..., variableN=valueN");	Set operating conditions as follows: **DIE** - Set to the target die. **PACKAGE** - Set the value to the target package. **SPEED** is one of: STD, -1, -2, -3, -F. **VOLTAGE** is one of: 5.0, 3.0 or 3.3/5.0. **PCI** is one of: 1 to set PCI compliance and 0 to turn off PCI compliance. **JTAG** is one of: 1 to Reserve JTAG Pins and 0 to use JTAG Pins. **PROBE** is one of: 1 to Reserve Probe Pins and 0 to use Probe Pins. **TEMPR** is one of: COM, IND, MIL or a three temperature range. **VOLTR** is one of: COM, IND, MIL or a three voltage range. Mixed voltage parts accept a six voltage range. The first three values apply to the I/O voltage range and the next three values apply to the core voltage range.
unfix_all_placed_iopins();	Unfixes all placed I/O.

Set Command Variables

The following command variables are used when setting the set(varname, varval) command, as described under the heading "set(varname, varval);" in Table B-1, "Supported Script Commands," on page 128.

USRDIR	*default directory*
DESIGN	*design name*

Import Variables

EDNINFLAVOR	*generic viewlogic mgc*
EDNINCFGFILE	*any unique edif configuration file*
NETLIST_NAMING_STYLE	*generic verilog VHDL*

Compile Variables

NL_PINS_REPLACE_ADB_PINS	0 *or* 1

Place and Route Variables

LAYOUT_MODE	STANDARD *or* TIMING_DRIVEN
PLAINC	0 *or* 1 *(incremental mode on or off)*
RESTRICTPROBEPINS	0 *or* 1
RESTRICTJTAGPINS	0 *or* 1

Timing Report Variables

TRPT_STHD_CHECK	0 *or* 1 *(report set-up and hold violations)*
TRPT_PATH_BRIEF	0 *or* 1 *(show critical paths only)*
TRPT_PASS_CLOCK	0 *or* 1 *(allow timing to pass thru clock pins)*
TRPT_PASS_ASYNC	0 *or* 1 *(allow timing to pass thru PRE and CLR pins)*
TRPT_PATH_EXPND	0 *or* 1 *(expand failed paths)*
PROC	WORST TYP BEST

Extraction Variables

`BA_DIR`	*directory for the extractor*
`BA_NAME`	*design name for the extractor*
`BA_CAE`	*type of CAE system*

Fuse Variables

`SIG`	*silicon signature value*

Script Examples

Following are two examples of Actel script files. The first is a very basic script that you would run to create an Actel database for a design. The second is a more complex script that you would run to modify an existing Actel database.

Basic Script for a New Design

The following basic script file is for a new design on a workstation. This script sets up the design, device, and operating conditions. The netlist is then imported from the local directory, and the design is run through Compile, Layout, and Extract. Finally, the database is saved to preserve the information.

```
main()
{
new_design();

set( "NETLIST_NAMING_STYLE", "VHDL" );
setup_design( "test", "42MX" );

import_netlist( "test.edn", "EDIF" );
set_device( "DIE = A42MX36, PACKAGE = 208 PQFP, SPEED = -2,
VOLTAGE = 3.3/5.0, PCI = 1, JTAG = 1, PROBE = 1, TEMPR = COM,
VOLTR = 3.00 3.30 3.60 4.75 5.00 5.25" );


set( "SIG", "1984" );
compile();
layout();
set("BA_CAE", "GENERIC");
extract();
fuse();
save_as("test.adb");

close();
}
```

There are several drawbacks to this basic example. The first is that the design name is hard coded (TEST), so you would have to make a script for each design. The second drawback is that if a failure occurs (in Compile for example), the subsequent functions (Layout and Extract) will also attempt to rerun Compile because of the demand-driven nature of the software. Also, you would use this script for one time only, as you would use the resulting database from this run for future iterations. The next example, which is more complex, shows methods for overcoming these drawbacks.

Script for an Existing Design

The following script is a more complex example for a design iteration on an existing database on PC. The database must be first opened, the modified netlist imported, and the functions through Extract executed. This script has examples of passing arguments and terminating execution as soon as a failure is detected.

