XILINX®

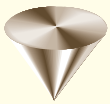
ONLINE

XEPLD SCHEMATIC DESIGN GUIDE

FOR WINDOWS

TABLE OF CONTENTS

INDEX



GO TO OTHER BOOKS

XACT™ STEP

0401391

Contents

Chapter 3 EPLD Architecture and Design Trade-offs

Getting Started with Schematic Design

This chapter will help you quickly understand how to develop a schematic design using XEPLD. Brief schematic design examples are included, illustrating the device-independent schematic library and the automatic PAL conversion process.

An Overview of Schematic Design Methods

A schematic design defines the functionality of a logic circuit using one or more schematic files, each of which contains components whose functions are already defined (74xx TTL or similar functions) and components for which you define the function using behavioral modules. Figure 1-1 summarizes the design flow.

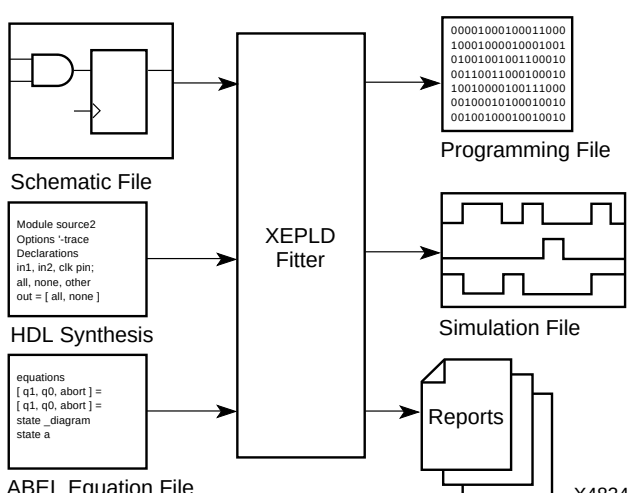


Figure 1-1 Basic Schematic Design Flow

The Viewlogic, OrCAD, Mentor, and Cadence software packages are supported directly by Xilinx for design entry and simulation. Other compatible interfaces are available from their manufacturers.

The following sections provide an overview of methods for creating schematics and behavioral modules for schematic designs.

Using the Unified Library

The Unified Library allows you to create a device-independent design to test how the design works in different devices, or to create a device-specific design to take full advantage of a device's unique architectural features.

The Unified Library contains all the component symbols for all the available device families. As illustrated in Figure 1-3, most of the symbols can be used in designs targeted for any Xilinx device, but some of the symbols are specific to one or more device families.

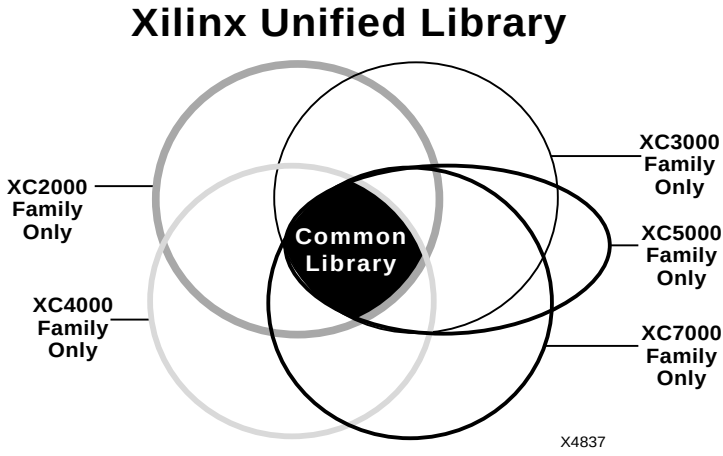


Figure 1-2 Device Families and the Unified Library

The common symbols are automatically mapped to the chosen target device. The same common symbol may be mapped differently to target devices with different architectures.

The "Device-Independent Design" chapter describes the library components and how to retarget an existing FPGA design to an EPLD

device.For more information about the Unified Library, refer to the *Libraries Guide* and *Libraries Supplement Guide*.

Behavioral Modules and PAL Conversion

You can include a behavioral module in a schematic design. You may want to include behavioral designs because:

- They are a more convenient method to describe state machines.
- They may already exist as PAL equations or HDL.

To create a custom component from a behavioral module, use the **`Symbol Generation Utility`** found in the XACT*step* icon group. This utiltity creates a custom symbol from behavioral design files. The symbol can then be placed in a schematic.

The original design file can be a new file created using XABEL The method of including an XABEL design in a schematic design is described in the chapter entitled "Using Behavioral Modules in Schematics."

Figure 1-3 shows the design flow for including a behavioral module.

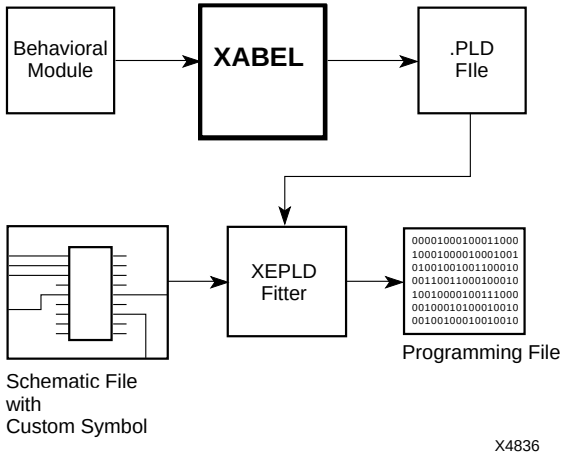


Figure 1-3 Schematic PAL Conversion Flow

Schematic Design Flow Example

This section runs through the entire schematic design process, from creating a design to programming and simulating the design. The following device-independent design, a 4-bit Johnson counter, is used as an example:

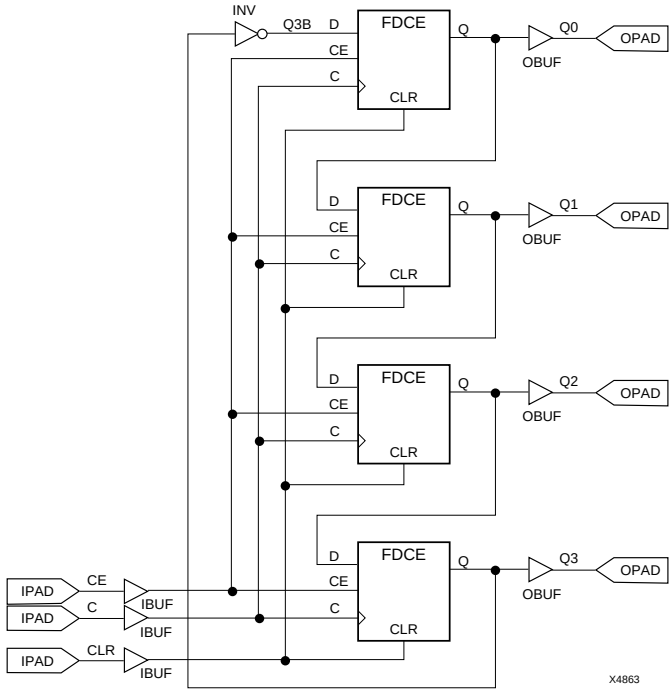


Figure 1-4 Example 4-Bit Johnson Counter Design

Simulation results for this design are shown in Figure 1-5.

This design contains no behavioral modules. For an example of a design that includes a behavioral module, see "Using Behavioral Modules in Schematics."

The steps are summarized for Viewlogic ProSeries and OrCAD.

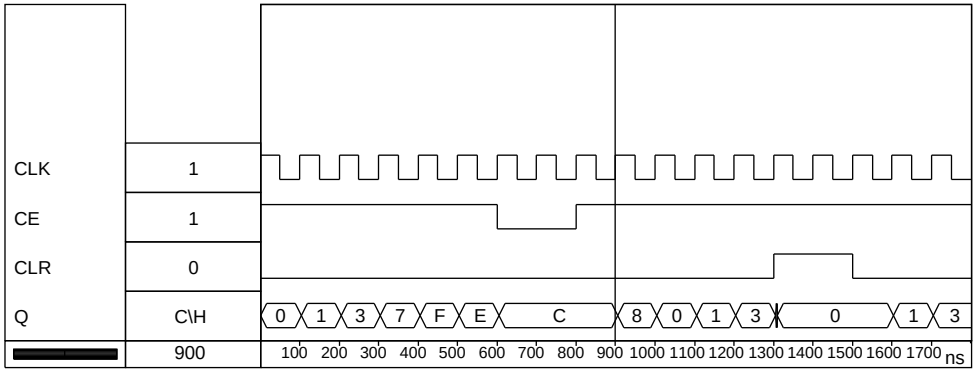


Figure 1-5 Example Viewlogic Functional Simulation Results

Viewlogic Procedure

Step 1 — Enter PROflow and Configure ProCapture

Create a directory for your design. If you are using PROflow, your Xilinx libraries will be configured for you automatically whenever you create a new project. Otherwise, use the **`Project Manager`** utility found in Design Entry.

If you use the Project Manager, the Xilinx library required for this example is installed under the path `unified\xc7000`. It is in megafile format. The required viewdraw.ini alias is `xc7000`. If you plan to simulate, you should also include the Viewlogic built-in library.

Enter PROflow by selecting the **`PROflow`** icon. From PROflow:

Select **`Design Entry`**, then select the **`Project Manager`** button**.**

To create a directory for your project, press the **`Create`** button and select a directory for your project.

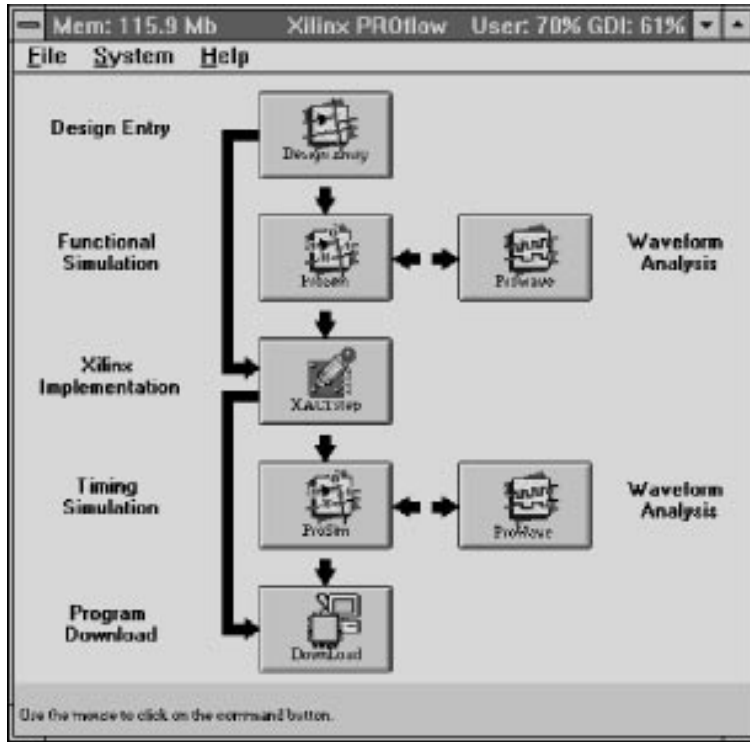


Figure 1-6 Xilinx PROflow

Step 2 — Select the Device Family

From **`Design Entry`**, click on **`Select Family`**.

Select XC7000, then select **`OK`**.

Under **`Design Type`**, select **`Schematic`**.

In the **`Design Name`** line, type **`JCOUNT.1`**

Select **`OK`**. PROcapture will be invoked and a design entry window will appear.

Step 3 — Draw the Design

Draw the design as shown in Figure 1-4. For more information about using PROcapture, see the *Viewlogic Interface Guide* and *Viewlogic Tutorials* manual.

If you do not wish to draw the design, you can copy the jcount.1 schematic file in the `\xact\example\vwlogic\jcount\sch` directory.

It is important that you label the nets between the IPADs and IBUFs and between the OPADs and OBUFs, because these names will appear in reports and on simulation traces.

Save your design using the **File → Save As** command.

Step 4 — Perform Functional Simulation (Optional)

From Design Entry, link the design to the simulator.

Select **PROsim** from the **PROflow** menu.

Figure 1-7 ProSim Icon

Click on **Select Part**. A menu for selecting parts and packages appears. Select a part and a package from the lists and press **OK**.

Place a check in the **Command File** box and select **Browse**. Go to the JCOUNT directory. The file JCOUNT.CMD should appear in the files display. Select JCOUNT.CMD and press **OK**.

Make sure the **Execute Netlister** box is checked, then press **OK.**

The design is simulated using the following command file:

```
restart
vector Q Q[3:0]
wave jcount.wfm CLK CE CLR Q
clock c 1 0
step 50ns
h prld
```

```
h CE
l CLR
cycle
l prld
cycle 5
l CE
cycle 2
h CE
cycle 5
h CLR
cycle 2
l CLR
cycle 3
```

To view the waveform file, go to PROflow and select **PROwave.**

Step 5 — Fit the Design and Create a Programming File

In steps 1 and 2 you specified the XC7000 family. You now need a specific part number for which to fit the design. To select XC7300:

Select **Xilinx Implementation** from PROflow. Next, create a new project.

File → New Project

The **New Project** dialog box will appear. Click on the **Input Design** line and click on **Browse**. Select JCOUNT.1 from the list. Set the **Target Family** to XC7000. Select a directory to work in, then click on **Translate**. The program creates a new Xilinx project.

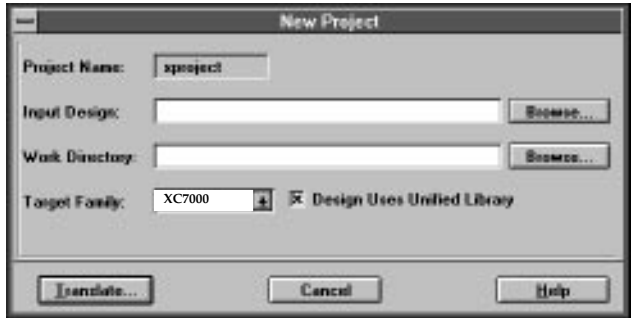


Figure 1-8 New Project Window

To translate the design to a specific device, select **Design → Translate**. Click on **Select Part**. Under **Family**, select **XC7300**. Select a specific **Device** and **Package** from the lists given.

Set the speed to **-5**. Select **OK**, then wait for the message that the translation has successfully completed. Select **OK** again. You should now be back in the Xilinx Design Manager.

Note: If you select **ALL** under **Device**, **Package**, and/or **Speed**, the program will automatically select an economical solution to meet the needs and constraints of your design. In general, it will select the smallest part (starting with the XC7318 pc44) and work its way up until it has the part that will satisfy the design. You may place **ALL** in any combination of these boxes (**ALL** is default for all three).

Select **Tools → Flow Engine**. Set **Stop After** to **Bitstream** and select **RUN**. The program optimizes the design, fits the design, creates timing file and programming files all in one step. The interface keeps you updated on the progress of processing.

Step 6 — Examine the Reports

Examine the reports to verify that the design was implemented as you expected. To examine reports from the Xilinx Design Manager select **Utilities → Report Browser,** or select the report browser icon. The following reports are most useful for schematic designs:

Figure 1-9 Report Browser

- Translation Report — The translation report informs you of how the translation went and lists any errors or problems that occured during translation.

- Fitting Report — This fitting report shows the allocation of Function Block resources.
- Resource Report — The resource report lists the device resources used by the design and the resources remaining.
- Pinout Report — The pinlist report shows how the external nets in your design were mapped to the device pins.
- Mapping Report — The mapping report tells you how the logic in the design was mapped to the device.
- Timing Report — Shows the calculated worst-case timing for the logic paths in your design.
- Timespec Report — The timespec report tells you if you have made any errors in assigning TSPEC attributes in your design and list the TSPECS that the fitter used.

Step 7 — Timing Simulation

From PROflow select the **PROsim** icon from the Timing Simulation block. Put a check in the **Command File** box and select **Browse**.

Select the JCOUNT.CMD file and press **OK**. Make sure **Execute Netlister** is selected, then press **OK**. The simulation runs and generates a report.

Note: Xilinx has a separate Timing Analyzer for further analysis of timing. See the "EPLD Architecture and Design Trade-offs" chapter for information on the Timing Analyzer, or go to the *XEPLD Reference Guide*.

OrCAD Procedure

Step 1 — Enter and Configure OrCAD

Enter OrCAD. If you have a Windows version of OrCAD, simply double-click the **OrCAD** icon. If you have a DOS version of OrCAD, exit Windows first, then type at the DOS prompt:

```
orcad
```

Note: Double-clicking on an **MS-DOS** icon does not exit you from Windows; it merely opens a DOS window. You must exit Windows completely using **File** → **Exit Windows**.

Double click on **Design Management Tools**. Click on **Create Design**.

In the prompt box that appears, type **jcount**↵, then click on **OK**.

