Using DesignWare Library IP in coreAssembler

coreAssembler Tutorial

A coreTools Application
Copyright Notice and Proprietary Information Notice

Copyright © 2014 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader’s responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/Company/Pages/Trademarks.aspx.

All other product or company names may be trademarks of their respective owners.

Synopsys, Inc.
700 E. Middlefield Road
Mountain View, CA 94043

www.synopsys.com
Preface

About This Manual

This manual contains a tutorial and information about using DesignWare Library IP in coreAssembler. This manual is intended for designers and integrators who want to learn how to quickly assemble an AMBA® subsystem using DesignWare Library IP in coreAssembler. AMBA is a registered trademark of ARM Limited and is used under license.

This manual does not contain exhaustive information about coreAssembler. For more detailed information about coreAssembler, refer to the coreAssembler User Guide.

Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2014</td>
<td>J-2014.09</td>
<td>Version change. Removed ‘Define Interface’ from coreAssembler GUI.</td>
</tr>
<tr>
<td>September 2013</td>
<td>I-2013.09</td>
<td>Platform and version change. Removed VMT testbench creation.</td>
</tr>
<tr>
<td>March 2013</td>
<td>H-2013.03</td>
<td>Updates to screenshots and some tutorial steps.</td>
</tr>
<tr>
<td>June 2012</td>
<td>G-2012.06</td>
<td>Date / Version update. Updated graphics and Appendix A menu items.</td>
</tr>
<tr>
<td>December 2011</td>
<td>F-2011.12</td>
<td>Date / Version update. Replaced “Perform ASIC Synthesis” window graphic on page 43 (new default Design_Compiler_Reference_Methodology)</td>
</tr>
<tr>
<td>June 2011</td>
<td>F-2011.06</td>
<td>STAR fixes; Major GUI updates (merging to Specify Subsystem activity and tabbed sub-activities); Many updated figures.</td>
</tr>
<tr>
<td>September 2010</td>
<td>E-2010.09</td>
<td>STAR fixes; GUI updates (Technology dialog and report, Enable Activity); Activity changes (Perform ASIC Synthesis)</td>
</tr>
<tr>
<td>March 2010</td>
<td>D-2010.03</td>
<td>GUI updates (Complete Connections, Change Connections, etc.)</td>
</tr>
<tr>
<td>January 2010</td>
<td>C-2009.06-SP2</td>
<td>Copyright update.</td>
</tr>
<tr>
<td>October 2009</td>
<td>C-2009.06-SP1</td>
<td>Activity reports and design views are not automatically generated. New Reports tab behavior. Complete Connections window reorganized. Export interface dialog has new naming “format” options.</td>
</tr>
</tbody>
</table>
Preface

Using DesignWare Library IP in coreAssembler

Web Resources

❖ [http://www.designware.com](http://www.designware.com) - Datasheets, verification models, and DesignWare Synthesizable IP
❖ [http://www.synopsys.com](http://www.synopsys.com) - General information about Synopsys and its products

Manual Overview

This manual contains the following chapters and appendixes:

- **“Introduction and Setup” on page 9**
  - Give details of DesignWare Library IP in coreAssembler to get you using the tool.

- **“Tutorial 1: Creating a Hierarchical Subsystem” on page 13**
  - Shows how to create a multi-layer subsystem that consists of the following DesignWare synthesizable IP components: `DW_axi`, `DW_axi_x2h`, `DW_ahb`, `DW_apb`, `DW_ahb_dmac`, `DW_ahb_ictl`, `DW_apb_timers`, and `DW_apb_uart`.

- **“Tutorial 2: Creating a Testbench” on page 33**
  - Shows how to create both a VMM testbench for your subsystem design. You will also simulate a single component in a non-Synopsys simulator.

- **“Tutorial 3: Synthesizing Your Design” on page 39**
  - Shows how to create a new subsystem workspace using a batch script, prepare for the Synthesize activity, and perform a Synthesis run.

- **“Menus and Toolbars” on page 45**
  - Provides a table of toolbar icons and corresponding menu choices to accomplish subsystem assembly.

Tutorial Conventions

Each tutorial in this document is structured with the following topics:

- **Description** – briefly describes the design you are going to create
- **Objective** – highlights the specific tasks you will perform to create the tutorial design
- **Tutorial Procedures** – steps you through all tasks to complete the tutorial
- **Tutorial Summary** – summarizes the activities you completed during the tutorial

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 2009</td>
<td>B-2008.12-SP1</td>
<td>Changed Big- to Little-Endian in tutorial to fix erroneous simulation results.</td>
</tr>
<tr>
<td>June 2008</td>
<td>B-2008.06</td>
<td>Updated Screenshots, Memory Map, Slave Visibility, etc.</td>
</tr>
</tbody>
</table>
Getting Help

If you have a question about using Synopsys products, please consult product documentation that is installed on your network. You can also access documentation for DesignWare products on the Web:

- **Product documentation for many DesignWare products:**
- **Datasheets for individual DesignWare IP components, located using “Search for IP”:**
  [http://www.designware.com](http://www.designware.com)
- **License Information -- Synopsys Common Licensing Quickstart:**

You can also contact the Synopsys Support Center in the following ways:

- **Obtain Customer Education information, SolvNet access to documentation and software, or open a call to your local support center using this page:**
- **Send an e-mail message to support_center@synopsys.com.**
- **Global Support Centers -- contact information and hours of operation:**
This chapter provides an overview of DesignWare Library IP in coreAssemble, which can help you accelerate your SoC design flow. This chapter also describes setup and other general information to get you started using DesignWare Library IP in coreAssemble.

The topics discussed in this chapter include:

- Overview of Using DesignWare Library IP in coreAssemble
- “Installation and Setup” on page 9
- “Using coreAssemble Workspaces” on page 10

1.1 Overview of Using DesignWare Library IP in coreAssemble

coreAssemble is a highly flexible, integrated and feature-rich design environment that allows you to select, configure, interconnect, simulate, and synthesize DesignWare Synthesizable Components for AMBA® 2 and AMBA® 3 AXI. AMBA is a registered trademark of ARM Limited and is used under license.

Synopsys provides coreAssemble and coreConsultant, both from the Synopsys coreTools suite, for the purposes of configuration, synthesis, and verification of synthesizable IP. For information on the different coreTools applications, refer to Guide to coreTools Documentation.

A workspace is your custom-configured version of a component or subsystem, where you connect, configure, simulate, and synthesize your implementation. coreConsultant is the basic tool used to create a workspace for a single IP component. coreAssemble enables you to create a workspace with multiple IP components connected within a subsystem.

1.2 Installation and Setup

You must install and properly set up all the components and tools used with coreAssemble before you can create a subsystem using DesignWare Library IP. For installation instructions and information about required tools and versions, refer to “Setting up Your Environment” in the DesignWare Synthesizable Components for AMBA 2/AMBA 3 AXI Installation Guide. For general information about the contents of the coreTools release, refer to the coreTools Release Notes.
1.2.1 Licensing

Table 1-1 describes the required licenses for coreAssembler.

<table>
<thead>
<tr>
<th>Type of IP</th>
<th>Required License</th>
<th>Usage Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DW Library IP</td>
<td>DesignWare (or coreAssembler)</td>
<td>coreAssembler User Guide</td>
</tr>
<tr>
<td>Other coreBuilder packaged IP</td>
<td>coreAssembler (or coreBuilder*)</td>
<td>coreAssembler User Guide</td>
</tr>
<tr>
<td>Non-DW IP or IP-XACT IP</td>
<td>coreAssembler</td>
<td>coreAssembler User Guide</td>
</tr>
</tbody>
</table>

* Note: coreAssembler can use a coreBuilder license for DesignWare and coreKit IP. Synthesis does not require a DesignWare license if source code license and source for DW Building Blocks is used.

For a more details or for licensing of other tools within coreTools family, refer to “Licensing” in the coreTools Release Notes.

1.3 Using coreAssembler Workspaces

A workspace is your custom-configured version of a subsystem where you connect, configure, simulate, and synthesize your subsystem. Workspaces contain directories, files, and symbolic links to IP component installations (in $DESIGNWARE_HOME for DesignWare Library IP). Workspaces are of two types:

- Subsystem workspace – this is the workspace for your synthesizable IP design
- Testbench workspace – this is where you build a testbench around your Device Under Test (DUT) and simulate your subsystem.

Figure 1-1 shows the directories for both types of coreAssembler workspaces. Some of the directories are created after you perform the related activities (for example, the export and syn (synthesis) directories).

**Figure 1-1 coreAssembler Workspace Tree**
1.4 Overview of coreAssembler Tutorials

The tutorials in this chapter are designed to get you quickly up-to-speed on how to use coreAssembler to create your DesignWare SoC subsystem. Each tutorial shows specific tasks that you will encounter when you begin using the tool to build your own design. The tutorials are as follows:

❖ “Tutorial 1: Creating a Hierarchical Subsystem” – Provides the necessary steps to create a hierarchical subsystem design, configure the design, create a memory map, and generate the subsystem RTL.

