Ultra38000™

Place and Route

User's Guide

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
(408) 943-2600
August 1996

Cypress Software License Agreement

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.

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.

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.

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.

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.

RESELLING. The reselling or distribution of this product can be done by Cypress authorized distributors only.

BENCHMARKING. This license Agreement does not convey to you the right to publish performance benchmarking results involving any Cypress *Warp* products. Permission to publish performance benchmarking results involving any Cypress *Warp* products must be received in writing from Cypress Semiconductor prior to publishing.

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.

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.

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

Contents

Ultra38000 Place and Route User's Guide

1 Introduction

This guide describes the following topics in *Warp's* SpDE tool kit:

- SpDE Design Flow
- SpDE Graphical Interface: the SpDE Window
- SpDE Design Tools
- SpDE Analysis Tools
- Design Considerations: Speeding Up High-Fanout Nets

Warp's SpDE tool kit provides a complete suite of FPGA place and route tools, allowing the user to complete the FPGA design process, to perform timing analysis on completed FPGA designs, to generate simulation models for timing simulation, and to generate programming files for device programming.

1.1 Design Placement and Routing

The final step in the FPGA design flow is to map logic fragments generated by the *Warp* compiler into logic cells of the target FPGA device. This process is called place and route. SpDE provides automatic placement and routing, requiring virtually no user intervention.

1.2 Design Viewing and Path Analysis

After automatic placement and routing, SpDE's Physical Viewer displays the graphed results. Along with the Path Analyzer, the user performs static timing analysis on critical paths and highlights them in the Physical Viewer.

1.3 Design Simulation and Programming

After design completion and analysis, the user can choose to generate simulation models for timing simulation in many environments for device programming.

2 SpDE Design Flow

The following is a typical SpDE design flow. For details on the description of each tool, please refer to Section 4, "SpDE Design Tools."

2.1 Starting SpDE

To start SpDE in *Warp3*, double-click on the Ultra38000 icon in the Workview PLUS Cockpit (on IBM PCs and compatibles) or the Powerview Cockpit (on UNIX workstations). SpDE can also be started from the SpDE program group or Galaxy's SpDE menu item on IBM PCs (and compatibles) and from the command line on UNIX workstations.

For *Warp2*, SpDE can be started from the SpDE program group or Galaxy's SpDE menu item on IBM PCs (and compatibles) and from the command line on UNIX workstations.

2.2 Importing Files

Warp designs targeted for FPGA devices are first compiled and synthesized by the *Warp* compiler into logic fragments and then saved as QDIF files. These files must be imported into SpDE before placement and routing can take place.

To import a QDIF or EDIF file, use:

File ->Import QDIF / EDIF

During import, the Design Verifier is automatically invoked, which checks for design errors, architectural violations, and potential problems.

2.3 Running the Tools

Once a QDIF file has been imported, the user starts the SpDE design tools by using the following:

Tools-> Run Selected Tools...

A summary of the available tools and their respective tasks are listed below.

Table 1 Summary of SpDE tools

SpDE Tool	Function
Logic Optimizer	Logic Optimization
Placer and Router	Automatic Placement & Routing
Delay Modeler	Timing Analysis
Back-Annotation	Simulation Model Generation
Automatic Test Vector Generator	Test Vector Generation
Sequencer	Programming File Generation

2.4 Design Viewing and Path Analysis

After automatic placement and routing, the user can use the Physical Viewer and the Path Analyzer to view the physical placement and routing of the design logic and to examine the timing for critical paths. The Path Analyzer can be invoked as follows:

Tools->Path Analyzer

The user can also examine resource utilization and the report file by using:

Info-> Utilization.../ Report File…

2.5 Saving

After automatic placement and routing, the final design implementation can be saved as a *.chp* file. This can be done by selecting the following:

File->Save/ Save As...

File Formats

Here are the various file formats that SpDE uses as input or generates as output.

Table 2 SpDE file formats

File Type	Format(s)	Description
Input	QDIF (*.qdf*) EDIF (*.ed**)	*Warp* and some third-party tools (e.g., Exemplar's Galileo) generate QDIF files, while others (e.g., Synopsys'® Design Compiler™) generate an EDIF file as input to SpDE.
Output / Input	CHIP (*.chp*)	Post-place and route designs are saved in this format. Designs can then be reloaded at a later time.
Output / Input	QCF (.qcf)	Path constraint file. Gets saved when the user puts path constraints in the Path Analyzer and is read back on reload of the design.
Simulation	varies	Different types of simulation timing models are generated depending on the target simulator. Refer to section 4.5 "Delay Modeler and Back Annotation" for more information.

3 SpDE Graphical Interface: The SpDE Window

3.1 The File Menu

The *File* menu includes commands related to creating, opening, and saving completed designs; importing synthesized designs for placement and routing; exporting programming files; printing the physical view of the layout; and exiting SpDE.

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

Open brings up a dialog box for selecting an existing CHIP (*.chp*) file. The specified CHIP file will then be loaded, replacing the current design.

Save saves the currently completed design as a CHIP file.

Save As brings up a dialog box, allowing the user to save the current completed design as a CHIP file with a user-specified name. This menu item gives the user the capability of saving backup or reference copies of a design.

Import loads files generated by *Warp* or other synthesis tools in QDIF or EDIF format for placement and routing.

*Print Setup (*PC version only) invokes the printer setup dialog box for setting the default printer driver.

Print Preview (PC version only) shows a picture of the matter to be printed for the user's perusal.

Print (PC version only) prints the current physical view using the parameters set in Print Setup.

Exit terminates SpDE and prompts the user to save the design if it has been modified since the last save. For PC users, 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 (numbered 1-5). 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.

3.2 The Edit Menu

The *Edit* menu is currently non-functional and is reserved for later use.

3.3 The View Menu

The *View* menu (Figure 1) includes commands for manipulating the physical view of the design, highlighting specific nets, and specifying physical view options.

Zoom In magnifies the physical view of the current design. The user can simply click on a specific spot as the center of the view and zoom in by a scale factor of 1.25, or click and drag the mouse pointer to define a viewing rectangle. The view adjusts to fit the specified rectangle as closely as possible. The shortcut key for *Zoom In* is F11.

Zoom Out de-magnifies the view. The shortcut key for *Zoom Out* is F12.

Pan allows the user to drag the view with the help of the cursor. The shortcut key for *Pan* is F10.

Center moves the point under the cursor to middle of the SpDE window. The shortcut key for *Center* is Ctrl-C.

Full Fit modifies the scale factor to fit a view of the entire Ultra38000 layout on the screen. The shortcut key for *Full Fit* is F9.

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.

Lock View freezes the layout view in full fit mode and disables other view maneuvering functions like zoom in etc. The shortcut key for Lock View is Ctrl-L.

Refresh redraws the physical view. The shortcut key for Refresh is F5.

Highlight Net allows the user to select nets to be highlighted in the physical view. This menu item is discussed in greater detail in section 5.1.

Verifier messages brings up the message box with list of design errors. This is discussed in detail in the section 4.1 on “Design Verifier”.

Preferences sets physical view options. These are discussed in greater detail in the next section, ”Preferences.”

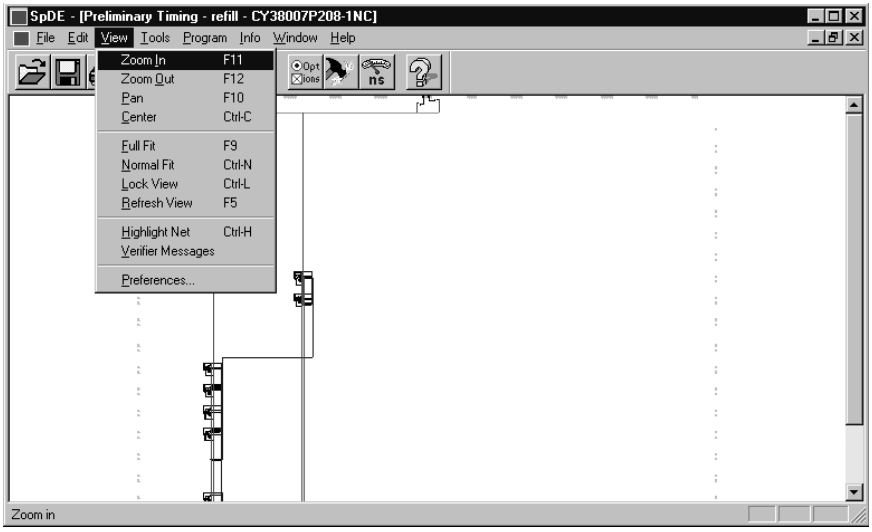


Figure 1 View menu

Preferences

Selecting *Preferences* from the *View* menu brings up the *Preferences* dialog box (Figure 2).

