Chapter 11

Schematic-on-Top with VHDL Tutorial

This chapter contains the following sections:

Introduction

This chapter guides you through a typical field-programmable gate array (FPGA) and complex programmable logic device (CPLD) design procedure from schematic entry with instantiated HDL 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 link HDL entities to Mentor symbols and instantiate those symbols into the Calc design. Next, you use QuickSim Pro, the Mentor Graphics schematic/HDL 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 using pld_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. Unlike the "Schematic Design Tutorial", this tutorial focuses completely on FPGAs. For information on retargeting a design to a different device family, see the "Targeting the Design for the XC9000 Family" section of the "Schematic Design Tutorial".

This tutorial includes instructions on the following:

- Installing the tutorial files
- Using Mentor Graphics Design Manager
- Using Mentor Graphics Design Architect
- Completing the SEG7DEC and ALU blocks in the Calc design
- Performing functional simulation on the Calc design in QuickSim Pro
- 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
- Command summaries

Required Background Knowledge

Note: This tutorial focuses specifically on the procedures for importing VHDL modules into Design Architect schematics. Refer to the “Schematic Design Tutorial” for information on basic object-management procedures in Design Manager, schematic-entry procedures in Design Architect, or Xilinx-specific concepts such as CONFIG, STARTUP, and incremental design.

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.
- Basic knowledge of Mentor Graphics tools, such as editing MGC location maps, manipulating design objects and launching applications in Design Manager, and adding and working with design elements in Design Architect. If you are not familiar with these procedures, see the “Schematic Design Tutorial” chapter.

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 top-level schematic design with instantiated VHDL modules using the Mentor interface.

Note: To instantiate Verilog modules into a schematic in Design Architect, you must first encapsulate them within a VHDL entity. You then encapsulate the VHDL entity into the schematic.

The tutorial design is targeted for an XC4000E 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.

Software Installation

Required Software

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

- Mentor Graphics Version C.2, 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
- A third-party FPGA / CPLD synthesis tool, such as LeonardoSpectrum (Exemplar Logic)
- Model Technology ModelSim 5.2d
- 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, 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
   - Model Technology ModelSim 5.2d
   - A third-party FPGA / CPLD synthesis tool, such as LeonardoSpectrum (Exemplar Logic)
3. Verify the installation, using the “Configuring Your System” section of the “Getting Started” chapter as a guide.

4. Add a reference to $XILINX_TUTORIAL to your MGC_LOCATION_MAP file.

   All of the tutorial designs use the variable $XILINX_TUTORIAL as part of their path references. For example, the design object alu in the $XILINX/mentor/tutorial/calc_sot directory uses the path reference $XILINX_TUTORIAL/calc_sot/alu to define where it is located in the directory structure.

   With this definition added to the location map as defined in the "Getting Started" chapter, 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
   ```

   This assumes the Xilinx tutorial files have been placed under /home/bclinton/mentor/xtutorial. For example, the full path to the schematic-on-top tutorial would be /home/bclinton/mentor/xtutorial/calc_sot

   Refer to the "Schematic Design Tutorial" chapter or 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 a design object is created 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 a schematic named calc is created, 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_sot 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 11-1 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 directories, and the bitstream file. Different intermediate files are created for different device families. Do not overwrite any files in the solutions directories.

The calc_sot directory contains the incomplete copy of the tutorial design. The installation program copies a few intermediate files to the calc_sot tutorial directory, and you create the remaining files when you perform the tutorial. In a later step, you copy the calc_sot directory to another area and perform the tutorial in this new area. Each design component directory has associated with it a file called *file-name*.mgc_component.attr. This file identifies the corresponding directory as a Mentor Graphics design component.

Starting the Design Manager

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

```
pld_dmgr
```

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

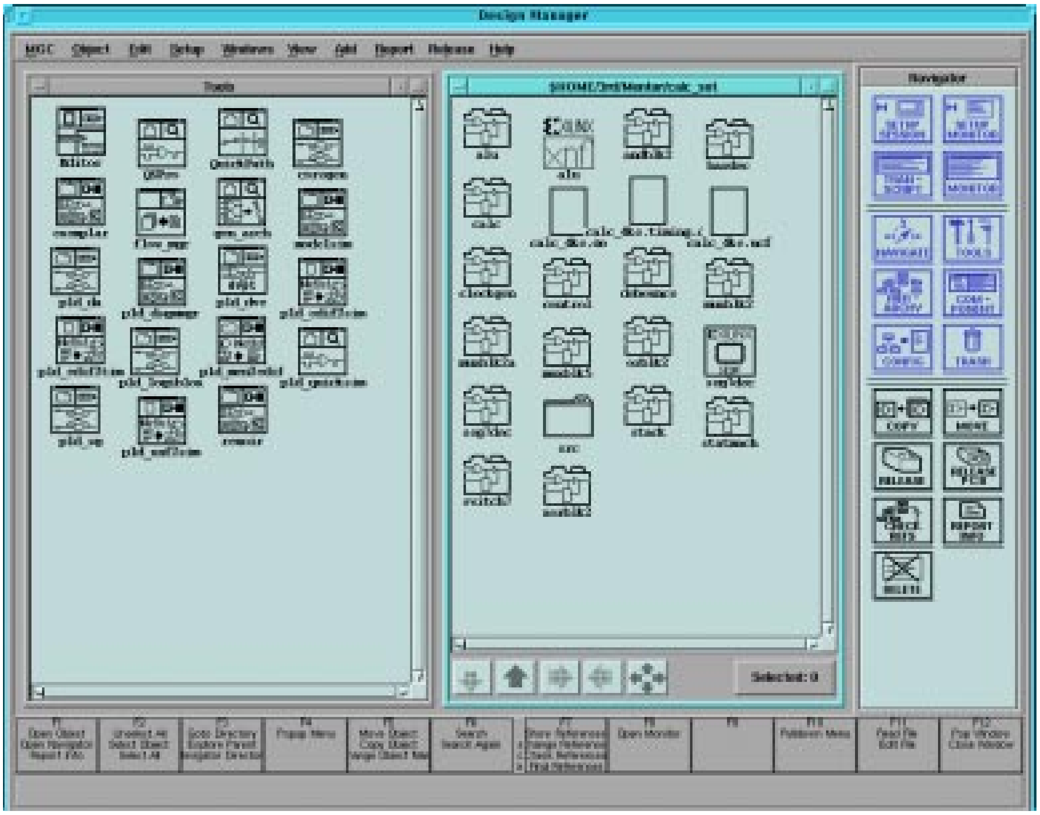

Design Manager
MGC Object Edit Setup Windows View Add Report Release Help
Tools
$HOME/3rd/Mentor/calc_sel
Navigator
Selected: 0

Figure 11-1 Mentor Design Manager Window

Note: This tutorial assumes you are familiar with basic Design Manager procedures. For more information on these procedures, see the “Schematic Design Tutorial”.

Copying the Tutorial Files

The schematic-on-top Calc tutorial files are located in the /tutorial/calc_sot directory. To use the Copy operation in Design Manger to copy these files to a local area, perform the following steps:

1. In the Navigator window, move to the directory where the tutorial files were installed.
2. Select the calc_sot 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_sot directory selected in the Navigator window, choose **Right Mouse Button** → **Edit** → **Copy**.
6. In the dialog box that appears, type the directory path where you want to copy the working copy of the tutorial files.

   For example, to copy the files to /home/dum/tutor/mentor, enter the following:

   /home/dum/tutor/mentor/calc_sot
7. To keep the design references intact, select **Options** → **Convert References? No** from the Copy dialog box.

   Normally, you allow Design Manager to update design references when you copy a design directory. In this case, however, you should leave the $XILINX_TUTORIAL reference intact. To make sure the proper directory is referenced, modify the MGC_LOCATION_MAP accordingly after you copy the tutorial design.

8. Click **OK** to exit the Options dialog box.
9. Click **OK** again to exit the Copy dialog box and start the Copy process.
10. Use the Navigator to change directories to the location of the working copy of calc_sot. (In the example above, you would click the **four-arrow button** at the bottom of the Navigator window, then type **/home/dum/tutor/mentor/calc_sot** in the dialog box.)
11. Modify your MGC_LOCATION_MAP file so that the $XILINX_TUTORIAL variable points to the directory where the copy of calc_sot 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 that opens, type:

    $MGC_LOCATION_MAP

14. Click **OK**.

    The $XILINX_TUTORIAL soft name now points to the new tutorial area.

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_sot** in the Directory field of the dialog box, then select **OK** or press return. This sets the

working directory to the directory where you work on the tutorial.

3. Select the **`$XILINX_TUTORIAL/calc_sot/calc`** design object in the Navigator window.
4. Select **`Right Mouse Button`** → **`Open`** → **`pld_da`**.

   The Design Architect window appears and displays the Calc design as shown in the following figure.

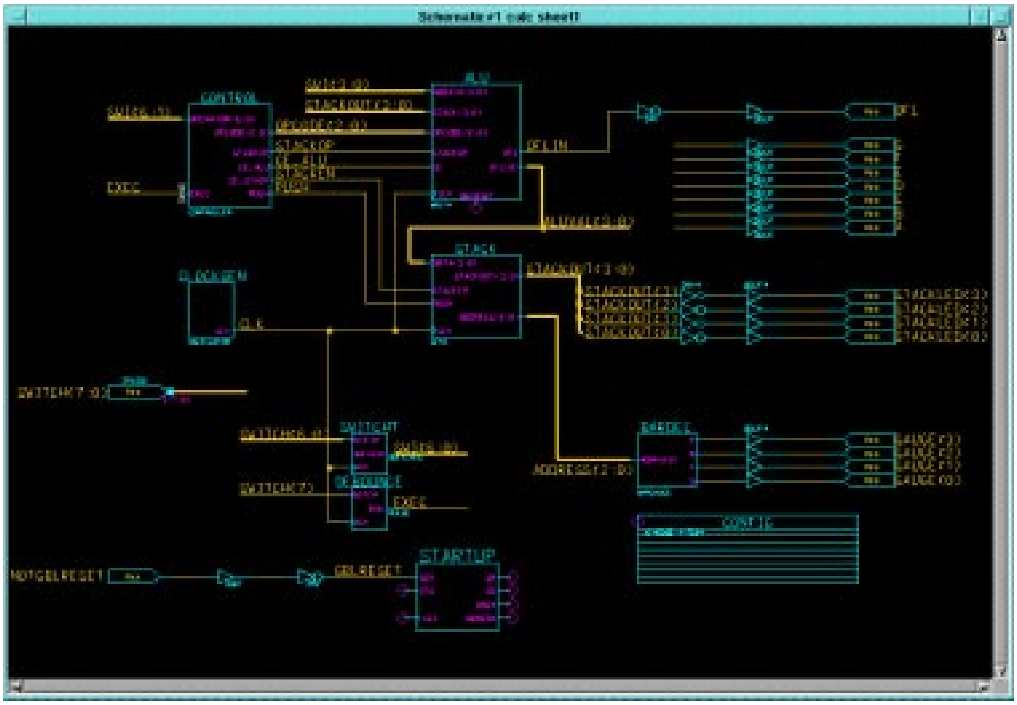


Figure 11-2 Top-Level Schematic for Calc

5. Resize the Design Architect window to cover the entire screen.

Completing the Calc Design

To complete the tutorial design, you need to link VHDL entities to symbols in the schematic using Vcom and 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.

Design Description

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.

In this tutorial, you create a new symbol for the SEG7DEC component from its associated seg7dec.vhd description, then instantiate that symbol onto the Calc schematic. You then link an existing ALU symbol to its associated alu.vhd description. A CONFIG block and a STARTUP block have already been added to the Calc design as well. For more information on using CONFIG and STARTUP, see the “Schematic Design Tutorial”.

The design consists of the following functional blocks:

- **ALU**—The arithmetic functions of the processor are performed in this block. This block is defined by the VHDL file alu.vhd.
- **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.
- **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. You generate the symbol for this module from its behavioral VHDL description in seg7dec.vhd.
- **CLOCKGEN**—This block uses an internal oscillator circuit in XC4000E devices to generate the clock signal.
- **BARDEC**—This block shows how many items are on the stack on a “gauge” of four LEDs.
- **SWITCH7**—This is a user-defined module consisting of seven input flip-flops used to latch the switch data.

Note: Basic Xilinx design concepts such as assignment of pin properties, use of STARTUP and CONFIG, and incremental design methodology as well as details about Xilinx device architecture are not covered in this chapter. For more information on these topics, see the “Schematic Design Tutorial” chapter.

Adding the SEG7DEC Component

On the Calc schematic, notice a space near the upper-right corner of the schematic between the ALUVAL(3:0) bus and the inputs to seven OBUF elements that feed LED outputs A through G. This is where the SEG7DEC symbol must be placed.

This component has a VHDL file, seg7dec.vhd, that describes its behavior. In this exercise, you compile the seg7dec.vhd file for simulation, generate a symbol for it, then instantiate the entity into the Calc schematic so that it may be simulated and built into the implemented device.

Compiling the VHDL Entity

To compile the VHDL file for SEG7DEC, follow these steps:

1. Take a look at the src/seg7dec.vhd file with the text editor you normally use.

   This entity includes a case statement that decodes a four-bit number into a set of seven signals suitable for display on a seven-segment LED display. The src/seg7dec.vhd file is as follows:

   ```
   process (Q)
     begin
   ```

