Platform Express User's Guide

Software Version 1.1f

Release 1.1f

November 2002

End-User License Agreement

Trademark Information

Table of Contents

Table of Contents (cont.)

Table of Contents (cont.)

Table of Contents (cont.)

Table of Contents (cont.)

List of Figures (cont.)

List of Figures (cont.)

List of Figures (cont.)

List of Tables (cont.)

List of Tables (cont.)

About this Manual

The Platform Express User’s Guide describes how to use Platform Express to quickly create platform-based designs based on open source or other third-party components. This manual is targeted to hardware and software engineers working in the Electronic Design Automation (EDA) industry. As such, this manual assumes a basic knowledge of System-On-Chip (SoC) embedded-processor electronic designs.

Using the Documentation

To accommodate differences in accessibility, viewing, and printing, the Platform Express User's Guide is delivered in two formats:

- Adobe PDF
- Sun JavaHelp

The technical content in each format is essentially the same. The two formats are provided as alternate delivery mechanisms of the same content.

Using Acrobat

The Adobe PDF format requires the Adobe Acrobat Reader application. The Adobe Acrobat Reader, which is available at no cost, is available at:

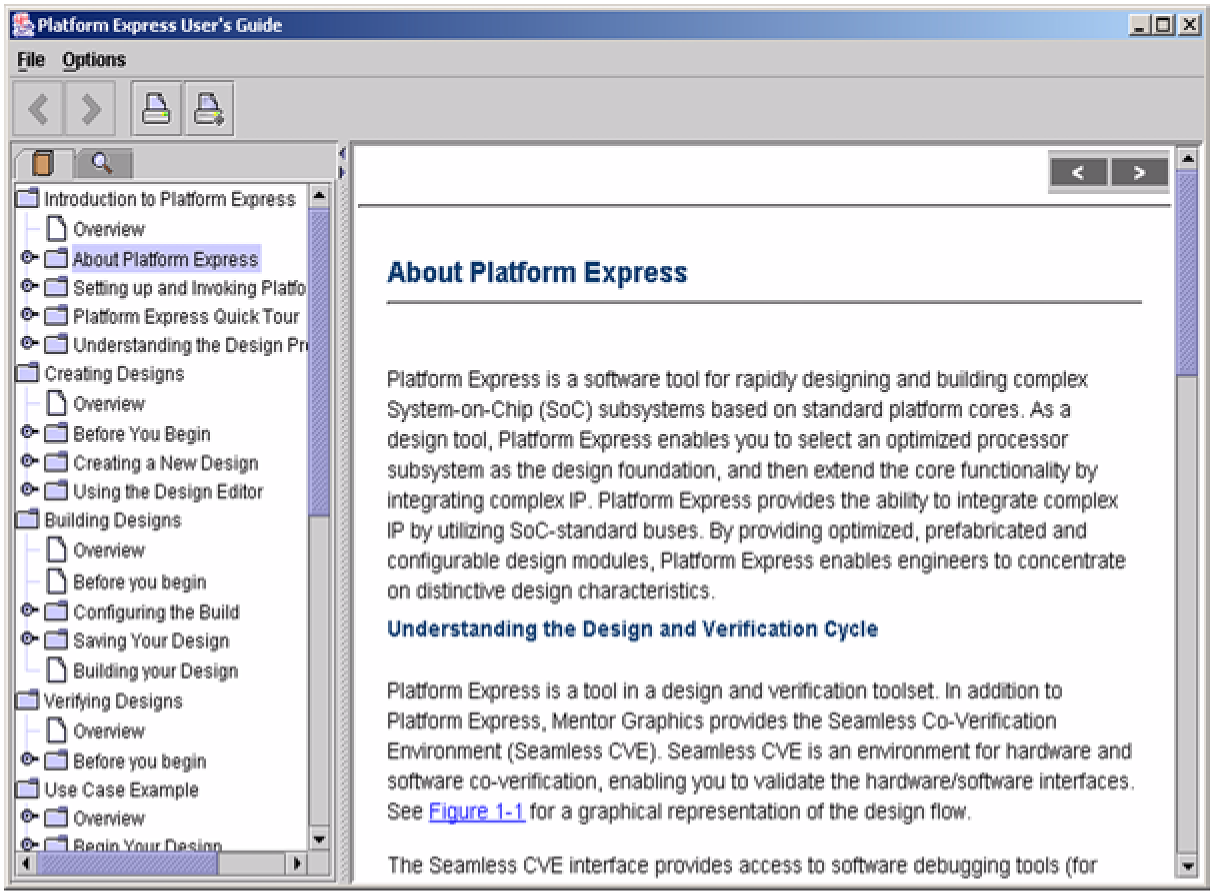
http://www.adobe.com/products/acrobat/readermain.html

The PDF version of the User’s Guide is available in the `/doc` directory of your Platform Express installation. The User’s Guide in PDF format is ideal for printing because it maintains the page paradigm.

Using JavaHelp

The User's Guide in JavaHelp is easily accessible from the Help menu of the Platform Express user interface. No third-party application is required. The Platform Express online help is ideal for accessing documentation directly from the Platform Express interface, and is customized for online viewing. For better printing results, use the PDF format instead of JavaHelp.

Select **Help > User's Guide** to invoke the JavaHelp viewer and display the first help topic. The viewer window contains two panes: the Content pane and the Navigation pane.



Introduction to Platform Express
Overview
About Platform Express
Setting up and Invoking Platfo
Platform Express Quick Tour
Understanding the Design Pr
Creating Designs
Overview
Before You Begin
Creating a New Design
Using the Design Editor
Building Designs
Overview
Before you begin
Configuring the Build
Saving Your Design
Building your Design
Verifying Designs
Overview
Before you begin
Use Case Example
Overview
Begin Your Design

About Platform Express

Platform Express is a software tool for rapidly designing and building complex System-on-Chip (SoC) subsystems based on standard platform cores. As a design tool, Platform Express enables you to select an optimized processor subsystem as the design foundation, and then extend the core functionality by integrating complex IP. Platform Express provides the ability to integrate complex IP by utilizing SoC-standard buses. By providing optimized, prefabricated and configurable design modules, Platform Express enables engineers to concentrate on distinctive design characteristics.

Understanding the Design and Verification Cycle

Platform Express is a tool in a design and verification toolset. In addition to Platform Express, Mentor Graphics provides the Seamless Co-Verification Environment (Seamless CVE). Seamless CVE is an environment for hardware and software co-verification, enabling you to validate the hardware/software interfaces. See Figure 1-1 for a graphical representation of the design flow.

The Seamless CVE interface provides access to software debugging tools (for

Figure 1. JavaHelp viewer interface

- Content Pane

  Contains topic text. Within the topic text, there may be blue, underlined links that you can click to display related topics.

- Navigation Pane

  Provides two tabs: Contents and Search; and four buttons: Back, Next, and two Print buttons.

  - Contents Tab

Displays the Table of Contents. In the table of contents, click on a topic title to display that topic's contents in the Content pane. Double-click a folder to display the topic tree.

- Search Tab

  Displays the Search tab and invokes the global search tool. Search through the entire User's Guide for an item by entering text in the Find box and pressing Return. The viewer displays titles of all topics containing the item in the Navigation pane. Click on a topic title to display the topic.

- Previous and Next buttons

  Cycle forwards and backwards through previously displayed topics in the order in which they were displayed.

- Print buttons: Print and Print Setup

  The left **Print** button displays a print dialog box for configuring and initiating printing; the right **Print Setup** button sets the page format.

Typographical Conventions

Convention	**Meaning**	**Example**
italic text	Indicates a variable or user-supplied argument.	`-f` *filename*
square brackets [ ]	Encloses optional commands, parameters, or arguments.	`filename [`*list*`]`
menu > menu choice > sub-menu choice	Shows the top-level menu on the main menu, to a menu choice, to a sub-menu choice.	**File > Save**
Bold	Indicates selectable items within the interface, such as menus.	In the **File** menu, select **New**.

Note

Screen captures in the *Platform Express User's Guide* reflect the Solaris platform with the Common Desktop Environment (CDE) windowing system. The actual appearance of GUI elements on other platforms may vary.

Manual Organization

The Platform Express User's Guide contains the following topics:

- Introduction to Platform Express
- Creating Designs
- Building Designs
- Verifying Designs
- Use Case Example
- Directory Structures
- Frequently Asked Questions (FAQs)
- Build Log File
- Using the Documentation Generator
- Contacting the SupportCenter

Related Publications

The following Mentor Graphics publications contain information relevant to the usage of Platform Express:

- The *Platform Express Installation Manual* explains how to install Platform Express and how to set up your environment.

- The *Platform Express Release Notes* contain changes and enhancements, as well as caveats specific to Platform Express or its operating environment.
- The *Platform Express Integrator's Guide* is targeted specifically to IP providers for the express purpose of integrating custom core components into the Platform Express interface. End-users creating designs with the provided third-party and open source components do not need the *Integrator's Guide*.
- The *Seamless CVE User's and Reference Manual* contains an overview of the product, explains how to prepare designs for cosimulation, and provides procedures for workstation and design configuration and workstation configuration.

About the Platform Express User's Guide

Product Version: 1.1f

Release Date: November 2002

Chapter 1 Introduction to Platform Express

Overview

This chapter contains conceptual and high-level procedural information on Platform Express. It contains the following sections:

- About Platform Express
- Setting up and Invoking Platform Express
- Platform Express Quick Tour
- Understanding the Design Process

About Platform Express

Platform Express is a software tool for rapidly designing and building complex System-on-Chip (SoC) subsystems based on standard platform cores. As a design tool, Platform Express enables you to select an optimized processor subsystem as the design foundation, and then extend the core functionality by integrating complex IP. Platform Express provides the ability to integrate complex IP by utilizing SoC-standard buses. By providing optimized, prefabricated and configurable design modules, Platform Express enables engineers to concentrate on distinctive design characteristics.

Understanding the Design and Verification Cycle

Platform Express is a tool in a design and verification toolset. In addition to Platform Express, Mentor Graphics provides the Seamless Co-Verification

Environment (Seamless CVE). Seamless CVE is an environment for hardware and software co-verification, enabling you to validate the hardware/software interfaces. See Figure 1-1 for a graphical representation of the design flow.

The Seamless CVE interface provides access to software debugging tools (for example, XRAY Debugger and Green Hills MULTI), as well as to hardware logic simulation tools (for example, ModelSim and NCSim). Platform Express provides the essential functionality for designing and building complex SoC subsystems, and Seamless CVE provides the interface for design verification.

Automating the Design Cycle

Platform Express assists the system designer with tools that automate the design cycle. All relevant design elements are presented in detail within the Platform Express interface, accelerating design creation. The Design Editor is context-aware and allows on-the-fly configuration. After the design is created, Platform Express provides tools for automating the build of the design database and tool configuration files. The build files generated by Platform Express are used directly in concert with the Seamless Co-Verification environment. Platform Express provides the following automatic functionality:

- Bus decoding
- Bus bridging
- Interrupt bridging

Designing with Various Platform Cores

The Platform Express interface provides the flexibility of offering a selection of platform cores (for example, ARM966 and ARM926) to use as the basis of your design. The default Platform Express application ships with a number of libraries (which contain platform core components), and alternate third-party libraries are also available. Many platform cores are provided as “open source,” while others are proprietary. (Proprietary components are shipped as simulation models and do not provide sources). For information on creating and implementing libraries, consult the *Platform Express Integrator’s Guide*.

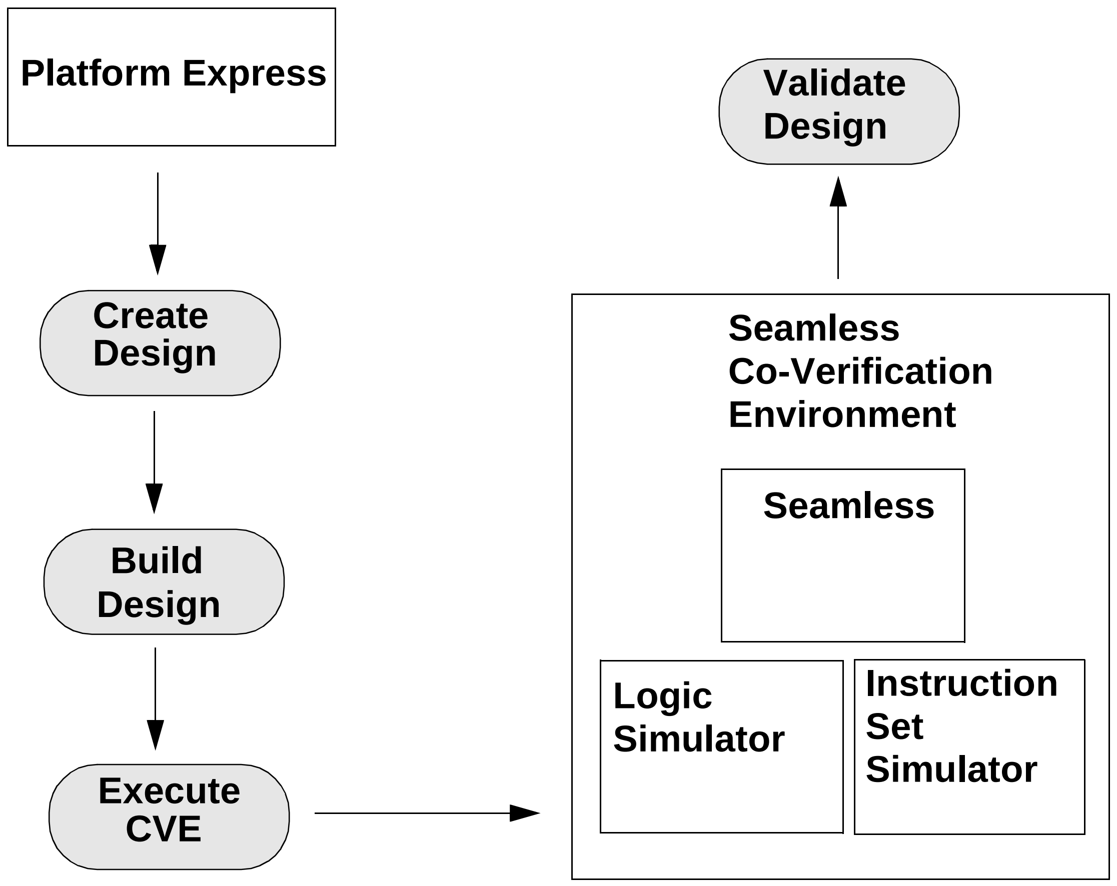


Figure 1-1. Design and Verification Flow

Setting up and Invoking Platform Express

Licensing

Platform Express utilizes the FlexLM License Manager software. Refer to the *Platform Express Installation Manual* for licensing information.

Defining Environment Variables

Platform Express is a Java SDK application. In addition to `JAVAHOME`, Platform Express utilizes environment variables to function properly. The required `PXHOME` environment variable points to the root directory of the Platform Express installation. Immediately after installing Platform Express, you must manually set the `PXHOME` environment variable.

Other optional environment variables which are relevant to Platform Express include:

- `JAVAHOME`

  Points to your JRE or Java SDK installation.

- `PXPATH`

  Points to your component libraries.

- `CVE_HOME`

  Points to your Seamless CVE installation.

- `MODELTECH`

  Points to your ModelSim installation.

- `ARMTOOLS`

  Points to your ARM build tools, such as the ARM compiler.

For further information on defining your environment to run Platform Express, refer to the *Platform Express Installation Manual*.

Note Platform Express is validated against specific versions of Seamless CVE. Consult the *Platform Express Release Notes* to ensure you have installed a version of Seamless CVE that has been certified to work with Platform Express.

Invoking Platform Express

To invoke Platform Express, enter the following:

```
$PXHOME/bin/px
```

For enhanced control of the Platform Express invocation, additional command-line switches are available:

```
$PXHOME/bin/px [-d] [-v] [-l logfilename] [-i] [-G
generatorchain] [[-b|-r] designname]
```

See Table 1-1 for a listing and description of the invocation switches.

.

Table 1-1. Invocation Switches

Switch	Description
`-d`	Enables debug message output. Use the -l option to specify output to a log file. Otherwise, the messages appear in the output pane.
`-v`	Enables additional (verbose) debug messages, including the names of generators as they execute during the build. Use the -l option to specify output to a log file. Otherwise, the messages appear in the output pane.
`-l`	Creates a log file. A default log file is created (or overwritten) unless a logfilename is specified. The default log file is located in your home directory in `.pxrc/pxlogfile`.
`logfilename`	Specifies the filename and optional path for debug messages.
`-i`	Ignores generator errors and continues build. If not used, the build stops if an error occurs.

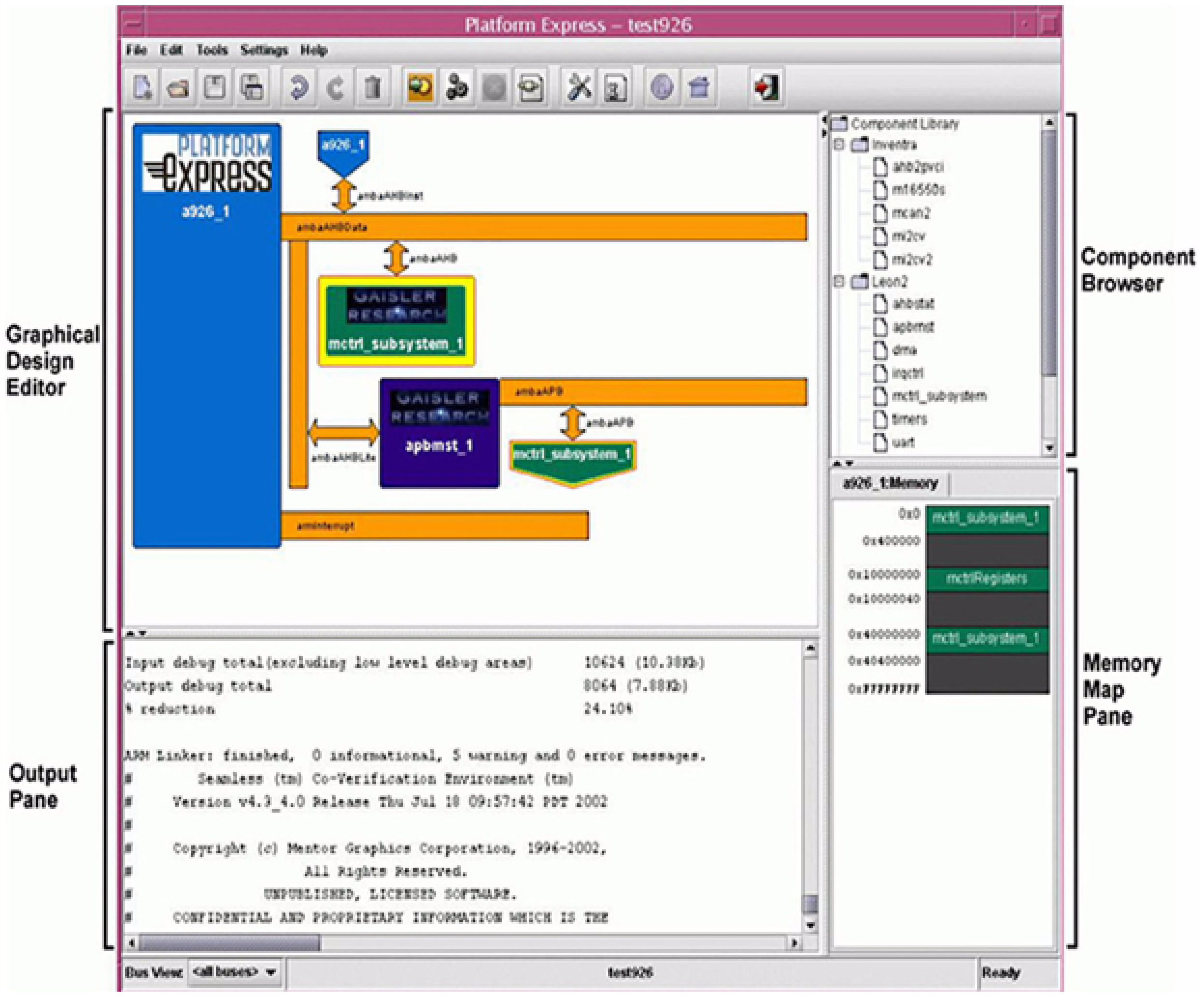
Table 1-1. Invocation Switches

Switch	Description
`-G`	Invokes Platform Express in batch mode. All required drivers must be specified in the design before using this option.
`generatorchain`	One of `Build` or `Execute`.
`-b`	Automatically builds `designname`.
`-r`	Automatically executes a co-verification session on `designname`.
`designname`	The root directory of an existing Platform Express design.

Platform Express Quick Tour

The following is an overview of the Platform Express Graphical User Interface (GUI). In addition to the standard menu bar, tool bar, and status bar of modern application interfaces, the Platform Express interface is composed of four basic elements:

- Graphical Design Editor
- Component Browser
- Memory Map Pane
- Output Pane



```
Input debug total(excluding low level debug areas)      10624 (10.38Kb)
Output debug total                                      8064 (7.88Kb)
% reduction                                             24.10%

ARM Linker: finished,  0 informational, 5 warning and 0 error messages.
#          Seamless (tm) Co-Verification Environment (tm)
#       Version v4.3_4.0 Release Thu Jul 18 09:57:42 PDT 2002
#
#       Copyright (c) Mentor Graphics Corporation, 1996-2002,
#                        All Rights Reserved.
#                    UNPUBLISHED, LICENSED SOFTWARE.
#       CONFIDENTIAL AND PROPRIETARY INFORMATION WHICH IS THE
```

Figure 1-2. Platform Express Interface

Graphical Design Editor

The Graphical Design Editor enables you to create a block diagram of your design. The components in the design editor are configurable. To configure the components in the design editor, right-click on a component and select the desired configuration.