The JCOUNT design appears, highlighted, in the list. Click on **Suspend to System**. Type the following at the DOS prompt:

```
xdraft 7
```

(wait for the command to complete)

```
exit
```

You are back in the Design Management Tools window. Click on **OK** to exit.

Step 2 — Enter Draft and Draw the Design

Double click on **Schematic Design Tools**. Double Click on **Draft**. A blank schematic window appears.

Draw the design as shown in Figure 1-4. For more information about using Draft, see the *OrCAD Interface/Tutorial Guide*.

If you do not wish to draw the design, you can copy the jcount.sch schematic file in the \xact\tutorial\orcad\jcount directory.

Note: It is important that you label the nets between the IPADs and IBUFs and between the OPADs and OBUFs, because these names will appear in reports and on simulation traces.

Step 3 — Add Simulation Information

To add trace information, follow these steps for each net that is between an IPAD and an IBUF or between an OBUF and an OPAD:

1. Place the mouse cursor over the net and select the **Place** → **Trace Name** command.
2. In response to the `Trace Name?` prompt, type the name of the net, for example **CE**.
3. Double click to place the trace marker, then click with the right button to exit the command.

To add stimulus information, follow these steps for each net that is between an IPAD and an IBUF.

1. Place the mouse cursor over the net and select the **Place → Stimulus** command.
2. In response to the Stimulus? prompt, type the following:

   For CE:

   ```
   0:1 6000:0 8000:1
   ```

   For CLK:

   ```
   0:0 500:T 1000:G:500
   ```

   For CLR:

   ```
   0:0 13000:1 15000:0
   ```

3. Double click to place the stimulus marker, then click with the right button to exit the command.

Save your design using the **Quit → Update File** command. Exit Draft using **Quit → Abandon Edits**.

Step 4 — Prepare Simulation Vectors (Optional)

While still in OrCAD select **Digital Simulation Tools**. A menu will appear. From the menu select **Local Configuration**. A configuration menu will appear. Select **Configure ASCTOVST**.

A dialog box will appear. Check **Source is a Stimulus File**, and make sure the file adjacent to it reads jcount.ast. If it does not read jcount.ast, highlight the entry and type **jcount.ast**.

Select **Digital Simulation Tools** again. From the menu select **Local Configuration**. Select **Configure ASCTOVST**.

Check **Source is a Trace File**, and make sure the file adjacent to it reads jcount.atr. If it does not read jcount.atr, highlight the entry and type **jcount.atr**.

Step 5 — Create Functional Simulation Model

If you have OrCAD for Windows:

1. Open the **Xilinx Tools** Window.
2. Click on the **Simulation Utility** icon.
3. Select **jcount.sch**

4. Select **OrCAD**
5. Select **Functional Simulation**

If you have a DOS version of OrCAD:

1. Exit Windows (or suspend) and go to DOS.
2. Type at the DOS prompt: **xsimmake -f oef6 jcount**
3. Press return. After processing has completed, exit back to OrCAD.

Step 6 — Perform Functional Simulation (Optional)

Double click on **Design Management Tools**, select **JCOUNT** from the list, and click on **OK**.

Double Click on **Digital Simulation Tools**.

Click once on **Simulate**. Select **Local Configuration** and then **Configure Simulate**. Change jcount.inf to **jcount.vst**, then select **OK**.

Double click on **Simulate**. A blank simulation waveform window appears with the R, Q3, Q2, Q1, Q0, CE, and CLK nets listed.

Select **Edit Stimulus** → **Yes**. The Stimulus Editor window appears.

Select **Add**. Press the down arrow key to highlight the **Signal Name** field, then select **Browse**. Type a **P** to scroll down to the net names beginning with P. Press the down arrow key to highlight **PRLD**, then press ↵.

Press the down arrow key to highlight the **Initial Value** field. Type **1**.

Press the down arrow key again. Select **Add**. In response to `Time of Function?` type **10**, then press ↵ to accept a 0 (zero) value.

Select **Return** to return to the main stimulus editor, select **Write** → **Yes** to add PRLD to the stimulus file, and select **Use** to return to the waveform window.

Select **Trace** → **Change View** and enter **150**. Select **Run Simulation** and enter **18000**. The simulation waveforms appear.

Exit Simulate using **Quit** → **Abandon Simulation**.

If you are in OrCAD for Windows, open the **Xilinx Tools** icon and start the **Design Manager**.

To exit the DOS version of OrCAD, double click on **To Main**, then on **Exit ESP**. You are now back in DOS; start Windows, open the **Xilinx Tools** icon, and start the **Design Manager**.

Step 7 — Fit the Design and Create a Programming File

From the Design Manager create a new project.

File → New Project

The new project dialog box appears. **Browse** on **Input Design** to select **jcount.sch**. Select **XC7000** as the **Target Family.** Use the **Work Directory** and **Browse** key to find a directory to place the output in. Then select **Translate**.

Select **Tools → Flow Engine**. Set **Stop After** to **Bitstream** and select **RUN**. The program optimizes the design, fits the design, creates timing file and programming files all in one step. The interface keeps you updated on the progress of processing.

Step 8 — Examine the Reports

Examine the reports to verify that the design was implemented as you expected. To examine reports from the Xilinx Design Manager select **Utilities → Report Browser,** or select the report browser icon. The following reports are most useful for schematic designs:

Figure 1-10 Report Browser

- Translation Report — The translation report informs you of how the translation went and lists any errors or problems that occured during translation.

- Fitting Report — This fitting report shows the allocation of Function Block resources.
- Resource Report — The resource report lists the device resources used by the design and the resources remaining.
- Pinout Report — The pinlist report shows how the external nets in your design were mapped to the device pins.
- Mapping Report — The mapping report tells you how the logic in the design was mapped to the device.
- Timing Report — The timing report shows the calculated worst-case timing for the logic paths in your design.
- Timespec Report — The timespec report tells you if you have made any errors in assigning TSPEC attributes in your design and list the TSPECS that the fitter used.

Step 9 — Timing Simulation

If you skipped functional simulation, follow the instructions for functional simulation found in this procedure. If you did perform functional simulation, continue with the following instructions.

Create Timing Simulation Model

To run a timing simulation on the design:

1. Open **Xilinx Tools**
2. Select the **Simulation Utility** icon
3. Select **jcount.sch**
4. Select **OrCAD**
5. Select **Timing Simulation**

If you are running OrCAD from DOS:

1. Exit Windows. Start OrCAD, then double click on **Design Management Tools**, select **JCOUNT** from the list, and click on **OK**.
2. Double Click on **Digital Simulation Tools**. Double click on **Simulate**. A blank simulation waveform window appears with the R, Q3, Q2, Q1, Q0, CE, and CLK nets listed.

3. Select **Trace → Change View** and enter **125**. Select **Run Simulation** and enter **15000**. The simulation waveforms appear.
4. Exit Simulate using **Quit → Abandon Simulation**.

Chapter 2

Device-Independent Design

This chapter discusses how and why to create a device-independent or device-specific design. It also explains how to take an existing FPGA design and retarget it to an EPLD device.

Choosing Components

The Unified Library contains all Xilinx component symbols, some of which are specific to one device family, some of which are common to two or more families, and some of which are common to all families.

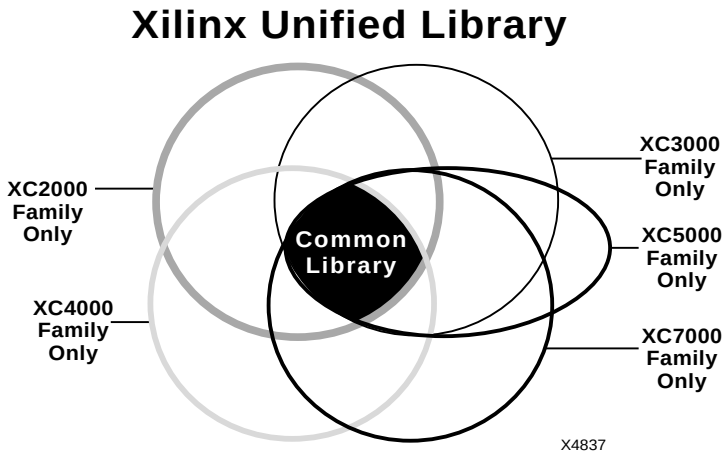


Figure 2-1 Common Symbols in the Unified Library

When a component of the same name is present in multiple families' libraries, it has the same functionality and graphic symbol body, and similarly named pins. However, the component's implementation,

including whether the symbol is a primitive or macro (with underlying schematic), may vary between families.

When to Use EPLD-Specific Components

In general, common library components work well for EPLD designs. You should use XC7000-specific components only under these special conditions:

- If you are cascading arithmetic components, you should use EPLD-specific arithmetic components, because the carry lines that go between components are mapped to the carry chain.
- If you are cascading up/down counters, you should use EPLD-specific counters, because the separate up and down terminal counts can be placed in the UIM for greater speed and density.

For example, suppose you were working on a design that needed an 8-bit full adder. In most designs, you could use a device-independent adder, such as ADD8. If this component is used in an EPLD design, the internal logic is mapped onto the special EPLD arithmetic carry lines; if used in an FPGA design, the logic is mapped in the way that is most efficient for the FPGA's architecture.

However, if you needed to cascade two 8-bit adders, it would be most efficient to use EPLD-specific adders, because the carry lines that go between components would be mapped to the carry chain.

When to Use Common Components

To make your design device-independent, use only the symbols common to all device families. The XACT*step* software automatically maps the symbols in your design onto the chosen target device. Creating a device-independent design allows you to easily test your design with different Xilinx devices.

For a complete list of Unified Library components and their compatibility with the different Xilinx device families, refer to the *Libraries Guide*.

Attributes and Device Independence

If you want your design to be completely device-independent, do not use schematic attributes. There are a few attributes common to FPGA and EPLD devices:

- PART=*device_name* (or PARTTYPE in OrCAD)
- INIT=R|S
- Timing specifications for TIMESPEC and TIMEGRP symbols
- LOC=*device_pin*
- FAST

For the PART and LOC attributes you must change the values when you change devices. All other EPLD-compatible attributes are EPLD-specific. For more information about these attributes, refer to the "Attributes" appendix. These attributes are also described in the context of how they are used in the "EPLD Architecture and Design Tradeoffs" and "Design Applications" chapters of this manual.

General Conversion Procedure: FPGA to EPLD

The basic steps for retargeting an FPGA design to an EPLD device are as follows. Examples in this section assume you are converting from the XC3000 family.

Viewlogic Procedure

1. Enter your CAE tool and open your design.
2. Use the conversion program to make the symbols in your schematic reference the reconfigured libraries.
   a) Select the **`Design Entry`** button.
   b) Click on **`Select Family`**. A list of families will appear. Select **`XC7000`** and click on **`OK`**. The **`Altran`** menu will appear.
   c) Select XC3000 under **`Current Technology Aliases`**.
   d) Select XC7000 under **`Target Technology Aliases`**, then select **`OK`**. The conversion program will execute.
3. Create a new Xilinx project for the converted schematic design.

From the Design Manager click on the **File** menu and select **New Project**.

Change the name of the project so that you can recognize the project as XC7000 (to a name such as CALC7K).

From the **Target Family** select **XC7000**.

4. Find all the library components in the schematic that are not EPLD-compatible, and use the *XACT Libraries Guide* to find EPLD-compatible equivalents.
   - If you have already run Xaltran (in the previous steps), incompatible components appear as white squares in the schematic.
5. If necessary, use EPLD-specific components to obtain a more efficient design. In most cases, device-independent components are mapped onto the EPLD architecture efficiently, but there are exceptions, which are described in the "When to Use EPLD-Specific Components" section earlier in this chapter.
6. Remove all attributes except INIT, FAST, and timing specifications. Make sure timespec syntax is up to date because EPLD software does not accept old timespec syntax (use syntax as described in this manual). Change the values of PART and LOC as needed, or remove them.
7. Enter the Flow Engine and press **RUN**.

   From the Design manager click the **Tools menu** and select **Flow Engine.** The flow engine window will appear.

   Set the **Stop After** selection is to **Fit**,**Timing** or **Bitstream**, depending on where you want processing to stop.
8. When you perform either functional or timing simulation, you must set the MRESET signal Low then High instead of setting GSR (global reset).
9. If you wish to perform timing simulation, you may have to change the internal nodes you drive and monitor. The EPLD fitter optimizes the logic, which makes many of the internal nodes in the design disappear. However, all external signals are always visible.

Note: If your FPGA design has RAM, ROM, or other elements that do not have EPLD equivalents, you cannot retarget your design unless you redesign those parts.

OrCAD Procedure

1. Change (cd) to your working directory and reconfigure the libraries in your CAE tool for the XC7000 family.
   - Type the following at the DOS prompt while in your working directory:

     ```
     xdraft 7
     ```
2. Enter your CAE tool and open your design.
3. Find all the library components in the schematic that are not EPLD-compatible, and use the *XACT Libraries Guide* to find EPLD-compatible equivalents.
   - If you have run XDraft 7 on an OrCAD schematic, the incompatible components generate error messages in OrCAD and do not appear in the schematic.
4. If necessary, use EPLD-specific components to obtain a more efficient design. In most cases, device-independent components are mapped onto the EPLD architecture efficiently, but there are exceptions, which are described in the "When to Use EPLD-Specific Components" section earlier in this chapter.
5. Remove all attributes except INIT, FAST, and timing specifications. Change the values of PART and LOC as needed, or remove them.
6. Enter the Flow Engine and press RUN. Make sure the **`Stop After`** selection is set to **`Fit,Timing`** or **`Bitstream`**, depending on where you want processing to stop.
7. When you perform either functional or timing simulation, you must set the MRESET signal Low then High instead of setting GSR (global reset).
8. If you wish to perform timing simulation, you may have to change the internal nodes you drive and monitor. The EPLD fitter optimizes the logic, which makes many of the internal nodes in

the design disappear. However, all external signals are always visible.

Note: If your FPGA design has RAM, ROM, or other elements that do not have EPLD equivalents, you cannot retarget your design unless you redesign those parts.

Converting Behavioral Modules

If your design contains Xilinx-ABEL or XBLOX modules, you must perform these additional steps before your normal schematic flow:

1. Change the encoding of state machine modules. You do not have to rewrite the logic, just the state assignment. For FPGAs, which are rich in registers, one-hot encoding (a symbolic state machine) is most efficient. For EPLDs, which are rich in product terms, maximal encoding (an encoded state machine) is most efficient. Conversion may be unnecessary for very simple state machines.

   For more information about symbolic and encoded state machines, see the "Simulating an ABEL-HDL Design" and "Converting Encoded State Machine to Symbolic State Machine" sections in the "Design Examples" chapter of the *Xilinx ABEL User Guide*. You should follow the steps of the latter section in reverse.

2. Rerun XABEL specifying Xilinx EPLD as the target architecture. XABEL will create an equation file with extension *.pld*. In your schematic, continue to use the existing Xilinx-ABEL symbols, but change the DEF=XABEL attributes to DEF=PLD and FILE=*filename*. To ensure that the software does not process old files, delete the *file_name*.xnf files in the xnf directory.

3. Convert the XBLOX modules. There is no straightforward conversion—you can rewrite the logic using a behavioral entry tool (Xilinx-ABEL) or create an EPLD-compatible lower-level schematic.

FPGA to EPLD Conversion Example: CALC Design

The CALC design is extensively documented in the tutorial chapters of the Xilinx Interface Guide for your CAE tool. It is used as an example of an FPGA design. This section describes the steps necessary to convert the XC3000 version of this design to an EPLD design.

The steps are different for each CAE tool. The procedure for Viewlogic users is described first, followed by OrCAD.

Procedure for Viewlogic Users

Reconfiguring the Libraries and Schematic Symbols

To reconfigure the libraries for the XC7000 family and update all the symbols so their properties are compatible with XC7000 devices, follow these steps:

1. Copy the CALC tutorial files to a directory under your home directory as described in the *Viewlogic Interface Guide* and *Viewlogic Tutorials* manual. Change (cd) to the calc\soln_3k directory.
2. Use the conversion program to make the symbols in your schematic reference the reconfigured libraries.
   a) Select the **Design Entry** button.
   b) Click on **Select Family**. A list of families will appear. Select **XC7000** and click on **OK**. The **Altran** menu will appear.
   c) Select XC3000 under **Current Technology Aliases**.
   d) Select XC7000 under **Target Technology Aliases**, then select **OK**. The conversion program will execute.

Editing the Schematic

To edit the schematic so all its symbols are compatible with XC7000 devices, follow these steps:

1. Start up Design Manager. Start up **PROFlow** and select the **PROCapture** button from the menu.
2. Open the CALC schematic and select the OSC_3K file.

   Select the **Files** menu and click on **Open Project**.

   Find the CALC directory and select OSC_3K.

