Xilinx/ Concept-HDL Interface Guide

Xilinx/Concept-HDL Interface Guide — 2.1i *Printed in U.S.A.*

About This Manual

This document describes the Xilinx/Concept-HDL interface, an HDL-based design creation tool.

Before using this manual, you should be familiar with the operations that are common to all Xilinx software tools: how to bring up the system, select a tool for use, specify operations, and manage design data. These topics are covered in the *Alliance Quick Start Guide* and the *Foundation Quick Start Guide*. Other publications you can consult for related information are the *Libraries Guide*, *Design Manager/Flow Engine User Guide*, and the *Development System Reference Guide*.

Additional Resources

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

Resource	Description/URL
Tutorial	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
Application Notes	Descriptions of device-specific design techniques and approaches http://support.xilinx.com/apps/appsweb.htm

Resource	Description/URL
Data Book	Pages from *The Programmable Logic Data Book*, which describe device-specific information on Xilinx device characteristics, including read-back, 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
Tech Tips	Latest news, design tips, and patch information on the Xilinx design environment http://support.xilinx.com/support/techsup/journals/index.htm

Manual Contents

This manual covers the following topics.

- Chapter 1, "Getting Started" chapter—Provides an overview of Concept-HDL and describes the Xilinx/Concept-HDL design flow.
- Chapter 2, "Using Setup" chapter—Describes how to set up a new Concept-HDL project or change existing setup information for projects with Xilinx designs.
- Chapter 3, "Using Concept-HDL with Xilinx Designs" chapter—Describes how to enter Xilinx-specific design information in Concept-HDL.
- Chapter 5, "Conducting Simulation" chapter—Provides information required to simulate Xilinx designs in Concept-HDL.
- Chapter 4, "Using Genview" chapter—Describes how to use Genview to create and change views used in Concept-HDL.
- Chapter 6, "Upgrading to Concept-HDL" chapter—Describes how to convert Concept-SCALD designs to Concept-HDL designs.

Conventions

This manual uses the following typographical and online document conventions. An example illustrates each typographical 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 loc2* `...` *locn;*

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.

Chapter 1

Getting Started

Cadence Concept-HDL is a graphical user interface (GUI) that presents a complete mixed-level design environment. Concept-HDL supports both behavioral and structural design descriptions in text and graphics and incorporates block editing functions for quick architectural design. It runs on Windows NT and on UNIX workstations.

Concept-HDL is based on the older SCALD architecture. Where SCALD was a proprietary format, Concept-HDL directly outputs the industry standards for Verilog-HDL and VHDL when you save your design. This eliminates the need for a secondary expansion step for generating the netlist for simulation.

This chapter describes getting started with Concept-HDL and contains the following sections.

- “Understanding Concept-HDL”
- “Starting Concept-HDL”

Understanding Concept-HDL

Concept-HDL operates under the Cadence Project Manager to provide a wide range of design functions. Using Concept-HDL you can quickly and efficiently instantiate components, create and name nets, and see your design elements. The following figure shows a sample design in Concept-HDL.

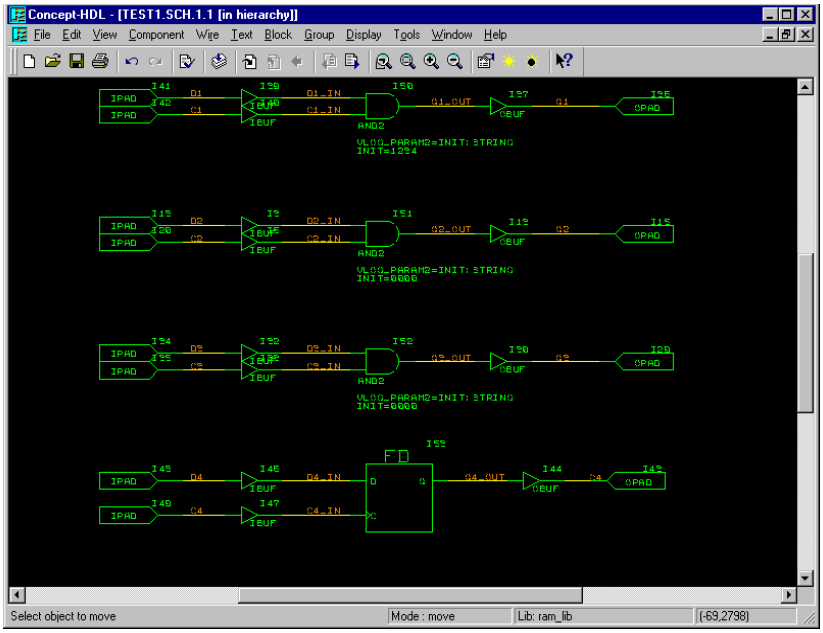


Figure 1-1 Concept-HDL Design Entry Window

Concept-HDL offers the following major features.

- Top-down (hierarchical) design allowing you to quickly draw blocks and connect wires between blocks.
- Cross-view generation of blocks from HDL descriptions or automatic generation of HDL text from high-level diagrams.
- Cross-probing between Concept-HDL and other tools.

Understanding Library.Cell:View

Concept-HDL's library structure is organized according to Library.Cell:View (L.C:V).

A library is a collection of related cells that describe components of a single design (a design library) or common components used in many designs (a reference library).

You reference each library by a logical name and each library has a unique physical directory associated with it. You define library names and map them to physical directories in the cds.lib file.

A cell is an object with a unique name stored in a library. Each Verilog module, macromodule, or UDP, or each VHDL entity, architecture, package, package body, or configuration is a unique cell.

Each cell within a library is a separate file system directory. Every unique element of a design is its own cell and, therefore, has its own cell directory.

You use views to delineate between representations, such as schematic, VHDL, and Verilog, abstraction levels such as behavior, RTL, post-synthesis, and status.

Each view within a cell is a separate file system directory that contains all of the files pertaining to a particular representation of a given design element. For example, one view directory might contain the RTL representation of a particular Verilog module, while the behavioral representation is stored in another view directory.

Following the Xilinx/Concept-HDL Design Flow

Concept HDL supports top-down, mixed-level, and bottom-up FPGA design flows. You use Xilinx Design Manager in the place and route part of the flow.

In the bottom up design methodology, you enter the design using Xilinx provided primitives and macros.

In the top-down and mixed-level design methodology, the logic to be implemented is entered as a block in Concept-HDL schematic editor along with other design units. This block can either be described in HDL or entered as a schematic.