Component Browser

The component browser lists all the available platform cores. (The available platform cores vary depending on your installed libraries.) Initially, only core platforms are visible. Additional components (such as memory, bus bridges, interrupt controllers, and peripherals) become visible *after* you have selected a core platform and placed it in the Design Editor.

Memory Map Pane

Memory and other peripheral components automatically connect to the appropriate bus on the platform. The memory map pane is a Read-Only pane that displays where a given component fits into the platform address space in relation to other components. The memory map pane displays address locations in hexadecimal format.

Output Pane

The output pane functions as a standard output display. Build statuses are automatically displayed in the output pane.

Note The contents of the output pane do not clear automatically when you close the current design.

Clearing the Contents of the Output Pane

1. Right-click within the output pane.
2. Select **Clear** from the context menu.

   You cannot clear a selection of the output pane. The Clear function deletes the entire contents.

Copying the Contents of the Output Pane

1. Select the desired contents in the output pane.
2. Right-click within the output pane.
3. Select **Copy** from the context menu.
4. Paste the contents into a text editor or shell.

Additional User Interface Elements

In addition to the graphical design editor, component browser, output pane, and memory map pane, there are several common elements in the Platform Express interface:

- Menu Bar
- Button Bar
- Status Bar

Menu Bar

The main menu bar is used for creating and saving designs, configuring tool settings, and building and executing finished designs.

File Menu

Table 1-2. File Menu

Menu Item	**Description**
New	Closes current design and creates a new design using your default project settings.
Open	Opens an existing design (`*.plx`).
Save	Saves the current design.

Save As	Allows you to save the current design with a new name.
Exit	Exits Platform Express.

Edit Menu

Table 1-3. Edit Menu

Menu Item	Description
Undo	Undoes the last change.
Redo	Undoes the last Undo.
Delete	Deletes the selected component.

Tools Menu

Table 1-4. Tools Menu

Menu Item	Description
Build	Builds the current design, using the specified verification tools.
Execute	Simulates the current design; executes Seamless CVE.
Stop	Stops the current build.

Note

Depending on your installed components, you may see additional items on the Tools menu. For example, the Platform Express Documentation Generator (which is an add-on component) is accessible from the Tools menu. See Appendix D, Using the Documentation Generator, for more information.

Settings Menu

Table 1-5. Settings Menu

Menu Item	**Description**
User Preferences	Configures Platform Express environment settings; does not affect the settings of the current design.
Project Settings	Configures design-related settings for the current design.
Save As Default	Saves the current design settings as the default for all your new designs.
Log Messages	Configures the level of log message output. The three choices are: • None: outputs no messages • Info: sets logmask to “8” • Debug: sets logmask to “4” The logmask is a filter setting that defines which types of messages are displayed. The “8” logmask displays informational messages only. The “4” logmask displays the most in-depth messages, useful for debugging.
Configure All	Configures all design-related settings for the current design. This dialog box enables you to configure *all* settings of your design from a single location. The three top-level settings are: • design • component • bus

Help Menu

Table 1-6. Help Menu

Menu Item	Description
User’s Guide	Invokes JavaHelp and loads the User's Guide.
About	Displays Platform Express version numbers and legal information.
Px Web	Invokes a browser and loads the Platform Express website.

Button Bar

Contains buttons for managing designs and configuring Platform Express.

Status Bar

The status bar has three active areas, from left to right:

- o Bus View menu, which allows the Design Editor to show all buses or only one bus type.
- o Displays the current design name.
- o Displays one of the following states:
  - Ready
  - Building
  - Executing

Note The states displayed in the Status Bar are dependent on the generation tool. As the generation tool may vary, the values displayed in the Status bar that reflect the state may vary.

Size-Configurable Panes

The Platform Express interface is configurable. Resize the Design Editor, Component Library, Memory Map, and Output panes by dragging the dotted bars between the panes:

Minimize or restore panes by clicking the arrows in the bar next to the pane(s):

Figure 1-3 depicts the Platform Express interface with the Design Editor expanded (and the Component Library, Memory Map, and Output panes minimized).

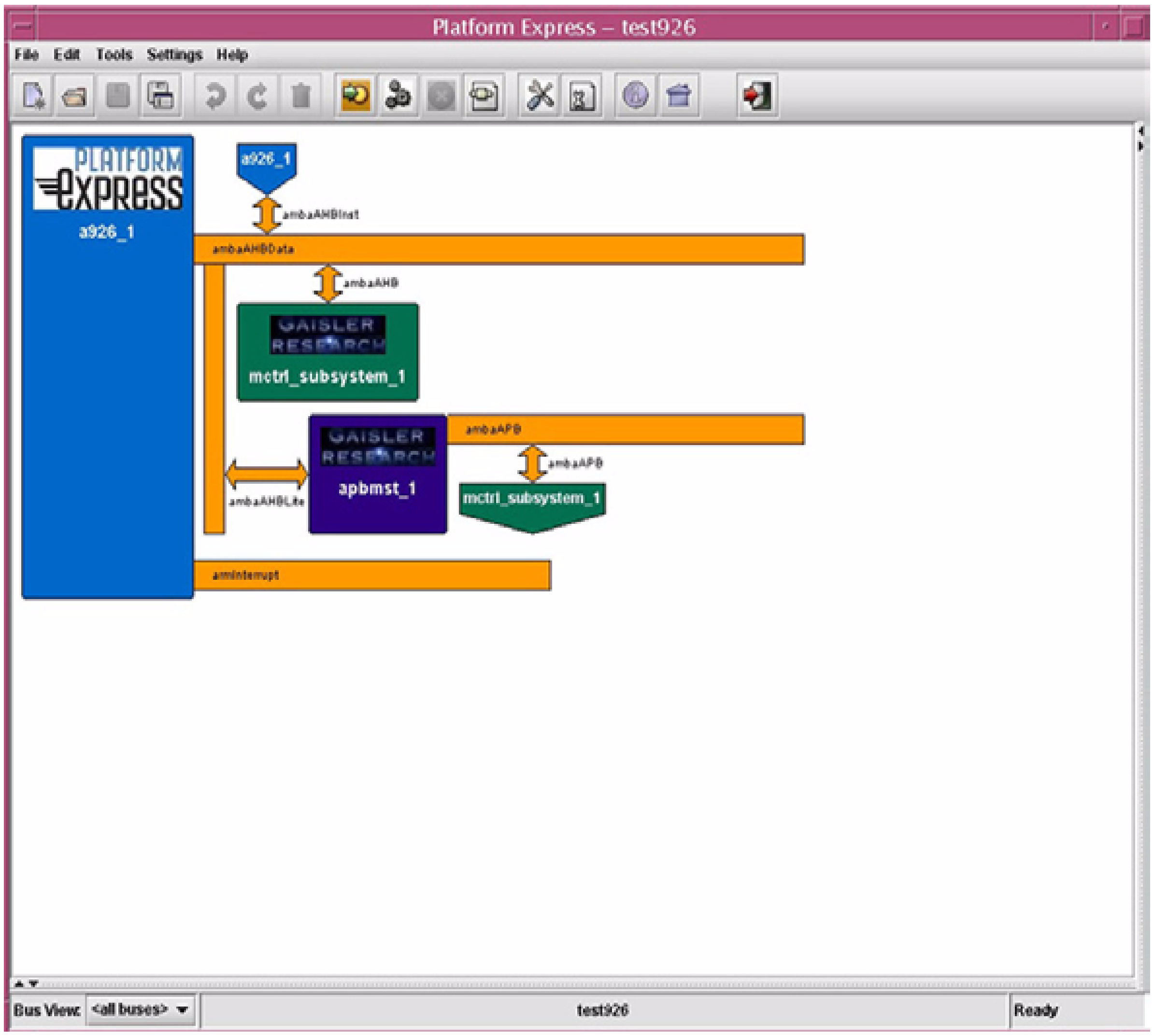


Figure 1-3. Platform Express, with Maximized Design Editor

Understanding the Design Process

The platform design cycle consists of three main processes:

1. Creating the Design
2. Building the Design
3. Executing the Co-Verification Environment

Step One: Create the Design

Platform Express provides a set of platform cores and complex IP as building blocks to create your design. The Component Browser lists your licensed platform cores. After you have selected the desired core platform and moved it to the graphical design editor, the Component Browser automatically lists all the associated memories and peripherals.

1. Select the platform core on which to base your design.
2. Double-click or drag-and-drop the desired platform core into the design editor.
3. Specify addresses and other information in the core's configuration dialog box.

Once the platform is in the design editor, you may begin adding memory and peripheral components to the desired buses. Most components have dialog boxes that allow you to specify addresses and other configuration information.

Step Two: Build the Design

Platform Express generates an HDL system design based on the contents of the Design Editor and the component configurations. To initiate the build process, select **Tools** > **Build** or click the **Build** button on the toolbar. The generated design files include most of the basic requirements of an embedded system design, such as component instantiation and connection within a top-level design, decode logic, and interrupt setup.

Step Three: Invoke the Co-Verification Environment

Platform Express provides an interface for executing a co-verification environment that invokes Seamless CVE. All the required scripts and configuration files, which are necessary for running the co-verification session, are automatically generated by Platform Express. Use the co-verification environment to verify the design you created and built with Platform Express.

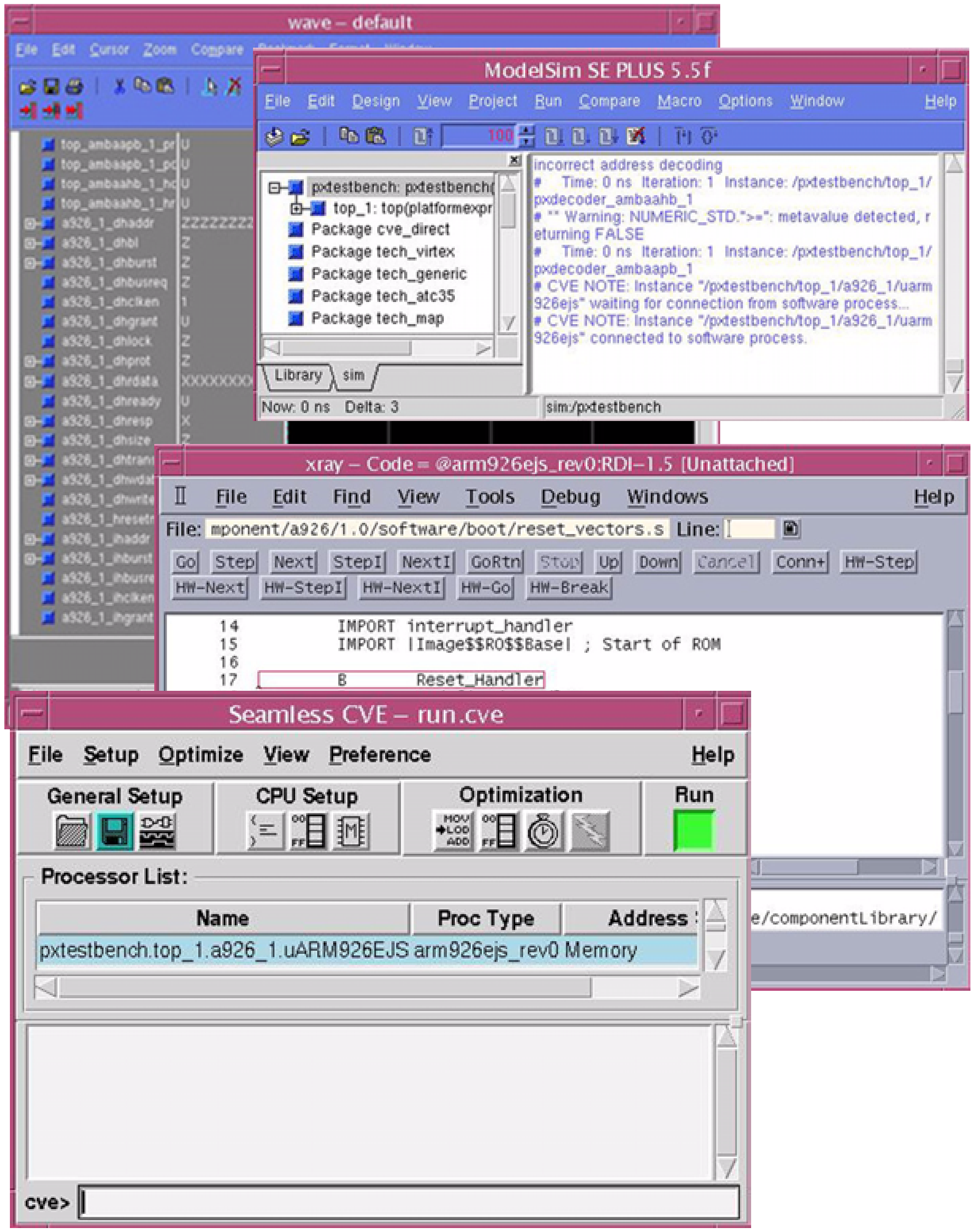
To execute the co-verification session, select **Tools > Execute** or click the **Execute** button on the toolbar. When you invoke a co-verification session, the following applications are automatically loaded:

- Seamless CVE
- Logic simulator interface (for example, ModelSim)
- Wave viewer

In addition, log windows also appear. Depending on your environment, you may have to issue additional commands to invoke the Instruction Set Simulator (ISS). For example, to invoke the XRAY Debugger ISS, enter the following command at the ModelSim command prompt:

```
run -all
```

Note Platform Express relies on environmental settings to invoke Seamless CVE. Ensure that you have set all the required environment variables to properly execute a co-verification session. For more detailed information, consult the *Platform Express Installation Manual*.



```
incorrect address decoding
#    Time: 0 ns  Iteration: 1  Instance: /pxtestbench/top_1/
pxdecoder_ambaahb_1
# ** Warning: NUMERIC_STD.">=": metavalue detected, r
eturning FALSE
#    Time: 0 ns  Iteration: 1  Instance: /pxtestbench/top_1/
pxdecoder_ambaapb_1
# CVE NOTE: Instance "/pxtestbench/top_1/a926_1/uarm
926ejs" waiting for connection from software process...
# CVE NOTE: Instance "/pxtestbench/top_1/a926_1/uarm
926ejs" connected to software process.
```

```
        IMPORT  interrupt_handler
        IMPORT  |Image$$RO$$Base| ; Start of ROM

        B       Reset_Handler
```

Name	Proc Type	Address
pxtestbench.top_1.a926_1.uARM926EJS	arm926ejs_rev0	Memory

Figure 1-4. Seamless Co-Verification Environment

Chapter 2 Creating Designs

Overview

This chapter contains procedural and conceptual information on creating a design with Platform Express. It contains the following sections:

Before You Begin

This chapter assumes you have properly installed Platform Express (including libraries), and have configured your environment to run Seamless CVE. Check that you have defined the following environement variables: `PXHOME`, `PXPATH`, `CVE_HOME`, `MODELTECH` (for ModelSim logic simulation), and `ARMTOOLS` (if you are using ARM tools). Consult the *Platform Express Installation Manual* for complete instructions.

Defining User Preferences

Before creating your first design, check your user preferences. User preference settings affect the operation of Platform Express. (All of your designs use the same user preferences.)

To configure user preferences, select **Settings** > **User Preferences**:

- Browser Command

  The Browser Command is the system command for invoking an Internet browser (such as Netscape), directly from within the Platform Express interface. (To invoke the browser, right-click a component in the Design Editor, select Browse, and then select a menu choice.) The direct browser access enables you to view information and documentation provided by third-party core vendors.

  To define a browser, enter the full path to the browser and all command parameters, including an embedded URL tag (`$url`). The URLs from the component configuration files will be inserted when the browser is invoked.

Defining Project Settings

Before creating your first design, check your project settings. When you invoke Platform Express, it creates a directory named `.pxrc` in your home directory (if `.pxrc` does not already exist). Your default project settings are stored in the `.pxrc` directory. Platform Express applies your default project settings to all new projects. (The words *project* and *design* are used interchangeably.)

Note

A warning appears if the .pxrc directory does not exist:

```
[WARN]    - Error opening settings file
"/user/michael/.pxrc/preferences.xml"
/user/michael/.pxrc/preferences.xml (No such
file or directory)
[WARN]    - Error opening settings file
"/user/michael/.pxrc/projectSettings.xml"
/user/michael/.pxrc/projectSettings.xml (No such
file or directory)
```

Changing Current Project Settings

1. Select **Settings** > **Project Settings**.
2. In the Edit Properties dialog box, enter the desired settings.
3. Click OK.

The new project settings are stored in the project directory when you save your design.

Changing Default Project Settings

1. Select **Settings** > **Project Settings**.
2. In the Edit Properties dialog box, enter the desired settings.
3. Click OK.
4. Select **Settings > Save As Default**.

The design settings are listed in Table 2-1.

Table 2-1. Project Settings

Setting Name	Description	Default
HDL Top Module Name	Name of HDL top module.	`top`
HDL Language	One of: `VHDL`, `Verilog`.	`VHDL`
Clock Period	Period, in nanoseconds, for one-shot or clock signals that require drivers.	`40 ns`
Clock Pulse Offset	Pulse offset, in nanoseconds, for clock signals that require drivers.	`10 ns`
Clock Pulse Value	Pulse value for clock signals that require drivers.	`0`
Clock Pulse Duration	Pulse duration, in nanoseconds, for clock signals that require drivers.	`20 ns`
Reset Offset	In nanoseconds.	`40 ns`
Reset Assertion Value	Reset assertion value.	`0`
Current Verification Environment	Used by **Tools** > **Build**. `ModelSim` or `NCSim`.	`ModelSim`

Creating a New Design

To create a design with Platform Express, you use three areas of the interface:

- Graphical Design Editor

  Create, view, and save your design, starting from a new or existing design.

- Component Browser

  Select the components for your design.

- Memory Map Pane

  Displays how the components map into your design’s memory space. The memory map refreshes when components are added or component configurations are modified. The memory map is “Read-Only”; you cannot edit it directly.

Note

If a design is already open, select **File** > **New** or click the **New** button. This step clears the Design Editor and allows you to create a new design. Platform Express applies the default design settings to all new designs.

1. Browse the available core platforms from the list of components in the component browser.
2. Double-click the desired core or click-and-drag the component to the design editor.

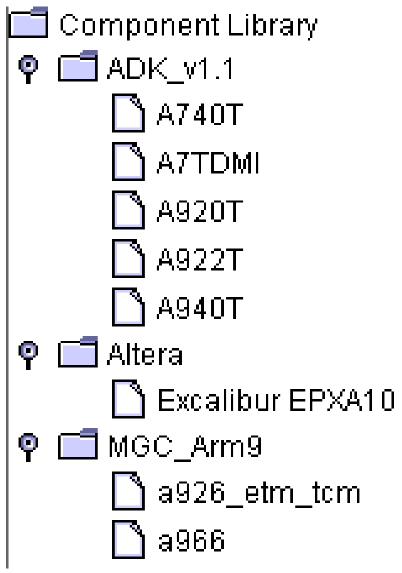


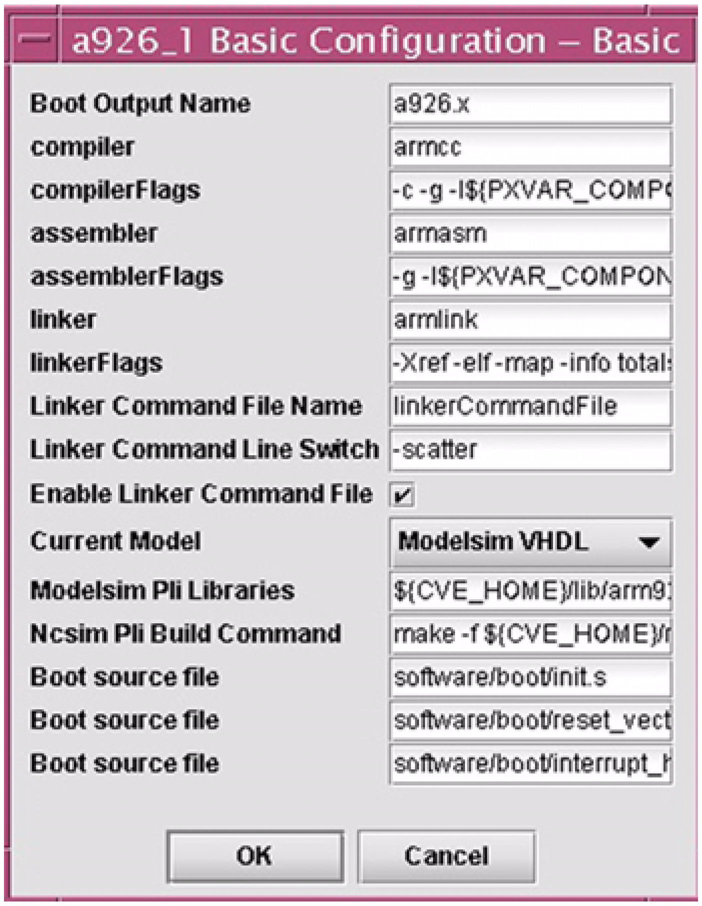
Figure 2-1. Component Browser

3. Enter your selections in the provided core configuration dialog box.

   See Figure 2-2 for a sample configuration dialog box based on the ARM926 core.

4. Click **OK**.

Note The fields in the core configuration dialog boxes vary depending on the selected core.



a926_1 Basic Configuration – Basic

Boot Output Name	a926.x
compiler	armcc
compilerFlags	-c -g -I${PXVAR_COMP
assembler	armasm
assemblerFlags	-g -I${PXVAR_COMPON
linker	armlink
linkerFlags	-Xref -elf -map -info total
Linker Command File Name	linkerCommandFile
Linker Command Line Switch	-scatter
Enable Linker Command File	✓
Current Model	Modelsim VHDL
Modelsim Pli Libraries	${CVE_HOME}/lib/arm9
Ncsim Pli Build Command	make -f ${CVE_HOME}/
Boot source file	software/boot/init.s
Boot source file	software/boot/reset_vect
Boot source file	software/boot/interrupt_

Figure 2-2. Example Core Configuration Dialog Box

When the dialog box closes, the core appears in the design editor.

