ULTRA LOGIC

WARP2™ VHDL Compiler for PLDs and CPLDs

USER'S GUIDE

CYPRESS

Warp2^{TM}

VHDL Development System

User's Guide

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
(408)943-2600

Warp2 Installation

For PC's:

Installing *Warp2* on your IBM PC or compatible computer requires about 55 Megabytes of hard disk space. The installation procedure is as follows:

1. Write down the serial number found on the first disk; you will be asked for it later.
2. Insert disk 1 into a floppy disk drive.
3. Select File/Run from the Windows program manager.
4. Type `a:install` or `b:install`, as appropriate, and select OK.
5. Answer questions from the installation script.

Note that if you choose the option to create example DOS configuration files, the files are created with a ".exm" extension, e.g., the example file for "autoexec.bat" would be "autoexec.exm".

Note also that the example file only contains the difference between the old and new files, not a complete copy. Because of this, Cypress recommends that you simply choose the option to go ahead and modify the originals. The installation procedure copies the original file to a ".bak" extension. Example files "win.exm" and "system.exm," found in your Windows directory, are also created.

NOTE

IMPORTANT DOS REMINDER...

Your PATH must be shorter than 128 characters. This is a DOS limitation.

IMPORTANT WINDOWS TIPS...

1. The Optimizing Windows chapter of the Microsoft Windows User's Guide includes numerous ideas to increase performance. We especially recommend creating a SMARTdrive of 1 to 2 Megabytes. We also recommend a Windows 3.1 virtual memory size of 16 Megabytes.

2. Some PC clones utilize the upper portion of the first megabyte of memory in a manner that conflicts with Windows. If any *Warp2* (or other) application "hangs," try adding the following line to your SYSTEM.INI file (found in the Windows directory) in the section marked as indicated:

```
[386Enh]
        EMMexclude=A000-FFFF
```

After adding this line, re-start Windows. If the problem persists, consult your PC manufacturer for additional assistance.

For Sun:

Installing *Warp2* on your Sun workstation requires about 38 Megabytes of hard disk space. The installation procedure is as follows:

1. If it does not already exist, create the directory where you want the software to be located.
2. For installation from a tape, tar the contents of the tape, run the install script and answer the questions:

   ```
   tiger% mkdir /usr/local/warp
   tiger% cd /usr/local/warp
   tiger% tar xvf /dev/rst8
   tiger% ./install
   ```

   For installation from disks, insert the first disk and bar the contents before running the install script:

   ```
   tiger% mkdir /usr/local/warp
   tiger% cd /usr/local/warp
   tiger% bar xvfZ /dev/rfd0
   tiger% ./install
   ```

3. When asked for the serial number, simply enter your company name, or copy the number from the tape (or first disk) label.
4. In your .cshrc or .login file, set a variable called CYPRESS_DIR pointing to your install directory. Make sure your path includes */your_Cypress_directory/bin.*

   For example:

   ```
   setenv CYPRESS_DIR /usr/local/warp
   set path=($path $CYPRESS_DIR/bin)
   ```

If you are installing *Warp2*, skip immediately to Step 7. If you are installing *Warp2+*, perform Steps 5 and 6 at this time.

5. Set your pASIC environment variables in your .cshrc file as follows:

   ```
   setenv SPDE_ROOT $CYPRESS_DIR/spde
   setenv XVTPATH $SPDE_ROOT/print
   setenv UIDPATH $SPDE_ROOT/%N.uid
   set path=($path $SPDE_ROOT)
   ```

6. Copy the pASIC configuration file into your home directory:

   ```
   cp $CYPRESS_DIR/spde/.spderc $HOME
   ```

7. By default on the Sun Sparc, the Galaxy and Nova interfaces use the OpenLook toolkit. If the user wishes to use the Motif interface instead, set the environment variable CYPRESS_MOTIF to TRUE.

   For example:

   ```
   setenv CYPRESS_MOTIF TRUE
   ```

   Adding this command to your .login or .cshrc files will make this interface change permanent.

Warp2™

VHDL Development System

Warp2 Overview

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
(408) 943-2600

Cypress Software License Agreement

1. LICENSE. Cypress Semiconductor Corporation ("Cypress") hereby grants you, as a Customer and Licensee, a single-user, non-exclusive license to use the enclosed Cypress software program ("Program") on a single CPU at any given point in time. Cypress authorizes you to make archival copies of the software for the sole purpose of backing up your software and protecting your investment from loss.

2. TERM AND TERMINATION. This agreement is effective from the date the diskettes are received until this agreement is terminated. The unauthorized reproduction or use of the Program and/or documentation will immediately terminate this Agreement without notice. Upon termination you are to destroy both the Program and the documentation.

3. COPYRIGHT AND PROPRIETARY RIGHTS. The Program and documentation are protected by both United States Copyright Law and International Treaty provisions. This means that you must treat the documentation and Program just like a book, with the exception of making archival copies for the sole purpose of protecting your investment from loss. The Program may be used by any number of people, and may be moved from one computer to another, so long as there is **No Possibility** of its being used by two people at the same time.

4. DISCLAIMER. THIS PROGRAM AND DOCUMENTATION ARE LICENSED "AS-IS," WITHOUT WARRANTY AS TO PERFORMANCE. CYPRESS EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTY OF MERCHANTABILITY OR

Table of Contents

Chapter 1

Warp2 Process Overview

About This Chapter

Overview

This chapter provides an overview of the design process using *Warp2* tools.

1.1. What do you do with *Warp2*?

Warp2 is a collection of tools for designing and synthesizing circuits to be transferred to programmable logic devices. The *Warp2* tool set comes in two versions: *Warp2* (for targeting PLDs) and *Warp2+* (for targeting PLDs and pASIC380 FPGAs).

With *Warp2*, you can:

- use VHDL descriptions (structural, behavioral, or both) to describe a design.
- compile and synthesize the resulting design description. "Compile" means check the design description to make sure it's syntactically correct and contains no obvious logical errors, such as unconnected inputs, etc. "Synthesize" means convert the design description into a mapping file that can be transferred to a programmable device.
- create a program file for any supported programmable logic device (PLD).

With *Warp2+*, you can also target pASIC380 FPGAs.

There are other tasks you can perform, but at the highest conceptual level, that's it in a nutshell.

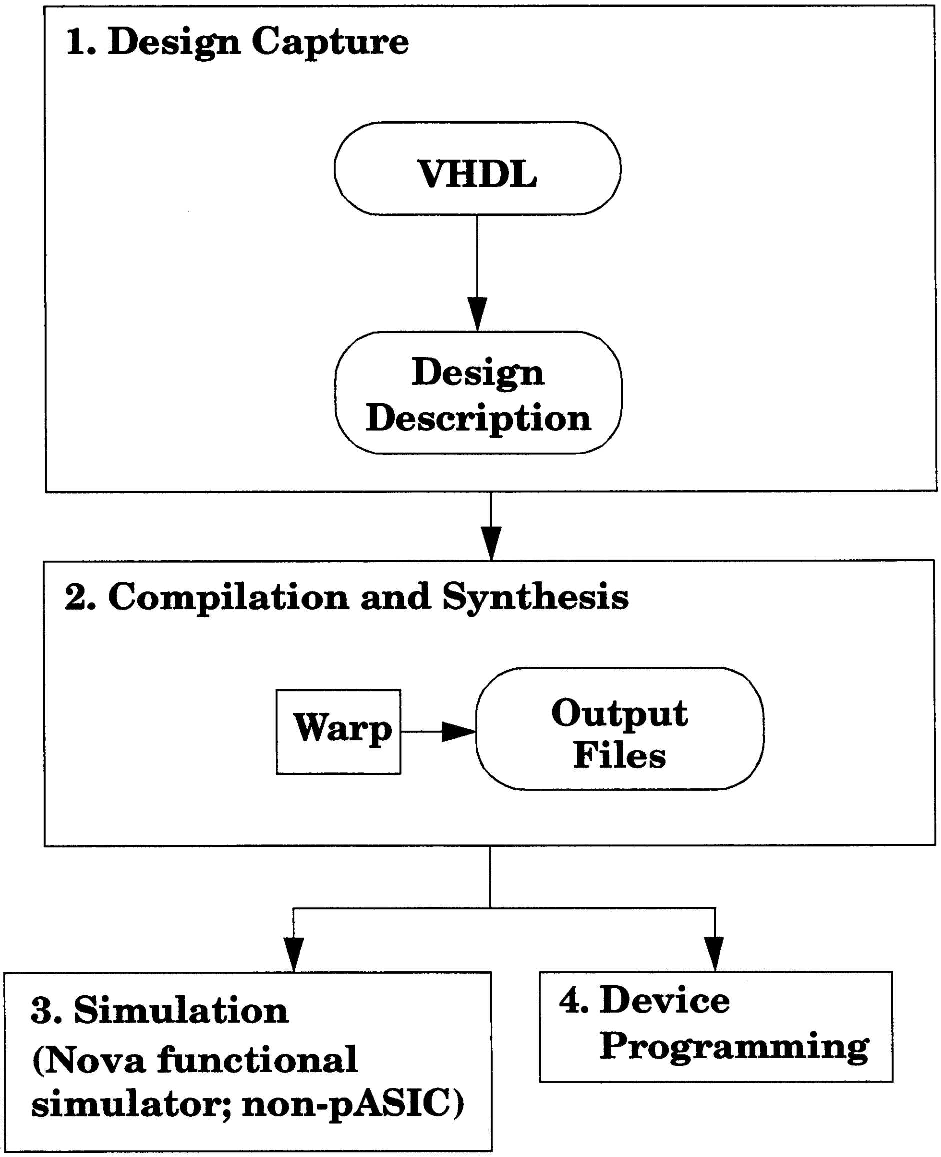

1. Design Capture
VHDL
Design Description
2. Compilation and Synthesis
Warp
Output Files
3. Simulation (Nova functional simulator; non-pASIC)
4. Device Programming

Figure 1-1. Warp2 Conceptual Process Flow.

1.2. What kinds of devices can I target using Warp2?

Warp2 supports a wide variety of Cypress programmable devices. *Warp2* features a common user interface so that the process of targeting one device is much like that of targeting any other.

Table 1-1 lists the parts for which you can target designs when using *Warp2*. These parts can be divided into five families:

- PLDs: the bulk of the Cypress programmable logic device family.
- MAX CPLDs: the Cypress high-density complex PLD family.
- FLASH370 CPLDs: the Cypress high-speed, high-density FLASH complex PLD family.
- pASIC380 FPGAs: the Cypress very-high-speed CMOS FPGA family (supported in *Warp2+* and *Warp3*).

Table 1-1.
***Warp2*-Supported Parts**

Family	Part #'s			
PLDs	C16L8 C16V8 C22V10	C16R4 C20G10 C22VP10	C16R6 C20G10C C331	C16R8 C20RA10 C335
MAX CPLDs	C341 C346	C342 C346H	C343	C344
FLASH370 CPLDs	C371	C374	C375	
pASIC380 FPGAs[1]	C381A C385A	C382A C386A	C383A	C384A

1. Supported in *Warp2+* only.

1.3. What's the design process in *Warp2*?

The usual order of design in *Warp2* is as follows: (1) describe the design in VHDL; (2) compile and synthesize; (3) process QDIF file (when targeting pASIC380 FPGAs only); and (4) program the target device.

You can describe designs in *Warp2* behaviorally, structurally, or using a combination of both.

As their name implies, *behavioral descriptions* specify the way a design behaves. Behavioral descriptions take the form of text files containing statements written in a hardware description language. In the case of *Warp2*, this hardware description language is VHDL. Automated tools convert the behavioral description into circuit structure that implements the specified behavior.

In the *Warp2* system, you write a VHDL source file that describes your design's behavior, then you use the *Warp* VHDL synthesis compiler to synthesize a circuit that implements that behavior.

Structural descriptions specify the components that make up a design, as well as how they are connected (like a netlist). Structural descriptions can be included in VHDL source files.

Descriptions can be hierarchical; that is, a design at one level can use a component described at a lower level. Once a design is completely described, whether behaviorally or structurally, *Warp* compiles and synthesizes this representation, i.e., converts it into a format that can either be directly programmed onto a PLD or CPLD, or that can be processed further to program pASIC380 FPGAs (*Warp2+* and *Warp3* only).

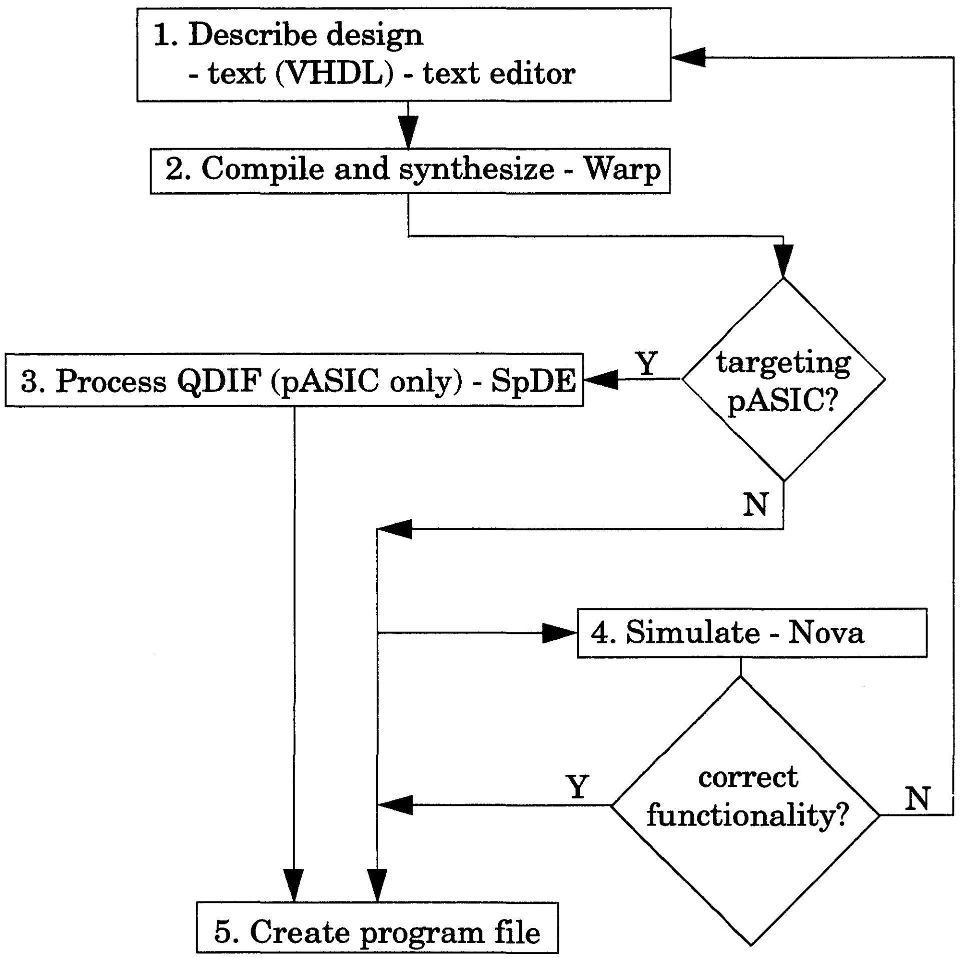


Figure 1-2. Warp2 Process Overview (Detailed).

1.4. What are "parts libraries," and why should I know about them?

Parts libraries are directories in which VHDL descriptions of components are stored. A comprehensive set of parts libraries comes with *Warp2*.

Warp2 provides libraries containing the following types of components:

- adders
- multipliers
- counters
- common logic gates
- input/output components
- memory elements
- multiplexers
- registers
- shifters
- TTL components.

You can open the VHDL description of any library component for viewing with any text editor. The default location of the VHDL descriptions of library components is:

```
\WARP\LIB\COMMON\FAMILY.VHD
```

where *family* is the generic name of a particular component group (e.g., `COUNTER.VHD`, `GATES.VHD`).

To instantiate a component in a VHDL description, use a VHDL component instantiation statement. See the description of components in Chapter 5 of the *Warp Synthesis Compiler Manual*.

For more information about the libraries, refer to the *Warp System Library Manual.*

1.5. Where can I call if I have problems?

Cypress Semiconductor has offices around the world. If you need information about (or if you experience any difficulties using) any Cypress product, call one of the numbers listed below.

Country	Toll- Free Number	Country	Toll-Free Number
N. America	1-800-419-1481	Japan	0031-11-1731
Australia	0014-800-125-203	Korea	008-1-800-942-8203
Belgium	11-8729	Netherlands	06-022-5303
Denmark	8001-0413	Norway	050-12068
Finland	9800-10065	Singapore	800-1758
France	05-90-1251	Spain	900-99-1163
Germany	0130-81-1902	Sweden	020-795-637
Hong Kong	800-7214	Switzerland	045-05-8808
Israel	00-17-942-1803	UK	0800-89-7339
Italy	1678-97-034		

Chapter 2

Warp2 Tools

About This Chapter

Overview

Warp2 comprises a set of tools for creating and synthesizing designs for programmable logic devices. This chapter describes each tool in the set.

2.1. Galaxy

Galaxy is the graphical user interface for *Warp*, *Warp2*'s VHDL compilation and synthesis tool.

To bring up Galaxy on an IBM PC or compatible, double-click on the GALAXY icon within the *Warp2* or *Warp2+* program group.

To bring up Galaxy on a Unix workstation, type "galaxy &<CR>" from the command line.

To run *Warp* directly (i.e., without the Galaxy user interface), type a *Warp* command line at the shell prompt. See Chapter 2 of the *Warp Synthesis Compiler Manual* for more information about the Warp command line.

For complete information on *Warp* and Galaxy, see the *Warp Synthesis Compiler Manual*.

2.2. *Warp*

Warp is *Warp2*'s VHDL synthesis compiler.

Warp takes a VHDL description of a circuit as input. *Warp* produces JEDEC files (used for programming PLDs) or QDIF files (used by the SpDE tools when targeting pASIC380 FPGAs).[1]

For complete documentation of *Warp* and Galaxy, see the *Warp Synthesis Compiler Manual.*

1. available with *Warp2*+ and *Warp3* only.

2.3. Nova

Nova is *Warp2*'s JEDEC functional simulator.

The Nova user interface gives you an easy way to:

- specify JEDEC files to simulate;
- read or write stimulus files;
- edit input waveform traces;
- simulate the behavior of a design;
- alternate between various views (i.e., collections of signals), and specify signals to include in each view;
- specify the length and resolution of a simulation;
- specify segments, where you can re-apply and edit initial conditions, in order to compare results of differing initial conditions side-by-side;
- and lots of other useful capabilities.

For complete documentation of the Nova simulator, see the *Nova User's Manual*, which comes as part of the *Warp2* documentation set.

2.4. Place & Rte

The "Place&Rte" tool processes a QDIF file (generated by *Warp*) in order to target pASIC380 FPGA's. This tool is available with *Warp2+* and *Warp3* only.

The "Place&Rte" tool is a tool provided by QuickLogic, a Cypress technology partner.

Place&Rte is used to perform automatic placement and routing, delay modeling, critical-path timing analysis, automatic test vector generation, and device programming and test.

For more information about the Place&Rte tool, see the *SpDE / Warp System User's Manual*, included in the *Warp2* documentation set.

Index

Warp2™

VHDL Development System

Tutorial

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
(408) 943-2600

Cypress Software License Agreement

1. LICENSE. Cypress Semiconductor Corporation ("Cypress") hereby grants you, as a Customer and Licensee, a single-user, non-exclusive license to use the enclosed Cypress software program ("Program") on a single CPU at any given point in time. Cypress authorizes you to make archival copies of the software for the sole purpose of backing up your software and protecting your investment from loss.

2. TERM AND TERMINATION. This agreement is effective from the date the diskettes are received until this agreement is terminated. The unauthorized reproduction or use of the Program and/or documentation will immediately terminate this Agreement without notice. Upon termination you are to destroy both the Program and the documentation.

3. COPYRIGHT AND PROPRIETARY RIGHTS. The Program and documentation are protected by both United States Copyright Law and International Treaty provisions. This means that you must treat the documentation and Program just like a book, with the exception of making archival copies for the sole purpose of protecting your investment from loss. The Program may be used by any number of people, and may be moved from one computer to another, so long as there is **No Possibility** of its being used by two people at the same time.

4. DISCLAIMER. THIS PROGRAM AND DOCUMENTATION ARE LICENSED "AS-IS," WITHOUT WARRANTY AS TO PERFORMANCE. CYPRESS EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTY OF MERCHANTABILITY OR

Table of Contents

Chapter 1 Introduction

About This Chapter

Overview

The *Warp2* Tutorial walks you through a common sequence of operations in using *Warp2*. The Tutorial shows you how to create, compile, synthesize, and simulate a design.

This chapter presents:

- introductory information about *Warp2*;
- a discussion of what this manual is and where it fits in the larger scheme of *Warp2* documentation;
- our assumptions about tools you should be familiar with to run the Tutorial, including sources of information about those tools;
- some conventions about typography, wording, and illustrations used in this manual;
- installation and licensing requirements for running the Tutorial;
- the objectives of the Tutorial; and
- the contents of the Tutorial.

1.1. Introduction to *Warp2*

Warp2 is a Cypress Semiconductor software product that enables users to describe electronic designs using VHDL, then compile and synthesize those descriptions to program Cypress parts, such as PLD's and pASIC380 FPGA's.[1]

Warp2 consists of three programs:

- The *Warp* VHDL compiler translates VHDL text descriptions into files that can be mapped onto programmable parts.
- The Nova JEDEC functional simulator allows you to verify the correctness of a design by simulating its behavior.
- The SpDE toolkit contains a set of tools for fitting designs onto pASICs. This tool set includes a placer, router, logic optimizer, path analyzer, and an automatic test vector generator, among others. (SpDE is only available with *Warp2+* and *Warp3*.)

1. Available with *Warp2+* and *Warp3* only.

1.2. Conventions

The following conventions are used throughout this manual:

Notational Conventions:

Menu items	Whenever an item from a menu is referenced, the reference takes the form menu-name/item-name/item-name... The first entry in the reference is the name of the menu; the second is the name of the menu item; the third and succeeding entries indicate choices from sub-menus. Example: "Select File/Open..." tells you to pull down the File menu, and select the "Open..." item from it.
Italic	Words are italicized for emphasis or to draw attention to new terms.
`Courier`	Denotes the contents of a text file, or indicates that displayed text is system output. Also indicates the text of typed commands.

File-Naming Conventions

Warp2 runs on two different types of platforms: IBM PC's and compatible computers, and Sun workstations.

IBM PC's and compatible computers specify file locations by designating a disk drive and using a backslash (\) character to distinguish directory levels, e.g., "c:\level1\level2\myfile".

Sun workstations specify file locations using a forward slash (/) and no disk drive designator, e.g., "/level1/level2/myfile".

For consistency and brevity, this manual uses the same notation as IBM PC's and compatibles when referring to file locations. Unix platform users are asked to make appropriate translations.

Mouse Conventions

The following terms describe common actions you might perform in using this manual:

Click	Place the mouse cursor over an object, then press and release the appropriate mouse button.
Double-click	Position the mouse cursor over an object, then press and release the appropriate mouse button twice in rapid succession.
Drag	Position the mouse cursor over an object or at a specified location. Press and hold the appropriate mouse button. While holding down the mouse button, move the cursor to the new location. Finally, release the mouse button.
Press	When you are instructed to "press" a specified key, locate the key on the keyboard and press it.
Select	When you are instructed to "select" an option, move the cursor over the option, then click the appropriate mouse button.

Other Conventions

The following visual indicators denote special situations you may encounter while using this manual:

NOTE

This icon indicates a *note*: a point in the tutorial where you must exercise special caution, or where the procedure might vary depending upon the platform you're using, or where there's something else you should know about that doesn't fit into the main flow of the text.

This icon indicates a *hint*: a point in the tutorial where you could save a little time, a few keystrokes or much frustration, if you only knew the hint that's being explained.

This icon indicates a *shortcut*. This tutorial has been designed so that you can choose which parts of the *Warp2* design, compilation, synthesis, and simulation process you want to work on. When you see this symbol, you're being given a choice to skip to another section of the tutorial.

1.3. Objectives

The objective of the *Warp2* Tutorial is to demonstrate the use of *Warp2* tools for creating, compiling, synthesizing, and simulating designs.

Tools to be demonstrated in the *Warp2* Tutorial include:

- *Warp*: compiles VHDL descriptions, produces JEDEC files to program PLD's, and produces QDIF files used in programming pASIC380 FPGA's.
- Nova: simulates the JEDEC file created in the *Warp* step.

1.4. About the Tutorial

The *Warp2* Tutorial shows how to use tools to create, compile, synthesize, and simulate designs.

Chapter 2 of this Tutorial demonstrates, in step-by-step fashion, how to create designs using *Warp2* tools.

Consider the following design problem: we wish to design a controller for a soft-drink dispensing machine. The machine has two bins to dispense Pepsi and Coca-Cola, respectively. Each bin holds three cans of soft drink. (This could be any positive integer, but three is an easy number to simulate with.)

We want the circuit to dispense a beverage when the user presses a button for that beverage and one or more cans of the beverage are available. We want the circuit to NOT dispense a beverage when no cans of that beverage are available. We want to get a REFILL signal when both bins are empty. We also want to be able to press a RESET signal to tell the circuit that the machine has been replenished and that the bins are full again.

The solution, described in Chapter 2, will proceed as follows:

First, we'll describe a circuit in VHDL that controls the operation of one bin. It will respond appropriately to a "get_drink" signal (i.e., by giving a drink when one is available), keep count of the number of cans left in the bin, and set an EMPTY signal when its bin becomes empty and is in need of resetting. We'll call this circuit "binctr". We will also add appropriate VHDL to make a "package" out of the circuit, which will allow us to use the circuit in higher-level designs.

Then, we'll write a structural description of a circuit that instantiates two binctrs and other logic as appropriate, to describe the larger design that we have in mind. We'll call the larger design "refill".

After that, we'll synthesize the binctr and refill VHDL descriptions into a JEDEC file, and simulate the behavior of the resulting design.

Figure 1-1 shows the sequence of operations used in the Tutorial exercises.

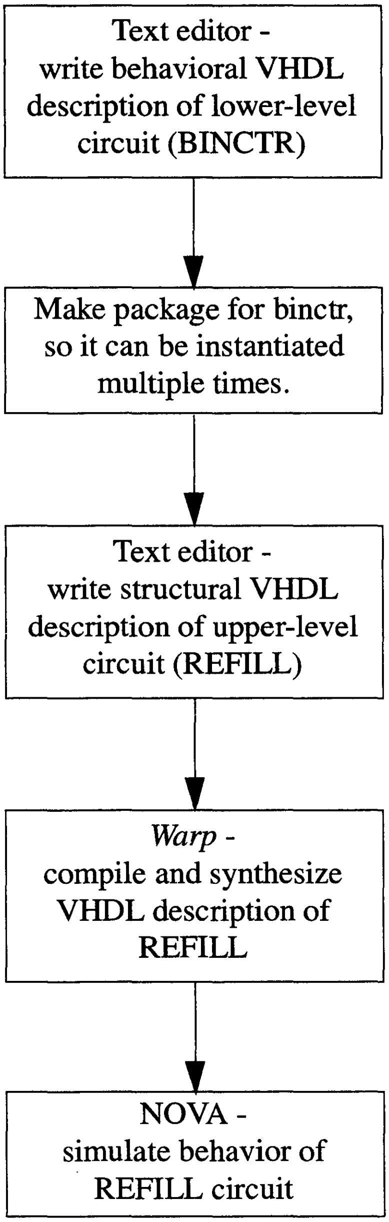

Text editor -
write behavioral VHDL description of lower-level circuit (BINCTR)
Make package for binctr, so it can be instantiated multiple times.
Text editor -
write structural VHDL description of upper-level circuit (REFILL)
Warp -
compile and synthesize VHDL description of REFILL
NOVA -
simulate behavior of REFILL circuit

Figure 1-1. Flow Diagram of Warp2 Tutorial, Chapter 2.

1.4. About the Tutorial

1.4.1. File/Directory Management

As you work through the tutorial, don't write anything into your *Warp* directory or its sub-directories. Instead, copy the information you need into a different directory and work from there.

By default, the *Warp2* installation procedure for IBM PC's and compatible computers installs *Warp2* software into a directory named c:\warp. On Sun workstations, the default location for the *Warp2* software is the directory pointed to by the CYPRESS_DIR environment variable.

When doing the Tutorial exercises (or any other time, for that matter), don't write anything into the *Warp* directory. Instead, create a separate directory to practice in (e.g., c:\w2tutor).

1.4.2. Differences in Operating Systems

Warp2 operates identically on both Unix and Windows systems. However, there are differences in the way you start *Warp2* and in the appearance of various objects on the display.

Starting *Warp2*

On Windows systems, you start *Warp2* by double-clicking on the Galaxy icon from the *Cypress* program group.

On Unix systems, you start *Warp2* by typing "galaxy<CR>" from within a shell window.

Differences in Display

Although dialog boxes and prompts may differ in appearance between the two platforms, their functionality is identical. Screen captures in this manual are taken from the Windows version of *Warp2*. Differences in the Unix version are identified when necessary. In most cases, any adjustments needed to go from Windows to Unix versions of *Warp2* displays are obvious.

Naming Restrictions

When working in the Windows environment, you MUST keep file names restricted to eight or fewer alpha-numeric characters (of which the first must be an alphabetic character), plus a file extension of up to three characters (e.g., ".exe", ".vhd", etc.). This is important to remember if you might transfer data between Windows and Unix implementations of *Warp2*.

NOTE

Keep in mind also that some Unix systems have trouble with spaces embedded in file names. That can be true for IBM PCs and compatible systems, too. Don't use embedded spaces in file names.

Chapter 2 The Tutorial

About This Chapter

Overview

This chapter takes you step-by-step through the tutorial example, using a low-level behavioral description and a high-level structural one. When you complete this chapter, you will know how to:

- write an entity declaration, architecture, and package declaration for a VHDL behavioral description of a simple circuit;
- write an entity declaration and architecture that instantiates the simple circuit in a structural description of a higher-level circuit;
- run *Warp* to synthesize the schematic's VHDL description;
- run Nova to simulate the behavior of the design.

2.1. Write the “binctr” VHDL description

The “binctr” VHDL description controls the behavior of a single bin in the drink machine. It acknowledges drink requests, dispenses drinks, keeps count of the number of drinks remaining, and sets an output signal when the bin becomes empty.

The “binctr” VHDL description will be written in three parts:

- the ENTITY declaration declares the name, direction, and data type of each port of the component;
- the ARCHITECTURE describes the behavior of the component;
- the PACKAGE declaration provides the information to *Warp* to allow binctr to be used as a component in a higher-level design.

The following pages discuss the contents of each of these sections of the VHDL description. You can use any ASCII text editor to type BINCTR.VHD.

NOTE

If you would rather not type the BINCTR.VHD file yourself, you can copy it from the Warp directory. The default location for the BINCTR.VHD file is c:\warp\examples\w2tutor\binctr.vhd on PC’s, and $CYPRESS_DIR/examples/w2tutor on UNIX workstations.

From either of these locations, copy binctr.vhd to your project directory. Then, read along for the next few pages, to help you understand the purpose of each section of a VHDL source file.

2.1. Write the “binctr” VHDL description

2.1.1. Write the Entity Declaration

The ENTITY declaration declares the name, direction, and data type of each port of the component.

Figure 2-1 lists the entity declaration for the binctr component.

The ENTITY declaration declares that entity “binctr” has five external interfaces, or “ports.” It has three input ports of type BIT, named reset, get_drink, and clk, respectively. It has two output ports, also of type BIT, named give_drink and empty, respectively.

```
entity binctr is port(
  reset, get_drink, clk : in bit;
  give_drink, empty: out bit);
end binctr;
```

Figure 2-1. ENTITY Declaration of binctr Component.

2.1.2. Write the Architecture

The ARCHITECTURE portion of a VHDL description describes the behavior of the component.

Figure 2-2 lists the architecture portion of the binctr component's VHDL description. The architecture appears after the entity declaration in the .VHD file.

The first line declares an architecture named "archbinctr" of entity "binctr".

The next two lines declare a constant and a signal, respectively.

- The constant, named "full", determines how many drinks are in a full bin.
- Signal "remaining", of type integer with a range from 0 to the value of constant full, keeps track of how many drinks are left in the bin.

The BEGIN that follows the signal declaration marks the start of the architecture body.