Concept-HDL works with Synplicity’s Synplify for design synthesis. Concept-HDL writes Verilog and VHDL directly from a block diagram schematic. Synplify then synthesizes the structural netlist of the entire design.

You can specify synthesis constraints and attributes on component instances in Concept-HDL. When you save the design, HDL Direct creates a netlist and a synthesis constraints file. This provides Concept-HDL with the ability to pass constraints to the synthesis engine to control the way you implement the design in the physical device.

A mixed-level design consists of macros defined in Xilinx primitives and macros described using Verilog or VHDL. Concept-HDL can also treat HDL macros as black boxes for designs where the FPGA layout or synthesis is complete.

The following figure shows the Xilinx/Concept-HDL design flow.

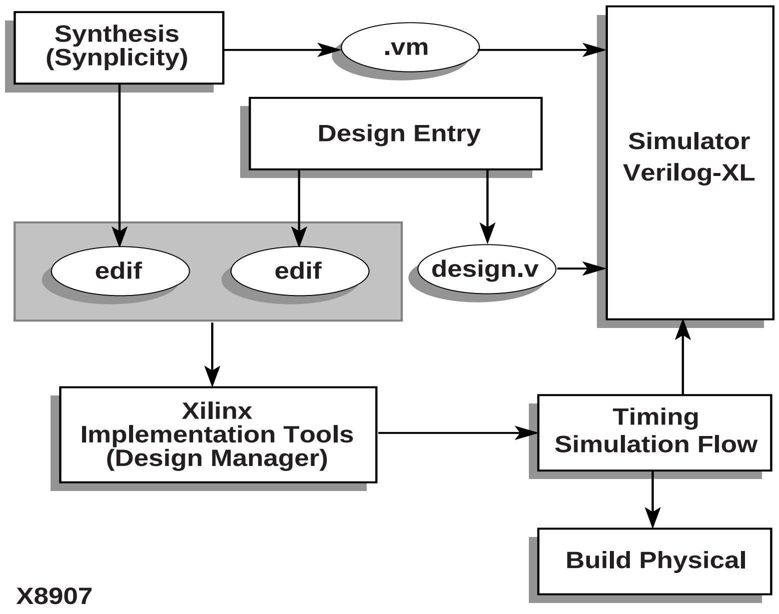


Figure 1-2 Xilinx/Concept-HDL Design Flow

System Requirements

The Xilinx/Concept-HDL design flow runs on the Sun Solaris SPARC5 with the following minimum requirements.

- 128 MB of disk space
- 64 MB of RAM
- 70 MHz
- 150 MB of swap space

The Xilinx/Concept-HDL design flow also runs on Windows NT 4.0, and requires the following minimum hardware requirements.

- Pentium P5 processor—166 MHz (minimum)

- 128 MB of disk space
- 64 MB of RAM
- 150 MB of swap space

Starting Concept-HDL

Concept-HDL runs under the Cadence Project Manager. The main Project Manager window appears in the following figure.

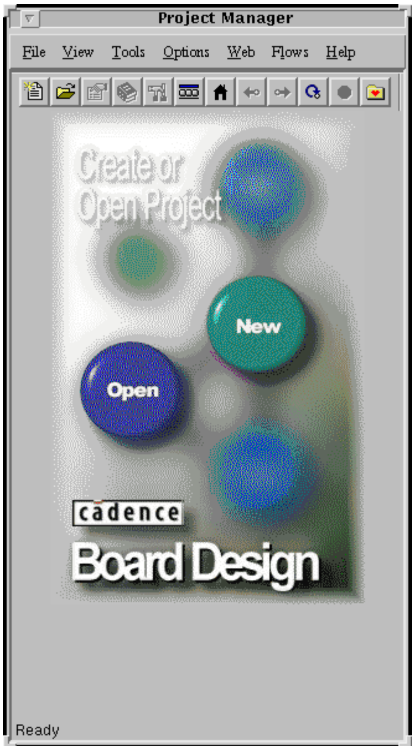


Figure 1-3 Project Manager Open Project Window

Starting A New Design

You can start the design process in Concept-HDL by using the Board Design flow or the Programmable IC flow in Project Manager.

Use the following steps to start a new design in Project Manager using the Board Design flow.

1. Select the New button. A New Project Wizard dialog box appears prompting you for the name and location of your new project.

   Enter this information. You can get an informational dialog box describing the location of errors such as missing libraries.

2. Follow the prompts in the New Project Wizard to create a new project.

Use the following steps to start a new design in Project Manager using the Programmable IC flow.

1. Create a project using the steps outlined previously.
2. Select **`Flows`**→**`Programmable IC`**.

Opening an Existing Design

To open an existing design in Project Manager, select the Open button. Select the design you want to open from the dialog box that appears.

After starting a new design or opening an existing design, the window shown in the following figure appears.This is the default Board Design window.

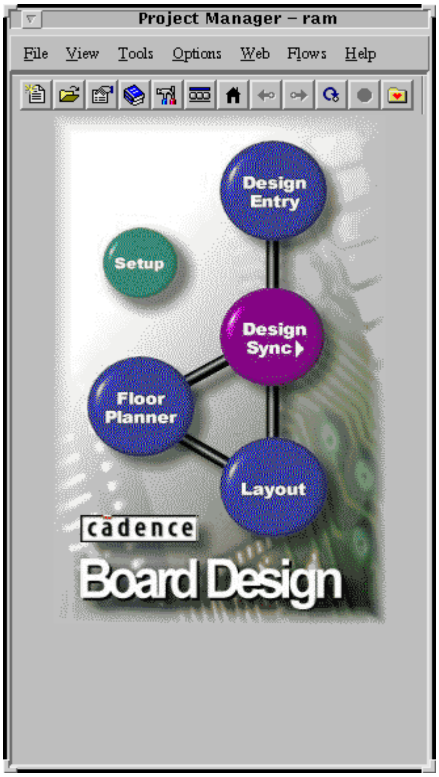


Figure 1-4 Board Design Flow Window

Select **`Flows`**→**`Programmable IC`** to launch the Programmable IC Board Design window, shown in the following figure.