```
    case Q is
      when "0000" => DISPLAY <= "0000001";
      when "0001" => DISPLAY <= "1001111";
      when "0010" => DISPLAY <= "0010010";
      when "0011" => DISPLAY <= "0000110";
      when "0100" => DISPLAY <= "1001100";
      when "0101" => DISPLAY <= "0100100";
      when "0110" => DISPLAY <= "0100000";
      when "0111" => DISPLAY <= "0001101";
      when "1000" => DISPLAY <= "0000000";
      when "1001" => DISPLAY <= "0000100";
      when "1010" => DISPLAY <= "0001000";
      when "1011" => DISPLAY <= "1100000";
      when "1100" => DISPLAY <= "0110001";
      when "1101" => DISPLAY <= "1000010";
      when "1110" => DISPLAY <= "0110000";
      when others => DISPLAY <= "0111000";
    end case;
end process;
```

2. Close the src/seg7dec.vhd file.
3. To create a VHDL work library where compiled entities will reside, type the following at the system prompt from within the $XILINX_TUTORIAL/calc_sot directory:

   `vlib` *mywork*

   A library directory called “mywork” now exists in the tutorial project directory.
4. Map this directory to “work” so that the VHDL compilation programs can recognize it:

   `vmap work` *mywork*

   You should now have a new file in your tutorial directory called modelsim.ini, which contains the following library entry:

   ```
   [Library]
   work = mywork
   ```

   This allows the VHDL-simulation programs to recognize this as the working directory.
5. Compile the src/seg7dec.vhd file with the vcom command:

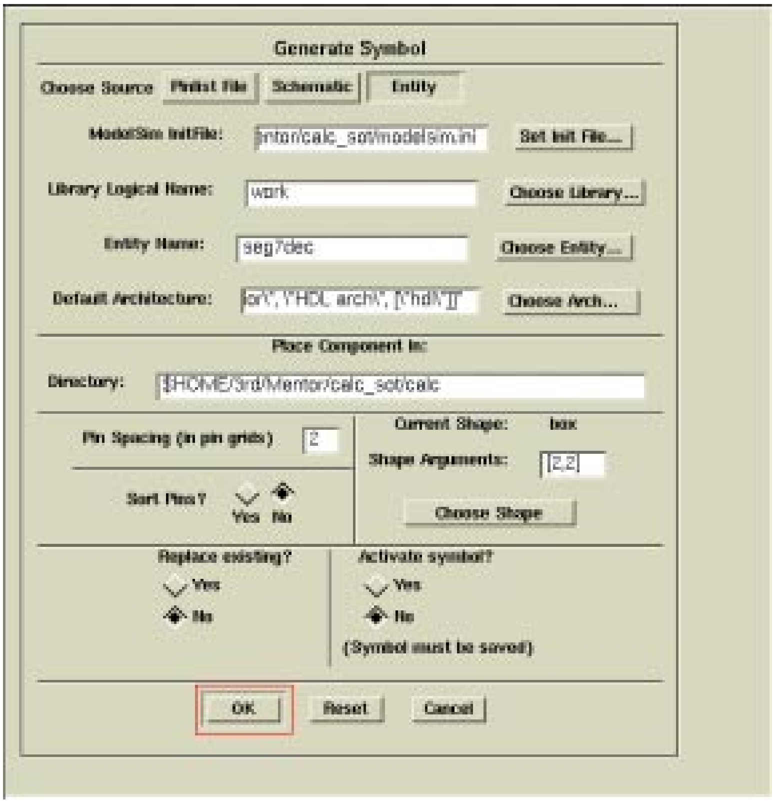
   `vcom –qspro_syminfo src/seg7dec.vhd`

The –qspro_syminfo option tells VCOM to write out information needed by the Design Architect Symbol Generator.

6. With the Calc schematic still open in Design Architect, select **`Miscellaneous`** → **`Generate Symbol`** from the menu bar.

7. In the Generate Symbol dialog box that appears, select **`Choose Source`** → **`Entity`**.

   The dialog box fields change as shown in the Generate Symbol dialog box in the following figure.



ModelSim InitFile: ntor/calc_sot/modelsim.ini
Library Logical Name: work
Entity Name: seg7dec
Default Architecture: or\",\"HDL arch\", [\"hdl\"]]"
Directory: $HOME/3rd/Mentor/calc_sot/calc
Pin Spacing (in pin grids) 2
Current Shape: box
Shape Arguments: [2,2]
Sort Pins? Yes [✓ No]
Replace existing? Yes [✓ No]
Activate symbol? Yes [✓ No]
(Symbol must be saved)

Figure 11-3 Generate Symbol Dialog Box

8. In the Generate Symbol dialog box, make sure the fields are set as shown in the following table:

Table 11-2 Generate Symbol Settings for SEG7DEC

Field	Value
QVHDL InitFIle	$XILINX_TUTORIAL / calc_sot / modelsim.ini
Lib. Logical Name	work
Entity Name	seg7dec
Deflt. Architecture	“[\”behavior]”, \“HDL arch\”, [\“\hdl\”]]”
Place Comp. in Dir.	$XILINX_TUTORIAL / calc_sot
Pin Spacing	2
Shape Arguments	[2,2]
Sort Pins?	Yes
Replace existing?	No
Activate symbol?	No

Note: You can select the architecture setting from a list by click on the **`Choose Arch`** button. Since only one architecture has been compiled for the SEG7DEC entity, you may also leave this field blank.

9. Click **`OK`**.

   After a few moments, the Symbol Editor appears with the newly created SEG7DEC component. Note the properties attached to this symbol and its pins. These properties allow the underlying entity and the upper-level schematic portion to be simulated concurrently in QuickSim Pro.

Note: If you get the error “Entity source work_library / _parsed.vhd does not exist,” make sure you specified the –qspro_syminfo option on the VCOM command line.

10. Add “SEG7DEC” text to the symbol as shown. (If you do not know the procedure for this, refer to the instructions in the “Schematic Design Tutorial”.)

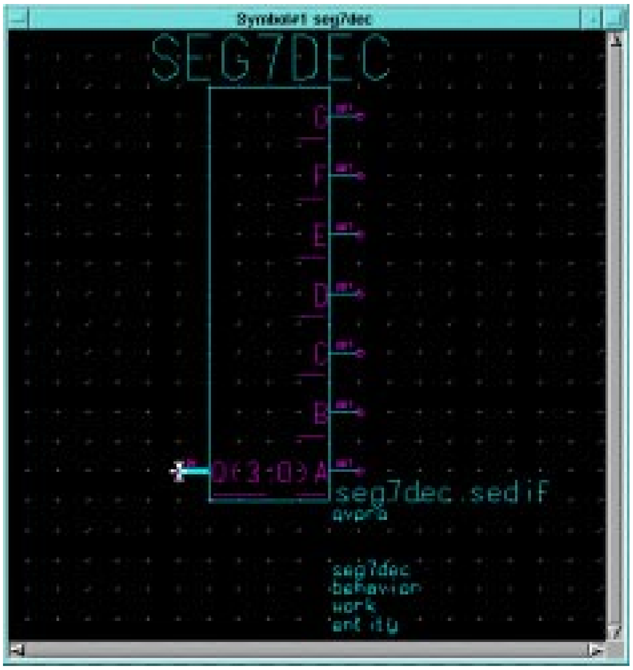


Figure 11-4 Generated SEG7DEC Symbol

All symbols that have an associated non-schematic model must have a FILE property attached to them so that the Xilinx netlister (NGDBUILD) can incorporate these portions of the design into the implemented design. The value of this property is the actual filename of the submodule netlist. This filename can have one of several different extensions, based on the netlist format it uses.

Table 11-3 Common FILE Property Extensions

Extension	Netlist Format
.edif	Generic Xilinx-compatible EDIF 2.0
.xnf	Generic XNF (Xilinx Netlist Format) 6.*x*
.sedif	Synopsys Xilinx-compatible EDIF 2.0
.sxnf	Synopsys XNF 6.*x*

The presynthesized SEG7DEC module included with this tutorial was generated by Synopsys' Design Compiler, which uses SEDIF.

11. Select the body of the SEG7DEC symbol and add the following property:

    Name: **`FILE`**
    Value: **`seg7dec.sedif`**

12. Check and Save the symbol.

    Now that a symbol exists for this component, you can instantiate it onto the top-level Calc schematic.

13. With the Calc schematic window active, click **`CHOOSE SYMBOL`** from the Schematic Palette and select the **`seg7dec`** component.

14. Instantiate this component between the ALUVAL(3:0) bus and the nets that lead to the outputs A-F as shown below.

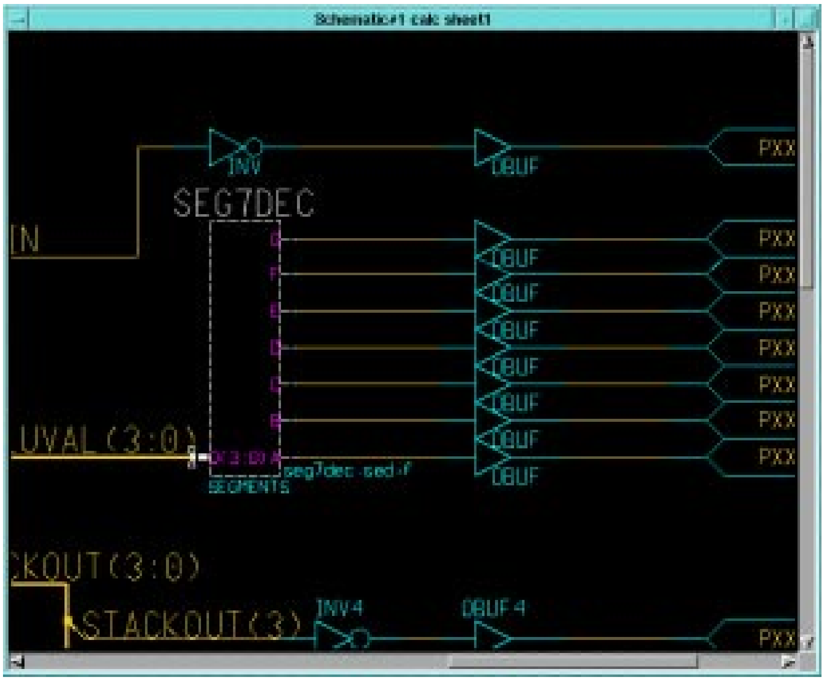


Figure 11-5 Adding the SEG7DEC Symbol

15. Add the instance name **`SEGMENTS`** to the SEG7DEC symbol. (Add property **`INST`**, value **`SEGMENTS`**.)

16. Check and Save the Calc schematic. Leave the schematic open.

Linking a VHDL Entity to the ALU Component

VHDL can also be associated with a pre-existing symbol. The following procedure links the alu.vhd entity with the ALU symbol, which has already been instantiated on the Calc schematic.

As with SEG7DEC, ALU has a VHDL file, alu.vhd, that describes its behavior. In this exercise, you compile the alu.vhd file for simulation, link the compiled model to the existing ALU symbol, then update the instantiated entity in the Calc schematic so that it may be simulated and built into the implemented device.

Compiling the VHDL Entity

To compile the VHDL file for ALU, follow these steps.

1. Take a look at the alu or seg7dec.vhd file with the text editor you normally use.

   This entity includes a four-bit data register, several multi-bit gate functions (AND, OR, and XOR), and an adder-subtractor. To model the device’s system-wide global set/reset, an additional port, GBLRESET, has been added from the schematic-based ALU design. (See the “Schematic Design Tutorial”.) This signal is brought into the process block that controls the four-bit data register:

   VHDL ALU Register Description:

```
process (CLK, GBLRESET)
begin
  if (GBLRESET='1') then
    QIN <= "0000";
    OFL <= '0';
  elsif (CLK'event and CLK='1') then
    if (CE='1') then
      if (QRESET='1') then
        QIN <= "0000";
        OFL <= '0';
      else
        QIN <= MUX;
        OFL <= OVER;
      end if;
    end if;
  end if;
end process;
```

The GBLRESET signal is written as an explicit asynchronous clear. This allows you to connect the registers in the ALU entity to the system-wide global-set / reset signal from the schematic. In an all-schematic design, the system-wide global-set / reset net does not need to be connected, since it is implicitly connected to all flip-flops in a device, both for simulation and implementation. For simulation purposes, however, this signal needs to be explicitly connected to all flip-flops in all VHDL modules. The Xilinx Core Tools recognize this as a redundant connection and subsequently trim it out of the implemented design.

2. When you are finished perusing the ALU VHDL description, close the file.

   Since a work directory already exists for compiling VHDL modules, you can go directly to compiling the ALU entity.

3. Compile the src / alu.vhd file with the VCOM command:

   ```
   vcom –explicit –qspro_syminfo src/alu.vhd
   ```

   The "-explicit" option allows VCOM to tolerate multiply-defined standard functions. This is required because the equality operator ("=") is defined in both the ieee.std_logic_1164 and ieee.std_logic_unsigned packages, both of which are called out in the alu.vhd file.

4. With the Calc schematic still open in Design Architect, select the ALU symbol.

5. Open down into the symbol by selecting **Right Mouse Button** → **Open Down**.

6. Select **symbol:alu**.

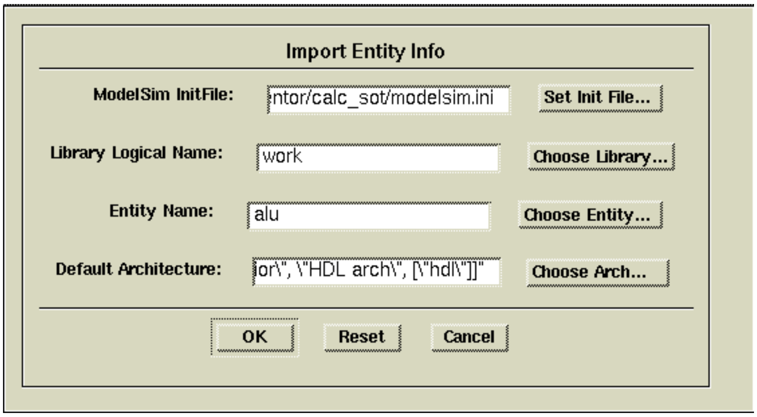
7. Click **OK**.

   The Symbol Editor appears.

   If you performed the Schematic Design Tutorial with the all-schematic-based Calc design, you should notice that the ALU symbol has an extra pin, GBLRESET, that corresponds to the extra VHDL port mentioned above.

8. From the menu bar, select **File** → **Import VHDL Entity**.

   The Import VHDL Entity dialog box appears.



Import Entity Info

ModelSim InitFile: ntor/calc_sot/modelsim.ini

Library Logical Name: work

Entity Name: alu

Default Architecture: ior\", \"HDL arch\", [\"hdl\"]]"

Figure 11-6 Import VHDL Entity Dialog Box

9. Set the following values in the dialog box:

Table 11-4 Import VHDL Entity Settings for ALU

Field	Value
QVHDL InitFIle	$XILINX_TUTORIAL / calc_sot / modelsim.ini
Lib. Logical Name	work
Entity Name	alu
Deflt. Architecture	“[\”behavior\”,\“HDL arch\”, [\“hdl\”]]”

Note: You can select the architecture setting from a list by clicking on the **`Choose Arch`** button. Since only one architecture has been compiled for the ALU entity, you may also leave this field blank.

10. Click **`OK`**.

    In the Symbol Editor, the ALU symbol’s body and pins are annotated with properties similar to those that Generate Symbol attached to the SEG7DEC symbol. These properties allow the underlying entity and the upper-level schematic portion to be simulated concurrently in QuickSim Pro.

Note: If you get the error “Entity source *work_library* / _parsed.vhd does not exist,” make sure you specified the –qspro_syminfo option on the VCOM command line.

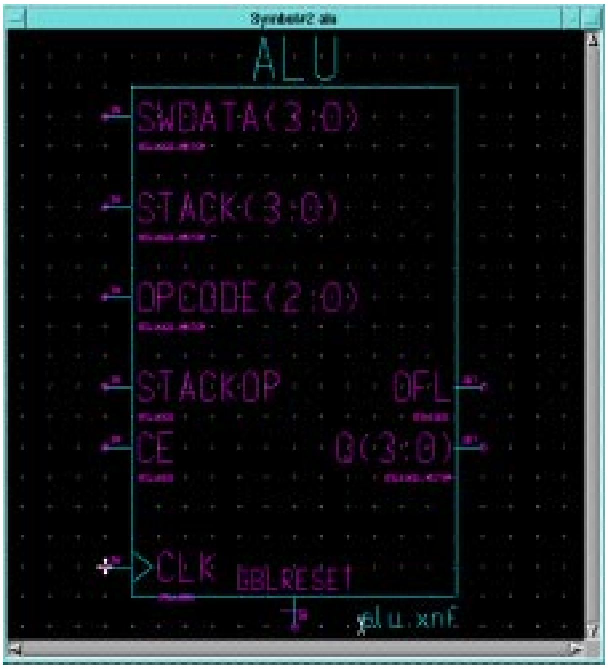


Figure 11-7 ALU Symbol with VHDL-Import Properties

Since the ALU module is also a non-schematic element, you must attach a FILE property to it so the Xilinx netlister (NGDBUILD) can incorporate this portion of the design into the implemented netlist. The ALU netlist included with this tutorial was synthesized by Exemplar’s Galileo, and thus uses generic XNF.

11. Select the body of the ALU symbol and add the following property:

    Name: **`FILE`**
    Value: **`alu.xnf`**

12. Check and Save the symbol.

    Since the symbol has now changed, you must reflect the change in the top-level Calc schematic.

13. With the Calc schematic window active, select the ALU symbol, then select **Right Mouse Button** → **Update** → **Auto**.

    This updates the ALU instantiation with the new symbol properties.

    The GBLRESET pin on the ALU symbol must be attached to the global set / reset signal from the top-level design. This signal is the one connected to the GSR input on the STARTUP block. In the case of Calc, this net is also called GBLRESET.

14. Attach a new net to the GBLRESET pin and name it GBLRESET.

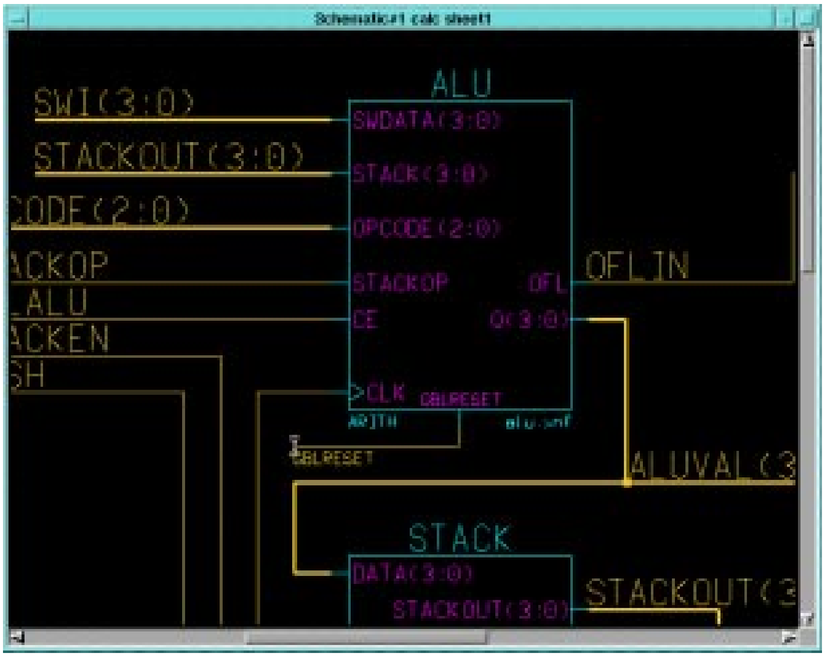


Figure 11-8 Updating the ALU Symbol

15. Check and Save the Calc schematic.

16. Exit Design Architect.

Using a Constraints File

Using a constraints file, you can supply constraints information in a textual form. An example of a constraints file is shown below. The example shows the user constraints file, calc_4ke.ucf, that is supplied with this tutorial. The constraints file syntax is the same for all device families.

Note: You may also place location constraints directly on the schematic. For more information, see the “Schematic Design Tutorial” chapter.

The place and route software must be instructed to read and apply the .ucf file when the design is read into the Xilinx Design Manager. The procedure for doing this is detailed later in the “Using the Xilinx Design Manager”.

Example Constraints File:

```
# CALC_4KE.UCF
# User constraints file for CALC, XC4003E-PC84
# Uses angle brackets, as per PLD_MEN2EDIF option

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;
NET NOTGBLRESET   LOC=P56;
```

Performing Functional Simulation

Functional simulation is performed before design implementation to verify that the schematic that you have designed is logically correct. All components in the Calc design have built-in simulation models so little pre-processing is necessary. However, every top-level schematic design in Mentor Graphics must have a simulation viewpoint before you can simulate it in QuickSim Pro. 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.

Note: The following instructions refer to two related but different programs: ModelSim and QuickSim Pro. ModelSim is Mentor Graphics' HDL-only simulator, while QuickSim Pro is a co-simulation tool that runs both ModelSim to simulate the HDL portions of a design and QuickSim to simulate the schematic portions. QuickSim Pro runs a process known as a FlexSim backplane through which the two "solvers" (ModelSim and QuickSim) communicate with each other.

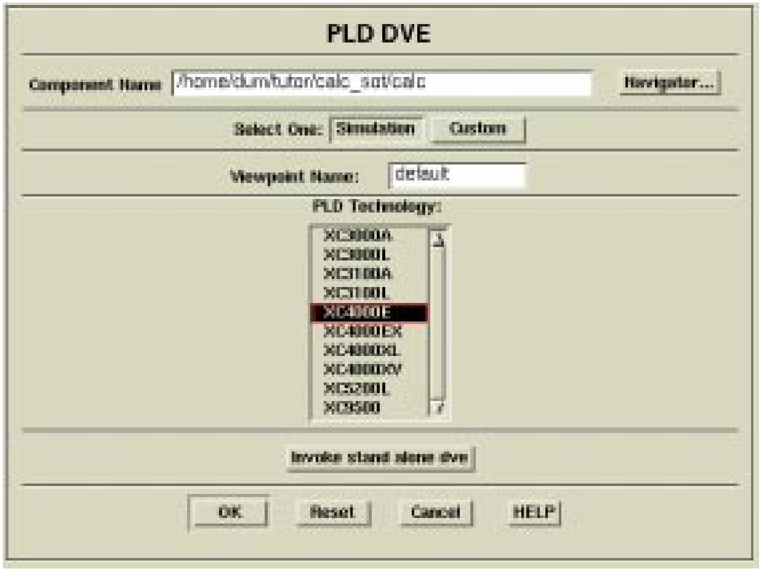
Also note that the instructions that follow do not have detailed information on basic QuickSim operations such as selecting nets and displaying Trace and List windows. For information on these procedures, please refer to the "Schematic Design Tutorial".

Using Pld_dve

To use the PLD Design Viewpoint Editor to generate a design viewpoint to tell QuickSim (as run by QuickSim Pro) how to interpret certain Xilinx-specific design properties, follow these steps.

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.



Component Name /home/dum/tutor/calc_sot/calc

Viewpoint Name: default

PLD Technology:

XC3000A
XC3000L
XC3100A
XC3100L
XC4000E
XC4000EX
XC4000XL
XC4000XV
XC5200L
XC9500

Figure 11-9 Invoking Pld_dve for Functional Simulation

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

4. Click OK.

   The pld_dve script executes.

5. Once pld_dve completes, dismiss the shell window in which it executed.

Invoking QuickSim Pro

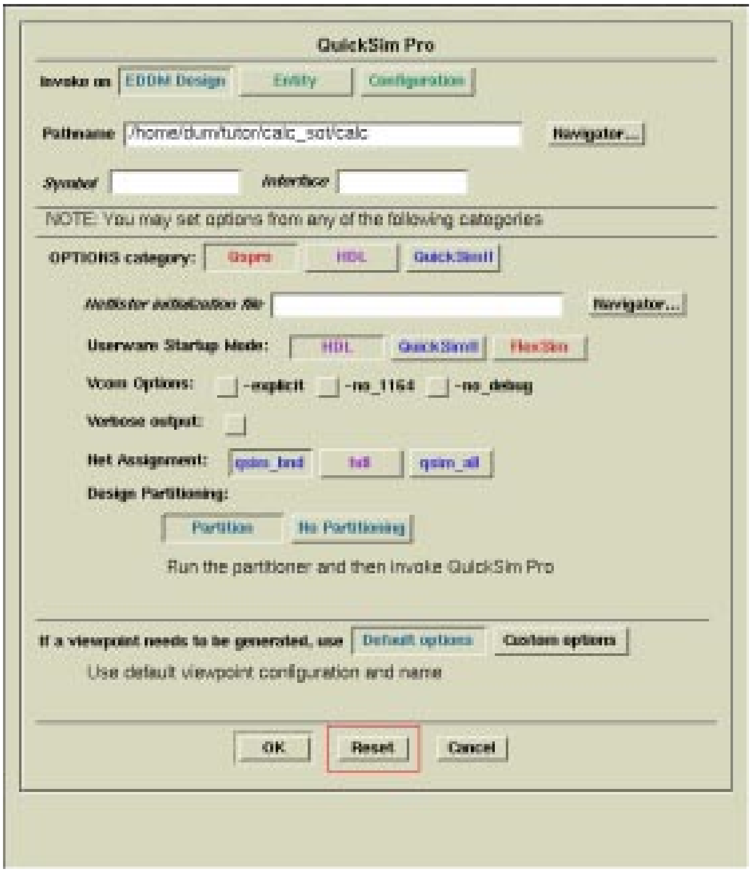
To invoke QuickSim Pro for functional simulation on the Calc design, follow these steps:

1. Select the Calc design object in the Navigator window.

2. Invoke QuickSim Pro on the design by selecting **`Right Mouse Button`** → **`Open`** → **`QSPro`**.

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

3. Since the top-level Calc design is a schematic (EDDM model), verify that EDDM Design is the design set to Invoke on.



QuickSim Pro

Invoke on [✓ EDDM Design] Entity Configuration

Pathname /home/dum/tutor/calc_sol/calc Navigator...

Symbol *Interface*

NOTE: You may set options from any of the following categories

OPTIONS category: [✓ Qspro] HDL QuickSimII

Netlister initialization file Navigator...

Userware Startup Mode: [✓ HDL] QuickSimII FlexSim

Vcom Options: -explicit -no_1164 -no_debug

Verbose output:

Net Assignment: [✓ qsim_bnd] hdl qsim_all

Design Partitioning:

[✓ Partition] No Partitioning

Run the partitioner and then invoke QuickSim Pro

If a viewpoint needs to be generated, use [✓ Default options] Custom options

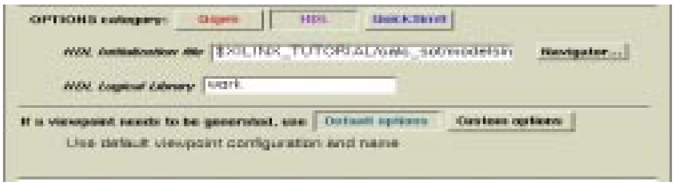
Use default viewpoint configuration and name

OK Reset Cancel

Figure 11-10 Invoking QuickSim Pro for Functional Simulation

Note the OPTIONS category buttons. These buttons allow you to set options for QuickSim Pro, as well as for each of the two simulation “solvers” (ModelSim and QuickSim). Each button brings up its own set of options in the OPTIONS panel. Each set of options is independent of the other two sets of options, and options in all three panels are applied concurrently when QuickSim Pro is run.

4. In the QuickSim Pro dialog box, select **`OPTIONS category`** → **`ModelSim`** to see the ModelSim options.


OPTIONS category: Qspro HDL QuickSimII

HDL Initialization file $XILINX_TUTORIAL/calc_sot/modelsim.ini Navigator...

HDL Logical Library work

If a viewpoint needs to be generated, use: Default options Custom options

Use default viewpoint configuration and name

Figure 11-11 ModelSim Options

5. Under ModelSim Initialization file, enter:

   `$XILINX_TUTORIAL/calc_sot/modelsim.ini`

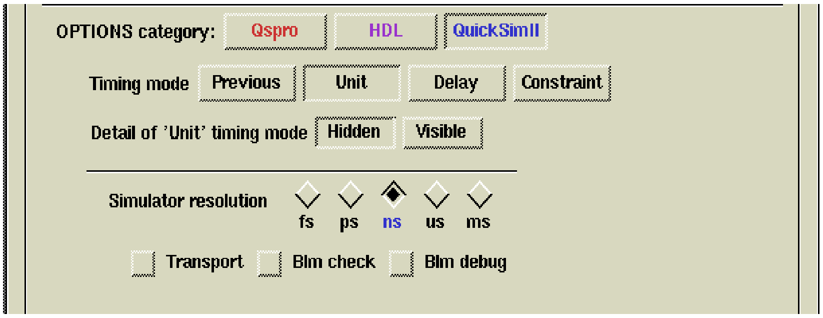
6. Under ModelSim Logical Library, enter:

   `work`

Note: The ModelSim initialization file defaults to the modelsim.ini file in the working directory. The ModelSim logical library defaults to the “work” directory as called out in the initialization file.

7. In the QuickSim Pro dialog box, select **`OPTIONS category`** → **`QuickSim`** to see the QuickSim options.

   The Unit timing mode should already be set, since this is the default. This tells the QuickSim solver to run in functional-simulation mode.


OPTIONS category: Qspro HDL QuickSimII

Timing mode Previous [✓ Unit] Delay Constraint

Detail of 'Unit' timing mode [✓ Hidden] Visible

Simulator resolution fs ps [✓ ns] us ms

Transport Blm check Blm debug

Figure 11-12 QuickSim Options

8. In the QuickSim Pro dialog box, select **`OPTIONS category`** → **`ModelSim`** and note that the settings you put in the ModelSim Options panel are still in place.

9. Click **`OK`** to invoke the QuickSim Pro simulator.

Two windows open, QSPro(QuickSim II) and QSPro(HDL). These represent the QuickSim and ModelSim simulation solvers, respectively, running within the integrated QuickSim Pro environment.

Viewing the Calc Schematic

When QSPro(QuickSim) starts, no schematic windows are open. 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.

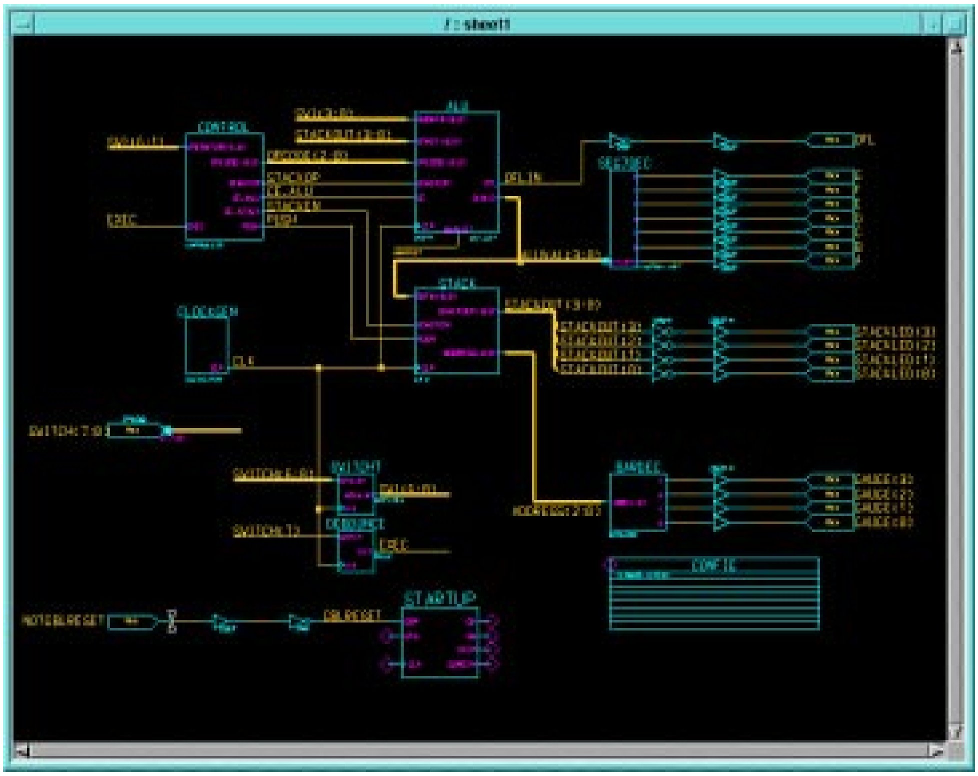


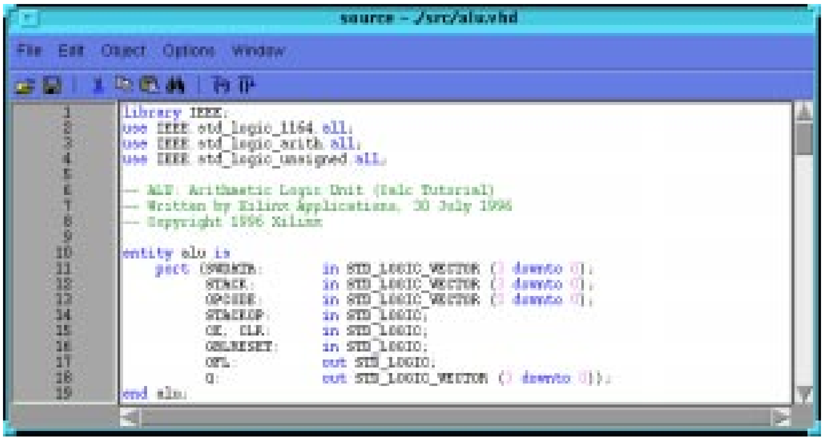
Figure 11-13 Top-Level Calc Schematic

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. Select the ALU symbol, then select Right Mouse Button → Open → Down.

   If this were a schematic module, the ALU schematic would appear in the QuickSim window. Instead, a Source window appears displaying the ALU VHDL description. This window is actually displayed through the ModelSim solver.



