Mentor Graphics Interface Guide

The Xilinx logo shown above is a registered trademark of Xilinx, Inc.

ASYL, FPGA Architect, FPGA Foundry, NeoCAD, NeoCAD EPIC, NeoCAD PRISM, NeoROUTE, Timing Wizard, TRACE, XACT, XILINX, XC2064, XC3090, XC4005, XC5210, and XC-DS501 are registered trademarks of Xilinx, Inc.

The shadow X shown above is a trademark of Xilinx, Inc.

All XC-prefix product designations, A.K.A Speed, Alliance Series, AllianceCORE, BITA, CLC, Configurable Logic Cell, CoolRunner, CORE Generator, CoreLINX, Dual Block, EZTag, FastCLK, FastCONNECT, FastFLASH, FastMap, Fast Zero Power, Foundation, HardWire, IRL, LCA, LogiBLOX, Logic Cell, LogiCORE, LogicProfessor, MicroVia, MultiLINX, PLUSASM, PowerGuide, PowerMaze, QPro, RealPCI, RealPCI 64/66, SelectI/O, SelectRAM, SelectRAM+, Silicon Xpresso, Smartguide, Smart-IP, SmartSearch, Smartspec, SMARTSwitch, Spartan, TrueMap, UIM, VectorMaze, VersaBlock, VersaRing, Virtex, WebFitter, WebLINX, WebPACK, XABEL, XACT*step*, XACT*step* Advanced, XACT*step* Foundry, XACT-Floorplanner, XACT-Performance, XAM, XAPP, X-BLOX, X-BLOX plus, XChecker, XDM, XDS, XEPLD, Xilinx Foundation Series, XPP, XSI, and ZERO+ are trademarks of Xilinx, Inc. The Programmable Logic Company and The Programmable Gate Array Company are service marks of Xilinx, Inc.

All other trademarks are the property of their respective owners.

Xilinx, Inc. does not assume any liability arising out of the application or use of any product described or shown herein; nor does it convey any license under its patents, copyrights, or maskwork rights or any rights of others. Xilinx, Inc. reserves the right to make changes, at any time, in order to improve reliability, function or design and to supply the best product possible. Xilinx, Inc. will not assume responsibility for the use of any circuitry described herein other than circuitry entirely embodied in its products. Xilinx, Inc. devices and products are protected under one or more of the following U.S. Patents: 4,642,487; 4,695,740; 4,706,216; 4,713,557; 4,746,822; 4,750,155; 4,758,985; 4,820,937; 4,821,233; 4,835,418; 4,855,619; 4,855,669; 4,902,910; 4,940,909; 4,967,107; 5,012,135; 5,023,606; 5,028,821; 5,047,710; 5,068,603; 5,140,193; 5,148,390; 5,155,432; 5,166,858; 5,224,056; 5,243,238; 5,245,277; 5,267,187; 5,291,079; 5,295,090; 5,302,866; 5,319,252; 5,319,254; 5,321,704; 5,329,174; 5,329,181; 5,331,220; 5,331,226; 5,332,929; 5,337,255; 5,343,406; 5,349,248; 5,349,249; 5,349,250; 5,349,691; 5,357,153; 5,360,747; 5,361,229; 5,362,999; 5,365,125; 5,367,207; 5,386,154; 5,394,104; 5,399,924; 5,399,925; 5,410,189; 5,410,194; 5,414,377; 5,422,833; 5,426,378; 5,426,379; 5,430,687; 5,432,719; 5,448,181; 5,448,493; 5,450,021; 5,450,022; 5,453,706; 5,455,525; 5,466,117; 5,469,003; 5,475,253; 5,477,414; 5,481,206; 5,483,478; 5,486,707; 5,486,776; 5,488,316; 5,489,858; 5,489,866; 5,491,353; 5,495,196; 5,498,979; 5,498,989; 5,499,192; 5,500,608; 5,500,609; 5,502,000; 5,502,440; 5,504,439; 5,506,518; 5,506,523; 5,506,878; 5,513,124; 5,517,135; 5,521,835; 5,521,837; 5,523,963; 5,523,971; 5,524,097; 5,526,322; 5,528,169; 5,528,176; 5,530,378; 5,530,384; 5,546,018; 5,550,839; 5,550,843; 5,552,722; 5,553,001; 5,559,751; 5,561,367; 5,561,629; 5,561,631; 5,563,527; 5,563,528; 5,563,529; 5,563,827; 5,565,792; 5,566,123; 5,570,051; 5,574,634; 5,574,655; 5,578,946; 5,581,198; 5,581,199; 5,581,738; 5,583,450; 5,583,452; 5,592,105; 5,594,367; 5,598,424; 5,600,263; 5,600,264; 5,600,271; 5,600,597; 5,608,342; 5,610,536; 5,610,790; 5,610,829; 5,612,633; 5,617,021; 5,617,041; 5,617,327; 5,617,573; 5,623,387; 5,627,480; 5,629,637; 5,629,886; 5,631,577; 5,631,583; 5,635,851; 5,636,368; 5,640,106; 5,642,058; 5,646,545; 5,646,547; 5,646,564; 5,646,903; 5,648,732; 5,648,913; 5,650,672; 5,650,946; 5,652,904; 5,654,631; 5,656,950; 5,657,290; 5,659,484; 5,661,660; 5,661,685; 5,670,896; 5,670,897; 5,672,966; 5,673,198; 5,675,262; 5,675,270; 5,675,589; 5,677,638; 5,682,107; 5,689,133; 5,689,516; 5,691,907; 5,691,912; 5,694,047; 5,694,056; 5,724,276; 5,694,399; 5,696,454; 5,701,091; 5,701,441; 5,703,759; 5,705,932; 5,705,938; 5,708,597; 5,712,579; 5,715,197; 5,717,340; 5,719,506; 5,719,507; 5,724,276; 5,726,484; 5,726,584; 5,734,866; 5,734,868; 5,737,234; 5,737,235;

About This Manual

This manual explains how to use the Xilinx / Mentor Graphics Interface software with Mentor Graphics® software version C.4.

Before using this manual, you should be familiar with the operations that are common to all Xilinx software tools. These operations are covered in the *Quick Start Guide*.

For detailed tutorials showing how to use the Mentor Graphics Interface, see the *Mentor Schematic Tutorial* and the *Mentor Schematic-on-Top with VHDL Macro Tutorial* on the Xilinx Web site at `http://support.xilinx.com/support/techsup/tutorials/index.htm`.

Manual Contents

This manual covers the following topics.

- Chapter 1, "Introduction," describes the Mentor Graphics Design Manager™ Interface, the Xilinx design flow, key features, inputs and outputs, and the architectures with which they work.
- Chapter 2, "Getting Started," describes how to configure your system for the Mentor Graphics Design Manager, and how to invoke the Mentor Graphics Design Manager.
- Chapter 3, "Schematic Designs," describes how to use the Mentor Graphics Design Manager and Design Architect™ to design with pure schematic designs. It covers schematic design entry, functional simulation, implementation, and timing simulation.

- Chapter 4, "HDL Designs," describes how to use the Mentor Graphics Interface to design with pure HDL designs. It covers HDL design entry, functional simulation, implementation, and timing simulation.
- Chapter 5, "Mixed Designs with VHDL on Top," describes how to use the Mentor Graphics Interface to design with mixed schematic and VHDL designs with VHDL on Top. It covers design entry, functional simulation, implementation, and timing simulation.
- Chapter 6, "Mixed Designs with Schematic on Top," describes how to use the Mentor Graphics Interface to design with mixed schematic and VHDL designs with a schematic on top. It covers design entry, functional simulation, implementation, and timing simulation.
- Chapter 7, "Mentor / Xilinx Flow Manager," describes how to use the Mentor / Xilinx Flow Manager to guide you through the design process.
- Chapter 8, "Advanced Techniques," describes useful design and simulation techniques that were not covered in the other sections of this manual.
- Chapter 9, "Manual Translation," describes how to manually process your design from the operating system command line.

Additional Resources

For additional information, go to `http://support.xilinx.com`. The following table lists some of the resources you can access from this Web site. You can also directly access these resources using the provided URLs.

Resource	Description/URL
Tutorials	Tutorials covering Xilinx design flows, from design entry to verification and debugging `http://support.xilinx.com/support/techsup/tutorials/index.htm`
Answers Database	Current listing of solution records for the Xilinx software tools Search this database using the search function at `http://support.xilinx.com/support/searchtd.htm`

Resource	Description/URL
Application Notes	Descriptions of device-specific design techniques and approaches `http://support.xilinx.com/apps/appsweb.htm`
Data Book	Pages from *The Programmable Logic Data Book*, which contain device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging `http://support.xilinx.com/partinfo/databook.htm`
Xcell Journals	Quarterly journals for Xilinx programmable logic users `http://support.xilinx.com/xcell/xcell.htm`
Technical Tips	Latest news, design tips, and patch information for the Xilinx design environment `http://support.xilinx.com/support/techsup/journals/index.htm`

Conventions

This manual uses the following conventions. An example illustrates each convention.

Typographical

The following conventions are used for all documents.

- `Courier font` indicates messages, prompts, and program files that the system displays.

  ```
  speed grade: - 100
  ```

- **`Courier bold`** indicates literal commands that you enter in a syntactical statement. However, braces "{ }" in Courier bold are not literal and square brackets "[ ]" in Courier bold are literal only in the case of bus specifications, such as bus [7:0].

  ```
  rpt_del_net=
  ```

  `Courier bold` also indicates commands that you select from a menu.

  `File` → **`Open`**

- *Italic font* denotes the following items.

  - Variables in a syntax statement for which you must supply values

    `edif2ngd` *`design_name`*

  - References to other manuals

See the *Development System Reference Guide* for more information.

- Emphasis in text

  If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.

- Square brackets "[ ]" indicate an optional entry or parameter. However, in bus specifications, such as bus [7:0], they are required.

  edif2ngd *[option_name] design_name*

- Braces "{ }" enclose a list of items from which you must choose one or more.

  lowpwr ={**on**|**off**}

- A vertical bar " | " separates items in a list of choices.

  lowpwr ={**on**|**off**}

- A vertical ellipsis indicates repetitive material that has been omitted.

  ```
  IOB #1: Name = QOUT'
  IOB #2: Name = CLKIN'
  .
  .
  .
  ```

- A horizontal ellipsis "…." indicates that an item can be repeated one or more times.

  `allow block` *block_name loc1 loc2locn;*

Online Document

The following conventions are used for online documents.

- Red-underlined text indicates an interbook link, which is a cross-reference to another book. Click the red-underlined text to open the specified cross-reference.

- Blue-underlined text indicates an intrabook link, which is a cross-reference within a book. Click the blue-underlined text to open the specified cross-reference.

Contents

Chapter 7 Mentor/Xilinx Flow Manager

Chapter 8 Advanced Techniques

Chapter 9 Manual Translation

Chapter 1

Introduction

This chapter describes the Mentor Graphics® Design Manager™ interface, a Mentor Graphics tool enhanced by the addition of Xilinx features.

You can invoke all individual tools from the Xilinx-enhanced Design Manager or from the shell.

This chapter contains the following sections.

Architecture Support

The software supports the following architecture families in this release.

- Spartan™ / XL / -II
- Virtex™ / -E / -II
- XC9500™ / XL / XV

- XC4000™E / L / EX / XL / XV / XLA
- XC3000™A / L
- XC3100™A / L
- XC5200™

Platform Support

The Mentor interface is supported on Sun SPARCstations using the Solaris operating system versions 2.5 and 2.6. It is also supported on HP workstations using the HPUX operating system version 10.2.

Library Support

The following libraries are available in the Mentor interface.

- Unified Libraries, which contain the symbol models for schematic entry and simulation
- SimPrim library, which contains the symbol models for timing (EDDM) simulation
- VITAL VHDL SimPrim library for top-down timing simulation
- Verilog SimPrim library for top-down Verilog timing simulation

Features

The following sections describe the major features available in this release.

Mentor Software Release Support

This interface supports the Mentor C.4 software release.

HDL Support

This release offers a number of features that allow you to process a design through a VHDL or Verilog netlist.

ModelSim and QuickSim Pro

This release supports the ModelSim™ simulator, which simulates behavioral VHDL, Verilog, VHDL-based, and Verilog-based gate-level designs composed of SimPrim elements. In addition, LogiBlox elements can be simulated at the behavioral level.

It also supports QuickSim Pro™ for mixed mode simulations for schematic-based and VHDL-based designs. QuickSim Pro can invoke ModelSim to simulate VHDL-based elements, or quicksim to simulate Unified Libraries elements.

Note This documentation assumes that you are using ModelSim and QuickSim Pro. However, QuickHDL and QuickHDL Pro™ provide the same functionality as ModelSim and QuickSim Pro. If you are using QuickHDL instead of ModelSim or QuickHDL Pro in place of QuickSim Pro, see the "ModelSim" section for details on how to use QuickHDL and QuickHDL Pro in place of ModelSim and QuickSim Pro.

Co-simulation may require that you use a specific version of the ModelSim software with Mentor C.4. Check with your Mentor representative for details.

VHDL Gate-Level Simulation Support

This release supports VHDL simulation, including IEEE-standard 1076.4 VHDL libraries of SimPrim models. Xilinx implementation tools output timing simulation VHDL netlists by using structural VHDL models of SimPrim VHDL models and an SDF file.

Verilog Gate-Level Simulation Support

This release supports Verilog simulation, including Verilog libraries for use with SimPrim models. Xilinx implementation tools output timing Verilog netlists by using structural Verilog models with SimPrim Verilog models and an SDF file.

Links to EDIF from Synthesis Tools

The Mentor interface can accept XNF and EDIF files from various synthesizers that are compatible with the Xilinx implementation software. These files can be directly submitted to the Xilinx Design Manager for placement and routing of the design.

You can also simulate these EDIF or SXNF files by submitting them to the pld_edif2sim or pld_xnf2sim utility, which creates EDDM components for use with pld_quicksim.

In addition, after place and route, you can output VHDL and Verilog netlists, which can be submitted to ModelSim for simulation with SDF files providing the back-annotation information.

Mentor Design Manager

The Mentor Graphics Design Manager is an easy-to-use interface that represents applications and design files as icons. You can now perform many tasks in the Design Manager window that were previously done at the operating system level. The Design Manager runs in a window on your workstation display and makes it easy for you to invoke applications and to manage designs, files, and directories. The Design Manager lets you do these tasks by using graphical point-and-click actions. You can run applications by selecting an application icon, or a design object icon and a menu item.

Note A design object consists of the files and directories that make up your design.

The Xilinx script, pld_dmgr, configures the Design Manager for the creation, implementation, and simulation of Xilinx designs. This manual describes only the Xilinx-configured Design Manager; refer to Mentor Graphics documentation for a more comprehensive description of the Mentor Design Manager.

The Design Manager includes a Tools window, a Navigator window, and a Design Manager palette, as shown in the following figure.

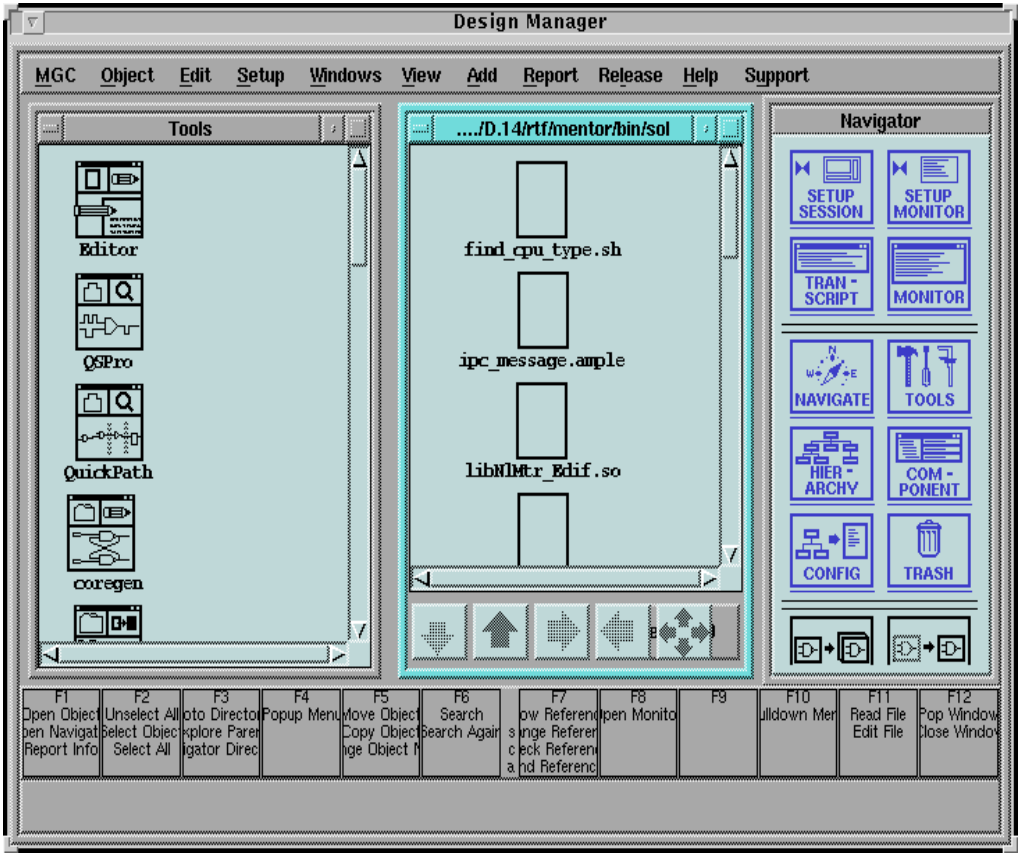


Figure 1-1 Mentor Design Manager Window

The Tools window contains icons representing all the Mentor Graphics and Xilinx applications that you need to execute the steps in the design flow. The Navigator window contains design object icons, including original schematics as well as files created during translation and simulation. This window makes it easy to access files in different directories. The Design Manager palette provides easy access to the most commonly used Design Manager menu items.

The remainder of this section briefly describes the icons in the Tools window and the Mentor programs they represent. The tools with names that begin with PLD are configured through scripts for working with Xilinx designs.

Coregen (CORE Generator)

This is a stand-alone Xilinx tool for generating VHDL and Verilog models of CORE modules. Schematic models can be created by invoking CORE Generator from within pld_da under the Xilinx Libraries Palette menu.

Editor

The Editor icon represents the Mentor Graphics Notepad editor. Notepad is a full-featured, window-based text editor. For more information on Notepad, refer to the Mentor Graphics *Notepad User's and Reference Manual*.

Exemplar

The Exemplar™ icon opens the Leonardo™ Spectrum synthesis tool.

Flow_mgr (Mentor/Xilinx Flow Manager)

The Mentor / Xilinx Flow Manager is a dialog box that provides a visual guide of the steps you need to perform for five common design flows. Each step contains buttons to launch the appropriate tool and to display a visual record of your progress in the flow. It does not automatically perform the steps for you. It lists the steps in the correct order that you need to perform. For each step there is a button that launches the appropriate tool. When you are finished with the tool, you click the Finished button for that step and the description for that step changes to indicate that it is finished.

Gen_Arch

Gen_Arch creates a VHDL architecture from a Mentor schematic (EDDM) component for use in mixed schematic and HDL simulations within QuickSim Pro.

ModelSim

ModelSim™ (vsim) is Mentor's simulator for behavioral VHDL, Verilog, or VHDL-based and Verilog-based gate-level designs composed of SimPrim elements.

QuickHDL previously provided this same functionality as ModelSim. The design flows in this user guide are based on ModelSim. If you have not upgraded to ModelSim and are still using QuickHDL, you can substitute QuickHDL into your design flows as described below.

ModelSim and QuickHDL have the same functionality, but the commands you use to control these tools are different. The Mentor Design Manager contains icons for QHDL as well as ModelSim so you can access either tool depending on what you have installed.

The following table provides a mapping between ModelSim and QuickHDL commands. In the procedures in this manual that use ModelSim commands, you can substitute QuickHDL by substituting the corresponding QuickHDL commands in those procedures.

ModelSim Commands	QuickHDL Commands
vlib	qhsim
vlib	qhlib
vmap	qhmap
vcom	qvhcom
vlog	qvlcom

Pld_da

Pld_da is Mentor's Design Architect®, a schematic editor configured for Xilinx designs. The Xilinx-configured Design Architect is identical to the Mentor Graphics version except for the addition of a Xilinx library of primitives, macros, and utilities such as Convert Design. For more information on creating Xilinx designs with Design Architect, refer to the "Design Entry" section of the "Schematic Designs" chapter in this manual and the *Mentor Graphics Schematic Design Tutorial* on the Xilinx Web site at `http://support.xilinx.com/support/techsup/tutorials/index.htm`. For a more detailed description of Design Architect commands and processes, refer to the Mentor Graphics *Design Architect User's Manual*.

Pld_dsgnmgr

The Mentor Design Manager interface contains a Pld_dsgnmgr icon for the Xilinx Design Manager. Pld_dsgnmgr is the Xilinx Design Manager, which implements the design. You can access any individual Xilinx tool from the Xilinx Design Manager.

Pld_dve

Pld_dve is the Mentor Graphics Design Viewpoint Editor (DVE) configured for Xilinx designs. When you invoke this application from within the Mentor Design Manager, a dialog box appears and you are asked to create either a simulation or custom viewpoint. Refer to the "Functional Simulation" section of the "Schematic Designs" chapter and the "Timing Simulation for Schematic Designs" section of the "Schematic Designs" chapter in this manual for more information on pld_dve. For detailed information on DVE, refer to the Mentor Graphics *Design Viewpoint Editor User's and Reference Manual*.

Pld_edif2sim

Pld_edif2sim is a utility that converts a Mentor, Synopsys, or other Xilinx compatible EDIF files into a Mentor EDDM single-object simulation model, VHDL netlist, or Verilog netlist. Pld_edif2sim is for functional simulation only.

Pld_edif2tim

Pld_edif2tim is the Mentor EDIF netlist reader, which converts a placed and routed EDIF netlist to a Mentor single-object EDDM file that can be submitted to pld_quicksim for timing simulation.

Logiblox (LogiBLOX GUI)

This is a stand-alone Xilinx tool for generating VHDL and Verilog models of LogiBlox components. Schematic models can be created by invoking LogiBLOX from within pld_da under the Xilinx Libraries Palette menu.

Pld_men2edif

Pld_men2edif converts a Mentor schematic to a hierarchical EDIF netlist that is ready for implementation.

Pld_xnf2sim

Pld_xnf2sim is a utility that converts an unrouted XNF file to a Mentor EDDM single-object simulation model. This conversion can only be done on chip-level XNF files with EXT records, not on lower level modules embedded in a schematic. VHDL or Verilog simulation models can also be generated. Pld_xnf2sim is for functional simulation only.

Pld_quicksim

Pld_quicksim is an interactive logic simulator that performs functional or timing simulation on your designs. For more information on pld_quicksim, refer to the "Functional Simulation" section of the "Schematic Designs" chapter, the "Timing Simulation for Schematic Designs" section of the "Schematic Designs" chapter, and the *Mentor Graphics Schematic Design Tutorial* on the Xilinx Web site at http://support.xilinx.com/support/techsup/tutorials/index.htm. For a detailed description of pld_quicksim, refer to the Mentor *Graphics QuickSim II User's Manual*.

Pld_sg

Pld_sg is the Mentor schematic generator (SG), which creates a schematic from an EDDM single object netlist. You can use this tool to generate a schematic for the timing simulation netlist.

QuickPath

QuickPath™ performs static and slack timing analysis on designs. For more information on QuickPath, refer to the "Performing a Timing Analysis" section of the "Schematic Designs" chapter. For a detailed description of QuickPath, refer to the Mentor Graphics *QuickPath User's and Reference Manual*.

Renoir

Renoir™ is the Mentor Graphics HDL graphical design tool for generating Verilog and VHDL.

Tau

Tau is a board-level timing analysis tool from Mentor Graphics designed to do system timing analysis, as opposed to transmission line analysis which is the focus of IS Analyzer / Floorplanner. Tau checks that timing constraints such as setup and hold requirements on component inputs are met. For more details, refer to the "Using TAU" section of the "Advanced Techniques" chapter.

LogiBLOX Modules

You can enter a schematic using LogiBLOX symbols along with other Unified Libraries elements. For schematics, invoke LogiBLOX from within pld_da by using the Xilinx Libraries menu (**`Libraries`** → **`Xilinx Libraries`** → **`Logiblox`**). In addition, EDDM simulation models are automatically created for LogiBLOX symbols during symbol creation.

For VHDL or Verilog LogiBlox models, invoke LogiBlox from the pld_dmgr's tool window, or from the popup session window within pld_da.

CORE Generator Modules

You can enter a schematic using CORE Generator symbols along with other Unified Libraries elements. For schematics, invoke CORE Generator from within pld_da by using the Xilinx Libraries menu (**`Libraries`** → **`Xilinx Libraries`** → **`Coregen`**). In addition, EDDM simulation models are automatically created for CORE Generator symbols during symbol creation.

For VHDL or Verilog CORE Generator models, invoke CORE Generator from the pld_dmgr's tool window, or from the popup session window within pld_da.

EDIF

This release supports EDIF 2 0 0 for design implementation. Refer to the Xilinx EDIF specification for supported constructs.

Cross-Probing

Cross-probing is a way of cross-referencing between the original schematic and the timing simulation model after placement and routing. Once a Mentor design is translated, expanded, mapped, placed, and routed, you can extract the back-annotation information and create a hierarchical EDIF netlist. After you convert this EDIF netlist to an EDDM model using pld_edif2tim, you submit it to pld_dve to create a viewpoint and then to pld_quicksim for timing simulation. The resulting data base preserves the design hierarchy, and although it is created in terms of the SimPrim library, most of the original net names are still available. You enable cross-probing by invoking QuickSim with the -cp option. This option invokes pld_dve as well as pld_quicksim. You then open the original design viewpoint in pld_dve and view the desired design sheet. If you display the original schematic in pld_dve, you can select nets on the original schematic and view them in the QuickSim trace window.

See the “Cross-Probing” section of the “Schematic Designs” chapter for more details on cross-probing.

Timing Simulation

This release supports back-annotated timing simulation after placement and routing. Pld_edif2tim translates the routed EDIF file to an EDDM single-object netlist.

Schematic Generator

The schematic generator is a utility that you can optionally use to generate a hierarchical schematic from a back-annotated EDDM model. This is not a required step since you can instead use cross-probing with the back-annotated EDDM model and the original schematic for simulation without generating a back-annotated schematic. You can invoke the schematic generator from within the design manager or from a shell by typing pld_sg. You must have a Mentor schematic generator license in order to use this tool.

Timing Constraints

You can add timing constraints to the schematic as properties. You can also place them in a UCF (user constraints file) that NGDBuild can process. If a conflict arises between the timing information in the EDIF file and in the constraints file, the information in the constraints file prevails.

Design Flows

You use different PLD design flows for performing design entry, implementation and simulation depending on whether you use schematic design entry or HDL design entry.

In either case, the easiest and most automatic way is to use the application icons in the Design Manager window. You can also run the various programs in the design flow manually from the UNIX shell. The shell commands are described in the "Manual Translation" chapter.

The Mentor interface supports the following design flows.

- Schematic entry with the Unified Libraries components, LogiBLOX symbols, CORE Generator symbols or a combination of these symbols.
- Schematic entry with Unified Library components with some models expressed in Xilinx compliant EDIF or XNF
- Top-down HDL (Verilog / VHDL) design entry and synthesis
- Mixed schematic and VHDL design with VHDL on top
- Mixed schematic and VHDL design with schematic on top

Schematic Entry Design Flows

The schematic entry design flows are illustrated in the following three figures.

