AMBA Designer (FD001)

Revision: r2p0

User Guide

ARM®

ARM DUI 0333C

AMBA Designer (FD001)
User Guide

Release Information

The following changes have been made to this document.

Release history

Date	Issue	Confidentiality	Change
18 May 2006	A	Non Confidential	First release for revision r0p0
19 May 2006	B	Non Confidential	Minor technical corrections
20 September 2006	C	Non Confidential	Updated for r2p0

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
AMBA Designer (FD001) User Guide

List of Tables
AMBA Designer (FD001) User Guide

List of Figures
AMBA Designer (FD001) User Guide

Preface

This preface introduces the *AMBA Designer (FD001) User Guide*. It contains the following sections:

About this guide

This is the *User Guide* for *AMBA Designer (FD001)*.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this guide, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

Intended audience

This guide is written for system designers and system integrators who are using the AMBA Designer tool to either build a PrimeCell® *High-Performance Matrix* (HPM) based design or configure the PrimeCell components that AMBA Designer supports. System verification engineers who want to verify the generated RTL, for the PrimeCell components, also require this guide. It is assumed that the audience is familiar with the RTL generation, synthesis, and verification processes of an ASIC design flow.

Using this guide

This guide is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for a description of the AMBA Designer tool.

Chapter 2 *Installation*

Read this chapter for a description of the installation process for AMBA Designer.

Chapter 3 *Using AMBA Designer*

Read this chapter for a description about how to invoke AMBA Designer, configure its preferences, and operate it in batch mode.

Chapter 4 *Configuring an HPM*

Read this chapter for a description of how to create and configure AMBA 3 bus interconnects using the HPM.

Chapter 5 *Configuring a DMC*

Read this chapter for a description of how to configure a PrimeCell *Dynamic Memory Controller* (DMC).

Chapter 6 *Configuring an SMC*

Read this chapter for a description of how to configure a PrimeCell *Static Memory Controller* (SMC).

Chapter 7 *Creating a System Model*

Read this chapter for a description of how to create a system model using the 4x3 interconnect that Chapter 4 *Configuring an HPM* describes. The system is then simulated.

Chapter 8 *RTL Design Flow*

Read this chapter for a description of how to use the RTL Design Flow Manager.

Glossary Read the Glossary for definitions of terms used in this guide.

Conventions

Conventions that this guide can use are described in:

- *Typographical*
- *Signals* on page xiv
- *Numbering* on page xv.

Typographical

The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.

< and > Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example:

- `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
- The Opcode_2 value selects which register is accessed.

Signals

The signal conventions are:

Signal level The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means HIGH for active-HIGH signals and LOW for active-LOW signals.

Lower-case n Denotes an active-LOW signal.

Prefix A Denotes global *Advanced eXtensible Interface* (AXI) signals.

Prefix AR Denotes AXI read address channel signals.

Prefix AW Denotes AXI write address channel signals.

Prefix B Denotes AXI write response channel signals.

Prefix C Denotes AXI low-power interface signals.

Prefix H Denotes *Advanced High-performance Bus* (AHB) signals.

Prefix P Denotes *Advanced Peripheral Bus* (APB) signals.

Prefix R Denotes AXI read data channel signals.

Prefix W Denotes AXI write data channel signals.

Numbering

The numbering convention is:

<size in bits>'<base><number>

This is a Verilog method of abbreviating constant numbers. For example:

- 'h7B4 is an unsized hexadecimal value.
- 'o7654 is an unsized octal value.
- 8'd9 is an eight-bit wide decimal value of 9.
- 8'h3F is an eight-bit wide hexadecimal value of `0x3F`. This is equivalent to b00111111.
- 8'b1111 is an eight-bit wide binary value of b00001111.

Further reading

This section lists publications by ARM, and by third parties.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets, addenda, and the Frequently Asked Questions list.

ARM publications

This guide contains information that is specific to AMBA Designer. See the following documents for other relevant information:

- *AMBA AXI Protocol Specification* (ARM IHI 0022)
- *AMBA 3 APB Protocol Specification* (ARM IHI 0024)
- *AMBA 3 AHB-Lite Protocol Specification* (ARM IHI 0033)
- *ARM1176JZF-S Technical Reference Manual* (ARM DDI 0301)
- *PrimeCell High-Performance Matrix (PL301) Technical Reference Manual* (ARM DDI 0397)
- *PrimeCell High-Performance Matrix (PL301) Integration Manual* (ARM DII 0157)
- *PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual* (ARM DDI 0331)
- *PrimeCell Static Memory Controller (PL350 series) Technical Reference Manual* (ARM DDI 0380)

- *SoC Designer User Guide* (ARM DUI 0316)
- *ARM FLEXlm License Management Guide*.

Feedback

ARM welcomes feedback on AMBA Designer (FD001) and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier giving:

- the product name
- a concise explanation of your comments.

Feedback on this guide

If you have any comments on this guide, send e-mail to `errata@arm.com` giving:

- the title
- the number
- the relevant page number(s) to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces AMBA Designer. It contains the following sections:

1.1 Overview

AMBA Designer is a *Graphical User Interface* (GUI) tool that enables users to:

- configure PrimeCell components
- create complex AMBA interconnects
- create a model of the component, or interconnect. AMBA Designer can then use this in a transaction level modeling environment to produce profiling and monitoring data.
- generate RTL and testbenches for PrimeCell components.

Figure 1-1 on page 1-3 shows a typical design flow for AMBA Designer and the interaction between AMBA Designer and SoC Designer Simulator. After a system model is created, AMBA Designer enables you to invoke SoC Designer Simulator, which performs the system simulation. This provides you with detailed visibility of system transactions that enable you to optimize the system architecture early on in the design process.

If the initial simulation results are unsuitable then an iterative process commences, where the component or interconnect is modified in AMBA Designer and the simulation is run again in SoC Designer Simulator.

When the simulation results are satisfactory, AMBA Designer produces the RTL Verilog files and the associated testbenches for verifying the RTL.

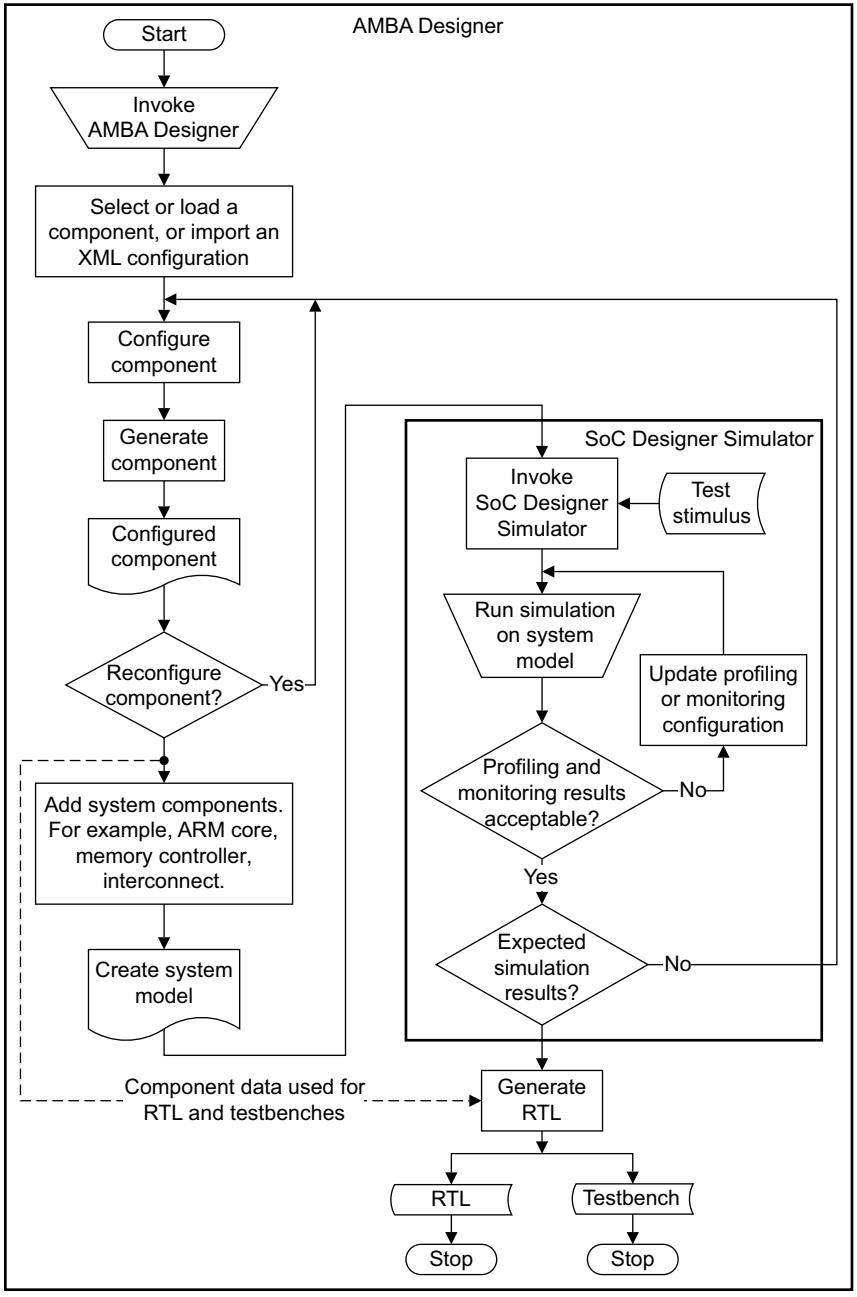


Figure 1-1 Typical design flow

1.2 Product revisions

This section describes the differences in functionality between product revisions of the program:

r0p0-r2p0 Contains the following differences in functionality:

- Enables you to configure the following devices:

  DMC (PL340) r1p0

  See Chapter 5 *Configuring a DMC* for more information.

  SMC (PL350 series) r0p0

  See Chapter 6 *Configuring an SMC* for more information.
- Updated to support version r1p0 of the HPM (PL301).
- The program does not generate a `.ini` file for each design configuration. The `.mxp` configuration file now contains the information that was previously stored in the `.ini` file.

  Note

  The AMBA Designer package contains a .ini to .mxp conversion program named `ini2mxp`. See *Converting the configuration file format* on page 4-49 for more information about this program.
- Adds a HPM reconfiguration option that enables you to add or remove masters, or slaves, from a previously generated interconnect. See *Reconfiguring an interconnect* on page 4-51 for more information.

Chapter 2
Installation

This chapter describes the steps required to install the program on your computer. It contains the following sections:

2.1 UNIX: Linux and Solaris

You can install AMBA Designer on the following operating systems:

- Red Hat® Enterprise Linux® 3.0
- Sun Solaris™ 8, 32bit.

The UNIX package is a file named:

`AMBA-Designer_`*`version`*`_`*`platform`*`.tgz`

where:

version Tool version information.

platform `Linux` or `Solaris`.

You must download this package from the ARM *ESL* (Electronic System Level) Support website at:

`https://esl-support.arm.com`

The installation process is described in:

- *Installation*
- *Configuration* on page 2-6.

2.1.1 Installation

Before installing AMBA Designer, the following conditions apply:

- you must install SoC Designer or have it installed on your system
- install or have installed on your system the AMBA2 and AMBA3 model libraries. During a multi-user installation, the installer requests the location of these model libraries.

Note

The AMBA Designer Release Note provides information about the versions required for:

- SoC Designer
- AMBA2 model library
- AMBA3 model library.

To install AMBA Designer:

1. Unzip the `.tgz` file in a directory of your choice.
2. Unpack the resulting `.tar` file.

The AMBA-Designer_*version_platform* directory is created. This contains the Linux and Solaris directories, with their associated install scripts and files.

3. Depending on your operating system, navigate to the appropriate directory:

   Linux AMBA-Designer_*version*_Linux/Install/Linux

   Solaris AMBA-Designer_*version*_Solaris/Install/Solaris

4. Enter the command ./install_AMBA_Designer.sh. On some systems you must enter /usr/bin/sh install_AMBA_Designer.sh
   - A welcome message is displayed.
   - A prompt displays the version of SoC Designer that is required.

5. Press **Enter** on the keyboard. The installer then displays:

```
Do you wish to:
  1. Install AMBA Designer for a single user
  2. Install AMBA Designer in a central location for use by multiple users

Please enter 1 or 2
```

6. Enter the appropriate option number and press **Enter** on the keyboard. The installer searches for SoC Designer on your device.

   Depending on the option that is chosen, one of the following message displays:

   1. Installing for single user

   2. Installing centrally for multiple users

   Note

   The installation aborts if SoC Designer is not found.

7. When the installer finds SoC Designer it then displays output similar to:

```
Existing installation of SoC Designer found. $MAXSIM_HOME is currently set
to:
      /home/user/ARM/SoCDesigner_version/

Checking installed version...

SoC Designer version found: version

This release of AMBA Designer requires version version or later.

If the installed version is sufficient, please press enter to continue.
Otherwise, please press CTRL-C to quit the installation process, then
install the latest version of SoC Designer and re-run the AMBA Designer
installation.

Please press enter to continue.
```

8. Press **Enter** on the keyboard to continue with the installation. A warning message displays if a previous installation of AMBA Designer is found, otherwise the installation continues at step 10.

```
**WARNING**
An existing installation of AMBA Designer has been detected at:
 /home/user/ARM/SoCDesigner_version/AMBADesigner

Do you wish to:
  1. Over-write your current AMBA Designer installation
  2. Quit the installation process (no files will be installed)

Please enter 1 or 2
```

9. Enter the appropriate option number and press **Enter** on the keyboard. Depending on the option that is chosen, one of the following message displays:

   1. `Over-writing current AMBA Designer installation`
   Installation continues at step 12.

   2. `AMBA Designer Installation Aborted`
   The installation process aborts.

10. The installer then displays:

```
AMBA Designer requires TCL version (or later) in order to run. By default
the scripts will execute 'tclsh' from the user's path.  Do you want to
specify a fixed location for the tclsh executable instead?
Please enter y or n
```

11. Enter `y` or `n` and press **Enter** on the keyboard. If you enter `n` the installation continues at step 12. If you enter `y` then the following message displays:

```
Please enter the full path to the tclsh executable.  E.g.:
    /eda/tools/opensource/tcl/8.4/bin/tclsh
```

    Enter the path to your `tclsh` executable and press **Enter** on the keyboard.

12. AMBA Designer file installation now starts. The displayed message depends on your choice of installation during step 6. The message displayed is described in:

    Single-user install Installation continues at step 13.

    Multi-user install Installation continues at step 14.

13. For a single-user installation the message displayed is similar to:

```
Installing AMBA Designer runtime files...

Installing AMBA Designer documentation...

Updating TCL scripts to call tcl executable from:
    <path to tclsh>
```



```
Generating AMBA Designer setup.csh file...

------------------------------------------------------------
AMBA Designer installation is complete.
------------------------------------------------------------
```

For a single-user installation then the file install process is complete.

14. For a multi-user installation the message displayed is similar to:

```
Installing AMBA Designer runtime files...

Installing AMBA Designer documentation...

Updating TCL scripts to call tcl executable from:
    <path to tclsh>
Setting global execute permission on scripts for all users...

Setting global read permission on files for all users...

Do you want to associate this installation of AMBA Designer with specific
installations of the following RealView MaxLib libraries?  Note that the
libraries must already be installed in your system
    AMBA3
    AMBA2

It is recommended that you do associate this installation with specific
installations of the RealView MaxLib libraries.  If you do not, then
each user will need to set-up their personal library selection, which
may lead to inconsistencies between users

Proceed with association?  Please press y or n (y)
```

15. Enter `y` or `n` and press **Enter** on the keyboard. Depending on the option that is chosen, one of the following message displays:

    `n` `Not proceeding with library association`

    Installation continues at step 16.

    `y` `Proceeding with library association...`

    ```
    Please enter the path to the AMBA3 installation, or press enter
    to skip
    ```

    Enter the path to your AMBA3 installation and press **Enter** on the keyboard. If the `maxlib.conf` file is found the following message displays:

    ```
    Adding <AMBA3_PATH>/etc/maxlib.conf to adcanvas

    Please enter the path to the AMBA2 installation, or press enter
    to skip
    ```

Enter the path to your AMBA2 installation and press **Enter** on the keyboard. If the `maxlib.conf` file is found the following message displays:

```
Adding <AMBA2_PATH>/etc/maxlib.conf to adcanvas
```

16. After the library association stage, the messages displayed are:

```
Generating AMBA Designer setup.csh file...

------------------------------------------------------------
AMBA Designer installation is complete.
------------------------------------------------------------
```

For a multi-user installation then the file install process is complete.

2.1.2 Configuration

After the file installation process completes the installer then displays information about how to run AMBA Designer.

To configure the shell:

1. The environment setup message is displayed:

```
You now have to run the following command in order to setup your
environment to correctly run the AMBA Designer tool set.

For c-shell (csh, tcsh) users:

> source /home/user/ARM/SoCDesigner_version/AMBADesigner/etc/setup.csh

NOTE: Bourne shell is not supported in this release of AMBA Designer.
```

If you are the root user then place the appropriate source command into the global logins of all users who require access to the software. If you are a sole user then place the source command in your own local login file.

To invoke the program:

2. The run message is displayed:

```
To run AMBA Designer use the command:
    adcanvas
To run RealView SoC Designer (with AMBA Designer features disabled) use
the command:
    sdcanvas
```


2.2 Windows

You can install AMBA Designer on Microsoft® Windows XP Professional.

The Windows package is a file named:

```
AMBA-Designer_version_Win32.zip
```

where *version* is the tool version information. You must download this package from the ARM ESL Support website at:

`https://esl-support.arm.com`

2.2.1 Installation

Before installing AMBA Designer, the following conditions apply:

- You must install SoC Designer or have it installed on your system.
- Install or have installed on your system the AMBA2 and AMBA3 model libraries. During installation, the installer requests the location of these model libraries.
- Prior to installing the latest version, ARM recommends that you uninstall any previous versions of AMBA Designer.

Note

The AMBA Designer Release Note provides information about the versions required for:

- SoC Designer
- AMBA2 model library
- AMBA3 model library.

To install the program:

1. Unzip the `.zip` file in a directory of your choice.
2. Double-click on AMBA-Designer-Lite_*version*_Win32.exe, the AMBA Designer installation executable.
3. Follow the procedures of the InstallShield Wizard.

If you require additional models or add-on software installed, see *Installing additional component models* on page 2-8 or *Obtaining and installing licenses* on page 2-10.

2.3 Installing additional component models

You must install licensed or third-party models separately because these are not included in the AMBA Designer tool package. For example, to create the 4x3 interconnect that *Example 4x3 interconnect* on page 4-4 describes and integrate it in a system as Chapter 7 *Creating a System Model* describes then you must install the following libraries:

- AMBA2 model library
- AMBA3 model library
- ARM11 model library
- AMBA Designer Support library.

To install a model library, you must add the `maxlib.conf` configuration files located in the `etc` directory of each model package to the SoC Designer library search list.

To add the `maxlib.conf` files required for AMBA Designer:

1. Start AMBA Designer with the following method:

   UNIX Invoke AMBA Designer from the console with the command:
   ```
   > adcanvas
   ```

   Windows Click on the Windows **Start** button and then using the **Start** menu click on **All Programs → ARM → AMBA Designer**.

2. Select **File → Preferences** from the Main menu.

3. Select **Model Library** from the **General** category in the SoC Designer Preferences dialog box.

4. Click on **Add ...** located in the Additional maxlib.conf Files panel. This opens the Add Model Library Search Path dialog box.

5. Click on **Browse...** in the Add Model Library Search Path dialog box. This opens a file browser dialog box.

6. Navigate to the directory containing the new models and add the `maxlib.conf` file by selecting the file and clicking on the **Open** button.

7. Click on **OK** in the Add Model Library Search Path dialog box. The `maxlib.conf` file is added to the list of files in the Additional maxlib.conf Files: panel.

8. Repeat steps 4 to 7 for the other library files.

9. Click on **OK & Save** in the SoC Designer Preferences dialog box to save the changes.

10. You must install the license file for the new model by adding the license file to the license path list.

Figure 2-1 shows an example of the SoC Designer Preferences dialog box.

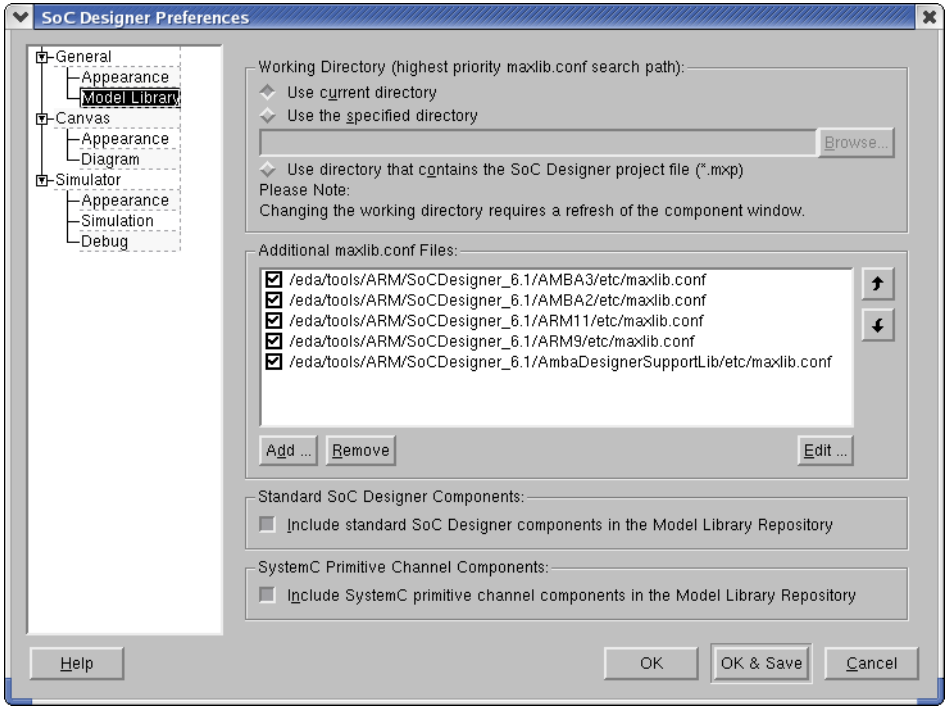


Figure 2-1 Preferences dialog with list of Model Library configuration files

2.4 Obtaining and installing licenses

This section describes the steps to install the license for AMBA Designer on your workstation. It contains the following sections:

- *Licensing overview*
- *Obtaining a floating license* on page 2-11
- *Installing the FLEXlm utilities on a license server* on page 2-12
- *Invoking AMBA Designer for the first time after installation* on page 2-16
- *Windows environment variable* on page 2-16.

Note

See the documentation supplied with your third-party add-on or model for additional information about installing the licenses for those products.

2.4.1 Licensing overview

You must obtain licenses to:

- run AMBA Designer
- use licensed ARM models in your system designs
- use licensed ARM or third-party add-ons
- use licensed third-party models, for example, third-party processor cores.

You can obtain a temporary license from ARM that enables you to evaluate AMBA Designer and the standard models for a period of 30 days.

License management consists of the following components:

- a license file that lists the software you are entitled to run
- software that validates your application against the license file.

ARM provides a floating license to enable you to use AMBA Designer.

Note

Node-locked licenses are not available for AMBA Designer.

Floating license

Using a floating license does not require a license on the local workstation. The application communicates with license management software located on a license server. The server runs the validation software, `lmgrd` and `armlmd`, and reads the contents of a license file that is stored on the server. The floating license is also locked to the server, by the hostid or network card id of the server.

The application can be installed on multiple workstations and each of them communicates with the license server to validate the shared license. There might be a limit on how many client workstations can simultaneously use the shared license.

Licensing a floating application requires:

1. Identifying the network card or disk id of the server, or servers, that will run the validation software.
2. After the product serial number is sent to you, you must generate the license file by using the ARM Web Licensing Portal.
3. The FLEXlm license management demon, `lmgrd`, and the ARM license management demon, `armlmd`, must be installed on the server.
4. The demon software on the server must be started when the server boots or started manually whenever the application on the client workstation requires validation of the license file.
5. For Windows workstations, setting an environment variable on the client workstation to point to the server that will validate the license file for the client application.

2.4.2 Obtaining a floating license

Once your order for AMBA Designer has been processed then ARM provides a product serial number. You must use this serial number when generating your license.

To generate your licence:

1. Using a browser, navigate to the ARM Web Licensing Portal at:

   `http://license.arm.com`

   A welcome page is displayed.

Note

See the ARM Web Licensing Portal User Guide for more information about using the licensing portal. The welcome page provides links to enable you to access the document.

2. To login to the portal, enter your User Name and Password. The ARM License Management Home Page is displayed.

On the ARM License Management Home Page:

3. Enter the AMBA Designer serial number in the Serial Number field.

4. Click on the **Get License** button. The Host ID Information Page is displayed.

On the Host ID Information Page:

5. Enter the hostid information for your server.

6. Click on the **Generate** button. The Confirmation Page is displayed.

On the Confirmation Page:

7. To generate the license, click on the **Confirm** button. The generated license is displayed.

To save the license to your device:

8. Click on the **Save to File** button. Save the file, `license.dat`, to your device.

2.4.3 Installing the FLEXlm utilities on a license server

A floating license enables many users to use AMBA Designer based on a central license stored on a server. The license is locked to the hostid of the server.

Contact your system administrator to set up the FLEXlm license server if one is not already available. Provide your system administrator with:

- the information in this chapter
- the license utilities provided in the following directory:

  UNIX *SoC_Designer_installation_directory*`/bin`

  Windows *SoC_Designer_installation_directory*`/Tools`

The followings sections describe where to place the license file and the installation of the FLEXlm utilities on either:

- *Solaris or Linux server*
- *Windows Server* on page 2-14.

Solaris or Linux server

Additional information about setting up a license server is in the *ARM FLEXlm License Management Guide*.

To create a license server:

1. Obtain a floating license for AMBA Designer. Follow the instructions that *Obtaining a floating license* on page 2-11 describes.

2. Place the floating licence, `license.dat`, in a directory on each of the servers that you are using.

Note

You might have to assign a different name to the licence text file, for example `license_ad.dat`, if a `license.dat` file exists for a previous installation of SoC Designer.

ARM recommends that you put the license file in the same directory as your license utilities.

Note

You might need to make some changes to the SERVER line in the license file:

a. Replace *this_host* with the corresponding server name.

b. Enter the host id name.

c. Enter the server port number. For ARM licenses this is usually `8224`.

Your modified server line must use the following format:

```
SERVER <servername> <host ID> <port>
```

3. Copy the following files to a directory, for example `/FLEXlm`, on the license server:
   - `lmgrd`
   - `armlmd`
   - `lmutil`
   - `makelinks.sh`
   - `rmlinks.sh`

   Note

   The destination directory must be on your `PATH`.

4. On the license server machine, navigate to your license utilities directory and type:

   ```
   sh ./makelinks.sh
   ```

5. To start the license server software on a UNIX server, navigate to the directory containing the license server software and type:

   ```
   nohup lmgrd -c license_file_name -l logfile_name
   ```

   Where:

   license_file_name

   Specifies the fully qualified path name of the license file.

   logfile_name

   Specifies the fully qualified path name to a log file.

6. Since `lmgrd` does not require root privileges, it should be started by a non-privileged user, not by root.
7. After you have started the license server, you can type, for example:

   ```
   tail -f logfile_name
   ```

   to see the most recent output from the license server.
8. If you are installing a multiple server system, repeat these steps for all of the license servers.

Windows Server

Additional information about setting up a license server is in the *ARM FLEXlm License Management Guide*.

Note

ARM does not support the use of a Windows 95, 98 or Me machine as a license server.

To create a license server:

1. Obtain a floating license for AMBA Designer. Follow the instructions that *Obtaining a floating license* on page 2-11 describes.
2. Place the floating licence, `license.dat`, in a directory on each of the servers that you are using.

   Note

   You might have to assign a different name to the licence text file, for example `license_ad.dat`, if a `license.dat` file exists for a previous installation of SoC Designer.

   ARM recommends that you put the license file in the same directory as your license utilities.

   Note

   You might need to make some changes to the SERVER line in the license file:
   a. Replace *`this_host`* with the corresponding server name.
   b. Enter the host id name.
   c. Enter the server port number. For ARM licenses this is usually `8224`.

   Your modified server line must use the following format:

   ```
   SERVER <servername> <host ID> <port>
   ```


3. Copy the following files to the `C:\FLEXlm` directory on the license server:
   - `lmgrd.exe`
   - `armlmd.exe`
   - `lmutil.exe`
   - `lmtools.exe`
4. Add `C:\FLEXlm` to the `PATH`.
5. Start the `lmtools.exe` utility and select the **Configuration using Services** option.
6. Enter the paths that specify the required files, or click the **Browse** button to locate and select the files. You must specify paths for:
   - executable file, `lmgrd.exe`
   - license file, `license.dat`
   - log file, `debug.log`
7. If you require the server software to start running automatically whenever the server is powered up, click the **Use Services** check box, then click the **Start Server at Power Up** check box.
8. Click **Save Service**.
9. When prompted, confirm that you want to save the FLEXlm License Manager service.
10. Click the **Start/Stop/Reread** tab, and ensure that the FLEXlm License Manager service is selected.
11. Click **Start Server** to start running the license server software.

    Note

    You can also start the license server software from a command line. From the directory where the license server software is installed, type:

    `lmgrd -c` *`license_file_name`* `-l` *`logfile_name`*

    Where:

    `license_file_name`
    : Specifies the fully qualified path name of the license file.

    `logfile_name`
    : Specifies the fully qualified path name to a log file.

12. After you have started the license server, you can inspect the log file by using a text editor, such as Notepad.

13. If you are installing a multiple server system, repeat these steps for all of the license servers.

2.4.4 Invoking AMBA Designer for the first time after installation

After installing AMBA Designer, the first time the program is run then it might prompt you to enter the location of the license file. You must enter the address of your server, at the prompt.

2.4.5 Windows environment variable

When the program is invoked on a Windows workstation, the program uses the `ARMLMD_LICENSE_FILE` environment variable to enable it to locate the license file on the server.

To update the environment variable:

1. Select **Control Panel** from the Start menu, to open the Control Panel window.
2. Double-click on **System** from the Control Panel, to open the System Properties dialog.
3. Click on the **Advanced** tab.
4. Click on **Environment Variables** to display the Environment Variables dialog.
5. Using the System variables pane, select the `ARMLMD_LICENSE_FILE` variable.
6. Click on **Edit** to display the Edit System Variable dialog.
7. In the Variable value field, enter the location on your server that contains the license file.
8. Click on **OK** to close the Edit System Variable dialog.
9. Click on **OK** to close the Environment Variables dialog.
10. Click on **OK** to close the System Properties dialog.

Chapter 3
Using AMBA Designer

This chapter describes how to invoke the program, configure the preferences, and use the program in batch mode. It contains the following sections:

- *Invoking AMBA Designer* on page 3-2
- *AMBA Designer preferences* on page 3-3
- *Batch mode operation* on page 3-5.

Note

The AMBA Designer graphical user interface is based on SoC Designer. See the *SoC Designer User Guide* for general information about the menu system and window layout.

3.1 Invoking AMBA Designer

To start AMBA Designer:

Windows To invoke SoC Designer Canvas on Windows, either:

- Click on the Windows **Start** button and then using the **Start** menu click on **All Programs → ARM → AMBA Designer**. This is the usual way of starting AMBA Designer.
- Double-click on the AMBA Designer icon located on the desktop.
- Drag an MXP file, with extension `.mxp`, on to the AMBA Designer icon.
- Double-click on an MXP file. The file associations must have been set by the installer.

Linux, Solaris

On UNIX, invoke AMBA Designer from the console with the command:

```
> adcanvas filename
```

Table 3-1 lists the supported command line options for AMBA Designer.

Table 3-1 Command line options

Option	Example use	Description
`-v`, `--version`	`adcanvas --version` `adcanvas -v`	Prints the version status of AMBA Designer and SoC Designer without starting the tool.
`-n`, `--nomaxlib`	`adcanvas --nomaxlib` `adcanvas -n`	Starts AMBA Designer without loading any components.
`-b`	`adcanvas -b -pl`*xxx* *filename* `[options...]`	Runs AMBA Designer in batch mode. See *Batch mode operation* on page 3-5 for more information.
filename	`adcanvas` *mySystem*`.mxp`	Starts AMBA Designer with the specified file open. The specified file must be a `.mxp` file.

3.2 AMBA Designer preferences

You can configure the directories that AMBA Designer uses for reading and writing files and the location of the `maxlib.conf` file, by using the General Preferences dialog.

To open the General Preferences dialog box:

1. Click on the **Designer Prefs** button that is located on the Main Toolbar, as Figure 3-1 shows. This opens the General Preferences dialog box.

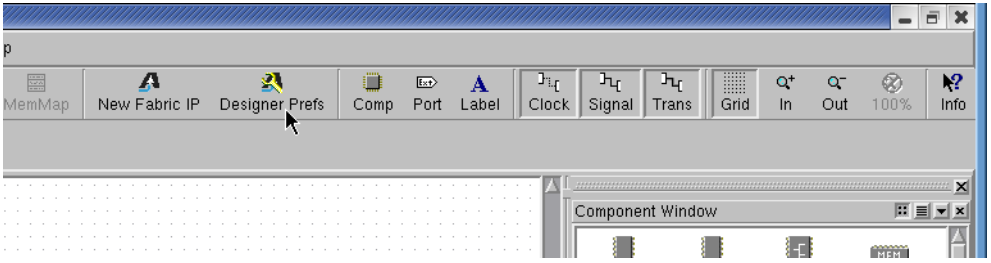


Figure 3-1 Designer Prefs selection

Figure 3-2 shows the General Preferences dialog box.

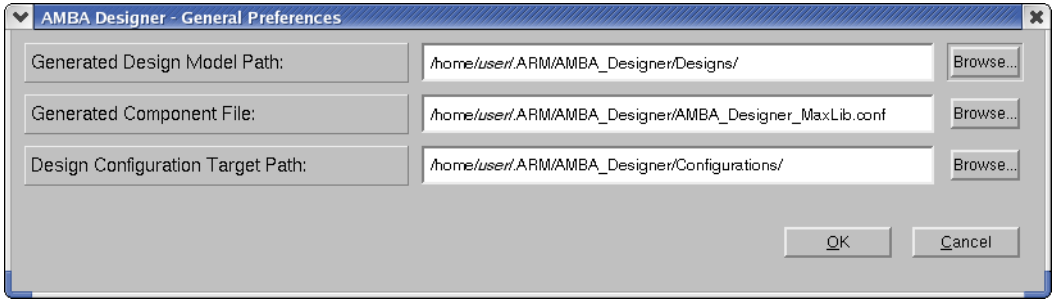


Figure 3-2 General Preferences dialog

The preferences are:

Generated Design Model Path:

After you configure a PrimeCell component, the program generates the component model and saves it in this directory.

During the RTL generation process, the program creates in this directory the relevant RTL directory structure for the PrimeCell component. The program uses this directory structure to save the RTL files that it generates.

Generated Component File:

This file contains the MaxLib entry information for generated models that have been added to the component library. See the *SoC Designer User Guide* for more information about `maxlib.conf` files.

Design Configuration Target Path:

This directory contains the design configuration files. These files contain the parameter, address, and sparse information that AMBA Designer uses to generate the PrimeCell models.

To change the defined Paths or File:

1. Either:
   - enter the new directory path in the associated field.
   - click on **Browse...**, to open a file browser dialog box. Using this dialog box:
     1. Navigate to the appropriate directory or file.
     2. Select the appropriate directory or file.
     3. Click on **OK** to close the file browser dialog box.
2. Click on **OK** to save the preferences and close the General Preferences dialog box.

AMBA Designer saves the preferences in the `AMBA_Designer_Prefs.ini` file. Figure 3-3 shows the location of the `.ini` file.

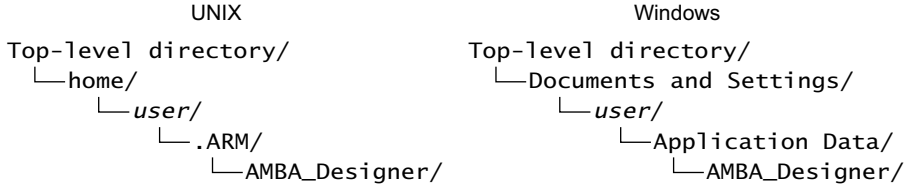


Figure 3-3 Location of AMBA Designer preferences file

Note

On UNIX, the `.ini` file also contains some other preferences, that the RTL Design Flow Manager uses during simulation and synthesis.

3.3 Batch mode operation

AMBA Designer supports batch mode operation, when the program runs on a UNIX operating system. This enables you to start regression tests without having to use the graphical user interface.