```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

-- ALU: Arithmetic Logic Unit (Calc Tutorial)
-- Written by Xilinx Applications, 30 July 1996
-- Copyright 1996 Xilinx

entity alu is
    port (SWDATA:        in STD_LOGIC_VECTOR ([illegible] downto [illegible]);
          STACK:         in STD_LOGIC_VECTOR ([illegible] downto [illegible]);
          OPCODE:        in STD_LOGIC_VECTOR ([illegible] downto [illegible]);
          STACKOP:       in STD_LOGIC;
          CE, CLK:       in STD_LOGIC;
          GBLRESET:      in STD_LOGIC;
          OFL:           out STD_LOGIC;
          Q:             out STD_LOGIC_VECTOR ([illegible] downto [illegible]));
end alu;
```

Figure 11-14   ALU Source Window

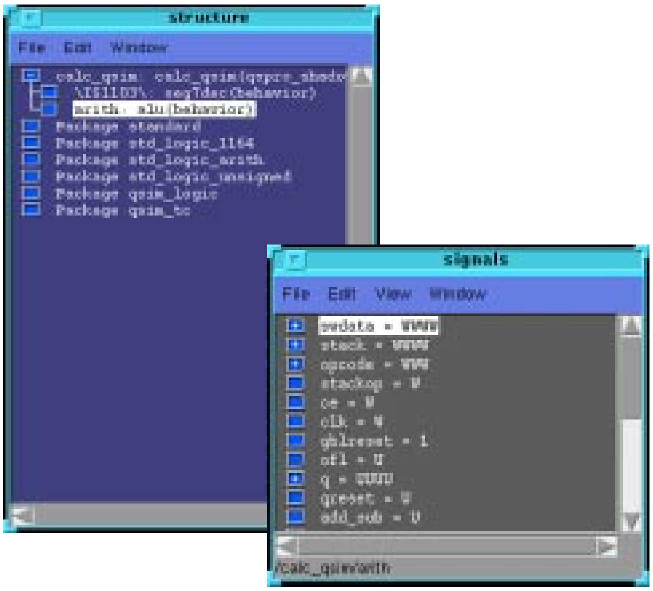
Viewing and Navigating the VHDL Hierarchy

ModelSim allows you to view the design hierarchy of the HDL portion of a mixed design. To see this hierarchy and its associated signals follow these steps:

1. From the QSPro(HDL) menu bar, select **`View`** → **`Structure`**.

   The Structure window appears as shown. Since the ALU source code is displayed in the Source window, the ALU entity is high-lighted.

2. Select **`View`** → **`Signals`** to display the Signals window which shows a list of all signals underneath the ALU entity.



```
calc_qsim: calc_qsim(qspro_shado
  \I$1103\: seg7dec(behavior)
  arith: alu(behavior)
Package standard
Package std_logic_1164
Package std_logic_arith
Package std_logic_unsigned
Package qsim_logic
Package qsim_tc
```

