

Mentor Schematic Design Tutorial

This chapter contains the following sections:

- “Introduction”
- “Required Background Knowledge”
- “Design Flow”
- “Software Installation”
- “Starting the Design Manager”
- “Copying the Tutorial Files”
- “Starting Design Architect”
- “Targeting the Design for the XC9000 or XCR Family”
- “Completing the Calc Design”
- “Controlling FPGA/CPLD Layout from the Schematic”
- “Modifying the Design for Non-XC4000E/EX Devices”
- “Using LogiBLOX”
- “Other Special Components”
- “Using a Constraints File”
- “Performing Functional Simulation”
- “Using Pld_men2edif”
- “Using the Xilinx Design Manager”
- “Performing Timing Simulation”
- “Examining Routed Designs with FPGA Editor”
- “Verifying the Design Using a Demonstration Board”

- “Making Incremental Design Changes”
- “Command Summaries”
- “Further Reading”

Introduction

This chapter guides you through a typical field-programmable gate array (FPGA) and complex programmable logic device (CPLD) design procedure from schematic entry to completion of a functioning device. It uses a design called Calc, a 4-bit processor with a stack. In the first part of the tutorial, you use the Design Architect, the Mentor Graphics design entry tool, to create the schematics and symbols for the Calc design. Next you use `p1d_quicksim`, the Mentor Graphics simulator, to perform a functional simulation on it. In the third step, you use the Xilinx Design Manager to implement the design. Finally, you verify the design’s timing by again using `p1d_quicksim`. The simple design example used in this tutorial demonstrates many system features that you can apply to more complex FPGA and CPLD designs.

Note: Although this tutorial describes creating and processing FPGA designs, you can apply most of the steps to CPLD designs.

This tutorial includes instructions on the following:

- Installing the tutorial files
- Using Mentor Graphics Design Manager
- Targeting the tutorial design (Calc) for an XC4000E or an XC9000 device
- Using Design Architect
- Completing the ALU block in the Calc design
- Adding the STARTUP block to tie signals to the global reset
- Adding device information in the Calc design
- Exploring Xilinx library elements
- Exploring the XC4000E oscillator
- Controlling device layout from the schematic
- Editing the Calc design for a non-XC4000E/EX device

- Performing functional simulation on the Calc design in pld_quicksim
- Converting the design to an EDIF file using pld_men2edif
- Implementing the design using pld_dsgnmgr
- Configuring the Xilinx Design Manager/Flow Engine
- Performing timing simulation on the routed Calc design in pld_quicksim
- Examining routed designs with the Editor for Programmable ICs (FPGA Editor)
- Verifying the Calc design on a demonstration board
- Making incremental design changes
- Command summaries

Required Background Knowledge

This tutorial assumes that you have a basic understanding of the following:

- UNIX operating system
- Motif Windows. Mentor Graphics applications conform to the Motif window style.

Note: When you are instructed to close a window, it is important that you exit from the window rather than iconize it.

Design Flow

See the “Design Flows” section of the “Introduction” chapter for the design flow involved in using the Mentor Graphics interface. That chapter also describes the general steps for creating a design using the Mentor interface.

This tutorial describes an incremental design methodology. In incremental design, you process the design, make a small change to the design, and process the design again. You use place and route information from the previous design processing cycle to constrain subsequent cycles of the same design. When you use this method, timing information in a design remains relatively stable through many

processing cycles. Also, place and route time is considerably reduced since much of the processing is done in previous cycles.

You can target the tutorial design for an XC4000E or XC9000 device. You can use a Xilinx demonstration board to test the functionality of your design. Make sure your demonstration board and software support your selected device. To determine compatibility, refer to the release notes that came with your software package.

This tutorial uses the following conventions to refer to the various device families:

- **XC3000 family**—includes XC3000, XC3000A, XC3000L, XC3100, and XC3100A devices
- **XC4000 family**—includes XC4000, XC4000E, XC4000EX, XC4000L, and XC4000XL devices
- **XC5200 family**—includes XC5200 devices
- **XC9000 family**—includes XC9500 and XC9500F devices

Software Installation

Required Software

The following versions of software are required to perform this tutorial:

- Mentor Graphics Version C.2 or later, including Mentor Design Manager, Design Architect, QuickSim, QuickPath, as well as the programs needed to read and write EDIF netlists (ENRead and ENWrite), which require special licensing
- Xilinx/Mentor Graphics Interface Version 2.1i
- Xilinx Development System Version 2.1i

Before Beginning the Tutorial

Before beginning the tutorial, set-up your workstation to use Mentor Graphics and Xilinx Development System software as follows:

1. Verify that your system is properly configured. Consult the release notes that came with your software package for more information.

2. Install the following sets of software:
 - Xilinx Development System Version 2.1i
 - Xilinx/Mentor Graphics Interface Version 2.1i
 - Mentor Graphics Version C.2 or later, including Mentor Design Manger, Design Architect, QuickSim, QuickPath, as well as the programs needed to read and write EDIF netlists (ENRead and ENWrite), which require special licensing
3. Verify the installation, using the “Configuring Your System” section of the “Getting Started” chapter of the Mentor Graphics Interface Guide, as a guide.
4. Add a reference to \$XILINX_TUTORIAL to your MGC_LOCATION_MAP file.

Every symbol and schematic in your design contains references which indicate where design objects reside on your disk or network. The tutorial designs use variables in their reference definitions so they can be easily relocated. All of the tutorial designs use the variable \$XILINX_TUTORIAL in their path references. \$XILINX_TUTORIAL must be defined in the file pointed to by \$MGC_LOCATION_MAP. For example, the design object seg7dec in the \$XILINX/mentor/tutorial/calc_sch directory uses the path reference \$XILINX_TUTORIAL/calc_sch/seg7dec to define where it is located in the directory structure. If the tutorial directories were copied to the path /home/bclinton/mentor/xtutorial, the following two lines must be added to the file pointed to by \$MGC_LOCATION_MAP:

```
$XILINX_TUTORIAL  
/home/bclinton/mentor/xtutorial
```

If you make a query to determine where the design object “\$XILINX_TUTORIAL/calc_sch/stack” is located, the Mentor Graphics tools use this definition to determine that stack is at /home/bclinton/mentor/xtutorial/calc_sch/stack.

With this definition added to the location map as defined in the “Getting Started” chapter of the Mentor Graphics Interface Guide, the complete location-map file should, at a minimum, look like:

```
MGC_LOCATION_MAP_1  
(empty line)
```

```
$MGC_GENLIB  
(empty line)  
$LCA  
(empty line)  
$SIMPRIMS  
(empty line)  
$XILINX_TUTORIAL  
/home/bclinton/mentor/xtutorial
```

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

Installing the Tutorial

If you have not already done so, download the tutorial files from ftp://ftp.xilinx.com/pub/documentation/M2.1i_tutorials/men_tut_files_21i.tar.Z. Once they are downloaded un-compress and un-tar the files.

```
uncompress men_tut_files_21i.tar.Z  
tar xvf men_tut_files_21i.tar
```

Standard Directory Structure

When you create a design object in Mentor Graphics, a directory is created in the project directory with the same name as the design object. This directory contains a schematic directory, symbol files, viewpoint files, and part interfaces. The directory is identified as a design object by the file, `design_name.mgc_component.attr`, that resides at the same level as the directory which has the name. For example, if you create a schematic named `calc`, a `calc` directory is created, and at the same level the file, `calc.mgc_component.attr`, is created. The `calc` directory contains all the files that describe `calc`.

Note: In this tutorial, file names and directory names are in lower case and the design example is referred to as `Calc`.

Tutorial Directory and Files

You will complete the `Calc` design in this tutorial. During the tutorial installation, the `/tutorial` directory is created; design object directories are created; and the tutorial files needed to complete the design are copied to the `calc_sch` directory. Some of the files you need to complete the tutorial design are not copied, because you create these

files in the tutorial. However, solutions directories with all input and output files are provided. They are located in the /tutorial directory and are listed in the following table.

Table 10-1 Tutorial Design Directories

Directory	Description
calc_sch	Schematic (Design Architect) tutorial directory
calc_4ke	Schematic solution directory for XC4003E-PC84
calc_9k	Schematic solution directory for XC95108-PC84
calc_sot	Schematic-on-top tutorial directory (uses XC4003E)

The solution directories contain the design files for the completed tutorial, including schematics, intermediate files, and the bitstream file. Different intermediate files are created for different device families. Do not overwrite any files in the solutions directories.

The calc_sch directory contains the incomplete copy of the tutorial design. The installation program copies a few intermediate files to the calc_sch tutorial directory, and you create the remaining files when you perform the tutorial. As described in a later step, you copy the calc_sch directory to another area and perform the tutorial in this new area. The following table lists and describes the directories and files in the calc_4ke solution directory.

Table 10-2 Tutorial Directories/Files in the Calc_4ke Directory

Directory or File Name	Description
calc	Top-level design directory
control	Design directory for control module
statmach	Design directory for state controller module
alu	Design directory for ALU module
muxblk2	Design component for arithmetic function in ALU
andblk2	Design component for arithmetic function in ALU
clockgen	System-clock generator
orblk2	Design component for arithmetic function in ALU

Table 10-2 Tutorial Directories/Files in the Calc_4ke Directory

Directory or File Name	Description
xorblk2	Design component for arithmetic function in ALU
muxblk5	Design component for multiplexer arithmetic outputs in ALU
muxblk2a	Design component for multiplexer operator function in control
stack	Design component for stack
seg7dec	Design component for 7-segment decoder
debounce	Design component for debounce circuit
calc.edif	EDIF netlist files created by pld_men2edif
pld_men2edif.log	pld_men2edif log file
calc.ngo	Native Generic Object created by EDIF2NGD
calc_4ke.ucf	User Constraints File
calc.bld	Design database report generated by NGDBUILD
calc.ngd	Native Generic Design created by NGDBUILD
calc.mrp	Mapping report generated by MAP
calc.pcf	Physical Constraints File created by MAP
calc_map.ncd	Native Circuit Description created by MAP
calc.par	Place-and-Route report generated by PAR
calc.pad	Pinout description generated by PAR
calc.ncd	Routed NCD file created by PAR
calc.twr	Timing report generated by Trace (TRCE)
calc.bit	Configuration bitstream created by BITGEN
calc.edn	Timing-model EDIF netlist created by NGD2EDIF
calc_lib/calc	QuickSim timing simulation model created by pld_edif2tim
pld_edif2tim.log	pld_edif2tim log file

In addition to the files listed above, there is a file called *filename.mgc_component.attr* associated with each design component directory. This file identifies the corresponding directory as a Mentor Graphics design component.

Starting the Design Manager

To start the Design Manager configured for Xilinx designs, type the following at the operating system command line:

```
p1d_dmgr
```

The Design Manager Window appears as shown in the following figure.

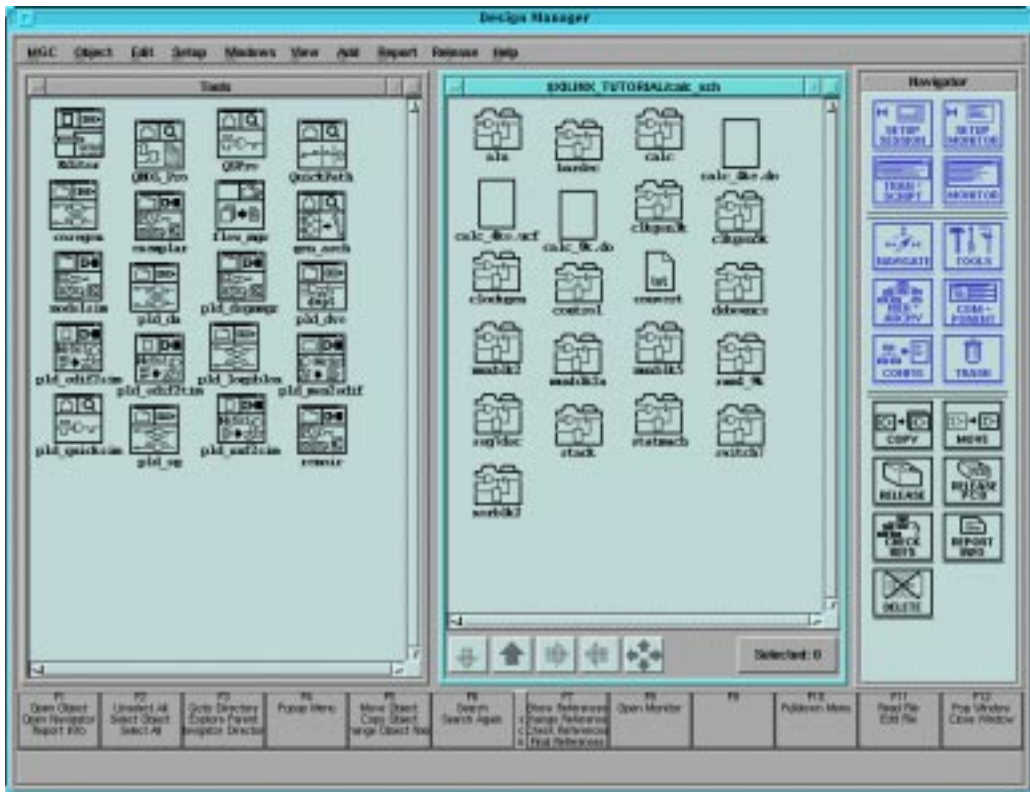


Figure 10-1 Mentor Design Manager Window

The Design Manager window contains the following three sub-windows:

- Tools Window
- Navigator Window
- Command Palette

Each sub-window is described below.

Mentor Graphics windows conform to Motif standards. You should know how to move, close, and minimize (or iconize) the Motif windows. When multiple windows are open, the active window has a blue border and inactive windows have a grey-brown border. For more information on Design Manager operation, refer to the Mentor Graphics documentation.

Tools Window

The Tools window on the left contains icons representing all the Mentor Graphics and Xilinx applications you need to execute the steps in the tutorial. A description of these programs is given in the “Features” section of the “Introduction” chapter.

Navigator Window

Use the Navigator window to move around the directory hierarchy and select files, folders, and other types of design objects.

The Navigator has five buttons located at the bottom of the window, three of which are most used. The two buttons on the left have up and down arrows on them. Use these buttons to move up and down the directory hierarchy. To move down the hierarchy with the down arrow, you must first select the desired folder in the Navigator. The rightmost button has four arrows on it, one pointing in each direction. When you select this button, a dialog box appears and you can type in the path to the directory you want to display in the Navigator window. Using this button is sometimes quicker and easier than using the up and down arrows.

Command Palette

Use the Command Palette to access the most commonly used Design Manager menu items.

Copying the Tutorial Files

Mentor Graphics design objects contain absolute directory path references. Since many of these paths are self-referencing, using the `cp` or `mv` command in Unix to copy or move these design objects to new directories can break those references. The Mentor Graphics Design Architect allows design objects to be copied or moved across directories by adjusting path references within each design object as it is relocated.

To demonstrate the Copy operation in Design Manager, perform the following steps:

1. In the Navigator window, move to the directory where the tutorial files were installed.
2. Select the `calc_sch` directory.
3. To see the references in this design, choose **Right Mouse Button** → **Report** → **Show References** → **For Design**.

A List of Unique References underneath the `calc_sot` directory is displayed. An example reference item might be:

```
$XILINX_TUTORIAL/calc_sot/alu/alu:mgc_symbol[6]
```

This indicates that an ALU symbol (version 6 under Mentor Graphics' versioning system) is referenced by the path `$XILINX_TUTORIAL/calc_sot/alu/alu`.

All references in the design should contain either `$LCA` or `$XILINX_TUTORIAL`.

4. Close the List of Unique References window.
5. With the `calc_sch` directory selected in the Navigator window, choose **Right Mouse Button** → **Edit** → **Copy**.

A dialog box appears.

6. In the dialog box, type the directory path where you want the working copy of the tutorial files copied. For example, if you want to copy the files to `/home/dum/tutor/mentor`, enter `/home/dum/tutor/mentor/calc_sch`. Click OK.
7. Use the Navigator to change directories to the location of the working copy of `calc_sch`. In the example above, you would click

the “four-arrow” button at the bottom of the Navigator window, then type `/home/dum/tutor/mentor` in the dialog box.

8. Select the `calc_sch` directory.
9. As before, choose **Right Mouse Button** → **Report** → **Show References** → **For Design**. The example reference above with the ALU symbol appears:

```
/home/dum/tutor/mentor/calc_sch/alu/alu:mgc_symbol[6]
```

10. Close the List of Unique References window.
11. Modify your `MGC_LOCATION_MAP` file so that the `$XILINX_TUTORIAL` variable points to the directory where the copy of `calc_sch` is located. In the example above, change the `$XILINX_TUTORIAL` section of the file so that it reads:

```
$XILINX_TUTORIAL  
/home/dum/tutor/mentor
```

12. Read the newly modified location map into Design Architect by selecting **MGC** → **Location Map** → **Read Map** from the menu bar.
13. In the dialog box, type `$MGC_LOCATION_MAP`, then click **OK**.

The `$XILINX_TUTORIAL` soft name now points to the new tutorial area. However, references in the `calc_sch` directory use `/home/dum/tutor/mentor` instead of its new equivalent, `$XILINX_TUTORIAL`. While this is legal, it is best in Mentor to use soft names wherever possible.

14. To convert the hard name back into a soft name, select the `calc_sch` directory and choose **Right Mouse Button** → **Edit** → **Change** → **References**.
15. In the Change References dialog box, enter for From: `/home/dum/tutor/mentor` (or whatever directory is applicable to your case). For To, enter `$XILINX_TUTORIAL`.

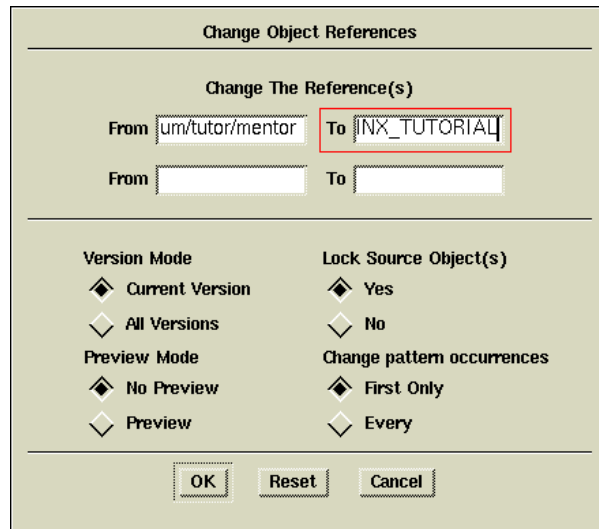


Figure 10-2 Change References Dialog Box

16. Click **OK**.

The Change References process begins.

17. After the process is finished, you can do another Show References operation to verify that all references have been changed properly.

Note: You can copy or move a design object without rewriting path references by selecting **Options** → **Convert References?** **No** from the Copy or Move dialog box.

Starting Design Architect

To open the Calc design in Design Architect, perform the following steps:

1. Select **MGC** → **Location Map** → **Set Working Directory** from the menu bar.

A small dialog box appears at the bottom of the screen.

2. Type **\$XILINX_TUTORIAL/calc_sch** in the Directory field of the dialog box, then select **OK** or press return.

Using the Mouse in Design Architect

Left Mouse Button

Use this button to select or de-select objects on a sheet. A selected object has a white dashed outline. Hold down this button and drag the mouse to select multiple objects.

Middle Mouse Button (Strokes)

Use the middle mouse button to perform actions known as strokes. You can use strokes as shortcuts to perform common tasks. Perform a stroke by pressing and holding the middle mouse button while moving the mouse to draw a line with a specific shape. Design Architect converts the shape you draw to a number string to determine which command to execute. The number is determined as shown in the following figure:

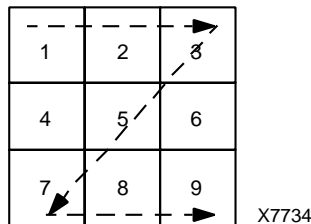


Figure 10-4 Using Strokes, Example of “Z” stroke (1235789)

For example, a “Z” stroke represents the number 1235789. To determine the commands that the strokes represent, select **Help** → **On strokes** from the menu bar at the top of the screen. You can also hold down the middle mouse button and draw the shape of a question mark “?” to display the stroke help screen. When applicable, this tutorial uses strokes and describes them using the numbering system shown in the “Using Strokes, Example of “Z” stroke (1235789)” figure.

Right Mouse Button

You can use the right mouse button to display different menus depending on the object(s) selected on the schematic sheet. For

example, if a net is selected when you press the right mouse button, the Net menu appears. You can access other menus, regardless of what is selected, by using the “Other Menus” selection that appears at the top of each menu.

Using the Function Keys

You can use the keyboard function keys to execute many Design Architect commands. The boxes at the bottom of the Design Architect window each contain up to four commands which you execute as follows:

- To execute the top command, press the associated Function key.
- To execute the middle command, press the associated Function key while holding down the Shift key.
- To execute the third command, press the associated Function key while holding down the Control key.
- To execute the bottom command, press the associated Function key while holding down the Alternate key.

Selecting Commands from the Menu Bar

Use the left mouse button to select commands from the menu bar at the top of the screen.

Selecting Commands from the Palette

Use the left mouse button to select commands from the Command Palette at the right side of the screen. The set of red buttons at the top of the palette change the commands that are available in the palette. The commands displayed in the palette vary depending on what type of window is active in Design Architect. For example, if a symbol editor window is active, commands such as Add Pin, Draw Rectangle, and other commands associated with creating symbols are available in the palette. If there are no windows open in Design Architect, commands such as OPEN SHEET or OPEN SYMBOL are available.

You may need to scroll the palette to access some of the commands by moving the cursor into the palette and using the PageUp and PageDown keys. You can also select **Right Mouse Button** → **Show Scroll Bars** to display scroll bars.

Entering Commands from the Keyboard

You can type commands anywhere in the Design Architect window. A dialog box appears at the cursor location to capture the command text. For example, you can open a schematic sheet by typing **open sheet** in the Design Architect window.

Cancelling Commands

When you select a command, it is displayed in either a small rectangular box in the lower-left area of the screen, or in a larger dialog box. In either case, you can cancel commands by selecting the cancel button in the box or by pressing the escape key.

Repeating Menu Commands

You can repeat commands that were executed by using either the menu bar or the menus accessed through the right mouse button by holding down the control key, moving the cursor to the appropriate area, and pressing the right mouse button. For example, if **Right Mouse Button** → **Properties** → **Add** was the last command sequence performed, you can repeat this sequence by holding down the control key and pressing the right mouse button with the cursor in the window where the command was last executed. You can also perform this function with the stroke 12369, which looks like an upside-down “L”.

Manipulating the Screen

To zoom in on a specific area of the screen, hold down the F8 key and move the mouse to create a box around the area you want to zoom on. To view the entire schematic, hold down the shift key and press F8. You can also perform these commands with the strokes 159 and 951, respectively. You can also zoom the schematic in or out with the menu bar commands **View** → **Zoom In** and **View** → **Zoom Out** or the strokes 357 and 753.

Targeting the Design for the XC9000 or XCR Family

The incomplete calc_sch design is configured for an XC4003E-PC84 part. If you want to target a demonstration board with this device, go to the “Completing the Calc Design” section. If you are targeting the

tutorial design for an XC95108-PC84 (no demonstration board available) or other device family, you must convert the design to reference the XC9000 library instead of the XC4000E library.

The Xilinx CoolRunner (XCR) series can be targeted by using the XC9000 library. The XCR series currently uses XPLA Workstation for implementing designs on the workstation. Xilinx Design Manager does not support XCR designs.

With three exceptions, the calc design can target the XCR as defined in this tutorial. XPLA Workstation cannot compile designs which use LogiBLOX. The config cell is not supported. Timing simulation uses ModelSim rather than QuickSim. The tutorial entitled Workstation Flow for Xilinx CoolRunner CPLDs provides steps to implement and do a timing simulation of the calc design.

The procedure provided below allows you to change every Xilinx component in the Calc design from the XC4000E library to the XC9000 library. Since the designs were created using the Unified Libraries, the parts in the XC4000E and XC9000 libraries have identical footprints and pinouts. This allows you to easily retarget designs to a different device family, provided only library parts common to the two families are used. You must manually replace any library parts that are not common to both families. This example shows a situation where this may happen.

Note: Although an XC4000E-to-XC9000 conversion is shown here, this procedure may be used to retarget from any family to any other family.

To retarget the Calc design to the XC9000 or XCR family:

1. Close the Calc schematic window by selecting **C**lose from the window's control menu. This is the menu accessed by clicking on the button in the upper left-hand corner of the window.
2. In the gray desktop area, choose **R**ight **M**ouse **B**utton → **C**onvert **D**esign.
3. A dialog box appears as shown in the figure below. Fill in the fields as shown and click **O**K.



Figure 10-5 Convert Design Dialog Box

Convert Design begins by displaying a Hierarchy Window as it is taking account of the design hierarchy. After this, the Calc schematic and all lower-level design schematics are displayed while Design Architect is retargeting design components.

4. After this process is completed, perform the following for each schematic:
 - a) Select **C**heck → **S**heet from the menu bar. A window appears containing the results of the design rule check.
 - b) After reviewing the contents of this window, close it and reselect the schematic window.
 - c) Select **F**ile → **S**ave from the menu bar to save the schematic.
 - d) Close the schematic window and repeat the process for the next schematic.

Note: Keep the top-level Calc design open, since you need it in the next section. You can also save yourself the step of checking and saving each design sheet by setting the Check & Save switch to Yes in the Convert Design dialog box.

Warning: Although turning the Check & Save switch on makes the Convert Design process more automatic, it is more dangerous since it prevents you from inspecting a converted design before saving it.

Use this setting with caution, and turn it on only when you are certain you wish to overwrite your original design.

Completing the Calc Design

To complete the tutorial design, you need to add a few design objects to the Calc schematic using Design Architect.

If you need to stop the tutorial at any time, be sure to save your work as follows:

1. Select **Check** → **Sheet** from the menu bar.
A window appears containing the results of the design rule check.
2. After reviewing the contents of this window, close it and reselect the schematic window.

Warning: It is important to check your design first before saving it.

3. Select **File** → **Save** from the menu bar to save the design.
4. Before proceeding to the next part of this tutorial, close (quit) the Calc schematic window.
5. If a dialog box appears asking if you want to save any changes, choose **NO**.

Design Description

The top-level schematic of the Calc tutorial design has been created for you. Each of the blocks in the schematic, such as CONTROL or ALU, is linked to a second-level module that describes its logic. Additionally, any second-level module can contain another block that references a third-level drawing, and so on. This organization is known as a hierarchical structure.

In this tutorial, you add three symbols to the ALU block schematic to complete it. First, you create the ANDBLK2 and ORBLK2 symbols and their underlying schematics and then add them to the schematic. Additionally, you add the FD4RE symbol from the Unified Libraries to the ALU block. After the ALU block is finished, you add the STARTUP block to the top-level Calc schematic to tie the device's global reset network to a device pin. To complete design entry, you

add a CONFIG block, which lists a set of instructions that dictate how the core tools should process the design.

The Calc design is a four-bit processor with a stack. The processor performs functions between an internal register and either the top of the stack or data input from external switches. The results of the various operations are stored in the register and displayed in hexadecimal on a seven-segment display. The top value in the stack is displayed in binary on a bar LED. A count of the items in the stack is displayed as a “gauge” on another bar LED.