When the Generate RTL process of the RTL Design Flow Manager is initially invoked, it creates an XML configuration file for the configured PrimeCell component. The Generate RTL process then creates the RTL based on the information contained in the XML file. The program creates the XML file in a directory under the `Designs` directory. Figure 8-4 on page 8-6 shows the location of the XML file, when the program creates RTL for an HPM.

During batch mode, AMBA Designer uses the XML configuration file to perform any of the following functions:

- Generate RTL
- Generate RTL and simulate
- Generate RTL and synthesis
- Generate RTL, simulate, and synthesis.

You can also specify the simulator that AMBA Designer uses and whether it uses OVL assertions and LEC.

3.3.1 Batch mode usage and commands

Invoke AMBA Designer from the console with the following command:

```
> adcanvas -b -plxxx filename [options...]
```

Where:

`-b` AMBA Designer runs in batch mode.

`-plxxx` The model number of the configured PrimeCell. For example:

- `-pl301` when using an HPM
- `-pl340` when using a DMC.

`filename` Name of the XML file.

options... Table 3-2 on page 3-6 lists the supported batch mode options.

Table 3-2 Batch mode options

Option	Example use	Description
`-gen`, `-generate`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -gen`	Generate RTL.
`-sim`, `-simulate`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -sim`	Generate RTL and simulate.
`-synth`, `-synthesise`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -synth`	Generate RTL and synthesize.
`-all`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -all`	Generate RTL, simulate, and synthesize. This is the default, if no other batch mode options are specified.
OVL options		
`-ovl`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -ovl`	Enables OVL assertions during simulation.[a]
`-noovl`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -noovl`	Disables OVL assertions during simulation.[a]
LEC options		
`-lec`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -lec`	Enables LEC during synthesis.[b]
`-nolec`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -nolec`	Disables LEC during synthesis.[b]
Simulator selection options		
`-mti`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -mti`	Selects the Mentor ModelSim simulator.[c]
`-nc`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -nc`	Selects the Cadence NCsim simulator.[c]
`-vcs`	`adcanvas -b -pl`*`xxx`* *`mySystem`*`.xml -vcs`	Selects the Synopsys VCS simulator.[c]

a. This option overrides the Use OVL Assertions flag in the RTL Design Flow Manager Preferences.
b. This option overrides the Run LEC flag in the RTL Design Flow Manager Preferences.
c. This option overrides the Simulator setting in the RTL Design Flow Manager Preferences.

Chapter 4
Configuring an HPM

This chapter describes how to create and configure interconnects using the HPM. It contains the following sections:

4.1 About the HPM

The HPM is a highly configurable AMBA 3 bus interconnect. The interconnect consists of a central AXI cross-bar switch, known as the AXI bus matrix, and AMBA infrastructure components to enable connection to other AMBA devices.

Figure 4-1 shows an example of the top-level hierarchy of an interconnect.

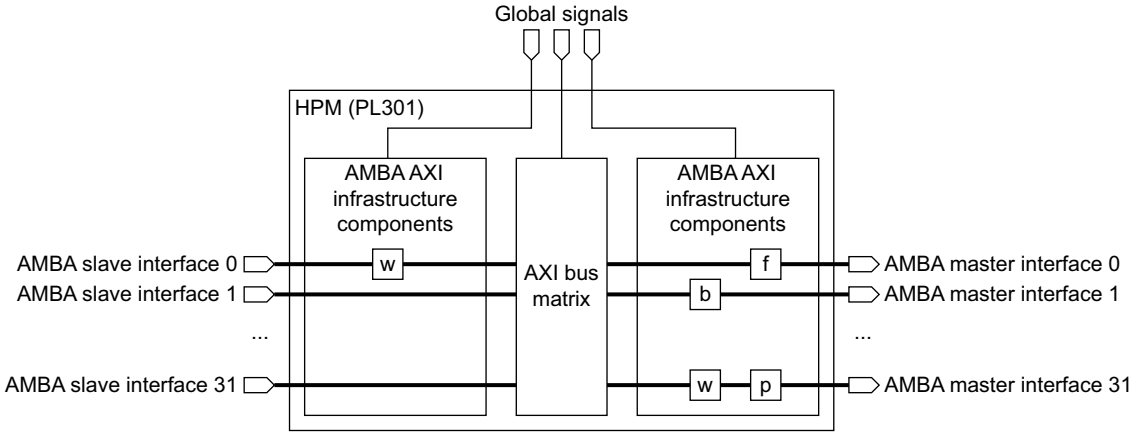


Figure 4-1 Example top-level hierarchy of the HPM

In Figure 4-1 the AMBA AXI infrastructure components types are:

b buffering component

f frequency conversion component

p data bus protocol conversion component

w data bus width conversion component.

This combination of IP provides support for other AMBA bus protocols including AHB-Lite and APB.

See the *PrimeCell High-Performance Matrix (PL301) Technical Reference Manual* for more information about the HPM.

Note

AMBA Designer does not support use with the *PrimeCell AXI Configurable Interconnect (PL300).*

4.1.1 HPM parameters

AMBA Designer enables the configuration of the following HPM parameters:

- number of slave interfaces, to a maximum of 32
- number of master interfaces, to a maximum of 32
- one of the following cyclic dependency schemes:
  - SingleSlave
  - UniqueID
  - HigherRank
  - Hybrid
  - SlavePerID.
- address widths from 32-bit to 64-bit, inclusive
- additional register stage for the address decoder
- ID width of slave interface
- naming of master and slave interfaces
- routing data width of 32, 64, or 128 bits
- AXI and AHB-Lite master interface data widths of 32, 64, or 128 bits
- AXI and AHB-Lite slave interface data width of 32, 64, or 128 bits
- initial arbitration order for slave interface
- registered I/O for slave and master interfaces
- interface protocols, AXI, AHB-Lite, and APB
- clock domain crossing
- write issuing capability of the master interface
- write issuing depth of the master interface
- read acceptance capability for the slave interface
- write acceptance capability for the slave interface
- **REMAP** bus selection.

4.2 Example 4x3 interconnect

This example describes how to create and configure an HPM that consists of four slave interfaces and three master interfaces. The components that attach to the HPM are:

- an ARM1176JZF® processor core that connects to the four HPM slave interfaces
- a DMC that connects to two of the HPM master interfaces
- an AXI-compliant memory that connects to the remaining HPM master interface.

Chapter 7 *Creating a System Model* describes how you connect these components to the 4x3 interconnect and use the system model in a transaction level modeling environment to produce profiling data.

Note

When referring to the size of an interconnect, this document refers first to the number of slave interfaces, that masters connect to, followed by the number of master interfaces, that slaves connect to. For example, a 4x3 interconnect would comprise of four slave interfaces and three master interfaces and would therefore connect to four masters and three slaves.

To create the 4x3 interconnect then you must install the following libraries:

- AMBA2 model library
- AMBA3 model library
- AMBA Designer Support library.

See *Installing additional component models* on page 2-8 for information about how to install these libraries.

Creating the 4x3 example interconnect involves the following steps:

- *Instantiating the AXI-core bus matrix*
- *Configuring the Master Params* on page 4-8
- *Configuring the Slave Params* on page 4-9
- *Creating the address map* on page 4-12
- *Configuring the sparse interconnect* on page 4-18
- *Generating the interconnect* on page 4-23
- *Adding the interconnect to the component library* on page 4-26
- *RTL generation or system modeling* on page 4-27.

4.2.1 Instantiating the AXI-core bus matrix

Start AMBA Designer by using one of the following methods:

UNIX Enter `adcanvas` in a console window and press **Enter** on the keyboard.

Windows Click on the Windows **Start** button and then using the **Start** menu click on **All Programs → ARM → AMBA Designer**.

After starting AMBA Designer, you have an empty Diagram Window, with the tab name of **Untitled** in the lower-left corner, as Figure 4-2 shows.

Figure 4-2 Untitled tab

To instantiate the bus matrix:

1. Click on the **New Fabric IP** button on the Main Toolbar, as Figure 4-3 shows. This opens the Configure new PrimeCell IP dialog box.

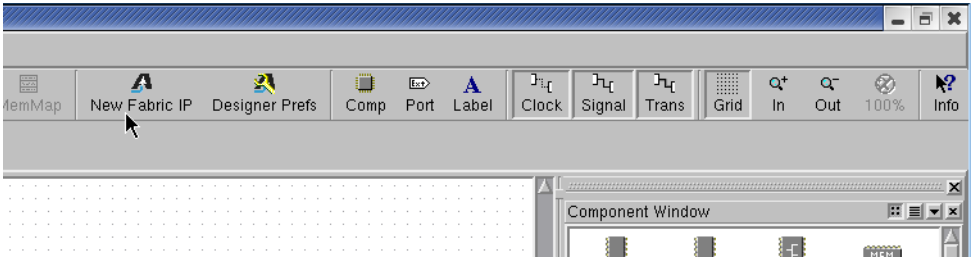


Figure 4-3 New Fabric IP selection

2. In the Configure new PrimeCell IP dialog box:
   a. Click on the row containing a PrimeCell name of PL301, as Figure 4-4 shows.

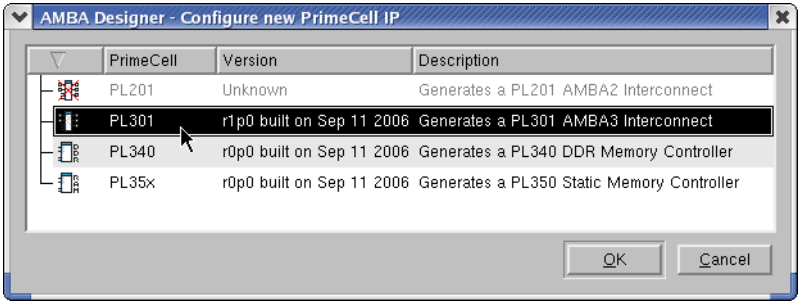


Figure 4-4 Configure new PrimeCell IP dialog

   b. Click on **OK** to close the dialog box and open the New PL301 AXI-core Bus Matrix dialog box.

3. In the New PL301 AXI-core Bus Matrix dialog box:
   a. Enter 4 in the Number of Masters field, as Figure 4-5 shows.
   b. Enter 3 in the Number of Slaves field.

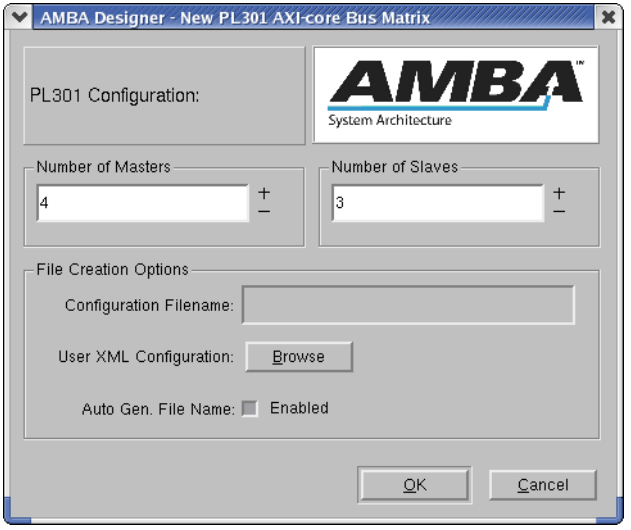


Figure 4-5 New PL301 AXI-core Bus Matrix dialog

   c. Click on **OK** to close the dialog box.

Note

By default, AMBA Designer creates an autogenerated filename. To create your own filename then using the New PL301 AXI-core Matrix dialog box:

a. Deselect the **Auto Gen. File Name** check box, to disable the automatic generation of filenames.

b. Enter your chosen filename in the Configuration Filename field.

   The characters you enter in this field must comply with the file naming conventions for the file system where the design file is located. Using a reserved character from the file system, might cause the program to terminate unexpectedly.

The Diagram Window displays the HPM as Figure 4-6 on page 4-7 shows.

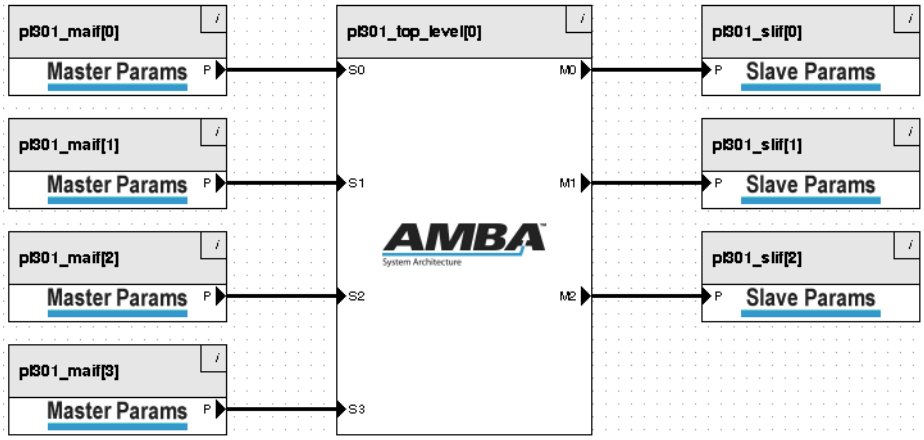


Figure 4-6 Unconfigured 4x3 HPM

This bus matrix shows the HPM component in the center, with the four masters on the left-hand side and the three slaves on the right-hand side. Table 4-1 lists the port interfaces for this unconfigured HPM.

Table 4-1 Port interfaces of the 4x3 HPM

Port name	Function	Connects externally to:
S0	Slave interface 0	Master, ARM1176JZF data bus
S1	Slave interface 1	Master, ARM1176JZF DMA bus
S2	Slave interface 2	Master, ARM1176JZF instruction bus
S3	Slave interface 3	Master, ARM1176JZF peripheral bus
M0	Master interface 0	Slave, DMC peripheral bus
M1	Master interface 1	Slave, DMC AXI bus
M2	Master interface 2	Slave, AXI-compliant memory, for system ROM

Note

When AMBA Designer creates the filename then the tab name changes to **PL301_*xxx*_4x3**, where:

- **PL301** is the prefix the program assigns to all the HPM files it creates
- ***xxx*** is a random generated sequence of alphanumeric characters

- **4x3** is the size of the HPM, that is, the number of slave interfaces followed by the number of master interfaces.

An asterisk (*) after the **PL301_*xxx*_4x3** name indicates that there are unsaved changes, either from an auto-saved file or the saved file.

4.2.2 Configuring the Master Params

The master and slave interfaces of the HPM are all programmed with their default values. The following sections describe how to change the Master Params that then configure the corresponding HPM slave interfaces:

- *HPM slave interfaces 0, 1, and 2 configuration*
- *HPM slave interface 3 configuration*.

HPM slave interfaces 0, 1, and 2 configuration

These three interfaces connect to the ARM1176JZF processor's data, DMA, and instruction buses. The default parameter values are correct for these interfaces and therefore no changes are required.

HPM slave interface 3 configuration

This interface connects to the ARM1176JZF peripheral bus so you must change the data_width parameter from 64 to 32. To make this change:

1. Double-click on the pl301_maif[3] component to open the Edit Parameters dialog box.

In the Edit Parameters dialog box:

2. Set the data_width parameter to 32, as Figure 4-7 on page 4-9 shows.
3. Click on **OK** to close the dialog box.

Note

To edit parameter values, double-click on the current value setting. For example, to change the data_width parameter from the default value of 64, double-click on 64 and select a new value from the drop-down list.

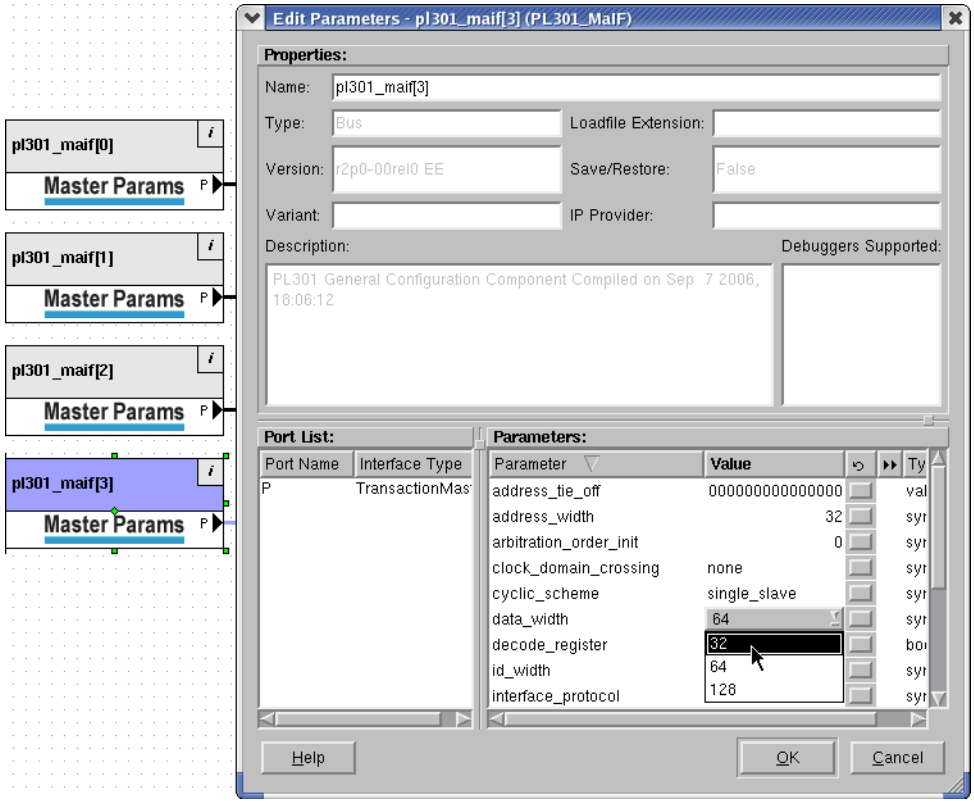


Figure 4-7 Configuring slave interface 3

4.2.3 Configuring the Slave Params

The three master interfaces of the HPM connect to two slaves, the DMC and the AXI-compliant memory. The following sections describe how to change the Slave Params that then configure the corresponding HPM master interfaces:

- *HPM master interface 0 configuration*
- *HPM master interface 1 configuration* on page 4-11
- *HPM master interface 2 configuration* on page 4-12.

HPM master interface 0 configuration

This master interface connects to the peripheral bus on the DMC and you must make the following changes:

data_width Change from 64 to 32.

interface_protocol Change from AXI to APB.

x_apb_slot_00_name

Change from (apb_slot0) to DMC.

Note

When master and slave parameters are changed, AMBA Designer verifies that the change does not violate an AMBA protocol. Therefore, the sequence you use to change the parameters is important.

For example, an APB interface using a data bus of 64 bits is not permitted. If the interface_protocol changes to APB when the data_width is 64, AMBA Designer displays a warning message in the Output Window and the interface_protocol remains unchanged.

To make these changes:

1. Double-click on the pl301_slif[0] component to open the Edit Parameters dialog box.

In the Edit Parameters dialog box:

2. Set the data_width parameter to 32.
3. Set the interface_protocol parameter to apb.
4. Enter DMC in the x_apb_slot_00_name Value field, as Figure 4-8 on page 4-11 shows.

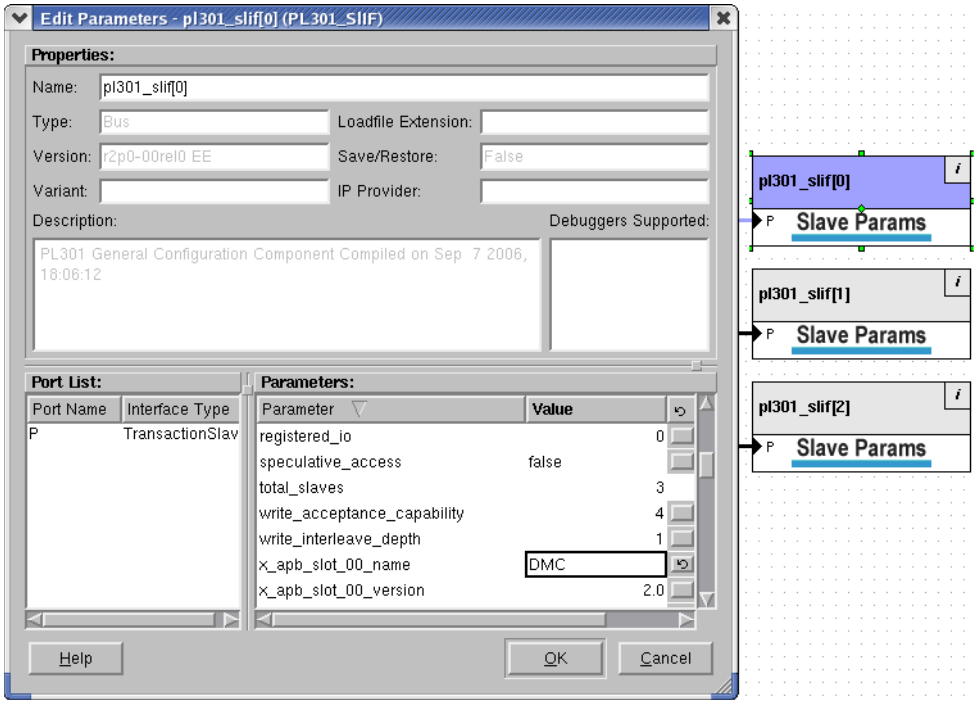


Figure 4-8 Configuring master interface 0

5. Click on **OK** to close the dialog box.

HPM master interface 1 configuration

This interface connects to the AXI bus on the DMC and you must make the following change:

registered_io Change from 0 to 1.

To make this change:

1. Double-click on the pl301_slif[1] component to open the Edit Parameters dialog box.

In the Edit Parameters dialog box:

2. Set the registered_io parameter to 1, as Figure 4-9 on page 4-12 shows.

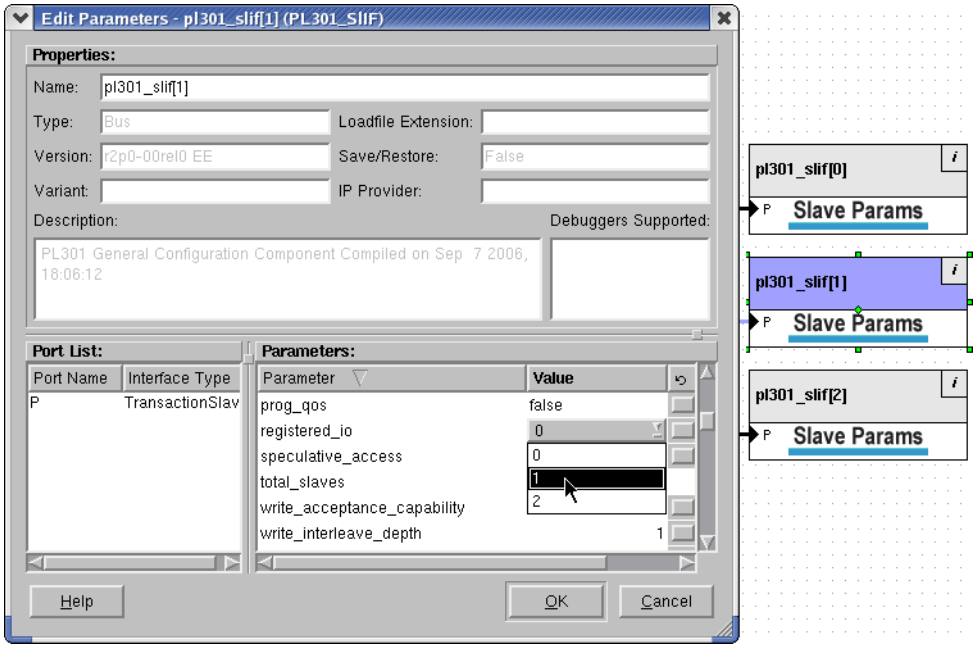


Figure 4-9 Configuring master interface 1

3. Click on **OK** to close the dialog box.

HPM master interface 2 configuration

This interface connects to the AXI bus on the AXI-compliant memory. The default parameter values are correct and therefore no configuration changes are required.

4.2.4 Creating the address map

To create the address map you must use the Memory Map Editor dialog. You can open the Memory Map Editor for the interconnect by either:

- double-clicking on the pl301_top_level[0] component
- using **MemMap**:
  1. Click on the pl301_top_level[0] component to select the component.
  2. Click on the **MemMap** button on the Main Toolbar.
- using the context menu:
  1. Right-click on the pl301_top_level[0] component to display the context menu.

2. Select **AMBA Designer → PL301 Interconnect Address Map** from the context menu, as Figure 4-10 shows.

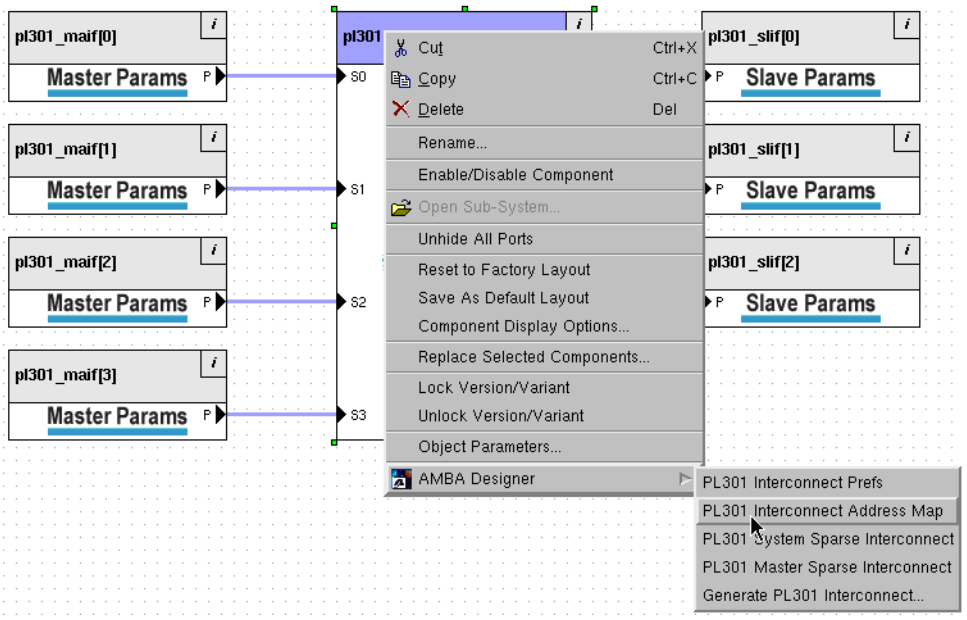


Figure 4-10 Accessing the address map

All these methods open the Memory Map Editor dialog box, as Figure 4-11 shows.

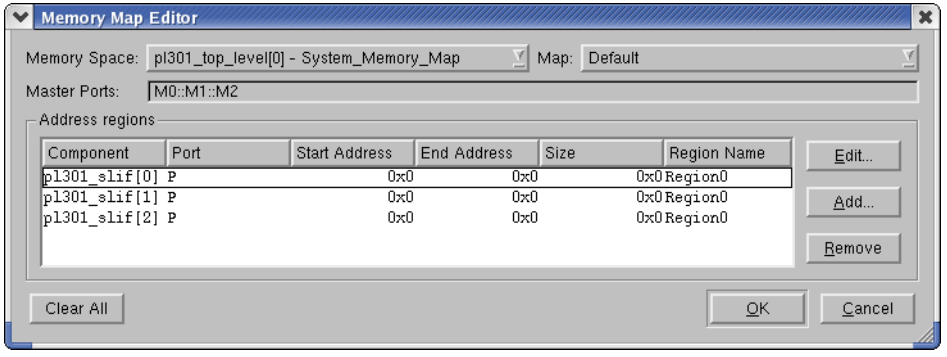


Figure 4-11 Memory Map Editor dialog

In the Memory Map Editor dialog box:

3. Click on **Edit...** to open the Edit Address Region dialog box. Figure 4-12 on page 4-14 shows the dialog box that is displayed.

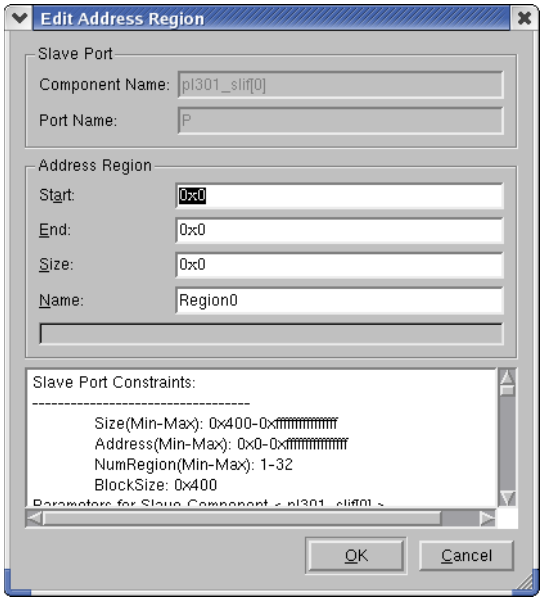


Figure 4-12 Edit Address Region dialog for slave 0

In the Edit Address Region dialog box:

4. Enter `0x10110000` in the Start field, to set the base address for the pl301_slif[0] component to 0x10110000.
5. Enter `0x10000` in the Size field, as Figure 4-13 on page 4-15 shows.

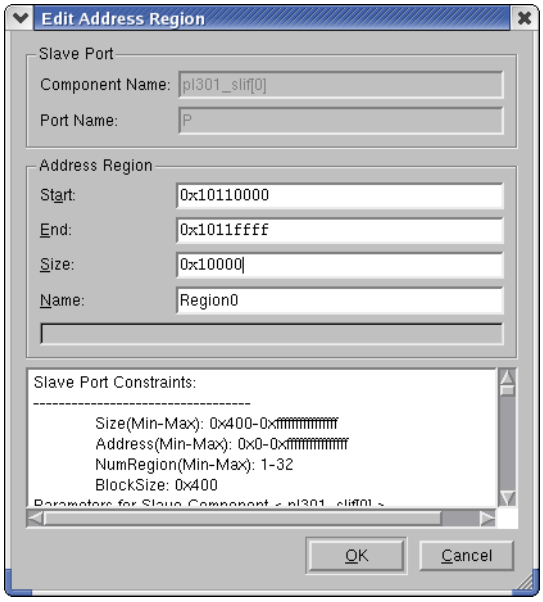


Figure 4-13 Address map for slave 0

6. Click on **OK** to add this address region and close the dialog box.

Note

See *Address map* on page 4-55 for more information about the Memory Map Editor dialog box.

The Memory Map Editor displays the new information for pl301_slif[0]. To update the information for the next slave:

7. In the Memory Map Editor dialog box, click on the Component field that contains the value, pl301_slif[1].

8. Click on **Edit...** to open the Edit Address Region dialog box as Figure 4-12 on page 4-14 shows.

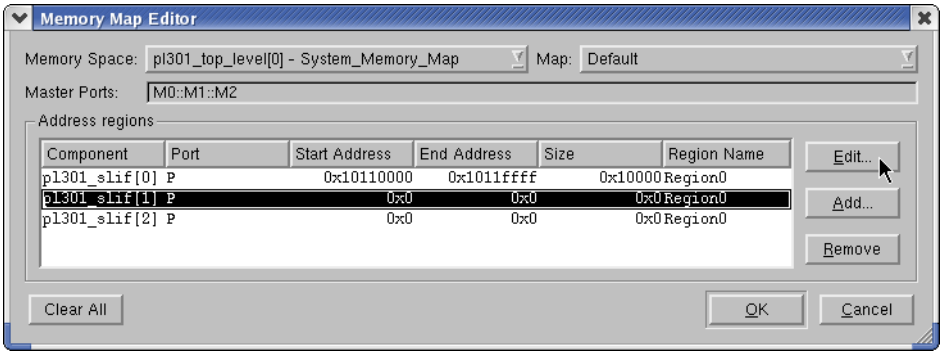


Figure 4-14 Accessing the address map editor for slave 1

In the Edit Address Region dialog box:

9. Enter `0x01000000` in the Start field, to set the base address for the pl301_slif[1] component to 0x01000000.

10. Enter `0x04000000` in the Size field, as Figure 4-15 shows.

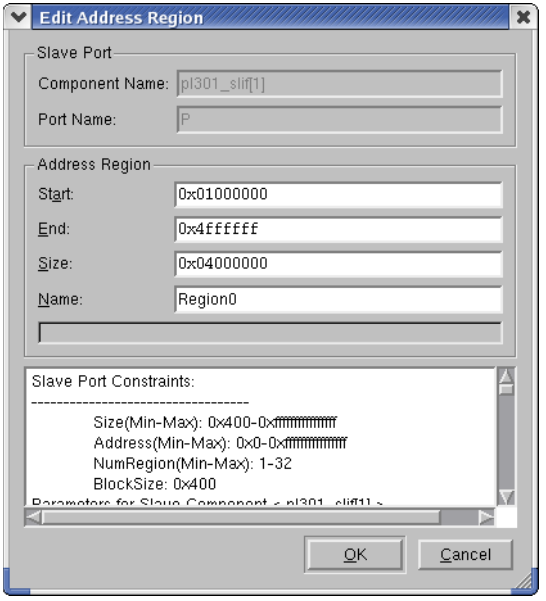


Figure 4-15 Address map for slave 1

11. Click on **OK** to add this address region and close the dialog box.

The Memory Map Editor displays the new information for pl301_slif[1]. To update the information for the next slave:

12. In the Memory Map Editor dialog box, click on the Component field that contains the value, pl301_slif[2].

13. Click on **Edit...** to open the Edit Address Region dialog box.

In the Edit Address Region dialog box:

14. If the Start field does not contain `0x0` then enter `0x0` in this field. This sets the base address for the pl301_slif[2] component to 0x00000000.

15. Enter `0x01000000` in the Size field, as Figure 4-16 shows.

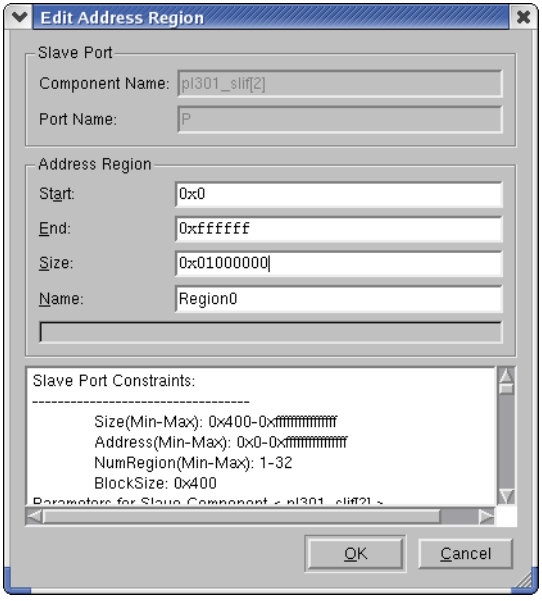


Figure 4-16 Address map for slave 2

16. Click on **OK** to add this address region and close the dialog box.

The address map for all of the slaves is now configured. Figure 4-17 on page 4-18 shows the fully configured Memory Map Editor dialog box.

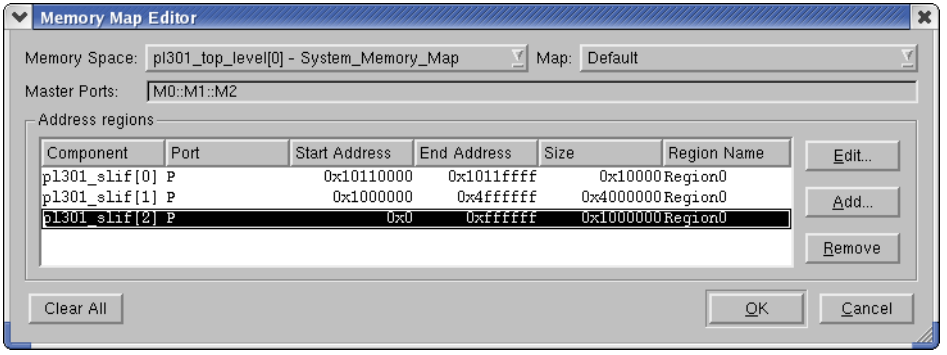


Figure 4-17 Configured address map

17. Click on **OK** to close the dialog box.

4.2.5 Configuring the sparse interconnect

Sparse interconnect enables you to define which master is connected to which slave. This enables you to minimize the interconnect logic in the HPM.

AMBA Designer provides two methods to enable you to configure sparse connectivity. For small interconnects it is usually preferable to use the PL301 System Sparse Interconnect dialog. For interconnects that use more than 16 master or slave interfaces then AMBA Designer only permits use of the PL301 Master Sparse Interconnect dialog. See *Sparse connectivity* on page 4-39 for more information about using these dialog boxes.

To configure the master to slave interconnections:

1. Right-click on the pl301_top_level[0] component to display the context menu.
2. Select **AMBA Designer → PL301 System Sparse Interconnect** from the context menu, as Figure 4-18 on page 4-19 shows.