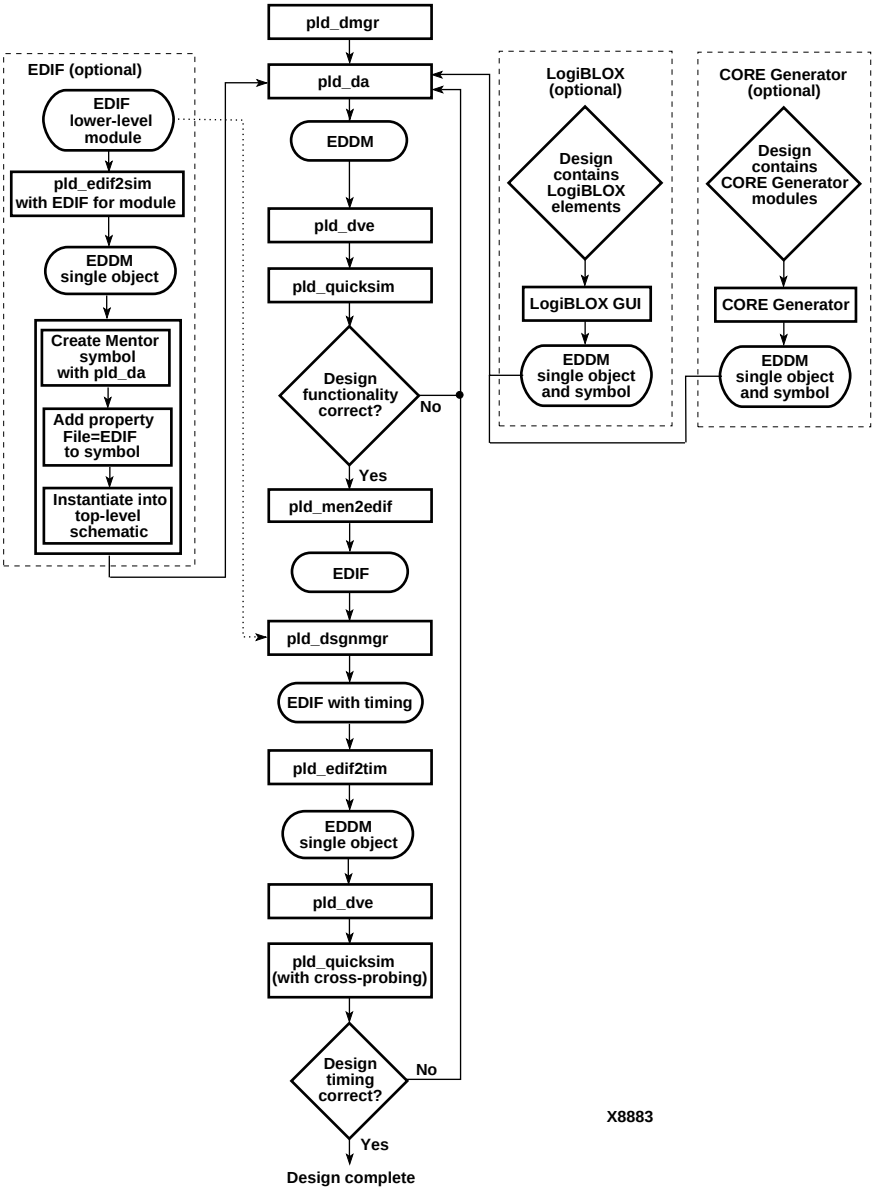


Figure 1-2 Schematic Design Entry Including EDIF-Based and LogiBLOX Modules

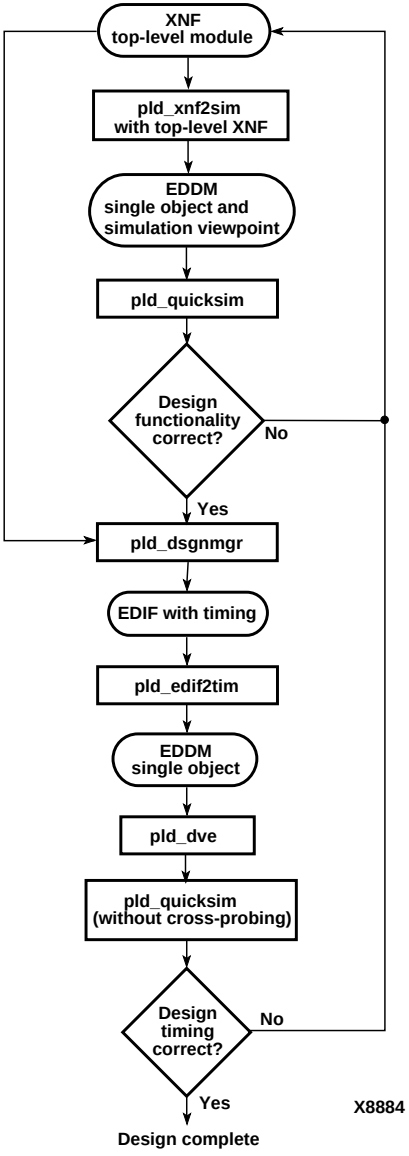


Figure 1-3 Design Entry with XNF Top-Level Module

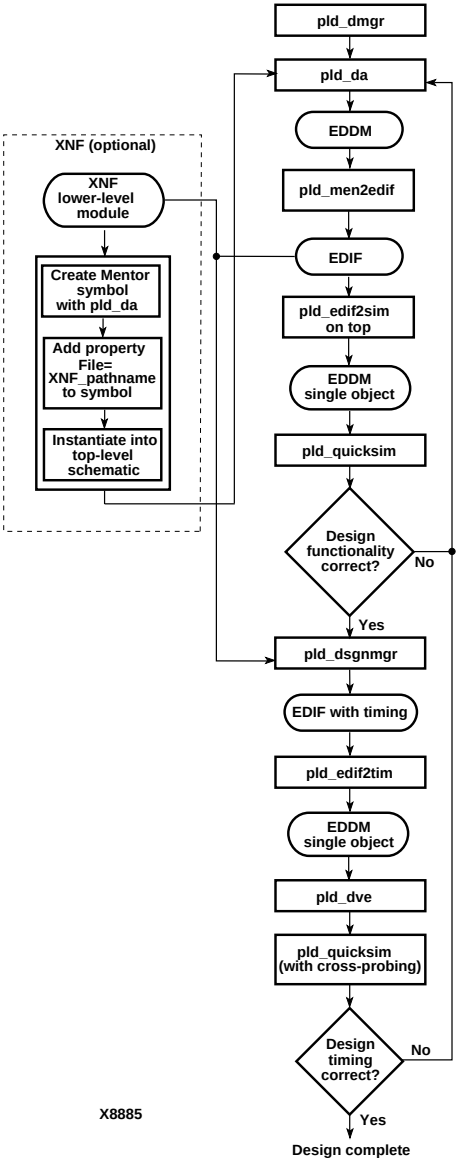


Figure 1-4 Schematic Design Entry with XNF Module

HDL Entry

The following figure shows the design flow for VHDL and Verilog design entry and synthesis for all supported technologies.

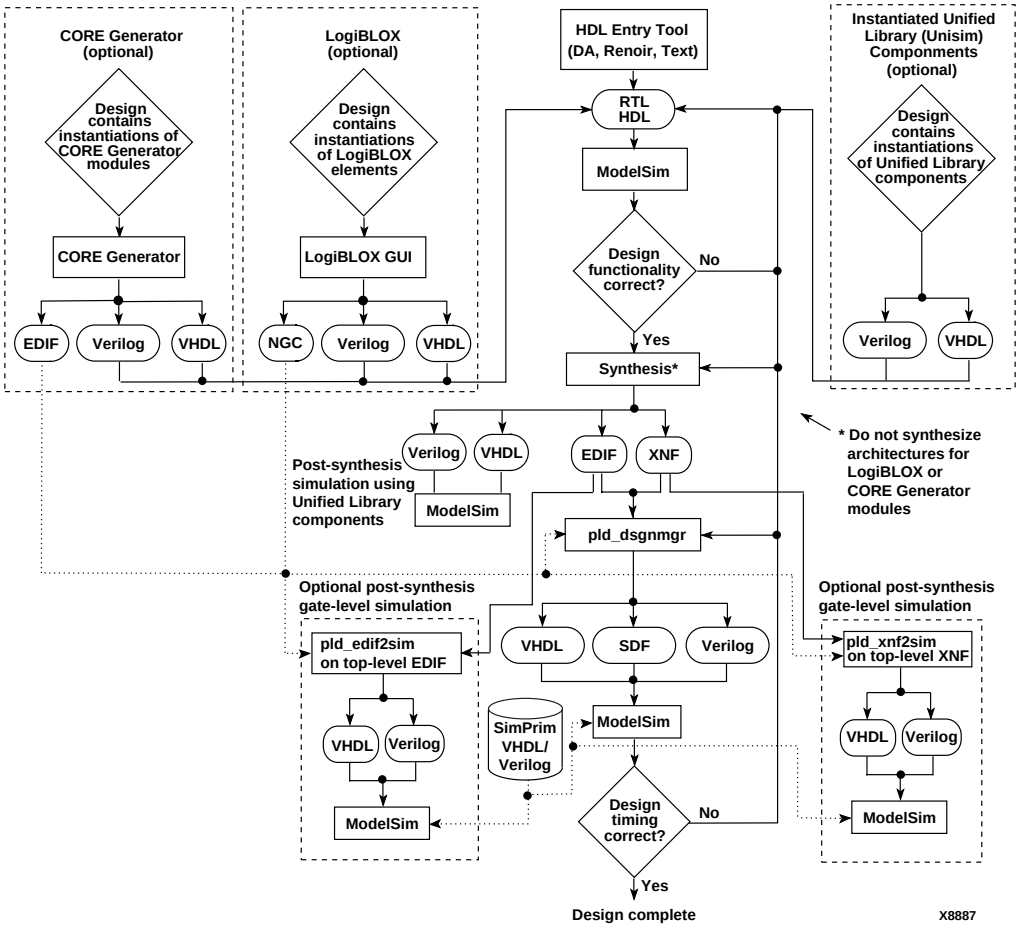


Figure 1-5 HDL (Verilog/VHDL) Design Entry and Synthesis

Mixed Schematic and VHDL Flow with VHDL on Top

The design flow for design entry of a top-level VHDL design with a schematic sub-module embedded within is illustrated in the following figure.

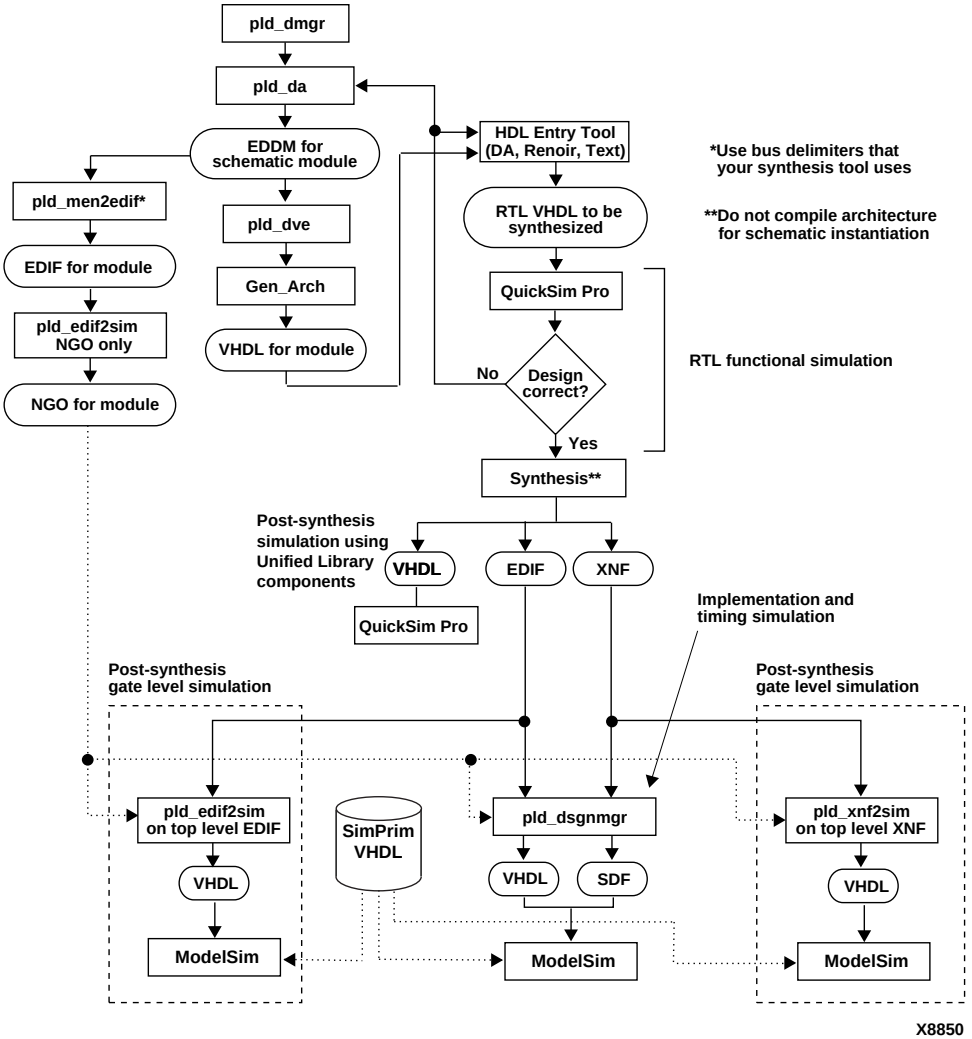


Figure 1-6 Mixed Schematic and VHDL Design with VHDL on Top

Mixed Schematic and VHDL Flow with Schematic on Top

The design flow for design entry using a mixture of schematics, VHDL, and Verilog is illustrated in the following figure.

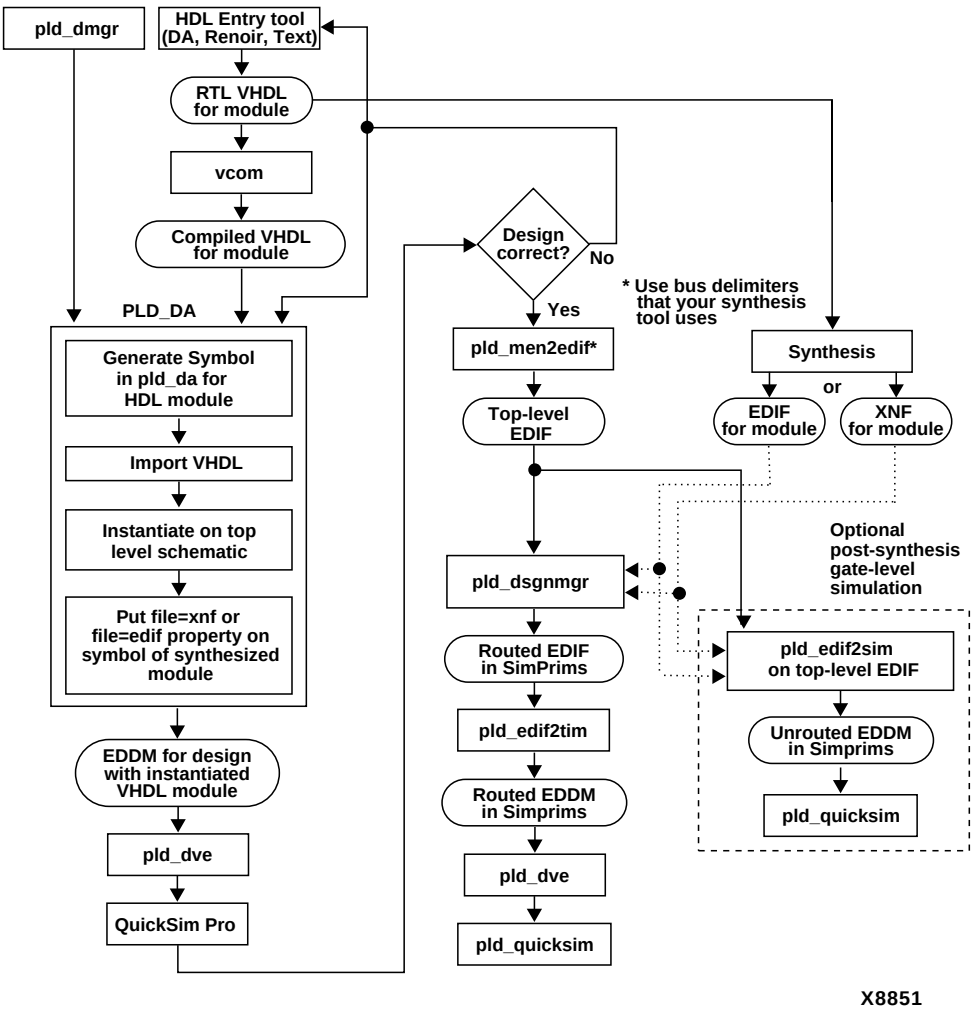


Figure 1-7 Mixed Schematic and VHDL Design with Schematic on Top

Inputs

The Mentor interface accepts netlists in EDIF or XNF format.

EDIF

You can submit an EDIF Level 2 0 0 netlist based on a design using Unified Libraries components. The following restrictions apply.

- Only the netlist and schematic types of EDIF are supported.
- Only one design per EDIF file is allowed.
- An EDIF file can contain one design component or multiple components. The EDIF2NGD utility converts each design component to an NGO file. NGDBUILD uses a top-level NGO file, which refers to the other NGO files, to create the NGD file.

XNF

The Mentor interface can accept one of the following XNF netlists.

- An XNF netlist created by third-party netlist writers that meet the specifications of XNF version 6.1
- An XFF netlist created by XNFMerge version 6.1
- An XTF netlist created by XNFPrep version 6.1

An XNF netlist can represent all or part of a design. To be included in the netlist of a schematic design, a component must be tagged with the FILE property indicating the path name of the XNF file.

If a lower module is expressed in XNF, the top level must be run through EDIF2SIM in order to create a simulation netlist. The lower-level XNF file can not be run through XNF2SIM by itself since its lack of EXT records prevents XNF2SIM from knowing which signals should become module pins.

Outputs

Xilinx Design Manager generates a back-annotated simulation netlist:

- SimPrim-based EDIF
- A structural VHDL SimPrim netlist and an SDF delay file
- A structural Verilog SimPrim netlist and an SDF delay file

Files

The following Xilinx specific files are involved in processing a design through the Mentor interface.

- The EDN file is a post-route EDIF netlist file that expresses timing in SimPrim library elements instead of Unified Libraries elements.
- The NCD file contains a representation of the physical design.
- The NGA file contains physical timing delay information.
- The NGD file contains a logical design hierarchy expressed in the Xilinx implementation primitives.
- The NGM file contains a representation of the logical design. It also contains optimization information.
- The NGO file contains netlist information in a proprietary data base format; it is a binary file.
- The SDF file contains timing delay information.
- The V file contains the structural design based on Verilog-based SimPrim models.
- The VHD file contains the structural design based on VHDL-based SimPrim models.
- The XNF file is the Xilinx netlist format used prior to the use of EDIF in the current release. In the current Mentor Interface flow, XNF is only used as an import format option.
- The PCF file is the physical constraints file.
- The UCF file is the User Constraint File for specifying the user's timing and placement constraints for place and route.

Tutorials

It is highly recommended that you perform the Mentor Interface tutorials provided on the Xilinx Web site to become familiar with the basic concepts of PLD design, verification, and implementation. The tutorials are located at `http://support.xilinx.com/support/techsup/tutorials/index.htm`.

Online Help

The Mentor interface contains online help which is available from each application's dialog box. Help contains information about the Mentor features offered in the interface but does not contain information about the Xilinx features. The Mentor software is supplied with a set of online manuals in PDF format. This online manual is the documentation for the Xilinx features.

Chapter 2

Getting Started

This chapter describes how to configure your system for the Mentor Graphics Design Manager, and how to invoke the Mentor Graphics Design Manager. This chapter contains the following sections.

Configuring Your System

Install the appropriate software and verify that your system is properly configured as described in the release notes that came with your software package. When you have finished the installation, verify that your .cshrc or setup file contains lines similar to the following.

```
setenv XILINX location_of_Xilinx_software
set path=($XILINX/bin/sol \
$XILINX/mentor/bin/sol $MGC_HOME/bin $path)
```

Note Path names of directories will vary. (For example, $XILINX/bin/sol would be $XILINX/bin/hp if you are running the Xilinx software on an HP workstation.) For more information on paths and environment variables, refer to the release notes that came with your software package.

XILINX is the directory where all Xilinx software is located.

Make sure that the following Mentor Graphics specific variables are set correctly.

- **EXEMPLAR**

  This variable should point to the location where the Exemplar software is installed. For example.

  ```
  setenv EXEMPLAR /products/leonardo.ver4_2
  ```

- **LCA**

  In addition to instantiating it in the file pointed to by MGC_LOCATION_MAP, the LCA environment variable should point to the directory where the Xilinx/Mentor Graphics software is installed, typically $XILINX/mentor/data.

- **LD_LIBRARY_PATH**

  This variable is used by Mentor Graphics and Xilinx programs. On Solaris platforms, this variable is set as follows.

  ```
  setenv LD_LIBRARY_PATH $MGC_HOME/shared/
  lib:$MGC_HOME/lib:$XILINX/bin/sol:/usr/openwin/
  lib: ${LD_LIBRARY_PATH}
  ```

  On HP workstations, the variable is SHLIB_PATH. Leave out /usr/openwin/lib.

- **MGC_GENLIB**

  This should point to the Mentor Graphics gen_lib library, normally $MGC_HOME/gen_lib.

- **MGC_HOME**

  This should point to the Mentor Graphics software tree.

- **MGC_LOCATION_MAP**

  This variable should point to a valid location map file.

  Each component in a design contains a reference indicating where it resides on the disk or network. All components in designs created in the Mentor Graphics C.4 environment reference the variable $LCA, while back-annotated timing models reference the variable $SIMPRIMS. It is also important that the $LCA and $SIMPRIMS variables be instantiated, but not defined, in the file pointed to by $MGC_LOCATION_MAP. With all these elements, the location-map file should, at a minimum, look like:

```
MGC_LOCATION_MAP_1
(empty line)
$MGC_GENLIB
(empty line)
$LCA
(empty line)
$SIMPRIMS
(empty line)
```

The MGC_LOCATION_MAP_1 line indicates that this is a version 1 location-map file. (You can also use the version MGC_LOCATION_MAP_2, which adds features such as outside file inclusion.) The three soft names with blank lines indicate that the Mentor Graphics software should pull the associated values from the parent environment.

Refer to the Mentor Graphics documentation for more information on location maps.

- **MGC_WD** (Optional)

  This variable should point to the working directory. You can have this variable always point to your current directory by setting it to “.”

  Xilinx tools ignore the MGC_WD variable.

- **MGLS_LICENSE_FILE**

  This variable must point to a valid FlexLM license file that lists the Mentor Graphics license daemon and licensed software features, as supplied by Mentor Graphics. A sample license file may begin as follows.

```
SERVER tequiero 9542df17 1700
DAEMON mgcld /tools/mentor/lib/mgcld
 /usr/local/data/mentor.opt
FEATURE falconfw_s 8.0 31-dec-1997 10 ...
```

- **MODELTECH**

  This variable should point to the directory where the Modeltech software is installed. For example,

  ```
  setenv MODELTECH /products/modeltech_ver5
  ```

- **MTI_HOME**

  This variable should point to the directory where the Modeltech software is installed. For example,

  ```
  setenv MTI_HOME /products/modeltech_ver5
  ```

- **RENOIRHOME**

  This variable should point to the directory where the Renoir software is installed. For example,

  ```
  setenv RENOIRHOME /products/renoir
  ```

- **SIMPRIMS**

  This points to the directory where Xilinx simulation models are located. This should be set to $LCA/simprims.

Refer to the release notes for additional information on paths and environment variables.

Invoking the Design Manager

To invoke the Design Manager from the operating system, type **`pld_dmgr`**.

The Design Manager window appears, as shown in Figure 1-1.

Invoking Applications in the Design Manager

You can use either an icon or the Navigator to invoke an application from the Design Manager.

Tools Window Icons

To use an icon to open an application, double-click the left mouse button on the icon in the Tools window.

A dialog box appears that allows you to set options, or the application is executed.

Navigator Window

If you want to load a specific design, you can also use another method of invoking an application.

1. Select the design object in the Navigator window with the left mouse button and press the right mouse button.
2. Select **Open** from the Navigator menu.
3. Select the appropriate application from the popup menu.

   Only the applications that can be executed on the selected object will be displayed in the popup menu.

   A dialog box appears that allows you to set options, or the application is executed.

Chapter 3

Schematic Designs

This chapter describes how to use the Mentor Graphics Design Manager and Design Architect to design with pure schematic designs. It contains the following sections.

Design Flows

Three pure schematic design flows are shown in the "Schematic Entry Design Flows" section in the "Introduction" chapter. This chapter describes how to work with designs using the pure schematic design flows.

Design Entry

The following subsections describe how to access and use the design entry tools.

Invoking Design Architect

You can use either the pld_da icon or the Navigator to invoke Design Architect from the Design Manager.

To invoke Design Architect with the pld_da icon in the Tools Window, double-click the left mouse button on the pld_da icon. A Design Architect window displays but without a schematic. You can use the Open Sheet icon in the Session Palette to open a schematic sheet.

If you want to load a specific design, you can invoke Design Architect from the Navigator as follows.

1. Select the design in the Navigator window and press the right mouse button.
2. Select **Open** → **pld_da** from the Navigator pop-up menu.

Adding Components

This section explains how to add Xilinx library components and describes how to use CORE Generator and LogiBLOX.

Adding Xilinx Library Components

To add Xilinx library components, perform the following steps.

1. To add a component from the Xilinx libraries, select **XILINX Libraries** from the Libraries pull-down menu.
2. In the Schematic Palette, click the desired technology library.

   Note Do not mix components from different technologies (families).
3. Click **BY TYPE** to select a category of element, or **ALL PARTS** to select a specific element.
4. Click the desired element, move the cursor to the desired location on the schematic, and click the left mouse button to place it.

Note For Virtex2, ten I/O primitives have default attributes when added to the schematic. The default attribute displays next to the primitive when placed on the schematic. For OBUF, OBUFDS, OBUFTDS, OBUFT, IOBUF, and IOBUFDS, there are default attributes for IOSTANDARD, DRIVE, and SLEW. For IBUF, IBUFDS, IBUFG, IBUFGDS, there is a default attribute for IOSTANDARD. For details on valid entries for each of these attributes, see the *Xilinx Libraries Guide*.

Bus Rippers

Bus rippers are Mentor Graphics-supplied special components that connect nets to specific signals on a bus. You can obtain bus rippers by selecting the rip component in the Logic submenu in the Unified Libraries. These components are the same as rip components in the MGC Digital Libraries gen_lib.

Xilinx Libraries

In Design Architect, the Xilinx Libraries menu contains the Unified Libraries. The Unified Libraries are a collection of libraries that conform to standards set for the appearance, function, and naming conventions of the library elements. This standardization allows you to easily convert from one Xilinx architecture to another. You should use the primitives and the macros in the Unified Libraries to create new designs. Refer to the *Xilinx Libraries Guide* for detailed information on the Xilinx Libraries.

Primitives and Macros

The Xilinx Libraries contain the following types of components.

- **Primitives**—These are pads and basic logic elements, such as gates, latches, flip-flops, buffers, and oscillators.
- **Soft macros**—These are schematics that contain primitives and other soft macros. Soft macros have pre-defined functionality and often have fixed mapping, placement, and routing to provide the most efficient use of resources and the fastest speed.

LogiBLOX

LogiBLOX allows you to synthesize common data functions such as addition, that are optimized for a particular family. Refer to the *LogiBLOX Guide* for information on LogiBLOX components.

CORE Generator System

The CORE Generator system allows you to use complex functions such as math functions, memories, or DSP functions that are optimized for a particular family. Refer to the *CORE Generator System User Guide* for information on CORE Generator modules.

You might like to create various functions using Xilinx Cores and put them on a schematic sheet in Mentor Design Architect, and simulate them using Mentor's QuickSim.

To place Core Generator modules into your design, follow these steps.

1. Invoke pld_da and open a schematic sheet for your design.
2. In the Xilinx library menu in the schematic sheet, click the Coregen... Palette menu item.

   The Create / Instantiate COREGEN Symbol dialog box opens as shown in the following figure.