The design consists of the following functional blocks:

- **ALU**—The arithmetic functions of the processor are performed in this block.
- **CONTROL**—The opcodes are decoded into control lines for the stack and ALU in this module.
- **STACK**—The stack is a four-nibble storage device. It is implemented using synchronous RAM in the XC4000E design. You can substitute the RAM4_9K module, which uses flip-flops, in place of the RAM16X4S macro in the STACK schematic to implement the stack in an XC9000 or other non-XC4000E device.

Note: If RAM4_9K is used in a non-XC9000 device, it must be retargeted using Convert Design.

- **DEBOUNCE**—This circuit debounces the “execute” switch, providing a one-shot output.
- **SEG7DEC**—This block decodes the output of the ALU for display on the 7-segment decoder.
- **CLOCKGEN**—This block uses an internal oscillator circuit in XC4000E devices to generate the clock signal. In XC9000 designs, it is replaced by an input pad and a clock buffer.

Note: The XC3000 and XC5200 FPGA families also have on-board oscillators. See the CLKGEN3K and CLKGEN5K components included in the calc_sch directory to see how the oscillators in these families are used.

- **BARDEC**—This block shows how many items are on the stack on a “gauge” of four LEDs.
- **SWITCH7**—This block is a user-defined module consisting of seven input flip-flops used to latch the switch data.

Creating the ANDBLK2 Symbol

Opening a Symbol Window

1. Use the left mouse button to select Open Symbol in the Command Palette.
2. Type `$XILINX_TUTORIAL/calc_sch/andblk2` in the Component Name box.
3. Select **OK**.

A symbol editor window appears.

Creating the Symbol Outline

1. Zoom in until the grid space markers, represented by small crosses, are visible in the symbol window.
2. Select **ADD RECTANGLE** from the palette.
3. Position the cursor in the upper left corner of the symbol window and press the left mouse button.
4. While holding down the left mouse button, move the cursor diagonally to the opposite corner of the symbol window to draw a rectangle that is six grid squares high by eight grid squares wide. Be sure to measure using the grid marks, and not the small dots that define fractions of grid spacing.

Adding Pins to the ANDBLK2 Symbol

1. Select Add Pin from the palette.
The dialog box in the following figure appears.
2. Fill in the Dialog box exactly as shown in the following figure and then select **OK**.

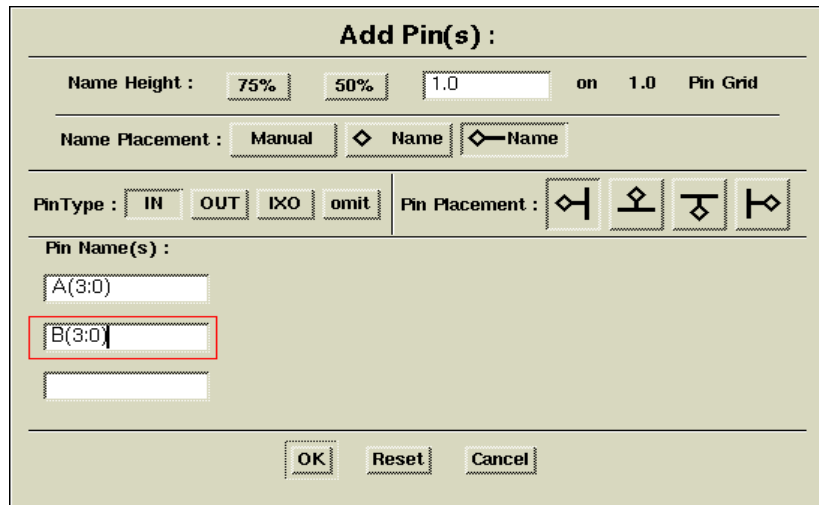


Figure 10-6 Add Pin(s) Dialog Box for A(3:0) and B(3:0)

A small crosshair appears under the cursor, and a rectangular box appears stating that the first pin, A(3:0), is to be placed.

- Place pin B(3:0) as shown in the figure below by moving the cursor to the position where the diamond appears in the figure (one grid space to the left of the rectangle) and pressing the left mouse button. Small purple diamonds indicate pins.

If you make a mistake before placing a pin, press the escape key to cancel the command, then repeat the above steps. If you make a mistake after placing a pin, press the F2 key to unselect everything. Select the pin (diamond) and the line next to it and press and hold CTRL-F2 to execute a move command. Move the pin to the correct position and release the keys.

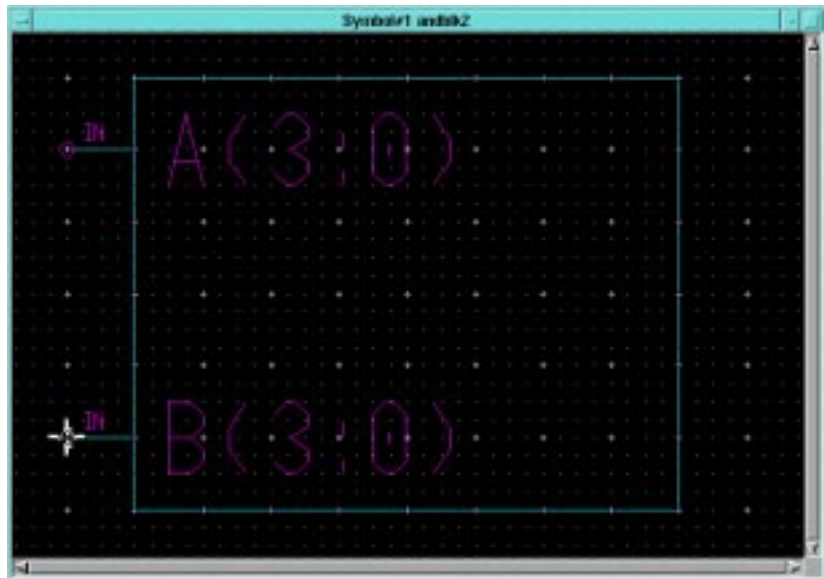


Figure 10-7 Adding Pins A(3:0) and B(3:0)

4. Select **Add Pin** from the palette and fill in the dialog box as shown in the following figure, then select **OK**. Be sure to set the name height to 1.0.

The image shows a dialog box titled "Add Pin(s) :". It has several sections:

- Name Height :** Three buttons labeled "75%", "50%", and "1.0", followed by the text "on 1.0 Pin Grid".
- Name Placement :** A dropdown menu set to "Manual", a diamond icon, and a button labeled "Name" with a diamond icon.
- PinType :** Four buttons labeled "IN", "OUT", "IXO", and "omit".
- Pin Placement :** Four icons representing different pin placement styles: a diamond with a vertical line, a diamond with a horizontal line, a diamond with a diagonal line, and a diamond with a vertical line and a horizontal line.
- Pin Name(s) :** A text input field containing "Q(3:0)", which is highlighted with a red rectangle. Below it is an empty text input field.
- Buttons:** "OK", "Reset", and "Cancel" buttons at the bottom.

Figure 10-8 Add Pin(s) Dialog Box for Q(3:0)

5. Place the pin Q(3:0) as shown in the figure below.
6. To adjust the positioning of the pin names, move the mouse over the text, press and hold the F7 function, and move the mouse to reposition the text. Release the F7 key to place the text at the new location.

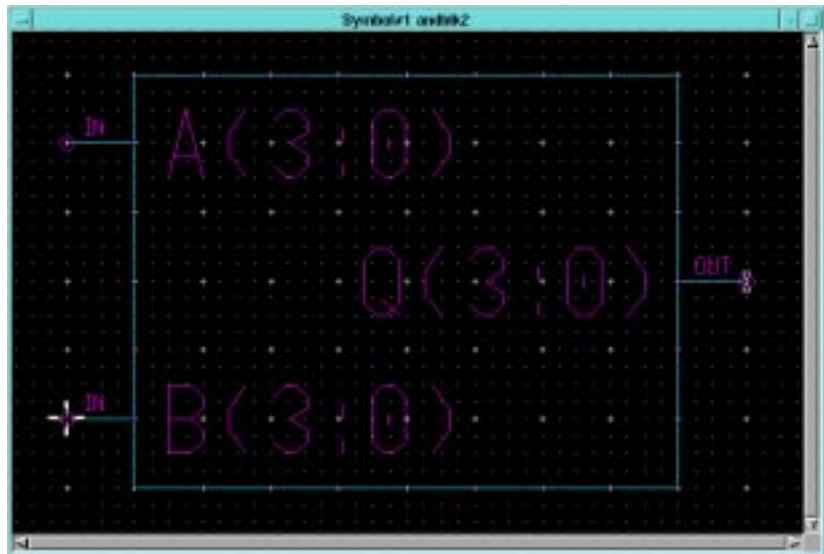


Figure 10-9 Adding Pin Q(3:0)

Adding Text

You can add comment text to a symbol to make it more easily identifiable on a schematic, or to annotate it without modifying its function. To add text to the symbol, perform the following steps:

1. Select the red **TEXT** button at the top of the palette to display the text editing icons.
2. Choose **ADD TEXT** from the palette.

A small rectangular dialog box appears in the lower left portion of the window.

3. Type **ANDBLK2** in the Text field of the dialog box, then press return or select **OK**.
4. Move the cursor into the symbol editor window and place the text directly above the symbol body by moving the mouse to the proper position and pressing the left mouse button.

If you make a mistake while typing the text and the text has already been placed, move the mouse over the text and press the F7 key while holding down the shift key. A small dialog box

appears at the bottom of the screen containing the selected text. Modify the text in the dialog box. Select OK to change the text on the symbol. You can use this method to modify any text on the symbol, such as pin names.

Modifying Text Size

To modify symbol text size, perform the following steps:

1. Press the F2 key to unselect everything.
2. Use the left mouse button to select the text, **ANDBLK2**, at the top of the symbol.
3. Select **Right Mouse Button** → **Change Height** → **1.5 X pin spacing**.
4. Place the cursor over the text and press and hold the F7 key.
5. While still holding down the F7 key, move the text so that it is centered above the symbol body, as shown in the following figure.

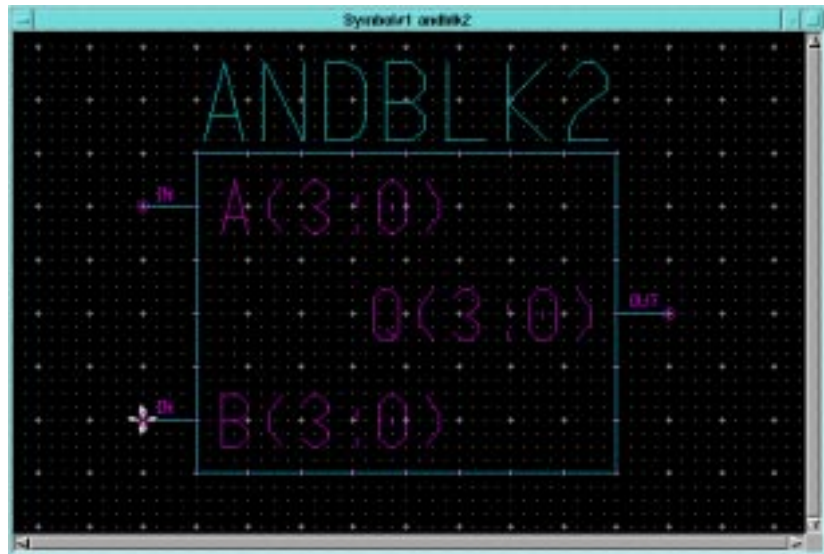
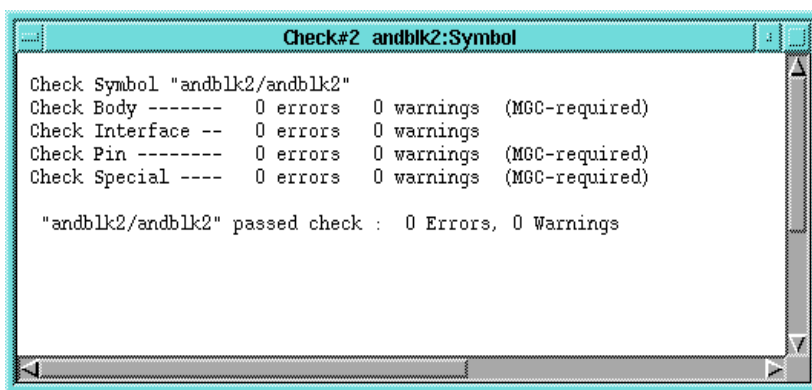


Figure 10-10 Completed ANDBLK2 Symbol

Saving the ANDBLK2 Symbol

To save the ANDBLK2 symbol, perform the following:

1. From the menu bar, select **Check** → **With Defaults**.
A text window appears containing the results of the design rule check.
2. Check to see that the information displayed is the same as that in the following figure. If you do not have the same output, correct the symbol to eliminate the differences and then check the symbol again.



```
Check#2 andblk2:Symbol
Check Symbol "andblk2/andblk2"
Check Body ----- 0 errors 0 warnings (MGC-required)
Check Interface -- 0 errors 0 warnings
Check Pin ----- 0 errors 0 warnings (MGC-required)
Check Special ---- 0 errors 0 warnings (MGC-required)

"andblk2/andblk2" passed check : 0 Errors, 0 Warnings
```

Figure 10-11 Output from Check

3. Close the text window by selecting **Close** from the menu that appears when the left mouse button is pressed in the box in the upper left hand corner of the text window.
4. Select **File** → **Save Symbol** from the menu bar to save the symbol.

Creating the ORBLK2 Symbol

The next step is to create the symbol for ORBLK2, as shown in the following figure. Since ORBLK2 is similar to ANDBLK2, use the ANDBLK2 symbol and modify the text as described below.

1. Move the cursor above the ANDBLK2 text.
2. Press the F7 key while holding down the shift key to select the Change Text Value command.

- In the small dialog box that appears in the lower left corner, type **ORBLK2** in the New Text field, then select **OK**.

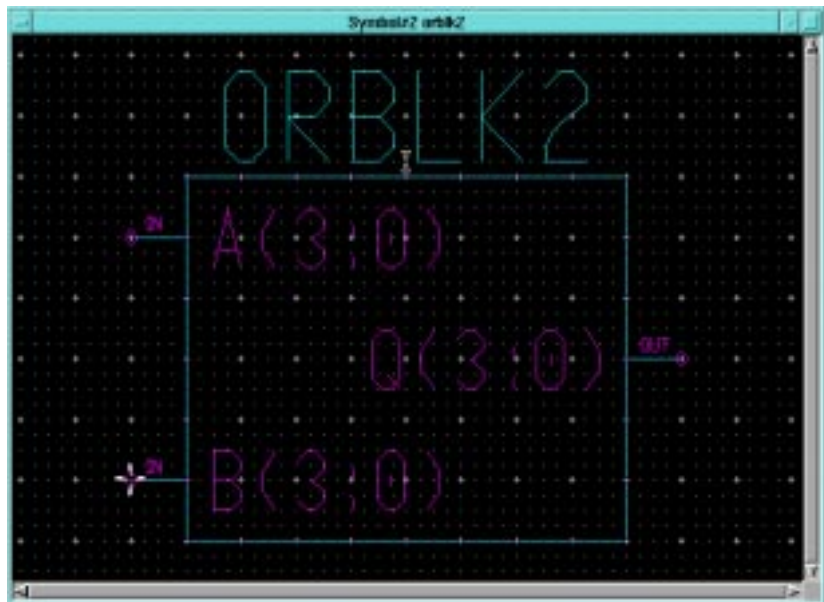


Figure 10-12 Completed ORBLK2 Symbol

- If necessary, use the cursor and F7 key to move and center the text, as described earlier.
- From the menu bar, select **Check** → **With Defaults**.
A text window appears containing the results of the design rule check. Since you are modifying the ANDBLK2 symbol, the text still refers to ANDBLK2.
- If any errors are reported in the Check text window, correct them on the symbol and check the schematic again. Otherwise, close the text window.
- To save the symbol as ORBLK2, select **File** → **Save Symbol AS**.

A dialog box appears.

Warning: It is important that you select the Save Symbol As command instead of Save to prevent overwriting the original ANDBLK2 file.

8. Enter `$XILINX_TUTORIAL/calc_sch/orblk2` in the component name field and enter `orblk2` in the interface name field.
9. Select **OK** to execute the command.
This saves the symbol as `ORBLK2`.
10. Close the window containing the symbol.
A dialog box appears prompting you to save the changes to `ANDBLK2`. Since the symbol for `ANDBLK2` was saved prior to modifying it for the `ORBLK2` symbol, it is not necessary to save changes to the `ANDBLK2` symbol.
11. In the dialog box, select **No**.

Creating Schematics for ANDBLK2 Symbol

You have created symbols for `ANDBLK2` and `ORBLK2`. The next step is to create schematics for these blocks. You can then reference the schematics in a higher-level schematic by placing the symbols.

Opening a Schematic Window

1. To open a schematic window, select **OPEN SHEET** from the palette.
2. In the dialog box that appears, type `$XILINX_TUTORIAL/calc_sch/andblk2` in the Component Name field.
3. Select **OK**.
A blank schematic sheet appears.

Adding the First Component to a Schematic

1. From the menu bar, select **Libraries** → **XILINX Libraries**.
The Xilinx Libraries menu appears.
2. Select the correct library for the device you are targeting, either `XC4000E` or `XC9000`.
If you select the wrong library, use the **PageUp** key to go to the top of the Library Palette menu and click the left mouse button on the **Back** option. This moves the library menu back up the hierarchy.
3. Choose **BY TYPE** from the palette.

This option organizes the library parts into categories. The ALL PARTS option displays all the library parts at once. A menu appears similar to that shown in the figure below.

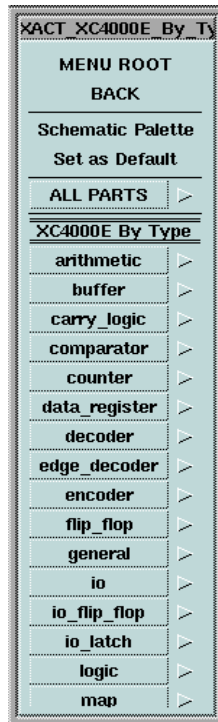


Figure 10-13 XC4000E Library BY TYPE Menu

4. To move up and down in the menu, turn on the scroll bars by moving the cursor into the menu window and selecting **Right Mouse Button** → **Show Scroll Bars**. You can also move up and down using the PageUp and PageDown keys.
5. Click the left mouse button on the Set As Default option. This option allows you to return to this area and view of the library menu by clicking on the Library icon in the Schematic Palette.
6. Choose the logic category from the **BY TYPE** menu.
7. Select **and2**.
8. In the small dialog box that appears on the screen, move the cursor into the schematic window.

The outline of a 2-input and gate appears.

9. Move the symbol outline to the location shown in the following figure and then click the left mouse button to place the object.

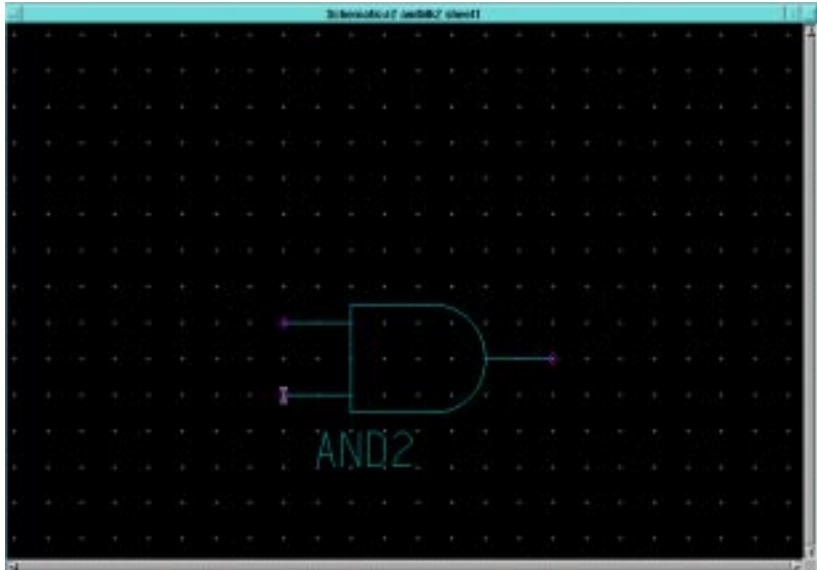


Figure 10-14 Placing a Component

Placing Additional Components

After placing the and2, note that a picture of it appears in the small window in the upper right area of the screen. The last library element selected appears in this window.

1. To select another component of the same type, move the mouse inside this window, and click the left mouse button.
2. Then move the cursor to the schematic window, position the component, and release the mouse button to place it on the sheet.
3. Using this method, select and place a second and2 symbol as shown in the following figure.

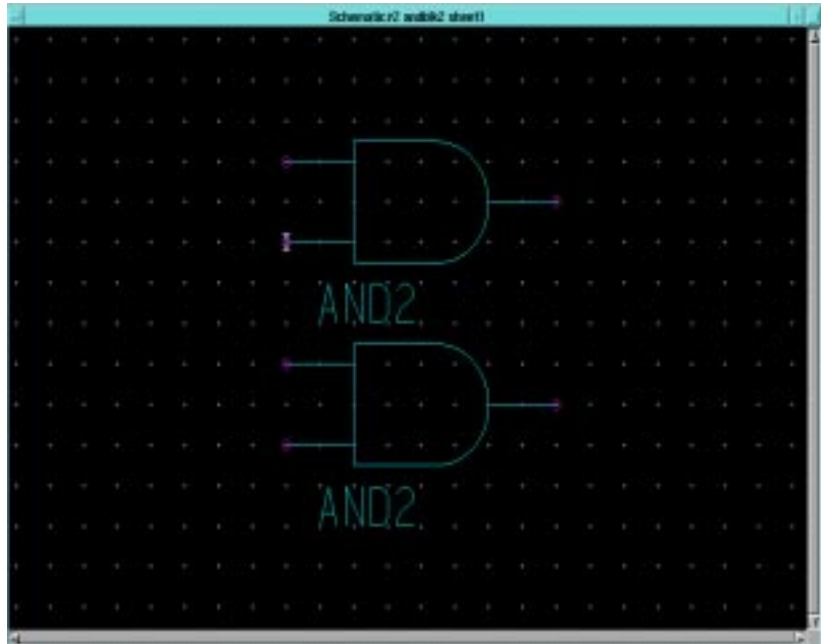


Figure 10-15 Placing a Second Component

Copying a Component

Use the Copy command to add more components by copying a component that already appears on the schematic.

1. Press the F2 function key to ensure that nothing is selected.

It is important to use the F2 key before selecting objects because objects selected in previous steps are sometimes not deselected.

2. Move the mouse above and to the left of the two symbols on the sheet.
3. While holding down the left mouse button, move the mouse below and to the right of the two symbols.

A white box appears surrounding the two symbols.

4. Release the mouse button to select the objects.
5. Select **Right Mouse Button** → **Copy**. Alternatively, use stroke 3214789, a stroke in the form of a “C”, to select the copy command.

A small dialog box appears at the bottom of the screen.

6. Place the two copied gates above the original two using the left mouse button. If necessary, use the 753 stroke to zoom out.

The dialog box disappears after you place the gates.

7. Press Shift - F8 to view the entire schematic.

The schematic now looks like the following figure.

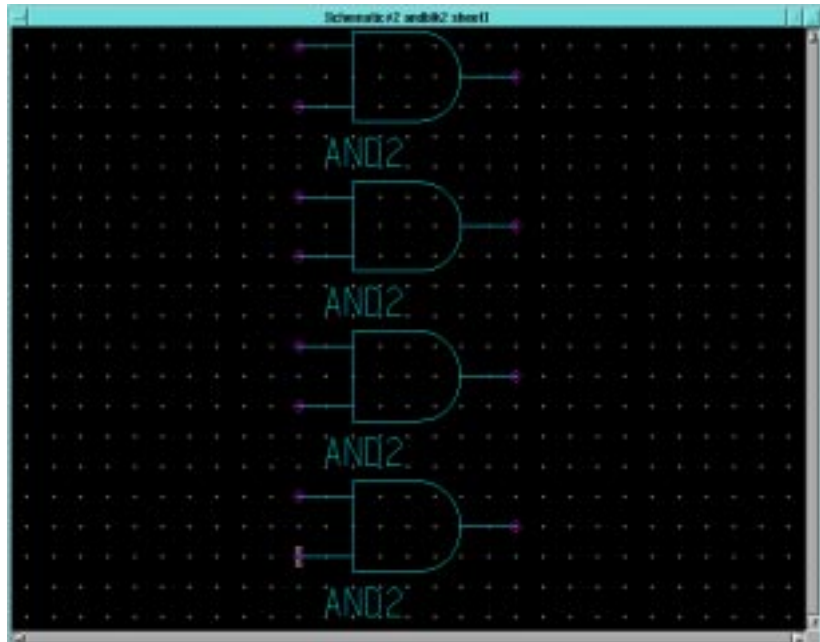


Figure 10-16 Component Placements for ANDBLK2

Moving a Component

If you make a mistake when placing a component, you can use the menu commands to move the component.

1. Use the F2 key to deselect.
2. Select the component by clicking on it with the left mouse button.

The component appears highlighted, indicating that it has been selected.

3. Select **Right Mouse Button** → **Move**, or use the stroke 74159.

A small dialog box appears.

4. Click the left mouse button to correctly place the component.

The dialog box disappears after the component is placed.

Adding Buses to a Schematic

Sometimes it is convenient to draw a set of signals as a bus rather than as several separate wires. It is not necessary to physically connect a bus to the nets that make up the bus. There are several schematics in the Calc design that have short bus segments that are not connected to anything. This is done so that a bus pin can be used to represent the bus on the symbol. A bus must exist on the schematic if a bus pin is to be used for a set of signals.

Add buses to the schematic as follows:

1. After pressing the F2 key, select **Right Mouse Button** → **Bus**.

A small dialog box appears, and a white cross appears under the cursor.

2. Draw a bus by clicking the left mouse button to specify the starting point, moving the mouse to a new position, and then clicking the button again to make a bend in the bus or to connect it to a pin. Terminate the bus is by clicking the mouse button in the same place twice. Add the three buses shown in the figure below. You may want to zoom the schematic view out before performing adding the buses.

If you make a mistake, press the F2 key to deselect everything on the sheet. Then click on the bus segments you want to delete so that they appear highlighted. Press the Delete key and then redraw them correctly.