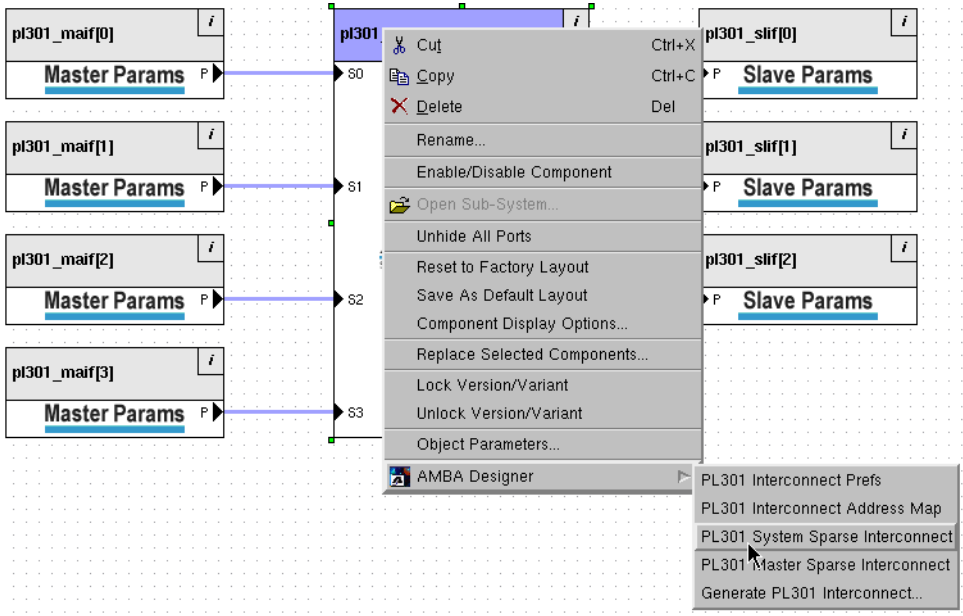


Figure 4-18 Accessing the sparse interconnect

The PL301 System Sparse Interconnect dialog box is displayed, as Figure 4-19 shows.

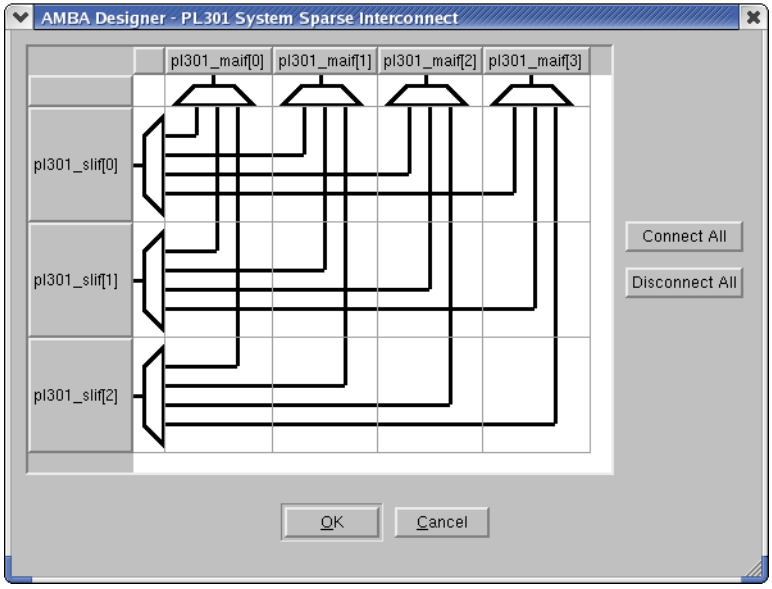


Figure 4-19 Fully connected interconnect

Figure 4-19 on page 4-19 shows a fully connected interconnect where each master connects to every slave.

3. Click on **Disconnect All** to remove all connections in the interconnect, as Figure 4-20 shows.

Note

Use the **Connect All** button in the PL301 System Sparse Interconnect dialog box when you require each master to be connected to every slave.

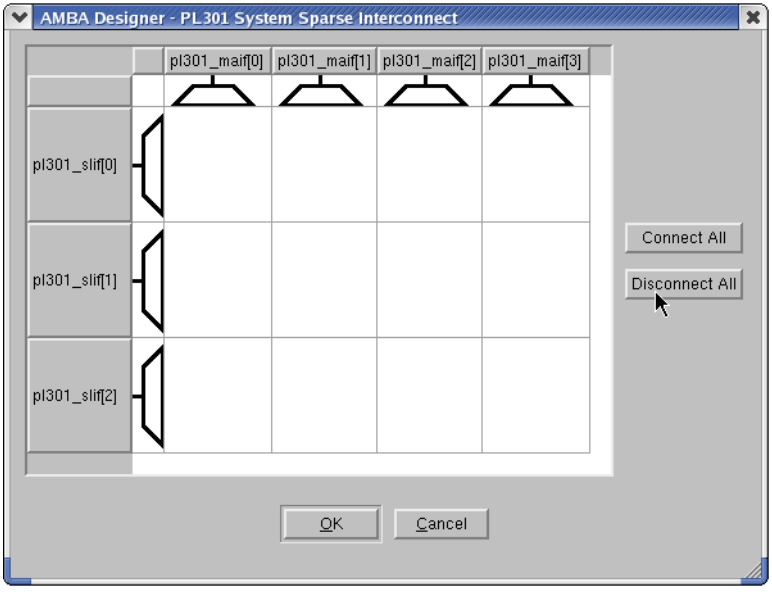


Figure 4-20 Unconfigured sparse interconnect

To connect a master interface to a slave interface, double-click on the grid where the corresponding rows and columns intersect. The grid consists of:

Rows That represent the slaves that connect to the HPM. In this example they are named pl301_slif[*n*].

Columns That represent the masters that connect to the HPM. In this example they are named pl301_maif[*n*].

For this example, you must select the area of the grid that Figure 4-21 on page 4-21 shows.

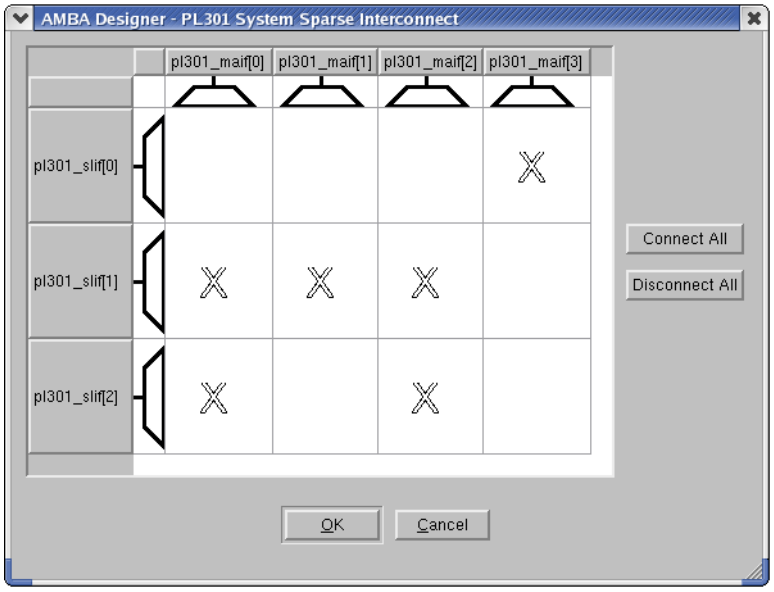


Figure 4-21 Sparse interconnect grid for 4x3 example

To select these areas on the grid:

4. Double-click on the each of the Xs that Figure 4-21 shows. Figure 4-22 on page 4-22 shows the resulting configured sparse interconnect.

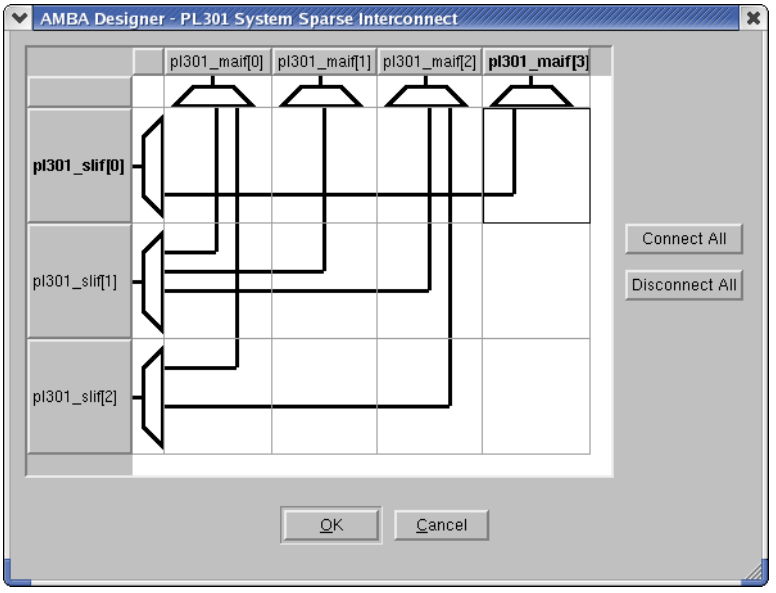


Figure 4-22 Configured sparse interconnect

5. Click on **OK** to close the PL301 System Sparse Interconnect dialog box. The program saves the configured interconnect.

Table 4-2 lists the connections in the configured sparse interconnect.

Table 4-2 Configured sparse interconnect

		ARM1176JZF processor interfaces			
		pl301_maif[0] data bus	**pl301_maif[1] DMA**	**pl301_maif[2] instruction bus**	**pl301_maif[3] peripheral bus**
DMC	**pl301_slif[0] peripheral bus**	-	-	-	Connection
	pl301_slif[1] AXI bus	Connection	Connection	Connection	-
Memory	**pl301_slif[2] AXI bus**	Connection	-	Connection	-

4.2.6 Generating the interconnect

After configuring the master and slave interfaces, it is now possible to create the interconnect.

To generate the interconnect:

1. Right-click on the pl301_top_level[0] component to display the context menu.
2. Select **AMBA Designer → Generate PL301 Interconnect...** from the context menu, as Figure 4-23 shows.

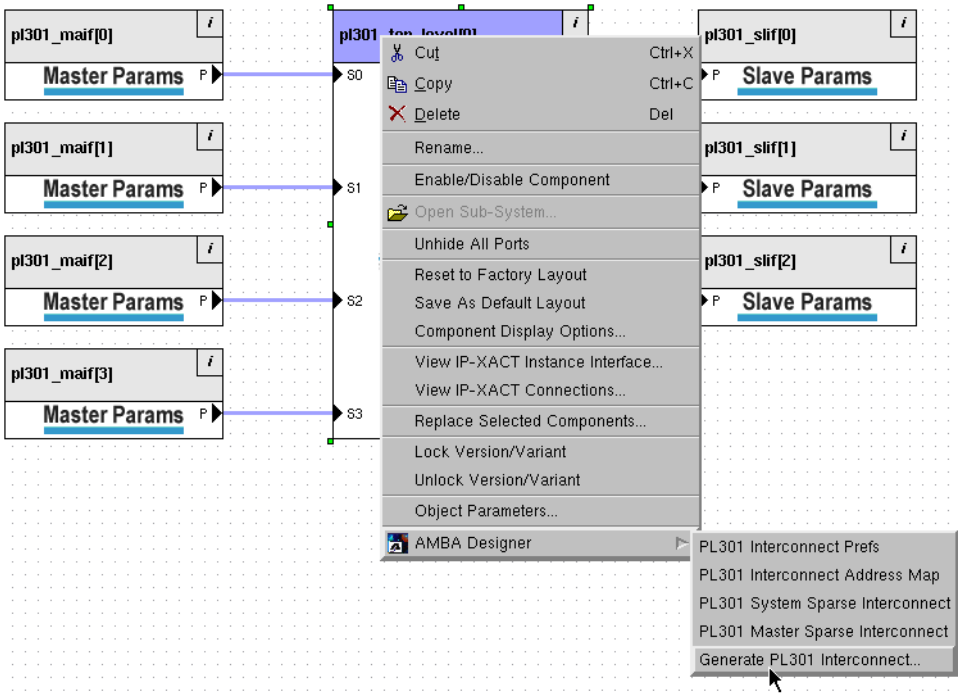


Figure 4-23 Generating the 4x3 interconnect

The Generate PL301 Interconnect process produces and functionally verifies all of the configured interconnect. Figure 4-24 on page 4-24 shows the generated interconnect.

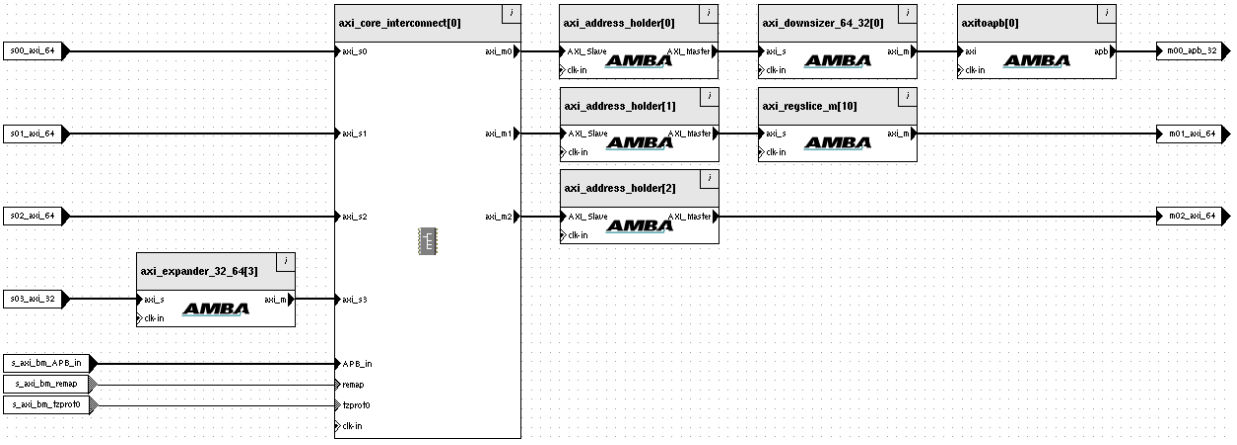


Figure 4-24 Generated 4x3 interconnect

In Figure 4-24 the interconnect consists of the following components:

axi_core_interconnect[0]

The HPM, configured with a 64-bit data bus.

axi_expander_32_64[3]

An expander component that enables transfers between the 32-bit peripheral bus on the ARM1176JZF and the 64-bit bus of the HPM.

axi_address_holder[0], axi_address_holder[1], and axi_address_holder[2]

AMBA Designer provides these components to enable it to store the address information for the master interface ports of the HPM. SoC Designer Simulator uses this information when it reconstructs the address map for the HPM. See *System modeling and the address map* on page 4-63 for more information about the axi_address_holder component.

axi_downsizer_64_32[0]

A downsizer component that enables transfers between the 64-bit bus of the HPM and the 32-bit bus on the DMC.

axitoapb[0] A protocol conversion component that enables transfers between the AXI interface on the HPM and the APB interface on the DMC.

axi_regslice_m[10]

A register slice component that enables timing isolation between the HPM and the DMC. AMBA Designer always configures the operating mode of the register slice to be fully registered, for all five AXI channels.

Note

AMBA Designer does not enable you to configure the operating mode of the register slice to be registered forward path or static bypass.

After selecting **Generating Interconnect...**, AMBA Designer displays the interconnect in a new Diagram Window with the tab name of **PL301_*xxx*_4x3_1**, where ***xxx*** is the sequence of alphanumeric characters used prior to generating the interconnect. The **_1** suffix appended to the tab name signifies that this is a generated interconnect.

The program saves the generated interconnect using a `PL301_xxx_4x3_1.mxp` filename in the default `Designs` directory:

UNIX `home/user/.ARM/AMBA_Designer/Designs/`

Windows `C:\Documents and Settings\user\My Documents\AMBA_Designer\Designs\`

Where *user* is the login name of the current person running AMBA Designer.

If more interconnects are generated from **PL301_*xxx*_4x3** then by default AMBA Designer uses the same tab name and design filename. Prior to overwriting the previous design, AMBA Designer requests confirmation that you want to proceed, as Figure 4-25 shows.

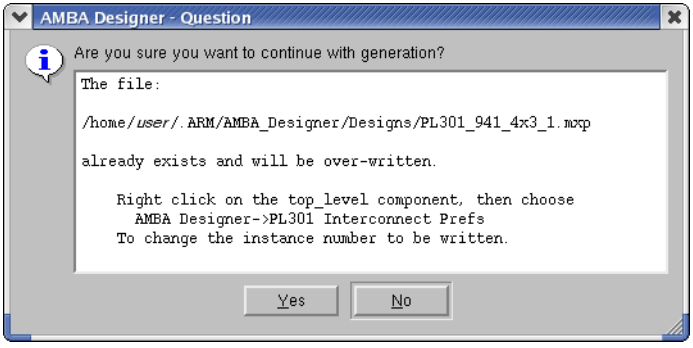


Figure 4-25 Generation instance number confirmation

Click on **Yes** to overwrite the previous design. Clicking on **No** cancels the **Generating Interconnect...** process.

Note

See *PL301 interconnect preferences* on page 4-47 for information about how to change the default behavior for interconnect instance numbering.

4.2.7 Adding the interconnect to the component library

To enable the program to use the generated interconnect in modeling scenarios then you must add the interconnect to the component library.

To add this component to the component library:

1. Right-click on the axi_core_interconnect[0] component to display the context menu.
2. Select **AMBA Designer → Add PL301 to Component Library...** from the context menu, as Figure 4-26 shows.

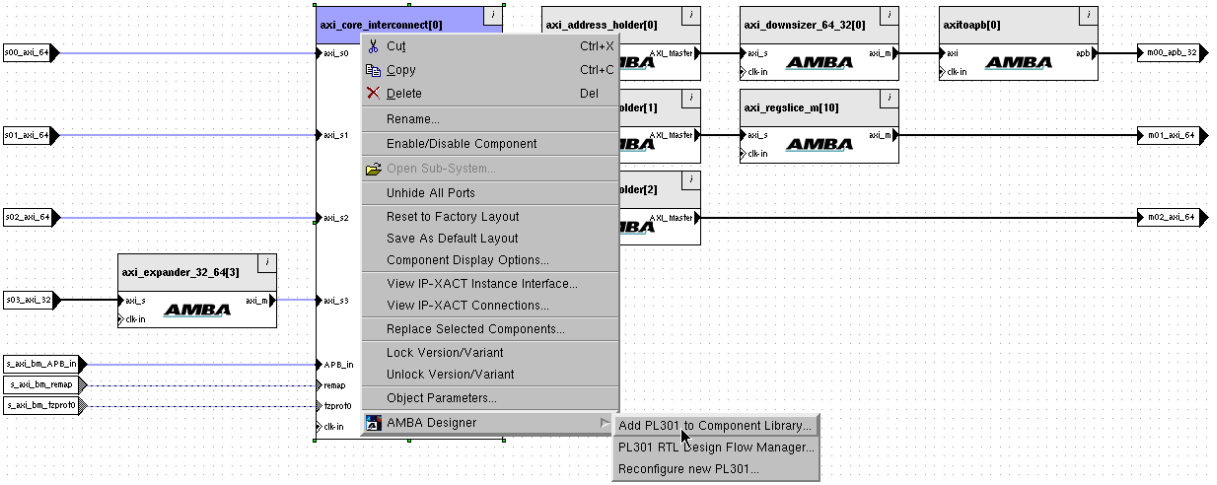


Figure 4-26 Adding the interconnect to the component library

After the program completes this process it displays the name of your component in the Component Window. Figure 4-27 on page 4-27 shows a Component Window that contains a PL301_*xxx*_4x3_1 interconnect model name.

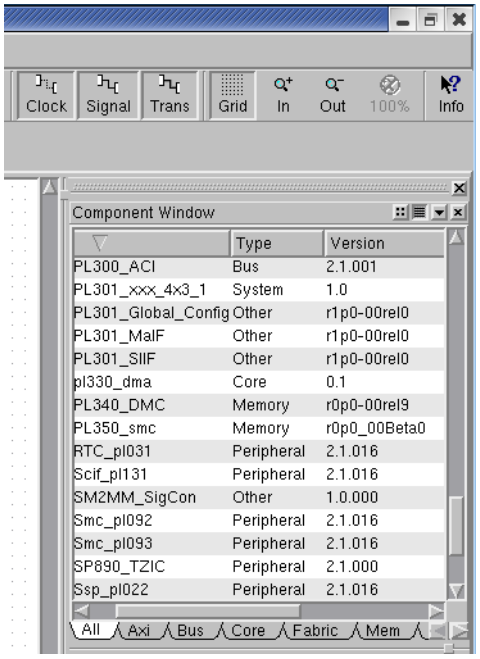


Figure 4-27 Component Window

Note

Figure 4-27 shows a Component Window using a listed output. Depending on your AMBA Designer application settings then the components might display as icons.

4.2.8 RTL generation or system modeling

After the interconnect is created you can then either generate the RTL for the interconnect or incorporate the interconnect model into your system model to enable you to evaluate the performance of your system design.

The RTL and modeling process flows are described in:

- Chapter 8 *RTL Design Flow*
- Chapter 7 *Creating a System Model*.

Note

AMBA Designer only enables the generation of RTL, when the program runs on a UNIX operating system. You must also ensure that the AMBA Designer license file, usually named `licence.dat`, permits the program to generate RTL.

4.3 Clock domain crossing example

This example describes how to configure and produce a 2x2 interconnect using the clock domains that Figure 4-28 shows.

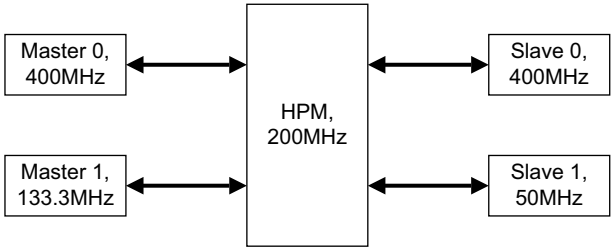


Figure 4-28 Example 2x2 system

Note

The example only configures the minimum number of parameters that are sufficient to describe the use of clock domain crossing.

If you intend to model your system in SoC Designer Simulator then you must ensure that you configure the model_clock_divider_*x* parameters depending on the simulation frequency that is applied to the bridge. The model_clock_divider_*x* parameters are calculated using:

$$\texttt{model\_clock\_divider\_}x = \frac{F_S}{F_O}$$

Where:

F_S Simulation frequency applied to the bridge.

F_O Operating frequency of the component, that attaches to the bridge interface where the clock divider is located.

This example uses the default setting for the clock slave port on each bridge, so all the bridges are clocked by the master clock in SoC Designer Simulator. The master clock operates at the same rate as the scheduler in SoC Designer Simulator and therefore the master clock can be considered to operate at the clock speed of the fastest component in the system. The 2x2 example uses a maximum clock speed of 400MHz so the model_clock_divider_*x* parameters are derived using a simulation frequency, F_S=400MHz.

Note

By default, the models for the clock domain crossing bridges operate at the same rate as the scheduler in SoC Designer Simulator. If you use another clock to operate the clock domain crossing bridge model then:

- you must ensure that this clock frequency is equal to, or an integer multiple of, the highest AXI bus clock frequency that is applied to the bridge
- for asynchronous bridges, you must also ensure that this clock frequency is an integer multiple of the lowest AXI bus clock frequency that is applied to the bridge.

The 2x2 example is described in the following sections:

- *Configuring the HPM*
- *Generating the interconnect* on page 4-35.

4.3.1 Configuring the HPM

The process of configuring the HPM is described in:

- *Instantiating the AXI-core bus matrix*
- *Configuring master 0* on page 4-30
- *Configuring master 1* on page 4-31
- *Configuring slave 0* on page 4-33
- *Configuring slave 1* on page 4-34.

Instantiating the AXI-core bus matrix

To create the HPM:

1. Select **File** → **New** from the Main menu. This creates a new Diagram Window.
2. Click on the **New Fabric IP** button on the Main Toolbar to open the Configure new PrimeCell IP dialog box.
3. In the Configure new PrimeCell IP dialog box:
   a. Click on the row containing a PrimeCell name of PL301.
   b. Click on **OK** to close the dialog box and open the New PL301 AXI-core Bus Matrix dialog box.
4. In the New PL301 AXI-core Bus Matrix dialog box:
   a. Enter 2 in the Number of Masters field.
   b. Enter 2 in the Number of Slaves field.

c. Click on **OK** to close the dialog box.

Configuring master 0

Master 0 operates at 400MHz and the HPM operates at 200MHz. Because master 0 is operating faster than the HPM then the clock_domain_crossing parameter must be set to fast.

From Table 4-6 on page 4-65, when clock_domain_crossing=fast then you must enter a value for model_clock_divider_1. To determine the model_clock_divider_1 setting it is necessary to know the values of F_S and F_O.

For model_clock_divider_1, Table 4-6 on page 4-65 states that when clock_domain_crossing=fast then F_O is the operating frequency of the component that attaches to the master interface of the downwards synchronizing bridge. That component is the HPM and therefore F_O = 200MHz.

The clock domain crossing configuration options for master 0 are:

- clock_domain_crossing = fast
- model_clock_divider_1 = F_S/F_O = 400/200 = 2.

To enter the clock domain crossing options for master 0:

1. Double-click on the pl301_maif[0] component to open the Edit Parameters dialog box.

In the Edit Parameters dialog box:

2. Set the clock_domain_crossing parameter to fast.
3. Enter 2 in the model_clock_divider_1 parameter field, as Figure 4-29 on page 4-31 shows.

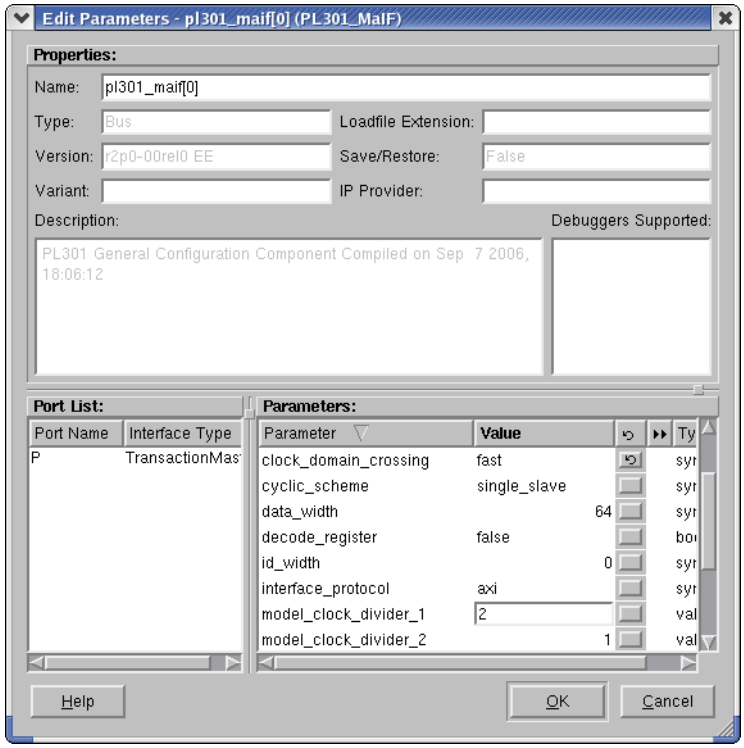


Figure 4-29 Clock domain crossing for master 0

4. Click on **OK** to close the dialog box.

Configuring master 1

Master 1 operates at 133.3MHz and the HPM operates at 200MHz. Because master 1 is operating asynchronous to the HPM then the clock_domain_crossing parameter must be set to async.

From Table 4-6 on page 4-65, when clock_domain_crossing=async then you must enter values for model_clock_divider_1 and model_clock_divider_2.

For model_clock_divider_1, Table 4-6 on page 4-65 states that when clock_domain_crossing=async then F_O is the operating frequency of the component that attaches to the slave interface of the asynchronous bridge. That component is master 1 and therefore F_O = 133.3MHz.

For model_clock_divider_2, Table 4-6 on page 4-65 states that F_O is the operating frequency of the component that attaches to the master interface of the asynchronous bridge. That component is the HPM and therefore F_O = 200MHz.

The clock domain crossing configuration options for master 1 are:

- clock_domain_crossing = async
- model_clock_divider_1 = F_S/F_O = 400/133.3 = 3
- model_clock_divider_2 = F_S/F_O = 400/200 = 2.

To enter the clock domain crossing options for master 1:

1. Double-click on the pl301_maif[1] component to open the Edit Parameters dialog box.

In the Edit Parameters dialog box:

2. Set the clock_domain_crossing parameter to async.
3. Enter 3 in the model_clock_divider_1 parameter field.
4. Enter 2 in the model_clock_divider_2 parameter field, as Figure 4-30 shows.

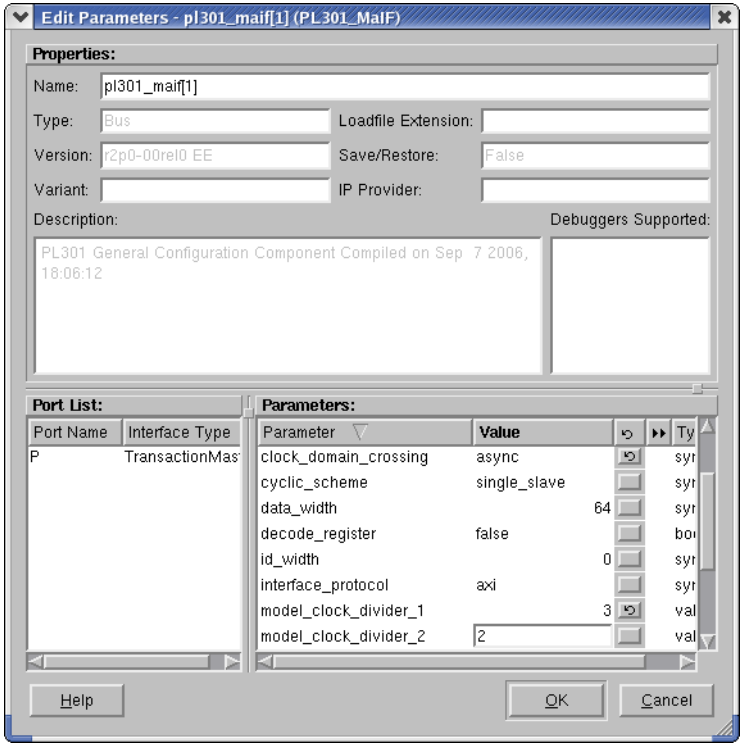


Figure 4-30 Clock domain crossing for master 1

5. Click on **OK** to close the dialog box.

Configuring slave 0

Slave 0 operates at 400MHz and the HPM operates at 200MHz. Because slave 0 is operating faster than the HPM then the clock_domain_crossing parameter must be set to fast.

From Table 4-7 on page 4-69, when clock_domain_crossing=fast then you must enter a value for model_clock_divider_1. To determine the model_clock_divider_1 setting it is necessary to know the values of F_S and F_O.

For model_clock_divider_1, Table 4-7 on page 4-69 states that when clock_domain_crossing=fast then F_O is the operating frequency of the component that attaches to the slave interface of the upwards synchronizing bridge. That component is the HPM and therefore F_O = 200MHz.

The clock domain crossing configuration options for slave 0 are:

- clock_domain_crossing = fast
- model_clock_divider_1 = F_S/F_O = 400/200 = 2.

To enter the clock domain crossing options for slave 0:

1. Double-click on the pl301_slif[0] component to open the Edit Parameters dialog box.

In the Edit Parameters dialog box:

2. Set the clock_domain_crossing parameter to fast.
3. Enter 2 in the model_clock_divider_1 parameter field, as Figure 4-31 on page 4-34 shows.

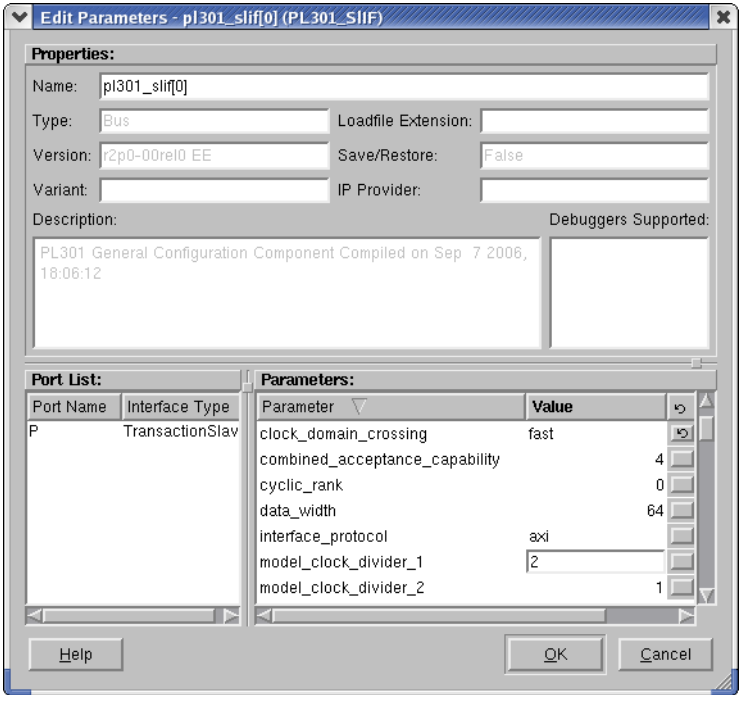


Figure 4-31 Clock domain crossing for slave 0

4. Click on **OK** to close the dialog box.

Configuring slave 1

Slave 1 operates at 50MHz and the HPM operates at 200MHz. Because slave 1 is operating slower than the HPM then the clock_domain_crossing parameter must be set to slow.

From Table 4-7 on page 4-69, when clock_domain_crossing=slow then you must enter a value for model_clock_divider_1. To determine the model_clock_divider_1 setting it is necessary to know the values of F_S and F_O.

For model_clock_divider_1, Table 4-7 on page 4-69 states that when clock_domain_crossing=slow then F_O is the operating frequency of the component that attaches to the master interface of the downwards synchronizing bridge. That component is slave 1 and therefore F_O = 50MHz.

The clock domain crossing configuration options for slave 1 are:

- clock_domain_crossing = slow
- model_clock_divider_1 = F_S/F_O = 400/50 = 8.

To enter the clock domain crossing options for slave 1:

1. Double-click on the pl301_slif[1] component to open the Edit Parameters dialog box.

In the Edit Parameters dialog box:

2. Set the clock_domain_crossing parameter to slow.
3. Enter 8 in the model_clock_divider_1 parameter field, as Figure 4-32 shows.

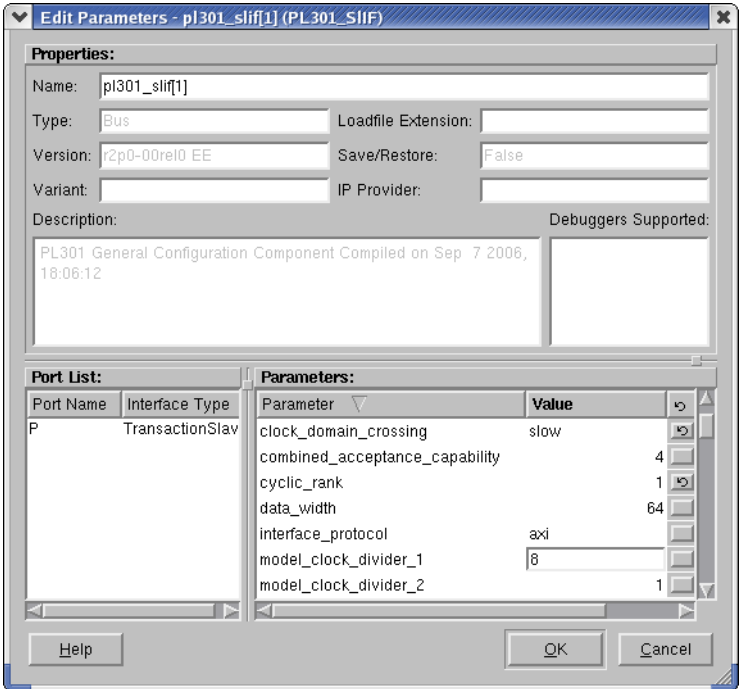


Figure 4-32 Clock domain crossing for slave 1

4. Click on **OK** to close the dialog box.

4.3.2 Generating the interconnect

To generate the interconnect:

1. Right-click on the pl301_top_level[0] component to display the context menu.
2. Select **AMBA Designer → Generate Interconnect...** from the context menu, as Figure 4-33 on page 4-36 shows.