```
swdata = WWWW
stack = WWWW
opcode = WWW
stackop = W
ce = W
clk = W
gblreset = 1
ofl = U
q = UUUU
qreset = U
add_sub = U
```

/calc_qsim/arith

Figure 11-15 Structure and Signals Windows

3. With the left mouse button, select the seg7dec(behavior) entity in the Structure window.

   Notice that the Source window now changes to display the SEG7DEC VHDL file, and the Signals window changes to display the associated signals.

4. Select the top-level calc_qsim(structure) entity in the Structure window.

   Notice that the Source window displays VHDL code even though the top level was originally drawn on a schematic. This VHDL file was written by QuickSim Pro upon invocation so that ModelSim could simulate all instantiated VHDL modules from the top level.

5. Reselect the alu (behavior) module in the Structure window.

   The Signals listing displays the ALU signals. Note that buses in the Signals listing have "+" icons next to them. If you click on one of these icons, it changes to a "-" icon, and the list expands to include all bit signals within the corresponding bus.

6. Click the "-" icon to collapse the listing.
7. Select the following signals in the Signals window. You may have to scroll down the list to see them all.

   ```
   clk
   ce
   opcode
   q
   stack
   stackop
   swdata
   ```

   You can select a collection of signals by clicking on the first signal name with the left mouse button, then use a ctrl-key left mouse click combination on subsequent signal names. If you select a signal name you don't want selected, click on the signal name again with a ctrl-key left mouse button combination to unhighlight it.

   The waveforms for these signals can be traced in a ModelSim Wave window.
8. In the Signals window, select **`View`** → **`Wave`** → **`Selected`** signals.

   The Wave window appears as shown. Since the simulation has not yet been run, no waveforms are displayed.

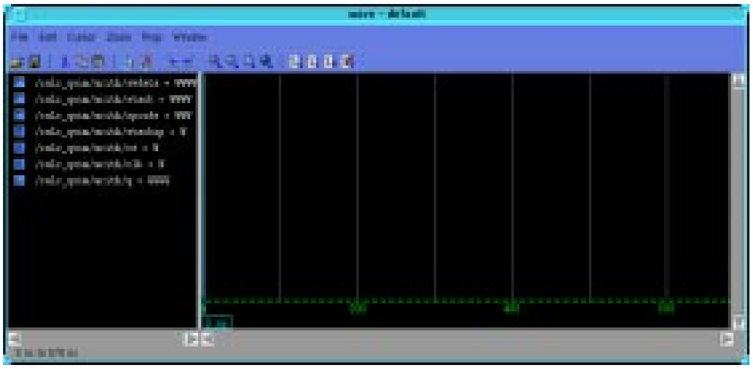


Figure 11-16 Wave Window at Time 0

In the next section, you complete the functional simulation and view the waveforms in the Trace (QuickSim) and Wave (ModelSim) windows.

Completing the Functional Simulation

You can automate the simulation by using a QuickSim command file. A “dofile,” calc_4ke.do, has been supplied with this tutorial for this purpose. This file opens Trace and List windows, sets up simulation vectors, and runs the simulation for a time duration of 3,400 ns. Use it as follows:

1. From the QuickSim menu bar, select **`MGC`** → **`Transcript`** → **`Replay`**.
2. In the dialog box, enter **`calc_4ke.do`** and click **`OK`**.

   The functional simulation runs automatically with Trace and List windows set up in the QuickSim window. Note that the Wave window under ModelSim is also updated to reflect the signals underneath the ALU component.

Note: For more specific information on using QuickSim features, see the “Schematic Design Tutorial”. The simulation command file executed here is similar to the one run in that chapter.

3. If the signal names and values in the Wave window are not fully visible, drag the divider between the signal list and the waveforms to see more of the signal names.
4. You can use the scroll bars to view different parts of the simulation in the Wave window. You can also zoom in and zoom out by selecting from the Zoom menu.
5. As with the Signals windows, the signal listing in the Wave window is expandable and collapsible. Click the “+” icon beside the /calc_qsim/arith/swdata signal name to view the individual signal waveforms in the Wave window.

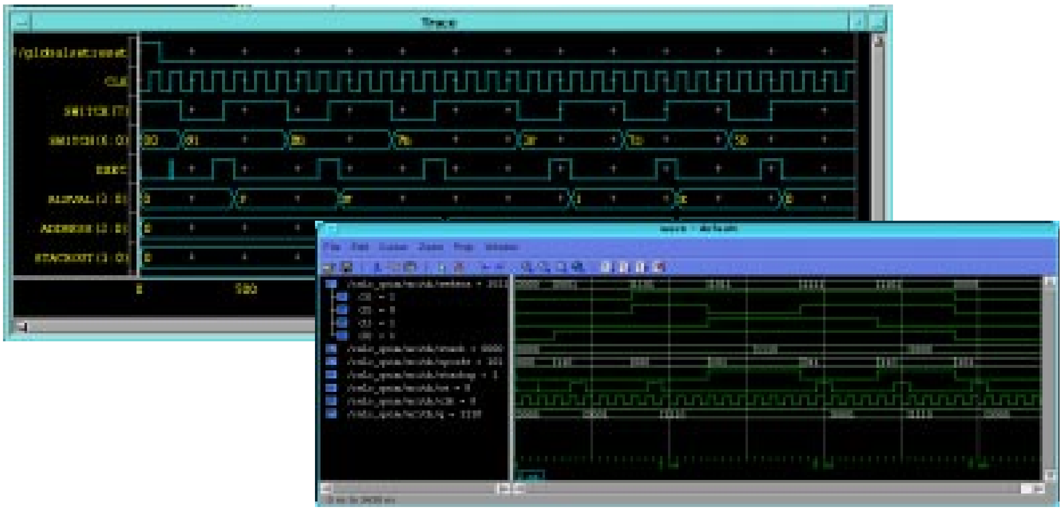


Figure 11-17 QuickSim Trace and ModelSim Wave Windows

6. The SWDATA waveform corresponds to the lower four bits of the SWITCH(6:0) bus on the top-level schematic. SWITCH(3:0) is connected to the SWDATA(3:0) port on the ALU in the top-level schematic.