3. Edit the schematic so it looks like Figure 2-2:

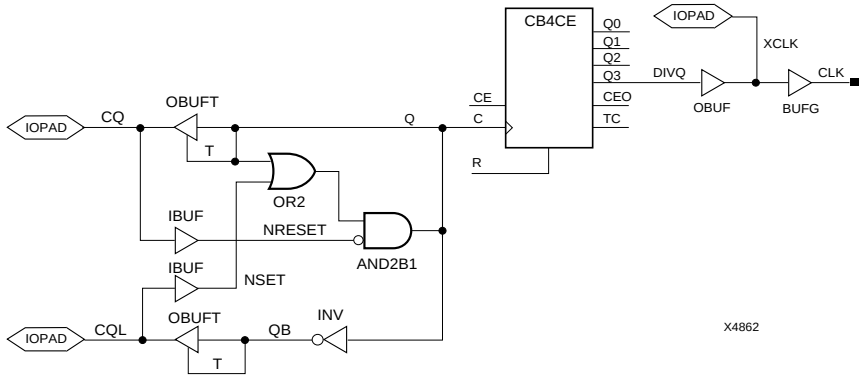


Figure 2-2 New Viewlogic Oscillator Schematic for EPLD CALC

4. Be sure to label the new IOPAD net "XCLK".
5. Select the **File** menu and select **Save.** Select the **View→ Pop** command. (If you wish, you can use the **File → Save As** command to save this schematic as OSC_7K. If you save this schematic to a new name, be sure to save the corresponding symbol file to the same name. In addition, make sure the new symbol is added to and saved in the top-level CALC schematic.)
6. Search the top-level schematic and every sub-schematic for attributes, including LOC= pin assignments. Delete all except the PART= attribute. Attributes are displayed in yellow to distinguish them from labels, which are white. Save each schematic that you change using the **File → Save** command.
7. Use the **Change → Text** command to change the part name as follows:

   PART=7372-10PC68

8. Save the top-level schematic using the **File → Save** command.

Entering Xilinx Design Manager Environment

To map the design onto a target XC7000 device, follow these steps:

1. Make sure the calc.1 project is the active project on the Design Manager.

2. Enter the Flow Engine:

   Select **Tools → Flow Engine.**

   The Flow Engine menu appears.

3. To run optimization, the fitter, timing and bitstream:

   Select the down arrow beside **Stop After** and select **BITSTREAM.**

   Select the **RUN** softkey.

   The optimization will take some time. The Flow Engine lets you know what it is working on.

Performing Timing Simulation

To perform timing simulation, follow these steps:

1. Go to Viewlogic PROflow and select the **PROsim** button (adjacent to Timing Simulation). Place a check in the **Command File** box and click on **Browse**.

   Select the **calc.vsm** file from the list and select **OK.**

2. Run timing simulation as described in the PROseries tutorial in the *Viewlogic Interface Guide*, except change the lines in the Vector Definition, Simulation Output Definition, Clock Definition, and Global Reset & Initial Input Values to those shown in the following file. The ... indicates comments that have been removed.

   You must make these changes because the EPLD fitter optimizes the logic, removing many internal nodes. This makes the design more efficient, but harder to simulate.

```
|--------------------VECTOR DEFINITION-------------------
...
vector SW sw7\sw6_p sw7\sw5_p sw7\sw4_p sw7\sw3_p +
sw7\sw2_p sw7\sw1_p sw7\sw0_p
| You can also use bus syntax when defining vectors
vector ALU alu[3:0]
vector LED_P led\led[3:0]_p
| Set radices for vectors
|   The default radix is binary for input, hex for output
radix hex SW ALU
radix bin LED_P
|--------------SIMULATION OUTPUT DEFINITION-------------
...
```

```
wave calc_7k.wfm osc_7k\xclk SW exc_p ALU LED_P we rst
| Save simulation values for these nodes
watch osc_7k\xclk SW exc_p ALU LED_P we rst
| Output the values of all watched signals each time
|   "xclk" goes high.  Create tabular output.
break osc_7k\xclk 1 do (print > calc_7k.tab)
| Output node and vector transitions and simulation time
|   whenever any of the nodes or vectors changes state
trace osc_7k\xclk SW exc_p ALU LED_P we rst > calc_7k.trc
|-------------------CLOCK DEFINITION----------------------
clock osc_7k\xclk 1 0
| Use a clock period of 100ns.  Set stepsize=50ns
step 50ns
|------------GLOBAL RESET & INITIAL INPUT VALUES----------
| Set initial values for all inputs using the "H" and "L"
|   commands for nets and "assign" for vectors
h exc_p
assign SW 00\h
| Initialize all flip-flops (preload- is active high
|   for 7k designs, you can abbreviate to prld)
h prld
| Viewsim uses units of 0.1 ns, so this statement
|   simulates for 100 ns.
cycle
l prld
cycle
```

Procedure for OrCAD Users

Reconfiguring the Libraries and Schematic Symbols

To reconfigure the libraries for the XC7000 family and update all the symbols so their properties are compatible with XC7000 devices, follow these steps:

1. Copy the CALC tutorial files to a directory under your home directory as described in the *OrCAD Interface/Tutorial Guide*. Change (cd) to the calc\soln_3k directory.
2. While in your working directory, type the following at the DOS prompt:

   `xdraft 7`

Editing the Schematic

To edit the schematic so all its symbols are compatible with XC7000 devices, follow these steps:

1. Invoke **`OrCAD`** from Windows or from DOS.
2. Double click on the **`Design Management Tools`** button. Select **`CALC`** from the list on the left, then select the **`OK`** button.
3. Enter **`Schematic Design Tools`** and then **`Draft`**. The top-level schematic of the CALC design appears. If you see a message about an X being deleted, do not worry; this is an FPGA-specific property that you would have to delete anyway.
4. Change the PART= text to **`7372-10PC68`**.
5. Enter the OSC_3K sheet symbol using the **`Quit`** → **`Enter Sheet`** → **`Enter`** command.

   The OSC_3K schematic is displayed. There may be messages telling you that two components, ACLK and GCLK, are unavailable. The spaces where these two components were on the original schematic are left open, with unconnected nets.
6. Edit the schematic so it looks like Figure 2-3. The part of the schematic that is not visible is unchanged except for LOC= properties being removed.

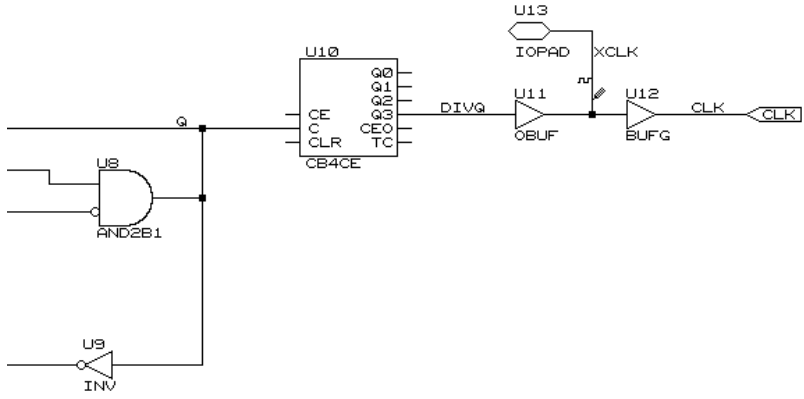


Figure 2-3 New OrCAD Oscillator Schematic for EPLD CALC

7. Be sure to label the new IOPAD net "XCLK".
8. Be sure to delete the LOC=... properties from the two IOPADs at the left end of the schematic. (Use the **Edit → Edit → 1st Part Field → Name** command.)
9. Select **Quit → Update File → Leave Sheet**.
10. Use the **Quit → Enter Sheet**, **Edit → Edit→ 1st Part Field → Name**, and **Quit → Update File → Leave Sheet** commands to delete all the LOC= and FAST statements in the CALC schematic and all its sub-schematics.
11. When you are finished editing all the schematics, select **Quit → Update File → Abandon Edits** from the top-level schematic, double click on the **To Main** button, double click on the **Exit ESP** button, and press any key to return to Design Manager.

Running the Fitter Commands

From the Design Manager create a new project.

File → New Project

The new project dialog box appears. **Browse** on **Input Design** to select **jcount.sch**. Select **XC7000** as the **Target Family.** Use the **Work Directory** and **Browse** key to find a directory to place the output in. Then select **Translate**.

Select **Tools → Flow Engine**. Set **Stop After** to **Bitstream** and select **RUN**. The program optimizes the design, fits the design, creates timing file and programming files all in one step. The interface keeps you updated on the progress of processing.

Performing Timing Simulation

To perform timing simulation, follow the instructions in the "VST Tutorial" chapter of the *OrCAD Interface/Tutorial Guide*, with the exceptions in the following steps.

1. Place stimulus and trace information on the XCLK signal in the OSC_3K schematic instead of on the CLK signal, and on the LED_P signals in the LED schematic instead of on the STACK signals.

2. In addition, you will have to move the stimulus information in the SW7 schematic from the SW signals to the SW_P signals and subtract one clock cycle. Follow these steps:

   a) Place the mouse cursor on the stimulus symbol on the SW4 signal and select the **Inquire** command. The following line is displayed in the top left corner of the screen:

   ```
   Stimulus: 0:0 6000:1
   ```

   b) Subtract 500 time units, which is equal to one clock cycle, from the times at which the signal changes value. Replace all time values less than or equal to 500 with 0. In this case, 0 remains 0 and 6000 becomes 5500.

   c) Use the **Place** → **Stimulus** command to add the following stimulus to the SW4_P signal:

   ```
   0:0 5500:1
   ```

   d) Delete the stimulus indicator on the SW4 signal.

   e) Repeat steps a through d to delete all stimulus information from each SW signal and add it to each SW_P signal.

   f) Use **Quit** → **Update File** to save your edits.

   The EPLD fitter software optimizes away many internal nodes such as the CLK, STACK, and SW7 signals, but cannot optimize external signals such as the XCLK, LED_P, and SW_P signals.

3. Substitute “EPLD” wherever the VST Tutorial says “FPGA”. For example, select **Orcad_Epld_Timing** as the XSIMMAKE flow name instead of Orcad_Fpga_Timing.

4. Instead of adding a GR or GSR stimulus (p. 12-21), add a PRLD stimulus. Like GSR, the PRLD stimulus has an initial value of 1 and is brought down to 0 at 1 nanosecond.

Converting a Xilinx-ABEL Module (Optional)

You can use a Xilinx-ABEL module instead of the STATMACH schematic in the CALC design. The stat_abl.abl file specifies the logic. Substituting this Xilinx-ABEL module is described in the Interface User Guide for your CAE tool.

1. Use Xilinx-ABEL to create a stat_abl.pld file.

2. Start up your CAE tool.
3. Open the schematic named CONTROL. Remove the DEF=XABEL attribute from the STATMACH symbol, which referenced the Xilinx-ABEL module, and change to FILE=stat_abl and DEF=PLD. For specific instructions on changing attributes, see the "Attributes" appendix.

Note: If you wish, you can convert the stat_abl.abl file from symbolic (one-hot) encoding to maximal encoding. It is not really necessary, however, because this state machine has only three states, and therefore the conversion spares only one register.

Note: The Xilinx initial state property is not supported for XC7000 designs. To initialize this one-hot encoded state machine, use XEPLD register preload statements as shown below.

```
xepld property 'sother.prld=ucc';
xepld property 'swe.prld=gnd';
xepld property 'spush.prld=gnd';
```

EPLD Architecture and Design Trade-offs

This chapter discusses EPLD architecture and trade-offs in fitting your design to the EPLD architecture: designing for speed, density, or pinout preservation; controlling power consumption; and controlling preload values.

The tips and techniques in this chapter are guidelines only. They are general principles that work in most cases. They may or may not be applicable to a particular design.

EPLD Architecture

EPLD devices have special architectural features that can make your design faster and more efficient. The XEPLD fitter software automatically analyzes your design, optimizes the logic, and maps functions into the appropriate device resources. However, an understanding of the EPLD architecture can help you exercise complete control of design optimization.

For more detailed information about EPLD architecture, refer to the EPLD device data sheets.

Figure 3-1 is a simplified diagram of the XC7354 device that shows the main architectural features of EPLD devices.

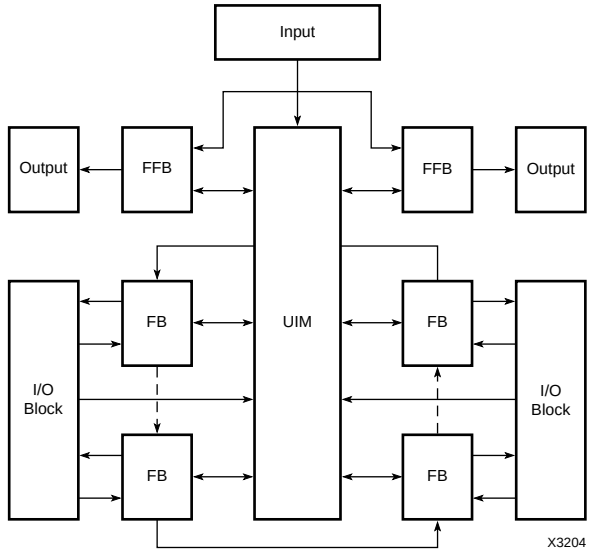


Figure 3-1 EPLD Device Structure

The five basic architectural features in an XC7300 family EPLD device are as follows:

- Input Pads
- Output Pads
- High Density Function Blocks (HDFBs)
- Fast Function Blocks (FFBs)
- The Universal Interconnection Matrix (UIM)

This section describes these features and how designs are mapped onto them for best results.

Note: For a complete explanation of the XC7000 architectural features, see the EPLD device data sheets.

Input Pad Structures

The XC7000 devices have two types of input pads: Fast Inputs and standard inputs.

Fast Input pins have two paths through the device. One path drives directly into the Fast Function Blocks, bypassing the UIM, and is used for signals that require the fastest propagation delays and shortest macrocell register setup times.The second path drives all function blocks (both FFBs and HDFBs) through the UIM.

Standard inputs and UIM paths of Fast Inputs can be configured as follows:

- Registered
- Registered with clock enable
- Latched
- Combinatorial

Registering and/or latching signals at the input pad shortens register setup times and is used most often to pipeline data on-chip or synchronize asynchronous inputs. The input pad registers can also store data, making more macrocells available for implementing logic.

Output Pad Structures

The XC7000 devices have two types of output pads: those driven by HDFBs, which have standard drive capability; and those driven by FFBs, which have higher drive capability. These outputs can be configured as follows:

- 3-state with individual p-term control (HDFB only)
- 3-state with FOE control
- Direct (always on)

Each output pad driven by a HDFB can be 3-stated by its own macrocell product term for maximum flexibility. The global FOE net offers maximum speed.

Bidirectional pins have both input pad structures and output pad structures. See your device data sheet for details.

High-Density Function Blocks

High Density Function Blocks provide the maximum amount of logic resources for use in your design. They are well-suited for arithmetic functions, counters, and other kinds of complex logic.

High Density Function Blocks contain special fast carry lines for arithmetic logic. These lines extend between High-Density Function Blocks, allowing fast carry for very large arithmetic functions.

Fast Function Blocks

The XC7300-series devices have a combination of High Density Function Blocks and Fast Function Blocks; this is called "Dual Block Architecture."

Logic placed in Fast Function Blocks performs faster than logic in High Density Function Blocks. Fast Function Blocks are well-suited for critical decoding and ultra-fast state machine applications.

The Universal Interconnect Matrix (UIM)

The Universal Interconnect Matrix, or UIM™, provides a 100% interconnection matrix allowing any function block output to drive any function block input in the device; routing is never blocked. All function block inputs (except for the Fast Inputs) come from the UIM.

The UIM can perform wired-AND functions, which the software uses automatically when possible to improve resource utilization.

Designing for Speed

To optimize for speed (faster pin-to-pin and register setup times), follow these guidelines:

- Use General Timing Optimization
- Use XACT Performance (Timespecs)
- Use EPLD-specific arithmetic functions when cascading.
- Use EPLD-specific bidirectional counters when cascading.
- Reduce levels of logic.
- Use active-High clock inputs and active-High output-enable control inputs.

If you want to use General Timing Optimization without specifying any Timing Specifications, you can simply go to the **XC7000 Implementation Template** and turn on the **Timing Optimization** button. General Timing Optimization will shorten your critical paths as much as it can. In general, density optimization is the default for the fitter.