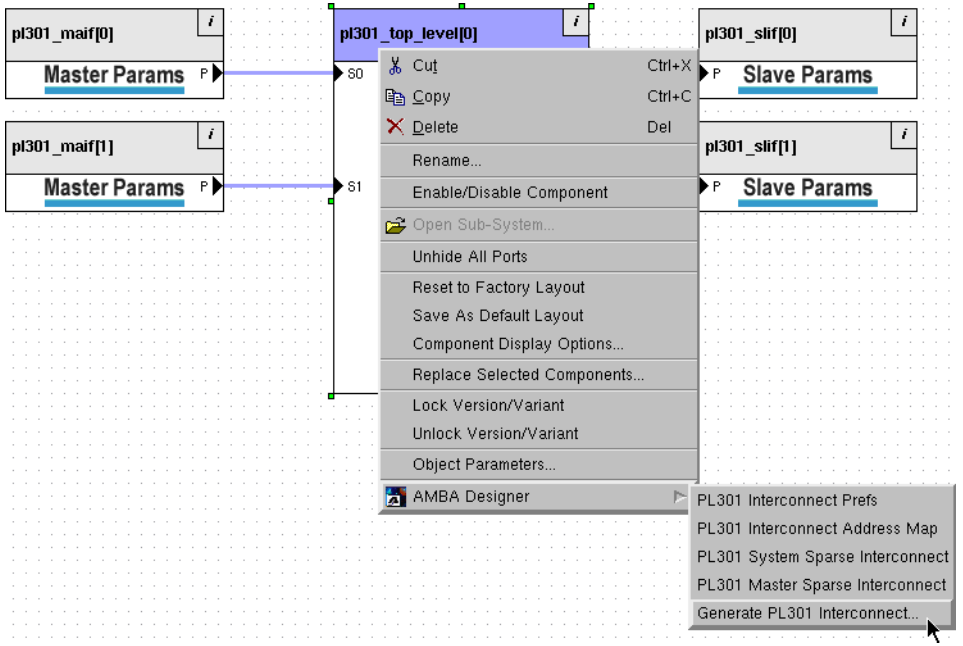


Figure 4-33 Generating the 2x2 interconnect

3. In the Question dialog box, click on **Yes** as Figure 4-34 on page 4-37 shows.

Note

The Question dialog is displayed because this example only configures the minimum number of parameters that are sufficient to demonstrate the use of clock domain crossing.

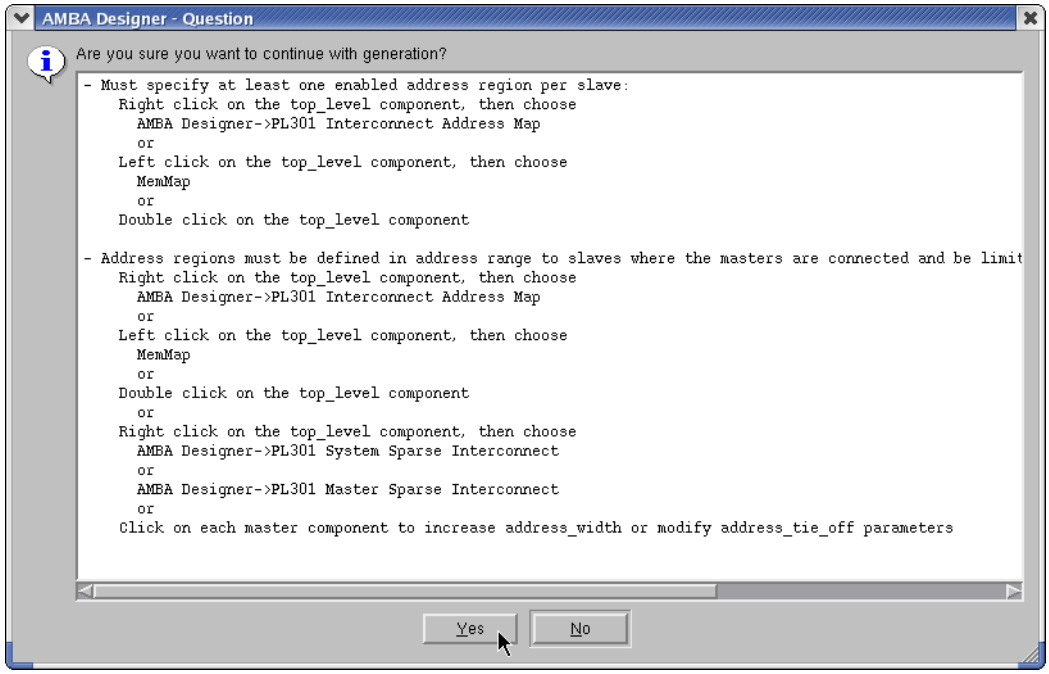


Figure 4-34 Question dialog

AMBA Designer creates a model view of the configuration, as Figure 4-35 shows.

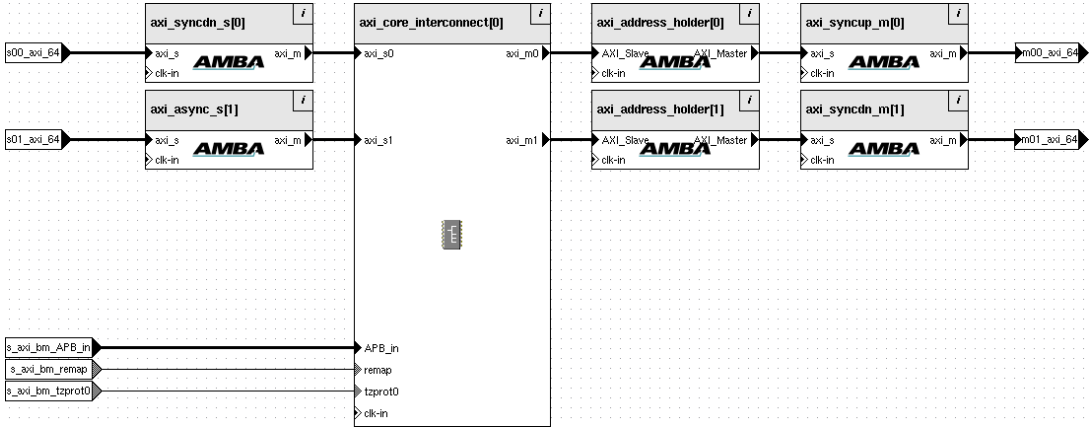


Figure 4-35 Generated 2x2 interconnect

In Figure 4-35 on page 4-37 the interconnect consists of:

axi_core_interconnect[0]

The HPM.

axi_syncdn_s[0]

A downwards synchronizing bridge component that enables transfers between master 0 operating at 400MHz and the HPM operating at 200MHz.

axi_async_s[1]

An asynchronous bridge component that enables transfers between master 1 operating at 133.3MHz and the HPM operating at 200MHz.

axi_address_holder[0] and axi_address_holder[1]

AMBA Designer provides these components to enable it to store the address information for the master interface ports of the HPM. See *System modeling and the address map* on page 4-63 for more information about the axi_address_holder component.

axi_syncup_m[0]

An upwards synchronizing bridge component that enables transfers between the HPM operating at 200MHz and slave 0 operating at 400MHz.

axi_syncdn_m[1]

A downwards synchronizing bridge component that enables transfers between the HPM operating at 200MHz and slave 1 operating at 50MHz.

Figure 4-36 shows a block diagram of the configured system.

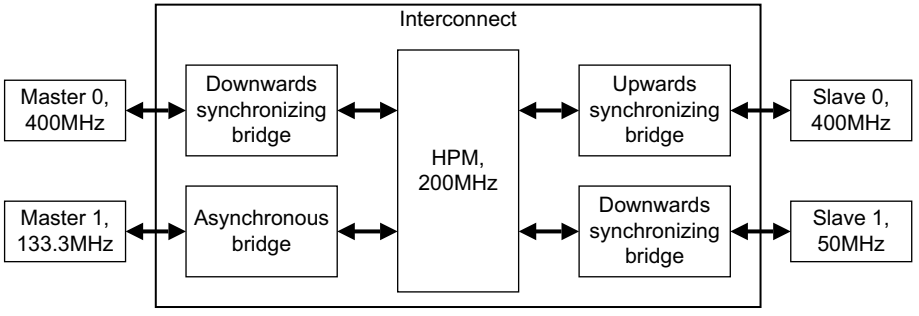


Figure 4-36 Configured system

4.4 Sparse connectivity

You can configure the sparse connectivity of an interconnect by using the dialog boxes described in:

- *System sparse interconnect*
- *Master sparse interconnect* on page 4-41.

4.4.1 System sparse interconnect

The PL301 System Sparse Interconnect dialog box displays a graphical representation of the HPM that enables you to configure the connectivity of an interconnect.

To open the PL301 System Sparse Interconnect dialog box:

1. Right-click on the HPM component to display the context menu. By default, the name of an HPM component is pl301_top_level[0].
2. From the context menu, select **AMBA Designer → PL301 System Sparse Interconnect**. This opens the PL301 System Sparse Interconnect dialog box.

Figure 4-37 on page 4-40 shows an example 6x6 interconnect that is displayed using the PL301 System Sparse Interconnect dialog box.

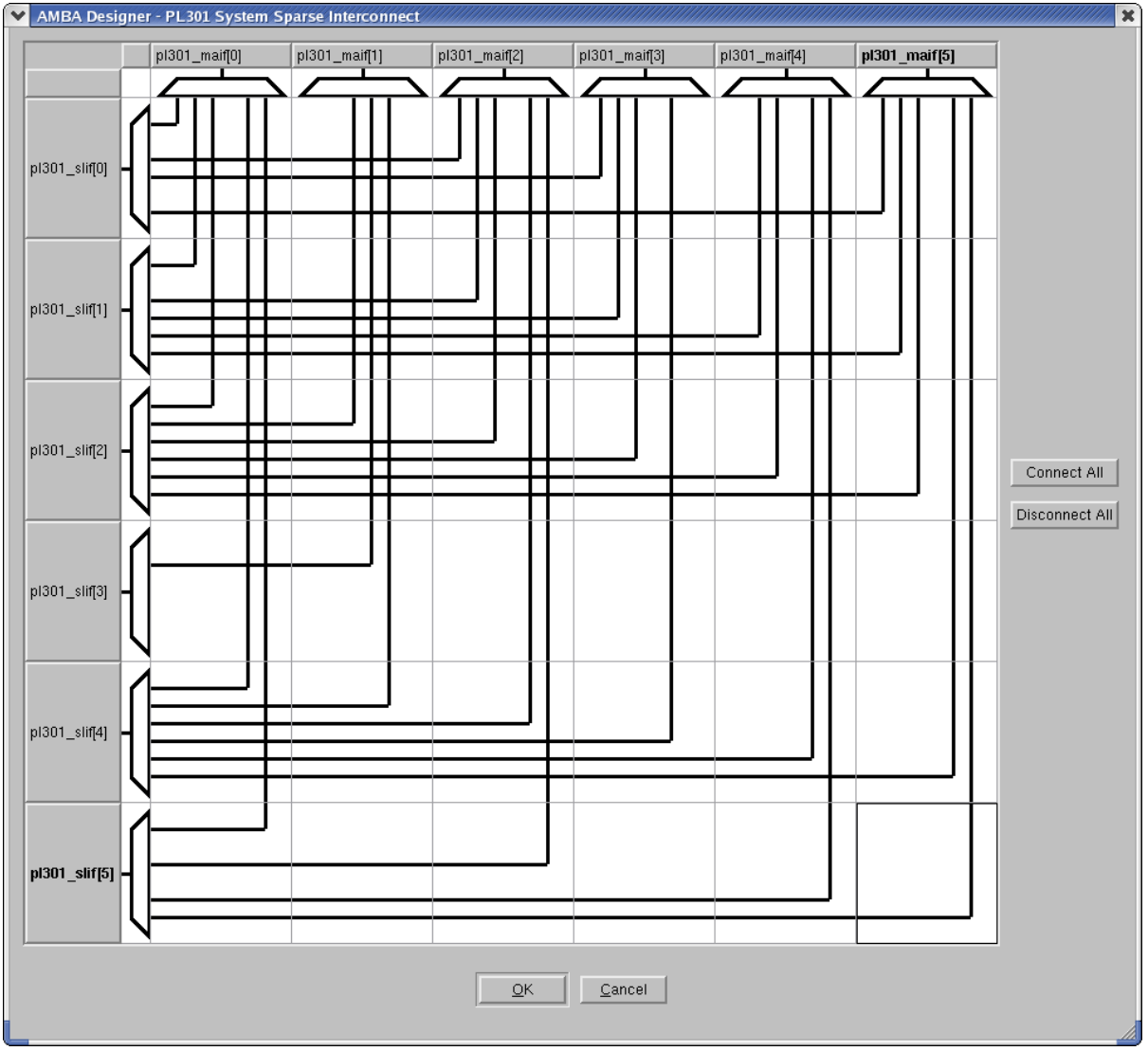


Figure 4-37 Example 6x6 sparse interconnect

For small interconnects, ARM recommends that you use the PL301 System Sparse Interconnect dialog box to configure the connectivity of an interconnect.

For large interconnects then it becomes increasingly difficult to display and enter the configuration using a graphical representation of the interconnect. Therefore, when the number of masters or slaves exceeds 16, AMBA Designer only permits configuration of the sparse connectivity by use of the PL301 Master Sparse Interconnect dialog box.

Using the PL301 System Sparse Interconnect dialog box

See the example that *Configuring the sparse interconnect* on page 4-18 describes, for information about how to use the PL301 System Sparse Interconnect dialog box.

4.4.2 Master sparse interconnect

The PL301 Master Sparse Interconnect dialog box displays a list of slaves that are connected to the selected master. By using the Master drop-down list, you are able to select each master and therefore configure the connectivity of the entire interconnect.

The features and how to use the Master Sparse Interconnect dialog box are described in:

- *Features*
- *Using the Master Sparse Interconnect dialog box* on page 4-43.

Features

To open the PL301 Master Sparse Interconnect dialog box:

1. Right-click on the HPM component to display the context menu. By default, the name of an HPM component is pl301_top_level[0].
2. From the context menu, select **AMBA Designer → PL301 Master Sparse Interconnect**. This opens the PL301 Master Sparse Interconnect dialog box as Figure 4-38 shows.

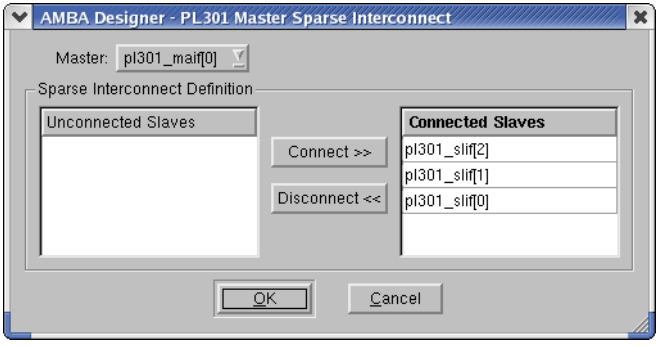


Figure 4-38 PL301 Master sparse interconnect dialog

Figure 4-38 on page 4-41 shows that this dialog box contains a Master drop-down list and a Sparse Interconnect Definition pane. These are described in:

- *Master*
- *Sparse Interconnect Definition*.

Master

The Master drop-down list contains all the masters that are connected to the HPM. After you select a master, AMBA Designer displays the current configuration for this master in the Sparse Interconnect Definition pane.

Sparse Interconnect Definition

This pane contains the following buttons:

Unconnected Slaves

Click on **Unconnected Slaves** to select all the slaves that are listed under the **Unconnected Slaves** button.

Connected Slaves

Click on **Connected Slaves** to select all the slaves that are listed under the **Connected Slaves** button.

Connect Click on **Connect** to move the slaves that are selected in the Unconnected Slaves column to the Connected Slaves column.

Disconnect Click on **Disconnect** to move the slaves that are selected in the Connected Slaves column to the Disconnected Slaves column.

Figure 4-39 shows the PL301 Master Sparse Interconnect dialog box for master interface 3, using configuration data taken from the 4x3 example in *Example 4x3 interconnect* on page 4-4.

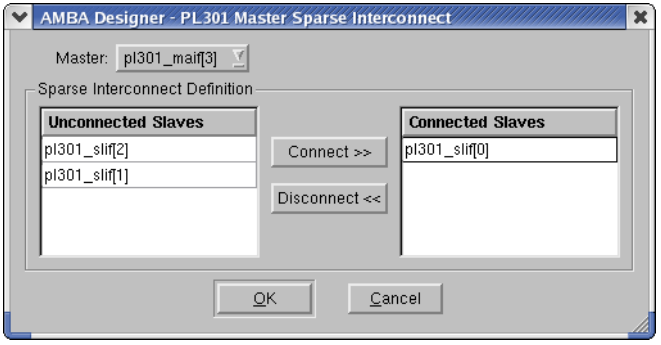


Figure 4-39 Master sparse interconnect for master interface 3

In Figure 4-39 on page 4-42:

- the Master drop-down list shows that the selected master is pl301_maif[3]
- the Sparse Interconnect Definition pane shows that pl301_maif[3] connects to one slave, pl301_slif[0].

Using the Master Sparse Interconnect dialog box

This section describes how to use the PL301 Master Sparse Interconnect dialog box to configure a 4x3 interconnect. You create the example using the interconnect connections that Table 4-2 on page 4-22 lists.

Note

When you configure the connectivity of small interconnects, it is usually preferable to use the PL301 System Sparse Interconnect dialog box.

To create the 4x3 interconnect:

1. Select **File** → **New** from the Main menu. This creates a new Diagram Window.
2. Click on the **New Fabric IP** button on the Main Toolbar to open the Configure new PrimeCell IP dialog box.
3. In the Configure new PrimeCell IP dialog box:
   a. Click on the row containing a PrimeCell name of PL301.
   b. Click on **OK** to close the dialog box and open the New PL301 AXI-core Bus Matrix dialog box.
4. In the New PL301 AXI-core Bus Matrix dialog box:
   a. Enter 4 in the Number of Masters field.
   b. Enter 3 in the Number of Slaves field.
   c. Click on **OK** to close the dialog box.

By default, AMBA Designer connects each master to every slave. To disconnect each master from every slave:

5. Right-click on the pl301_top_level[0] component to display the context menu.
6. From the context menu, select **AMBA Designer** → **PL301 System Sparse Interconnect**, to open the PL301 System Sparse Interconnect dialog box.
7. Click on **Disconnect All** to remove all connections in the interconnect.

8. Click on **OK** to close the PL301 System Sparse Interconnect dialog box. The program saves the configured interconnect.

Note

To disconnect each master from every slave you can use the PL301 Master Sparse Interconnect dialog box, however using the PL301 System Sparse Interconnect dialog box usually involves less steps.

To open the PL301 Master Sparse Interconnect dialog box:

9. Right-click on the pl301_top_level[0] component to display the context menu.

10. Select **AMBA Designer → PL301 Master Sparse Interconnect** from the context menu, as Figure 4-40 shows.

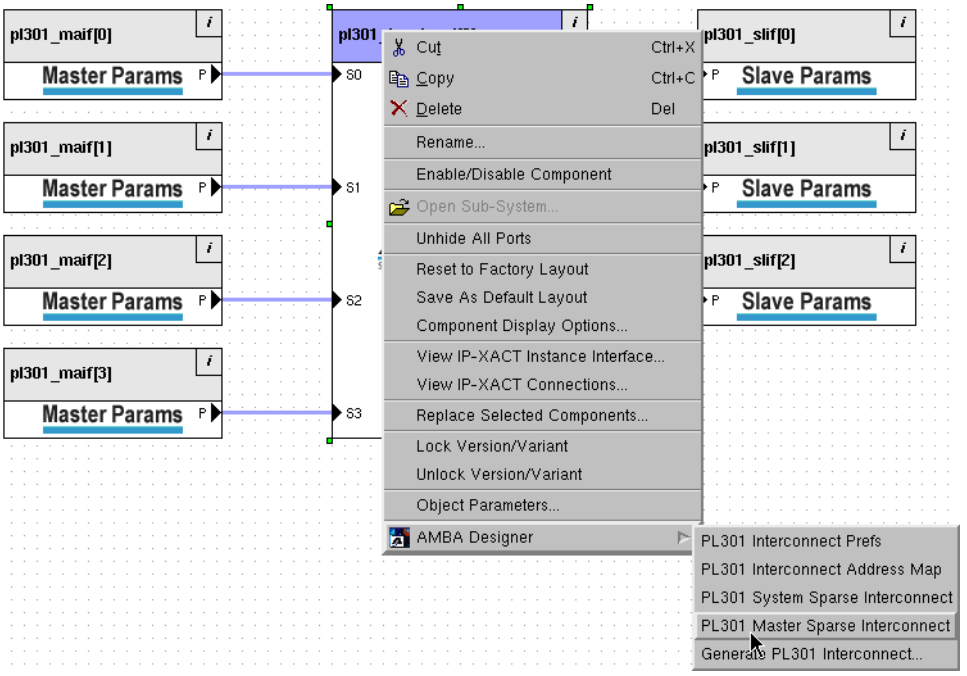


Figure 4-40 Accessing the master sparse interconnect

The PL301 Master Sparse Interconnect dialog box is displayed, as Figure 4-41 on page 4-45 shows.

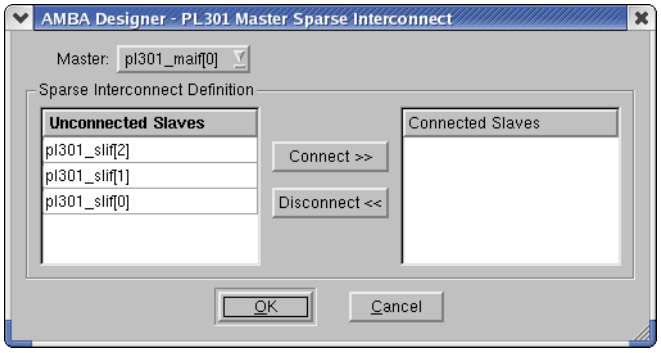


Figure 4-41 Master sparse interconnect example

To configure the connections for pl301_maif[0]:

11. In the Sparse Interconnect Definition pane, click on pl301_slif[1] in the Unconnected Slaves column, to select the slave.
12. Click on **Connect** to add this slave to the Connected Slaves column.
13. Click on pl301_slif[2] in the Unconnected Slaves column, to select the slave.
14. Click on **Connect** to add this slave to the Connected Slaves column.

To configure the connections for pl301_maif[1]:

15. Set Master to pl301_maif[1], using the drop-down list.
16. Click on pl301_slif[1] in the Unconnected Slaves column, to select the slave.
17. Click on **Connect** to add this slave to the Connected Slaves column.

To configure the connections for pl301_maif[2]:

18. Set Master to pl301_maif[2], using the drop-down list.
19. Click on pl301_slif[1] in the Unconnected Slaves column, to select the slave.
20. Click on **Connect** to add this slave to the Connected Slaves column.
21. Click on pl301_slif[2] in the Unconnected Slaves column, to select the slave.
22. Click on **Connect** to add this slave to the Connected Slaves column.

To configure the connections for pl301_maif[3]:

23. Set Master to pl301_maif[3], using the drop-down list.
24. Click on pl301_slif[0] in the Unconnected Slaves column, to select the slave.
25. Click on **Connect** to add this slave to the Connected Slaves column.

To save the changes and close the dialog box:

26. Click on **OK**.

The sparse connectivity is now configured for the 4x3 interconnect. To display the connectivity using the PL301 System Sparse Interconnect dialog box:

1. Right-click on the pl301_top_level[0] component to display the context menu.
2. From the context menu, select **AMBA Designer → PL301 System Sparse Interconnect**, to open the PL301 System Sparse Interconnect dialog box.

Figure 4-42 shows a graphical representation of the interconnect connectivity that was configured using the PL301 Master Sparse Interconnect dialog box.

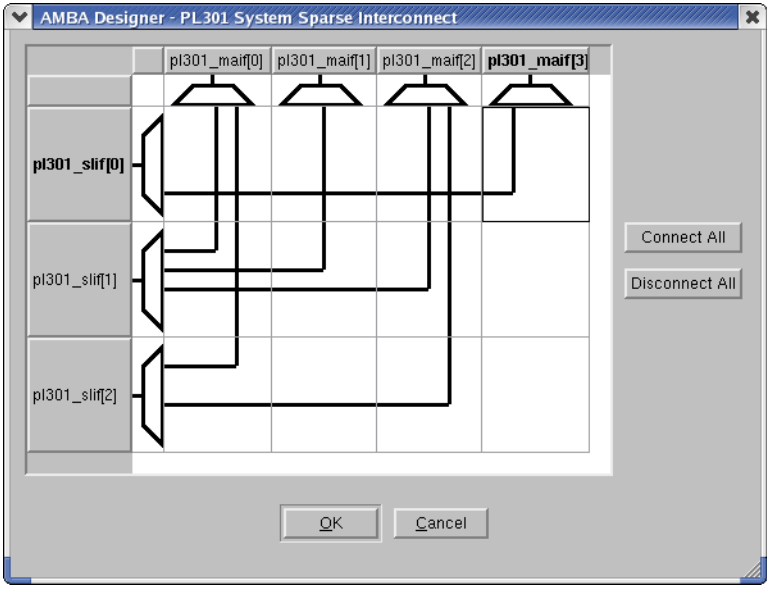


Figure 4-42 Configured sparse interconnect

3. Click on **OK** or **Cancel**, to close the PL301 Sparse Interconnect dialog box.

Note

The connectivity that Figure 4-42 shows is identical to that shown in Figure 4-22 on page 4-22, which was configured using the PL301 System Sparse Interconnect dialog box.

4.5 PL301 interconnect preferences

The PL301 Interconnect Prefs option enables you to change the filename and instance number that AMBA Designer uses when it generates an interconnect.

To open the PL301 Interconnect Prefs dialog box:

1. Right-click on the pl301_top_level[0] component to display the context menu.
2. Select **AMBA Designer → PL301 Interconnect Prefs** from the context menu, as Figure 4-43 shows.

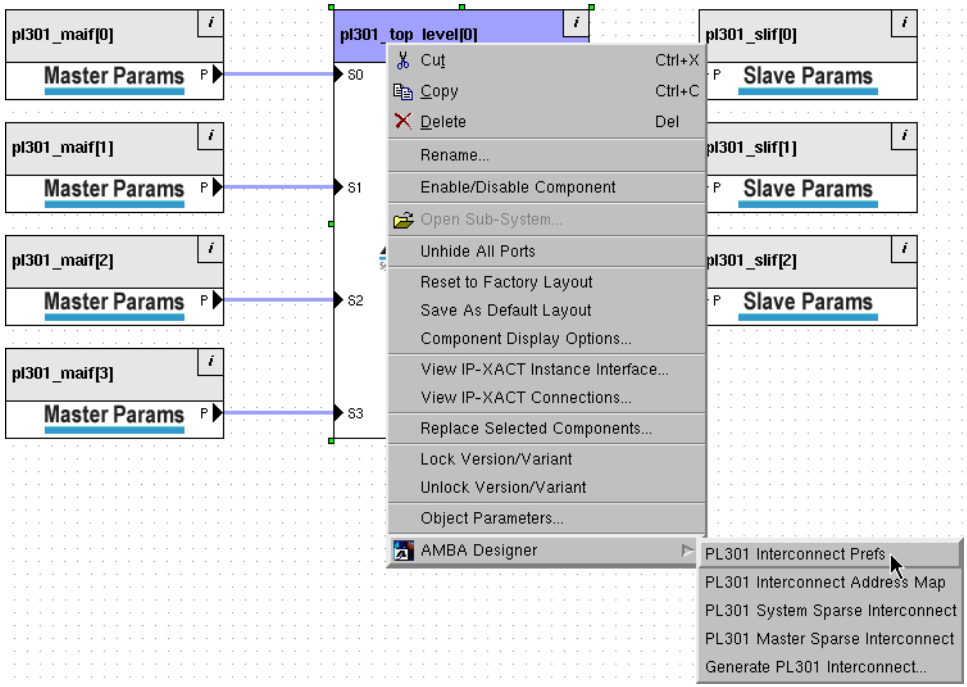


Figure 4-43 PL301 Interconnect Prefs selection

The PL301 Interconnect Preferences dialog box is displayed, as Figure 4-44 on page 4-48 shows.

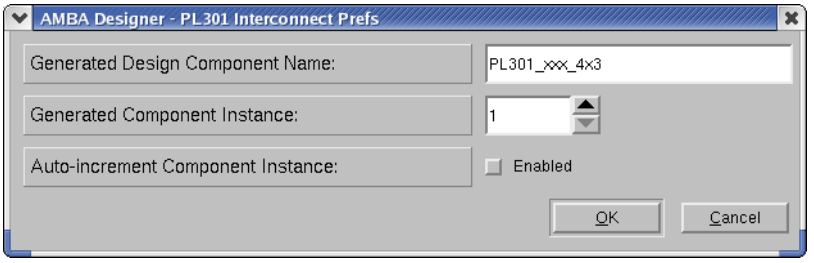


Figure 4-44 PL301 Interconnect Prefs dialog

The preferences in the PL301 Interconnect Prefs dialog box are:

Generated Design Component Name:

The program uses this design name as the filename for the generated interconnect. The filename has a `.mxp` suffix. See *AMBA Designer preferences* on page 3-3 for information on where the program saves the design file.

Enter the name of your generated interconnect in this field.

Note

The characters you enter in this field must comply with the file naming conventions for the file system where the design file is located. Using a reserved character from the file system, might cause the program to terminate unexpectedly when the interconnect is generated.

Generated Component Instance:

The numerical suffix that is appended to the Generated Design Component Name. The default value for this field is 1.

Enter a maximum of six digits in this field. AMBA Designer removes any leading zeros before it appends the suffix to the Generated Design Component Name.

Auto-increment Component Instance:

Select this check box, so that AMBA Designer increments the Generated Component Instance number. For example, if starting with a Generated Component Instance number of 1, when further interconnects are generated from PL301_*xxx*_4x3, they are assigned incrementing suffices, that is, _2, _3, _4, ...

AMBA Designer saves the PL301 interconnect preferences in the current `.mxp` configuration file. See *AMBA Designer preferences* on page 3-3 for information about where the program saves the configuration file.

4.6 Converting the configuration file format

Prior to the r2p0 release of AMBA Designer, when the program saved a configuration of an HPM it created two files. These two files used the following file extensions:

`.ini` Contains the sparse connectivity information.

`.mxp` Contains the main configuration parameters.

Figure 4-45 shows the Warning dialog that displays when you select **Generate Interconnect...** for a `.mxp` configuration file that was saved using an earlier release of the program.

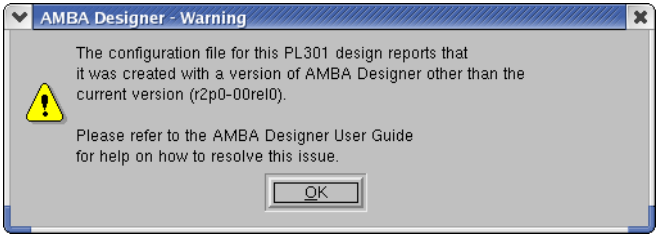


Figure 4-45 Warning dialog

You must use the `ini2mxp` program to convert the a `.mxp` configuration file into a format that enables the program to generate an interconnect. The following section describes how to use this program:

- *The ini2mxp program*.

4.6.1 The `ini2mxp` program

The AMBA Designer installation package includes the `ini2mxp` program. Figure 4-46 shows the default location of this program.

UNIX

```
Top-level directory/
  └─home/
      └─user/
          └─ARM/
              └─SoCDesigner_version/
                  └─AMBADesigner/
                      └─bin/
```

Windows

```
Top-level directory/
  └─Program Files/
      └─ARM/
          └─SoCDesigner_version/
              └─AMBADesigner/
                  └─bin/
```

Figure 4-46 Location of the `ini2mxp` file

Using a console window for UNIX or a Command Prompt for Windows, invoke `ini2mxp` by using the following command:

```
> ini2mxp filename.mxp
```

Where:

filename Name of the MXP file to convert.

The program performs the following steps:

1. Creates a backup of the original configuration and saves the configuration file using a `.bak` extension.
2. Converts the configuration to the new file format.
3. Saves the new configuration, using the original filename and directory location.

Note

The `ini2mxp` program makes no changes to the `.ini` file.

For example, using the following command:

```
> ini2mxp /work/Configurations/PL301_941_4x3.mxp
```

generates the following files in the /work/Configurations directory:

`PL301_941_4x3.bak` A copy of the original configuration.

`PL301_941_4x3.mxp` A new version of the configuration that is now compatible with the current version of AMBA Designer.

4.7 Reconfiguring an interconnect

The program provides a reconfiguration option to enable you to add or remove master interfaces, or slave interfaces, on a previously generated interconnect.

To reconfigure an interconnect:

1. Click on **Open** on the Main Toolbar to open a file browser dialog box.
2. Load an interconnect that you have previously created. These files are named similar to `PL301_xxx_4x3_1.mxp` and are located in the default location:

   UNIX `home/user/.ARM/AMBA_Designer/Designs/`

   Note

   You must enter `home/user/.ARM` in the file browser dialog box because the dialog does not display the `.ARM` directory.

   Windows `C:\Documents and Settings\user\My Documents\AMBA_Designer\Designs\`

   Where *user* is the login name of the current person running AMBA Designer.
3. Right-click on the axi_core_interconnect[0] component to display the context menu.
4. Select **AMBA Designer → PL301 Reconfiguration...** from the context menu, as Figure 4-47 on page 4-52 shows. This opens the PL301 Reconfiguration dialog box.

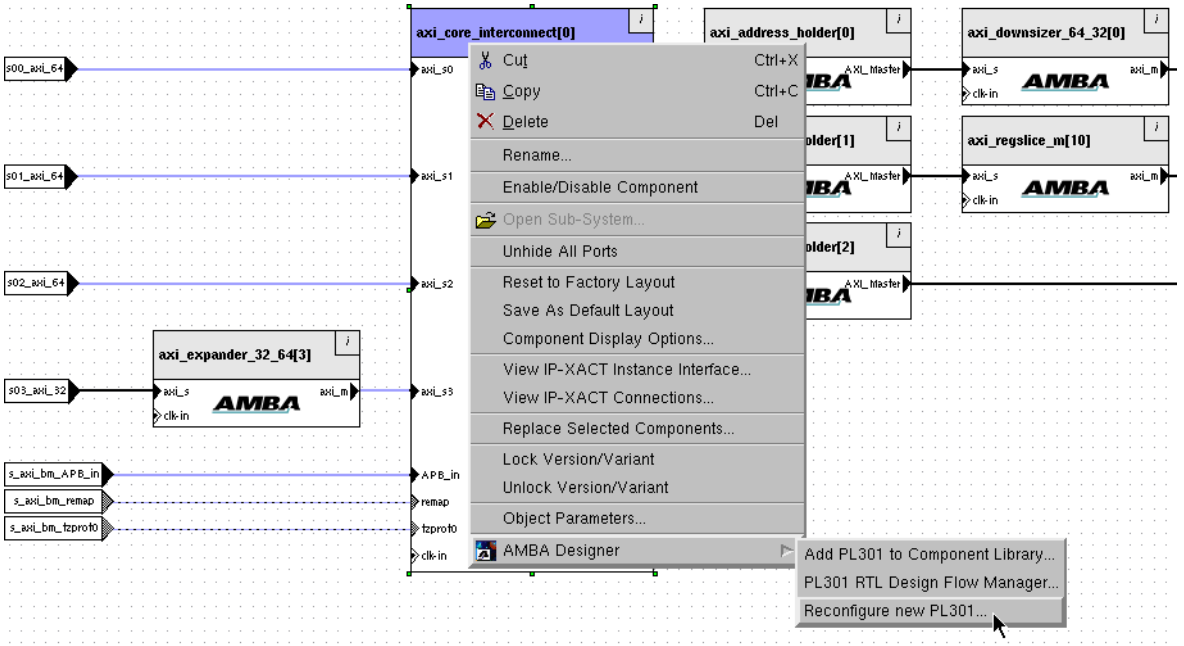


Figure 4-47 PL301 Reconfiguration selection

Figure 4-48 shows the PL301 Reconfiguration dialog box.

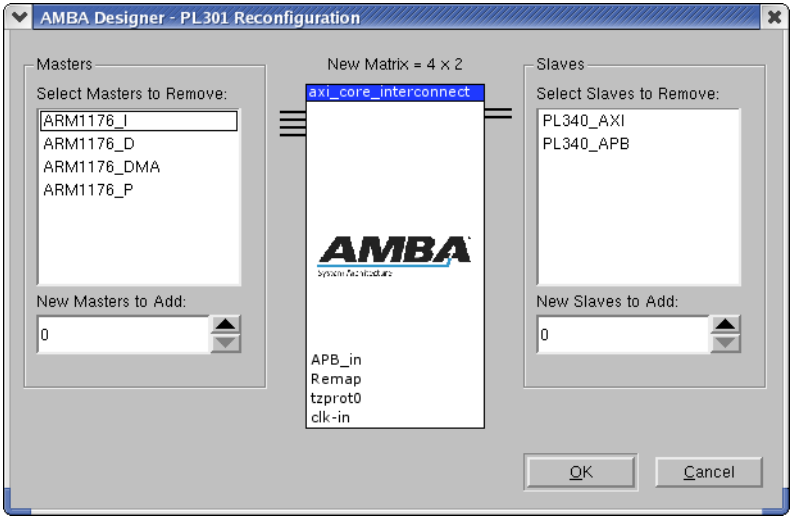


Figure 4-48 PL301 Reconfiguration dialog

In Figure 4-48 on page 4-52, the dialog consists of a Masters pane and a Slaves Pane and situated between these panes there is a graphical representation of the current configuration.