5. Add memory and peripheral components to the buses.

   Drag components from the component browser into the Design Editor. As you drag the component over the buses, the cursor changes to indicate a valid or invalid connection.

Valid connection:	↖
Invalid connection:	⊘

   If you drop the component while a bus is highlighted, the component attaches to that bus. If you drop a component while no bus is highlighted, Platform Express makes the required connections by default.

1. If prompted with a dialog box, specify the component’s Base Address or other configuration settings and click OK.

   When you close the dialog box, the component automatically connects to the selected bus on the platform. Figure 2-3 shows a simple design in the Design Editor.

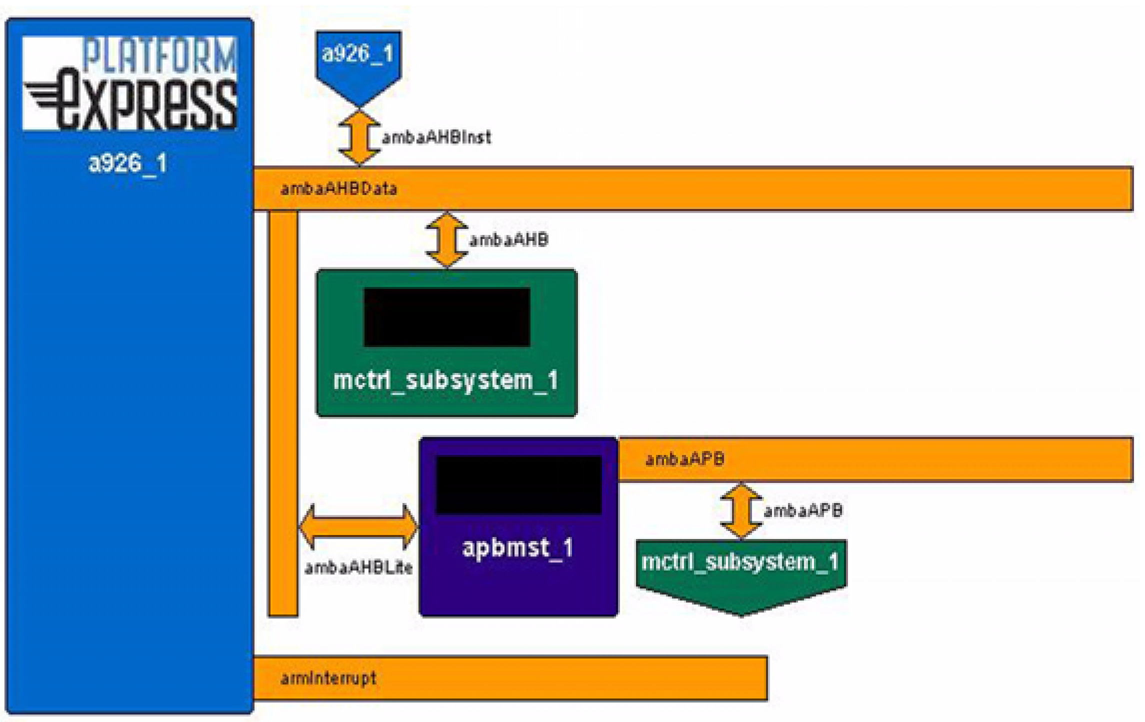


Figure 2-3. Simple Design with core and peripherals

Opening an Existing Design

1. Select **File** > **Open**.

2. Navigate to the desired design (a file with the `*.plx` extension).

   You can also open a design from a shell by invoking Platform Express with a specified design (`*.plx`). For more information on invocation switches, refer to Table 1-1.

Using the Design Editor

The Design Editor pane is the working space for creating and manipulating your design. You can add and configure components and buses in the Design Editor. Figure 2-3 shows a basic design in the Design Editor. After you have placed components into the Design Editor, you can right-click on the components and buses for further configuration.

The Design Editor enables you to build a design with the following building blocks:

- Components
- Connectors
- Buses

The Design Editor does not allow:

- Repositioning of components, except when moving them to a new bus
- Re-routing of connections

Working with Componets

The Platform Express Component Browser displays a list of components containing platform cores, memories, and peripherals. (Refer to the Component Browser to browse the available cores for your design.) The components in the Component Browser are essentially the building blocks of your design. These components have bus interfaces. The bus interfaces connect to buses. The connections to buses are represented by connectors.

When you begin a design, only platform cores are visible in the Component Browser. Once a platform has been placed in the Design Editor, additional components appear in the Component Browser, and you can begin adding memory and peripheral components and buses. Only those components that can be connected to the currently available buses appear in the Component Browser. Components can be connected directly or through bus bridges.

Besides adding components, you may need to configure components, including master and slave components, or add bus bridges to adapt the components to the buses in the design.

Color Coding for Components

Color-coding of the cores, memories and peripherals is provided as a means for distinguishing them. Platform cores are blue. Any component that provides a bus bridge is displayed in one of eight shades of purple. All other components are assigned one of eight shades of green. Buses and connectors in the Design Editor are displayed in yellow.

Creating Component Instances

Note The available platform cores listed in the Component Browser vary depending on your installed libraries. If you do not see components which you believe are installed, check your `PXPATH` environment variable.

Insert a component into the Design Editor in one of the following ways:

- Double-click the component name in the Component Browser.
- Right-click on the component name in the Component Browser and select **Add** from the context menu.
- Select the component and drag it to the Design Editor.

Memory and peripheral components automatically connect to the appropriate bus on the core. All other connections designated as required in the component's definition file are made, such as clock and interrupt connections. If the component has connections that are not required, those connections are not made automatically.

The memory map is automatically updated when you add a component to a core. It shows where the component fits into the core address space in relation to the other components.

Configuring Components

Most components have dialog boxes that appear automatically when you place a component in the Design Editor. The configuration dialog box allows you to specify addresses and other parameters. These parameters are described and defined in the component data sheets available from the component vendor or on the component vendor's website. (To go to a component vendor's website, right-click on the component and select an entry under the Browse menu.)

1. In the Design Editor, select the desired component.
2. Right-click on the component and select **Configure** > **Basics** from the context menu.
3. Enter the desired parameters, based on data sheets provided by the vendor.
4. Click **OK**.

Configuring the Pin Settings

1. In the Design Editor, select the desired component.
2. Right-click on the component and select **Configure** > **Pins** from the context menu.
3. Select pins to export to the test bench.
4. Click **OK**.

   Unconnected pins are not automatically connected or exported to the test bench. Pin names for pins that may be exported are displayed in bold.

Note Setting values for required drivers overwrites default values only if the driver is connected to the pin.

Deleting Components

You can delete a component by selecting the desired component and performing one of the following actions:

- Press the Del key on your keyboard.
- Select **Edit** > **Delete**.
- Right-click and select **Delete** from the context menu.
- Click the **Delete** button on the toolbar.

When you delete a component with a bus, the “delete” action is propagated throughout the design. The component’s bus is deleted, which then deletes all the connections associated with the bus, which then deletes all the components that had a required connection to the bus.

Accessing Component Information

From the Platform Express interface, you can access information about a component directly from the vendor’s web site. For example, to access information from the ARM web site:

1. Select the desired ARM-based component in the Design Editor.
2. Right-click and select **Browse** > **+ ARM926EJS Technical Refenence Manual** from the context menu.

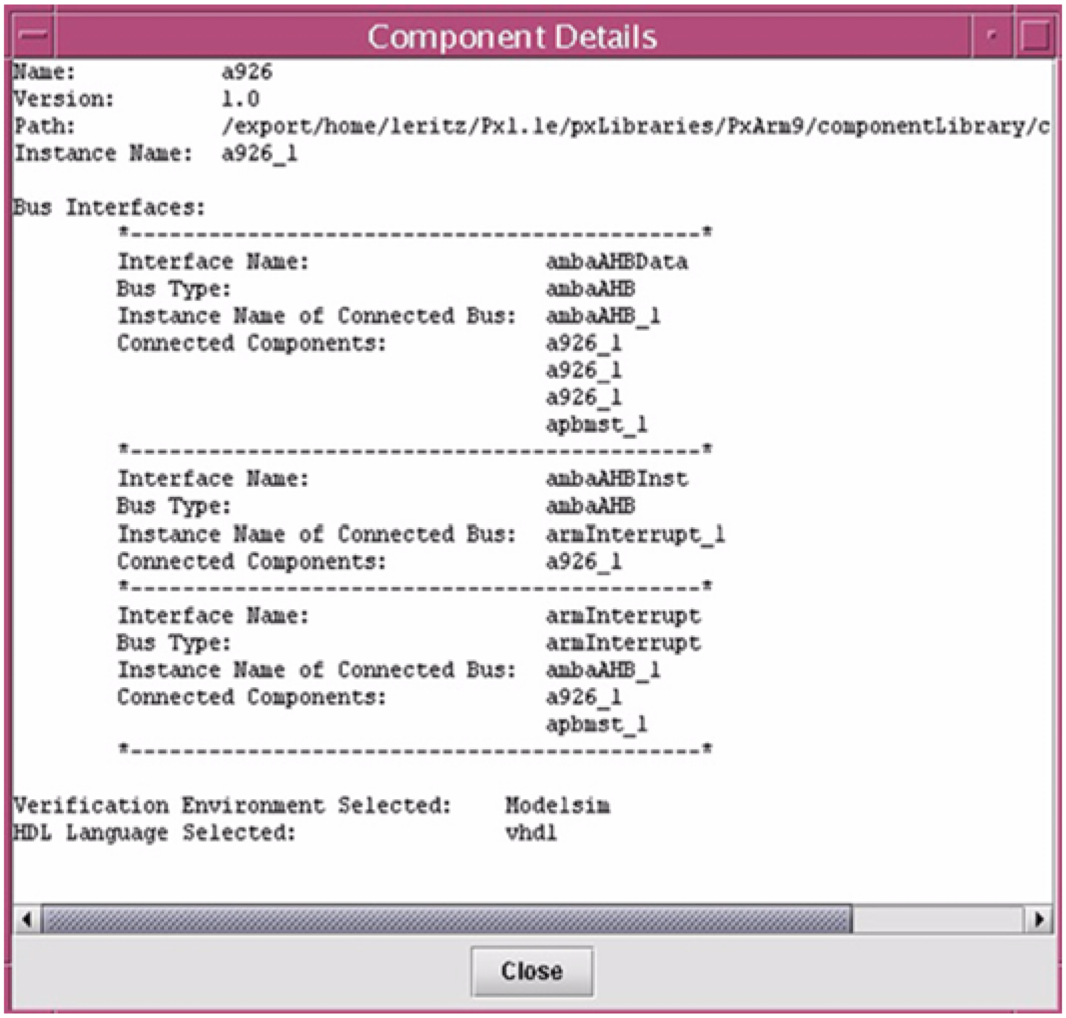
   Platform Express invokes a browser and loades the specified URL.

Note The exact menu choice of the context menu varies depending on the selected core.

Accessing the Details Window

You can also display the Details window that contains information on the selected core. The Details window displays details about the component or bus such as instance names, types, and connections.

1. Select the desired component in the Design Editor.
2. Right-click and select **Details** from the context menu.



```
Name:           a926
Version:        1.0
Path:           /export/home/leritz/Px1.1e/pxLibraries/PxArm9/componentLibrary/c
Instance Name:  a926_1

Bus Interfaces:
        *---------------------------------------------*
        Interface Name:                 ambaAHBData
        Bus Type:                       ambaAHB
        Instance Name of Connected Bus: ambaAHB_1
        Connected Components:           a926_1
                                        a926_1
                                        a926_1
                                        apbmst_1
        *---------------------------------------------*
        Interface Name:                 ambaAHBInst
        Bus Type:                       ambaAHB
        Instance Name of Connected Bus: armInterrupt_1
        Connected Components:           a926_1
        *---------------------------------------------*
        Interface Name:                 armInterrupt
        Bus Type:                       armInterrupt
        Instance Name of Connected Bus: ambaAHB_1
        Connected Components:           a926_1
                                        apbmst_1
        *---------------------------------------------*

Verification Environment Selected:   Modelsim
HDL Language Selected:               vhdl
```

Figure 2-4. Details window

Working with Connectors

Creating Connections

When you instantiate a component, Platform Express automatically creates all the required connections. A bus (“Bus A”) can be dragged to another compatible bus (“Bus B”). This action removes “Bus A” and inserts a secondary master connection on “Bus B.” The bus interface that was previously represented by “Bus A” becomes connected to “Bus B” and is now represented by a connector.

Connecting an Unconnected Component

- Drag the component to a bus.

  Platform Express automatically selects the component interface to connect to the bus, and displays the bus interface ID next to the connector.

- To change the component interface, right-click the bus interface ID and select the desired bus interface.

  (This works *only* if there is more than one available, compatible bus interface. If so, an asterisk appears next to the bus interface ID. It is changed by clicking the interface ID, not the connector.)

Deleting Connections

When you delete a required connection, the component to which it belongs is automatically deleted. You can delete a connection by selecting the desired connection and performing one of the following actions:

- Press the Del key on your keyboard.
- Select **Edit** > **Delete**.
- Right-click and select **Delete** from the context menu.
- Click the **Delete** button on the toolbar.

When you delete a required connection, the component to which it belongs is deleted automatically.

Working With Buses

A *bus* refers to a path used for transmitting signals from any of several sources to any of several destinations. A *bus bridge* is a component with a master bus interface and a slave bus interface in which signals arriving on the slave interface are translated into signals on the master interface. This allows slaves connected to the master interface to respond to signals from the master of the bus to which the slave interface is connected.

The Design Editor provides several ways of working with buses:

- Bus Views
- Automatic Bus Connections
- Changing Bus Connections
- Bus Bridges
- Multiple Bus Connections

Bus Views

The Design Editor displays the bus connections for all components in the design. It provides platform-defined views, which may include:

- All buses (default view)
- System bus only
- Interrupt bus only
- Reset bus only
- Timer bus only

You can switch between the various views using the Bus View menu in the status bar. Depending on the platform characteristics specified by the vendor in the core component's definition file, some buses may only be visible in the “all buses” view.

Automatic Bus Connections

As the component is dragged over the buses, the cursor changes, indicating where the component can or cannot be attached.

If a component is dropped while a bus is highlighted, the component attaches to that bus. If no bus is highlighted when the component is dropped, Platform Express selects a bus to connect to.

If a component can be connected to more than one available bus, Platform Express automatically connects to one of the buses. Afterwards, you can move the component to a different bus, if desired.

Changing Bus Connections

To change a component's bus connection, drag the connector (not the component) to the preferred bus.

To make connections to additional buses, drag the component to the preferred bus. If the component has additional connections available for that bus type, it remains connected to the original bus and an additional connection is made to the second bus. If the component does not have any additional connections available for that bus type, it is not moved.

Bus Bridges

Bus bridges create a bus-to-bus connection that allows components that can only connect to one bus type to connect to a different bus type. Bus bridges are displayed perpendicular to the parent bus, with the extension parallel to the parent bus.

If you add a component that requires a bus bridge to connect to an existing bus, a bus bridge selection dialog box may appear, and a dialog box for configuring the bus bridge may also appear. After the bus bridge is configured, the new bus and the component appear in the Design Editor.

Multiple Bus Connections

Components with connections to multiple buses (for example, to the system and interrupt buses) are represented once for each bus type. The first representation contain all labels and logos. Subsequent representations are arrow-shaped, with the instance name only. All representations are highlighted if any of them is selected. Components with multiple connections to a single bus show all connections from a single representation.

Master and Slave Components

Bus interfaces can be masters or slaves. Unconnected master interfaces are represented as buses. Primary bus masters appear as components on the left side of the Design Editor. Secondary bus masters are displayed above the bus. Bus slave components are displayed below the bus. When the maximum number of master or slave components have been added to a bus, an error message appears.

Configuring Buses

Right-click on the bus and select **Configure** > **Decoder Selection** or **Required Drivers** from the context menu to configure the bus. The required drivers are used when you build the design. The "Decoder Selection" determines which decoder module to apply to the selected bus. (The default selection is based on the HDL language you have chosen for your design.)

Warning The full path to the decoder is stored in the design. If you move the location of the libraries or the design, you may have to reconfigure the decoders.

The Required Drivers configure the generated drivers for unconnected bus pins, or for bus pins that require a driver. There are three types of drivers: Clock, Reset, and Fixed Value. The Clock and Reset drivers are exported to the test bench level. You can select whether to export the Fixed Value driver. See Configuring the Decoder Selection and Configuring the Required Drivers in Chapter 3 for more information.

Chapter 3 Building Designs

Overview

This chapter contains procedural information on configuring and building your design with Platform Express. It contains the following sections:

- Before you begin
- Configuring the Build
- Building your Design

Before you begin

This chapter assumes that you have properly installed Platform Express (including libraries), and have configured your environment to run Seamless CVE and any other required build tools (for example, ARM compilers and assemblers). Check that you have defined the following environement variables: `PXHOME`, `PXPATH` and `MODELTECH`. Consult the *Platform Express Installation Manual* for complete instructions. This chapter also assumes that you have created a design. See Chapter 2, Creating Designs, for information.

Note You cannot perform a build unless your environment is properly configured. For example, if you select an ARM core, your environment must be configured to run the ARM tools.

Configuring the Build

The configurable design elements of your project are entirely dependent on the core platform you have selected for your design. Before building, you may need to set signals for driving unconnected pins, specify where to load the software, or generate code for remapping address ranges.

Defining all configurable components

Select **Settings > Configure All** to access all of the configurable components of your entire design. The Required Configuration dialog box appears. The Required Configuration dialog box enables you to configure your entire project from one dialog box interface. The following configurable top-level elements are:

- Design
- Component
- Bus

The exact fields listed in the Required Configuration dialog depend on the core and other modules of your design.

Defining configurable components separately

In addition to the all-encompassing Required Configuration dialog, other separate build configuration dialogs include:

Table 3-1. Configuration Dialog Boxes

Dialog Box Name	Access	Purpose
<*design file name*>Project Settings Configuration — Basic	**Settings > Project Settings**	Define settings for the current project.

Table 3-1. Configuration Dialog Boxes

Dialog Box Name	Access	Purpose
<*core name*>Basic Configuration — Basic	Right-click on the core and select **Configure > Basics** from the context menu.	Define basic settings for the selected component.
<*core name*>Pin Configuration — Pins	Right-click on the core and select **Configure > Pins** from the context menu.	Select which pins to export to the test bench.
<*core name*>Scatter Loader Configuration — Scatter Loader Panel	Right-click on the core and select **Configure > Scatter Loader** from the context menu.	Define the load addresses for software modules.
<*core name*>Software Memory Configuration — Software Memory Configuration Panel	Right-click on the core and select **Configure > Software Memory** from the context menu.	Define custom memory configurations.
<*bus_name*>Decoder Selection Configuration — Decoder Selection Panel	Right-click on a bus and select **Configure > Decoder Selection** from the context menu.	Choose the decoder module to apply for the selected bus.
<*bus_name*>Required Driver Configuration	Right-click on a bus and select **Configure > Required Drivers** from the context menu.	Configure the generated drivers for unconnected bus pins (or for pins marked as *requiresDriver*).

Setting Driver Signals

Drivers can be connected to components or buses. There are three types of drivers:

- Fixed

- Single-shot
- Clocks

If not otherwise configured using the Required Drivers or Pin Configuration dialog boxes, single-shot and clock drivers will be initialized with the design's default values.

Pins with complex drivers (for example, clock drivers and single-shot drivers) must be exported to the test bench to be configured. The HDL code generated for complex drivers cannot be synthesized. Pins with simple (fixed value) drivers may be initialized in Platform Express or exported to the test bench. The pin drivers will only be created if the pin is not connected to a bus. If the pin is connected to a bus, the bus decoder logic must drive the pins as necessary.

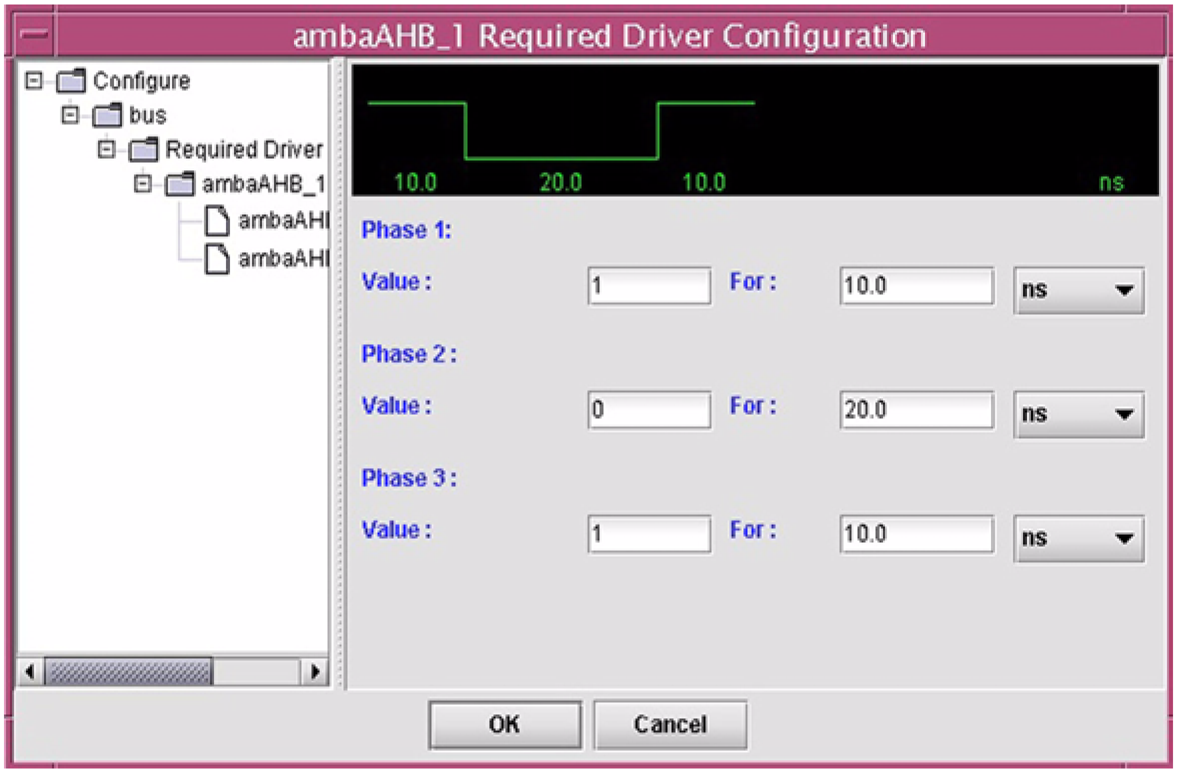
Configuring the Required Drivers

At the beginning of the build process, Platform Express verifies that all required signal levels and the duration for certain signals within your design (as required by the components' definition files) have been configured either by default or by the user. (The "Required Driver Configuration" dialog box appears if any required signals have not been configured.)