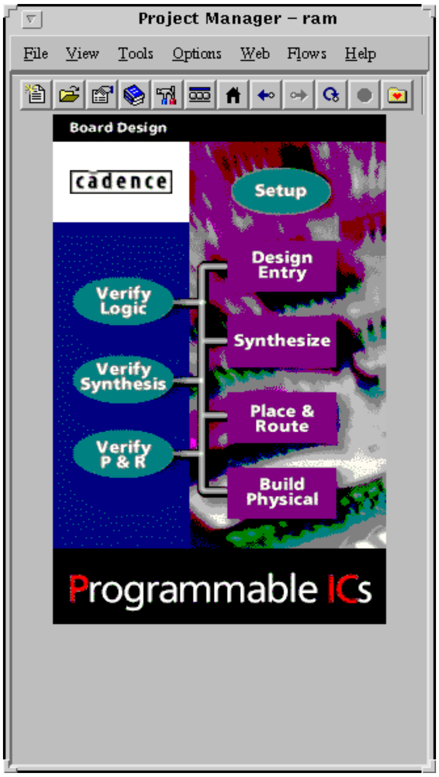


Figure 1-5 Programmable IC Board Design Window

You can perform the following design steps from the Project Manager Board Design Window.

- Setup

  Opens the Setup dialog boxes, allowing you to add and remove libraries
- Design Entry

  Launches Concept-HDL's schematic editor
- Verify Logic

  Lets you perform pre-synthesis functional simulation
- Synthesize

  Launches Synplify

- Verify Synthesis

  Allows you to check the post-synthesis Verilog netlist

- Place & Route

  Launches Xilinx Design Manager

- Verify P&R

  Allows you to check the P&R Verilog netlist and an SDF file

- Build Physical

  Allows you to create library components or use standard library components

The “Using Setup” chapter provides information for using the Setup dialog boxes. The “Using Concept-HDL with Xilinx Designs” chapter describes how to use Concept-HDL to prepare designs for Design Manager.

Using Setup

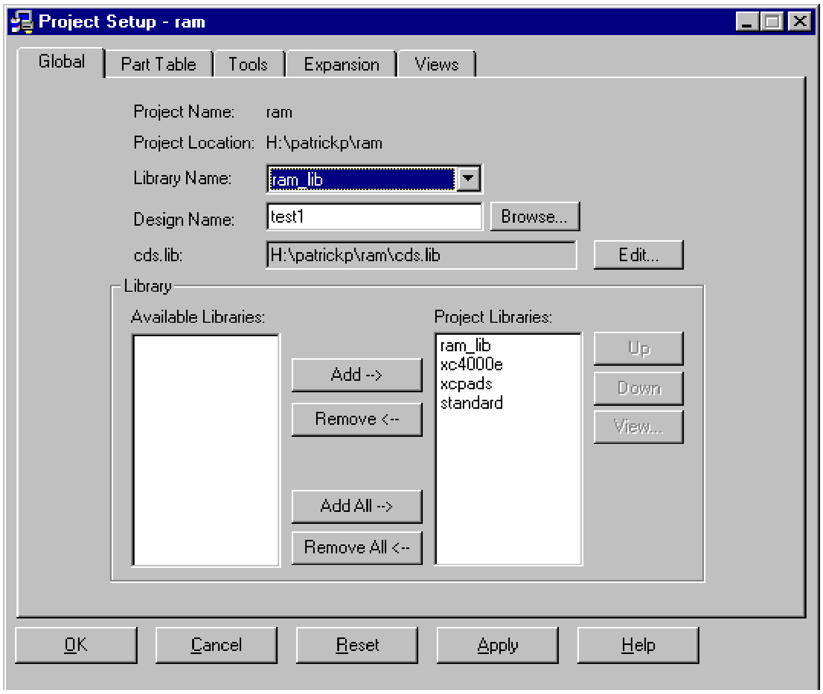
Use the Setup feature of Concept-HDL for selecting libraries, setting up design configurations, and specifying options for other tools.

This chapter describes the steps you take when using the Setup feature when creating Xilinx designs. This chapter contains the following sections.

- “Opening Setup”
- “Modifying the cds.lib File”
- “Working with Xilinx Libraries”
- “Selecting Xilinx as the PIC Vendor”
- “Using Setup for Synthesis”

Opening Setup

You open Setup from the Pic Design window by clicking the Setup oval. The Project Setup window appears, as shown in the following figure.



Project Name: ram

Project Location: H:\patrickp\ram

Library Name: ram_lib

Design Name: test1

cds.lib: H:\patrickp\ram\cds.lib

Available Libraries:

Project Libraries:
ram_lib
xc4000e
xcpads
standard

Figure 2-1 Project Setup Window

You can change a wide range of Concept-HDL settings and tools access in this window. Make sure you select the appropriate Xilinx project libraries by adding them to the Project Libraries list.

You can add or remove libraries at any time in the project.

Modifying the cds.lib File

This section describes how to modify the cds.lib file after you add or remove libraries.

1. Open the project.
2. Select **`Tools`**→**`Setup`**. The Project Setup dialog box appears.
3. Select the Global tab.
4. Click the Edit button next to the cds.lib box.
5. Edit the cds.lib file.

Add libraries to the cds.lib file directly by specifying their logical names and their physical locations, or you can add files that contain a list of libraries and their locations.

6. Select **File**→**Save**.
7. Select **File**→**Exit**.
8. Click Yes in the confirmation window to update the library list.
9. Click Apply in the Project Setup window to save your changes, or OK to save your changes and exit Project Setup.

When using cds.lib with Xilinx, include a reference to the following libraries.

```
DEFINE <library> worklib

INCLUDE $CDS_INST_DIR/share/cdssetup/cds.lib

DEFINE <device_family> $XILINX/cadence/data/<device_family>

DEFINE xcpads  $XILINX/cadence/data/xcpads
```

Make modifications in the physical paths to point to the appropriate locations.

Working with Xilinx Libraries

Use the following instructions to add or remove libraries you can use with Concept-HDL on existing projects.

1. Open the project.
2. Select **Tools**→**Setup**. The Project Setup dialog box appears.
3. Select the Global tab.

   You can view the contents of a library by selecting the library and clicking View. A window displaying the contents of the library appears. You cannot make any changes in this window.
4. Modify the Project Libraries list under Library.

   Add one library by selecting the library in the Available Libraries list and clicking Add. You can add all the libraries in the Available Libraries list by selecting Add All.

   Remove one library by selecting the library in the Project Libraries list and clicking Remove. The Remove All button removes all the libraries in the Project Libraries list.

5. Choose the search order for the project libraries.

   The order of the libraries in the Project Libraries list determines their search order. To move a library one level up, select the library and then click Up. To move a library one level down, select the library and then click Down.