A process declaration follows, marked by the keyword PROCESS and an ensuing BEGIN.

```
architecture archbinctr of binctr is
  constant full:integer:=3; -- max of 3 drinks/bin
  signal remaining:integer range 0 to full;
begin
  proc_label:process begin
    wait until clk = '1';
    if reset = '1' then
      remaining <= full;
      empty<='0';
      give_drink<='0';
    elsif remaining=0 then
      empty <= '1';
      give_drink <= '0';
    elsif get_drink = '1' then
      remaining <= remaining - 1;
      give_drink <= '1';
    elsif get_drink = '0' then
      give_drink <= '0';
    else
      give_drink <= give_drink;
      end if;
    end process;
  end archbinctr;
```

Figure 2-2. ARCHITECTURE of binctr Component.

The line "WAIT UNTIL clk='1'" synchronizes all activity within the process with transitions of signal clk to '1'.

Subsequent signal activity is handled in the following order:

- If signal "reset" is '1', then signal "remaining" is set to full, signal "empty" is set to '0', and signal "give_drink" is set to '0'. (Notice that this means reset has priority over get_drink; if reset and get_drink are both true on the same clock cycle, the bin is reset, but no drink is given.)

- Otherwise, if signal “remaining” has value 0, then signal “empty” is set to ‘1’ and signal “give_drink” is set to ‘0’.
- Otherwise, if signal “get_drink” is ‘1’, then signal “remaining” is decremented by one and signal “give_drink” is set to ‘1’.
- Otherwise, if signal “get_drink” is ‘0’, the signal “give_drink” is set to ‘0’.
- Otherwise, give_drink is set to the current value of give_drink. Inclusion of this clause is not strictly necessary, but it’s good VHDL coding practice .

Several lines ending the IF statement, process, and architecture follow. Note that the END statement ending the architecture must be accompanied by the name of the architecture, which MUST match the name shown on the first line of the architecture.

2.1. Write the "binctr" VHDL description

2.1.3. Write the Package Declaration

The PACKAGE declaration provides the information to the *Warp* compiler to allow binctr to be used as a component in a higher-level design.

Figure 2-3 lists the package declaration for the binctr component.

NOTE

The package declaration must appear before the entity declaration or architecture in the .VHD file.

```
package binctr_pkg is
  component binctr
    port(reset, get_drink, clk : in bit;
         give_drink, empty: out bit);
  end component;
end binctr_pkg;
```

Figure 2-3. PACKAGE Declaration of binctr Component.

The first line in Figure 2-3 declares the name of the package. The name of the package must be distinct from the name of any component declared within that package. Using the convention "<*entity*>_pkg" works nicely.

The second line declares a component named "binctr". The component name that appears on this line must match the name of an accompanying entity.

The port statement declares the name, direction, and type of each port in the component. You can copy the port statement from the entity declaration for this purpose.

An “end component” and “end binctr_pkg” statement conclude the package declaration. Note that the package named in the END package statement must match that shown in the first line of the package delcaration.

At this point, save the file as c:\w2tutor\binctr.vhd.

2.2. Run Warp

Warp is the VHDL synthesis compiler for *Warp2*.

Warp takes VHDL descriptions as input. *Warp* has two functions in the *Warp2* system: (1) to verify that VHDL descriptions are syntactically correct, and (2) to produce JEDEC or QDIF files to program output devices.

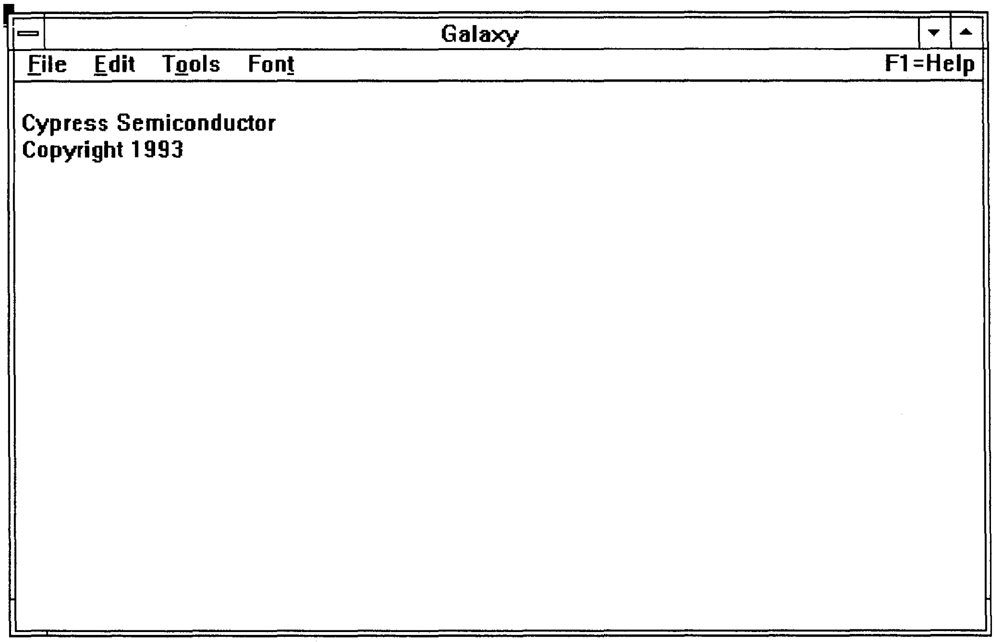
At this point in the Tutorial, we'll use *Warp* to verify that the BINCTR.VHD file is syntactically correct. This step isn't strictly necessary; we could simply go on and build the higher-level design for the "refill" circuit. But it's always a good idea to compile and simulate any VHDL description once you've completed it. That way, you can spot problems in your VHDL description when they are easiest to identify and correct. Later on, should you encounter problems with the larger circuit, you can at least be assured that you have taken care of any bugs at the lower levels of the hierarchy.

2.2. Run Warp

2.2.1. Start Galaxy

Galaxy is the user interface for the *Warp* VHDL synthesis compiler.

To start Galaxy, double-click on the Galaxy icon in the Cypress program group. (If you're using a Unix workstation, type "galaxy<CR>" at a shell prompt). The Galaxy window appears (Figure 2-4).



```
Cypress Semiconductor
Copyright 1993
```

Figure 2-4. The Galaxy window (#1).

2.2.2. Compile binctr.vhd

To bring up the *Warp* dialog box, select Tools/Run Warp Menu from the Galaxy menu bar.

The *Warp* dialog box appears (Figure 2-5).

This dialog box lists the VHDL files available in the current directory on the left side, and the VHDL files selected for compilation/synthesis on the right side.

To select the binctr.vhd file for compilation, select binctr.vhd from the list of files on the left-hand side, then click on the “Add” button.

Then, select “Compile Only” from the “Build” button group near the lower-left corner of the dialog box.

Then, click on “OK”. *Warp* runs, printing messages to keep you appraised of its progress.

The compilation process should run to completion, without any error messages.

NOTE

If you do get error messages, check to make sure that the various parts of the binctr.vhd file read EXACTLY as they are listed on the preceding pages. Better yet, copy the binctr.vhd file from the \w2tutor sub-directory of the *Warp* directory, then run *Warp* again.

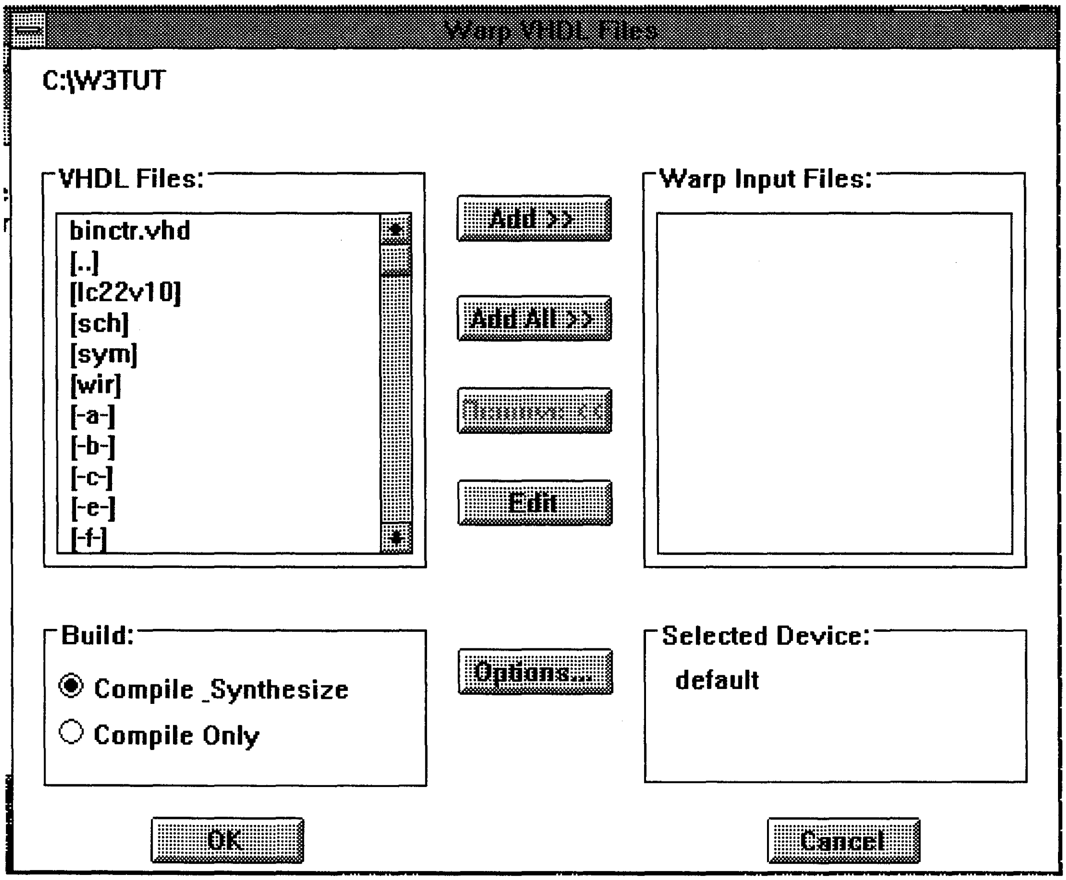


Figure 2-5. Warp dialog box (#1).

2.3. Generate a VHDL Description of the Higher-Level Circuit

Having defined the behavior of the lower level of the circuitry, we can describe the upper level structurally by instantiating various components and connecting them with appropriate internal signals.

Figure 2-6 shows file REFILL.VHD, a structural description of the highest level of our tutorial circuit.

The first five lines tell *Warp* to use various packages containing components and functions referred to in the design. Specifically, the CKPAD component is in the IOPKG package, the AND2 component is in the GATESPKG package, and the binctr component is in the BINCTR package. The CYPRESS and RTLPKG packages contain definitions that are used in almost all designs.

The entity declaration declares an entity named REFILL and defines the names, types, and direction of its seven input and output ports.

The architecture that follows starts by declaring three internal signals. These are used to connect the output of one component to the input of another.

For example, the EMPTY port of the binctr component labeled bin_1 connects to internal signal empty_1. Signal empty_1, in turn, connects to the A port of the AND gate named and_1.

Instantiation of two BINCTRs, one AND gate, and a CKPAD (clock pad) completes the structural description of this circuit.

```
use work.IOPKG.all;
use work.GATESPKG.all;
use work.BINCTR_PKG.all;
use work.cypress.all;
use work.rtlpkg.all;

entity REFILL is
    port(GIVE_PEPSI: INOUT bit;
         GIVE_COKE: INOUT bit;
         REFILL_BINS: INOUT bit;
         RESET: IN bit;
         CLK: IN bit;
         GET_COKE: IN bit;
         GET_PEPSI: IN bit);
end REFILL;

architecture archREFILL of REFILL is
    signal i_clk: bit;
    signal empty_1: bit;
    signal empty_2: bit;
begin
    bin_1: BINCTR
        port map(RESET => RESET,
                 GET_DRINK => GET_PEPSI,
                 CLK => i_clk,
                 GIVE_DRINK => GIVE_PEPSI,
                 EMPTY => empty_1);
    and_1: AND2
        port map(A => empty_1,
                 B => empty_2,
                 Q => REFILL_BINS);
    bin_2: BINCTR
        port map(RESET => RESET,
                 GET_DRINK => GET_COKE,
                 CLK => i_clk,
                 GIVE_DRINK => GIVE_COKE,
                 EMPTY => empty_2);
    clk_pad: CKPAD
        port map(P => CLK,
                 Q => i_clk);
end archREFILL;
```

Figure 2-6. REFILL.VHD file

2.4. Run Warp to compile and synthesize entire circuit

Warp is the VHDL synthesis compiler for *Warp2*.

Warp takes VHDL descriptions as input and produces JEDEC or QDIF files as output. JEDEC files are used to program PLD's. QDIF files are used as input to the SpDE tools, which produce output files used to program pASIC380 FPGA's.

The first time you ran *Warp*, earlier in the Tutorial, it was simply to verify that the BINCTR.VHD file was syntactically correct.

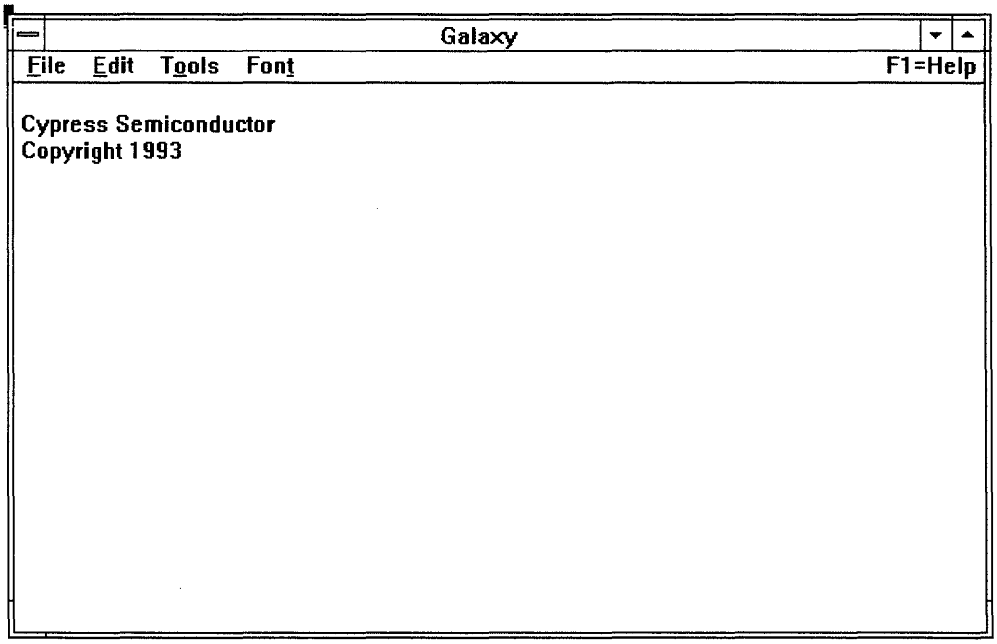
On the following pages, you'll run *Warp* to produce a JEDEC file for a specific target device (in this case, a C22V10).

2.4. Run Warp to compile and synthesize entire circuit

2.4.1. Start Galaxy

Galaxy is the user interface for the *Warp* VHDL synthesis compiler.

To start Galaxy, double-click on the Galaxy icon in the Cypress program group. (On Unix workstations, type "galaxy<CR>" at a shell prompt.) Close the ensuing "About" window. The Galaxy window appears (Figure 2-7).



Cypress Semiconductor
Copyright 1993

Figure 2-7. The Galaxy window (#2).

2.4. Run Warp to compile and synthesize entire circuit

2.4.2. Start *Warp*

To bring up the *Warp* dialog box, select Tools/Run Warp Menu from the Galaxy menu bar.

The *Warp* dialog box appears (Figure 2-8).

This dialog box lists the VHDL files available in the current directory on the left side, and the VHDL files selected for compilation/synthesis on the right side.

To select a VHDL file for compilation/synthesis, click on the name of the file on the left side, then click on the “Add” button.

To de-select a file for compilation/synthesis, click on the file’s name on the right side, then click on the “Remove” button.

The asterisk (*) alongside the name of the REFILL.VHD file indicates that *Warp2* has processed this file and identified it as the highest-level description of a multi-file VHDL hierarchy. (The asterisk is NOT part of the file name, however; it’s just a visual aid, to help you identify the highest-level files in projects.)

Now, let’s specify a target device...

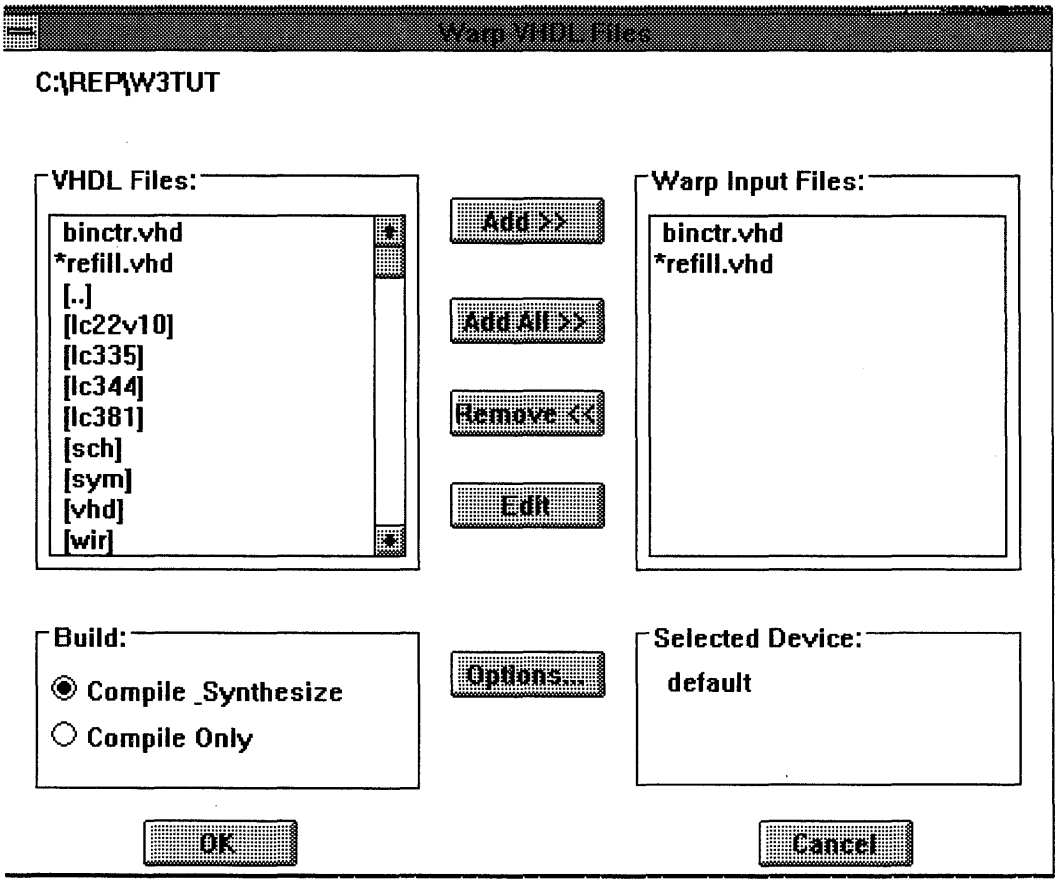


Figure 2-8. Warp dialog box (#2).

2.4. Run Warp to compile and synthesize entire circuit

2.4.3. Synthesize a VHDL Description

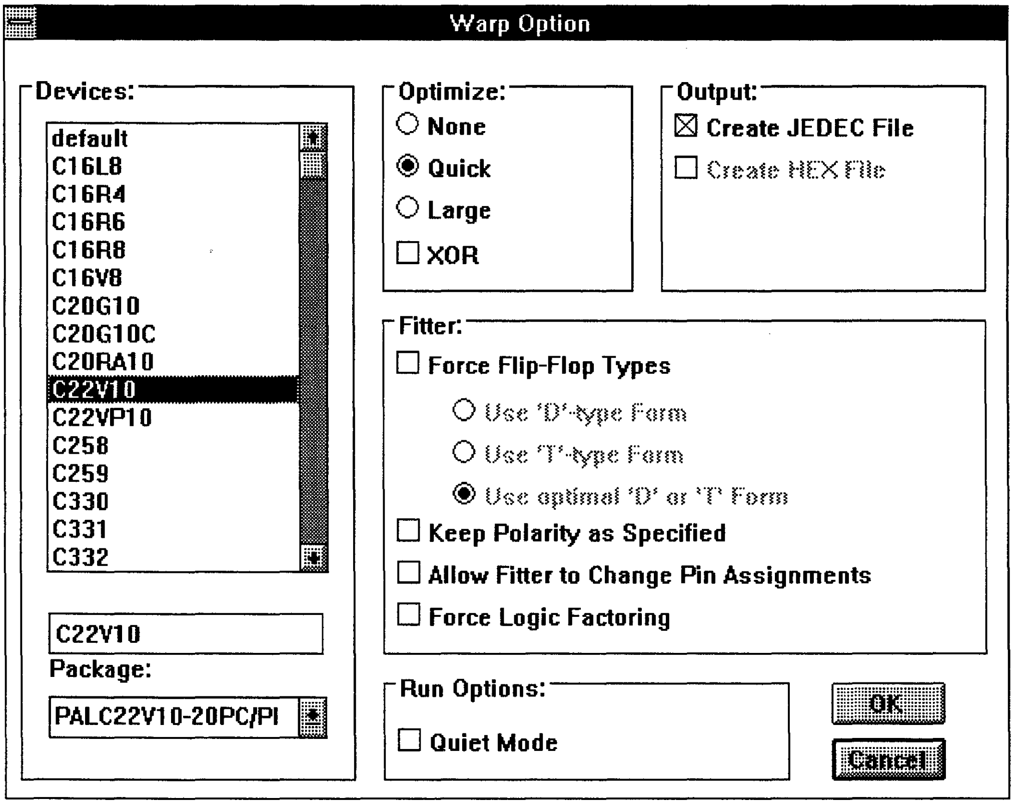
To specify a target device (along with other compilation/synthesis options), click on the Options button from the *Warp* dialog box.

The *Warp* options dialog box appears (Figure 2-9).

For this tutorial, you will synthesize the VHDL description in file REFILL.VHD to a C22V10 PLD.

To select the C22V10, scroll through the "Devices:" list and highlight "C22V10", then click "OK".

When the *Warp* dialog box re-appears, click on "OK". *Warp* runs, printing messages to keep you appraised of its progress.


Warp Option

Devices:

default
C16L8
C16R4
C16R6
C16R8
C16V8
C20G10
C20G10C
C20RA10
C22V10
C22VP10
C258
C259
C330
C331
C332

C22V10

Package: PALC22V10-20PC/PI

Optimize:
- ○ None
- ● Quick
- ○ Large
- ☐ XOR

Output:
- ☒ Create JEDEC File
- ☐ Create HEX File

Fitter:
- ☐ Force Flip-Flop Types
  - ○ Use 'D'-type Form
  - ○ Use 'T'-type Form
  - ● Use optimal 'D' or 'T' Form
- ☐ Keep Polarity as Specified
- ☐ Allow Fitter to Change Pin Assignments
- ☐ Force Logic Factoring

Run Options:
- ☐ Quiet Mode

Figure 2-9. Warp Options dialog box.

This operation generates two files of particular interest (among others):

- The first is named REFILL.JED. The .JED file can be used to program the 22V10.
- The second file is named REFILL.RPT. It contains pinout and timing information, along with lots of other information about the final synthesized design.

Exit the Galaxy window. (Choose "Exit" from the File menu.)

NOTE

If compilation errors occur, do the following:
1. make sure the text of your binctr.vhd file is entered exactly as shown earlier in this chapter; better yet, copy it from the \w2tutor sub-directory of the *Warp* directory.

2. run *Warp* again.

2.5. Run Nova to simulate the behavior of the design

Once the design is synthesized, it's a good idea to simulate its behavior to ensure that it functions as intended.

We'll use the Nova simulator to test the behavior of the design. In this tutorial, we'll perform the following steps:

- start Nova;
- open the REFILL.JED file;
- create a new view and populate it with the signals we're interested in (and only those signals);
- designate and edit a clock signal;
- set the values of the stimulus signals in the simulation;
- simulate the design;
- examine results to figure out what happened.

2.5. Run Nova to simulate the behavior of the design

2.5.1. Start Nova and Open the REFILL.JED File

To start Nova, double-click on the Nova icon in the Cypress program group. (From Unix workstations, type “nova<CR>” at a shell prompt.

The Nova screen appears, followed by the Nova “About” box. The “About” box goes away by itself in a few seconds. If you want to make it go away faster, click anywhere in the “About” box.

To open the REFILL.JED file, first choose the “Open...” item from the File menu. Find the appropriate directory in the ensuing Open File dialog box, select the REFILL.JED file, then click on “Open”. The results should look like Figure 2-10.

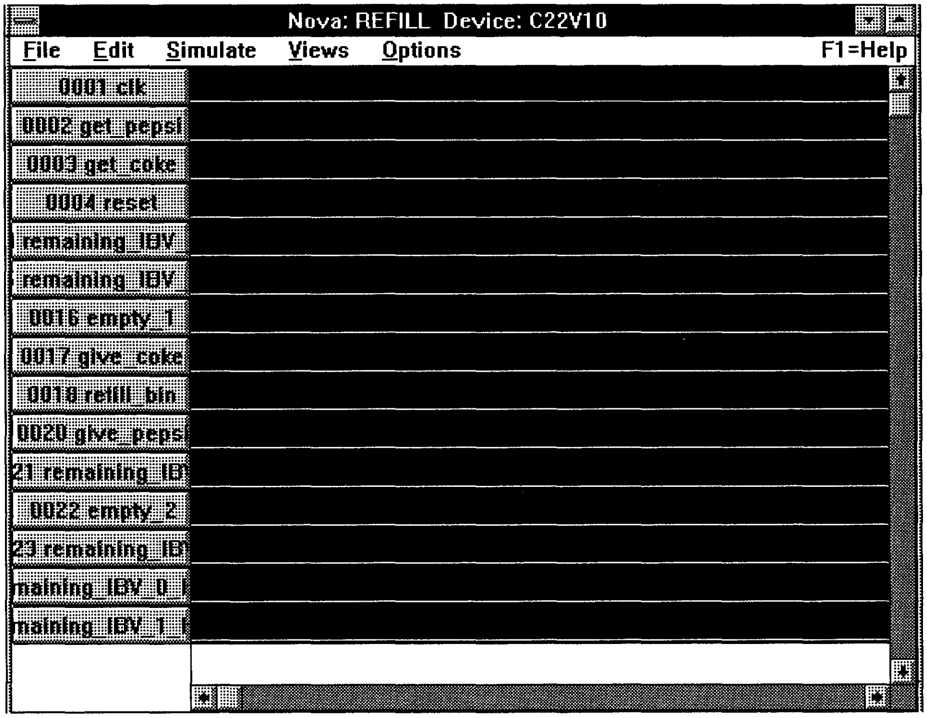


Figure 2-10. Nova Screen When REFILL.JED File Is First Opened.

2.5. Run Nova to simulate the behavior of the design

2.5.2. Create a View

The View is the collection of signals available for viewing on the Nova screen. To make it easier to see what's going on in our simulation, we'll create our own view, filling it with just the signals we're interested in.

1. Choose the "Edit Views" item from the Views menu.
2. Click on the "New View" button.
3. In the ensuing dialog box, give the new view any unique name; "tutview" will do nicely.
4. Click on the signal name in the "Full View" portion of the window, then on the "Add" button, for each of the following signals:

   clk;
   reset;
   get_pepsi;
   get_coke;
   give_pepsi;
   give_coke;
   refill_bin.

5. Click on "OK". The new view should appear on the Nova display screen, and should look like Figure 2-11.

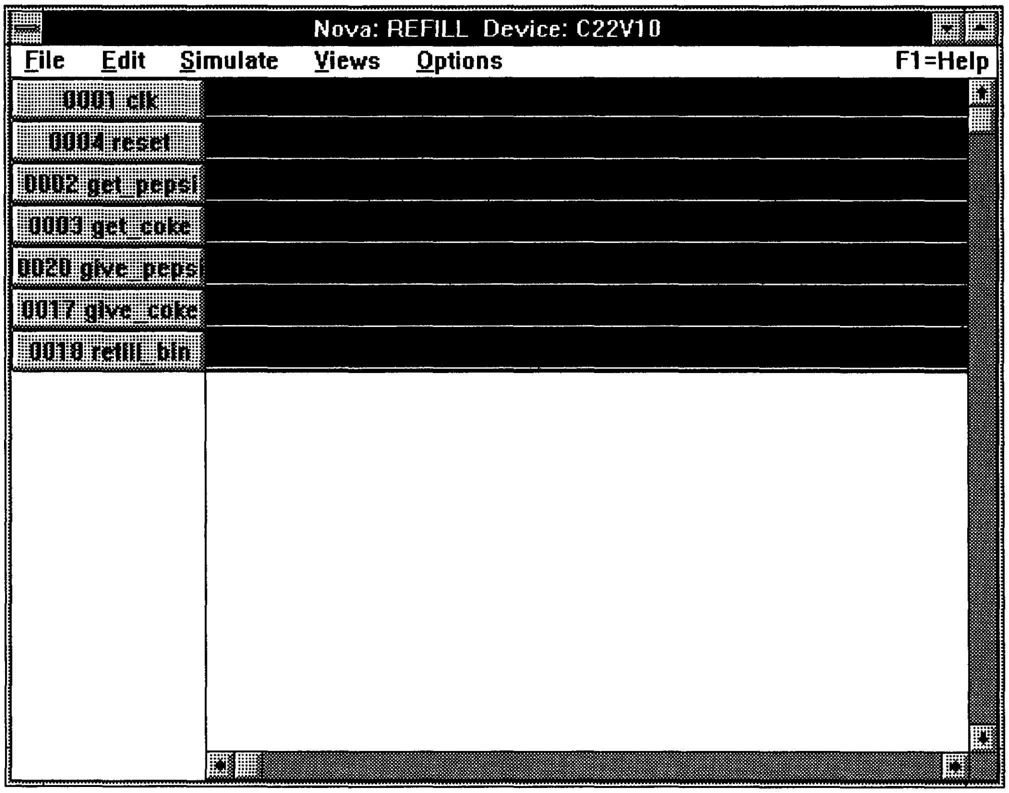


Figure 2-11. Nova Display Screen With New View Defined.

2.5.3. Designate a Clock Signal

We'll use the Edit/Clock menu item to give signal "clk" a series of alternating highs and lows.

1. Select signal "clk" by clicking on its name in the Nova display. The signal trace should turn blue when the entire signal is selected in this manner. (The trace will become a dashed line on monochrome monitors.)
2. Select the "Clock" item from the Edit menu.
3. When the Edit/Clock dialog box appears, click on "OK" to accept the default clock values.

Signal "clk" appears as a series of equally-spaced, alternating highs and lows (Figure 2-12).

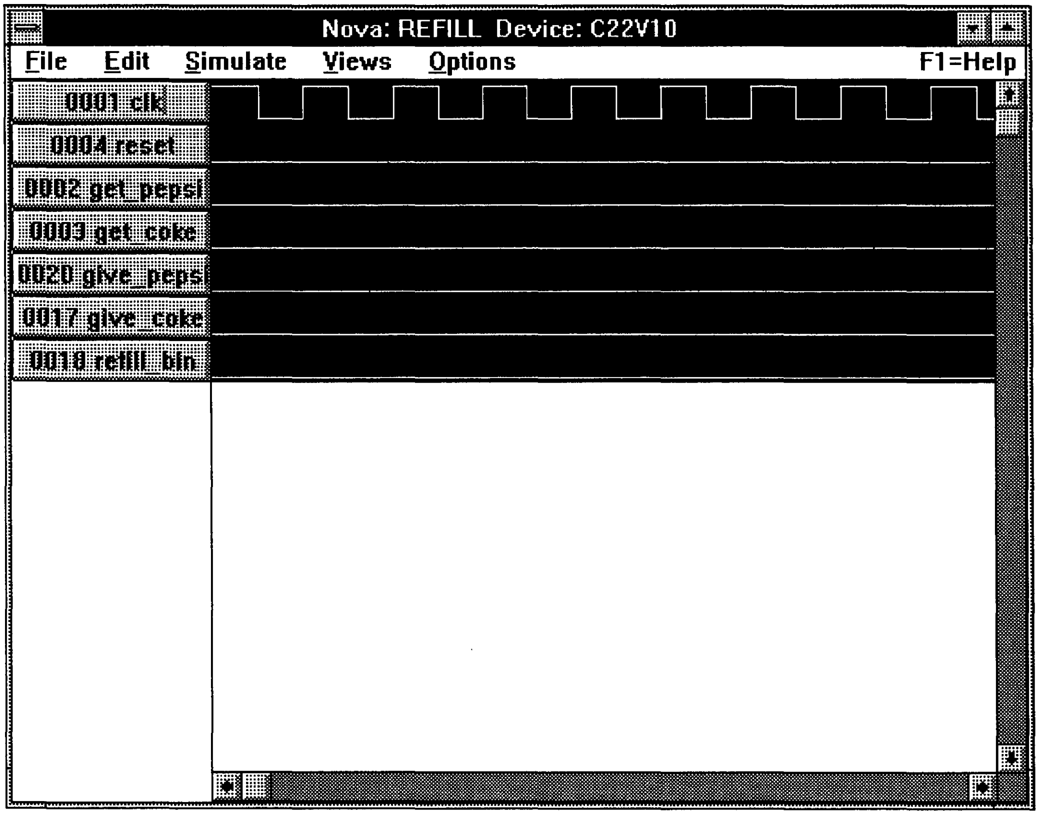


Figure 2-12. Clock Signal Designated.