To add new masters or slaves then either:

Add Masters Enter the number of additional masters that you require in the New Masters to Add field.

Add Slaves Enter the number of additional slaves that you require in the New Slaves to Add field.

To remove masters or slaves from the current configuration then either:

Remove Masters Click on the name of the appropriate master in the Select Masters to Remove field. Use control-click when removing more than one master.

Remove Slaves Click on the name of the appropriate slave in the Select Slaves to Remove field. Use control-click when removing more than one slave.

5. Enter the number of additional masters or slaves that you require in the New Masters to Add and New Slaves to Add fields.
6. Click on any masters or slaves that you require to be removed using the Select Masters to Remove and Select Slaves to Remove fields.
7. Click on **OK** to create a new interconnect. The program appends a `_reconfigured` suffix to the original filename and by default it stores the file in the `Designs` directory.

4.8 Installing synthesis libraries

To synthesize the generated RTL for the HPM then you must set up some environment variables. Configure these variables for your chosen foundry process.

Table 4-3 lists the synthesis environment variables that you must configure.

Table 4-3 Synthesis environment variables for the HPM

Variable	Description
PL301_A3BM_SYNTH_DWROOT	Defines the location of your Synopsys DesignWare library directory
PL301_A3BM_SYNTH_LIB_NAME	Set to use your foundry process
PL301_A3BM_SYNTH_LIB_PATH	Defines the location of the file to be sourced for your synthesis tool
PL301_A3BM_SYNTH_LIB_HDL_DIR	Defines the location of your synthesis library directory
PL301_A3BM_SYNTH_LIB_HDL_FILE	Defines the name of your synthesis entity file

4.8.1 Synthesis of the 4x3 example

To synthesize the example configuration in *Example 4x3 interconnect* on page 4-4 then you must download and install the ARM Artisan SAGE-HS™ CL013G TSMC 0.13um libraries from the following location:

`http://www.arm.com`

Unpack the library and install it in a directory of your choice.

You must then configure the environment variables to point to the location of your installed library files. Table 4-4 lists example variable definitions for use with the SAGE-HS library files.

Table 4-4 Example synthesis environment variables for the HPM

Variable	Description
PL301_A3BM_SYNTH_DWROOT	/synopsys/synthesis/2004.12-SP5-4
PL301_A3BM_SYNTH_LIB_NAME	tsmc13_g_fsg_sage_hs
PL301_A3BM_SYNTH_LIB_PATH	/artisan/tsmc13g/sage-hs/2005q2v1/aci/sc-x/synopsys
PL301_A3BM_SYNTH_LIB_HDL_DIR	/artisan/tsmc13g/sage-hs/2005q2v1/aci/sc-x/verilog
PL301_A3BM_SYNTH_LIB_HDL_FILE	tsmc13_hs.v

4.9 Address map

The address map controls the routing of transactions to the slaves. You configure the address map for bus master components by using the Memory Map Editor dialog box. The functionality of the Memory Map Editor dialog box is described in the following sections:

- *Default address map*
- *Remap* on page 4-56
- *Remap examples* on page 4-57.

4.9.1 Default address map

Figure 4-49 shows the Memory Map Editor dialog box with an unconfigured address map for an HPM.

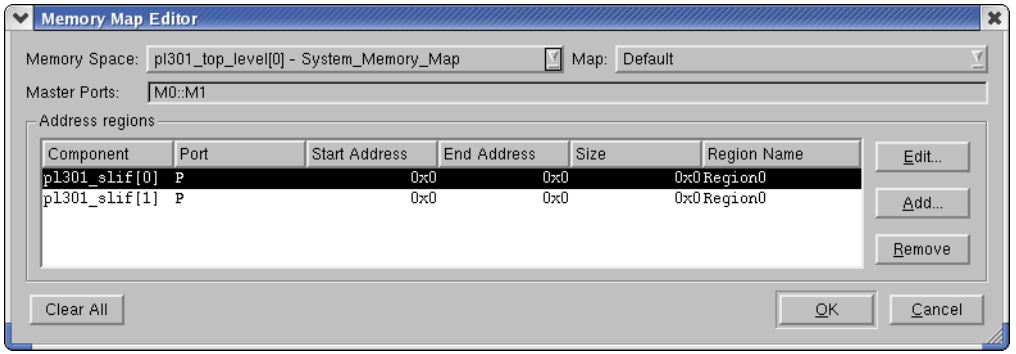


Figure 4-49 Unconfigured address map

In the Address regions pane for an HPM, the fields displayed are:

Component The name of the slave.

Start Address The start address for the selected Component and Region Name. The default setting is `0x0`.

End Address The end address for the selected Component and Region Name. The default setting is `0x0`.

Size The address size for the selected Component and Region Name. The default setting is `0x0`.

Note

When the Size field is `0x0` it indicates that the address region referenced by the Component and Region Name is not active.

Region Name The name of the address region. By default this is set to Region0. For each Component, you can define a maximum of 32 Region Names in each Map. If you use all ten remap address maps, then including the default map, a maximum number of 352 Region Names can be defined for each Component.

Note

See the *SoC Designer User Guide* for more information about using the Memory Map Editor.

4.9.2 Remap

The Memory Map Editor dialog box provides the Map drop-down list that enables you to access the remap address maps. Figure 4-50 shows the options available in the Map drop-down list.

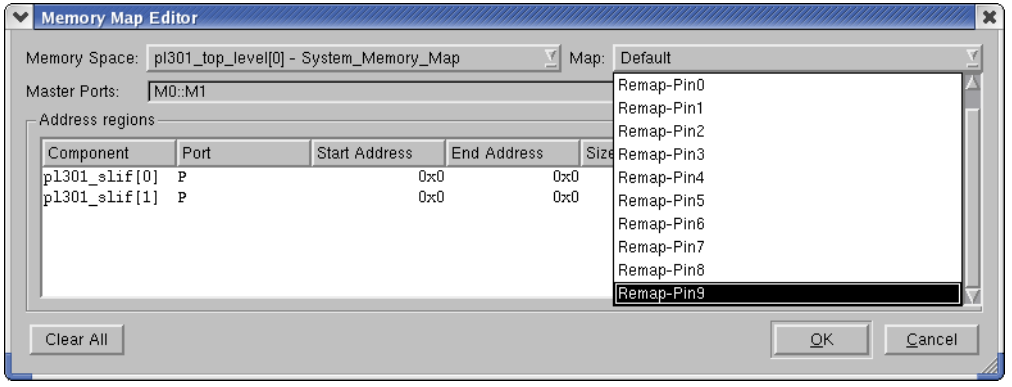


Figure 4-50 Map list

When Map is set to Default, the Memory Map Editor dialog displays the address map that is active when all the **REMAP[9:0]** signals are LOW.

When Map is set to Remap-Pin*n*, the Memory Map Editor dialog displays the address map that is active when the **REMAP[*n*]** signal is HIGH.

4.9.3 Remap examples

To clarify the use of the remap functionality when using the Memory Map Editor dialog box, the following sections describe some example configurations:

- *Adding address regions*
- *Moving address regions* on page 4-60.

Adding address regions

This example describes how to set up a slave with one remap address map. The slave is configured with two default address regions and one additional region is enabled during remap.

The example assumes that the **REMAP[0]** signal controls the remapping and therefore Remap-Pin0 is configured.

To create this example:

1. Select **File → New** from the Main menu. This creates a new Diagram Window.
2. Click on the **New Fabric IP** button on the Main Toolbar to open the Configure new PrimeCell IP dialog box.
3. In the Configure new PrimeCell IP dialog box:
   a. Click on the row containing a PrimeCell name of PL301.
   b. Click on **OK** to close the dialog box and open the New PL301 AXI-core Bus Matrix dialog box.
4. In the New PL301 AXI-core Bus Matrix dialog box:
   a. Enter `2` in the Number of Masters field.
   b. Enter `1` in the Number of Slaves field.
   c. Click on **OK** to close the dialog box.
5. Right-click on the pl301_top_level[0] component.
6. From the context menu, select the **AMBA Designer → PL301 Interconnect Address Map** option, to open the Memory Map Editor dialog.

To create the default address regions when all **REMAP** signals are inactive, then in the Memory Map Editor dialog box:

7. Using the Map pulldown window, select Default.
8. Click on **Edit...** to open the Edit Address Region dialog box. In the Edit Address Region dialog box:
   a. Enter `0xA0000000` in the Start field.
   b. Enter `0x1000` in the Size field.

c. Enter `Region0` in the Name field.
d. Click on **OK** to add this address region and close the dialog box.

9. Click on **Add...** to open the Edit Address Region dialog box. In the Edit Address Region dialog box:
   a. Enter `0xE0000000` in the Start field.
   b. Enter `0x1000` in the Size field.
   c. Enter `Region1` in the Name field.
   d. Click on **OK** to add this address region and close the dialog box.

This enters the second address range for slave 0, when no remap is selected. Figure 4-51 shows the base address map for slave 0.

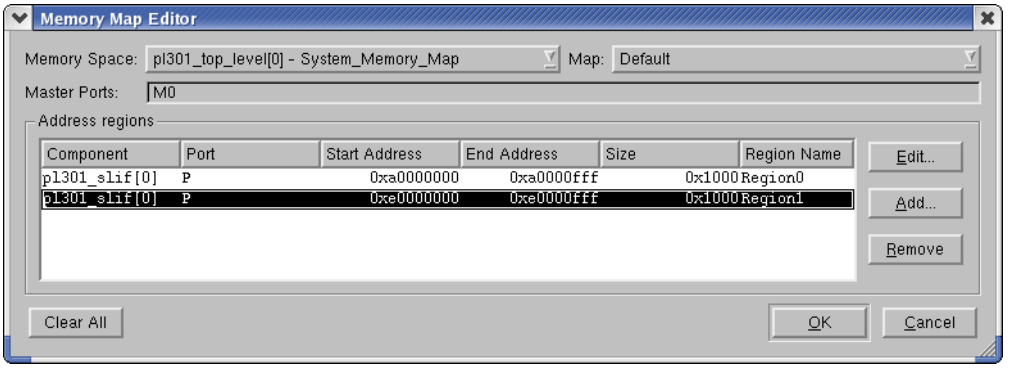


Figure 4-51 Default address map for remap example

To create the default address regions when **REMAP[0]** is active, then in the Memory Map Editor dialog box:

10. Using the Map pulldown window, select Remap-Pin0.
11. Click on **Edit...** to open the Edit Address Region dialog box. In the Edit Address Region dialog box:
    a. Enter `0xA0000000` in the Start field.
    b. Enter `0x1000` in the Size field.
    c. Enter `Region0` in the Name field.
    d. Click on **OK** to add this address region and close the dialog box.
12. Click on **Add...** to open the Edit Address Region dialog box. In the Edit Address Region dialog box:
    a. Enter `0xE0000000` in the Start field.
    b. Enter `0x1000` in the Size field.
    c. Enter `Region1` in the Name field.
    d. Click on **OK** to add this address region and close the dialog box.

To create an additional address region when **REMAP[0]** is active, then in the Memory Map Editor dialog box:

13. Click on **Add...** to open the Edit Address Region dialog box. In the Edit Address Region dialog box:
    a. Enter `0x20000000` in the Start field.
    b. Enter `0x1000` in the Size field.
    c. Enter `Region2` in the Name field.
    d. Click on **OK** to add this address region and close the dialog box.

Figure 4-52 shows the configured Memory Map Editor dialog box when Remap-Pin0 is selected.

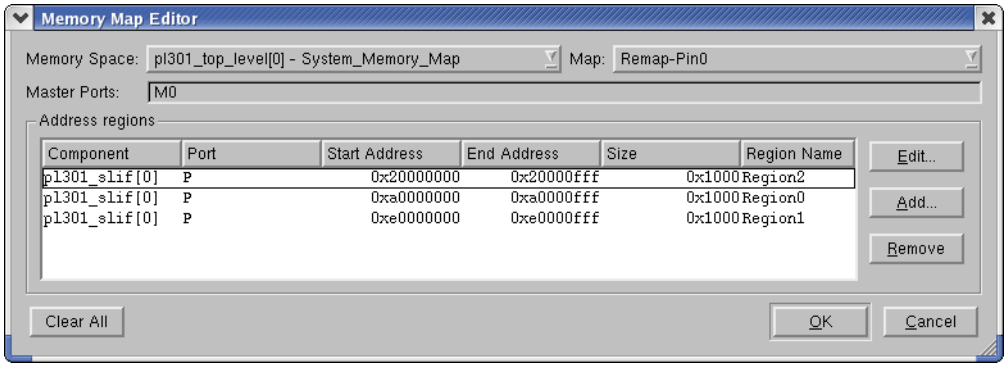


Figure 4-52 Remap Pin0

Figure 4-53 shows the address map that this example creates.

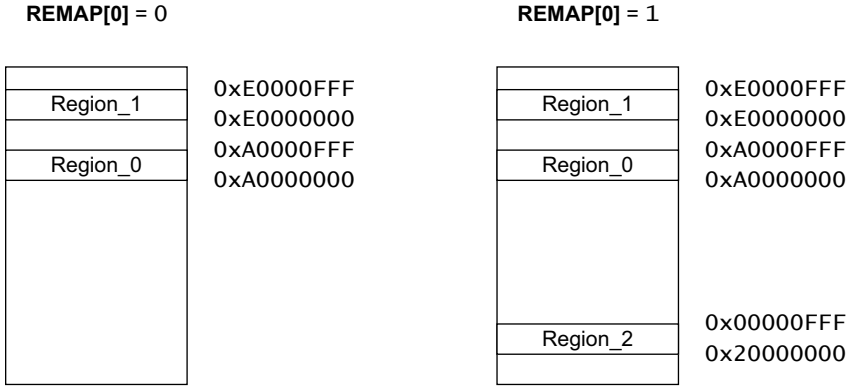


Figure 4-53 Remap Pin0 address map

Moving address regions

This example describes how to set up a slave with one remap address map. The slave is configured with two address regions, one of which is moved during remap.

The example assumes that **REMAP[4]** controls the remapping and therefore Remap-Pin4 is configured.

To create this example:

1. Select **File → New** from the Main menu. This creates a new Diagram Window.
2. Click on the **New Fabric IP** button on the Main Toolbar to open the Configure new PrimeCell IP dialog box.
3. In the Configure new PrimeCell IP dialog box:
   a. Click on the row containing a PrimeCell name of PL301.
   b. Click on **OK** to close the dialog box and open the New PL301 AXI-core Bus Matrix dialog box.
4. In the New PL301 AXI-core Bus Matrix dialog box:
   a. Enter `2` in the Number of Masters field.
   b. Enter `1` in the Number of Slaves field.
   c. Click on **OK** to close the dialog box.
5. Right-click on the pl301_top_level[0] component.
6. From the context menu, select the **AMBA Designer → PL301 Interconnect Address Map** option, to open the Memory Map Editor dialog.

To create the default address regions when all **REMAP** signals are inactive, then in the Memory Map Editor dialog box:

7. Using the Map pulldown window, select Default.
8. Click on **Edit...** to open the Edit Address Region dialog box. In the Edit Address Region dialog box:
   a. Enter `0xA0000000` in the Start field.
   b. Enter `0x1000` in the Size field.
   c. Enter `Region0` in the Name field.
   d. Click on **OK** to add this address region and close the dialog box.
9. Click on **Add...** to open the Edit Address Region dialog box. In the Edit Address Region dialog box:
   a. Enter `0xC0000000` in the Start field.
   b. Enter `0x1000` in the Size field.
   c. Enter `Region1` in the Name field.

d. Click on **OK** to add this address region and close the dialog box.

This enters the second address range for slave 0, when no remap is selected. Figure 4-54 shows the base address map for slave 0.

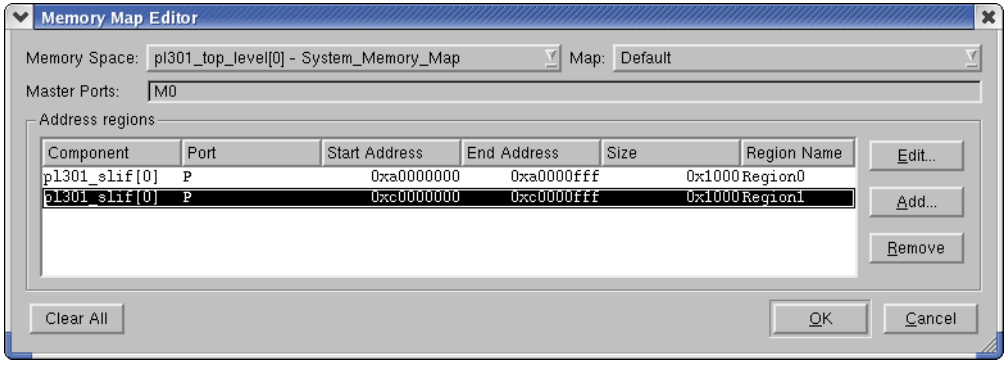


Figure 4-54 Default address map for remap move example

To create the default address region when **REMAP[4]** is active, then in the Memory Map Editor dialog box:

10. Using the Map pulldown window, select Remap-Pin4.
11. Click on **Edit...** to open the Edit Address Region dialog box. In the Edit Address Region dialog box:
    a. Enter `0xA0000000` in the Start field.
    b. Enter `0x1000` in the Size field.
    c. Enter `Region0` in the Name field.
    d. Click on **OK** to add this address region and close the dialog box.

To move address region Region1, when **REMAP[4]** is active, then in the Memory Map Editor dialog box:

12. Click on **Add...** to open the Edit Address Region dialog box. In the Edit Address Region dialog box:
    a. Enter `0x70000000` in the Start field.
    b. Enter `0x1000` in the Size field.
    c. Enter `Region1` in the Name field.
    d. Click on **OK** to add this address region and close the dialog box.

Figure 4-55 on page 4-62 shows the configured Memory Map Editor dialog box when Remap-Pin4 is selected.

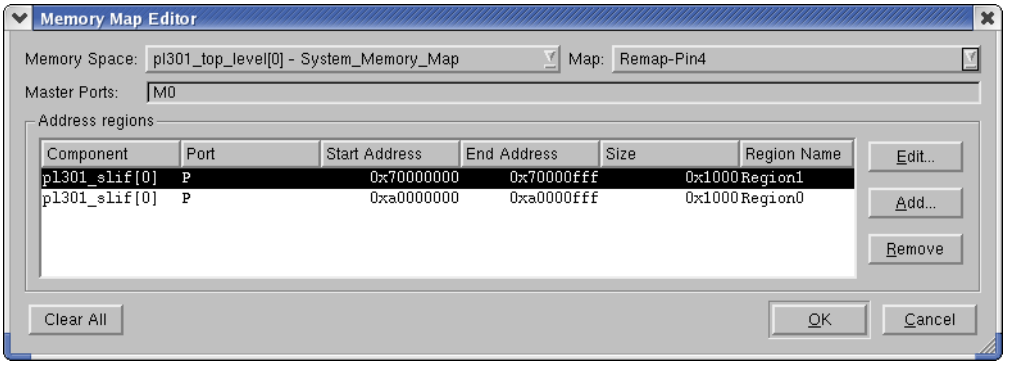


Figure 4-55 Remap Pin4

Figure 4-56 shows the address map that this example creates.

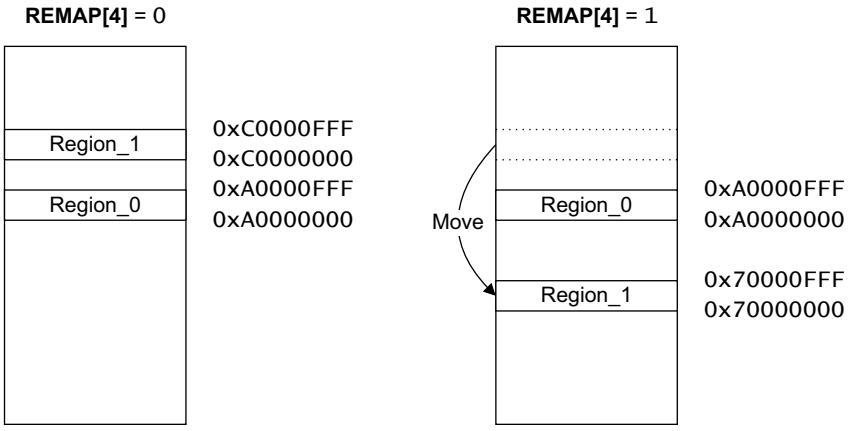


Figure 4-56 Remap move address map

4.10 System modeling and the address map

In AMBA Designer, when you enter the address map information for an HPM the program saves this information in an `.mxp` file. When AMBA Designer creates an interconnect model then for each master interface on the HPM it inserts an axi_address_holder component. The program uses the axi_address_holder component to store the address information for that interface. No address information is stored in the axi_core_interconnect component.

In SoC Designer Simulator, the HPM obtains its address data by polling the slave components that connect to its master interfaces. The HPM is then able to poll the axi_address_holder components and reconstruct its address map.

Figure 4-57 shows an interconnect model with the axi_address_holder components containing the address map information.

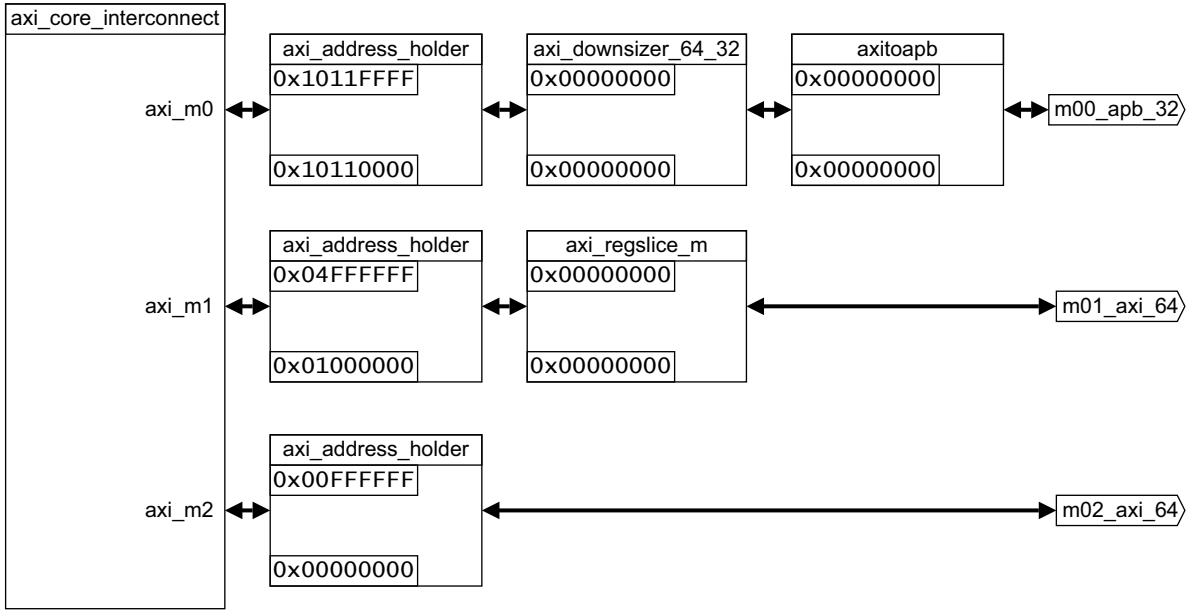


Figure 4-57 Address map in an interconnect model

4.11 Configuration options for the HPM

This section provides information about the complete range of configuration options in AMBA Designer, that are specific to the HPM. It contains the following sections:

- *Global configuration options for the HPM*
- *Master Params for the HPM* on page 4-65
- *Slave Params for the HPM* on page 4-68
- *Parameter naming cross reference* on page 4-75.

4.11.1 Global configuration options for the HPM

Table 4-5 lists the configuration options in AMBA Designer that configure the global configuration options of the HPM. You can access these options by selecting the pl301_top_level[0] component and using the **Object Parameters...** context menu.

Table 4-5 Global configuration options for the HPM

Configuration option	Range	Default value	Description
`model_apb_tz_enable`	`true` `false`	`false`	When set to `true`, AMBA Designer enables the display of the TrustZone ports on the AXI to APB protocol bridge models.
`model_max_slave_id_width`	Integer, 0-16	0	The maximum ID width that the HPM uses on its slave interfaces. The program uses this value, in conjunction with `total_masters`, to compute the width of the ID field required for the master interfaces on the HPM model.
`routing_address_width`[a]	Integer, 32-64	Assigned	Address bus width that the HPM uses for internal routing. The program sets this to equal the largest address width of any slave that connects to the HPM. You can alter the calculated default value but it must be greater than or equal to the largest address width used by a slave.
`routing_data_width`	32, 64, 128, or `Auto`	`Auto`	Data bus width that the HPM uses for internal routing. The program sets this to equal the largest data width of any slave that connects to the HPM. You can alter the calculated default value.
`total_masters`	Integer, 1-32	Assigned	Total number of masters that connect to the HPM.
`total_slaves`	Integer, 1-32	Assigned	Total number of slaves that connect to the HPM.
`user_signal_width`[a]	Integer, 0-32	0	Width of the AXI **xUSER** signals for all master and slaves. If this option is zero then these signals are not present.

a. See the *PrimeCell High-Performance Matrix (PL301) Technical Reference Manual* for more information.

4.11.2 Master Params for the HPM

Table 4-6 lists the Master Params configuration options in AMBA Designer that configure the slave interface configuration options of the HPM. You can access these options by selecting the appropriate pl301_maif[*n*] component and using the **Object Parameters...** context menu.

Table 4-6 Master Params for the HPM

Configuration option	Range	Default value	Description[a]
address_tie_off	64-bit hexadecimal, without 0x prefix	16'x0	Sets the value of the tie-off address bits. When the address_width of the master is less than the routing_address_width, the HPM prepends the address with the appropriate tie-off values.
address_width	Integer, 32-64	32	Set this value to the number of address bits that the master provides. This value must be less than or equal to the routing_address_width.
arbitration_order_init	Integer, 0-(total_masters-1)	0	Sets the priority of the master. The lower the value, the higher the priority of the master. Enables implementation of fixed-priority, round-robin, or concurrent arbitration schemes.
clock_domain_crossing	none fast slow async	none	The clock domain crossing bridge[b] options are: none: Master and HPM operate on the same clock domain so no bridge component is inserted. fast: Master operates at a higher clock frequency with respect to the HPM. A downwards synchronizing bridge is inserted. slow: Master operates at a lower clock frequency with respect to the HPM. An upwards synchronizing bridge is inserted. async: Master operates asynchronously to the HPM. An asynchronous bridge is inserted.
cyclic_scheme	single_slave unique_id higher_rank hybrid slave_per_id	single_slave	Scheme that the HPM implements to avoid cyclic dependency deadlock.

Table 4-6 Master Params for the HPM (continued)

Configuration option	Range	Default value	Description[a]
data_width	32, 64, or 128	64	Set this value to the number of data bits that the master provides. For APB masters then this value is limited to 32. If data_width is not the same as the routing_data_width then AMBA Designer inserts a downsizer or expander.
decode_register	true false	false	Provides a single register stage in the address decoder, but at the expense of extra latency.
id_width	Integer, 0-16	0	Number of bits for the ID bus of the master. If this option is zero, these signals are removed.
interface_protocol	axi ahb_lite_master ahb_lite_slave ahb_lite_mem	axi	Interface protocol of the master. The options are: axi — No protocol bridge is instantiated. ahb_lite_master — An AHB-Lite master to AXI protocol bridge is instantiated. ahb_lite_slave — An AHB-Lite slave to AXI protocol bridge is instantiated. ahb_lite_mem — An AHB-Lite to AXI protocol bridge is instantiated. This bridge is optimized for use with AHB memory controllers.

Table 4-6 Master Params for the HPM (continued)

Configuration option	Range	Default value	Description[a]
model_clock_divider_1	Integer, 1-*n*	1	Clock divider value that is used in the model for the clock domain crossing bridge. You must select an appropriate value[c] for your chosen clock_domain_crossing bridge type: fast — Sets the value of the clock divider for the master interface on the downwards synchronizing bridge. slow — Sets the value of the clock divider for the slave interface on the upwards synchronizing bridge. async — Sets the value of the clock divider for the slave interface on the asynchronous bridge. AMBA Designer does not use this configuration option when clock_domain_crossing is set to none.
model_clock_divider_2	Integer, 1-*n*	1	Clock divider value that is used in the model for the asynchronous bridge. You must select the divider value[c] so that the appropriate clock frequency is generated for the master interface on the asynchronous bridge. AMBA Designer only uses this configuration option when clock_domain_crossing is set to async.
port_name	String of less than 16 characters[d]	pl301_maif[*n*]	Name of the master port. Default values use an incrementing numerical suffix, *n*, that starts from 0. You can rename the default.

Table 4-6 Master Params for the HPM (continued)

Configuration option	Range	Default value	Description[a]
read_issuing_capability	Integer, 1-16	4	Read FIFO depth of the master. AMBA Designer configures the corresponding slave interface in the HPM to have an equivalent read acceptance capability.
registered_io	0, 1, or 2	0	Inserts the required number of register slices. The options are: 0 No register slice inserted 1 One register slice inserted as close as possible to the HPM 2 Two register slices inserted, one as close as possible to the HPM and one as close as possible to the AXI boundary.
write_issuing_capability	Integer, 1-16	4	Write FIFO depth of the master. AMBA Designer configures the corresponding slave interface in the HPM to have an equivalent write acceptance capability.

a. See the *PrimeCell High-Performance Matrix (PL301) Technical Reference Manual* for more information.
b. See *Clock domain crossing example* on page 4-28 for more information about clock domain crossing.
c. Calculate the clock divider setting using:

$$model\_clock\_divider\_x = \frac{Fs}{Fo}$$

Where:
F_S Simulation frequency applied to the bridge.
F_O Operating frequency of the component, that attaches to the bridge interface where the clock divider is located.
d. The name must adhere to the Verilog signal naming convention.

4.11.3 Slave Params for the HPM

The Slave Params are configured using several dialog boxes and are described in:

- *Object parameters* on page 4-69
- *Address map* on page 4-73
- *Sparse connect* on page 4-74.

Object parameters

Table 4-7 and Table 4-8 on page 4-72 list the Slave Params configuration options in AMBA Designer. You can access these options by selecting the appropriate pl301_slif[*n*] component and using the **Object Parameters...** context menu. These configuration options set the master interface configuration options of the HPM.

Table 4-7 Slave Params for the HPM

Configuration option	Range	Default value	Description[a]
address_tie_off	64-bit hexadecimal, without 0x prefix	16'x0	Sets the value of the tie-off address bits. When the routing_address_width is less than the address_width of the slave, the HPM prepends the address with the appropriate tie-off values.
address_width	Integer, 32-64	32	Set this value to the number of address bits that the slave requires.
clock_domain_crossing	none fast slow async	none	The clock domain crossing bridge[b] options are: none — Slave and HPM operate on the same clock domain so no bridge component is inserted. fast — Slave operates at a higher clock frequency with respect to the HPM. A downwards synchronizing bridge is inserted. slow — Slave operates at a lower clock frequency with respect to the HPM. An upwards synchronizing bridge is inserted. async — Slave operates asynchronously to the HPM. An asynchronous bridge is inserted.
combined_acceptance_capability	Integer, 1-32	4	Maximum number of active transactions that a slave can accept. You must specify this if read and write address storage is combined. AMBA Designer configures the corresponding master interface in the HPM to have an equivalent combined issuing capability.
cyclic_rank	Integer, 0-(total_slaves-1)	*n*, where *n* increments from 0	Number of the master interface. This must be unique. You can assign default values from 0 up to (total_slaves-1). This controls the cyclic priority and *Quality of Service* (QoS).

Table 4-7 Slave Params for the HPM (continued)

Configuration option	Range	Default value	Description[a]
data_width	32, 64, or 128	64	Set this value to the number of data bits that the slave requires. For APB slaves then this value is limited to 32. If data_width is not the same as the routing_data_width then AMBA Designer inserts a downsizer or expander.
interface_protocol	axi ahb_lite_master ahb_lite_slave apb[c]	axi	Interface protocol of the slave. The options are: axi — No protocol bridge is instantiated. ahb_lite_master — An AXI to AHB-Lite master protocol bridge is instantiated. ahb_lite_slave — An AXI to AHB-Lite slave protocol bridge is instantiated. apb — An AXI to APB protocol bridge is instantiated.
model_clock_divider_1	Integer, 1-*n*	1	Clock divider value that is used in the model for the clock domain crossing bridge. You must select an appropriate value[d] for your chosen clock_domain_crossing bridge type: fast — Sets the value of the clock divider for the slave interface on the upwards synchronizing bridge. slow — Sets the value of the clock divider for the master interface on the downwards synchronizing bridge. async — Sets the value of the clock divider for the slave interface on the asynchronous bridge. AMBA Designer does not use this configuration option when clock_domain_crossing is set to none.

Table 4-7 Slave Params for the HPM (continued)

Configuration option	Range	Default value	Description[a]
model_clock_divider_2	Integer, 1-*n*	1	Clock divider value that is used in the model for the asynchronous bridge. You must select the divider value[d] so that the appropriate clock frequency is generated for the master interface on the asynchronous bridge. AMBA Designer only uses this configuration option when clock_domain_crossing is set to async.
port_name	String of less than 16 characters[e]	pl301_slif[*n*]	Name of the slave port. Default values use an incrementing numerical suffix, *n*, that starts from 0. You can rename the default.
prog_qos	true false	false	When set to true, AMBA Designer configures the HPM to use the programmable QoS scheme.
registered_io	0, 1, or 2	0	Inserts the required number of register slices. The options are: 0 No register slice inserted. 1 One register slice inserted as close as possible to the HPM. 2 Two register slices inserted, one as close as possible to the HPM and one as close as possible to the AXI boundary.
speculative_access	true false	false	Set this parameter to true, if you configure a master to use the ahb_lite_mem interface protocol.
write_acceptance_capability	Integer, 1-16	4	Maximum number of active write transactions that the slave can accept at any one time. AMBA Designer configures the corresponding master interface in the HPM to have an equivalent write issuing capability.
write_interleave_depth	Integer, 1-16	1	Number of active write transactions for which the slave is capable of receiving data. AMBA Designer configures the corresponding master interface in the HPM to have an equivalent write interleave capability.

a. See the *PrimeCell High-Performance Matrix (PL301) Technical Reference Manual* for more information.
b. See *Clock domain crossing example* on page 4-28 for more information about clock domain crossing.

c. If you configure an APB bridge, you require the parameters that Table 4-8 on page 4-72 lists for each active APB peripheral slot.
d. Calculate the clock divider setting using:

$$model\_clock\_divider\_x = \frac{Fs}{Fo}$$

Where:

F_S Simulation frequency applied to the bridge.

F_O Operating frequency of the component, that attaches to the bridge interface where the clock divider is located.

e. The `name` must adhere to the Verilog signal naming convention.

For an APB slave, Table 4-8 lists the additional Slave Params configuration options in AMBA Designer that configure the master interface parameters of the HPM.

Table 4-8 Additional APB Slave Params for the HPM

Configuration option	Range	Default value	Description
`x_apb_slot_nn_name`	String of less than 16 characters[a]	`(None)`	The name of the APB slot. AMBA Designer appends this name to the APB signal names at the HPM top-level. If this slot is active then you must rename the default value.
`x_apb_slot_nn_version`	2.0 3.0	2.0	Configures the APB protocol version for the specified slot. You must set this to the same as the slave's APB protocol version.

a. The `name` must adhere to the Verilog signal naming convention.

Address map

The address map controls the routing of transactions to the master interfaces. You can access the address map editor by using the **AMBA Designer → PL301 Interconnect Address Map** context menu.

Table 4-9 lists the Memory Map Editor configuration options in AMBA Designer that configure the address map master interface configuration options of the HPM.

Table 4-9 Address map Slave Params for the HPM

Memory Map Editor dialog box	Range	Default value	Description
`Component`	See `port_name` in Table 4-7 on page 4-69	`pl301_slif[`*n*`]`	Displays the name of the slave.
`Start Address`	64-bit hexadecimal, with `0x` prefix	`16'x0`	Lower bound of the address region of the slave, for the address map selected by the `Map` setting.
`End Address`	64-bit hexadecimal, with `0x` prefix	`16'x0`	Upper bound of the address region of the slave, for the address map selected by the `Map` setting.
`Size`	64-bit hexadecimal, with `0x` prefix	`16'x0`	Size of the address region of the slave, for the address map selected by the `Map` setting.
`Map`	`Default` or `Remap-Pin`*n*, *n* = integer, 0-9	`Default`	Selects the address map to be displayed. When `Map` = `Default`, this corresponds to when the **REMAP** bus is LOW. When `Map` = `Remap-Pin`*n*, this corresponds to when the **REMAP[***n***]** signal is HIGH.
`Region Name`	String of characters	`Region0`	Defines a name for the address region specified by the `Start Address` and `End Address` parameters.