6. Clicking Apply saves your changes. Selecting OK saves your changes and exits Project Setup.

Selecting Xilinx as the PIC Vendor

To specify the Xilinx flow, use the following instructions.

1. Select the Tools tab and Click on PIC Setup in the Tools tab. Select Xilinx as the PIC vendor.

   A vendor options form appears with the Xilinx tab selected. Select the appropriate device family. The form also shows the default values of the property format file, package file, and pin file.

2. Enter the package file.

   `$XILINX/cadence/data/<device_family>.pkg`

3. Enter the property format file (pff).

   `$XILINX/cadence/data/xilinx.pff`

4. Enter the pin file.

   `$XILINX/cadence/data/xilinx.pga.pin`

Select the radio button options for User And Programming Pins and Create Custom Library Component.

Using Setup for Synthesis

Set Setup Options for synthesis by clicking on the synthesis tab. Use synthesis setup to create a Synplify project file with options for the following.

- Synplify Home

  Specifies the location of Synplify installation

- Synthesis View

Used as run directory for Synplify (location of the post-synthesis Verilog file)

- Configuration

  Specifies the configuration used by Hierarchy Editor to pick up various Verilog files that constitute the design. You create a project file in the synthesis view using this set of files along with other options.

The other options available under Configuration include the following.

- Input HDL

  Specifies the HDL used as the input to Synplify

- Output HDL

  Specifies the name given by Synplify when generating the post-synthesis netlist

- Output Format

  Specifies the output format of the netlist given to Design Manager as input.

Chapter 3

Using Concept-HDL with Xilinx Designs

This chapter describes how to use Concept-HDL to create your Xilinx designs. This chapter includes the following sections.

- “Understanding Library and Database Structure”
- “Using Design Entry”
- “Verifying Logic”
- “Synthesizing Your Design”
- “Verifying Synthesis”
- “Using Place and Route”
- “Verifying Place and Route”
- “Building the Physical Design”

For detailed information about to create designs and enter design elements in Design Entry or to use other parts of the Cadence Project Manager, refer to the Cadence-HDL documentation.

Understanding Library and Database Structure

Design Entry places data according to Library.Cell:View. So, in a design with a schematic macro named *foo*, the data hierarchy exists as illustrated in the following figure.

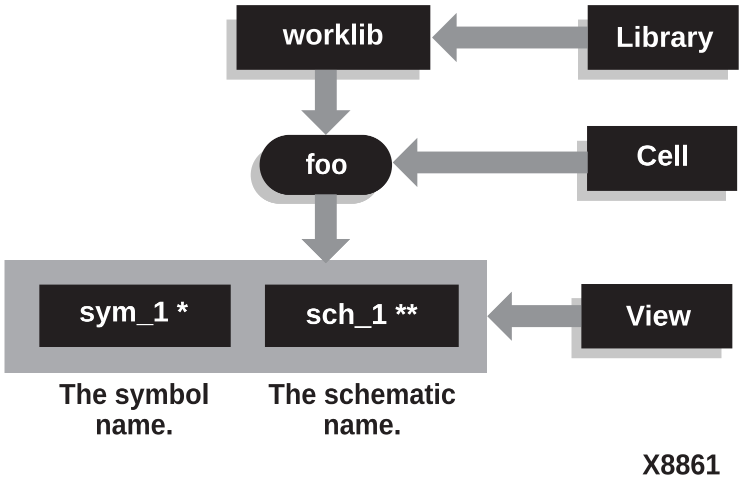


Figure 3-1 Schematic Macro Library.Cell:View Approach

Consider an example of a design with a schematic top-level named foo, shown in the following figure.

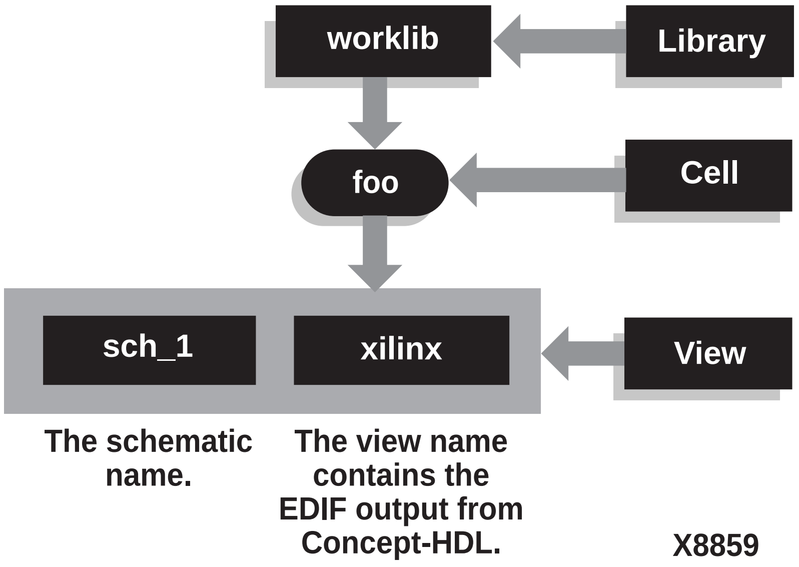


Figure 3-2 Schematic Top-Level Library.Cell:View

Using Design Entry

For details about how to use the Concept-HDL Design Entry, refer to Cadence documentation.

When using Design Entry in the Xilinx/Concept-HDL flow, add synthesis constraints to the HDL or provide these constraints to Synplify with the Synplify Constraints editor before passing schematic properties to the P&R tool.

Verifying Logic

You conduct functional simulation to verify design logic after completing design entry.

To verify the logic, use the default configuration for functional simulation, cfg_verilog. VHDL is not supported.

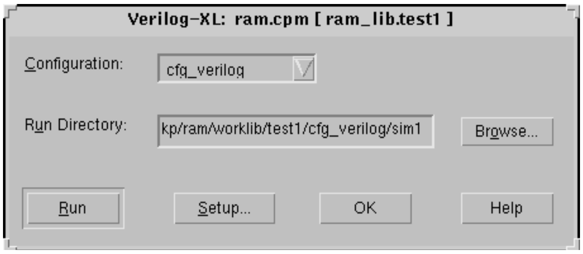
Click on Setup to invoke the project setup tool. Choose setup options for simulation by clicking on the simulation tab. In the Addt'l Cmd Line Options field, add the following.

```
+define+XILINX
```

If your design contains LogiBLOX modules, add the SIMPRIM libraries to your simulation environment using the following steps.