2.5.4. Set the Values of Stimulus Signals

Besides the clock signal, we will set the values of the following input signals for our simulation: reset, get_pepsi, and get_coke.

1. Set "reset" to high for one rising clock edge. To do so:
   a. position the cursor on the "reset" trace, just to the left of a rising clock edge;
   b. click and hold the left mouse button;
   c. drag the cursor along the trace to the right of the rising clock edge, then release the mouse button.
   d. type a '1'.

<> Repeat this procedure to set the values of other signals.

2. Set "get_pepsi" to high for four non-consecutive rising clock edges.
3. Set "get_coke" to high for four non-consecutive rising clock edges.
4. Set "reset" to high for one rising clock edge after the last "get_coke" request.
5. Set "get_pepsi" and "get_coke" to high for one rising clock edge, respectively, after the second reset.

After you have set the values of the input signals, you may wish to change the screen resolution, in order to fit all activity in the waveforms on one screen.

To do so, select the "Resolution" item from the Options menu, then set the resolution to, say, two pixels per simulation tic. The result, when complete, should look like Figure 2-13.

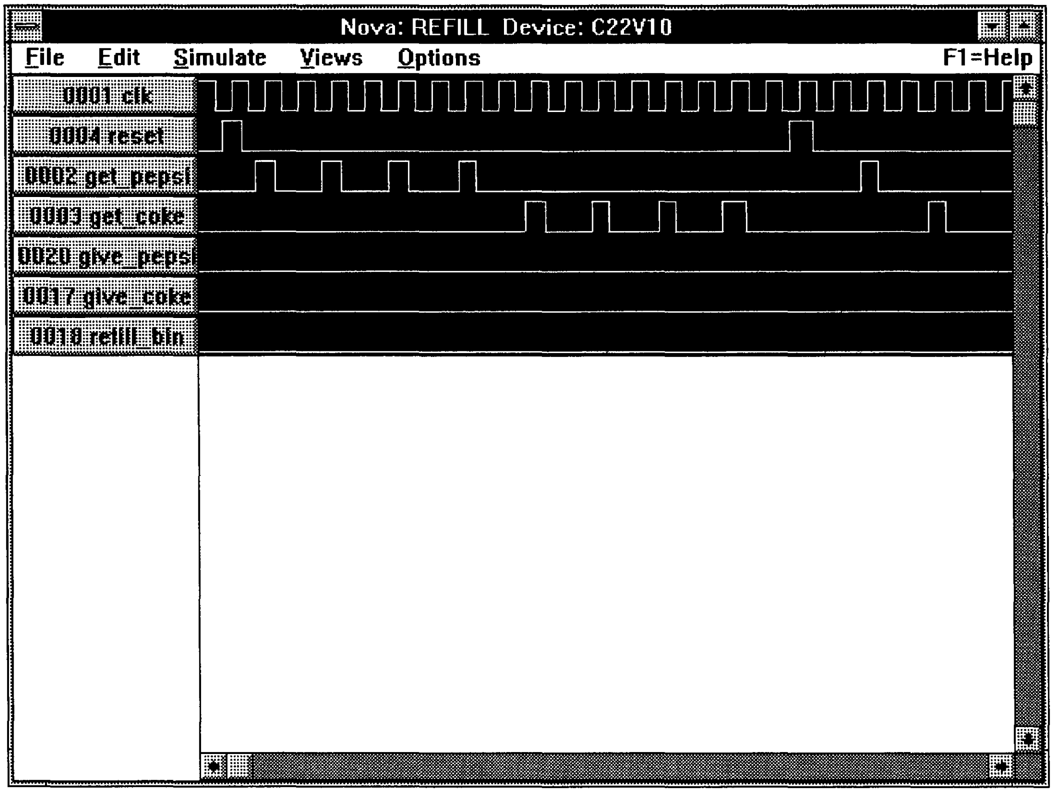


Figure 2-13. Input Signals Set.

2.5. Run Nova to simulate the behavior of the design

2.5.5. Simulate and Examine Results

To simulate the design, select the "Execute" item from the Simulate menu.

The results should look similar to Figure 2-14.

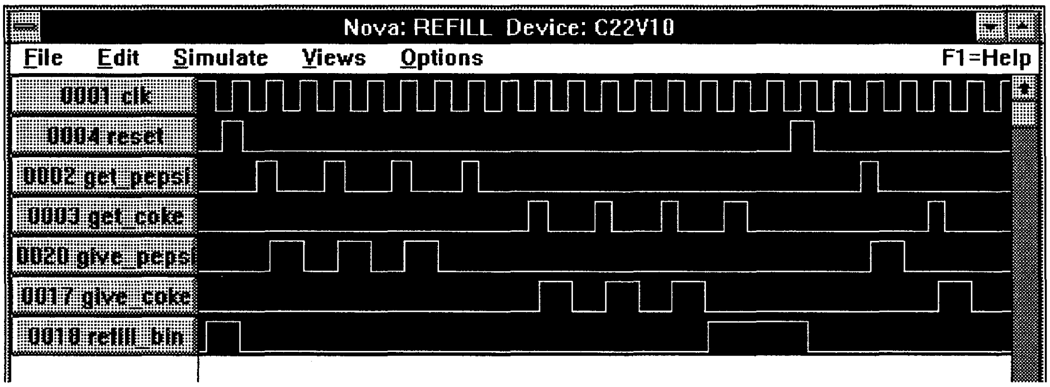


Figure 2-14. Final Simulation Results.

The simulation starts with the drink machine empty. (Notice the state of the refill_bin signal at the start of the simulation.)

When the reset signal goes high briefly near the start of the simulation, the refill_bin signal is set to 0. The drink machine is now ready to dispense drinks.

The drink machine dispenses a Pepsi in response to the first three requests for a Pepsi. (Note the relationship between the pulses in the get_pepsi and give_pepsi signals.) After the fourth request for a Pepsi, however, the machine does not dispense a drink; the Pepsi bin is empty.

Similary, the drink machine dispenses a Coke in response to the first three requests for a Coke. After the fourth request for a Coke, the machine does not dispense one; the Coke bin is empty.

With both bins empty, the refill_bin signal goes high with the last drink dispensed. It stays high until the reset signal goes high again, telling the machine that the bins have been replenished. The next two requests, for a Pepsi and a Coke respectively, are honored.

2.6. Conclusion

Now that we have verified the correct behavior of the design, the REFILL.JED file can be transferred to a device programmer.

JEDEC files (.JED extension) generated by *Warp* can be transferred to device programmers for PLDs and PROMs. See the manual that came with your device programmer for information on transferring and using this file to program devices.

This concludes the *Warp2* Tutorial. For more VHDL examples and discussion, see Chapter 4 of the *Warp Synthesis Reference Manual*.

Index

Warp2™

VHDL Development System

Nova User's Manual

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
(408)943-2600

Cypress Software License Agreement

1. LICENSE. Cypress Semiconductor Corporation ("Cypress") hereby grants you, as a Customer and Licensee, a single-user, non-exclusive license to use the enclosed Cypress software program ("Program") on a single CPU at any given point in time. Cypress authorizes you to make archival copies of the software for the sole purpose of backing up your software and protecting your investment from loss.

2. TERM AND TERMINATION. This agreement is effective from the date the diskettes are received until this agreement is terminated. The unauthorized reproduction or use of the Program and/or documentation will immediately terminate this Agreement without notice. Upon termination you are to destroy both the Program and the documentation.

3. COPYRIGHT AND PROPRIETARY RIGHTS. The Program and documentation are protected by both United States Copyright Law and International Treaty provisions. This means that you must treat the documentation and Program just like a book, with the exception of making archival copies for the sole purpose of protecting your investment from loss. The Program may be used by any number of people, and may be moved from one computer to another, so long as there is **No Possibility** of its being used by two people at the same time.

4. DISCLAIMER. THIS PROGRAM AND DOCUMENTATION ARE LICENSED "AS-IS," WITHOUT WARRANTY AS TO PERFORMANCE. CYPRESS EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTY OF MERCHANTABILITY OR

FITNESS OF THIS PROGRAM FOR A PARTICULAR PURPOSE.

5. LIMITED WARRANTY. The diskette on which this Program is recorded is guaranteed for 90 days from date of purchase. If a defect occurs within 90 days, contact the representative at the place of purchase to arrange for a replacement.

6. LIMITATION OF REMEDIES AND LIABILITY. IN NO EVENT SHALL CYPRESS BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM PROGRAM USE, EVEN IF CYPRESS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. CYPRESS'S EXCLUSIVE LIABILITY AND YOUR EXCLUSIVE REMEDY WILL BE IN THE REPLACEMENT OF ANY DEFECTIVE DISKETTE AS PROVIDED ABOVE. IN NO EVENT SHALL CYPRESS'S LIABILITY HEREUNDER EXCEED THE PURCHASE PRICE OF THE SOFTWARE.

7. ENTIRE AGREEMENT. This agreement constitutes the sole and complete Agreement between Cypress and the Customer for use of the Program and documentation. Changes to this Agreement may be made only by written mutual consent.

8. GOVERNING LAW. This Agreement shall be governed by the laws of the State of California. Should you have any question concerning this agreement, please contact:

   Cypress Semiconductor Corporation
   Attn: Legal Counsel
   3901 N. First Street
   San Jose, CA 95134-1599

   408-943-2600

Table of Contents

Chapter 1

Introduction

1.1. Introduction

Nova is Cypress Semiconductor Corporation's name for its behavioral simulator.

The Nova user interface gives you an easy way to:

- specify JEDEC files to simulate;
- read or write stimulus files;
- convert files from .JED to ViewSim format;
- edit input waveform traces;
- simulate the behavior of a design;
- alternate between various views (i.e., collections of signals), and specify signals to include in each view;
- specify the length and resolution of a simulation;
- specify segments, where you can re-apply and edit initial conditions, in order to compare results of differing initial conditions side-by-side;
- and lots of other useful capabilities.

This manual tells you how to use Nova to simulate designs. It assumes that you are already familiar with common user interface operations for your computer, such as the use of scroll bars, menu buttons, opening and closing windows, etc.

Chapter
2

Using Nova

2.1. Starting Nova

On Sun workstations, typing “nova” on the command line brings up the Nova window. On PC’s and compatibles, double-clicking on the Nova icon in the Cypress directory brings up the Nova window.

By default, Nova comes up ready to run on a color screen.

To start Nova on a monochrome Sun workstation, type “nova -m” on the command line.

To set Nova to come up in monochrome mode when running Windows on an IBM PC or compatible computer:

1. select the Nova icon from the Cypress window;
2. select “Properties” from the File menu;
3. edit the “Command Line” entry to include the -m option;
4. click OK.

2.2. The Nova Window

The Nova window (Figure 2-1) consists of a menu bar with several items across the top; a column of buttons along the left side, listing pin and node numbers and signal names; an area for displaying traces, and scroll bars across the bottom and right sides.

Menu Bar

The menu items are File, Edit, Simulate, Views, and Options. Under each of these items are menus for selecting related actions. The menus are ordered so that the most common operation is at the top. The contents of each menu are described in greater detail later in this manual.

Only two menu items, Open and Exit, are enabled in the File menu when you first enter Nova. When you open a ".jed" file or ".cyp" file. the other menu items will be enabled.

Node Numbers, Signal Names

The left-hand side of the Nova window consists of a column of buttons, displaying pin and node numbers and their associated signal names. A "node" is an area of a circuit containing one or more points whose locations you may wish to trace. (For information about different values within a node, refer to Section 2.4.4., "Nodes".)

To change the width of the buttons where signal names are displayed, use the Signal Name Size item in the Options menu.

Trace Area

The trace area displays the values of the nodes/signals listed in the left-hand column.

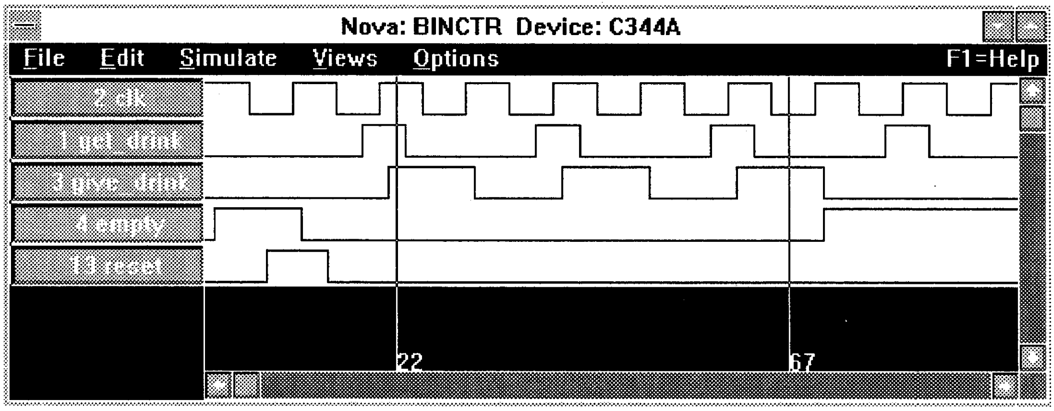


Figure 2-1. Main Nova Window.

You can display up to two measuring cursors, which allow you to see precisely the value(s) of several signals at a single time. To display the first cursor, click at the bottom of the trace window. To display a second cursor, click at the bottom of the trace window while pressing and holding the Shift key.

To change the position of either cursor, click and hold on the cursor at the bottom of the trace window, then drag the cursor to its new position. The cursor's horizontal position in simulation tics is displayed next to each cursor.

Note that a simulation tic does not represent any set amount of real-time delay. Instead, a simulation tic is simply a unit delay of simulation time.

2.3. The File Menu

The File Menu contains items related to opening JEDEC files for simulation, reading and writing stimulus files, and saving output files in various formats.

The File menu (Figure 2-2) in the Nova dialog box contains the following items:

- **Open...**
- **Write Sim (*.sim)**
- **Write Trace (*.psd)**
- **Read Stimulus File**
- **Write JEDEC Vectors**
- **Write JEDEC File (*.jed)**
- **Disassemble to ViewSim Format (*.vhd)**
- **Exit**
- **About...**

The operations of each of these menu items are discussed in greater detail on the next few pages.

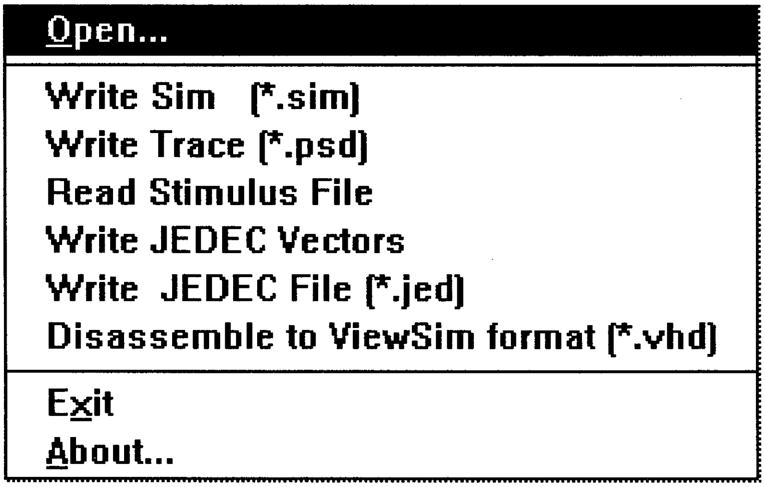


Figure 2-2. Nova File Menu.

2.3. The File Menu

2.3.1. Opening Files

The **Open...** item in the Files menu selects which .JED or .CYP file to open, and tells Nova what device is targeted in simulation.

Selecting **Open...** brings up the Open Files dialog box (Figure 2-3). The "File Name" line specifies the names of files to view in the Files window, or to open. By default, this line reads "*.JED".

To open a file, you can select a file from the list shown in the Files window, or type the name of the file on the "File name" line. Selecting a .JED file and clicking on Open closes the dialog box and displays traces. (If a stimulus file of the form *filename*.sim or *filename*.stm exists, it is also read automatically.) Clicking on Cancel closes the dialog box without opening a file.

The Select Device dialog box (Figure 2-4) comes up when you click on Open in the Open Files dialog box, and the file to be opened is a .JED file not created by *Warp2*. The Select Device dialog box maps a JEDEC file to a device.

Selecting a device with the wrong number of fuses brings up a message box stating: "Wrong device type for this jedec - QF doesn't match." This indicates that the number of fuses in the selected device don't match the number in the JEDEC file.

NOTE

If Nova says that it can't find file DEVICES.DAT, check to make sure your CYPRESS_DIR environment variable is set correctly. Nova uses this file to find the proper pin names and numbers for each target device and package.

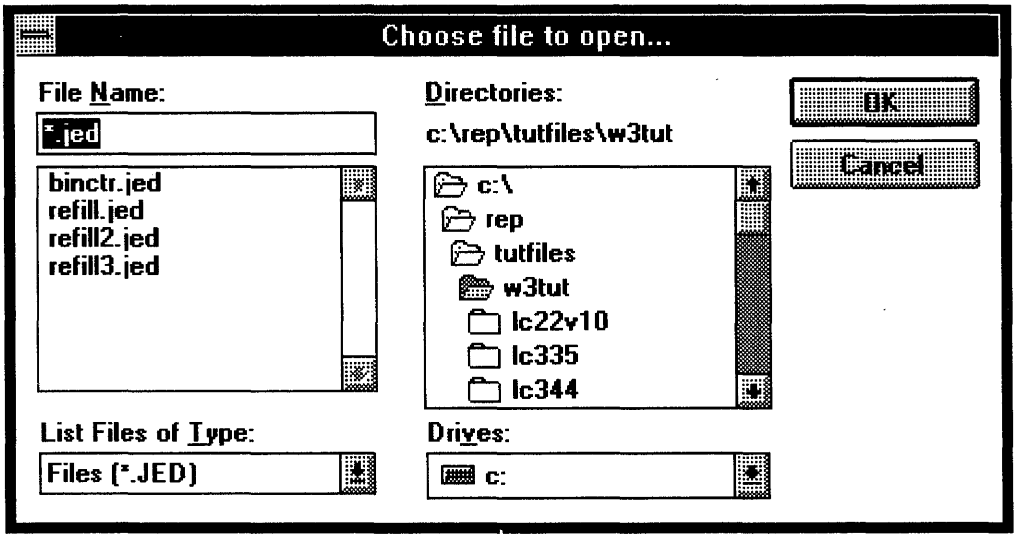


Figure 2-3. Open Files Dialog Box.

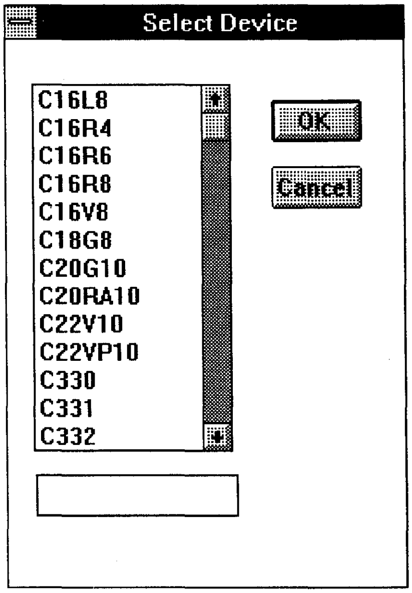


Figure 2-4. Select Device Dialog Box.

2.3.2. Reading and Writing Stimulus Files

Write Sim and **Write Trace** save simulation data. **Read Stimulus File** reads data stored by a previous Write Sim operation.

Write Sim saves the current simulation data to *filename*.sim, where *filename* is the prefix of the file you are simulating. If a ".sim" file already exists with this filename, the new simulation data overwrites the old. The .sim file (see Figure 2-5) can be re-read by **Read Stimulus File**.

Write Trace saves the trace information to *filename*.psd, where *filename* is the prefix of the file you are simulating. The .psd file (see Figure 2-6) provides a column-oriented, human-readable record of trace values during the simulation. Bus values are not written to the file.

Read Stimulus File reads simulation data from a .sim file. Because reading in the simulation file may change some of the settings the user has set for the current simulation, a message box is displayed, asking if the stimulus file should be read in. A "Yes" reply reads in the .sim file. A "No" reply returns you to the main Nova window. The *filename*.sim file is automatically read when the *filename*.jed file is opened.

```
1
clock_pin1
F83E0F83E0F83E0F83E0F83E0F83E0F83E0F83E0F83E0F83E0F83E0F83E0F83E
0000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
2
pin2
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
3
nickel_pin3
00FF800000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
```

Figure 2-5. Portion of .sim File.

```
 0: 1 0 0 0 L L L
 1: 1 0 0 0 L L L
 2: 1 0 0 0 L L L
 3: 1 0 0 0 L L L
 4: 1 0 0 0 L L L
 5: 0 0 0 0 L L L
 6: 0 0 0 0 L L L
 7: 0 0 0 0 L L L
 8: 0 1 0 0 L L L
 9: 0 1 0 0 L L L
10: 1 1 0 0 L L L
11: 1 1 0 0 L L L
12: 1 1 0 0 L L L
13: 1 1 0 0 L L L
14: 1 1 0 0 L L L
15: 0 1 0 0 L L L
16: 0 1 0 0 L L L
17: 0 0 1 0 L L L
18: 0 0 1 0 L L L
19: 0 0 1 0 L L L
20: 1 0 1 0 L L L
21: 1 0 1 0 L L L
22: 1 0 1 0 L L L
23: 1 0 1 0 L L L
24: 1 0 1 0 L L L
25: 0 0 1 0 L L L
```

Figure 2-6. Portion of .psd File.

2.3.3. Writing JEDEC Vectors

Write JEDEC Vectors appends vector information to the JEDEC file. The vectors can be used to test parts after they are programmed.

Write JEDEC Vectors appends vector information to the JEDEC file that you are simulating. If the JEDEC file already contains vector information, the new vector information overwrites the old.

2.3.4. Converting Between File Formats

The File menu includes items that allow you to convert vector information into different file formats, depending what you want to do with it.

Figure 2-7 shows the various file types that can be input to or output from *Warp*, Nova, and a device programmer.

Write JEDEC File (*.jed) writes out a JEDEC file from the data available to the simulator. You would typically use this item if you have input a .CYP file and want a JEDEC file to use on a device programmer. The dialog box includes options to include an instruction in the JEDEC file to blow the security fuse when the device is programmed, and to write the JEDEC file using a compressed, “K-field” hexadecimal representation. (This is the default representation for the C340 family; it reduces the JEDEC file size by nearly a factor of four.)

Disassemble to ViewSim format (*.vhd) writes a ViewSim source file (with .VHD extension) from the data available to the simulator. The resulting file can be revised with a text editor (to hand-optimize the design, for example) and re-input to *Warp*.

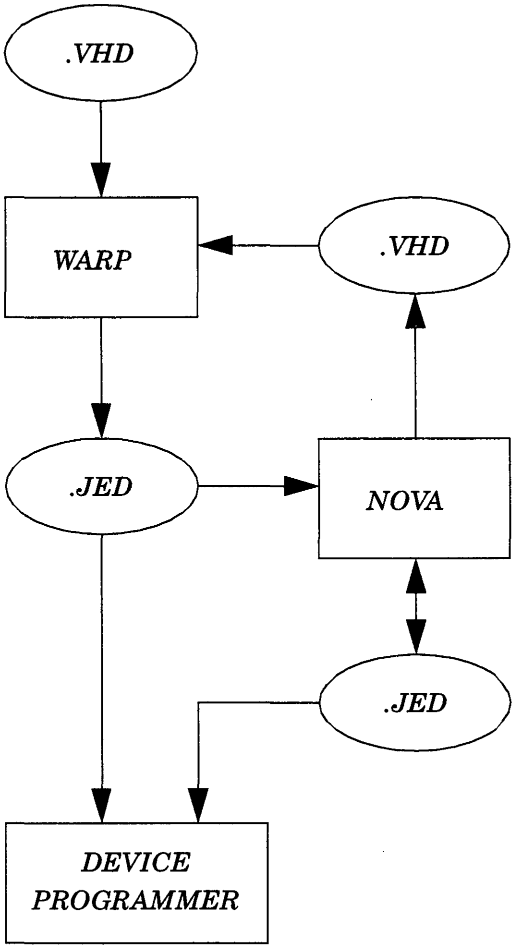


Figure 2-7. Possible Data Paths and File Formats.

2.3.5. About and Exit

The File menu's About item displays some basic information about the Nova simulator. The Exit item exits the simulator.

Besides displaying version information about the Nova simulator, the About dialog box also includes a Help button. Clicking on Help brings up help about Nova.

2.4. The Edit Menu

Use the items in the Edit Menu to modify trace information displayed on the screen. With the Edit menu, you can set the selected range of a trace, create and delete view nodes, create, delete and edit buses, and change the bus radix.

Items in the Edit Menu (Figure 2-8) include:

- **High, Low:** sets the selected trace or portion of a trace to 1 or 0, respectively.
- **Clock:** sets up repetitive pulses.
- **Pulse:** sets up a single pulse.
- **Node Defaults:** specifies the default source for the displayed value of a node.
- **Create View Node:** creates a new trace, selects the point within a node at which the displayed value is measured.
- **Delete View Node:** deletes traces from the simulation.
- **Create Bus:** groups traces for display as a single entity called a bus; used when it is more convenient to think of groups of signals as a single value. Bus values are only displayed when a measuring cursor is present.
- **Delete Bus:** un-defines a previously defined bus.
- **Edit Bus:** adds or removes signals from a bus.
- **Bus Radix:** specifies radix used to display a bus value.

These items are described in greater detail on the following pages.

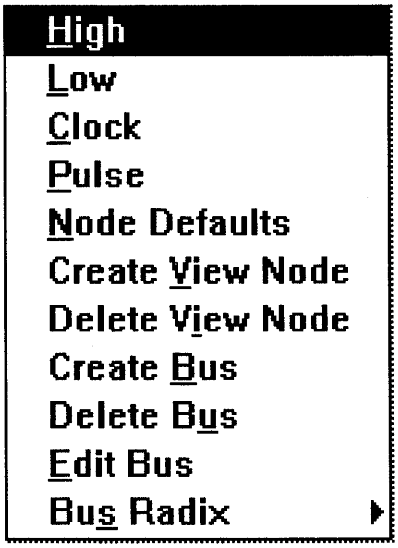


Figure 2-8. Nova Edit Menu.

2.4. The Edit Menu

2.4.1. Setting Signals High or Low

With the Nova user interface, you can easily set the value of all or a selected portion of an input signal high or low.

To set an entire input signal high or low:

- click on the button containing the name of the signal in the Nova window to select it. On color monitors, the button changes color and the trace turns blue when selected. On monochrome monitors, the button goes to inverse video and the trace changes to a dotted line when selected.
- select High or Low from the Edit menu as desired, or type '1' or '0'.

To set a portion of an input signal high or low (see Figure 2-9):

- de-select the signal;
- click and hold the mouse button on the trace at the point where you want the left edge of the selected area to be;
- drag the mouse to the point where you want the right edge of the selected area to be;
- then, select High or Low from the Edit menu, as appropriate, or type '1' or '0'.

Both the left and middle buttons of a 3-button mouse perform the same action when clicked to position an edge.

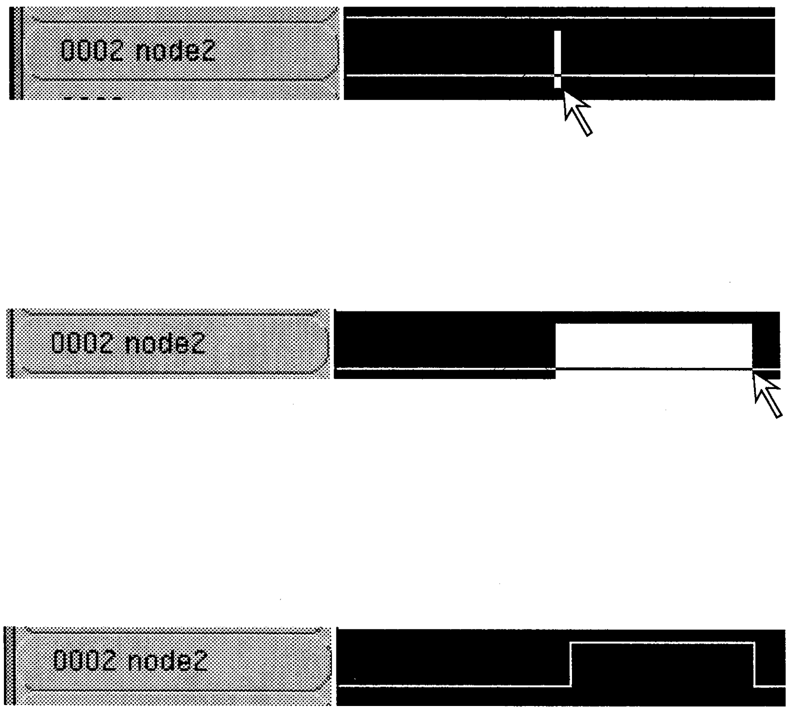


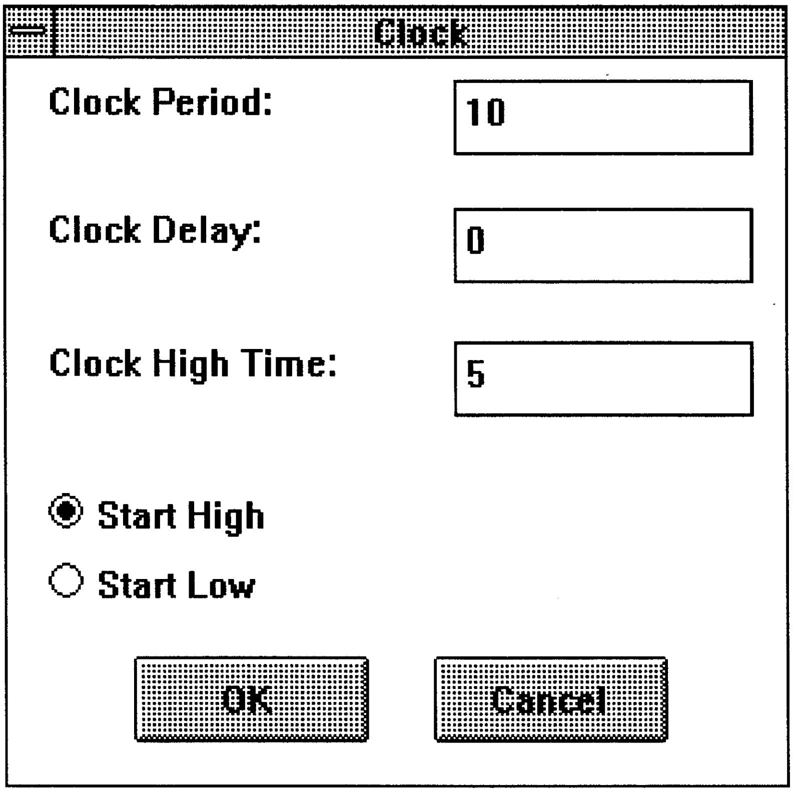
Figure 2-9. Setting a Portion of a Signal High or Low. Top: press and hold the mouse button at the left edge of the selected area. Middle: drag to the right edge of the selected area, and release the mouse button. Bottom: select High or Low from the Edit menu, or type '1' or '0' from the keyboard.

2.4.2. Setting Up Clock Signals (Repetitive Pulses)

The Clock item under the File menu lets you set up repetitive pulses on a selected signal or portion thereof.

To set up a repetitive pulse or clock signal, select a signal or a portion of a signal, then select the Clock item under the File menu. This brings up the Clock dialog box (Figure 2-10), which lets you fill in various parameters about the repetitive pulse you wish to set up:

- **Clock Period** specifies the period of repetition for the pulse in simulator "tics".
- **Clock Delay** specifies the number of simulator tics to wait (beginning with the left edge of the selected area) before starting the repetition. The default is 0 tics.
- **Clock High Time** specifies the amount of time that the selected signal should be set to '1' during each repetition. The default is 5.
- **Start High** and **Start Low** specify whether each repetition starts with the signal set to '0' or '1';
- **OK** sets up the repetitive pulse.
- **Cancel** closes the Clock dialog box without affecting the trace.