Sparse connect

Sparse connect is a feature that controls which masters are connected to which slaves. You configure this feature by using either:

- **AMBA Designer → PL301 Master Sparse Interconnect** context menu
- **AMBA Designer → PL301 System Sparse Interconnect** context menu.

Table 4-10 lists the configuration dialog boxes in AMBA Designer that configure the sparse connect master interface parameters of the HPM.

Table 4-10 Sparse connect Slave Params for the HPM

Configuration dialog box	Range[a]	Default value	Description
`PL301 Master Sparse Interconnect`	User-defined list of slaves	Master connects to all slaves	The PL301 Master Sparse Interconnect dialog box enables you to select the slaves that connect to a single master.
`PL301 System Sparse Interconnect`	`Disconnect All`, User-defined array of slaves and masters, or `Connect All`	`Connect All`	The PL301 System Sparse Interconnect dialog box enables you to select which masters the slave connects to. The preset settings are: `Disconnect All` Slave does not connect to any master. `Connect All` Slave connects to all masters.

a. Each master must connect to one or more slaves.

4.11.4 Parameter naming cross reference

In most cases, the configuration options that AMBA Designer provides for an HPM, use the same naming as the parameters that the HPM provides for the RTL.

Table 4-11 provides a cross reference that lists the differences in naming between the program and the generated RTL parameters.

Table 4-11 Naming cross reference list for the HPM

Configuration option in AMBA Designer	Description	RTL parameter in HPM
`read_issuing_capability`	AMBA Designer configures the corresponding slave interface in the HPM to have an equivalent read acceptance capability	`read_acceptance_capability`
`write_issuing_capability`	AMBA Designer configures the corresponding slave interface in the HPM to have an equivalent write acceptance capability	`write_acceptance_capability`
`combined_acceptance_capability`	AMBA Designer configures the corresponding master interface in the HPM to have an equivalent combined issuing capability	`combined_issuing_capability`
`write_acceptance_capability`	AMBA Designer configures the corresponding master interface in the HPM to have an equivalent write issuing capability	`write_issuing_capability`
`write_interleave_depth`	AMBA Designer configures the corresponding master interface in the HPM to have an equivalent write interleave capability	`write_interleave_capability`

Chapter 5
Configuring a DMC

This chapter describes how to use the program to configure a DMC. It contains the following sections:

5.1 About the PrimeCell DMC

The DMC is an AMBA compliant *System-on-Chip* (SoC) peripheral. You can configure the device to support the following memory types:

- SDRAM
- DDR
- Mobile DDR.

The device has one APB port to enable you to configure the DMC registers and one AXI port for the memory transfers to the external dynamic memory devices.

Figure 5-1 shows the main bus interfaces on the DMC and the connection to the memory devices and optional PrimeCell (PL220) *External Bus Interface* (EBI).

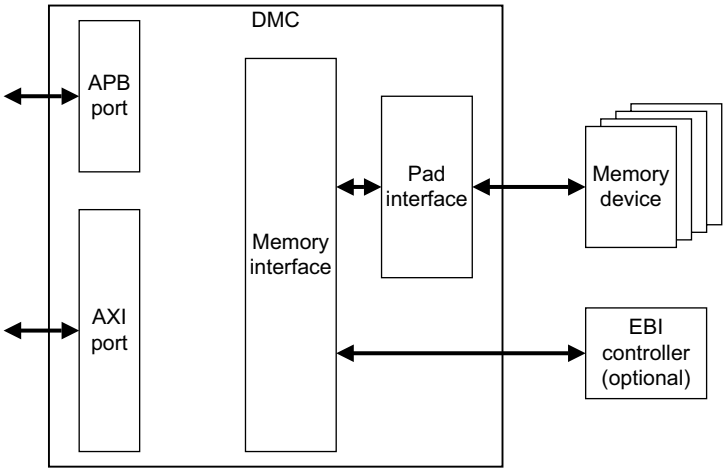


Figure 5-1 DMC system connections

See the *PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual* for more information about this device.

5.1.1 DMC parameters

AMBA Designer enables you to configure the following DMC parameters:

- AXI data width
- memory type
- memory data width
- number of memory devices
- arbiter depth
- number of exclusive access monitors

- command FIFO depth
- read FIFO depth
- write FIFO depth
- RID FIFO depth.

5.2 Configuration process

The DMC configuration process involves the following steps:

- *Configuring the DMC*
- *Adding the configured DMC to the component library* on page 5-7
- *RTL generation or system modeling* on page 5-8.

5.2.1 Configuring the DMC

Start AMBA Designer by using one of the following methods:

UNIX Enter `adcanvas` in a console window and press **Enter** on the keyboard.

Windows Click on the Windows **Start** button and then using the **Start** menu click on **All Programs → ARM → AMBA Designer**.

To configure the DMC:

1. Click on the **New Fabric IP** button on the Main Toolbar, as Figure 5-2 shows. This opens the Configure new PrimeCell IP dialog box.

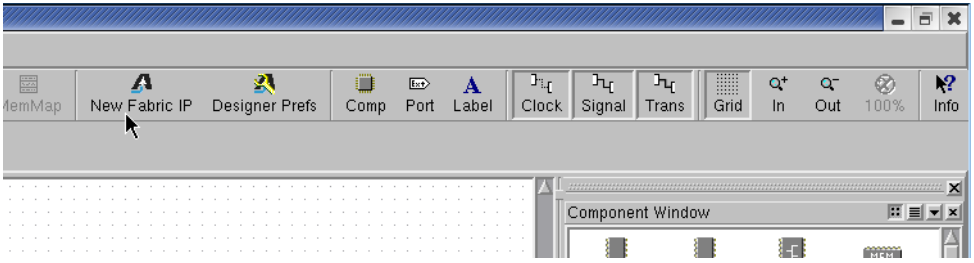


Figure 5-2 New Fabric IP selection

2. In the Configure new PrimeCell IP dialog box:
   a. Click on the row containing a PrimeCell name of PL340, as Figure 5-3 shows.

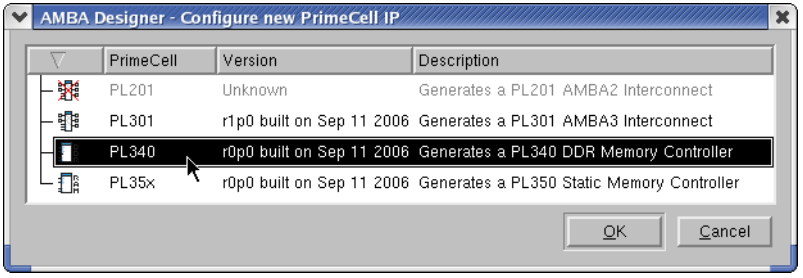


Figure 5-3 Selecting the PL340

b. Click on **OK** to close the dialog box and open the New PL340 DDR Memory Controller dialog box.

Figure 5-4 shows the New PL340 DDR Memory Controller dialog box.

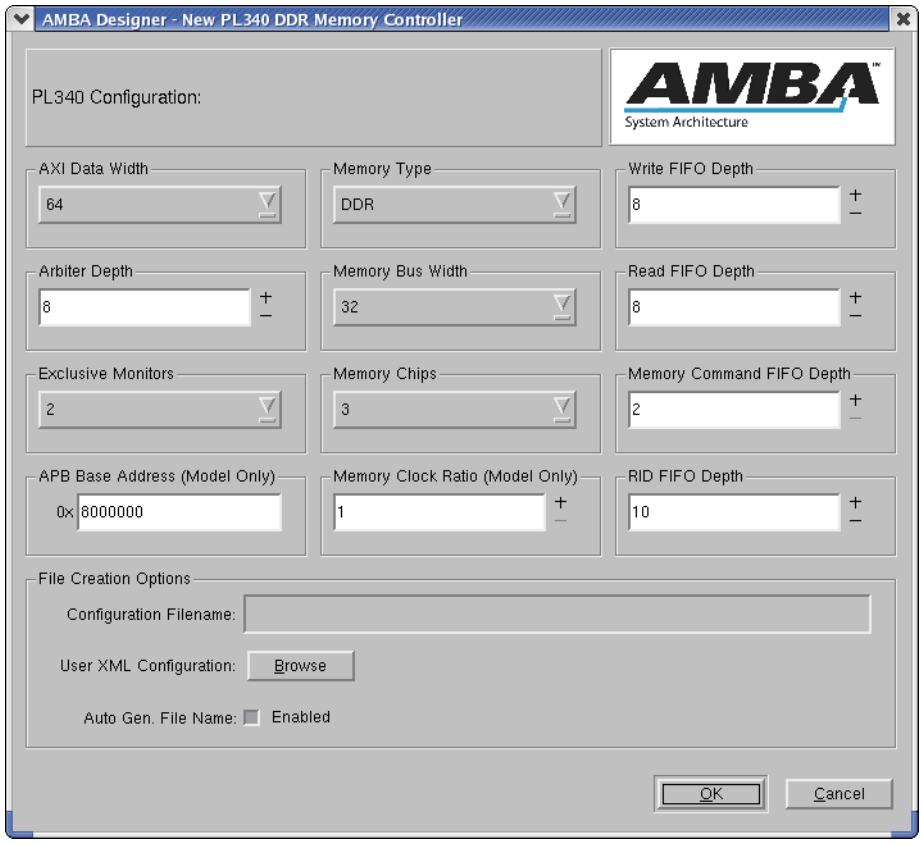


Figure 5-4 New PL340 DDR Memory Controller dialog

To configure the device you must modify the fields and drop-down lists that the New PL340 DDR Memory Controller dialog box provides. In Figure 5-4, the configuration options are:

AXI Data Width

This drop-down list selects the data bus width for the AXI slave interface.

Memory Type

This drop-down list selects the type of memory interface.

Memory Bus Width

This drop-down list selects the data bus width for the memory interface on the DMC.

Memory Chips

This drop-down list selects the number of chip selects that the memory interface on the DMC provides.

Arbiter Depth

This field defines the number of outstanding accesses that the device can support. The length of the arbiter queue defines the write interleaving capability of the DMC. Long arbiter depths enable better prioritization of reads in a busy system but this has an adverse affect on timing closure.

Exclusive Monitors

This drop-down list selects the number of exclusive access monitors to be implemented in the DMC.

Memory Command FIFO Depth

This field defines the number of pending memory interface commands that the device can support. Increasing the FIFO length can increase the latency for high priority reads and reduce the ability of the device to mask commands, such as precharges, when the memory data bus is busy. However, when you use the device in an asynchronous clocking mode, a FIFO depth of six is required to ensure continuous data flow on the memory data bus.

Write FIFO Depth

This field defines the number of buffers for the write FIFO and must be capable of storing an entire memory burst. Therefore, if you use a memory burst length of eight, the FIFO must have a minimum depth of eight for a SDRAM interface and a minimum depth of four for DDR and Mobile DDR interfaces. To ensure continuous data flow on the memory data bus the FIFO depth must be 2.5 times your intended memory burst length.

Read FIFO Depth

This field defines the number of buffers for the read FIFO and must be capable of storing an entire memory burst. Therefore, if you use a memory burst length of eight, the FIFO must have a minimum depth of eight for a SDRAM interface and a minimum depth of four for DDR and

Mobile DDR interfaces. To ensure continuous data flow on the memory data bus the FIFO depth must be 2.5 times your intended memory burst length.

RID FIFO Depth

This field defines the number of buffers for the read ID tag. To prevent the FIFO from becoming a performance bottleneck when issuing a series of very short reads, you must configure the FIFO depth using:

RID FIFO Depth = Read FIFO Depth + Command FIFO Depth

APB Base Address (Model Only)

This field defines the base address for the APB interface. SoC Designer Simulator uses this address when modeling the component.

Note

The base address might be updated depending on your system model configuration in SoC Designer Simulator. For example, if the DMC model connects to an HPM then the HPM provides the base address for the DMC.

Memory Clock Ratio (Model Only)

This field defines the integer value that the program uses to divide down the AXI clock frequency and generate the clock frequency for the memory interface.

3. Click on **OK** to save the configuration and close the dialog box.

Note

When AMBA Designer creates the filename then the tab name for the Diagram Window changes to **PL340*_xxx_abcd***, where:

- **PL340** is the prefix the program assigns to all DMC files that it creates
- ***xxx*** is a random generated sequence of alphanumeric characters
- ***abcd*** is a sequence of numbers that depends on the configuration of the device.

An asterisk (*) after the **PL340*_xxx_abcd*** name indicates that there are unsaved changes, either from an auto-saved file or the saved file.

5.2.2 Adding the configured DMC to the component library

To enable the program to use the configured DMC in modeling scenarios then you must add the component to the component library.

To add this component to the component library:

1. Right-click on the pl340_dmc[0] component to display the context menu.
2. Select **AMBA Designer → Add PL340 to Component Library...** from the context menu, as Figure 5-5 shows.

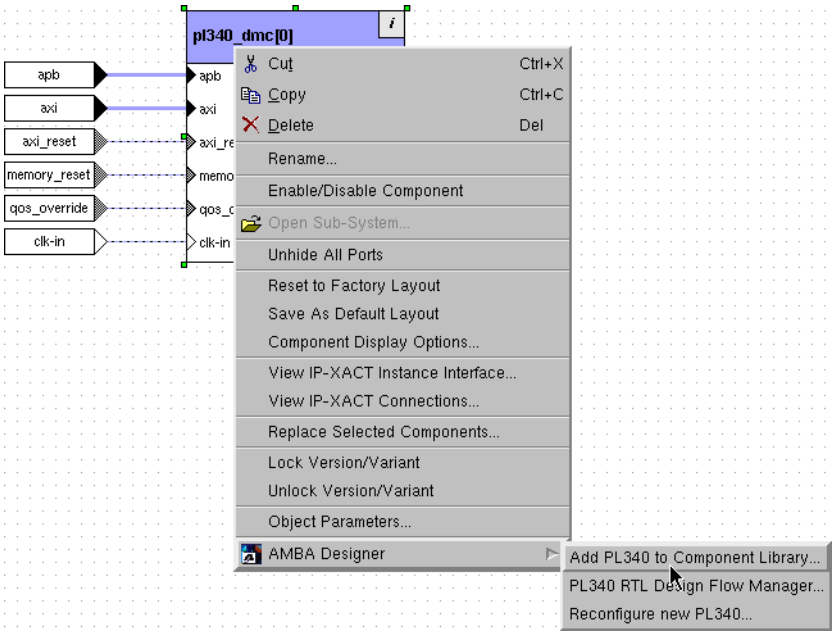


Figure 5-5 Adding the PL340 to the component library

AMBA Designer adds your PL340_*xxx_abcd* component model to the component library and inserts it into the list of components that the Component Window contains.

5.2.3 RTL generation or system modeling

After the configured component is created you can then either generate the RTL for the component or incorporate the component model into your system model to enable you to evaluate the performance of your system design.

The RTL and modeling process flows are described in:

- *Generating RTL for the DMC* on page 5-10
- Chapter 7 *Creating a System Model*.

Note

AMBA Designer only enables the generation of RTL, when the program runs on a UNIX operating system. You must also ensure that the AMBA Designer license file, usually named `licence.dat`, permits the program to generate RTL.

5.3 Generating RTL for the DMC

To generate RTL for the DMC, the program uses the RTL Design Flow Manager that Chapter 8 *RTL Design Flow* describes, but with the following exceptions described in:

- *Invoking the RTL Design Flow Manager*
- *Location of the Verilog and synthesis directories* on page 5-11
- *Denalisoft models* on page 5-12
- *OVL assertions* on page 5-12
- *Installing synthesis libraries* on page 5-12
- *Location of the summary file* on page 5-12
- *Location of the acceptable messages directory* on page 5-12.

5.3.1 Invoking the RTL Design Flow Manager

To access the RTL Design Flow Manager for your DMC:

1. Click on **Open** on the Main Toolbar to open a file browser dialog box.
2. Load the component that was previously created in *Adding the configured DMC to the component library* on page 5-7. This file is named `PL340_xxx_abcd.mxp` and is located in the default location:

   UNIX `home/user/.ARM/AMBA_Designer/Designs/`

   Note

   You must enter `home/user/.ARM` in the file browser dialog box because the dialog does not display the `.ARM` directory.

   Windows The program does not permit the generation of RTL on a Windows operating system. To generate RTL, you must use the UNIX version of the program.

   Where *user* is the login name of the current person running AMBA Designer.
3. Right-click on the pl340_dmc[0] component to display the context menu.
4. Select **AMBA Designer → PL340 RTL Design Flow Manager...** from the context menu, as Figure 5-6 on page 5-11 shows. This opens the RTL Design Flow Manager dialog box.

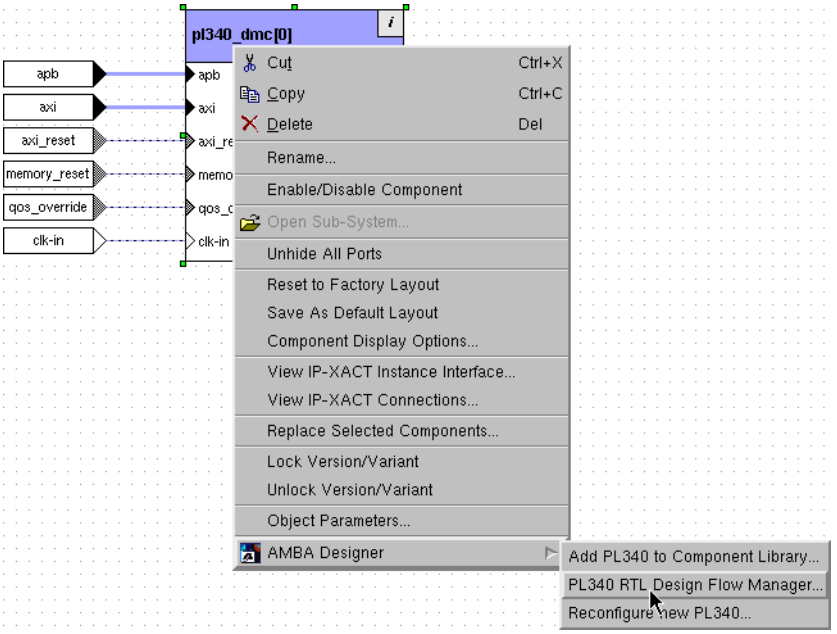


Figure 5-6 PL340 RTL Design Flow Manager selection

The program creates an XML configuration file. Figure 5-7 shows the location of the XML configuration file on a standard installation.

UNIX

```
Top-level directory/
  └─home/
      └─user/
          └─.ARM/
              └─AMBA_Designer/
                  └─Designs/
                      └─PL340_xxx_abcd_PL340_RTL/
```

Figure 5-7 Location of XML file for the DMC

To generate the RTL you must continue at step 5 in *Generate* on page 8-4.

5.3.2 Location of the Verilog and synthesis directories

The configured Verilog and synthesis are generated locally and stored in the `logical` and `implementation` directories. Figure 5-8 on page 5-12 shows the default location of these directories.

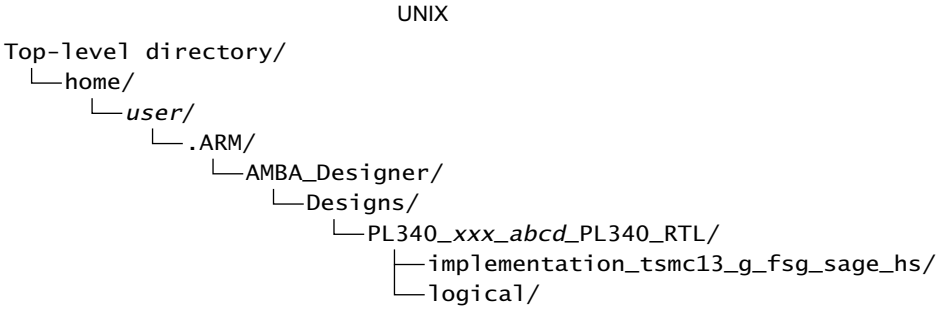


Figure 5-8 Location of Verilog and synthesis directories for the DMC

5.3.3 Denalisoft models

The program must have access to Denalisoft memory models to enable it to simulate the DMC. See the DMC installation package for information about how to configure the environment variable that AMBA Designer requires.

5.3.4 OVL assertions

The DMC does not support the use of OVL assertions during simulation. Therefore, you must ensure that the **Use OVL Assertions** check box in the RTL Design Flow Manager Preferences dialog box is not selected.

5.3.5 Installing synthesis libraries

To synthesize the generated RTL for the DMC then you must set up some environment variables. Configure these variables for your chosen foundry process. See the DMC installation package for information about how to configure the environment variables that AMBA Designer requires.

5.3.6 Location of the summary file

AMBA Designer does not generate a summary report file because the current release of the DMC does not support this feature.

5.3.7 Location of the acceptable messages directory

AMBA Designer does not generate an acceptable messages directory because the current release of the DMC does not support this feature.

5.4 Reconfiguring a DMC

The program provides a reconfiguration option to enable you to modify a previously configured DMC.

To reconfigure a DMC:

1. Click on **Open** on the Main Toolbar to open a file browser dialog box.
2. Load a DMC component that you have previously created. These files are named `PL340_xxx_abcd.mxp` and are located in the default location:

   UNIX `home/user/.ARM/AMBA_Designer/Designs/`

   Note

   You must enter `home/user/.ARM` in the file browser dialog box because the dialog does not display the `.ARM` directory.

   Windows `C:\Documents and Settings\user\My Documents\AMBA_Designer\Designs\`

   Where *user* is the login name of the current person running AMBA Designer.
3. Right-click on the pl340_dmc[0] component to display the context menu.
4. Select **AMBA Designer → Reconfigure new PL340...** from the context menu, as Figure 5-9 on page 5-14 shows. This opens the PL340 reconfiguration dialog box.

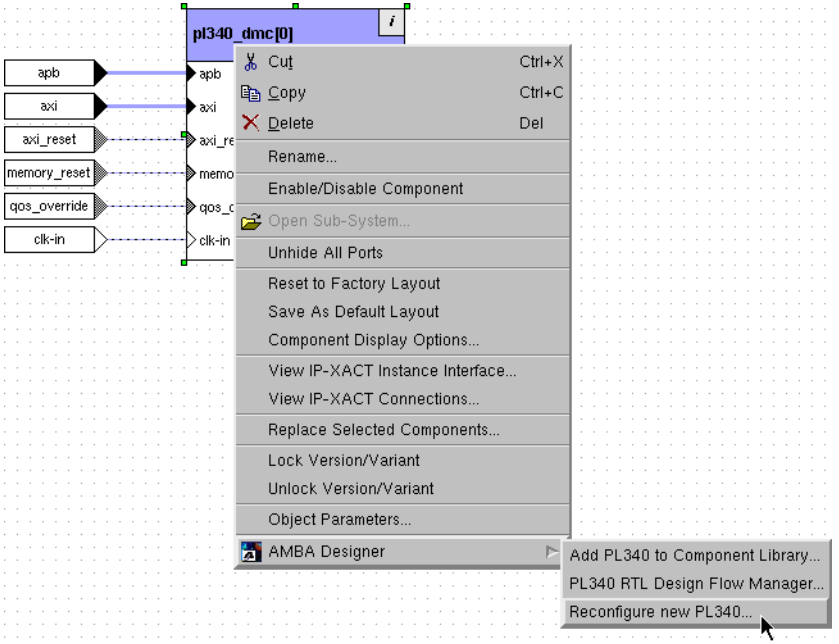


Figure 5-9 Reconfigure PL340 selection

Figure 5-10 on page 5-15 shows the PL340 reconfiguration dialog box.

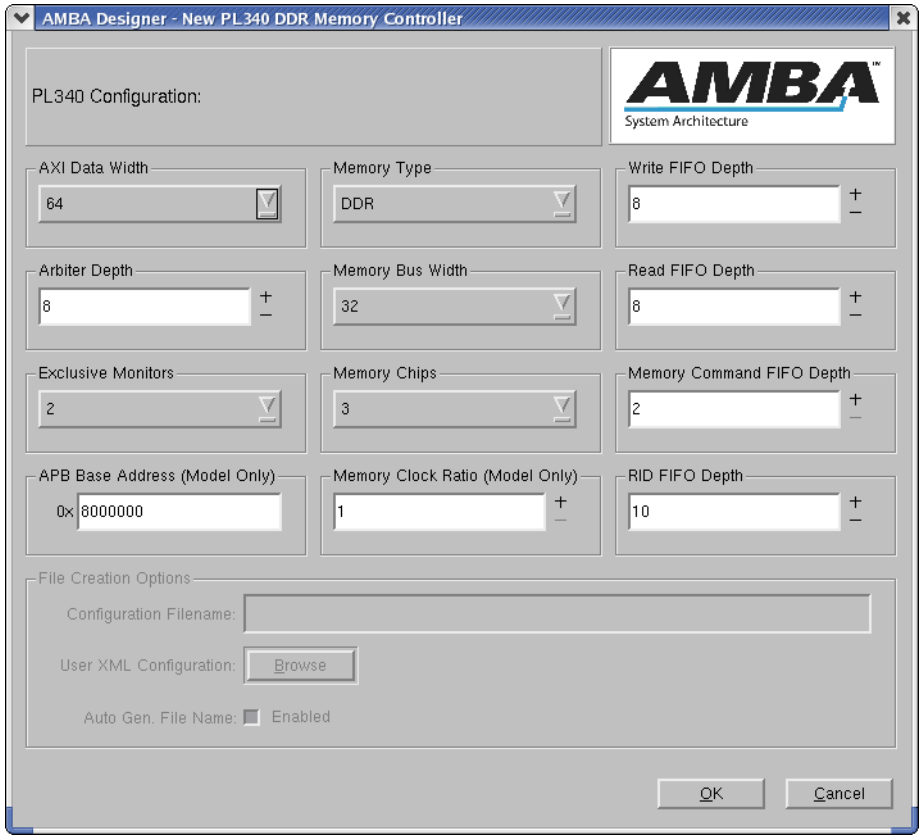


Figure 5-10 PL340 reconfiguration dialog

Note

In Figure 5-10:

- The configuration options are identical to those described for Figure 5-4 on page 5-5.
- The File Creation Options pane is grayed-out because this feature is not available during reconfiguration.

To modify the DMC:

5. Enter the modifications you require using the fields and drop-down lists that the dialog box provides.

6. Click on **OK** to create a new DMC. The program appends a `_reconfigured` suffix to the original filename and by default it stores the file in the `Designs` directory.

5.5 Configuration options for the DMC

Table 5-1 lists the configuration options in AMBA Designer that configure the configuration options of the DMC.

The configuration options are accessed by selecting the pl340_dmc[0] component and using the **Object Parameters...** context menu.

Table 5-1 Configuration options for the DMC

Configuration option	Range	Default value	Description
Arbiter Depth	Integer, 4-12	8	Number of outstanding accesses that the DMC must support.
AXI Data Width	32, 64, or 128	64	Size in bits of the AXI data bus.
Command FIFO Depth	Integer, 2-6	2	Number of buffers available for the command data.
Exclusive Monitors	0, 1, 2, 4	2	Number of monitors available for monitoring exclusive accesses on the AXI port.
Memory Chips	Integer, 1-4	3	Total number of dynamic memory devices that connect to the DMC.
Memory Data Width	16, 32, or 64	32	Size in bits of the data bus width for the memory interface on the DMC.
Memory Type	DDR SDR Mobile DDR	DDR	Type of dynamic memory device that connects to the pad interface on the DMC. All the memory devices must be one of the following types: SDR Single data rate SDRAM. DDR Double data rate SDRAM. Mobile DDR Mobile double data rate SDRAM.
RData FIFO Depth	Integer, 4-16	8	Number of buffers available for the read data.
RID FIFO Depth[a]	Integer, 4-20	10	Number of buffers available for the read ID tag.
WData FIFO Depth	Integer, 4-16	8	Number of buffers available for the write data.

Table 5-1 Configuration options for the DMC (continued)

Configuration option	Range	Default value	Description
The following parameters are only used when modeling the component in SoC Designer Simulator			
`APB Base Address`	32-bit hexadecimal, without `0x` prefix	`08000000`	The base address of the APB interface.
`Memory Clock Ratio`	Integer, 1-255	1	The number used to divide down the AXI clock frequency. The memory interface uses this generated clock frequency.

a. The DMC component model does not currently support this parameter. However, the program saves this parameter in the XML configuration file for the DMC and therefore it uses this parameter when it generates the RTL.

Chapter 6
Configuring an SMC

This chapter describes how to use the program to configure an SMC. It contains the following sections:

6.1 About the PrimeCell SMC

The SMC is an AMBA compliant SoC peripheral. The SMC consists of several preconfigured devices that use the PL35*x* model name and support one or two memory interfaces of type SRAM or NAND. The SMC series consists of the following devices:

PL351 NAND memory interface.

PL352 SRAM memory interface.

PL353 SRAM and NAND memory interfaces.

PL354 SRAM and SRAM memory interfaces.

The device has one APB port to enable you to configure the SMC registers and one AXI port for the memory transfers to the external static memory devices.

Figure 6-1 shows the main bus interfaces on the SMC and the connection to the memory devices and optional PrimeCell (PL220) EBI.

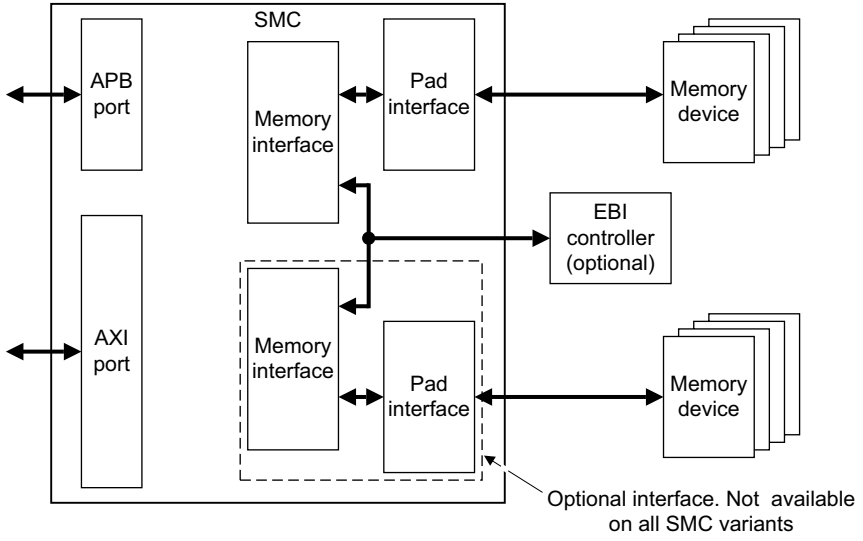


Figure 6-1 SMC system connections

See the *PrimeCell Static Memory Controller (PL350 series) Technical Reference Manual* for more information about this device.

6.1.1 SMC parameters

AMBA Designer enables you to configure the following SMC parameters:

- AXI data width
- memory data width

- number of memory devices
- command FIFO depth
- read FIFO depth
- write FIFO depth.
- inclusion of an entry block pipeline stage, for SMCs with two memory interfaces
- number of exclusive access monitors.

6.2 Configuration process

The SMC configuration process involves the following steps:

- *Configuring the SMC*
- *Adding the configured SMC to the component library* on page 6-8
- *RTL generation or system modeling* on page 6-8.

6.2.1 Configuring the SMC

Start AMBA Designer by using one of the following methods:

UNIX Enter `adcanvas` in a console window and press **Enter** on the keyboard.

Windows Click on the Windows **Start** button and then using the **Start** menu click on **All Programs → ARM → AMBA Designer**.

To configure the SMC:

1. Click on the **New Fabric IP** button on the Main Toolbar, as Figure 6-2 shows. This opens the Configure new PrimeCell IP dialog box.

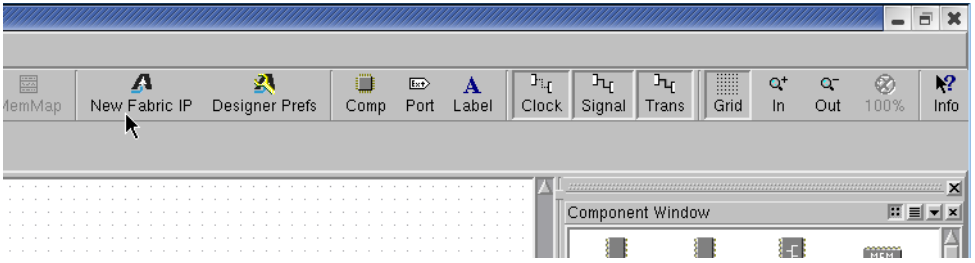


Figure 6-2 New Fabric IP selection

2. In the Configure new PrimeCell IP dialog box:
   a. Click on the row containing a PrimeCell name of PL35x, as Figure 6-3 shows.

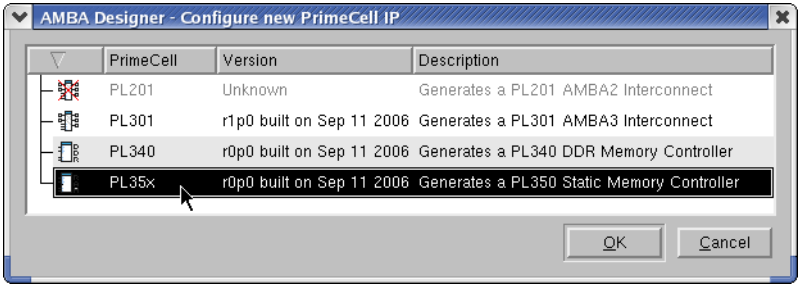


Figure 6-3 Selecting the PL35x

b. Click on **OK** to close the dialog box and open the New PL350 DDR Memory Controller dialog box.

Figure 6-4 shows the New PL350 DDR Memory Controller dialog box.

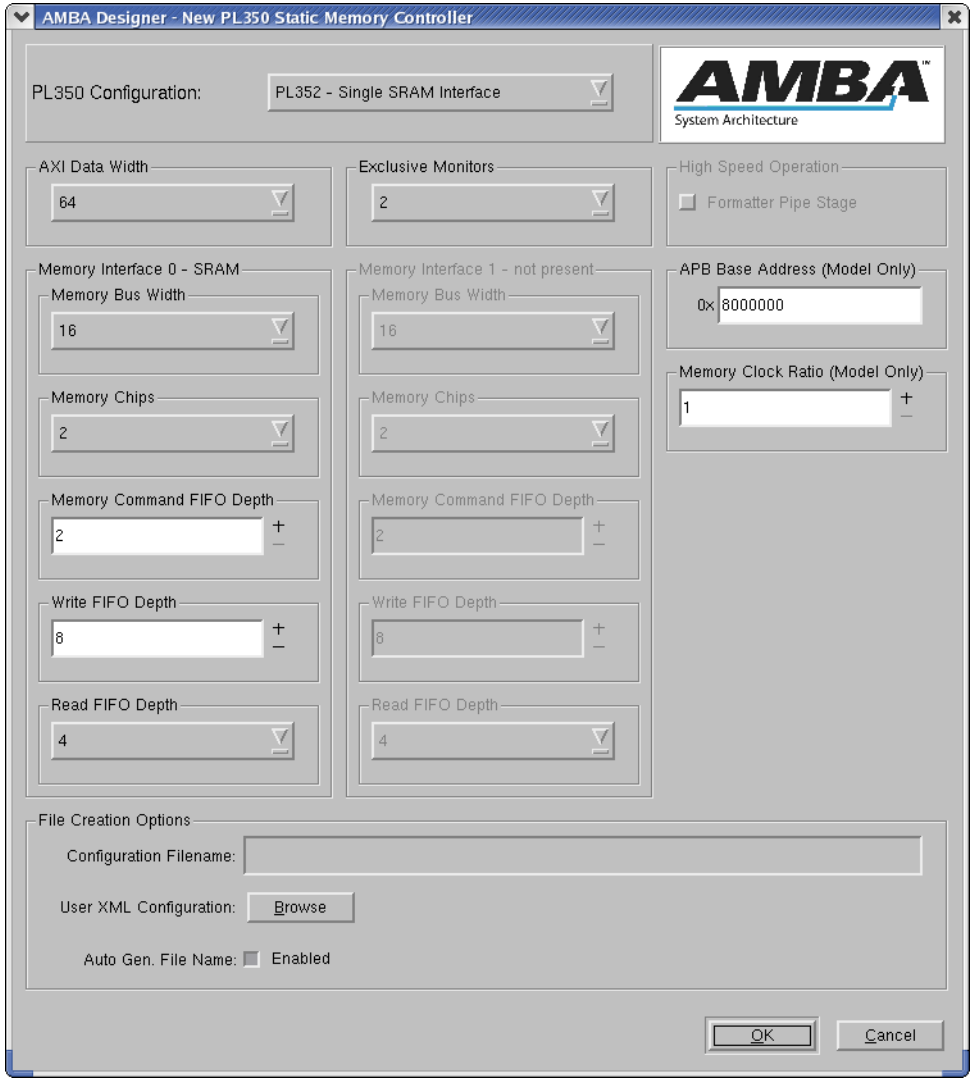


Figure 6-4 New PL350 DDR Memory Controller dialog

To configure the device you must modify the fields and drop-down lists that the New PL350 DDR Memory Controller dialog box provides. In Figure 6-4 on page 6-5, the configuration options are:

PL350 Configuration

This drop-down list selects the SMC variant to configure. The possible options are:

- PL351 - Single NAND Interface
- PL352 - Single SRAM Interface
- PL353 - SRAM and NAND Interfaces
- PL354 - Dual SRAM Interfaces.