To manually access the "Required Drivers" dialog box:

1. Select the desired bus.
2. Right-click on the bus and select **Configure** > **Required Drivers** from the context menu.

   The Required Driver Configuration dialog box appears.



Phase 1:

Value : 1 For : 10.0 ns

Phase 2 :

Value : 0 For : 20.0 ns

Phase 3 :

Value : 1 For : 10.0 ns

Figure 3-1. Required Driver Configuration dialog box

3. In the “Required Drivers” folder in the left-hand column, select the desired driver from the bus folder.

   By default, the first driver in the bus folder is displayed automatically.

4. In each “Value:” field(s), supply a value for each required signal (delineated in the dialog box with “Phase N”).

   The allowed values for the “Value” field are: 0, 1 or nothing.

5. In the “For:” field corresponding to each “Value:” field, enter the desired duration.

6. In the drop-down menu to the right of each “For:” field, select one of the following time units:

   > ns (nanosecond: one billionth)

> fs (femtosecond: one quadrillionth)

> ps (picosecond: one trillionth)

> us (microsecond: one millionth)

> ms (millisecond: one thousandth)

> sec (second)

> min (minute)

> hr (hour)

7. Click **OK**.

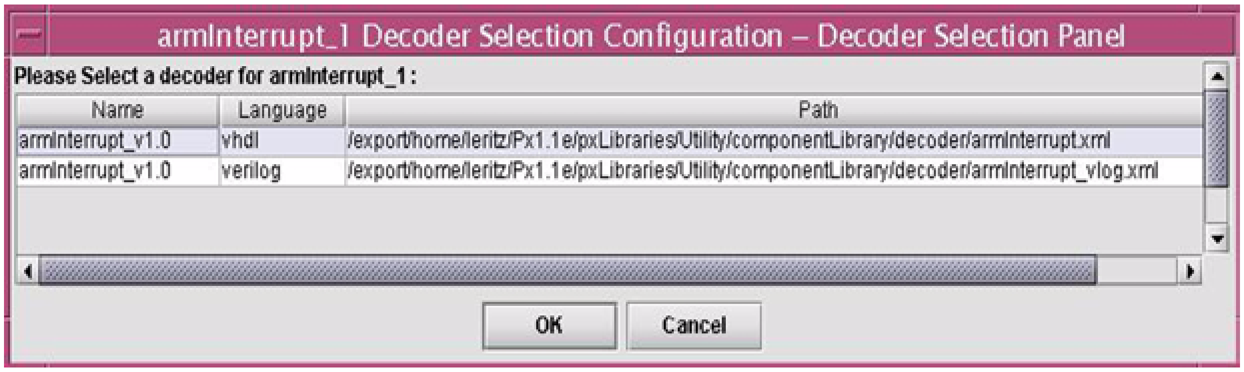
Note You must specify all required signals *before* building your design.

Configuring the Decoder Selection

The "Decoder Selection" refers to the specific decoder module that is applied to a given bus. The default selection is based on the current design's HDL language.

1. Select the desired bus.
2. Right-click on the bus and select **Configure > Decoder Selection** from the context menu.

   The Configure Decoder Selection dialog box appears.


armInterrupt_1 Decoder Selection Configuration – Decoder Selection Panel

Please Select a decoder for armInterrupt_1 :

Name	Language	Path
armInterrupt_v1.0	vhdl	/export/home/leritz/Px1.1e/pxLibraries/Utility/componentLibrary/decoder/armInterrupt.xml
armInterrupt_v1.0	verilog	/export/home/leritz/Px1.1e/pxLibraries/Utility/componentLibrary/decoder/armInterrupt_vlog.xml

Figure 3-2. Configure Decoder Selection Dialog Box

3. Select the desired decoder from the list and click **OK**.

Note The values in the Name, Language and Path fields in the "Configure Decoder Selection" dialog box are Read-Only.

Defining the Pin Configuration

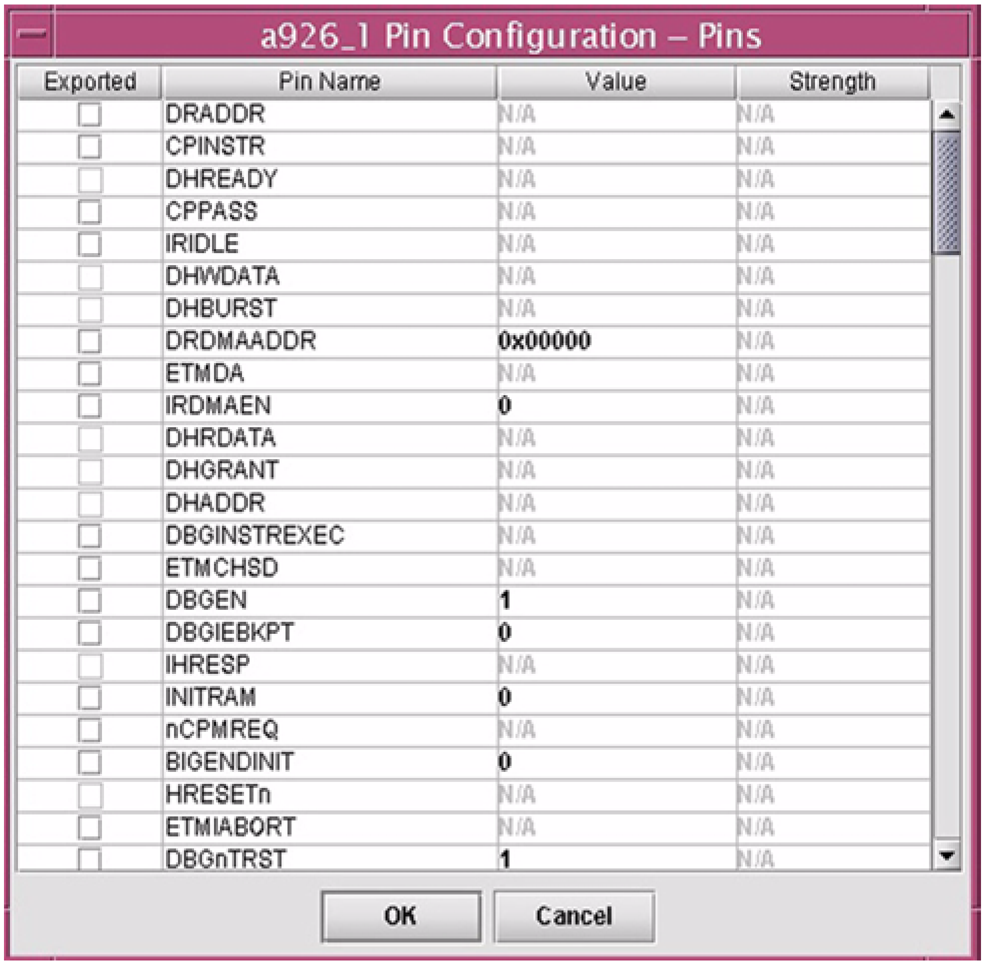
Some pins in a component's definition file may be specified to allow exporting. Exported pins emerge in the HDL-generated code. Unconnected pins are not automatically connected or exported.

1. Select the desired component.
2. Right-click on the component.
3. Select **Configure** > **Pins** from the context menu to set pin driver values.

Note Pins that can be configured use a bold font for their values.

Pin Configuration Example

In Figure 3-3, DRDMAADDR, IRDMAEN, DBGEN, DBGEIBKPT, INITRAM, BIGENDINIT, and DBGnTRST are all configurable pins.



a926_1 Pin Configuration – Pins

Exported	Pin Name	Value	Strength
☐	DRADDR	N/A	N/A
☐	CPINSTR	N/A	N/A
☐	DHREADY	N/A	N/A
☐	CPPASS	N/A	N/A
☐	IRIDLE	N/A	N/A
☐	DHWDATA	N/A	N/A
☐	DHBURST	N/A	N/A
☐	DRDMAADDR	0x00000	N/A
☐	ETMDA	N/A	N/A
☐	IRDMAEN	0	N/A
☐	DHRDATA	N/A	N/A
☐	DHGRANT	N/A	N/A
☐	DHADDR	N/A	N/A
☐	DBGINSTREXEC	N/A	N/A
☐	ETMCHSD	N/A	N/A
☐	DBGEN	1	N/A
☐	DBGIEBKPT	0	N/A
☐	IHRESP	N/A	N/A
☐	INITRAM	0	N/A
☐	nCPMREQ	N/A	N/A
☐	BIGENDINIT	0	N/A
☐	HRESETn	N/A	N/A
☐	ETMIABORT	N/A	N/A
☐	DBGnTRST	1	N/A

Figure 3-3. Pin Configuration dialog box

Using the Scatter Loader

The scatter loader is used to specify the load addresses for software modules. (See Table 3-2 for a list of supported data types in the scatter loader.)

Follow these steps to access and manipulate the data in the scatter loader dialog box:

1. Right-click on the platform core and select **Configure** > **Scatter Loader** from the context menu.

The Scatter Loader Configuration - Scatter Loader Panel dialog box appears with the current CPU.

2. Assign software modules to execution regions by highlighting unassigned files in the left panel, highlighting a target execution region in the right panel, and clicking the add (>>>) button.

3. Remove software modules by highlighting the filename in the right panel and clicking the remove (<<<).

Platform Express creates default load and execution regions based on the memory map of the address space.

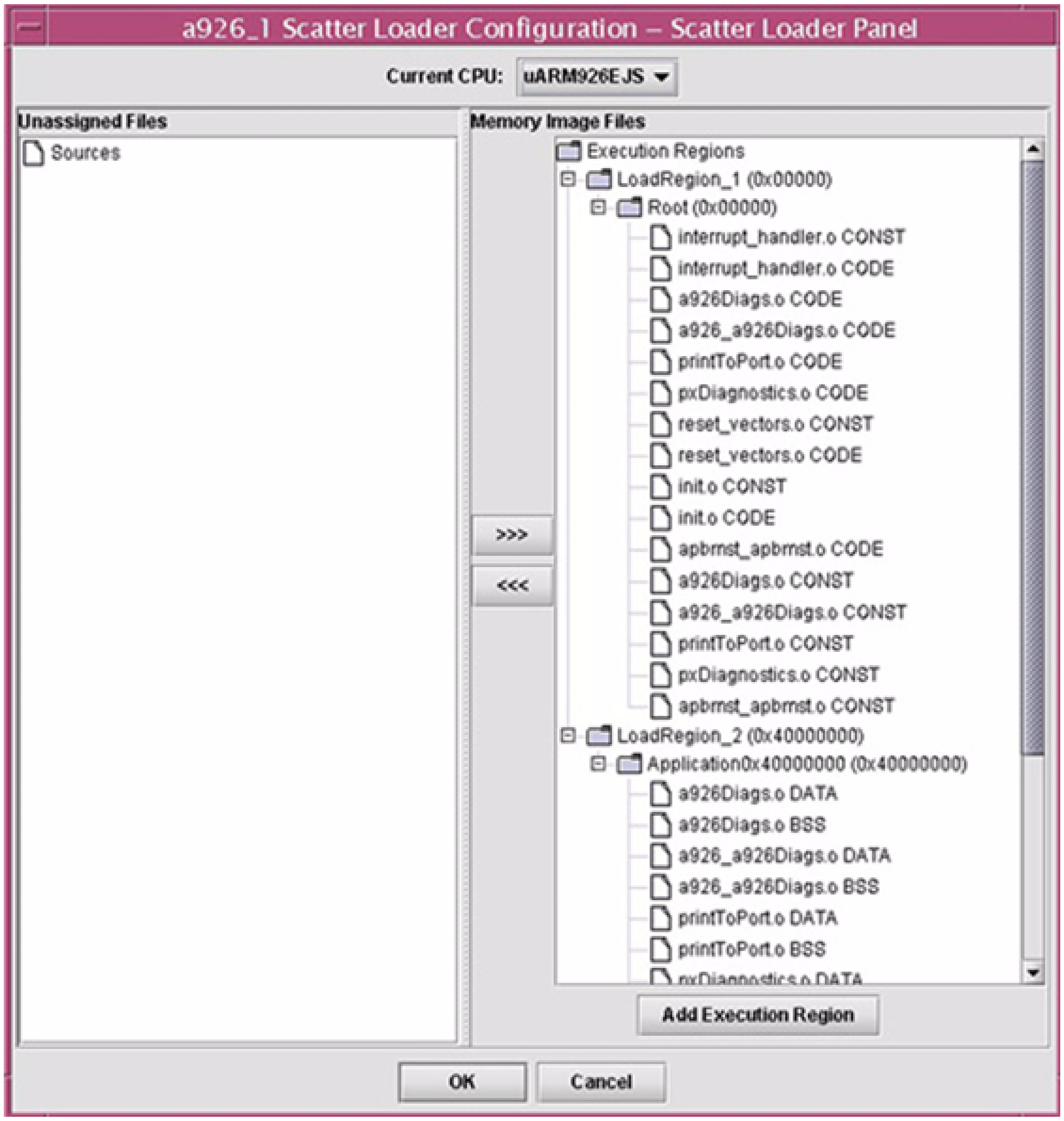


Figure 3-4. Scatter Loader dialog box

Adding Execution Regions

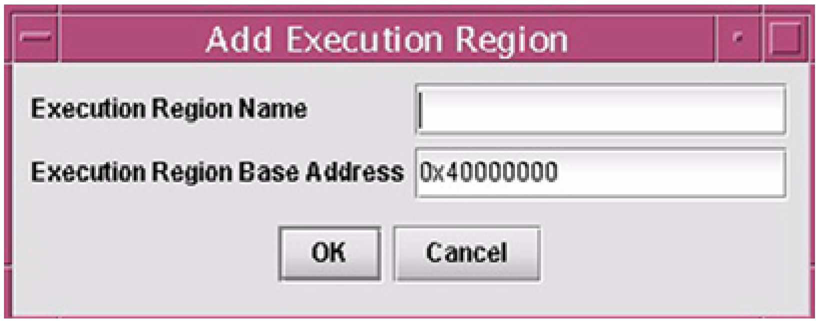
Each memory unit in your design is displayed in its own Load Region. (See the Memory Image Files column in the Scatter Loader dialog box.) To place code into a given Load Region, you must ensure that it is also placed in an Execution Region.

Note Platform Express automatically creates an Execution Region for the default RAM and ROM in the design.

Follow these steps to add a new software Execution Region:

1. Right-click on the platform core and select **Configure > Scatter Loader** from the context menu.
2. Select the desired Load Region folder in the Memory Image Files column.
3. Click the **Add Execution Region** button below the Memory Image Files column.

   The Add Execution Region dialog box appears.



Execution Region Name
Execution Region Base Address 0x40000000

Figure 3-5. Add Execution Region dialog box

4. Enter a name and base address for the new execution region.

5. Click **OK**.

Scatter Loader Data Types

The Scatter Loader uses a limited set of data types. Table 3-2 lists the set of data types.

Table 3-2. Scatter Loader Data Types

Data Type	Description
CODE (RO-CODE)	Read-Only Instruction Code: for loading into ROM only.
CONST (RO-DATA)	Constant Data - Strings, Constants, and Literal Pool data: can be loaded into either ROM or RAM.
DATA (RW)	Read/Write areas of your code: for loading into RAM only.
BSS (ZI)	ZI (Zero Initialization) Read/Write areas of code: for loading into RAM only.

Remapping Address Ranges

Platform Express provides support for peripherals that can change the address where they reside based on the state of other signals (remap signals). The bus decoder generates the appropriate HDL code.

Configuring Software Memory

Platform Express enables you to add custom memory configurations with the “Software Memory Configuration Panel.” (See Figure 3-6.) You have the ability to add, edit and remove your custom software memory. The Memory Map Pane immediately refreshes to display your custom software memory configuration. Custom software memory configurations appear in blue.

1. Right-click on the core and select **Configure** > **Software Memory** from the context menu.
2. In the Software Memory Configuration dialog box, click **Add**.
3. In the Software Memory Properties dialog, enter a value for the Name, Address, and Size fields.
4. Click **OK**.
5. Click **OK** again.

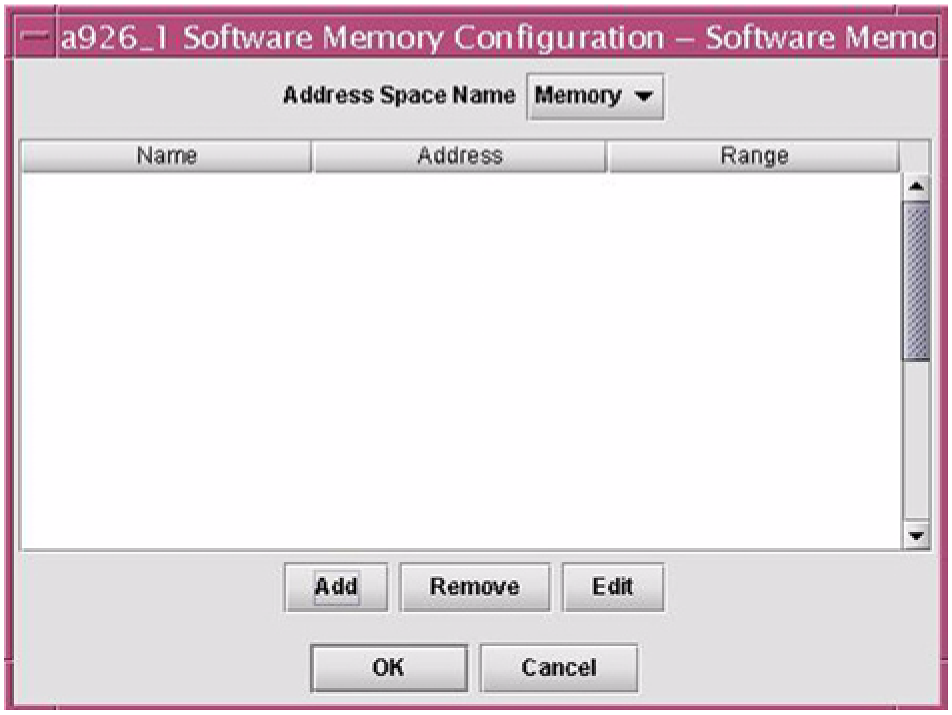


Figure 3-6. Sofware Memory Configuration Panel

Saving Your Design

When you save a design, Platform Express creates a project directory for your design. Platform Express names the project directory using the design name, *project_name*, which you specified when you saved your design. The project directory can include:

- The project design file. *project_name*`.plx`, which is created when you select **File** > **Save**, and specifies:
  - o Components contained in your design
  - o Parameters for those components
  - o Component and interrupt connections

  Platform Express uses the information in the project design file to build the design.

- The project settings that apply specifically to that project.
- The *verificationEnv* directory, which is created when you select **Tools** > **Build**, contains the following subdirectories:
    - o *verificationEnv/*`{simulator}/hdl`

      where {simulator} is either Modelsim or Ncsim.

      Contains Verilog and VHDL source files for your hardware design.
    - o *verificationEnv/*`software`

      Contains software source code and related include files for the software design that is to be executed on your design. There is a subdirectory for each supported simulator containing build scripts, compiled HDL, and Seamless configuration files. Refer to “Appendix A, Directory Structures,” for a complete list of directories.
    - o *verificationEnv/*`exec`

Note If you select **Tools** > **Build** before you have saved your design, Platform Express creates a unique directory for the design.The directory name begins with `plx`.

Hardware Design Files

Hardware design files are top-level HDL files that instantiate components and define how the platform core connects to any decoders, bus bridges, and interrupt-handling structures in your design.

- Decoder-logic HDL design files, which implement the interconnection logic between components and the platform buses.
- Top-level HDL
- Build scripts

- HDL test bench

Software Design Files

Software design files can contain:

- Boot code, both assembly language (for cores) and C-language (for peripherals and memories).
- Build scripts for compiling, linking, and assembling.
- Top-level C-language code for calling initialization and diagnostics functions.
- Interrupt-service routines.

Use the Scatter Loader to specify where to load the software. See Using the Scatter Loader for more information.

Building your Design

Platform Express builds the design files based on the the component definition files, project settings, scatter loader, and any other configuration settings that you make during design creation. Building a design generates hardware *and* software files, and can also set up signals for driving unconnected pins, and generate code for remapping address ranges.

Note The **Tools** > **Build** menu is disabled until you have inserted a platform core into the Design Editor.

1. Select **Tools** > **Build**.

   The Output pane displays the build status. See Appendix C, Build Log File, for an example log from a successful build.

2. To stop the build process before it completes, select **Tools** > **Stop**.

   After a design has been built with no errors, it can be run in batch mode from the command line. Refer to Table 1-1 for a complete list of invocation switches.