❖ “Tutorial 2: Creating a Testbench” – Uses the Tutorial 1 design as the device under test (DUT), and steps you through the process of creating a testbench workspace, adding verification models, performing a simulation run, and saving the testbench code.

❖ “Tutorial 3: Synthesizing Your Design” – Uses the Tutorial 1 design, and guides you through the steps to set up and perform a synthesis run.

1.5 Before You Begin

⚠️ Attention

It is important that you have correctly installed and set up coreAssembler and your DesignWare Library IP environment to use DesignWare Library IP in coreAssembler. Please refer to the installation information in the coreTools Release Notes and the DesignWare Synthesizable Components for AMBA 2/AMBA 3 AXI Installation Guide.
2

Tutorial 1: Creating a Hierarchical Subsystem

2.1 Description
In this tutorial, you will create a multi-layer subsystem that consists of the following DesignWare synthesizable components for AMBA 2/AMBA 3 AXI: DW_axi, DW_axi_x2h, DW_ahb, DW_apb, DW_ahb_dmac, DW_ahb_ictl, DW_apb_timers, and DW_apb_uart.

2.2 Objective
The purpose of this tutorial is to show the coreAssembler activities used for subsystem RTL creation using the following “Create RTL” group of activities:

❖ “Procedure 1: Create a New Workspace” on page 15
❖ “Procedure 2: Specify Subsystem -- Add Components” on page 17
❖ “Procedure 3: Specify Subsystem -- Memory Map Specification” on page 25
❖ “Procedure 4: Specify Subsystem -- Configure Components” on page 26
❖ “Procedure 5: Complete Connections” on page 28
❖ “Procedure 6: Generate Subsystem RTL” on page 30

You will need to complete all of the steps in this tutorial in order to begin Tutorial 2.

2.3 Before You Begin
Because the tutorials are streamlined to help you complete them in minimum time, some of the general information about coreTools and coreAssembler is not provided in the tutorial flow. Links are provided to the appropriate section in the coreAssembler User Guide.
2.4 Tutorial 1 Subsystem Design

Figure 2-2 shows the layout of the design that you create in Tutorial 1. It contains a hierarchical APB block, and both AXI and AHB buses. A DMA controller provides direct access to external memory.

Testbenches:

- tb_cA_Tutorial_2b

Diagram:

- **DUT**
- **i_AhbMasterInterface** (custom AHB master)
- **DW_ahb**
  - **i_ahb**
- **Bridge**
  - **DW_apb**
    - **i_apb**
- **mychip_top**
  - **DW_axi**
    - **i_axi**
  - **DW_axi_x2h**
    - **i_axi_x2h**
- **i_AhbSlaveInterface** (custom AHB slave)
- **i_AxiMasterInterface** (custom AXI master)
- **DW_ahb_dmac**
  - **i_dmac**
- **DW_apb_dmac**
- **i_DMAC**
- **i_AxiSlaveInterface**
- **i_AxiMasterInterface** (custom AXI master)
- **i_AhbSlaveInterface** (custom AHB slave)

The diagram illustrates the connections and interfaces used in the tutorial. The manually exported interfaces are shown in red, while the automatically exported interfaces are shown in blue. The annotations `s` and `m` represent slave and master respectively.

Testbenches:

- **DW_apb_timers**
  - **i_timers**
- **DW_apb_uart**
  - **i_uart_1**
  - **i_uart_2**
  - **i_uart_3**
- **DW_axi_x2h**
  - **i_axi_x2h**
- **DW_axi**
  - **i_axi**
- **DW_ahb**
  - **i_ahb**
- **DW_apb_dmac**
  - **i_dmac**
- **DW_ahb_dmac**
  - **i_dmac**

These testbenches are used to verify the functionality of the design. The tutorial provides a comprehensive overview of how to use DesignWare Library IP in coreAssembler.
2.5 Procedure 1: Create a New Workspace

Complete the following steps:

1. Invoke the coreAssembler environment.
   
   ```
   % coreAssembler
   ```

2. Click on the File > New Workspace menu item to display the “Create a New Workspace” dialog:

   ![Create a New Workspace dialog](image)

3. Change the following fields and then click OK:

   **Table 2-2 Tutorial 1 Workspace Naming**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace name</td>
<td>cA_Tutorial_1</td>
</tr>
<tr>
<td>Design name</td>
<td>mychip_top</td>
</tr>
</tbody>
</table>

4. The Add Components tab should be active (if not, click on the Add Components tab).
You should see a window layout similar to the following figure. The tabs at the top of the Activity View pane allow you to easily move between adding components, making configuration changes, and updating a memory map.

Some activity tabs, such as the Memory Map tab, may not be visible until you enable the activity. This will be explained in more detail in the Memory Map tutorial steps.
2.6 Procedure 2: Specify Subsystem -- Add Components

1. From the Add Components tab view, right-click to show the schematic menu, then choose Add New Component -or- click the on the toolbar.

2. From the list of available components, click on the DW_ahb component; the instance name appears in the Component Name(s) column. You can edit the default instance name (i_ahb) to be whatever name you want before adding the component, but for this tutorial, use the default name. Click OK. It may take a few seconds to add a component.

You will see a Schematic View of a single DW_ahb component. The Schematic View graphically shows the components and connections within your subsystem.

However, before you begin building the subsystem, it is important that you check to see if you have properly set up your environment and installed the correct versions of supported tools (such as simulators, coreAsmbl, Design Compiler, and so on).
3. In the Help menu, select Help for /i_ahb > Check Environment...

Checks are performed, and a pop-up window similar to the following figure is displayed. This report lists the specific tools and versions installed in your environment. It also displays errors when a specific tool is not installed or if you are using an older tool version.

For some blocks you might not see this html loading .... dont worry

Note

If you have any warnings or errors in this report, you must first correct them. For example, if you have errors with implementation tool checks, you must fix them to complete Tutorial 3. If you have simulation tool issues, then you must fix these before performing Tutorial 2. Refer to the DesignWare Synthesizable Components for AMBA 2/AMBA 3 AXI Installation Guide, which contains information about downloading and installing required tools, such as Vera, Synopsys VCS, (optionally MTI, and NC) simulators, and Design Compiler.

After you have resolved these errors and/or warnings, make sure to run the Design IP Environment Check again. Click OK in the checkEnv.html window to close it.
4. Toggle over to the tree view (illustrated in the following figure) by clicking on the Toggle View toolbar icon. The tree view shows a text view of the component(s) and connections. Click on the “+” box to expand the view. Red text indicates that information about the component’s interface connections is missing. As you build the subsystem in this tutorial, you will resolve these issues. An interface connection is the relationship between two components. For instance, in a few moments you will be adding an APB to the system. The single interface connection between the AHB and the APB represents the AHB slave interface, which includes the address, data, and control signals between the two bus components.

5. Toggle back to the Schematic View.

Note You do not have to toggle back to the Schematic View in order to use the Add New Components dialog. Components can be added in either view.
6. Open the **Add New Components** box again and click on the following components:
   - DW_axi_x2h
   - DW_axi

7. Click **OK** to add these components to your subsystem. The connections are automatically made from the AXI Slave through the X2H component to the AHB_Master1 interface. Clock and reset connections are also made.
   (NOTE: OK the dialog that may appear, and you will handle the connections later.)

8. Open the **Add New Components** box again and add the DW_AHB_dmac.
   The DMAC connects to a Master port and a Slave port on the i_AHB component. The combined interrupt line on the DMAC is not automatically connected. Next you add an interrupt controller to make the necessary interrupt connections.

9. Open the **Add New Components** box again and add the DW_AHB_ictl.
   This component is added and shows a large number of interrupt ports available. You won’t need all of these ports, so next you need to configure ICTL for the correct number of ports.

10. Click on the **Configure Components** tab; then expand the i_ictl instance and click on the Top-level Parameters item. Change the number of “irq” and “fiq” interrupts as shown in the diagram below. Do not click Apply yet.

11. Click back to the **Add Components** view and apply the changes that you made in the Configure Components view by clicking OK on the dialog that appears. You should see that the i_ictl instance now shows fewer interrupt ports. (Note: If you don’t want to see this “Apply” dialog in the future, check the “Don’t Show Again” box.)

   You will hook up the i_ictl interrupts later, when all of the components have been added. Next you will export some interfaces as shown in Figure 2-2.

12. Select the AXI_Master01 port on i_axi (you may need to zoom in to see port names). Right click and choose “Export Interface.” Click the OK dialog button, keeping the default export name.
   The i_axi component turns from red to yellow, indicating that no interface issues remain.
13. In a similar manner select and export the CPU port on i_ictl. Use default naming. The i_ictl component is now green, indicating that there are no issues. You may need to zoom your view to see the port name text.

14. Finally, select the Remap-Pause signal on i_aeh. But instead of exporting, right-click and choose “Set Unused” from the menu. The i_aeh component is now yellow.

Next, you will build the DW_apb and peripherals in a hierarchical cell. But, before you do this, it is a good idea to save your work.

15. From the File menu, choose “Save Workspace.” Your work to this point is now saved in the cA_Tutorial_1 workspace directory.