Note

The list only displays the SMC variants that you have licensed.

If you choose an SMC variant that only has one memory interface then the options in the Memory Interface 1 pane are not applicable and the program grays them out.

AXI Data Width

This drop-down list selects the data bus width for the AXI slave interface.

Exclusive Monitors

This drop-down list selects the number of exclusive access monitors to be implemented in the SMC.

Memory Bus Width

This drop-down list selects the data bus width for the memory interface on the SMC.

Memory Chips

This drop-down list selects the number of chip selects that the memory interface on the SMC provides.

Memory Command FIFO Depth

This field defines the number of pending memory interface commands that the device can support.

Write FIFO Depth

This field defines the number of buffers for the write FIFO and must be capable of storing an entire memory burst. To ensure continuous data flow on the memory data bus the FIFO depth must be 2.5 times your intended memory burst length.

Read FIFO Depth

This field defines the number of buffers for the read FIFO and must be capable of storing an entire memory burst. To ensure continuous data flow on the memory data bus the FIFO depth must be 2.5 times your intended memory burst length.

Formatter Pipe Stage

Click on this check box, in the High Speed Operation pane, to include an entry block pipeline stage.

Note

This option is only available for SMC variants that have two memory interfaces.

APB Base Address (Model Only)

This field defines the base address for the APB interface. SoC Designer Simulator uses this address when modeling the component.

Note

The base address might be updated depending on your system model configuration in SoC Designer Simulator. For example, if the SMC model connects to an HPM then the HPM provides the base address for the SMC.

Memory Clock Ratio (Model Only)

This field defines the integer value that the program uses to divide down the AXI clock frequency and generate the clock frequency for the memory interfaces.

3. Click on **OK** to save the configuration and close the dialog box.

Note

When AMBA Designer creates the filename then the tab name for the Diagram Window changes to **PL350*_xxx_abcde***, where:

- **PL350** is the prefix the program assigns to all SMC files that it creates
- ***xxx*** is a random generated sequence of alphanumeric characters
- ***abcde*** is a sequence of numbers that depends on the configuration of the device.

An asterisk (*) after the **PL350*_xxx_abcde*** name indicates that there are unsaved changes, either from an auto-saved file or the saved file.

6.2.2 Adding the configured SMC to the component library

To enable the program to use the configured SMC in modeling scenarios then you must add the component to the component library.

To add this component to the component library:

1. Right-click on the pl35*x*_smc[0] component to display the context menu.
2. Select **AMBA Designer → Add PL350 to Component Library...** from the context menu, as Figure 6-5 shows.

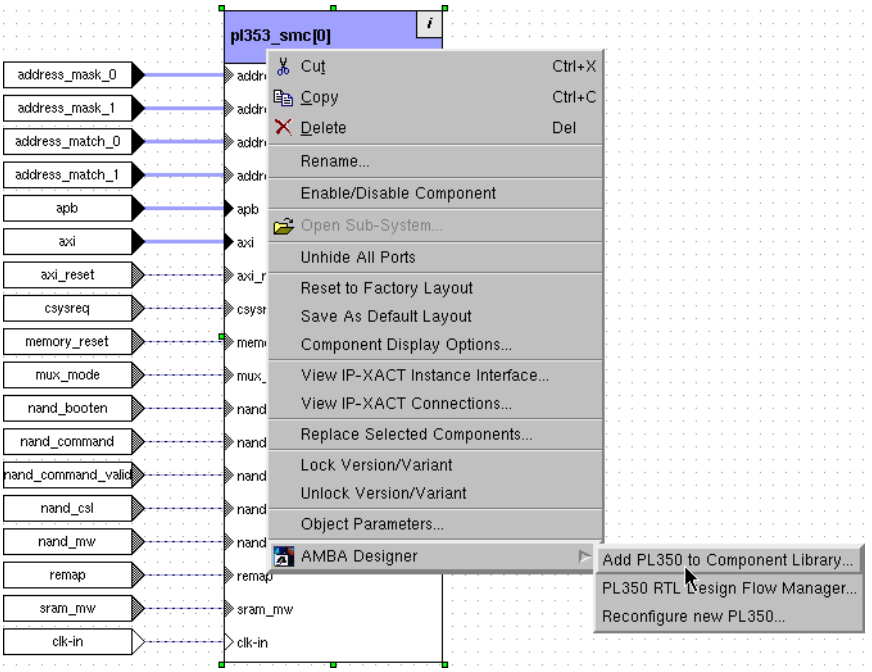


Figure 6-5 Adding the PL350 to the component library

AMBA Designer adds your PL35*x*_*xxx*_*abcde* component model to the component library and inserts it into the list of components that the Component Window contains.

6.2.3 RTL generation or system modeling

After the configured component is created you can then either generate the RTL for the component or incorporate the component model into your system model to enable you to evaluate the performance of your system design.

The RTL and modeling process flows are described in:

- *Generating RTL for the SMC* on page 6-10
- Chapter 7 *Creating a System Model*.

Note

AMBA Designer only enables the generation of RTL, when the program runs on a UNIX operating system. You must also ensure that the AMBA Designer license file, usually named `licence.dat`, permits the program to generate RTL.

6.3 Generating RTL for the SMC

To generate RTL for the SMC, the program uses the RTL Design Flow Manager that Chapter 8 *RTL Design Flow* describes, but with the following exceptions described in:

- *Invoking the RTL Design Flow Manager*
- *Location of the Verilog and synthesis directories* on page 6-11
- *Denalisoft models* on page 6-12
- *OVL assertions* on page 6-12
- *Installing synthesis libraries* on page 6-12
- *Location of the summary file* on page 6-12
- *Location of the acceptable messages directory* on page 6-13.

6.3.1 Invoking the RTL Design Flow Manager

To access the RTL Design Flow Manager for your SMC:

1. Click on **Open** on the Main Toolbar to open a file browser dialog box.
2. Load the component that was previously created in *Adding the configured SMC to the component library* on page 6-8. This file is named `PL350_xxx_abcde.mxp` and is located in the default location:

   UNIX `home/user/.ARM/AMBA_Designer/Designs/`

   Note

   You must enter `home/user/.ARM` in the file browser dialog box because the dialog does not display the `.ARM` directory.

   Windows The program does not permit the generation of RTL on a Windows operating system. To generate RTL, you must use the UNIX version of the program.

   Where *user* is the login name of the current person running AMBA Designer.
3. Right-click on the pl35*x*_smc[0] component to display the context menu. Where the number *x* signifies the variant of the SMC you are using.
4. Select **AMBA Designer → PL350 RTL Design Flow Manager...** from the context menu, as Figure 6-6 on page 6-11 shows. This opens the RTL Design Flow Manager dialog box.

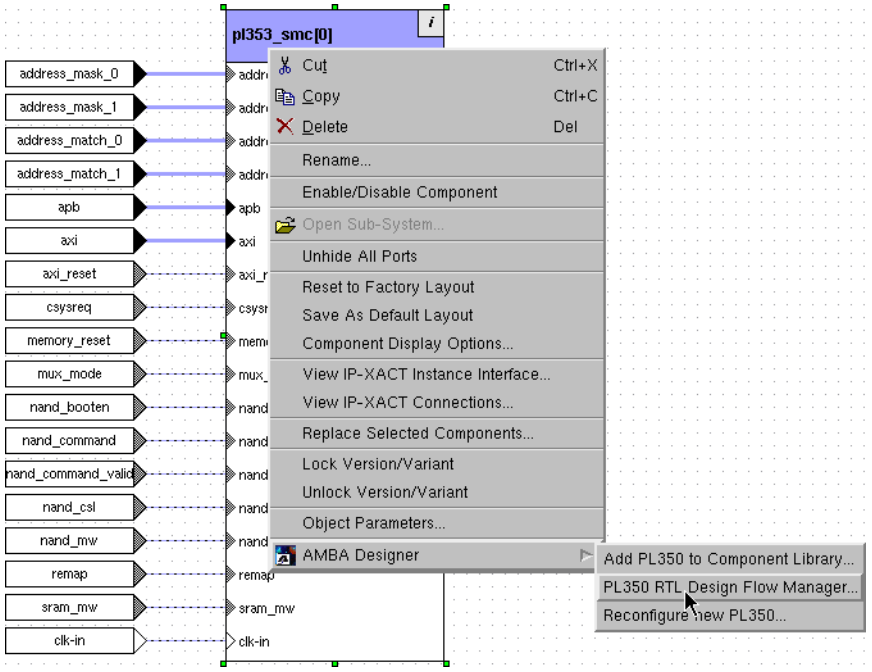


Figure 6-6 PL350 RTL Design Flow Manager selection

The program creates an XML configuration file. Figure 6-7 shows the location of the XML configuration file on a standard installation.

UNIX

```
Top-level directory/
  └─home/
      └─user/
          └─.ARM/
              └─AMBA_Designer/
                  └─Designs/
                      └─PL350_xxx_abcde_PL35x_RTL/
```

Figure 6-7 Location of XML file for the SMC

To generate the RTL you must continue at step 5 in *Generate* on page 8-4.

6.3.2 Location of the Verilog and synthesis directories

The configured Verilog and synthesis are generated locally and stored in the `logical` and `implementation` directories. Figure 6-8 on page 6-12 shows the location of these directories.

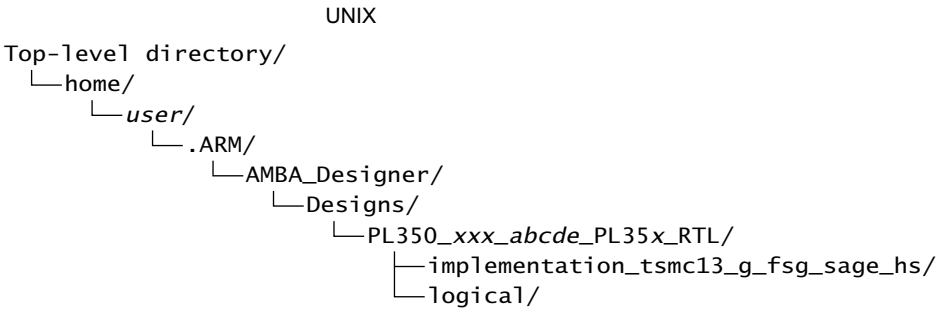


Figure 6-8 Location of Verilog and synthesis directories for the SMC

The naming of the parent directory for the `logical` and `implementation` directories is dependent on the chosen SMC variant. For example, when using the PL353 the parent directory is named `PL350_xxx_abcde_PL353_RTL`.

6.3.3 Denalisoft models

The program must have access to Denalisoft memory models to enable it to simulate the SMC. See the SMC installation package for information about how to configure the environment variable that AMBA Designer requires.

6.3.4 OVL assertions

The SMC does not support the use of OVL assertions during simulation. Therefore, you must ensure that the **Use OVL Assertions** check box in the RTL Design Flow Manager Preferences dialog box is not selected.

6.3.5 Installing synthesis libraries

To synthesize the generated RTL for the SMC then you must set up some environment variables. Configure these variables for your chosen foundry process. See the SMC installation package for information about how to configure the environment variables that AMBA Designer requires.

6.3.6 Location of the summary file

AMBA Designer does not generate a summary report file because the current release of the SMC does not support this feature.

6.3.7 Location of the acceptable messages directory

AMBA Designer does not generate an acceptable messages directory because the current release of the SMC does not support this feature.

6.4 Reconfiguring an SMC

The program provides a reconfiguration option to enable you to modify a previously configured SMC.

To reconfigure an SMC:

1. Click on **Open** on the Main Toolbar to open a file browser dialog box.
2. Load an SMC component that you have previously created. These files are named `PL350_xxx_abcde.mxp` and are located in the default location:

   UNIX `home/user/.ARM/AMBA_Designer/Designs/`

   Note

   You must enter `home/user/.ARM` in the file browser dialog box because the dialog does not display the `.ARM` directory.

   Windows `C:\Documents and Settings\user\My Documents\AMBA_Designer\Designs\`

   Where *user* is the login name of the current person running AMBA Designer.
3. Right-click on the pl35*x*_dmc[0] component to display the context menu.
4. Select **AMBA Designer → Reconfigure new PL350...** from the context menu, as Figure 6-9 on page 6-15 shows. This opens the PL350 reconfiguration dialog box.

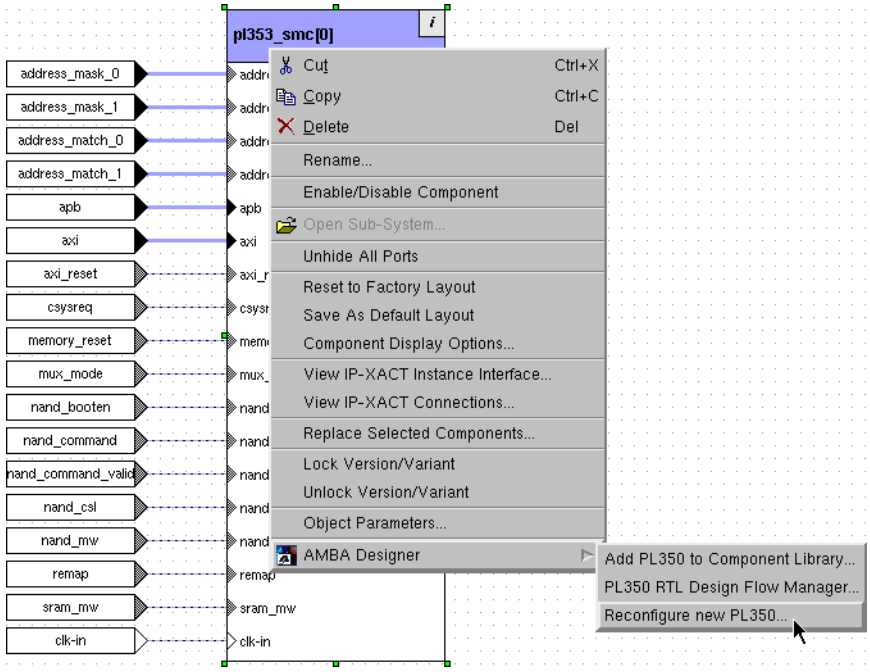


Figure 6-9 Reconfigure PL350 selection

Figure 6-10 on page 6-16 shows the PL350 reconfiguration dialog box.

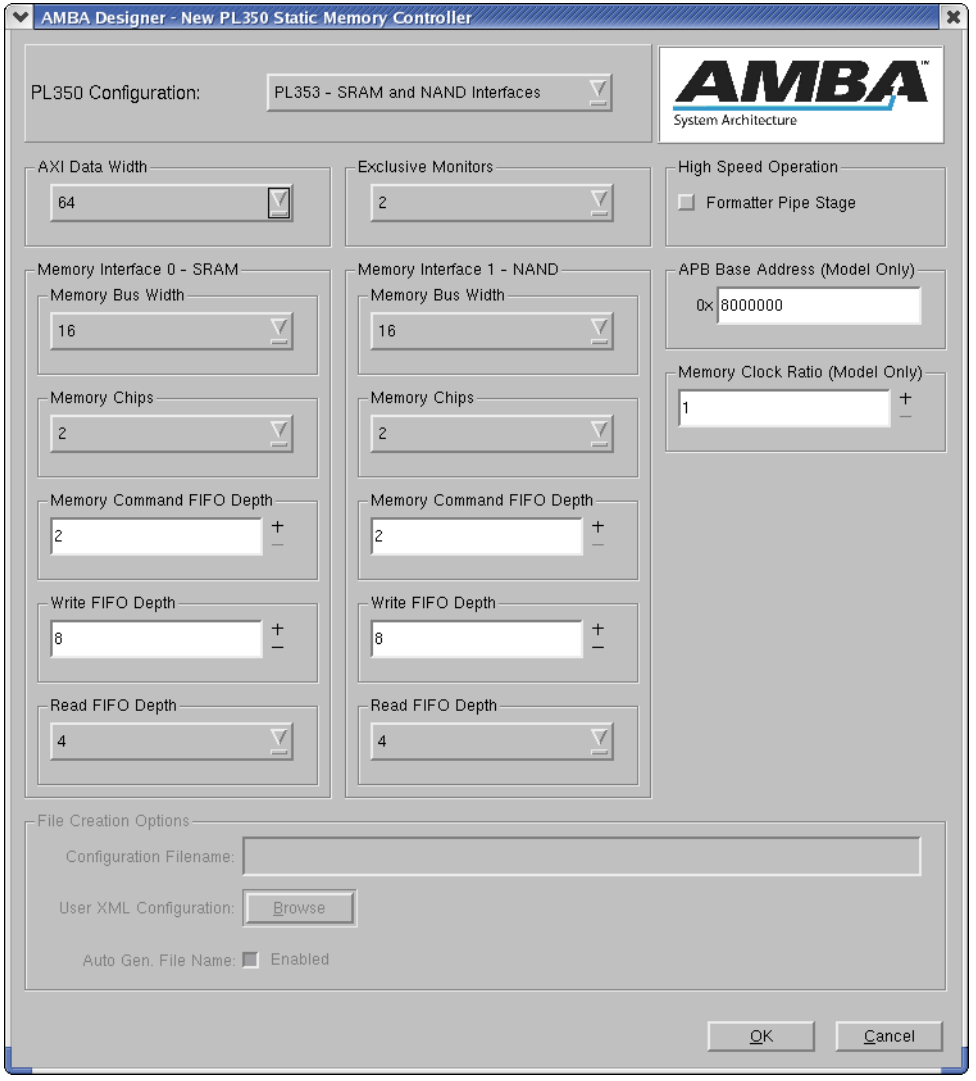


Figure 6-10 PL350 reconfiguration dialog

Note

In Figure 6-10:

- The configuration options are identical to those described for Figure 6-4 on page 6-5.

- The File Creation Options pane is grayed-out because this feature is not available during reconfiguration.

To modify the SMC:

5. Enter the modifications you require using the fields and drop-down lists that the dialog box provides.
6. Click on **OK** to create a new SMC. The program appends a `_reconfigured` suffix to the original filename and by default it stores the file in the `Designs` directory.

6.5 Configuration options for the SMC

Table 6-1 lists the configuration options in AMBA Designer that configure the configuration options of the SMC.

The configuration options are accessed by selecting the pl35*x*_smc[0] component and using the **Object Parameters...** context menu.

Table 6-1 Configuration options for the SMC

Configuration option	Range	Default value	Description
`AXI Data Width`	32, 64	64	Size in bits of the AXI data bus.
`Command FIFO depth`	Integer, 2-6	2	Number of buffers available for the command data.
`Extra Pipeline Stage`	`true` `false`	`false`	Provides an entry block pipeline stage in the format block, but at the expense of extra latency. This option is only available for SMC variants with two memory interfaces.
`Exclusive Monitors`	0, 1, 2, 4	2	Number of monitors available for monitoring exclusive accesses on the AXI port.
`Memory Chips 0`, `Memory Chips 1`	Integer, 1-4	2	Total number of static memory devices that connect to the appropriate memory interface on the SMC.
`Memory Width 0`, `Memory Width 1`	8, 16, 32[a]	16	Size in bits of the data bus width for the appropriate memory interface on the SMC.
`Read FIFO depth`	Integer, 4-16	8	Number of buffers available for the read data.
`Write FIFO depth`	Integer, 4-16	8	Number of buffers available for the write data.
The following parameters are only used when modeling the component in SoC Designer Simulator			
`APB Base Address`	32-bit hexadecimal, without `0x` prefix	`08000000`	The base address of the APB interface.
`Memory Clock Ratio`	Integer, 1-255	1	The number used to divide down the AXI clock frequency. Each memory interface uses this generated clock frequency.

a. A value of 32 is not available when configuring a NAND memory interface.

Chapter 7
Creating a System Model

This chapter describes how to create a system model using the example 4x3 interconnect that *Example 4x3 interconnect* on page 4-4 describes. The interconnect model is incorporated into a system and the system is then simulated using SoC Designer Simulator. It contains the following sections:

- *Building the system* on page 7-2
- *Simulating the system* on page 7-6.

Note

The AMBA Designer graphical user interface is based on SoC Designer. See the *SoC Designer User Guide* for general information about the menu system, window layout, and model simulation implementation.

7.1 Building the system

After creating the interconnect model, you can now add the ARM1176JZF processor, DMC, and AXI-compliant memory models to complete the system. The following sections describe this process:

- *Adding the system components*
- *Saving the system* on page 7-5.

Note

To create this system model then your device must have access to RealView® models for the following components:

- ARM1176JZF processor
- PL340 DMC
- AXI-compliant memory.

See *Installing additional component models* on page 2-8 for information about installing component models.

7.1.1 Adding the system components

To add the system components:

1. Click on the **Open** button on the Main Toolbar. This opens a file browser dialog box.
2. Load the `1176_PL340_world.mxp` file as Figure 7-1 shows.

   Figure 7-2 on page 7-3 shows the location of the `1176_PL340_world.mxp` file, when AMBA Designer is installed in the default directory.

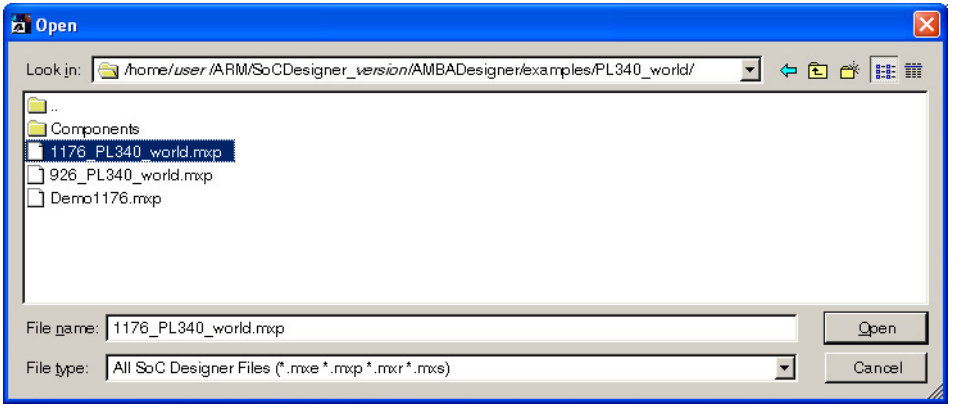


Figure 7-1 Loading the ARM1176 and PL340 world

UNIX

```
Top-level directory/
  └─home/
      └─user/
          └─ARM/
              └─SoCDesigner_version/
                  └─AMBADesigner/
                      └─examples/
                          └─PL340_world/
```

Windows

```
Top-level directory/
  └─Program Files/
      └─ARM/
          └─SoCDesigner_version/
              └─AMBADesigner/
                  └─examples/
                      └─PL340_world/
```

Figure 7-2 Location of the 1176_PL340_world.mxp file

You now have a new Diagram Window, with the tab name of **1176_PL340_world**, containing the three system components that Figure 7-3 shows.

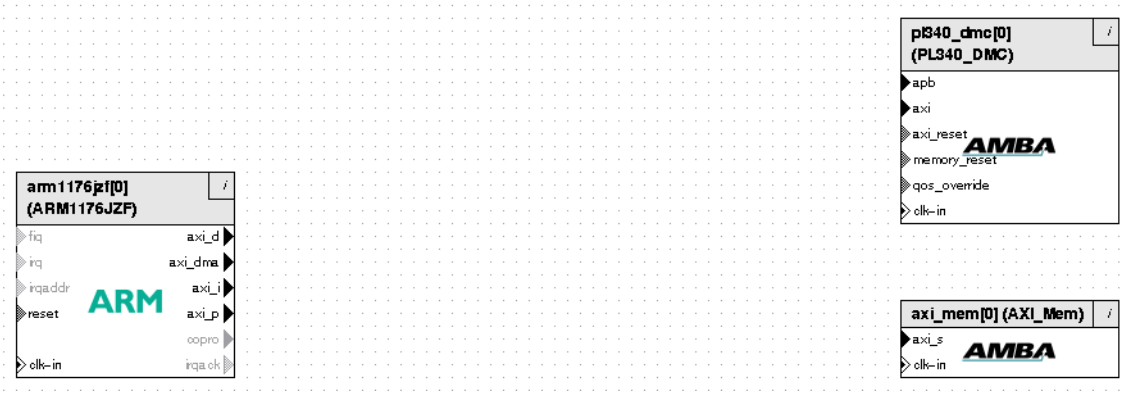


Figure 7-3 ARM1176, DMC and AXI memory

In *Adding the interconnect to the component library* on page 4-26, the 4x3 interconnect was added to the component library. To use this interconnect model:

3. Press and hold the left mouse button on the interconnect component PL301_*xxx*_4x3_1 in the Component Window and drag it on to the Diagram Window. Release the left mouse button, so that it places the interconnect between the ARM1176 processor and DMC, as Figure 7-4 on page 7-4 shows.

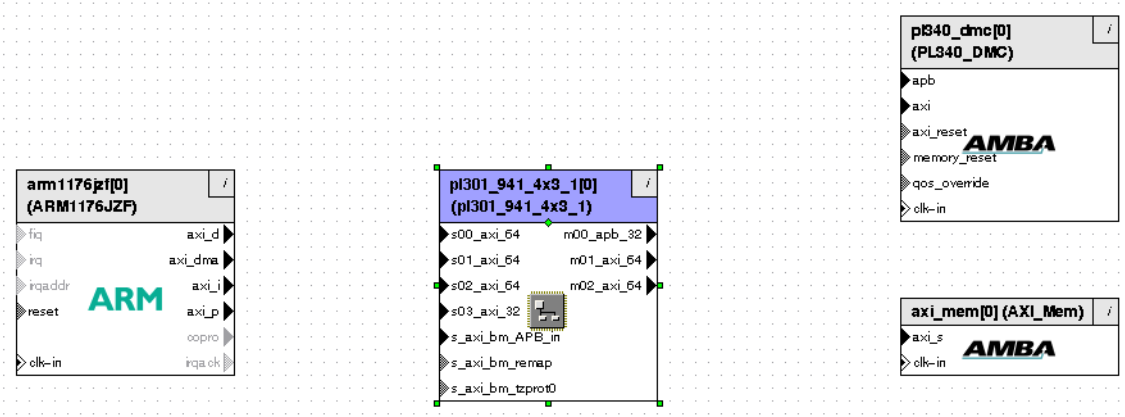


Figure 7-4 ARM1176JZF, interconnect, DMC and AXI memory

Note

If your operating platform does not support drag-and-drop then:

a. Select **Insert** → **Add Component...** from the Main menu.

b. Click on the PL301_*xxx*_4x3_1 model in the Select Component dialog box.

c. Click on **OK** to close the Select Component dialog box.

d. Click on the Diagram Window, so that it places the interconnect between the ARM1176 processor and DMC, as Figure 7-4 shows.

To connect the components to each other:

4. Click on the **Connect** button on the Main Toolbar.

5. Click on the ports of each component to connect them as follows:
   a. axi_d → s00_axi_64.
   b. axi_dma → s01_axi_64.
   c. axi_i → s02_axi_64.
   d. axi_p → s03_axi_32.
   e. m02_axi_64 → axi_s.
   f. m01_axi_64 → axi.
   g. m00_apb_32 → apb.

Note

To connect the ports, ensure the **Connect** button is selected, click on the start port and then click on the end port. For example, to connect the axi_d port to the s00_axi_64 port, first click on axi_d and then click on s00_axi_64.

If you accidentally connect the wrong ports together then immediately click on the **Undo** button to remove the connection.

Figure 7-5 shows the correct port connections for this system.

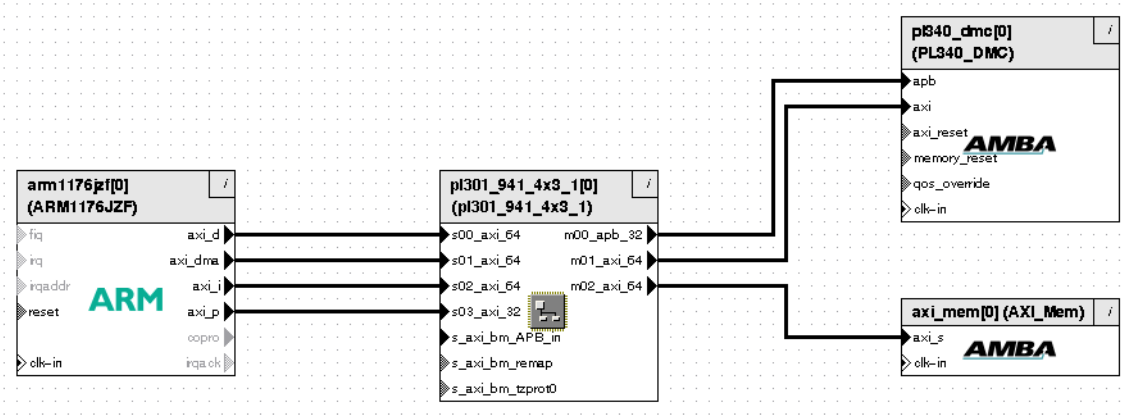


Figure 7-5 Connected system

7.1.2 Saving the system

To save the completed system that Figure 7-5 shows:

1. Select **File** → **Save As...** from the Main menu to open the Save As... dialog box.
2. Enter `Demo1176` in the Project Name field of the Save As... dialog box, as Figure 7-6 shows.

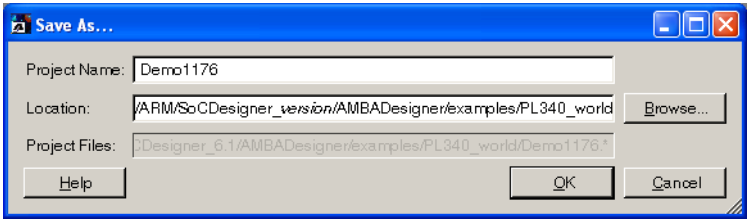


Figure 7-6 Saving the system

3. Click on **OK** to close the Save As... dialog box.

Note

The program appends the `.mxp` suffix to the filename you enter in the Project Name field.

7.2 Simulating the system

After creating the system model it is advisable to simulate the system to verify that the performance is satisfactory. The following sections describe this process:

- *Simulation environment*
- *Running the simulation* on page 7-8
- *Profiling and monitoring* on page 7-10.

7.2.1 Simulation environment

To simulate the system, you must invoke the SoC Designer Simulator application. Do this from AMBA Designer, either by pressing **F5** on your keyboard or selecting **Simulation → Simulate System...** from the Main menu, as Figure 7-7 shows.

Note

AMBA Designer must have a project file open, for example, a file with the `.mxp` extension, to enable SoC Designer Simulator to run.

See the *SoC Designer User Guide* for more information about using SoC Designer Simulator.

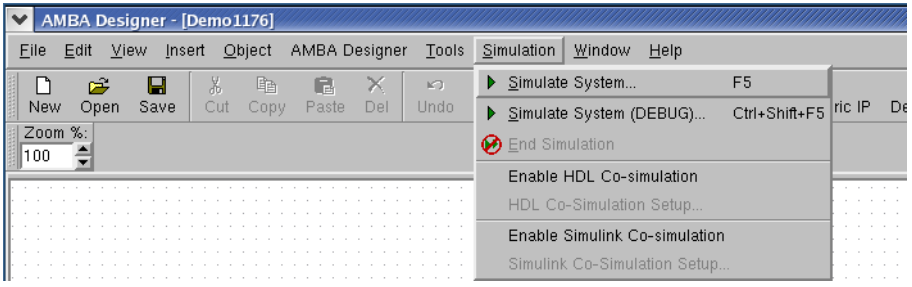


Figure 7-7 Simulate system

To set up the simulation environment:

1. Ensure that the `Demo1176.mxp` file from *Saving the system* on page 7-5 is open in AMBA Designer and then press **F5** on your keyboard.

After SoC Designer Simulator opens, it prompts you for the name of an executable file by opening the Select Application Files dialog box.

In the Select Application Files dialog box:

2. Select the arm1176jzf[0] device that is listed under the Component column.

3. Click on **Select File** as Figure 7-8 shows. This opens a file browser dialog box named Select The Application Code.

Note

This file is known as The Application Code and in this example, is the software code that runs on the ARM1176ZJF processor during a simulation.

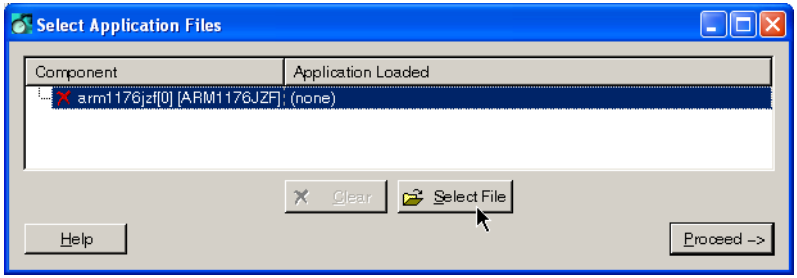


Figure 7-8 Select application files

In the Select The Application Code dialog box:

4. Navigate to the `PL340_world` directory.
5. Select the `1176_PL340_World.axf` file, as Figure 7-9 on page 7-8 shows.
6. Click on **Open**.

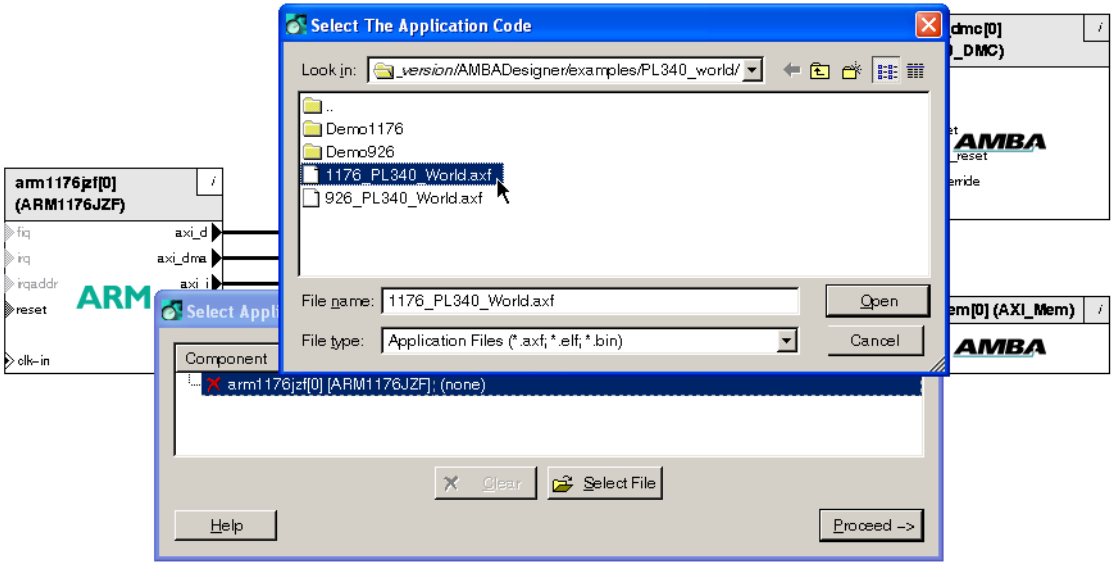


Figure 7-9 Selecting the application code

After the Select Application Code dialog box closes:

7. Click on **Proceed ->** in the Select Application Files dialog box.

Note

If the Output Window displays:

```
RemSIM: cannot find nor start RVBROKER - will be unavailable.
```

You can ignore this message, because this simulation does not require RVBROKER.

SoC Designer Simulator is now ready to simulate the system using the software code that the `1176_PL340_World.axf` file contains.

Note

The `1176_PL340_World.axf` application file is compiled code, specifically created for this 4x3 example using ARM RealView.

7.2.2 Running the simulation

Prior to running the simulation, it is advisable to disable the display of the unconnected port warning messages, that would otherwise occur when the simulation results are displayed.

To disable the Output Window display:

1. Select **Window from the Main menu and ensure that the Output Window** menu item is not selected, as Figure 7-10 shows.