1. Select Verify Logic from the Programmable ICs window.

   A Verilog-XL dialog box appears where you can specify configuration and the run directory, as shown in the following figure.



Verilog-XL: ram.cpm [ram_lib.test1]

Configuration: cfg_verilog

Run Directory: kp/ram/worklib/test1/cfg_verilog/sim1

Figure 3-3 Verilog-XL Verify Logic Dialog Box

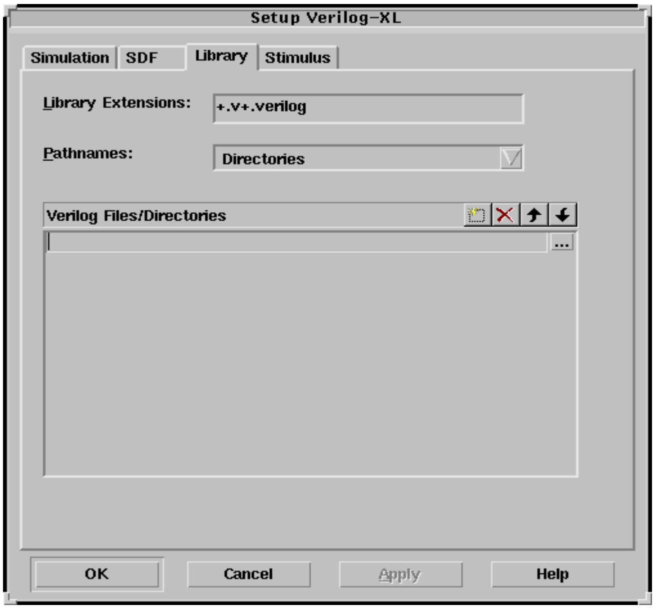
2. Select the Setup pushbutton in the Verilog-XL dialog box.

   The Setup Verilog-XL window appears.

3. Select the Library tab in the Setup Verilog-XL window, then enter the path to the SIMPRIM libraries in the Verilog Files / Directory-field.

   Alternatively, click on the small folder icon on the Verilog Files / Directory data entry field.

   A browse button (ellipses) appears, as shown in the following figure.



Library Extensions: +.v+.verilog

Pathnames: Directories

Verilog Files/Directories

Figure 3-4   Setup Verilog-XL Window

4. Use the ellipses icon to toggle to the location of the SIMPRIM libraries, then select the OK pushbutton.

You can find additional information in the “Conducting Simulation” chapter.

Synthesizing Your Design

If your design is purely schematic, you can skip synthesis and move on to Place and Route. If your design contains HDL blocks, you can synthesize after conducting functional simulation

You synthesize Xilinx designs in Concept-HDL with Synplify. You start synthesis by clicking on the Synthesize button in the Programmable IC window.

Creating the Synthesis View

The first step in design synthesis with Concept-HDL is to create a synthesis view. Use the following steps to create a synthesis view.

1. Click on Synthesize. Select the various options provided in the setup.

   A project file, syn.prj, is created in the synth view. By default, this project file contains all source files, constraints files, top level module, target technology, result format and the result file location.

2. Click on Run. This launches Synplify.
3. Within the Synplify Project window, click on **`Target`**$\rightarrow$**`Set Device Options`**. Set the appropriate options for your target family.
4. Click on Run. Synplify creates the implementation netlist in the run directory.

Understanding the Synthesis Data Flow

The library structure is organized according to a Library.Cell:View approach. Consider an example of a design with a HDL macro named foo, shown in the following figure.

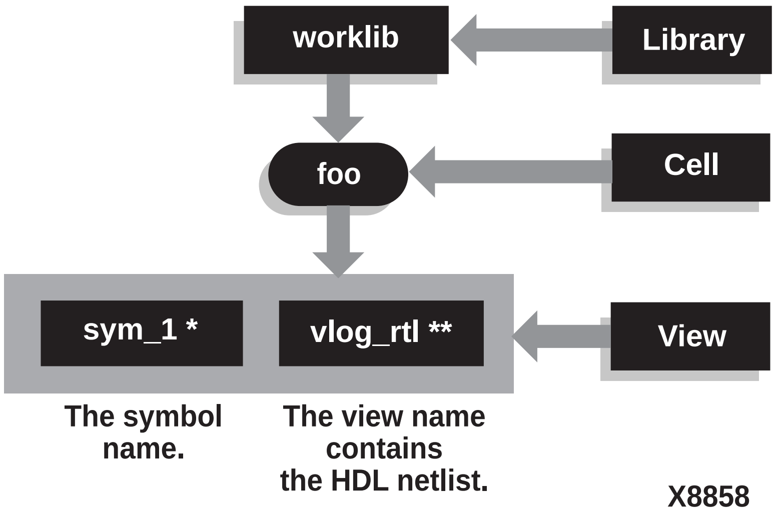


Figure 3-5 Macro Library.Cell:View Approach

Additionally, consider an example of a design with a HDL top-level named foo, shown in the following figure.

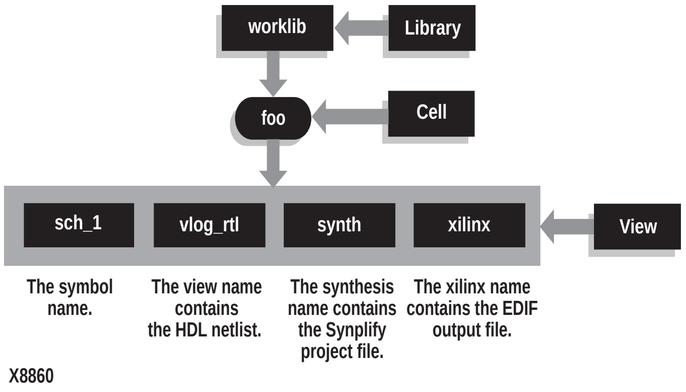

worklib
Library
foo
Cell
sch_1
vlog_rtl
synth
xilinx
View
The symbol name.
The view name contains the HDL netlist.
The synthesis name contains the Synplify project file.
The xilinx name contains the EDIF output file.
X8860

Figure 3-6 HDL Top-Level Design with Library.Cell:View

Verifying Synthesis

You can perform post-synthesis simulation to verify synthesis. The default configuration used for functional simulation is cfg_synth.

Synplify creates a post-synthesis verilog netlist with a .vm extension. This netlist is moved into the synthesis view. VHDL is not supported.