The *Texting* tab (Figure 2) groups check boxes which include items relating to text labels in the physical view. Table 3 lists the check boxes and examples of the items they control. Text items can be turned off to increase redraw speed or to simplify the physical view.

Table 3 Examples of check box items

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

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 except for VCC or GND. To find out VCC or GND connections use the *VCC GND Net Names* check box.

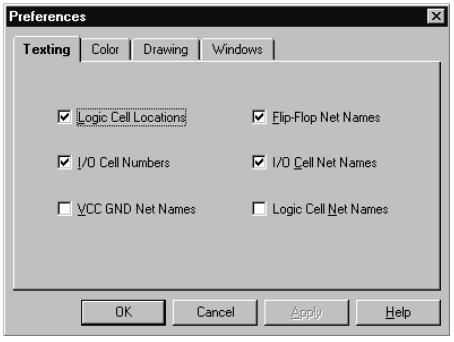


Figure 2 Preferences dialog box

The *Color* tab (Figure 3) contains the controls to assign the color to various resources used in the physical viewer and the path analyzer. To change the color of a resource, select it from the appropriate list and assign a color from the color pallets.

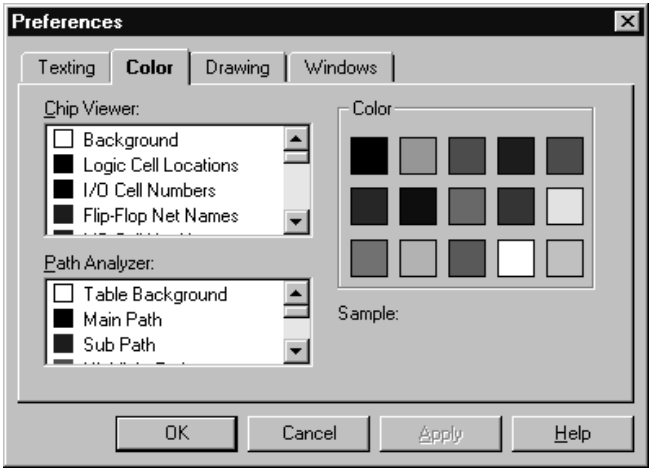


Figure 3 Color tab from Preferences dialog box

The *Drawing* tab (Figure 4) includes the *Cell Filtering* check box. Deselecting this check box replaces the detailed logic cells with simple boxes, thereby increasing redraw speed.

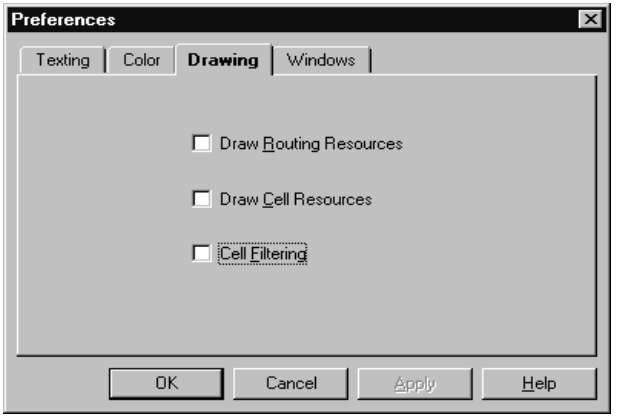


Figure 4 Drawing tab from Preferences dialog box

The *Windows* tab (Figure 5) includes two check boxes to customize SpDE window. *Show Toolbar* when unchecked will remove the toolbar and *Show Status Bar* when unchecked will remove the status bar from the SpDE window.

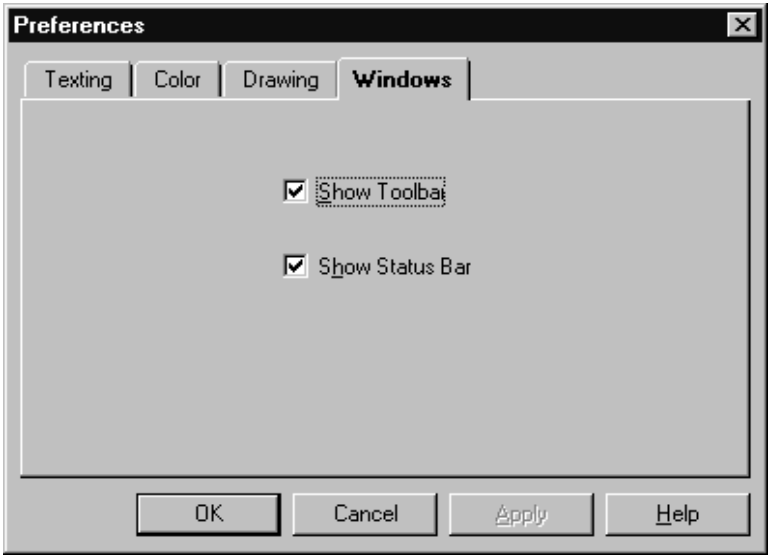


Figure 5 Windows tab from Preferences dialog box

Clicking *Apply* temporarily accepts all the preference settings and then the effects are displayed for the user's perusal. Clicking the *OK* button accepts all the preferences for subsequent use. It also records the preference settings. These settings are then used the next time SpDE is invoked. Click the *Cancel* button to discard the changes.

3.4 The Tools Menu

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

The *Tools* menu (Figure 6) contains three items: *Options*, *Run Selected Tools* and *Path Analyzer*.

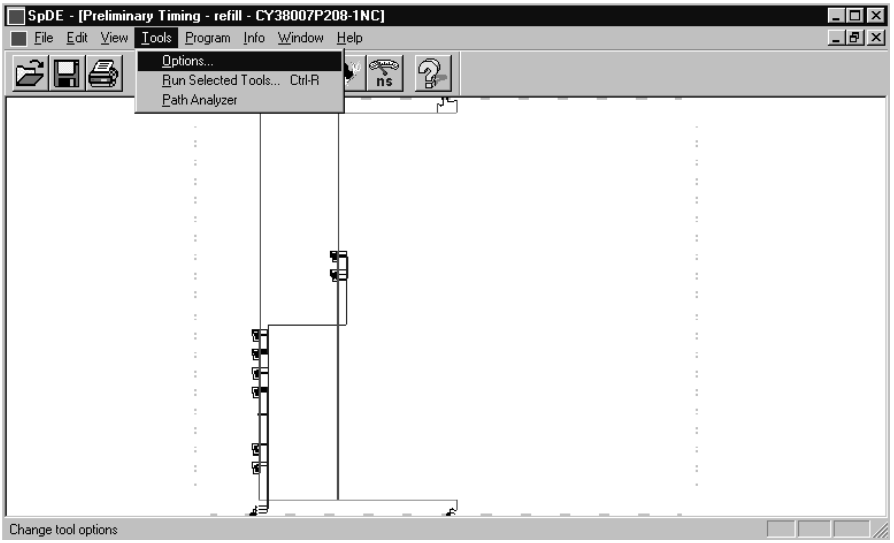


Figure 6 Tools menu

Run Selected Tools opens the *Run Tools* window (Figure 7). This window is used to select which tools are to be run on the design. Two types of logic optimization can be run. This selection is made in the *SpDE Tools Options* window (Figure 8). Disabled tools are grayed out. Tools that have already been run are in black color. To run a tool check the box in front of it. Detailed descriptions of the Logic Optimizer, Placer, Router, Delay Modeling, Back-Annotation, and ATVG tools can be found in Section 4, "SpDE Design Tools." The Sequencer tool is only used to create data needed to program devices and has a short description in that section.

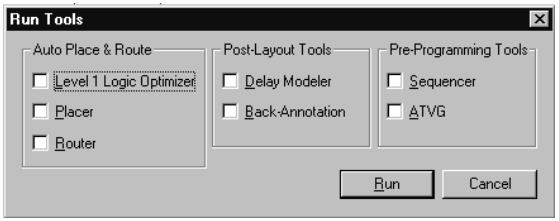


Figure 7 Select Tools To Run dialog box

The 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 5.2, "Path Analyzer."