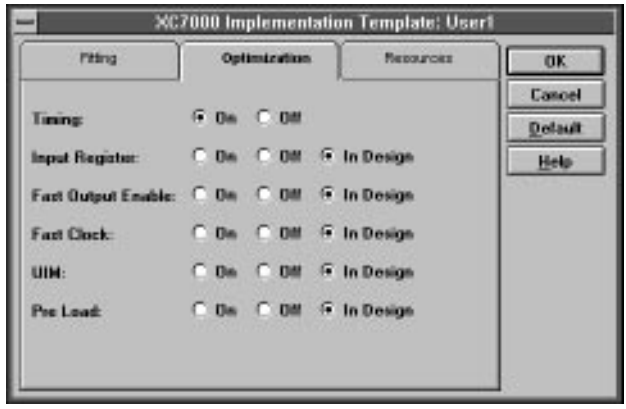


Figure 3-2 Optimization Template

Using XACT Performance

You can use Timing Specifications (T-Specs) to specify the maximum allowable delay between groups of components in your design. The software then optimizes and maps your design to achieve the timing defined by these specifications. This Xilinx EPLD timing-driven optimization is called XACT-Performance.

T-Specs can be applied directly to a schematic design or they can be specified in a constraints file for behavioral and VHDL designs. Do not attempt to edit a constraints file (.CST) when using schematic design. Always change the timing specifications within the schematic.

Timing Definitions

Delays and times are calculated as defined by the path types in this section.

The path types are defined as follows:

- Clock to Setup — Register to register cycle time, including clock to output and setup delay.

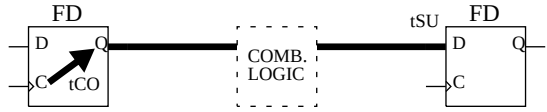


Figure 3-3 Clock to Setup Path

- Pad to Pad — Combinational pad to pad delay.

Figure 3-4 Pad to Pad Path

- Clock to Pad — Delay from the register clock input to the output pad.

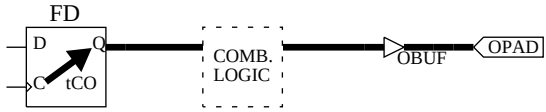


Figure 3-5 Clock to Pad Path

- Pad to Setup — Data path delay from the pad to the register data input. Includes the register setup time.

Figure 3-6 Pad to Setup Path

- Setup to Clock at the Pad — Setup time of data at the pad to clock at the pad. This path type includes only global clocks and product

term clocks driven directly from input pads. If the data input is signal A and the clock input is signal CLK, the timing calculation for this type of path is as follows:

Max(A to D) – Min(CLK to C)

Max and Min are maximum and minimum propagation delays through the combinational logic.

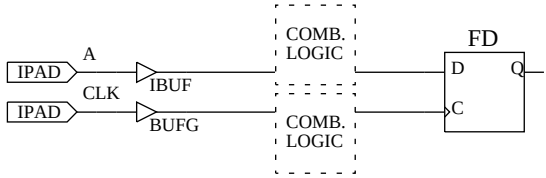


Figure 3-7 Setup to Clock at the Pad Path

- Clock Pad to Output Pad — Clock pad to output pad propagation delay. The clock can be a global or product term clock.

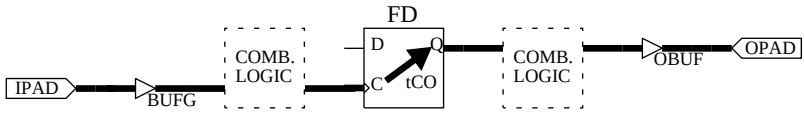


Figure 3-8 Clock Pad to Output Pad Path

- Paths Ending at Clock Pins of Flip-Flops — Delay from clock pad to register clock input. The clock can be a global or product term clock.

Figure 3-9 Path Ending at the Clock Pin of a Flip-Flop

The TIMESPEC Primitive

Timing specifications are placed on your schematic using a TIMESPEC primitive that contains the TIMESPEC Attribute Definitions which control the timing for paths between defined groups of components.

The TIMESPEC primitive, as illustrated in Figure 3-10, is 30 characters wide and contains TS attribute definitions. Each TIMESPEC primitive can hold up to eight TS attribute definitions. If you want to include more than eight TS attribute definitions, you can use multiple TIMESPEC primitives in your schematic.

Note: Though the TIMESPEC primitive is only 30 characters wide, you can create TS attribute definitions of any length by continuing on the next line.

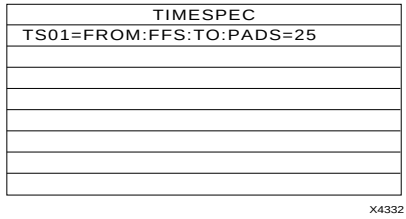


Figure 3-10 TIMESPEC Primitive

How you add a TIMESPEC primitive to your schematic depends on your specific schematic-entry software. Refer to the appropriate Xilinx Interface User Guide for step-by-step instructions.

Defining Timing Path End Points

Specify the start and end points of your timing paths using one of the following methods:

- Refer to a predefined group by specifying one of the corresponding keywords — FFS, PADS, LATCHES.
- Create arbitrary groups within a predefined group by tagging symbols with TNM (pronounced *tee-name*) attributes.

- Create groups that are combinations of existing groups by using TIMEGRP symbols.
- Create groups by pattern matching on signal names.

Using Predefined Groups

You can refer to a group of flip-flops, input latches, or I/O pads, by using the corresponding keywords:

Table 3-1 Predefined Groups

Keyword	Group
FFS	Macrocell or IOB flip-flops
LATCHES	input latches only; not latches built from macrocell registers
PADS	input/output pads

These predefined groups represent all symbols of that type. For example the following TS Attribute means that the delay between any two flip-flops must be no greater than 30 ns.

```
TS01=FROM:FFS:TO:FFS=30
```

And the following TS attribute means that the delay between any I/O pad and any flip-flop must be no greater than 25ns.

```
TS_OTHER=FROM:PADS:TO:FFS=25
```

For more information on using TNM attributes and Time GRP, please refer to the *XEPLD Reference Guide*.

Specifying Time Delay Units

Nanoseconds are the default units for specifying delay times in TS attributes. However, after specifying the maximum delay or minimum frequency numerically, you can enter the unit of measure by specifying the following:

- NS for nanoseconds
- MHZ for megahertz
- US for microseconds

- KHZ for kilohertz

The software converts all units to nanoseconds and rounds them to 0.1 ns accuracy.

CST Files

You can create a constraints file for use in a VHDL or behavioral design by using an ASCII editor. Save the file as *file_name*.CST in your project directory. The constraints file can be used in processing your design if you specify it from the **Design Implementation Dialog** box as shown in Figure 3-12. See the *XEPLD Reference Guide* for more information. An example constraints file is shown below.

```
TIMESPEC="TS02=FROM:FF_1:TO:FF_4=20";
TIMESPEC="TS08=FROM:FFS:TO:PADS=TS05*10";
TIMESPEC="TS59=FROM:G_1A:TO:G_8C=15";
TIMESPEC="TS62=FROM:LATCHES:TO:PADS=25";
TIMESPEC="TS65=FROM:PADS:TO:PADS=18";
TIMESPEC="TS73=FROM:FFS(a*):TO:FFS(b*)=20"
;
TIMEGRP="many_ffs=ffs1:ffs2";
TIMEGRP="group1=ffs(ctr*)";
TIMEGRP="group2=group1:except:ffs(ctr0)";
TIMEGRP="ff1=FFS(a*)";
```

Figure 3-11 Sample Contraints File

Figure 3-12 Design Implementation Options

Note: For VHDL design and behavioral design, .cst files are the only mechanism to enter your timing specifications. For schematic designs you have attributes and T-specs.

Enabling Timing Specifications

To enable global timing-driven optimization (XACT-Performance), do the following:

1. Open the DESIGN MANAGER window.
2. Select Design from the tool bar.
3. Select Implement from the Implementation menu, which brings up the XC7300 Design Implementation Dialog Window, as follows:

Figure 3-13 XC7300 Design Implementation Dialog Window

4. Select Edit Template which brings up the XC7300 Implementation Template Window for Fitting, as follows:

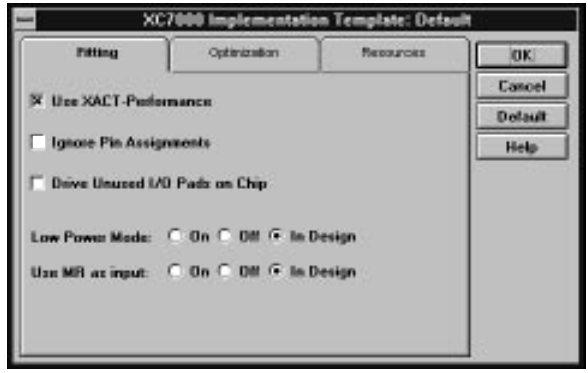


Figure 3-14 XC7300 Implementation Template — Fitting

5. If you place an X on **`Use XACT Performance`** the software will use your T-Spec information (if it exists) when fitting your design. If you do not select this option, the software will ignore all T-Spec information.

Using EPLD-Specific Arithmetic Functions

The schematic entry libraries supporting XC7000 EPLDs contain two types of arithmetic components, generic symbols shared in common with their Xilinx family libraries, and EPLD-specific symbols.

XC7000-specific arithmetic components use the fast carry chain for their CI and CO pins. This allows you to build up an arithmetic function of any desired size by cascading these library symbols and achieving the maximum possible speed supported by the EPLD.

Equivalent common components do not use the fast carry chain for their CI and CO pins and therefore are slower and use more device resources when cascaded. If you are not cascading, however, the common arithmetic components work just as well.

Note: If you use EPLD specific arithmetic components, you might have to make changes before moving to other families.

Cascading Counters

Likewise, the library contains both common and EPLD-specific versions of up/down counter components. If you are cascading bidirectional or down counters, you should use XC7000-specific counters. These counters have separate up and down terminal counts (CEOU and CEOD) that can be cascaded in the UIM. The up terminal count is generated by ANDing all of the counter output bits. The down terminal count lookahead is generated in a macrocell. These terminal counts are then ANDed with the count enable inputs (CEU and CED) to produce the component's up and down terminal count outputs as shown in Figure 3-15.

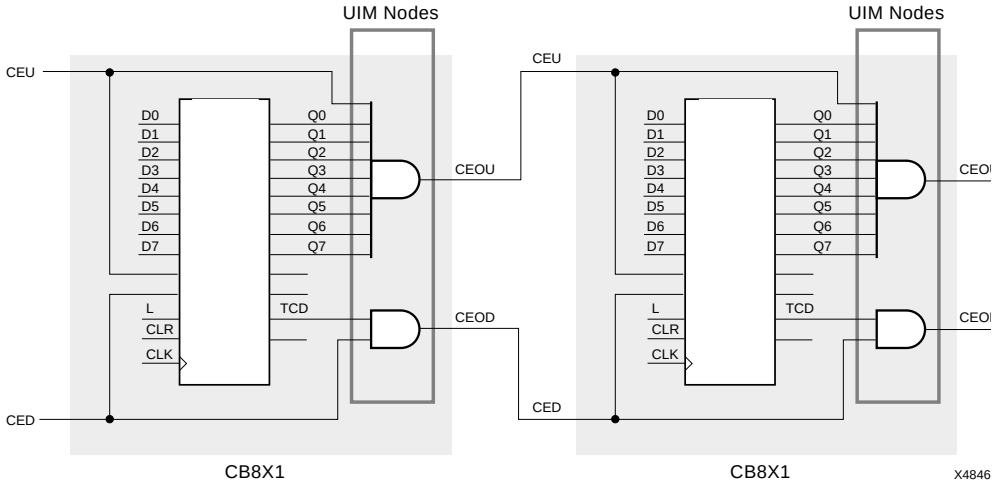


Figure 3-15 Cascading EPLD-Specific Up/Down Counters

Because the XEPLD optimization software collapses the cascaded AND-gate logic into a single level of logic, the speed of the cascaded counter is constant, no matter how many bits it has. Very large counters can still be implemented at maximum speed because the software can implement large AND-gate functions in the UIM array as needed.

However, in common library bidirectional counters, the up and down terminal counts are combined into a single terminal count. This terminal count uses both the true and the complement of the counter bits, which makes the terminal count impossible to place in the UIM.

Reducing Levels of Logic

Each EPLD macrocell has several levels of logic followed by a register. If you put the logic first, the XEPLD fitter software maps the logic and register into the same macrocell. If you put logic after the registers, however, the XEPLD fitter software is generally forced to use additional macrocells for the logic that follows the registers, decreasing both the speed and density of your design. Figure 3-16 shows two equivalent circuits, one that is efficient and one that is inefficient.

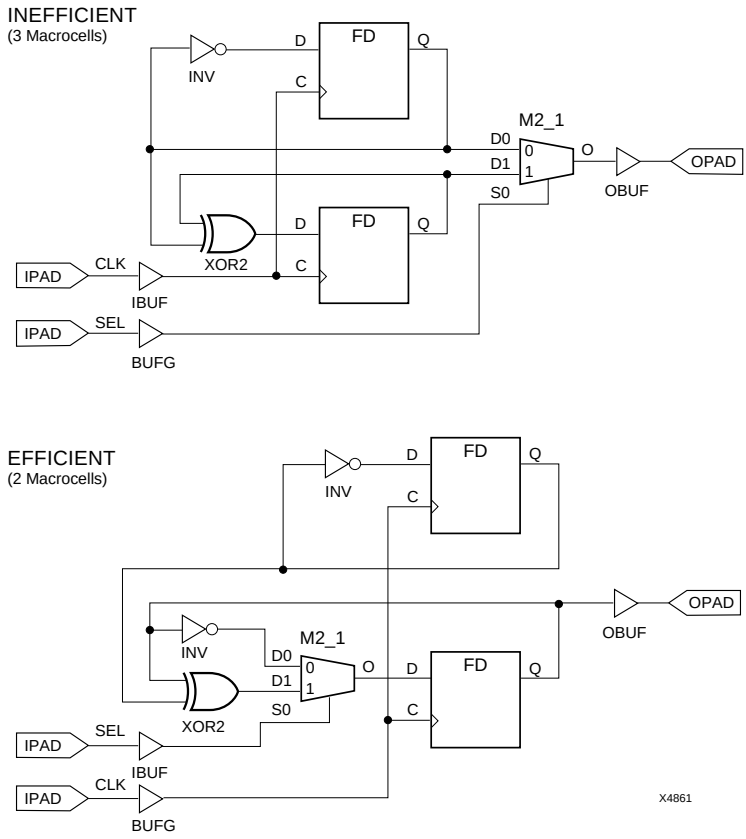


Figure 3-16 Reducing Levels of Logic

Timing Analysis

After you have fitted your design, you can use the timing analyzer to find and report the timing of paths through the device. You can select specific paths or general types of paths to examine, and you can choose from several report formats.

If you included time specification attributes in your design (also called TimeSpecs or T-Specs), you can use the Timing Analyzer to determine whether the XEPLD fitter was able to meet these specifications.

Timing Analysis Procedure

The typical procedure for using the EPLD timing analyzer is as follows:

1. Make sure the design you want to analyze is the current design in the Design Manager, then open the EPLD timing analyzer.
2. Choose the kind of report you want from the Analyze menu.
3. Use the report window commands to view the report in more detail, save the report, and print the report.
4. (Optional) If having all the paths in your design reported is more information than you need, select filters from the Path Filters menu to determine which types of paths will be analyzed and reported. You can "filter out" paths you are not interested in. Then repeat step 2.
5. (Optional) Select options from the Options menu to fine-tune the analysis even more. Then repeat step 2.

These steps are described in more detail in the following sections.

Opening the EPLD Timing Analyzer

Open the EPLD timing analyzer by double clicking on the Timing Analysis icon in the Tools subwindow of the main Design Manager window. Figure 3-17 shows the icon.

Figure 3-17 Timing Analyzer Icon

The EPLD Timing Analysis window appears. The design that is loaded into the timing analyzer is the current design established in the Design Manager. To load a different design into the timing analyzer, return to the Design Manager's main menu and open another project.

EPLD Timing Analysis Window Features

The EPLD Timing Analysis window looks like this:

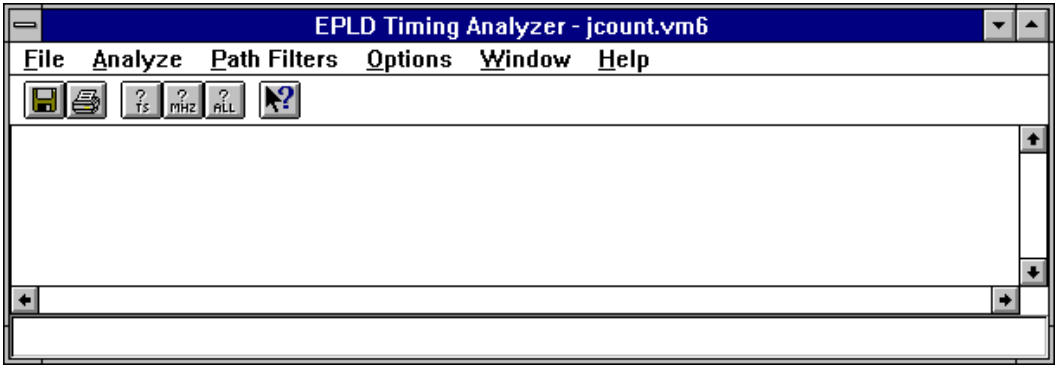


Figure 3-18 EPLD Timing Analysis Window

The Toolbar and Statusbar are optional; you can display or hide them using the Toolbar and Statusbar commands on the View menu. The Statusbar is displayed by default; the Toolbar is not.