See the “Conducting Simulation” chapter for more information.

Using Place and Route

Xilinx-specific options for the P&R tool include the target family, interface view (P&R tool run directory), property format file, package file and pin file.

When selecting a part, ensure that you set the simulation template to Concept Verilog XL. Unselect the following Include.

```
‘uselib Directive in Verilog file
```

Set the simulation netlist name to the following.

`<`*`design`*`>_routed`

You do this in Xilinx Design Manager by selecting the templates from the Options pull-down menu.

Earlier versions of Design Manager ran on the EDIF netlist created by Synplify for top-down and mixed-level flows, and the SIR2edf netlist for bottom-up flows. However, you can now create netlists for each schematic block and synthesize individual HDL blocks. Concept-HDL partitions the design into pure schematic and HDL blocks. SIR2edf creates netlists using the schematic and HDL blocks pass on to Synplify. This provides you the ability to use a previously generated EDIF view for blocks, rather than requiring a Verilog, VHDL, or schematic view.

Verifying Place and Route

You specify the appropriate P&R options according to the device type. Cadence Project Manager renames the timing verification files in the interface view to the following

- **<*design*>_routed.v**
- **<*design*>_routed.tv**

- **<*design*>_routed.pin**
- **<*design*>_routed.sdf**

You use these files in timing verification and building physical views for the design.

Refer to the Design Manager documentation for specific instructions about using Design Manager.

Building the Physical Design

You can use either use a standard library component or create a custom component for the design implemented.

If you choose the Create Custom Library component option, Concept-HDL creates a separate cell with the following views.

- Symbol
- Chips
- Entity

The PIC design retains the vlog_routed view created in the Verify P&R step. The vlog_routed view contains an implemented Verilog file and SDF file. In addition to creating this custom library component, the Build Physical Design process creates a pic_1 schematic view containing master.tag and page1.csb, instantiating this custom library component.

If you use a standard library component the pic_1 schematic view instantiates this component.

Open the schematic view pic_1 in Concept-HDL and save it to obtain the netlist.

You perform the build physical step after implementing the design creates a schematic view pic_1. This schematic instantiates the custom component created or the standard component selected at the time of building the physical views. While the sch_1 schematic view contains the logical design, implementation occurs in the physical device instantiated in the pic_1 view. Choose the view pic_1 for packaging the implemented design.

Chapter 4

Conducting Simulation

This chapter describes how to simulate Xilinx designs entered in Concept-HDL. This chapter includes the following sections.

- “Supported Simulators”
- “Using Global Signals in the Xilinx Design Flow”
- “Simulating a Verilog Design”
- “Setting Verilog Global Set/Reset”

Supported Simulators

The Concept-HDL schematic editor and simulation interface provide the capability to create a netlist of your design and take it into a simulation environment. You can set up and launch either Verilog-XL or Leapfrog as your primary simulator. However, Xilinx supports Verilog-XL; the Leapfrog flow is solely supported by Cadence.

Using Global Signals in the Xilinx Design Flow

You use Xilinx global signals for both functional simulation and timing simulation of the designs having Xilinx primitives and macros. Depending upon the family of Xilinx devices used, you declare appropriate global signals in the global module.

The simulation interface provided in Concept HDL runs **edbconfig** to create the global signals module. The global module (glbl) contains the following lines with the term $XILINX replaced with its value.

```
‘ifdef XILINX

‘include "$XILINX\cadence\data\global.v"

‘endif
```

The included file $XILINX\cadence\data\global.v is the list of global signals in all the families, not a complete Verilog module.

To include the global signals in the global module, you must specify the following additional command line option.

```
+define+XILINX
```

When you use global signals from the Xilinx installation do not name any of the symbols, such as the signal connected to the startup symbol **`<global_signal>/g'`**. Using this name creates multiple declarations of this signal in the glbl module (one created by edbconfig and the other included from the Xilinx installation).

Simulating a Verilog Design

Before you simulate a Verilog design in Concept-HDL, make sure you set up Project Manager with Verilog-XL as the primary simulator (select Verilog-XL from the Tools tab in the Setup window). Ensure that you saved the schematic you created in Design Entry with the proper libraries. Saving the schematic in Design Entry creates a netlist.

You can optionally override default bindings by launching Hierarchy Editor, modifying the various cells and instances, and saving the configuration.

Use the following steps to start simulation.

1. Start the Verilog-XL interface by selecting **`Tools`→`Simulate`** from the Programmable IC Window pulldown menu.

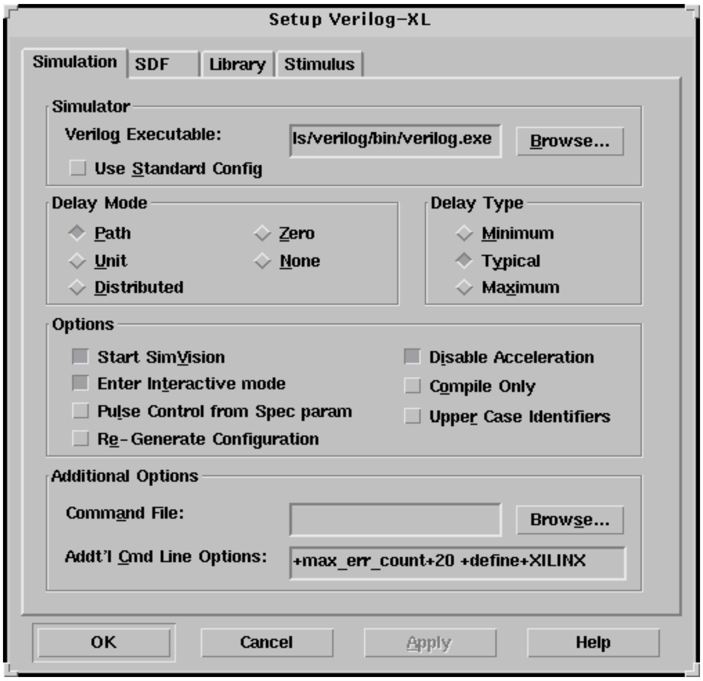
   This launches the Verilog-XL interface.

2. Select the design configuration from the Verilog-XL interface. The default configuration for Verilog simulation is cfg_verilog.

3. Select the Setup button in the Verilog-XL interface.

   The Setup Verilog window appears, as shown in the following figure.

From this window you can set Verilog-XL command line options, chose to include the Verilog model libraries, specify parameters for SDF back-annotation, and generate and include a Verilog testbench.