Arguments can be passed in the command line or from the Execute Script command from the File menu on the Designer main screen. In this case, the argument is the design name (des_name), making this script generic for all designs. The only caveat is that default extension names (.adb, .edn, etc.) must be used. The script also takes advantage of the globally-defined working directory by having “.\” as the directory containing the design. The working directory is defined using the Preferences command from the File menu in the Designer Main window.

When you open the design, the netlist will automatically be imported if it has been modified. Taking advantage of the demand-driven feature of Designer, only Layout (which will force a Compile) and Extract are launched. If either of these functions fail, a *.log file is exported and the script terminated (return command). The design is saved before termination if further investigation is required.

```
main()
{
  des_name=(arg_value(1));

/* Open the design  */

  if(!(open(".\"+des_name+".adb")))
      {export(".\"+des_name+".log","log");
       save();
       return(1); }

  /* Process the design */

  set("LAYOUT_MODE","STANDARD");
  if(  (layout()!=1)
      { export(".\"+des_name+".log","log");
        save();
        return(1); }

  /* Generate file for back-annotation */
  set("BA_CAE","GENERIC");
  extract();
  export(".\"+des_name+".log","log");
  save();
  close();
  }
```

Log Files

You can generate a log file of the session by using the following run time parameter:

`logfile:<filename>`

C

Product Support

Actel backs its products with various support services including Customer Service, a Customer Applications Center, a Web and FTP site, electronic mail, and worldwide sales offices. This appendix contains information about using these services and contacting Actel for service and support.

Actel U.S. Toll-Free Line

Use the Actel toll-free line to contact Actel for sales information, technical support, requests for literature about Actel and Actel products, 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) 1256 305600.
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.

Customer Applications Center

The Customer Applications Center is staffed by applications engineers who can answer your hardware, software, and design questions.

All calls are answered by our Technical Message Center. 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 7:30 a.m. to 5 p.m., Pacific Standard Time, Monday through Friday.

The Customer Applications Center number is (800) 262-1060.

European customers can call +44 (0) 1256 305600.

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. Guru is available 24 hours a day, seven days a week.

Web Site

Actel has a World Wide Web home page where you can browse a variety of technical and non-technical information. Use a Net browser (Netscape recommended) to access Actel's home page.

The URL is **http://www.actel.com**. You are welcome to share the resources we have provided on the net.

Be sure to visit the “Actel User Area” on our Web site, which contains information regarding: products, technical services, current manuals, and release notes.

FTP Site

Actel has an anonymous FTP site located at **ftp://ftp.actel.com**. You can directly obtain library updates, software patches, design files, and data sheets.

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. The e-mail account is monitored several times per day.

The technical support e-mail address is **tech@actel.com**.

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

San Diego
Tel: 619.938.9860
Fax: 619.938.9887

Thousand Oaks
Tel: 805.375.5769
Fax: 805.375.5749

Colorado

Tel: 303.420.4335
Fax: 303.420.4336

Florida

Tel: 407.677.6661
Fax: 407.677.1030

Georgia

Tel: 770.831.9090
Fax: 770.831.0055

Illinois

Tel: 847.259.1501
Fax: 847.259.1572

Maryland

Tel: 410.381.3289
Fax: 410.290.3291

Massachusetts

Tel: 978.244.3800
Fax: 978.244.3820

Minnesota

Tel: 612.854.8162
Fax: 612.854.8120

North Carolina

Tel: 919.376.5419
Fax: 919.376.5421

Pennsylvania

Tel: 215.830.1458
Fax: 215.706.0680

Texas

Tel: 972.235.8944
Fax: 972.235.965

International Sales Offices

Canada
Suite 203
135 Michael Cowpland Dr,
Kanata, Ontario K2M 2E9

Tel: 613.591.2074
Fax: 613.591.0348

France
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
Bahnhofstrasse 15
85375 Neufahrn

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

Hong Kong
Suite 2206,
Parkside Pacific Place,
88 Queensway

Tel: +011.852.2877.6226
Fax: +011.852.2918.9693

Italy
Via Giovanni da Udine No. 34
20156 Milano

Tel: +39 (0)2.3809.3259
Fax: +39 (0)2.3809.3260

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
135-090, 18th Floor,
Kyoung Am Building
157-27 Samsung-dong
Kangnam-ku, Seoul