The Window menu controls the display of report windows generated by commands on the Analyze menu. Because the report windows can be moved and sized independently of the main timing analyzer window, the Cascade and Tile commands arrange report windows below and to the right of the main window rather than within it.

However, when you collapse report windows, their icons appear in the main timing analyzer window, and you can use the Arrange Icons command to arrange the icons and line them up.

The Help menu is standard for Microsoft Windows.

Generating Reports

Selecting one of the commands on the Analyze menu is all you need to do to generate a report.

Each of these commands generates a different timing report. If you have previously chosen path filters or other options, the report reflects these choices. After a report is generated, it appears in a separate window, from which you can save and plot the report.

Designing for Density

The XEPLD software optimizes for density by default, but there are a few things you can do to improve density optimization:

- Maximally encode all state machines.
- Use global nets.
- Use the Master Reset pin if your design requires device reinitialization, or use this pin as a regular input if your design requires an additional I/O pin.

Maximally Encoding State Machines

If your design contains behavioral modules written as state machines, be sure that the functions are maximally encoded. This works best for EPLDs, which are rich in product terms. (For FPGAs, it is best to use one-hot encoding, because FPGAs are rich in registers.)

Using Global Nets

Clock and output enable signals mapped to global nets do not consume function block resources. Optimization software automatically maps the most used rising edge clock inputs to the FastCLK nets and the active-High output enable inputs to the FOE nets. Consider using active-high output enables and clock signals, and inverting the signals off-chip.

Master Reset Pin Trade-offs

This discussion describes trade-offs of using the XC7000 MR pin as a 3-state control. Consult your device data sheet for specific requirements of the MR pin during power-up.

The XC7000 devices feature a master reset pin that can be used to reinitialize the device. When the device is reinitialized, all device pins are 3-stated and registers are preloaded. When initialization is complete, the register preload is released and the outputs become operational.

The master reset pin can completely 3-state the device during board testing. The advantage is that no product terms or Function Block inputs are required for 3-state the device pins.

On the 7354, 7336, and 7318 devices, the master reset pin can also be programmed as a logic pin by assigning the MRINPUT=ON global attribute to a TBLOCK symbol. If this attribute is specified, the device is initialized only on power-up.

Designing to Preserve the Pinout

In the XC7000 devices, logic capacity and device pinout are determined only by the resources available in the Function Blocks; the logic mapped into the Function Blocks is always guaranteed to route in the UIM. The factors that determine the logic capacity of the Function Block are as follows:

- Number of Function Block inputs
- Number of product terms available to each macrocell in the Function Block
- Number of macrocells in the Function Block
- Number of device pins driven by the Function Block

Resource Reservation

Xilinx EPLDs are 100% routable due to the unique structure of the Universal Interconnection Matrix. This means that you can use all of the macrocells in a device. However, this section shows you how to reserve macrocells for design iteration, without changing your pinout.

To access the **Resources** options, do the following:

1. Open the Design Manager and select your design.
2. Next, you need to change the implementation of your design. Select **Implementation Template** by the following method:

   a) Select **Utilities** → **Template Manager.**

Figure 3-19 XC7300 Design Implementation Dialog Window

b) Select your design from the list and press **Edit.**

c) Click on Resources to get the pin and macrocell reservation template.

3. Change the values as needed.

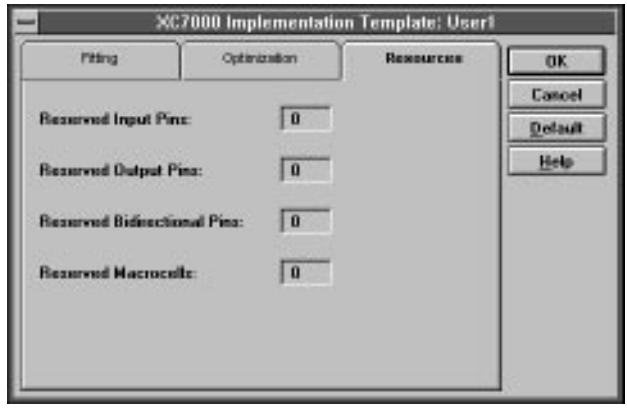


Figure 3-20 Resources Dialog Window

From here you can control the amount of device resources that will be left unused in each device. If your design uses more than one device,

these resource reservations are applied to each device; the resource reservations are not divided among the total number of devices used. The resource reservation options are:

- *Reserved Input Pins* represents the total number of Input pins including Fast Input pins that are reserved in each device.
- *Reserved Output Pins* represents the total number of output pins reserved in each device including those connected to Fast Function Blocks and those connected to High Density Function Blocks. The macrocell that drives the output pin is also reserved.
- *Reserved Bidirectional Pins* represents the total number of I/O pins reserved in each device including those connected to Fast Function Blocks and those connected to High Density Function Blocks. The macrocell that drives the output pin is also reserved.
- *Reserved Macrocells* represents the total number of macrocells reserved in each device including those macrocells set aside for reserved pins, those contained in Fast Function Blocks and those connected to High Density Function Blocks.

Using Pinouts from an Earlier Design Iteration

Pinouts from an iteration (version) of a design are saved automatically during processing. As long as the iteration is not deleted from the directory, a guide file containing pinouts from the design will exist. If you want to use the pinout of a earlier version to process a current revision, it can be selected before processing.

1. From the Design Manager, enter the Flow Engine. **Tools → Flow Engine**.
2. Select the down arrow adjacent to **Guide Design**. This will display a list of versions and revisions associated with the design.
3. Select the version and revision you want pinouts from and select **OK**. This will return you to the Flow Engine. When you **RUN** the design through, the pinouts from the selected revision will be used.

Manual Pin Assignment

Note: Manual pin assignment can restrict the layout capability of the software. It is usually best to allow XEPLD to automatically assign pins based on the most efficient placement of logic in the device.

XEPLD automatically assigns device pins for you, based on the most efficient usage of device resources. This is usually the best method for pin assignment if you do not have specific pinout requirements. Automatic pin assignment is performed only for those pins that have not been assigned through some other method.

If you have specific pinout requirements you can use the LOC=*pin_number* attribute to assign the signal to a specific pin.

Note: LOC attributes override the pin assignments in the guide file. This allows you to make changes to your fixed pin specifications. However, if you override the guide file with LOC attributes, the software will issue a warning. Also, trying to use LOC attributes to preserve a previously achieved pinout does not provide the software with sufficient history information (as does the guide file) to allow the software to successfully fit the design.

Manual Pin Assignment Precautions

When you manually assign output and I/O pins, you force the software to place logic functions into specific function blocks. If the logic does not exceed the function block resources (macrocells, product terms, and UIM inputs) and the function block has the correct external pin resources to meet the logic I/O requirements, the logic is mapped into the function block and the design will route in the UIM.

Try to place product term intensive logic onto pins that are driven by High Density Function Blocks. Be sure that the Function Block's shared product term resources and UIM inputs will not be exhausted. You may also wish to leave additional room in the Function Block for design iterations.

Assign your external rising-edge clocks and active-High output enable signals to the FastCLK and FOE pins on the device.

Evaluate the requirements of your logic assigned to pins that are driven by the Fast Function Blocks. Functions mapped to an FFB can

be clocked only by global clocks, 3-stated only by FOEs, and, for the 7354, 7372, and 73108 devices, only asynchronously set. Plan ahead for design iterations which may create functions that require the exported product terms from an adjacent macrocell.

The LOC Attribute

Use the LOC=*pin_name* attribute on a PAD symbol to assign the signal to a specific pin. The pin name is P*nn* for PC and PQ packages; the *nn* is a pin number. The pin name is *rc* (rowcolumn) for PG and BGA packages. Examples are LOC=P24 and LOC=G2.

You can apply the LOC attribute to as many PADs in your design as you like. However, each pin assignment further constrains the software as it automatically allocates logic and I/O resources to internal nodes and I/O pins with no LOC attributes.

Note: Pin assignment using the LOC attribute is not supported for bus components such as OBUF8.

Controlling Power Consumption

You control power consumption for specific macrocell outputs using the LOWPWR=ON attribute. This attribute is valid only for XC7300 designs. This attribute is either a global or component attribute.

The default is LOWPWR=OFF (high speed) for all macrocells used in the design unless otherwise specified in the Design Manager.

To make low power the global default power setting, set **`Low Power Mode`** to **`On`** in the **`Fitting`** menu of the **`Implementation Template`** in the Design Manager.

To set the template to use low power mode as set by attributes in the design, set the template to **`In Design`**.

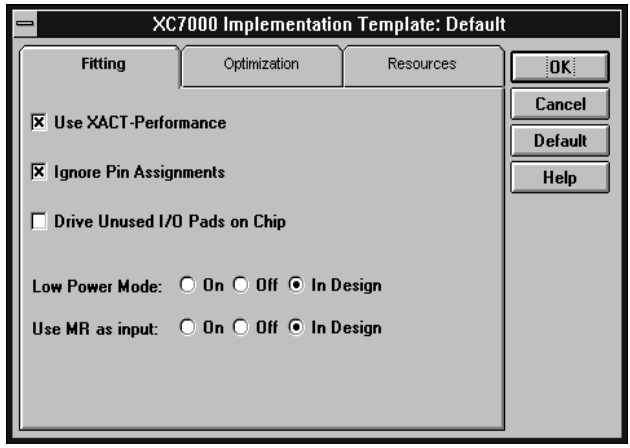


Figure 3-21 Low Power Mode Set to In Design

To control the power setting of the macrocells used by an individual symbol, use LOWPWR=ON or LOWPWR=OFF (unless global **`Low Power Mode`** was selected). This attribute is ignored if assigned to a symbol that uses no macrocells, such as an inverter.

Note: Low-power macrocells are slower than standard-power outputs. If you have a mixture of low- and standard-power macrocells, pay close attention to simulation results or the timing report to see how the power settings affect timing interactions.

Controlling Preload Values

The preload values used in the implementation of your design depend on these factors:

- INIT = R/S. The INIT attribute specifies the initialization value to be preloaded into a register upon power-up or Master Reset. INIT=R specifies a preload value of 0 (reset) and INIT=S specifies a preload value of 1 (set). This attribute can be applied to flip-flops or any component containing a register.
- Whether preload optimization is on. By default, the XEPLD fitter performs preload optimization, ignoring the library defaults, but honoring INIT attributes, to produce the most efficient mapping of components to available device resources. You can control

preload optimization by turning off **Pre Load** in the **Optimization** menu of the **Edit Template** in the Design Manager. To do this

- From the Design Manager, **Tools → Flow Engine**
- From the Flow Manager, **Setup → Options**
- Select **Edit Template**
- Select the **Optimization** template
- Place a check in the **off** circle next to **Pre Load**

Attributes for Controlling Preload Values

If your design is not sensitive to preload values, it is best to allow preload optimization, because this produces efficient results. However, if you want to control register preload states, you can prevent preload optimization as follows:

- Use the INIT=S (preload=1) and INIT=R (preload=0) attributes to specify the preload values of individual registers. The INIT attributes are always obeyed by the fitter, regardless of whether preload optimization is disabled.

Note: You cannot change the preload value of an input register to 0 using the INIT=R attribute because input registers physically do not support preload to the 0 state.

Preload Values for Functional and Timing Simulation

The only differences expected between functional and timing simulation involve the initial states of registers and latches in the design. Functional simulation assumes that preload values are undefined (X). Timing simulation uses the actual preload values implemented by the fitter.

For example, if an FDR component is mapped to a Fast Function Block, the FDR will appear to preload to X during functional simulation, because the value is undefined. However, during timing simulation, this FDR may preload to 1, because that is physical preload state of an FFB macrocell register where the FD component was mapped.

Chapter 4

Design Applications

This chapter describes some of the most useful techniques for making your EPLD design more efficient. These examples are suggestions and guidelines only, and may not apply to your particular design.

Reset and Preload Control in XC7000 FFB and Input Pad Registers

Use the following reset emulation technique to do these things:

- To allow logic to fit into a Fast Function Block register (for speed or because its an unused resource), emulate reset or clear.
- To spare macrocell registers when using input pad registers, emulate reset or clear.
- To change preload values in input pad or FFB registers to 0 (the preload value for these registers is normally 1).

Figure 4-1 shows how to set up reset emulation. An additional macrocell register from a High-Density Function Block provides the reset input. You AND the output of this HDFB register with the outputs of the FFB or input pad registers. These AND gates end up in the UIM, so there is no additional delay.

The XEPLD software automatically tries to map FD-type registers into input pad registers or FFB macrocell registers before it maps to HDFB macrocell registers. However, if you want to explicitly specify input pad registers, use IFD-type registers. To explicitly specify FFB registers, use the F attribute on the register outputs (see the "Attributes" appendix for more about the F attribute).

Note: If you are changing preload values, you do not need an HDFB macrocell register with a reset; a simple FD will work.

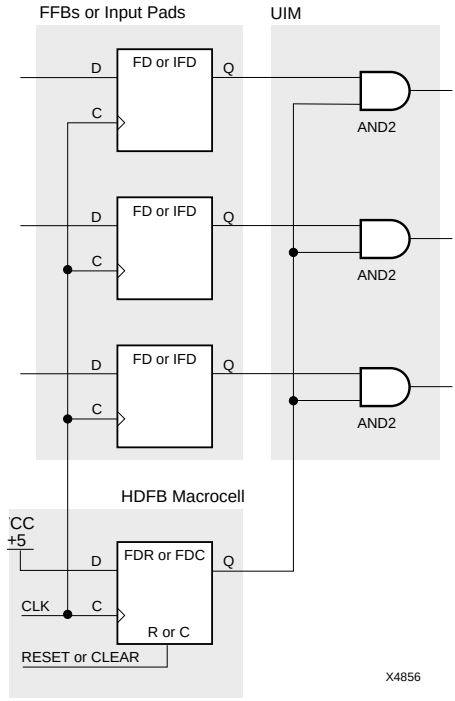


Figure 4-1 Reset and Clear Emulation for FFB or Input Registers

Read-Back Registers

Figure 4-2 shows a simple read-back register. Data is written from the IOPAD to the register on the rising edge of the clock if READ_ENABLE is inactive and WRITE_ENABLE is active. Data is read from the IOPAD when READ_ENABLE is active.

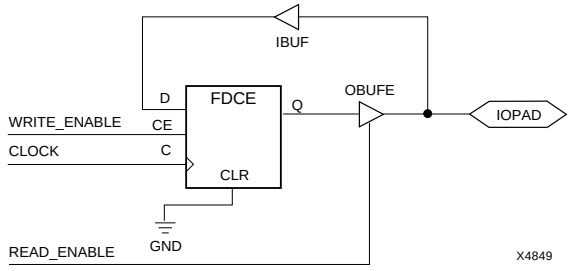


Figure 4-2 Read-Back Register Example

Bidirectional Signals and Buses

Figure 4-2A shows how to specify a bidirectional pin. Figure 4-2B shows that you can have a bidirectional signal passing through the chip. To make a bidirectional bus, use bus components as shown in Figure 4-2C.

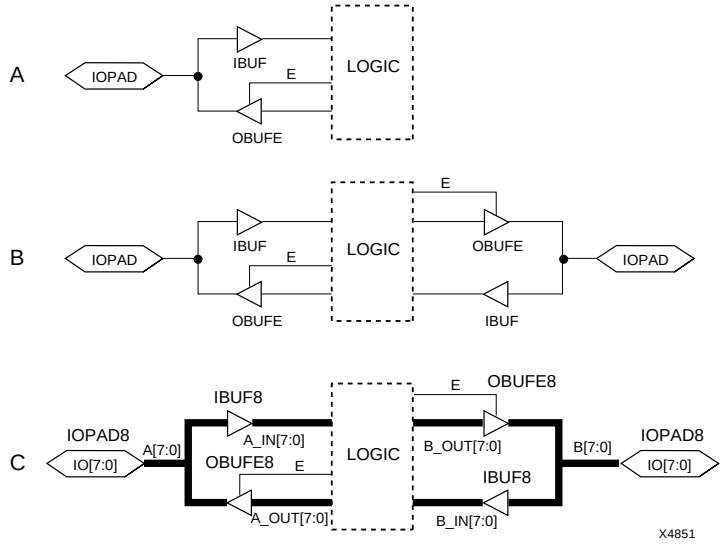


Figure 4-3 Bidirectional Signals and Buses

Bidirectional Signals in PLDs

If you want to use a PLD output with a TRST equation to control a bidirectional I/O pin of the EPLD, connect the OBUF output to an IOPAD and IBUF (or IFD/ILD). If the same PLD symbol that generates the output is also to receive the I/O pin input, you must use a separate pin of the PLD symbol to receive the signal from the IBUF. Do not tie the signal received from an IBUF to the net driving the OBUF of the same IOPAD as shown in Figure 4-3A; these input and output nets must remain separate as shown in Figure 4-3B.