3. After adding the three buses, press the **Escape** key to exit the bus adding mode.

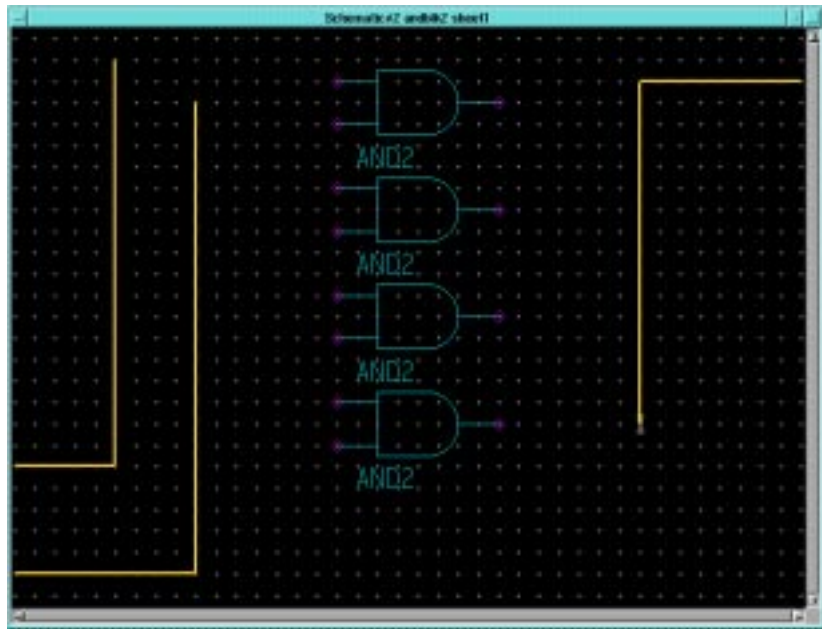


Figure 10-17 ANDBLK2 Schematic with Buses

Adding Nets to a Schematic

Next, nets must be added to attach the appropriate pins on the gates to the buses. You may want to enlarge the view of the components to make it easier to draw the nets.

1. Press the F2 key.
2. Select **Right Mouse Button** → **Wire** from the ADD menu.

A small dialog box appears, and a white cross appears under the cursor.

Note: If the ADD menu does not appear, it may be that something is still selected, resulting in a different menu appearing on the screen. If this happens, press the F2 key and repeat step one.

3. Move the cursor to the top input pin of the top and2 gate, then click the left mouse button.

4. Move the cursor to the left, so that the pointer is laying atop the leftmost bus. (The wire should form a ninety-degree angle with the bus.) Click the left mouse button twice to terminate the wire.

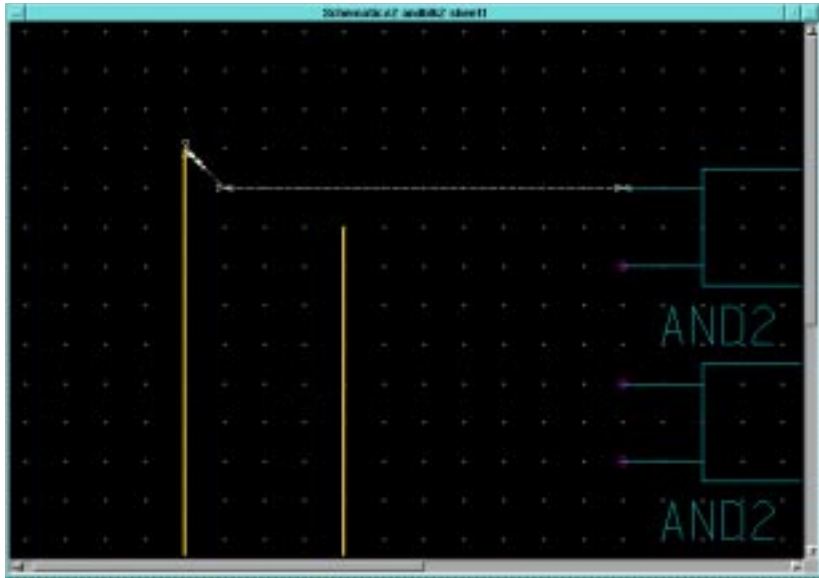


Figure 10-18 Connecting a Net

A bus ripper is inserted automatically between the wire and the bus as shown in the “Connecting a Net” figure. A bus ripper defines which bit of the bus is connected to the wire. Automatically inserting bus rippers is referred to as autoripping.

If the bus ripper did not automatically get inserted, make sure that you clicked on the pin first and then on the bus to attach a net between the two. If the net is attached to the bus first, autoripping does not occur. Also, check the Setup menu to make sure that autoripping is turned on. “Set Autoripping Off” should be displayed in the menu to indicate that autoripping is turned on. If “Set Autoripping On” is displayed, select it to turn autoripping on. Also, the `$MGC_GENLIB` environment variable must be set correctly for the autoripping function.

A dialog box as shown in the “Label Bus Dialog Box” figure below appears. This box allows you to label a net which has a bus ripper connected to it. Labeling is the process of identifying a net or a component by assigning a text string to it. It is recommended

that you label all nets on the schematic, to simplify debugging and simulation. To specify the bus signals they are related to, all nets that are attached to buses must have a number in parentheses at the end of their names. For example, a net that is bit zero of bus A must be labeled A(0).

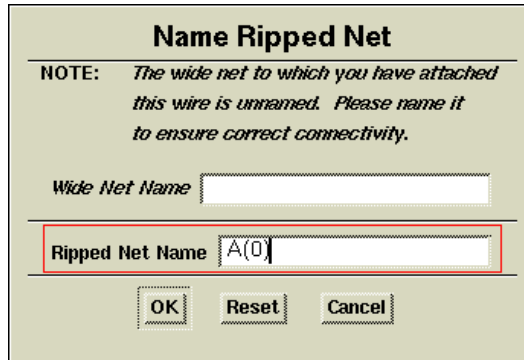


Figure 10-19 Label Bus Dialog Box

Fill out the fields as shown in this example, so that this first net is labeled A(0). Note that you can label the bus as well as the net, but you will do this later.

If the dialog did not appear as shown, choose SETUP RIPPER from the palette and make sure that Ripper Mode is set to "Implicit" and that Ripper Query is set to "On". After setting these parameters, select OK, then delete the net and start again from step one.

5. After filling out the dialog box, you are asked to place the net label, or net name, on the schematic. Place the label as shown in the following figure and click the left mouse button.

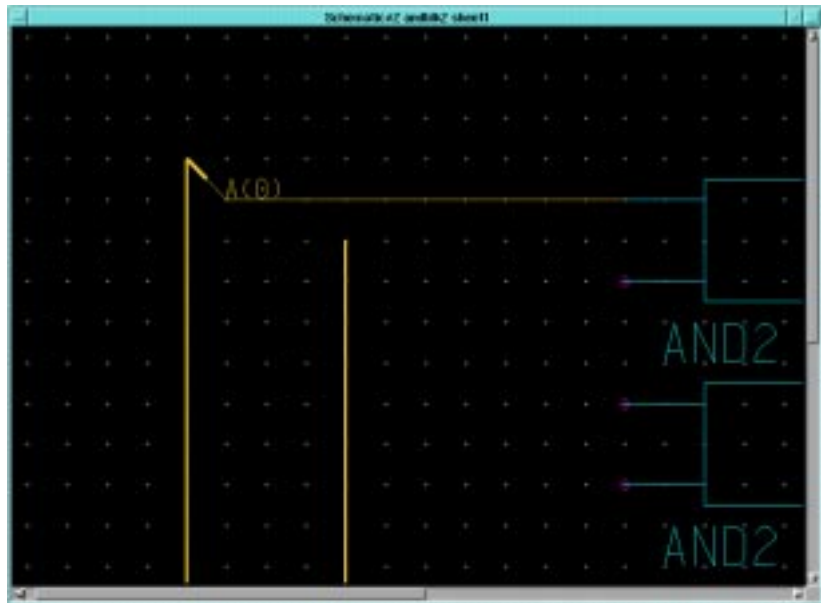


Figure 10-20 Adding and Placing a Net Name

6. Press the Escape key to exit the wire-adding mode.

Completing the Net Connections

Add the remaining nets to the schematic as follows:

1. Press the F3 key to execute the Add Wire command, or use the downward stroke 258.
2. Add the remaining nets as shown in the “ANDBLK2 with All Wires and Buses Connected” figure below.

Note: When a wire is properly attached to a symbol pin, the small diamond that specifies the connection point for the pin disappears. If any of the diamonds are still visible, delete the associated net and reattach it.

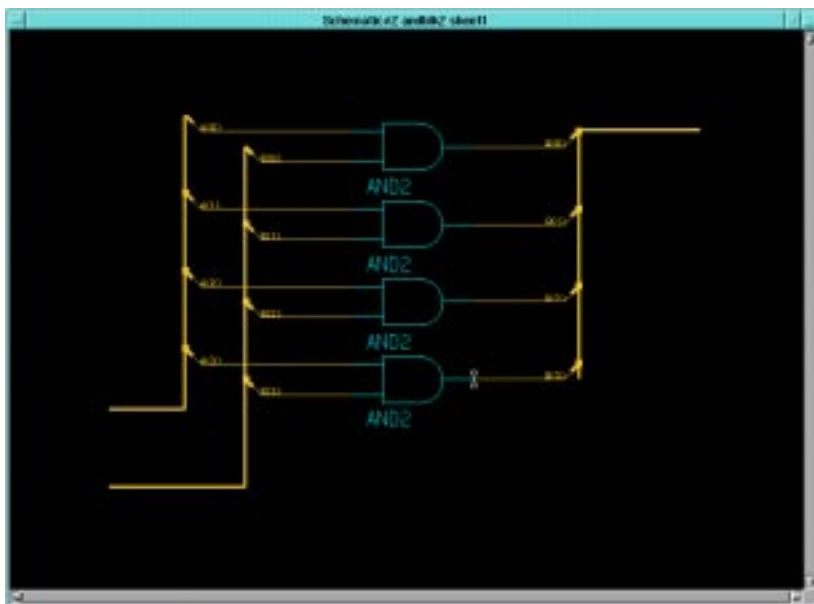


Figure 10-21 ANDBLK2 with All Wires and Buses Connected

Increasing Text Size

At this point, all nets in the ANDBLK2 schematic have been labeled. However, the text of these labels is quite small compared to the other elements in the schematic.

To make the labels more readable:

1. Select all nets in the design by dragging with the left mouse button over the $A(x)$ and $B(x)$ nets then releasing, then doing the same with the $Q(x)$ nets. In each case, a selection marquis appears as shown in the “Selecting Nets” figure. This is an additive selection; elements selected previously remain selected. Note that the marquis need only touch elements you wish to select; it need not enclose these elements completely.

Note: If you accidentally select any elements besides the nets (bus rippers, buses, or gates), press F2 to unselect everything, then repeat the selection procedure.

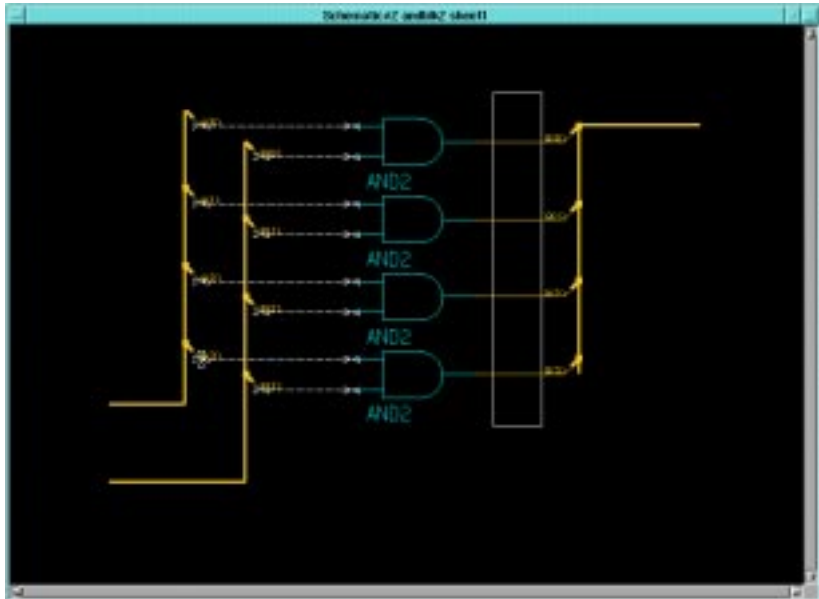


Figure 10-22 Selecting Nets

2. To change the size of the net labels, choose **Right Mouse Button** → **Properties** → **Change Text Height** → **1.0 x Pin Spacing**.
3. In the Change Property Height dialog box that appears as shown below, type **NET** in the Property Name field as indicated to increase the height of all NET properties on all selected elements.



Figure 10-23 Increasing Text Size

All net and bus labels are attached as a property called “NET” with a value equivalent to the net or bus label.

The label sizes are increased as shown in the figure below.

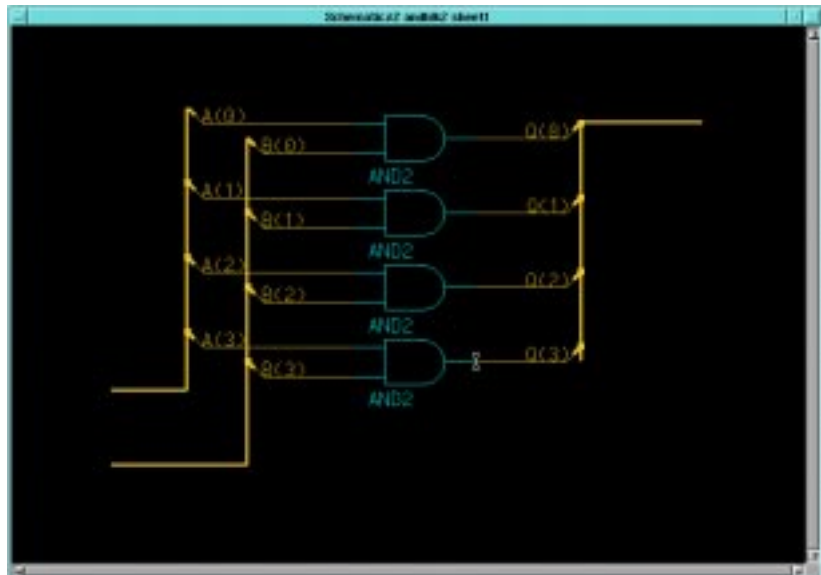


Figure 10-24 Schematic with Larger Net Names

Adding Ports

You must add port symbols to nets and buses to define the connectivity between a schematic and its associated symbol. For the ANDBLK2 schematic, all three buses need ports. Input signals are given PORTINs and output signals are given PORTOUTS.

Add ports to the schematic as follows:

1. If the appropriate Unified library is not displayed in the palette, use the menu bar command **Libraries** → **XILINX Libraries** to select it. Select the Unified Libraries and the appropriate library for the part being used.
2. If the library is already visible, you may need to choose the **BACK** option from the top of the Library Palette to move up to the general library categories. Continue selecting **BACK** until the **ALL PARTS** and **BY TYPE** selections are displayed.
3. Select **BY TYPE**, and then choose the io category.
4. Select the portin library part from the menu.

5. Place the portin so that the white crosshair is *exactly* above the left end of the upper input bus, on the left side of the window.
6. Place another portin at the end of the lower input bus, on the left side of the window.
7. Select a portout symbol from the library and place it at the end of the output bus.
8. Press **Shift-F8** to view the entire schematic.

The schematic appears as in the following figure.

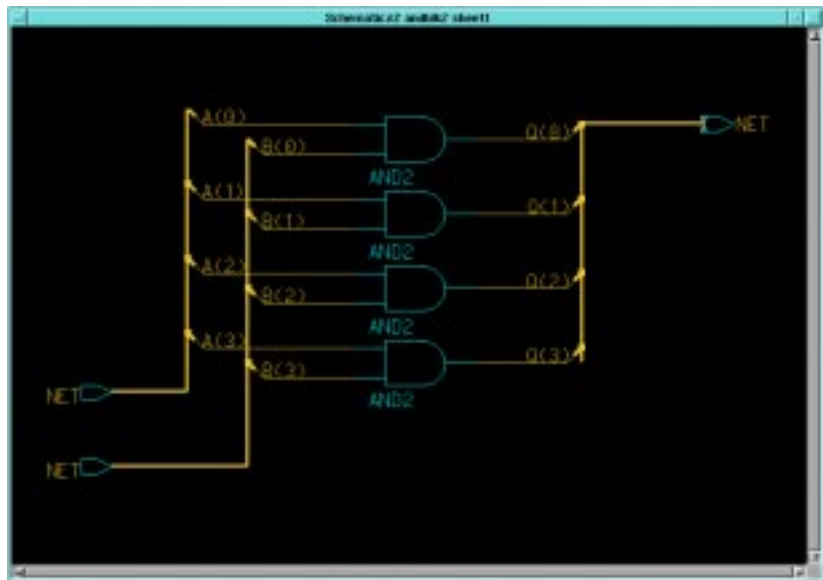


Figure 10-25 Adding Ports

Labeling Ports

Normally, you label nets and buses by selecting them, then executing the menu selection **Right Mouse Button** → **Name Nets** as shown later in this tutorial. However, the addition of the port symbols to the buses has automatically assigned a default name of “NET” to each bus. This simplifies the process since you can modify the existing names rather than add new ones.

1. Press the **F2** key to unselect everything on the schematic sheet.

2. Move the cursor so that it sits above the NET label on the output bus.
3. Press **Shift-F7** to choose the Text Change Value command.
A small dialog box appears.
4. In the New Value field, change the text to Q(3:0).
5. Press return or choose **OK** in the dialog box.
6. Repeat this procedure on the two remaining buses, giving them names as shown in the following figure.

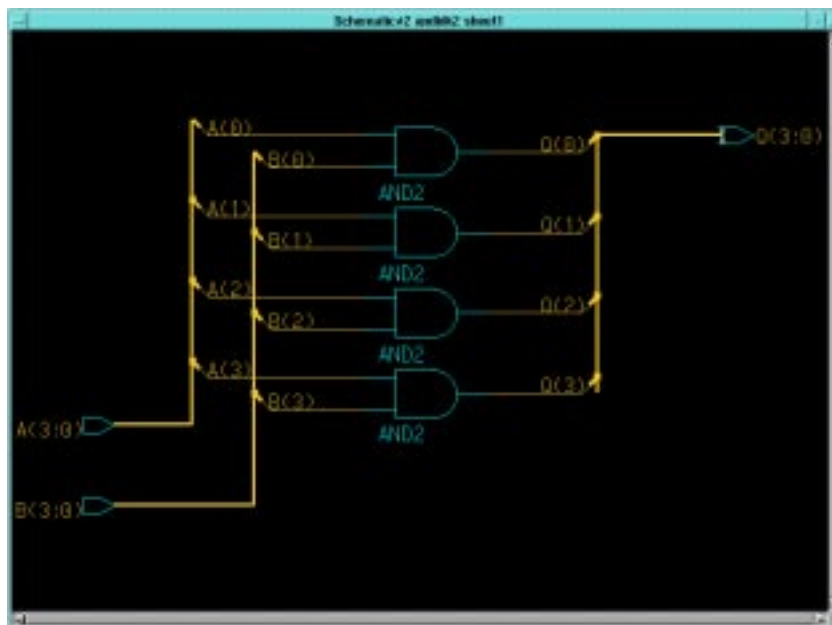


Figure 10-26 Labeling Buses

Saving the Schematic

The schematic is now complete. Check and save the schematic as follows:

1. Select **Check** → **Sheet**.

The text window that appears should not contain any warnings about unnamed or dangling net vertices, since all pins in the schematic should be connected.

2. If these or any other warnings or errors occur, recheck the schematic against the following figure.

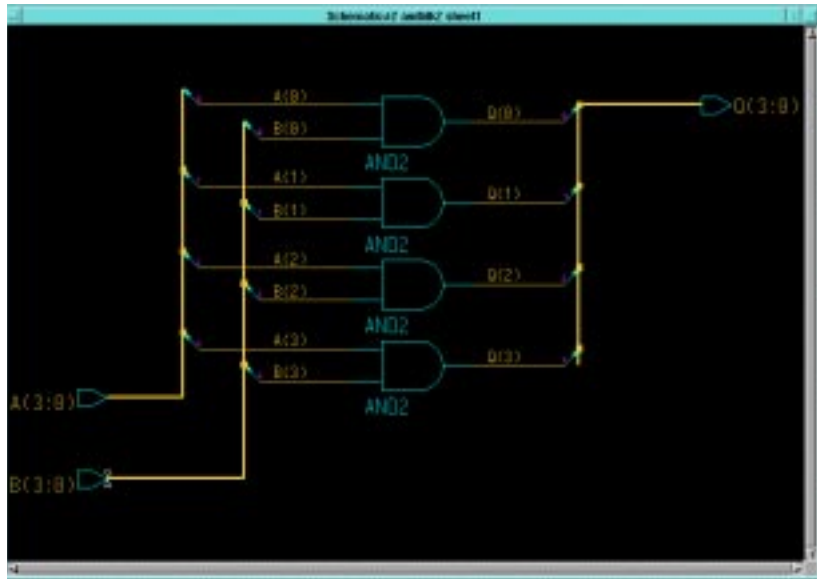


Figure 10-27 Completed ANDBLK2 Schematic

The check sheet window is also linked to the schematic window. Any net, vertex, or instance names can be highlighted in the check sheet window by clicking the left mouse button on it. The corresponding net, vertex, or instance on the schematic is highlighted. This is useful for relating an error message in check sheet to the schematic.

3. Once all schematic errors have been corrected, check the design again if necessary, and close the check sheet text window.
4. Select **File** → **Save Sheet** from the menu bar to save the schematic.

Creating Schematics for ORBLK2 Symbol

The ORBLK2 schematic is similar to the ANDBLK2 schematic. To create schematics for the ORBLK2 symbol, you can use the ANDBLK2 schematic and simply replace the four and2 gates with or2 gates as described in the following procedure.

1. Press the **F2** key to unselect everything on the ANDBLK2 schematic.
2. Display the **BY TYPE** library menu and select the logic category.
3. Press and hold the left mouse button and move the mouse to create a rectangle to include part of all four and2 gates, as shown in the following figure. It is not necessary to box the entire gate to select it. Do not include any part of the attached nets in the rectangle.

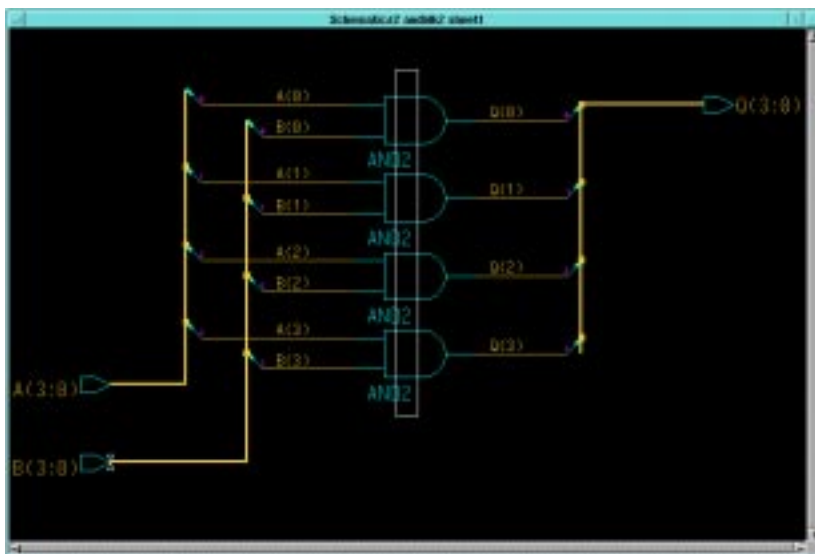


Figure 10-28 Selecting Gates

4. When the rectangle is positioned correctly, release the left mouse button to select all four and2 gates.
5. Select **Right Mouse Button** → **Replace** → **From Library Menu**.

A message appears at the bottom of the screen requesting that you select the replacement library part from the menu.

6. Use the **PageUp** and **PageDown** keys to scroll the component list. Select the or2 component.

The four and2 gates are replaced with or2 gates. The ORBLK2 schematic is complete.

7. Select **Check** → **Sheet** from the menu bar.
The check program refers to the ANDBLK2 schematic, since this was modified to create the ORBLK2 schematic.
8. Close the text window containing the results of check sheet.
9. Select **File** → **Save Sheet As**.
10. In the dialog box that appears, type `$XILINX_TUTORIAL/calc_sch/orblk2` in the Component name field and leave all other fields blank.
11. Press return to save the schematic.

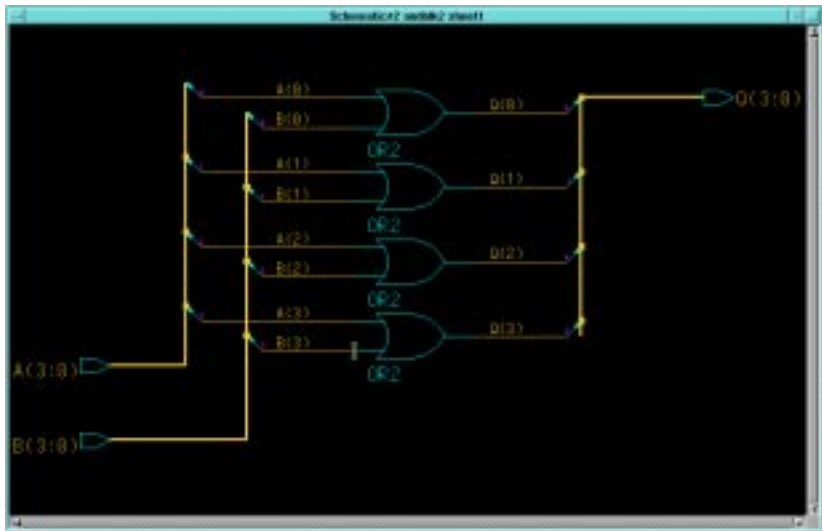


Figure 10-29 Completed ORBLK2 Schematic

Editing the ALU Schematic

So far you have created symbols for ANDBLK2 and ORBLK2. You have also created underlying schematics for these symbols. The next step is to place the symbols in the ALU block schematic.

1. Close the only open window, which is the modified ANDBLK2 schematic, using the button in the upper left corner of the window.

2. In the dialog box that appears asking whether to save the changes to the schematic, select **No**, since the ANDBLK2 schematic was saved earlier, but then modified for use as the ORBLK2 schematic.
3. Choose **OPEN SHEET** from the Session Palette.
4. Press the **Navigator** button to open a Navigator window.
5. If necessary, change directories to the `$XILINX_TUTORIAL/calc_sch` directory using the up arrow to move up one directory level and double-clicking folders to push into them. Then, select the Calc design, which is represented by a folder with a “c” on it and with the name “calc” next to it. The “c” specifies that it is a component, and not just a directory.
6. Press **return** or select **OK** from the Navigator window.

The Component Name field of the OPEN SHEET dialog box is automatically back-filled with “`$XILINX_TUTORIAL/calc_sch/calc`”.
7. Press **return** or select **OK** from the OPEN SHEET dialog box. The top level Calc design appears in a window. Press Shift - F8 to view the entire schematic, if necessary.
8. Press **F2** key to unselect everything on the schematic.
9. Select the **ALU** symbol.
10. The additions you need to make are all in the ALU schematic, so choose **File** → **Open Down** from the menu bar using the left mouse button.

The Open Down dialog box appears as shown in the following figure.