Clock Period: 10

Clock Delay: 0

Clock High Time: 5

[✓ Start High]

Start Low

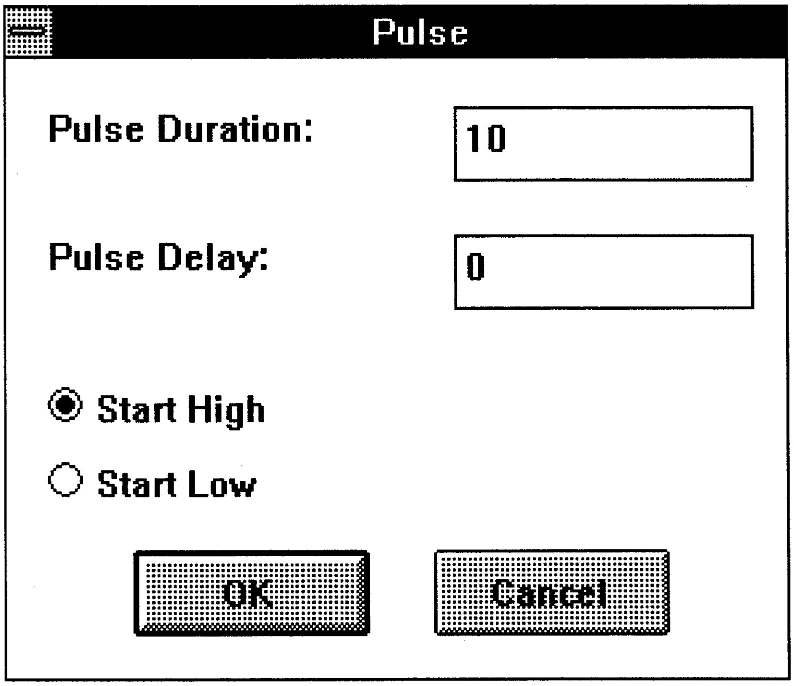
Figure 2-10. Clock Dialog Box.

2.4.3. Setting Up Non-Repetitive Pulses

The Pulse item under the File menu lets you set up single pulses on a selected signal.

To set up a single pulse on a signal, select a signal or a portion of a signal, then select the Pulse item under the File menu. This brings up the Pulse dialog box (Figure 2-11), which lets you fill in various parameters about the pulse you wish to set up:

- **Pulse Duration** specifies the length of the pulse, in simulator "tics".
- **Pulse Delay** specifies the number of simulator tics to wait (starting from the start of the simulation) before applying the pulse. The default is 0.
- **Start High** and **Start Low** specify whether the pulse sets the signal to '0' or '1';
- **OK** sets up the pulse.
- **Cancel** closes the Pulse dialog box without affecting the trace.


Pulse

Pulse Duration: 10

Pulse Delay: 0

● Start High

○ Start Low

Figure 2-11. Pulse Dialog Box.

2.4.4. Nodes

A node is an area of a circuit containing one or more points at which you may wish to trace a signal. Nova lets you specify the exact point or points within a node at which to trace signal values. It also gives you facilities for setting the default value of a node and for forcing one or more positions in a node to known values.

To Nova, a node is:

1. any input to an array;
2. any output from an array;
3. any pin on the device;
4. any other electrical position that needs to be modeled, but doesn't meet the first three criteria.

Figure 2-12 diagrams a simple PLD to illustrate what constitutes a node. The pin to the left of the array meets criteria 1 and 3; the macrocell to the right of the array meets criterion 2; and the macrocell drawn below the array meets criterion 4.

For each node, you can:

- create a view node, i.e., specify one or more positions within a node from which to trace values;
- specify the means by which a node is assigned its value;
- force any position in a node to a known value. This is often useful for multi-segment simulations.

Each of these capabilities is discussed in greater detail on the following pages.

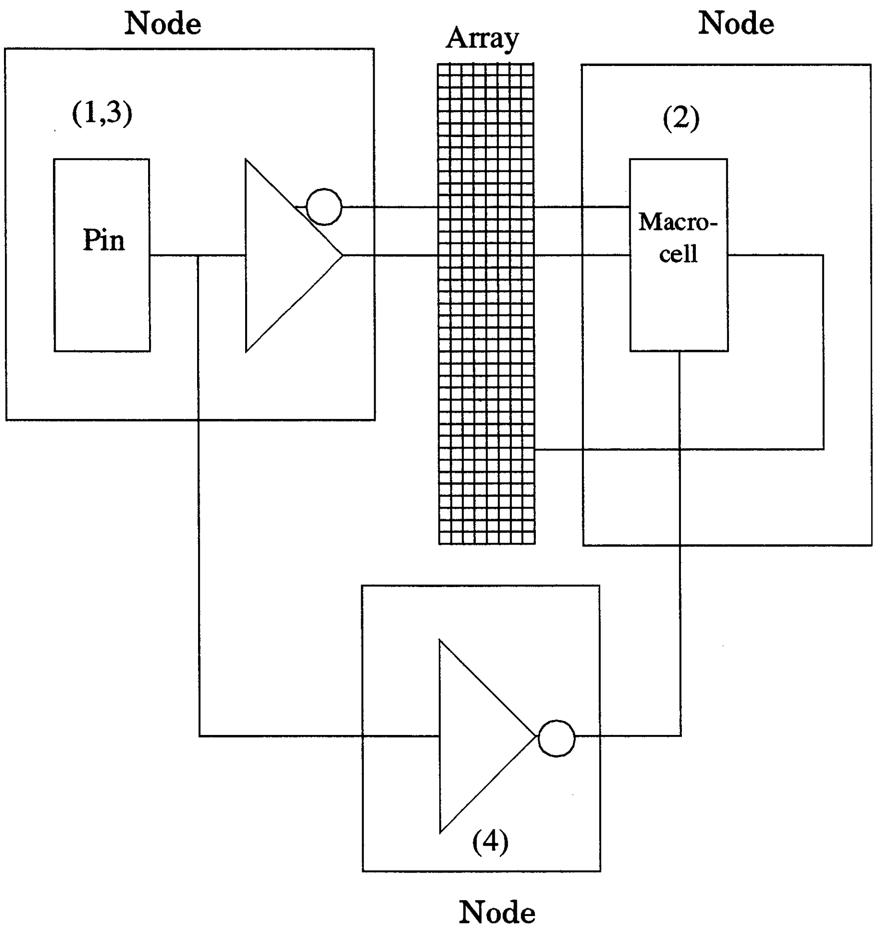


Figure 2-12. Logic Diagram of a Simple PLD, Showing Nodes.

2.4. The Edit Menu

2.4.4. Nodes

2.4.4.1. Selecting Node Points to View

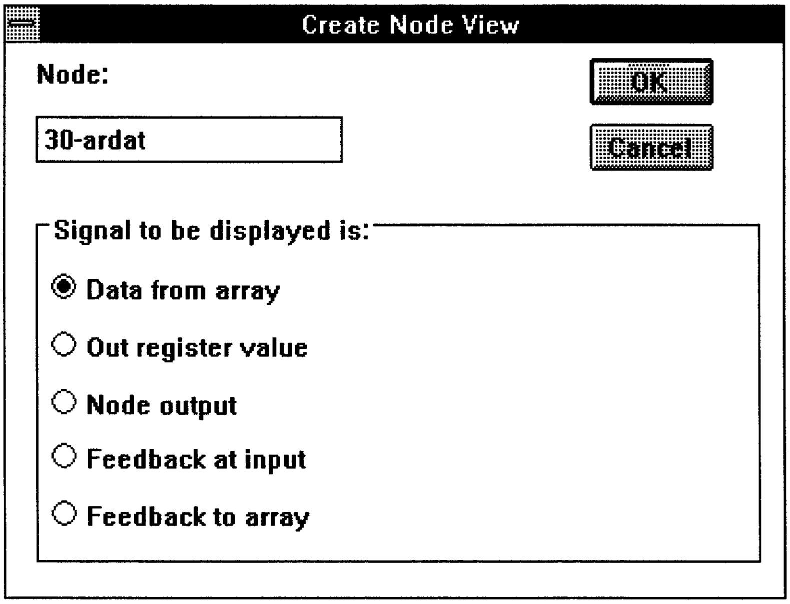
Many nodes contain several points at which you can trace simulation values. **Create View Node** lets you select which of those points to view.

A view node allows you to see what is happening at various points inside a node. Selecting **Create View Node** brings up the Create Node View dialog box (Figure 2-13), which allows you to select points to view within a selected node. To bring up this dialog box, you must select a node with the current view set to FULL. (See Section 2.6., "The Views Menu", for information about changing views.)

The Create Node View dialog box displays the node name with the view node name to be created directly below it. Nova creates the view node name by taking the node number, followed by a '-' and an extension to represent the selected signal to be displayed.

The view node points that can be displayed depend upon the selected node. Examples of view node points that can be displayed include:

- Data from Array - This is the data at the output of an OR-XOR combination of gates. Extension is "ardat".
- Out value before OE - This is the data on the output pin if the output enable is asserted. This includes the output buffer inversion, if there is one. Extension is "b_oe".
- OE Value - This is the state of the output enable. If high, OE is asserted so the output is driven. Extension is "oe".
- Node Output - This is the data on the pin. This is the default view for output nodes. Extension is "out".


Create Node View

Node:

30-ardat

Signal to be displayed is:

[✓ Data from array]

Out register value

Node output

Feedback at input

Feedback to array

Figure 2-13. Create Node View Dialog Box.

- Feedback at input - This is the data at the D input of the input register, if there is one. If there is no input register, feedback at input and feedback to array are identical. Extension is “fbkin”.

- Feedback to array - This is the data that is being fed to the array. It differs from feedback at input because it may be the other side of a register. Extension is “fbk_ar”.

Selecting “OK” closes the Create Node View dialog box and creates a view node, displayed at the end of the node list. Selecting “Cancel” closes the Create Node View dialog box without creating the view node.

To delete a view node, select the view node to delete, then select **Delete View Node** from the Edit menu.

2.4. The Edit Menu

2.4.4. Nodes

2.4.4.2. Setting Input Node Values

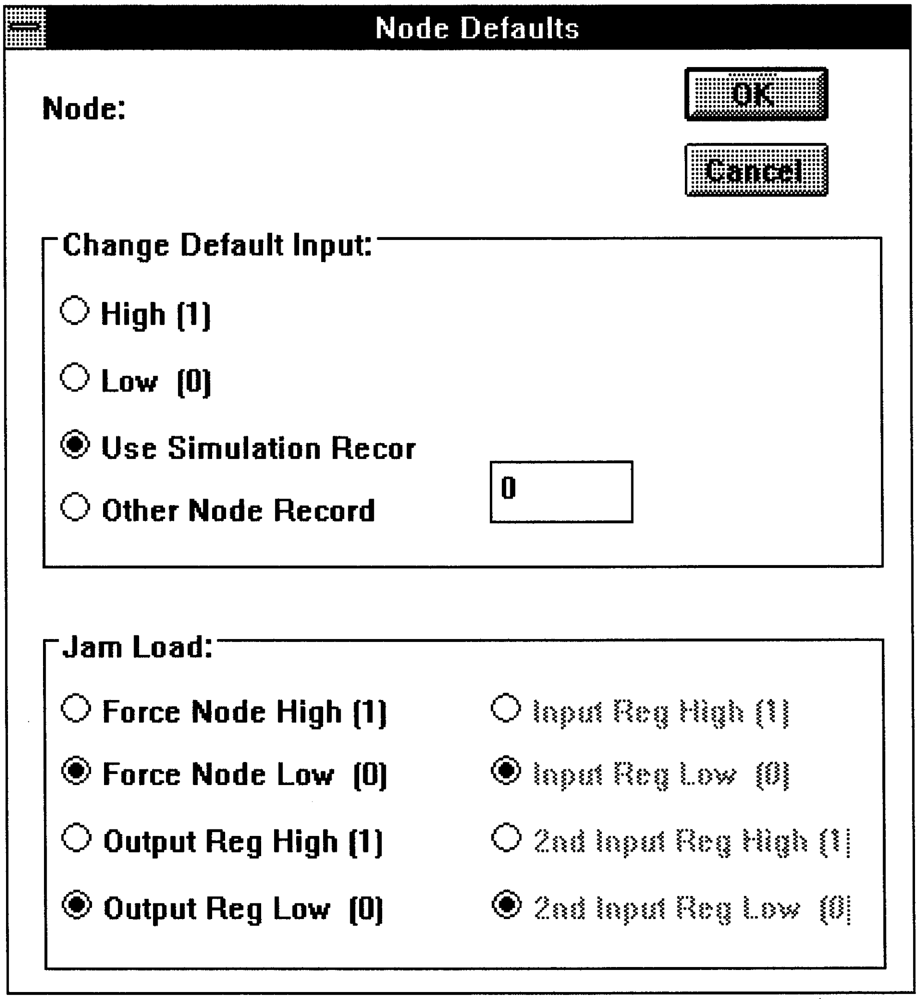
Node Defaults lets you specify the default source for the displayed value of a node.

Selecting **Node Defaults** brings up the Node Defaults dialog box (Figure 2-14).

You use the Change Default Input window of the Node Defaults dialog box to specify the high-impedance source for the value of an input node. The current setting is shown highlighted within this window.

There are four possible settings for each input. They are:

- High (1): tie the signal to Vcc.
- Low (0): tie the signal to Vss (ground)..
- Use Simulation Record: use the value(s) in the simulation record (.sim file).
- Other Node Record: tie the signal to another node. Enter the node number on the line to the right of the Other Node Record button.



Node Defaults

Node:

OK

Cancel

Change Default Input:

○ High (1)

○ Low (0)

● Use Simulation Recor

○ Other Node Record

0

Jam Load:

○ Force Node High (1)

● Force Node Low (0)

○ Output Reg High (1)

● Output Reg Low (0)

○ Input Reg High (1)

● Input Reg Low (0)

○ 2nd Input Reg High (1)

● 2nd Input Reg Low (0)

Figure 2-14. Node Defaults Dialog Box (1).

2.4. The Edit Menu

2.4.4. Nodes

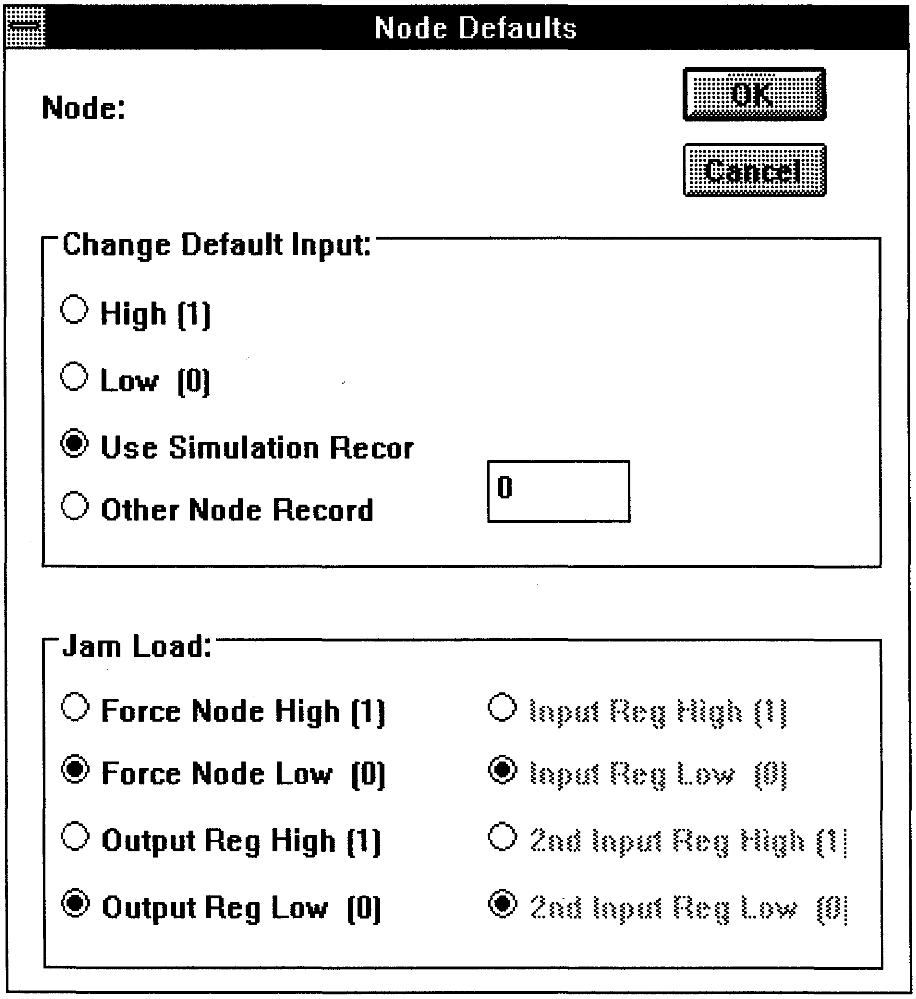
2.4.4.3. Forcing Output Node Values

Node Defaults lets you force the value of an output node at a specified point.

Selecting Node Defaults brings up the Node Defaults dialog box (Figure 2-15).

You use the Jam Load window of the Node Defaults dialog box to force an output node to a specified value. Values of these nodes rarely need to be touched for normal simulations. However, for multi-segment simulation (for long counters and other long-period design) or if there are problems in simulating the start-up of a circuit, the values may need to be changed. The current setting is shown highlighted.

Depending on the type of node, it may be possible to select from Force Node High(1), Force Node Low(0), Output Reg High(1), Output Reg Low (0), Input Reg High (1), Input Reg Low (0), 2nd Input Reg High (1), and 2nd Input Reg Low(0).


Node Defaults

Node:

OK

Cancel

Change Default Input:

○ High (1)

○ Low (0)

[✓ Use Simulation Recor]

○ Other Node Record

0

Jam Load:

○ Force Node High (1)

[✓ Force Node Low (0)]

○ Output Reg High (1)

[✓ Output Reg Low (0)]

○ Input Reg High (1)

[✓ Input Reg Low (0)]

○ 2nd Input Reg High (1)

[✓ 2nd Input Reg Low (0)]

Figure 2-15. Node Defaults Dialog Box (2).

2.4.5. Working With Buses

Sometimes, it's more convenient to group several traces in a simulation and view them as a single trace. You can do this with the Create Bus, Delete Bus, and Edit Bus items in the Edit menu.

Selecting **Create Bus** brings up the Bus dialog box (Figure 2-16). This dialog box combines nodes into a user-named bus. The View list in the dialog box contains the names of all nodes in the current view. The Bus list holds the names of each node in the bus. A bus may be made up of any number of nodes.

Selecting OK closes the Bus dialog box and creates a bus with the specified bus name. Buses are placed at the top of the trace area. Selecting Cancel closes the Bus dialog box without changing the trace area. Bus values are not displayed unless a measuring cursor is present.

To add a node to the bus: select the node from the View list and select the "Add>>" button. Double-clicking on the node name also adds the selected node to the bus. The new node is added below the selected node of the bus.

NOTE

You can only add nodes in the current view to a bus in that view.

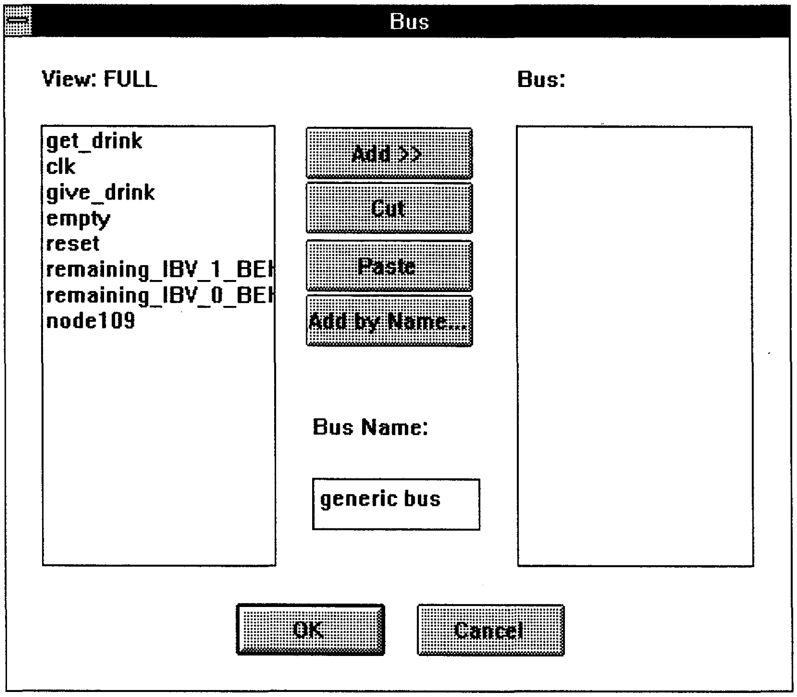


Figure 2-16. Bus Dialog Box.

Clicking on the Add-by-Name button brings up a dialog box that asks you to specify the name(s) of signals to add to the bus. The use of wild card characters is permitted. A "?" matches a single character; a "*" matches any string of characters. The construct *name*[*m*:*n*] denotes a range of signals, numbered from *m* through *n*, beginning with the characters *name*. For example, "input[0:3]" matches signals input0, input1, input2, and input3.

To remove a node from the bus: select the node to be removed and select the "Cut" button. Double-clicking on the node name in the Bus list also removes the node from the bus.

To change a node's position in the bus: Select the node, then click "Cut." Select another node, then click "Paste". The node that was previously cut will be inserted below the newly selected node.

To name the bus: click on the line below the words “Bus Name” and enter the name for the bus. If no name is provided, the bus is named “generic bus”.

To delete a bus: Select a bus trace by clicking on the bus name button or the bus trace. After the bus is selected, selecting the **Delete Bus** item from the Edit menu brings up a dialog box with which you can remove the bus from the trace area.

Edit Bus brings up the same Bus dialog box used for creating the bus. The bus name line is filled in, and the nodes in the bus are displayed in the Bus list. Add and remove nodes from the bus in the same way as when you create a bus. You may also change the bus name when editing the bus.

After all changes have been completed, selecting OK closes the bus dialog box and applies the modifications to the selected bus. Selecting Cancel closes the dialog box without updating the bus.

Bus Radix brings up a submenu that allows you to choose how bus information is displayed. The three choices are binary, octal and hexadecimal. Hexadecimal is the default.

2.5. The Simulate Menu

The Simulate Menu has only one menu item: Execute.

Selecting **Execute** from the Simulate Menu (Figure 2-17) simulates the design's operation. The Nova screen is redrawn, and the resulting waveforms are displayed.

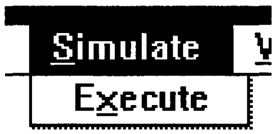


Figure 2-17. Simulate Menu.

2.6. The Views Menu

Items in the Views menu let you select the views (i.e., groupings of traces) that you see in the trace area.

The View menu (Figure 2-18) contains five items:

- **Edit Views**: lets you create and edit views.
- **Select View:** lets you select a view to display.
- **Delete View:** lets you remove one or more views from the list.
- **Zoom In (2X):** multiplies the displayed timescale resolution factor by two.
- **Zoom Out (1/2X):** divides the displayed timescale resolution factor by two.

Each of these items is discussed in greater detail in the following pages.

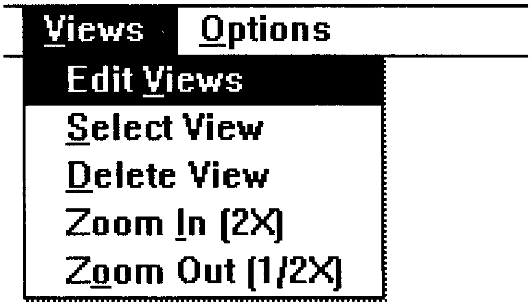


Figure 2-18. Views Menu.

2.6. The Views Menu

2.6.1. Editing Views

Edit Views lets you create new views, and add, remove, or exchange traces in existing views.

Three views are automatically created with each .JED file: full, pins-only, and pins & registers. The full view (default) lists all nodes in the design. This view cannot be edited. The pins-only view contains only nodes that are attached to pins. The pins & registers view contains all nodes attached to registers or pins.

Selecting **Edit Views** displays the Edit Views dialog box (Figure 2-19), used to edit the current view. The view list on the left displays the FULL view, which contains the default traces for all nodes. Use this list, along with appropriate buttons, to add or remove traces from the view list on the right.

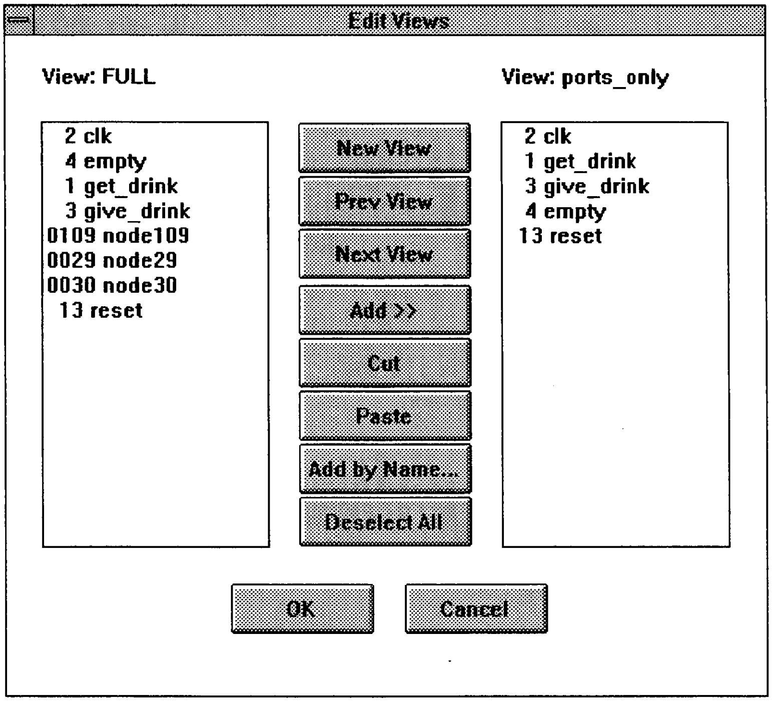
To create a new view: click on New View. You will be prompted for a name, which is placed at the top of the right-hand view list.

To move between views: click on Next View or Previous View.

To add a trace to a view: select one or more traces from the left (Full) view window, then click Add>>. If a trace is also selected in the right window, the new traces are inserted after the selection; otherwise, the new traces are added to the end of the view.

To remove traces from a view: select the traces in the right window, then click on Cut.

To exchange (i.e., re-order) traces within a view: Select one or more traces from the right view window, then click Cut. Then, select another trace from the right view window and click Paste. The previously cut trace(s) are inserted after the selected trace.



View: FULL

```
   2 clk
   4 empty
   1 get_drink
   3 give_drink
0109 node109
0029 node29
0030 node30
  13 reset
```

View: ports_only

```
 2 clk
 1 get_drink
 3 give_drink
 4 empty
13 reset
```

Figure 2-19. Edit Views Dialog Box.

Add-by-Name brings up a dialog box that asks you to specify the name(s) of traces to add. The use of wild card characters is permitted. A "?" matches a single character; a "*" matches any string of characters. The construct *name*[*m*:*n*] denotes a range of signals, numbered from *m* through *n*, beginning with the characters *name*. For example, "input[0:3]" matches signals input0, input1, input2, and input3. You can also use multiple expressions separated by spaces.

Deselect All unselects all selected traces in either window.

Selecting OK closes the Edit Views dialog box and updates the trace area to reflect changes made to the view. Selecting Cancel closes the Edit Views dialog box without making any changes to the view.

2.6.2. Selecting and Deleting Views

Select View lets you change the active view. **Delete View** lets you remove a view from the list of available views.

Select View brings up the Select View dialog box (Figure 2-20). The View line gives the name of the current view. To change the current view, select the desired view from the list, then click OK or type a carriage return. Clicking Cancel closes the Select View dialog box without affecting the active view.

Delete View also brings up the Select View dialog box. Select the view to delete from the scrollable list. The FULL view may not be removed, and so it is not included in this list. Clicking OK or typing carriage return applies the change to the list of views. If you remove the current active view, the active view changes to FULL. There is no undo for Delete View, so be certain the view you are deleting is correct before clicking on OK or typing a carriage return. Cancel closes the Select View dialog box without deleting the selected view.

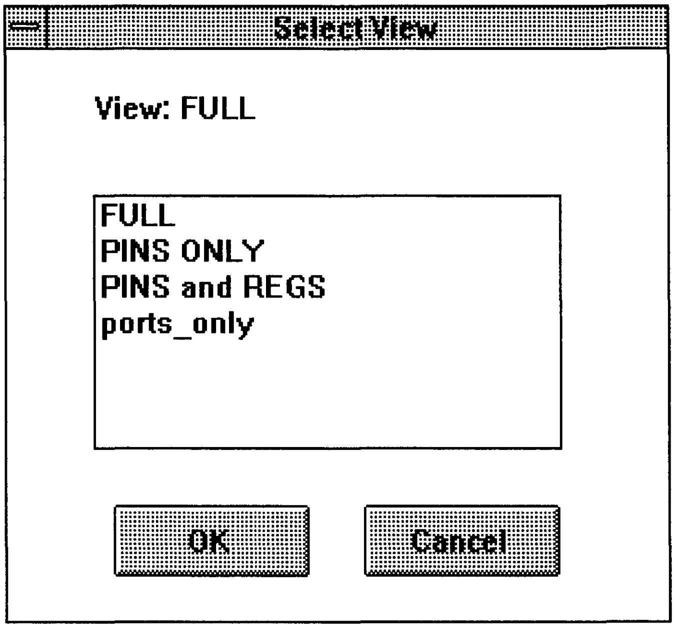


Figure 2-20. Select View Dialog Box.

2.6.3. Zoom In, Zoom Out

Zoom In doubles the time scale resolution of the trace window. **Zoom Out** halves the time scale resolution of the trace window.

Zoom In doubles the time scale resolution of the trace window, i.e., it doubles the number of pixels in the X-axis used to display one tic of simulation time. The result is to "zoom in" the view of displayed traces.

Zoom Out does just the reverse.

The resolution setting must be 1 or greater. The default is 5. Attempting to set the time scale resolution lower than 1 has no effect.

2.7. The Options Menu

The Options Menu contains items that let you specify the simulation length, create or delete simulation segments, and specify the viewing resolution of the trace area.

The Options Menu (Figure 2-21) contains five items:

- **Simulation Length:** lets you set the length of the simulation.
- **Create Segment:** lets you create a segment, or "new-start-point", within the simulation.
- **Delete Segment:** lets you delete a previously created segment from the simulation.
- **Resolution:** lets you stretch and compress displayed traces.
- **Signal Name Size:** lets you specify the width in characters of Nova's signal name buttons.

Each of these items is described in greater detail in the following pages.

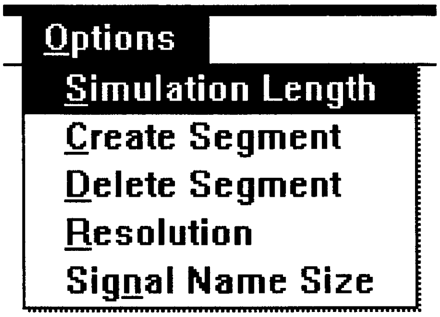


Figure 2-21. Options Menu.

2.7. The Options Menu

2.7.1. Simulation Length

Simulation Length lets you set the length of the simulation.

Selecting **Simulation Length** brings up the Simulation Length dialog box (Figure 2-22).

The minimum and default simulation length is 256 tics. The maximum simulation length is 9984. Clicking the up arrow adds 64 tics to the simulation length, to a maximum of 9984. Clicking the down arrow subtracts 64 tics from the simulation length, to a minimum of 256 tics.

You can also set the simulation length by typing a number on the line next to the up and down arrows. The number will be rounded downward to the nearest multiple of 64.

Clicking OK closes the Simulation Length dialog box and sets the simulation length, to be used on the next simulator run. Clicking Cancel closes the Simulation Length dialog box without affecting the simulation length.

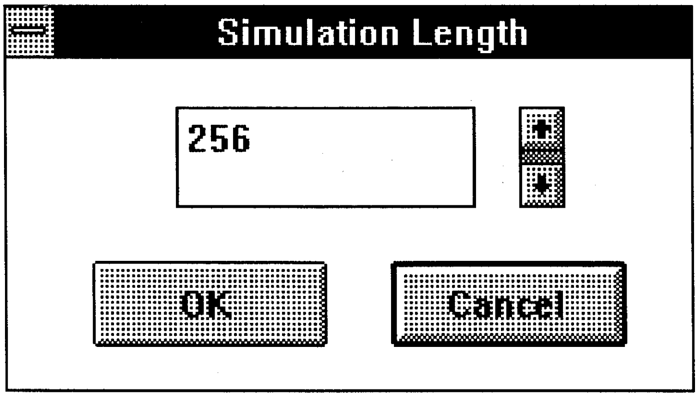


Figure 2-22. Simulation Length Dialog Box.