Chapter 4 Verifying Designs

Overview

This chapter contains high-level conceptual and basic procedural information on verifying a design created with Platform Express. It contains the following sections:

- Invoking a Co-Verification Session
- Using the Co-Verification Session

This chapter does not cover specific usage instructions for Seamless CVE. For application-specific functionality, refer to the *Seamless CVE User's Guide and Reference Manual*, which is supplied in both PDF and HTML, and is accessible online from the **Help** menu in the Seamless CVE main window.

Before you begin

This chapter assumes that you have properly installed Platform Express (including libraries), and have configured your environment to run Seamless CVE and any other required build tools (for example, `ARM` compilers and assemblers). Check that you have defined the following environement variables: `PXHOME`, `PXPATH` and `MODELTECH`. Consult the *Platform Express Installation Manual* for complete instructions. This chapter also assumes that you have created and successfully built a design. See Chapter 2, Creating Designs, and Chapter 3, Building Designs, for information.

Invoking a Co-Verification Session

Note

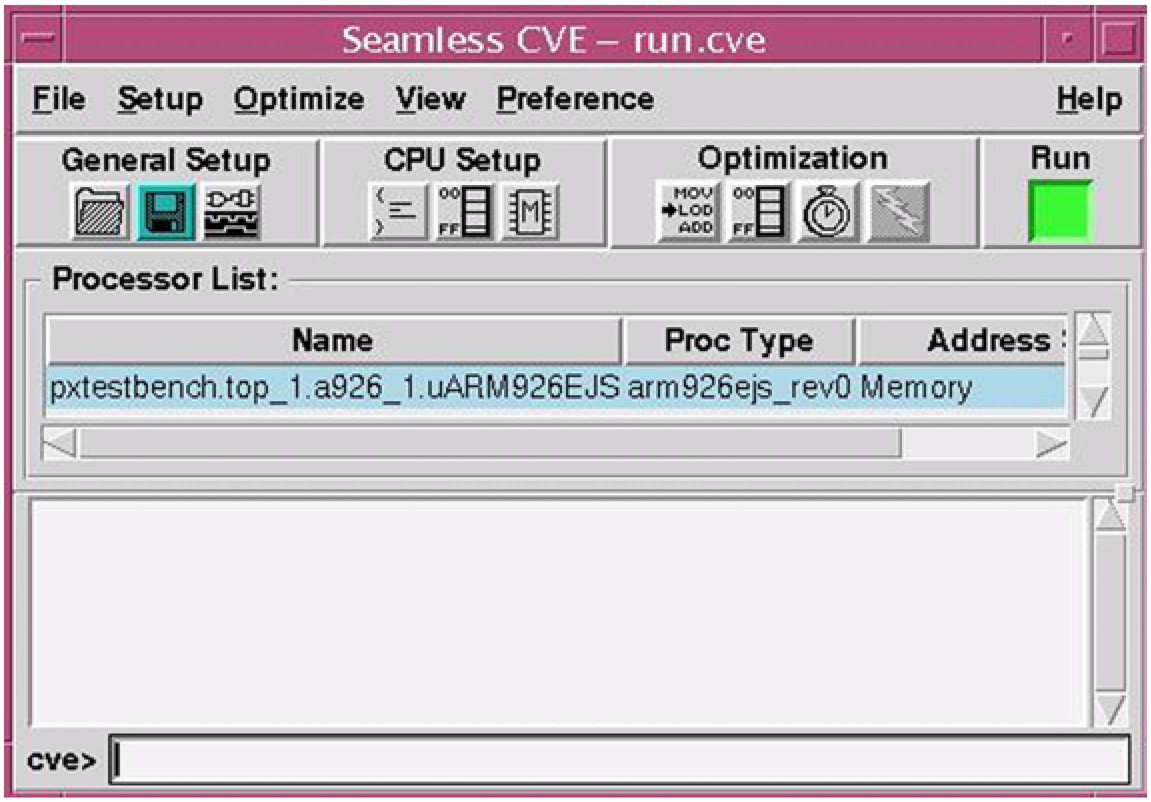
The **Tools** > **Execute** menu is disabled until you have successfully built your design. If you have not built your design, you must Select **Tools > Build** *before* selecting **Tools > Execute**.

Before attempting to invoke a co-verification session, ensure that you have properly built your design. Refer to Chapter 4, "Building Designs," for more information.

To launch a hardware/software co-verification session that uses Seamless CVE, select **Tools** > **Execute** or click the **Execute** button from the Platform Express interface. If your environment is configured properly, Platform Express invokes the Seamless co-verification tools, which use the design files generated during the Platform Express build process.

Note

Platform Express uses environment variables to invoke Seamless CVE. If Platform Express fails to invoke Seamless CVE, the likely cause is an improperly configured environment. Consult the *Platform Express Installation Manual* for information on environment settings.

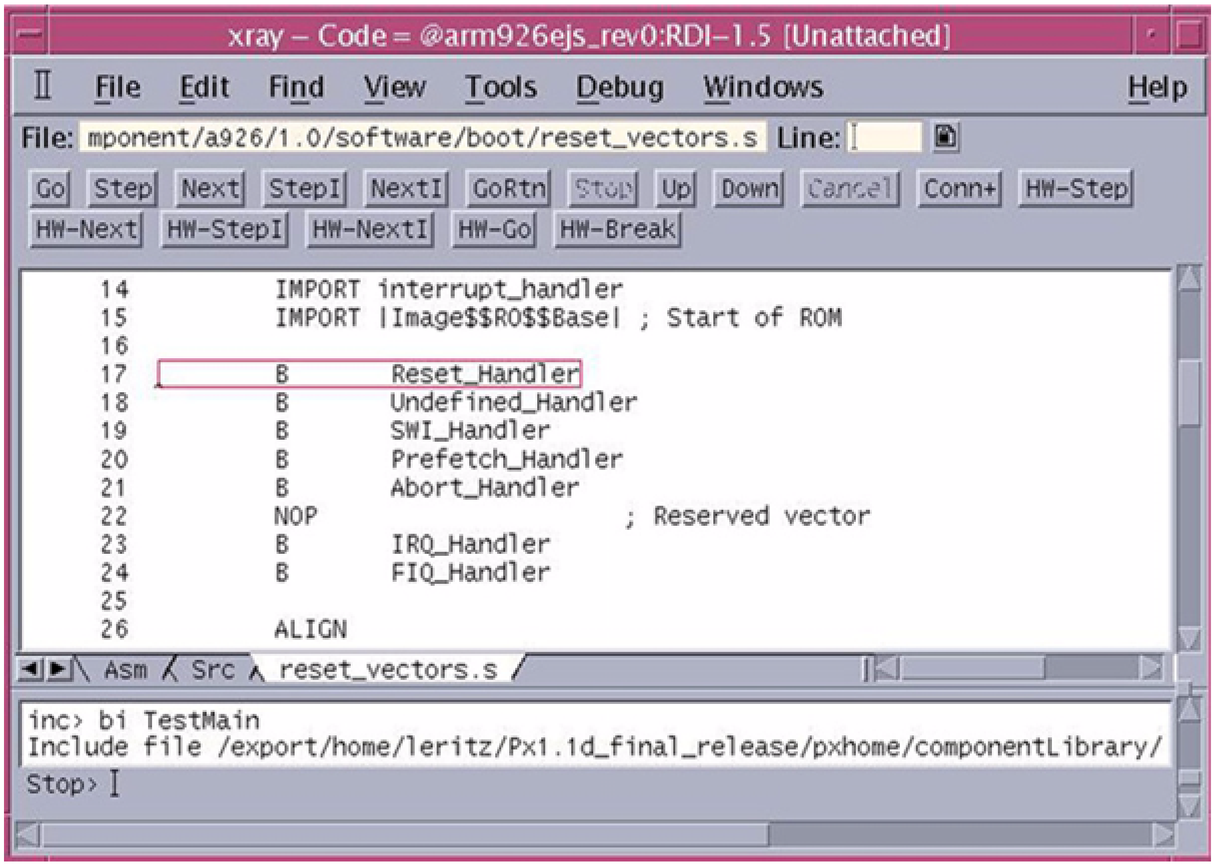


Name	Proc Type	Address
pxtestbench.top_1.a926_1.uARM926EJS	arm926ejs_rev0	Memory

Figure 4-1. Seamless CVE Interface

See Figure 4-1 for a screen captures of the Seamless CVE interface. Other applications within the Seanless Co-Verification Environment include the XRAY Debugger (Figure 4-2), ModelSim (Figure 4-3), and the Wave viewer (Figure 4-4).

After a design has been executed with no errors, it can be built and loaded into Seamless in batch mode from the command line. Refer to the *Platform Express Installation Manual* for information on invocation switches.


xray – Code = @arm926ejs_rev0:RDI–1.5 [Unattached]
File Edit Find View Tools Debug Windows Help
File: mponent/a926/1.0/software/boot/reset_vectors.s Line:
Go Step Next StepI NextI GoRtn Stop Up Down Cancel Conn+ HW-Step
HW-Next HW-StepI HW-NextI HW-Go HW-Break
Asm Src reset_vectors.s

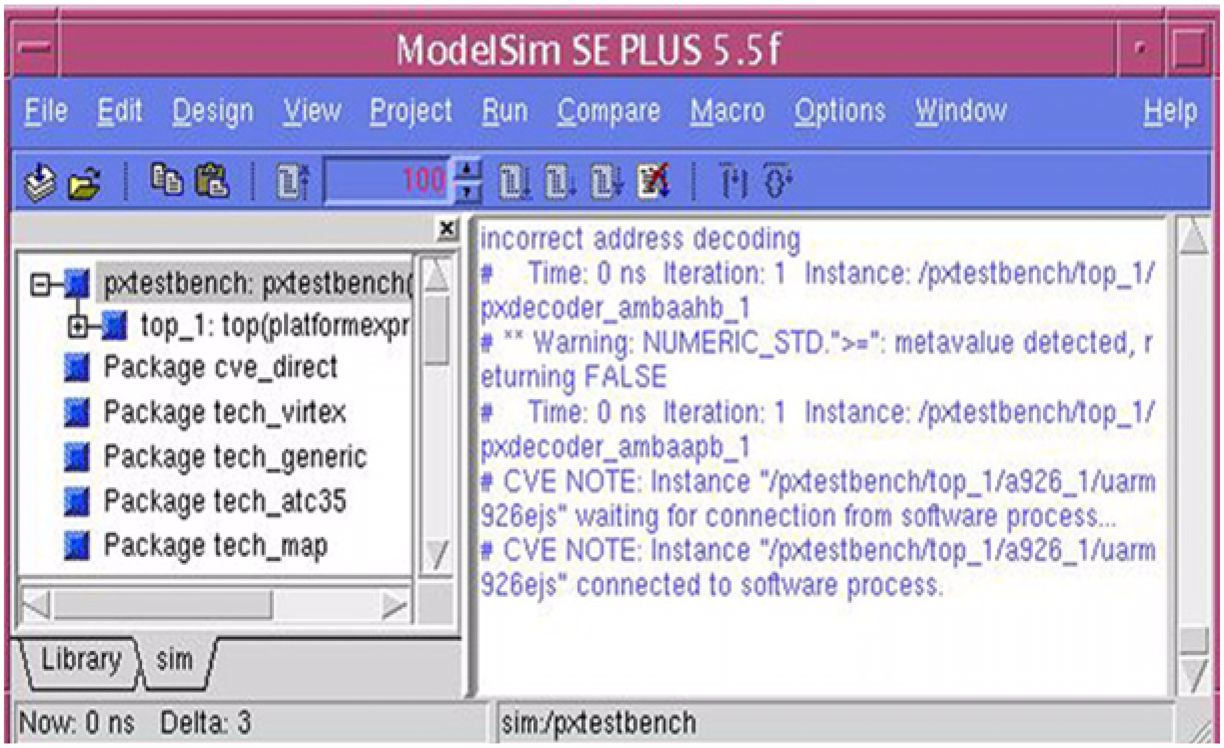
```
        IMPORT  interrupt_handler
        IMPORT  |Image$$RO$$Base| ; Start of ROM

        B       Reset_Handler
        B       Undefined_Handler
        B       SWI_Handler
        B       Prefetch_Handler
        B       Abort_Handler
        NOP                       ; Reserved vector
        B       IRQ_Handler
        B       FIQ_Handler

        ALIGN
```

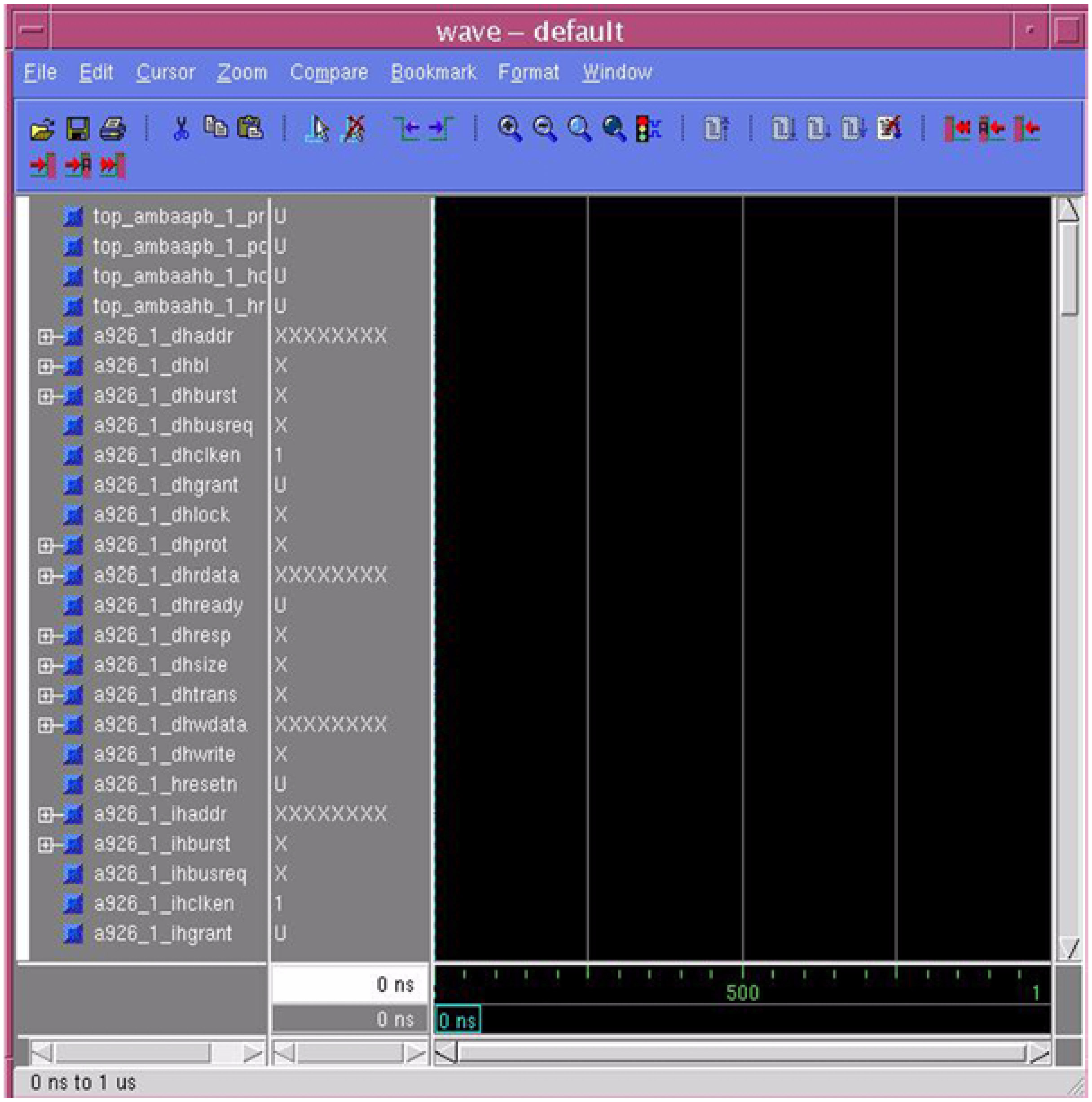
```
inc> bi TestMain
Include file /export/home/leritz/Px1.1d_final_release/pxhome/componentLibrary/
Stop>
```

Figure 4-2. XRAY Debugger interface



```
incorrect address decoding
#    Time: 0 ns  Iteration: 1  Instance: /pxtestbench/top_1/
pxdecoder_ambaahb_1
# ** Warning: NUMERIC_STD.">=": metavalue detected, r
eturning FALSE
#    Time: 0 ns  Iteration: 1  Instance: /pxtestbench/top_1/
pxdecoder_ambaapb_1
# CVE NOTE: Instance "/pxtestbench/top_1/a926_1/uarm
926ejs" waiting for connection from software process...
# CVE NOTE: Instance "/pxtestbench/top_1/a926_1/uarm
926ejs" connected to software process.
```

Figure 4-3. ModelSim interface



```
top_ambaapb_1_pr U
top_ambaapb_1_pc U
top_ambaahb_1_hc U
top_ambaahb_1_hr U
a926_1_dhaddr    XXXXXXXX
a926_1_dhbl      X
a926_1_dhburst   X
a926_1_dhbusreq  X
a926_1_dhclken   1
a926_1_dhgrant   U
a926_1_dhlock    X
a926_1_dhprot    X
a926_1_dhrdata   XXXXXXXX
a926_1_dhready   U
a926_1_dhresp    X
a926_1_dhsize    X
a926_1_dhtrans   X
a926_1_dhwdata   XXXXXXXX
a926_1_dhwrite   X
a926_1_hresetn   U
a926_1_ihaddr    XXXXXXXX
a926_1_ihburst   X
a926_1_ihbusreq  X
a926_1_ihclken   1
a926_1_ihgrant   U
```

Figure 4-4. Wave viewer interface

Using the Co-Verification Session

The actual design verification is conducted with Seamless CVE, using files generated by Platform Express. A Seamless co-verification session consists of three major processes:

- Seamless CVE, which coordinates the hardware and software simulation processes and provides optimizations that speed simulation.
- An Instruction-Set Simulator (ISS), which models the processor's instruction set and executes target processor code. Examples of supported instruction set simulators include the XRAY Debugger from Mentor Graphics and MULTI from Green Hills Software.
- A logic simulator, which simulates the hardware design, including a Seamless-compatible bus interface model of the processor and Seamless compatible memories. Examples of supported logic simulators include ModelSim from Model Technology and NC-Sim from Cadence Design Systems.

The co-verification process requires an interaction between the hardware and software simulations. The software simulator must execute instructions and the hardware simulator must run the corresponding logic simulation for the co-verification to progress.

The simulators may start automatically, because of the default behavior for the simulator, or because of commands contained in scripts that Platform Express generates and specifies as part of the simulator invocation. Alternatively, it may be necessary to start co-verification by clicking the **Run** button on the Seamless main window. When Seamless starts co-verification, the **Run** button changes from red to green.

It may also be necessary to start simulation manually from the hardware and software simulators.

Starting HW/SW Simulators Examples

- To start the ModelSim hardware simulator, move the cursor to the ModelSim command line and enter the command `run 10000`.
- To start an ISS software simulator, from the graphical user interface of the ISS/debugger:
    - click the **Go** button.

or

- repeatedly click the **Step Instruction** button.

For detailed information on the Seamless Co-verification Environment (CVE), refer to the *Seamless CVE User's Guide and Reference Manual*, which is supplied in both PDF and HTML, and is accessible online from the **Help** menu in the Seamless CVE main window.

Chapter 5 Use Case Example

Overview

This chapter contains a step-by-step tutorial for creating a design with Platform Express using the PxArm9 component library. This "use case" scenario provides step-by-step procedures as well as some conceptual explanations behind the procedures.

Set up your environment

This "Use Case" uses the following components and tools:

- PxARM9 Library
- ARM Build tools
- Seamless CVE
- ModelSim (logic simulator)
- XRAY (instruction set simulator)

Use the following commands to set up your environment, using your specific path locations.

1. `setenv PXHOME /user/name/platform_express/pxhome`

2.
   ```
   setenv PXPATH
   /user/name/platform_express/pxLibraries/AMBA:/user/name/pl
   atform_express/pxLibraries/Inventra:/user/name/platform_ex
   press/pxLibraries/Leon2:/user/name/platform_express/pxLibr
   aries/PxArm9:/user/name/platform_express/pxLibraries/Utili
   ty:/export/home/leritz/Px1.1e/pxLibraries/PxSample
   ```
3. `setenv MODELTECH /user/name/model_5.5f/modeltech`

```
4. setenv CVE_HOME /user/name/seamless_cve/cve_home.ss5
5. setenv ARMTOOLS /user/name/arm_tools/arm2.51/ss5/bin
```

Invoke Platform Express

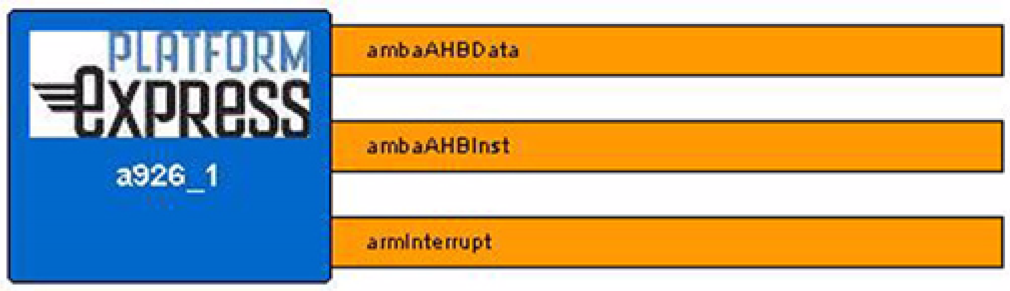
```
$PXHOME/bin/px
```

Begin Your Design

The following procedures require you to select components from the Component Browser. You will also manipulate the components in the Design Editor.