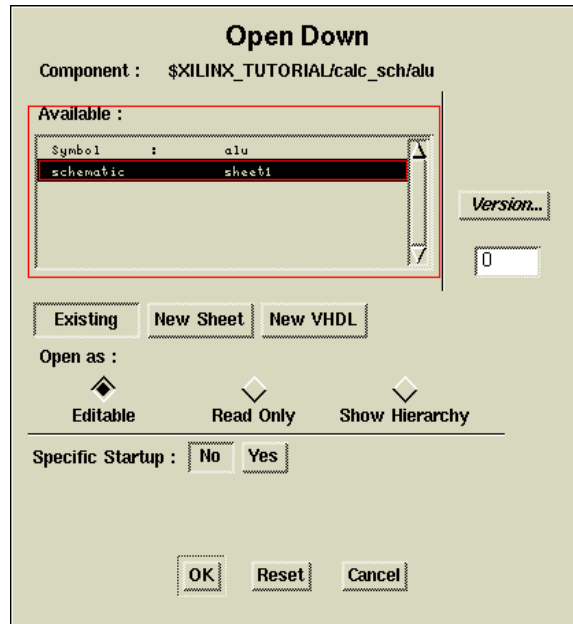


Figure 10-30 Open Down Dialog Box

11. In the Open Down dialog box, specify whether to modify the symbol or the schematic for ALU by selecting **schematic sheet1**, with the left mouse button. The selected line appears highlighted in the dialog box.
12. Press **return** or select **OK**.

A second schematic window appears containing the ALU schematic.

Placing User-Created Components

You can now place the ANDBLK2 and ORBLK2 symbols on the schematic as shown in the figure below. You place the symbols using the same procedure you used to place the and2 gate from the Xilinx libraries when you created the ANDBLK2 schematic.

1. Use the **F8** key to zoom into the empty area near the center of the schematic, between the XORBLK2 and ADSU4 symbols.
2. Press the **F2** key to ensure that nothing is selected.

3. Choose **Right Mouse Button** → **Instance** → **Symbol by Path**.

In the Add Instance dialog box that appears, use the Navigator button to select the \$XILINX_TUTORIAL/calc_sch/andblk2 component, or type the name in the Component Name field.

4. Press **return** or select **OK** to execute the command.
5. Move the cursor to the correct location as shown in the following figure.

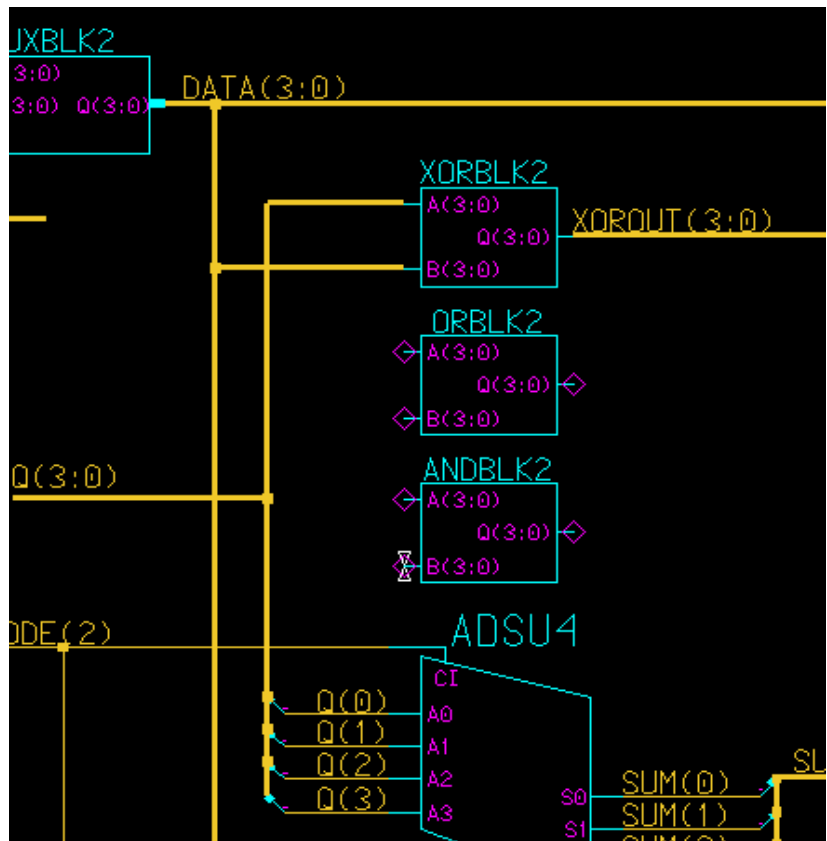


Figure 10-31 Adding ANDBLK2 and ORBLK2 to ALU Schematic

6. Press the left mouse button to place the component.
7. Follow the same procedure to add the ORBLK2 symbol. Refer to the ALU schematic in the figure above for proper placement.

8. If you make a mistake when placing a component, select it (after pressing F2 key) and use **Right Mouse Button** → **Move** to reposition it.

Placing Library Components

The next step in the tutorial is to add the fd4re and and5b2 components to the ALU schematic. The fd4re component is available in the Xilinx Unified Libraries and consists of four flip-flops with a clock enable. The and5b2 component is a five-input AND gate with two inputs inverted (“bubbled,” hence the “b”).

Note: These components are available in all libraries, including those for the XC4000E and XC9000.

1. Use the **shift -F8** keys to display the entire ALU schematic.
1. Use the **F8** key to zoom into the open area in the lower right-hand corner.
2. Select **Libraries** → **XILINX Libraries** from the menu bar.
3. Select the Unified Libraries and the appropriate family library using the left mouse button.
4. Choose **BY TYPE** → **flip_flop** → **fd4re** from the Library menu.
5. Move the cursor into the schematic window.
An outline of the fd4re component appears.
6. Move the component to lower right corner of the schematic, approximately to the location shown the “Adding FD4RE and AND5B2 to ALU Schematic” figure.
7. Press the left mouse button to place the component.
8. Repeat steps two through six to place the and5b2 component next to the fd4re as shown in the figure below.

When choosing the component from the library menu, use the selection **BY TYPE** → **logic** → **and5b2**.

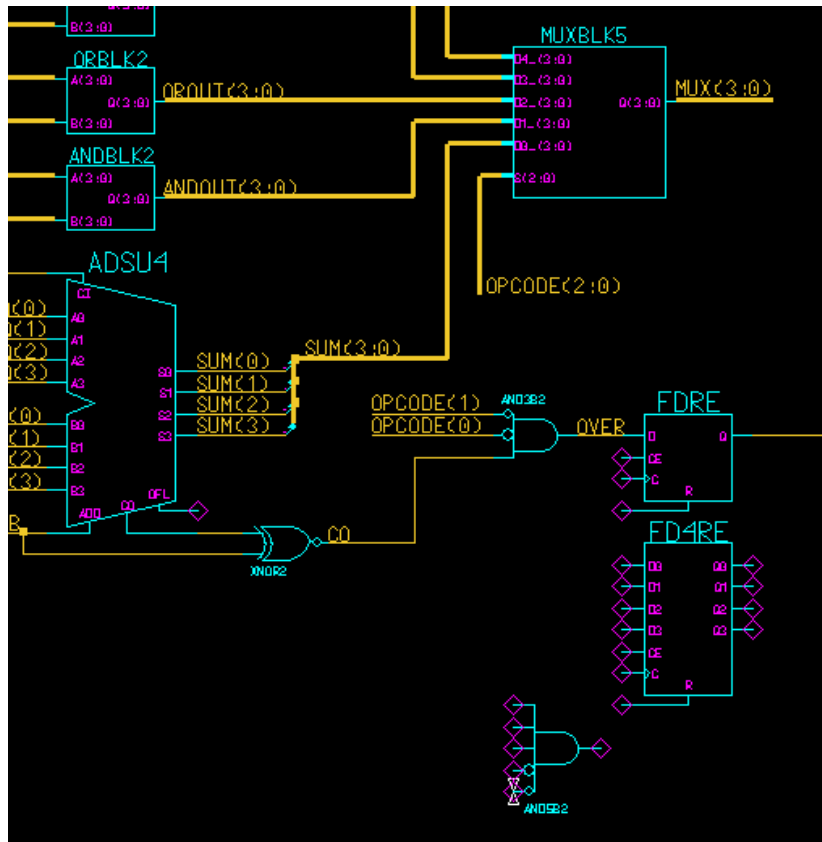


Figure 10-32 Adding FD4RE and AND5B2 to ALU Schematic

Adding Nets, Buses, Ports and Labels

FD4CE and AND5B2

Next, complete adding the fd4re and and5b2 symbols by adding nets, buses, and labels as follows:

1. Add the necessary nets and buses to complete connections for fd4re and and5b2 as you did for the previous schematic.

The figure below displays the labeled nets and buses for fd4re and and5b2.

2. Add ports to the nets and buses attached to the fd4re and and5b2, as shown in the figure below.
3. Change the default “NET” properties to the proper names using the **Shift-F7** key, as shown in the following figure.
4. To add net labels to nets not connected to ports or buses, select the nets as described earlier, then select **Right Mouse Button** → **Name Nets**.

For each selected net, the schematic editor asks you for a net label, then offers you the opportunity to place each label on the schematic. The nets should be labeled as shown.

5. Increase the size of the label text as described earlier.

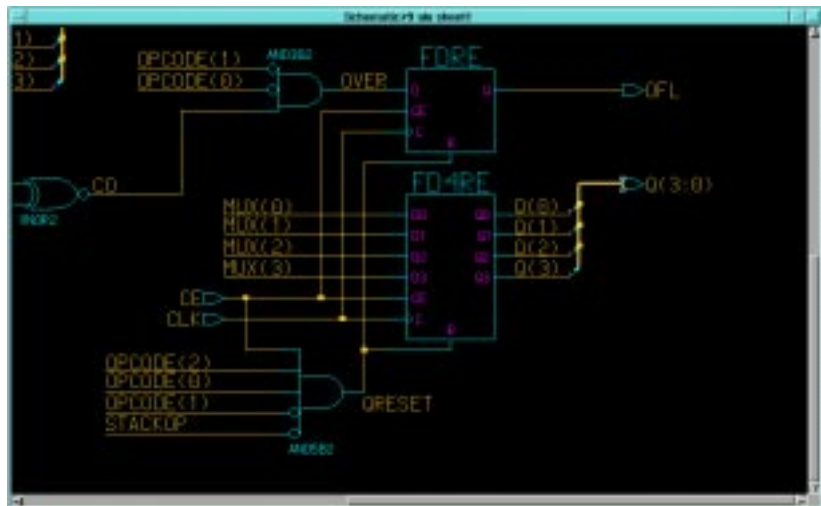


Figure 10-33 Nets, Buses, and Ports for FD4RE and AND5B2

ANDBLK2 and ORBLK2

Next, complete the addition of ANDBLK2 and ORBLK2 to the ALU schematic.

1. Add the necessary buses to complete connections for ANDBLK2 and ORBLK2. The figure below displays the labeled nets and buses for ANDBLK2 and ORBLK2.
2. Use the figure below to name the added buses by using the same **Right Mouse Button** → **Name Nets** from the previous

section. You only need to label the output buses of the two components, since the inputs to these components are connected to pre-labeled buses.



Figure 10-34 Nets, Buses and Labels for ANDBLK2 and ORBLK2

Adding Labels to Components

It is important to add labels to components. Error and warning messages often reference component labels, and labels also appear in simulation netlists. Also, net names at lower levels of hierarchy are referenced using the following format:

```
...component_label/component_label/net_label
```

In the ALU schematic, labels have already been added to the MUXBLK2, XORBLK2, and MUXBLK5 blocks.

To add a label to the ANDBLK2 placement, follow these steps.

1. Press the **F2** key to unselect everything.
2. Use the left mouse button to select the ORBLK2 symbol.
3. Select **Right Mouse Button** → **Properties** → **Add**.

A dialog box appears.

4. In the window labeled “Existing Property Name”, choose the `INST` property with the left mouse button. It appears highlighted.
5. In the Property Value field, type `ORBLK2`, then press `return` or choose `OK`.
6. Move the text to position it as shown in the following figure and click the left mouse button to place the text.
7. Label the `ANDBLK2` symbol the same way using the label `ANDBLK2`, as shown in the following figure.
8. Give the `FD4RE` component the label `ALUVAL`.

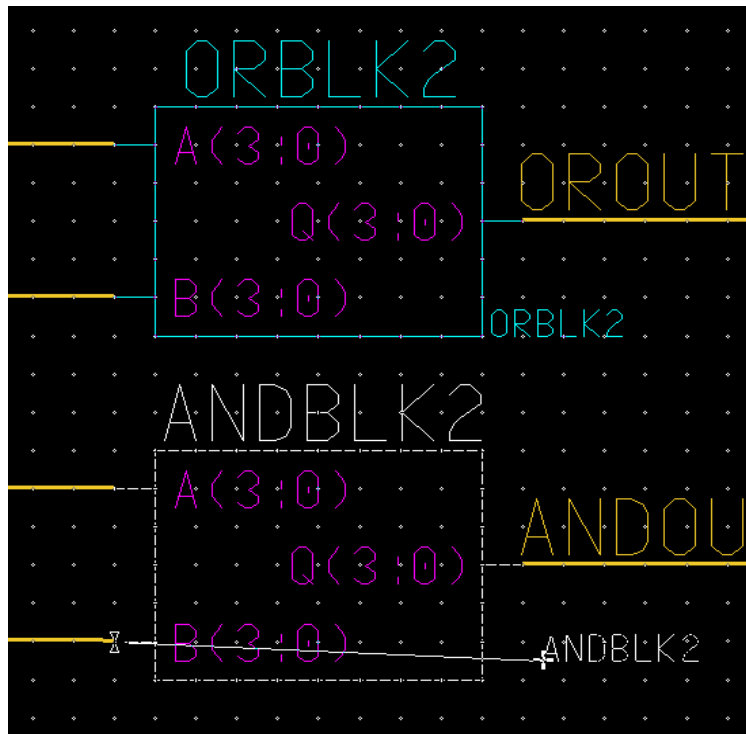


Figure 10-35 Adding Component Labels to ALU Schematic

The completed ALU schematic is shown in the following figure.

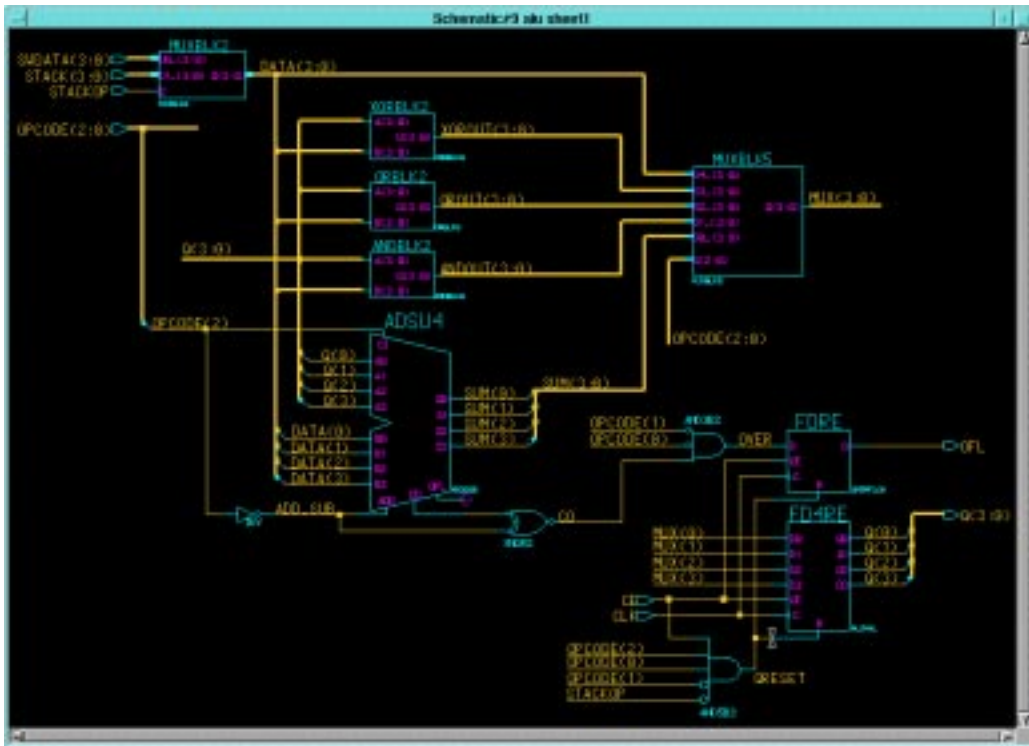


Figure 10-36 Completed ALU Schematic

Saving the ALU Schematic

Check the schematic. If errors occur, resolve them and then check and save the schematic.

Exploring Xilinx Library Elements

The Xilinx libraries contain three types of elements. Primitives are basic logic elements such as the and2 and or2 gates that you previously placed in ANDBLK2 and ORBLK2. Soft macros are schematics created by combining primitives and other soft macros. Relationally Placed Macros (RPMs) are soft macros that contain placement information. RPMs are currently only available in the XC4000E library.

All three types of library elements are placed on a schematic in exactly the same way.

Viewing a Xilinx Soft Macro Schematic

Soft macro schematics are schematics such as you might make for your own designs. In fact, you can load one of these schematics and use the File Save As command to save it under another name, and then edit this new schematic to customize it to your needs.

Open the schematic underneath the fd4re symbol as follows:

1. Press the **F2** key to unselect everything.
2. Select **fd4re** with the left mouse button.
3. Select **File** → **Open** **Down** from the menu bar.
4. In the dialog box that appears, select the schematic sheet and click **OK**.

As shown in the following figure, fd4re consists of four fdre symbols.

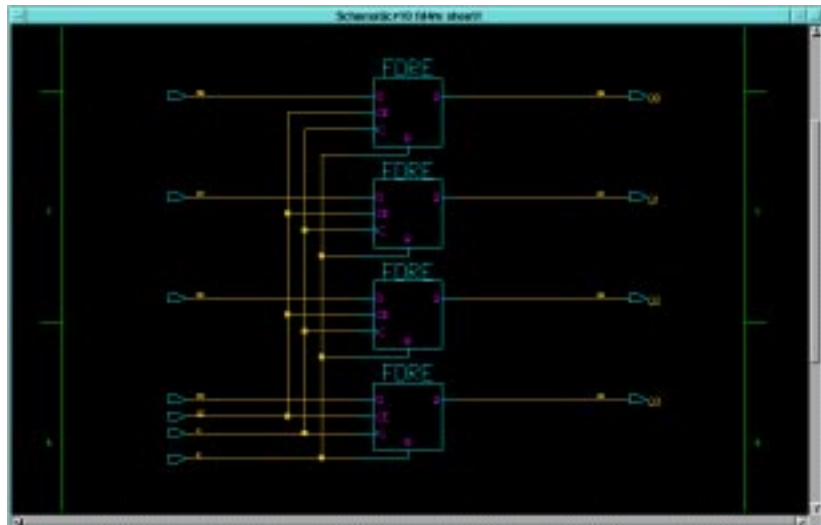


Figure 10-37 FD4RE Schematic from XC4000E Library

Viewing a Xilinx RPM (XC4000-Based Families Only)

Note: The following description of RPMs contains detailed information on the XC4000E architecture. Refer to *The Programmable Logic*

Data Book for more information on the XC4000E CLB structure and fast carry logic.

If your design is not targeted for the XC4000E family, read this section, but do not perform any of the commands. Continue the tutorial with the “Opening the Calc Schematic” section (the next section).

Note: The XC5200 library also contains RPMs. If you have an XC5200 schematic, you may open the ADSU4 component as described here to see how this RPM is implemented in that family.

The ALU contains a component from the Xilinx library, adsu4, which is a four-bit wide adder/subtractor. If your design is targeted for the XC4000E library, this schematic is implemented as a Relationally Placed Macro (RPM). If your design is not targeted for the XC4000E or XC4000EX library, adsu4 is implemented without this placement information.

RPM schematics are schematics such as you might make for your own designs. In fact, you can load one of these schematics and use the File Save As command to save it under another name. You can then edit this new schematic to customize it to your needs.

Elements placed in the ADSU4 RPM schematic include CY4 components and FMAPs. The CY4 symbol gives you the ability to specify fast carry logic functionality from the schematic. Fast carry logic is a hardware feature in XC4000E parts that allows very fast arithmetic-type functions.

The FMAPs map logic functions to function generators in Configurable Logic Blocks (CLBs), which are arranged in a rectangular grid in the die. Both CY4 symbols and FMAP symbols have RLOC attributes. RLOCs are attached to the symbols that assign relative locations to the CLBs. You can use carry symbols as well as FMAPs and other mapping components in your own schematics. However, knowledge of them is not necessary to use RPMs. Only expert users should create macros containing carry logic and FMAPs. For a description of these components, see the XACT Libraries Guide.

Push into the ADSU4 schematic as follows:

1. Press **F2** key.
2. Select **ADSU4**.
3. Open the schematic underneath adsu4.

- Use the **F8** key (or stroke 159) to zoom into the upper portion of the schematic as shown in the following figure.

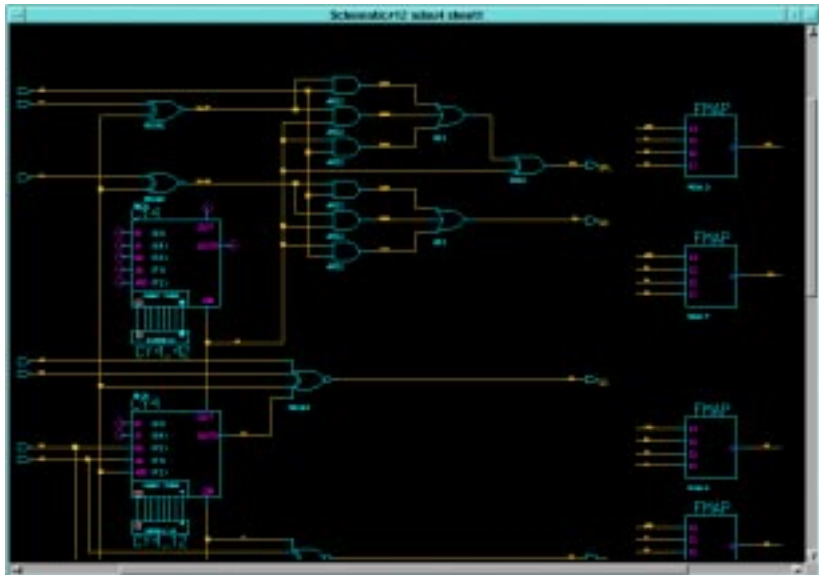


Figure 10-38 Upper Portion of the ADSU4 RPM Schematic

- Press **F2** key to unselect everything.
- Select the FMAP component in the upper right corner.
- Select **Report** → **Object** → **Selected** → **All**.

A text window appears displaying the attributes on the symbol, as shown in the following figure. The RLOC attribute is set to R0C0.G, indicating that this function is mapped to the G function generator of the upper-left corner (row zero, column zero) CLB in the RPM. RPM origins are in the upper left-hand corner. (You can also call up a report with the 1474123 stroke, which looks like a lowercase “r”.)

Reporting: Instance, Net, Pin, Property, Comment, Frame,
Attached pin, Attached property,

Instance	Name	Property	Name	Value	Type
I\$5	\$LCA/xc4000e/fmap/fmap	TS\$703	LIBVER	2.0.0	string
		TS\$700	MODEL	null	string
		TS\$699	COMP	FMAP	string
		TS\$698	INST	I\$5	string
		TS\$697	RLOC	R0C0.0	string
Pins:	Name	Property	Name	Attached Vertex	of Net
P\$704	I1	TS\$706	PIN	V\$193	N\$21
		TS\$705	PINTYPE	in	
P\$707	I2	TS\$709	PIN	V\$198	N\$22
		TS\$708	PINTYPE	in	
P\$710	I3	TS\$712	PIN	V\$203	N\$23
		TS\$711	PINTYPE	in	
P\$713	I4	TS\$715	PIN	V\$208	N\$24
		TS\$714	PINTYPE	in	
P\$716	0			V\$212	N\$25

Figure 10-39 RLOC Attribute on FMAP Component

8. Close the text window to return to the adsu4 schematic window.
9. Use the scroll bars on the sides of the window or double-click the middle mouse button to pan around the schematic and look at the RLOCs.

Note that logic is mapped to three CLBs, designated as R0C0, R1C0, and R2C0. Therefore, this RPM uses three CLBs that are arranged in a column. Information on the number of CLBs used and the shape of the logic block is available for each RPM in the XACT Libraries Guide. These locations are relative, not absolute.

The macro is not defined as placed in the uppermost CLB in the left most column. Regardless of the RPM's absolute location, the logic associated with the FMAP with the location R0C0 is always at the top, R1C0 is in the CLB directly below, and so on.

10. Close the adsu4 schematic and return to the ALU schematic.

Opening the Calc Schematic

Close all open schematic or symbol windows except for the top-level Calc schematic window. If the Calc window is closed, open it. The Calc schematic appears on the screen.

Using the XC4000E Oscillator

If your design is not targeted for the XC4000E family, read this section, but do not perform any of the commands.

The XC4000E family devices contain an on-chip clock generator, which makes it unnecessary to use an external circuit for this purpose. The on-board clock circuitry is not precise, but is suitable for designs that do not need a highly accurate clock, such as the Calc design.

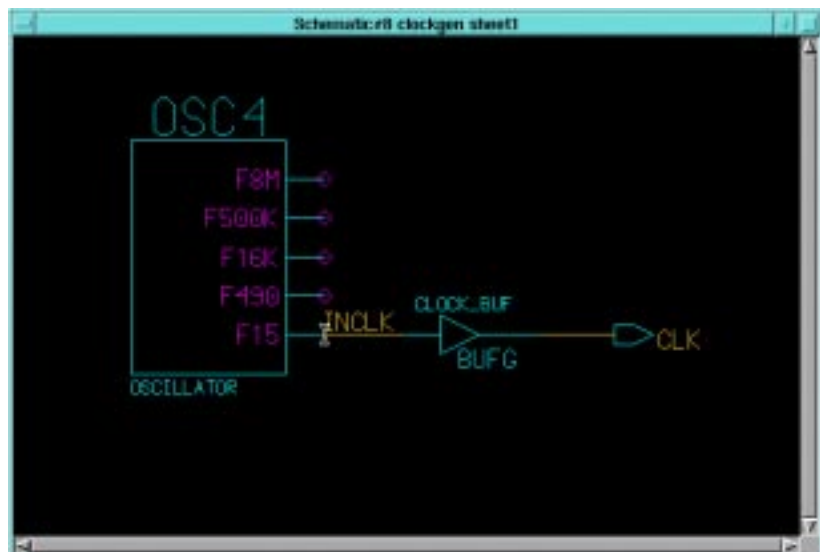


Figure 10-40 CLOCKGEN Schematic

The CLOCKGEN schematic contains an XC4000E library part, OSC4. This symbol represents the on-chip oscillator that generates nominal clock frequencies of 8 MHz, 500 KHz, 16 KHz, 490 Hz, and 15 Hz. The Calc design uses the 15-Hz output from this component when targeted for XC4000E family designs. The clock output from OSC4 is buffered through a BUFG global clock buffer.