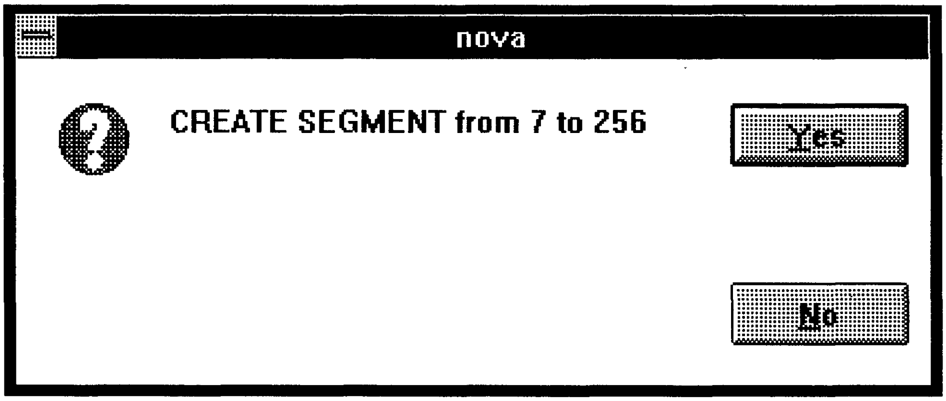
2.7.2. Creating and Deleting Segments

Create Segment lets you create a segment, or "new start point," within the simulation. **Delete Segment** deletes a previously created start boundary.

A segment is a point in the simulation at which various nodes are reset to their "jam load" values (set through the Node Defaults dialog box).

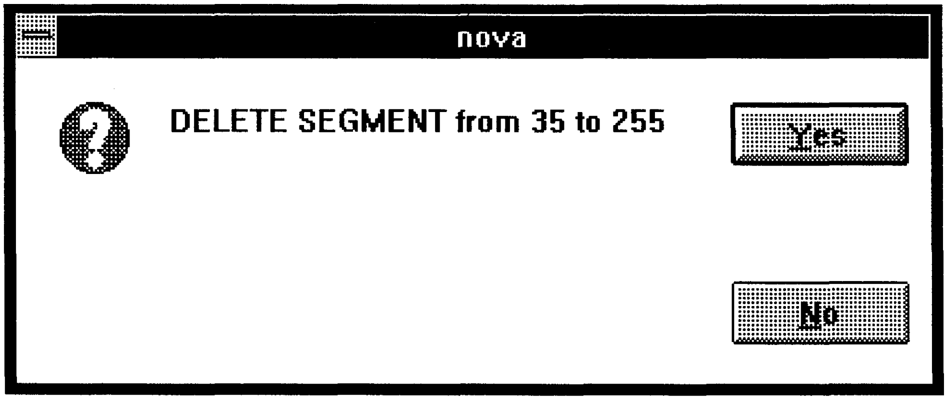
To create a segment, position the leftmost measuring cursor where you would like the segment to start, then select **Create Segment** from the Options menu to bring up the Create Segment dialog box (Figure 2-23). The dialog box indicates the starting and ending boundaries of the segment. Selecting Yes closes the dialog box and creates the new simulation segment. Selecting No closes the dialog box box without creating the segment. You may create up to 15 segments.

To delete a segment, position the leftmost measuring cursor within the segment to be deleted, then select Delete Segment to bring up the Delete Segment dialog box (Figure 2-24). The dialog box indicates the segment boundaries for the segment you wish to delete. Selecting Yes closes the dialog box and deletes the segment. Selecting No closes the dialog box without removing the segment.


nova

CREATE SEGMENT from 7 to 256

Figure 2-23. Create Segment Dialog Box.


nova

DELETE SEGMENT from 35 to 255

Figure 2-24. Delete Segment Dialog Box.

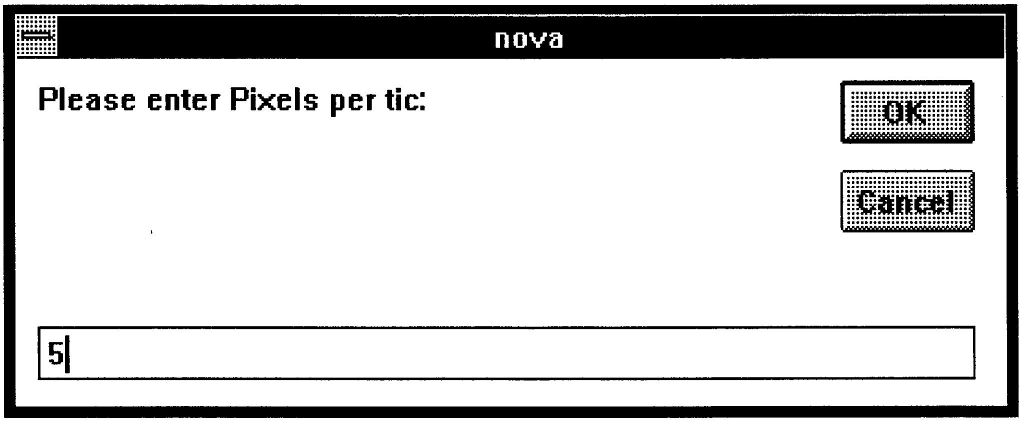
2.7.3. Resolution

Resolution lets you stretch and compress displayed traces.

Selecting Resolution from the Options menu brings up the Resolution dialog box (Figure 2-25). This dialog box lets you set the number of screen pixels on the X-axis to be used per simulation tic. Varying this number effectively stretches or compresses the traces displayed on the screen.

The pixels-per-tic setting may be any number between 1 and 100. The default is 5. The larger the number, the more "stretched" the traces appear; the smaller the number, the more compressed the traces appear.

Selecting OK closes the Resolution dialog box and updates the trace display. Selecting Cancel closes the Resolution dialog box without updating the trace display.


nova

Please enter Pixels per tic:

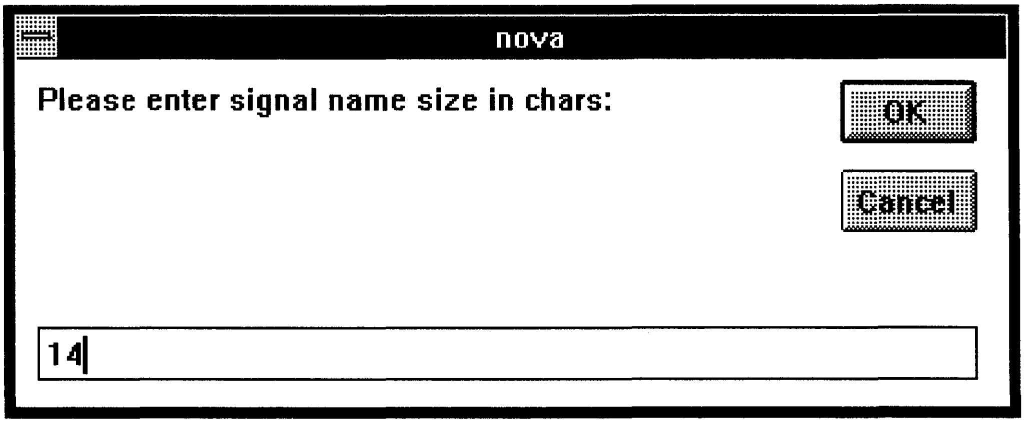
5

Figure 2-25. Resolution Dialog Box.

2.7.4. Signal Name Size

Signal Name Size lets you specify the width in characters of Nova's signal name list buttons.

Selecting **Signal Name Size** from the Options menu brings up the Signal Name size dialog box (Figure 2-26). This dialog box lets you set the width in characters of the buttons in Nova's signal name list.



Please enter signal name size in chars:

14

Figure 2-26. Signal Name Size Dialog Box.

Index

Warp2™

VHDL Development System

SpDE/Warp System User's Manual

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
(408)943-2600

Cypress Software License Agreement

1. LICENSE. Cypress Semiconductor Corporation ("Cypress") hereby grants you, as a Customer and Licensee, a single-user, non-exclusive license to use the enclosed Cypress software program ("Program") on a single CPU at any given point in time. Cypress authorizes you to make archival copies of the software for the sole purpose of backing up your software and protecting your investment from loss.

2. TERM AND TERMINATION. This agreement is effective from the date the diskettes are received until this agreement is terminated. The unauthorized reproduction or use of the Program and/or documentation will immediately terminate this Agreement without notice. Upon termination you are to destroy both the Program and the documentation.

3. COPYRIGHT AND PROPRIETARY RIGHTS. The Program and documentation are protected by both United States Copyright Law and International Treaty provisions. This means that you must treat the documentation and Program just like a book, with the exception of making archival copies for the sole purpose of protecting your investment from loss. The Program may be used by any number of people, and may be moved from one computer to another, so long as there is **No Possibility** of its being used by two people at the same time.

4. DISCLAIMER. THIS PROGRAM AND DOCUMENTATION ARE LICENSED "AS-IS," WITHOUT WARRANTY AS TO PERFORMANCE. CYPRESS EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTY OF MERCHANTABILITY OR

FITNESS OF THIS PROGRAM FOR A PARTICULAR PURPOSE.

5. LIMITED WARRANTY. The diskette on which this Program is recorded is guaranteed for 90 days from date of purchase. If a defect occurs within 90 days, contact the representative at the place of purchase to arrange for a replacement.

6. LIMITATION OF REMEDIES AND LIABILITY. IN NO EVENT SHALL CYPRESS BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM PROGRAM USE, EVEN IF CYPRESS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. CYPRESS'S EXCLUSIVE LIABILITY AND YOUR EXCLUSIVE REMEDY WILL BE IN THE REPLACEMENT OF ANY DEFECTIVE DISKETTE AS PROVIDED ABOVE. IN NO EVENT SHALL CYPRESS'S LIABILITY HEREUNDER EXCEED THE PURCHASE PRICE OF THE SOFTWARE.

7. ENTIRE AGREEMENT. This agreement constitutes the sole and complete Agreement between Cypress and the Customer for use of the Program and documentation. Changes to this Agreement may be made only by written mutual consent.

8. GOVERNING LAW. This Agreement shall be governed by the laws of the State of California. Should you have any question concerning this agreement, please contact:

   Cypress Semiconductor Corporation
   Attn: Legal Counsel
   3901 N. First Street
   San Jose, CA 95134-1599

   408-943-2600

Table of Contents

Chapter 1 Introduction

About This Chapter

This chapter introduces the purpose, elements, and design flow of the *Warp* system's SpDE tools.

1.1 Introduction

SpDE is the name of Cypress Semiconductor Corporation's pASIC processing toolkit.

This chapter tells you how to use the SpDE Toolkit to process pASIC designs. It assumes that you are already familiar with common user interface elements for your platform, such as scroll bars, menu buttons, windows, etc. The SpDE toolkit runs within a rich graphical environment that provides a consistent user interface and a unified pASIC processing environment.

The SpDE Toolkit gives you the following pASIC processing capabilities:

- Design verification
- Logic optimization
- Automatic placement and routing
- Physical viewing of pASIC layout
- Post-layout delay modeling and back-annotation
- Timing analysis
- Automatic test vector generation (ATVG)

The typical order of use of the SpDE tools in *Warp* is:

- import a .QDF file, using Import QDIF from the File menu;
- run all the SpDE tools in sequence, using Run All Tools from the Tools menu;
- export the .LOF file using Export/LOF from the File menu.

Design Verifier

The Design Verifier analyzes a user design for common design errors, architectural violations, and architectural warnings. In addition to catching errors prior to running the other SpDE tools, the Design Verifier alerts the user to such potential problems as excessive fanout.

Logic Optimizer

Logic optimization is the first step in automatic placement and routing. The Logic Optimizer utilizes two modules to efficiently map logic into the pASIC logic cell: the Packer and the Technology Mapper.

Starting with the user design, the Packer maps logic symbols (hard macros) into logic cells. As many as four macro symbols may be packed into a single logic cell. The Packer allows users to design with easy and convenient macros, while achieving high utilization of the pASIC resources.

The Technology Mapper provides automatic logic cell optimization. For example, NOR gates can be implemented more efficiently than OR gates. The Technology Mapper introduces and removes inversion bubbles in order to improve capacity and performance. In more general terms, the Logic Optimizer merges gates in order to achieve more efficient implementations.

Placer

Placement is the second step in automatic placement and routing (APR). Starting with the fully packed design, the Placer moves cells around the pASIC to minimize routing delays. This not only produces extremely fast pASICs, it also allows completely automatic routing. The Placer offers fixed placement of I/O cells as well as logic cells. Fixed I/O cell placement lets you take circuit board considerations into account, while fixed logic cell placement offers precise control over internal routing delays.

Working closely with the Path Analyzer, the Placer offers timing-driven placement. Timing constraints are specified directly in the Path Analyzer; these are passed to the Placer to insure that critical timing goals are met automatically.

Router

The Router connects cells by routing nets within the pASIC matrix. The Router's optimization capabilities minimize routing delays and routing resources required.

Physical Viewer

The Physical Viewer displays the physical layout of a pASIC part. The viewer allows the user to inspect the results of APR for placement efficiency and routing congestion. Timing-critical paths can easily be identified. This information can be fed back into APR in order to improve the physical layout.

Delay Modeler

The Delay Modeler performs precise post-layout delay calculations using state-of-the-art circuit analysis techniques. Processing the complete results of APR, the Delay Modeler analyzes packing, placement, and routing to determine intrinsic delays and routing delays for the entire design. These precisely calculated delays are written directly into the logic simulator netlist for timing-accurate results.

Path Analyzer

The Path Analyzer performs path analysis of the circuit delays from the Delay Modeler. The Path Analyzer offers automatic analysis of the complete design, as well as interactive analysis of user-specified portions of the design.

Automatic Test Vector Generator

The Automatic Test Vector Generator (ATVG) provides convenient and thorough test coverage of programmed pASIC parts. Using the internal scan path circuitry built into every pASIC, the Automatic Test Vector Generator achieves excellent test coverage. The degree of test coverage can be easily controlled, and user-specified test patterns can be included. ATVG provides both ideal and actual coverage statistics, allowing the design to be tuned for optimal testability.

Starting SpDE

To start SpDE, double-click on the “SPDE” icon in the *Warp2* program group.

1.2 The File Menu

The File menu (Figure 1-1) includes commands relating to: creating, opening, and saving binary chip files: importing/exporting QDIF, EDIF, and LOF files; printing the physical view of the layout; and exiting SpDE.

SpDE operates primarily on chip files (file extension .CHP) and QDIF files (file extension .QDF). Chip files have a compact and efficient binary file format, while QDIF files have an open ASCII file format. Chip files are used within SpDE for reasons of efficiency, while QDIF files are used to import the output from *Warp* synthesis operations.

New clears any current design and initializes SpDE to operate on a new design.

Open loads an existing chip file (file extension *.CHP*). Selecting **Open** brings up a dialog box, from which you can select a chip file. Doing so re-initializes SpDE and loads the specified chip file.

Save saves the current design as a chip file. **Save** uses the format specified by the current selection in the Tools-Options-Simulator menu item. The default is Viewsim. Once a design has been run through the tools, a saved file contains placement, routing, timing, testing, and programming information.

Save As saves the current design as a chip file with a user-specified name. Selecting **Save As** brings up a dialog box, which allows you to specify the name for the chip file. This menu item gives you the capability to save backup or history copies of a design for later reference.

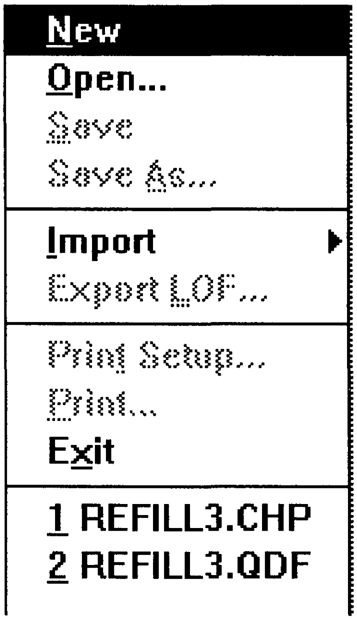


Figure 1-1. File Menu

Import loads files saved in QDIF or EDIF format. **Export LOF** saves the current design in LOF-file format. The **Import** and **Export LOF** menu items are discussed in greater detail in Section 1.2.1, "Importing, Exporting".

Print Setup invokes the printer setup dialog box for the default printer driver. **Print** prints the current physical view using the parameters set in **Print Setup**.

Exit terminates SpDE, prompting you to save the design if it has been modified since the last save. On PC's and compatibles, double-clicking the Control-Menu box (in the upper-left corner of the SpDE window) is a shortcut for **Exit**.

Below the **Exit** menu item, the last five accessed files are listed. This provides a shortcut for the **Open** and **Import** commands. Clicking on a chip file from this list opens the file; clicking on another file type from this list imports the file.

1.2 The File Menu

1.2.1. Importing, Exporting

Import lets you import files written in QDIF or EDIF formats. **Export LOF** writes designs into LOF-format files.

The **Import** menu item features a pop-up sub-menu with QDIF and EDIF entries. Position the cursor on the **Import** menu item, click and hold the button, drag the cursor to select one of QDIF or EDIF. This bring up a dialog box, from which you can select a design. Doing so re-initializes SpDE and loads the specified file.

As the file is loaded, the Design Verifier analyzes it for common design errors, architectural violations, and architectural warnings. A dialog box lists any problems found.

Problems fall into one of four categories: Notice, Warning, Error, or Fatal Error:

- Notices and Warnings serve as alerts; they may or may not require action (e.g., excessive fanout).
- Errors prevent devices from being programmed, but do not prevent the toolkit from running (e.g., floating inputs).
- Fatal Errors prevent the toolkit from being run (e.g., net with multiple drivers).

The Design Verifier removes unused gates from the design; any gate that cannot affect an output is removed. Unused counter and register bits, for example, are automatically removed. “Stripped” gates are flagged with a Notice to ensure that their removal is sound.

Export LOF saves the current design as a Link Object Format file (file extension .LOF) with the specified name. This format yields a compressed file that contains all the data required to program and test pASIC devices. Use this file to program the pASIC on a device programmer.

1.3 The View Menu

The View menu (Figure 1-2) includes commands for controlling the physical view and highlighting nets.

Zoom In "magnifies" the Physical View. The command supports two modes: "click" and "drag". To use click mode, select **Zoom In**, then click the mouse button once. This increases the scale factor by 1.25, and centers the view on the current cursor position.

Drag mode allows an arbitrary rectangle to specify the desired view. To use drag mode, select **Zoom In**, position the cursor where you want the viewing rectangle to begin, and click and hold the mouse button. Then, drag the cursor to the diagonally opposite corner of the viewing rectangle and release the mouse button. The view adjusts to fit the specified rectangle as closely as possible. The shortcut key for **Zoom In** is Ctrl-Z.

Zoom Out "de-magnifies" the view. The shortcut key for Zoom Out is Ctrl-X.

Full Fit modifies the scale factor to fit a view of the entire pASIC chip on the screen. The shortcut key for **Full Fit** is Ctrl-F.

Normal Fit sets the scale factor to its initial value, and centers the view on the selected position. The shortcut key for **Normal Fit** is Ctrl-N. (Note, though, that you must still click the mouse button to select the viewing position.)

Preferences sets physical view options. These are discussed in greater detail in Section 1.3.1, "Preferences".

Hilight Net allows you to select nets to be highlighted in the physical view. This menu item is discussed in greater detail in Section 1.3.2, "Hilight Net".

Redraw erases and redraws the physical view.

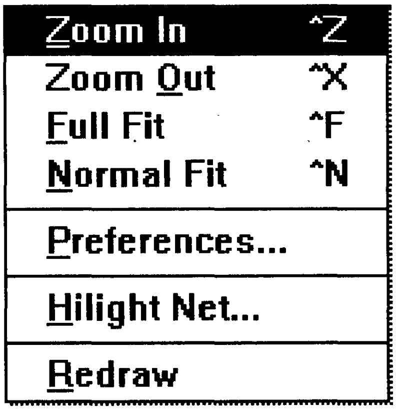


Figure 1-2. View Menu

1.3 The View Menu

1.3.1. Preferences

Preferences sets physical view options.

Selecting **Preferences** brings up the Preferences dialog box (Figure 1-3).

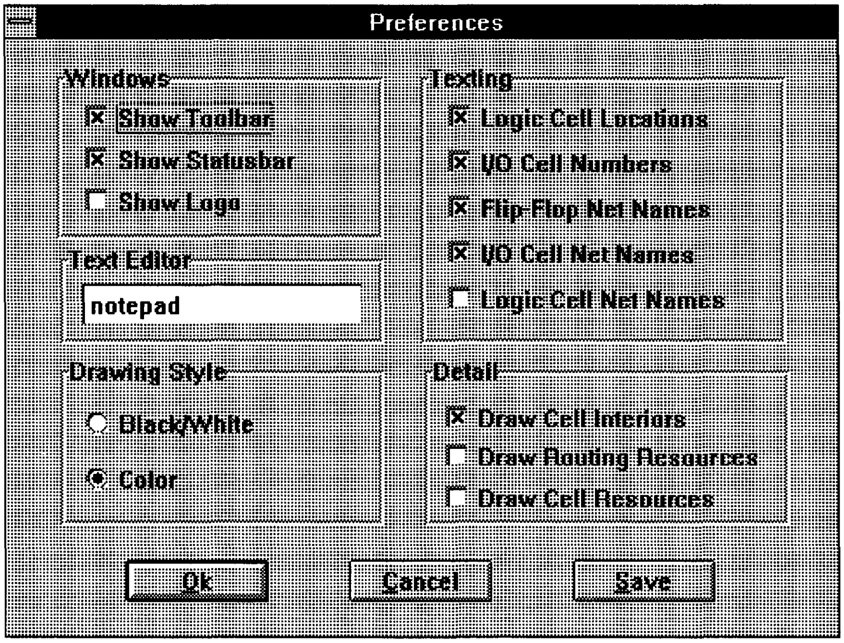
The **Texting** group of check boxes includes items relating to text in the physical view. Table 1-1 lists the check boxes and examples of the items they control. Text items can be turned off to increase redraw speed or simplify the physical view.

Table 1-1. Texting Options in the Physical View

Item	Example
Logic Cell Locations	A1, A2, B1, C1, H12
I/O Cell Numbers	3, 12, 24, 42
Flip-Flop Net Names	specified in schematic
I/O Cell Net Names	specified in schematic
Logic Cell Net Names	specified in schematic

The **Flip-Flop Net Names** option only includes the net names of logic cell flip-flop outputs. The **Logic Cell Net Names** option includes the net names of all logic cell inputs and outputs.

The **Drawing Style** group of radio buttons controls the use of color in classifying interconnects. In Black/White mode, all wires are shown in black. In Color mode, short wires are shown in blue and black, express wires are shown in green, I/O wires are shown in red, and clock traces are shown in mauve. (That's purple with a college education.)



Windows
- [x] Show Toolbar
- [x] Show Statusbar
- [ ] Show Logo

Text Editor
notepad

Drawing Style
- ○ Black/White
- ● Color

Texting
- [x] Logic Cell Locations
- [x] I/O Cell Numbers
- [x] Flip-Flop Net Names
- [x] I/O Cell Net Names
- [ ] Logic Cell Net Names

Detail
- [x] Draw Cell Interiors
- [ ] Draw Routing Resources
- [ ] Draw Cell Resources

Figure 1-3. Preferences Dialog Box

The **Detail** group includes the **Draw Cell Interiors** check box. De-selecting this check box replaces the detailed logic cells with simple boxes, thereby increasing redraw speed.

Clicking **OK** accepts all preference settings for tools that are subsequently executed. Clicking the **Save** button does the same, and also records the preference settings in a file. These settings are then used the next time SpDE is invoked.

1.3.2. Hilight Net

Hilight Net allows you to specify nets for SpDE to highlight in the physical view.

Selecting Hilight Net brings up the Highlight Net dialog box (Figure 1-4), and redraws the Physical View in light gray, which allows the highlighted nets to stand out. The net list box on the left of the dialog box contains the names of nets not currently highlighted, while the net list box on the right contains the names of nets that are currently highlighted.

To highlight nets, select one or more nets from the box on the left, or specify a net name in the “Wildcard Selection” field on the left side of the dialog box, and click on the right arrow button.

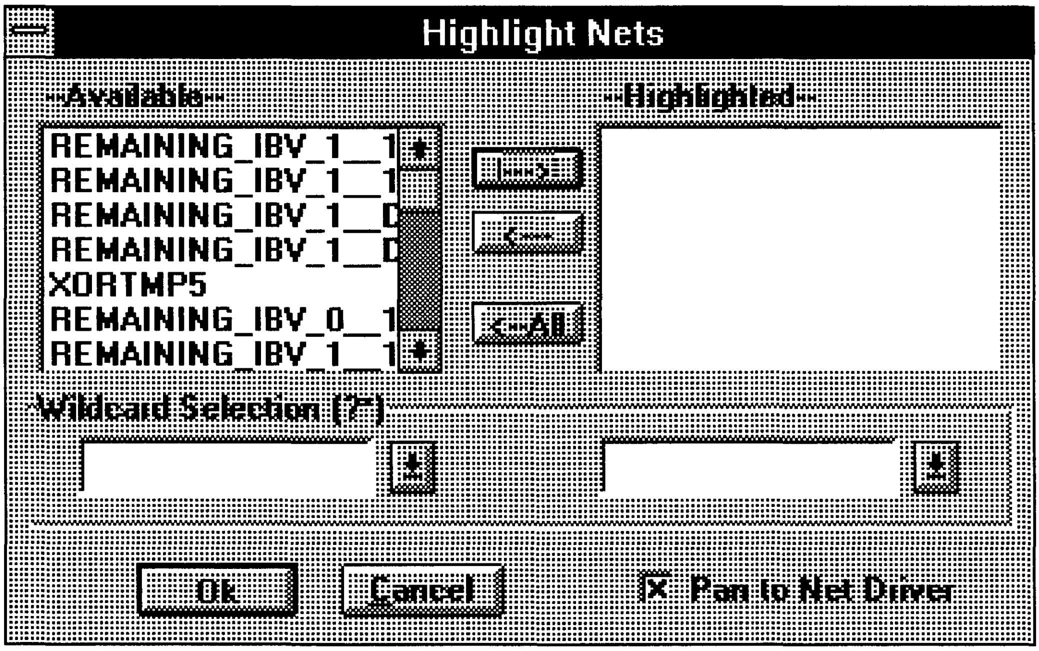
To remove nets from the highlight list, select one or more nets from the box on the right, or specify a net name in the “Wildcard Selection” field on the right side of the dialog box, and click on the left arrow button.

When using the “Wildcard Selection” fields, the wildcard characters “*” and “?” are accepted. The “*” character matches 0 or more occurrences of any character. The “?” character matches a single occurrence of a single character.

Double-clicking on a net name in either list moves it to the other list.

All nets can be removed from the highlight list by clicking on the ALL button.

Once in highlight net mode, the highlight status may be toggled by clicking directly on the desired nets in the physical view.



REMAINING_IBV_1__1
REMAINING_IBV_1__1
REMAINING_IBV_1__D
REMAINING_IBV_1__D
XORTMP5
REMAINING_IBV_0__1
REMAINING_IBV_1__1

Figure 1-4. Highlight Net Dialog Box

To exit highlight net mode, click OK. The Physical View will be redrawn in the normal mode. To exit without redrawing the Physical View, click Cancel.

1.4 The Tools Menu

The Tools menu is used to set up and run the optimizing, placing, routing, sequencing, delay modeling, back annotation, and path analysis tools.

The Tools menu (Figure 1-5) contains three items: Run Tools, Path Analyzer, and Options.

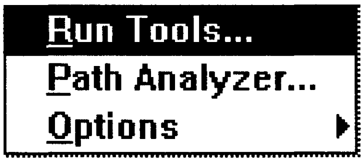
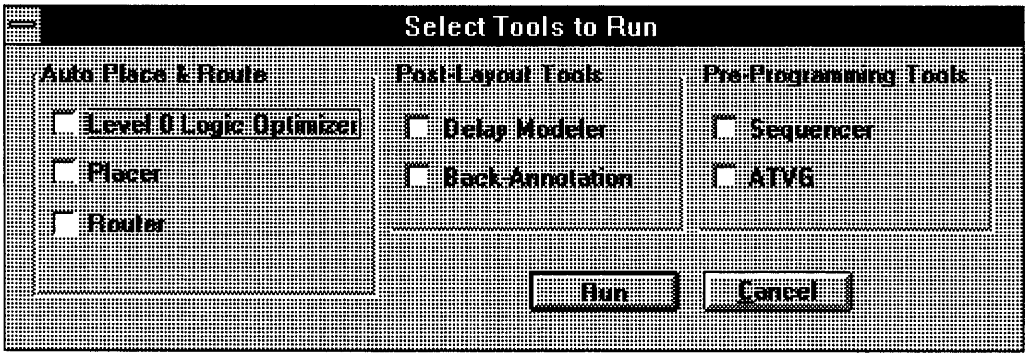


Figure 1-5. Tools Menu.

Run Tools opens the **Select Tools to Run** window (Figure 1-6). This window is used to select which tools are to be run on the design. Disabled tools are grayed out. Tools that have already been run are un-checked. Detailed descriptions of the Logic Optimizer, Placer, Router, Delay Modeling, Back Annotation, and ATVG tools can be found in Sections 2 and 3 of this manual. The Sequencer tool is only used to create data needed to program devices. It has no options, and is not documented.

Path Analyzer can be used to determine operating frequency, setup and hold times, and clock skew. This menu item is discussed in greater detail in Section 3.2., "Path Analyzer".

Options lets you select **General** or **Simulator** options. **General** options are the options for all tools. **Simulator** options affect only the Back-Annotation tool. The **General Tools** Options and **Simulator** Options windows are shown in Figures 1-7 and 1-8, respectively.



Select Tools to Run

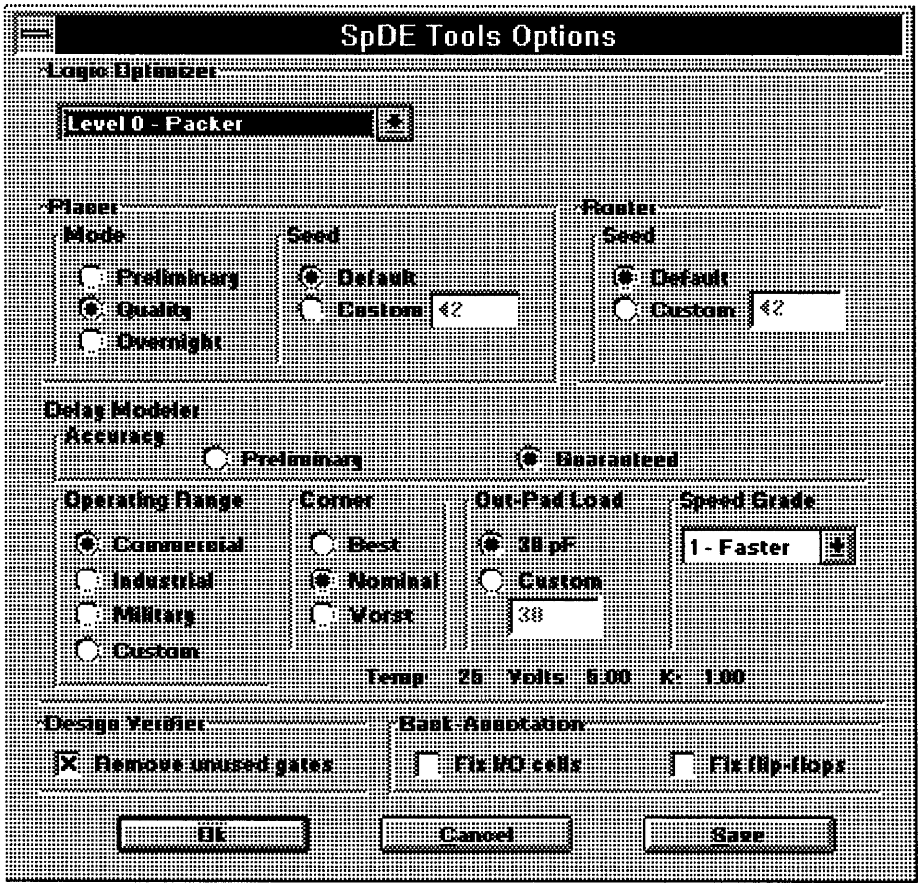
Auto Place & Route
- [ ] Level 0 Logic Optimizer
- [ ] Placer
- [ ] Router

Post-Layout Tools
- [ ] Delay Modeler
- [ ] Back Annotation

Pre-Programming Tools
- [ ] Sequencer
- [ ] ATVG

Run | Cancel

Figure 1-6. Select Tools to Run Dialog Box.