Options pops up the *Tools Options* window. The *Tools Options* window is shown in Figure 8. The *Tools Options* window is used to configure each of the SpDE tools. A detailed description of each tool's options can be found in section 4, along with the description of the tool.

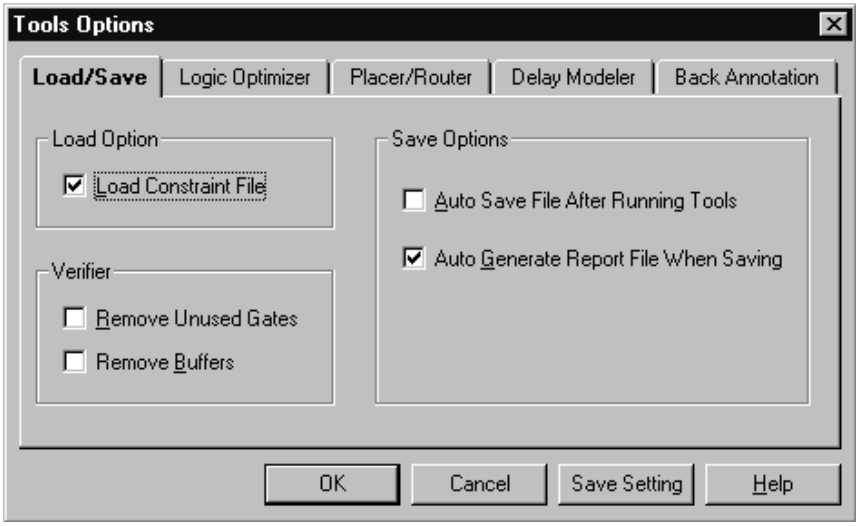


Figure 8 General Tools Options window

3.5 The Program Menu

The *Program* menu is currently disabled and will be available in a future release. To find out more information on programming support, please call Cypress's Applications help-line at (408) 943-2821.

3.6 The Info Menu

The *Info* menu (Figure 9) includes items that provide statistics and other information about a design.

Utilization reports the number of cells of different types used in the design (Figure 10) as well as the number of logic, input-only, clock-only, and bidirectional cells. In addition it reports the routing and the via-link resource utilization.

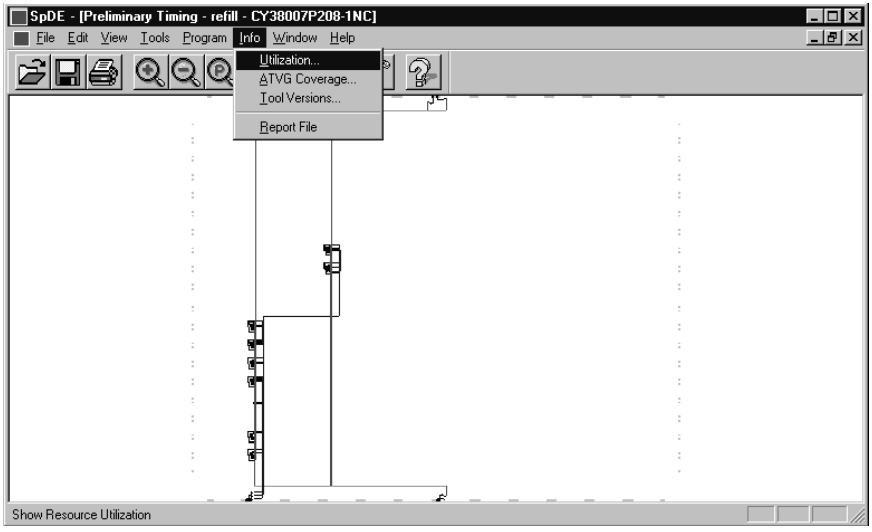


Figure 9 Info menu

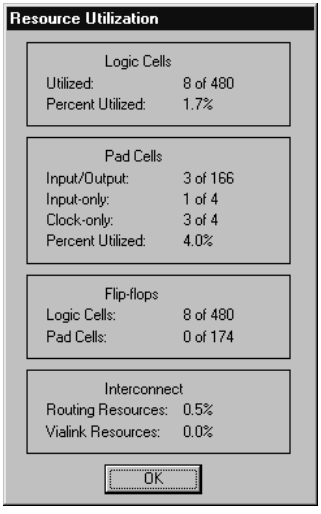


Figure 10 Typical Utilization dialog box

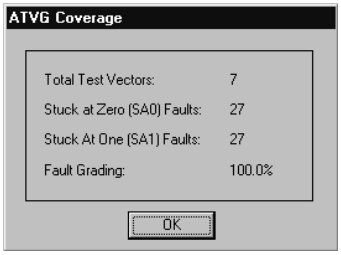


Figure 11 Typical ATVG Coverage dialog box

ATVG Coverage reports on the design's test coverage (Figure 11). These statistics may be used as guidelines to improve the design in order to increase test coverage. Of particular interest is the Fault Grading statistic, which indicates the quality of test coverage produced by ATVG.

Tool Versions is provided for diagnostic purposes. *Tool Versions* lists the tools that have been run on the current design, including the version number of each tool listed (Figure 12).

Report File displays the Cypress report file produced by the *Warp* compiler, including a SpDE tools appendix.

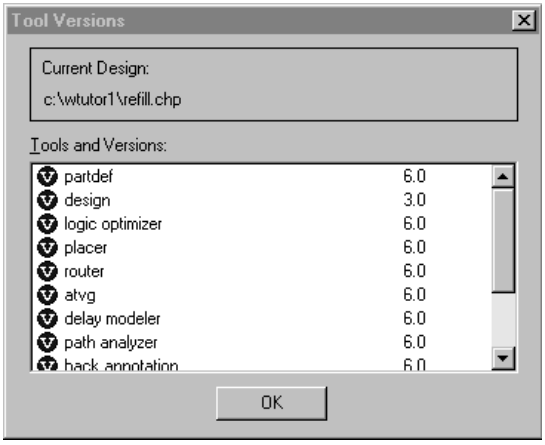


Figure 12 Typical Tools Versions dialog box

3.7 The Windows Menu

The *Windows* menu provides standard windows environment functionality. Available menu items are: *New Window, Cascade, Tile Horizontally, Tile Vertically, Arrange Icons* and a list of open *windows.*

3.8 The Help Menu

The *Help* menu (Figure 13) includes commands for on-line help on SpDE.

Online Documentation instructs the user on how to access the Acrobat file for this document.

About SpDE provides information on the SpDE Toolkit.

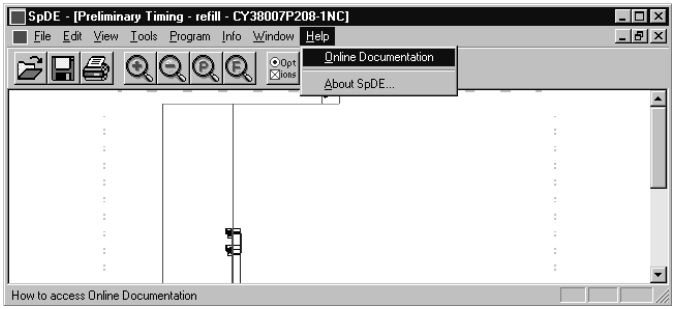


Figure 13 Help menu

4 SpDE Design Tools

This section describes the SpDE design tools:

The Design Verifier analyzes the file for design errors, architectural violations, and potential problems.

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 Ultra38000 interconnect resources.

The Delay Modeler calculates delays for the Path Analyzer and writes the delays and delay scale to a file in the ViewSim simulator.

The Back-Annotation tool creates simulation models with timing. In addition, it writes pin numbers and fixed flip-flop numbers to a file which is later used to back-annotate schematics.

The Automatic Test Vector Generator generates test vectors that can be used to test Ultra38000 devices after they have been programmed.

The Sequencer is the tool that generates the programming file used to program the Ultra38000 devices.

4.1 Design Verifier

When a design is imported from *Warp* into SpDE as a *.qdf* file, the Design Verifier analyzes the file for design errors, architectural violations, and potential problems. If any exist, a window will pop up displaying the appropriate messages. The messages are categorized as follows, according to their severity:

Notes simply provide information which might be of interest to the user, such as the removal of unused gates. No user action is required.