Tel: +82 (0)2.555.7425
Fax: +82 (0)2.555.5779

Taiwan
4F-3, No. 75, Sec. 1,
Hsin-Tai-Wu Road,
Hsi-chih, Taipei, 221

Tel: +886 (0)2.698.2525
Fax: +886 (0)2.698.2548

United Kingdom
Daneshill House,
Lutyens Close
Basingstoke,
Hampshire RG24 8AG

Tel: +44 (0)1256.305600
Fax: +44 (0)1256.355420

Index

A

Actel
  FTP Site 139
  Libraries 19
  Manuals xiii
  Web Based Technical Support 138
  Web Site 138
ACTgen
  Buffering 46
  Fan-In Control 46
  Features 41
  Generating a Macro Report 48
  Generating New Macros 43
  LOG File 48
  Main Window 42
  Modifying Existing Macros 44
ACTmap
  Batch Mode 62
  Compiling VHDL 54
  Configuration Files 61
  Defining I/Os 58
  Features 51
  Hierarchical Project 59
  Main Window 53
  Optimizing Netlists 55
  Translating a Netlist 57
Actual Delay 98, 100
Adding
  Global Networks 20
  Ground 19
  Pins 32
  Power 19
  Properties 32
Assigning Pins 80
Assumptions xii
Automatic
  Fan-In Reduction 22
  Logic Reduction 22
  Module Reduction 22
  Pin Assignment 33
Auxiliary Files 64, 70

B

Back Annotation 87
  Effects of Combiner 27
Batch Mode
  ACTmap 62
  Designer 125–135
  Designer Commands 128
  Designer Variables 132
Batch Timer 112
Behavioral Simulation 11
Buffering 30
  ACTgen 46

C

Calculating Module Utilization 31
Capturing a Design
  HDL-Based 11
  Schematic-Based 8
Changing Design Information in Designer 71
ChipEdit 117–123
  Clock Module Symbol 119
  Color Definitions 119
  Combinatorial Module Symbol 119
  Fixed Logic Module Symbol 119
  Fixing Macros 121
  Input/Output Module Symbol 119
  Moving Macros 121
  Multiple Windows 120
  Placing Macros 120
  Reserved Symbol 119

- Selected Objects Symbol 119
- Sequential Module Symbol 119
- Symbol Definitions 119
- Unplaced Module Symbol 119
- View Options 122
- Window 117

CLK 21
CLKA 21
CLKB 21
Clock
- Dedicated 21
- Exceptions 76
- Routed 21

Clock Constraints 76
Clock to Output Delay 105
Combinatorial Module Reduction 23
Combiner 22
- Back Annotation Effects 27

Combining Modules 22
- 40MX 28
- ACT 1 28

Compile 2, 63–68
- A New Design 63
- Options 64
- VHDL 54

Configuration Files 61
Constant Input Reduction 26
Constraint
- Clock 76
- Definitions 79
- Entering in DT Edit 76
- Exceptions 76
- Name 99
- Path 77

Contacting Actel
- Customer Service 137
- Electronic Mail 139
- Technical Support 138
- Toll-Free 137
- Web Based Technical Support 138

Conventions xiii
CRT File 70
Cumulative Delay 100
Customer Service 137

D

DCF File 70
DCLK 37
Dedicated Clocks 21
- HCLK 21
- IOCLK 22

Defining I/Os 58
Delay 100
- Clock to Out 105
- Estimating 38
- Input to Output 107
- On-Chip 109
- Type 100

Design Creation/Verification 8, 11
- Behavioral Simulation 11
- EDIF Netlist Generation 8, 12
- Functional Simulation 8
- HDL Source Entry 11
- Schematic Capture 8
- Structural Netlist Generation 12
- Structural Simulation 12
- Synthesis 12

Design Flow
- Design Creation/Verification 8, 11
- Design Implementation 9, 12
- Schematic-Based 7–9
- Synthesis-Based 10–13