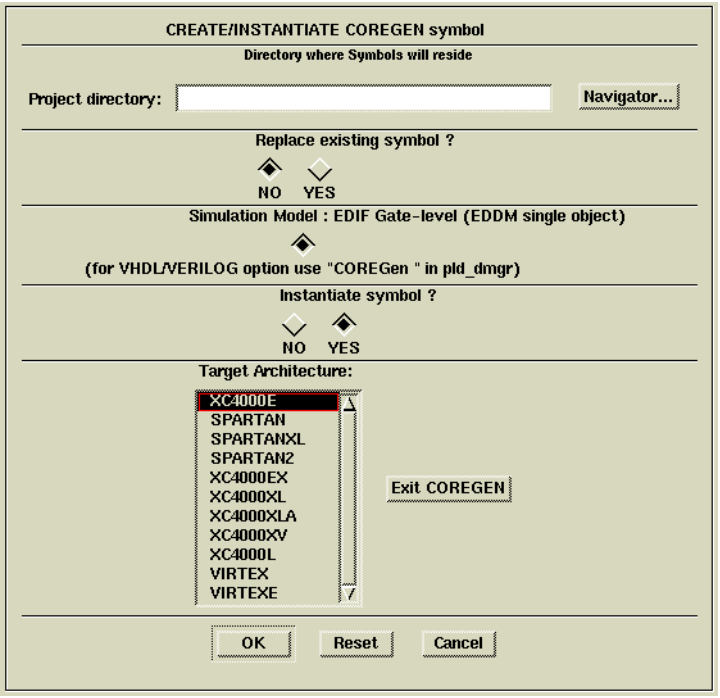


Figure 3-1 Create/Instantiate COREGEN Symbol

Note When you invoke Coregen from within the Schematic Editor, a revolving cursor indicates that Design Architect is waiting on the Coregen GUI and for you to create a core. At times, Coregen GUI may be hidden behind the Design Architect window. If the Coregen GUI does not come up, press control-s to stop the waiting process. When you press control-s, the revolving cursor stops moving but it does not return to the normal arrow shape. This does not mean that Design Architect is not functional; you can continue working in the Schematic Editor. To restore the cursor to the normal arrow, press control-k at any time. If you wish to continue waiting for interaction with Coregen GUI, press control-q to resume the waiting process.

The Create / Instantiate COREGEN Symbol dialog box contains the following fields.

- **Project directory**—This is the directory in which you would like the symbol component to reside. Xilinx refers to this directory as the symbol project directory.
- **Replace existing symbol**—This button determines if the new symbol can overwrite the same name symbol in the symbol project directory.
- **Simulation Model**—This switch cannot be changed. It indicates that the cores are created from EDIF files and read into Mentor to create EDDM SINGLE OBJECT simulation model.
- **Instantiate symbol**—This choice button determines if the created core symbol is to be placed on the sheet.
- **Target Architecture**—This is a list gadget from which the desired Xilinx family is selected.

3. Fill out this form and click **OK**.

   The CORE Generator system is invoked and Design Architect is disabled.

4. In the CORE Generator, select and generate the desired core.
5. Once the CORE Generator edif output is created, control will be back in the Design Architect, a symbol gets created and you are able to place the symbol on the schematic sheet.

6. Upon placement of the symbol an automatic popup message inquires if you would like to create another core. If you select YES, the control will be transferred back to CORE Generator and the same process repeats. If you select NO, the CORE Generator GUI automatically closes. This dialog box often appears on the top left corner of the schematic window.
7. After the CORE Generator closes, if you need to create another core, go back to step 2 above.

Properties

This section describes the properties that are unique to Mentor or that are required when working with Xilinx PLDs using Mentor.

Properties, or attributes, are instructions placed on symbols or nets in an FPGA or CPLD schematic that allow you to control aspects of software processing. They express information specific to each design, unlike run-time options entered in the Xilinx Design Manager.

This section describes the properties that are unique to Mentor schematics or that are required. The *Xilinx Libraries Guide* describes the other attributes that you can place on a Mentor schematic.

INST

Use the INST property to uniquely identify an instantiation of a component or symbol in a design. Design Architect assigns a default INST property to the symbol of each instantiation (I$1, I$2, and so forth), and the INST value is appended to the hierarchical path.

COMP

Use the COMP property to indicate that a simulation model exists for a primitive. All Xilinx primitives have a COMP property.

Do not place the COMP property on user symbols since COMP indicates that the symbol is a Xilinx library primitive.

CYMODE

Use the CYMODE property on the Carry Mode symbol to identify the mode for the dedicated carry logic in an XC4000 CLB.

INTERNAL

Use the INTERNAL property to identify unbonded IOBs.

Entering Timing Specifications

The Mentor netlist writer program (ENWRITE) converts all property names to lowercase letters, and the Xilinx netlist reader EDIF2NGD then converts the property names to uppercase letters. To ensure references from one constraint to another are processed correctly, observe these guidelines.

- A *TSidentifier* name should contain only uppercase letters on a Mentor Schematic (TSMAIN, for example, but not TSmain or TSMain).
- If a *TSidentifier* name is referenced in a property value, it must be entered in uppercase letters. For example, the TSID1 in the second constraint below must be entered in uppercase letters to match the TSID1 name in the first constraint.

```
TSID1 = FROM: gr1: TO: gr2: 50;
TSMAIN = FROM: here: TO: there: TSID1: /2;
```

Creating New Groups from Existing Groups

The Mentor netlist writer program (ENWRITE) converts all property names to lowercase letters, and the Xilinx netlist reader EDIF2NGD then converts the property names to uppercase letters. To ensure references from one constraint to another are processed correctly, observe these guidelines.

- Group names should contain only uppercase letters on a Mentor Schematic (MY_FLOPS, for example, but not my_flops or My_flops).
- If a group name appears in a property value, it must also be expressed in uppercase letters. For example, the GROUP3 in the first constraint below must be entered in uppercase letters to match the GROUP3 in the second constraint.

```
TIMEGRP GROUP1 = gr2: GROUP3;
TIMEGRP GROUP3 = FFS: except: grp5;
```

Functional Simulation

Functional simulation provides an effective means of identifying logic errors in your design before it is implemented in a Xilinx device. Since timing information for the design is not available, the simulator tests the logic in the design using unit delays. Finding errors before routing your design saves debugging time later in the design process.

You can functionally simulate XNF or EDIF based designs by using pld_xnf2sim or pld_edif2sim to convert the designs to a Mentor simulation model. The EDIF design must be Xilinx compatible and expressed in Unified Library components. The following figure illustrates the design flow for these types of designs.

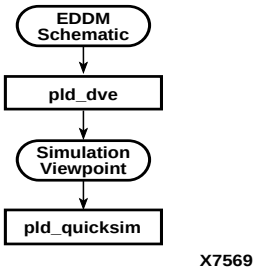


Figure 3-2 Functional Simulation Flow Diagram

The *Mentor Graphics Schematic Design Tutorial* on the Xilinx Web site at `http://support.xilinx.com/support/techsup/tutorials/index.htm` provides a detailed example of the steps involved in functional simulation.

Simulating Pure Schematic Designs

This section describes how to simulate purely schematic designs—designs that are composed solely of elements from the Unified Libraries and that have been entered through Design Architect. Performing functional simulation on a pure schematic design consists of creating a viewpoint in pld_dve from the schematic that you created in Design Architect and using pld_quicksim to simulate the design.

Creating the Viewpoint

After creating a schematic design with Design Architect and a Xilinx library, the next step in the functional simulation flow is to configure a viewpoint for the simulator. Without a correct simulation viewpoint, you will not be able to simulate your design. The viewpoint defines primitives and parameters for design evaluation and analysis.

Pld_dve invokes the Mentor Graphics Design Viewpoint Editor (DVE) to configure a viewpoint for Xilinx designs.

Create the viewpoint for the top-level component that was created in Design Architect.

1. To invoke DVE, double-click the left mouse button on the pld_dve icon in the Design Manager Tools window.

   Alternatively, you can select the top-level component in the Navigator window and click the right mouse button to invoke pld_dve.

   The dialog box shown in the following figure appears. For a more detailed description of DVE, refer to the *Mentor Graphics Design Viewpoint Editor Users Manual* and *Reference Manual*.

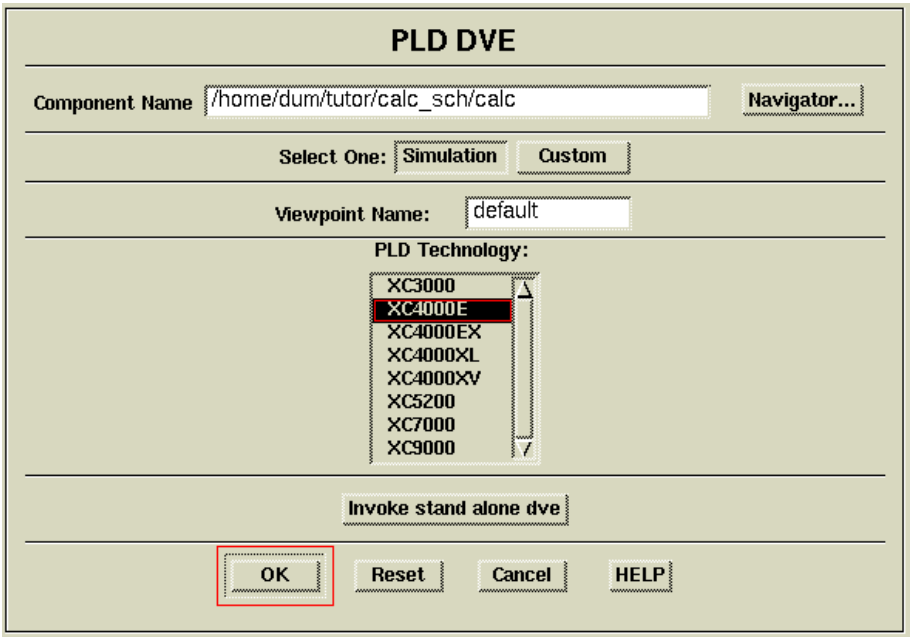


Figure 3-3 Pld_dve Dialog Box

2. Enter the design name in the Component Name field, or click **`Navigator`** to browse a list of design names. If you invoked pld_dve from the Navigator window, the component is already selected.

   If you click the Navigator, you can select the component name, and the corresponding viewpoint name will appear in the Viewpoint Name field.

3. In the Select One field, select **`Simulation`**.

   Select **`Custom`** if you want to open the selected viewpoint in DVE so that you can interact with it rather than accept the pld_dve default. Selecting Custom invokes Mentor's DVE and opens the named viewpoint. You could use this to select a different model for a specific sub-module.

4. In the Viewpoint Name field, you can enter the viewpoint name if you do not want to use the default viewpoint.

5. In the PLD Technology field, select a technology.

6. Click **Invoke Stand-Alone DVE** only if you want to invoke DVE to interact with Mentor's user interface instead.

   This command brings up the DVE window to allow you to customize the viewpoint. For information on customizing a viewpoint, see the Mentor Graphics DVE user documentation.

7. Select **OK** to start pld_dve.

   Pld_dve now generates a viewpoint with the same name as that entered in the Viewpoint Name field. It is in the format *component_name*/*viewpoint_name*.

   You can also access pld_dve from a UNIX shell.

   If you are converting a top-level XNF or EDIF netlist with pld_xnf2sim or pld_edif2sim, the simulation viewpoint is created for you automatically.

Simulating the Design

After creating the viewpoint, you can submit pure schematic designs to pld_quicksim for functional simulation.

1. To invoke pld_quicksim, double-click the left mouse button on the pld_quicksim icon in the Design Manager Tools window.

   Alternatively, you can select the top-level component in the Navigator window and click the right mouse button to invoke pld_quicksim.

   The PLD_QuickSim II dialog box, shown in Figure 3-4, appears on the screen. For more detailed information on the dialog box options, refer to the Mentor Graphics QuickSim documentation.

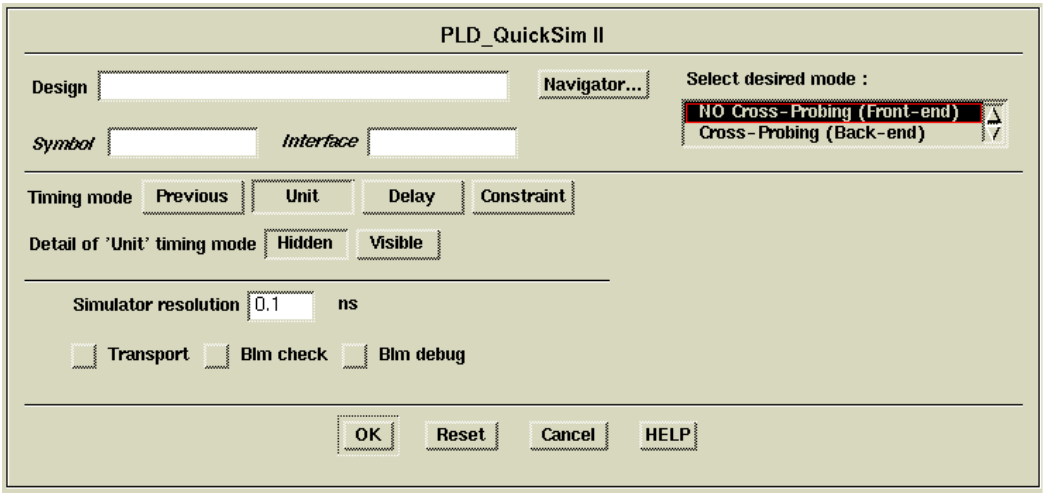


Figure 3-4 PLD_QuickSim II Dialog Box

2. In the Design field, enter the design name. If you selected the component in the Navigator window, the design name is already set.
3. In the Select Desired Mode box, click **No Cross-Probing**, if it is not already selected (This is the default setting).

   You can only select cross-probing for timing simulation for schematic designs, not for functional simulation. See the "Cross-Probing" section for more details about cross-probing.
4. In the Timing Mode field, select **Unit** for functional simulation.
5. In the Detail of "Unit" Timing Mode field, click **Hidden**.
6. In the Simulator Resolution box, enter the smallest unit of time that you want to be visible in the simulator.

   The smallest resolution allowed for Xilinx designs is 0.1 ns.
7. Click **OK**.

   Pld_quicksim now starts, and the QuickSim II window appears. The QuickSim II window functions as a waveform viewer; you can bring up the schematic and view the signals, or you can view the waveforms generated by the simulation. Consult the Mentor Graphics documentation for more information on how to view waveforms in this window.

Simulating Schematic Designs with LogiBLOX Elements or CORE Generator Modules

LogiBLOX creates a simulation model for LogiBLOX elements and the CORE Generator system creates a simulation model for CORE Generator modules. However, you must still create a viewpoint on the top-level design with pld_dve before functionally simulating the design. Follow the instructions in the "Creating the Viewpoint" section of the "Simulating Pure Schematic Designs" section in this chapter. Then submit the design to pld_quicksim, following the procedure given in the "Simulating the Design" section of the "Simulating Pure Schematic Designs" section in this chapter.

Simulating Schematic Designs with XNF Elements

To functionally simulate a pre-route XNF design, follow the steps in this section. The steps are illustrated in the following figure.

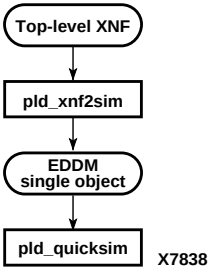


Figure 3-5 XNF Functional Simulation Flow

Creating the Design Component

Create the top-level design component as described in the "Creating the Design Component" section in this chapter. This provides an "anchor" for the converted design.

Converting the XNF File

The next step is to convert the XNF file to a simulation model.

1. In your schematic, create a symbol for each XNF element in your design.

2. Attach a FILE=*xnf_file_pathname* property to each symbol. (In this manual, file=*value* means to add the file property and set its value to *value*.)
3. Double-click the left mouse button on the pld_xnf2sim icon in the Design Manager Tools window.

   The resulting dialog box is shown in the following figure.

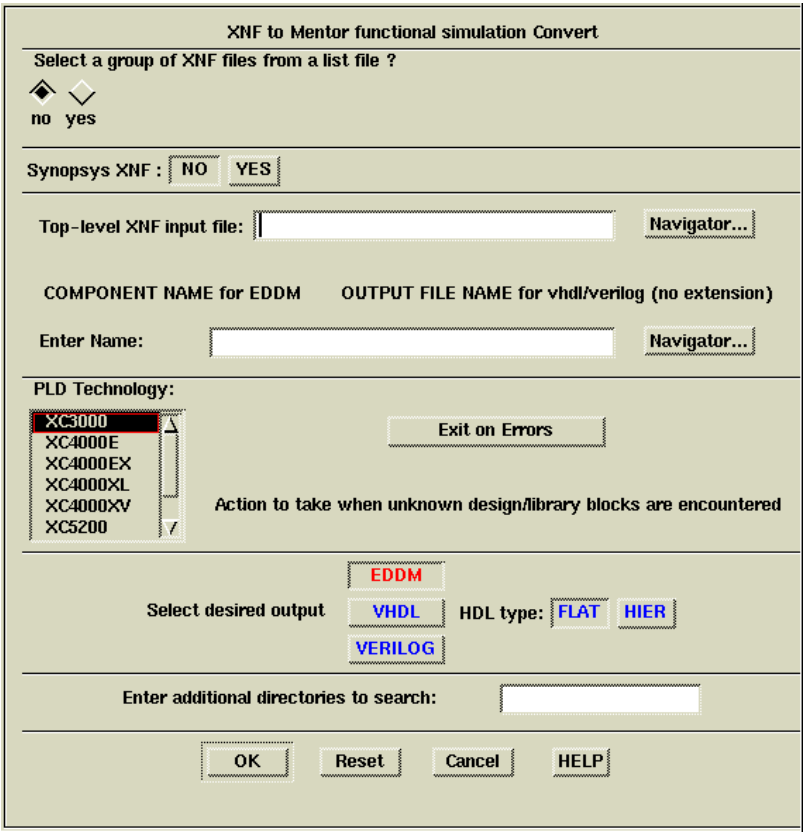


Figure 3-6 PLD XNF to Mentor Convert Dialog Box

Pld_xnf2sim uses all supporting XNF files from the directory in which the top-level XNF input file was submitted.

4. If the required XNF files are not in that directory, click **Yes** in the field asking "Select a group of XNF files from a list file?," and specify the path name of a file that lists the path names of all needed XNF files. Each path name is specified on a separate line in this file, for example.

   ```
   /x/y/z/abc.xnf
   /x/y/z/def.xnf
   ```

5. In the Synopsys XNF field, select **No** if the XNF does not come from Synopsys.

6. In the Top-level XNF Input File field, type the name of your top-level XNF file or click **Navigator** to find it.

7. In the Enter Name field, enter the name of the symbol that you created in step 1 or click **Navigator** to find it.

   Note If the symbol has not yet been created, a Mentor component is created with an EDDM-single-object model. At this point, you can use Design Architect to create a symbol for it. Click **Open Symbol** and specify the path name of this component. A symbol is automatically created. Check the symbol, add the file=*xnf_file_pathname* property, and save it if the XNF file represents the entire design (If it has EXT statements for IO pins.). However, if the XNF does not contain EXT statements, you must manually create the symbol and assign the pins. In this case, the simulation model (EDDM_single_object) created by pld_xnf2sim will not correspond with this symbol, and functional simulation must be done by converting the entire design to EDIF and submitting the EDIF to pld_edif2sim to create a top-level component and use pld_quicksim to simulate. This top-level component and all its submodules will be expressed in terms of SimPrim primitives rather than the Unified Library components used for design entry.

8. In the PLD Technology field, select the appropriate architecture.

9. Leave the Exit on Errors button enabled if you want the program to exit when it encounters an unresolved block. Otherwise, click the Exit on Errors button and it changes to Continue (Ignore Errors).

10. In the Select Desired Simulation Model field, select **EDDM**.

11. In the “Enter additional directories to search” field, enter all the directory pathnames that the program should search to find EDIF, XNF, and NGO files that define blocks in your design that have the File property on them.
12. Click **OK**.

    This procedure produces a single-object simulation model for the specified symbol component.

Creating the Viewpoint

If you are converting a top-level XNF or EDIF netlist with pld_xnf2sim or pld_edif2sim, the simulation viewpoint is created for you automatically.

Simulating the Design

The rest of the simulation procedure is the same as that described in the “Simulating Pure Schematic Designs” section earlier in this chapter.

Simulating Schematic Designs with EDIF Elements

To functionally simulate a pre-route EDIF design, follow the steps in this section. The steps are illustrated in the following figure.

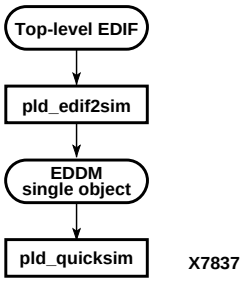


Figure 3-7 EDIF Functional Simulation Flow

Creating the Design Component

Create the top-level design component as described in the "Creating the Design Component" section in this chapter. This provides an "anchor" for the converted design.

Converting the EDIF File

The next step is to convert the EDIF file to a simulation model.

1. In your schematic, create a symbol for each EDIF element in your design.
2. Attach a FILE property with the value of *edif_file_pathname* to each symbol.
3. Double-click the left mouse button on the pld_edif2sim icon in the Design Manager Tools window.

   The resulting dialog box is shown in the following figure.

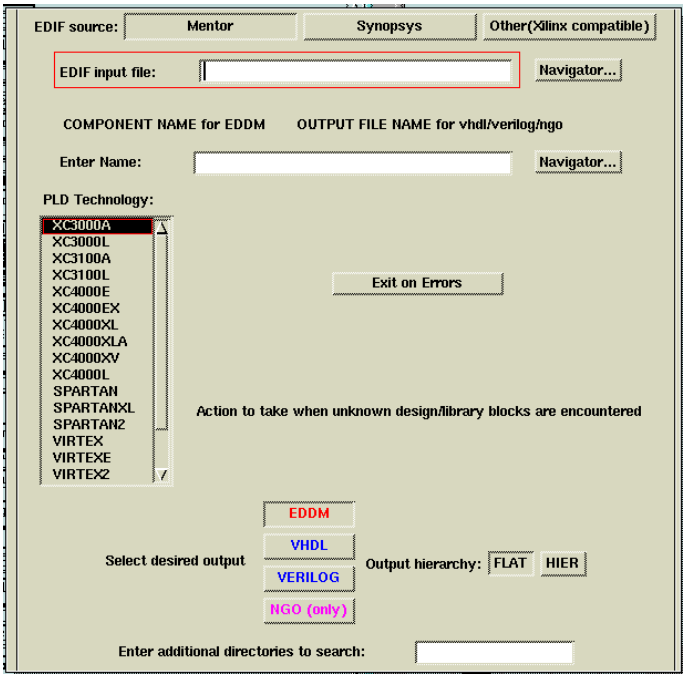


Figure 3-8 PLD EDIF to Mentor Convert Dialog Box

Pld_edif2sim uses all supporting EDIF files from the directory in which the top-level EDIF input file was submitted.

4. In the EDIF source field, select **Mentor**, **Synopsys**, or **Other** to specify the source from which the EDIF was generated. Specify Other if the EDIF comes from a vendor other than Mentor or Synopsys. When selecting Other, you must ensure that the EDIF is compatible with Xilinx EDIF.
5. In the Top-level EDIF Input File field, type in the name of your top-level EDIF file, or click **Navigator** to find it.
6. In the Enter Name field, enter the name of the symbol that you created in step 1, or click **Navigator** to find it.

   Note If the symbol has not yet been created, a Mentor component is created with an EDDM-single-object model. At this point, you can use Design Architect to create a symbol for it. Click **Open Symbol** and specify the path name of this component. A symbol is automatically created. Check the symbol, add the file=*edif_file_pathname* property, and save it.
7. In the PLD Technology field, select the appropriate architecture.
8. Leave the Exit on Errors button enabled if you want the program to exit when it encounters an unresolved block. Otherwise, click the Exit on Errors button and it changes to Continue (Ignore Errors).
9. In the Select Desired Output field, select **EDDM**.
10. In the "Enter additional directories to search" field, enter all the directory pathnames that the program should search to find EDIF, XNF, and NGO files that define blocks in your design that have the File property on them.
11. Click **OK**.

    This procedure produces a single-object simulation model for the specified symbol component.

    If you are converting an EDIF with pld_edif2sim, the simulation viewpoint is created for you automatically.

Simulating the Design

The rest of the simulation procedure is the same as that described in the "Simulating the Design" section of the "Simulating Pure Schematic Designs" section earlier in this chapter.

Implementing Schematic Designs

Once you complete functional simulation for schematic designs, you are ready to implement your design. You perform implementation with the Xilinx Design Manager, pld_dsgnmgr, which you invoke from the Mentor Design Manager or from a UNIX shell. Pld_dsgnmgr first translates the design into a flattened or hierarchical netlist, then optimizes, places, and routes the design. It creates delay data for timing simulation and physical (bitstream) design data for downloading.

Design entry of pure schematic designs, or schematic designs with LogiBLOX elements, CORE Generator modules, EDIF submodules, or XNF submodules produces an EDDM file that you must convert to EDIF with the pld_men2edif utility before implementing the design with pld_dsgnmgr. The following figure shows the design flow involved in implementing a design.

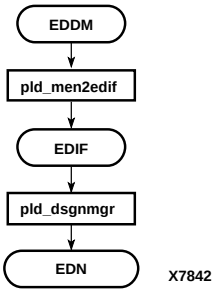


Figure 3-9 Design Implementation

Converting the EDDM Design to EDIF

To convert your design to EDIF, follow these steps.

1. In the Mentor Design Manager, double-click the left mouse button on the pld_men2edif icon.

   The dialog box shown in the following figure appears.

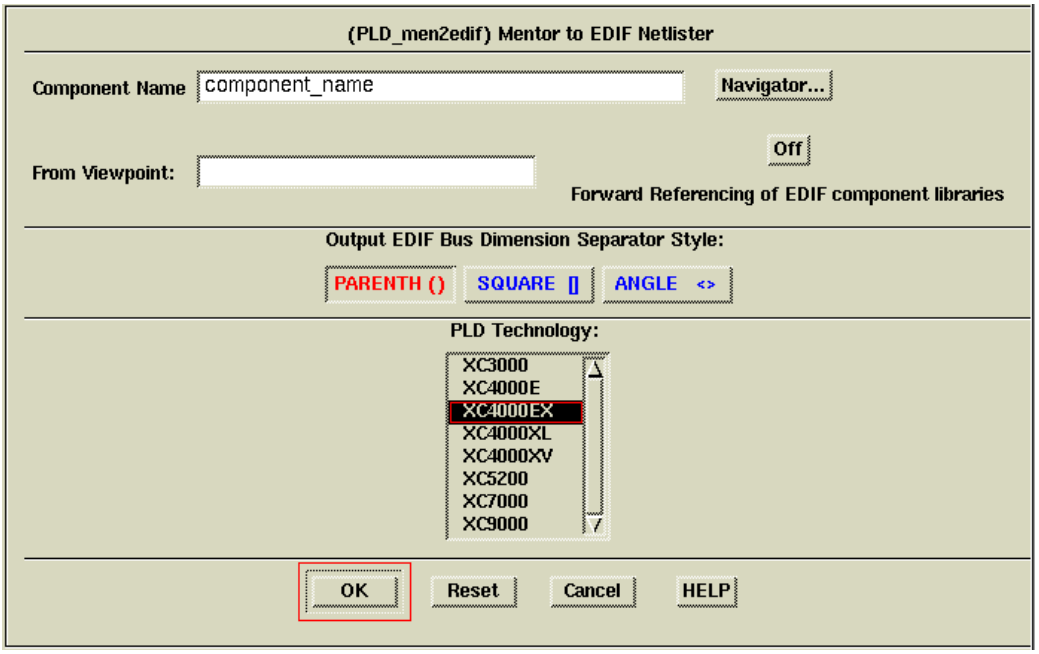


Figure 3-10 Mentor to EDIF Netlist Dialog Box

2. In the Component Name field, enter the component name or click **Navigator** to browse a list of design names.
3. In the From Viewpoint field, you can enter the viewpoint name if you do not want to use the default viewpoint. Alternatively, in step 2 you can select a viewpoint under the component.
4. Select the appropriate architecture for your design in the PLD Technology field.
5. Select the appropriate Bus Dimension Separator Style.

   This is important if you are merging components from other design-entry tools into a single design. Choosing a bus-index delimiter lets you insure that the bus-index delimiters that pld_men2edif writes out are consistent with those of any other design-entry tools with which you are interfacing. Mentor EDIF uses parentheses. Synopsys EDIF uses angle brackets.
6. Click **OK**.

Pld_men2edif now produces an EDIF file that you can submit to the Xilinx Design Manager, pld_dsgnmgr. The output name is *component_name*.edif.

Implementing the Design

The Xilinx Design Manager is a graphical design flow and project manager. The Xilinx Design Manager takes your design, represented by the EDIF file from pld_men2edif, and implements it in an FPGA or CPLD. You can also use the Xilinx Design Manager to generate timing information that you can import into QuickSim or ModelSim.

The Xilinx Design Manager, pld_dsgnmgr, can accept an EDIF file, or if your design is a pure XNF design, it can accept an XNF file.