Warnings provide information on potential design problems, such as excessive fanout, which could impact performance and might require user intervention.

Errors provide information on design problems, such as floating inputs, that prevent a part from being programmed, even though the tools can still be run.

Fatal Errors provide information on extreme errors, such as excessive resource usage, which prevent the tools from being run.

Figure 8 shows two options which direct SpDE to take certain actions on the design while importing it. If the *Remove Unused Gates* option is checked, all gates which are floating or are driving nothing will be removed from the design. If the *Remove Buffers* option is checked, all buffers will be removed. Buffers are defined as one input one output non-inverting gates which are implemented in A or F fragments and are such that their output does not drive their input.

4.2 Logic Optimizer

The Logic Optimizer is the first tool to be run after a design netlist has been loaded into SpDE. Logic optimization is the first step in automatic placement and routing (APR). The Logic Optimizer uses sophisticated technology mapping algorithms to partition the design into logic cells.

Two levels of optimization are available in the Logic Optimizer: Level 0 - Packer and Level 1 - Technology Mapper. The optimization levels can be selected from the *Logic Optimizer* tab from the *Tools Options* window (Figure 14), which appears after selecting the *Options* item from the *Tools* menu.

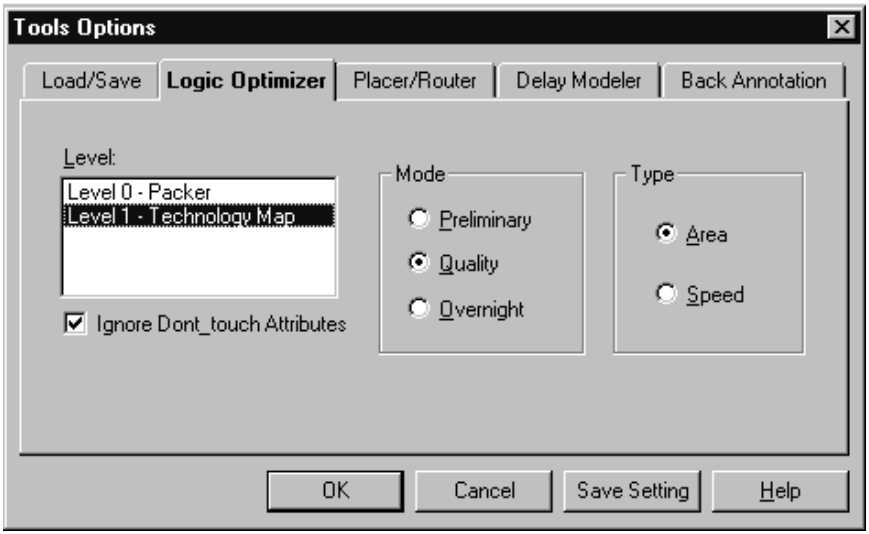


Figure 14 SpDE Tools Options window: Logic Optimizer options

Level 0 Optimization: The Packer

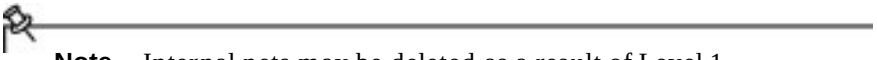
The **Level 0 Packer** simply "packs" logic symbols (hard macros) from the imported QDIF file into logic cells, leaving all net connections intact. As many as four macro symbols may be packed into a single logic cell. No logic optimizations are done. The Level 1 Optimizer is the preferred logic optimizer in almost every case, but Level 0 optimization is provided for versatility and compatibility with old designs.

Level 1 Optimization: The Technology Mapper

The **Level 1 Technology Mapper** provides automatic logic cell optimization. The Technology Mapper introduces and removes inversion bubbles in order to improve capacity and performance. In more general terms, the Technology Mapper merges gates when possible in order to achieve more efficient implementations. Because the Technology Mapper uses a more sophisticated algorithm than the Packer, the 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.

Area / Speed Optimization

The selection of Area versus Speed is passed directly from *Warp* and is simply used here in SpDE to reflect which option was selected. Changing this option in SpDE has no effect on the results of your design.



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

Logic Optimization Modes

For Level 1optimization, the user 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 takes 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.

Ignore Dont_Touch Attributes

If this check box is checked then the Technology Mapper will treat the gates with *Dont_Touch* attribute just like any other gate and will try to optimize them. Uncheck this box to keep the original implementation of the hard macros. This check box has no impact on the Packer.

4.3 Placer

Placement is the second step in automatic placement and routing. The Placer places the logic fragments generated by the Logic Optimizer in optimal locations of the FPGA device to minimize routing delays. The Placer also allows users to fix the placement of I/O cells and registers. Fixing I/O cell placement can ease circuit board design, while fixing register placement offers precise control over internal routing delays.

Via the Path Analyzer, the Placer offers timing-driven placement. The user can enter timing constraints for specific nets in the Path Analyzer, and the information is then passed to the Placer to ensure that critical timing goals are considered with a higher priority during placement. 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 15 shows the dialog box by which the user selects Placer options. Two kinds of options are available: the user can select the seed value for the Placer, the placement mode, or both.

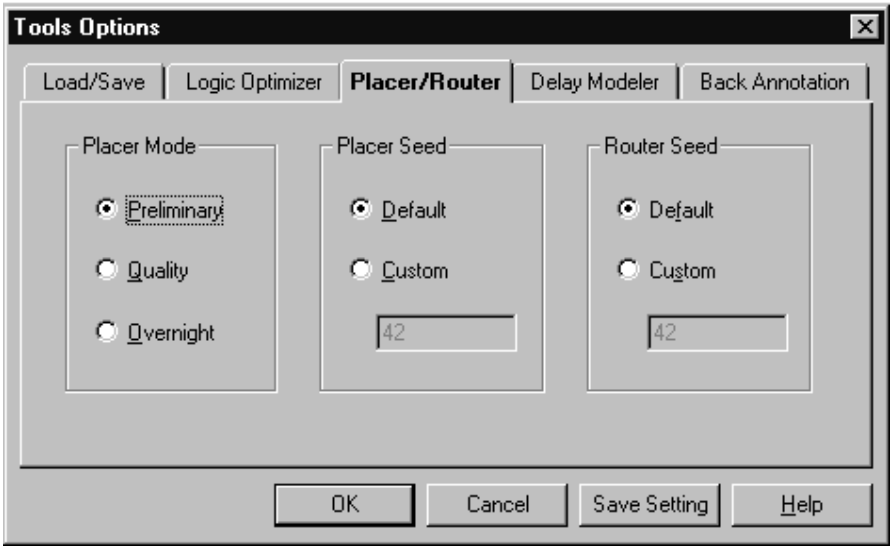


Figure 15 Placer options

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 an integer between 0 and 32767. The user chooses either 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.

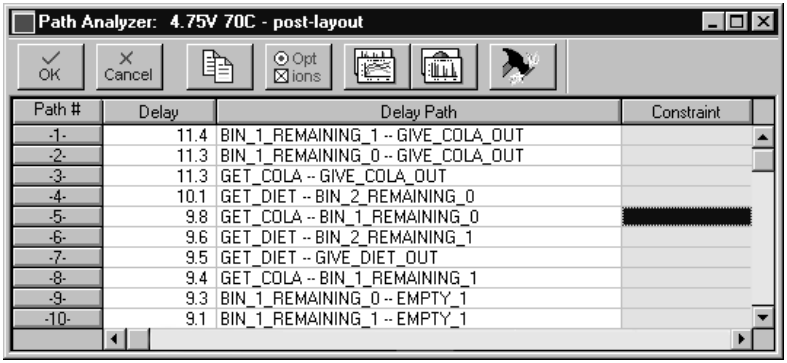
Placement Mode - Three placement modes are available: *Preliminary*, *Quality*, and *Overnight*. These three modes have the following characteristics:

- *Preliminary* placement is faster than quality placement, but usually by only a few minutes. Results are not as predictable as Quality placement. Cypress recommends at least Quality placement of designs before programming chips.
- *Quality* placement is the default placement mode. As its name implies, it produces high-quality placements.
- *Overnight* placement exists primarily for the curious. Results of Overnight mode placement are usually about 4% better than Quality placement, but at a significant cost in run time (about ten times). 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 (Figure 16) to provide timing-driven placement, an advanced technique issued 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-	11.4	BIN_1_REMAINING_1 -- GIVE_COLA_OUT	
-2-	11.3	BIN_1_REMAINING_0 -- GIVE_COLA_OUT	
-3-	11.3	GET_COLA -- GIVE_COLA_OUT	
-4-	10.1	GET_DIET -- BIN_2_REMAINING_0	
-5-	9.8	GET_COLA -- BIN_1_REMAINING_0	
-6-	9.6	GET_DIET -- BIN_2_REMAINING_1	
-7-	9.5	GET_DIET -- GIVE_DIET_OUT	
-8-	9.4	GET_COLA -- BIN_1_REMAINING_1	
-9-	9.3	BIN_1_REMAINING_0 -- EMPTY_1	
-10-	9.1	BIN_1_REMAINING_1 -- EMPTY_1	

Figure 16 Using timing constants in the Path Analyzer

Hint – 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.