Select the Core

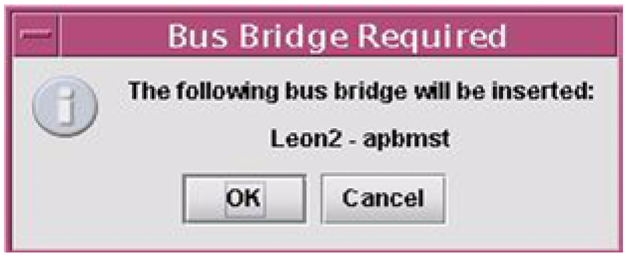
1. In the Component Browser, open the PxArm9 folder.
2. Double-click the ARM926EJ-S core.
3. In the “Configure a926 - Basic” dialog box, accept the default values and click OK.



The ARM926 core appears in the design editor. Notice that the Component Browser refreshes with all the available IP components that are compatible with the ARM926 core. Also notice that the memory map pane refreshes to display the occupied address locations. The available address locations are also displayed.

Add a Memory Component

1. In the Component Browser, open the Leon2 folder.
2. Double-click the mctrl_subsystem memory component.
3. In the "Bus Bridge Required" dialog box, click OK.

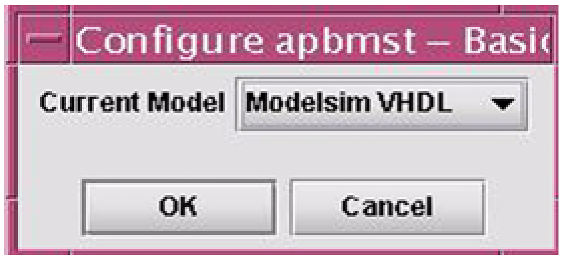


Bus Bridge Required

The following bus bridge will be inserted:

Leon2 - apbmst

4. In the "Configure apbmst - Basic" dialog box, accept ModelSim as the logic simulator model and click OK.



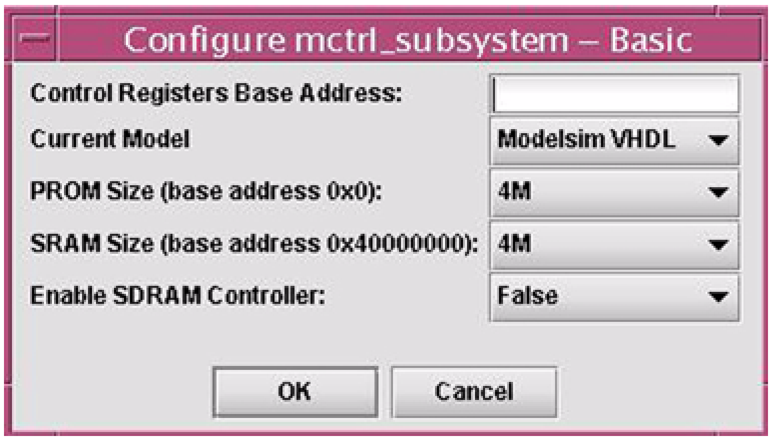
Configure apbmst – Basic

Current Model: Modelsim VHDL

5. In the "Configure mctrl_subsystem - Basic" dialog box, enter a value for the "Control Registers Base Address" field.

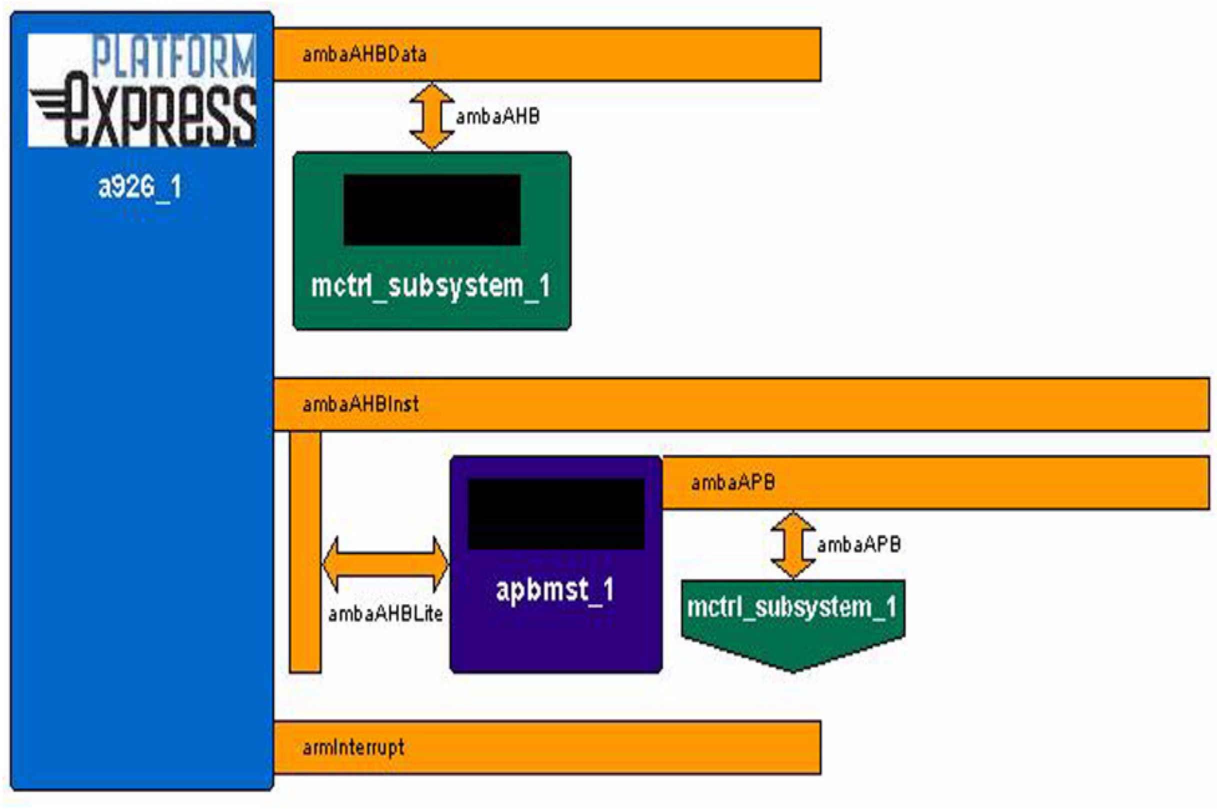
   For example: 0x10000000

   Accept the default values for the Current Model, PROM Size, SRAM Size, and Enable SDRAM Controller.

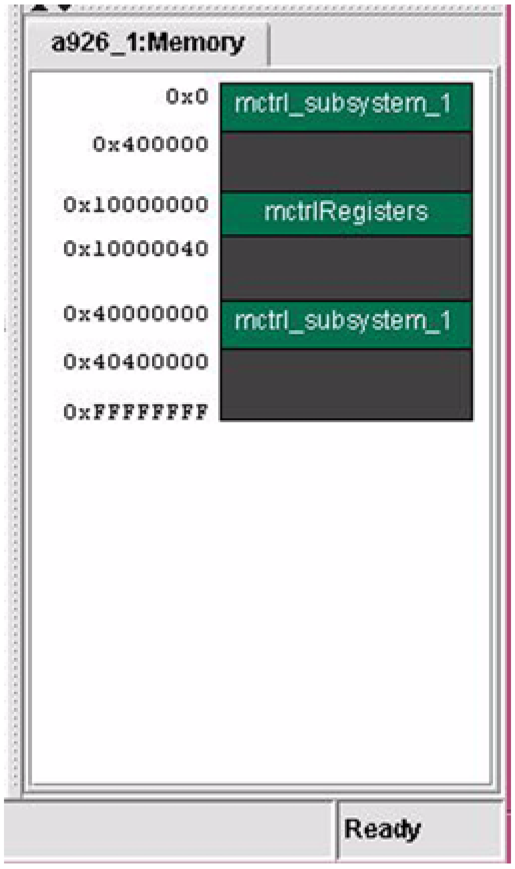

Configure mctrl_subsystem – Basic

Control Registers Base Address:	
Current Model	Modelsim VHDL
PROM Size (base address 0x0):	4M
SRAM Size (base address 0x40000000):	4M
Enable SDRAM Controller:	False

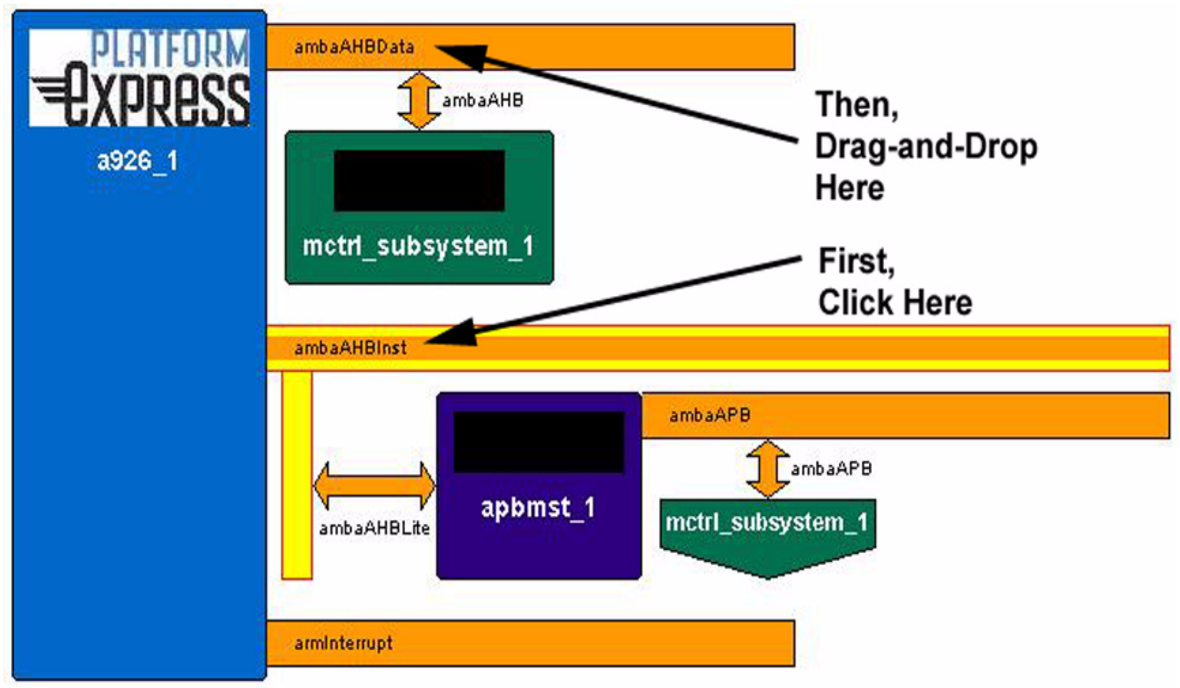
Your design now looks like this in the Design Editor:



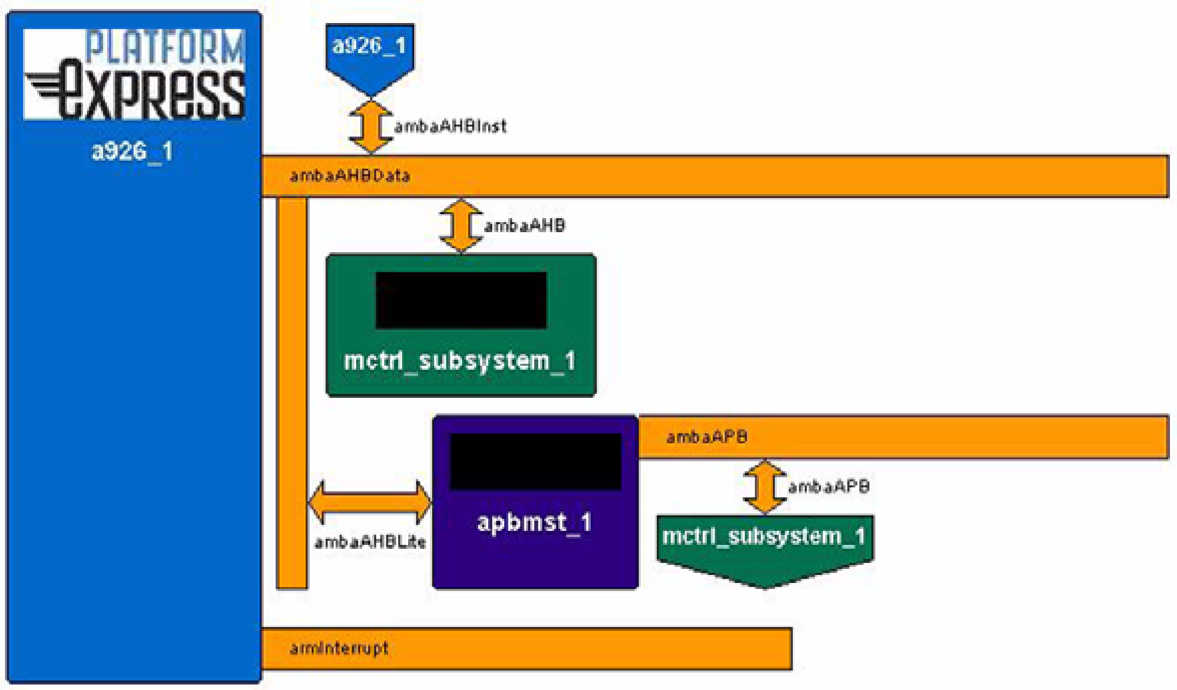
And the Memory Map Pane now looks like this:



6. Click the ambaAHBInst bus and drag-and-drop it on the ambaAHBData bus. (See the figure below.)



Your design now appears as follows:



Save the Design

Select **File** > **Save** and provide a design name in the Save dialog box.

Design Explanation

The preceding design has two buses: one for instruction, the other for data. In this design, all peripherals are accessible to both the instruction bus and the data bus. This particular scenario (in which ROM must be visible to both AMBA buses) is specific to the ARM926 core.

Build Your Design

1. Select **Tools** > **Build** to execute the build of your newly-created design.

2. In the “Required Configuration” dialog box, accept all the default values and click OK.

   The Output pane displays the build log. (See Appendix C, Build Log File, for a complete build log of a successful build.) The last few lines of your successful build should be something like:

```
Debug Area Optimization Statistics

Input debug total(excluding low level debug areas)
10976 (10.72Kb)
Output debug total                                 8416
(8.22Kb)
% reduction
23.32%

ARM Linker: finished,  0 informational, 5 warning and 0
error messages.
```

The build process generates the following:

- HDL description of the design

- software, such as low-level design diagnostics
- complete Seamless CVE configuration files (including the logic simulator and instruction set simulator configuration files)

As part of the build process, Platform Express identifies all the software code modules in your design and places them appropriately into ROM and RAM. The build process also checks several other design details, including which bus decoders to use, as well as clock and reset stimuli.

Simulate Your Design

After the build process is complete, you can now verify that the design works. Select **Tools** > **Execute** to invoke the Seamless Co-Verification Environment (CVE). The following applications are loaded:

- Seamless CVE
- ModelSim
- Waveform viewer

  To invoke the XRAY Debugger, issue the following command from the ModelSim command prompt:

  ```
  run -all
  ```

Verify Your Design

Use Seamless CVE to begin verifying your design created with Platform Express. For example, you may want to examine the HDL design hierarchy created in ModelSim.

1. In ModelSim main window, select **View** > **Structure**.
2. In the Structure window, expand the “top_1” component.

Note that each bus in the design has a component called “pxdecoder_*busname*.”

3. In the ModelSim main window, select **View** > **Source**.

   A source window appears with the “top.vhd” HDL displayed.

4. When are finished with your Seamless CVE session, select File > Exit from the Seamless CVE window.

Note Before you make any enhancements to your Platform Express design, you must close Seamless CVE. Until the verification process has stopped, you cannot add components or reconfigure existing components.

Cycle through the Design Flow

Now that you have completed one design cycle, you can make changes and enhancements to your Platform Express design and cycle through the design flow again.

Appendix A Directory Structures

Overview

This appendix contains the directory structures of:

- the default Platform Express installation
- the directories created from a build of a design

Platform Express Directory Structure

The following is the default list of directories and sub-directories in `$PXHOME`:

Table 1-1. Platform Express Installation

Directory	Description
`Pxkey`	Key file used for licensing
`api`	Application Programming Interface for component developers and integrators
`bin`	Contains Platform Express startup command (`px`) and a script for the default browser
`doc`	Online help files and schema diagrams for component developers
`etc`	Miscellaneous files
`images`	Icon image files

Table 1-1. Platform Express Installation

Directory	Description
`mgls`	Mentor Graphics licensing server
`schema`	XML schema documents
`tools/bin`	Tools and applications for component developers and integrators.

Directory Structure of Generated Design

Select **File** > **Save** to save your design files after you have built your design. A directory is created with your project name (for example, myProject). Your design is saved in the project directory (for example, `myProject.plx`).

Note The design file must have the same name as the project directory and must have the .plx extension.

After the design is built and saved, the resulting directory structure is similar to the following:

```
|myProject
 myProject.plx
   |--verificationEnv
      |
      |--Modelsim
         |
         |--exec
         |
         |--hdl
         |  |
         |  |--work
         |     |
         |     |--a926
         |     |
         |     |--arm926ejs
         |     |
         |     |--arm926ejs_lib
         |     |
         |     |--arm926ejs_rev0_engine_e
         |     |
         |     |--pxconfig_pxtestbench
         |     |
         |     |--pxconfig_top
         |     |
         |     |--pxdecoder_ambaahb_1_module
         |     |
         |     |--pxdecoder_ambaapb_1_module
         |     |
         |     |--pxtestbench
         |     |
         |     |--top
         |
         |--software
            |
            |--a926_Memory
               |
               |--coreDiagnostics
                  |
                  |--_a926_1
```

Table 1-2. Directory Structure of Platform Express Design

Directory	Description
`myProject`	Root project directory (example project name)
`verificationEnv`	Contains generated design files.
`Ncsim`	Contains design for NC-sim simulation (if selected)
`Modelsim`	Contains design for Modelsim simulation (if selected)
`exec`	Contains generated files for Seamless CVE execution
`hdl`	Generated HDL files, build scripts and compiled models
`software`	Generated software files, build scripts, object files and executable images

Appendix B Frequently Asked Questions (FAQs)

Overview

As an alternate approach to traditional documentation, the *Platform Express User's Guide* contains a list of Frequently Asked Questions (FAQs) to assist the user with common questions and/or problems. This set of FAQs is categorized by:

- Product Questions
- General Usage
- Error Messages

Product Questions

Q. Can I use other co-verification tools besides Seamless CVE?

A. No, Platform Express is designed specifically to work with the Seamless Co-Verification environment.

Q. Can I use other logic simulators besides ModelSim?

A. Yes, in addition to ModelSim, Platform Express currently works with NCSim. Support for other logic simulators is planned for future releases.

General Usage

Q. Can I simultanesously run more than one instance of Platform Express?

A. Yes, as long as you do not try to open the same design in two separate invocations of Platform Express.

Platform Express supports only one instance of a design at a time. Future releases of Platform Express will support file locking, which will gracefully warn the user when a design is already open.

Q. I've successfully built my design. How do I know if it built correctly?

A. Barring any error messages in the output pane, you can assume your design was built correctly. Refer to "Appendix C, Build Log File," to review a sample log file from a properly-built design.

Q. I've properly defined the `PXPATH` environment variable. Why aren't the libraries displayed in the Component Browser?

A. Check the Output pane's log messages for validation errors on the libraries. If there are no errors, remember that peripheral and memory components are not displayed in the Component Browser until you have instantiated the platform core. Refer to "Chapter 2, Creating Designs," for instructions on instantiating the platform core.

Q. Why didn't the Instruction Set Simulator (ISS) of the Seamless CVE automatically load when I selected **Tools > Execute**?

A. Some Instruction Set Simulators do not start immediately when Seamless starts. You may need to start the logic simulator and advance until the processor comes out of reset. For example, for ModelSim, enter the following at the command prompt:

```
run -all
```

Error Messages

Q. I've defined my environment to point to my Seamless CVE and ModelSim installation. Why am I still getting build errors? For example, I'm getting the following error below:

```
################################################################
##                 Px Generated File
##
```

```
##           Platform Express, Version 1.1d
##
##           SoC Verification Division
##
##           Mentor Graphics Corporation
##
##
##
## Generated on: September 19, 2002 3:46:21 PM PDT
##
## Generated by: leritz
##
## Modelsim HDL compile script
##
#############################################################

if [ -f pxenv ] ; then . ./pxenv; fi
if [ ! -d work ] ; then vlib ./work; fi
./compileHdl.sh: vlib: not found
cveRunsh:cve:not found
```

A. Ensure that you have defined the bin directories of Seamless CVE, ModelSim, and any other required build tools (such as ARM tools) in your path. For example, the following path statement enables Platform Express to invoke the Seamless CVE tools:

```
setenv PATH
/user/tools/modeltech/bin:/user/tools/cve_home.ss5/bin
```

Q. I've defined my environment to point to my Seamless CVE and ModelSim installation, and also included the bin directories of Seamless CVE and ModelSim in my path. Why am I still getting errors?

A. Ensure that you are using the supported versions of Seamless CVE and logic simulators. Platform Express is certified to run with specific versions only. Consult the *Platform Express Release Notes* for the latest information on supported Seamless CVE and logic simulator versions.

Q. Why do I get the following error message when I invoke Platform Express?

```
[ERROR]   - Validation failed on path
/export/home/leritz/Px1.1d_3/pxhome/pxLibraries
```

A. When it is invoked, Platform Express loads the libraries defined by the PXPATH environment variable. The PXPATH variable is a colon-separated list of paths in which each path points to the top of a component library. For example:

```
setenv PXPATH
/home/user/apps/pxLibraries/AMBA:/home/user/apps/pxLibraries/
INVENTRA:/export/home/leritz/Px1.1d3/pxLibraries/Utility
```

Warning

Do not install component libraries (pxLibraries) in a sub-directory under `PXHOME`.

If you do not set the `PXPATH` variable, the primary default is as follows: if the current working directory (where you invoked Platform Express) contains a sub-directory named pxLibraries, a default PXPATH is constructed from all the subdirectories of ./pxLibraries. The secondary default is as follows: if there is a directory named pxLibraries in the same parent as `PXHOME`, then a default `PXPATH` is constructed from all the subdirectories of `$PXHOME/..pxLibraries`.