16. Right click in the Schematic window, and choose “Add Hierarchical Level” from the menu. A dialog box appears for you to name the cell.

17. Add the “Name” and “Design Name” as shown, and OK the dialog box.

coreAssembler creates the cell and automatically puts you into the new level of hierarchy (so the previous schematic view is gone). You use the popup menu item “Exit cell” to move up and out of a hierarchical level, and double-click on the cell to descend into the cell hierarchy.

18. Add the following components in this level of hierarchy
   ✦ dw_apb ( i_apb )
   ✦ dw_apb_uart ( i_uart_1 ) (rename this instance “i_uart_1” before adding)
   ✦ dw_apb_uart ( i_uart_2 ) (rename this instance “i_uart_2” before adding)
   ✦ dw_apb_timers ( i_timers )
   HINT: Add i_apb, i_uart_1 and i_timers in the first add, then i_uart_2 in the second add.

19. Notice that i_uart_1 output is named “SIO” and i_uart_2 output is named “i_uart_2_SIO.” To keep naming consistent,
   a. select and remove the exported “SIO” port symbol on i_uart_1 (using “Remove Item...”)
   b. select this port again, right click and choose “Export Interface”
   c. specify the exported name as “i_uart_1_SIO.” OK the dialog box.

   This method is preferable to renaming the port, since it makes all signals unique.
20. Export these interfaces out of the i_apb_sys subsystem using the Schematic menu:
   - i_apb_sys/i_apb/AHB_Slave (name = AHB_Slave)
   - i_apb_sys/i_apb/HCLK (name = HCLK)
   - i_apb_sys/i_apb/HRESETn (name = HRESETn)
   - i_apb_sys/i_uart_1/Intr (default name = ex_i_uart_1_Intr)
   - i_apb_sys/i_uart_2/Intr (default name = ex_i_uart_2_Intr)
   - i_apb_sys/i_timers/Intr0 (default name = ex_i_timers_Intr0)
   - i_apb_sys/i_timers/Intr1 (default name = ex_i_timers_Intr1)

Make sure that none of the components are still shown in ‘red’ indicating missing information.

Next, you will save your work, and move up a level in hierarchy and view the ports and connections.


22. Right-click to obtain the menu, and click on the Exit Cell menu item to move to the top level.

   Notice that this cell did not automatically connect every port. Next you will manually make these connections to clocks and the interrupt controller.

23. Toggle to the schematic view from the tree view if not already in the schematic view.

24. Select i_ictl, right-click and choose Change Connection from the menu. The following dialog appears, where you can make these connections.
a. Make the following connections by checking the appropriate box for each of these interrupts:
   ✧ i_ictl/FIQ0 = i_dmac/int_combined
   ✧ i_ictl/IRQ0 = i_ahb/Itr
   ✧ i_ictl/IRQ1 = i_apb_sys/ex_i_uart_1_Intr
   ✧ i_ictl/IRQ2 = i_apb_sys/ex_i_uart_2_Intr
   ✧ i_ictl/IRQ3 = i_apb_sys/ex_i_timers_Intr0
   ✧ i_ictl/IRQ4 = i_apb_sys/ex_i_timers_Intr1
   ✧ i_ictl/IRQ5 (leave as <set unused> for now)

   Click OK in the Interface Connections dialog to make these interrupt connections.

25. Export the remaining unconnected IRQ5 off-chip (default naming).

26. Finish the connections to the i_apb_sys block by exporting the remaining signals, naming them:
   ✧ PCLK = PCLK
   ✧ PRESETn = PRESETn
   ✧ i_uart_1_SIO = uart_1_SIO
   ✧ i_uart_2_SIO = uart_2_SIO

27. Finally, export the remaining interface ports:
   ✧ i_ahb/AHB_Master3 (default naming)
   ✧ i_ahb/AHB_Slave4 (default naming)

28. Oops, we forgot to export an APB Slave interface, as shown in Figure 2-2.
   a. Double-click on the hierarchical cell “i_apb_sys” to enter it again.
   b. Select and export (using default naming) the “APB_Slave3” interface port.
   c. Exit the cell, using the menu. Note that i_apb_sys has a new port which needs to be dealt with.
   d. Select and export this new port naming it “APB_Slave3.”

Note

You have now added and configured interfaces for all the components in this subsystem.
29. Although coreAssembler performs automatic saves after each added component and when an activity successfully completes, save your workspace at this key point using the File > Save Workspace menu item.

30. Choose the Help > Check for IP Updates... menu item to see if your IP is current. You will want to know this information and possibly make updates before you synthesize or simulate your design.

You are presented an IP Update Check report window, comparing your components to those in your DESIGNWARE_HOME tree, and also the latest components that are currently available from Synopsys (via the web). STAR updates are also listed in this report, to help you determine if you need to make an update. Viewing STARs and downloading components from Synopsys requires SolvNet authentication.

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Instance Name</th>
<th>Instantiated Version</th>
<th>Search Path Version</th>
<th>Latest Available Version</th>
<th>Reported Stars (New stars in bold)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C01_timers</td>
<td>i_timers</td>
<td>2.05a</td>
<td>2.05a</td>
<td>2.05a</td>
<td>900D455189</td>
</tr>
</tbody>
</table>

The first time you use this feature, you are prompted to enable automatic update checking, and to specify the interval between checks. You can change these preferences at any time using the Edit > Preferences menu item.

For more information on the IP Update Checking feature, refer to “Component Update Checking” in the coreAssembler User Guide.

31. Close the IP Update Check Report window.

32. Click on the Report tab at the bottom of the schematic window; then click on “(re)generate report” for the Subsystem Contents to create/view the Subsystem Report. Make sure that there are no subsystem errors, as illustrated in the following figure.

### Subsystem Report

#### Subsystem Summary

<table>
<thead>
<tr>
<th>Workspace Name</th>
<th>cA_Tutorial_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Design Name</td>
<td>mychip_top</td>
</tr>
<tr>
<td><strong>Subsystem Errors</strong></td>
<td><strong>0</strong></td>
</tr>
<tr>
<td>Parameter Incompatibility Errors</td>
<td>0</td>
</tr>
<tr>
<td>Unconnected Interfaces</td>
<td>0</td>
</tr>
<tr>
<td><strong>Subsystem Components</strong></td>
<td><strong>6</strong></td>
</tr>
<tr>
<td>Exported Interface Connections</td>
<td>14</td>
</tr>
<tr>
<td>Component Interface Connections</td>
<td></td>
</tr>
<tr>
<td><strong>Subsystem ports to be created</strong></td>
<td><strong>96</strong></td>
</tr>
</tbody>
</table>

#### Hierarchical Subsystem Reports

- mychip_top
- i_apb_sys

**This is important to verify**

33. After viewing the report, click on the Dialog tab (next to the Report tab) to return to the editing environment.
2.7 Procedure 3: Specify Subsystem -- Memory Map Specification

1. Click on the Memory Map tab. (Note: if there is no “Memory Map” tab, use the Edit > Workspace Options menu and check the “Show Memory Map Specification activity” box.) The memory map view appears, as shown. This map allows you to initialize memory map values, customize those values, and validate the final memory map.

2. Click the “Initialize All” button to auto-initialize the map.

   You are prompted to provide a base address. Set it to 0x10000 and click OK.

   The spreadsheet is filled with values based on the requirements for each of the components. You can manually change any of these values to fit your needs.
The **note** APB slaves are mapped into the AHB Slave address range (beginning at 10000, and each APB slave is given a range of 1000 in the AHB Slave space).

3. Click on the “Validate All” button to see if the values create a valid memory map. A valid indication is when no message is reported.

**hint** You must wait until you have “Applied” the activity to generate and view the Memory Map report. You will do this in Procedure 4 next.

### 2.8 Procedure 4: Specify Subsystem -- Configure Components

1. Click on the **Configure Components** tab. A left pane shows the components that are configurable in a collapsible list. Here you will make several configuration changes.