For a more in-depth discussion of the flow, including advanced implementation options, see the *Development System Reference Guide*.

To implement your design, follow these steps.

1. Within the Mentor Design Manager, select the EDIF icon for your design in the Navigator, then select **Right Mouse Button** → **Open** → **pld_dsgnmgr**. The tool automatically creates a Xilinx project called *your_design_name*. Xilinx project information is kept in a file called xproj/*your_design_name*.prj by default. For implementation details, see the Xilinx *Design Manager/Flow Engine Guide*.

2. In the Xilinx Design Manager, select **Design** → **Options**.

   The Options dialog box appears as shown in the following figure. This dialog box contains many options. Most of these you can set as you see fit for you design. The following steps list some recommended settings for working in the Mentor environment.

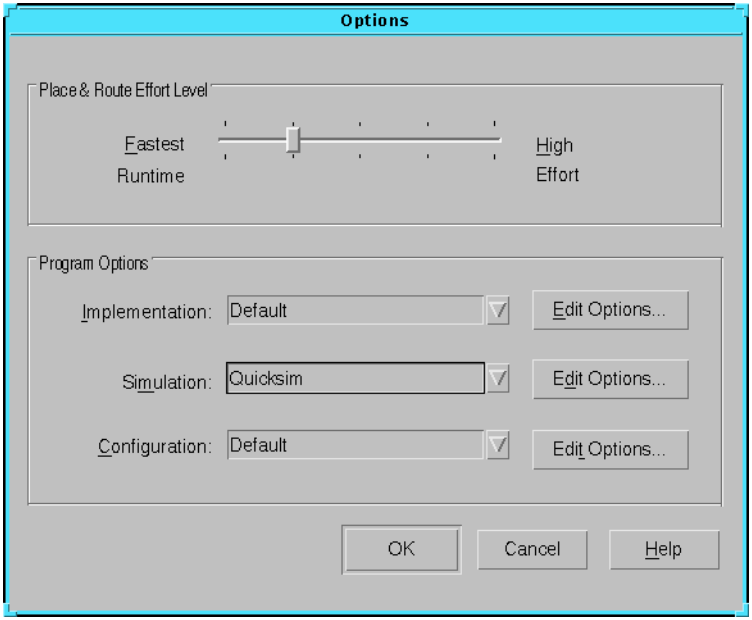


Figure 3-11 Options Dialog Box

1. Select Quicksim in the Options dialog box Simulation popup menu.
2. Click the Simulation Edit Options button to open the Simulations Options dialog box.
3. In the EDIF pane, make sure the CAE Vendor is set to Mentor and that the Retain Hierarchy in Netlist option is enabled.

Timing Simulation for Schematic Designs

Timing simulation verifies design functionality by using delay information from the EDIF, VHDL, or Verilog file created during design implementation. It also describes how to perform a timing analysis with Mentor's QuickPath tool.

During design implementation, the Xilinx Design Manager can produce an EDIF (EDN) file. For EDIF files, the process of timing simulation consists of converting the EDIF netlist to a Mentor EDDM model, creating a component and a viewpoint, and simulating the design with pld_quicksim. The timing simulation process for EDIF files is shown in Figure 3-12.

The *Mentor Graphics Schematic Design Tutorial* on the Xilinx Web site at `http://support.xilinx.com/support/techsup/tutorials/index.htm` illustrates the steps involved in timing simulation.

This section describes how to use QuickSim to perform timing simulation on designs described in EDIF.

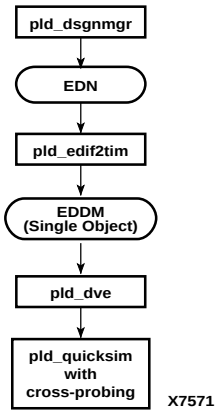


Figure 3-12 Timing Simulation for Schematics

Creating the EDDM Model and the Viewpoint

The first step in performing a timing simulation on your design is to use the pld_edif2tim utility to convert the EDIF netlist created by the Xilinx Design Manager to a Mentor EDDM model. At the same time, pld_edif2tim automatically creates a viewpoint which is subsequently processed by pld_dve –s to prepare it for timing simulation.

1. Double-click the left mouse button on the pld_edif2tim icon in the Design Manager Tools window.

   The dialog box shown in the following figure appears.

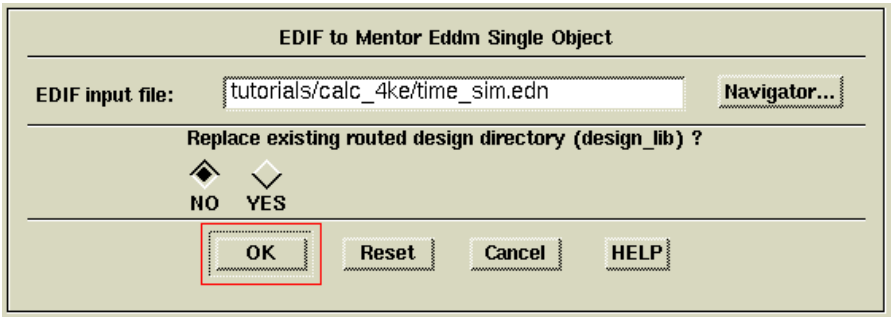


Figure 3-13 EDIF to Mentor Eddm Single Object Dialog Box

2. Enter the name of the EDN file in the EDIF Input File field, or click **Navigator** to browse the available files.

   The component created from the EDN file is put into a design library called *my_design_lib*. If you have already implemented your design at least once, this directory already exists. If you do not wish to copy over it, move it to another directory before proceeding.

3. Click **OK**.
4. Invoke DVE, by double-clicking the left mouse button on the pld_dve icon in the Design Manager Tools window.
5. Enter the top-level component name created by pld_edif2tim in the *my_design_lib* directory.
6. Use the **Navigate** button to navigate all the way down to the "default" viewpoint and select the viewpoint.
7. Select the **Simulation** Button.
8. Select the appropriate technology from the PLD Technology box.
9. Click **OK**.

Simulating the Design

You can now submit the design to pld_quicksim for timing simulation.

1. To invoke pld_quicksim, double-click the left mouse button on the pld_quicksim icon in the Design Manger Tools window.

2. In the Design field, enter the name of the top-level design created by pld_edif2tim.
3. In the Select Desired Mode field, select **Cross-Probing**.

   Normally, you select cross-probing for back-end EDDM designs but not for front-end designs. You can only cross-probe back-end designs that contain either pure schematic or schematics that contain EDDM hierarchical models. You cannot cross-probe designs written in HDL or that contain HDL models. See the "Cross-Probing" section for more details about cross-probing.

   Caution: In order for cross-probing to work, other sessions of Design Viewpoint Editor and QuickSim must be closed. Otherwise, the interprocess communication gets confused. This includes another user's session invoked by rlogin from another workstation.
4. Set the timing modes as desired.
5. Click **OK**.

   Pld_quicksim now simulates the design. The QuickSim graphical user interface appears. If you selected cross-probing, DVE is invoked as well.
6. In DVE, open the viewpoint of the front-end schematic design, that is, the viewpoint submitted to pld_men2edif.
7. Open the sheet of the design, and select signals that you wish to trace.

   These signals are automatically added to the QuickSim trace window. If you have a file that sets up your trace window and stimulus, use that instead. Any signals selected in the trace window select the same on the schematic on which they reside in the DVE window. If such sheets have not been opened in DVE yet, you must open them to see the selections.

Cross-Probing

Cross-probing is a way of cross-referencing between the original schematic and the timing simulation model after placement and routing. Once a Mentor design is translated, expanded, mapped, placed, and routed, you can extract the back-annotation information and create a hierarchical EDIF netlist. After you convert this EDIF to an EDDM model using pld_edif2tim, you submit it to pld_dve to create a viewpoint and then to pld_quicksim for timing simulation. The resulting data base preserves the design hierarchy, and although it is created in terms of the SimPrim library, most of the original net names are still available. You enable cross-probing by invoking QuickSim with the -cp option. This option invokes pld_dve as well as pld_quicksim. You then open the original design viewpoint in pld_dve and view the desired design sheet. If you display the original schematic in pld_dve, you can select nets on the original schematic and view them in the QuickSim trace window.

You may optionally create a schematic model using Mentor's schematic generator (sg) from the Eddm model created by pld_edif2tim. This schematic is only for viewing the backend schematic and is not required for the Xilinx flow to work. With cross-probing, you can use your original schematic for this purpose.

You should usually be able to reapply your original test vectors to the new Eddm_single_object design model for timing and/or functional simulation in QuickSim.

When you create the trace/list window in QuickSim, selecting signals from the original selected test vectors should cause the corresponding net on the original schematic sheet in pld_dve to be selected. If it is unselected in the trace/list window, it is also unselected on the original schematic.

If a net is selected in the pld_dve schematic sheet window, the net is automatically added to QuickSim trace window. If the net due to optimization or other complexities has been eliminated in the post-layout design, QuickSim issues an Error message of the type.

```
Error: $$add_traces returned error status at line
440 of file /tools/...

Error: Unable to resolve string '/ALU/I$10/C2' to a
signal or expression
```

No trace is displayed for this net.

When a net is selected on the original schematic sheet in pld_dve and if the corresponding signal is already added to the trace/list window, the net will not be added again; instead, it is highlighted in the trace/list window.

Adding list windows in quicksim is your choice. List windows are not automatically created. If you do create a list window, it is your choice which signals to add to the list window. Opening a list window does not automatically show or add the signals from the trace window. However once you have added signals to the list window, selecting such signals will interact with the original schematic exactly the same way as the ones in trace window.

Caution: In order for cross-probing to work, other sessions of Design Viewpoint Editor and QuickSim must be closed. Otherwise, the interprocess communication gets confused. This includes another user's session invoked by rlogin from another workstation to your workstation.

Note If you flatten your design during netlist generation, you lose hierarchical aliases for signals that span multiple hierarchy levels; only the name of the signal at its highest level is preserved.

While 100% back-annotation is possible, certain limitations of simulators, optimization process, and modelling of complex functions can make 100% back annotation impossible.

For more details on cross-probing, see the *Mentor Graphics Schematic Design Tutorial* on the Xilinx Web site at `http://support.xilinx.com/support/techsup/tutorials/index.htm`.

Performing a Timing Analysis

To perform static timing analysis, you may use Xilinx "trace" or Mentor's QuickPath or Sst Velocity.

Chapter 4

HDL Designs

This chapter describes how to use the Mentor Graphics Interface to design with pure HDL designs. It contains the following sections.

Design Flow

The following figure shows the design flow for VHDL and Verilog design entry, functional simulation, synthesis, and timing simulation for all supported technologies.

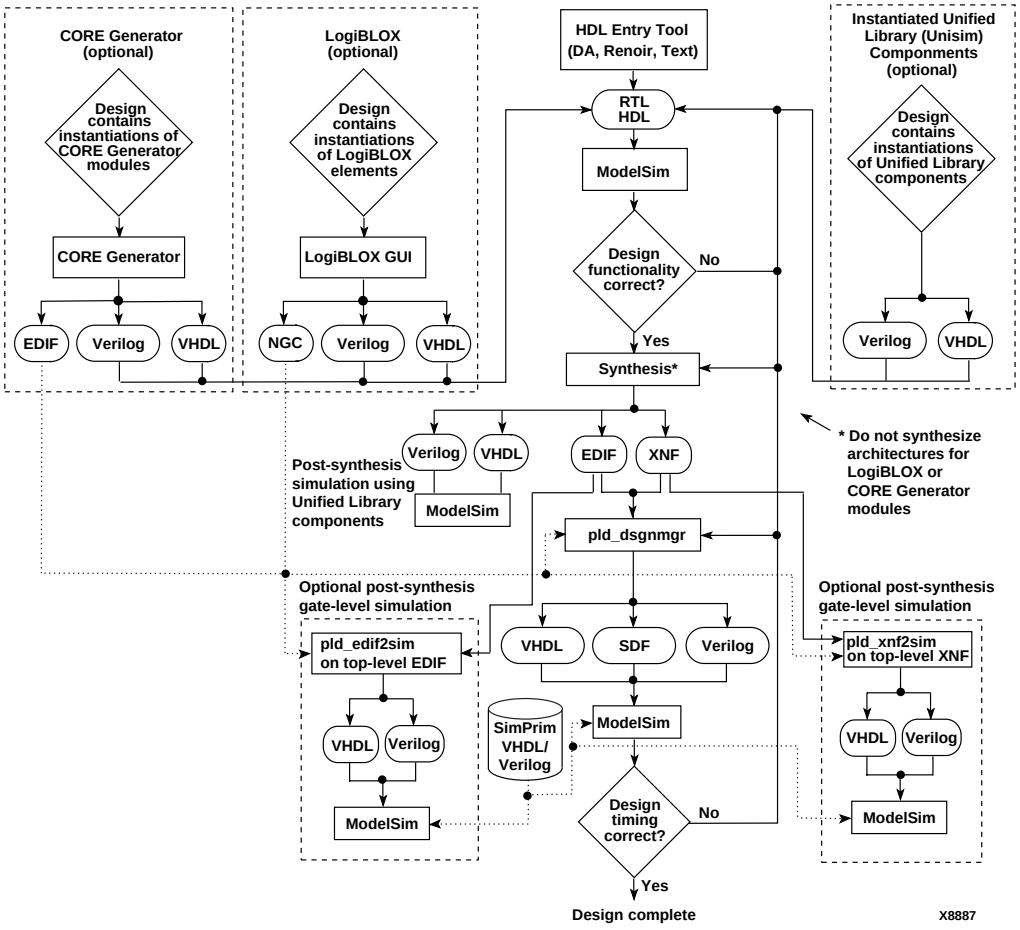


Figure 4-1 HDL (Verilog/VHDL) Design Flow

HDL Design Entry

This section describes the basic process of entering HDL designs. In addition to this chapter, the HDL design entry techniques in this section apply to the “Mixed Designs with VHDL on Top” chapter and “Mixed Designs with Schematic on Top” chapter.

Overview of HDL Design Entry

Use a text editor, pld_da, Renoir, or other HDL entry tool that is compatible with your synthesizer to create synthesizable VHDL or Verilog. Pld_da can be better than a plain text editor for editing your source. With pld_da you can submit the source to be compiled as you edit it (see the Mentor *Design Architect User's Guide* for details). When performing HDL design entry, observe the following requirements.

- The synthesizers must create EDIF or XNF that is compatible with Xilinx implementation tools.
- Xilinx-specific properties and timing constraints cannot be added in a VHDL or Verilog description, but synthesizers do have the capability of adding them to the output EDIF or XNF via constraint setting options. These constraint settings must be consistent with the current Xilinx implementation tools requirements. Otherwise, implementation can be controlled within the implementation tools themselves or with a UCF (user constraint file) file.

As an optional part of the Xilinx HDL design entry flow, you can instantiate LogiBLOX modules in your VHDL or Verilog designs and simulate the HDL output from LogiBLOX in your HDL simulators. When using LogiBLOX modules in HDL design entry, observe the following requirements.

- Create the NGC from LogiBLOX for later use in the implementation tools. LogiBLOX NGC files must be placed in your top level directory or you must modify the macro search path in the Xilinx Design Manager to include the location of NGC files. You must have the LogiBLOX NGC files in the same directory as your top-level EDIF or XNF.
- Your synthesizer must not read in or synthesize the HDL description of the LogiBLOX modules. These descriptions are for simulation only. The modules must be treated as black boxes by the synthesizer.

As an additional option in the Xilinx HDL design entry flow, you can instantiate CORE Generator modules in your VHDL or Verilog designs and simulate the HDL output from the CORE Generator system in your HDL simulator. When using CORE Generator modules in HDL design entry, observe the following requirements.

- Create the EDF from CoreGen for later use in the implementation tools. The CoreGen EDF file must be placed in your top level directory or you must modify the macro search path in the Xilinx Design Manager to include the location of the EDF files. You must have the CORE Generator EDF files in the same directory as your top-level EDF or XNF.
- Your synthesizer must not read in or synthesize the HDL description of the CoreGen modules. These descriptions are for simulation only. These are templates for you to cut and paste CoreGen declarations, instantiations, and configurations into the testbench. The modules must be treated as black boxes by the synthesizer.

Note In order to simulate CORE Generator modules, you need to first extract HDL simulation models using get_models. Please see the *CORE Generator System User Guide* for details.

HDL Design Entry Stages

HDL design entry has two stages as shown in Figure 4-2. The first stage is the Register Transfer Level (RTL). At this level, the design behavior is described in a high-level, non-technology-specific manner. Instantiation of specific components is the exception. An example would be RAMs or LogiBLOX modules. This design entry step is generally followed by a functional simulation.

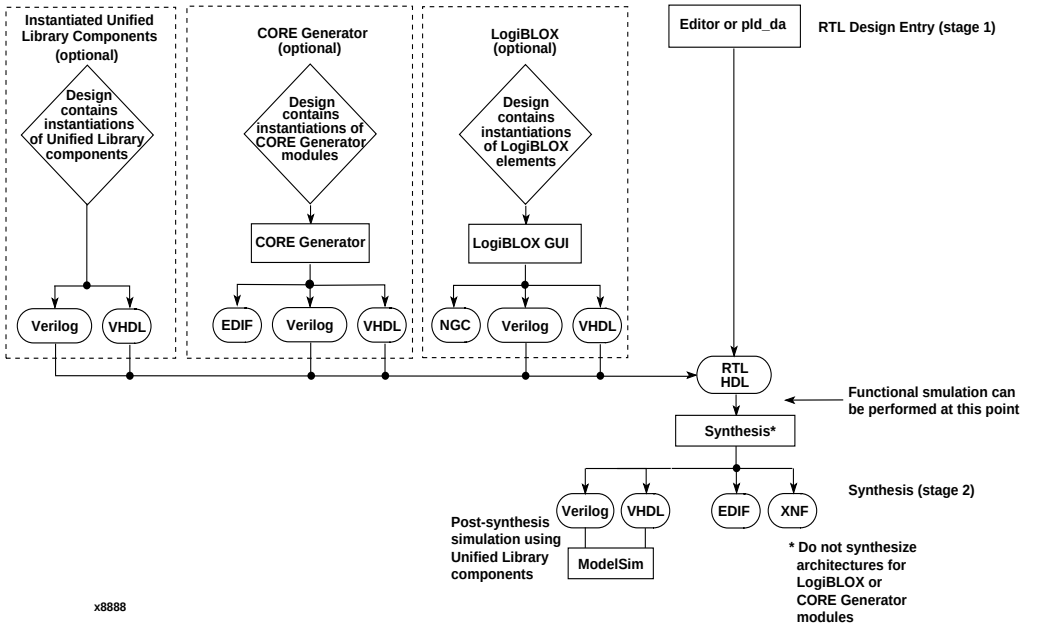


Figure 4-2 HDL (Verilog/VHDL) Design Entry and Synthesis

During design entry, you may check out the syntax correctness of your code by compiling it for your synthesizer and / or ModelSim without doing either the synthesis or simulation.

You may find the syntax checkers are different. The synthesizer may check for certain constructs it cannot synthesize like textio in VHDL, but these constructs may be perfectly good for simulating functionality as you develop the circuit. Many synthesizers have pragma or meta comments that allow you to keep this code in your HDL description but tell the synthesizer to ignore it.

Also there may be significant differences in how thoroughly the compilers check the code against the VHDL and Verilog IEEE standards or even how they interpret certain sections.

It is good practice to do occasional compiles for both the synthesis tool and ModelSim as you develop large sections of your HDL code.

Once the RTL simulation is correct, the second stage of design entry is to submit the RTL code to a synthesizer where the general functionality is synthesized and mapped to gates in a specific technology. At this point you have the option of performing a second functional simulation of the post-synthesis gate level description. However, this step is not necessary since no additional timing information is available before place and route.

Once you are satisfied with the behavior of the circuit, you can send the gate level output of the synthesizer to the implementation tool as either an XNF or EDIF file.

Stage 1: RTL Behavioral Code Development

The first stage of HDL design entry is developing an RTL behavioral description. Code created at the RTL entry is generally non-technology specific. Two exceptions worth noting are as follows.

- Coding style favoring one technology strengths over another
- Instantiating technology specific components

Your coding style should take into account your targeted technology's architecture specifics to achieve the best performance or smallest size for your design.

For example, a style that infers latches unintentionally or even on purpose would be just fine for a XC5200 or XC4000EX part but would be a trouble spot with an XC3000. It is a problem for the XC3000 because it would be implemented as cross-coupled logic creating a host of timing analysis issues. In a XC4000 it would take up valuable resources since each latch could be implemented as a SRAM cell, taking up a whole function generator. Synthesizers may vary in how they implement latches in technologies that do not have explicit latches. Some may use cross-coupled logic as in the XC4000E. Others may use a RAM cell.

Another RTL coding style problem might be describing the functionality in a manner that infers lots of MUXES. A better choice in some technologies would be to infer internal tri-states and use the high-performance tri-state lines.

Some synthesizers may not have a means of inferring or targeting high performance technology components like wide-edge decoders, I/O muxes, or latches. In that case you may need to instantiate these components to get your best chip performance.

In summary, observe the following RTL coding guidelines.

- Avoid using latches in technologies without specific architecture components to support them like the XC5200 and XC4000EX have.
- Infer tri-state buses instead of muxes for technologies with good internal tri-state structures.
- Instantiate high performance architecture features if your synthesizer cannot infer them.

Stage 2: Synthesis

The second stage of HDL design entry is synthesis of the RTL behavioral description down to technology specific gates. Specific synthesis design entry steps for Xilinx parts are highly dependent on which synthesizer you use. Generally you can break synthesis design entry into the following steps.

1. Tailor your RTL code for both the synthesizer and the Xilinx technology's capabilities. For example, if your synthesizer can insert the STARTUP component, you need not instantiate it.

   When you simulate in ModelSim with an instantiated startup block, you get a warning because the startup module does not have a simulation module. You may ignore this warning since the startup block is only used to direct the implementation tool and does not change the logical functionality of the circuit. The warning looks something like this.

   ```
   # ** Warning: Component startupblk is not bound.
   ```

2. Guide the synthesis process with timing and/or size requirements.
3. Guide the output process to select XNF or EDIF outputs to insert I/O buffers, global buffers, STARTUP blocks, and to output timing constraints either within the netlist itself or to a separate file readable by Xilinx implementation tools. Make sure the timing constraint style is the correct one for the current version of Xilinx implementation tools.

You should read your synthesizer manual for specific details, especially the sections about targeting Xilinx devices. Be aware that any one of these three steps can greatly affect the quality of synthesis and/or implementation results.

LogiBLOX Design Entry

Some synthesizers are not capable of using Xilinx carry chains properly and tend to infer inefficient structures (for example, for adders and counters). Other synthesizers are able to infer incrementors or decrementors, but do not use efficient logic for the control logic of the loadable counter.

To guarantee optimal synthesis for certain modules in a Xilinx technology, you may use the LogiBLOX module generator and instantiate the resulting module in your HDL code.

You may invoke the stand-alone Graphic User Interface of LogiBLOX in the tool window of pld_dmgr by clicking on the pld_logiblox icon. This is not the same Graphic User Interface you get in the Design Architect Schematic Palette. The stand-alone version is for VHDL or Verilog models only. Another way to invoke the stand-alone GUI is from the Design Architect pop-up menu in the Session Window.

LogiBLOX requires two outputs for proper implementation in a HDL design.

The first output is the HDL behavior description for simulating either VHDL or Verilog. These HDL descriptions only support HDL functional simulations. You should not send them to the synthesizer for synthesis. The entities can be used for component instantiation purposes, but the architectures should be treated as black boxes within the synthesizer. The following is an example of a LogiBLOX component declaration and instantiation in VHDL.

```
------------------------------------------------
-- Component Declaration
------------------------------------------------
component RAM16X1
  PORT(
    A: IN std_logic_vector(3 DOWNTO 0);
    DI: IN std_logic_vector(15 DOWNTO 0);
    WR_EN: IN std_logic;
    DO: OUT std_logic_vector(15 DOWNTO 0));
end component;

------------------------------------------------
-- Component Instantiation
------------------------------------------------
```

```
instance_name : RAM16X1 port map
    (A => ,
    DI => ,
    WR_EN => ,
    DO => );
```

The second output is the NGC file. The implementation tools use this file to pull the LogiBLOX module into the top-level design. Since these NGC files are technology specific, you should generate a new NGC file each time you select a new Xilinx architecture. The HDL behavioral descriptions do not change.

CORE Generator Module Design Entry

The Xilinx CORE Generator system provides designers with a catalog of ready-made functions ranging in complexity from simple arithmetic operators such as adders, accumulators, and multiplexers to system-level building blocks such as filters, transforms, and memories. These functions are optimized to deliver the highest levels of performance and efficiency for a particular Xilinx FPGA architecture and can be quickly integrated into your schematic or HDL designs.

You can invoke the Xilinx CORE Generator system in standalone mode by typing **coregen** at the UNIX prompt, by typing **coregen** in a session pop-up window in Design Architect, or by clicking on the pld_coregen icon in the Mentor Design Manager.

For Verilog and VHDL flows, the CORE Generator system creates both types of instantiation templates (.veo and .vho), and you simply use the one that is appropriate for your design flow. The instantiation templates contain example code for instantiating the CORE Generator module into a synthesis netlist. They also contain additional directives that must be added to a behavioral simulation testbench to perform an HDL behavioral simulation of your top level design. For example, in the .vho instantiation template file, there is a component declaration for CORE Generator module and a module instantiation block which you must use to instantiate the module in your HDL design for synthesis and behavioral simulation.

There is also a Xilinx CoreLib simulation library declaration, an include statement pointing to the VHDL behavioral model for the module in the Xilinx CoreLib library, and a VHDL CONFIGURATION entry which defines VHDL generics used in the module's VHDL behavioral simulation model.

The EDIF implementation netlist (.edn) is always produced for all design flows and contains information used by the Xilinx implementation tools to implement the module. Because the information in the EDN file is technology specific, a given CORE Generator module must always be regenerated whenever you retarget your design to a different Xilinx architecture.

Unified Library Instantiated Components

If you prefer, you may instantiate Unified Library components into the RTL design. The components you use should be primitives supported in the Xilinx family being targeted and also present in the synthesis tool's target library. For more information on Unified Library components, see the *Development System Reference Guide*.

Functional Simulation

Pure HDL designs consist of a RTL VHDL or Verilog model. You can optionally convert the synthesis output netlist to a gate-level HDL model and functionally simulate it. The flow diagram for performing functional simulation on pure HDL designs is shown in the following figure.

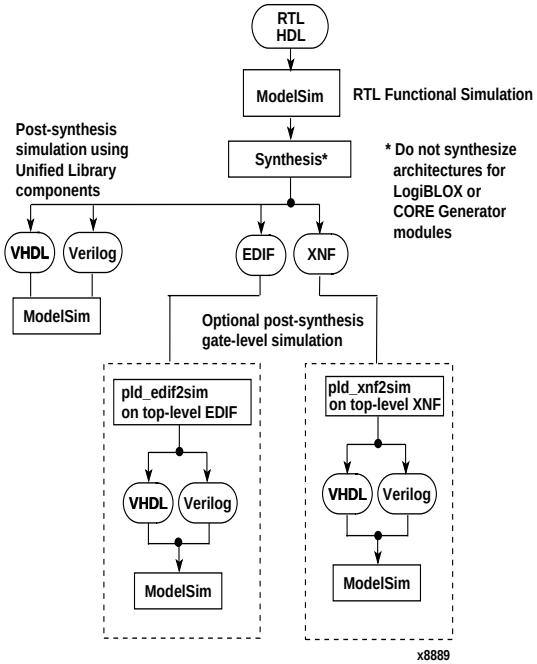


Figure 4-3 Performing Functional Simulation on a Pure HDL Design

Pre-Synthesis Functional Simulation

To perform a pre-synthesis functional simulation on a pure HDL design follow these steps.

Note This procedure assumes that you are using ModelSim. QuickHDL provides the same functionality as ModelSim. If you are using QuickHDL instead of ModelSim, see the "ModelSim" section of the "Introduction" chapter for details on how to modify this procedure.

1. Create a working library with vlib.

   `vlib` *mywork*

2. Map the library with vmap.

   `vmap work` *mywork*

3. If you are using LogiBLOX modules, use vmap to map to the compiled LogiBLOX modules location.

   vmap *logiblox compiled_logiblox_area*

   Your system administrator can tell you the location of the compiled version(s) of the LogiBLOX library. Instructions for compiling are in the Mentor Graphics Installation section of the *Alliance Installation Guide*. You may have to recompile the library for each version of ModelSim you use. The default directory for the compiled LogiBLOX library is as follows.