XC4000E family devices have eight on-chip clock buffers, one BUFGP (primary global buffer) and one BUFGS (secondary global buffer) in each corner of the device. Although it is possible to use them for other purposes, BUFGPs are best used to route externally-generated clock signals. BUFGSs have more flexibility, and can be used to route any large fan-out net, even if it is internally sourced. A BUFG symbol can represent either type of buffer, and allows the implementation software to choose which type of global buffer is best in each situation. BUFG also facilitates design retargeting to other Xilinx device families, since it can represent any type of global buffer in any family. The BUFG in the Calc design is substituted for a BUFGS during design implementation, because the clock is generated internally by the on-chip oscillator. See the *Xilinx Libraries Guide* and the *The Xilinx Programmable Logic Data Book* for more information on global clock buffers for Xilinx devices.

Controlling FPGA/CPLD Layout from the Schematic

Assigning Pin Locations

It is highly recommended that you let the automatic placement and routing program, PAR, define the pinout. Pre-assigning locations to the I/Os can sometimes degrade the performance of the place and route tools. However, it is usually necessary, at some point, to lock the pinout of a design so that it can be integrated into a board design. You should define the initial pinout by running the place and route tools without pin assignments, then locking down the I/O placement so that it reflects the locations chosen by the tools. I/O in the tutorial schematics must be assigned pin locations so that the Calc design can function in the Xilinx demonstration boards. Because the design is fairly simple, these pin assignments do not adversely affect the ability of PAR to place and route the design completely.

Pin locations are specified by attaching a LOC property to the net attached to the pad. LOC properties should not be attached directly

to I/O pads. Properties are not associated with nets, only with vertices on nets. When attaching properties, if the center of a net is selected, the entire net segment appears highlighted, indicating that two net vertices are selected, one at each end of the net segment. If a property is then attached to the net, it appears twice when placed, indicating it has been attached to both net vertices associated with the segment. While this is not illegal, it does clutter the schematic. To prevent this, select only one vertex before attaching properties. To select a net vertex, position the cursor exactly above the point where the net attaches to the pin, or above the point where the net bends. Otherwise, an entire net segment is selected. This operation is simplified because default pin locations are included with the I/O pins; for example, the “PXX” on the OPAD symbols. You can modify the existing property, rather than adding a new one.

Modify the LOC property on the pad associated with the STACKLED(0) signal on the Calc schematic as follows:

1. Position the mouse over the “PXX” text to the right of the pad attached to net F; this is the default location property attached to the net. Refer to the following figure.

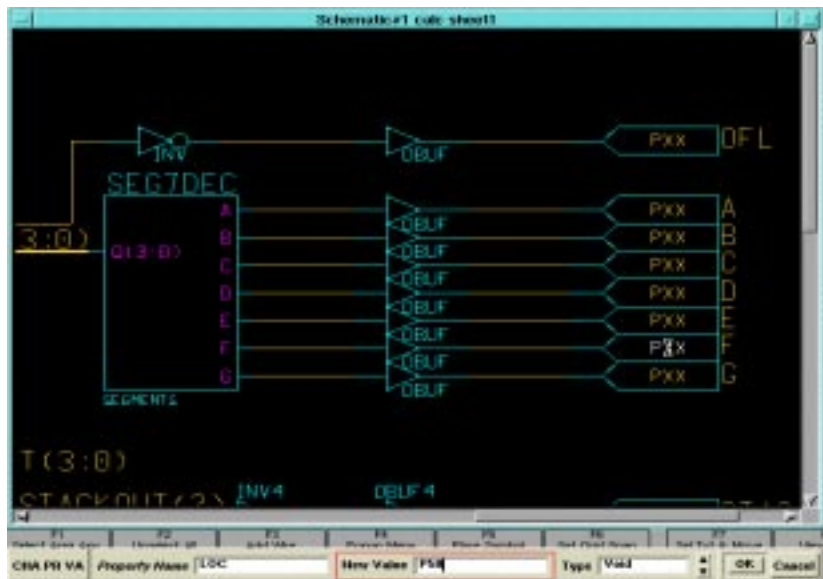


Figure 10-41 Assigning a Location to an Output Net

2. Without moving the mouse, press **Shift-F7**.

3. In the dialog box that appears, modify the “PXX” text to read P50.
4. Click **OK** or press return to execute the command.

For simplicity, the other pin locations for the Calc design have been placed in a data file known as a constraint file, which is described in a later section. You can leave the other location values undefined. Valid pin locations vary depending on the package. PLCC, HQFP, and other “numeric-only” package pins are designated with a P followed by the pin number, such as P17. PGA and other grid-array package pins use alphanumerics such as A12. The Programmable Logic Data Book lists the pinouts of each FPGA and CPLD for each package that Xilinx supplies.

Designating FAST Pads

You can modify the output slew rate by assigning a FAST attribute to the output buffer, as shown in the “Designating a FAST Pad” figure. The default slew rate is SLOW. “Fast” pads have different timing specifications and draw more current than “slow” (slew-rate-limited) pads. Slow pads are used by default. See *The Xilinx Programmable Logic Data Book* for timing specifications for the various slew rate modes.

Add a FAST attribute to the led output display drivers attached to the STACK(3:0) bus as follows:

1. Press **Shift-F8** to display the entire Calc schematic.
2. Click the left mouse button on the **OBUF4** symbol attached to the stack (3:0) bus.
3. Select **Right Mouse Button** → **Properties** → **Add**.
4. In the dialog box that appears, type the word **FAST** in *both* the Property Name and Property value fields. (This double entry is applied to any property that does not take a value.)
5. Press return or select **OK** to execute the command.
6. Use the left mouse button to place the text near the OBUF4 symbol, as shown in the following figure.

Since the property is attached to the OBUF4 symbol, it affects all four of the LED outputs.

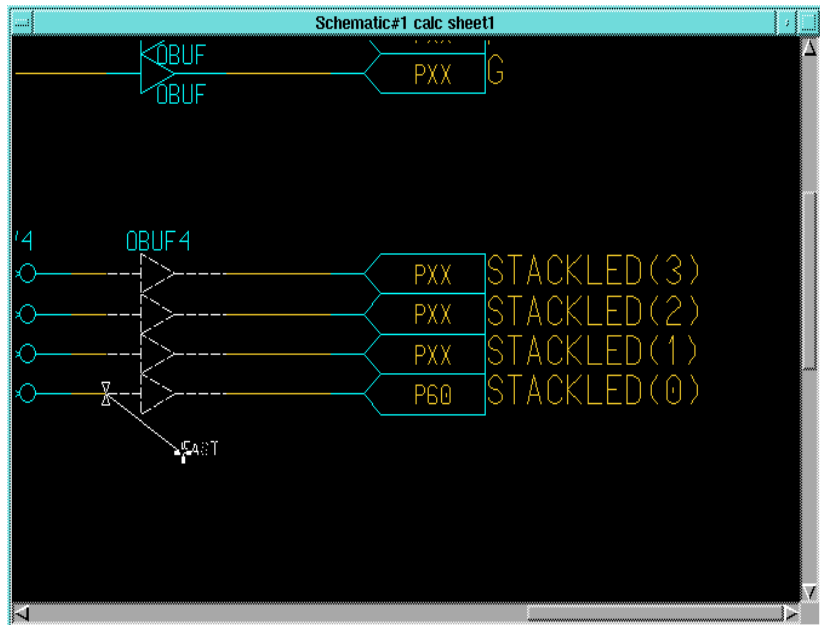


Figure 10-42 Designating a FAST Pad

Using the I/O Flip-Flops

Xilinx XC3000A and XC4000E devices have two flip-flops in each Input Output Block (IOB). Each pad has an associated input flip-flop and output flip-flop. You can also configure input flip-flops as latches and output flip-flops as 3-state. You access these elements using the library components IFD, ILD, OFD, and OFDT, as well as other higher-level macros that contain these components. For more information on these library elements, consult the *Xilinx Libraries Guide*.

IOB flip-flops are used whenever possible to free up internal CLB resources. IOB flip-flops are used to register the switch inputs. As shown in the figure below, the SWITCH7 macro attached to the input bus SW(7:0) in the lower-left area of the schematic has an underlying schematic that consists of seven IFD (input flip-flop D-type) Xilinx primitives. If similar flip-flops, such as FDs, had been used instead, the flip-flops in the IOBs would be wasted and would occupy valuable CLB resources.

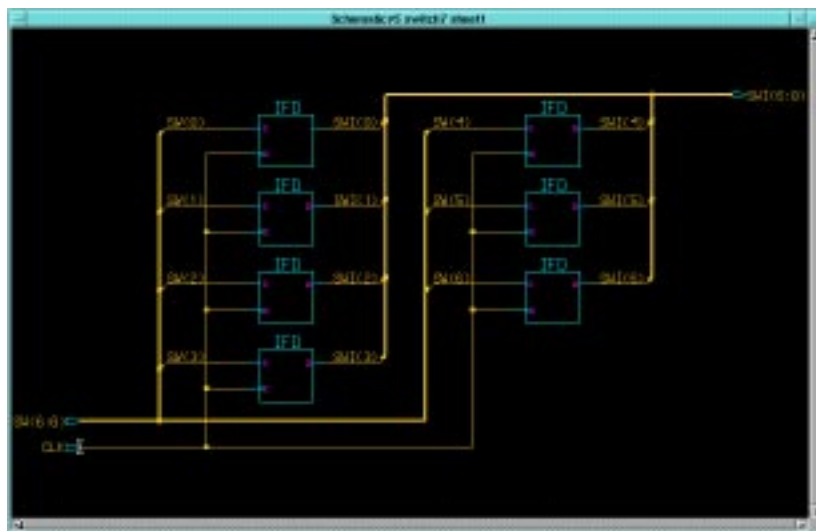


Figure 10-43 SWITCH7 Schematic Using Input Flip-Flops

Saving the Calc Schematic

Before continuing, check and save the changes made to Calc, as shown earlier in this tutorial.

Modifying the Design for Non-XC4000E/EX Devices

At this point in the tutorial, you have created or edited the following four schematic files: calc, alu, andblk2, and orblk2. The design, at this point, is suitable for use only in an XC4000E or XC4000EX device. This is because these devices have several advanced features not found in other Xilinx device families. Two of these advanced features are the on-chip memory built into the XC4000E CLB and wide-edge decoders.

RAM Stack Implementation

The RAM stack is implemented using a 16x4 RAM macro from the XC4000E library. Although the stack is 4x4, RAM and ROM are only available in 16x1 or 32x1 increments, so only one fourth of the memory addresses are used. A stack four times as deep could be implemented while still using only two CLBs. An equivalent flip-flop

implementation would require 64 flip-flops or 32 CLBs. In this case, with a stack only four words deep, using the static memory feature of the XC4000E CLB still reduces the stack from eight CLBs to two CLBs.

To view the XC4000E stack implementation, follow these steps:

1. Make sure the STACK symbol is selected in the Calc schematic, and select **Right Mouse Button** → **Open Down**.
2. Choose the schematic to modify and click on **OK**.
3. On the stack schematic is a RAM16X4S component, which represents four 16x1 synchronous RAMs. Select this component and **Open Down** into its schematic.

The schematic for RAM16X4S is shown below.

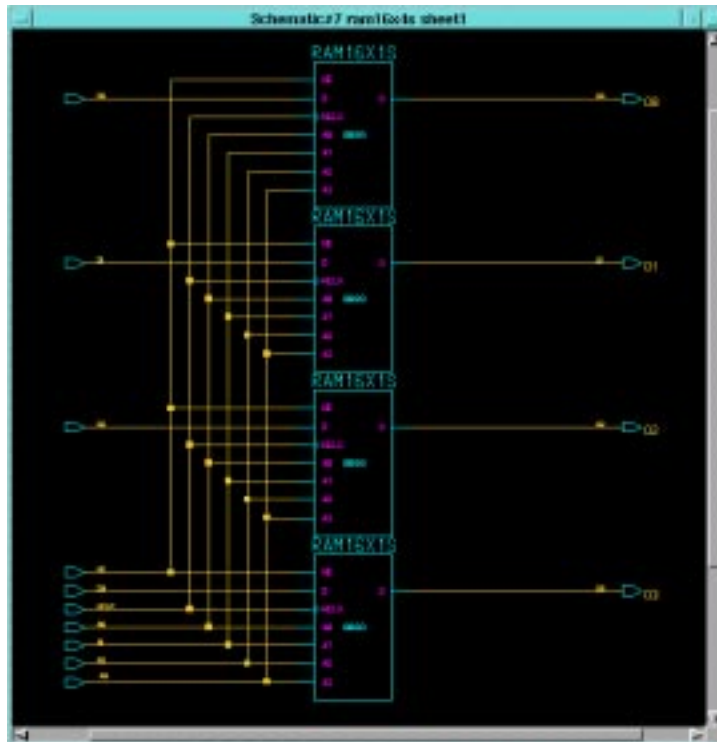


Figure 10-44 RAM16X4S, XC4000E Implementation

Using the Device-Independent Register File

The device-independent stack is implemented by replacing the RAM16X4S with a register file that emulates a synchronous RAM with a set of flip-flops and multiplexers. This implementation can be used for any Xilinx device, even one from the XC4000E family.

If you are targeting an XC4000E device, you may skip this section to take advantage of the RAM feature of the XC4000E.

Make the stack a device-independent schematic as follows:

1. Return to the stack schematic and use the left mouse button to select the RAM16X4S.
2. Select **Right Mouse Button** → **Replace** → **Other**.
3. In the Replace Instance dialog box that appears, fill out the fields as indicated in the following figure and click **OK**.

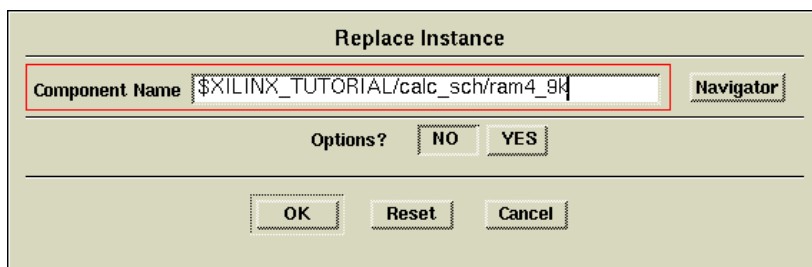


Figure 10-45 Replace Instance Dialog Box

The RAM16X4S is replaced with the device-independent RAM4_9K as shown below. Note that the label RAM_BLOCK has been removed by Design Architect. Add an INST property to this component in the same manner as was done to the components in the ALU schematic.

Note: If you are targeting a device family other than the XC9000, be sure to run Convert Design on the RAM4_9K schematic as described before.

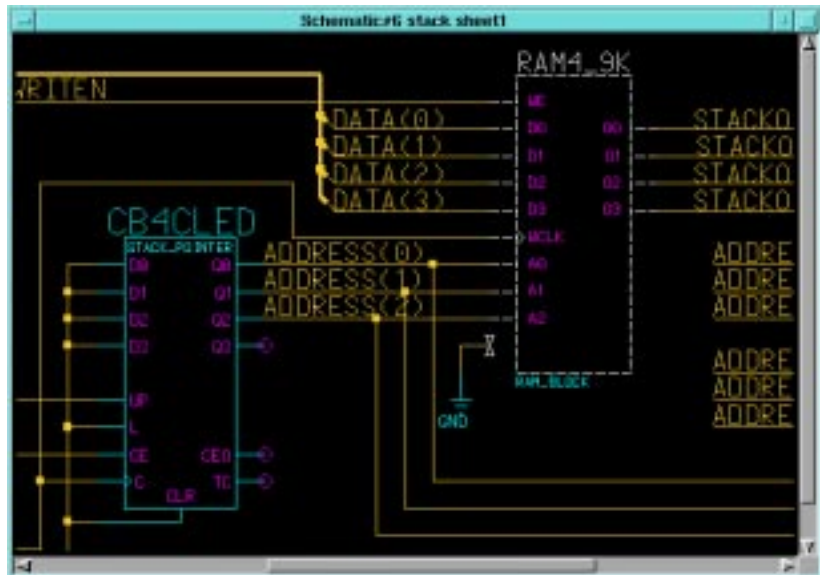


Figure 10-46 Device Independent RAM4_9K

The unused A3 pin that exists on RAM16X4S does not exist on RAM4_9K. Although the detached GND symbol and net are trimmed during the implementation process, you can clean-up the schematic by deleting them.

4. To clean-up the schematic, unselect everything by pressing **F2**, then select **Right Mouse Button** → **Delete** → **Selected** (or use the stroke 741236987, which looks like an uppercase “D”).
5. Check and save the updated stack schematic.

Removing the XC4000E Oscillator

If you are targeting the Calc design to the XC9000 family (or other non-XC4000 family), you must remove the CLOCKGEN circuitry, which includes the OSC4 component, and replace it with an external source.

Note: The XC3000 and XC5200 families also have internal, on-chip oscillators. See the CLKGEN3K and CLKGEN5K components to see how these are used. You may choose to replace the CLOCKGEN component with one of these alternative macros with the Replace

selection from the Instance pop-up menu, instead of following the instructions below.

1. On the Calc schematic, press **F2** to make sure nothing else is selected.
2. Select the CLOCKGEN component with the left mouse button.
3. Delete it by selecting **Right Mouse Button** → **Delete** → **Selected** (or use the stroke 741236987, which looks like an uppercase “D”).
4. Add components, nets, and labels as shown in the following figure. The IPAD symbol may be selected from the library menu under **BY TYPE** → **io**, while the BUFG symbol may be selected under **BY TYPE** → **buffer**.
5. Check and save the Calc schematic.

Since the CLK signal is now sourced by a pad, it must be generated externally.

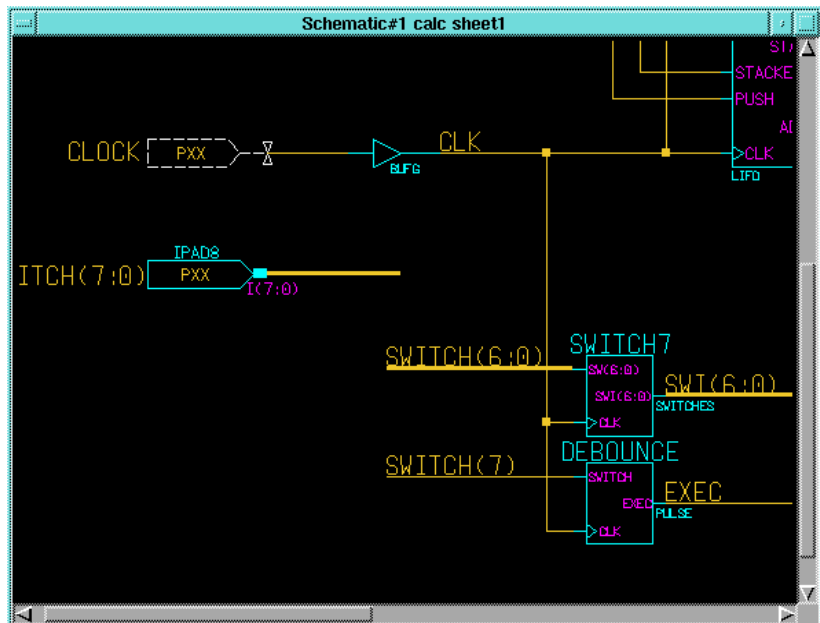


Figure 10-47 Device-Independent Clock Source

Using LogiBLOX

LogiBLOX is a tool that allows you to quickly synthesize modules for common functions such as adders, counters, and multiplexers. It allows you to create components of arbitrary bus width (e.g., a 17-bit adder) and automatically uses the best architectural resources for a particular target device family. In this optional section, you replace the ADSU4 component in the ALU schematic with a LogiBLOX adder. If you choose to leave the ALU schematic in its original form, read this section but do not make save any changes.

Creating and Instantiating a LogiBLOX Module

To replace the ADSU4 symbol with a LogiBLOX module:

1. Bring the ALU schematic into view.
2. Select the **ADSU4** component.
3. Select **Right Mouse Button** → **Delete** to delete the symbol from the schematic.
4. Select **Libraries** → **XILINX Libraries** from the menu bar.
5. From the Xilinx Unified Libraries menu, select **LogiBLOX**.

The Create/Modify/Instantiate LogiBLOX Symbol dialog box appears.

6. In the Symbol component field, type **addsub4**.

This is the user-given component name for the new LogiBLOX-generated module.

7. Under Instantiate symbol, choose **YES**.

This gives you the opportunity to place the component on the schematic immediately after LogiBLOX exits.

8. For PLD Technology, select the family to which you are targeting the design, e.g., XC4000E.

The dialog box should now appear as shown in the following figure.

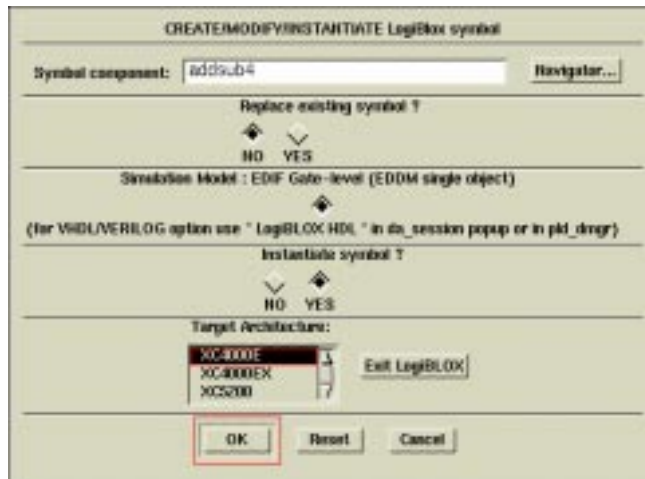


Figure 10-48 Instantiate LogiBLOX Dialog Box

9. Click **OK** to invoke the The LogiBLOX program.

In about 1 minute, the LogiBLOX Module Selector appears.

10. Set the options in this dialog box as shown in the following figure.

You are making a non-registered adder/subtractor module of four bits.

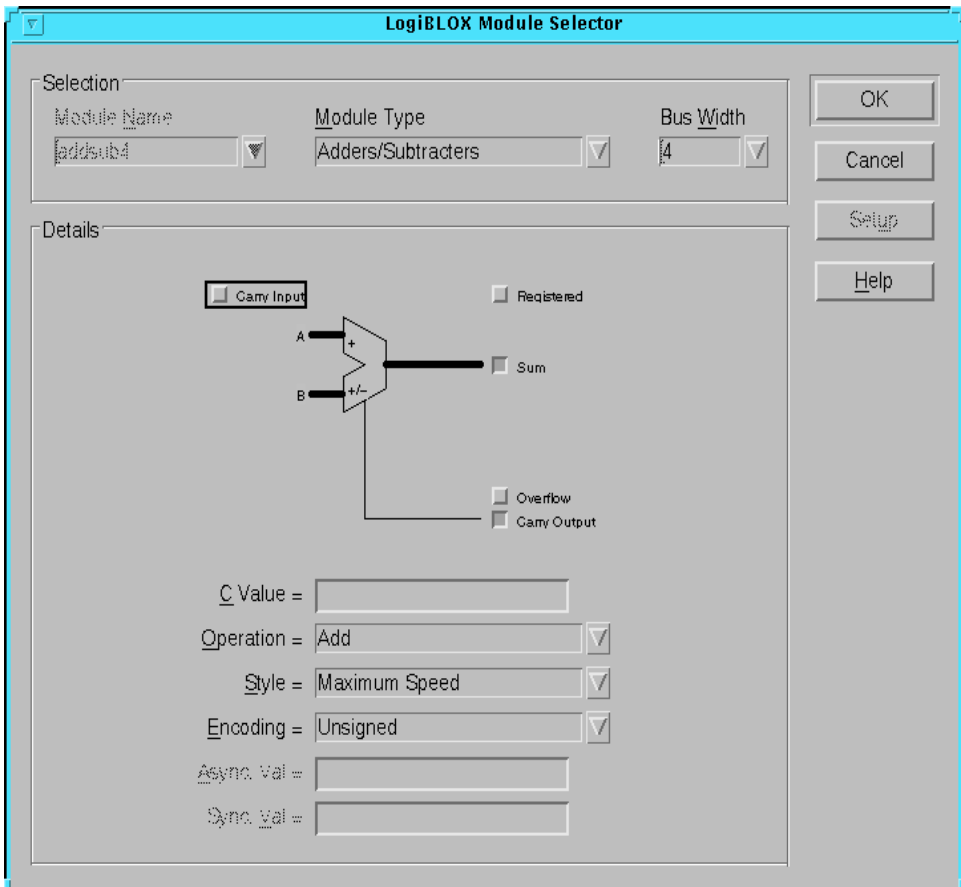


Figure 10-49 Using the LogiBLOX Module Selector

11. Click **OK**.

Design Architect takes a minute or so to generate a symbol for this new module.

12. When the module appears on your screen, place it in the space left by the ADSU4.

Do not worry about lining up pins with nets right now.

13. Arrange the surrounding nets as shown in the figure.
14. Check and save the ALU schematic.

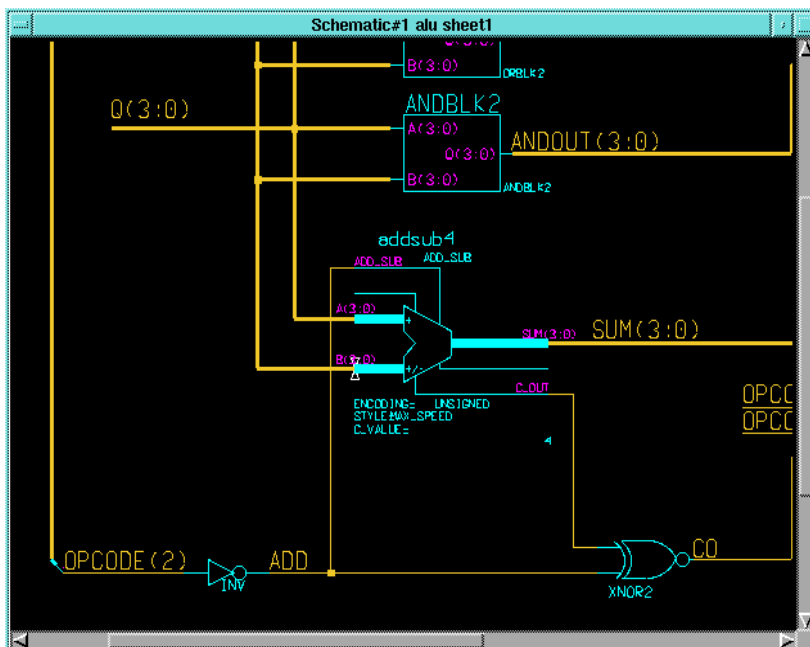


Figure 10-50 Adding the ADDSUB4 LogiBLOX Component

Other Special Components

In this section, you complete the Calc design by adding a STARTUP symbol to make the logic resettable and a CONFIG symbol to specify the Xilinx part number on the schematic.

The STARTUP Block (Optional: XC4000E/EX and XC5200 only)

The STARTUP block allows you to globally control different aspects of a design. This example uses STARTUP to connect an external signal to the global set/reset net built into the XC4000 family and XC5200 architectures. This global net connects to all flip-flops in the device and sets or resets them asynchronously. (Set or reset is determined at the flip-flop level.) An advantage to using this built-in resource is that no routing resources are wasted tying a system-wide reset signal to all flip-flops in the design. For more information on STARTUP, see the *Xilinx Libraries Guide*.

The STARTUP symbol is used here to implement a system-wide reset signal called NOTGBLRESET. This signal is active-low; therefore, when NOTGBLRESET is low, the Calc circuitry is reset.