2. Expand **i_ahb** (DW_ahb <version>) and click on the “Arbiter Priority Assignments.” Change the assignments as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Interface</th>
<th>Address Range</th>
<th>BaseAddress (Default)</th>
<th>BaseAddress (Normal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_ahb/AHB_Slave</td>
<td>i_ahb/Arbiter</td>
<td>0x1000</td>
<td>0x14000</td>
<td>0x14000</td>
</tr>
<tr>
<td>i_dmec/AHB_Slave</td>
<td></td>
<td>0x1000</td>
<td>0x15000</td>
<td>0x15000</td>
</tr>
<tr>
<td>i_icu/AHB_Slave</td>
<td></td>
<td>0x1000</td>
<td>0x16000</td>
<td>0x16000</td>
</tr>
<tr>
<td>i_apb_sys/i_apb/AHB_Slave</td>
<td></td>
<td>0x4000</td>
<td>0x10000</td>
<td>0x10000</td>
</tr>
<tr>
<td>i_apb_sys/i_apb/AHB_Slave</td>
<td>ex_i_ahb_AHB_Slave</td>
<td>0x1000</td>
<td>0x17000</td>
<td>0x17000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Interface</th>
<th>Address Range</th>
<th>BaseAddress (Default)</th>
<th>BaseAddress (Normal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_apb_sys/i_apb/APB_Slave</td>
<td>i_apb_sys_timers/APB_Slave</td>
<td>0x1000</td>
<td>0x10000</td>
<td>0x10000</td>
</tr>
<tr>
<td>i_apb_sys/i_uart_1/APB_Slave</td>
<td></td>
<td>0x1000</td>
<td>0x11000</td>
<td>0x11000</td>
</tr>
<tr>
<td>i_apb_sys/i_uart_2/APB_Slave</td>
<td></td>
<td>0x1000</td>
<td>0x12000</td>
<td>0x12000</td>
</tr>
<tr>
<td>APB_Slave3</td>
<td></td>
<td>0x1000</td>
<td>0x13000</td>
<td>0x13000</td>
</tr>
<tr>
<td>i_axi/AXI_Slave</td>
<td>i_axi_x2h/AXI_Slave</td>
<td>0x8000</td>
<td>0x10000</td>
<td>0x10000</td>
</tr>
</tbody>
</table>
3. Click on the Slave Arbiter Configuration item and check the following boxes:
   “Use Hard-coded Arbiter Priorities?”
   “Use Hard-coded Default Master?”

4. Expand the i_apb_sys subsystem, expand i_timers and click on “Timer 1 Configuration.”
   Set the “Width of Timer #1” to 16 bits.

5. Click on “Timer 2 Configuration” and set the width of timer #2 to 22 bits.

6. Expand i_axi (DW_axi <vers>), expand “Arbitration Options” and click on “Arbitration Type.”
   Change all Priorities for Slave Port 0 and 1 to “Fair Among Equals” priority.

7. Expand the i_ctl instance and click on “Top Level Parameters.” Check the box for “Install Priority
   Controller?”

8. Click on the “Priority Controller Configuration” and change the priority of the IRQs as follows:
   Priority level of IRQ Source 0 = 0
   Priority level of IRQ Source 1 = 3
   Priority level of IRQ Source 2 = 2
   Priority level of IRQ Source 3 = 1
   Priority level of IRQ Source 4 = 4
   Priority level of IRQ Source 5 = 5

9. Now, click “Apply” to complete the Specify Subsystem activity. You can always click on the Specify
   Subsystem activity later to return to this activity to modify any of these values.

The transcript shows that coreAssembler is Elaborating the components and making Auto-
connections. This creates a minimal design using your configuration and interface connections. After
a minute or two, your view changes to the Report tab.

10. Click the Report tab; then click “generate report” for the “Memory Map” to generate and view the
    Memory Map Specification Report:

    | Memory Map Specification Report |
    |--------------------------------|
    | Index By Master                |
    | /1_axi/AHB_Master              |
    | /ex_1_axi_APB_Master           |
    | Memory Map Diagrams            |
    | /1_axi/AHB_Master              |
    | /ex_1_axi_APB_Master           |
    | Index By Interface             |
    | /1_axi/AXT_Slave              |
    | /2_axi/AHB_Slave              |
    | /3_axi_AHB_Slave/1_axi_APB_Slave |
    | Index By Mode                 |
    | Default                       |
    | Boot                          |
    | Memory map for /1_axi/AHB_Master |
    | Interface                     |
    | Address Range                 |
    | Base address (Normal)         |
    | Base address (Boot)           |
    | 1_axi/Arbiter                 |
    | 0x1000                        |
    | 0x10000                       |
    | 1_axi/AHB_Slave               |
    | 0x1000                        |
    | 0x10000                       |

11. Click on “generate report” for all of the available reports to generate/view these reports. (Hint: Click
    the back arrow to return to the main report summary page.

12. Click on the “Component Configuration” report page and then on “/i_axi” HTML report link.
    Note: greyed entries are disabled in your current configuration.
These reports are a specification of the interfaces and registers in your subsystem design. You can also use the “DocBook” formatted XML files of these reports for use in your component specifications and databooks.

### 2.9 Procedure 5: Complete Connections

1. Click on the “Complete Connections” activity in the Activity List pane. The connections control panel appears in the view pane, as shown below:

   ![Connections Control Panel](image.png)

   This dialog has separate tabs for “Manual Connect” and “Manual Disconnect” changes, and also a schematic view (“Manual Connect Schematic” tab) where you can make graphical connections.

2. Under “Available Loads” select `i_ictl/scan_mode` and connect to ‘0’ by clicking “Tie low.” The signal is removed from the window, indicating that it now has a valid connection.

3. Click the “Show the hierarchy below the current context” checkbox for “Available Loads” to view into the `/i_apb_sys` hierarchy.

4. Set “scan_mode” for `i_timers`, `i_uart_1` and `i_uart2` to “0”. (Hint: select all these signals using ctrl-click before clicking on “Tie low”)

5. Click on the “Manual Disconnect” tab. This is where you can disconnect any manual connection. (NOTE: You must traverse the design to see signals in the views.)

6. Use the dropdowns to choose “Component” `i_ictl` and then “Port/Pin” `ictl/scan_mode`. In the left view, select the “Type” field “L” (load). The load status appears in the right window.

7. Click **Apply** to accept all of the connection changes you just made. The view now shows the Report tab.

   **No need to do anything if your design does not have these**
8. Click on the “generate report” link text to generate/view the Connections report for mychip_top.
9. Click on the “(html)” link for mychip_top “Report Link.”
   This report shows you the Auto and Manual connections within your subsystem design, and those
   ports and signals that remain unconnected.

### Connection Details Report for mychip_top

- **Automatic Connections**
- **All Connections**
- **Unconnected Ports**
- **Unconnected Pins**

### Automatic Connections

**Status:** Finished successfully.

### Auto-Created Subsystem Ports

<table>
<thead>
<tr>
<th>Direction</th>
<th>Name</th>
<th>Range</th>
<th>Interface Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>aclk</td>
<td>single bit</td>
<td>ACLK/clk</td>
</tr>
<tr>
<td>output</td>
<td>APB_Slave3_paddr</td>
<td>31:0</td>
<td>APB_Slave3/paddr</td>
</tr>
<tr>
<td>output</td>
<td>APB_Slave3_penable</td>
<td>single bit</td>
<td>APB_Slave3/penable</td>
</tr>
<tr>
<td>input</td>
<td>APB_Slave3_prdata</td>
<td>31:0</td>
<td>APB_Slave3/prdata</td>
</tr>
<tr>
<td>output</td>
<td>APB_Slave3_psel</td>
<td>single bit</td>
<td>APB_Slave3/pssel</td>
</tr>
<tr>
<td>output</td>
<td>APB_Slave3_pwdata</td>
<td>31:0</td>
<td>APB_Slave3/pwdata</td>
</tr>
<tr>
<td>output</td>
<td>APB_Slave3_pwrite</td>
<td>single bit</td>
<td>APB_Slave3/pwrite</td>
</tr>
<tr>
<td>input</td>
<td>aresetn</td>
<td>single bit</td>
<td>ARESETn/aresetn</td>
</tr>
<tr>
<td>input</td>
<td>ex_i_ahb_AHB_Master_haddr</td>
<td>31:0</td>
<td>ex_i_ahb_AHB_Master/haddr</td>
</tr>
<tr>
<td>input</td>
<td>ex_i_ahb_AHB_Master_hburst</td>
<td>2:0</td>
<td>ex_i_ahb_AHB_Master/hburst</td>
</tr>
<tr>
<td>input</td>
<td>ex_i_ahb_AHB_Master_hbusreq</td>
<td>single bit</td>
<td>ex_i_ahb_AHB_Master/hbusreq</td>
</tr>
</tbody>
</table>

10. Click on several of the report links to familiarize yourself with the report views.
2.10 Procedure 6: Generate Subsystem RTL

1. Click on the “Generate Subsystem RTL” activity. The following choices are available:

   - create the top-level design in Verilog, VHDL, System Verilog, or Verilog 1995.
   - if VHDL, you can assign a VHDL Library for each level, preferred logic type, and probes.
   - you can also add a comment for insertion into the generated files.

2. Use the defaults, except add the comment: “Synopsys coreAssembler Tutorial 1: <date>

3. Click Apply to create the design RTL.

   When the processing is complete, the view changes to the Report tab.