Rules for connecting PLD symbols also apply to any custom symbols defined by equation files or macro schematics.

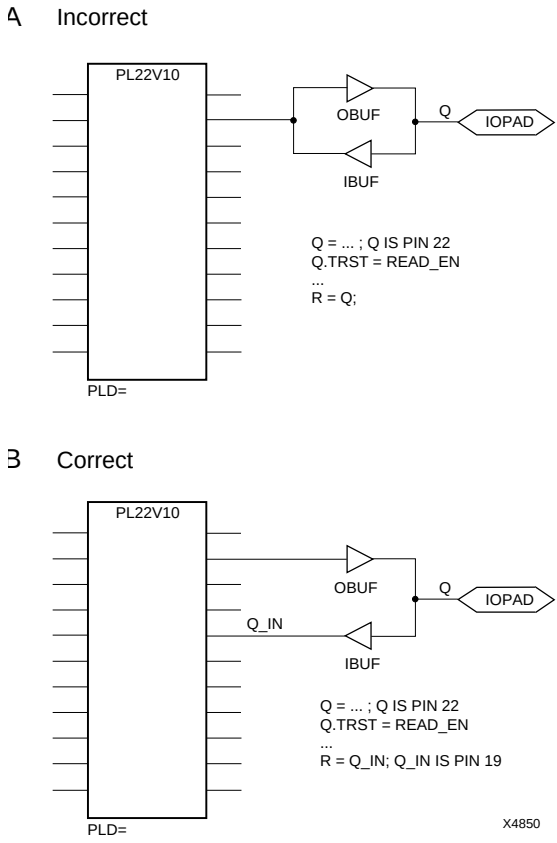


Figure 4-4 How to Control a Bidirectional PLD Pin

Multiplexing 3-State Signals

Three methods of multiplexing 3-state signals are shown in Figure 4-5 on the next page. Which method you choose depends on your application, resources, and speed requirements, although method C, which uses a multiplexer, is usually best for EPLD designs.

Method A, shown in Figure 4-5A, uses 3-state buffers instead of a multiplexer. The advantage of method A over method C is that method A uses only one Function Block input in the macrocell that sends the signal off-chip. The disadvantage of method A is that macrocell feedback is lost because the outputs are 3-stated; therefore counters will not work with Method A, but will work with Method C.

Method B, shown in Figure 4-5B, requires that you tie the signals together off-chip. This method results in a short clock-to-out delay and uses fewer macrocells than methods A and C. However, it uses more pins than method A or C.

Method C, shown in Figure 4-5C, uses a multiplexer instead of 3-state buffers. This method results in a longer clock-to-out delay than method B, although you can shorten this delay to that of method B by registering the output of the multiplexer and asserting the select signals one clock cycle in advance. This method uses more macrocells than method B, but uses fewer pins.

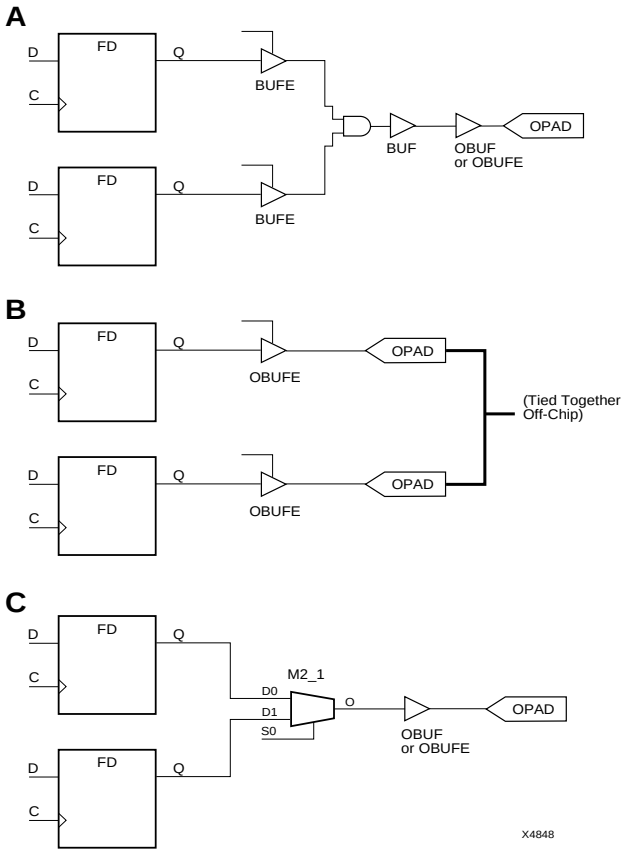


Figure 4-5 Methods of Multiplexing 3-State Signals

Optimizing XC7000 Registered Arithmetic Performance

The XEPLD software optimizes adders and subtracters into FD, FDC, and FDP registers. If your arithmetic component drives any other register type, the arithmetic and register functions are implemented in separate macrocells, impacting both speed and density.

The example in Figure 4-6 shows an adder/subtracter driving a register with clock enable and synchronous clear. When the logic in these components is broken down, each bitslice is represented as shown in Figure 4-7.

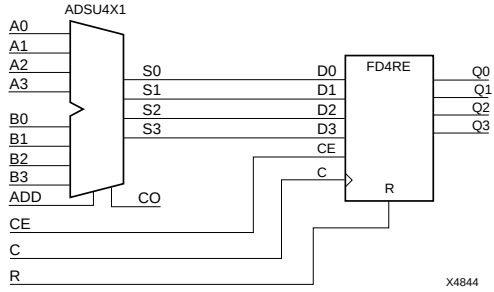


Figure 4-6 Using ADSU4X1 and FD4RE Library Components

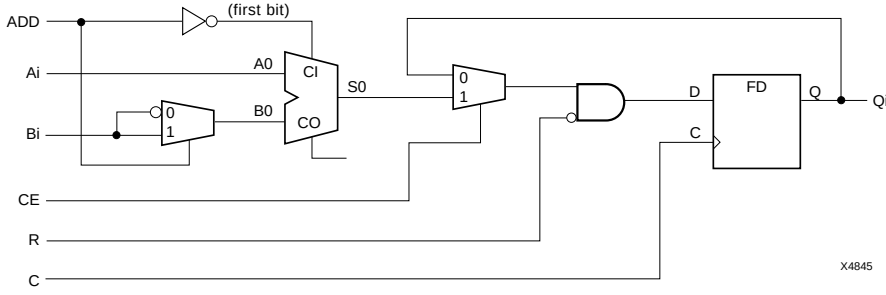


Figure 4-7 ADSU4X1 and FD4RE Equivalent Logic

In XC7000 High-Density Function Blocks, the arithmetic logic physically occurs just before the register, as shown in Figure 4-8. This means that, because of the reset and clock enable on the register, the

ADSU4X1 and FD4RE are implemented as two levels of logic in an XC7000 device.

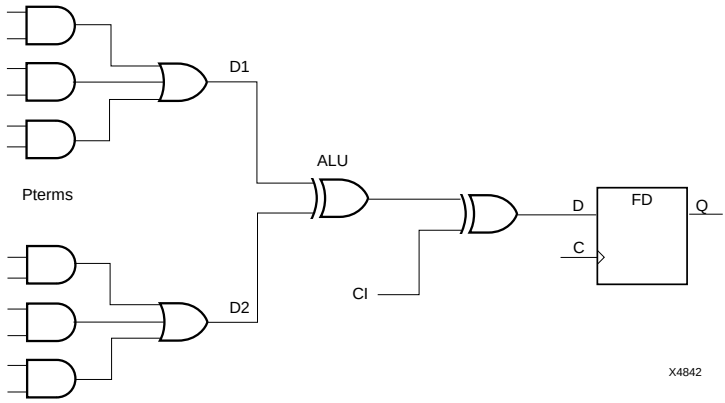


Figure 4-8 EPLD High-Density Function Block Architecture

Because the logic in the adder must be performed in the ALU block of the macrocell, the fitter cannot collapse the logic in Figure 4-7 into the same macrocell. As a result, the logic formation requires two macrocells and two macrocell delays.

You can achieve more efficient results by placing the register's reset and clock enable logic in front of the arithmetic logic as shown in Figure 4-9. Whenever you connect combinational logic to the inputs

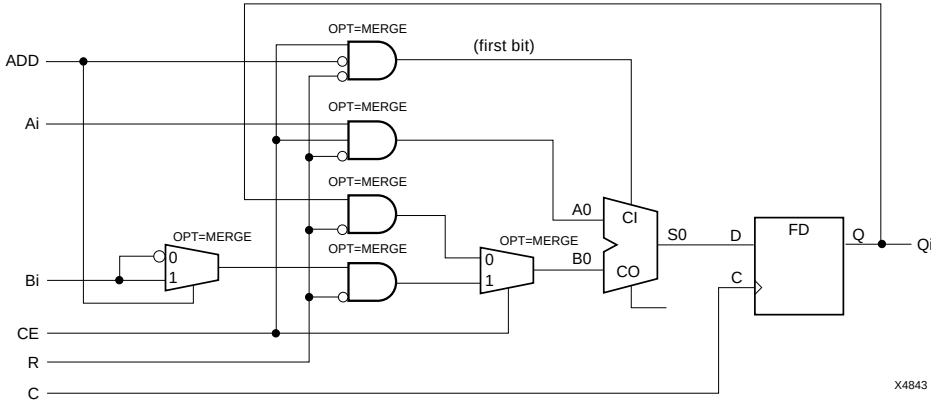


Figure 4-9 ADSUR4 Custom Symbol Logic Implementation

of an arithmetic component, you must place the OPT=MERGE attribute on each of the logic gates that you want implemented in the same macrocell as the arithmetic. Otherwise, the software will not automatically combine the combinational and the arithmetic functions because of the special mapping requirements of the EPLD arithmetic carry chain. Specifying OPT=MERGE forces the software to combine the logic. If you try to include more logic than can be fit into the macrocells, the software will issue an error.

Note: This is the only situation that requires the use of the OPT=MERGE attribute.

Combinational Feedback Loops

The simple expression of a D-type latch contains inherent logic hazards which could result in unpredictable results when run through the fitter.

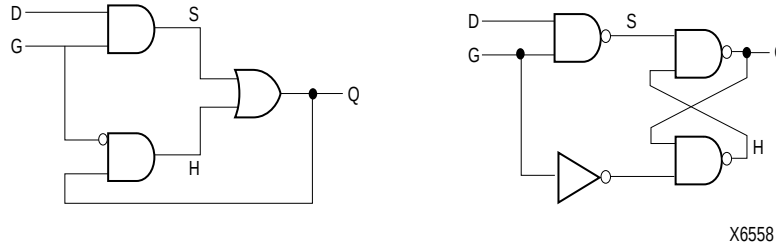


Figure 4-10 Simple Mux and Cross-Coupled-NAND Latches

A timing malfunction can occur if the logic is divided between two separate macrocells by the fitter. Figure 4-11 illustrates what can happen.

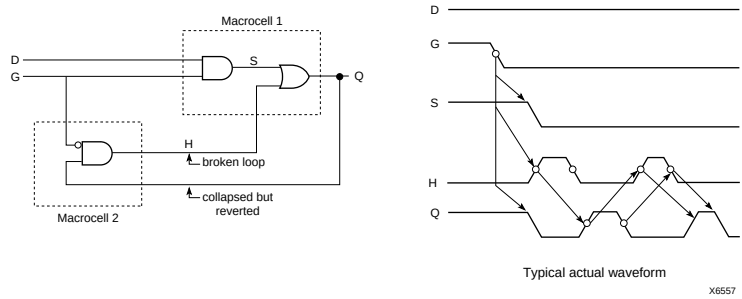


Figure 4-11 Malfunction of Physical Implementation

If you implement the D-type latches with proper redundant logic, the problem will not occur. Figure 4-12 shows two solutions for schematic implementation of D-type latches.

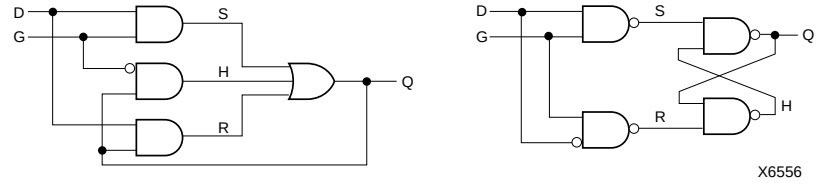


Figure 4-12 D-type Latch Solutions

When you create redundant logic in a schematic, remember to specify the MINUM=OFF attribute on the final output gate to prevent the software's Boolean minimization routine from removing the redundant logic.

Hierarchical Design

You can create symbols with schematics under them and place these symbols in your top-level schematic. This can make your design more modular and easier to understand.

User-created symbols are termed *custom components*. Custom components with schematics under them are termed *macros*, as opposed to *behavioral modules*, which are custom components with

equations under them. For information about creating behavioral modules, see "Using Behavioral Modules in Schematics."

The procedure for creating a symbol with an underlying schematic is the same for EPLD and FPGA except for the library you use (XC7000 instead of XC3000 or XC4000):

1. Create a lower-level schematic using XC7000 library symbols. To make a device-independent custom macro, use only device-independent symbols.
2. Create a symbol for the schematic.
3. Add attributes that the symbol needs to work in your CAE tool.

Notes for Viewlogic users:

- Label the nets in your lower-level schematic with the same names as the pins on the symbol.
- The block type of the symbol must be Composite (not Module). Use the **Change → Symbol Type → Composite** command to change this.
- If you copy a Xilinx-supplied library symbol to use as the basis for your custom macro, make sure you delete the invisible symbol attribute LEVEL=XILINX, as this marks the symbol as a primitive.

  Use **Change → Object Attribute → Dialog** to view and delete this attribute. If necessary, scroll through the window to find the `LEVEL=XILINX` attribute. Then mark the attribute with the mouse and then click on **Delete**.

Notes for OrCAD users:

- Label the module ports in your lower-level schematic with the same names as the pins on the symbol.
- Use the Edit Library (or LIBEDIT) utility to create your symbol.

Note: For information about storing custom components, see the "Using Behavioral Modules in Schematics" chapter. Custom components with underlying schematics are stored in the same way as custom components with underlying behavioral modules.

Schematic Custom Component Example

This next example shows you how to create a custom symbol with an underlying schematic. The steps for Viewlogic users are shown, with notes at the end for OrCAD users. Follow these steps:

1. Create the schematic using common symbols from the XC7000 library. It should look something like this:

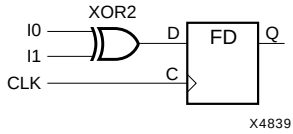


Figure 4-13 The REGXOR Schematic

2. Create a symbol with pin names that match the inputs and outputs of the schematic.

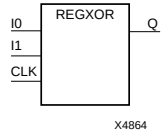


Figure 4-14 The REGXOR Symbol

3. Use the **Change → Symbol Type → Composite** command to change the symbol's block type to composite.

Notes for OrCAD users:

- Label the module ports in your lower-level schematic with the same names as the pins on the symbol.
- Use the Edit Library (or LIBEDIT) utility to create your symbol.

Chapter 5

Using Behavioral Modules in Schematics

This chapter discusses how to include behavioral (equation-based) modules in schematic designs. There are three reasons why you may want to use behavioral modules in your schematic:

- If portions of your design are already implemented using conventional programmable logic devices (PLDs), you can re-use your existing PLD equations without having to redraw the same logic schematically.
- Easier to express combinational functions and FSMs.
- You may wish to create a "custom primitive" symbol using equations (instead of a schematic-based "macro" symbol) because of the efficient sum-of-products logic equations. You can often achieve better logic density and performance for custom logic functions in an EPLD by using equation-based modules due to the inherent sum-of-products logic structures comprising the EPLD Function Block architecture. Custom primitives are easy to create, and can be used just like regular library components.

This chapter shows you how to use PLD symbols, create new components, edit library components, store custom components, and adapt behavioral logic to schematic designs.

This chapter includes design examples that use PAL devices in a schematic.

Preparing a Component

To prepare a PLD or custom primitive for use in a schematic design, follow these steps. After you have prepared your component, you can use it in a design just as you can any library component.

1. Create an ABEL or XABEL equation file. Name the file *symbolname*.abl.

   The MODULE statement in this file must specify the symbol name:

   ```
   MODULE symbolname
   ```

2. Compile the XABEL equation file to create a .xsf file, and a .pld file. From XABEL 5.2, the procedure is:

   Compile → Xilinx EPLD Netlist

Note: XABEL-CPLD 6.0 is for behavioral designs only, and will not generate a schematic module.