SpDE Tools Options

Logic Optimizer: Level 0 - Packer

Placer
- Mode: Preliminary [✓ Quality] Overnight
- Seed: [✓ Default] Custom 42

Router
- Seed: [✓ Default] Custom 42

Delay Modeler
- Accuracy: Preliminary [✓ Guaranteed]
- Operating Range: [✓ Commercial] Industrial Military Custom
- Corner: Best [✓ Nominal] Worst
- Out-Pad Load: [✓ 30 pF] Custom 30
- Speed Grade: 1 - Faster

Temp: 25 Volts: 5.00 K: 1.00

Design Verifier
- [x] Remove unused gates

Back-Annotation
- [ ] Fix I/O cells
- [ ] Fix flip-flops

Ok | Cancel | Save

Figure 1-7. General Tools Options Window.

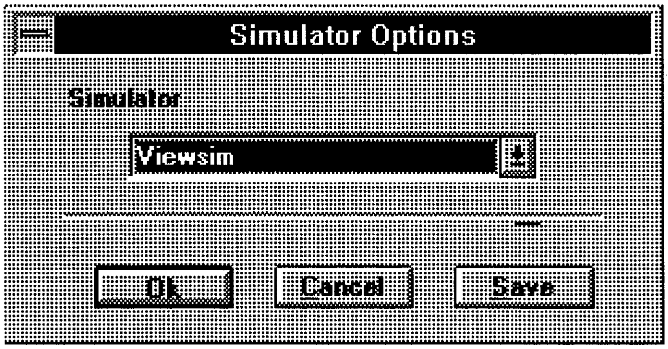


Figure 1-8. Simulator Options Window.

The **SpDE General Tools Options** window is used to configure each of the SpDE tools. A detailed description of each tool's options can be found in the section of this manual pertaining to that tool.

The **Simulator Options** window is used to select the simulator desired for back-annotated simulation. The Back Annotation tool uses this information to produce the proper timing netlist. See the section on the Delay Modeler and Back Annotation for details.

1.5 The Info Menu

The Info menu (Figure 1-7) includes items that provide statistics and other information about a design.

Cell Utilization reports on the number of cells of different types used in the design (Figure 1-8). It reports on the number of logic, input-only, clock-only, and bidirectional cells. Also included is a count of "partially-free" cells; these are logic cells with a free AND fragment. These cells can accept any macro that can be implemented in a single AND fragment. This information is provided to allow fine-tuning of high-utilization designs.

ATVG Coverage (Figure 1-9)reports on the design's test coverage, i.e., the percentage of detectable stuck-at faults) in the design. These statistics may be used as guidelines to improve design testability.

Tool Versions is provided for diagnostic purposes. **Tool Versions** lists the tools run on the current design, including the version number of each tool listed (Figure 1-10). Note that this is the version number that was run on the design, not the version number of the current tool.

Report File displays the Cypress report file produced by the *Warp* compiler and appended to by the SpDE tools.

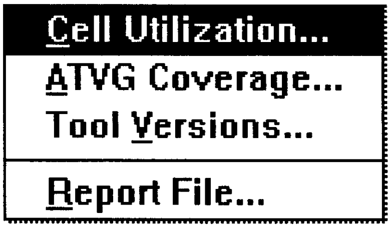
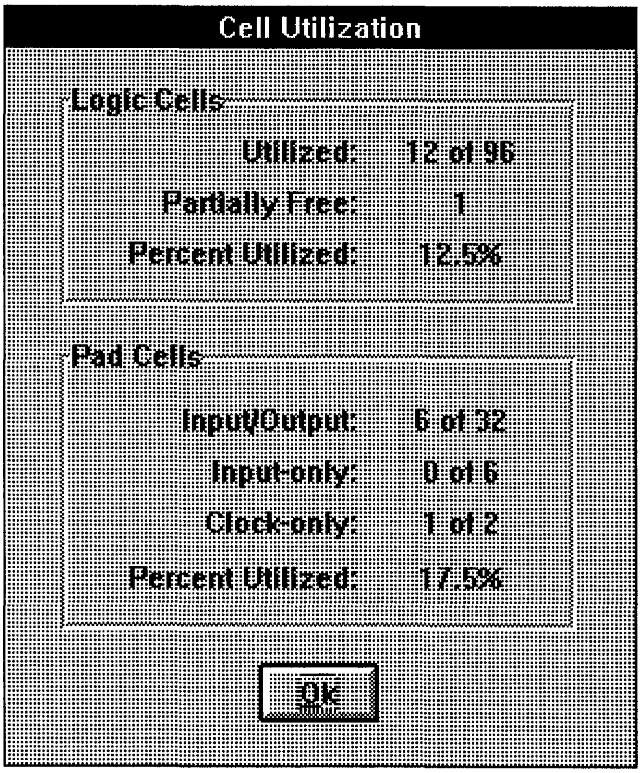


Figure 1-7. Info Menu


Cell Utilization

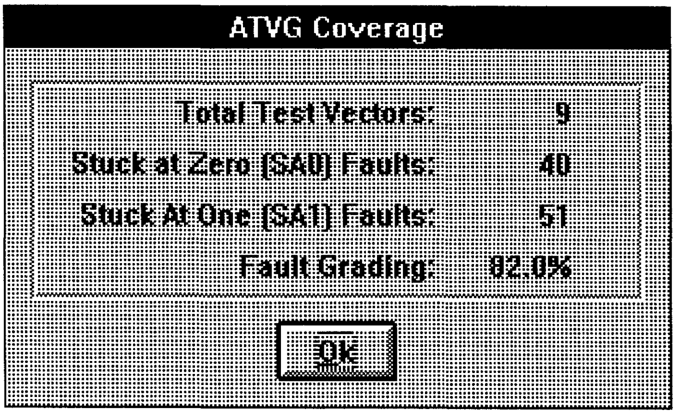
Logic Cells

Utilized:	12 of 96
Partially Free:	1
Percent Utilized:	12.5%

Pad Cells

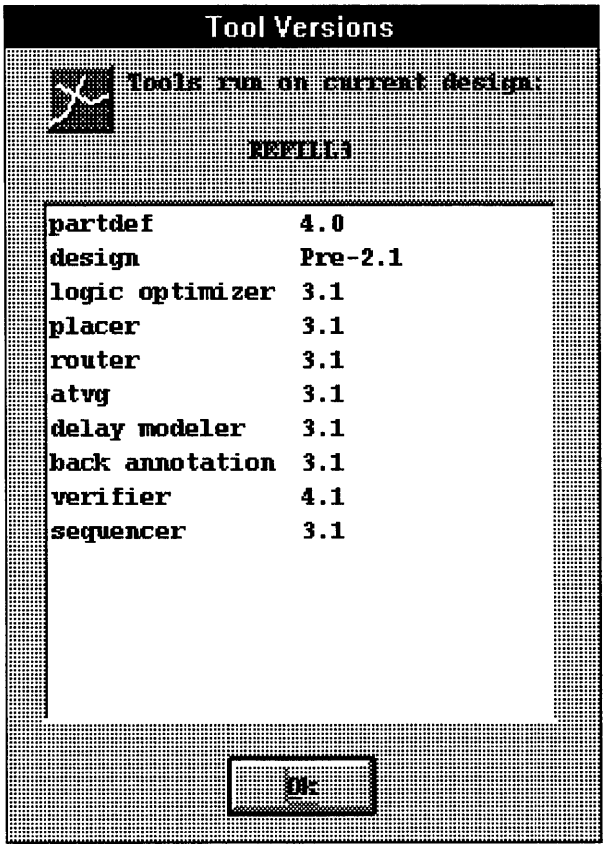
Input/Output:	6 of 32
Input-only:	0 of 6
Clock-only:	1 of 2
Percent Utilized:	17.5%

Figure 1-8. Typical Cell Utilization Dialog Box


ATVG Coverage

Total Test Vectors:	9
Stuck at Zero (SA0) Faults:	40
Stuck At One (SA1) Faults:	51
Fault Grading:	92.0%

Figure 1-9. Typical ATVG Coverage Dialog Box



Tools run on current design:

REFILL3

```
partdef           4.0
design            Pre-2.1
logic optimizer   3.1
placer            3.1
router            3.1
atvg              3.1
delay modeler     3.1
back annotation   3.1
verifier          4.1
sequencer         3.1
```

Figure 1-10. Typical Tool Versions Dialog Box.

1.6 The Help Menu

The Help Menu (Figure 1-11) includes commands to provide on-line help on SpDE.

Index invokes Help with SpDE's on-line help.

Macro Library brings up help on using the macro library cells.

Using Help provides introductory information on using the Help facility itself.

About SpDE provides information on the SpDE Toolkit, including the revision number of each tool and the configuration.

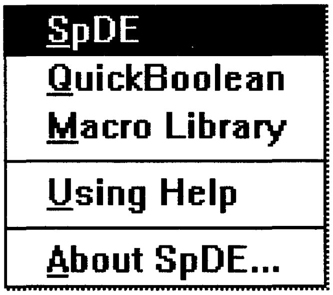


Figure 1-11. Help Menu.

NOTE

Cypress Semiconductor's *Warp* products do not support QuickBoolean.

Chapter
2
SpDE Design Tools

About This Chapter

This chapter describes the five SpDE design tools:

1. Logic Optimizer;
2. Placer;
3. Router;
4. Delay Modeler/Back Annotation;
5. Automatic Test Vector Generator.

2.1. Introduction

This chapter describes the five SpDE design tools: (1) Logic Optimizer; (2) Placer; (3) Router; (4) Delay Modeler/Back Annotation; and (5) Automatic Test Vector Generator.

The Logic Optimizer partitions designs into logic cells, using sophisticated technology mapping algorithms.

The Placer takes the design from the Logic Optimizer and places the logic cells in optimal locations on the chip.

The Router connects I/O and logic cells, using the pASIC interconnect resources.

The Delay Modeler calculates delays for use by Path Analyzer, and writes the delays and delay scale to a file for use by the Viewsim simulator.

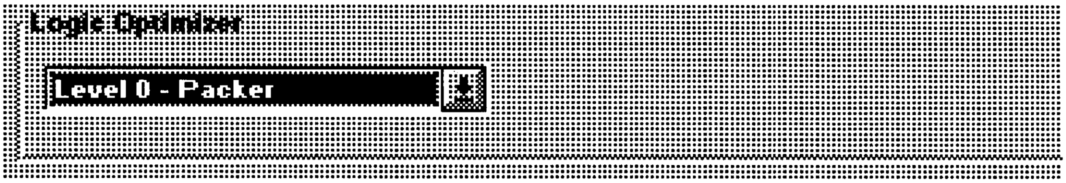
The Back Annotation tool writes pin numbers and fixed flip-flop numbers to a file that Cypback uses to back-annotate schematics.

The Automatic Test Vector Generator generates test vectors that can be used to test pASIC devices after they have been programmed.

2.2. Logic Optimizer

The Logic Optimizer is the first tool to be run after a design netlist has been loaded into SpDE. The Logic Optimizer uses sophisticated technology mapping algorithms to partition the design into logic cells.

There are two levels of optimization available in the Logic Optimizer: Level 0 - Packer, and Level 1 - Technology Mapper. The optimization levels can be selected from the SpDE Tools Options window (Figure 2-1), which comes up when you select Options/General from the Tools menu.


Logic Optimizer

Level 0 - Packer

Figure 2-1. SpDE Tools Options Window: Optimizer Options.

Level 0 Optimization: The Packer

The Packer "packs" logic into logic cells, but always leaves the original nets in the design intact. While the Level 1 Optimizer (the Technology Mapper) is the preferred logic optimizer in almost every case, Level 0 optimization is provided for versatility and compatibility with old designs.

Level 1 Optimization: The Technology Mapper

The Technology Mapper uses a more sophisticated algorithm than the Packer. Consequently, the Technology Mapper sometimes takes longer to run. For example, the Packer never takes more than a few seconds to run, but the Technology Mapper can take from a few seconds to several minutes, depending on the complexity of the design. This optimization level changes a design to use optimal gate selection (e.g., NOR vs. OR, etc.).

NOTE

Internal nets may be deleted as a result of Level 1 optimization.

Logic Optimization Modes

For Level 1 optimization, you may choose Preliminary, Quality, or Overnight mode from the SpDE Tools Options window. (Level 0 optimization is a simple, predictable algorithm that does not require different modes.)

Preliminary Level 1 optimization completes in about half the time of the Quality mode.

Quality Level 1 optimization is recommended for high-quality results.

Overnight (or Exhaustive) mode produces slightly better results than Quality mode on some designs, but with a significantly longer run time.

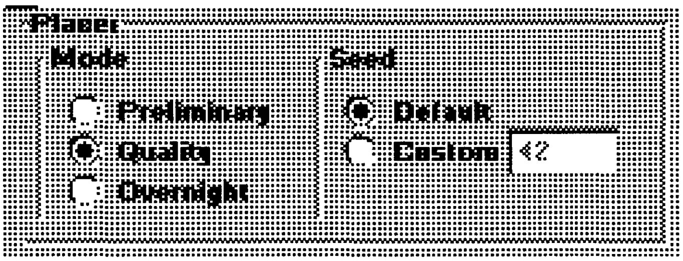
2.3. Placer

The Placer places cells output from the Logic Optimizer, using internal algorithms. For placing critical-path elements, use timing-driven placement from the Path Analyzer.

When run from the Path Analyzer, the Placer determines optimal locations by looking at the nets connecting logic cells, and by looking at timing constraints added by the designer. (See "Timing-Driven Placement," later in this section.)

Placer Options

Figure 2-2 shows the dialog box by which you select Placer options. Two kinds of options are available: you can select the seed value for the Placer, or the placement mode, or both.



Placer

Mode: Preliminary [✓ Quality] Overnight

Seed: [✓ Default] Custom 42

Figure 2-2. Placer Options.

Placer Seed - The placement seed initializes the placement process and sets a starting point for the decisions made during automatic placement. The seed for the Placer is a integer between 1 and 32767. You can use a custom seed, or the default seed value (42). Changing the seed value sets a different starting point for the placer, which can produce a slightly different (and possibly improved) placement.

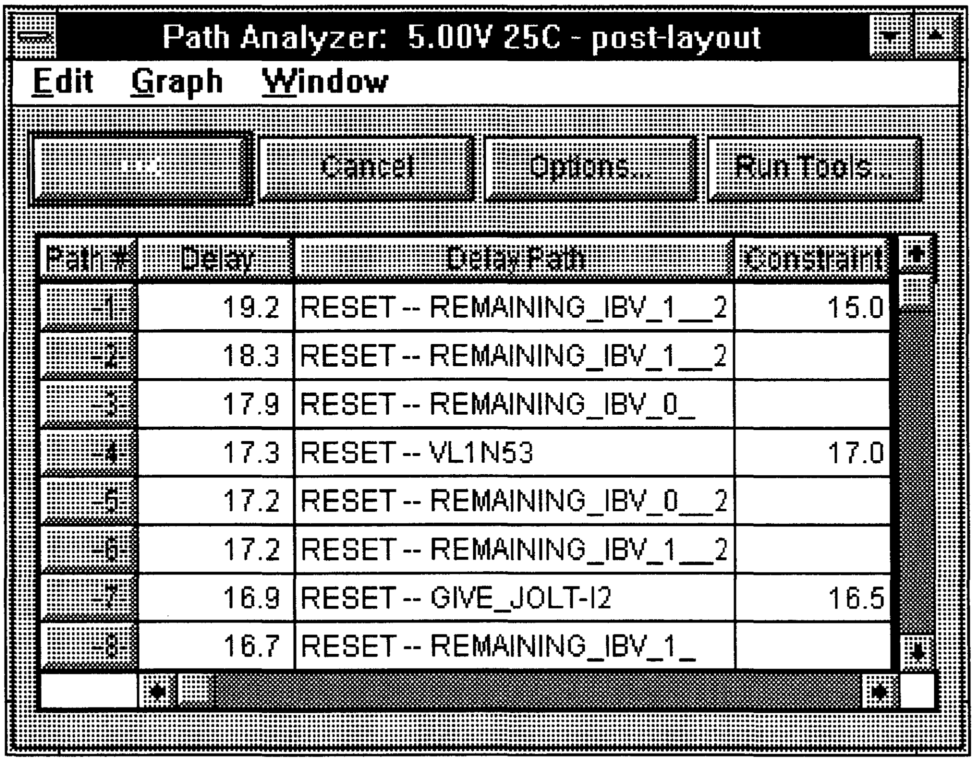
Placement Mode - Three placement modes are available: Preliminary, Quality, and Overnight. These three modes have the following characteristics:

- **Preliminary** ("Get a cup of coffee") placement is faster than quality placement, but usually by only a few minutes. Results are not as predictable and usually not as good as Quality placement. Cypress recommends that you place designs using at least Quality mode before programming chips.
- **Quality** ("Go to lunch") placement is the default placement mode. As its name implies, it produces high-quality placements.
- **Overnight** ("Go home") placement exists primarily for the curious, or just in case you're going home for the evening and want to keep your computer busy. Results of Overnight mode placement are usually about 4% better than Quality placement, but at a significant cost in run time (about 10X). Thus, a design that places in six minutes in Quality mode will take about an hour in Overnight placement mode.

Timing-Driven Placement

The Placer works closely with the Path Analyzer to provide timing-driven placement, an advanced technique that has been used in gate array placement to produce optimal results. User-specified constraints are fed from the Path Analyzer to the Placer. Paths not meeting the specified constraints are automatically boosted in priority until the constraint is met. Timing-driven placement allows the user to obtain peak performance without resorting to fixed placements.

Constraints can be entered for these paths directly in the Path Analyzer. Once the Path Analyzer has been run, paths not meeting the desired goal can be easily identified.



Path #	Delay	Delay Path	Constraint
1	19.2	RESET -- REMAINING_IBV_1__2	15.0
2	18.3	RESET -- REMAINING_IBV_1__2	
3	17.9	RESET -- REMAINING_IBV_0_	
4	17.3	RESET -- VL1N53	17.0
5	17.2	RESET -- REMAINING_IBV_0__2	
6	17.2	RESET -- REMAINING_IBV_1__2	
7	16.9	RESET -- GIVE_JOLT-I2	16.5
8	16.7	RESET -- REMAINING_IBV_1_	

Figure 2-3. Using Timing Constraints in the Path Analyzer.

It is important to set the constraints realistically—set each constraint at or just slightly below the required value. One of the keys to timing-driven placement is the concept of "good enough." Once a critical path has met its constraint, the Placer boosts the priority elsewhere in order to optimize all critical paths.

For each path with a constraint, the Placer estimates the delay throughout the placement process. If a constraint is met, the Placer continues to optimize the nets in the path normally. If a constraint is not met, the Placer boosts the priority of the nets in the paths; the boost in priority is proportional to the difference between the constraint and the estimated value. In other words, paths near their constraints are boosted less than paths far from their constraints.

Add constraints only where required. The dynamic delay estimation mentioned above adds work to the placement process. Each constraint specified slows the placement process.

Constraints are stored with the design database. Once the constraints have been specified, all subsequent Placer runs operate in timing-driven mode. This can be verified during placement from the **SpDE Status** window—under normal placement the heading is "Placer," while under timing-driven placement the heading is "Timing-Driven Placer."

NOTE

Constraints are not saved to a separate file, so each time you import a QDIF file into SpDE, all constraints are cleared.

Fixed Placement

Although the Placer automatically determines placement for logic cells and I/O pads, it also supports fixed assignments of both I/O and flip-flops when required. As the name implies, fixed assignments made by the designer are not modified by the Placer.

Fixing the Placement of I/O Pads

Design constraints sometimes require some or all I/O cell locations to be fixed. For example, an existing printed circuit board (PCB) might dictate a precise pinout. Alternatively, a high-speed PCB might require fixing a small number of critical pins in order to limit skew. The SpDE Placer can handle these cases.

To fix I/O pad (i.e., pin) placements, use the pin_numbers attribute in *Warp* VHDL. (See the *Warp Synthesis Compiler Reference Manual* for syntax information about this attribute.)

Fixing the Placement of Flip-Flops

Design constraints rarely require logic cell locations to be fixed. However, to allow designers a greater degree of flexibility, the Placer allows some or all of a pASIC's flip-flop macros to be fixed. One scenario that would dictate fixed flip-flops would be where all the bits of an 8-bit register need to appear on the output pins with absolute minimum skew. The Placer, not realizing this design constraint, might sacrifice the skew on the outputs in order to produce a circuit that was faster overall. By manually fixing the flip-flops on logic cell locations adjacent to the output pins, the designer can meet the design constraint.

To fix flip-flop (i.e., internal) placements, use the fixed_ff attribute in *Warp* VHDL. (See the *Warp Synthesis Compiler Reference Manual* for syntax information about this attribute.)

NOTE

In order for placement to proceed correctly, you must apply the fixed_ff attribute to each element of a bus, and not to the bus as a whole.

Keep in mind that two flip-flop macros cannot be assigned to the same location. The naming convention for pASIC logic cells assigns a character to each column and a decimal number to each row. SpDE verifies the uniqueness of location assignments for fixed placement with the Design Verifier.

Locking Down a Previous Placement

It is sometimes necessary to “lock down” an I/O placement or a logic cell placement. This means that for all subsequent place and route runs, you do not want the I/O pins and/or the logic cells to change their locations.

Use the SpDE back annotation tool and CypBack to fix I/O or logic cell placements, using the following steps:

1. Select the appropriate options for back-annotation in the SpDE Tools Options window and click OK.
2. Run the Back-Annotation Tool in SpDE (creates, writes out placement information to .ATR file).
3. Run CypBack after exiting SpDE (back-annotates

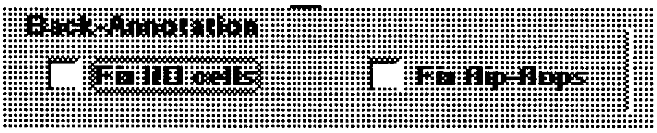


Figure 2-4. Back Annotation Options

placement information to schematic).

NOTE

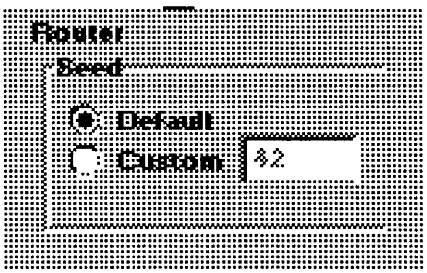
The back-annotator only writes to the schematic, not to the VHDL file.

2.4. Router

The Router employs highly optimized algorithms to connect I/O and logic cells using the pASIC interconnect resources. This finely tuned arrangement produces excellent performance with high utilization.

Seed Value

Figure 2-5 shows the area of the SpDE Tools Options window that allows you to set the seed value for the Router. The seed value may be an integer between 0 and 32767, inclusive. If the router seed is changed, the resulting route may be slightly different. This option is rarely needed, but is provided for versatility.



Router

Seed

[✓ Default]

Custom 42

Figure 2-5. Router Options.

Interconnect Resources

The Router uses four different types of routing resources for fast and efficient connection between logic cells and I/O pads. These resources are **clock networks**, **express wires**, **quad wires**, and **segmented wires**.

Clock Networks: Two clock networks exist on each pASIC device. Both of these dedicated resources are capable of connecting to the clock, set, or reset of any flip-flop in the pASIC device. Each clock network must be driven by one of the clock cells (CKPADs) located at specific pins depending on the package used. (Notice the pins labeled CLK in the device pinout appendix.)

Segmented Wires: Segmented wires are the most abundant routing resource. These wires traverse the distance of one logic cell. High-drive pads cannot drive segmented wires, so the Router restricts nets on High-Drive pads to be routed on quad or express wires.

Quad Wires: Quad wires span four times the distance on the chip that segmented wires do (four logic cells). Quad wires may be used in routing any net in the design, including nets driven by High-Drive pads and parallel logic (see "Special Routing Cases," below).

Express Wires: Express wires span the entire length or height of the pASIC device. They are used for high-fanout nets, or nets that need to travel across the device.

Special Routing Cases

High-Drive Pads: High-Drive Pads (HDPADs) must drive either quad wires or express wires. On devices that do not have quad wires, high-drive pads must drive express wires.

Parallel Logic: The pASIC architecture allows quad or express wires to be driven from higher-drive sources, such as HDPADs or parallel logic. Parallel logic is a logic configuration in which two identical gates (with the same inputs) have their output nets attached for higher drive capability. There is a restriction on the type of gates that can be tied in parallel. For more information, refer to the **Design Techniques** chapter of this manual and its discussion on **double-buffering**.

SpDE warns you if you use more than the recommended limit of high-drive nets (nets driven by high-drive pads or parallel logic). The router may have difficulty completing successfully in these cases.

2.5. Delay Modeler and Back Annotation

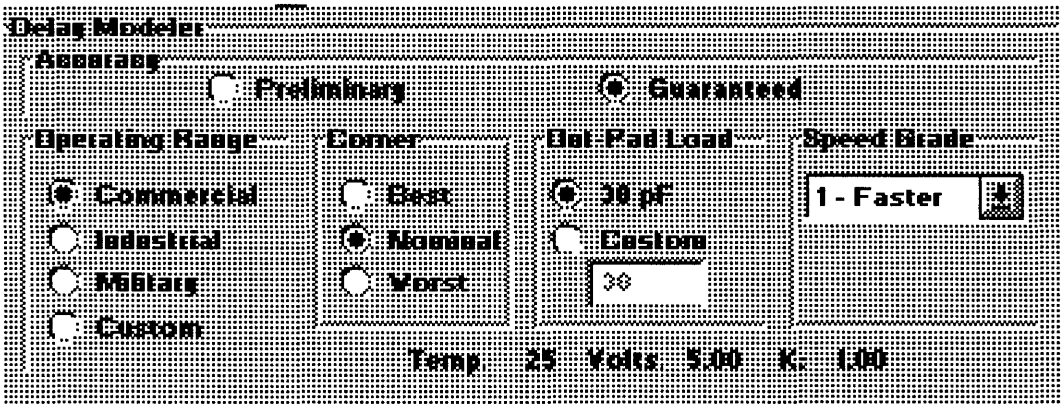
The Delay Modeler and Back Annotation tools are used to calculate the specific timing delays in the pASIC device and to send these timing numbers to a simulator for back-annotated simulation.

Delay Modeler

The Delay Modeler performs a comprehensive timing analysis, accounting for load, slew rate, signal propagation, and intrinsic delay. The tool uses a precise model of the pASIC device and calculates the effects of fanout, packing, placement, and routing.

The Delay Modeler can perform best-case, nominal, or worst-case analysis. The results of the worst-case analysis account for process variation, temperature, and voltage.

The Delay Modeler may be run in **Preliminary** or **Guaranteed** mode (see Figure 2-6). In Preliminary mode, the Delay Modeler uses statistical estimates for the impedance of ViaLinks in the device. In Guaranteed mode, the more accurate ViaLink impedances calculated by the Sequencer are used.



Delay Modeler

Accuracy: Preliminary [✓ Guaranteed]

Operating Range: [✓ Commercial] Industrial Military Custom

Corner: Best [✓ Nominal] Worst

Out-Pad Load: [✓ 30 pF] Custom 30

Speed Grade: 1 - Faster

Temp 25 Volts 5.00 K: 1.00

Figure 2-6. Delay Modeler Options.

If your machine does not have a math co-processor, the run time of the Delay Modeler may be prohibitive to use while you are debugging your design. In that case, use the Preliminary setting. Once the design is stable, use the Guaranteed setting to ensure proper timing performance.

The **Operating Range** radio button group controls the voltage and temperature ranges used by the Delay Modeler. The default setting is **Commercial**. The **Custom** setting allows a user-specified temperature and voltage to be employed. See "Custom Temperature and Voltage," later in this section.

The **Corner** radio button group selects the corner of the selected operating range. The default is **Nominal** (25 degrees Celsius and 5 volts, regardless of the operating range selected). **Best** selects the lowest temperature and highest voltage in the selected operating range; **Worst** selects the highest temperature and lowest voltage in the operating range. Simulation should be performed at the **Worst** corner.

The **Speed Grade** radio button group selects the pASIC speed grade to be analyzed.

The **Out-Pad Load** radio button group selects the capacitive loading on the output pins. The default is 30 pF. The Custom setting allows a user-specified load in the range of 0 pF to 150 pF to be employed for all output pins.

NOTE

The Delay Modeler has been tuned for peak accuracy within the recommended fanout ranges. High-fanout nets that produce fanout warnings are calculated to the highest accuracy possible, but these results are not guaranteed.

Custom Temperature and Voltage

To change the temperature and voltage setting for the Custom operating range, you must edit the spde.ini file located in *install_directory*\spde\data. For the SUN platform, this file should be named .spderc, and should be in your home directory. The following lines should be changed:

PC: spde.ini

```
[delay modeler]
...
CustomVCCBest=5.0
CustomVCCNominal=5.0
CustomVCCWorst=5.0
CustomTempBest=25.0
CustomTempNominal=25.0
CustomTempWorst=25.0
...
```

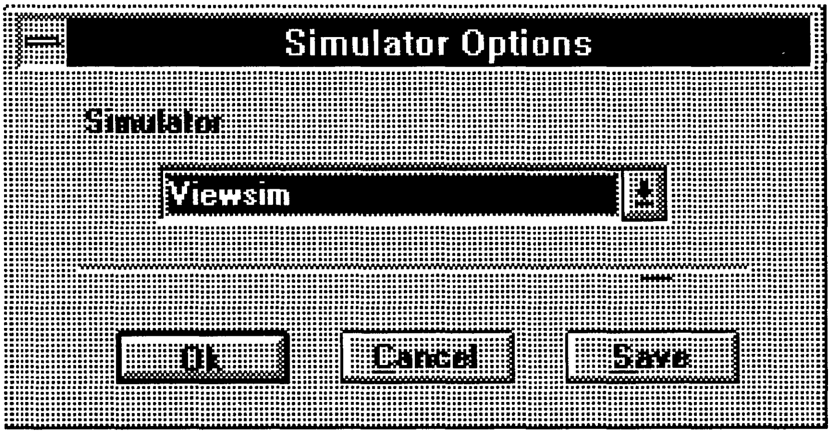
SUN: .spderc

```
...
delay modeler.customvccbest=5.0
delay modeler.customvccnominal=5.0
delay modeler.customvccworst=5.0
delay modeler.customtempbest=25.0
delay modeler.customtempnominal=25.0
delay modeler.customtempworst=25.0
...
```

The measurement units for CustomVCC variables are Volts. The measurement units for CustomTemp variables are Celsius degrees. The voltage range should not vary more than +/- 10% from nominal (5V). The Best, Nominal, and Worst extensions of these variables represent best, nominal, and worst process factors for the devices. SpDE selects which variable to use, based on the Corner setting from the SpDE Tools Options window.

Back Annotation

The Back Annotation tool produces files to send timing and placement information back to the design entry and simulation tools. A variety of simulation tools are supported for back-annotated simulation. The simulator can be selected in spDE by selecting the Options/Simulator items from the Tools menu (see Figure 2-7).



Viewsim

Figure 2-7. Simulator Options for Back Annotation.

If you change the default simulator by making a selection from the Simulator Options window, click on the Save button to write this information into the spde.ini file (.spderc for SUN workstations). Table 2-1 lists the files created by the Back Annotation tool for each of several simulator settings.

NOTE

After you change the simulator, you must still run the Back Annotation tool to create the simulation netlist.

The Verilog simulator also requires a primitive file, which describes the functionality of the primitive components specified in the *design*.v file. This primitive file is design-independent, and is shipped with SpDE. The filename for this primitive file is *pasic_directory*\spde\data\qlprim.v, where *pasic_directory* is the directory where the pASIC Toolkit is installed.

Table 2-1. Back-Annotation Files

Simulator Setting	Files Created	Function
Verilog	*design*.v *design*.sdf	verilog netlist delay back annotation file
ViewSim	*design*.vl *design*.dtb *design*.var	intermediate file for spde2vl delay back annotation file variable values for .dtb file

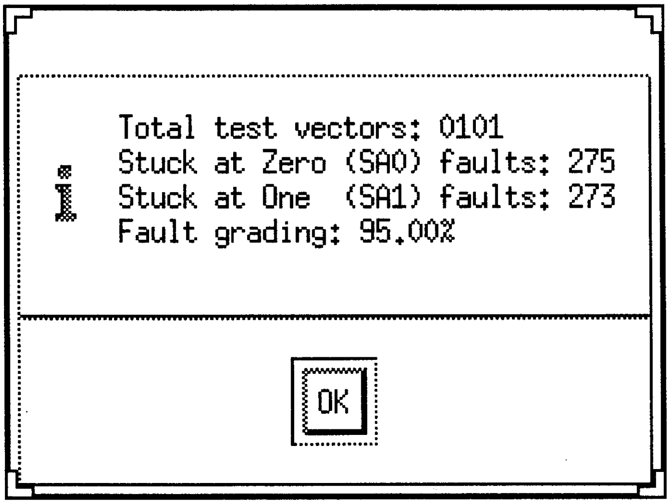
When back annotating to Viewsim, the spde2vl program must be run after back annotation to create the Viewlogic .vsm file needed for simulation.