Hint – Constraints should be added only where required. The dynamic delay estimation mentioned above adds work to the placement process. Each constraint specified slows the placement process. The design may have to be restructured in order to achieve the desired performance.

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.

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. The Placer does not modify fixed assignments by the designer.

Fixing the Placement of I/O Pads

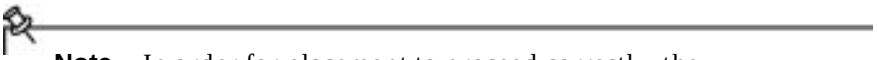
Design constraints sometimes require some or all I/O cell locations to be fixed. For example, an existing printed circuit board (PCB) might dictate a precise pinout. Alternatively, a high-speed PCB might require fixing a small number of critical pins in order to limit skew. The SpDE Placer can handle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

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

Fixing the Placement of Flip-Flops

Design constraints rarely require logic cell locations to be fixed. To allow designers a greater degree of flexibility, however, the Placer allows some or all of an Ultra38000's flip-flop macros to be fixed. One scenario that would dictate fixed flip-flops would be if 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 VHDL. (See the *Warp Reference Manual* for syntax information about this attribute.)



Note – In order for placement to proceed correctly, the **`fixed_ff`** attribute must be applied to each element of a bus and not to the bus as a whole.

Two flip-flop macros cannot be assigned to the same location. The naming convention for Ultra38000 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 Pin Assignment

Sometimes an I/O placement must be "locked down." This means that for all subsequent place and route runs, the I/O pins do not change their locations.

To back-annotate I/O pin locations, the following steps need to be followed:

- Run the *Back-Annotation Tool* in SpDE (which creates and writes out placement information to the *.atr* file). Amount of fixed placement is controlled by the options in the Back-Annotation options box (Figure 18). Two options are provided in the Fix Placement area: *I/O Cells* and *Flip-Flops*. When no option is checked, placement information for the cells which are currently fixed in the design is written out. When *I/O Cells* is checked all *I/O* cell locations are written out. When *Flip-Flops* is checked all flip-flop locations are written out.
- Select *Annotate...* from the Galaxy *File* menu to annotate to your control file, or select *Back Annotation...* from the *Cypress* menu in ViewDraw to annotate pins to your schematic.

4.4 Router

The Router employs highly optimized algorithms to connect I/O and logic cells using the Ultra38000 interconnect resources. The Router's optimization capabilities minimize routing delays and routing resources required. This finely-tuned arrangement produces excellent performance with high utilization.

Seed Value

Figure 15 shows the area of the *SpDE Tools Options* window that allows the user 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.

Interconnect Resources

The Router uses five 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**, **dual wires** and **segmented wires**.

Global Clock Networks: Two global clock networks exist on each FPGA in the Ultra38000 family. Both of these dedicated resources are capable of being connected to the clock, set, or reset of any flip-flop of the FPGA. Each of these clock networks can be driven by a clock generated anywhere on the chip or a clock coming from any pin. In the case where the design is using one of the global clock cells (CKPADs) located at a specific pin depending on the package used, nothing special needs to be done. In any other case, the user should instantiate a CLKBUF symbol to use the global clock network.

Array Clock Networks: In addition to the global clock networks, two array clock networks exist on each FPGA in the Ultra38000 family. Both of these dedicated resources are capable of being connected to the clock, set, or reset of any flip-flop in the array of the FPGA. Note that though these networks can not drive the *I/O* flip-flops, they are faster than the global clock networks. Each of these clock networks must be driven by one of the array clock cells (CKPADs) located at a specific pin depending on the package used.

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.

Dual Wires: Dual wires span two times the distance on the chip that segmented wires do (two logic cells). High-drive pads cannot drive dual wires either.

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 FPGA device. They are used for high-fanout nets or for 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 Ultra38000 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 Section 6, "Design Considerations: Speeding Up High-Fanout Nets" and its discussion on **double-buffering**.

Hint – SpDE warns the user if he uses 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.

4.5 Delay Modeler and Back-Annotation

The Delay Modeler performs precise post-layout delay calculations using state-of-the-art circuit analysis techniques. Processing the complete results of place and route, the Delay Modeler analyzes the results of packing, placement, and routing to determine intrinsic delays and routing delays for the entire design. The Back-Annotation tool writes these precisely calculated delays directly into the timing simulation output for accurate simulation results.

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 FPGA 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.

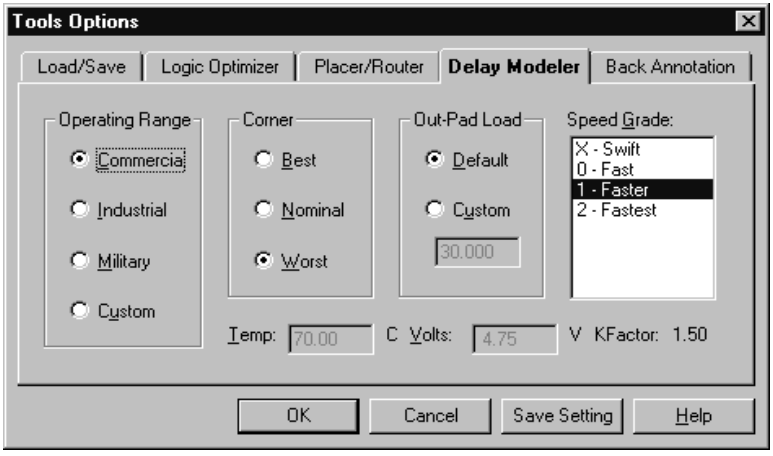


Figure 17 Delay Modeler options

Note – For a machine without a math co-processor, the run time of the Delay Modeler may be prohibitive as it uses floating point calculations.

The *Operating Range* radio button group controls the voltage and temperature ranges that the Delay Modeler uses. The default setting is *Commercial*. The *Custom* setting allows a user-specified temperature and voltage. See "Custom Temperature and Voltage," below.

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 always be performed at the *Worst* corner.

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 for all output pins.

The *Speed Grade* radio button group selects the Ultra38000 speed grade to be analyzed.

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, the user must edit the *spde.ini* file located in *$CYPRESS_DIR\spde\data*. For the Sun platform, this file should be named *.spderc* and should be in the home directory. The following lines should be changed:

Table 4 Edits to spde.ini and .spderc files

PC: *spde.ini*	**SUN:** *.spderc*
[delay modeler]	**...**
...	**delay modeler.customvccbest=5.0**
CustomVCCBest=5.0	**delay modeler.customvccnominal=5.0**
CustomVCCNominal=5.0	**delay modeler.customvccworst=5.0**
CustomVCCWorst=5.0	**delay modeler.customtempbest=25.0**
CustomTempBest=25.0	**delay modeler.customtempnominal=25.0**
CustomTempNominal=25.0	**delay modeler.customtempworst=25.0**
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.

Simulation 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 with *Warp3* is ViewSim.) The simulator in SpDE can be specified by selecting the *Back Annotation* Tab item from the *Tools Options* menu (see Figure 18).