   Verify that the SWDATA value in the Wave window corresponds to the lower four bits of SWITCH(6:0) in the QuickSim Trace window.

Note: You can make this task a easier by selecting the **`SWDATA signal`** in the Wave window, then choosing **`Prop`** → **`Radix: Hex`**. The hexadecimal value displayed in the Wave window now matches up with the lower-order hexadecimal digit from the SWITCH(6:0) value.

7. Close the QuickSim and ModelSim applications by choosing one of the two application windows and selecting **`Quit`** or **`Exit`** (depending on your workstation platform) from its control menu (the drop-down menu on the titlebar). It is not necessary to save any simulation results.

Note: Since both QuickSim and ModelSim are bound to the same process, closing one application automatically closes the other.

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 pld_men2edif dialog box appears.

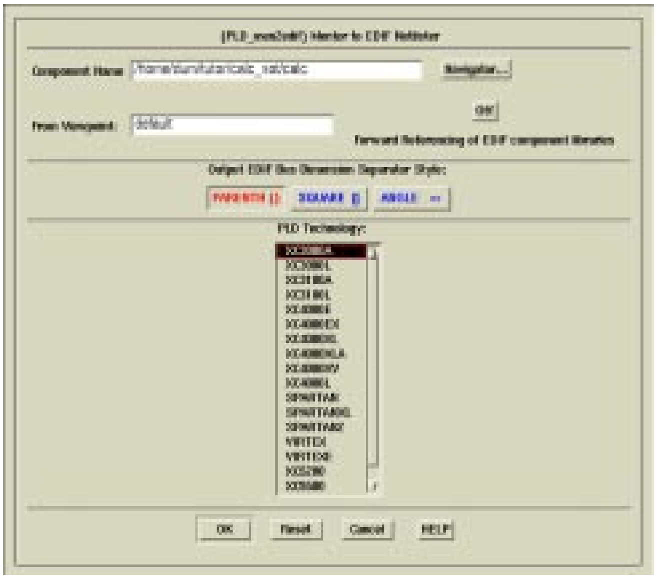


Figure 11-18 Pld_men2edif Dialog Box

Here is an explanation of some of the fields and buttons.

- **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 contains instantiated HDL components from other vendors, make sure that the bus delimeters written for symbols in the schematic are the same as those written in the synthesized HDL netlists; otherwise, NGDBUILD sees a pin mismatch and does not correctly connect buses that pass up or down through levels of hierarchy. Both of the synthesis tools used to generate the seg7dec.edif and alu.xnf files in this tutorial use angle brackets. If you are using these pre-synthesized files in your tutorial design, change this setting to ANGLE.
- **PLD Technology**—Select the architectural family from this list, in this case XC4000E.
- **HELP**—If the HELP button is clicked, a short help listing is produced by the pld_men2edif script.

To create an EDIF netlist for Calc, perform these steps:

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

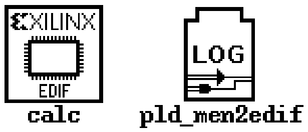


Figure 11-19 Files Created by Pld_men2edif

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

   A window appears displaying the log file. 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.

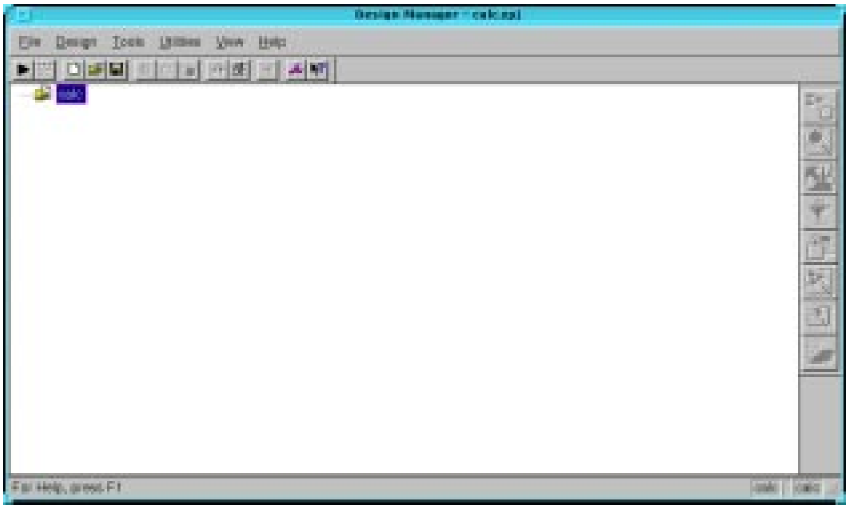

Design Manager - calc.prj
File Design Tools Utilities View Help
calc
For Help, press F1

Figure 11-20 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.`**

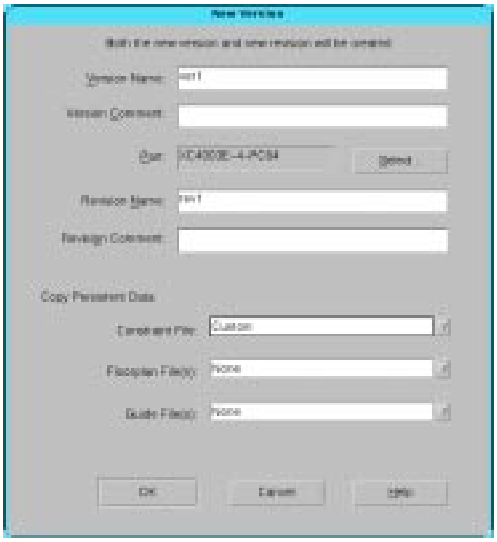
   In the current release of software, the Xilinx Design Manager does not read the part type from the design.

Note: The PART property in the CONFIG symbol does 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 Implement dialog box.

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 11-21.



Version Name: ver1

Part: XC4003E-4-PC84

Revision Name: rev1

Constraint File: Custom

Floorplan File(s): None

Guide File(s): None

Figure 11-21 New Version Dialogue Box

9. Click on OK. The Design Manager window should now look like the following Figure 11-22.

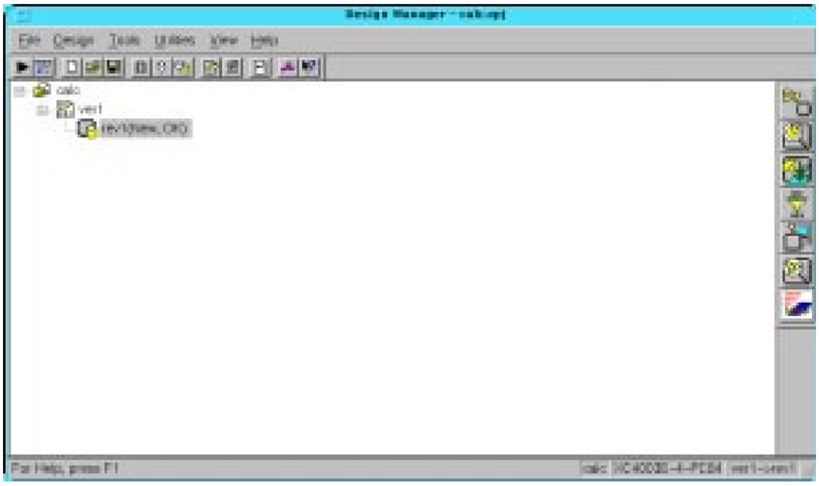
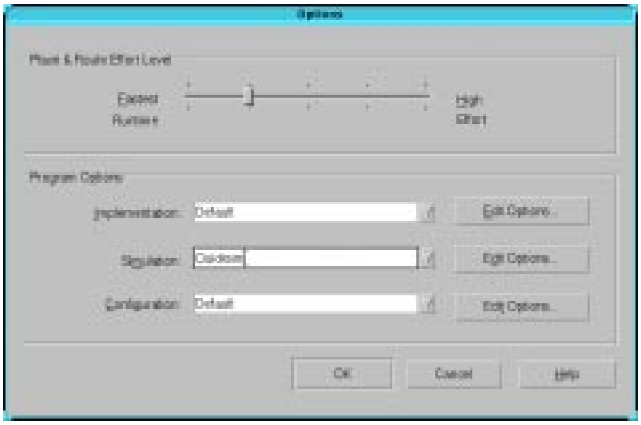


Figure 11-22 Design Manager with a New Version

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 11-23.



Program Options

Implementation: Default

Simulation: Quicksim

Configuration: Default

Figure 11-23 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):

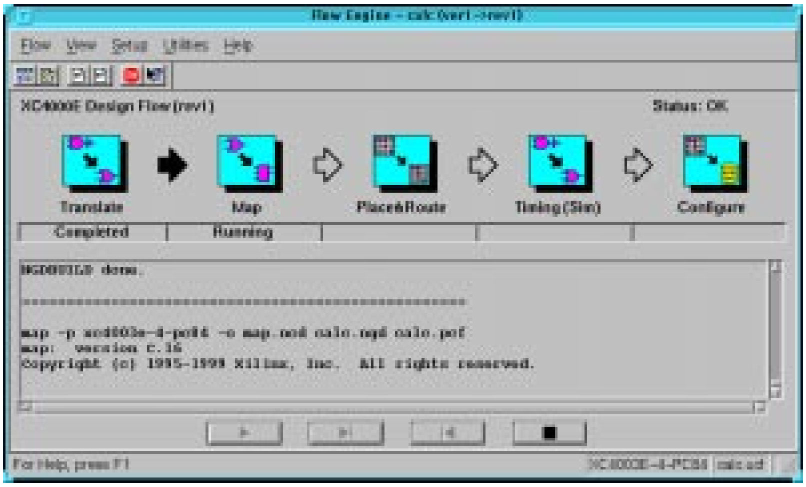
Produce Logic Level Timing Report—This generates a preliminary (post-map, pre-place and route) timing report based on the number of logic levels in each signal path. Since it is generated after the mapping step but before the place-and-route layout step, it does not contain information on device routing. 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 following figure.