3. Use the Symbol Generation Utility to create a symbol from the .xsf file. On the PC, go to your XACT*step* program group and select the **Symbol Generation Utility** icon (if you are working from a workstation you can select **DesignEntry → SYMGEN**). From a PC:

   a) Select the **Symbol Generation Utility** icon.

   b) Use Browse to find and display the .xsf file on the **Input** line.

   c) Select Orcad or Viewlogic from **Symbol Type**.

   d) Select **OK**.

4. Add attributes that the symbol needs to work in your CAE tool.

   For Viewlogic symbols, SymGen automatically adds the LEVEL=XILINX symbol attribute to mark the symbol as a primitive.

   When you use the symbol in a schematic, add the DEF=PLD and FILE=*symbolname* attributes to the symbol instance (do not add them using the symbol editor).

Behavioral Module Example

This first simple example shows you how to create a custom symbol with an underlying equation file. The steps for Viewlogic and OrCAD users are shown. Follow these steps:

1. Create the XABEL file, regxor.abl.

   ```
   MODULE regxor
   ```

```
TITLE ‘Registered XOR gate’
regxor device;
IO pin;
I1 pin;
CLK pin;
Q pin istype ‘reg’;
EQUATIONS
Q := IO $ I1;
Q.C = CLK;
end
```

2. Compile the file to create the .xsf and .pld files.

   Compile → Xilinx EPLD Netlist

3. Run SymGen by selecting the **Symbol Generation Utility** icon from the XACT*step* program group. Create the symbol below.

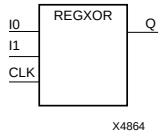


Figure 5-1 The REGXOR Symbol Created by SymGen

   If you are an OrCAD user, you must perform additional steps to prepare your symbol. See “Using SymGen to Create Custom Symbols” later in this chapter for details.

4. You can turn off the display of the LEVEL=XILINX attribute and other attributes using the **Change → Object Attributes → Visibility → All Attrs Off** command (for Viewlogic).

5. Add the DEF=PLD and FILE=regxor attributes to the symbol instance when you use the symbol in a schematic.

Using Xilinx ABEL

Use Xilinx ABEL 5.2 (or 5.1) for creating behavior modules for use in schematic design. The example used earlier in this chapter illustrates the process of using Xilinx ABEL to create a schematic symbol that can be placed in a Viewlogic or OrCAD schematic.

Xilinx ABEL is useful for developing behavioral modules if you want to take advantage of its high-level language capability but do not need to access device-specific features such as the high-speed carry path.

The SymGen Utility (Symbol Generation Utility) reads a Xilinx ABEL-generated or user-created .xsf file containing the symbol name and input and output names, then creates a macro file for OrCAD and a symbol file for Viewlogic. The OrCAD Draft schematic editor reads this macro file and creates a functional block that references a Xilinx ABEL-created .xnf file. Viewlogic Viewdraw reads the symbol and incorporates it into the schematic. For more information see the *Xilinx ABEL User Guide*.

Using SymGen to Create Custom Symbols

Behavioral modules use the Symbol Generation Utility (Symgen) to create a custom symbol. SymGen processes an .XSF:

SymGen is executed by selecting the **Symbol Generation Utility** icon from the XACT*step* program group.

SymGen also produces a report, *design_name*.SMR, which explains how the symbol was created and displays a diagram of the pinout. The Symbol Generation Utility menu allows you to specify generation of a .smr file.

Viewlogic Symbols

If you are a Viewlogic user, SymGen creates the symbol and places it in the sym directory below your design directory. You can add it to any schematic in your design just as you would any other symbol. No special conversion steps are necessary.

OrCAD Symbols

If you use the Symbol Generation Utility with OrCAD selected as the **Symbol Type**, SymGen creates a .CMD file and places it in your design directory. To convert the .CMD file into a symbol, you must perform these additional steps:

1. Enter the OrCAD Edit Library utility.

   File → New → Library

2. In response to the `Read Library?` prompt, type the following:

   `.\userlib.lib`↵

   This creates a library called userlib in your design directory.

3. The Library Edit screen appears. Select the **`Import`** command and type *file_name* **`.cmd`**↵ to invoke the command file.

4. The symbol appears. Select **`Library`** → **`Update Current`** to save the symbol to memory.

5. Select **`Quit`** → **`Update File`** to save the library to disk.

6. Select **`Abandon Edits`** to exit the library editor.

You can now add this symbol to any schematic in your design just as you would any other symbol.

Storing Custom Components

After you create your custom component, you should store it in each design directory where you need to access it.

If you have defined the underlying logic differently for targeting two or more different device families, you should store the component in two or more different project directories or library directories. Each directory would contain the underlying logic for one device family.

Viewlogic Components

You should store your custom library files in your project directory. You cannot add custom symbols to the XC7000 library directory or modify any of the Xilinx-supplied symbols or macros. However, you can copy Xilinx-supplied symbols or macros to your directory, rename them, and edit them.

OrCAD Components

You can store your library file and your macro schematics under the \XACT\XC7000 library directory. Do not add to or modify the xc7000.lib file or any of the library macro schematics supplied by Xilinx. Store equation files for custom primitives in each design directory for which you want to use the component.

Chapter 6

Design Verification

This chapter describes the simulation methods, and reports available to help you analyze and verify your design.

Simulating Designs

XEPLD supports a variety of third-party simulators, allowing you to perform functional or timing simulation of your finished design.

To perform simulation on a design, you must first translate it into a netlist consisting of XC7000 library models. XEPLD automatically creates simulation files in the XNF netlist format, which can be exported to the Viewlogic ViewSim simulator (.WIR), or the OrCAD simulator (.VST) using the Xilinx-supplied CAE tool interfaces and libraries. You can also use .XNF files with other simulators that support Xilinx.

Note: When XEPLD processes your design, some of your original nodes may be removed due to circuit optimization. These nodes cannot be viewed or stimulated. All of the external I/O signals are always maintained.

Making a Device Functional Simulation Model in ProSim or VST

To create a Viewlogic functional simulation model, select the **`Xilinx PROflow`** icon and select the **`ProSim`** button adjacent to **`Functional Simulation`**. This will run XSIMMAKE and generate a report in the notepad. When you exit the notepad, PROsim appears. If you want to view waveforms, select the **`ProWave`** button. You may also run a function simulation by selecting the **`Simulation Utility`** from the XACT*step* icon group.

To create an OrCAD functional simulation model, select the **Simulation Utility** from the XACT*step* icon group. The simulation menu appears. Use the **Browse** button to locate and select your schematic file (`design.sch`). Select the XC7000 family. Under **Output Data**, select **Produce Functional Simulation File**. Under **Format**, select **OrCAD**. Press the **Run** button to generate a report file.

See the *OrCAD Interface/Tutorial Guide*, *Viewlogic Interface Guide* and *Viewlogic Tutorials* for more information on simulation.

Making a Device Timing Simulation Model in ViewSim or VST

You must successfully fit your design to an EPLD using the Design Manager before proceeding with timing simulation

Viewlogic Procedure

To create a Viewlogic timing simulation model, follow these steps; for all commands, use the default options:'

1. Select the **Xilinx PROflow** icon from the **XACT*step*** icon group.
2. Select the **ProSim** button adjacent to Timing Simulation.
3. Place a check in the **Command File** box and click on **Browse**. Find the command file for your design and click on **OK**. Timing simulation will run and display a report.

See the *Viewlogic Interface Guide* and *Viewlogic Tutorials* for more information on simulation.

OrCAD Procedure

To create an OrCAD timing simulation model, follow these steps; for all commands, use the default options:

1. Select the **Simulation Utility** from the **XACT*step*** icon group.
2. Use the **Browse** button to locate and select your schematic file (`design.sch`). Select the XC7000 family.
3. Under **Output Data**, select **Produce Timing Simulation File**.

4. Under **Format**, select **OrCAD**.
5. Select your file name from the list of .sch files that are displayed. Select the **Run** button.

   XEPLD creates a *design_name*.VST file.

See the *OrCAD Interface/Tutorial Guide* for more information.

Preload Values in Functional and Timing Simulation

The only functional differences expected between functional and timing simulation involve the initial states of registers and latches in the design. Functional simulation assumes that preload values are undefined unless specified by an INIT attribute; timing simulation uses the actual preload values implemented by the fitter.

For example, an FDR component will appear to preload to X during functional simulation. However, during timing simulation, this FDR may preload to 1 because that is the physical preload state of an FFB macrocell register where the FD component was mapped.

See the "EPLD Architecture and Design Tradeoffs" chapter and the PRELOAD_OPT and INIT attribute descriptions in the "Attributes" appendix for more information about preload values.

Verifying Designs

After you have compiled your design using the Flow Engine, XEPLD generates reports that tell you how your design fits in the target device and how fast the design will run.

- The Resource Report, *design_name*.RES, gives you a summary of the logic utilization of the device, your I/O usage, and the resources that were left unused.
- The Pinlist Report *design_name*.PIN shows the final XC7000 device pinout of your design.
- The Timing Report *design_name*.TIM shows the calculated worst-case timing based on the physical implementation of your design.

Verifying Design Fit

When XEPLD has successfully compiled your design, you will see the following message on your screen:

```
Design Successfully Mapped. Examine the following
report files:
```

Examine the Resource Report to determine the amount of chip resources used to implement your design and how much remain. An example Resource Report is shown on the next page. The schematic for this report is the Johnson counter example in "Getting Started with Schematic Design." This design was targeted for the XC7318-5PC44.

The Logic Resources section of the Resource Report shows that 4 macrocells were used in the design and 14 remain available for additional logic. The Pin Resources section shows the types of signals required by the design, the types of device pins used to satisfy the signal requirements, and the remaining device pins that can be used for additional signals.

This report shows that the 2 input signals were placed on input pins, 3 of the 4 output signals were placed on I/O pins and the remaining output was placed on an FOE pin. A FastCLK pin was also used. A total of 28 pins (14 input and 14 I/O) remain available for additional input signals.

```
XEPLD, Version 6.0.0                                    Xilinx Inc.
                          Resource Report
 Circuit name: JCOUNT
Target Device: XC7318-5PC44        Integrated:  8- 9-95, 11:38AM


LOGIC RESOURCES

                 Required  Used      Remaining
Function Blocks     1         1         1
Macrocells          4         4        14


PIN RESOURCES:

Type   Req   --------Used--------------  --------Remaining------
             I  O  I/O Fclk Foe Cen Tot  I  O  I/O Fclk Foe CenTot
Inputs   2   2       0                 2 14      14                28
Outputs  4      0    3    0   1   0    4     0   14    0   0   0  14
I/Os     0           0                 0         14                14
Fclks    1                1            1                1           1
Foes     0                    0        0                    1       1
Cens     0                         0   0                        0   0
       --- --- --  ---  --- --- --- ---
         7   2  0    3    1   1   0   7


Note:The design requires 0 pins with Fast Input capability.
     This device has 11 pins with Fast Input capability.

 End of Resource Report
```

Common Questions and Answers

This appendix lists frequently asked questions about EPLD software and its CAE tool interfaces, and gives explanations and solutions.

Drawing the Design

This section lists problems you may encounter because your CAE tool drawing package is not properly configured for XEPLD software.

Why Do I See White Boxes Instead of Components?

If you are a Viewlogic user and your schematic contains symbols from a device family library (such as XC7000) that is not included in your viewdraw.ini file, you see white boxes when you view your schematic.

A likely cause of this problem is forgetting to run the Altran program when converting from one device family to another; see the "Device-Independent Design" chapter for details. Even after you run Altran, components from the old library that are not in the new library appear as white boxes — you should find equivalent components that are compatible with the new library.

Another likely cause is not configuring viewdraw.ini properly, with correct pathnames and library aliases. The example in the "Getting Started with Schematic Design" chapter includes information about how to configure Viewlogic software for the XC7000 device family.

Why Are Some of My Components Missing?

If you are an OrCAD user and your schematic contains symbols from a device family library (such as XC7000) for which OrCAD is not configured, you see missing components when you view your schematic.

You probably were converting this design to a new device family and forgot to do one of the following:

- Configure OrCAD using the XDraft command.
- Substitute symbols compatible with the new library for symbols compatible with the old library.

The example in the "Getting Started with Schematic Design" chapter includes information about how to configure OrCAD software for the XC7000 device family.

Fitting the Design

This section lists problems you may encounter when you fit the design.

What Does "Unrecognized Symbol" Mean?

If you get this error message:

```
xr55: [Warning]'ADECODE is an unrecognized symbol
for the EPLD family. If this symbol is a
behavioral module, make sure you have added
FILE=module and DEF=PLD attributes to the symbol.
If the symbol is a custom schematic component,
check that a schematic exists for it. If the
symbol is a standard library component, make sure
the target EPLD family supports it. If you are
using a synthesis tool, resynthesize th logic
targeting an EPLD family.
```

it means one of the following occurred:

- You did not properly link a behavioral design to a schematic symbol. You need to add attributes FILE=*filename* and DEF=PLD to the schematic symbol.

- If the symbol is a user macro, check that a schematic exists for it and that an .XNF file was generated.
- If the symbol is a Xilinx library component, make sure the target EPLD family supports it.
- If you are using a synthesis tool, recompile the logic targeting an EPLD family.

Simulating the Design

This section lists problems you may encounter during functional or timing simulation.

Why Are My Registers Stuck at the Preload Value?

At the beginning of a simulation, you must either pulse the MRESET signal Low then High, or pulse the PRLD signal, which is MRESET inverted, High then Low.

The assertion of the pulse forces all registers to a known state. The deassertion allows the registers to change states. If you do not deassert MRESET (or PRLD), your registers cannot change state.

Why Are My Internal Nodes Not Visible During Timing Simulation?

The EPLD fitter optimizes your design for efficiency, eliminating many internal combinatorial nodes. (If you are a Viewlogic user and you view your back-annotated schematic during timing simulation, these nodes appear with a "?".)

Nets between IPADs and IBUFs are observable, as are nets between OBUFs and OPADs. However, the outputs of IBUFs and the inputs of OBUFs may not be observable. Other nets may or may not be observable, depending on the results of optimization.

Why Do Functional and Timing Simulation Yield Different Results?

The only differences expected between functional and timing simulation involve the initial states of registers and latches in the design. Functional simulation assumes that preload values are undefined (X) unless specified using an INIT attribute. Timing simulation uses the actual preload values implemented by the fitter.

See the "EPLD Architecture and Design Tradeoffs" chapter and the PRELOAD_OPT and INIT attribute descriptions in the "Attributes" appendix for more information about preload values.

Appendix B

Attributes

There are several methods of controlling the software using attributes. There are Global Attributes which enable you to control the implementation process from a global level, and there are Component Attributes which allow you to control implementation at a detailed level. Global Attributes can be defined either on the schematic or from the Design Manager.

Attributes, which you place in your schematic, allow you to control the following aspects of how the software processes your design:

- Target device
- Linking of behavioral module symbols and their behavioral description files
- Register initial state
- Pinout and internal logic placement
- Power consumption
- Output pin slew-rate
- Timing constraints

Attributes are used to express information specific to each design, as opposed to run-time options entered through the Design Manager interface. There are two ways that attributes are placed in the schematic:

- Component attributes, such as FILE, OPT, and LOC, affect only the component instances on which they are placed.
- Global attributes set from the Implementation Template of the Design Manager.

Component Attributes

The component attributes specific to EPLD designs are as follows:

- DEF=PLD
- FAST
- FILE=*file_name*
- INIT={R | S}
- LOC=*pin_name*
- LOWPWR={ON | OFF}
- MINIM
- OPT={MERGE | OFF | UIM}

Viewlogic Procedure

Use the **Add → Object Attribute** command to assign a component attribute by using the following procedure:

1. Select the schematic component by placing the mouse arrow on the component and clicking the left mouse button.
2. Select **Add → Object Attribute.**
3. Type the attribute string, for example **LOWPWR=OFF**. Press Enter.
4. Move the arrow to where you would like the text and click the left mouse button.
5. If you are assigning more than one attribute to the same component, repeat steps 2 through 4 for each attribute.

OrCAD Procedure

Use the OrCAD **Edit** command to assign a component attribute by using the following procedure:

1. Position the cursor over the schematic symbol.
2. Select **Edit**.
3. Select **Edit**(again).
4. Select ***n*th Part Field**, where *n* is any part field you choose from 1st to 8th.

5. In the Name field enter:

   attribute_name=*value*

Implementation Template (for Global Attributes)

The Design Manager has an **`Implementation Template`** containing options for global attributes. Global attributes have the following selection options:

- *On*: The attribute is applied throughout the whole design.
- *Off*: The attribute is turned off.
- *In Design*: This is provided only for backward compatibility to earlier versions of the fitter in which global attributes were specified in the schematic itself.

Note: Attributes specified in the Design Manager's **`Implementation Template`** override any old-style global attributes placed in the schematic designs unless the **`In Design`** option is selected.