Design Implementation 9, 12
  Place and Route 9, 13
  Timing
    Analysis 9
  Timing Analysis 13
  Timing Simulation 9, 13
Design Information
  Changing in Designer 71–75
Design Layout 9, 13
Design Setup 65, 71
Design Synthesis 12
Designer
  Assigning Pins 82
  Auxiliary Files 64, 70
  Batch Mode 125–135
  Batch Mode Commands 128
  Batch Mode Variables 132
  Batch Timer 112
  Changing Design Information 71–75
  ChipEdit 117–123
  Compile 63
  Compiling a New Design 63–68
  Converting a Design 69
  Converting Designs 69
  Device Variations 67
  DT Analyze 9, 13, 95–111
  DT Edit 76–80
  Existing Designs 69–70
  Exiting 94
  Exporting Files 89, 115
  Extract 87
  Fixing PIns 82
  Flip Flop Report 90
  Fuse 113–115
  Import Netlist Options 64
  Importing a Design 63–68
  Incremental Placement 85
  Layout 84–87
  Layout Failures 86
  Main Window 63
  Naming a Design 65
  Opening a Design 69–70
  Overview 1–5
  Pin Report 90
  PinEdit 81–83
  Place and Route 9, 13
  Preferences 94
  Programming File 115
  Reports 90
  Script Language 125–135
    Commands 128
    Variables 132
  Selecting a Package 65, 72
  Selecting Device Family 65
  Selecting Device Variations 73
  Selecting Die 65, 72
  Selecting Die Voltage 67, 73
  Selecting Operating Conditions 68, 74
  Selecting Speed Grade 65, 72
  Status Report 90
  Timing Analysis 9, 13
  Timing Constraints 76–80
  Timing Report 91
Designs
  Hierarchical 19
  Multiple Sheet 19
Device
  Programming 9, 13
  Selection 65, 72
  Variations 67, 73
  Verification 9, 13
Device Selection 65

Device Setup Wizard 72
  Device Selection 65, 72
  Device Variations 67, 73
  Operating Conditions 68, 74
Device Variations 67, 73
DirectTime Analyze. See DT Analyze
DirectTime Edit. See DT Edit
Document Assumptions xii
Document Conventions xiii
Document Organization xi
DT Analyze 4, 95–111
  Analyzing Paths 96
  Clock to Output Delay 105
  Examples 101–111
  Expanding Paths 95
  Input to Output Delay 107
  Maximum Operating Frequency 104
  On Chip Delay 109
  Preferences 97
  Register to Register Delay 103
  Static Timing Analysis 9, 13
  Viewing Results 98
DT Edit 76–80
  Clock Constraints 76
  Constraint Definitions 79
  Exceptions 76
  Guidelines 78
  Path Constraints 77
  Window 76
DT Layout 85
Duplicating Logic 30

E
EDIF Netlist Generation
  Schematic-Based 8
  Synthesis-Based 12
Electronic Mail 139
End Terminal 98
Entering Constraints in DT Edit 76
Estimating Delays 38
Exiting Designer 94
Expanding Paths 99
Expanding Pins in DT Analyze 100
Exporting Files 89, 115
Extract 4, 87

F
Failures, Layout 86
Fan Out 30
Fan-In Control
  ACTgen 46–48
Fan-In Reduction 22, 27
Files
  Auxiliary 64, 70
  Configuration 61
  Exporting 89
Fixing
  Macros 121
  Pins 82
Flip Flop Report 90
Functional Simulation 8
Fuse 4, 113–115
  Generating Programming Files 113
  Silicon Signature 113

G
Gate-Level Netlist 12
Generating 38
  ACTgen Macros 43
  EDIF Netlist 8, 12
  Gate-Level Netlist 12
  Reports in Designer 90

- Silicon Signature 113
- Structural Netlist 12

Generating Programming Files 113
Global Network 20
GND 19
Ground 19

H
HCLK 21
HDL Source Entry 11
Hierarchical
- Designs 19
- Project 59

High Fan Out 30

I
I/O
- Clock 22
- Defining 58
- Module Utilization 31

Implementing a Hierarchical Project 59
Importing
- Auxiliary Files 64
- Netlist into Designer 63

Importing Auxiliary Files 70
Incremental Placement 85
Input to Output Delay 107
IOCLK 22
IOPCL 22

J
JTAG Pins 38
- Reserving 67, 73

L
Layout 3, 84–87
- Failures 86
- Incremental Placement 85

Libraries 19
Load 100
LOG File
- ACTgen 48

Logic Module Utilization 31
Logic Reduction 22, 26

M
Macro 100
- ACTgen 43, 44
- Fixing 121
- Moving 121
- Placing 120

Manual Pin Assignment 33
Maximum Operating Frequency 104
Modifying ACTgen Macros 44
Module Reduction 22
- 40MX 28
- ACT 1 28
- Combinatorial 23