   $XILINX/mentor/data/vhdl/logiblox

4. If you are using CORE Generator modules, use vmap to map to the compiled CoreGen modules location.

   Your system administrator can tell you the location of the compiled version(s) of the Xilinx CORE Generator library. Instructions for compiling are in the CORE Generator Installation section of the *Alliance Installation Guide*. You may have to recompile the library for each version of ModelSim you use. You can copy the files to a directory of your choice using the get_models command provided by the Xilinx install. This utility is used to extract and collect the Verilog or VHDL behavioral models which have been installed in a user's Core Generator tree.

   get_models [**-verilog** | **-vhdl**] *directory_name*

   For example, the following collects the VHDL models and puts them in the directory /usr/tmp/.

   get_models -vhdl /usr/tmp

   Compile the libraries as follows.

   vlib XilinxCoreLib

   vmap XilinxCoreLib *path/to/XilinxCoreLib*

   vcom -work XilinxCoreLib /usr/tmp/files.vhd

   In a similar manner, you can use the vlog command to compile the verilog libraries.

5. If you are using Unified Library components, use vmap to map to the compiled Unified Library location by executing the appropriate line below for the device family that you are using.

For vhdl:

```
vmap unisim $XILINX/mentor/data/vhdl/unisim
```

```
vmap unisim_5k $XILINX/mentor/data/vhdl/
unisim_5k
```

Map to unisim for the XC3000 and XC4000 series, or unisim_5k for the XC5200 series.

For verilog:

```
vmap unisim_ver $XILINX/mentor/data/verilog/
unisims
```

Map to uni3000 for the XC3000 series or uni5200 for the XC5200 series. All XC4000, Spartan, and Virtex families are mapped to unisims.

Note The preceding locations for the compiled libraries are the default locations for a default software installation. However, your system administrator can install them in other locations. Your system administrator can tell you the location of the compiled version(s) of the Unified Library. Instructions for compiling are in the Mentor Graphics Installation section of the *Alliance Installation Guide*. You may have to recompile the library for each version of ModelSim you use.

6. Compile the HDL source files with vcom (VHDL) or vlog (Verilog).

   vcom [*options*] *design_file(s)*

   vlog [*options*] *design_file(s)*

   See the Mentor documentation for a description of the available options.

7. Compile your testbench with vcom (VHDL) or vlog (Verilog).

   vcom [*options*] *testbench_file(s)*

   vlog [*options*] *testfixture_file(s)*

8. Select the appropriate architecture configuration or module for your testbench and select ModelSim in the pld_dmgr tools window. You can alternatively invoke the ModelSim simulator using vsim on the command line.

   See the Mentor documentation for ModelSim instructions.

9. After the RTL level simulation is correct, you may proceed to synthesis and to implementation or optional post-synthesis functional simulation.

Synthesis

You may use the tool of your choice for synthesis. For detailed information on performing synthesis, refer to the following Xilinx documents.

- *Synthesis and Simulation Design Guide*
- *Exemplar Tutorial* on the Xilinx Web site at `http://support.xilinx.com/support/techsup/tutorials/index.htm`

Optional Post Synthesis Functional Simulation

You can optionally perform a post-synthesis functional simulation on a pure HDL design, by following these steps.

1. Run pld_edif2sim on your top-level EDIF or pld_xnf2sim on your top level XNF file from synthesis.
2. Specify either VHDL or Verilog output in the pld_edif2sim or pld_xnf2sim dialog box.
3. Choose the Flat or Hierarchical option and click **OK** to create the structural HDL netlist.

Design Implementation

Once you complete functional simulation for HDL designs, you are ready to implement your design in an FPGA or CPLD. You perform implementation with the Xilinx Design Manager, a graphical design flow and project manager. In the Mentor interface, the Xilinx Design Manager is called pld_dsgnmgr. You invoke pld_dsgnmgr from the Mentor Design Manager or from a UNIX shell.

Figure 4-4 shows the design flow for implementing a design. The Xilinx Design Manager accepts your design, represented by the XNF or EDIF file from the synthesis tool. Design entry of pure HDL designs, or HDL designs with LogiBLOX elements or CORE Generator modules produces an EDIF or XNF file that you can submit to pld_dsgnmgr. Pld_dsgnmgr first translates the design into a flattened

or hierarchical netlist, then optimizes, places, and routes the design. You can also use the Xilinx Design Manager to generate SDF timing information that you can import into ModelSim along with your VHDL or Verilog netlist. For a more in-depth discussion of the flow, including advanced implementation options, see the *Development System Reference Guide*.

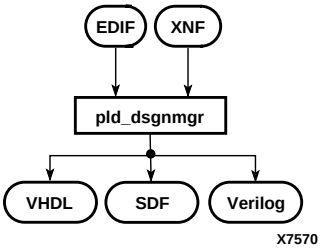


Figure 4-4 HDL Design Implementation

To implement your design within the Mentor Design Manager, select the EDIF icon for your design in the Navigator, then select **`Right Mouse Button`** → **`Open`** → **`pld_dsgnmgr`**. The Xilinx Design Manager displays. The tool automatically creates a Xilinx project called *your_design_name*. Xilinx project information is kept in a file called xproj/*your_design_name*.prj by default.

For implementation details, see the Xilinx *Design Manager/Flow Engine Guide*.

Note Be sure to choose the desired simulation option in the Options dialog box of the Xilinx Design Manager.

Timing Simulation

For HDL designs, the Xilinx Design Manager produces a V (Verilog) file or a VHDL file expressed in SimPrims and a corresponding SDF file that contains the timing data. The SDF file for VHD and V files are not interchangeable since the models they annotate follow different modeling standards.

Passing Timing Generics to Special Cells—ROC, OSC, OSC4, and OSC5

If your designs do not have an external global set or reset port or a user defined internal net driving the global set / reset net, then a ROC (Reset on Configuration) cell is automatically added to your VHDL netlist. This cell enables you to toggle the global set / reset net at the beginning of simulation by defining the pulse width of the signal pulse starting at time 0. By default, the pulsewidth is 0 which enables simulation to proceed but does not reset the circuit. To properly simulate the reset behavior of the chip, the pulse width generic should be set to a value within the range found in the Xilinx Databook for the particular device.

You can modify the following configuration for the technology's specific pulse width and user's testbench and compile it before you compile the testbench.

```
CONFIGURATION cfg_my_timing_testbench OF my_testbench IS
  FOR my_testbench_architecture
    FOR ALL:my_design USE ENTITY work.my_design(structure);
      FOR structure
        FOR ALL:roc USE ENTITY work.roc(roc)v)
          GENERIC MAP ( width => 100 ms);
        END FOR;
      END FOR;
    END FOR;
  END FOR;
END cfg_my_timing_testbench;
```

Verilog designs do not require ports to drive the global / set reset net from a testbench. Therefore, Verilog designs do not contain the ROC cell. The same signal name found in the front end can be used to drive the signal in the back-annotated design. The signal must be driven, or all flip-flops will initialize as X.

VHDL designs that contain oscillator cells like OSC, OSC4, or OSC5, must have the clock period set with a configuration statement. By default, the period is 0, disabling the oscillator. You should carefully select the period from the range of viable periods found in the Xilinx Databook for the particular technology. A specific period is not guaranteed because the cell is subject to process variations. You should select the value that best meets your simulation requirements.

You can use the following configurations for either the OSC, OSC4, or OSC5 cells by just changing the name of the cell and modifying the pulse width to the correct value.

```
CONFIGURATION cfg_my_functional_testbench OF my_testbench IS
  FOR my_testbench_architecture
    FOR ALL:my_design USE ENTITY work.my_design(my_design_rtl);
      FOR my_design_rtl
        FOR ALL:my_submodule USE ENTITY
        work.my_submodule(my_submodule_rtl);
          FOR my_submodule_rtl
            FOR all: osc4 USE ENTITY work.osc4(structure)
              GENERIC MAP ( period_8m => 125 NS);
            END FOR;
          END FOR;
        END FOR;
      END FOR;
    END FOR;
  END FOR;
END cfg_my_testbench_functional;
```

You can drive Verilog designs by the signal name used to drive the front-end simulation since the hierarchical name is preserved.

Simulating the Design

Simulate with ModelSim using vsim. To include the timing information in the SDF file, invoke vsim with the –sdftyp option. Refer to the Mentor documentation for information on available options. To simulate a Verilog based design, invoke vsim with the –L simprim option to choose the Verilog SimPrim libraries models.

Chapter 5

Mixed Designs with VHDL on Top

This chapter describes how to use the Mentor Graphics Interface to design with mixed schematic and VHDL designs with VHDL on Top. It contains the following sections.

Design Flow

The design flow for a top-level VHDL design with a schematic sub-module embedded within is illustrated in the following figure.

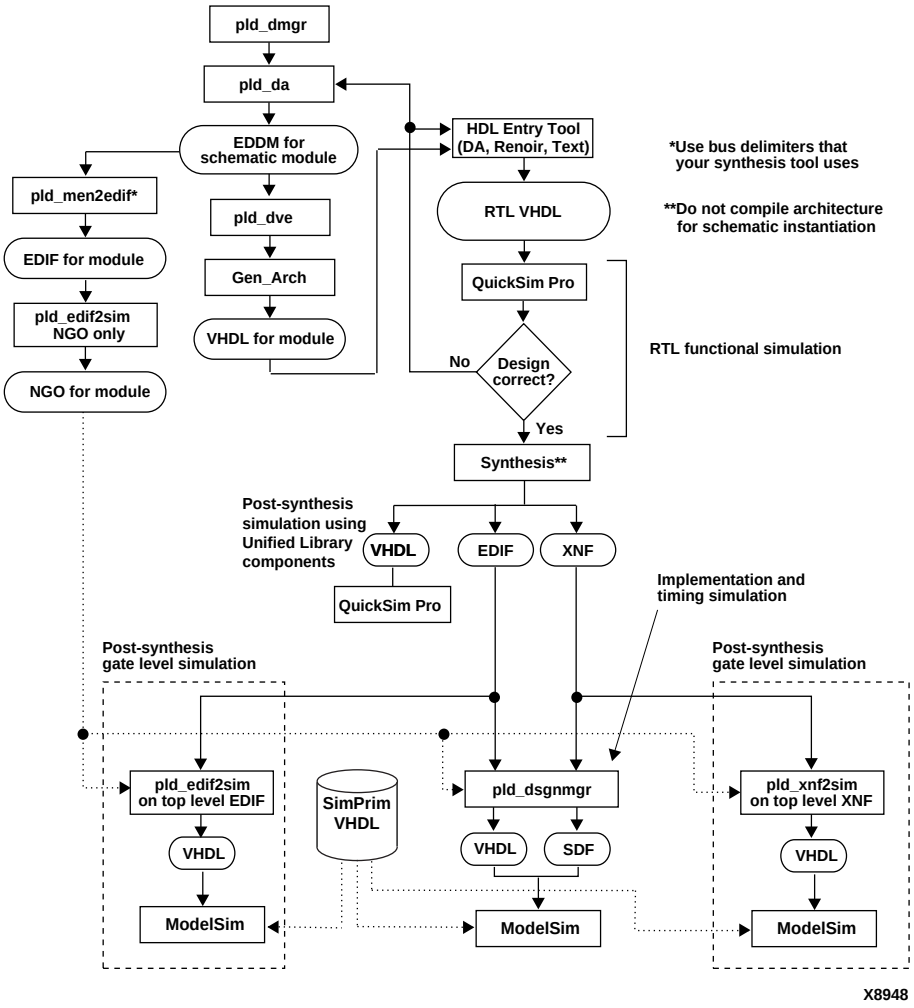


Figure 5-1 Mixed Schematic and VHDL Design with VHDL on Top

Design Entry

Enter your pure VHDL design as described in the “HDL Design Entry” section of the “HDL Designs” chapter.

If you wish to insert a schematic module into your VHDL code, Mentor QuickSim Pro allows you to co-simulate your VHDL portion in ModelSim with your schematic portion in QuickSim II.

Your synthesizer requires you to treat the schematic module as a black box. You must use pld_men2edif and pld_edif2sim to create a NGO file for the schematic component so the Xilinx implementation tools can merge it in the module during implementation.

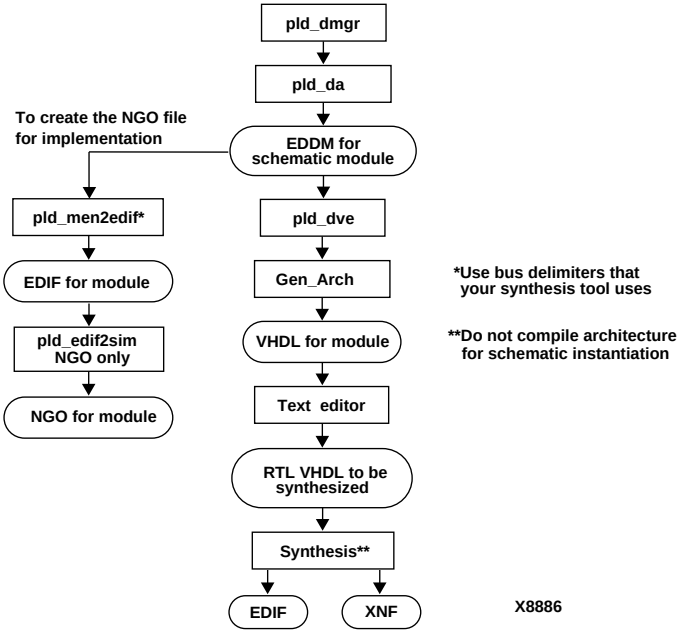


Figure 5-2 Design Entry for a Mixed Schematic and VHDL Design with VHDL on Top

To enter a mixed schematic and HDL design with VHDL on top, perform the following procedure. Figure 5-2 shows the flow diagram for this procedure.

1. Open pld_dmgr.
2. Open pld_da and generate EDDM for the schematic module.
3. Create the NGO file for implementation. To accomplish this, you use pld_men2edif to convert the EDDM for schematic module to EDIF and then use pld_edif2sim to create the NGO file. The procedure for doing this is as follows.

   a) Open pld_men2edif.

   b) Fill in the component name of the existing schematic based module. The module must have a symbol for its top-level netlist. There can be no chip-level I/Os.

   c) Select a viewpoint that properly sets the schematic parameters such that the EDIF is properly generated.

   d) Select the Bus Dimension Separator Style that matches your synthesizer. This is important; if your synthesizer uses one bus style and the EDIF/NGO from your schematic uses another style, the implementation tool does not merge the schematic module with the rest of the design, thus leaving it unexpanded.

   e) Choose the technology.

   f) Click **OK**.

   g) Create the NGO from EDIF2SIM and XNF2SIM for later use in the implementation tool. EDIF2SIM and XNF2SIM NGO files must be placed in your top level directory or you must modify the macro search path in the Xilinx Design Manager to include the location of the NGO files. EDIF2SIM or XNF2SIM do not have the macro search path functionality. You must have the EDIF2SIM and XNF2SIM NGO files in the same directory as your top-level EDIF or XNF.

   h) Open pld_edif2sim.

      A Pld_edif2sim dialog box opens.

   i) Specify the source of the EDIF file as either a Mentor, Synopsys, or Xilinx compatible EDIF. This step selects the appropriate implementation libraries.

   j) Enter the name for the EDIF file created above in step b that will be used for the NGO file.

k) Enter the name of the NGO file based on the component name used in the VHDL instantiation.

l) Select a Xilinx technology.

m) Select the NGO (only) output.

n) In the “Enter additional directories to search” field, enter all the directory pathnames that the program should search to find supporting EDIF, XNF, and NGO files.

o) Click **OK** to produce the NGO macro file of the schematic component.

4. Use pld_dve to set the simulation viewpoint.
5. Open GEN_ARCH to generate the VHDL for module.

   The dialog box opens as shown in the following figure.

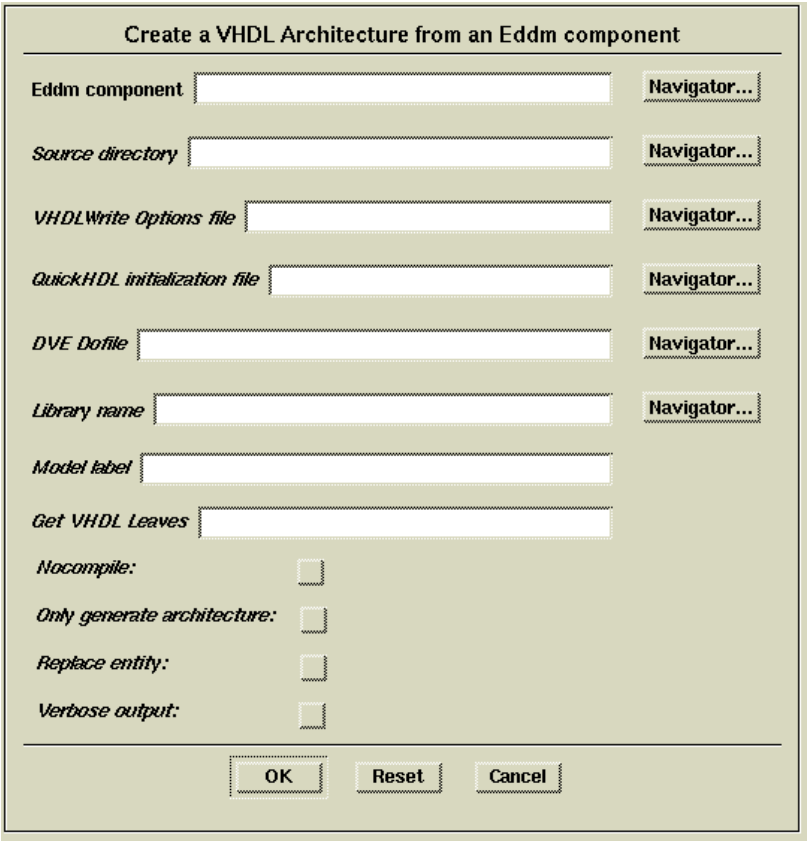


Figure 5-3 Create a VHDL Architecture from an EDDM Component Dialog Box

6. Enter the EDDM component name for the schematic.
7. Indicate the directory where the VHDL source files from GEN_ARCH are to be placed.
8. Specify the appropriate ModelSim initialization file. See the Mentor Graphics Documentation for details.
9. Enter the library name in which the compiled code will be placed. You can place it in the work library.
10. Leave the other boxes blank and click **OK** to produce the required output.

11. Use a Text Editor to create RTL VHDL to be synthesized for the rest of the design. Include the component declaration and instantiation for the schematic module.
12. Perform synthesis to generate EDIF or XNF for the whole design with a black box for the schematic module.

Functional Simulation

VHDL-on-top designs consist of a VHDL based design referencing EDDM components.

Simulating the Design

To simulate VHDL-at-top designs, invoke QuickSim Pro, which in turn invokes QuickSim to simulate the Unified Libraries elements and ModelSim to simulate the VHDL-based blocks as needed.

1. Double-click the left mouse button on the QuickSim Pro icon in the Design Manager Tools window.

   Alternatively, you can select the top-level component in the Navigator window and click the right mouse button to invoke QuickSim Pro.

   The QuickSim Pro dialog box appears, as shown in Figure 5-4.
2. In the **Invoke On** field, click **Configuration**.
3. In the **Name** field, type the path name of the configuration from Gen_Arch.
4. Click **Qspro**.
5. Click **OK** to proceed with simulation.

   For details on using QuickSim Pro, refer to the Mentor Graphics Documentation.

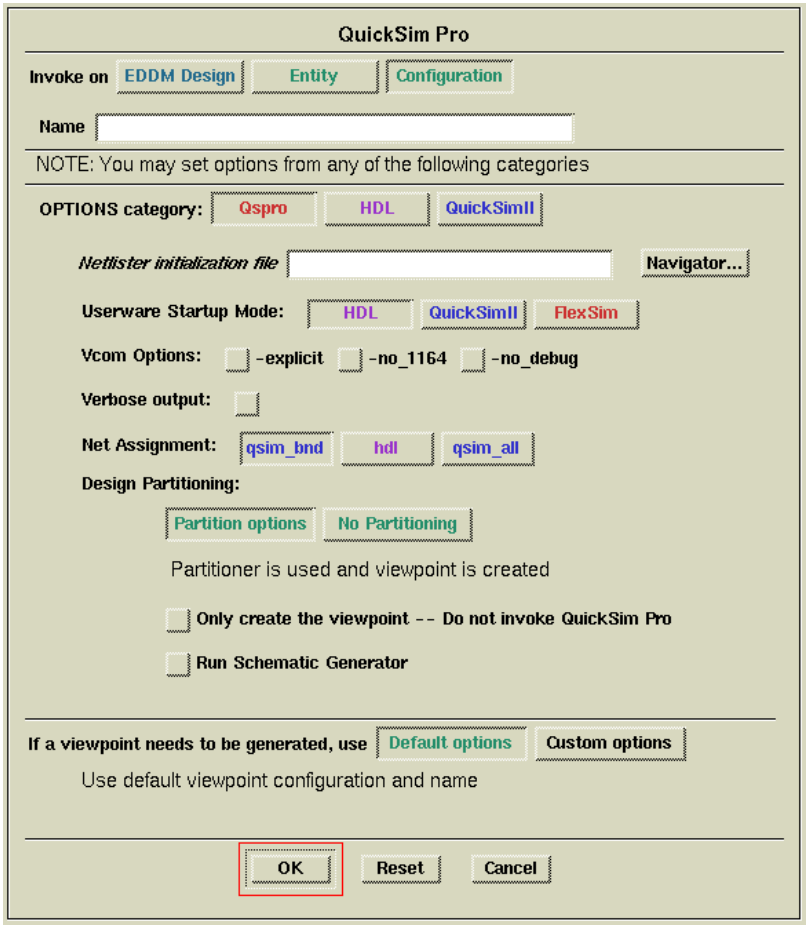


Figure 5-4 QuickSim Pro Dialog Box

Synthesis

You may use the tool of your choice for synthesis of the HDL component. For detailed information on performing synthesis, refer to the following Xilinx documents.

- *Synthesis and Simulation Design Guide*
- *Exemplar Tutorial* on the Xilinx Web site at `http://support.xilinx.com/support/techsup/tutorials/index.htm`

Optional Post-Synthesis Functional Simulation

You can optionally re-simulate the design after synthesis to an EDIF or XNF file to ensure that the design's functionality remains optimal. To do so, follow these steps.

1. Create the NGO from EDIF2SIM and XNF2SIM for later use in the implementation tool. EDIF2SIM and XNF2SIM NGO files must be placed in your top level directory or you must modify the macro search path in the Xilinx Design Manager to include the location of NGO files. EDIF2SIM or XNF2SIM do not have the macro search path functionality. You must have the EDIF2SIM and XNF2SIM NGO files in the same directory as your top-level EDIF or XNF.
2. If the synthesis tool created an EDIF file, submit the file to pld_edif2sim, then submit it to ModelSim.
3. If the synthesis tool created an XNF file, submit the file to pld_xnf2sim, then submit it to ModelSim. You can alternatively invoke the ModelSim simulator using vsim on the command line.

Design Implementation

Once you complete the functional simulation and synthesis steps for a VHDL-on-top design, you are ready to implement your design in an FPGA or CPLD. You perform implementation with the Xilinx Design Manager, a graphical design flow and project manager. In the Mentor interface, the Xilinx Design Manager is called pld_dsgnmgr. You invoke pld_dsgnmgr from the Mentor Design Manager or from a UNIX shell.

Within the Mentor Design Manager, select the EDIF icon for your design in the Navigator, then select **`Right Mouse Button`** → **`Open`** → **`pld_dsgnmgr`**. The Xilinx Design Manager displays. The tool automatically creates a Xilinx project called *your_design_name*. Xilinx project information is kept in a file called xproj/*your_design_name.prj* by default.

Design entry of VHDL-on-top designs produces NGO files for schematic modules and XNF or EDIF files for the synthesized portion of the design. The following figure shows the design flow for implementing such a mixed design.

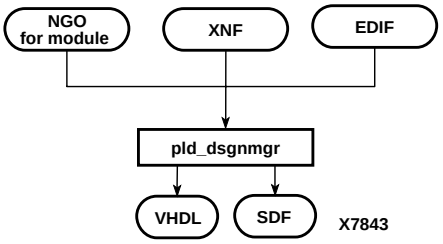


Figure 5-5 Design Implementation

The Xilinx Design Manager takes in your design, represented by the EDIF or XNF file from synthesis and the NGO file for the schematic module from pld_edif2sim. It first translates the design into a flattened or hierarchical netlist, then optimizes, places, and routes the design. You can also use the Xilinx Design Manager to generate SDF timing information that you can import into ModelSim. For a more in-depth discussion of the flow, including advanced implementation options, see the *Development System Reference Guide.*

By default, the Xilinx Design Manager looks for the NGO files for the schematic modules in the directory where it was invoked. You have the option of putting all of the NGO files in another directory. To direct the Xilinx Design Manager to look for the NGO files in another directory, follow these steps.

1. In the Xilinx Design Manager window, select **`Utilities`** → **`Template Manager`**.
2. Select the Family for implementation.
3. Select Implementation Templates.
4. Select the Template you wish to modify.

   If you have not created your own template, you may modify the default one.
5. Select Edit.
6. Select Interface.
7. Fill in the Macro Search Path box with the path to the NGO files.
8. Under simulation Data Options, select the VHDL Format.

Timing Simulation

You can now submit the VHDL and SDF files to ModelSim for timing simulation. You no longer need to use QuickSim Pro.

Compiling the SimPrim Libraries

Before performing timing simulation on an HDL-based design, the VHDL SimPrim libraries must be compiled.

Passing Timing Generics to Special Cells—ROC, OSC, OSC4, and OSC5

If your designs do not have an external global set or reset port or a user defined internal net driving the global set / reset net, then a ROC (Reset on Configuration) cell is automatically added to your VHDL netlist. This cell enables you to toggle the global set / reset net at the beginning of simulation by defining the pulse width of the signal pulse starting at time 0. By default, the pulsewidth is 0 which enables simulation to proceed but does not reset the circuit. To properly simulate the reset behavior of the chip, the pulse width generic should be set to a value within the range found in the Xilinx Databook for the particular device.

You can modify the following configuration for the technology's specific pulse width and user's testbench and compile it before you compile the testbench.

```
CONFIGURATION cfg_my_timing_testbench OF my_testbench IS
  FOR my_testbench_architecture
    FOR ALL:my_design USE ENTITY work.my_design(structure);
      FOR structure
        FOR ALL:roc USE ENTITY work.roc(roc)v)
          GENERIC MAP ( width => 100 ms);
        END FOR;
      END FOR;
    END FOR;
  END FOR;
END cfg_my_timing_testbench;
```

Verilog designs do not require ports to drive the global/set reset net from a testbench. Therefore Verilog designs do not contain the ROC cell. The same signal name found in the front end can be used to drive the signal in the back-annotated design. The signal must be driven, or all flip-flops will initialize as X.

VHDL designs that contain oscillator cells like OSC, OSC4, or OSC5, must have the clock period set with a configuration statement. By default, the period is 0, disabling the oscillator. You should carefully select the period from the range of viable periods found in the Xilinx Databook for the particular technology. A specific period is not guaranteed because the cell is subject to process variations. You should select the value that best meets your simulation requirements.

You can use the following configurations for either the OSC, OSC4, or OSC5 cells by just changing the name of the cell and modifying the pulse width to the correct value.

```
CONFIGURATION cfg_my_functional_testbench OF my_testbench IS
  FOR my_testbench_architecture
    FOR ALL:my_design USE ENTITY work.my_design(my_design_rtl);
      FOR my_design_rtl
        FOR ALL:my_submodule USE ENTITY
          work.my_submodule(my_submodule_rtl);
          FOR my_submodule_rtl
            FOR all: osc4 USE ENTITY work.osc4(structure)
              GENERIC MAP ( period_8m => 125 NS);
            END FOR;
          END FOR;
        END FOR;
      END FOR;
    END FOR;
  END FOR;
END cfg_my_testbench_functional;
```

You can drive Verilog designs by the signal name used to drive the front-end simulation since the hierarchical name is preserved.

Simulating the Design

Simulate with ModelSim using vsim. To include the timing information in the SDF file, invoke vsim with the –sdftyp option. Refer to the Mentor documentation for information on available options.