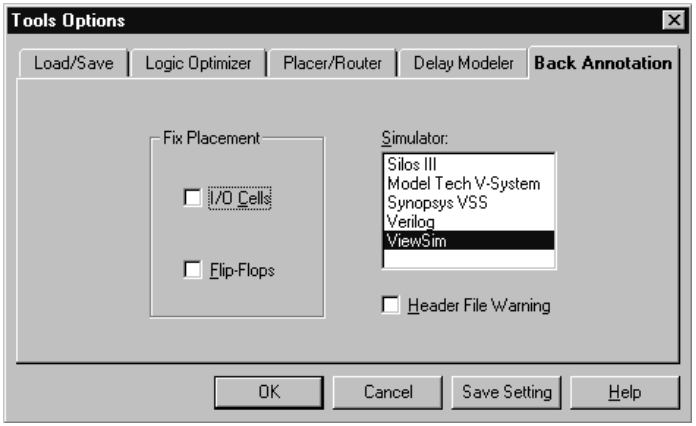


Figure 18 Simulator options for Back-Annotation

If the default simulator is changed by making a selection from the *Simulator* window, a click on the *Save Setting* button will write this information into the *spde.ini* file (*.spderc* for Sun workstations). Table 5 lists the files the Back-Annotation tool creates for each of several simulator settings.

Note – After changing the simulator, the user must still run the Back-Annotation tool to create the simulation netlist.

The Verilog® simulator and the Silos III simulator also require a primitive file, which describes the functionality of the primitive components specified in the *design.vq* file. This primitive file is design-independent, and is shipped with SpDE. The filename for this primitive file is *$CYPRESS_DIR\spde\data\qlprim.v*, where *$CYPRESS_DIR* is the directory in which the *Warp* software is installed.

Similarly, the Model Tech V-System simulator requires mtiprim.vhd primitive file and the Synopsys VSS simulator requires vssprim.vhd primitive file. Both of these primitive files reside in the *$CYPRESS_DIR\spde\data* directory.

Table 5 Files created during back-annotation

Simulator Setting	Files Created	Function
Verilog	design.*vq* *design.sdf*	verilog netlist delay back-annotation file
ViewSim	*design.vl* *design.dtb*	intermediate file for spde2vl delay back-annotation file
Synopsis VSS	design.*vhq* *design.sdf*	vhdl netlist delay back-annotation file
Model Tech V-System	design.*vhq* *design.sdf*	vhdl netlist delay back-annotation file
Silos III	design.*vq* *design.sdf*	verilog netlist delay back-annotation 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 icon that executes the spde2vl program from the Cockpit is labeled *pASIC-Vsim*. This icon is only available with the *Warp2sim* and *Warp3* products.

The Back-Annotation tool also supports the back-annotation of pin placement information back to the source design. For information on this process, see Section 4.3, “Placer.”

4.6 Automatic Test Vector Generator

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

The ATVG tool makes use of an internal scan path in Ultra38000 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.

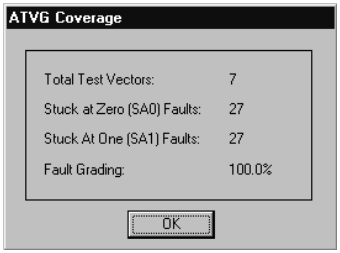


Figure 19 Sample ATVG Coverage window

Testing Overview

A major problem encountered in testing FPGAs is the inability to access directly 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 20. The truth table for this function is given in Table 6.

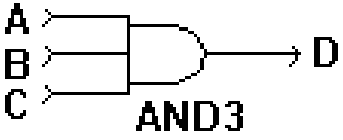


Figure 20 Stuck-at-Fault example

There are eight potential faults in this circuit, representing each node (A, B, C, D) stuck-at-0 and stuck-at-1. 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.

Table 6 Truth table for 3-input AND gate

A	B	C	D	Vector	Tests
0	0	0	0	1	D for SA1
0	0	1	0	2	D for SA1
0	1	0	0	3	D for SA1
0	1	1	0	4	A, D for SA1
1	0	0	0	5	D for SA1
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

By applying appropriate inputs to the circuit, the SA0 and SA1 faults may be detected at node D. Table 7 lists these inputs and their expected responses. Sets of input conditions and resultant output states are commonly referred to as **test vectors**.

The table indicates the fault(s) detected for each test vector.

Table 7 Detected faults

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:

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

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 counted 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 the table.

Design Considerations

The Ultra38000’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.

- Avoid combinatorial loops. The ATVG tool cannot test combinational loops such as those shown in Figure 21. Logic driven by or driving these loops may be untestable.
- 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 of the flip-flops in the FPGA device 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 21.

  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. A flip-flop clock, clear, or set driven by a bi-directional pin (BiPAD) also reduces fault coverage.

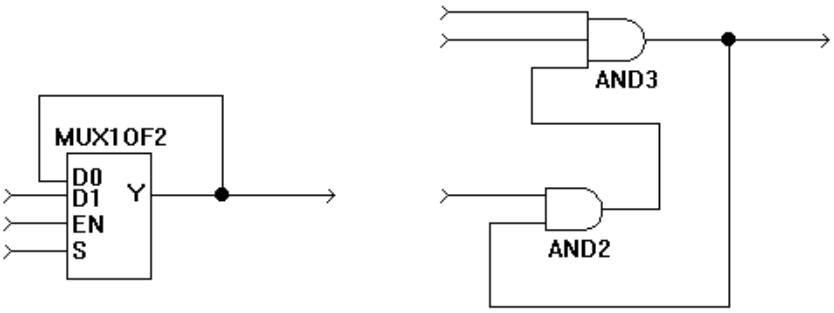


Figure 21 Examples of combinatorial loops (feedback)

4.7 Sequencer

The Sequencer is the tool that generates the programming file used to program the Ultra38000 devices.

5 SpDE Analysis Tools

After automatic placement and routing, SpDE allows the user to view logic cells and examine the timing of critical paths. Two tools are available for these purposes:

- Physical Viewer
- Path Analyzer

The following section discusses how to highlight critical nets for physical viewing using the Physical Viewer, as well as the calculation of critical timing parameters such as clock skew, operation frequency, setup time, and hold time using the Path Analyzer.

5.1 Highlight Net

Highlight Net mode helps the user analyze a design by highlighting and un-highlighting nets. Highlight Net mode cannot be used until a design has been placed and routed.

To open the *Highlight Nets* dialog box, select the *Highlight Net* item from the *View* menu in SpDE. Selecting *Highlight Net* brings up the *Highlight Net* dialog box (Figure 22) 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.

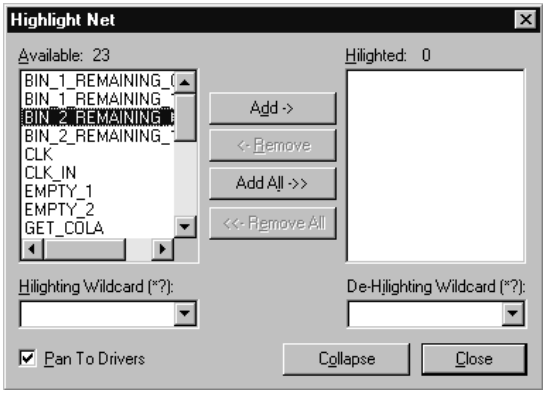


Figure 22 Highlight Nets dialog box

To highlight nets, select one or more nets from the box on the left, or specify a net name in the *Highlighting Wildcard* field on the left side of the dialog box, and click on the *ADD* 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 *De-Highlighting Wildcard* field on the right side of the dialog box, and click on the *REMOVE* button.

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

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

All nets can be added to the highlight list by clicking on the *ADD ALL* button.

All nets can be removed from the highlight list by clicking on the *REMOVE 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 *Close*. 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.

To shrink the window click on *Collapse* button. This makes more viewer area visible. To expand it again double-click on the Highlight Net title bar.

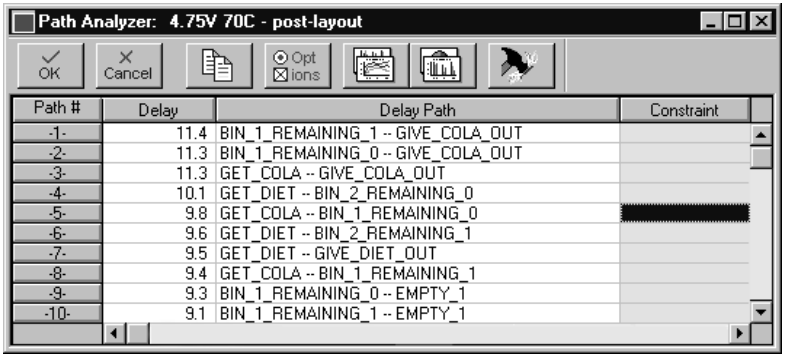
Pan to Net Driver

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.