Setup Verilog-XL

Simulation | SDF | Library | Stimulus

Simulator

Verilog Executable: ls/verilog/bin/verilog.exe Browse...

Use Standard Config

Delay Mode: [✓ Path] Zero Unit None Distributed

Delay Type: Minimum [✓ Typical] Maximum

Options

[✓ Start SimVision]
[✓ Enter Interactive mode]
Pulse Control from Spec param
Re-Generate Configuration
[✓ Disable Acceleration]
Compile Only
Upper Case Identifiers

Additional Options

Command File: Browse...

Addt'l Cmd Line Options: +max_err_count+20 +define+XILINX

OK Cancel Apply Help

Figure 4-1 Setup Verilog-XL Window

Setting Verilog Global Set/Reset

For Verilog simulation, ensure all behaviorally described (inferred) and instantiated registers have a common signal which asynchronously sets or resets the register. You must toggle the global set/reset signal (GSR) for XC4000E/L/X, Spartan/XL, and Virtex designs, or global reset (GR) for XC5200, XC3000A/L, or XC3100A/L designs. Toggling the global set/reset emulates the Power-On-Reset of the FPGA. If you do not do toggle the global set/reset signal, the flip-flops and latches in your simulation enter an unknown state.

The GSR signal in XC4000E/L/X, Spartan/XL, and Virtex devices, and the GR signal in XC5200 devices are active High. The GR signal in XC3000A/L and XC3100A/L devices are active Low.

Your implemented design contains the global set/reset net even if the design does not instantiate the STARTUP block. STARTUP gives you the option to control the global reset net from an external pin.

The Programmable Logic Data Book describes how to set the global set/reset pulse width so that it reflects the actual amount of time for the chip to go through the reset process after receiving power. The duration of the pulse is specified as T_{POR} (Power-On-Reset).

In the Xilinx software, use the Verilog UniSims library only in RTL simulations of your designs. Simulation at other points in the flow use the Verilog SimPrims libraries.

Defining GSR in a Test Bench

For pre-NGDBuild UniSims functional simulation, you must set the value of the appropriate Verilog global signals (glbl.GSR or glbl.GR) to the name of the GSR or GR net, qualified by the appropriate scope identifiers.

The scope identifiers are a combination of the test module scope and the design instance scope. You need the scope qualifiers because the Verilog UniSims simulation models interpret the glbl.GSR and glbl.GR wires to emulate a global reset signal using the scope information.

In your testfixture, enter the following commands to toggle the global signal, GSR or set the GSR for the 4000 series, Virtex, and Spartan designs.

```
reg GSR

assign glbl.GSR = GSR

GSR = 1’b1;

GSR = 1’b0;
```

To set the GR for the 3000 series, enter the following commands.

```
reg GR

assign glbl.GR = GR

#0 GR = 1’b0;

#100 GR = 1’b1;
```

To set the GR for the 5200 series, enter the following commands.

```
reg GR
assign glbl.GR = GR
#0 GR = 1'b1;
#100 GR = 1'b0;
```

To set the PRLD for the 9500 series, enter the following commands.

```
reg PRLD
assign glbl.PRLD = PRLD
#0 PRLD = 1'b1;
#100 PRLD = 1'b0;
```

For post-NGDBuild and post-route timing simulation, the testfixture template (.tv file) produced by running NGD2VER with the –tf option contains most of the code previously described for defining and toggling GSR or GR.

Use the following steps to define the global set/reset signals in a testfixture for your design.

Note: In the following steps, *testfixture_name* refers to the testfixture module name and *instance_name* refers to the designated instance name for the instantiated design netlist within the test bench.

1. For Verilog simulation without a STARTUP block in design, name the global set/reset net to *testfixture_name.instance_name.*GSR or *testfixture_name.instance_name.*GR (Verilog is case-sensitive). Declare the signal as a Verilog reg data-type.
2. For Verilog simulation with a STARTUP block in the design, the GSR/GR pin connects to an external input port, and glbl.GSR/glbl.GR is defined within the STARTUP block to make the connection between the user logic and the global GSR/GR net embedded in the Unified models. For post-NGDBuild functional simulation, post-Map timing simulation, and post-route timing simulation, glbl.GSR/glbl.GR is defined in the Verilog netlist created by NGD2VER.

   At the beginning of the simulation, you toggle the port or signal in your design that controls global set/reset, usually an external input port in the Verilog netlist. This signal can also be a wire if logic internal to your design controls global reset.

Give the name *test* to the main module in the testfixture file. This name is consistent with the name of the testfixture module written later in the design flow by NGD2VER during post-NGDBuild, post-MAP, or post-route simulation. If you maintain this naming consistency, you can use the same testfixture file for simulation at all stages of the design flow with minimal modification.

Chapter 5

Using Genview

This chapter describes how to use Genview with Concept-HDL when creating Xilinx designs. This chapter includes the following sections.

- “Creating Symbols with Genview”
- “Generating a Destination View”

Creating Symbols with Genview

Concept-HDL provides the Genview utility to create a symbol for user-written HDL designs. Genview lets you optionally place this HDL file in the HDL-centric library structure.

You can instantiate the symbols created by Genview just as you can with any other component. Genview supports creating views for a cell by specifying a HDL (Verilog or VHDL) file. Additionally, you can import the HDL and create a view corresponding to it for the cell.

Generating a Destination View

To generate a destination view from a HDL file, use the following instructions.

1. Open the Genview user interface from **`Tools`**→**`Generate View`**.
2. Select the File button to enable the file field.
3. Enter the path of the source HDL file in the File field. You can optionally browse to select the path of the source HDL file.
4. Select the Verilog or VHDL from the combo box depending upon the selection of the source HDL file.

If you want to import a Verilog or a VHDL view, check the ON button in the Import as View field and enter the view name where you want to import the source HDL file.

5. In the destination area, select the library from the combo box. This shows where the destination view generates.
6. Enter the destination view name or browse from the available path selections.
7. Select the destination view type from the combo box.
8. Click OK. Genview creates the destination view and returns with the status.

To generate a destination view from a View, use the following instructions.

1. Open the Genview user interface from **`Tools`**→**`Generate View`**.

   The Lib.cell:view radio button is the default selection.
2. You can specify the source view in the library.cell:view field, or use the Browse button to select a source view.
3. Specify the destination by either browsing to select the view or enter any view name.
4. Select the destination view type from the combo box. The available destination views are Schematic, Symbol, Verilog, and VHDL.
5. Click OK. Genview creates the destination view and returns with the status.