Chapter 6

Mixed Designs with Schematic on Top

This chapter describes how to use the Mentor Graphics Interface to design with mixed schematic and VHDL designs with schematic on top. It contains the following sections.

Design Flow

The design flow for designs containing a mixture of schematics and VHDL is illustrated in the following figure.

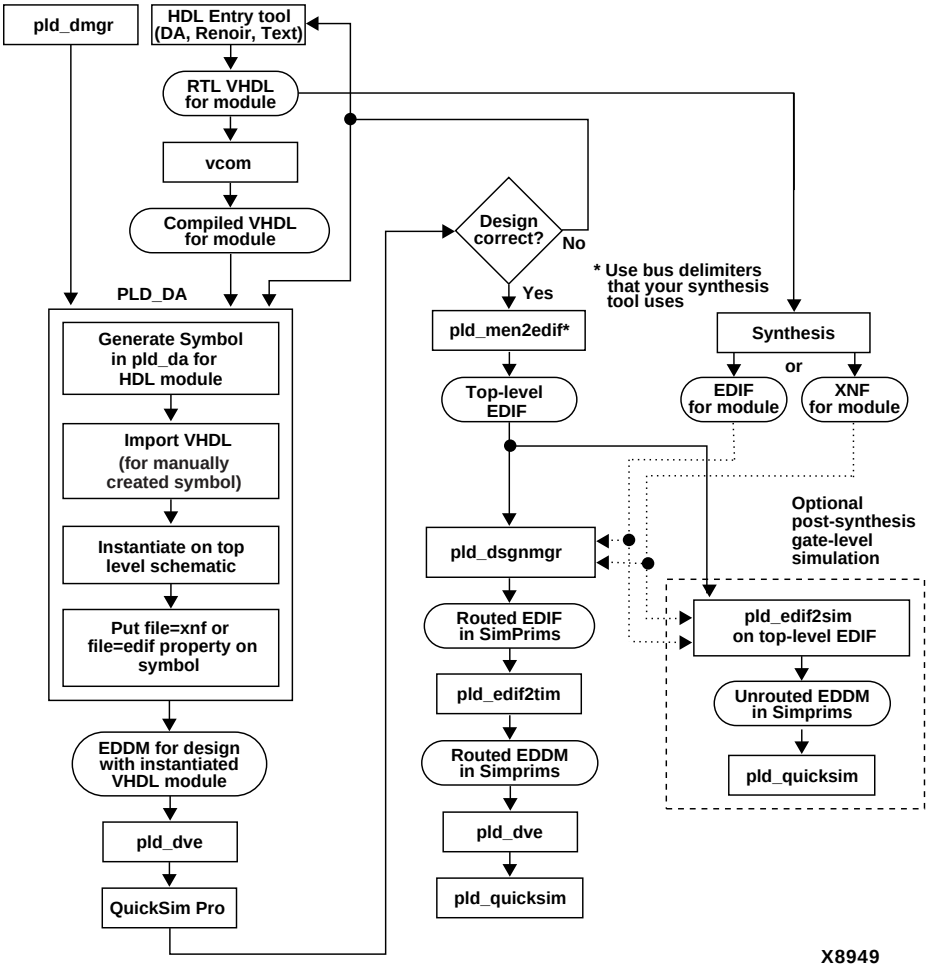


Figure 6-1 Mixed Schematic and VHDL Design with Schematic on Top

Design Entry

Design entry consists of two parts, VHDL module design entry and schematic entry.

VHDL Module Design Entry

To enter the VHDL module of your design and to get it ready for functional simulation and implementation, perform the following steps.

1. Enter the VHDL portion of your design.
2. Compile the VHDL source files with vcom.

   vcom [*options*] *design_file(s)* **–qspro_syminfo**
3. In pld_da, use **`File`** → **`Generate`** → **`Symbol`** to import VHDL and create a symbol for the VHDL module as shown in the following figure.

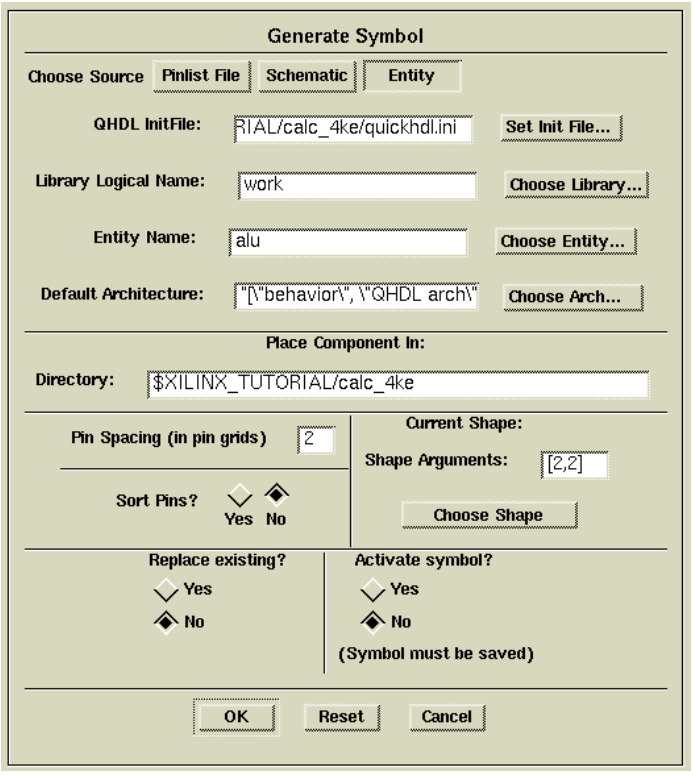


Figure 6-2 Generate Symbol Dialog Box

4. On the symbol, add the file=*xnf_file_pathname* or file=*edif_file_pathname* property with a value that specifies the path to the XNF or EDIF file that will be synthesized from the RTL description you created above. (In this manual, file=*value* means to add the file property and set its value to *value*.)

5. Check and save the new symbol.

Refer to the Mentor documentation for details on using Generate Symbol.

Schematic Entry

Perform the following steps:

1. Enter the top-level and lower-level schematic portions as described in the “Design Entry” section of the “Schematic Designs” chapter.
2. Instantiate the symbol created for the VHDL module on the top-level design.

Functional Simulation

Mixed-model schematic-based designs can be composed of schematic elements from the Unified Libraries, VHDL, XNF-based components, or EDIF-based components. The VHDL-based components will later have FILE=*edif_path* properties for implementation.

You can simulate the design either before or after you synthesize the HDL module.

Functional Simulation Before Synthesis

The flow diagram for this procedure is shown in Figure 6-4. Follow these steps to simulate your design before you synthesize it.

1. Generate a symbol for the HDL module with pld_da.
2. Instantiate the symbol on the schematic.
3. Put FILE=*xnf_file_pathname* or FILE=*edif_file_pathname* property on the symbol of the synthesized module. (In this manual, file=*value* means to add the file property and set its value to *value*.)
4. Create a viewpoint for the top-level design using pld_dve.

   pld_dve –s *design_name technology* [*viewpoint_name*]
5. Run QuickSim Pro to simulate the design by typing the following syntax.

   qhpro [*options*] *design_name*

   Alternate ways to invoke QuickSim Pro are to double-click the left mouse button on the QuickSim Pro icon in the Design Manager Tools window or to select the top-level component in the Navigator window and click the right mouse button.

The QuickSim Pro dialog box appears, as shown in the following figure.

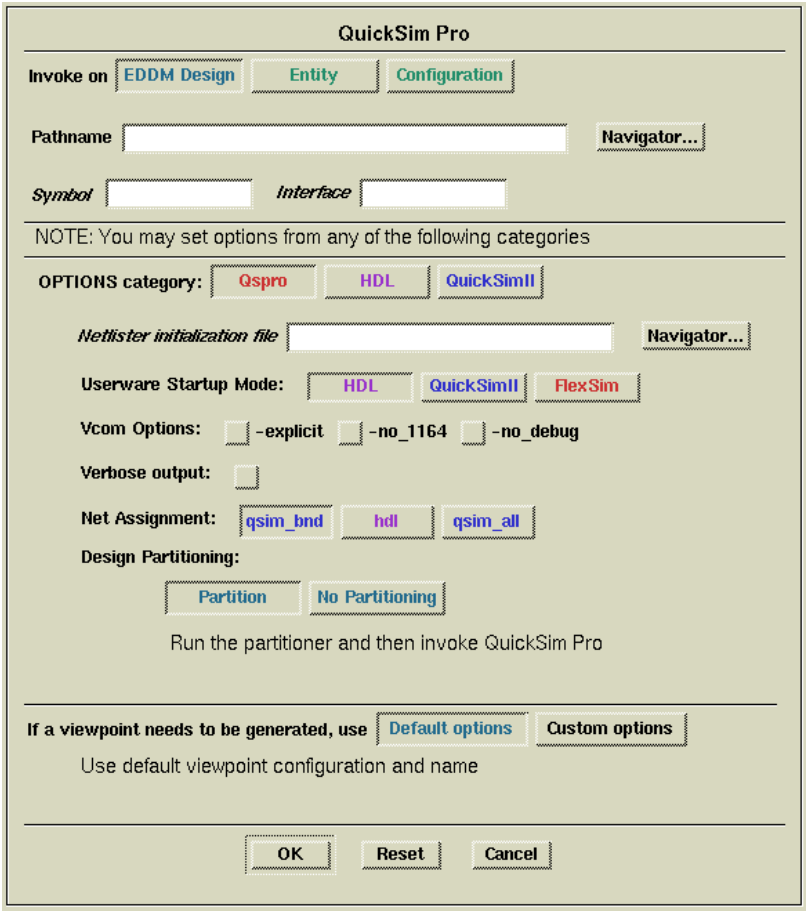


Figure 6-3 QuickSim Pro Dialog Box

6. In this dialog box, click **`EDDM Design`** in the Invoke On field.
7. In the Pathname field, type in the path name of the component.
8. Type the symbol name in the Symbol field only. This step is optional.
9. Type the interface name in the Interface field only. This step is optional.

10. Click **OK** to invoke the ModelSim simulator and perform simulation.
11. After simulation you may proceed to synthesis.

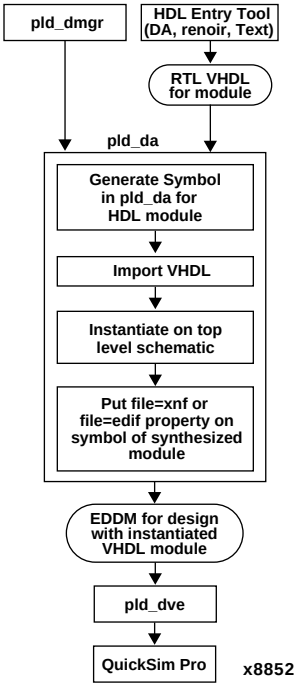


Figure 6-4 Performing Functional Simulation Before Synthesis on Mixed-Model Schematic-on-Top Designs

Synthesis

You may use the tool of your choice for synthesis of the HDL component. For detailed information on performing synthesis, refer to the following Xilinx documents.

- *Synthesis and Simulation Design Guide*
- *Exemplar Tutorial* on the Xilinx Web site at `http://support.xilinx.com/support/techsup/tutorials/index.htm`

Functional Simulation After Synthesis

You can optionally re-simulate the design at this point to ensure that the design's functionality remains optimal. This method for simulating your design does not require the use of QuickSim pro. The flow diagram for this procedure is shown in Figure 6-5.

If the synthesis tool created an EDIF file, you can include a symbol for the module within the top level design with file=*edif_file_name*. Then submit the whole design to pld_edif2sim, and then submit it to pld_quicksim.

If the synthesis tool created an XNF file, you can include a symbol for the module within the top level design with file=*xnf_file_name*. Then submit the whole design to pld_men2sim, pld_edif2sim, and then pld_quicksim.

Follow these steps to simulate by this method.

1. Synthesize the HDL module that is being included on the schematic, and create an EDIF or XNF file from that synthesis.
2. Create a symbol for the HDL module with pld_da and add the file=*edif_file_name* or file=*xnf_file_name* property to the symbol. Instantiate the symbol on the top level design.
3. Run pld_men2edif on the top level design to create an EDIF for the whole design. Make sure to specify the appropriate bus delimiter to match the synthesized module.
4. Run pld_edif2sim to convert it to a Mentor EDDM single object.

   pld_edif2sim *edif_file symbol_component_name technology* {**–m** | **–s**} **–eddm**

   Use –m if the synthesis was performed with a Mentor tool; use –s if the synthesis was performed with a Synopsys tool.
5. Perform functional simulation with pld_quicksim.

   pld_quicksim *design_name*[*/viewpoint_name*]

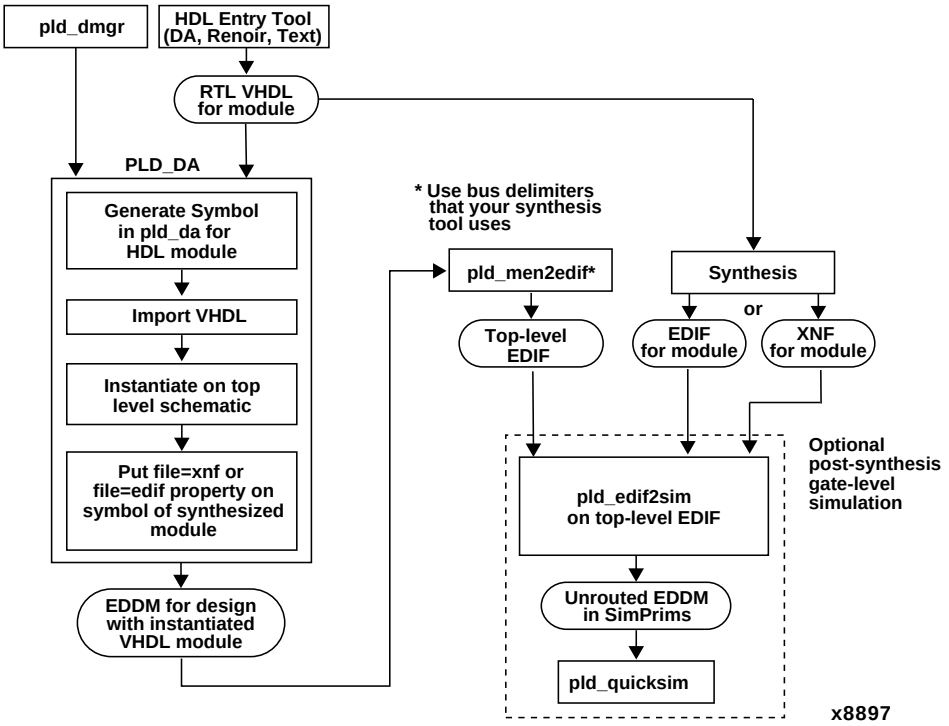


Figure 6-5 Performing Functional Simulation After Synthesis on Mixed-Model Schematic-on-Top Designs

Design Implementation

After functional simulation, use a synthesis tool that creates a Xilinx compatible EDIF or XNF file to synthesize certain blocks of the design described in VHDL.

After synthesis, you must attach a FILE=*design*.edif or FILE=*design*.xnf property to the VHDL-based block symbol in the schematic before you submit the top-level EDDM design to pld_men2edif.

Converting the EDDM Design

You convert the top-level EDDM design to EDIF with the pld_men2edif utility. To convert your design to EDIF, follow these steps.

1. In the Mentor Design Manager, double-click the left mouse button on the **pld_men2edif** icon.

   The Mentor to EDIF Netlist dialog box displays.

2. In the Component Name field, enter the component name, or click on **Navigator** to browse a list of design names.

3. In the From Viewpoint field, you can enter the viewpoint name if you do not want to use the default viewpoint. Alternatively, in step 2 you can select a viewpoint under the component.

4. Select the appropriate architecture for your design in the PLD Technology field.

5. Select the desired bus notation style.

   Be careful to select the Bus Dimension Separator Style that matches your synthesizer's style. Otherwise busses between the schematic portion and the HDL portion will not match up in the implemented design.

6. Click **OK**.

   pld_men2edif now produces an EDIF file that you can submit to the Xilinx Design Manager, pld_dsgnmgr. The output name is *component_name*.edif.

Implementing the Design

The Xilinx Design Manager, pld_dsgnmgr, can accept an EDIF file or if your design is a pure XNF design, it can accept an XNF file.

In the Mentor Design Manager, double-click the left mouse button on the pld_dsgnmgr icon.

Because the implementation is essentially the same as for a pure schematic design, follow the directions in the "Implementing Schematic Designs" section of the "Schematic Designs" chapter.

Normally you need an EDIF file to bring back the design into the EDDM environment. But you have the option of creating a VHDL or Verilog and an SDF file instead of an EDIF file, which you can submit to ModelSim for timing simulation.

Timing Simulation

This is the same as the “Timing Simulation for Schematic Designs” section of the “Schematic Designs” chapter. When reading this section, be aware that cross-probing does not apply to the VHDL component.

Chapter 7

Mentor/Xilinx Flow Manager

This chapter describes how to use the Mentor / Xilinx Flow Manager. It provides an overview of the Flow Manager followed by a description of each flow supported by the Flow Manager. It contains the following sections.

Flow Manager Overview

The Mentor / Xilinx Flow Manager is a dialog box that provides a visual guide of the steps you need to perform for five common design flows. Each step contains buttons to launch the appropriate tool and to display a visual record of your progress in the flow. It does not automatically perform the steps for you. It lists the steps in the correct order that you need to perform. For each step there is a button that launches the appropriate tool. When you are finished with the tool, you click the Finished button for that step and the description for that step changes to indicate that it is finished.

The following example shows the basic steps for using the Mentor / Xilinx Flow Manager.

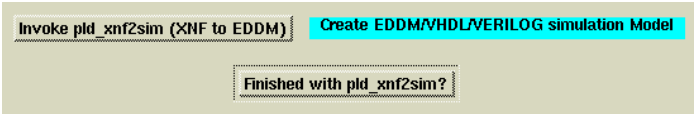
1. Click the Flow Manager icon in the Mentor Design Manager tools window to open the Mentor / Xilinx Flow Manager dialog box.

Since the Flow Manager is a dialog box, its size is determined by the size of the Mentor Design Manager Window. If necessary, use the Page Up / Page Down keys to scroll the Flow Manager dialog box.

2. Select the desired flow in the top portion of the Mentor / Xilinx Flow Manager dialog box.

The contents of the Mentor / Xilinx Flow Manager dialog box change to show the selected flow.

3. Click the first flow step button.

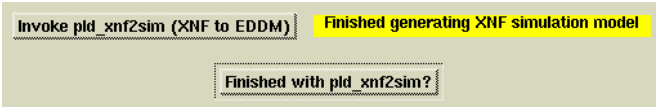


The Mentor / Xilinx Flow Manager launches the appropriate tool.

4. After you finish using the tool, exit the tool.

   After the tool closes, you are returned to the Mentor / Xilinx Flow Manager.

5. In the Mentor / Xilinx Flow Manager, click the Finished button for the current flow step.



Notice that the description and highlight color change to indicate that this step is complete.

6. Continue in the same manner with the remaining flow steps.

7. If you need to reset the flow, you can click the Reset button located at the bottom of the Mentor / Xilinx Flow Manager dialog box.

8. When you are finished with the Mentor / Xilinx Flow Manager, click the Exit Flow Manager button.

The details of each flow are described in the following pages.

Pure Schematic Design Flow

This flow consists of the following steps. Each step lists the name of the tool that you invoke followed by a reference to information in this manual about how to perform the step. The dialog box for this design flow is shown in Figure 7-1.

1. Start Schematic Design Entry
   - pld_da (Design Architect)
   - See the “Design Entry” section of the “Schematic Designs” chapter.
2. Create Functional Simulation Viewpoint
   - pld_dve (Viewpoint Editor)
   - See the “Creating the Viewpoint” section of the “Schematic Designs” chapter.
3. Run Functional Simulation
   - pld_quicksim (QuickSim)
   - See the “Simulating the Design” section of the “Schematic Designs” chapter.
4. Create EDIF from Schematic Design
   - pld_men2edif (EDIF writer)
   - See the “Converting the EDDM Design to EDIF” section of the “Schematic Designs” chapter.
5. Implement the EDIF / XNF Design
   - pld_dsgnmgr (Xilinx Design Manager)
   - See the “Implementing the Design” section of the “Schematic Designs” chapter.

6. Create the Timing Simulation Model
   - pld_edif2tim (EDIF reader)
   - See the “Creating the EDDM Model and the Viewpoint” section of the “Schematic Designs” chapter.
7. Run Timing Simulation
   - pld_quicksim (QuickSim)
   - See the “Simulating the Design” section of the “Schematic Designs” chapter.

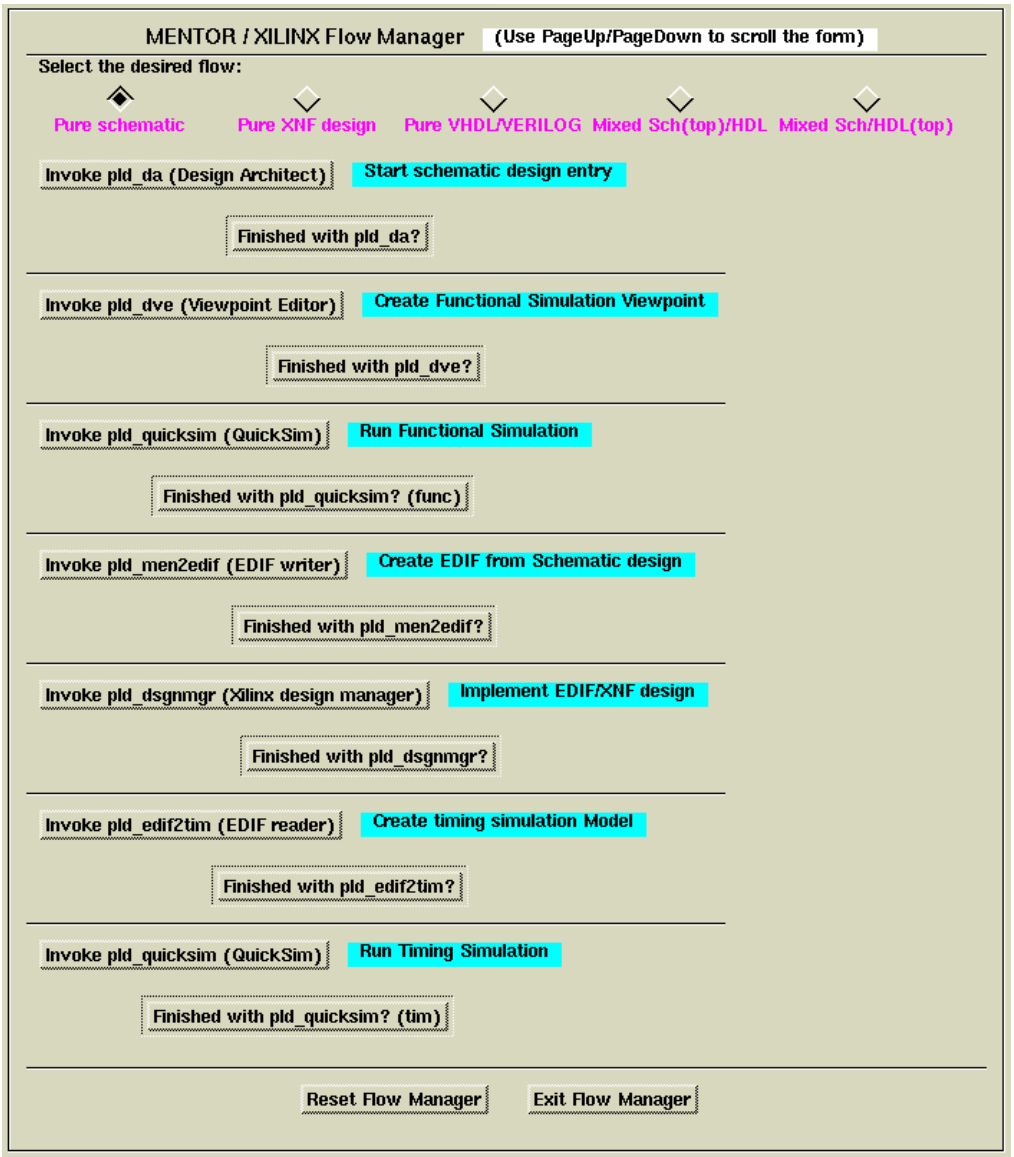


Figure 7-1 Mentor/Xilinx Design Manager Pure Schematic Mode

Pure XNF Design Flow

This flow consists of the following steps. Each step lists the name of the tool that you invoke followed by a reference to information in this manual about how to perform the step. The dialog box for this design flow is shown in Figure 7-2.

1. Create the EDDM / VHDL / Verilog Simulation Model
   - pld_xnf2sim (XNF to EDDM)
   - See the "Simulating Schematic Designs with XNF Elements" section of the "Schematic Designs" chapter.
2. Run Functional Simulation
   - pld_quicksim (QuickSim)
   - See the "Simulating the Design" section of the "Schematic Designs" chapter.
3. Implement the EDIF / XNF Design
   - pld_dsgnmgr (Xilinx Design Manager)
   - See the "Implementing the Design" section of the "Schematic Designs" chapter.
4. Create the Timing Simulation Model
   - pld_edif2tim (EDIF reader)
   - See the "Creating the EDDM Model and the Viewpoint" section of the "Schematic Designs" chapter.
5. Run Timing Simulation
   - pld_quicksim (QuickSim)
   - See the "Simulating the Design" section of the "Schematic Designs" chapter.

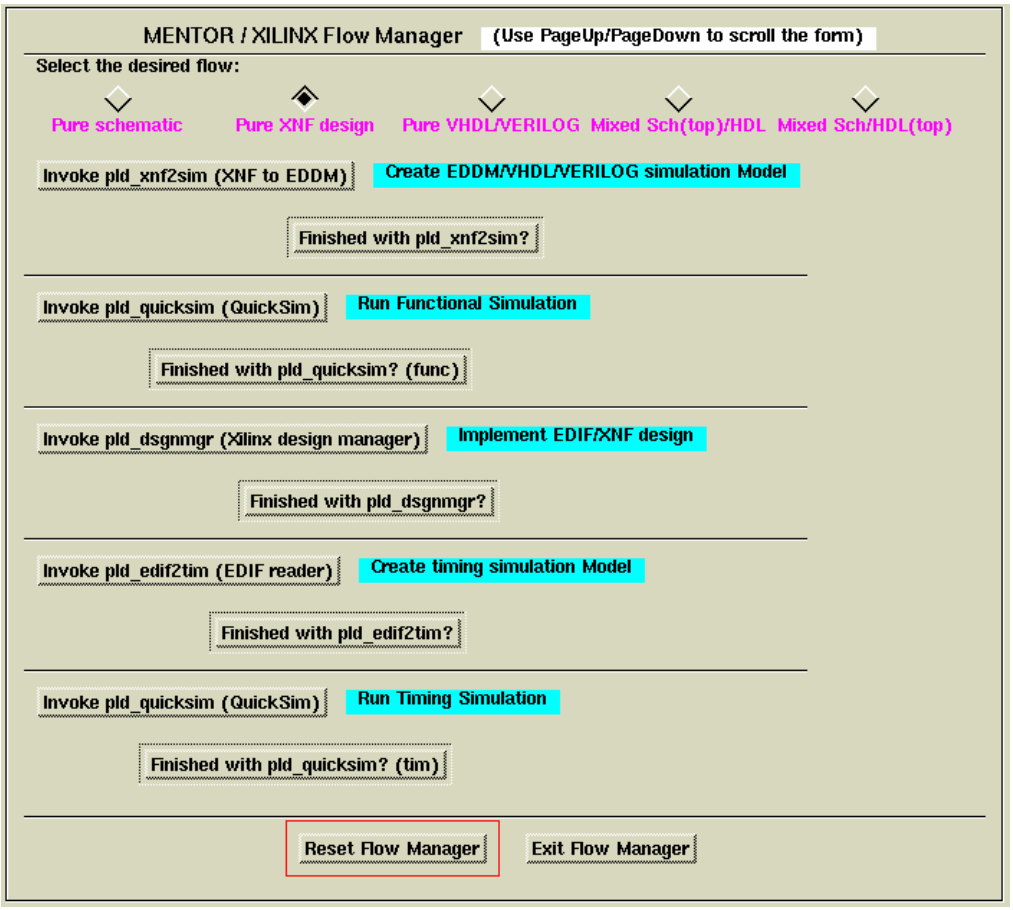


Figure 7-2 Mentor/Xilinx Design Manager Pure XNF Mode

Pure VHDL/Verilog Design Flow

This flow consists of the following steps. Each step lists the name of the tool that you invoke followed by a reference to information in this manual about how to perform the step. The dialog box for this design flow is shown in Figure 7-3.