1. In the Calc schematic, add the components, nets, and labels as shown in the following figure.

You may take IPAD and IBUF from the **BY TYPE** → **io** section of the Xilinx Library, INV from **BY TYPE** → **logic**, and STARTUP from **BY TYPE** → **general**.

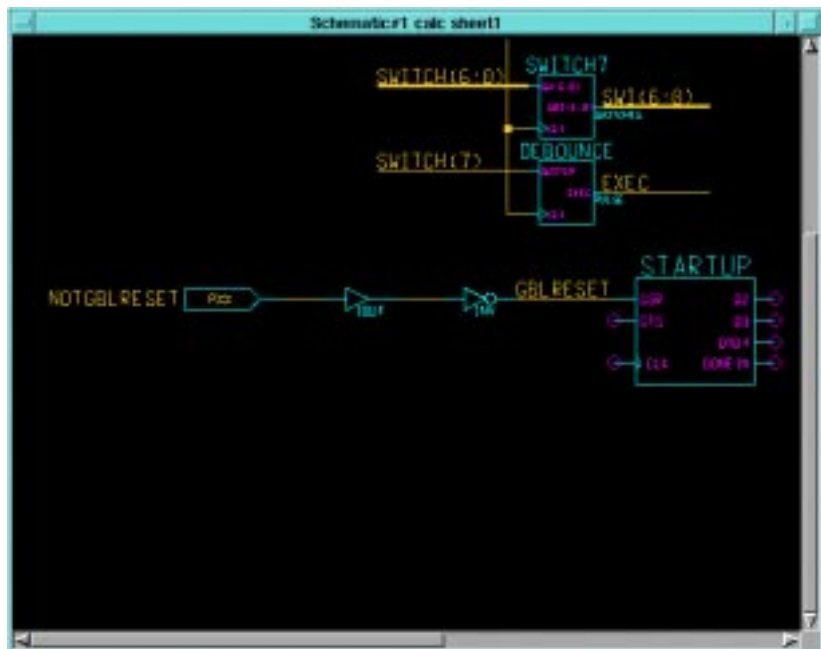


Figure 10-51 Adding the STARTUP Symbol

An inverter is added to the signal path since the GSR pin on STARTUP is active-high. Also, since GSR is *implicitly* connected to all reset logic throughout the device, GBLRESET is connected only to the GSR pin on the schematic.

Note: If you target an XC5200 device, connect your chip-wide reset signal to the GR pin on the STARTUP module.

2. Check and save the Calc design.

Adding the CONFIG Symbol (Optional)

The CONFIG symbol tells the place-and-route software how to process the design. This example uses CONFIG to specify the part number for this device.

To add the CONFIG symbol, follow these steps.

1. From the Calc schematic use the **BY TYPE** → **general** menu to call up the CONFIG symbol from the Xilinx Library.
2. Place this symbol in the lower-right hand corner of the Calc schematic.
3. With the CONFIG symbol still highlighted, select **Right Mouse Button** → **Properties** → **Add** → **Add Single Property**.
4. In the Add Property dialog box, enter the Property Name as “PART” and the Property Value as “XC4003E-4-PC84” and click OK.

This specifies an XC4003E device with –4 speed grade (approximately 4 nanoseconds delay through a CLB) in an 84-pin PLCC.

The PART value may take one of the following two formats:

[XC] *part_number-speed-package*

[XC] *part_number-package-speed*

Therefore, the following values for the PART property are all legal:

XC4003E-4-PC84 (recommended)

XC4003E-PC84-4

4003E-4-PC84

4003E-PC84-4

Note: If using a different device, type that device number into the Property Value field instead, *e.g.*, XC95108-10-PC84

5. Place the property text within the CONFIG symbol as shown in the following figure.

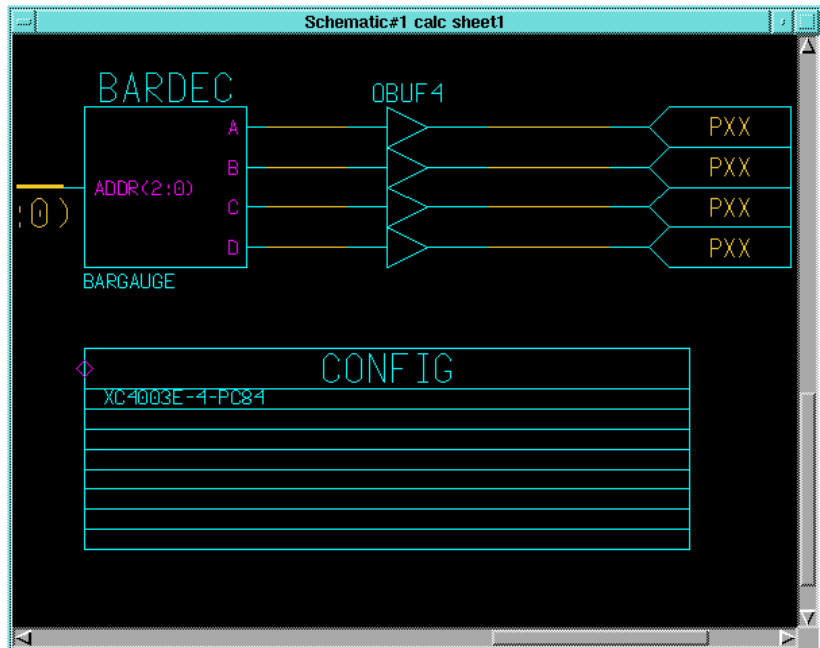


Figure 10-52 Adding the CONFIG Symbol

6. Check and save the Calc schematic.

Using a Constraints File

Using a constraints file, you can supply constraints information in a textual form rather than putting it on a schematic. Sometimes this method is more efficient than putting constraints on a schematic.

It is necessary to instruct the place and route software to read and apply the .ucf file when the Xilinx Design Manager reads the design. The procedure for doing this is detailed later in the “Using the Xilinx Design Manager” section.

The calc_4ke.ucf user constraints file which is supplied with this tutorial is shown below as an example of a constraints file. The constraints file syntax is the same for all device families. Since you only specified one pin location for one of the many inputs and outputs on the Calc schematic, you must use a constraints file to place the rest.

```
# CALC_4KE.UCF
# User constraints file for CALC, XC4003E-PC84
# If the F pin is not constrained on the schematic,
# remove the comment (#) from NET F LOC=P50;

NET SWITCH(7)      LOC=P19;
NET SWITCH(6)      LOC=P20;
NET SWITCH(5)      LOC=P23;
NET SWITCH(4)      LOC=P24;
NET SWITCH(3)      LOC=P25;
NET SWITCH(2)      LOC=P26;
NET SWITCH(1)      LOC=P27;
NET SWITCH(0)      LOC=P28;

NET A              LOC=P49;
NET B              LOC=P48;
NET C              LOC=P47;
NET D              LOC=P46;
NET E              LOC=P45;
# NET F            LOC=P50;
NET G              LOC=P51;
NET OFL            LOC=P41;

NET GAUGE(3)       LOC=P61;
NET GAUGE(2)       LOC=P62;
NET GAUGE(1)       LOC=P65;
NET GAUGE(0)       LOC=P66;

NET STACKLED(3)    LOC=P57;
NET STACKLED(2)    LOC=P58;
NET STACKLED(1)    LOC=P59;
NET STACKLED(0)    LOC=P60;

# Remove the NOTGBLRESET line if STARTUP
# is not used in the schematic

NET NOTGBLRESET    LOC=P56;
```

Performing Functional Simulation

You perform functional simulation before design implementation to verify that the schematic that you have designed is logically correct. All components in the Calc design, even the non-schematic Logi-BLOX module, have built-in simulation models so little pre-processing is necessary. However, every top-level design in Mentor

Graphics must have a simulation viewpoint before you can use it in QuickSim. The viewpoint describes how a design should be interpreted, including what components in the design are primitives, as well as how components within the design hierarchy should be modeled.

Using Pld_dve

You use the PLD Design Viewpoint Editor to generate a design viewpoint to tell QuickSim how to interpret certain Xilinx-specific design properties. Follow these steps to generate a viewpoint with pld_dve.

1. Select the calc design object from the appropriate directory in the Navigator window.
2. Invoke pld_dve on the design by selecting **Right Mouse Button** → **Open** → **pld_dve**.

A dialog box appears. Note that the component name, Calc, is entered automatically with a fully qualified path.

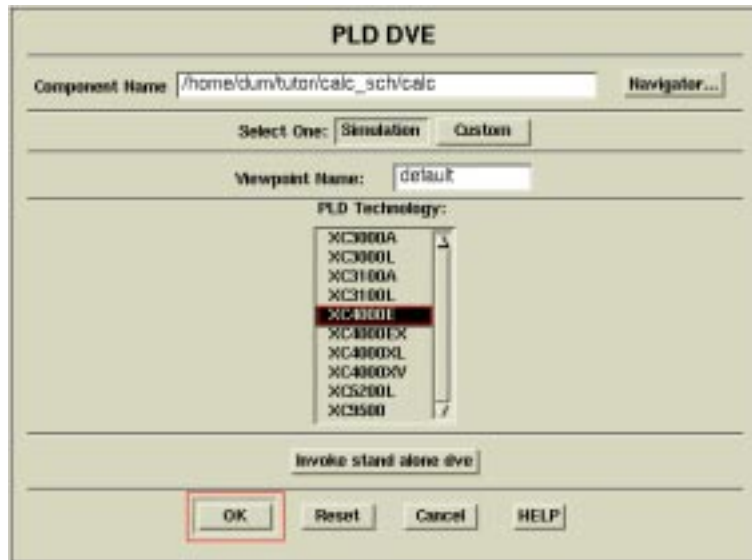


Figure 10-53 Invoking Pld_dve for Functional Simulation

3. Select the appropriate PLD Technology from the listing, *e.g.*, XC4000E, as shown in the figure above. (Leave other options set to their defaults, as shown in the figure.)

4. Click **OK** to execute the `pld_dve` script.
5. Once `pld_dve` completes, dismiss the shell window in which it executed.

Invoking Pld_quicksim

Invoke `pld_quicksim` for functional simulation on the Calc design using the following method:

1. Select the Calc design object in the Navigator window.
2. Invoke `pld_quicksim` on the design by selecting **Right Mouse Button** → **Open** → `pld_quicksim`.

A dialog box appears. Note that the component name, Calc, is entered automatically with a fully qualified path.

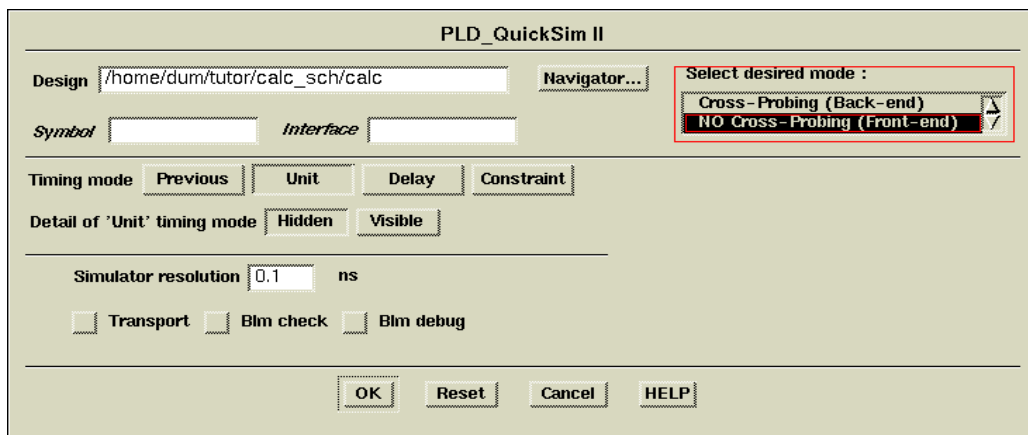


Figure 10-54 Invoking Pld_quicksim for Functional Simulation

3. Under Select desired mode, select **NO Cross-Probing (Front-end)**.

This runs `pld_quicksim` in functional simulation mode, which uses information from the original schematic (as opposed to a timing netlist generated by the Xilinx software) to model the design.

4. Click **OK** to start QuickSim II.

Viewing the Calc Schematic

When QuickSim starts, no windows are open. In this section, you open a window and view the top-level schematic for the Calc design. Displaying the schematic is convenient for viewing back-annotation during the simulation.

1. To open a window containing the Calc schematic, select **OPEN SHEET** from the palette.

This automatically opens the top-level sheet for Calc.

2. Move the window to the upper left corner of the QuickSim window.

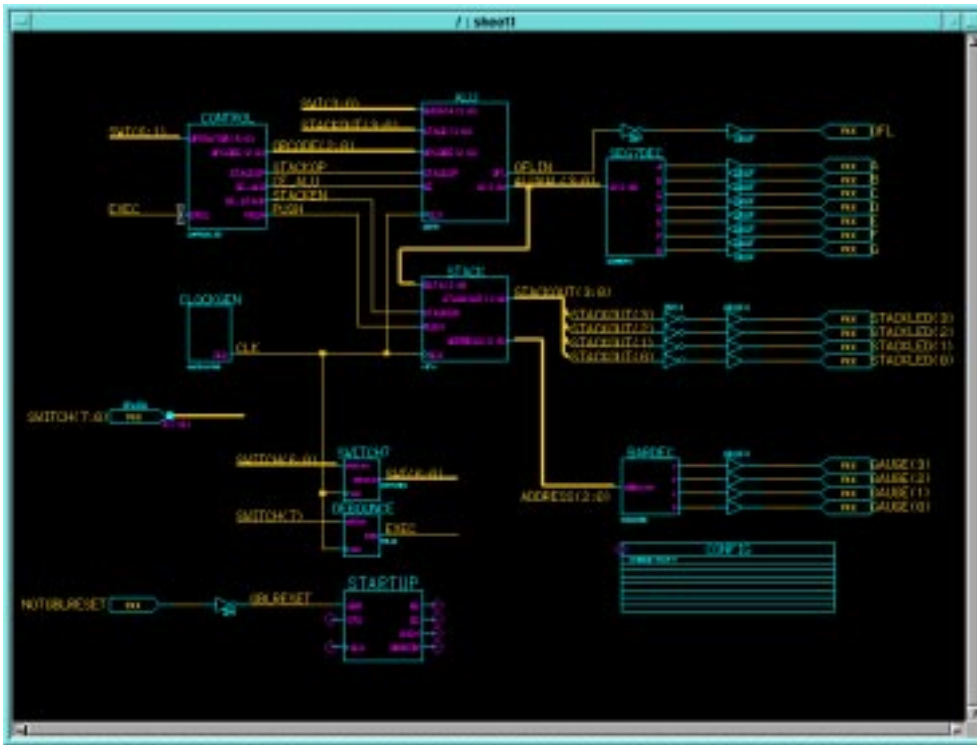


Figure 10-55 Top-Level Calc Schematic

Selecting Nets for Simulation

There are several ways to select the signals that you would like to monitor. One way is to select the **Right Mouse Button** → **Add** → **Traces** → **Specified** command, then type in the nets you want to view in simulation. Another way is to select the nets on the schematic. To select the signals, you may need to zoom and pan using the scroll bars or using strokes.

To select the nets on the schematic follow these steps:

1. Using the **F8** key, zoom in on the area pictured in the following figure.
2. Position the cursor on the net labeled CLK, and press the left mouse button.

The net appears highlighted, as in the figure below. Whenever any portion of a net is selected in QuickSim, the entire net appears highlighted. You can select additional nets using the same procedure. If you make a mistake, click the left mouse button a second time on the net or object to unselect it.

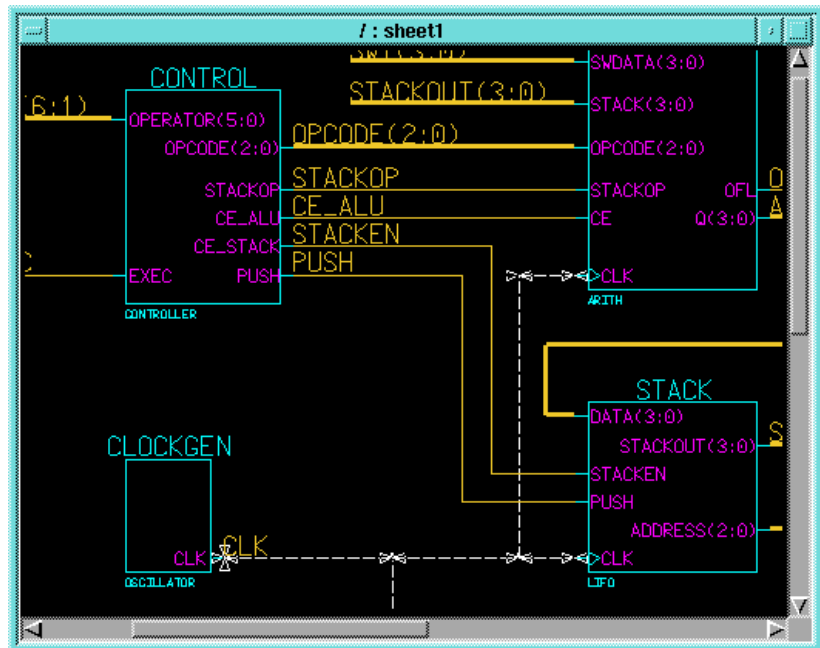


Figure 10-56 Selecting the CLK Net for Display in Trace Window

3. Use the left mouse button to select the following nets: STACKEN, PUSH.
4. Using the **Shift-F8** key, view the entire schematic.
5. Select the net labeled EXEC (output of the DEBOUNCE component) with the left mouse button.

One of the advantages of labeling all nets is now clear. When you select an unlabeled net for simulation display, note that a default name is used for the net, such as NS14. This name is not very useful for debugging, especially since making changes to the schematic may cause renumbering of net names.

6. You can also add buses to your list of signals to be monitored. Use the left mouse button to select the buses labeled ALUVAL(3:0) and STACKOUT(3:0).
7. Press the blue **TRACE** button in the palette to add all selected signals to the Trace window.

Opening Trace and List Windows

To view the waveforms of the selected signals, you must open a QuickSim Trace window.

To open a Trace window, perform these steps.

1. Select the blue button labeled **TRACE** in the palette with the left mouse button.

A Trace window appears displaying the waveforms selected on the schematic.

2. If necessary resize the Trace window to see all the signals at once. Otherwise, move the cursor into the Trace window and use the PageUp and PageDown keys to scroll through the signals in the window.

Note that all the signals in the Trace window are highlighted. Every window opened in QuickSim is dynamically linked to the others. The selection of a net on the schematic sheet, for example, is also reflected in the Trace window, and in any other window that is open. This is useful, for example, if a setup violation occurs. The instance name in the error message text is highlighted, and the related component on the schematic page also appears highlighted.

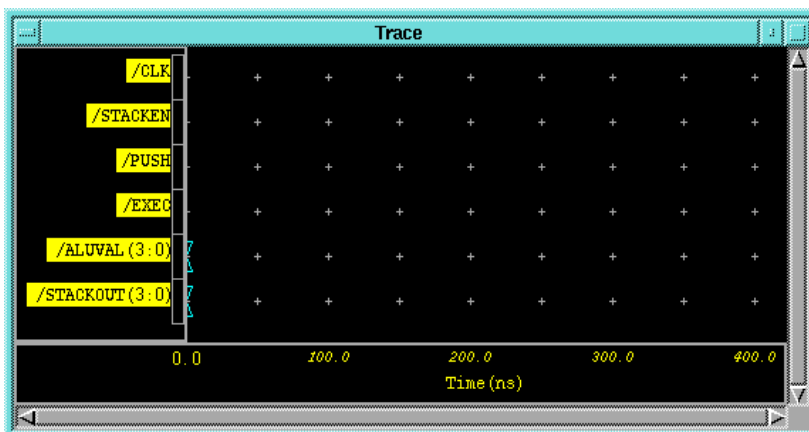


Figure 10-57 Trace Window

It is sometimes useful to obtain tabular output using a List window. A List window displays signal values and highlights the points at which a given signal value changes.

To open a List window, perform these steps.

1. Since the desired signals are already selected, select the blue LIST button in the palette with the left mouse button.

The list window appears, with the signal names at the bottom. The caret ('^') is an arrow pointing up to indicate the correct column for each signal.

2. Move the List window to the upper right-hand corner of the screen next to the Schematic window.

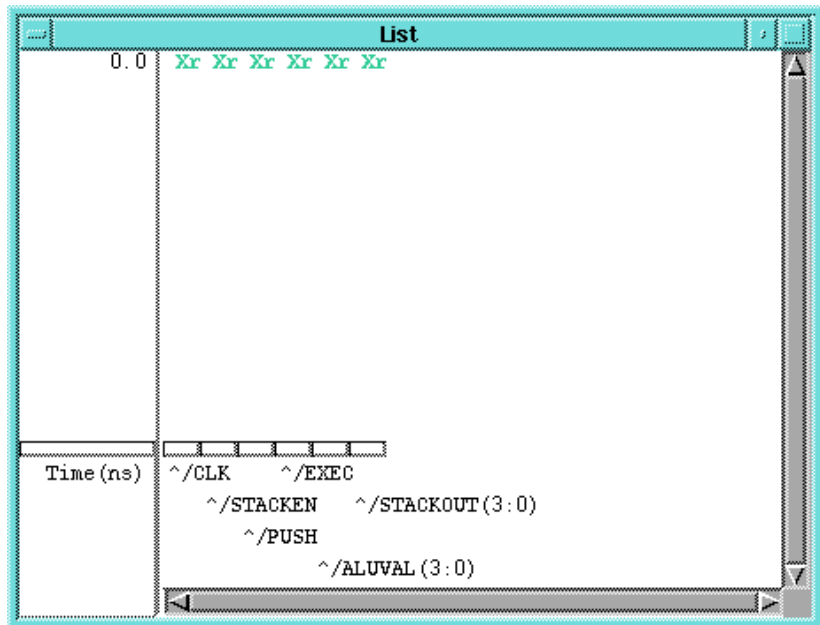


Figure 10-58 List Window

Adding Traces Manually

In the Calc design, inputs are entered via a set of eight switches, SWITCH(7:0). The lower seven switches (SWITCH(6:0)) define the opcode. The left-most switch (SWITCH(7)) is the execute switch.

When SWITCH(7) is toggled, the selected opcode on SWITCH(6:0) is executed. It is useful to view SWITCH(6:0) and SWITCH(7) separately in the Trace window.

Add these two traces to the Trace window as follows:

1. To add traces manually, the Stimulus Palette must be active. Click on the red STIMULUS button in the palette. The icons in the palette change.
2. Press the **F2** key to unselect everything.
3. Select the Trace window with the left mouse button.
4. Choose **Right Mouse Button** → **Add** → **Traces** → **Specified**.
5. In the dialog box that appears, select the Named Signals button with the left mouse button.
6. Fill in the dialog box as shown in the figure below.
7. Select **OK** or press Return.

The bus SWITCH(6:0) and the signal SWITCH(7) are added to the Trace window.

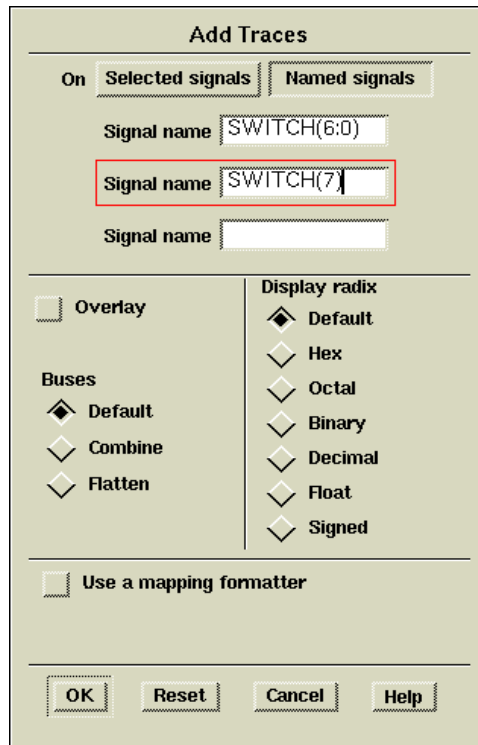


Figure 10-59 Adding Traces Manually

Assigning Values to the Clock

Define a clock for the circuit as follows:

1. Make sure that the Trace window is active (border appears blue). If not, select the window using the left mouse button.
2. Press the **F2** key to unselect everything, then select the **CLK** net in the Trace window using the left mouse button.
3. Select **ADD CLOCK** in the palette.
4. Fill in the dialog box that appears as shown in the following figure.

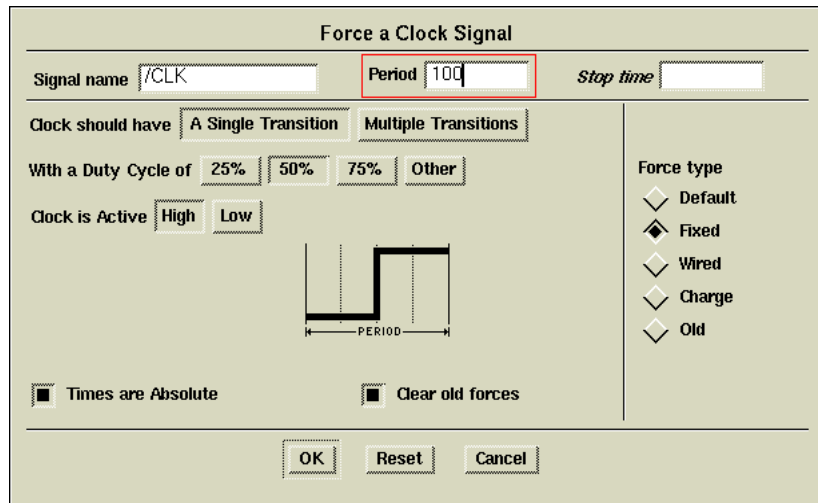


Figure 10-60 Adding a Clock Waveform

The dialog box selections give the clock a 100 ns period and a 50 percent duty cycle. At zero ns, the clock begins with a value of zero. The Absolute option indicates that the times are absolute, and not relative to the state of the simulator. For example, if you had already been simulating, the state of the simulator may not be at zero nanoseconds. If Absolute is selected, the times entered in the dialog box are referenced from time zero. If Absolute is not selected, the times entered in the dialog box are added to the present time in the simulation.

Selecting a Fixed Force type indicates that the signal is driven as if it were connected directly to VCC or GND. If Wired were selected, the signal would be driven as if it were connected to a pull-up or pull-down resistor. A Charge Force type represents a default charge on a floating signal. Wired values are overridden by Fixed values, and Charge values are overridden by both. In general, for Xilinx designs always use a force type of Fixed unless it is a bidirectional input, in which case a Wired force type should be used.

5. Press return or select **OK** to add the force to CLK.

Asserting Global Set/Reset (without STARTUP)

Note: This section applies to designs in which the STARTUP module has not been instantiated. If you have an XC4000 family or XC5200 design that has the STARTUP module in it, go to the “Asserting Global Set/Reset (with STARTUP)” section.

In every simulation, the first node you must assert is the global-reset signal. This signal does not exist on your schematic, but does exist in the device. This dedicated net is connected to every asynchronous reset pin on every flip-flop (including IOB flip-flops) in an FPGA or CPLD. The net is named differently and has a particular polarity depending on the device family used as shown in the following table.