```
NGDBUILD done.

==================================================

map -p xc4003e-4-pc84 -o map.ncd calc.ngd calc.pcf
map:  version c.16
Copyright (c) 1995-1999 Xilinx, Inc.  All rights reserved.
```

Figure 11-24 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 appears with:

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

14. Click on `View Logfile` to display the logfile from Flow Engine.

    The report is displayed in vi.

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

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.

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.

Note: All modules in a schematic-on-top design, including the HDL portions, are written as EDDM models. Since no HDL models are generated for timing simulation, only QuickSim is used to simulate the design, and neither ModelSim nor QuickSim Pro is required. You can however run the Xilinx tools with the Simulation option set to Modelsim VHDL or Modelsim Verilog, then run the timing simulation using ModelSim.

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. The 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.

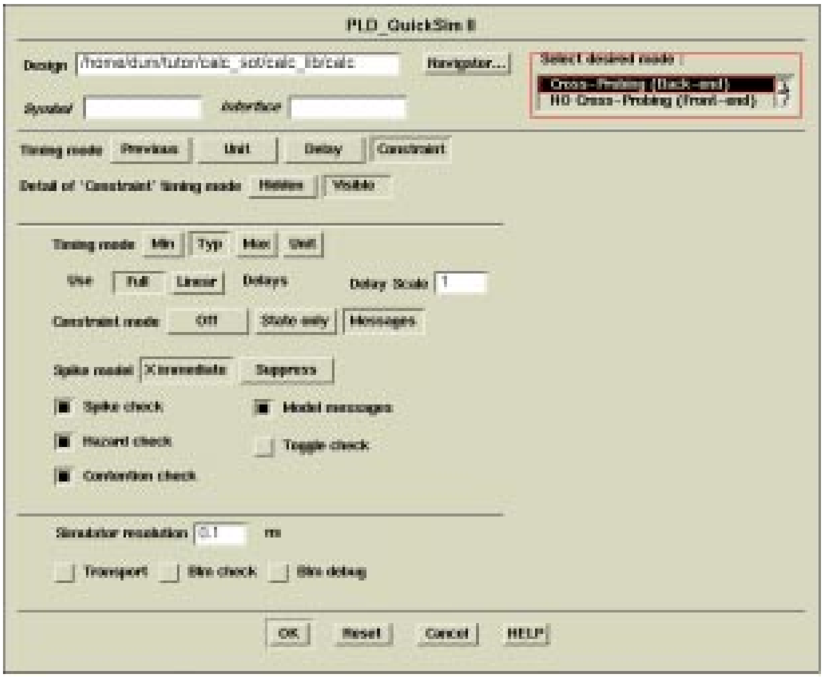
4. When you have finished looking at the file, close the Editor window.

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



Design: /home/dum/tutor/calc_sol/calc_lib/calc

Symbol:

Interface:

Select desired mode: Cross-Probing (Back-end) / NO Cross-Probing (Front-end)

Timing mode: Previous / Unit / Delay / Constraint

Detail of 'Constraint' timing mode: Hidden / Visible

Timing mode: Min / Typ / Max / Unit

Use: Full / Linear Delays

Delay Scale: 1

Constraint mode: Off / State only / Messages

Spike model: X immediate / Suppress

[✓] Spike check

[✓] Model messages

[✓] Hazard check

[ ] Toggle check

[✓] Contention check

Simulator resolution: 0.1 ns

[ ] Transport [ ] Bits check [ ] Bits debug

Figure 11-25 Invoking Pld_quicksim for Timing Simulation

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

   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 a message reading, "To start the cross-probing process, ..."

8. Click **OK** 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** from the palette.
12. In the dialog box that appears, enter the name of the *original* component in the Component field, *e.g.*, $XILINX_TUTORIAL / calc_sot / calc. (Do *not* enter the name of the simulation model, $XILINX_TUTORIAL / calc_sot / calc_lib / calc.)
13. Click OK to load the original viewpoint.
14. Select **OPEN SHEET** from the palette.

    The top-level Calc schematic appears.
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.timing.do** file.

    This replays a transcript file similar to the one played earlier, except that it does not open the schematic in QuickSim. 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.

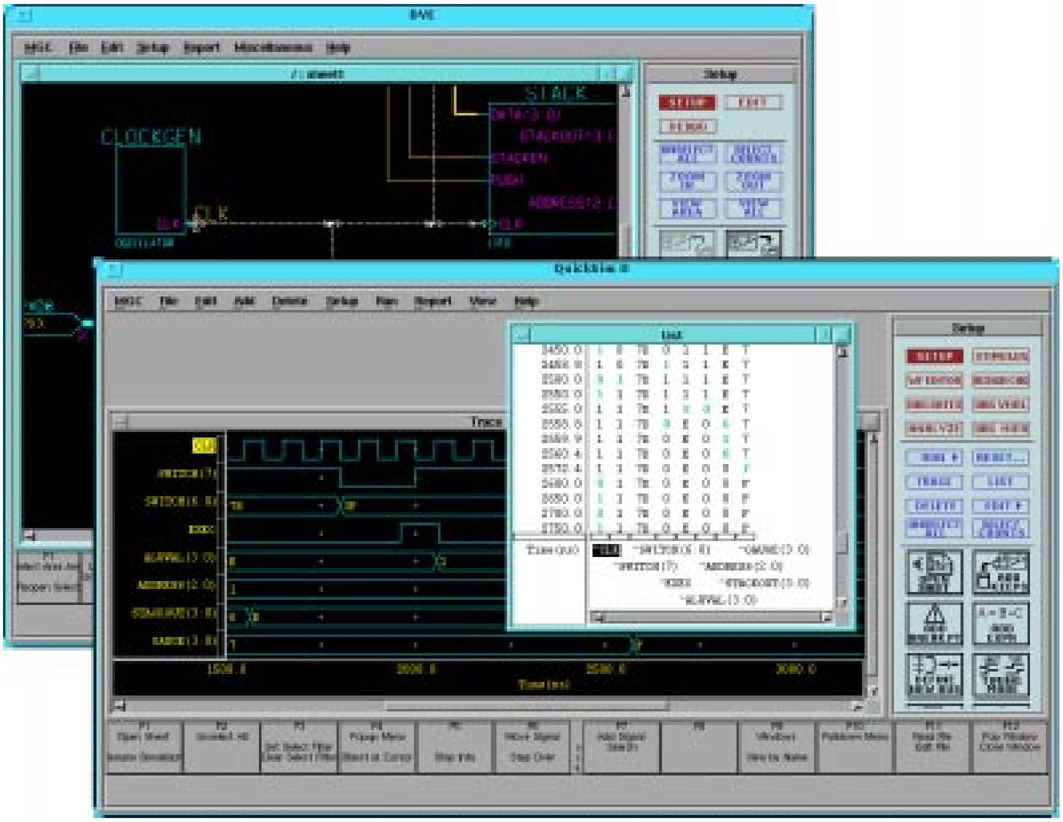


Figure 11-26 Cross-Probing with DVE and QuickSim

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 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

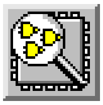


Figure 11-27 FPGA Editor Icon

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.

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.

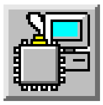


Figure 11-28 Hardware Debugger Icon

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.

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

```
vlib mywork
vmap work mywork
vcom –qspro_syminfo src/seg7dec.vhd
vcom –explicit –qspro_syminfo src/alu.vhd
pld_dve –s calc xc4000e
qspro calc –lib work –ini modelsim.ini
```

Basic Translation

```
[synthesize HDL modules]
pld_men2edif calc xc4000e –b a
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

```
pld_edif2tim calc.edn
pld_quicksim calc_lib/calc –cp –tim typ –consm
  messages
```

Further Reading

This Schematic-on-Top with VHDL Tutorial is intended 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, QuickSim, ModelSim, and QuickSim Pro 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, QuickSim, ModelSim, and QuickSim Pro documentation as well as the other chapters in this *Mentor Graphics Interface Guide*.