1. Select the desired VHDL / Verilog Creation Tool
   - DA, Renoir, or text editor
   - See the “HDL Design Entry” section of the “HDL Designs” chapter
2. Create the HDL Design
   - pld_da (Design Architect), Renoir, text editor
   - See the “HDL Design Entry” section of the “HDL Designs” chapter
3. Run VHDL / Verilog Functional Simulation
   - vsim (MTI Simulator modelsim)
   - See the “Pre-Synthesis Functional Simulation” section of the “HDL Designs” chapter
4. Run VHDL / Verilog Synthesis
   - Synthesis tool
   - See the “Synthesis” section of the “HDL Designs” chapter
5. Run ModelSim on post synthesis HDL

   YES or NO
6. Run VHDL / Verilog after synthesis simulation (OPTIONAL)
   - vsim (MTI Simulator modelsim)
   - See the “Functional Simulation” section of the “HDL Designs” chapter
7. Implement the EDIF / XNF Design
   - pld_dsgnmgr (Xilinx Design Manager)
   - See the “Design Implementation” section of the “HDL Designs” chapter
8. Run VHDL / Verilog Timing Simulation
   - vsim (MTI Simulator modelsim)
   - See the “Simulating the Design” section of the “HDL Designs” chapter.

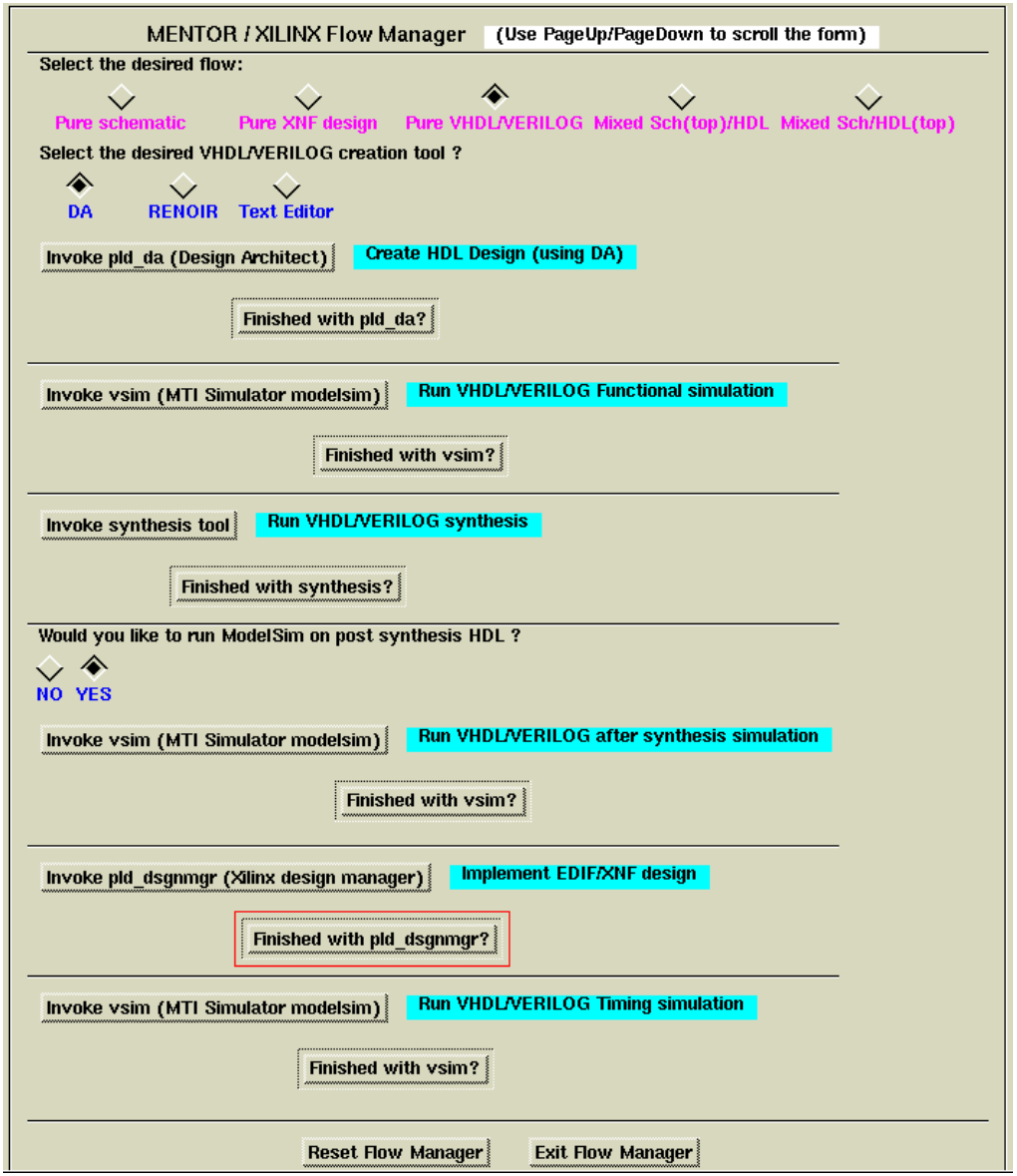


Figure 7-3 Mentor/Xilinx Design Manager Pure VHDL/Verilog Mode Running ModelSim on Post Synthesis HDL

Mixed Schematic (top)/HDL Design Flow

This flow consists of the following steps. Each step lists the name of the tool that you invoke followed by a reference to information in this manual about how to perform the step. The dialog box for this design flow is shown in Figure 7-4.

1. Select the desired VHDL / Verilog Creation Tool
   - DA, Renoir, or text editor
   - See the “Design Entry” section of the “Mixed Designs with Schematic on Top” chapter
2. Create the HDL Design
   - pld_da (Design Architect), Renoir, text editor
   - See the “Design Entry” section of the “Mixed Designs with Schematic on Top” chapter
3. Run VHDL / Verilog Synthesis
   - Synthesis tool
   - See the “Synthesis” section of the “Mixed Designs with Schematic on Top” chapter
4. Start the schematic design entry
   - pld_da (Design Architect)
   - See the “Design Entry” section of the “Mixed Designs with Schematic on Top” chapter
5. Generate Symbol for HDL Model
   - Generate Symbol
   - See the “VHDL Module Design Entry” section of the “Mixed Designs with Schematic on Top” chapter
6. Create the Functional Simulation Viewpoint
   - pld_dve (Viewpoint Editor)
   - See the “Functional Simulation” section of the “Mixed Designs with Schematic on Top” chapter

7. Run Functional Simulation
   - qspro (QuickSim Pro)
   - See the “Functional Simulation” section of the “Mixed Designs with Schematic on Top” chapter
8. Create EDIF from the Schematic Design
   - pld_men2edif (EDIF writer)
   - See the “Functional Simulation After Synthesis” section of the “Mixed Designs with Schematic on Top” chapter
9. Implement the EDIF / XNF Design
   - pld_dsgnmgr (Xilinx Design Manager)
   - See the “Design Implementation” section of the “Mixed Designs with Schematic on Top” chapter
10. Create the Timing Simulation Model
    - pld_edif2tim (EDIF reader)
    - See the “Design Implementation” section of the “Mixed Designs with Schematic on Top” chapter
11. Run Timing Simulation
    - pld_quicksim (QuickSim)
    - See the “Timing Simulation for Schematic Designs” section of the “Schematic Designs” chapter. When reading this section, be aware that cross-probing does not apply to the VHDL component.

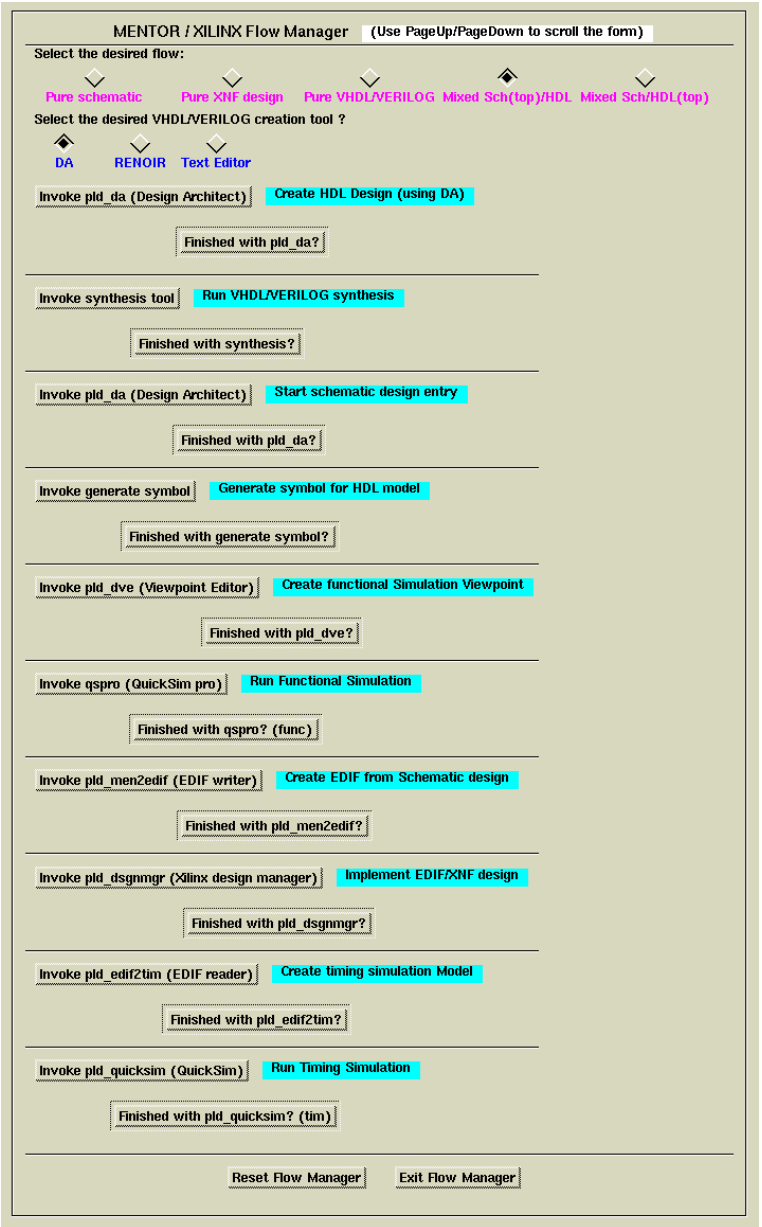


Figure 7-4 Mentor/Xilinx Design Manager Mixed Sch(top)/HDL Mode

Mixed Sch/HDL(top) Design Flow

In this mode, the Flow Manager dialog box has the following two sub-flows which you select at the top part of the dialog box.

- Create EDDM
- Simulate / Implement Top HDL.

To make the steps easier to understand, this document splits the Simulate / Implement Top HDL flow into a Simulate / Implement flow and an optional Simulate Synthesis Output flow.

This flow consists of the following steps. Each step lists the name of the tool that you invoke followed by a reference to information in this manual about how to perform the step.

The dialog box for the Create EDDM sub-flow is shown in Figure 7-5. The dialog box for the Simulate / Implement Top HDL sub-flow is shown Figure 7-6.

Create EDDM

To create EDDM, perform the following steps:

1. Create schematic for the EDDM module
   - pld_da (Design Architect)
   - See the “Design Entry” section of the “Mixed Designs with VHDL on Top” chapter
2. Create viewpoint for EDDM module
   - pld_dve (Viewpoint Editor)
   - See the “Design Entry” section of the “Mixed Designs with VHDL on Top” chapter
3. Create Entity / Architecture for EDDM module
   - genarch (Entity / Arch creation tool)
   - See the “Design Entry” section of the “Mixed Designs with VHDL on Top” chapter

4. Create EDIF for Schematic module
   - pld_men2edif (EDIF writer)
   - See the “Design Entry” section of the “Mixed Designs with VHDL on Top” chapter
5. Create NGO file from schematic module EDIF
   - pld_edif2sim (EDIF to EDDM, choose NGO only)
   - See the “Design Entry” section of the “Mixed Designs with VHDL on Top” chapter

Simulate/Implement Top HDL (Main Flow)

To simulate/implement top HDL, perform the following steps:

1. Select the desired VHDL/VERILOG creation tool
   - DA, Renoir, or text editor
   - See the “HDL Design Entry” section of the “HDL Designs” chapter
2. Create HDL Design
   - pld_da (Design Architect), Renoir, text editor
   - See the “HDL Design Entry” section of the “HDL Designs” chapter
3. Run Functional Simulation on top-level HDL
   - qspro (QuickSim Pro)
   - See the “Functional Simulation” section of the “Mixed Designs with VHDL on Top” chapter
4. Run VHDL/Verilog Synthesis
   - Synthesis tool
   - See the “Synthesis” section of the “Mixed Designs with VHDL on Top” chapter

5. Run simulation on post synthesis output?
   - YES or NO.
   - If you select YES, perform the steps in the following "Simulate Synthesis Output (Optional)" section before completing the following steps six and seven.
6. Implement EDIF / XNF design
   - Invoke pld_dsgnmgr (Xilinx Design Manager)
   - See the "Design Implementation" section of the "Mixed Designs with VHDL on Top" chapter
7. Run VHDL / Verilog Timing Simulation
   - Invoke vsim (MTI Simulator Modelsim)
   - See the "Timing Simulation" section of the "Mixed Designs with VHDL on Top" chapter

Simulate Synthesis Output (Optional)

This is an optional step that you can perform before implementation. For instructions on performing the following steps, see the "Optional Post-Synthesis Functional Simulation" section of the "Mixed Designs with VHDL on Top" chapter. This flow consists of the following steps.

1. Run simulation on post synthesis output. (Optional)

   YES or No. Select YES. (See step five in the preceding "Simulate / Implement Top HDL (Main Flow)" section.)
2. Select desired synthesis output

   HDL, EDIF, or XNF
3. Create EDDM / VHDL / VERILOG simulation model (EDIF)

   pld_edif2sim (EDIF to EDDM)
4. Create EDDM / VHDL / VERILOG simulation model (XNF)

   pld_xnf2sim (XNF to EDDM)
5. Run Functional Simulation (HDL)

   qspro (QuickSim Pro)

6. Run VHDL / Verilog Simulation (EDIF)
   vsim (MTI Simulator modelsim)
7. Run VHDL / Verilog Simulation (XNF)
   vsim (MTI Simulator modelsim)
8. Implement EDIF / XNF design
   Invoke pld_dsgnmgr (Xilinx Design Manager)
9. Run VHDL / Verilog Timing Simulation
   vsim (MTI Simulator modelsim)

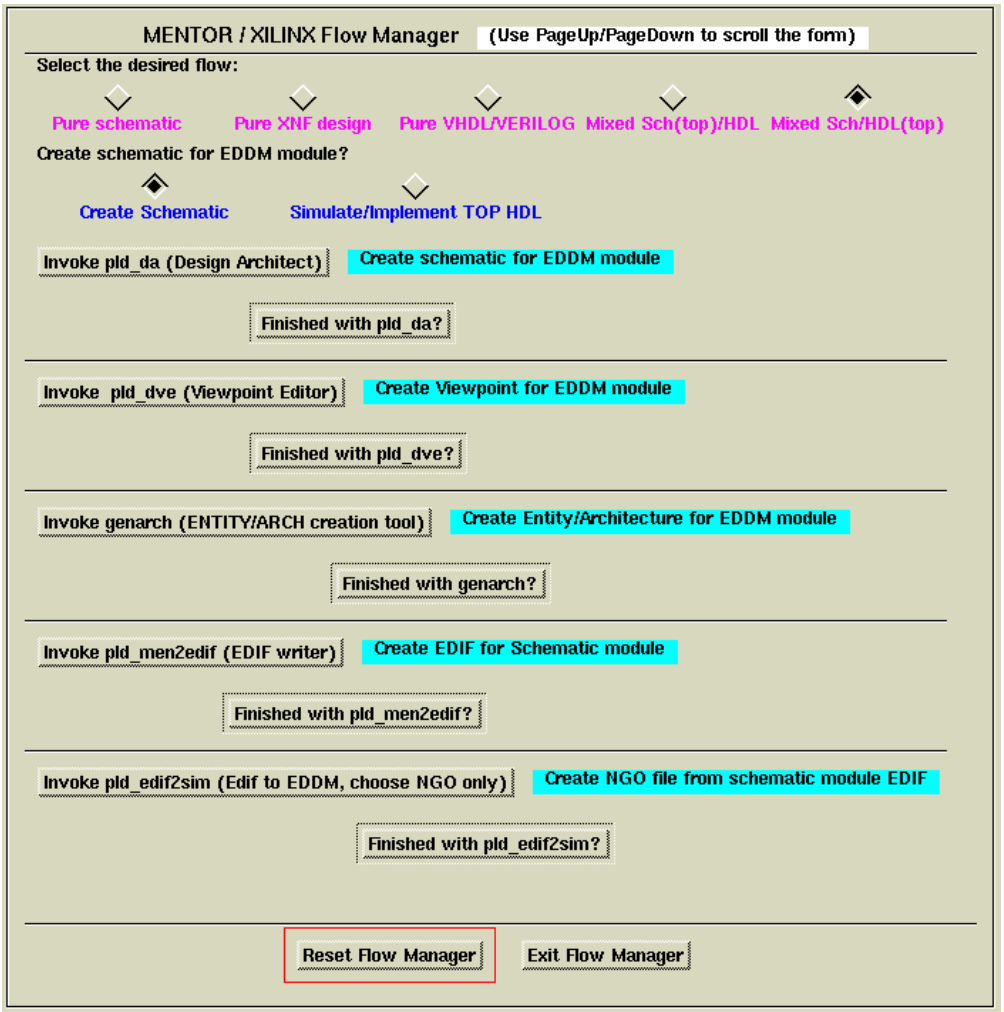


Figure 7-5 Mentor/Xilinx Design Manager Mixed Sch/HDL(top) — Create EDDM Mode

Figure 7-6 Mentor/Xilinx Design Manager Mixed Sch/HDL(top) — Simulate/Implement Top HDL Mode

Chapter 8

Advanced Techniques

This chapter discusses aspects of schematic entry and simulation that you should be familiar with to use Design Architect and pld_quicksim effectively.

This chapter contains the following sections.

Retargeting the Design to a Different Family

The Unified Libraries allow you to retarget your designs from one device family to another if both your source and target designs only include symbols from the Unified Libraries. Since most of the symbols in the Unified Libraries have the same footprint and name from one device family to another, you can easily convert your designs across Xilinx device families.

The procedure described in the following section uses Xilinx's Convert Design utility in Design Architect to retarget your schematic. It allows you to change every reference of every design object in your design directory from the source design library to the target design library. In your target design, the symbols that are common to the source and target families maintain their relative location and pin position in the schematic. Pins on these symbols retain their connectivity to the nets they were attached to in the source design.

You must manually replace symbols that are not common to your source and target families with equivalent logic. For example, if a GCLK was used in an XC3000A design that is retargeted for use in an XC4000E device, you must manually replace the GCLK symbol with a BUFGP, BUFG, or BUFGS, which is the XC4000E equivalent of a GCLK.

Note In the following procedures, XC4000 is used as the source design device family, and XC5200 is used as the target design device family. You can also retarget other device families.

To retarget a design to a different family, perform these steps.

1. Activate Design Architect by using either of the methods described in the "Invoking Design Architect" section of the "Schematic Designs" chapter. You do not have to open the schematic.
2. On Design Architect's desktop background (the area outside any schematic or symbol windows) press the right mouse button and select **`Convert Design`**.

   The dialog box shown in the following figure appears.

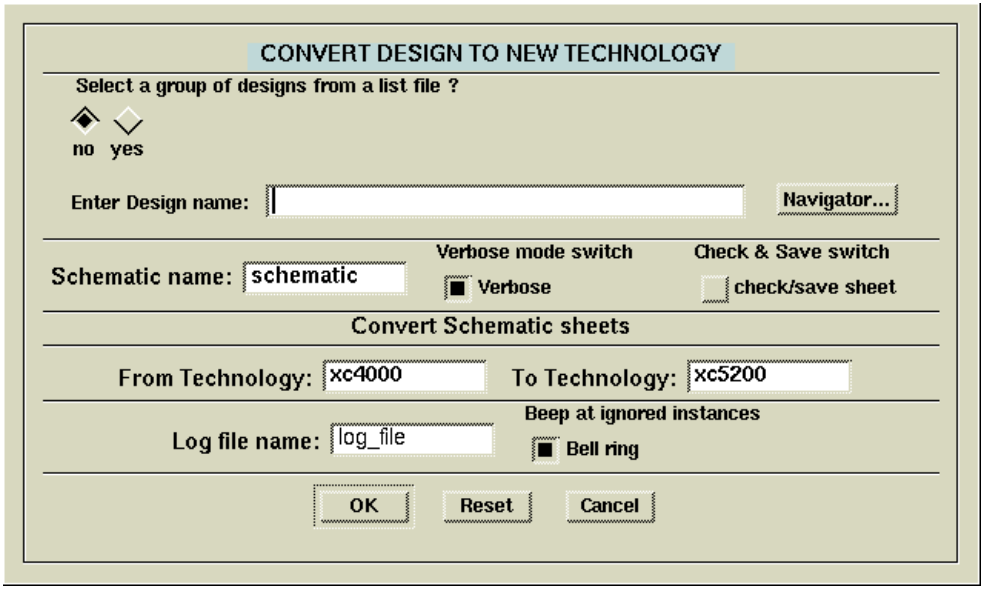


Figure 8-1 Convert Design To New Technology Dialog Box

3. In the field asking “Select a group of designs from a list file?,” click **yes** or **no**.

   - Click **no** if you want to retarget a single design. Convert Design utility traverses the hierarchy of a given schematic and converts the schematics of any hierarchical blocks found on the top-level schematic.
   - Click **yes** if you have a number of designs to retarget, and their names are contained in a file, one design per line. This file is useful if your design has many lower-level schematics.

   Note You can create a list file with the UNIX ls command. The ls command lists all the MGC components within a single directory, and the sed command strips the trailer from .mgc_compoennt.attr. The result is directed to the list file.

   ```
   ls *.mgc_component.attr | sed s/
   .mgc_component.attr//g > listfile
   ```

4. In the Enter Design Name field, enter the design name or the name of the file listing the designs to retarget.
5. In the Schematic Name field, enter the name of the schematic model.

   The default is Schematic.

6. Select the **Verbose** mode switch.
7. Leave the Check and Save Switch field set to its default setting, manual checking, to allow you to find Xilinx components that do not convert properly. Once you become familiar with Convert Design’s operation, you can select this field to have Convert Design automatically check and save the schematic.
8. In the From technology field, type the name of the device family from which you are converting. This field is case-insensitive.
9. In the To Technology field, type the name of the device family to which you are converting. This field is case-insensitive.
10. If you want the results of the conversion saved to a log file, type the name of the log file in the Log File Name field. The default is log_file.
11. Set a beep to sound for every un-matched symbol.
12. Click **OK** to start the conversion.

Merging Design Files from Other Sources

You can enter part of your design in a form other than schematics, such as text entry or a RAM or ROM description. You can also bring in netlist files produced by interface software other than Mentor Graphics. Whatever the form of entry, the starting point for inclusion into a Mentor Graphics schematic design must be a netlist file in EDIF format. EDIF netlist files must be located in the working directory. Without the EDIF file, this portion of the design cannot be included; with it, the origin of the logic becomes irrelevant. To incorporate the EDIF file into your schematic, you must create a symbol for the file and place it on your schematic as you would any other component. You also need to attach a FILE=*edif_file_pathname* property to the symbol.

Simulation Models

Most Xilinx simulation models are built with Mentor Graphics QuickPart tables. Flip-flops and memory elements are modeled with QuickPart tables and behavioral language models, while gates are modeled with QuickPart tables. All delay information is passed to Xilinx components through the routed EDIF, Verilog, or VHDL file.

Setting Global Reset and 3-State Signals

The way you set Global Reset and 3-State signals depends on which part type you are using. The methods are described below.

FPGA Designs

Before you simulate an FPGA design, you must force the / / globalsetreset (XC4000E designs) or the / / globalreset (XC5200 designs) or the / / globalresetb (XC3000 designs); otherwise, the flip-flops and latches do not function correctly.

1. Select your design directory icon in the Navigator window and select **`Right Mouse Button`** → **`Open`** → **`pld_quicksim`** to enter the pld_quicksim simulator.
2. Select the **`File`** → **`Open Sheet`** menu item to display the Design Architect schematic.

3. Select the **Add Force** menu from the pld_quicksim Stimulus palette.
4. Fill in the dialog box with the / /globalsetreset signal name, 25 for the first time, and 1 for the first value; n for the second time, and 0 for the second value.

   It is recommended that you do not force signals at time 0. See Mentor's *QuickSim User Guide* for details.

   The reset width emulates a power-on reset at the beginning of simulation. Globalsetreset is now forced High at n ns. If you want to reset the flip-flops after n ns, toggle the globalsetreset Low and High for the necessary pulse width specified in The *Xilinx Programmable Logic Data Book*.

The previous procedure is slightly different for XC4000 IOBs and 3-state I/O pins.

To set XC4000E/EX IOB flip-flops, follow these instructions.

1. Set the IOB flip-flops High or Low on power-up by using the INIT property on the IOB flip-flops.
2. To activate the signal and begin simulation, set globalsetreset by selecting the **Add Force** menu item from the pld_quicksim Stimulus palette.
3. Fill in the dialog box with the / /globalsetreset signal name, 25 for the first time and 1 for the first value; n for the second time and 25 for the second value.

   It is recommended that you do not force signals at time 0. See Mentor's *QuickSim User Guide* for details.

   N is the specified minimum reset pulse width for the given speed grade part of the design, specified in The *Xilinx Programmable Logic Data Book*.

   XC4000E/EX parts have a global input state to make all output pins 3-state, which allows the isolation of the XC4000E/EX part in board test. To simulate the global 3-state signal, force the signal named / /globalthreestate High using the Add Force command. Forcing the signal High holds all chip I/Os in a high-Z (3-state) state until / /globalthreestate is forced to zero.

CPLD Designs

Before you simulate a XC7000 or XC9000 CPLD design, you must force the / / prld; otherwise, the flip-flops do not function correctly.

1. Select your design directory icon in the Navigator window and select **`Right Mouse Button`** → **`Open`** → **`pld_quicksim`** to enter the pld_quicksim simulator.
2. Select the **`File`** → **`Open Sheet`** menu item to display the Design Architect schematic.
3. Select the **`Add Force`** menu from the pld_quicksim Stimulus palette.
4. Fill in the dialog box with the / / prld signal name, 25 for the first time, and 1 for the first value; *n* for the second time, and 0 for the second value.

   It is recommended that you do not force signals at time 0. See Mentor's *QuickSim User Guide* for details.

   The reset width emulates a power-on reset at the beginning of simulation. If you want to reset the flip-flops after n ns, toggle the prld High and Low for the necessary pulse width specified in The *Xilinx Programmable Logic Data Book*.

Using TAU

Tau is a board-level timing analysis tool from Mentor Graphics designed to do system timing analysis, as opposed to transmission line analysis which is the focus of IS Analyzer / Floorplanner. Tau checks that timing constraints such as setup and hold requirements on component inputs are met. To determine if these requirements are satisfied, it is necessary to take into account interconnect delay on the board, component delay, and the skew and phase shift between clocks. This system timing analysis is performed in the digital domain. You can perform this analysis before beginning physical design to identify interconnect delay constraints that must be satisfied by a board.

The Xilinx 3.1i development system also features Stamp Model generation which can be used with Tau. With Stamp Model generation, you can accelerate board level design verification for Xilinx programmable logic products.

For more information on using the Xilinx tools Trace to create Stamp models, and how to integrate with Tau please see the Application Note 166 at `http://www.xilinx.com/xapp/xapp166.pdf`.

The Xilinx program Trace (**`trce`**) can produce Stamp files which are used to pass timing data about the Xilinx FPGA to Tau. By default, Trace reports back all timing paths covered by constraints (.pcf file), but this may not be enough information in the STAMP file since it is possible that not all inputs and outputs are reported because they might not be covered by user constraints. You can use the following options to force Trace to evaluate all the paths.