Moving Macros 121
Multiple Sheet Designs 19

N
Naming a Design 65
Naming Conventions
- Schematic 15
- Verilog 15
- VHDL 17

Net
- Loading 30
- Name 100

Netlist
- Importing into Designer 63–68

  Importing Options 64
  Optimizing 55
  Translating 57
Netlist Generation
  EDIF 8, 12
  Gate-Level 12
  Structural 12
Network 20
New Design 63

O
On Chip Delay 109
On-Line Help xv
Opening a Design 69–70
  Converting 69
Operating Conditions, Selecting 68, 74
Optimizing Netlists 55
Output Load 100

P
Path
  Analyzing 96
  Constraints 77
  Expanding 95, 99
Pin
  Adding 32
  Assigning 80
  Assignment 32–38
  Automatic Assignment 33
  DCLK 37
  JTAG 38
  Name 100
  PRA 37
  PRB 37
  Printing a List 90
  Printing Layout 83
  Probe 37
  SDI 37
  TCK 38
  TDI 38
  TDO 38
  TMS 38
  Unused I/O 37
PIN File 70
PinEdit 3, 81–83
  Assigning Pins 82
  Committing Pin Assignments 83
  Fixing Pins 82
  Printing Pin Layout 83
  Window 81
Place and Route 9, 13
Placing Macros 120
Post-Synthesis Simulation 13
Power 19
PRA 37
PRB 37
Preferences
  Designer 94
  DT Analyze 97
Preserving Macros 32
Printing
  Pin List 90
  Timing Information 91
Printing Pin Layout 83
Probe Pins 37
  Reserving 67, 73
Product Support 137–140
  Customer Applications Center 138
  Customer Service 137
  Electronic Mail 139
  FTP Site 139
  Technical Support 138

- Toll-Free Line 137
- Web Site 138

Programming
- Device 13
- File Format 115

Programming a Device 9
Property, ALSPRESERVE 32

R

Rank 98
Register to Register Delay 103
Related Manuals xiii
Remapping 24
Reports
- ACTgen 48
- Designer 90

Required Delay 99
Routed Clocks 21
- CLK 21
- CLKA 21
- CLKB 21
- QCLK 21

S

Schematic Capture 8
Schematic-Based Design Flow 7–9
- Design Creation/Verification 8
- Design Implementation 9
- Programming 9
- System Verification 9

SDF File 87
SDI 37
Selecting
- Die 65, 72
- Die Voltage 67, 73
- Junction Temperature Range 68
- Operating Temperature Range 68
- Package 65, 72
- PCI Compliance 67, 73
- Speed Grade 65, 72
- Voltage Range 68

Selecting Junction Temperature Range 74
Selecting Operating Temperature Range 74
Selecting Voltage Range 74
Sequential Remapping 24
Setup Design 71
Silicon Signature 113
- Format 115

Simulation
- Behavioral 11
- Functional 8
- Schematic-Based 8, 9
- Structural 12
- Synthesis-Based 11, 12, 13
- Timing 9, 13

Slack 99
Special Clocks 22
- IOPCL 22

Standard Layout 84, 85
Start Terminal 98
Static Timing Analysis 9, 13, 95–111
Status Report 90
STF File 87
Structural Netlist Generation 12
Structural Simulation 12
Symbol, Top Level 38
Synthesis 12
Synthesis-Based Design Flow 10–13
- Design Creation/Verification 11
- Design Implementation 12
- Programming 13
- System Verification 13

System Verification 9, 13
  Silicon Explorer 9, 13

T
TCK 38
TDI 38
TDO 38
Technical Support 138
Timing Analysis 9, 13, 95–111
Timing Constraint 38
  Clock 76
  Definitions 79
  Entering in DT Edit 76
  Exceptions 76
  Path 77
Timing Driven Layout 85
Timing File 87
Timing Report 91
Timing Simulation 9
  Post-Synthesis 13
  Timing File 87
TMS 38
Toll-Free Line 137
Top Level Symbol 38
Total 100
Translating a Netlist 57

U
Unit Delays 8, 11
Unused Logic Removal 26
Utilization 31

V
VCC 19
VHDL, Compiling 54

W
Web Based Technical Support 138