The Back Annotation tool also supports the back-annotation of fixed placement information back to the source design. For information on this process, see Section2.3.

2.6. Automatic Test Vector Generator

The Automatic Test Vector Generator (ATVG) generates vectors that can be used on devices after they have been programmed.

The ATVG tools makes use of an internal scan path in pASIC devices that allows values to be applied to and read from each flip-flop on the device. Once the ATVG tool has been run in SpDE, fault coverage can be ascertained by selecting ATVG Coverage from the Info menu.



```
Total test vectors: 0101
Stuck at Zero (SA0) faults: 275
Stuck at One  (SA1) faults: 273
Fault grading: 95.00%
```

Figure 2-8. Sample ATVG Coverage Window (Sun version).

Testing Overview

A major problem encountered in testing FPGA's is the inability to directly access the vast majority of circuit nodes from the chip periphery. Internal faults, therefore, must be made observable at the output pins by creating a set of input stimuli that will exercise the appropriate path and cause faults to appear as invalid output level changes.

Stuck-At Faults

Stuck-at-0 (SA0) and stuck-at-1 (SA1) fault analysis is an effective means of evaluating test sequences for their ability to detect potential faults in circuits. SpDE's ATVG tool uses an advanced testing technique capable of detecting these faults.

A SA0 fault results from a condition that holds a given signal at a logical 0 regardless of the signal being asserted on that line. Similarly, a SA1 fault is a line that is held at logical 1 regardless of the asserted signal. To illustrate this concept, consider the simple case of the 3-input AND gate shown in Figure 2-9. The truth table for this function is given in Table 2-2.

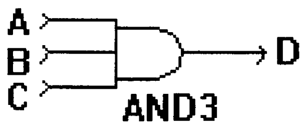


Figure 2-9. Stuck-at-Fault Example.

Table 2-2. Example Truth Table

A	B	C	D	Vector	Tests
0	0	0	0	1	
0	0	1	0	2	
0	1	0	0	3	
0	1	1	0	4	A, D for SA1
1	0	0	0	5	
1	0	1	0	6	B, D for SA1
1	1	0	0	7	C, D for SA1
1	1	1	1	8	A, B, C, D for SA0

There are eight potential faults in this circuit, representing each node (A, B, C, D) stuck-at-0 and stuck-at-1. It is apparent that if A or B or C were stuck at logical low, D would also assume logical low, indicating an incorrect state for the last state (vector 8) of the truth table, and hence a fault. Similarly, stuck-at-1 states for either A or B or C would show up as invalid outputs in vectors 4, 6, or 7 of the truth table, respectively.

By applying the appropriate inputs to the circuit, SA0 and SA1 faults may be detected at the output node D. These input stimuli, along with their expected responses, are listed in Table 2-3. These input conditions and their resultant output states are commonly referred to as test vectors. For each test vector in Table 2-3, the fault or faults detected are indicated.

Table 2-3. Test Vector Table

A	B	C	D	Vector	Tests
0	1	1	0	1	A, D for SA1
1	0	1	0	2	B, D for SA1
1	1	0	0	3	C, D for SA1
1	1	1	1	4	A, B, C, D for SA0

As this example demonstrates, a total of four test vectors are required to find all the potential faults in this simple AND gate example. Actually, SA0 and SA1 faults for node D are indistinguishable from faults in the input signals A, B, and C. From a functionality point of view, this is not important since sufficient information is generated to confirm correct or incorrect operation of the circuit.

Fault Grading

Fault grading is a quantitative measure of the testability of a circuit, and is defined by the following expression:

$$FG = \frac{\text{SA0 faults detected} + \text{SA1 faults detected}}{\text{total number of detectable faults}}$$

In this example, the total number of detectable faults is six (the total number of detectable faults is simply two times the number of inputs). Note that faults at the output D are not detectable because they are “covered” by faults at the inputs. The actual number of potential faults detected by these test vectors is also six, resulting in 100% fault coverage. Furthermore, these test vectors comprise an optimum set required to test the sample circuit, even though the truth table for it has the eight vectors shown in Table 2-2.

Design Considerations

The pASIC’s testability features allow the designer to achieve a high degree of fault coverage. Testability can be further increased by designing with a few simple rules in mind.

1. Avoid combinatorial loops.

   Combinatorial loops, such as those shown in Figure 2-10, cannot be tested by the ATVG tool. Logic driven by or driving these loops will therefore be untestable also.

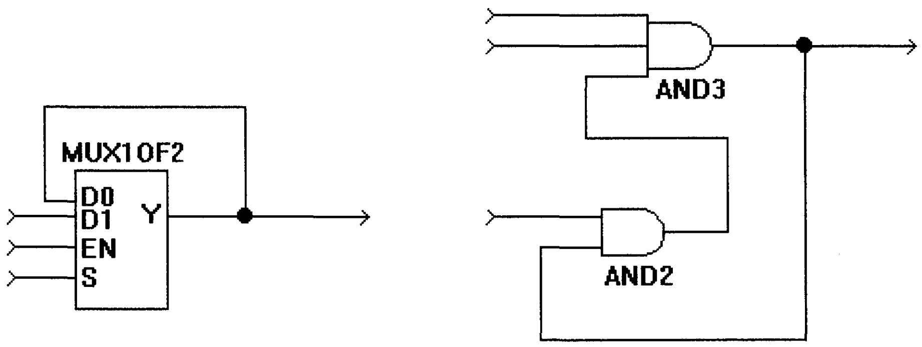


Figure 2-10. Examples of Combinatorial Loops (Feedback).

2. Using the output of a gate or flip-flop to clock, clear, or preset another flip-flop reduces testability.

   The ATVG tool tries to use all flip-flops in the pASIC for testing purposes. Clocking, setting, or clearing a flip-flop by internal logic renders the flip-flop useless to ATVG, reducing testability. Examples of logic structures that reduce testability in this way are shown in Figure 2-11. Although gated clocks reduce testability, a buffer or inverter in the path between an input pad and the clock of the flip-flop does NOT reduce testability. Notice also in Figure 2-11 that a flip-flop clock, clear, or set driven by a bi-directional piN reduces fault coverage.

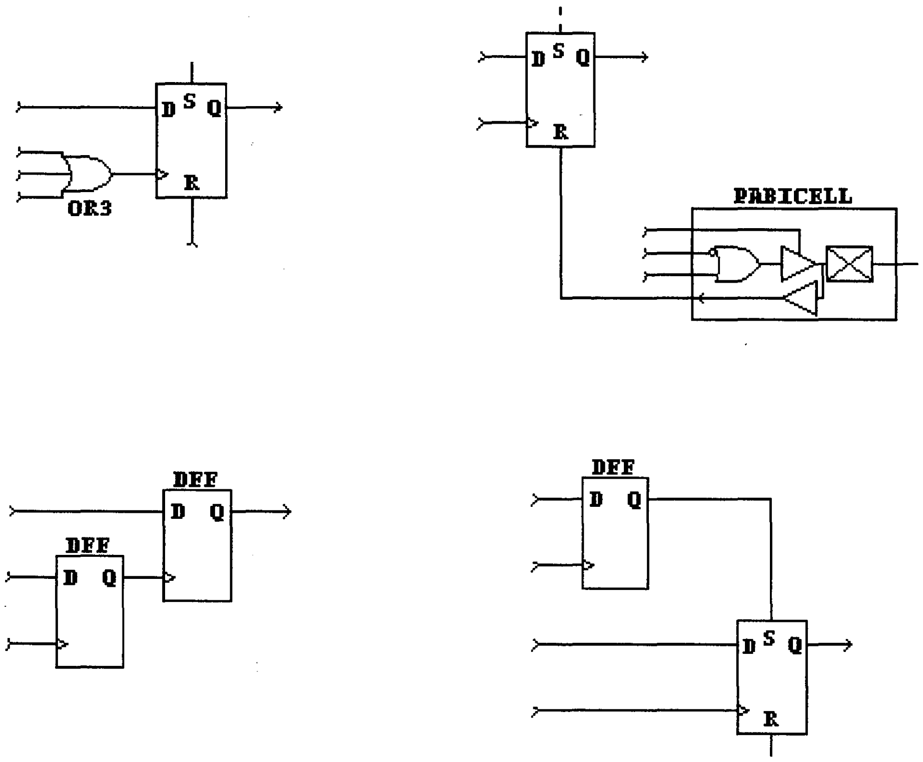


Figure 2-11. Examples of CLK, set, and reset controls that reduce testability.

3. Certain input-only pins (HDPADs) should not be used to drive logic that controls an asynchronous set or reset of a flip-flop, as doing so either disables ATVG or reduces coverage.

   A subset of the input-only pins (HDPADs) should not be used to drive asynchronous sets or resets (of flip-flops) directly or through a logic path. If so, ATVG will be disabled or fault coverage will be reduced. Table 2-4 shows the pin numbers for different packages.

Table 2-4. Input pins with restrictions in setting and resetting

Package	Input-pins (HDPADs) with restrictions driving sets or resets.	
	Disables ATVG	Lowers Coverage
44 pin PLCC	11	10
68 pin PLCC	17	16
68 pin CPGA	A7	B7
84 pin PLCC	22	21, 66
84 pin CPGA	C6	B7
100 pin TQFP	12	11, 65
144 pin TQFP	18	17, 93
144 pin CPGA	C8	B8, P7

Because of the test mode requirements of pASIC devices, certain input-only (also known as high-drive) pins cannot be used in a multiple HDPAD configuration without disabling ATVG, unless they are used only to drive flip-flop clock inputs. Table 2-5 lists the high-drive pins that cannot be used in HD2PADs or HD3PADs without disabling ATVG.

Table 2-5. Input pins with restrictions on HDxPADs

Package	Pins not to include in HD2PADs or HD3PADs, unless used as clock only (disables ATVG)
44 pin PLCC	10, 36
68 pin PLCC	16, 54
68 pin CPGA	B7, K7
84 pin PLCC	21, 66
84 pin CPGA	B7, K7
100 pin TQFP	11, 65
144 pin TQFP	17, 93
144 pin CPGA	B8, P7

Chapter 3 SpDE Analysis Tools

About This Chapter

After a design has been placed and routed, it is often necessary to analyze timing paths carefully to determine the speed of the design. SpDE contains two tools for this purpose—**Highlight Net** and the **Path Analyzer**.

3.1. Highlight Net

Highlight Net mode helps you analyze a design by highlighting and unhighlighting nets. Highlight Net mode serves no purpose until a design has been placed and routed.

To open the Highlight Nets window (Figure 3-1), select the **Highlight Net...** item from the View menu in SpDE.

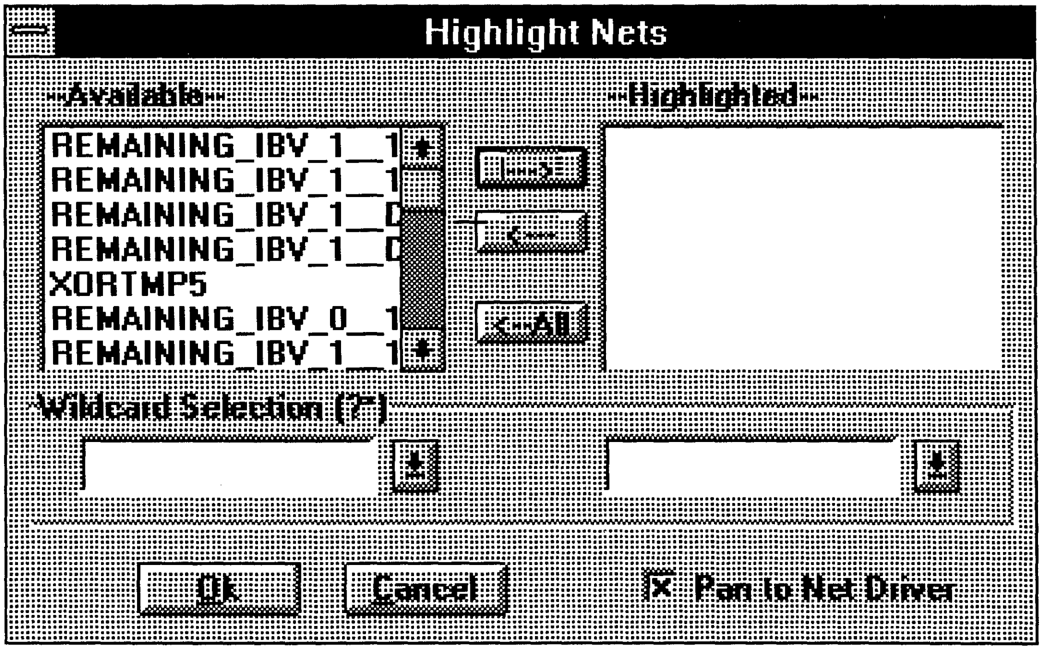


Figure 3-1. Highlight Nets Window.

To highlight nets, select one or more nets from the box on the left, or specify a net name in the "Wildcard Selection" field on the left side of the dialog box, and click on the right arrow button.

To remove nets from the highlight list, select one or more nets from the box on the right, or specify a net name in the “Wildcard Selection” field on the right side of the dialog box, and click on the left arrow button.

When using the “Wildcard Selection” fields, the wildcard characters “*” and “?” are accepted. The “*” character matches 0 or more occurrences of any character. The “?” character matches a single occurrence of a single character.

Double-clicking on a net name in either list moves it to the other list.

All nets can be removed from the highlight list by clicking on the ALL button.

Once in highlight net mode, the highlight status may be toggled by clicking directly on the desired nets in the physical view.

To exit highlight net mode, click Cancel. The Physical View will be redrawn in the normal mode.

Double-clicking on a net in the Highlighted box un-highlights the net in the physical view. Clicking on a highlighted wire in the physical view un-highlights the net.

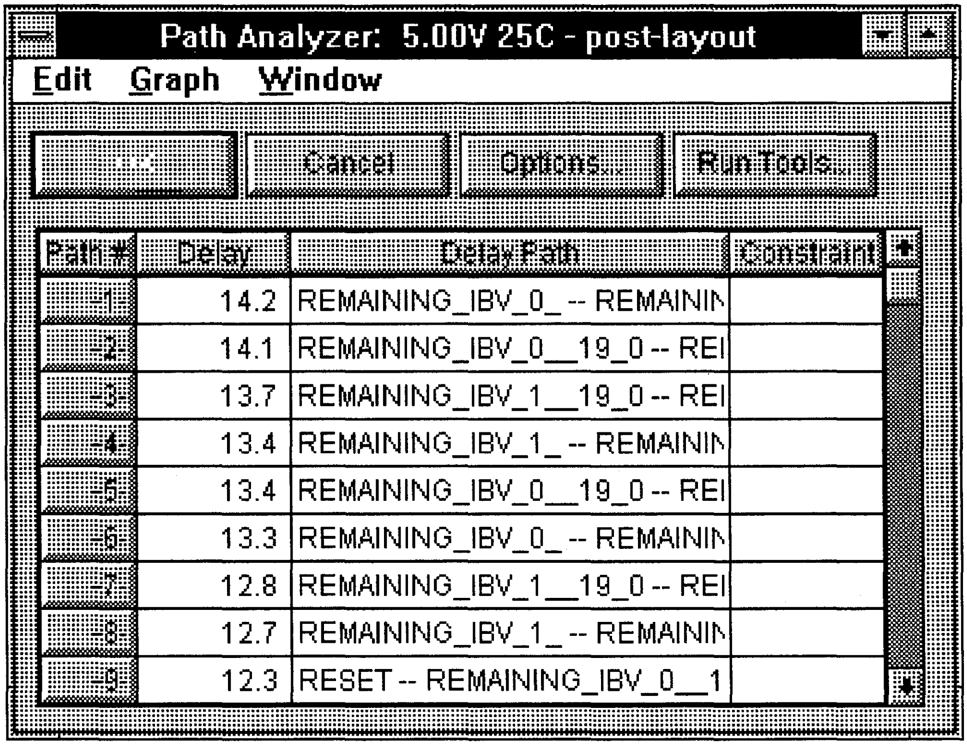
Pan to Net Driver

The Highlight Nets window contains a check box named **Pan to Net Driver**. When this box is checked, SpDE automatically pans to the driver of the net that is selected. This is true whether the net is selected by clicking on a wire in the physical view, or by selecting a net from the available list.

3.2. Path Analyzer

The Path Analyzer is a powerful static timing analyzer that can be used to determine operating frequency, setup and hold times, and clock skew. Working closely with the Physical Viewer, the Path Analyzer instantly identifies critical paths for optimization. Once the critical path has been identified, you can use the Timing-Driven Placer to optimize the placement in order to achieve specified operating constraints.

To run the Path Analyzer, select **Path Analyzer** from the Tools menu. Results are displayed in a four-column spreadsheet format (Figure 3-2).



Path #	Delay	Delay Path	Constraint
1	14.2	REMAINING_IBV_0_ -- REMAININ	
2	14.1	REMAINING_IBV_0__19_0 -- REI	
3	13.7	REMAINING_IBV_1__19_0 -- REI	
4	13.4	REMAINING_IBV_1_ -- REMAININ	
5	13.4	REMAINING_IBV_0__19_0 -- REI	
6	13.3	REMAINING_IBV_0_ -- REMAININ	
7	12.8	REMAINING_IBV_1__19_0 -- REI	
8	12.7	REMAINING_IBV_1_ -- REMAININ	
9	12.3	RESET -- REMAINING_IBV_0__1	

Figure 3-2. Path Analyzer window.

The **Path #** column is displayed in a push-button format, for reasons to be explained shortly. The **Delay** column indicates the delay in nanoseconds. For post-layout analysis, these delays are determined by the Delay Modeler. The **Delay Path** column displays the starting and ending nets of each path.

If the starting point is a pad, the net attached to the off-chip terminal on the pad will be used; if the starting point is a flip-flop, the net attached to the output of the flip-flop will be used. Likewise, if the ending point is a pad, the net attached to the off-chip terminal on the pad will be used; if the ending point is a flip-flop, the net attached to the output of the flip-flop will be used.

NOTE

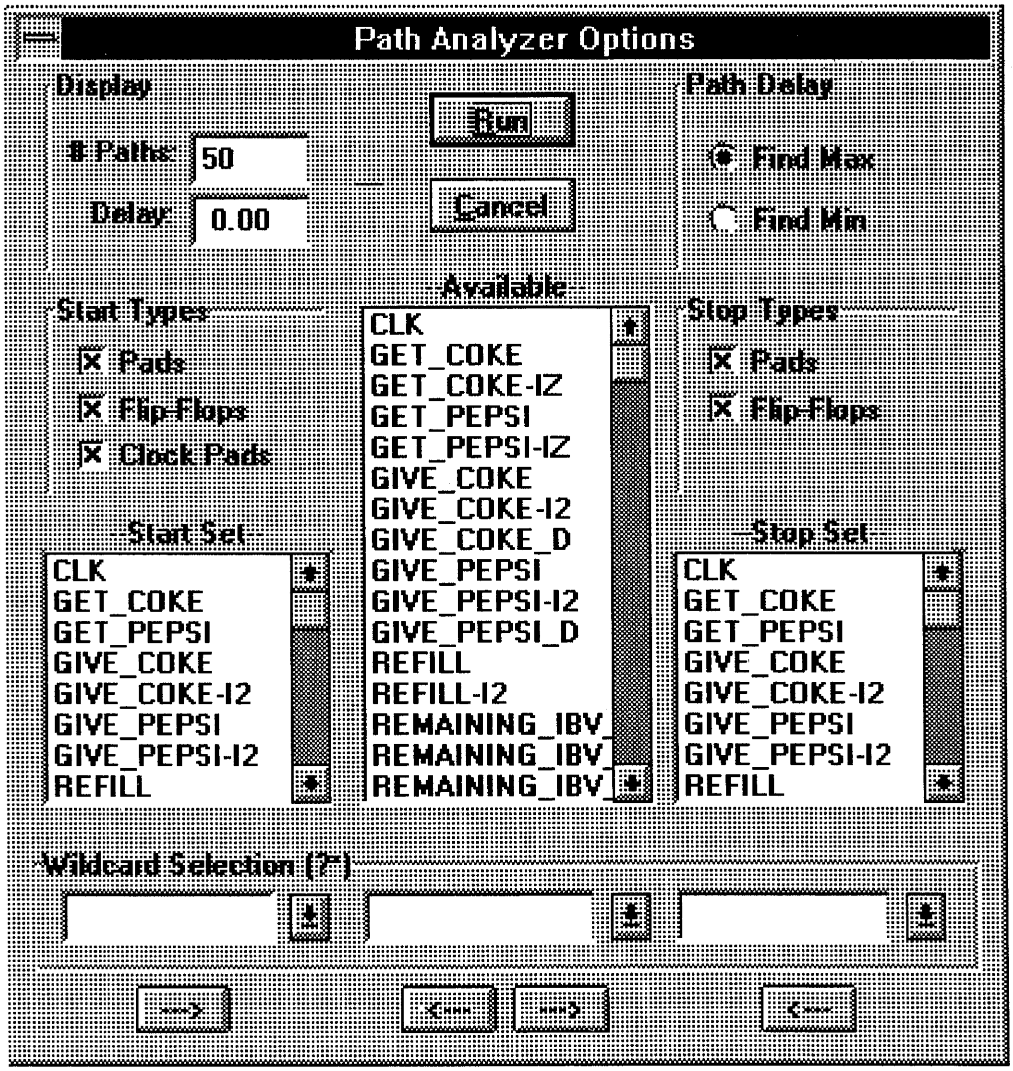
In order to be listed, I/O pads must have nets attached to their external terminals. (These "stub" nets are not required for placement and routing.) Naming these "stub" nets makes the Path Analyzer report easier to read.

Expanding Paths

To expand a path into its component trails, position the cursor over the desired button in the **Path #** column and double-click. The Path # button changes from -26 to +26+ (assuming path number "26") to indicate that the path has been expanded. The component trails are indented and listed in blue to differentiate them from the other **Delay Path** rows. Each trail lists a delay value in nanoseconds, along with an R or F token to denote a rising- or falling-edge delay.

Path Analyzer Options

All Path Analyzer options are set from the **Path Analyzer Options** dialog box (Figure 3-3), which appears when you click the **Options** button in the Path Analyzer window.



Display
Paths: 50
Delay: 0.00

Path Delay
[✓ Find Max] Find Min

Start Types
[✓] Pads
[✓] Flip-Flops
[✓] Clock Pads

Stop Types
[✓] Pads
[✓] Flip-Flops

Start Set
CLK
GET_COKE
GET_PEPSI
GIVE_COKE
GIVE_COKE-I2
GIVE_PEPSI
GIVE_PEPSI-I2
REFILL

Available
CLK
GET_COKE
GET_COKE-IZ
GET_PEPSI
GET_PEPSI-IZ
GIVE_COKE
GIVE_COKE-I2
GIVE_COKE_D
GIVE_PEPSI
GIVE_PEPSI-I2
GIVE_PEPSI_D
REFILL
REFILL-I2
REMAINING_IBV
REMAINING_IBV
REMAINING_IBV

Stop Set
CLK
GET_COKE
GET_PEPSI
GIVE_COKE
GIVE_COKE-I2
GIVE_PEPSI
GIVE_PEPSI-I2
REFILL

Figure 3-3. Path Analyzer Options Dialog Box.

The **Run** button at the top of the window re-runs the Path Analyzer with the newly specified options. The **Cancel** button returns to the Path Analyzer and discards any newly specified option selections.

The Path Delay group of radio buttons selects maximum or minimum path delays. Each trail along a given path includes a rising-edge delay and a falling-edge delay. If **Find Max** is selected, the Path Analyzer sums the larger of these edge delays at each trail; if **Find Min** is selected, the Path Analyzer sums the smaller of these edge delays. Note that this selection does not change the operating conditions. (In other words, it does not change worst-case commercial to best-case commercial.)

The **Display** group determines the number of paths calculated and listed in the path analyzer spreadsheet. The **# Paths** entry limits the number of paths to the specified value. The **Delay** entry is interpreted with regard to the **Path Delay** setting—if **Find Max** is selected, paths are listed if their delay is greater than or equal to the specified value; if **Find Min** is selected, paths are listed if their delay is less than or equal to the specified value.

The remaining lower sections of the dialog box are used to select the **Start Set** and **Stop Set** that specify the desired paths. The **Start Set** list box specifies the starting nets for path analysis, while the **Stop Set** list box specifies the ending nets for path analysis. Providing specific **Start Set** and **Stop Set** information limits the amount of data in the spreadsheet report, making it easier to interpret the results of the Path Analyzer.

The **Start Types** and **Stop Types** check boxes provide the easiest method for selecting the **Start Set** and **Stop Set** list box entries. By default, all of these check boxes are selected. The **Pads** check box selects all nets attached to the external terminals of all pads; this check box selects I/O pads, high-drive pads, and clock pads. The **Flip-Flops** check box selects all nets attached to the output terminals of all flip-flops. The **Clock Pads** check box selects all nets attached to the external terminals of any pad functioning as a clock (not only the internally buffered clocking networks).

Selecting one of these check boxes adds all of the appropriate nets to the desired set. De-selecting one of these check boxes removes all of the appropriate nets from the desired set. For example, assume none of the **Start Types** check boxes are selected. Selecting the **Pads** check box adds all pad nets to the **Start Set** list box. Selecting the **Clock Pads** check box results in no change, as all pad nets are already selected. De-selecting the **Clock Pads** check box, however, removes the clock pad nets from the **Start Set** list box, leaving only non-clock pad nets.

Nets can be selected manually using the **Available** list box in the center of the dialog box. Select a net or nets in this list box, then click on one of the arrow buttons below the **Available** list box. Clicking on the left-arrow button adds the selected nets to the Start Set list box, while clicking on the right-arrow button adds the selected nets to the **Stop Set** list box.

Likewise, the **Start Set** and **Stop Set** list boxes can be “pruned” by selecting a net or nets and clicking on the arrow button below the list box involved.

Groups of nets can be selected using the combo buttons below each list box. In Figure 3-3, for example, the bus IB[0:3] can be selected by cliking in the combo button just below the **Available** list box and typing IB* or IB[?] and pressing the **Tab** key. Once the desired nets are selected, they can be acted upon using the arrow buttons, as described previously.

NOTE

When entering text to select nets from the **Start Set**, **Available**, or **Stop Set** lists, wildcards can be used. An asterisk ("*") represents 1 or more characters; a question mark ("?") represents a single character (e.g., *addr** would select *addr[0]*, *addr[1]*, *addr[2]*, etc.

Graphing

The Path Analyzer provides essential information about the performance of the design. Occasionally, it is useful to view this information in graphical form. The Path Analyzer's Graph menu can be used to create two types of graphs: **Path vs Delay** and **Delay Histogram** graphs, shown in Figures 3-4 and 3-5.

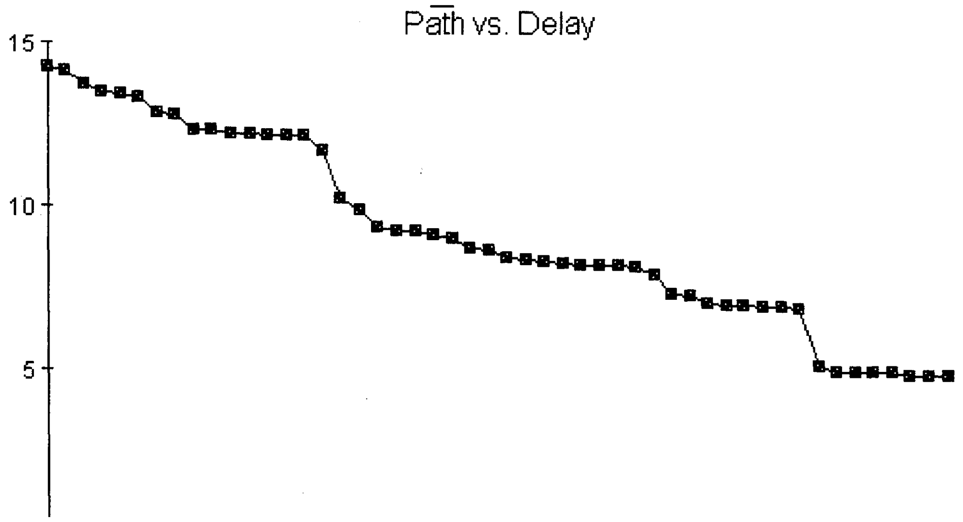


Figure 3-4. Path vs Delay Graph.

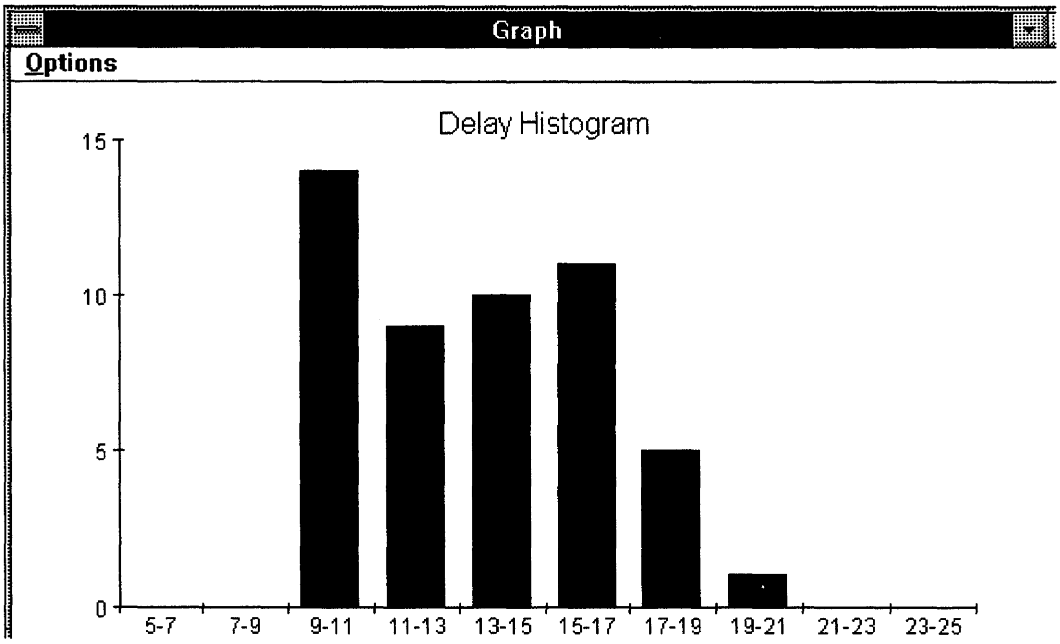


Figure 3-5. Delay Histogram Graph.

The Path vs Delay graph shows the path delays on the Y axis and the path numbers on the X axis. Double-clicking on the points in this graph has the same highlighting effect as double-clicking on the paths in the Path Analyzer.

The Delay Histogram graph uses a range of path delays as "buckets" on the X axis. The number of paths falling into a delay range "bucket" is shown as a Y value for each range.

Both graphs feature an options menu that provides the capability to copy the graph to the clipboard (as a bitmap) or to print the graph to the current printer. Also, each graph can be customized with the **Graph/Options** menu command from the main Path Analyzer window.

Key Calculations

Using the Path Analyzer, key information can be determined with simple arithmetic.

The results of these calculations will always be pessimistic, for reasons provided below. The calculations do, however, provide a quick and convenient means of determining worst-case design performance.

Clock Skew

Click the Path Analyzer's **Option** button to bring up the Path Analyzer Options dialog box.

1. Set the **Path Delay** radio button to **Find Max**.
2. In the **Start Types** check box group, activate only the **Clock Pad** check box.
3. In the **End Types** check box group, activate only the **Flip-Flops** check box.
4. Click the **Run** button to execute the Path Analyzer.
5. Make a note of the first path listed and the last path listed. (Note that the # Paths setting must be high enough to list all specified paths; in this case, it's the number of flip-flops used in the design.) The Clock Skew is given by

   SKEW = *first_path* - *last_path*

The clock skew calculation is always pessimistic, as the calculation ignores the fact that clock skew is meaningful only between flip-flops on a common path.

Operating Frequency

Click the Path Analyzer's **Option** button to bring up the Path Analyzer Options dialog box.