5.2 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. The Path Analyzer performs static timing analysis of the circuit delays from the Delay Modeler. The Path Analyzer offers automatic analysis of all signals or of a user-specified subset of signals in the completed design. Working closely with the Physical Viewer, the Path Analyzer instantly identifies critical paths for optimization. Once the critical path has been identified, the user 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 23).



Path #	Delay	Delay Path	Constraint
-1-	11.4	BIN_1_REMAINING_1 -- GIVE_COLA_OUT	
-2-	11.3	BIN_1_REMAINING_0 -- GIVE_COLA_OUT	
-3-	11.3	GET_COLA -- GIVE_COLA_OUT	
-4-	10.1	GET_DIET -- BIN_2_REMAINING_0	
-5-	9.8	GET_COLA -- BIN_1_REMAINING_0	
-6-	9.6	GET_DIET -- BIN_2_REMAINING_1	
-7-	9.5	GET_DIET -- GIVE_DIET_OUT	
-8-	9.4	GET_COLA -- BIN_1_REMAINING_1	
-9-	9.3	BIN_1_REMAINING_0 -- EMPTY_1	
-10-	9.1	BIN_1_REMAINING_1 -- EMPTY_1	

Figure 23 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.

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 -1- to +1+ (assuming path number “1”) 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 24), which appears when the user clicks the *Options* button in the Path Analyzer window.

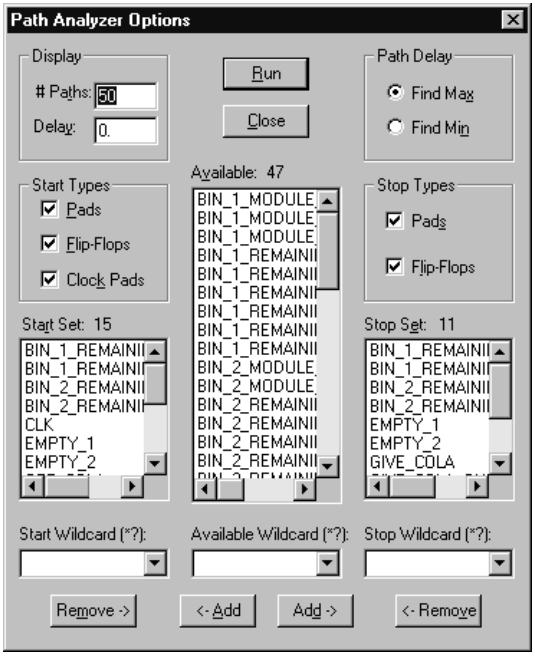


Figure 24 Path Analyzer Options dialog box

The *Run* button at the top of the window re-runs the Path Analyzer with the newly specified options. The *Close* 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. This selection does not change the operating conditions. (In other words, it does not change worst-case commercial to best-case commercial.) *Find Max* lists signals in order of longest delay to shortest. *Find Min* lists signals in order of smallest delay to longest.

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 interpreting the results of the Path Analyzer easier.

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. Deselecting one of these check boxes removes all of the appropriate nets from the desired set. For example, if none of the *Start Types* check boxes are selected, then 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. Deselecting 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 *ADD* buttons below the *Available* list box. Clicking on the left *ADD* button adds the selected nets to the *Start Set* list box, while clicking on the right *ADD* 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 *REMOVE* button below the list box involved.

Groups of nets can be selected using the combo buttons below each list box. In Figure 24, for example, the bus IB[0:3] can be selected by clicking in the combo button just below the *Available* list box and typing IB* or IB[?]. 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 one 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. SpDE’s Graph menu, when the Path Analyzer window is selected, can be used to create two types of graphs: *Path vs. Delay* and *Delay Histogram* graphs. The Path vs. Delay graph is shown in Figure 25.

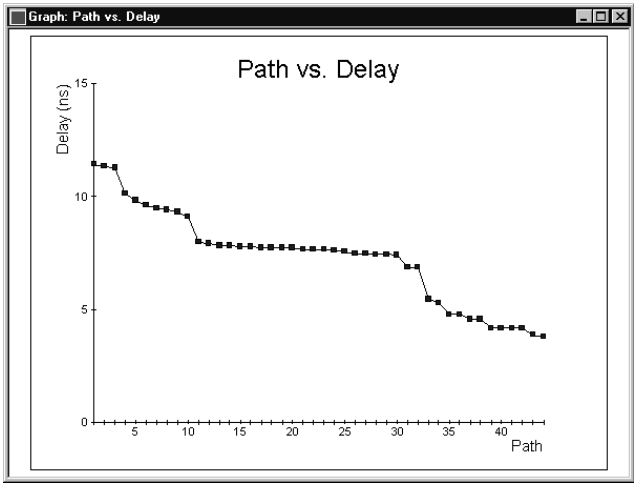


Figure 25 Path vs. Delay graph

The Path vs. Delay graph shows the path delays on the Y-axis and the path numbers on the X-axis. Double-clicking on points in this graph has the same highlighting effect as double-clicking on 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 can be customized with the *Graph/Options* menu command from the main window in the *Path Analyze*r mode. They can be printed by selecting the graph and selecting the *Print* menu item from the *File* menu.

Key Calculations

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

Hint – The results of these calculations will always be conservative, 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.

- Set the *Path Delay* radio button to *Find Max*.
- In the *Start Types* check box group, activate only the *Clock Pad* check box. The clock pad must be used as a clock not an input.
- In the *End Types* check box group, activate only the *Flip-Flops* check box.
- Click the *Run* button to execute the Path Analyzer.
- 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 conservative, 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.

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

$$F_{max} = 1/(critical\_path + clock\ SKEW)$$

The designer must determine the critical path in his design. This will come from a knowledge of the circuit function and its implementation. The designer may have to use some analysis to determine which paths are the frequency determining paths. Many designs contain false paths; therefore, the maximum delay path listed in the path analyzer may not be (and usually is not) the frequency determining path. False paths may include the following:

- Data paths with multiplexing and with various data sources and destinations where the longest path through the logic is never used.
- Long paths purposely given to signals which arrive long before they are required. Such paths are encountered in counters where high order stages reach their state long before the low order stage finally triggers a toggle in the counter.

Clock skew must be chosen for the critical path or paths identified. The largest skew identified is not necessarily the one to be used in the equation above. The number used must be the skew relevant to the critical delay path.

Setup Time

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

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

- Make a note of the first path listed; call it *pads_to_ffs*.
- Click the Path Analyzer's *Option* button to bring up the *Path Analyzer Options* dialog box.
- Set the *Path Delay* radio button to *Find Min*.
- In the *Start Types* check box group, activate only the *Clock Pad* check box.
- In the *End Types* check box group, activate only the *Flip-Flops* check box.
- Click the *Run* button to execute the Path Analyzer.
- 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 conservative, 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.

- Set the *Path Delay* radio button to *Find Min*.
- In the *Start Types* check box group, activate only the *Pads* check box.
- In the *End Types* check box group, activate only the *Flip-Flops* check box.
- Click the *Run* button to execute the Path Analyzer.
- Make a note of the first path listed; call it *pads_to_ffs*.
- Click the Path Analyzer's *Option* button to bring up the Path Analyzer Options dialog box.
- Set the *Path Delay* radio button to *Find Max*.
- In the *Start Types* check box group, activate only the *Clock Pad* check box.
- In the *End Types* check box group, activate only the *Flip-Flops* check box.
- Click the *Run* button to execute the Path Analyzer.
- 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.

6 Design Considerations: Speeding Up High-Fanout Nets

This section describes several techniques for speeding up the performance of designs created by the *Warp* system's SpDE tools. For more information, refer to Chapter 9, "Synthesis," in the *Warp User's Guide*.

For high-fanout, timing-critical nets, designers should consider improving design performance using buffering techniques. 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:

- double buffering
- split buffering
- selective buffering
- paralleling
- pipelining

6.1 Double Buffering

The Ultra38000 architecture allows two sources to drive a net 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 26 is an example of double buffering in a schematic.