4. Click on the “Subsystem Source Code” link to create/view the generated files that were created:

<table>
<thead>
<tr>
<th>Generated Subsystem</th>
<th>Hierarchical Subsystem Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace Name</td>
<td>mychip_top</td>
</tr>
<tr>
<td>Top Design Name</td>
<td>i_apb_sys</td>
</tr>
<tr>
<td>Language</td>
<td>verilog</td>
</tr>
<tr>
<td>Date</td>
<td>Tue Feb 19 14:59:00 PST 2013</td>
</tr>
</tbody>
</table>

| Generated Files |
|-----------------|-----------------|-----------------|
| Name            | Description     | Generated       |
| src/mychip_top.v| Verilog top-level subsystem RTL | 02/19/13 14:59:41 |
| export/mychip_top.list|Simulation file list | 02/19/13 14:59:42 |
5. Click on the “src/mychip_top.v” link to view this file (for Verilog RTL).
6. Choose **File > Save Workspace** to save your completed RTL design.
7. Choose **File > Write Batch Script** to create a batch script that can be used to recreate the RTL design. Enter the Output file name: “cA_Tutorial_1_batch.tcl”
   Check the box to Write default parameter values. OK the dialog box.
   This script is created in the current working directory, or optionally at a location you specify by clicking on the folder icon.
8. Open the batch script file that you just created in your text viewer of choice, and view the file structure and contents.
   Notice the command that was used to create this batch file is listed in the header. Also note the “create_workspace” command (first command below the header) that will be used to create the new workspace when this script is run.
9. Close this batch script file.
10. Choose **File > Exit** to close coreAssembler, and Save Workspace if needed.

Congratulations! You have just completed Tutorial 1. In this tutorial, you:

- Invoked coreAssembler
- Created a new workspace
- Created a hierarchical level containing the APB subsystem
- Added components which used automatic bus connections to create your subsystem
- Exported AHB/AXI master and AHB/APB slave interfaces
- Created a Memory Map
- Configured instances of components within your subsystem design
- Made manual connections within your subsystem design
- Saved the completed workspace (for later use)
- Created a batch script for recreating your subsystem
- Exited coreAssembler
Tutorial 2: Creating a Testbench

3.1 Description
In this tutorial, you will create a VMM testbench (defined in Verification Methodology Manual). The VMM testbench consists of the following DesignWare Verification IP (VIP) models: TB_clkgen, AxiMasterInterface, AhbMasterInterface, AhbSlaveInterface, ApbSlaveInterface, and Monitors.

3.2 Objective
The purpose of this tutorial is to show the creation of a testbench in coreAssembler. For this tutorial, you will go straight to the Create Testbench Workspace and Simulate Subsystem Testbench activity. Tutorial 2 contains the following procedures:

❖ “Procedure 1: Generate a Workspace from a Batch Script” on page 34
❖ “Procedure 2: Creating a VMM (random) Testbench Workspace” on page 35
❖ “Procedure 4: Individual Component non-VCS Simulation” on page 36

3.3 Before You Begin
Because the tutorials are streamlined to help you complete them in minimum time, some of the general information about coreTools and coreAssembler is not provided in the tutorial flow. Instead, links are provided to these documents:

❖ coreAssembler User Guide
❖ coreConsultant User Guide
### 3.4 Procedure 1: Generate a Workspace from a Batch Script

A batch script is an easy way to save and recreate your design workspace. When you completed Tutorial 1, you saved a batch script for your subsystem. In this procedure, you will create a new workspace using this saved batch file.

1. Invoke coreAssembler if not already open. Or if you have just finished Tutorial 1, you can click on the home icon to return to the coreAssembler home page.
2. If not already closed, be sure to save and close the cA_Tutorial_1 workspace.
3. In the coreAssembler command line, issue the following command:
   ```
coreAssembler> source cA_Tutorial_1_batch.tcl
   ```
   Because the “cA_Tutorial_1 workspace already exists, you are asked if you want to “Overwrite Existing Or Rename New Workspace”.

4. Use the drop-down box to change “Overwrite” to “New Workspace” enter “cA_Tutorial_2” as the new workspace name, and then OK the dialog box.

5. The script runs, and creates the new cA_Tutorial_2 workspace, leaving you at the last activity that you completed in Tutorial 1.

**Hint**

You can also run the batch script in non-GUI mode using the command:
```bash
%coreAssembler -shell -f cA_Tutorial_1_batch.tcl
```
But first you must open the file and change the target name (cA_Tutorial_1) for the create_workspace command to “cA_Tutorial_2” and then save the file.

Next you will use this subsystem RTL workspace to create a testbench to simulate your design.
3.5 Procedure 2: Creating a VMM (random) Testbench Workspace

In this procedure, you are now going to simulate the entire subsystem using the VMM environment.

1. Re-open the cA_Tutorial_2 subsystem design workspace.
2. Again, click on the “Create Testbench Workspace” activity.
3. Change these fields as listed below (from the previous settings) and then click Apply.
   - Testbench Workspace Name = tb_cA_Tutorial_2b
   - Testbench Strategy and Language = Constrained Random (VMM) / System Verilog
4. Again, if the DUT is red, satisfy all port connections (export, set unused, or tie low/high). The DUT turns purple when all port connections are made.
5. Click on the “Simulate Subsystem Testbench” activity in the Testbench workspace activity list.
6. You are asked if you want to auto-complete all of the testbench creation activities. Click Yes to use the defaults for these activities, auto-Generate Testbench HDL, and view the Simulate Subsystem Testbench activity. You can optionally create reports for each completed activity from the File > Generate Reports menu item.

When Simulate Subsystem Testbench is the current activity, notice that VCS is the only simulator choice for a VMM simulation.

7. Click the “Execution Options” tab, and choose the “Generate scripts only?” option.
8. Click Apply to generate the scripts.
9. Notice in the report that the Job Status says “Script(s) generated, but NOT run.”
10. Save your testbench workspace, and exit coreAssembler.
11. In your UNIX environment open a new term window and navigate to the tb_cA_Tutorial_2b/sim directory and view the contents of the run.scr file. This file launches the batch simulation run.
12. Run the simulation by executing this file: run.scr
3.6 Procedure 4: Individual Component non-VCS Simulation

In this procedure, you will perform an individual component simulation run using a non-VCS simulator.

1. Open coreAssembler on the cA_Tutorial_2 subsystem design.
   ```
   % coreAssembler cA_Tutorial_2
   ```

2. Expand the “Verify Component” group, and click on the “Setup and Run Simulation (for /i_ahb_sys/i_timers) activity.

   The text reveals that you must use a “GTECH” model or “source” code in order to use a non-Synopsys simulator. You must first create the “GTECH” model.

   **Note**
   The Synopsys VCS simulator reads the encrypted RTL files directly and does not require a GTECH conversion. All other supported simulators require a GTECH simulation model unless you have a source license. Additionally, you need a DesignWare license to complete the gate-level netlist generation process.

3. Expand the “Create Component GTECH Simulation Model” group and click on the “Generate GTECH Model (for /i_apb_sys/i_timers) activity. When asked if you want to complete the current activity, answer “No.”

4. Apply this activity to create a GTECH model for i_timers. The generator runs in the background, with a report window to monitor it:

   **Job Status**
   ```
<table>
<thead>
<tr>
<th>Run Style</th>
<th>Execution Host</th>
<th>Job Id</th>
<th>Job Status</th>
<th>Results File</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>us08slame00.internal.synopsys.com</td>
<td>13880</td>
<td>done</td>
<td>gtech/gtech.log</td>
</tr>
</tbody>
</table>
   ```

Last Update To Page: <date/time>
5. When the GTECH generation is done, return to the Verify Component activity for i_timers.
6. Click on the “View” item and choose “GTECH” instead of “RTL.”
7. Click on the “Simulator” item and choose a non-VCS simulator, such as MTI_Verilog or NC_Verilog. If neither are available, use VCSi.
8. Click on the “Testbench” item and un-select the Run test: “test_reset” item.
9. Click Apply to run the simulation on the i_timers GTECH Model.

10. Save your workspace (File > Save Workspace).
11. Close your cA_Tutorial_2 workspace (File > Close Workspace).

3.6.0.1 Tutorial 2 Summary

Congratulations! You have just completed Tutorial 2. In this tutorial, you:

- Created a VMM Testbench, only generating simulation scripts
- Performed the VMM simulation using the simulation scripts
- Created a GTECH Model for a subsystem instance
- Simulated a single component within the subsystem using a non-VCS simulator

---

Job Status

<table>
<thead>
<tr>
<th>Run Style</th>
<th>Execution Host</th>
<th>Job Id</th>
<th>Job Status</th>
<th>Results File</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>us03slamd00.internal.synopsys.com</td>
<td>15847</td>
<td>running (kill)</td>
<td>sim/runtest.log</td>
</tr>
</tbody>
</table>

Last Update To Page: <date/time>

---

This log file is created by the runtest.sh script found in

```
/remote/techpub-u/dthomas/Training/cA_Tutorial_2/components/ispb.sys/components/i_timers/
```

The simulator and testbench preparation steps should follow immediately.
Tutorial 3: Synthesizing Your Design

4.1 Description
This tutorial familiarizes you with the Synthesis activities within coreAssembler. It shows the activities you use to set up for synthesis, and presents the pre-defined strategies and strategy options you use to perform the synthesis run.

4.2 Objective
For this tutorial, you will perform the following tasks:
❖ Recreate the subsystem you created in Tutorial 1 using a batch script.
❖ View synthesis setup activities for the subsystem using the default settings
❖ Auto-complete some synthesis activities using defaults
❖ Synthesize the subsystem using a pre-defined strategy for Design Compiler
4.3 Procedure 1: Recreating a Workspace from Batch

A batch script is an easy way to save and recreate your design workspace. When you completed Tutorial 1, you saved a batch script for your subsystem. In this procedure, you will create a new workspace using this saved batch file.