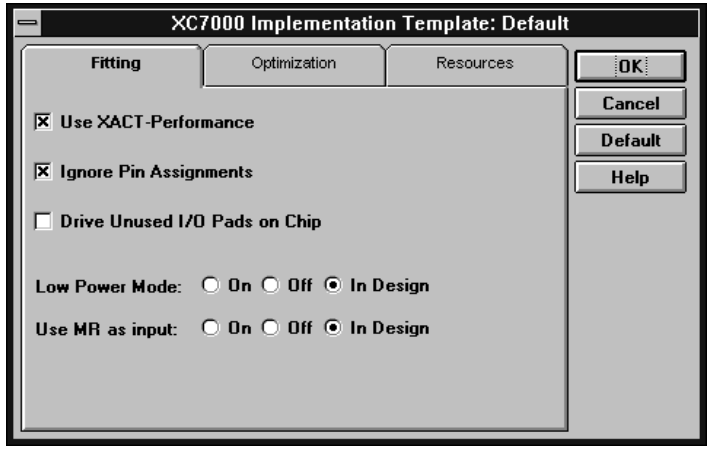


Figure B-1 Implementation Template

To get to the Implementation Template from the Design Manager:

`Utilities → Template Manager`

Click on the name of the an implementation template and select:

Edit

From the Fitting template you can control the fitting options which are set as follows:

- *Use XACT-Performance*: This indicates that you want to use timing driven optimization which causes the software to optimize your design to meet your specified timing constraints. For this option to be useful, you must have previously created a timing constraints file or you must have placed T-Spec information on your schematic. See Chapter 4 for information on using T-Specs.

Note: Processing time will potentially be longer.

  If this option is turned off, any timing specification attributes in the schematic are ignored.

- *Ignore Pin Assignments*: This indicates that you do not want to use any pinout information that may be in the design file or in a guide (.gyd) file. This allows the fitter to place pins anywhere.

  If this option is turned on, any LOC attributes in the schematic are ignored.

- *Drive Unused I/O Pads On Chip*: This indicates that you want all unused I/O pads to be actively driven by on chip circuitry. Because all unused I/O pads must be driven to a valid logic level, this option alleviates the need for external drivers or pullup resistors. This option should not be used with the XC7318 and XC7336 devices.

- *Low Power Mode*: Specifies the default power consumption mode for the whole design. The default can be overridden using the LOWPWR=ON or LOWPWR=OFF component attributes in the schematic:

  - On — Set Low Power mode as the default for the entire design.
  - Off — Set high performance (high power mode) as the default for the entire design.
  - In Design — Default power may be defined by the old-style LOWPWR=ALL global attribute in the schematic.

- *Use MR As Input*: Controls the use of the MR Input pin on the XC7318, XC7336, and XC7354:

- On — Allow the MR pin to be used as an input pin.
- Off — Use the MR pin as a Master Reset input pin.
- In Design — Allow the pin use to be determined by the old-style MRINPUT=ON global attribute in the schematic.

Click OK to accept the template, click Cancel to return to the previous menu, click Default to set the default options, or click Help to open the on-line help system.

Click Optimization and you will see the optimization template as shown in Figure B-2.

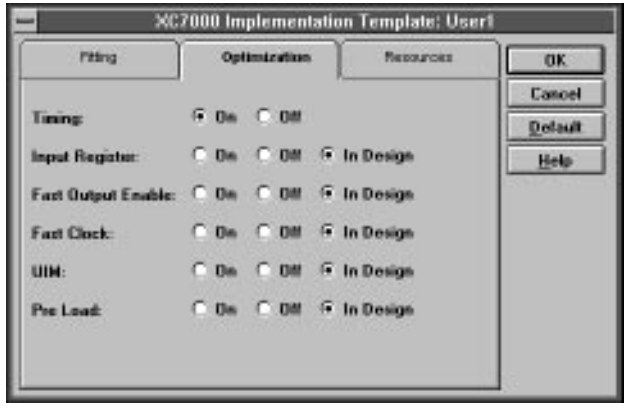


Figure B-2 Optimization Template

From the Optimization template you can control the optimization options which are set as follows:

- On — The option is activated for the whole design.
- Off — The option is turned off for the whole design.
- In Design — The option is provided for backward compatibility to earlier versions of the fitter in which global attributes were specified in the schematic itself.

The options are:

- Timing— Optimizes all logic paths for speed to achieve the fastest possible timing.

- Input Register — Determines if simple registers in your design can be implemented using IOB registers of the EPLD.
- Fast Output Enable — Determines if output enable signals can use the global high speed FOE nets.
- Fast Clock — Determines if clocks can use the global FastClock nets.
- UIM — Determines if the UIM can be used to implement basic boolean logic functions.
- Pre Load — Determines if the fitter can optimize registers that have no assigned preload values. This optimization allows the software to choose preload values that simplify fitting.

Click OK to accept the template, click Cancel to return to the previous menu, click Default to set the default options, or click Help to open the on-line help system.

Click Resources and you see the Resource Reservation template as shown in Figure B-3.

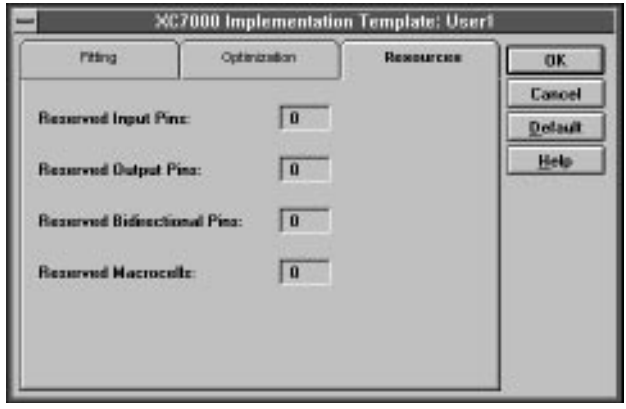


Figure B-3 Resources Template

From the Resources template you can control the amount of device resources reserved for future use as follows:

- Reserved Input Pins — Specifies the number of input pins to leave unused.

- Reserved I/O Pins — Specifies the number of I/O pins to leave unused.
- Reserved Output Pins — Specifies the number of Output pins to leave unused.
- Reserved Macro Cells — Specifies the number of Macro Cells to leave unused.

Click OK to accept the template, click Cancel to return to the previous menu, click Default to set the default options, or click Help to open the on-line help system.

Target Device Selection — The PART Attribute

You can place the global PART attribute in your schematic to select the target EPLD device for your design. Refer to the Release Notes or the Design Manager Part menu for a list of EPLD device names supported by the software.

Note: This attribute is called PARTTYPE in OrCAD.

Selecting a part type in the Design Manger menu other than InDesign before invoking the Fitter overrides any PART attribute in your schematic.

Note: The Design Manager will automatically select a part for you, choosing, in general, the smallest part that will satisfy the needs and constraints of your design.

The format of the PART value is as follows:

PART=*dddd*-*sspppp*

dddd is the device number, for example 7354

ss is the speed grade, for example 12

pppp is the package type and pin count, for example PC68

Viewlogic Procedure

Apply the PART attribute as an unattached schematic attribute. Follow these steps:

1. Deselect all components by clicking on a blank area of your schematic.

2. Select the **Add → Object Attribute** command.
3. Type the PART attribute string and press enter.
4. Position the attribute anywhere in the schematic and click with the left mouse button.

OrCAD Procedure

Unlike other global attributes used for EPLD designs, the PARTTYPE attribute is a stand-alone text string and should not be placed beneath the |GLOBAL keyword in the schematic. Simply place the text

|PARTTYPE=*dddd-ssppppp*

anywhere in your schematic other than the GLOBAL attribute list.

Behavioral Module File Name — The FILE Attribute

The FILE=*file_name* and DEF=PLD attributes on a PLD symbol specify the name of the file with the logic equations for that PLD.

Specify the directory path if necessary. Do not specify the file extension in the FILE=*file_name* attribute.

Pin Assignment — The LOC Attribute

Use the LOC=*pin_name* attribute on a PAD symbol to assign the signal to a specific pin. The PAD symbols are IPAD, OPAD, IOPAD, and UPAD. The pin name is P*nn* for PC packages; the *nn* is a pin number. The pin name is *rc* (rowcolumn) for PG packages. Examples are LOC=P24 and LOC=G2.

Pin assignments are unconditional in that the software will not attempt to relocate a pin if it cannot achieve the specified assignment. You can apply the LOC attribute to as many PADs in your design as you like. However, each pin assignment further constrains the software as it automatically allocates logic and I/O resources to internal nodes and I/O pins with no LOC attributes.

To assign logic to a specific macrocell of a specific function block, use the LOC=FB*x_y* attribute, where *x* is the function block and *y* is the macrocell. For instance, LOC=FB3_4 places the signal from the logic to which it is attached into macrocell 4 of function block 3.

All pin assignments for a revision are save in a .gyd file (guide file). If you want to create a new revision of a design and use the pin assignments from a previous version, go to the **`Implementation Menu`** and use the **`Guide Design`** option to pick up the guide file from the earlier version.

Note: Pin assignment using the LOC attribute is not supported for bus components such as OBUF8.

Power Setting — The LOWPWR Attribute

This attribute is valid for XC7300 designs, and is not valid for XC7200 designs. You can use this attribute to override the default power setting in the Design Manager **`Implementation Window.`** Normally the low power default is set to **`OFF.`**

To make low power the global default power setting, go to the **`Implementation Template`** (in the Design Manager), select the **`Fitting`** menu, and select **`On`** next to **`Low Power Mode`**.

To set the power of macrocells used by an individual symbol, use the LOWPWR=ON or LOWPWR=OFF component attribute. This attribute is ignored if assigned to a symbol that uses no macrocells, such as an inverter or an I/O buffer.

Logic Optimization Attributes

Use the logic optimization attributes to control optimization at specific points in your design. Logic optimization attributes are normally not required to process EPLD designs.

OPT=OFF

The OPT=OFF component attribute inhibits logic optimization of all macrocells used by a symbol.

The logic optimizer collapses the levels of logic to remove intermediate nodes. Components are optimized forward into components connected to their outputs.

If you build combinational logic using low-level gates and multiplexers, the software attempts to pack all logic bounded between device I/O pins and registers into a single macrocell.

The logic optimizer first removes all internal logic that is not used by any other logic or output buffer.

The logic optimizer moves logic forward by collapsing combinational expressions into their fanouts. If collapsing an expression into all fanouts succeeds, the original macrocell logic becomes unused and is removed.

The logic optimizer does not collapse an expression into its fanouts if the resulting expression uses too many product terms or inputs. Also, logic is not automatically collapsed forward into any arithmetic components (see the OPT=MERGE attribute).

The logic optimizer also moves forward any logic, whether combinational or sequential, that is buffered by a tri-state buffer. However, logic that itself contains a tri-state control is not moved forward.

The OPT attribute has no effect on any symbol that contains no macrocell logic, such as an I/O buffer.

The OPT=OFF attribute can be used to prevent optimization if it appears that the software is collapsing logic in a way that prevents a successful fit.

OPT=MERGE

Whenever you connect combinational logic to the inputs of an arithmetic component, you must place the OPT=MERGE attribute on each of the logic gates that you want implemented in the same macrocell as the arithmetic. Specifying OPT=MERGE forces the software to combine the logic. Otherwise, because of the special mapping requirements of the EPLD arithmetic carry chain, the software will not automatically combine the combinational and the arithmetic functions . If you try to include more logic that can be fit into the macrocells, the software will issue an error.

INIT

The INIT attribute specifies the intialization value to be preloaded into a register upon power-up or Master Reset. INIT=R specifies a preload value of 0 (Reset) and INIT=S specifies a preload value of 1 (Set). This attribute can be applied to flip-flops or any component

containing a register. However, INIT=R cannot be specified for an IOB register, or latch symbols such as IFD.

FAST

In devices that support output slew-rate control, the FAST attribute can be placed on an OPAD (output pad) or IOPAD symbol (primitive or macro) to select the fast slew-rate operation of the corresponding EPLD output-pin driver. The default is reduced (slower) slew-rate, which reduces output switching surges in the device.

MINIM

The MINIM=OFF attribute tells the fitter to disable Boolean logic minimization for the attached component. You need to use the MINIM=OFF attribute if you want to specify redundant logic in a portion of your design to avoid a potential race condition; for example, you would use MINIM=OFF when designing combinational feedback loops and latches.

TIMESPEC Attribute Syntax

The TIMESPEC attribute definitions specify the maximum delay between groups of components. They begin with the letters "TS" and a unique identifier that can consist of letters, numbers, or the underscore character (_). The value of the TIMESPEC attribute consists of a FROM-TO expression specifying the timing requirements between specific end points. The full syntax is shown as follows:

TS*identifier***=FROM:***group1***:TO:***group2***=***delay*

The parameters *group1* and *group2* can be any of the following:

- Predefined groups consisting of FFS, LATCHES, or PADS which are discussed in the "Using Predefined Groups" section.
- Previously created TNM identifiers which are introduced in the "Creating Arbitrary Groups Using TNMs" section.
- Groups defined in TIMEGRP symbols which are introduced in the "Creating New Groups from Existing Groups" section.

The *delay* parameter defines the maximum delay for the attribute, using nanoseconds as the default unit of measurement. Other units of measurement such as MHZ may also be used.

Note: Keywords, such as FROM and TO, appear in this document in upper case; however, you can enter them in the TIMESPEC primitive in either upper or lower case.

The following examples show typical TIMESPEC attribute definitions:

```
TS01=FROM:FFS:TO:FFS=30
TS_OTHER=FROM:PADS:TO:FFS=25
TS_THAT=FROM:PADS:TO:LATCHES=35
```

Note: Latches refer to input latches only.

OrCAD Users — The TIMESPEC primitive does not exist in the Xilinx OrCAD library. For details on how to use Time Specs in an OrCad design, refer to the *OrCAD Interface/Tutorial Guide*.

Mentor Graphics Users — The term *attribute* in this chapter is equivalent to *property* as used in the Mentor Graphics environment.

Index

W

X

Trademark Information

XILINX®, XACT, XC2064, XC3090, XC4005, and XC-DS501 are registered trademarks of Xilinx. All XC-prefix product designations, XACT-Floorplanner, XACT-Performance, XAPP, XAM, X-BLOX, X-BLOX plus, XChecker, XDM, XDS, XEPLD, XPP, XSI, BITA, Configurable Logic Cell, CLC, Dual Block, FastCLK, HardWire, LCA, Logic Cell, LogicProfessor, MicroVia, PLUSASM, SMARTswitch, UIM, VectorMaze, VersaBlock, VersaRing, and ZERO+ are trademarks of Xilinx. The Programmable Logic Company and The Programmable Gate Array Company are service marks of Xilinx.

IBM is a registered trademark and PC/AT, PC/XT, PS/2 and Micro Channel are trademarks of International Business Machines Corporation. DASH, Data I/O and FutureNet are registered trademarks and ABEL, ABEL-HDL and ABEL-PLA are trademarks of Data I/O Corporation. SimuCad and Silos are registered trademarks and P-Silos and P/C-Silos are trademarks of SimuCad Corporation. Microsoft is a registered trademark and MS-DOS is a trademark of Microsoft Corporation. Centronics is a registered trademark of Centronics Data Computer Corporation. PAL and PALASM are registered trademarks of Advanced Micro Devices, Inc. UNIX is a trademark of AT&T Technologies, Inc. CUPL, PROLINK, and MAKEPRG are trademarks of Logical Devices, Inc. Apollo and AEGIS are registered trademarks of Hewlett-Packard Corporation. Mentor and IDEA are registered trademarks and NETED, Design Architect, QuickSim, QuickSim II, and EXPAND are trademarks of Mentor Graphics, Inc. Sun is a registered trademark of Sun Microsystems, Inc. SCHEMA II+ and SCHEMA III are trademarks of Omation Corporation. OrCAD is a registered trademark of OrCAD Systems Corporation. Viewlogic, Viewsim, and Viewdraw are registered trademarks of Viewlogic Systems, Inc. CASE Technology is a trademark of CASE Technology, a division of the Teradyne Electronic Design Automation Group. DECstation is a trademark of Digital Equipment Corporation. Synopsys is a registered trademark of Synopsys, Inc. Verilog is a registered trademark of Cadence Design Systems, Inc.

Xilinx 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 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 will not assume responsibility for the use of any circuitry described herein other than circuitry entirely embodied in its products. Xilinx 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; 4,853,626;

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; RE 34,363, RE 34,444, and RE 34,808. Other U.S. and foreign patents pending. Xilinx, Inc. does not represent that devices shown or products described herein are free from patent infringement or from any other third party right. Xilinx assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or software support or assistance provided to a user.

Xilinx products are not intended for use in life support appliances, devices, or systems. Use of a Xilinx product in such applications without the written consent of the appropriate Xilinx officer is prohibited.