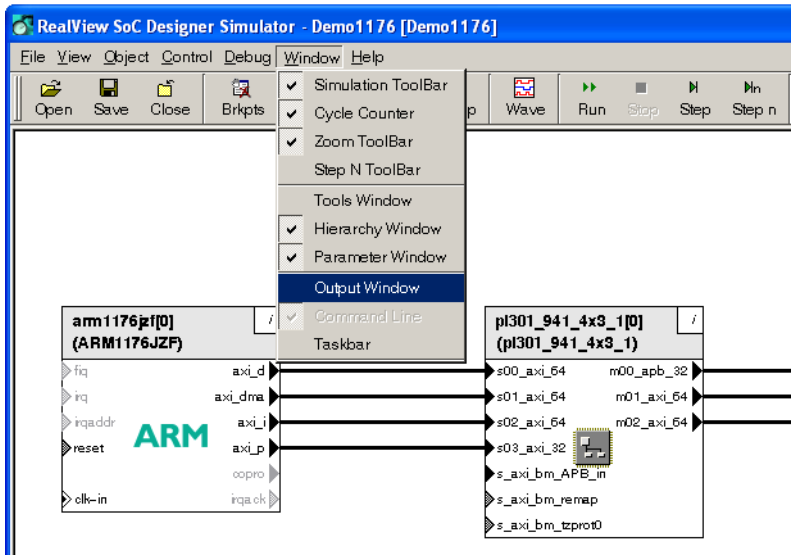


Figure 7-10 Disabling the Output Window

To run the simulation:

1. Press **F5** on your keyboard or click on **Run** located on the Simulation Toolbar.

The simulation starts, with the ARM1176JZF processor executing the application code that the `1176_PL340_World.axf` file contains.

A Console Window appears with the results of the simulation, as Figure 7-11 shows.

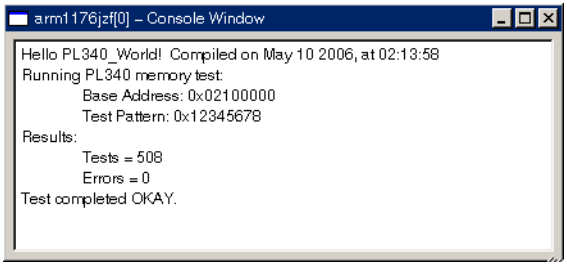


Figure 7-11 Simulation results

This Console Window shows that the ARM1176JZF processor has successfully initialized the system and output the text “Hello PL340_World!”.

Figure 7-12 shows the Cycle Counter window, adjacent to the Simulation Toolbar. For this example, the Cycle Counter value indicates the number of ARM1176JZF processor clock cycles that are executed during a simulation.

Figure 7-12 Cycle Counter

7.2.3 Profiling and monitoring

To assist the process of systems architectural design, it is beneficial to enable the profiling and monitoring features in SoC Designer Simulator, to display system transactions and refine the system architecture. The following sections describe these features:

- *Profiling*
- *Monitoring* on page 7-12.

Profiling

In this system, the ARM1176JZF processor and DMC models support profiling. The following sections describe the process for selecting memory command profiling on the DMC:

- *Selecting the profiling parameters*
- *Running the simulation* on page 7-11.

Selecting the profiling parameters

To select the DMC profiling parameters:

1. Select **Debug → Profiling Manager...** from the Main menu, or click on **Profile** on the Simulation Toolbar.
2. In the Profiling Manager dialog box, double-click on the third row containing a Component name of pl340_dmc[0] and a Stream name of Memory command, as Figure 7-13 on page 7-11 shows. This opens a Profiling window for the DMC.

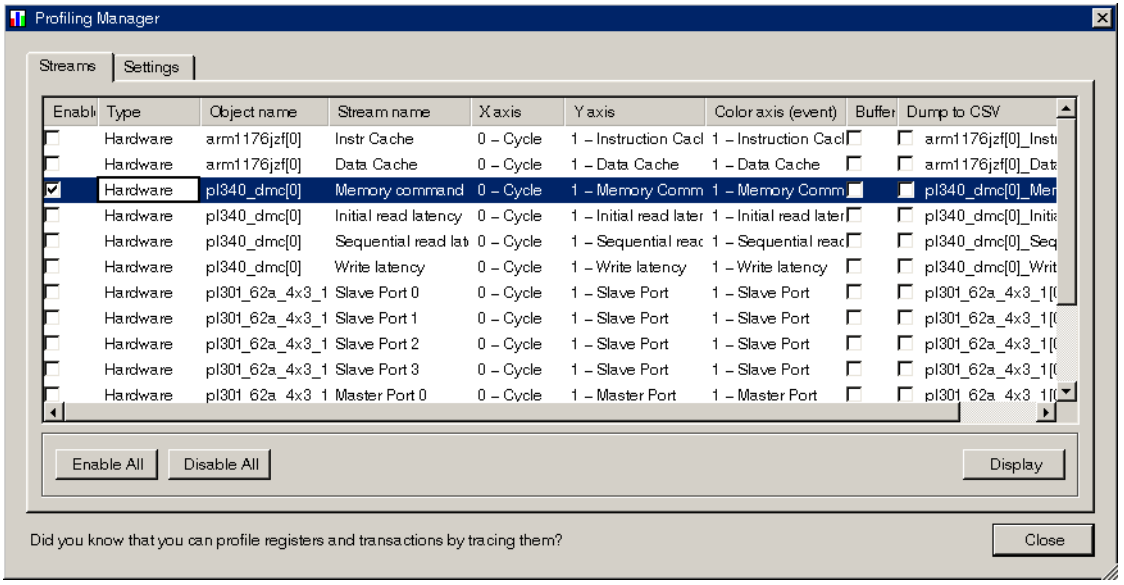


Figure 7-13 Profiling Manager dialog

3. Click on **Close** in the Profiling Manager dialog box.

Running the simulation

To run the simulation:

1. Click on **Reset** on the Simulation Toolbar. This clears the Console Window data and resets the ARM1176JZF processor's program counter.

2. Start the simulation by either:
   - press **F5** key on the keyboard
   - click on **Run** located on the Simulation Toolbar.

Figure 7-14 on page 7-12 shows an example of the profiling data that you can display in the Profiling window, when the Segment size is set to 1000 and the horizontal and vertical axes have been appropriately scaled.

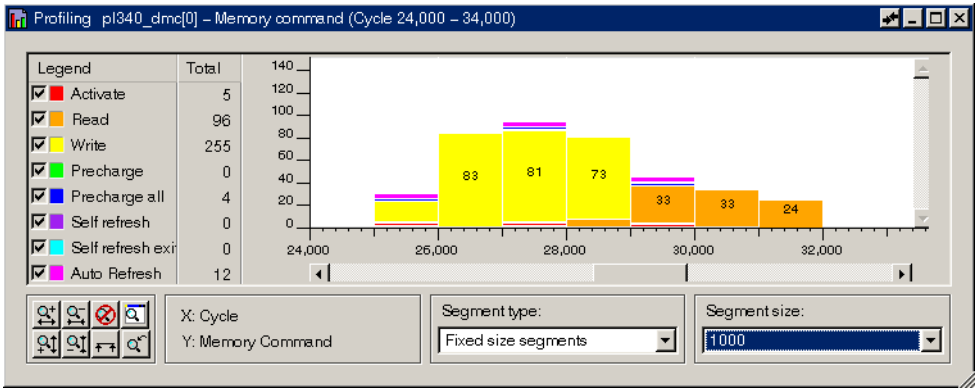


Figure 7-14 DMC profiling results

Note

See the *SoC Designer User Guide* for information about setting profiling options during simulations.

Monitoring

SoC Designer provides the ability to monitor the AMBA bus transactions. See the *SoC Designer User Guide* for information about enabling bus-master port monitoring.

Chapter 8
RTL Design Flow

This chapter describes the RTL generation process. It contains the following sections:

8.1 Registering the PrimeCell

You must register the PrimeCell component with AMBA Designer, to enable the program to generate, simulate, or synthesize the RTL.

The PrimeCell installation package provides a UNIX script that enables you to register the PrimeCell component with the program. See the Release Note of the PrimeCell, for information about how to use the script.

8.2 RTL design flow

The RTL Design Flow Manager dialog box controls the RTL design flow process. The design flow consists of the following subprocesses:

- Generate RTL
- Simulate
- Synthesize.

Figure 8-1 shows the RTL Design Flow Manager process.

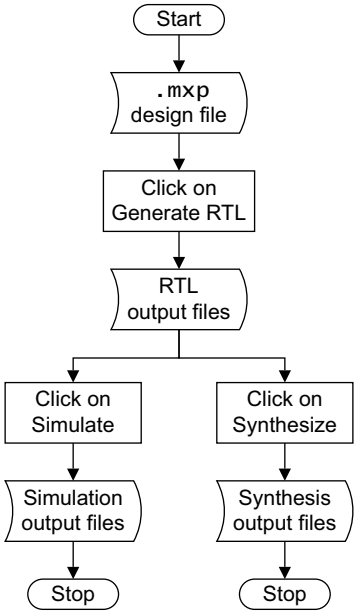


Figure 8-1 RTL Design Flow Manager process

Note

The RTL Design Flow Manager always executes the Generate RTL process before it permits the Simulate or Synthesis processes to start.

The following sections describe the process of generating, simulating, and synthesizing the RTL:

- *Generate* on page 8-4
- *Simulation* on page 8-7
- *Synthesis* on page 8-11.

8.2.1 Generate

This section describes how to generate the RTL for a PrimeCell component by using the 4x3 example interconnect that was created in *Generating the interconnect* on page 4-23.

Note

You must have the following programs installed:

- Perl
- TCL.

The AMBA Designer Release Note provides information about the program versions that are required.

To generate the RTL:

1. Click on **Open** on the Main Toolbar to open a file browser dialog box.
2. Load the interconnect that was previously created in *Generating the interconnect* on page 4-23. This file is named `PL301_xxx_4x3_1.mxp` and is located in the default location:

   UNIX `home/user/.ARM/AMBA_Designer/Designs/`

   Note

   You must enter `home/user/.ARM` in the file browser dialog box because the dialog does not display the `.ARM` directory.

   Windows The program does not permit the generation of RTL on a Windows operating system. To generate RTL, you must use the UNIX version of the program.

   Where *user* is the login name of the current person running AMBA Designer. Figure 8-2 on page 8-5 shows the interconnect that the Diagram Window displays.

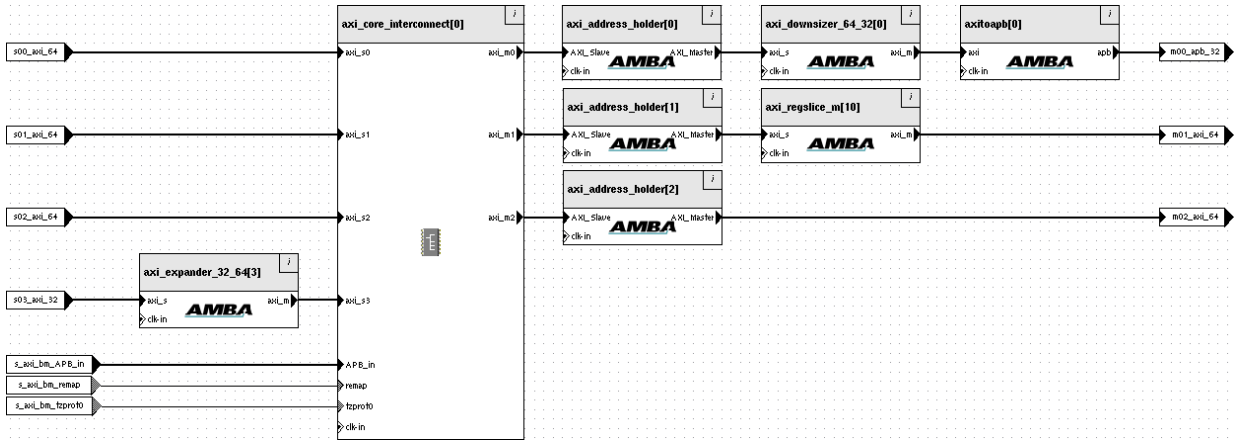


Figure 8-2 Interconnect

3. Right-click on the axi_core_interconnect[0] component to display the context menu.

4. Select **AMBA Designer → PL301 RTL Design Flow Manager...** from the context menu, as Figure 8-3 shows. This opens the RTL Design Flow Manager dialog box.

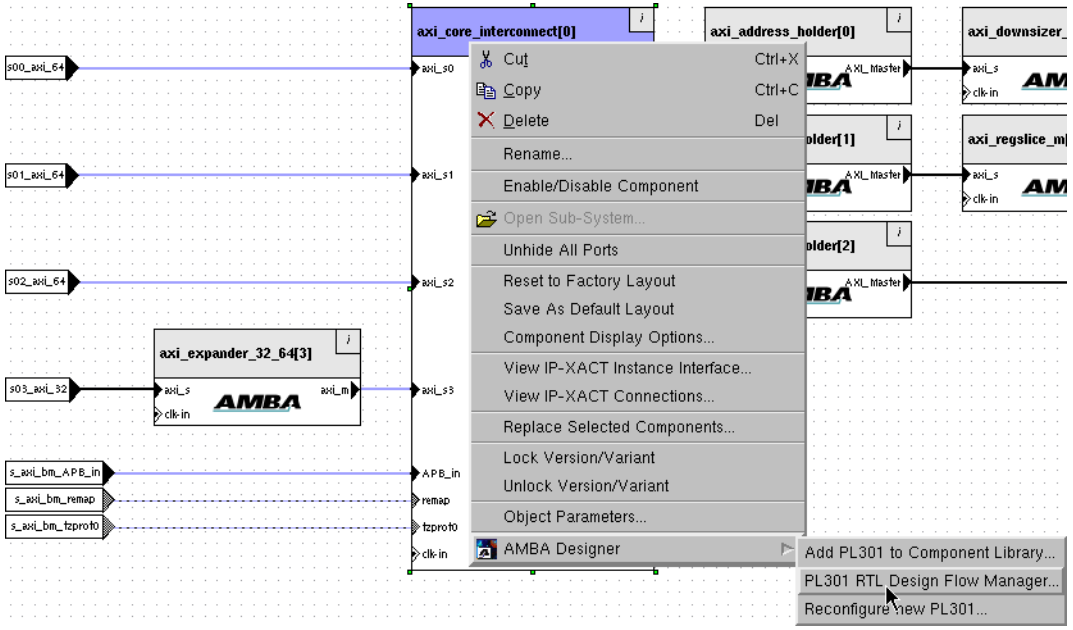


Figure 8-3 RTL Design Flow Manager

The program creates an XML configuration file. Figure 8-4 shows the location of the XML configuration file on a standard installation.

```
                                    UNIX
Top-level directory/
  └─home/
      └─user/
          └─.ARM/
              └─AMBA_Designer/
                  └─Designs/
                      └─PL301_xxx_4x3_1_RTL/
```

Figure 8-4 Location of XML file

Note

The program also uses the XML configuration file during batch mode operation. See *Batch mode operation* on page 3-5 for more information.

5. Click on **Generate RTL** in the RTL Design Flow Manager dialog box as Figure 8-5 shows.

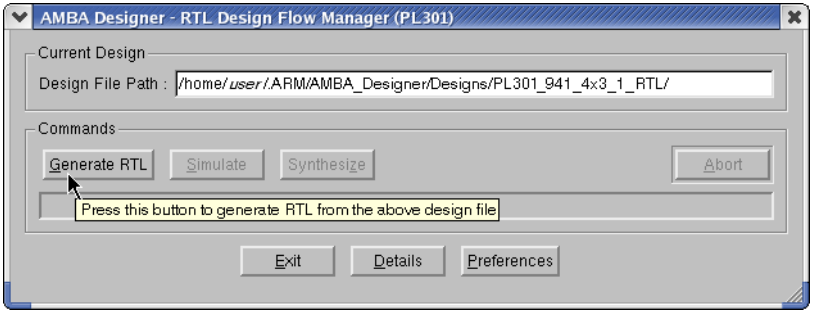


Figure 8-5 Generate RTL

The program reads in the XML configuration file and the RTL generation process starts. The progress bar in the Commands pane of the RTL Design Flow Manager dialog box displays the progress. On completion of the RTL generation process, the progress bar changes color to either:

Red Indicates that the program failed to generate the RTL. Click on the **Details** button to open the Output Window pane. This pane displays information about the status of the process.

Green Indicates that the program generated the RTL.

The configured Verilog and synthesis are generated locally and stored in the `logical` and `implementation` directories. Figure 8-6 shows the location of these directories.

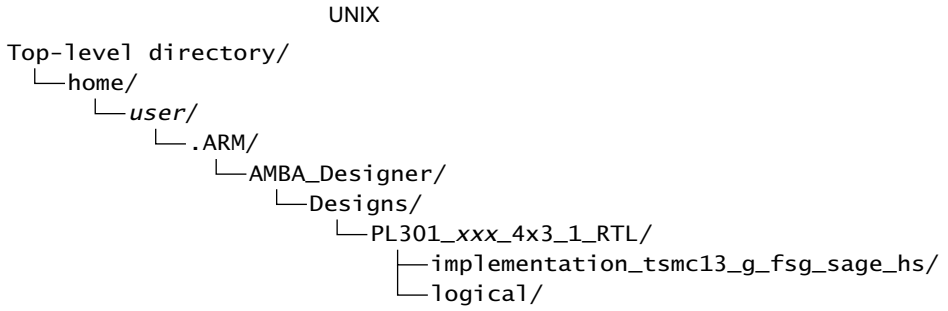


Figure 8-6 Location of Verilog and synthesis directories

Note

Figure 8-6 shows the directories for the PL301. For other PrimeCell devices, the directory structure is described in the relevant configuration chapter. For example, *Location of the Verilog and synthesis directories* on page 5-11 provides the information for the DMC.

The RTL generation process creates a *File Reader Master* (FRM) testbench based on the configured HPM. During the simulation process, the RTL Design Flow Manager uses the FRM testbench to verify the functionality of the configured RTL.

8.2.2 Simulation

You can only start the Simulate process when the Generate RTL process has been successfully completed.

Prior to running the simulation, you must verify that the simulation preferences are appropriately configured. The simulation stage is described in:

- *Simulation preferences*
- *OVL assertions* on page 8-8
- *Simulate* on page 8-10.

Simulation preferences

During the simulation, AMBA Designer invokes one of the following simulators:

- ModelSim from Model Tech, Inc.
- VCS from Synopsys, Inc.

- NC-Verilog from Cadence Design Systems, Inc.

Note

The AMBA Designer Release Note provides information about the simulator tool versions that were used during the development of AMBA Designer.

You must set the AMBA Designer simulation preferences so that AMBA Designer invokes a simulator that your device is configured for.

To configure the simulation preferences:

1. Click on **Preferences** in the RTL Design Flow Manager dialog box. This opens the RTL Design Flow Manager Preferences dialog box, as Figure 8-7 shows.

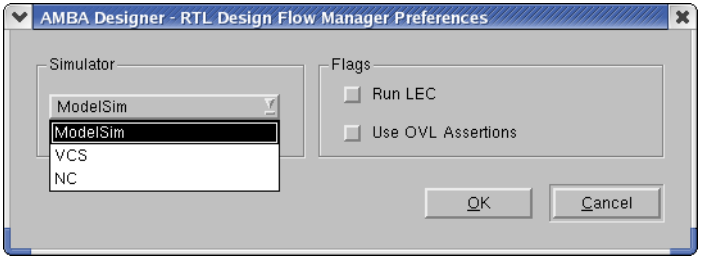


Figure 8-7 RTL Design Flow Manager Preferences

2. Using the Simulator drop-down list, select the simulator that corresponds to the simulation tool that your device is configured to use.
3. Click on **OK** to save the preferences and close the dialog box.

OVL assertions

If the PrimeCell device supports OVL assertions then if required, you can simulate with *Open Verification Library* (OVL) assertions. The procedure for using OVL assertions is described in:

- *Installing the OVL library* on page 8-9
- *Enabling OVL assertions* on page 8-9.

Note

It is not a requirement to install or enable OVL assertions, for the purpose of simulating the 4x3 example that *Example 4x3 interconnect* on page 4-4 creates.

Installing the OVL library

The AMBA Designer installation package does not include the OVL library. However, you can download the OVL assertion library from `http://www.eda.org`

You must install the OVL assertion library `*.vlib` and `*.h` components in the directory locations that Figure 8-8 shows.

Note

You must create the `OVL` and `verilog` directories.

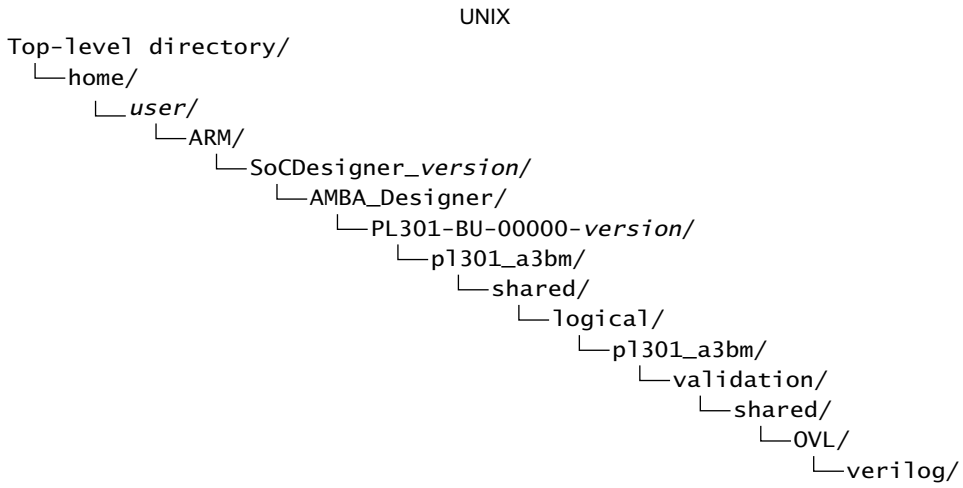


Figure 8-8 Location of the OVL and verilog directories

Note

Figure 8-8 shows the directory structure for the PL301. For other PrimeCell devices, see the relevant configuration chapter, for information about whether the device supports OVL assertions.

Enabling OVL assertions

To enable OVL assertions to be performed during simulation:

1. Click on **Preferences** in the RTL Design Flow Manager dialog box.
2. Select the **Use OVL Assertions** check box in the Flags panel, to enable simulation with OVL assertions, as Figure 8-9 on page 8-10 shows.

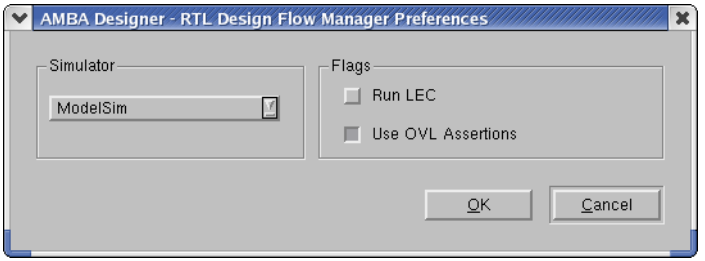


Figure 8-9 Enabling OVL assertions

3. Click on **OK** to save the preferences and close the dialog box.

Simulate

To simulate the RTL:

1. Click on **Simulate** in the RTL Design Flow Manager dialog box, as Figure 8-10 shows.

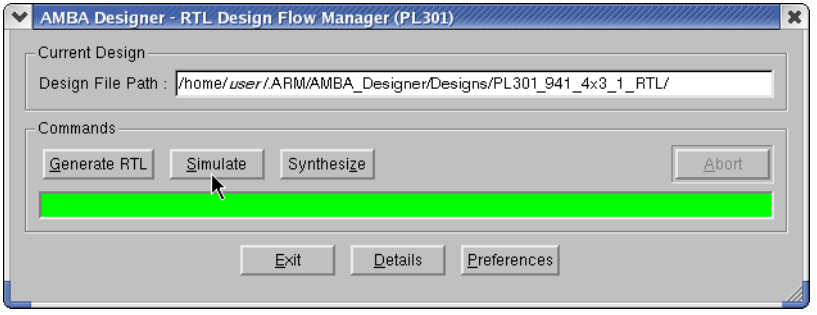


Figure 8-10 Simulate RTL

   The simulate RTL process starts. The progress bar in the Commands pane of the RTL Design Flow Manager dialog box displays the progress. On completion of the simulate RTL process, the progress bar changes color to either:

   Red Indicates that the program failed to simulate the RTL. Click on the **Details** button to open the Output Window pane. This pane displays information about the status of the process.

   Green Indicates that the program simulated the RTL.

2. Click on **Exit** to close the RTL Design Flow Manager dialog box.

8.2.3 Synthesis

You can only start the Synthesize process when the Generate RTL process has been successfully completed.

Note

You must configure your device with the appropriate location for your chosen EDA synthesis tool.

To synthesize the example created in *Generating the interconnect* on page 4-23 you must install the appropriate Artisan libraries, as *Installing synthesis libraries* on page 4-54 describes.

The RTL synthesis process in AMBA Designer was developed using the following tools:

- Synopsys Formality
- Synopsys Design Compiler
- Synopsys Physical Compiler.
- Atrenta Spyglass
- TransEDA VNavigator
- Cadence Specman Elite.

The AMBA Designer Release Note provides information about the tool versions that were used.

To synthesize the RTL:

1. Click on **Preferences** in the RTL Design Flow Manager dialog box to open the RTL Design Flow Manager Preferences dialog box.
2. If you require *Logical Equivalence Checking* (LEC) to be performed during synthesis then you must ensure that the **Run LEC** check box is selected in the Flags panel, as Figure 8-9 on page 8-10 shows.

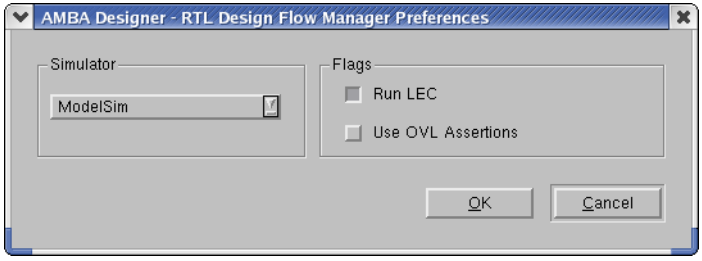


Figure 8-11 Enabling LEC

Note

You must configure your device with the appropriate location for your chosen LEC tool.

3. Click on **OK** to save the preferences and close the RTL Design Flow Manager Preferences dialog box.

4. Click on **Synthesize** in the RTL Design Flow Manager dialog box, as Figure 8-12 shows.

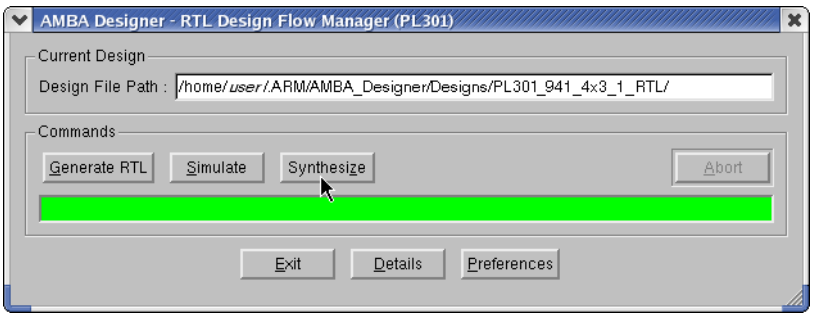


Figure 8-12 Synthesize RTL

The synthesize RTL process starts. The progress bar in the Commands pane of the RTL Design Flow Manager dialog box displays the progress. On completion of the synthesize RTL process, the progress bar changes color to either:

Red Indicates that the program failed to synthesize the RTL. Click on the **Details** button to open the Output Window pane. This pane displays information about the status of the process.

Green Indicates that the program synthesized the RTL.

5. Click on **Exit** to close the RTL Design Flow Manager dialog box.

8.2.4 Summary file

The RTL Design Flow Manager process updates the `amba_engine_summary.rpt` summary file, with pass or fail information at the end of each of the three stages.

Figure 8-13 on page 8-13 shows the location of the summary report file.

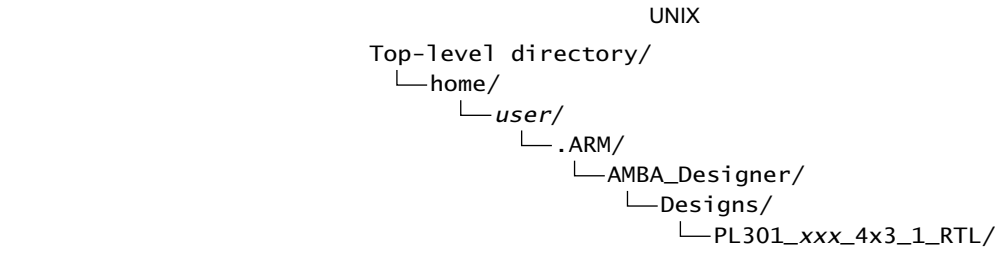


Figure 8-13 Location of summary file

When invoked, the RTL Design Flow Manager appends new information, if the summary file exists.

The tabulated text file contains information about the following types of message categories:

fatal A non-recoverable error. The RTL Design Flow Manager process aborts.

error A functional error is detected that prevents the RTL Design Flow Manager from proceeding to the next stage of the process.

warning A functional warning is detected. You must verify that it is safe to ignore this violation because it might indicate the presence of a functional error.

accepted warning

An error or warning occurred but on further processing of the input files the violation is found to be invalid. You can ignore these violations.

To check if an error or warning is acceptable, AMBA Designer uses the information contained in the `acceptable_messages` directory. Figure 8-14 on page 8-14 shows the location of this directory.

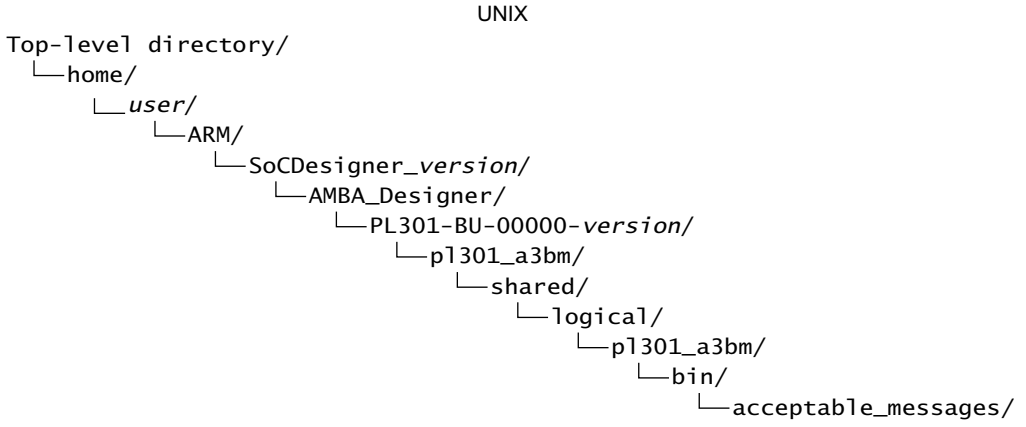


Figure 8-14 Location of the acceptable messages directory

OVL_ERROR

An OVL assertion error is detected. This prevents the RTL Design Flow Manager from proceeding to the synthesis process.

rule AMBA Designer does not currently support use of this feature.

guideline AMBA Designer does not currently support use of this feature.

Glossary

This glossary describes some of the terms used in technical documents from ARM.

Advanced eXtensible Interface (AXI)
A bus protocol that supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure. The AXI protocol also includes optional extensions to cover signaling for low-power operation.

AXI is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.

Advanced High-performance Bus (AHB)
A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol. The full AMBA AHB protocol specification includes a number of features that are not commonly required for master and slave IP developments and ARM recommends only a subset of the protocol is usually used. This subset is defined as the AMBA AHB-Lite protocol.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

Advanced Microcontroller Bus Architecture (AMBA)

A family of protocol specifications that describe a strategy for the interconnect. AMBA is the ARM open standard for on-chip buses. It is an on-chip bus specification that details a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.

Advanced Peripheral Bus (APB)

A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

AHB *See* Advanced High-performance Bus.

AHB-Lite A subset of the full AMBA AHB protocol specification. It provides all of the basic functions required by the majority of AMBA AHB slave and master designs, particularly when used with a multi-layer AMBA interconnect. In most cases, the extra facilities provided by a full AMBA AHB interface are implemented more efficiently by using an AMBA AXI protocol interface.

Aligned A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.

AMBA *See* Advanced Microcontroller Bus Architecture.

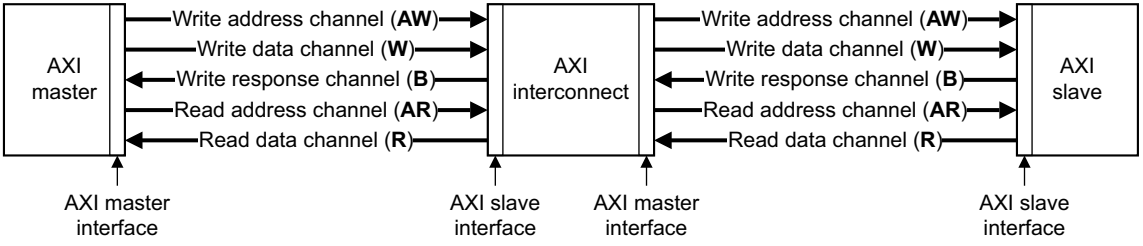
APB *See* Advanced Peripheral Bus.

AXI *See* Advanced eXtensible Interface.

AXI channel order and interfaces

The block diagram shows:

- the order in which AXI channel signals are described
- the master and slave interface conventions for AXI components.



AXI terminology

The following AXI terms are general. They apply to both masters and slaves:

Active read transaction

A transaction for which the read address has transferred, but the last read data has not yet transferred.

Active transfer

A transfer for which the **xVALID**[1] handshake has asserted, but for which **xREADY** has not yet asserted.

Active write transaction

A transaction for which the write address or leading write data has transferred, but the write response has not yet transferred.

Completed transfer

A transfer for which the **xVALID/xREADY** handshake is complete.

Payload The non-handshake signals in a transfer.

Transaction An entire burst of transfers, comprising an address, one or more data transfers and a response transfer (writes only).

Transmit An initiator driving the payload and asserting the relevant **xVALID** signal.

Transfer A single exchange of information. That is, with one **xVALID/xREADY** handshake.

The following AXI terms are master interface attributes. To obtain optimum performance, they must be specified for all components with an AXI master interface:

Combined issuing capability

The maximum number of active transactions that a master interface can generate. This is specified instead of write or read issuing capability for master interfaces that use a combined storage for active write and read transactions.

---

1. The letter **x** in the signal name denotes an AXI channel as follows:

AW Write address channel.
W Write data channel.
B Write response channel.
AR Read address channel.
R Read data channel.

Read ID capability

The maximum number of different **ARID** values that a master interface can generate for all active read transactions at any one time.

Read ID width

The number of bits in the **ARID** bus.

Read issuing capability

The maximum number of active read transactions that a master interface can generate.

Write ID capability

The maximum number of different **AWID** values that a master interface can generate for all active write transactions at any one time.

Write ID width

The number of bits in the **AWID** and **WID** buses.

Write interleave capability

The number of active write transactions for which the master interface is capable of transmitting data. This is counted from the earliest transaction.

Write issuing capability

The maximum number of active write transactions that a master interface can generate.

The following AXI terms are slave interface attributes. To obtain optimum performance, they must be specified for all components with an AXI slave interface

Combined acceptance capability

The maximum number of active transactions that a slave interface can accept. This is specified instead of write or read acceptance capability for slave interfaces that use a combined storage for active write and read transactions.

Read acceptance capability

The maximum number of active read transactions that a slave interface can accept.

Read data reordering depth

The number of active read transactions for which a slave interface can transmit data. This is counted from the earliest transaction.

Write acceptance capability

The maximum number of active write transactions that a slave interface can accept.

Write interleave depth

The number of active write transactions for which the slave interface can receive data. This is counted from the earliest transaction.

Beat Alternative word for an individual transfer within a burst. For example, an INCR4 burst comprises four beats.

See also Burst.

Burst A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals to indicate the length of the burst and how the addresses are incremented.

See also Beat.

Byte An 8-bit data item.

Byte lane strobe A signal that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of this signal corresponds to eight bits of the data bus.

Direct Memory Access (DMA)
An operation that accesses main memory directly, without the processor performing any accesses to the data concerned.

DMA *See* Direct Memory Access.

Halfword A 16-bit data item.

Host A computer that provides data and other services to another computer. Especially, a computer providing debugging services to a target being debugged.

Region A partition of instruction or data memory space.

Remapping Changing the address of physical memory or devices after the application has started executing. This is typically done to permit RAM to replace ROM when the initialization has been completed.

Unaligned A data item stored at an address that is not divisible by the number of bytes that defines the data size is said to be unaligned. For example, a word stored at an address that is not divisible by four.

Word A 32-bit data item.