1. Set your working directory to the directory containing the cA_Tutorial_1_batch.tcl.
2. Invoke coreAssembler if it’s not already open.
3. In the command line field (the coreAssembler > prompt) at the bottom of the window, issue the following commands:

   ```
   > source cA_Tutorial_1_batch.tcl
   ```

   If not source .... you can load the tutorial 1

4. When prompted, change the workspace name as “cA_Tutorial_3” and OK the dialog.

   This step takes several minutes, and refreshes your view as the process continues.

   You are now going to perform synthesis on the subsystem. But first you can streamline coreAssembler processing by removing unneeded processing steps.

5. Expand the Create Gate-Level Netlist group and then click on the Specify Target Technology activity.

   The Specify Target Technology activity specifies the technology libraries for ASIC and/or FPGA synthesis. When you select this activity, the Library Setup tabs are displayed. Prior to showing this pane for the first time, your setup file (.synopsys_dc.setup) is read to determine initial values for the fields.

   The following figure shows default library setup information for an environment that has the current DesignWare Foundation (DWBB) Library installed. If you want to add your own libraries, click to navigate to the location of the specific library.

   Path: "/project/linuxlab/cadence/vendors/VTVT/vtvt_tsmc180/Synopsys_Libraries/libs/vtvt_tsmc180.db"

   Path: "/project/linuxlab/cadence/vendors/VTVT/vtvt_tsmc180/Synopsys_Libraries/libs/vtvt_tsmc180.db"

   Path: "/project/linuxlab/cadence/vendors/VTVT/vtvt_tsmc180/Synopsys_Libraries/libs/vtvt_tsmc180.lib"
For this tutorial, you are going to use the default synthesis settings, so you will be auto-completing most of the activities in the Create Gate-Level Netlist activity group.

6. Click **Apply** for this activity.

7. When “Specify Target Technology” summary page appears, click the “Implementation Technology” report link; the Technology Report appears.

### Technology Report

**DC Technology Setup**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>search_path</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>/globals/apps/syn_2010.12/libraries/syn</td>
</tr>
<tr>
<td></td>
<td>/globals/apps/syn_2010.12/dw/syn_ver</td>
</tr>
<tr>
<td></td>
<td>/globals/apps/syn_2010.12/dw/sim_ver</td>
</tr>
<tr>
<td></td>
<td>/usi/small/libraries</td>
</tr>
<tr>
<td>target_library</td>
<td>lsi_10k.do</td>
</tr>
<tr>
<td>link_library</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>lsi_10k.do</td>
</tr>
</tbody>
</table>

For more information about the synthesis activities in coreAssembler, refer to the *coreAssembler User Guide* (“Creating the Gate-Level Netlist for a Subsystem” chapter). This manual can also be accessed from **Help > coreAssembler Tool Help > User’s Guide**. To obtain basic definitions for any of the activity parameters in the Create Gate-Level Netlist activity group, click on the Help tab for the specific activity window.

Also, the synthesis flow is documented in each DesignWare component’s databook. You can access the databook of any DW component that you include in your subsystem from the coreAssembler **Help** menu.

8. Click on the **Perform ASIC Synthesis** activity.
a. You are prompted to complete the remaining activities in the **Create Gate-Level Netlist** activity group. Click **Yes**.

---

**Enable activity**

Display the activity 'Perform ASIC Synthesis' automatically completing all activities that it depends on?

- Initialize Subsystem Constraints
- Specify Clock(s)
- Specify Operating Conditions & WireLoads
- Specify Timing Exceptions
- Specify Port Constraints
- Specify Synthesis Methodology
- Specify Test Methodology

Processing will stop at any activity which cannot be completed automatically.

[Yes] [No]

---

**Note**

This may take several minutes to complete. Depending on your Technology Library, you may be prompted to specify Operating Conditions. For example, you may have to specify a value for the `OperatingConditionsWorst` such as `WCCOM`. Once you enter a value, click Apply.

You may also pause with a Wireloads Report, in which case, click the “Perform ASIC Synthesis” activity again and Yes to finish automatic completion.

The **Perform ASIC Synthesis** activity window is displayed and becomes active when the auto-completion process has finished. Here you choose the Synthesis “Strategy.”.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Options</th>
<th>Licenses</th>
<th>Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategy:</td>
<td>Design_Compiler_Reference_Methodology</td>
<td>Options</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>Performs a one pass, top down compile from RTL using compile.Ultra. This flow implements the Design Compiler Reference Methodology associated with the version of Design Compiler being used.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9. Click the pull-down arrow on the Strategy field and examine the choices; each is described in the *coreAssembler User Guide*.

10. Click on and examine the controls available to you for each of the options tabs:
Using DesignWare Library IP in coreAssembler

- **Options** (generate scripts only, run style, notification)
- **Licenses** (indicate which licenses can be used during the Synthesis run)
- **Reports** (indicate which reports you will gather data for)

11. Click on the “Options” button to the right of the Strategy field (not the Options tab). This allows you access to all of the Strategy Parameters to control the synthesis run. It also allows inclusion of files and scripts to be executed during the synthesis run. We will use the default values.

⚠️ **Attention**
Before launching a synthesis run, you may want to only “generate scripts,” so that you can control the run yourself using the `<workspace>/syn/run.scr/Makefile`. If so, first choose this option from the **Options** tab.

12. In the **Perform ASIC Synthesis** activity window pane, click **Apply** to accept the default (Design Compiler Reference Methodology) settings and begin the synthesis run.

After several minutes, the Synthesize activity is marked completed and the “coreAssembler Synthesis Results” window is displayed. This *does not* mean that the synthesis run has completed, just the steps to launch the run have completed. Synthesis runs in the background and the window automatically refreshes its “Job Status” until synthesis has finished.

<table>
<thead>
<tr>
<th>Run Style</th>
<th>Execution Host</th>
<th>Job Id</th>
<th>Job Status</th>
<th>Results File</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>us03dwm031</td>
<td>22810</td>
<td>running</td>
<td>full</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>report/synthesis.html</td>
</tr>
</tbody>
</table>

**Note**

The synthesis results can be found beneath the workspace directory in the “syn” folder. For a description of the files that are created in this folder, refer to the *coreAssembler User Guide*. 

---

After you have synthesized the entire subsystem, you can find the gate-level netlist (db or Verilog) in the `<workspace>/syn/final/db` directory.
4.4 Tutorial 3 Summary

Congratulations! You have just completed Tutorial 3. In this tutorial, you:

❖ Created a workspace from a batch file
❖ Examined the libraries specified in the Specify Target Technology activity
❖ Specified synthesis constraints, clocks, and timing, using defaults.
❖ Examined the synthesis options and strategies for the Synthesize activity
❖ Synthesized the subsystem
❖ Saved your results, and examined the data created.
Table 5-1 lists toolbar icons and the corresponding menu items and commands to assemble a subsystem.

### Table 5-1  coreAssembler Menu and Toolbar Options

<table>
<thead>
<tr>
<th>Toolbar Icon</th>
<th>Menu Item</th>
<th>Description</th>
<th>Command Line Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Install coreKit</td>
<td>Install a coreKit. Note: DesignWare synthesizable IP for AMBA 2/AMBA 3 AXI does not require installation.</td>
<td>batch_install</td>
</tr>
<tr>
<td></td>
<td>New Workspace</td>
<td>Create a new coreAssembler generic workspace (not generally used in coreAssembler). To create an AMBA-specific coreAssembler workspace, use the <strong>create a new AMBA subsystem now</strong> link on the home page.</td>
<td>create_workspace</td>
</tr>
<tr>
<td></td>
<td>Open Workspace</td>
<td>Open an existing workspace.</td>
<td>open_workspace</td>
</tr>
<tr>
<td></td>
<td>Save Workspace</td>
<td>Save the currently loaded workspace.</td>
<td>save_workspace</td>
</tr>
<tr>
<td>None</td>
<td>Save Workspace As</td>
<td>Save the currently loaded workspace into a new directory.</td>
<td>duplicate_workspace</td>
</tr>
<tr>
<td>None</td>
<td>Close Workspace</td>
<td>Close a workspace.</td>
<td>close_workspace</td>
</tr>
<tr>
<td>None</td>
<td>Generate Reports</td>
<td>Allows selection and generation of activity reports (reports are not automatically generated).</td>
<td>generate_reports</td>
</tr>
<tr>
<td>None</td>
<td>Generate Optional Views</td>
<td>Allows selection and generation of alternate views (alternate views are not automatically generated)</td>
<td>generate_views</td>
</tr>
<tr>
<td>None</td>
<td>Write Batch Script</td>
<td>Create a TCL batch script to re-create the current state of your workspace.</td>
<td>write_batch_script</td>
</tr>
<tr>
<td>Toolbar Icon</td>
<td>Menu Item</td>
<td>Description</td>
<td>Command Line Mode</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>None</td>
<td>Build Debug Tarfile</td>
<td>Generates debug info for Synopsys support. Creates file named &lt;design&gt;_debug.tar.gz</td>
<td>build_debug_tarfile</td>
</tr>
<tr>
<td>None</td>
<td>Write IP-XACT File</td>
<td>Export the current subsystem using IP-XACT XML file format.</td>
<td>write_spirit_component</td>
</tr>
<tr>
<td>None</td>
<td>Read IP-XACT File</td>
<td>Import an IP-XACT formatted XML file for use in coreAssembler.</td>
<td>read_spirit_file</td>
</tr>
<tr>
<td>None</td>
<td>Remove IP-XACT File</td>
<td>Remove an IP-XACT generators and other elements from coreAssembler.</td>
<td>remove_spirit_file</td>
</tr>
<tr>
<td>None</td>
<td>Tables...</td>
<td>Submenu of read/write table commands allowing table-driven data entry.</td>
<td>read_attribute_table</td>
</tr>
<tr>
<td></td>
<td>&gt; Read Attribute Table</td>
<td></td>
<td>write_attribute_table</td>
</tr>
<tr>
<td></td>
<td>&gt; Write Attribute Table</td>
<td></td>
<td>read_pin_connection_table</td>
</tr>
<tr>
<td></td>
<td>&gt; Read Pin Connection Table</td>
<td></td>
<td>write_pin_connection_table</td>
</tr>
<tr>
<td></td>
<td>&gt; Write Pin Connection Table</td>
<td></td>
<td>read_subsystem_table</td>
</tr>
<tr>
<td></td>
<td>&gt; Read Subsystem Table</td>
<td></td>
<td>write_subsystem_table</td>
</tr>
<tr>
<td></td>
<td>&gt; Write Subsystem Table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>Exit</td>
<td>Exit coreAssembler.</td>
<td>exit or quit</td>
</tr>
</tbody>
</table>