1. Set the **Path Delay** radio button to **Find Max**.
2. In the **Start Types** check box group, activate only the **Flip-Flops** check box.
3. In the **End Types** check box group, activate only the **Flip-Flops** check box.
4. Click the **Run** button to execute the Path Analyzer.
5. Note the delay of the first path listed. The operating frequency is given by

$$F_{max} = 1/(\mathit{first\_path} + \mathrm{SKEW})$$

The operating frequency calculation is always pessimistic, because the clock skew component is always pessimistic. This calculation also assumes that the design is fully synchronous with a single clock signal.

Setup Time

Click the Path Analyzer's **Option** button to bring up the Path Analyzer Options dialog box.

1. Set the **Path Delay** radio button to **Find Max**.
2. In the **Start Types** check box group, activate only the **Pads** check box.
3. In the **End Types** check box group, activate only the **Flip-Flops** check box.
4. Click the **Run** button to execute the Path Analyzer.

5. Make a note of the first path listed; call it *pads_to_ffs*.
6. Click the Path Analyzer's **Option** button to bring up the Path Analyzer Options dialog box.
7. Set the **Path Delay** radio button to **Find Min**.
8. In the **Start Types** check box group, activate only the **Clock Pad** check box.
9. In the **End Types** check box group, activate only the **Flip-Flops** check box.
10. Click the **Run** button to execute the Path Analyzer.
11. Note the delay of the first path listed; call it *clock_to_ffs*. The setup time is given by

$$t_{setup} = pads\_to\_ffs - clock\_to\_ffs$$

The setup time calculation is always pessimistic, because the two calculations often apply to different flip-flops.

Hold Time

Click the Path Analyzer's **Option** button to bring up the Path Analyzer Options dialog box.

1. Set the **Path Delay** radio button to **Find Min**.
2. In the **Start Types** check box group, activate only the **Pads** check box.
3. In the **End Types** check box group, activate only the **Flip-Flops** check box.
4. Click the **Run** button to execute the Path Analyzer.
5. Make a note of the first path listed; call it *pads_to_ffs*.

6. Click the Path Analyzer's **Option** button to bring up the Path Analyzer Options dialog box.

7. Set the **Path Delay** radio button to **Find Max**.

8. In the **Start Types** check box group, activate only the **Clock Pad** check box.

9. In the **End Types** check box group, activate only the **Flip-Flops** check box.

10. Click the **Run** button to execute the Path Analyzer.

11. Note the delay of the first path listed; call it *clock_to_ffs*. The hold time is given by

$$t_{hold} = clock\_to\_ffs - pads\_to\_ffs$$

This calculation will typically yield a negative number. The Hold Time calculation is always pessimistic, as the calculation ignores the fact that the two measurements are likely along different paths.

Chapter 4

Design Techniques

About This Chapter

This chapter describes several techniques for speeding up the performance of designs created by the *Warp* system's SpDE tools.

4.1. Speeding Up High-Fanout Nets

For high-fanout, timing-critical nets, designers should consider improving design performance using buffering techinques. In some cases, solutions such as paralleling or pipelining can be used.

Five techniques that can be used to improve circuit performance are described on the following pages:

1. double buffering
2. split buffering
3. selective buffering
4. paralleling
5. pipelining

4.1. Speeding Up High-Fanout Nets

4.1.1. Double Buffering

The pASIC architecture allows a net to be driven by two sources in specific cases. This is called double buffering. Using two gates to drive a high-fanout net speeds up the performance of the net dramatically.

Figure 4-1 is an example of double buffering in a schematic.

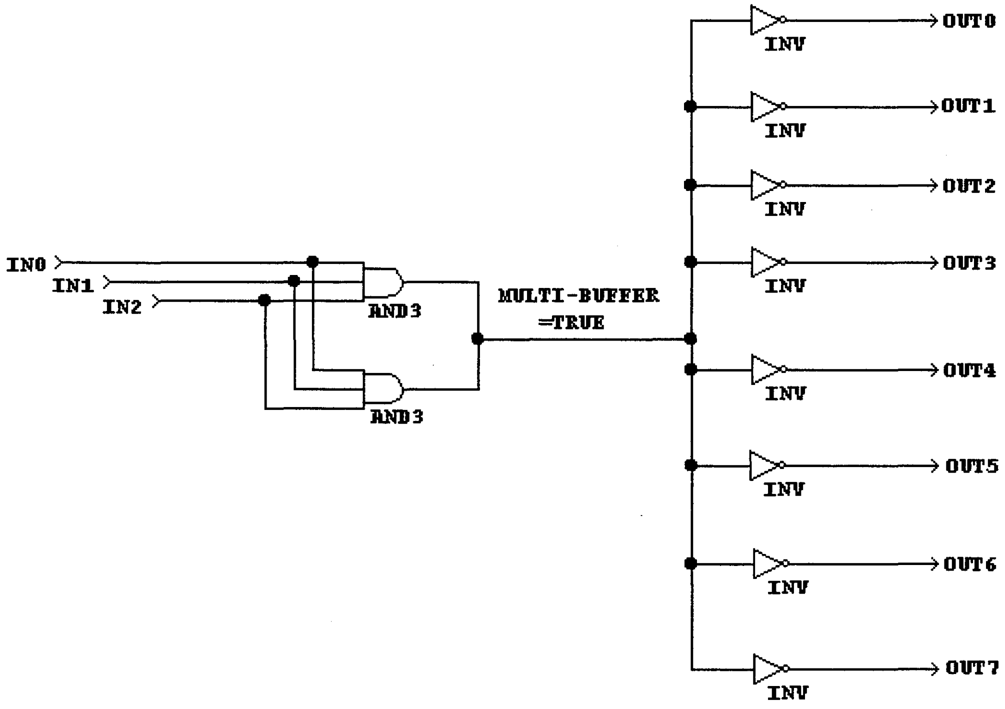


Figure 4-1. Double-Buffering Example.

Double buffering is legal as long as the two gates driving the high-fanout net are identical gates, with the same nets on the inputs and output. Each gate must fit into an AND-fragment (frag_a library element). Double buffering is an excellent performance solution, and offers the best skew and delay characteristics of all buffering solutions for fanouts of 8 to 16.

NOTE

Double buffering on an 8x12 (1000 usable gates) or 12x16 (2000 usable gates) device requires the use of express wires. These devices have limited express wire resources, so only a few double buffers should be used. Refer to the section on the Router for more information.

Using Double-Buffering With VHDL Source Files

In order to use double-buffering within VHDL source files, you must:

1. include the Cypress-provided package `resolutionpkg` in your VHDL source file, using the following statement:

```
USE work.resolutionpkg.all;
```

2. declare the name of the signal to be resolved (i.e., the signal to be driven by more than one driver), and declare it to be of sub-type `multi_buffer` and type `bit`.

The example in Figure 4-2 is a VHDL source file that implements the design shown in Figure 4-1.

```
-- Resolution function for wired-or.  Used to create
-- legal VHDL for double-buffering techniques
-- employed for pasic.
-------------------------------------------------------
use work.resolutionpkg.all;
use work.GATESPKG.all;
use work.cypress.all;
use work.rtlpkg.all;

entity DOUBLEBUF is
    port(IN0: IN bit;
         IN1: IN bit;
         IN2: IN bit;
         OUT7: INOUT bit;
         OUT6: INOUT bit;
         OUT5: INOUT bit;
         OUT4: INOUT bit;
         OUT3: INOUT bit;
         OUT2: INOUT bit;
         OUT1: INOUT bit;
         OUT0: INOUT bit);
end DOUBLEBUF;

architecture archDOUBLEBUF of DOUBLEBUF is
  -- net to be resolved
  signal multiple_driver: multi_buffer bit;
begin
  multiple_driver <= IN0 AND IN1 AND IN2; -- driver #1
  multiple_driver <= IN0 AND IN1 AND IN2; -- driver #2
  OUT0 <= NOT multiple_driver;
  OUT1 <= NOT multiple_driver;
  OUT2 <= NOT multiple_driver;
  OUT3 <= NOT multiple_driver;
  OUT4 <= NOT multiple_driver;
  OUT5 <= NOT multiple_driver;
  OUT6 <= NOT multiple_driver;
  OUT7 <= NOT multiple_driver;
end archDOUBLEBUF;
```

Figure 4-2. Example of Double-Buffering in VHDL Source File

4.1. Speeding Up High-Fanout Nets

4.1.2. Split Buffering

Split buffering breaks a wide-fanout net into two or more nets.

Figure 4-3 is an example of split buffering. Without the buffers, the DFF drives a fanout of 8. As configured in the illustration, the DFF drives a fanout of 2, and each buffer drives a fanout of 4.

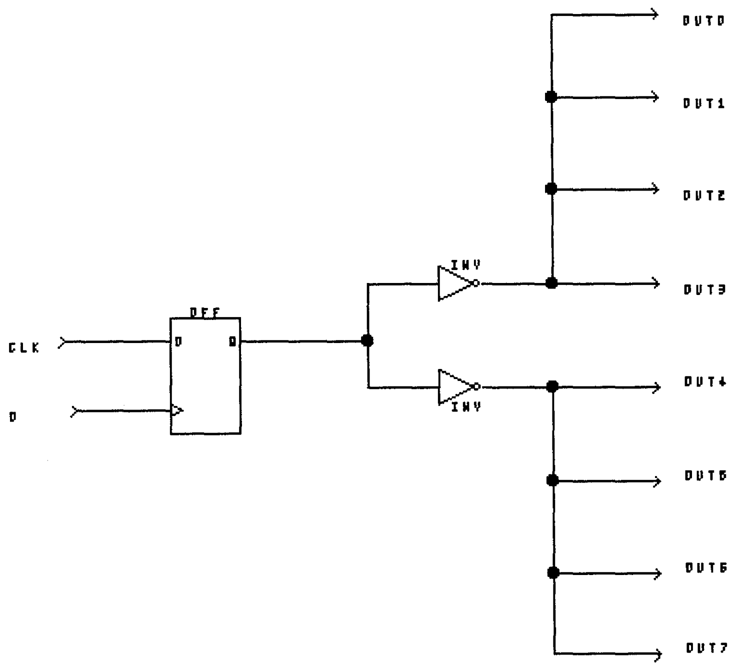


Figure 4-3. Circuit demonstrating split buffering.

NOTE

Adding buffers introduces a logic cell delay to the net. This added delay must be balanced against the gain in reducing the fanout. Simple split buffering (as demonstrated in Figure 4-3) is generally employed only with fanouts of 16 or greater.

4.1. Speeding Up High-Fanout Nets

4.1.3. Selective Buffering

Selective buffering is the selective use of buffers in situations where a high-fanout net has a small number of critical destinations, and a large number of less-critical ones.

Figure 4-4 is an example of selective buffering. The DFF drives a fanout of 8, but only one of the destinations is in the critical path of the circuit. Inserting a single buffer between the DFF output and the 7 non-critical destinations re-structures the circuit so that the DFF drives a fanout of two without adding any logic cell delay in the critical path.

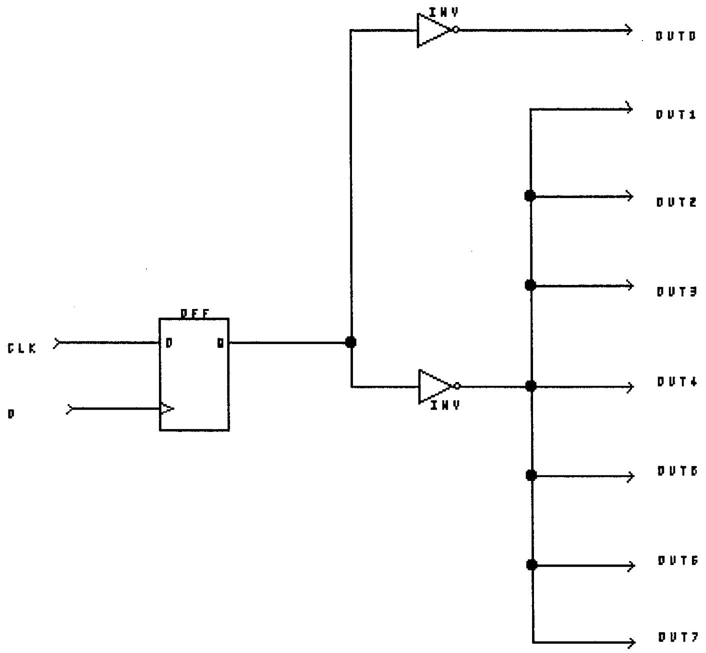


Figure 4-4. Circuit Demonstrating Selective Buffering.

Buffers should be introduced with care and skill. **Selective buffering** offers tremendous improvement in circumstances where the circuit has a few clearly identifiable critical paths.

4.1. Speeding Up High-Fanout Nets

4.1.4. Paralleling

Paralleling is a design technique that duplicates the logic driving a high-fanout load to reduce the effective fanout. Duplicating the logic avoids the delay introduced by adding buffers to the circuit.

Successful buffering must balance reduced fanout against the additional delay caused by the use of buffers. Paralleling is an alternative that does not introduce this added delay.

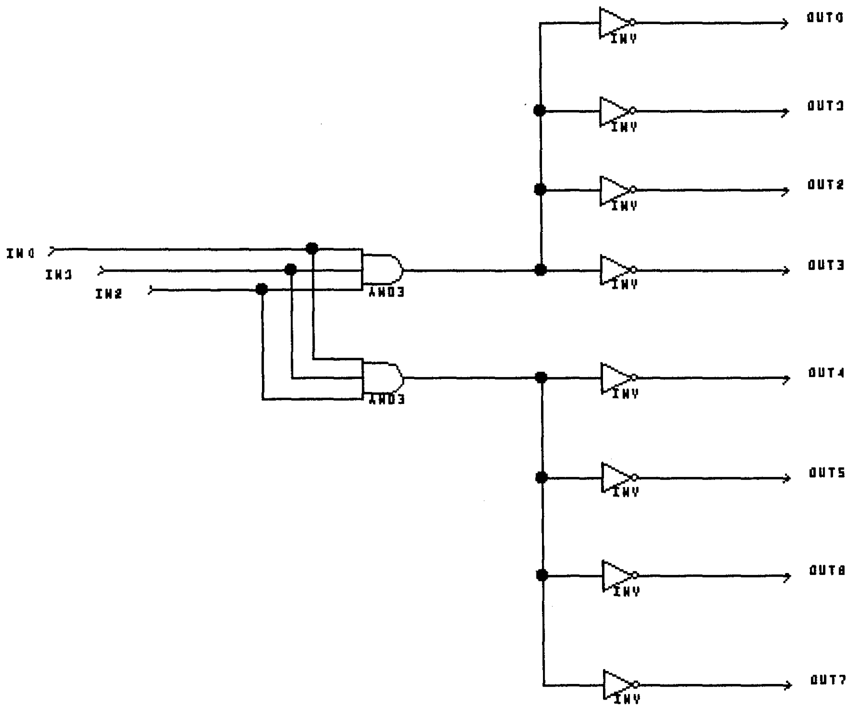


Figure 4-5. Circuit demonstrating paralleling.

Figure 4-5 is an example of paralleling. The AND gate has been duplicated, with each of its inputs tied to the corresponding input on the "twin" gate. Each AND gate drives a fanout of 8, effectively halving the fanout, without introducing the added delay associated with buffering. By duplicating the AND gate, however, the fanout on each of the input nets has been increased.

Notice that paralleling is similar to double buffering, except that the outputs are not tied together. Paralleling should be used instead of double buffering when:

1. skew is not critical;
2. too many express wires have already been used for high-drive inputs or double buffers (see the section on the Router); or
3. the logic to be replicated does not fit into an AND fragment of the larger cell (no larger than a PAfrag_a library element).

4.1.5. Pipelining

Pipelining is the technique of inserting registers in long combinatorial paths, effectively increasing the system clock rate.

Inserting registers in long combinatorial paths shortens the length of the critical path and allows operations to be overlapped, increasing the system clock rate. The pASIC architecture promotes pipelining, as each logic cell contains a D flip-flop. As a result, a design can be pipelined with little or no increase in the number of logic cells used.

Appendix A Error Messages

This appendix is a reference of all SpDE messages. You may get error messages after different actions using the SpDE toolkit.

1. **Import - QDIF** from the SpDE menu: refer to the section of this appendix titled **Import Design Verifier**.
2. All numbered error messages from SpDE: refer to the section of this appendix titled **User Errors**.
3. Error messages from other design tools: refer to the documentation for that tool.

A.1. Import Design Verifier

The Design Verifier, which runs when a design is loaded into SpDE, presents **Notes, Warnings**, and **Errors** in an interactive list box.

Notes

Notes are intended to bring a situation to the designer's attention. The situation is probably not a problem, but should be verified nevertheless.

Gate <*gate*> is not used, and is being removed.
The Design Verifier has determined that the gate is not being used. This "stripper" function can be deactivated from the SpDE Tools Options dialog box.

Warnings

Warnings alert the designer to a problematic situation, commonly associated with a real problem.

Exceeded recommended limit of high-drive nets.
Too many nets are sourced by HDPADs, CKDPADs, and/or double-buffers (parallel AND gates); the router may not be able to complete. Using fewer signals in tandem with these pads guarantees routability.

Gate <*gate*> cannot have a fixed placement.
The specified gate cannot have a fixed placement. Fixed placements can be applied to logic cells, which utilize the flip-flop in the logic cell.

Gate <*gate*> has no net on pad.

There is no external net defined for an input or an output of the design. The path analyzer will not be able to use this gate as a defined start or stop point in analysis. Add a net and a net name to the pad.

Net <*net*> drives no inputs.

The specified net has a fanout of zero. (The net has a driving gate, but no other connections.)

Net <*net*> has high I/O pad fanout of <*fanout*>.

The specified net has exceeded the recommended fanout limit for a bi-directional pad driver. If the net is speed-critical, employ buffering or paralleling techniques.

Net <*net*> has high input pad fanout of <*fanout*>.

The specified net has exceeded the recommended fanout limit for an input pad driver. If the net is speed-critical, employ buffering or paralleling techniques.

Net <*net*> has high logic cell fanout of <*fanout*>.

The specified net has exceeded the recommended fanout limit for a logic cell driver. If the net is speed-critical, employ buffering or paralleling techniques.

Pin <*pin#*> (<*gate*>) drives set or reset, disabling ATVG.

One of the restricted testing pins (labeled I/SCLK or I/SM in pinout diagrams) is driving a set or reset, directly or indirectly. These pads are restricted testing pins, and require that ATVG be disabled.

Pin *<pin#>* (*<gate>*) paralleled, disabling ATVG.
One of the restricted testing pins (labeled I/SCLK or I/SM in pinout diagrams) is wired in parallel with another pin. These pads are restricted testing pins, and require that ATVG be disabled.

Errors

Errors flag genuine error conditions that would prevent parts from being programmed. However, the tools can still be run for experimental purposes and examination.

Gate *<gate>* has floating input.
The specified gate has one or more unconnected inputs. Floating inputs are not allowed.

Net *<net>* driven by multiple I/O pads.
The specified net is driven by more than one I/O pad. A net cannot be driven by multiple I/O pads.

Net *<net>* has fanout of <24, but >2 drivers.
The specified net has too many high-drive pads. Remove one high-drive pad and re-try.

Net *<net>* has no driver.
The specified net does not have a driving cell; thus, the inputs of the attached cells are floating.

Fatal Errors

Fatal errors flag serious error conditions that prevent the tools from being run..

Clock net <*net*> has multiple drivers.
The specified net is driven by more than one clock pad. Clock nets must be driven by one and only one clock pad.

Dual drive gate <*gate*> is illegally connected.
You have tried to use double-buffering, but incorrectly, OR you illegally tied the outputs of two gates together.

Gate has illegally connected outputs.
Two gates have their outputs tied together illegally. (You may have double-buffered them incorrectly.)

Gate <*gate*> is placed on incompatible cell.
The specified gate has an invalid fixed placement. A bi-directional pad macro may have been placed on an input cell, or vice versa.

Gates <*gate*> and <*gate*> are placed on the same cell.
Two gates cannot be placed on the same cell.

High-drive net <*net*> has opposing pads in a corner.
A net, driven by a high-drive pad, cannot drive a pair of bi/tripads that are at a 90-degree angle to each other in a corner of the chip (e.g., one on the top, one on the right side). Move one of the pads away from the corner and re-try.

High-drive net <*net*> has pads on top and bottom.
Multiple high-drive pads (HD2PAD, HD3PAD, HD4PAD) must have fixed placements. Multiple high-drive pads must be placed on the same side of the chip (e.g., all on the top or all on the bottom of the chip). This error also occurs if you are driving tri-state enables directly from HDPADs. In this case, you cannot fix a pin driven from the HDPAD on the opposite side of the chip from the HDPAD.

Net <*net*> uses clock pad to drive logic inputs.
The clock output tree of the CKPAD cannot be used to drive any logic except for clock pins, asynchronous presets, and clears. Consider using one of the two high-drive outputs of the CKPAD (IN or IZ).

Net <*net*> driven by more than two logic outputs.
The specified net is driven by more than two logic cells. A net can be driven by two logic cells in the case of double-buffering, but a net can never be driven by more than two logic cells.

Net <*net*> driven by multiple sources.
The specified net has an illegal configuration of multiple drivers. The only valid configuration of multiple drivers is two, three, or four high-drive pads.

Net <*net*> is on both sides of an I/O pad.
The specified net has been wired both inside and outside the boundary of a single pASIC. Often, a net attached outside the chip (to a pad, for example) will be named accidentally with a name already used inside the chip.

Net <*net*> uses clock pad to drive logic inputs.

A clock net is being used to drive logic cells. The dedicated clocking structures (CKPADs) may only drive clocks, sets, or resets of logic cells.

Pad on net <*net*> must be pre-placed.

When using HDPADs to drive the enables of more than 16 tri/bipads, the tri/bipads must be pre-placed either on the same side or adjacent sides of the HDPAD placement. The tri/bipads may not be located on the opposite side of the HDPAD. If there are 16 or fewer pads driven from an HDPAD, the Design Verifier performs the placement automatically.

Used <*number*> bi-drectional pads with <*max*> available.

You have used more general I/O pads than are available on the chosen device. Remember that some pin positions require special pads, such as input-only pads or clock pads.

Used <*number*> clock pads with <*max*> available.

You have used more general I/O pads than are available on the chosen device. There are only two clock pads (CKPADs) on each pASIC device.

Used <*number*> flip-flops with <*max*> available.

You have used more flip-flops in your design than are available in the chosen device.

Used <*number*> input-only pads with <*max*> available.

You have used more HDPADs in your design than are available on the chosen device. There are six HDPADs available on all pASIC devices.

A.2. User Errors

SpDE reports user errors using an Error dialog box. These errors represent design or system errors that can be fixed by the user. The list below is organized by tool code; the first two letters of the error code indicate the tool.

XX—(starting with any two letters)

xx0100- xx0199	**Out of memory.**
	SpDE has requested more memory than Windows currently has available. Try closing other applications and re-running SpDE. If the problem persists, try re-starting Windows. Many memory problems can be solved by creating a larger Windows swap file. Windows offers very efficient memory management; refer to the **Microsoft Windows User's Guide** for complete details.

CH—Chip file to QDIF file converter (loads old design files)

CH0001- CH0002	**Error loading binary file:<*filename*>. Cannot save QDIF file: <*filename*>.**
	The converter software is having trouble loading the source design or saving to the destination. This could be due to a full disk, or to a lack of read or write access to the files.

DB—the SpDE Database Module

DB0001-DB0002	**Invalid package type.** An invalid package topic has been chosen for the pASIC chip being targeted.

ED—EDIF Netlist Reader

ED0002-ED0003	**Syntax error on line <*line number*>.** Illegal syntax has been used at line <*line number*> in the EDIF file.

ET—EDIF Netlist Reader (EDIF to SpDE Translator)

ET0006	**Unknown package type: <package>** A package that SpDE does not recognize is specified in the EDIF file.
ET0007	**Package has incorrect pin bonding** A pin that does not exist (or is not bonded out) on the selected package is used in the EDIF file. Either the pin number or package type are incorrect.

GP—Graphing Package

GP0001-GP0002	**Error opening clipboard** **Error opening picture** The Grapher could not properly open the picture of clipboard with Windows calls. Try re-booting your computer.

GP0003-GP0005	**Error closing picture** **Error closing clipboard**	The Grapher could not properly close the picture of clipboard with Windows calls. Try re-booting your computer.
GP0004	**Error putting picture onto clipboard**	The Grapher could not complete the operation of copying the graph to the clipboard. You may be low on memory, or Windows could be unstable. Try re-booting your computer.

JE—LOF Netlister

JE0001	**Could not open file <*filename*>**	<*filename*> specified by the user either does not exist, or does not have a read attribute.
JE0002	**No LOF support for part <*part*>**	The device used for the current design (<*part*>) is currently not supported by the LOF Netlister.

LS—Load and Save Files

LS0001-LS0004	**Could not open binary file <*filename*>**	<*filename*> specified by the user either does not exist, or does not have a read attribute.
LS0002-LS0005	**Wrong part file DB version in file <*file*>**	An old version of the specified part file exists in the SpDE data directory. Check your WIN.INI file to ensure that the ini-path entry in the [SpDE] section has been properly set.

LS0003-LS0006

Unknown part name *<part>*

The part specified in the design file does not exist, or does not have an associated part file. Check your WIN.INI file to ensure that the ini-path entry has been properly set.

LS0007-LS0010

Part File Errors

This error occurs if SpDE cannot find a current, valid part file. If this error occurs, you may want to re-install SpDE.

LS0011

Unknown package type: *<package>*

A package that spDE does not recognize is specified in the QDIF file.

LS0200

<error>* at approximately line *<line number>

The parsing error *<error>* occurred while reading line *<line number>* of the QDIF file.

PA—Path Analyzer

PA000x

Clipboard Errors

These errors indicate that the Path Analyzer could not use the Windows clipboard properly. Try re-booting your computer.

PK—Packer (Level 0 Optimizer)

PK0000	**Cannot pack - too many logic cells**
PK0001	**Too many HDPADs (input-only pads) used**
PK0002	**Too many I/O pads used**
PK0003	**Too many CKPADs (clock pads) used**
	The design requires more of the specified resources than are available in the selected pASIC device. Use fewer of the specified components, or select a larger device.
PK0004	**Illegal fixed I/O location**
	An I/O cell has been assigned to an incompatible pin location. For example, a high-drive pad was placed on a bi-directional pin. Move the fixed placement to an appropriate location.

RT—Router

RT0000	**Could not complete routing**
RT0001	**Could not complete clock routing**
RT0002	**Could not complete hi-drive routing**
	The router does not have enough resources to complete routing. In the case of hi-drive routing, refer to “Special Routing Cases” in Section 2.4., "Router". Otherwise, try re-placing after changing the placer seed.

RT0003	**Out of express wires in channel <x><y>. Re-run placer with another seed.**
	The router requires more express wires than are availabe in the specified channel. This problem is most often caused by an excess of signals attached to the high-drive input pads. Employing four or fewer signals in tandem with these pads guarantees routability of these signals.

SD—SDF Writer

SD0001	**Cannot open file: <*filename*>**
	The SDF writer cannot open the SDF file that it needs to create. This could be due to a full disk, or a write-protected file or directory.

SP—SpDE

SP0004	**SPDE.INI is read-only or does not exist.**
	SpDE could not find its initialization file SPDE.INI for saving defaults. This could mean that the file has been erased or that the file is read-only.

SQ—Sequencer

SQ0000	**Sequencer could not complete. Re-run Router with a different seed.**
	The sequencer could not determine an order in whcih to program the Via-Links in the part. Re-running the placer and/or the router with different seeds should correct the problem.

TM—Technology Mapper (Level 1 Optimizer)

TM0001 **TM0002** **TM0003** **TM0004**	**Cannot pack - too many high-drive pads** **Cannot pack - too many I/O pads** **Cannot pack - too many clock pads** **Cannot pack - too many logic cells**
	The Technology Mapper has determined that the design needs more resources than are available. You may need to select a larger device, or change the design accordingly.

UI—User Interface

UI0001	**There is (are) <*number*> dll(s) not in SpDE's path.**
	SpDE has detected <*number*> DLL's that it needs that are not in the current spDE directory.
UI0002	**Cannot convert chip file <*filename*>**
	SpDE could not convert the chosen chip (.CHP) file to the latest version. Possibly a non-chip file or a chip file from a very old version of SpDE has been selected.
UI0003	**Unable to complete command <*command*> successfully.**
	SpDE tried to execute the command <*command*>, without success.
UI0004	**Invalid directory: <*directory*>**
	The chosen directory cannot be accessed. This may happen if the chosen directory is a DOS drive that has been "joined" to a network directory. Also, the directory may not exist.

UI0005	**Can't change to specified directory** The chosen directory cannot be changed to properly. This will happen if the chosen directory is a DOS drive that has been "joined" to a network directory.
UI0006 UI0023 UI0024	**SPDE.INI is read-only. Cannot save options** SpDE could not read and/or modify the SPDE.INI file. Make sure this file is not in a read-only directory. You may also check the WIN.INI file under [SpDE] to see that the ini-path points to the directory where this file exists. (.spderc in home directory for SUN users).
UI0008	**PKZIP.EXE was not found in path** The LOF file cannot be properly compressed unless PKZIP version 1.01 is in the DOS path. Either put PKZIP in the path, or ZIP the file manually using PKZIP 1.01.
UI0009	**Unable to run command: *<command>*** **Reason: *<reason>*** SpDE could not run a Windows application because of *<reason>*. This could indicate an improper configuration.
UI0010	**No printer connected** SpDE could not detect a printer device under Windows. Check Printer Setup in the Windows Control Panel.
UI0011	**Printer not set up** SpDE could not print to the default device. Check Printer Setup in the Windows Control Panel.

UI0015	**File <*filename*> is from a later version of SpDE...**	You have chosen to open a file that was created from a later version of spDE than is running currently. If you did not intend this, check your configuration, or re-install the latest SpDE tools.
UI0016	**Unable to convert <*filename*>**	SpDE could not properly convert <*filename*> to the current version of SpDE.
UI0017	**Can't initialize gang programmers**	You may have a configuration problem with the gang programmers, or a problem with your serial card.
UI0018	**No gang programmers found**	Check to make sure all gang programmers are connected and plugged in, and that the correct COM port has been chosen.
UI0019	**No automatic place and route tools were run - Check Place and Route option settings.**	You have chosen to Run All Tools after all the tools have already been run. If you wish to iterate: change seeds in the Tools Options, then re-run tools with Run Selected Tools.
UI0020 **UI0021**	**Cannot process SPDE.INI file**	The SPDE.INI file has been corrupted.

UI0022	**Error opening report file <*filename*>**	SpDE could not open the report file it has created. This could happen if you were too low on memory to load the chosen editor, or if the chosen editor could not be loaded properly. Change the chosen editor from View/ Preferences.
UI0034	**Cannot load QDIF file <*filename*>**	An error was detected while reading a QDIF file. The file may have a syntax error, or the file may have been damaged.
UI0037	**Ini-path not found in win.ini. Using c:\pasic\spde\data**	SpDE expects to find the variable ini-path under the heading [SpDE] in the win.ini file. The installation program will do this automatically. Check the win.ini file, or re-install SpDE.
UI005x	**Save Error**	A DOS error was detected while trying to save a file. This may be caused by a write-protect violation or insufficient disk space.
UI006x	**Load Error**	A DOS error was detected while trying to load a file. This may be caused by choosing the wrong file type to load, or trying to load a file without a read attribute.

VE—SpDE Physical Viewer

VE0007 VE0010 VE0012	**Value must be between <*min*> and <*max*>**	The value you have entered is out of the allowable range. Enter a value between <*min*> and <*max*>.
VE0009 VE0011	**Bad (unsigned) integer value**	The value you have entered does not represent a proper integer value. If SpDE is expecting an unsigned integer, make sure the number is positive. Always make sure integers do not have decimal points.

VG—Verilog Netlister

VG0001	**Error: cannot open file: <*filename*>**	The Verilog netlister cannot open the output file it is trying to create. This could be due to a full disk or a read-only directory.

VL—Viewlogic Netlister

VL0000- VL0004, VL0006	**Error: cannot open file: <*filename*>**	The Viewlogic netlister could not access the specified file. Check Viewlogic environment variables and write-access of specified directory; also check to be sure specified file exists.
VL0005 VL0007	**Cannot write to file: <*filename*>**	The Viewlogic netlister could not write to the specified file. Check available disk space and write permission on the specified directory and file.

A.3. Internal Errors

These errors are indicated by the title bar **Internal Error**, as well as message of a (usually) cryptic nature. These error should not occur—they indicate an inconsistency in SpDE's data structures.

If you encounter one of these errors, record the text of the error completely and contact Cypress Semiconductor Corporation.

Index