Appendix C Build Log File

Overview

This appendix contains the output of a log file from a successful build.The following design example was based on the PxArm9 library, using ARM tools. Review this log file for an example of a properly-built design.

```
################################################################
##                   Px Generated File
##
##             Platform Express, Version 1.1d
##
##             SoC Verification Division
##
##             Mentor Graphics Corporation
##
##
##
## Generated on: October 3, 2002 10:28:40 AM PDT
##
## Generated by: leritz
##
## Modelsim HDL compile script
##
################################################################

if [ -f pxenv ] ; then . ./pxenv; fi
if [ ! -d work ] ; then vlib ./work; fi
vmap work ./work
Modifying modelsim.ini
vcom   -93   ${CVE_HOME}/vsim_vhdl/arm926ejs_rev0.vhd
Model Technology ModelSim SE vcom 5.5f Compiler 2002.01 Jan  7
2002
-- Loading package standard
-- Compiling entity arm926ejs_rev0_engine_e
-- Compiling architecture engine_a of arm926ejs_rev0_engine_e
-- Loading package textio
-- Loading package std_logic_1164
-- Compiling entity arm926ejs
-- Compiling architecture model of arm926ejs
-- Loading entity arm926ejs_rev0_engine_e
-- Compiling package arm926ejs_lib
vcom   -93
${a926_1}/componentLibrary/component/a926/1.0/hdlsrc/A926_mod
.vhd
Model Technology ModelSim SE vcom 5.5f Compiler 2002.01 Jan  7
2002
-- Loading package standard
-- Loading package std_logic_1164
-- Loading package arm926ejs_lib
-- Compiling entity a926
-- Compiling architecture struct of a926
-- Loading package textio
-- Loading entity arm926ejs
vcom  -93 PxDecoder_ambaAPB_1_module.vhd
Model Technology ModelSim SE vcom 5.5f Compiler 2002.01 Jan  7
2002
-- Loading package standard
-- Loading package std_logic_1164
-- Loading package numeric_std
-- Compiling entity pxdecoder_ambaapb_1_module
-- Compiling architecture platformexpress of
pxdecoder_ambaapb_1_module
vcom  -93 PxDecoder_ambaAHB_1_module.vhd
Model Technology ModelSim SE vcom 5.5f Compiler 2002.01 Jan  7
2002
-- Loading package standard
-- Loading package std_logic_1164
-- Loading package numeric_std
-- Loading package attributes
-- Loading package std_logic_misc
-- Compiling entity pxdecoder_ambaahb_1_module
-- Compiling architecture platformexpress of
pxdecoder_ambaahb_1_module
vcom  -93 top.vhd
Model Technology ModelSim SE vcom 5.5f Compiler 2002.01 Jan  7
2002
-- Loading package standard
-- Loading package std_logic_1164
-- Compiling entity top
-- Compiling architecture platformexpress of top
-- Loading package arm926ejs_lib
-- Loading entity a926
-- Loading package std_logic_arith
-- Loading package target
-- Loading package device
-- Loading package textio
-- Loading package config
-- Loading package sparcv8
-- Loading package iface
-- Loading package amba
-- Loading entity leon2_apbmst
-- Loading package std_logic_unsigned
-- Loading entity mctrl_subsystem
-- Loading package numeric_std
-- Loading entity pxdecoder_ambaapb_1_module
-- Loading package attributes
-- Loading package std_logic_misc
-- Loading entity pxdecoder_ambaahb_1_module
-- Compiling configuration pxconfig_top
-- Loading entity top
-- Loading architecture platformexpress of top
vcom  -93 pxtestbench.vhd
Model Technology ModelSim SE vcom 5.5f Compiler 2002.01 Jan  7
2002
-- Loading package standard
-- Loading package std_logic_1164
-- Compiling entity pxtestbench
-- Compiling architecture platformexpress of pxtestbench
-- Loading entity top
-- Compiling configuration pxconfig_pxtestbench
-- Loading entity pxtestbench
-- Loading architecture platformexpress of pxtestbench
-- Loading configuration pxconfig_top
pli parameter
value=${CVE_HOME}/lib/arm926ejs_rev0_vsim_vlog.slib
val=${CVE_HOME}/lib/arm926ejs_rev0_vsim_vlog.slib
#############################################################
###########
##                Px Generated File
##
```

```
##           Platform Express, Version 1.1d
##
##           SoC Verification Division
##
##           Mentor Graphics Corporation
##
##
##
## Generated on: October 3, 2002 10:28:59 AM PDT
##
## Generated by: leritz
##
## Software compile script
##
##############################################################
###########

if [ -f pxenv ] ; then . ./pxenv; fi
armasm -g -I${a926_1_1}/software/include -
I${a926_1}/common/include   ${a926_1_1}/software/boot/init.s
-o ${PX_BUILD}/init.o
armasm -g -I${a926_1_1}/software/include -
I${a926_1}/common/include
${a926_1_1}/software/boot/reset_vectors.s -o
${PX_BUILD}/reset_vectors.o
armcc -c -g -I${a926_1_1}/software/include -
I${a926_1}/common/include
${a926_1_1}/software/boot/interrupt_handler.c -o
${PX_BUILD}/interrupt_handler.o
"/export/home/leritz/Px1.1d_final_release/pxLibraries/PxArm9/
componentLibrary/component/a926/1.0/software/boot/interrupt_h
andler.c", line 6: Warning: inventing 'extern int
printToPort();'
/export/home/leritz/Px1.1d_final_release/pxLibraries/PxArm9/c
omponentLibrary/component/a926/1.0/software/boot/interrupt_ha
ndler.c: 1 warning, 0 errors, 0 serious errors
armcc -c -g -I${apbmst_1_1}/software/include -
I${apbmst_1}/common/include -
DFUNCNAME=apbmstCoreDiagnostics_1  ${PX_BUILD_1}/apbmst.c -o
${PX_BUILD}/apbmst_apbmst.o
armcc -c -g -I${a926_1_1}/software/include -
I${a926_1}/common/include -DFUNCNAME=a926Diags_1
${PX_BUILD_2}/a926Diags.c -o ${PX_BUILD}/a926_a926Diags.o
```

```
armcc -c -g -I${a926_1_1}/software/include -
I${a926_1}/common/include  ${PX_BUILD_3}/printToPort.c -o
${PX_BUILD}/printToPort.o
"/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test92
6/verificationEnv/Modelsim/software/a926_Memory/coreDiagnosti
cs/_a926_1/printToPort.c", line 25: Warning: variable 'i'
declared but not used
/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test926
/verificationEnv/Modelsim/software/a926_Memory/coreDiagnostic
s/_a926_1/printToPort.c: 1 warning, 0 errors, 0 serious
errors
armcc -c -g -I${a926_1_1}/software/include -
I${a926_1}/common/include  ${PX_BUILD_3}/pxDiagnostics.c -o
${PX_BUILD}/pxDiagnostics.o
"/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test92
6/verificationEnv/Modelsim/software/a926_Memory/coreDiagnosti
cs/_a926_1/pxDiagnostics.c", line 9: Warning: inventing
'extern int apbmstCoreDiagnostics_1();'
"/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test92
6/verificationEnv/Modelsim/software/a926_Memory/coreDiagnosti
cs/_a926_1/pxDiagnostics.c", line 21: Warning: inventing
'extern int a926Diags_1();'
"/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test92
6/verificationEnv/Modelsim/software/a926_Memory/coreDiagnosti
cs/_a926_1/pxDiagnostics.c", line 32: Warning: inventing
'extern int pxDiagsEnd();'
/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test926
/verificationEnv/Modelsim/software/a926_Memory/coreDiagnostic
s/_a926_1/pxDiagnostics.c: 3 warnings, 0 errors, 0 serious
errors

armlink ${PX_BUILD}/init.o \
${PX_BUILD}/reset_vectors.o \
${PX_BUILD}/interrupt_handler.o \
${PX_BUILD}/apbmst_apbmst.o \
${PX_BUILD}/a926_a926Diags.o \
${PX_BUILD}/printToPort.o \
${PX_BUILD}/pxDiagnostics.o \
 -Xref -elf -map -info totals  -scatter
${PX_BUILD}/linkerCommandFile  -o a926.xARM Linker: (Warning)
"/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test92
6/verificationEnv/Modelsim/software/a926_Memory/linkerCommand
File", line 13 (near column 33) No AREAs selected by
'a926Diags.o(...)'.
ARM Linker: (Warning)
"/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test92
6/verificationEnv/Modelsim/software/a926_Memory/linkerCommand
File", line 23 (near column 35) No AREAs selected by
'apbmst_apbmst.o(...)'.
ARM Linker: (Warning)
"/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test92
6/verificationEnv/Modelsim/software/a926_Memory/linkerCommand
File", line 28 (near column 36) No AREAs selected by
'a926_a926Diags.o(...)'.
ARM Linker: (Warning)
"/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test92
6/verificationEnv/Modelsim/software/a926_Memory/linkerCommand
File", line 29 (near column 31) No AREAs selected by
'a926Diags.o(...)'.
ARM Linker: (Warning)
"/mnt/techpub/work/leritz/Px_work/lab_files/from_jerry/test92
6/verificationEnv/Modelsim/software/a926_Memory/linkerCommand
File", line 30 (near column 39) No AREAs selected by
'interrupt_handler.o(...)'.

Inter-AREA References

init.o(Init2) refers to pxDiagnostics.o(C$$code) for TestMain
init.o(Init2) refers to reset_vectors.o(reset_vectors) for
Boot
init.o(Init2) refers to init.o(Stacks)
init.o(.debug_info) refers to init.o(.debug_abbrev)
init.o(.debug_info) refers to init.o(Init2)
init.o(.debug_info) refers to init.o(.debug_line)
init.o(.debug_line) refers to init.o(Init2)
reset_vectors.o(reset_vectors) refers to init.o(Init2) for
Reset_Handler
reset_vectors.o(reset_vectors) refers to
interrupt_handler.o(C$$code) for interrupt_handler
reset_vectors.o(reset_vectors) refers to
reset_vectors.o(irq_vectors2)
reset_vectors.o(.debug_info) refers to init.o(.debug_abbrev)
reset_vectors.o(.debug_info) refers to
reset_vectors.o(reset_vectors)
reset_vectors.o(.debug_info) refers to
reset_vectors.o(.debug_line)
reset_vectors.o(.debug_line) refers to
reset_vectors.o(reset_vectors)
interrupt_handler.o(C$$code) refers to printToPort.o(C$$code)
for printToPort
interrupt_handler.o(.debug_line) refers to
interrupt_handler.o(C$$code)
interrupt_handler.o(.debug_info) refers to
interrupt_handler.o(.debug_loc)
interrupt_handler.o(.debug_info) refers to
interrupt_handler.o(C$$code)
interrupt_handler.o(.debug_info) refers to
interrupt_handler.o(.debug_line)
interrupt_handler.o(.debug_info) refers to
interrupt_handler.o(.debug_macinfo)
interrupt_handler.o(.debug_info) refers to
interrupt_handler.o(.debug_abbrev)
interrupt_handler.o(.debug_pubnames) refers to
interrupt_handler.o(.debug_info)
interrupt_handler.o(.debug_frame) refers to
interrupt_handler.o(C$$code)
apbmst_apbmst.o(C$$code) refers to printToPort.o(C$$code) for
printToPort
apbmst_apbmst.o(.debug_line) refers to
apbmst_apbmst.o(C$$code)
apbmst_apbmst.o(.debug_info) refers to
apbmst_apbmst.o(.debug_loc)
apbmst_apbmst.o(.debug_info) refers to
apbmst_apbmst.o(C$$code)
apbmst_apbmst.o(.debug_info) refers to
apbmst_apbmst.o(.debug_line)
apbmst_apbmst.o(.debug_info) refers to
apbmst_apbmst.o(.debug_macinfo)
apbmst_apbmst.o(.debug_info) refers to
interrupt_handler.o(.debug_abbrev)
apbmst_apbmst.o(.debug_pubnames) refers to
apbmst_apbmst.o(.debug_info)
apbmst_apbmst.o(.debug_frame) refers to
apbmst_apbmst.o(C$$code)
```

```
a926_a926Diags.o(.debug_line) refers to
a926_a926Diags.o(C$$code)
a926_a926Diags.o(.debug_info) refers to
a926_a926Diags.o(C$$code)
a926_a926Diags.o(.debug_info) refers to
a926_a926Diags.o(.debug_line)
a926_a926Diags.o(.debug_info) refers to
a926_a926Diags.o(.debug_macinfo)
a926_a926Diags.o(.debug_info) refers to
interrupt_handler.o(.debug_abbrev)
a926_a926Diags.o(.debug_pubnames) refers to
a926_a926Diags.o(.debug_info)
a926_a926Diags.o(.debug_frame) refers to
a926_a926Diags.o(C$$code)
printToPort.o(C$$code) refers to printToPort.o(C$$data)
printToPort.o(.debug_line) refers to printToPort.o(C$$code)
printToPort.o(.debug_info) refers to printToPort.o(C$$data)
printToPort.o(.debug_info) refers to printToPort.o(C$$code)
printToPort.o(.debug_info) refers to
printToPort.o(.debug_line)
printToPort.o(.debug_info) refers to
printToPort.o(.debug_macinfo)
printToPort.o(.debug_info) refers to
interrupt_handler.o(.debug_abbrev)
printToPort.o(.debug_pubnames) refers to
printToPort.o(.debug_info)
printToPort.o(.debug_frame) refers to printToPort.o(C$$code)
pxDiagnostics.o(C$$code) refers to pxDiagnostics.o(C$$data)
pxDiagnostics.o(C$$code) refers to printToPort.o(C$$code) for
printToPort
pxDiagnostics.o(C$$code) refers to a926_a926Diags.o(C$$code)
for a926Diags_1
pxDiagnostics.o(C$$code) refers to apbmst_apbmst.o(C$$code)
for apbmstCoreDiagnostics_1
pxDiagnostics.o(.debug_line) refers to
pxDiagnostics.o(C$$code)
pxDiagnostics.o(.debug_info) refers to
pxDiagnostics.o(C$$data)
pxDiagnostics.o(.debug_info) refers to
pxDiagnostics.o(C$$code)
pxDiagnostics.o(.debug_info) refers to
pxDiagnostics.o(.debug_loc)
pxDiagnostics.o(.debug_info) refers to
pxDiagnostics.o(.debug_line)
pxDiagnostics.o(.debug_info) refers to
pxDiagnostics.o(.debug_macinfo)
pxDiagnostics.o(.debug_info) refers to
interrupt_handler.o(.debug_abbrev)
pxDiagnostics.o(.debug_pubnames) refers to
pxDiagnostics.o(.debug_info)
pxDiagnostics.o(.debug_frame) refers to
pxDiagnostics.o(C$$code)

AREA map of root:

Base     Size     Type RO? Name
0        10       DBUG RO  .debug_abbrev from object file
init.o
10       27c      DBUG RO  .debug_abbrev from object file
interrupt_handler.o
0        4c       DBUG RO  .debug_frame from object file
interrupt_handler.o
4c       4c       DBUG RO  .debug_frame from object file
apbmst_apbmst.o
98       3c       DBUG RO  .debug_frame from object file
a926_a926Diags.o
d4       3c       DBUG RO  .debug_frame from object file
printToPort.o
110      64       DBUG RO  .debug_frame from object file
pxDiagnostics.o
0        c4       DBUG RO  .debug_info from object file init.o
c4       cc       DBUG RO  .debug_info from object file
reset_vectors.o
190      120      DBUG RO  .debug_info from object file
interrupt_handler.o
2b0      108      DBUG RO  .debug_info from object file
apbmst_apbmst.o
3b8      f4       DBUG RO  .debug_info from object file
a926_a926Diags.o
4ac      144      DBUG RO  .debug_info from object file
printToPort.o
5f0      164      DBUG RO  .debug_info from object file
pxDiagnostics.o
0        184      DBUG RO  .debug_line from object file init.o
```

```
184      184      DBUG RO  .debug_line from object file
reset_vectors.o
308      198      DBUG RO  .debug_line from object file
interrupt_handler.o
4a0      184      DBUG RO  .debug_line from object file
apbmst_apbmst.o
624      184      DBUG RO  .debug_line from object file
a926_a926Diags.o
7a8      19c      DBUG RO  .debug_line from object file
printToPort.o
944      1b8      DBUG RO  .debug_line from object file
pxDiagnostics.o
0        2c       DBUG RO  .debug_loc from object file
interrupt_handler.o
2c       38       DBUG RO  .debug_loc from object file
apbmst_apbmst.o
64       38       DBUG RO  .debug_loc from object file
pxDiagnostics.o
0        1c0      DBUG RO  .debug_macinfo from object file
interrupt_handler.o
1c0      210      DBUG RO  .debug_macinfo from object file
apbmst_apbmst.o
3d0      1d4      DBUG RO  .debug_macinfo from object file
a926_a926Diags.o
5a4      1c0      DBUG RO  .debug_macinfo from object file
printToPort.o
764      1c0      DBUG RO  .debug_macinfo from object file
pxDiagnostics.o
0        28       DBUG RO  .debug_pubnames from object file
interrupt_handler.o
28       30       DBUG RO  .debug_pubnames from object file
apbmst_apbmst.o
58       24       DBUG RO  .debug_pubnames from object file
a926_a926Diags.o
7c       24       DBUG RO  .debug_pubnames from object file
printToPort.o
a0       30       DBUG RO  .debug_pubnames from object file
pxDiagnostics.o


AREA map of Root:

Base     Size     Type RO? Name
```

```
0        68       CODE RO  reset_vectors from object file
reset_vectors.o
68       58       CODE RO  C$$code from object file
interrupt_handler.o
c0       118      CODE RO  C$$code from object file
apbmst_apbmst.o
1d8      10       CODE RO  C$$code from object file
a926_a926Diags.o
1e8      1c       CODE RO  C$$code from object file
printToPort.o
204      100      CODE RO  C$$code from object file
pxDiagnostics.o
304      84       CODE RO  Init2 from object file init.o


AREA map of Application0x40000000:

Base     Size     Type RO? Name
40000000 4        DATA RW  C$$data from object file
printToPort.o
40000004 4        DATA RW  C$$data from object file
pxDiagnostics.o
40000008 40       DATA RW  Stacks from object file init.o
40000048 800      DATA RW  irq_vectors2 from object file
reset_vectors.o


Image entry point : 0
Entry area : "reset_vectors" from object file reset_vectors.o
                   code   inline   inline  ’const’      RW
0-Init    debug
                  size     data  strings     data     data
data     data
Object totals       540      44     320       0    2120
0    8416


Debug Area Optimization Statistics

Input debug total(excluding low level debug areas)     10976
(10.72Kb)
Output debug total                                      8416
(8.22Kb)
```

```
% reduction                                                23.32%

ARM Linker: finished,  0 informational, 5 warning and 0 error
messages.
```

Appendix D Using the Documentation Generator

Overview

This appendix contains the conceptual and procedural information on the Platform Express Documentation Generator. It contains the following sections:

- About PxDoc
- Installing PxDoc
- Using PxDoc
- Sample of Generated Documentation
- Customizing the Documentation Generator

About PxDoc

The Platform Express Documentation Generator (also called “PxDoc”) is an add-on component that enables you to generate design documentation in HTML. The Documentation Generator accesses design data and generates HTML pages based on the elements of your design.

The generated documentation adheres to the 3.2 HTML standard and is not browser-specific. Using a browser, you can access the generated documentation from any filesystem, or by means of a web server. Because PxDoc is not included with the basic Platform Express package, you must acquire and install the PxDoc package in order to access it from the Platform Express interface.

Installing PxDoc

The PxDoc package is delivered as an additional component library. To enhance your installation of Platform Express with the PxDoc component library, follow these steps:

1. Save your work and exit Platform Express.
2. Download the PxDoc component library from the following URL:

   http://www.mentor.com/platform_ex/download/download.cfm

3. Copy the PxDoc directory to your `$PXPATH` directory.
4. Edit your `$PXPATH` environment variable to include PxDoc.
5. Invoke Platform Express.

Using PxDoc

After you have successfully installed the pxDoc component, the Tools menu is enhanced with the following menu items related to PxDoc:

- Documentation

  Invokes the PxDoc dialog box.

- Mem DocGen

  Automatically builds and displays the memory map documenation.

- Part DocGen

  Automatically builds and displays the parts list documenation.

- Pin DocGen

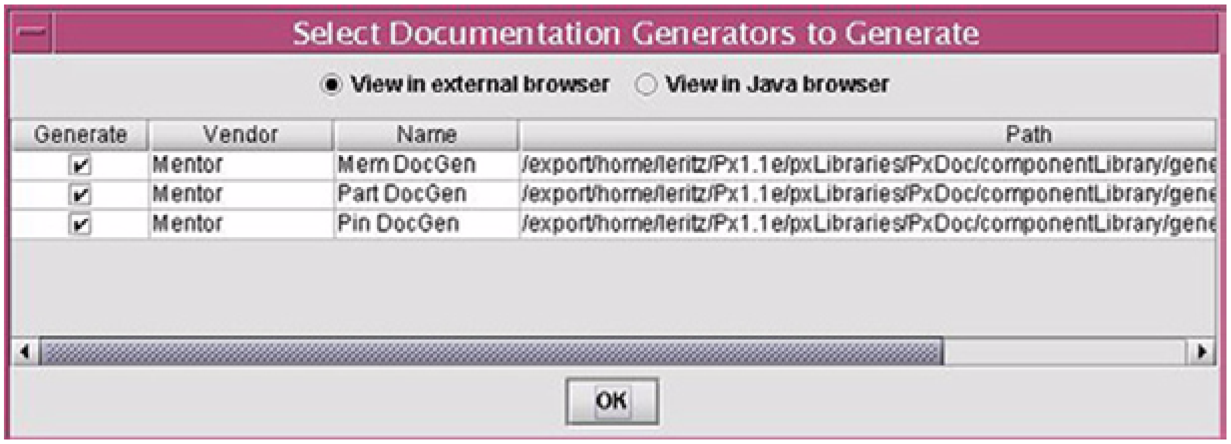
  Automatically builds and displays the pin list documenation.