**Edit Menu**

<table>
<thead>
<tr>
<th>Toolbar Icon</th>
<th>Menu Item</th>
<th>Description</th>
<th>Command Line Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Tool Installation Roots</td>
<td>Specify tool installation directories.</td>
<td>set_tool_root</td>
</tr>
<tr>
<td>None</td>
<td>Preferences</td>
<td>Specify GUI preferences and some verification, execution, synthesis, and IP-XACT options.</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Workspace Options</td>
<td>Customize which activities and functionality are available in the current workspace.</td>
<td>set_workspace_options</td>
</tr>
</tbody>
</table>

**View Menu**

<table>
<thead>
<tr>
<th>Toolbar Icon</th>
<th>Menu Item</th>
<th>Description</th>
<th>Command Line Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Home Page</td>
<td>Show the coreAssembler Home Page in the Activity View pane.</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Reports</td>
<td>View a list of the available reports for the current subsystem.</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Dialog Window Console Window Activity Window Application Toolbar Schematic Toolbar</td>
<td>Show or hide the toolbars and windows. Checkbox indicates the item is visible.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Table 5-1  coreAssembler Menu and Toolbar Options (Continued)

<table>
<thead>
<tr>
<th>Toolbar Icon</th>
<th>Menu Item</th>
<th>Description</th>
<th>Command Line Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Schematic Menu</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add New Component</td>
<td>Add component instance to subsystem. Also insert glue logic.</td>
<td>instantiate_component</td>
</tr>
<tr>
<td></td>
<td>Add Hierarchical Level</td>
<td>Add hierarchical cell to subsystem.</td>
<td>create_hierarchical_component</td>
</tr>
<tr>
<td></td>
<td>Group components</td>
<td>Move selected components into a hierarchical cell, preserving connections.</td>
<td>group_components</td>
</tr>
<tr>
<td></td>
<td>Ungroup component</td>
<td>Add hierarchical cell to subsystem.</td>
<td>ungroup_component</td>
</tr>
<tr>
<td></td>
<td>Change Views</td>
<td>Changes view in multi-view IP-XACT XML component.</td>
<td>set_component_view get_component_view</td>
</tr>
<tr>
<td></td>
<td>Import Component</td>
<td>Import an unpackaged component.</td>
<td>import_component</td>
</tr>
<tr>
<td></td>
<td>Duplicate Component</td>
<td>Duplicate selected component.</td>
<td>duplicate_component</td>
</tr>
<tr>
<td></td>
<td>Remove Item</td>
<td>Remove selected component or interface from subsystem.</td>
<td>remove_component remove_exported_interface</td>
</tr>
<tr>
<td></td>
<td>Replace Component</td>
<td>Replace selected component.</td>
<td>replace_component</td>
</tr>
<tr>
<td></td>
<td>Attach Interfaces</td>
<td>Attach interfaces to the selected component.</td>
<td>attach_interface</td>
</tr>
<tr>
<td></td>
<td>Rename</td>
<td>Rename selected component or exported interface.</td>
<td>rename_component rename_exported_interface</td>
</tr>
<tr>
<td></td>
<td>Enter Cell Exit Cell</td>
<td>You can navigate into and out of hierarchical components using the hierarchy context pull-down box just left of the navigation icons. For more information about adding hierarchical components to your design, see the “New Hierarchical Component” section in the coreAssembler User Guide.</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Edit Search Path</td>
<td>Edit component search path.</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Load Interface Definitions</td>
<td>Loads an interface definition file</td>
<td>load_interface_definitions</td>
</tr>
<tr>
<td></td>
<td>Change Connection</td>
<td>Change selected component/interface connection.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Table 5-1  coreAssembler Menu and Toolbar Options (Continued)

<table>
<thead>
<tr>
<th>Toolbar Icon</th>
<th>Menu Item</th>
<th>Description</th>
<th>Command Line Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Export Interface</td>
<td>Export selected interface.</td>
<td>export_interface</td>
</tr>
<tr>
<td>None</td>
<td>Export all Unconnected Interfaces</td>
<td>Export any unconnected interface that requires a connection.</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Set Unused</td>
<td>Set selected interface to unused</td>
<td>set_unused_interface</td>
</tr>
<tr>
<td>None</td>
<td>Mark All Unconnected Interfaces as Unused</td>
<td>Globally “set unused” for each unconnected interface.</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Detach Interface</td>
<td>Remove an attached interface from the given component</td>
<td>detach_interface</td>
</tr>
<tr>
<td>None</td>
<td>Port Associations</td>
<td>Associate and setup interface ports. Only enabled when you have selected an exported interface or an interface attached to an imported component.</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Filter</td>
<td>Filter displayed interfaces (Schematic View).</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Propagate Memory Map</td>
<td>Propagate the memory maps visible from an exported slave interface to that slave.</td>
<td>propagate_memory_map</td>
</tr>
<tr>
<td>None</td>
<td>Remove Memory Map</td>
<td>Remove a memory map.</td>
<td>remove_memory_map</td>
</tr>
<tr>
<td></td>
<td>Inspect &gt; Toggle View</td>
<td>Toggle Activity View pane between Schematic and Tree views.</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Inspect &gt; Inspect Component</td>
<td>Open Component Inspector.</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Inspect &gt; Component List</td>
<td>Open Component List Inspector.</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Inspect &gt; Exported Interface List</td>
<td>Open Exported Interface List Inspector.</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Open All</td>
<td>Expand all list items (Tree View).</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Close All</td>
<td>Close all list items (Tree View).</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Close All Components</td>
<td>Close all component list items (Tree View).</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Move &gt; Move Item Up</td>
<td>Move list item up list (Tree View).</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Move &gt; Move Item Down</td>
<td>Move list item down list (Tree View).</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Zoom &gt; Zoom In 2X</td>
<td>Zoom view in 2x (Schematic View).</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Zoom &gt; Zoom Full</td>
<td>Zoom view to fit all (Schematic View).</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Zoom &gt; Zoom Out 2X</td>
<td>Zoom view out 2x (Schematic View).</td>
<td>N/A</td>
</tr>
</tbody>
</table>
## Table 5-1  coreAssembler Menu and Toolbar Options (Continued)