Table 10-3 Net Names for the Global Set/Reset Signal

Device Family	Net Name	Polarity
XC3000	//globalresetb	Active Low
XC4000	//globalsetreset	Active High
XC5200	//globalreset	Active High
XC7000/XC9000 Pre NGDBuild (functional sim)	//prld	Active High
XC7000/XC9000 Post PAR (timing sim)	/prld	Active High

In each case, the global-reset net name is preceded by two forward slashes, indicating that the net is a global signal in QuickSim. The global-reset signal is part of the simulation models; you must toggle it at the beginning of every simulation. If you do not pulse globalresetb low, all flip-flop outputs are unknown at all times during your simulation.

In the following example, you set //prld, the XC9000 global-reset signal, high at time 0 and low at 100 ns:

1. With the Trace window selected, press the blue **Unselect All** button in the palette with the left mouse button.
2. Select the **Add Force** icon in the palette with the left mouse button.

The Force Multiple Values dialog box appears.

- Since a signal is not selected, the Signal name field is empty. Fill in the dialog box as shown in the following figure. The signal is to be forced low (asserted) at time zero and high at time 100ns.

The image shows a dialog box titled "Force Multiple Values". It has a "Signal name" field containing "//prld". Below it are three rows of "Value" and "Time" input fields. The first row has Value "1" and Time "0". The second row has Value "0" and Time "100", with a red rectangular box around the "100" value. The third row has empty fields. To the right is a "Force type" section with radio buttons for "Default" (selected), "Fixed", "Wired", "Charge", "Old", and a checkbox for "Absolute". At the bottom are four buttons: "OK", "Reset", "Cancel", and "Help".

Figure 10-61 Forcing the Global-Reset Signal (XC9000)

For other families, assert the appropriate signal with the appropriate polarity. For example, in the XC3000 family, the global-reset signal is //globalresetb, which must be forced low at time 0, and so on.

Asserting Global Set/Reset (with STARTUP)

Note: This section applies to designs in which the STARTUP module has been instantiated. If you have a design without a STARTUP module instantiated, follow the instructions in the “Asserting Global Set/Reset (without STARTUP)” section.

The global-reset signal must be forced at the beginning of all XC4000 family and XC5200 simulations. It is an active-High signal that sets or resets all flip-flops in the chip. Whether a flip-flop is set or reset depends on whether it is an FDP or an FDC flip-flop, or on the value of the flip-flop’s INIT attribute. The default configuration for all flip-flops is to function as a reset flip-flop.

Unlike other families, the global-reset signal in the XC4000 family and XC5200 family is not hard-wired to a package pin, and need not appear on one at all. If you want access to the global-reset net from an

external pin, place the STARTUP component in your schematic and attach an IPAD and IBUF to the GSR pin for XC4000 family designs, or to the GR pin for XC5200 family designs. This pad becomes an active-High Global Set Reset signal in XC4000 family devices and an active-High Global Reset signal in XC5200 family devices. You can also use an internally generated signal to drive the GSR or GR pin of the STARTUP component. There is also an active-High Global Three State signal (GTS) that you can access in the same way. See the *Xilinx Libraries Guide* for more information on the STARTUP symbol.

Since an external signal is connected to the global-reset net via the STARTUP symbol, you must pulse this external signal to activate global reset as opposed to the internal global-reset signal (explained in the “Asserting Global Set/Reset (without STARTUP)” section.

1. With the Trace window selected, press the blue **Unselect All** button in the palette with the left mouse button.
2. Select the **Add Force** icon in the palette with the left mouse button.

A dialog box appears.

3. Normally, the net name `//globalsetreset (XC4000)` or `//global-reset (XC5200)` would be added as the signal name. (The two leading forward slashes would indicate that this is a global signal.) However, since you have included the STARTUP symbol in this design, you must instead pulse whatever signal is driving the GSR pin on the STARTUP module. In this case, pulse the NOTGBLRESET signal.
4. Fill in the dialog box as shown in the following figure. The NOTGBLRESET signal is to be forced low (asserted) at time zero and high at 100ns. Note that, because of the inverter in the path from NOTGBLRESET to GSR or GR, this series of forces is equivalent to pulsing globalsetreset or globalreset high.

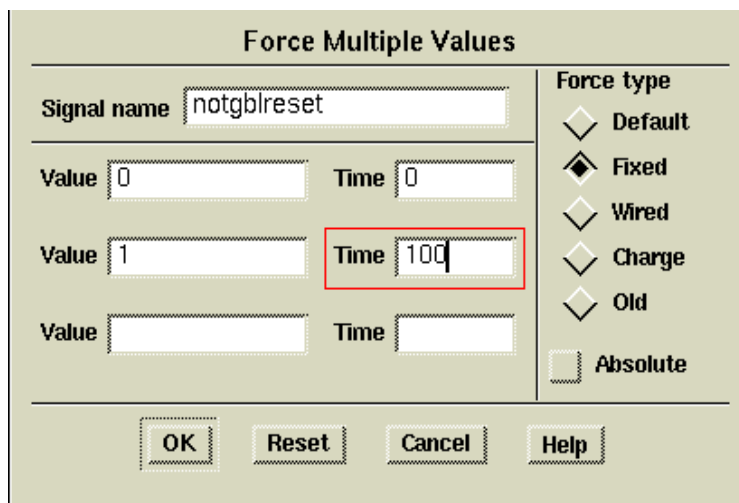


Figure 10-62 Forcing Globalsetreset via Notgblreset (XC4000E)

Design Description

The Calc design is a simple four-bit processor with a stack. The CONTROL module interprets the switch input and drives the control lines of the ALU and STACK components. The ALU performs functions between an internal register and either the top of the stack or data read in from the external switches. Outputs include ALUVAL(3:0), the current contents of the internal register, and STACKOUT(3:0), the top value in the stack.

For a more detailed description of the Calc design, see the “Design Description” section.

Simulating the Circuit

You are now ready to force the inputs to known values and simulate.

1. Press the blue **Unselect All** button in the palette.
2. Select the **Add Force** button in the palette with the left mouse button.
3. Fill in the dialog box as shown in the figure below.

All numbers entered are interpreted as hex. This sets opcode (SWITCH(6:0)) to perform the following actions:

00: ADD 0h to register value (should produce a zero).
 61: LOAD register with 1h.
 0D: ADD Dh to register value (1 + D should produce F).
 7B: PUSH register value to stack (top of stack=F).
 50: CLEAR register value.

Force Multiple Values	
Signal name	switch(6:0)
Value	00
Time	0
Value	61
Time	200
Value	0D
Time	700
Value	7B
Time	1200
Value	3F
Time	1800
Value	7B
Time	2300
Value	50
Time	2800
Value	
Time	

Force type

- Default
- Fixed
- Wired
- Charge
- Old

Absolute

OK Reset Cancel Help

Figure 10-63 Forcing Values to SWITCH(6:0)

For these commands to be executed, you must provide stimulus to SWITCH(7), the execute switch. Perform the following actions to force SWITCH(7) correctly:

1. Press the blue **Unselect All** button in the palette.
2. Select the **SWITCH(7)** signal from the Trace window. It may be necessary to use the PageDown key to scroll through the list of signals in the Trace window.
3. Select the red **WF EDITOR** button from the top of the Palette. The icons in the palette change.
4. Select the icon labeled **EDIT WAVEFORM**.

A new trace appears labeled `forces@@/SWITCH(7)`. While the `SWITCH(7)` trace represents the value of `SWITCH(7)` up to the present time in simulation, the trace `forces@@/SWITCH(7)` represents all values that will ever be forced on the signal. During simulation, this waveform can be edited to modify future values of `SWITCH(7)`.

A blue line appears extending from `SWITCH(7)` to indicate that it has not been given a value. First, force `SWITCH(7)` to a known value at time zero as follows:

1. Select the **ADD** icon in the palette.
2. Move the cursor into the Trace window.

A red vertical line appears under the cursor. The numbers in the grey box reflect the value and time that are pointed to as the cursor is moved.

3. Move the cursor close to the beginning of `forces@@/SWITCH(7)`, as shown in the figure below, and then press the left mouse button.

This indicates that you want to change the value from the nearest left edge (in this case, time zero is considered an edge) to the next right edge. Since the signal makes no transitions, you can assign the same value to the entire length of the signal.

4. Type a '1' in the value field of the small dialog box and click **OK**.

This indicates that you want to change the signal value between the two nearest edges to a one. The entire length of the signal changes color from dark blue to light blue, and the line moves up, indicating it will be driven to a one.

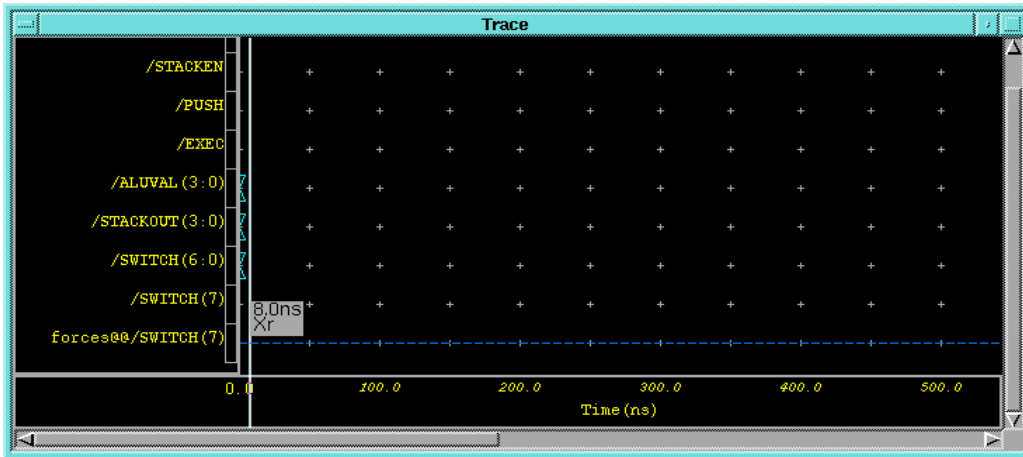


Figure 10-64 Forcing SWITCH(7) to Initial Value

5. Press the **Escape** key to end the Add Event operation.

Now that SWITCH(7) has been given an initial value, you must define when transitions occur on the signal as follows:

1. Select the **TOGGLE** icon from the palette.
2. Move the cursor to the Trace window.

A red vertical line appears with numbers indicating the value and time of the signal at the position beneath the cursor.
3. Move the cursor to the forces@@/SWITCH(7) signal at time 700ns and press the mouse button, as shown in the figure below.

A high to low transition is added to the force waveform at time 700 ns.

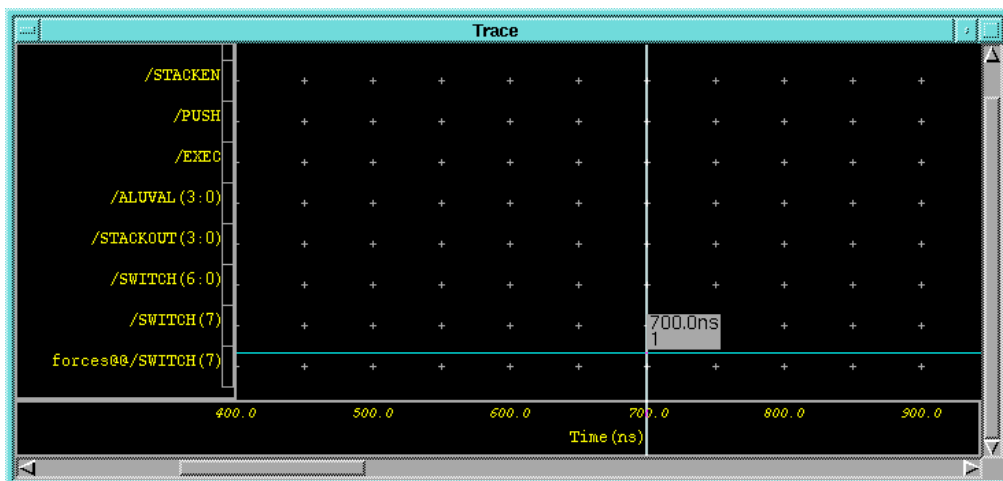


Figure 10-65 Adding the First Toggle to SWITCH(7)

Note: It is sometimes difficult to position the cursor at exactly the right value if you are zoomed in too close. If you zoom out, the numbers get rounded to the nearest 1.0 ns, making it easy to place the edges correctly. Use the stroke 753 to zoom out. If you still cannot place the edges exactly, err to the left of the desired location. If you make a mistake, select the CUT icon in the palette and click the left mouse button on the incorrectly placed edge. The edge disappears. Then, select TOGGLE to continue adding edges.

4. Without moving the cursor, use the right arrow key to scroll the window forward in time. Each press of the right arrow key advances the window (and, consequently, the position under the cursor) by 50 ns. Add toggles at times 900, 1200, 1400, 1800, 2000, 2300, 2500, 2800, and 3000 ns. The waveform then appears as in the figure below. Press Shift-F8 to view the entire waveform.
5. Press **Escape** to end the TOGGLE command

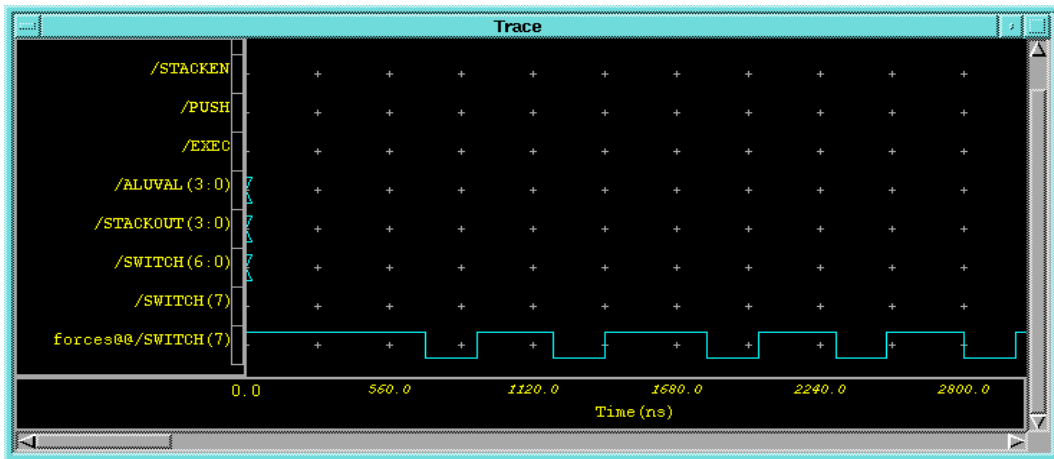


Figure 10-66 SWITCH(7) Force Waveform

Now that your inputs and clock are defined, you are ready to run the simulation.

6. Type **run 3400** at any location in the QuickSim window, then press **return**.

A window automatically appears containing the text. The results should look similar to those in the following figure.

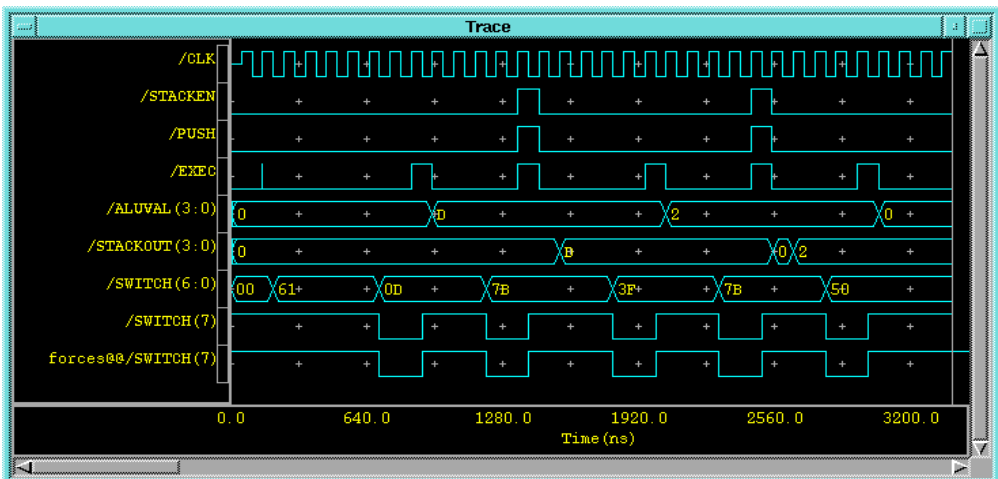


Figure 10-67 Output from Simulation (XC4000E design)

Saving the Results

If you were to exit QuickSim now, you would lose your waveform data. You can save the waveform information in a waveform database. To view the waveforms at a later time, you can use the **File** → **Load** → **Waveform DB** command found in the menu bar.

1. Select the red **STIMULUS** button from the palette.
2. Select the **SAVE WDB** icon from the palette.

The Save Waveform DB dialog box appears.

3. Fill in the Save Waveform DB dialog box as shown in the following figure.

This saves your results to the WaveForm Database, `simrun1`. This database is created in the directory specified by the `$MGC_WD` environment variable.

4. Press **return** or click **OK**.

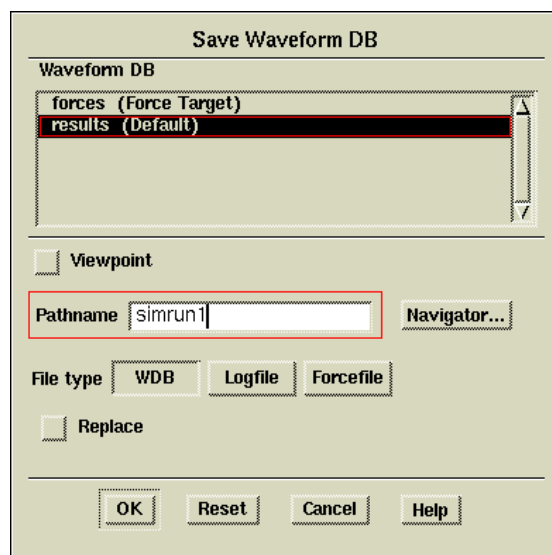


Figure 10-68 Saving Results

It may be useful to save the stimulus so that it can be run again. To do this perform the following steps:

1. Press the red **STIMULUS** button in the palette.

2. Select the **SAVE WDB** icon from the palette.
The Save Waveform DB dialog box appears.
3. Fill in the Save Waveform DB dialog box as shown in the figure below.
This saves the stimulus to the file, forces1. As with simrun1, this file is created in the directory specified by \$MGC_WD.
4. Press **return** or click **OK** to save the forces.

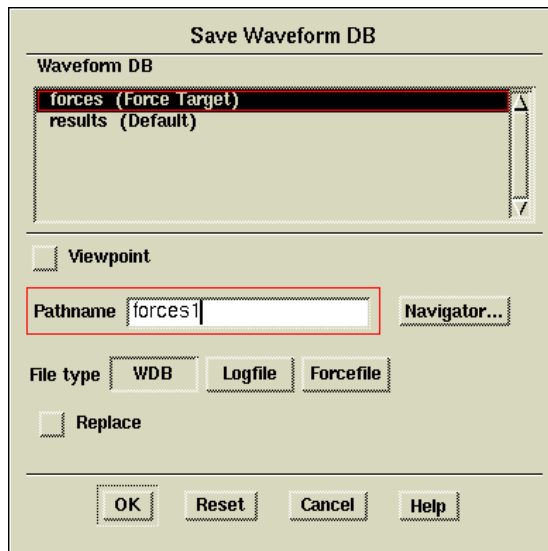


Figure 10-69 Saving Forces

After saving the results, reset the simulator to time zero as follows:

1. Press the blue **RESET** button in the palette.
2. In the dialog box that appears, select the **state** button so that it highlights, and deselect any highlighted buttons.
This forces the simulator to reset without saving.
3. Press **return** or choose **OK**.

The Trace window results disappear, while the forces waveform remains.

Using the Transcript

In addition to saving the results and forces, you can also save the actual transcript for the QuickSim session. Every mouse click and key press is recorded. This is sometimes useful for making macros to perform complicated, repetitive tasks. The saved transcript can then be replayed using the **MGC** → **Transcript** → **Replay** command found in the menu bar. Save the transcript as follows:

1. Select the **MGC** → **Transcript** → **Show Transcript** command from the menu bar.

A text window appears. In this text window are AMPLE commands. AMPLE (Advanced Multi-Purpose Language) is a C-like programming language used by all of the Mentor Graphics tools.

2. Select **Right Mouse Button** → **Export**.
3. In the dialog box that appears, type the file name **transcript.out** in the text field of the dialog box.

This saves the transcript to that file.

It is usually necessary to edit the transcript to make it useful. For example, if this transcript were re-run on the Calc design, it would setup the simulation, run, save the results, reset the simulator, and open a transcript window. Perhaps all you want it to do is setup and run the simulation. You would then have to delete the other commands from the transcript file before re-running it. For example, the `$show_transcript();` command at the end of the transcript file could be deleted to keep the transcript window from appearing. You would probably also want to delete the `$set_active_window("Transcript");` command as well if you did this. For more information on AMPLE, refer to the appropriate Mentor Graphics documentation.

4. Select **File** → **Quit** to exit QuickSim.

Using Pld_men2edif

Once your design is verified to be functionally correct, you use `pld_men2edif`, a tool in the Mentor Graphics Design Manager, to translate your Mentor design into a Xilinx-ready EDIF netlist. Running `pld_men2edif` is always the first step in implementing a

design. Whenever you make changes to your schematic, you must run `pld_men2edif` again so that the Xilinx software can process those changes.

When you run `pld_men2edif` from the Mentor Design Manager, the following dialog box appears:



Figure 10-70 Pld_men2edif Dialog Box

Here is an explanation of some of the fields and buttons in the `pld_men2edif` dialog box.

- **Component Name**—Enter the name of the component that you want to process here.
- **From Viewpoint**—If you are an advanced Mentor Graphics designer who uses viewpoints to organize design models and properties, enter the viewpoint name that you wish to use for this EDIF translation. If you do not use or are not familiar with viewpoints, leave this field blank and `pld_men2edif` will use a default value.
- **Forward Referencing of EDIF component libraries**—This option applies only in rare situations where design hierarchy has been structured in such a way that circular or recursive references exist. Normally, this option is set to Off.

- **Output EDIF Bus Dimension Separator Style**—This determines how bus-index delimiters are written into the output EDIF file. This is important if you are merging components from other design-entry tools into a single design. Choosing a bus-index delimiter lets you insure that the bus-index delimiters that `pld_men2edif` writes out are consistent with those of any other design-entry tools with which you are interfacing.

Since this design has been fully captured in Mentor Graphics, you need not worry about what type of bus delimiters are used. You may leave this setting on the default (PARENTH).

- **PLD Technology**—Select the architectural family from this list.
- **HELP**—If the HELP button is clicked, a short help listing is produced by the `pld_men2edif` script.

Perform the following steps to create an EDIF netlist for Calc:

1. Double-click on the `pld_men2edif` icon in Design Manager.
2. For the Component Name, type `$XILINX_TUTORIAL/calc_sch/calc` as shown above.
3. Select the architecture in the PLD Technology field, *e.g.*, XC4000E.
4. Select **OK**.

This opens a new shell window where `pld_men2edif` runs and reports its progress. When `pld_men2edif` has completed, the following should appear at the bottom of the shell window:

```
pld_men2edif ended with return code 0
Done.
```

5. Dismiss the `pld_men2edif` shell window by typing `Ctrl-C` in it or by selecting **Close** from the window's control menu (accessed through the button on the left side of the title bar).

Note: The output of `pld_men2edif` may be sent to the window from which the `pld_dmgr` was originally invoked. This behavior is dictated by the `$MGC_TERMINAL_WINDOW` environment variable; see the Mentor Graphics documentation for more details.

Examining Pld_men2edif Output Files

In addition to the EDIF netlist, `pld_men2edif` also creates a `pld_men2edif.log` file. This file contains a transcript of the processing

done by `pld_men2edif`. If the program fails to generate an EDIF netlist, any errors encountered are logged in this file.

Examine the `pld_men2edif.log` file for the Calc design as follows:

1. Select the Navigator window.
2. Choose **Right Mouse Button** → **Update Window**.

This updates the Navigator window to display the new files created by `pld_men2edif`, including an EDIF file for Calc, and a log file for `pld_men2edif`.

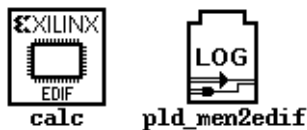


Figure 10-71 Files Created by Pld_men2edif

3. Select the LOG icon labeled `pld_men2edif` and choose **Right Mouse Button** → **Open** → **Editor**.

A window appears displaying the log file.

4. When you are done viewing the log, close the window.

Note: You can change the display font in this window by selecting **View** → **Fonts**.

Using the Xilinx Design Manager

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

This section gives a brief overview of the design implementation flow. For a more in-depth discussion of the flow, including advanced implementation options, see the *Development System Reference Guide*.

1. Within the Mentor Design Manager, select the **Calc EDIF** icon in the Navigator, then select **Right Mouse Button** → **Open** → **pld_dsgnmgr**.

The Xilinx Design Manager appears as shown. The tool automatically creates a Xilinx project called calc. Xilinx project information is kept in a file called xproject/calc.prj by default.

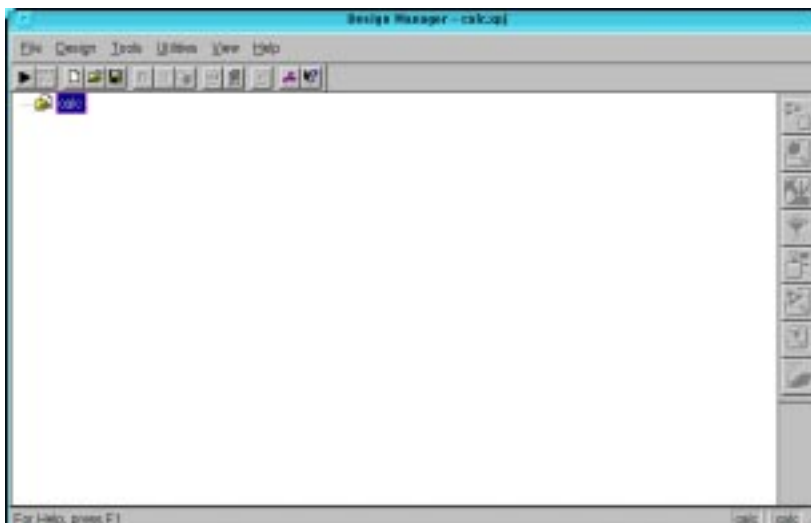


Figure 10-72 Xilinx Design Manager

Each project has associated with it objects known as “versions” and “revisions.” Versions represent logic changes in a design (for example, adding a new block of logic, replacing an AND gate with an OR gate, or adding a flip-flop); revisions represent different executions of the design flow on a single design version, usually with new implementation options (for example, higher place and route effort, a change in part type, or experimentation with new bitstream options). In the next stage, you make a new version and revision on which you run the implementation design flow.

2. Before you implement you must create a version in order to modify the implementation options. **Design** → **New Version**.

In the current release of software, the Xilinx Design Manager does not read the part type from the design.