Genview does not allow you to generate a Verilog or VHDL view from the schematic view. Use HDL Direct to do this by writing the schematic in Concept-HDL. Also, Genview does not allow you to generate the same destination view from the same source view.

Chapter 6

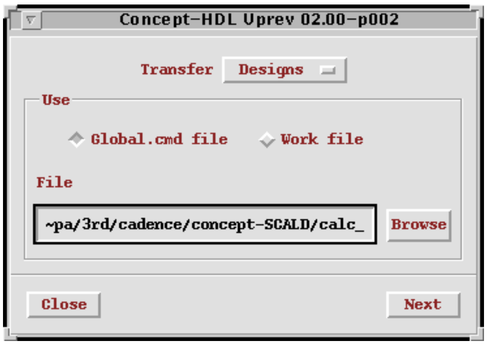
Upgrading to Concept-HDL

This chapter describes how to upgrade existing Concept-SCALD libraries and designs to Concept-HDL. This chapter includes the following sections.

- "Starting chdl_uprev"
- "Upgrading Design Parts"
- "Completing the Upgrade"

Starting chdl_uprev

Cadence provides a utility, chdl_uprev, to migrate existing Concept-SCALD designs to the Concept-HDL database. The following figure shows the primary chdl_uprev interface.



~pa/3rd/cadence/concept-SCALD/calc_

Figure 6-1 Concept-HDL Uprev Dialog Box

From this dialog box you can choose to transfer Concept-SCALD libraries or designs to Concept-HDL. You can enter the path to the libraries and designs or browse to choose the proper path. After you select a library or design, and select the path to it, press the Next button.

Upgrading Design Parts

After you select a design, and select the path to it, pressing the Next button takes you to the Concept-HDL Uprev Selection dialog box, shown in the following figure.

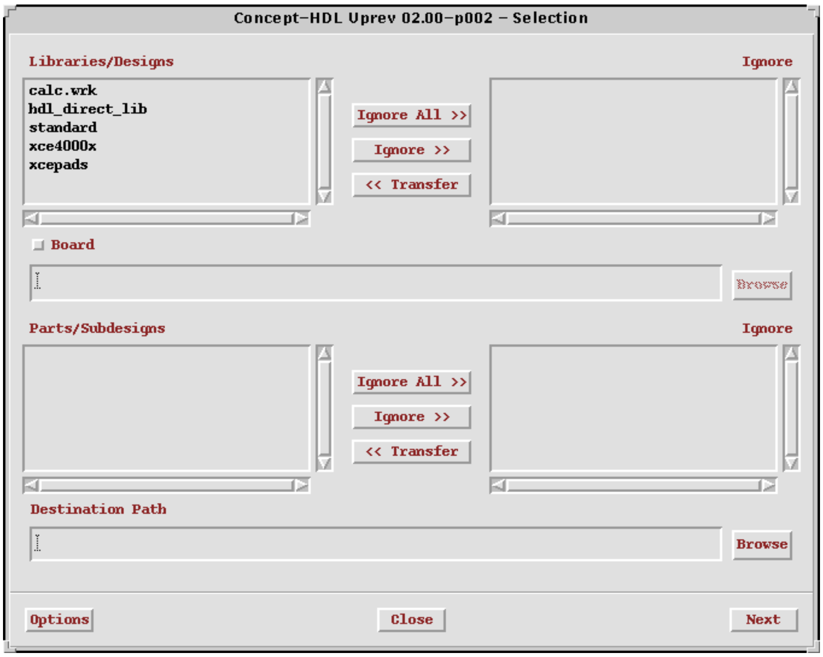
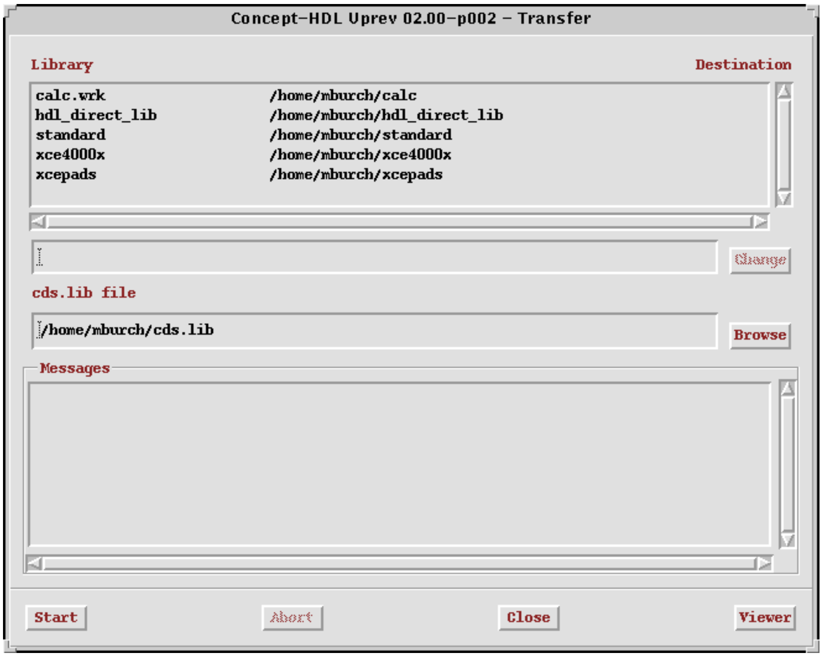


Figure 6-2 Concept-HDL Uprev Selection Dialog Box

In this dialog box you can select the parts you want to upgrade. You specify the new design directory in the Destination Path field. You press the Next button after selecting the parts you want to upgrade.

Completing the Upgrade

Pressing the Next button in the Concept-HDL Uprev Selection dialog box launches the dialog box shown in the following figure.



Library	Destination
calc.wrk	/home/mburch/calc
hdl_direct_lib	/home/mburch/hdl_direct_lib
standard	/home/mburch/standard
xce4000x	/home/mburch/xce4000x
xcepads	/home/mburch/xcepads

/home/mburch/cds.lib

Figure 6-3 Concept-HDL Uprev Transfer Dialog Box

This dialog box displays the location of the cds.lib file the chdl_uprev process creates; you can change the location of the new cds.lib file prior to generation. When you select the Viewer button, the upgrade begins. Error and warning messages display in the Messages field.