<table>
<thead>
<tr>
<th>Toolbar Icon</th>
<th>Menu Item</th>
<th>Description</th>
<th>Command Line Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Help Menu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>About</td>
<td>Displays the coreAssembler initial window.</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Check for IP Updates</td>
<td>Performs an IP check against the DesignWare Home Library, and against the latest IP on the Synopsys website.</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Show IP Update Report</td>
<td>Displays the most recent IP check report.</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>coreAssembler Tool Help &gt; Error Messages &gt; User’s Guide &gt; Command Reference &gt; Tutorial &gt; Release Notes &gt; Application Notes</td>
<td>Provides links to the various online manuals related to coreAssembler.</td>
<td>N/A</td>
</tr>
<tr>
<td>None</td>
<td>Help for component /&lt;instance_name&gt; Support Info for &lt;cmp&gt; Data Book Release Notes Check Environment</td>
<td>Provides access to documentation for each of the DESIGNWARE_HOME components used in your subsystem. Check Environment performs a tool version check (should be run before using simulation or synthesis).</td>
<td>N/A</td>
</tr>
<tr>
<td>Icon Only Reference</td>
<td>N/A</td>
<td>Turns the cursor into an arrow with a question mark that allows you to obtain “What's This?” help when you right-click on an item.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
## Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>import_component</td>
<td>47</td>
</tr>
<tr>
<td>instantiate_component</td>
<td>47</td>
</tr>
<tr>
<td>open_workspace</td>
<td>45</td>
</tr>
<tr>
<td>read_spirit_file</td>
<td>46</td>
</tr>
<tr>
<td>remove_component</td>
<td>47</td>
</tr>
<tr>
<td>rename_component</td>
<td>47</td>
</tr>
<tr>
<td>replace_component</td>
<td>47</td>
</tr>
<tr>
<td>save_workspace</td>
<td>45</td>
</tr>
<tr>
<td>set_tool_root</td>
<td>46</td>
</tr>
<tr>
<td>set_unused_interface</td>
<td>48</td>
</tr>
<tr>
<td>set_workspace_options</td>
<td>46</td>
</tr>
<tr>
<td>source</td>
<td>34, 40</td>
</tr>
<tr>
<td>ungroup_component</td>
<td>47</td>
</tr>
<tr>
<td>write_batch_script</td>
<td>45</td>
</tr>
</tbody>
</table>

## A

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Hierarchical Level</td>
<td>21</td>
</tr>
<tr>
<td>Add New Component</td>
<td>17</td>
</tr>
<tr>
<td>Add New Components</td>
<td>20</td>
</tr>
<tr>
<td>Adding components to subsystem</td>
<td>17</td>
</tr>
<tr>
<td>AhbMasterInterface</td>
<td>33</td>
</tr>
<tr>
<td>AhbSlaveInterface</td>
<td>33</td>
</tr>
<tr>
<td>ApbSlaveInterface</td>
<td>33</td>
</tr>
<tr>
<td>Arbiter Priority Assignments</td>
<td>26</td>
</tr>
<tr>
<td>AxiMasterInterface</td>
<td>33</td>
</tr>
</tbody>
</table>

## B

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch file</td>
<td>44</td>
</tr>
<tr>
<td>Batch Script</td>
<td>34</td>
</tr>
<tr>
<td>Before You Begin</td>
<td>11</td>
</tr>
</tbody>
</table>

## C

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cA_Tutorial_1</td>
<td>21</td>
</tr>
<tr>
<td>Change Connection</td>
<td>22</td>
</tr>
<tr>
<td>Check Environment</td>
<td>18</td>
</tr>
<tr>
<td>Check for IP Updates</td>
<td>24</td>
</tr>
<tr>
<td>checkEnv.html</td>
<td>18</td>
</tr>
<tr>
<td>Close Workspace</td>
<td>37</td>
</tr>
</tbody>
</table>

## Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch_install</td>
<td>45</td>
</tr>
<tr>
<td>close_workspace</td>
<td>45</td>
</tr>
<tr>
<td>coreAssembler</td>
<td>15</td>
</tr>
<tr>
<td>create_workspace</td>
<td>34, 45</td>
</tr>
<tr>
<td>duplicate_workspace</td>
<td>45</td>
</tr>
<tr>
<td>exit</td>
<td>46</td>
</tr>
<tr>
<td>export_interface</td>
<td>48</td>
</tr>
<tr>
<td>generate_reports</td>
<td>45</td>
</tr>
<tr>
<td>generate_views</td>
<td>45</td>
</tr>
<tr>
<td>group_components</td>
<td>47</td>
</tr>
</tbody>
</table>

## Connections

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connections</td>
<td>28</td>
</tr>
</tbody>
</table>

## Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AhbMasterInterface</td>
<td>33</td>
</tr>
<tr>
<td>AhbSlaveInterface</td>
<td>33</td>
</tr>
<tr>
<td>ApbSlaveInterface</td>
<td>33</td>
</tr>
<tr>
<td>AxiMasterInterface</td>
<td>33</td>
</tr>
<tr>
<td>DW_ahb</td>
<td>13</td>
</tr>
<tr>
<td>DW_ahb_ictl</td>
<td>13</td>
</tr>
<tr>
<td>DW_apb</td>
<td>13</td>
</tr>
<tr>
<td>DW_apb_timers</td>
<td>13</td>
</tr>
<tr>
<td>DW_apb_uart</td>
<td>13</td>
</tr>
<tr>
<td>DW_axi</td>
<td>13</td>
</tr>
<tr>
<td>DW_axi_x2h</td>
<td>13</td>
</tr>
<tr>
<td>listing</td>
<td>17</td>
</tr>
<tr>
<td>Monitors</td>
<td>33</td>
</tr>
</tbody>
</table>

## Constrained Random (VMM)

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constrained Random (VMM)</td>
<td>35</td>
</tr>
</tbody>
</table>

## Core Assembler

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>coreAssembler</td>
<td>10, 13, 31, 33, 39</td>
</tr>
<tr>
<td>adding components</td>
<td>17</td>
</tr>
<tr>
<td>tutorials</td>
<td></td>
</tr>
<tr>
<td>overview of</td>
<td>11</td>
</tr>
<tr>
<td>Tutorial 1</td>
<td>13</td>
</tr>
</tbody>
</table>
workspace directory 10
coreAssembler User Guide 13, 24, 33
coreConsultant User Guide 33
coreTools Release Notes 11
Create Gate-Level Netlist 40, 41, 42
Create RTL Activities 13
create_workspace 34
Creating a Testbench 33

D
DCTCL_opto_strategy 43
Design Compiler 17, 18, 39
Design name, definition of 15
DesignWare Library 11
DESIGNWARE_HOME 10, 24
Device Under Test 10
DocBook 28
DUT 10
DW_ahb 13
DW_ahb_dmac 20
DW_ahb_ictl 13, 20
DW_apb 13
dw_apb 21
DW_apb_timers 13
dw_apb_timers 21
DW_apb_uart 13
dw_apb_uart 21
DW_axi 13, 20, 27
dw_axi 13, 20, 27

E
Environment check 18
Execution Options 35
Exit 31
Exit Cell 22
Export Interface 20, 21

F
FPGA synthesis 40

G
gate-level netlist 43
Generate Subsystem RTL 30
Generated Files report 30
Generated Sysbsystem report 30
GTECH 36, 37

Simulation Model 36

H
HCLK 22
Help 41
HRESET 22

I
i_ahb 26
i_axi 20
Installation Guide 11
Interrupts 23
FIQ0 23
IRQ 27
IRQn 23
Priority Controller 27

J
Job Status 43

M
Manual Connect 28
Manual Connect Schematic 28
Manual Disconnect 28
Memory Map
  base address 25
  Initialize All 25
  Specification Report 27
  Validate All 25, 26
Memory Map Specification 25
Monitors 33
MTI 18
MTI_Verilog 37

N
NC Verilog 18
NC_Verilog 37
New Workspace 15
new workspace 34
non-VCS Simulation 36
non-VCS simulator 37

P
Parameters
  Strategy 43
PCLK 23
Preferences 24
PRESETn 23
Using DesignWare Library IP in coreAssembler

Release Notes 11
Reports
- generate report 24, 27
- Generated Files 30
- Generated Subsystem 30
- STARs 24
- Subsystem Report 24
- Technology 41
- Wireloads 42
RTL 30, 31, 34

Save Workspace 21, 22, 31, 37
Schematic
- menu 17
- view 17, 19
Signals
- HCLK 22
- HRESET 22
- PCLK 23
- PRESETn 23
- SIO 21, 35
Simulate Subsystem Testbench 35
Simulation
- GTECH 36
- non-VCS 36
Simulator
- MTI_Verilog 37
- NC_Verilog 37
- VCSi 37
SIO 21, 35
Slave Arbiter Configuration 27
source 40
Specify Subsystem
- Add Components 17
- Configure Components 26
- Memory Map 25
Specify Target Technology 40, 44
STARs 24
Strategy Parameters 43
Synthesis 39
- Licenses 43
- Options 43
- Reports 43
- Results 43
Synthesize 41

Synthesize activity 44
System variable
- DESIGNWARE_HOME 10
System Verilog 35

Technology Report 41
Testbench
- creating 33
- VMM 35
Testbench workspace 10
Toggle View 19
Tools
- checking for 18
- Design Compiler 18
- MTI 18
- NC Verilog 18
- VCS 18
- Vera 18
Tutorial 1
- block diagram 14
- description of 13, 33
- objective of 33
- procedure for 15
- purpose of 13, 33
Tutorial 2
- description 33
- objective 33
- Procedures 34
Tutorial 3
- description 39
- Objective 39
- procedures 40
Tutorial 4
- description of 39
- objective of 39
- procedure for 40
Tutorials
- Overview 11
- overview of 11
- Tutorial 1 13

UNIX 35
VCS 18
VCSi 37
Vera 18
Index

Verilog 30
VHDL 30
VMM 35
VMM Testbench 37

W
WCCOM 42
Workspace
  create new 15
  definition of 10
  directory structure 10
  naming 15
  Testbench 10
  Tree diagram 10
Workspace name, definition of 15
Write Batch Script 31

X
XML files 28