To generate documentation with PxDoc:

Warning PxDoc automatically overwrites any previous versions of the generated documentatoin. You are not prompted with an "overwrite" warning.

1. Select **Tools** > **Documentation**.

   The Generate PxDoc dialog box appears.



Generate	Vendor	Name	Path
✓	Mentor	Mem DocGen	/export/home/leritz/Px1.1e/pxLibraries/PxDoc/componentLibrary/gene
✓	Mentor	Part DocGen	/export/home/leritz/Px1.1e/pxLibraries/PxDoc/componentLibrary/gene
✓	Mentor	Pin DocGen	/export/home/leritz/Px1.1e/pxLibraries/PxDoc/componentLibrary/gene

Figure D-1. PxDoc Dialog Box

2. In the Generate column, select the desired choices from the list:

- memory map
- parts list
- pin list

By default, all three choices are selected.

1. Select either the **View in external browser** or **View in Java Browser** radio button.

2. Click **OK**.

The status bar reports “Generating Documents.” The output pane displays the path of the generated documentation.

Accessing the Generated Documentation

Immediately after PxDoc generates the HTML files, the generated documentation displays automatically (either with the external browser as defined by the user preferences or with the internal Java browser). If you generated two or more files, the browser or viewer displays the list of the generated HTML files. If you generated only one file, that file is automatically displayed in the brower.

Note If the external browser fails to invoke for any reason, the Java viewer is displayed instead.

Location of Generated Files

If you close the browser or viewer, you can access the documentation from the filesystem. The HTML files are written to the following location:

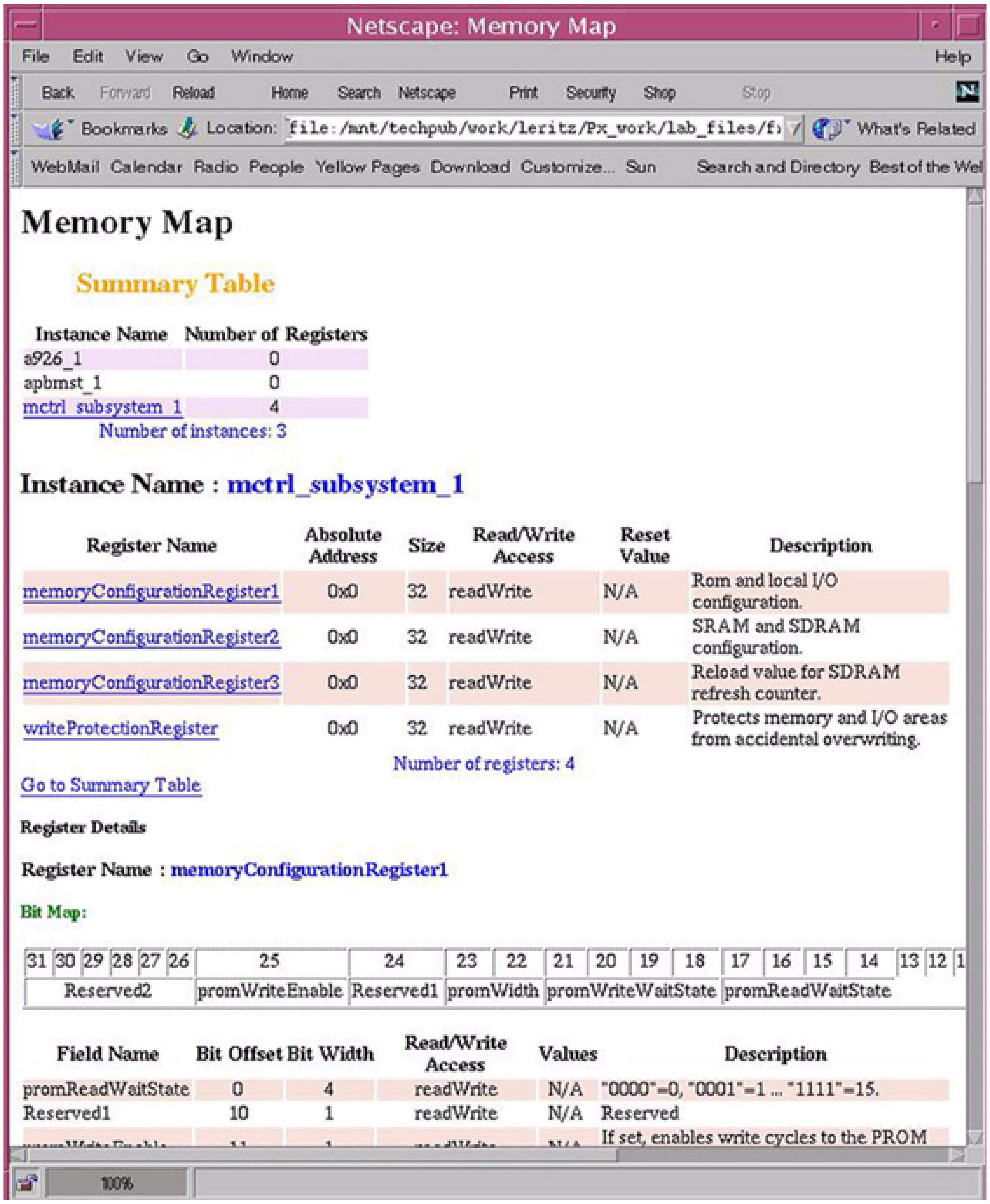
<project_dir>`verificationEnv/`*<Modelsim|Ncsim>*`/doc/`

Table 4-1. Generated HTML Filenames

Part	**Filename**
Memory map	`mem.html`
Parts list	`part.html`
Pin list	`pin.html`

Sample of Generated Documentation

See Figure D-2 and Figure D-2 for sample generated documentation.



Netscape: Memory Map
File Edit View Go Window Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: file:/mnt/techpub/work/leritz/Px_work/lab_files/fi What's Related
WebMail Calendar Radio People Yellow Pages Download Customize... Sun Search and Directory Best of the Wel
100%

Memory Map

Summary Table

Instance Name	Number of Registers
a926_1	0
apbmst_1	0
mctrl_subsystem_1	4

Number of instances: 3

Instance Name : mctrl_subsystem_1

Register Name	Absolute Address	Size	Read/Write Access	Reset Value	Description
memoryConfigurationRegister1	0x0	32	readWrite	N/A	Rom and local I/O configuration.
memoryConfigurationRegister2	0x0	32	readWrite	N/A	SRAM and SDRAM configuration.
memoryConfigurationRegister3	0x0	32	readWrite	N/A	Reload value for SDRAM refresh counter.
writeProtectionRegister	0x0	32	readWrite	N/A	Protects memory and I/O areas from accidental overwriting.

Number of registers: 4

Go to Summary Table

Register Details

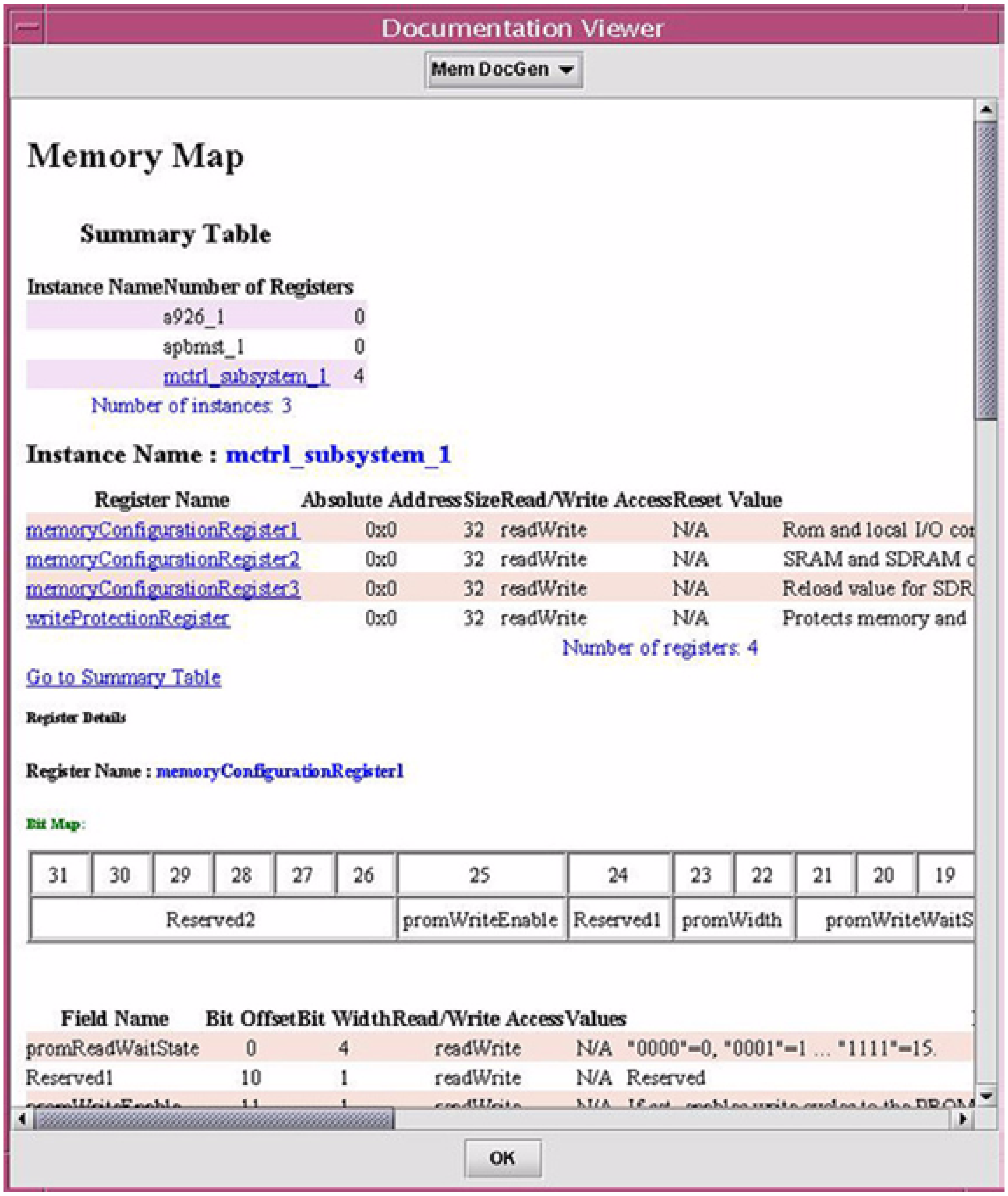
Register Name : memoryConfigurationRegister1

Bit Map:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	1
Reserved2						promWriteEnable	Reserved1	promWidth		promWriteWaitState				promReadWaitState						

Field Name	Bit Offset	Bit Width	Read/Write Access	Values	Description
promReadWaitState	0	4	readWrite	N/A	"0000"=0, "0001"=1 ... "1111"=15.
Reserved1	10	1	readWrite	N/A	Reserved
					If set, enables write cycles to the PROM

Figure D-2. Sample Memory Map documentation, Netscape



Memory Map

Summary Table

Instance Name	Number of Registers
a926_1	0
apbmst_1	0
mctrl_subsystem_1	4

Number of instances: 3

Instance Name : mctrl_subsystem_1

Register Name	Absolute Address	Size	Read/Write Access	Reset Value	
memoryConfigurationRegister1	0x0	32	readWrite	N/A	Rom and local I/O co
memoryConfigurationRegister2	0x0	32	readWrite	N/A	SRAM and SDRAM c
memoryConfigurationRegister3	0x0	32	readWrite	N/A	Reload value for SDR
writeProtectionRegister	0x0	32	readWrite	N/A	Protects memory and

Number of registers: 4

Go to Summary Table

Register Details

Register Name : memoryConfigurationRegister1

Bit Map:

31	30	29	28	27	26	25	24	23	22	21	20	19
Reserved2						promWriteEnable	Reserved1	promWidth		promWriteWaitS		

Field Name	Bit Offset	Bit Width	Read/Write Access	Values	
promReadWaitState	0	4	readWrite	N/A	"0000"=0, "0001"=1 ... "1111"=15.
Reserved1	10	1	readWrite	N/A	Reserved
promWriteEnable	11	1	readWrite	N/A	If set, enables write cycles to the PROM

Figure D-3. Sample Memory Map Documentation, Java

Customizing the Documentation Generator

PxDoc can be customized in terms of style and output to deliver the desired documentation. You must be a licensed Integrator to customize the Documentation Generator. Consult the *Platform Express Integrator's Guide* for further information.

Appendix E Contacting the SupportCenter

Technical support is available to all customers who have a support contract. Before contacting technical support, gather the following information:

- License ID
- Platform and version
- Product and version
- Any test files
- Exact steps or procedures causing the problem

North America Support

In North America, refer to the online SupportCenter at:

`http://www.mentor.com/supportnet/`

This website provides technical support information, including a solutions database, application notes, FAQs, patches, and an online form for submitting a technical support request.

International Support

For customer support outside North America, contact your local Mentor Graphics sales organization, or refer to the following website:

`http://www.mentor.com/supportnet/support_offices.html`

Product Licensing Assistance

Refer to the SupportCenter to obtain new product licenses or to address questions or issues related to product licensing. Include the following information with your request:

- Your name.
- Company name, address, telephone number, and email address.
- List of products you want to license or upgrade.
- Name of server(s) that you plan to use (if different from your current server, including any new licensing server configurations).
- The host ID numbers of client and licensing server workstations for node-locked licenses.
- The host ID number of the licensing server workstation for all floating licenses.

International customers should contact their local Mentor Graphics sales office for product licensing assistance.

Glossary

Address block

A two-dimensional block placed inside an address space. It has a two dimensional offset component defining its location in relation to the base of the address space in units of address and bit offset. It also has a size component to indicate how much of the address space it occupies in dimensions of address and bit width.

Address space

A two-dimensional magnitude in units of address and bit width. May be occupied by smaller address blocks.

Bus

A path used for transmitting signals from any of several sources to any of several destinations.

Bus Bridge

A Platform Express component having a master bus interface and a slave bus interface where signals arriving on the slave interface are translated into signals on the master interface. This allows slaves connected to the master interface to respond to signals from the master of the bus to which the slave interface is connected.

Component Library

A collection of files and directories that can be used by Platform express to create and validate SoC designs. Component libraries can include IP (components), bus definitions, decoder templates, generators and generator chains.

Interrupt Controller

A hardware module responsible for collecting interrupt requests from different components in the design and selects the proper one(s) to send to the processor.

Interrupt Service Routine (ISR)

An Interrupt Service Routine (ISR) is a software function that is called and run in response to an interrupt received by the processor.

Literal Pool

Constant and string information built into compiled source code. In most cases, the constants and strings in the code are required to be relatively close to the instruction code in the Memory Map. (Literal pools are specific to ARM processors.)

Memory Map

A representation of an address space and the address blocks it contains. Memory maps are usually represented graphically.

Scatter Loader

A module whose function is to arrange the software compiled objects in different load and execution regions.

Test Bench

(Also known as a “test harness” or “test fixure”). An HDL model used to verify the correct behavior of a hardware model. A test bench generates simulation input stimuli for the model test and applies this input stimuli to the model under test. Then it applies this input stimuli and collates output responses. Finally, it compares output responses with expected values and (possibly) automatically gives a pass or fail indication.

Trademark Information

Mentor Graphics Trademarks

Third-Party Trademarks

End-User License Agreement

IMPORTANT - USE OF THIS SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE SOFTWARE

This license is a legal "Agreement" concerning the use of Software between you, the end-user, either individually or as an authorized representative of the company purchasing the license, and Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Mentor Graphics (Singapore) Private Limited, and their majority-owned subsidiaries ("Mentor Graphics"). USE OF SOFTWARE INDICATES YOUR COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. If you do not agree to these terms and conditions, promptly return or, if received electronically, certify destruction of Software and all accompanying items within 10 days after receipt of Software and receive a full refund of any license fee paid

END-USER LICENSE AGREEMENT

1. **GRANT OF LICENSE**. The software programs you are installing, downloading, or have acquired with this Agreement, including any updates, modifications, revisions, copies, and documentation ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics or its authorized distributor grants to you, subject to payment of appropriate license fees, a nontransferable, nonexclusive license to use Software solely: (a) (in machine-readable, object-code form; (b) for your internal business purposes; and (c) on the computer hardware or at the site for which an applicable license fee is paid, or as authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Mentor Graphics' then-current standard policies, which vary depending on Software, license fees paid or service plan purchased, apply to the following and are subject to change: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be communicated and technically implemented through the use of authorization codes or similar devices); (c) eligibility to receive updates, modifications, and revisions; and (d) support services provided. Current standard policies are available upon request.

2. **ESD SOFTWARE**. If you purchased a license to use embedded software development (ESD) Software, Mentor Graphics or its authorized distributor grants to you a nontransferable, nonexclusive license to reproduce and distribute executable files created using ESD compilers, including the ESD run-time libraries distributed with ESD C and C++ compiler Software that are linked into a composite program as an integral part of your compiled computer program, provided that you distribute these files only in conjunction with your compiled computer program. Mentor Graphics does NOT grant you any right to duplicate or incorporate copies of Mentor Graphics' real-time operating systems or other ESD Software, except those explicitly granted in this section, into your products without first signing a separate agreement with Mentor Graphics for such purpose.

3. **BETA CODE**

   3.1. Portions or all of certain Software may contain code for experimental testing and evaluation ("Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to you a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and your use of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form.

   3.2. If Mentor Graphics authorizes you to use the Beta Code, you agree to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. You will contact Mentor Graphics periodically during your use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of your evaluation and testing, you will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

3.3. You agree that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceives or makes during or subsequent to this Agreement, including those based partly or wholly on your feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this subsection shall survive termination or expiration of this Agreement.

4. **RESTRICTIONS ON USE**. You may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. You shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. You shall not make Software available in any form to any person other than your employer's employees and contractors, excluding Mentor Graphics' competitors, whose job performance requires access. You shall take appropriate action to protect the confidentiality of Software and ensure that any person permitted access to Software does not disclose it or use it except as permitted by this Agreement. Except as otherwise permitted for purposes of interoperability as specified by the European Union Software Directive or local law, you shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive from Software any source code. You may not sublicense, assign or otherwise transfer Software, this Agreement or the rights under it without Mentor Graphics' prior written consent. The provisions of this section shall survive the termination or expiration of this Agreement.

5. **LIMITED WARRANTY**

5.1. Mentor Graphics warrants that during the warranty period Software, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Software will meet your requirements or that operation of Software will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. You must notify Mentor Graphics in writing of any nonconformity within the warranty period. This warranty shall not be valid if Software has been subject to misuse, unauthorized modification or installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND YOUR EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF SOFTWARE TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF SOFTWARE THAT DOES NOT MEET THIS LIMITED WARRANTY, PROVIDED YOU HAVE OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) SOFTWARE WHICH IS LOANED TO YOU FOR A LIMITED TERM OR AT NO COST; OR (C) EXPERIMENTAL BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

5.2. THE WARRANTIES SET FORTH IN THIS SECTION 5 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO SOFTWARE OR OTHER MATERIAL PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

6. **LIMITATION OF LIABILITY**. EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE STATUTE OR REGULATION, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL

THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT PAID BY YOU FOR THE SOFTWARE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER.

7. **LIFE ENDANGERING ACTIVITIES**. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF SOFTWARE IN ANY APPLICATION WHERE THE FAILURE OR INACCURACY OF THE SOFTWARE MIGHT RESULT IN DEATH OR PERSONAL INJURY. YOU AGREE TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE, OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH SUCH USE.

8. **INFRINGEMENT**

   8.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against you alleging that Software infringes a patent or copyright in the United States, Canada, Japan, Switzerland, Norway, Israel, Egypt, or the European Union. Mentor Graphics will pay any costs and damages finally awarded against you that are attributable to the claim, provided that you: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the claim; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the claim.

   8.2. If an infringement claim is made, Mentor Graphics may, at its option and expense, either (a) replace or modify Software so that it becomes noninfringing, or (b) procure for you the right to continue using Software. If Mentor Graphics determines that neither of those alternatives is financially practical or otherwise reasonably available, Mentor Graphics may require the return of Software and refund to you any license fee paid, less a reasonable allowance for use.

   8.3. Mentor Graphics has no liability to you if the alleged infringement is based upon: (a) the combination of Software with any product not furnished by Mentor Graphics; (b) the modification of Software other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of Software as part of an infringing process; (e) a product that you design or market; (f) any Beta Code contained in Software; or (g) any Software provided by Mentor Graphics' licensors which do not provide such indemnification to Mentor Graphics' customers.

   8.4. THIS SECTION 8 STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS AND YOUR SOLE AND EXCLUSIVE REMEDY WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT BY ANY SOFTWARE LICENSED UNDER THIS AGREEMENT.

9. **TERM**. This Agreement remains effective until expiration or termination. This Agreement will automatically terminate if you fail to comply with any term or condition of this Agreement or if you fail to pay for the license when due and such failure to pay continues for a period of 30 days after written notice from Mentor Graphics. If Software was provided for limited term use, this Agreement will automatically expire at the end of the authorized term. Upon any termination or expiration, you agree to cease all use of Software and return it to Mentor Graphics or certify deletion and destruction of Software, including all copies, to Mentor Graphics' reasonable satisfaction.

10. **EXPORT**. Software is subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products, information about the products, and direct products of the products to certain countries and certain persons. You agree that you will not export in any manner any Software or direct product of Software, without first obtaining all necessary approval from appropriate local and United States government agencies.

Index (cont.)

Index (cont.)

Index (cont.)

Index (cont.)