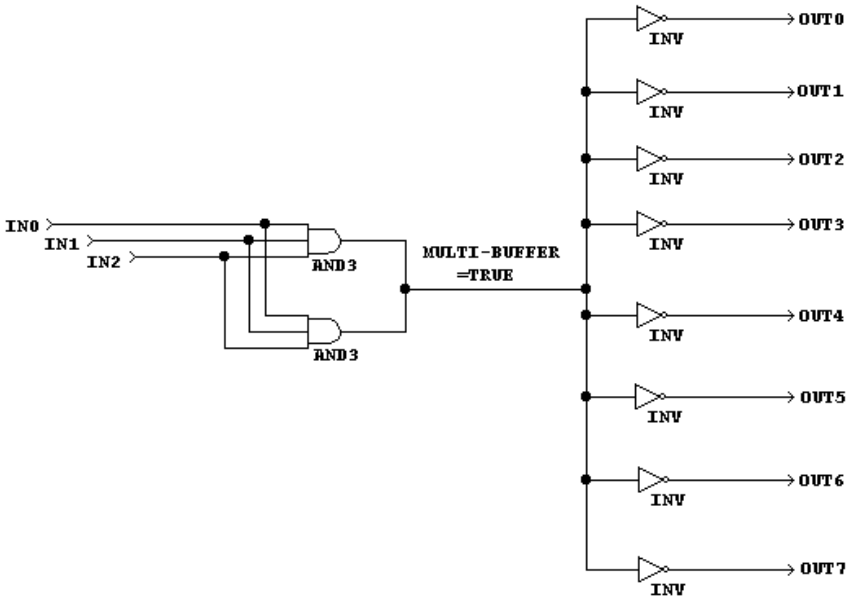


Figure 26 Double buffering example

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 (PAfrag_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. An example of double buffering in a VHDL source file is the following:

```
-- Resolution function for wired-or.  Used to create
-- legal VHDL for double-buffering techniques
-- employed for Ultra8000.
-------------------------------------------------------
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;
```

6.2 Split Buffering

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

Figure 27 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.

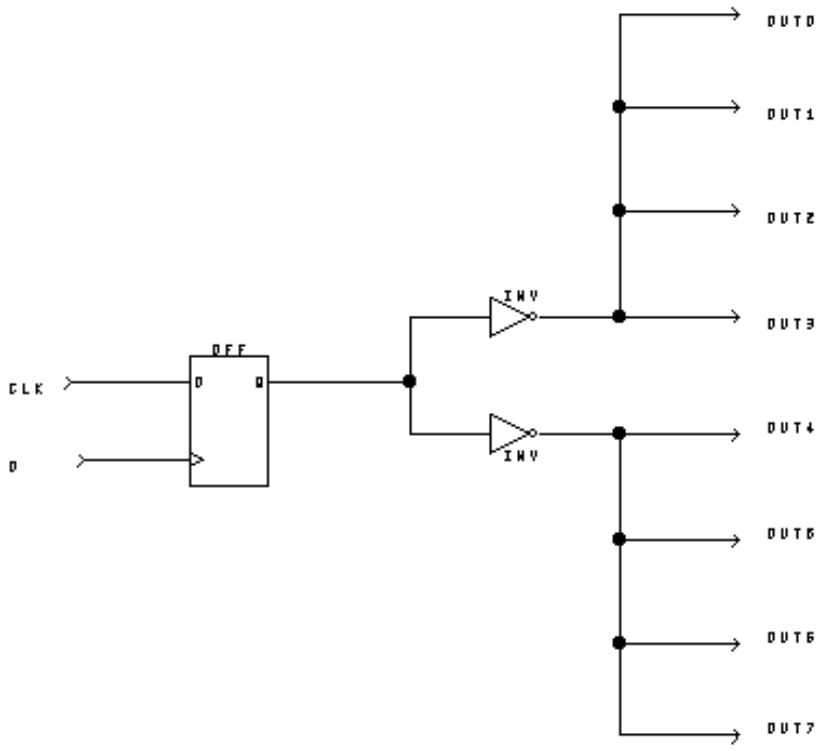


Figure 27 Circuit demonstrating split buffering

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 27) is generally employed only with fanouts of 16 or greater.

6.3 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 28 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 restructures the circuit, so that the DFF drives a fanout of two without adding any logic cell delay in the critical path.

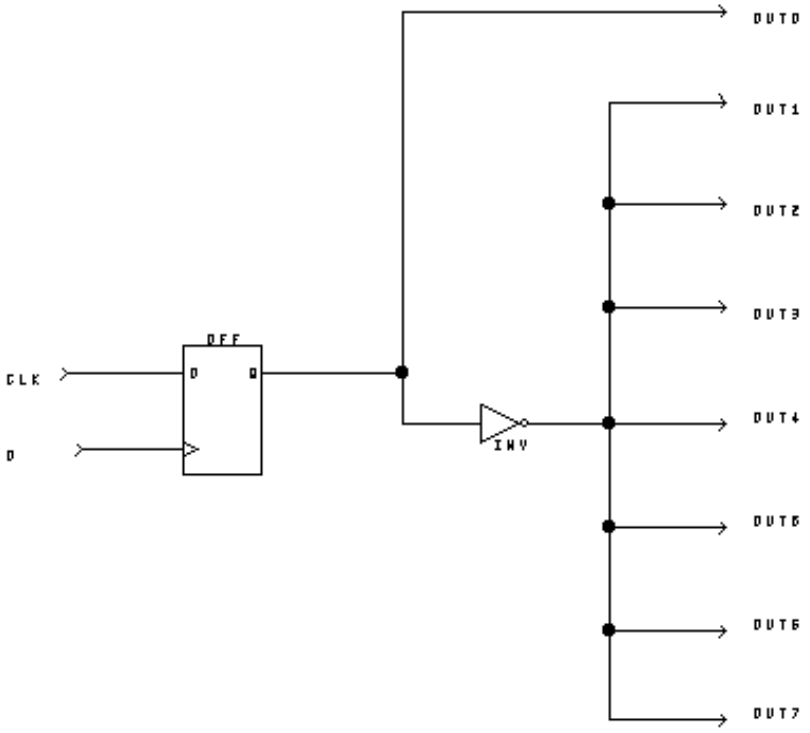


Figure 28 Circuit demonstrating selective buffering

Hint – 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.

6.4 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 that use of buffers causes. Paralleling is an alternative that does not introduce this added delay.

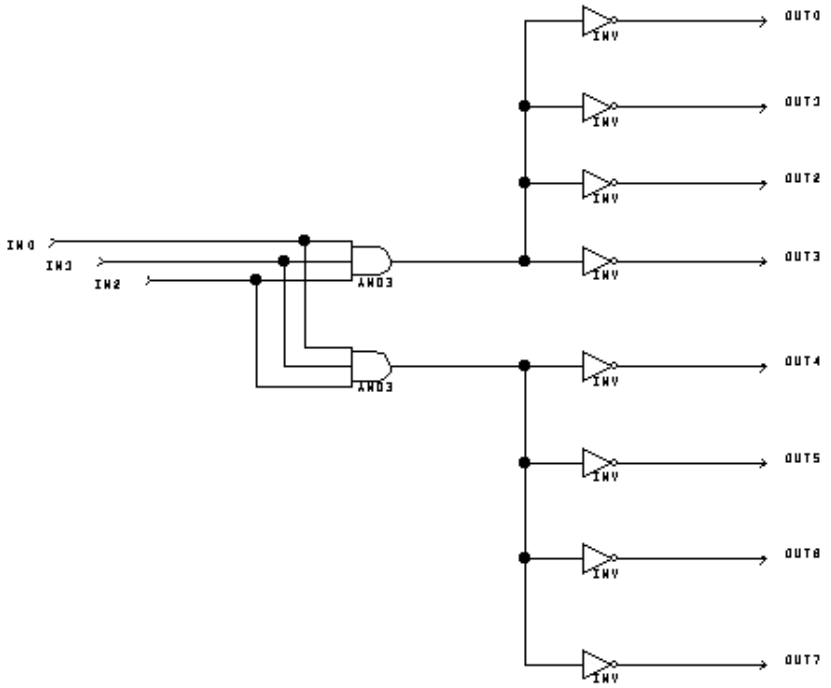


Figure 29 Circuit demonstrating paralleling

Figure 29 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.

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

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

6.5 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 Ultra38000 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.

For more information on achieving high performance or high utilization in designs, see Chapter 9, “Synthesis.” in the *Warp User’s Guide*.

Index

S

T

V

W

Z