- -a is used for advanced design analysis in the absence of a constraint file (.pcf)
- -u reports additional paths which are not covered by the constraints within the PCF file
- -s <speed> runs analysis with the specified speed grade

By default, Trace does not use the constraint file (.pcf) so you must specify it on the command line when running **`trce`**. Use the -stamp option to generate the Stamp .mod and .dat files. An example follows.

```
trce -u -s 1 -o report_filename -stamp stamp_filename
routed_file.ncd file.pcf
```

To create the Stamp files for minimum timing, use the -s min option and run Trace again. Support for the -s min option is only available for 4KXL and later families and may not currently be available for all technologies. Run Trace once to get min and run it again to get max timing. This creates two .dat and two .mod files. You only need one .mod and the two .dat files to import into Taulib. The following is an example of creating Stamp files for minimum timing.

```
trce -u -s min min_report_filename -stamp
min_stamp_filename routed_file.ncd file.pcf
```

Refer to the *Xilinx Development System Reference Guide* for more information about Trace.

Taulib is the tool you use to import the cell timing for the Xilinx FPGA cell. At the board-level, the chips (FPGAs in this case) are considered cells. Following are the steps for importing the cell timing with Taulib.

1. Within Taulib, import the Stamp information by selecting the cell name that represents the FPGA, if it already exists. Do this by clicking on the very left hand box next to the name. This highlights the entire row. If the cell does not already exist, taulib automatically creates one.
2. Import the Stamp model (.mod) and the data (.dat) file using the **File** → **Import** menus.
3. If only the timing information is to be read, choose Override Existing Timing model.
4. If a new timing model is to be created, choose Create New Timing Model.
5. Choose the appropriate Timing Value to interpret the delay values in the data file as either minimum or maximum.
6. Click **OK** and select the Cell Timing sheet to examine the imported timing information.

Refer to the Mentor Tau documentation for more information on running Tau.

Chapter 9

Manual Translation

You can access the programs required to simulate and implement your design through the graphical user interface of the Mentor Design Manager or through the UNIX command line.

The first half of this chapter discusses the program sequence for performing functional simulation, design implementation, and timing simulation from the UNIX command line for different types of designs. The second half describes the syntax of the individual programs.

This chapter contains the following sections.

Functional Simulation

The following sections describe functional simulation for a variety of designs.

Pure Schematic Designs

Perform functional simulation for schematic designs as follows:

1. Create a viewpoint using pld_dve.

   `pld_dve -s` *design_name technology* [*viewpoint_name*]

2. Perform functional simulation with pld_quicksim.

   `pld_quicksim` *design_name* [/ *viewpoint_name*]

Schematic Designs with XNF Elements

Perform functional simulation as follows:

1. Create a symbol in pld_da for each XNF element in your design.
2. To the symbols, add the FILE property with the path name of the XNF file as the value.
3. Run pld_men2edif to convert the entire design into EDIF.
4. Run pld_edif2sim on this EDIF file to create a design component that represents the entire design.

   pld_edif2sim *edif_file component_name technology* **–m –eddm** [**–sd dir**]

   Use –sd to search additional directories other than the one containing the source EDIF file to find supporting EDIF, NGO, or XNF files.
5. Perform functional simulation with pld_quicksim.

   pld_quicksim *design_name*[*/viewpoint_name*]

Schematic Designs with LogiBLOX or CORE Generator Elements

Schematic designs with LogiBLOX or CORE Generator elements already contain simulation models so you only need to create a viewpoint and then simulate.

1. Create a viewpoint using pld_dve

   pld_dve –s *design_name technology* [*viewpoint_name*]
2. Perform functional simulation with pld_quicksim.

   pld_quicksim *design_name*[*/viewpoint_name*]

Mixed Schematic and VHDL with Schematic-on-Top Designs

You can simulate the design either before or after you synthesize the HDL module.

Before Synthesis

Follow these steps to simulate your design before you synthesize it.

1. Compile the VHDL module into a work library. If using Mentor version B.2 and up, use –qhpro –syminfo when compiling, otherwise Generate Symbol in the Design Architect will fail.
2. Create a symbol for the HDL module with pld_da using **`File`** → **`Miscellaneous`** → **`Symbol`**.
3. The Generate Symbol dialog box opens as shown in Figure 9-1.
4. In the Generate Symbol dialog box, choose **`Entity`** as the source and specify the library logical name, entity name, and default architecture.
5. Instantiate the symbol on the schematic.
6. Create a viewpoint using pld_dve.

   `pld_dve -s` *design_name technology* [*viewpoint_name*]
7. Run QuickSim PRO to simulate the design by typing the following syntax.

   `qhpro` [*options*] *design_name*

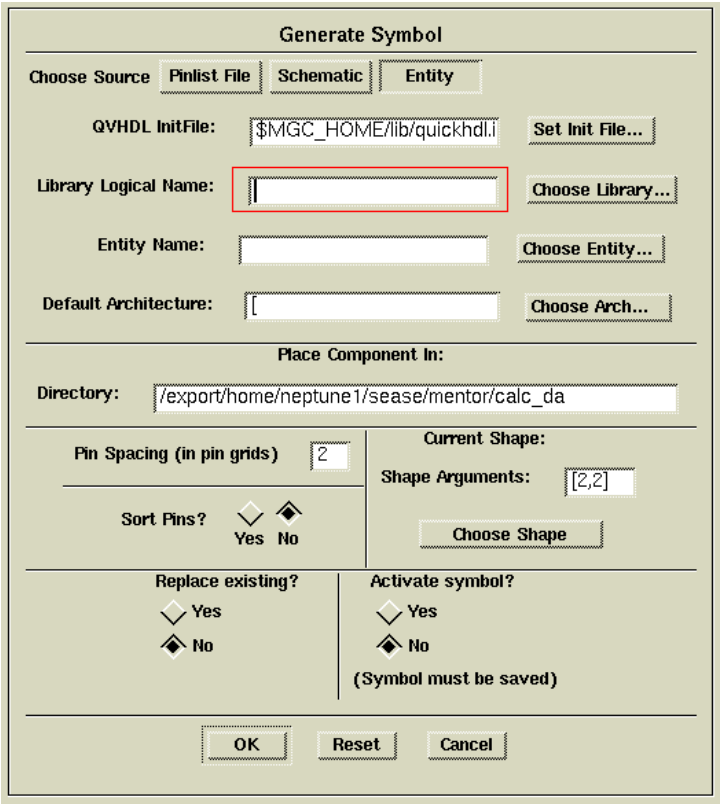


Figure 9-1 Generate Symbol Dialog Box

After Synthesis

To simulate your VHDL design after you synthesize it, follow these steps.

1. Synthesize the HDL module that is being included on the schematic, and create an EDIF file from that synthesis.
2. Create a symbol for the HDL module with pld_da.
3. If the synthesis output was an EDIF file, run pld_edif2sim to convert it to a Mentor EDDM single object.

   `pld_edif2sim` *edif_file symbol_component_name technology* {**`-m`**|**`-s`**} **`-eddm`** [**`-sd dir1 ... -sd dirn`**]

Use –m if the synthesis was performed with a Mentor tool; use –s if the synthesis was performed with a Synopsys tool.

4. Perform functional simulation with pld_quicksim.

   `pld_quicksim` *design_name*[*/viewpoint_name*]

   Where *design_name* is the EDDM design created by pld_edif2sim.

HDL-at-Top Designs

EDDM models must be inserted in the top-level HDL file.

1. Create a work library.
2. Perform the following steps for any schematic based components that need to be included in the top level VHDL.
   a) Run **`pld_dve -s`** to create a viewpoint for each EDDM component.
   b) Make sure the EDDM has an underlying symbol associated with it. If not create one using **`pld_da`** → **`Miscellaneous`** → **`Generate Symbol`**. Specify Schematics as the source in the dialog box.
   c) Run gen_arch to create entity and architecture source files.
   d) Instantiate this component into the top-level VHDL file.
3. Compile the VHDL source files with vcom.

   `vcom` [*options*] *design_name*

   See the Mentor documentation for a description of the available options.
4. Run QuickSim PRO to simulate the design by typing the following syntax.

   `qhpro` [*options*] *design_name*

   For a description of the QuickSim PRO options, see the Mentor Graphics documentation.

Pure HDL Designs

Perform the following steps.

1. Create a working library.
2. Compile the HDL source files with vcom.

   vcom [*options*] *design_name*

   See the Mentor documentation for a description of the available options.
3. Simulate the design by running ModelSim. Type the following syntax.

   vsim [*options*] [*–lib_name*] [**primary** [*architecture* [**primary**] ...]

Design Implementation

This section explains how to implement various design types.

Schematic Designs (FPGA)

The procedure for implementing pure schematic designs, designs with XNF elements, designs with LogiBLOX elements, designs with CORE Generator modules, and mixed-model schematic-at-top designs is the same. Follow these steps.

1. Convert the EDDM design to EDIF format with pld_men2edif.

   pld_men2edif *design_name technology* [*viewpoint_name*] [**–b** *bus_delimiter*]
2. Submit the design to NGDBuild, which reads a file in EDIF or XNF format, reduces all the components in the design to Xilinx primitives, runs a logical design rule check on the design, and writes an NGD file as output.

   ngdbuild –p *technology design_name*

   For example.

   ngdbuild –p xc4000ex test –sd dir
3. Map the logic to the components in the FPGA by typing the following syntax.

   map *design_name***.ngd –p** *partname*

For example.

map –p 4000EXHQ240-3 test.ngd

4. Place and route the design.

   par –w *design_name*.**ncd** *design_name*.**ncd**

   The first file is created by the MAP utility, and PAR creates the other one.

   For example.

   par –w test.ncd test.ncd (writes out test.ncd created by map)

   par –w test.ncd test_par.ncd (writes new file test_par.ncd)

5. Back-annotate the design.

   ngdanno *design_name*.**ncd** *design_name*.**ngm**

6. Convert the design to an EDIF file.

   ngd2edif –a –v mentor *design_name*.**nga –w**

7. Submit the design to pld_edif2tim, the Mentor EDIF netlist reader, which converts an EDIF netlist to a Mentor single-object EDDM file that can be submitted to pld_quicksim for timing simulation. Use this syntax.

   pld_edif2tim *design_name*.**edn**

   This step creates a design library, *design*_lib, containing the design on which you can perform timing simulation.

Schematic Designs (CPLD)

When using CPLDs, the procedure for implementing pure schematic designs, designs with XNF elements, and mixed-model schematic-at-top designs is the same. Follow these steps.

1. Convert the EDDM design to EDIF format with pld_men2edif.

   pld_men2edif *design_name technology* [*viewpoint_name*]

2. Submit the design to the CPLD fitter.

   cpld –p *partname design_name* [**–sd dir**]

3. Convert the design to an EDIF file.

   ngd2edif –a –v mentor *design_name*.**nga –w**

4. Submit the design to pld_edif2tim, the Mentor EDIF netlist reader, which converts an EDIF netlist to a Mentor single-object EDDM file that can be submitted to pld_quicksim for timing simulation. Use this syntax.

   pld_edif2tim *design_name*.**edn**

   This step creates a design library, *design*_lib, containing the design on which you can perform timing simulation.

HDL-at-Top Designs

To implement HDL-at-top designs, perform the following steps.

1. Synthesize the HDL modules in your design, and create an EDIF or XNF file from that synthesis.

2. Convert the EDIF or XNF file to an NGD file by using ngdbuild.

   ngdbuild –p *technology design_name*

   For example.

   ngdbuild –p XC4000E test (where test is the root name for the EDIF or XNF file)

   Note Referenced Mentor EDDM models must have their corresponding EDIF files created and residing in the same directory where the top level EDIF or XNF file resides. If they reside in other directories, you must use the –sd option to specify additional directories to search for such files.

3. Map the logic to the components in the FPGA by typing the following syntax.

   map *design_name*.**ngd –p** *partname*

   For example.

   map –p 4000EXHQ240-3 test.ngd

4. Place and route the design.

   par –w *design_name*.**ncd** *design_name*.**ncd**

The first file is created by the MAP utility, and PAR creates the other one.

For example.

`par –w test.ncd test.ncd` (writes out test.ncd created by map)

`par –w test.ncd test_par.ncd` (writes new file test_par.ncd)

5. Back-annotate the design.

   `ngdanno` *design_name*`.ncd` *design_name*`.ngm`

6. Back-annotate the design.

   `ngdanno` *design_name*`.ncd` *design_name*`.ngm`

7. Submit the design to pld_edif2tim, the Mentor EDIF netlist reader, which converts an EDIF netlist to a Mentor single-object EDDM file that can be submitted to pld_quicksim for timing simulation. Use this syntax.

   `pld_edif2tim` *design_name*`.edn`

   This step creates a design library, *design_lib*, containing the design on which you can perform timing simulation.

Pure HDL Designs

To implement pure HDL designs, perform the following steps.

1. Synthesize the HDL file, and create an EDIF or XNF file from that synthesis.
2. Convert the EDIF or XNF file to an NGD file by using ngdbuild.

   `ngdbuild –p` *technology design_name*

   For example,

   `ngdbuild –p XC4000E test` (where test is the root name for the EDIF or XNF file)

   Note Referenced Mentor EDDM models must have their corresponding EDIF files created and residing in the same directory where the top level EDIF or XNF file resides.

3. Map the logic to the components in the FPGA by typing the following syntax.

   map *design_name*.**ngd –p** *partname*

   For example.

   map –p 4000EXHQ240-3 test.ngd

4. Place and route the design.

   par –w *design_name*.**ncd** *design_name*.**ncd**

   The first file is created by the MAP utility, and PAR creates the other one.

   For example,

   par –w test.ncd test.ncd (writes out test.ncd created by map)

   par –w test.ncd test_par.ncd (writes new file test_par.ncd)

5. Back-annotate the design.

   ngdanno *design_name*.**ncd** *design_name*.**ngm**

6. Convert the design to an EDIF file.

   ngd2edif –a –v mentor *design_name*.**nga –w**

7. Submit the design to pld_edif2tim, the Mentor EDIF netlist reader, which converts an EDIF netlist to a Mentor single-object EDDM file that can be submitted to pld_quicksim for timing simulation. Use this syntax.

   pld_edif2tim *design_name*.**edn**

   This step creates a design library, *design_lib*, containing the design on which you can perform timing simulation.

Timing Simulation

This section explains how to perform timing simulation for various designs.

Schematic Designs

The procedure for performing timing simulation on pure schematic designs, designs with XNF elements, designs with LogiBLOX elements, designs with CORE Generator modules, and mixed-model schematic-at-top designs is the same. Follow these steps.

1. Use pld_edif2tim to create a Mentor EDDM model.

   `pld_edif2tim` *design_name*`.edn`

2. Create a viewpoint using pld_dve.

   `pld_dve -s` *design_lib/design technology* [*viewpoint_name*]

3. Run pld_quicksim to perform the timing simulation by using the following syntax.

   `pld_quicksim -cp` *design_lib/design_name*

   This command brings up DVE for cross-probing.

   For example,

   `pld_quicksim -cp test_lib/test`

4. Cross-probe between the original design and the new design.

5. Open the Viewpoint that was used to create the original design EDIF netlist.

6. Open the schematic sheet in pld_dve.

7. Select the signals to trace in the pld_dve schematic.

   Pld_quicksim automatically creates a trace window and adds the selected signals to it. Use pld_dve's schematic sheet window as if it were the sheet in the pld_quicksim window.

Pure HDL Designs

You can create either an output EDIF file or output VHDL / Verilog file from the Xilinx Design Manager (or Xilinx core tool scripts).

EDIF Method

To create an EDIF output file, perform the following steps.

1. Submit the design to pld_edif2tim, which converts an EDIF netlist to a Mentor single-object EDDM file that can be submitted to pld_quicksim for timing simulation. Use this syntax.

   pld_edif2tim *design_name*.**edn**

   This step creates a design library, *design_lib*, containing the design on which you can perform timing simulation.

2. Create a viewpoint with pld_dve.

   pld_dve -s *design_lib/design_name technology*

3. Simulate the timing with pld_quicksim.

   pld_quicksim *design_lib/design_name*

VHDL/Verilog Method

To create an VHDL / Verilog output file, perform the following steps.

1. Compile the HDL source files with vcom.

   vcom [*options*] *design_name*

   See the Mentor documentation for a description of the available options.

2. Simulate the timing with ModelSim.

   vsim *options* [*–lib_name*] [**primary** [*architecture* [**primary**] ...]

Program Summary

This section briefly describes the UNIX command-line syntax of the commands that activate the Mentor and Xilinx programs that you can use to process your designs manually. They are listed in alphabetical order.

CPLD

CPLD is a C-shell script for fitting into the XC7000 and XC9000 families. For a description of the CPLD command syntax and options, see the *CPLD Schematic Design Guide* or run the CPLD command with no parameters

Dsgnmgr

Dsgnmgr, the Xilinx Design Manager, is Xilinx's design implementation tool.

The dsgnmgr syntax can take the following three forms.

dsgnmgr

dsgnmgr *project*

dsgnmgr –design *design*.**edif**

When you use the first form of the syntax, the Design Manager appears with no project loaded. A project in this context means a Xilinx project.

When you use the second form of the syntax, the Design Manager appears but with the specified project loaded or opened. The project is a fully specified file name with a .prj extension. It is a file created by the Design Manager and contains the project information for a Xilinx project.

When you use the third form of the syntax, the Design Manager finds the design. A design in this context is a netlist file such as an EDIF file. If the design does not already have a Xilinx project associated with it, the Design Manager creates a project and appears with this project loaded. If the design already has a Xilinx project associated with it, the Design Manager appears with that project loaded.

EDIF2NGD

Edif2ngd converts an EDIF 2 0 0 netlist to a Xilinx NGO file. The EDIF file includes the hierarchy of the input schematic. The output NGO file is a binary database describing the design in terms of the components and hierarchy specified in the input design file.

For a description of the edif2ngd syntax and options, see the *Development System Reference Guide*.

Editor

The Notepad editor is a full-featured, window-based text editor. It is only available in the graphical user interface of the Mentor tools.

Gen_Arch

Gen_Arch creates VHDL entity and architecture from a Mentor (EDDM) component.

For a description of the Gen_Arch syntax and options, see the Mentor Graphics documentation.

MAP

MAP is a Xilinx tool that maps the logic to the components in an FPGA design.

For a description of the MAP syntax and options, see the *Development System Reference Guide*.

NGDAnno

NGDAnno is Xilinx's back-annotation utility.

For a description of the NGDAnno syntax and options, see the *Development System Reference Guide*.

NGDBuild

NGDBuild reads a file in EDIF or XNF format, reduces all the components in the design to Xilinx primitives, runs a logical design rule check on the design, and writes an NGD file as output.

For a description of the NGDBuild syntax and options, see the *Development System Reference Guide.*

NGD2EDIF

NGD2EDIF converts a Xilinx NGD or NGA file to an EDIF 2 0 0 netlist.

For a description of the NGD2EDIF syntax and options, see the *Development System Reference Guide.*

PAR

PAR is Xilinx's place and route tool.

For a description of the PAR syntax and options, see the *Development System Reference Guide.*

Pld_da

Pld_da is Design Architect, a schematic editor configured for Xilinx designs. For a description of Design Architect, see the *Mentor Graphics Design Architect Users Manual.*

Pld_dve

Pld_dve creates a simulation or custom viewpoint for a Xilinx design.

The pld_dve syntax is the following.

`pld_dve` [**`-s`**] *`design_name technology`* [*`viewpoint_name`*]

- –s creates a simulation viewpoint for pld_quicksim (chip-level / board-level functional / timing). It is optional. If you do not use –s but specify a viewpoint name, pld_dve opens in the interactive mode and opens the specified viewpoint.
- *design_name* is the name of your Mentor design component.
- *technology* specifies the PLD architecture.
- *viewpoint_name* specifies the name of the design viewpoint to generate. This is optional; pld_dve does not perform any customization on this viewpoint if –s is not specified.

When pld_dve creates a simulation viewpoint—that is, when you use the –s option—and if the viewpoint contains COMP or FILE primitives, pld_dve removes these primitives, then creates a viewpoint that can be submitted to pld_quicksim.

Pld_edif2sim

Pld_edif2sim is a utility that converts a Mentor, Synopsys, or any other Xilinx compatible EDIF file into a Mentor EDDM single object, VHDL netlist, Verilog netlist, or NGO file.

The pld_edif2sim syntax is the following.

```
pld_edif2sim edif_file symbol_component_name |
output_file_name technology {-s|-o|-m} {-eddm|-vhdl|-
verilog|-ngo} {-hier|-flat} {-ignore_unexpanded} [-sd
dir1 ... -sd dirn] [-help]
```

- *edif_file* is the name of the EDIF file from Mentor, Synopsys, or Data I/O
- *symbol_component_name* is the name of the component. This is used for the –eddm option.
- *output_file_name* is the name of the output VHDL or Verilog. This is used for the –vhdl or –verilog options.
- *technology* specifies the PLD architecture.
- –ngo specifies that pld_edif2sim should produce a *design_name*.ngo file only.
- –s indicates that the EDIF file is a Synopsys file.
- –o indicates that the EDIF file is any third party vendor's EDIF that is compatible with Xilinx.
- –m indicates that the EDIF file is a Mentor file.
- –eddm specifies that the EDIF file be converted to Mentor's EDDM single object.
- –vhdl specifies that the EDIF file be converted to a VHDL file.
- –verilog specifies that the EDIF file be converted to a Verilog file.
- –sd specifies additional directories to search to find any supporting EDIF, XNF, or NGO files.
- –hier specifies that the VHDL/Verilog netlist is hierarchical.

- –flat specifies that the VHDL / Verilog netlist is flat (default is flat).
- –ignore_unexpanded specifies that if there are any unknown primitives in the design, pld_edif2sim does not exit with an error status; instead, it ignores this condition and goes on. By default, it exits with an error message.
- –help allows you to obtain more information on pld_edif2sim and its options. It is optional.

Pld_edif2tim

Pld_edif2tim is the Mentor EDIF netlist reader, which converts an EDIF netlist to a Mentor single-object EDDM file that can be submitted to pld_quicksim for timing simulation.

The pld_edif2tim syntax is the following.

```
pld_edif2tim edif_file [-r] [-help]
```

- *edif_file* is the name of the EDIF file.
- –r specifies that if *design_lib* already exists, it will be replaced.
- –help allows you to obtain more information on pld_edif2tim and its options. It is optional.

Pld_men2edif

Pld_men2edif is the Mentor EDIF netlist writer, which creates a hierarchical EDIF netlist from a Mentor schematic design.

The pld_men2edif syntax is the following.

```
pld_men2edif design_name technology [viewpoint_name]
[-b 'delimiter'] -circular [-help]
```

- *design_name* is the name of your Mentor design component.
- *technology* specifies the PLD architecture.
- *viewpoint_name* specifies the name of the design viewpoint to use. It is optional. If a viewpoint does not exist, pld_men2edif will create one. If you do not specify the viewpoint, it will use the viewpoint called default.

- –circular overcomes the forward referencing problem that occurs if a primitive in one library is referenced in another library before its parent library is defined in EDIF. In this case the EDIF reader fails to process the EDIF file. The –circular switch prevents this problem.
- –b *'delimiter'* specifies the bus dimension separator style as an angle bracket, square bracket or paren.

  delimiter is one of the following: Angle | Square | Paren

  The –b option instructs the EDIF writer to convert the bus delimiters into the specified delimiter. If –b is not specified, '()' will be used for bus delimiters by default.
- –help allows you to obtain more information on pld_men2edif and its options. It is optional.

Pld_quicksim

Pld_quicksim is an interactive logic simulator that performs functional or timing simulation on your designs.

The pld_quicksim syntax is the following.

pld_quicksim [**–cp**] *design_name*[*/viewpoint_name*]

- –cp ensures that cross-probing is performed. It is optional. If you specify this option, QuickSim invokes DVE to allow viewing the front-end schematic for cross-probing. You must then open the viewpoint on the original design that was used to create the EDIF netlist.
- *design_name* is the name of your Mentor design directory.
- *viewpoint_name* specifies the name of the design viewpoint to use. It is optional. If you specify a viewpoint name, it must be preceded with a slash and appended to the design name, as in the following example.

  ```
  pld_quicksim test/myvpt
  ```

  For a description of the other options available in pld_quicksim, see the Mentor Graphics *QuickSim Users and Reference Manuals*.

To enable cross-probing between front-end and back-end designs in timing simulations, specify –cp. In this case, the syntax is the following.

```
pld_quicksim -cp test_lib/test
```

Pld_sg

Pld_sg invokes the Mentor schematic generator (SG), which creates a schematic from an EDDM model. You must have a Mentor schematic generator license in order to use this tool. Usage is as follows.

```
pld_sg [options] [viewpoint_path]
```

See the Mentor documentation for a description of the available options.

Pld_xnf2sim

Pld_xnf2sim is a utility that converts an XNF file to a Mentor EDDM single object, VHDL netlist, or Verilog netlist.

The pld_xnf2sim syntax is the following.

```
pld_xnf2sim top-level_xnf_file [-list listfile]
symbol_component_name | output_file_name technology
{-ignore_unexpanded} [-s] {-eddm|-vhdl|-verilog} {-
hier|-flat} [-sd dir1 ... -sd dirn] [-help]
```

- *top-level_xnf_file* is the top-level XNF file.
- –list *listfile* allows you to list all the related XNF files to be converted. It is optional. If you do not specify –list, all XNF files located in the directory in which the top-level XNF file resides are used as referenced by the top-level XNF file.
- *symbol_component_name* is the name of the Mentor component for which a simulation model is to be created.
- *output_file_name* is the name of output VHDL or Verilog (for –vhdl or –verilog option)
- *technology* specifies the PLD architecture.
- –s indicates that the XNF file is a Synopsys file. It is optional.
- –eddm specifies that the XNF file be converted to an EDDM single object.

- –vhdl specifies that the XNF file be converted to a VHDL file.
- –verilog specifies that the XNF file be converted to a Verilog file.
- –hier specifies that VHDL / Verilog is hierarchical.
- –flat specifies that VHDL / Verilog is flat (This is the default).
- –ignore_unexpanded specifies that if there are any unknown primitives in the design, pld_edif2sim does not exit with an error status; instead, it ignores this condition and goes on. By default, it exits with an error message.
- –sd specifies additional directories to search to find any supporting EDIF, XNF, or NGO files.

XNF file(s) submitted to pld_xnf2sim must represent the entire design, including the top-level IO ports (EXT statements). Feeding an XNF file that only represents one part of a design (with no IO pads) results in an invalid simulation model. You can use the following procedure to run functional simulation on a schematic design that consists of a partial XNF.

1. Create symbols representing the XNF files.
2. Add the FILE property with the value equal to the pathname of the XNF file.
3. Instantiate these symbols on your schematic.
4. Create an EDIF file with pld_men2edif (using the top-level schematic).
5. Feed this EDIF file to pld_edif2sim to create an EDDM model.
6. Simulate this EDDM model with pld_quicksim.

ModelSim

ModelSim (vsim), is Mentor's simulator for behavioral VHDL, Verilog, VHDL-based, and Verilog-based gate-level designs composed of Unified Libraries or SimPrim elements.

The ModelSim syntax is the following.

```
vsim options [-lib_name] [primary [architecture
[primary] ...]
```

For a description of the ModelSim options, see the Mentor Graphics documentation.

Note This documentation assumes that you are using ModelSim. QuickHDL provides the same functionality as ModelSim. If you are using QuickHDL instead of ModelSim, see the "ModelSim" section of the "Introduction" chapter for details on how to use QuickHDL in place of ModelSim.

QuickPath

QuickPath performs a static and slack timing analysis on designs. For a description of the QuickPath syntax and options, see the Mentor Graphics documentation.

QuickSim Pro

QuickSim Pro (qspro) is Mentor's simulator for mixed-model schematic, VHDL, and Verilog designs. It can invoke ModelSim to simulate HDL-based elements, or QuickSim to simulate gate-level schematics.

The QuickSim Pro syntax is the following.

```
qhpro options design_name
```

For a description of the QuickSim Pro options, see the Mentor Graphics documentation.

Vcom

Vcom compiles the VHDL to be able to run ModelSim (vsim) simulator.

```
vcom [options] design_name
```

See the Mentor documentation for a description of the available options.

Vlog

Vlog compiles the Verilog files to be able to run ModelSim (vsim) simulator.

```
vlog [options] design_name
```

See the Mentor documentation for a description of the available options.

Index

E

F

G

H

I

L

M

S

T