Note: The PART property in the CONFIG symbol does not get read properly when processed from the system prompt, if the “-p” command-line option is omitted from NGDBUILD and MAP. (See the “Command Summaries” section of this chapter.)

3. Click the **select** button to display a pull-down listing of available devices.
4. Choose a Family of **xC4000E**, a Device of **xC4003E**, a Package of **PC84**, and a Speed Grade of **-4**.
5. Click **OK**.

The part number is inserted into the Part field in the New Version window

6. Choose the existing ucf file to be used in implementation in the 'Copy Persistent Data' field. Click on the pull down arrow and choose 'Custom'. The Custom window should appear. Click on the Browse button and select the file 'calc_4ke.ucf' file in the calc_sch directory (one directory up from the xproj directory).
7. Click on **OPEN** to choose the file and return to the 'Custom' window
8. Click **OK** to use the selected file and return to the 'New Version' window, which should now show Custom for Constraint file. See the following Figure 10-73.



Figure 10-73 New Version Dialogue Box

9. Click on OK. The Design Manager window should now look like the following Figure 10-74.

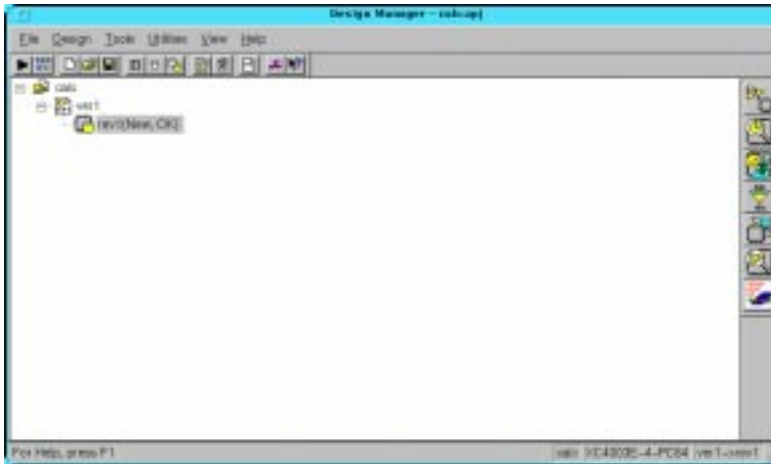


Figure 10-74 Design Manager With a New Version

10. Now that a version/revision has been created, the implementation options can be modified. With the rev1 highlighted, select **Design** → **Options**.
11. Choose the Simulation type to be Quicksim. This will automatically set the 'Correlate simulation data to input file' options and will also write out a Mentor EDIF. See the following Figure 10-75.



Figure 10-75 Implementation Options Dialogue Box

12. Click OK to return to the Design Manager main window.

Note: If you would like the tools to produce a Logic Level Timing Report then you will have to go to **Design** → **Options**, click on the implementation 'Edit Options' tab. Click on the Timing Reports tab and make the appropriate selection to Produce Logic Level Timing Report.

Note: By choosing 'QuickSim' for the simulation the Timing simulation data is created by default. Also the Post Layout Timing Report will be produced and the Configuration Data is created by default since the option is set to default as opposed to 'OFF'.

- **Produce Timing Simulation Data**—This generates a back-annotated EDIF netlist that can be imported into the Mentor Graphics tools.
- **Produce Configuration Data**—This generates a programming bitstream suitable for downloading into the Xilinx device.
- **Produce Post Layout Timing Report**—This generates a timing report file based on how the design is actually routed.

You can also select the following option (FPGAs only):

- **Produce Logic Level Timing Report**—This generates a preliminary (post-map, pre-place and route) timing report based on the number of logic levels in each signal path. Since it is generated after the mapping step but before the place-

and-route layout step, it does not contain information on device routing. Looking at this report before place and route can be useful for seeing how much “routing slack” you have in a design.

13. Within the Xilinx Design Manager, click on the right arrow



or select **Design** → **Implement** to run implementation.

The Flow Engine comes up as shown in the figure.

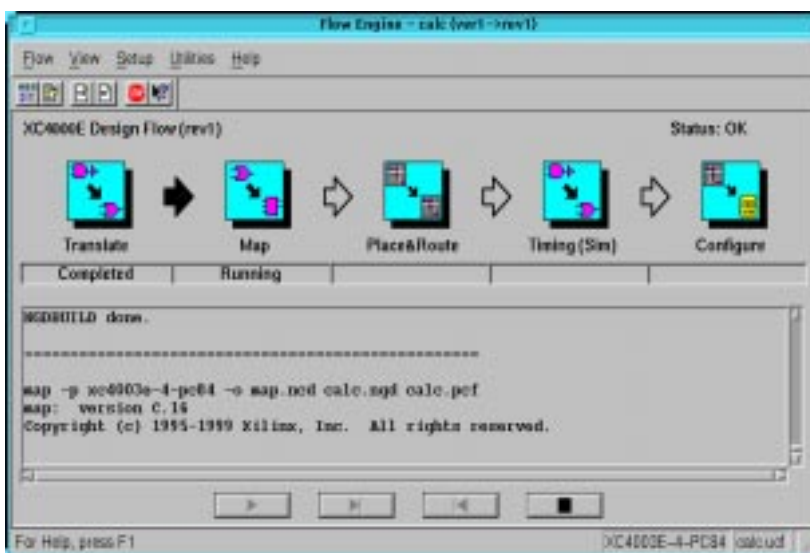


Figure 10-76 The Xilinx Flow Engine

The status bar shows the progress of the implementation flow with the following stages:

- **Translate**—convert the design EDIF file into an NGD (Native Generic Design) file
- **Map**—group basic elements (“bels”) such as flip-flops and gates into logic blocks (“comps”); also generate a logic-level timing report if desired
- **Place&Route**—place comps into the device, and route signals between them

- **Timing (Sim)**—generate timing simulation data and an optional post-layout timing report
- **Configure**—generate a bitstream suitable for downloading into and configuring a device

When the implementation completes, an Implementation Status box displays:

```
Implementing revision ver1->rev1 completed
successfully.
```

14. Click on View Logfile to display the logfile from the Flow Engine. The report is displayed in vi.
15. To exit the viewer, type `:q!` and press **Return**.
16. Click **OK** in the Implementation Status dialog to return to the Xilinx Design Manager.

Note: To use another text editor, such as Emacs, as the report viewer, select **File** → **Preferences** from the Xilinx Design Manager.

Performing Timing Simulation

Timing simulation uses the block and routing delay information from the routed design to give a more accurate assessment of the behavior of the circuit under worst-case conditions. Also, since the delay-annotated timing netlist is different from the original schematic design, the timing simulation uses a process called *cross-probing* to allow you to view simulation results on your schematic. In this section, you perform a timing simulation of the Calc design by first preparing the design using `pld_edif2tim`. Once this has been done, you run `pld_quicksim` with cross-probing to trace waveforms and annotate results onto your original schematic.

Using Pld_edif2tim to Prepare a Timing Simulation

`Pld_edif2tim` reads a routed EDN file and back-annotates the delays to the schematic. This includes a number of steps, all of which are automatically run by the `pld_edif2tim` script. This script is represented by the `pld_edif2tim` icon in `pld_dmgr`. The files necessary for back-annotation have either been created in the Design Architect tutorial or are included in the solution directories.

Use `pld_edif2tim` to prepare the design for timing simulation as follows:

1. In `pld_dmgr`, use the Navigator to find and select the EDN calc icon.

This represents the timing-annotated netlist generated by the Xilinx Design Manager.

Note: There may be two similar looking types of icons, one marked EDIF and the other marked EDN. An EDIF file represents a netlist translated from the original schematic, while an EDN file represents a netlist translated from a routed NCD file. Be sure you select an EDN file to prepare for timing simulation.

2. Select **Right Mouse Button** → **Open** → `pld_edif2tim`.

A dialog box appears. Design Manager automatically fills in the dialog box with the name of the EDN file.

3. Verify that the Replace existing routed design library field is set to **NO**.

On subsequent executions of `pld_edif2tim`, you may set this to **YES** if you are overwriting a previous timing model.

4. Press **return** or select **OK** to execute the command.

The script produces a shell and runs in it.

Examining the `Pld_edif2tim.log` File

Examine the `pld_edif2tim.log` file as follows:

1. In `pld_dmgr`, select **Right Mouse Button** → **Update Window**.

The window is updated with the files that `pld_edif2tim` generated.

2. Find the `pld_edif2tim` LOG file and select it with the left mouse button.

3. Choose **Right Mouse Button** → **Open** → **Editor** to open the file in the editor.

No errors or warnings should be reported. For a short summary of the commands executed by `pld_edif2tim` during the timing flow, see the “Command Summaries” section at the end of this

chapter. The timing flow is always the same since the starting point is always a routed EDN file with delays.

4. When you have finished looking at the file, close the Editor window.

Note: New for the 2.1i Xilinx/Mentor Interface `pld_edif2tim` now runs `pld_dve`, and users will not have to run this step separately.

Invoking QuickSim for Timing Simulation

1. With the timing-annotated calc component in the `calc_lib` directory selected, invoke `pld_quicksim` by selecting **Right Mouse Button** → **Open** → `pld_quicksim`.

A dialog box appears. The component name, Calc, is entered automatically with a fully qualified path.

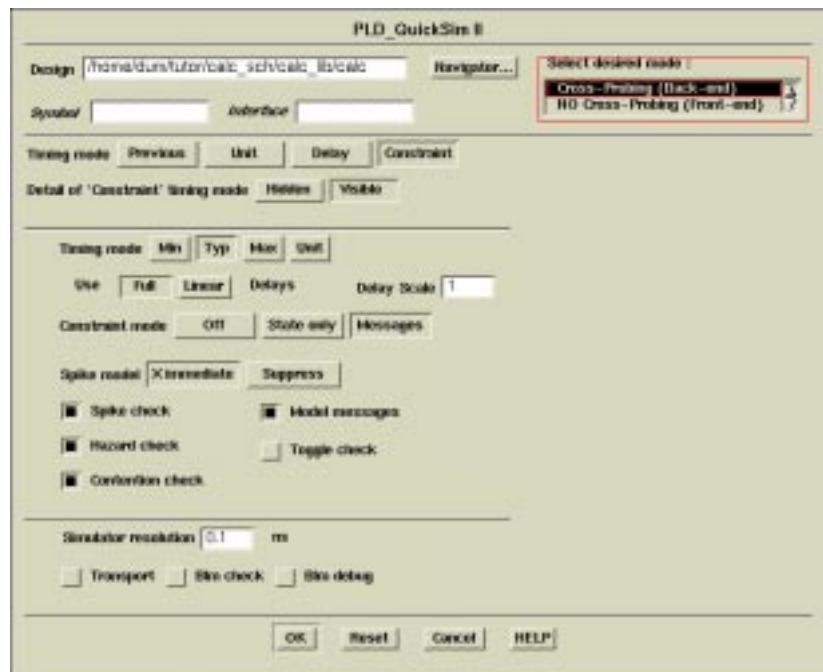


Figure 10-77 Invoking `Pld_quicksim` for Timing Simulation

2. Select desired mode as **Cross-Probing**.

This allows QuickSim to use the back-annotated timing model in QuickSim, while allowing you to view the original schematic in DVE. This process is necessary because your original schematic is expressed in Unified libraries, while the back-annotated timing model is generated using simulation primitives.

3. Select the **Constraint** option for Timing mode.
4. Select the **Visible** option for Detail of 'Constraint' timing mode.

A new set of buttons appears in the dialog box.

5. Select **Typ** for Timing mode.

This specifies the use of the back-annotated timing information.

6. Select **Messages** for Constraint mode.
7. Leave the rest of the buttons set at their defaults, and press **return** to start QuickSim.

For more information on these other options, refer to the Mentor Graphics documentation on QuickSim. For most Xilinx simulations, the above setup is appropriate.

The Design Viewpoint Editor (DVE) appears and gives an informational message reading,

```
To start the cross-probing process, ...
```

8. Click **Close** in this message window.
9. Resize the DVE window so that it is almost as large as the entire screen.

The QuickSim window also appears.

10. Resize the QuickSim window so that it is almost as large as the entire screen, allowing space for you to click on the DVE window to make it active and display in the foreground.
11. Bring the DVE window to the foreground and select **OPEN DESIGN VIEWPOINT** from the palette.
12. In the dialog box, enter the name of the *original* component in the Component field, e.g., `$XILINX_TUTORIAL/calc_sch/calc`. (Do *not* enter the name of the simulation model, `$XILINX_TUTORIAL/calc_sch/calc_lib/calc`.)
13. Click **OK** to load the original viewpoint.

14. Select **OPEN SHEET** from the palette to open the top-level Calc schematic.
15. To see the cross-probing process in action, click on the **CLK** net attached to the **CLOCKGEN** component.

A few seconds after this net is selected in DVE, a Trace window appears in QuickSim listing the CLK net.

16. Bring the QuickSim window to the foreground to see the Trace window.
17. Select **Transcript** → **Replay** from the QuickSim menu bar.
18. From the dialog box, choose the **calc_4ke.do** file. (Depending on your target device, select **calc_3ka.do**, **calc_5k.do**, or **calc_9k.do**.)

This replays a transcript file similar to the one created earlier. This transcript file opens the design; opens Trace and Monitor windows with the correct signals; assigns stimulus to the signals; and then runs the simulation. It should be obvious when you look at the Trace output that real delay values are being used. It may be useful to view the transcript file using the editor in **pld_dmgr** or another editor.

The figure below shows how DVE and QuickSim may look like running side by side.

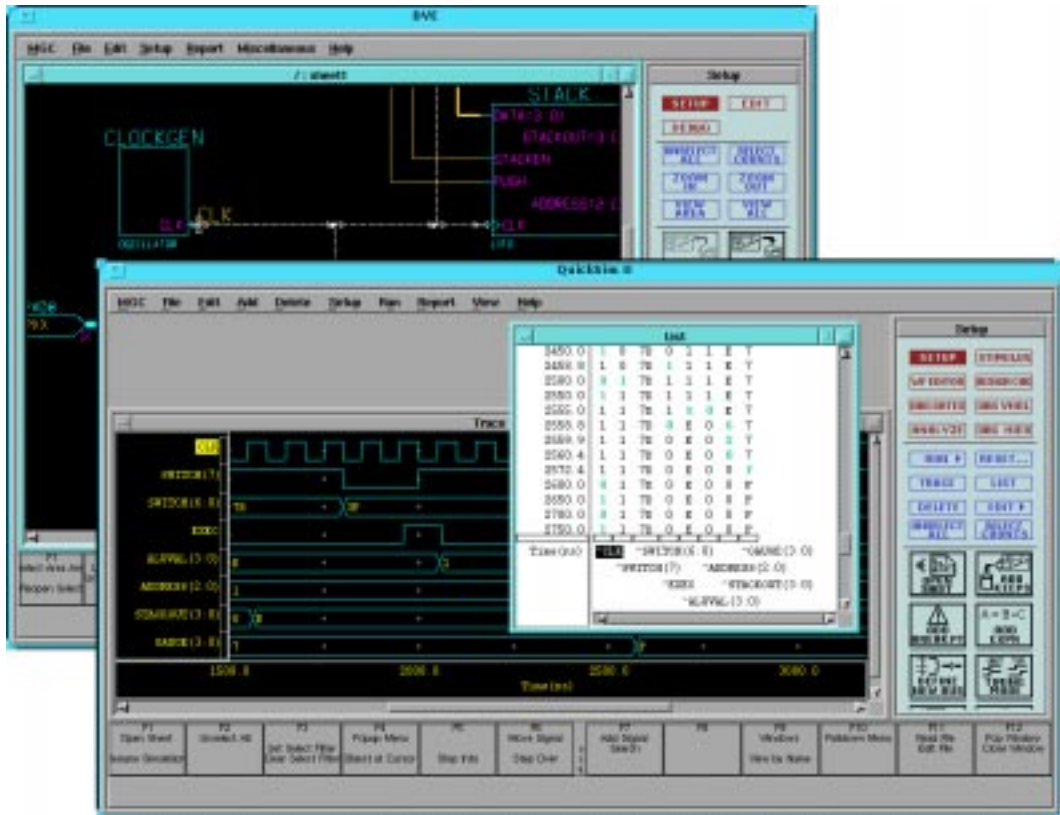


Figure 10-78 Cross-Probing with DVE and QuickSim

19. After examining the waveforms in timing simulation, close both the QuickSim and DVE windows.

Examining Routed Designs with FPGA Editor

Note: This section applies only to FPGA designs. If you are targeting a CPLD such as an XC9000 device, skip to the “Making Incremental Design Changes” section.

At this point in the tutorial, the design process is complete. If you would like to see how the design has been implemented by the Xilinx software, you can take a graphic look at your placed and routed design using the FPGA Editor. You can access the FPGA Editor from the Xilinx Design Manager.

FPGA Editor provides several useful functions, such as:

- Manual placement of a pre-routed design
- Manual editing of a routed design
- Static timing analysis



Figure 10-79 FPGA Editor Icon

FPGA Editor is explained in the Xilinx Software Manuals or online at http://support.xilinx.com/support/sw_manuals/2_1i/index.htm.

Verifying the Design Using a Demonstration Board

Note: This section applies only to FPGA designs. If you are targeting a CPLD such as an XC9000 device, skip to the “Making Incremental Design Changes” section

Creating and Downloading the Bitstream

A bitstream has been created during the Configure stage in Flow Engine. At this point, you are ready to download the bitstream using a parallel download cable or the more versatile XChecker cable connected to your workstation. The XC4000E version of the Calc design is suitable for download into an FPGA demonstration board available from Xilinx.

Downloading is accomplished with Hardware Debugger. To invoke Hardware Debugger, you select **Tools** → **Hardware Debugger** from the menu bar, or click the Hardware Debugger icon on the toolbar. If you are using an XChecker cable, you can also use the Hardware Debugger to read back information from the device to verify both the configuration as well as the state of memories and registers within the device.



Figure 10-80 Hardware Debugger Icon

Hardware Debugger is explained in the Hardware Debugger Guide, which can be found in the Xilinx Software manuals or online at http://support.xilinx.com/support/sw_manuals/2-1i/index.htm, in the Design Verification area. There is also a tutorial that you can follow along which is listed at the above URL.

Making Incremental Design Changes

After initially placing and routing a design, it is often necessary to go back to the schematic and make slight modifications to the original design. When this situation occurs, much of the place and route information from the previous design iteration can be “recycled,” since much of it is unchanged. This process is known as incremental design, and the NCD file (containing partition, placement, and routing information) from the prior place and route run is used as a guide.

Since much of the place and route information is extracted from the guide file, the place and route time is greatly reduced. The reuse of place and route information also results in more consistent timing over a number of guided place and route iterations. Once a section of your design passes your timing requirements, guided design ensures that it passes in the future, even if other parts of the design are modified.

In this section of the tutorial, you make a small change to the schematic and reprocess the design using the guide options available in the Xilinx Flow Engine.

Note: A small design change is the addition, removal, or replacement of only a small amount of logic in the design; the exact amount is dependent on the size of the design. If radical changes are made to a design, especially to existing portions of the design, it can be disadvantageous to guide the design. Also the device utilization will be a factor in guiding design. When a device is full it may be harder to guide when design changes are made.

Making an Incremental Schematic Change

Make a simple change to the Calc schematic that is visible immediately on the demonstration board. For example, assume that the reset opcode is no longer needed and needs to be removed from the design. This can be done by grounding the ‘R’ pins that are inputs to

the FDRE and FD4RE macros in the ALU schematic. The logic that generated the original reset signal, and the logic it drove, is automatically optimized out of the netlist by the MAP program.

Open `p1d_da` and load the Calc schematic as follows.

1. From `p1d_dmgr`, select the `$XILINX_TUTORIAL/calc_sch/alu` design object and choose **Right Mouse Button** → **Open** → `p1d_da`.

Design Architect appears with the ALU schematic loaded.

2. Use the **F8** key, or the stroke 159, to zoom in on the lower right quadrant of the schematic.
3. Press the **F2** key to make sure nothing is selected.
4. Select the **AND5B2** component that generates the QRESET net feeding the FDRE and FD4RE.
5. Press the **Delete** key to delete the component.
6. Connect a ground symbol to the dangling QRESET net.

The GND symbol can be found in the **BY TYPE** → **general** section of the Xilinx Library menu. See the figure below.

7. Check and save the schematic.
8. Exit `p1d_da` and return to `p1d_dmgr`.

Note: Be sure to run `p1d_men2edif` to write out a new EDIF netlist to implement with.

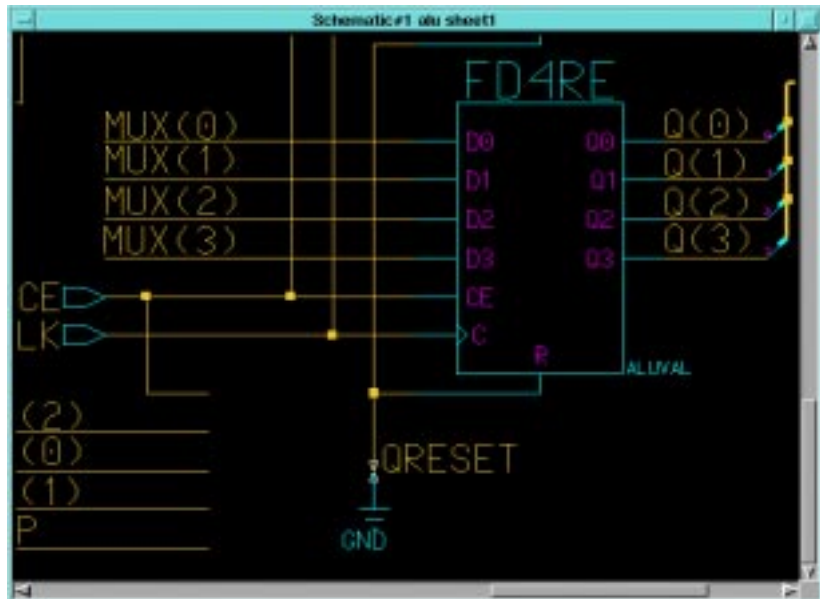


Figure 10-81 Grounding the Reset Logic

Translating the Incremental Design

Translate the guided Calc design by turning on the guide option in the New Version window.

1. In the Xilinx Design Manager, select **calc**, then choose **Design** → **New Version**.
2. The New Version dialog box appears with the Name field automatically filled in as **ver2**. You may also add a comment to the new version. This comment appears in the project view next to the version number. Click **OK**.

Note: You can add a comment to any version or revision in the project view by selecting that version or revision, then selecting **Right Mouse Button** → **Properties**.

3. In the New Version window, 'Copy Persistent Data' section use the button to choose **ver1** → **rev1** in the 'Guide File(s) dialogue box.
4. Click **OK** to close the dialog box.

5. Run the implementation as before by clicking the right arrow,



or by doing **Design** → **Implement**.

6. When all steps have completed successfully, you can check to see how much of the design was guided by looking at the Place and Route Report.

Verifying the Change in the Demonstration Board

Verify that the change was performed by downloading the new bitstream to the demonstration board, as you did previously. Make sure that the **ver2** → **rev1** revision is selected in the project view.

Downloading is accomplished with Hardware Debugger. To invoke Hardware Debugger, you select **Tools** → **Hardware Debugger** from the menu bar, or click the Hardware Debugger icon on the toolbar. If you are using an XChecker cable, you can also use the Hardware Debugger to read back information from the device to verify both the configuration as well as the state of memories and registers within the device.

Hardware Debugger is explained in the Hardware Debugger Guide, which can be found in the Xilinx Software manuals or online at http://support.xilinx.com/support/sw_manuals/2-1i/index.htm, in the Design Verification area. There is also a tutorial that you can follow along which is listed at the above URL.

Command Summaries

Although this tutorial uses the Mentor Graphics Design Manager and the Xilinx Design Manager to process the Calc design, you can also manually run the individual programs that these graphical tools run.

This section details command sequences that you can use to perform the translations the Xilinx Design Manager performs in this tutorial. The commands are written as you would type them at the system prompt or in a batch file. You may also see a summary of these system commands by using the **Utilities** → **Command History** and **Utilities** → **Command Preview** selections in Design

Manager or Flow Engine. Once you are in the Command History or Command Preview dialog box, select the display Mode to Command Line to see the detailed system commands, including command-line options, used by Flow Engine. You can cut and paste from these Command dialog boxes into your text editor to create batch files.

Note: The commands listed here are slightly different from the commands in the Command History and Command Preview windows. The commands listed here show how you would typically execute the Xilinx programs from the system prompt, outside of the Xilinx Design Manager framework. The commands listed in the Command History and Command Preview windows reflect how Flow Engine executes Xilinx programs to fit into the Xilinx Design Manager framework.

XC4000E Command Summaries

Functional Simulation

```
p1d_dve -s calc xc4000e
p1d_quicksim calc
```

Basic Translation

```
p1d_men2edif calc xc4000e
ngdbuild -p XC4000E -uc calc_4ke.ucf calc.edif
  calc.ngd
map -p XC4003E-4-PC84 -o calc_map.ncd -oe normal
  calc.ngd calc.pcf
trce calc_map.ncd -a -o calc_map.twr
par -w -l 4 calc_map.ncd calc.ncd calc.pcf
trce calc.ncd -a -o calc.twr
ngdanno calc.ncd calc_map.ngm
ngd2edif -v mentor -w calc.nga calc.edn
bitgen calc.ncd -l -w
```

Timing Simulation

```
p1d_edif2tim calc.edn
p1d_quicksim calc_lib/calc -cp -tim typ -consm
  messages
```


Incremental Translation

```
mv calc.ncd calc_guide.ncd
mv calc_map.mdf calc_guide.mdf
pld_men2edif calc xc4000e
ngdbuild -p XC4000E -uc calc_4ke.ucf calc.edif
calc.ngd
map -p XC4003E-4-PC84 -o calc_map.ncd -oe normal
-gf calc_guide.ncd calc.ngd calc.pcf
trce calc_map.ncd -a -o calc_map.twr
par -w -l 4 -gf calc_guide.ncd calc_map.ncd
calc.ncd calc.pcf
trce calc.ncd -a -o calc.twr
ngdanno calc.ncd calc_map.ngm
ngd2edif -v mentor -w calc.nga calc.edn
bitgen calc.ncd -l -w
```

XC9000 Command Summaries

Functional Simulation

```
pld_dve -s calc xc9000
pld_quicksim calc
```

Basic Translation

```
pld_men2edif calc xc9000
cpld -p XC95108-10-PC84 calc
```

Timing Simulation

```
ngd2edif -v mentor -w calc.nga calc.edn
pld_edif2tim calc.edn
pld_quicksim calc_lib/calc -cp -tim typ -consm
messages
```

Incremental Translation

```
cpld -p XC95108-10-PC84 -pinlock calc
```

Further Reading

The Schematic Design Tutorial is provided to give you the information necessary to begin a Xilinx design using Mentor Graphics software. It is important to note that tools as broad and complex as Design Architect and QuickSim cannot be fully explained in a single tutorial. There are many different ways to use the commands in these tools, and there are also many ways to customize these applications. It is strongly recommended that you read the Mentor Graphics Design Architect and QuickSim documentation as well as the other chapters in this *Mentor Graphics Interface Guide*.