



DesignWare® Cores LPDDR5/4/4X Memory Controller

User Guide

DWC LPDDR5/4/4X Controller - Product Code: E092-0
DWC LPDDR5/4/4X Controller AFP - Product Code: E093-0
DWC AP LPDDR5/4/4X Controller - Product Code: E094-0

Copyright Notice and Proprietary Information

© 2021 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

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 <https://www.synopsys.com/company/legal/trademarks-brands.html>

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

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.

www.synopsys.com

Contents

Revision History	7
Preface	9
User Guide Organization	9
Reference Documentation	10
Web Resources	10
STAR on the Web (SotW)	10
Synopsys Statement on Inclusivity and Diversity	10
Customer Support	10
Chapter 1	
Design Flow	13
1.1 Overview of coreConsultant	14
1.2 Design Flow	15
Chapter 2	
Configuring the Controller	17
2.1 Prerequisites for Configuring the Controller	18
2.1.1 Setting Up Your Environment	18
2.1.2 Creating a Workspace	18
2.2 Configuring the Controller	21
2.3 Running VCS Xprop Analyzer	24
2.4 Creating Optional Reports and Views	25
2.4.1 Activity-Specific Reports and Views	25
2.4.2 Format of Activity-Specific Reports	26
2.4.3 Setting Preferences for Report Generation	26
2.4.4 Generating Activity-Specific Reports	26
2.4.5 Accessing Reports	27
2.4.6 Generating and Accessing Activity-Specific Views	27
2.5 Running Spyglass Lint and CDC	28
2.5.1 Errors and Warnings	29
2.5.2 Lint	29
2.5.3 CDC	29
2.6 Running VC SpyGlass Verification	31
2.6.1 Errors and Warnings	32

2.6.2	Waivers and Constraints Used	32
Chapter 3		
	Simulating the Controller	33
3.1	PVE Testbench	34
3.1.1	PVE Testbench Details	34
3.1.2	PVE Testbench Files	34
3.1.3	List of PVE Tests	34
3.1.4	PVE Test Case Details	35
3.2	Running PVE Tests Using coreConsultant	36
3.2.1	Running the Simulation	36
3.2.2	Test Cases	36
3.2.3	Important Simulation Files	36
3.2.4	Checking Simulation Status and Results	36
3.3	Running PVE Tests from Command Line	38
Chapter 4		
	Performing Synthesis and Post-Synthesis	39
4.1	Synthesizing the Controller	40
4.1.1	Performing Synthesis	40
4.1.2	Setting Additional Synthesis Options	49
4.1.3	Checking Synthesis Results	50
4.1.4	Running Synthesis from a Unix Shell	51
4.1.5	Troubleshooting: Common Problems in this Step	51
4.2	Inserting Design for Test	52
4.3	Running Automatic Test Pattern Generation	53
4.3.1	Running ATPG using TetraMax	53
4.4	Running Low Power Static Verification Using VC LP	55
4.5	Running Static Timing Analysis	57
4.5.1	Performing Static Timing Analysis	57
4.5.2	Checking STA Results	59
4.5.3	Running STA from Unix Shell	60
4.6	Performing Formal Verification	61
4.6.1	Checking Formal Verification Results	62
4.6.2	Running Formality from a Unix Shell	62
Chapter 5		
	Integrating the Controller	63
5.1	Exporting Controller to Your Chip Design Database	64
5.1.1	Instantiating Your Controller	64
5.1.2	Exporting RTL Code	64
5.1.3	Synthesizing to a Device Outside of coreConsultant	64
5.1.4	Exporting Views and Reports	66
5.1.5	Simulating Outside coreConsultant	66
5.2	Host Interface (HIF) Integration	67
5.2.1	HIF Command Interface	67
5.2.2	HIF Write Data Request Interface	68

5.2.3	HIF Write Data Interface	68
5.2.4	HIF Read Data Interface	68
5.2.5	HIF Credit Increment Interface	69
5.3	PHY Integration.....	70
5.3.1	PHY Type	70
5.3.2	PHY Simulation Model	70
5.3.3	Imported PHY Location	70
5.3.4	Imported C code location	71
5.3.5	Imported Firmware Location	71
5.3.6	PHY Parameters that can be Controlled from coreConsultant	71
5.4	External RAM Integration	72
Appendix A		
Additional coreConsultant Information		73
A.1	Creating a Batch Script	74
A.2	Help Information	75
A.3	Workspace Directory Structure Overview	77
A.4	Dumping Debug Information when Problems Occur	79
Appendix B		
CDC Methodology		81
B.1	Clocks, Resets, and Clocking Scheme	82
B.2	Synchronizers Used in DWC_ddrctl	83
B.3	Synthesis Constraints	85
B.4	Synchronization Method	87
B.4.1	APB Static Registers	87
B.4.2	Dynamic Single Bit Signals	87
B.4.3	Pulse Signals	87
B.4.4	Data Bus Signals	88
B.4.5	Data FIFO and Replaceable Components	88
B.5	CDC Constraints and Report	89
Appendix C		
OCCAP Synthesis Constraints		91
C.1	Duplicated Modules	92
C.2	Automotive FIFO Controllers	93
C.3	Triple Module Replication (TMR) of Specific Registers	94

Revision History

The following table provides a summary of changes made to this User Guide.

Version	Date	Description
1.10a-lca00	September 2021	Added: <ul style="list-style-type: none"> ■ “STAR on the Web (SotW)” on page 10 ■ “Synopsys Statement on Inclusivity and Diversity” on page 10 Updated: <ul style="list-style-type: none"> ■ “Method 2: Use write_sdc Command” on page 65 ■ “PHY Integration” on page 70 ■ Table B-1 on page 83 ■ “Synthesis Constraints” on page 85 ■ “CDC Constraints and Report” on page 89 ■ “OCCAP Synthesis Constraints” on page 91
1.01a-lca01	January 2021	Added: <ul style="list-style-type: none"> ■ “Running VC SpyGlass Verification” ■ “Simulating Outside coreConsultant” Updated: <ul style="list-style-type: none"> ■ “Configuring the Controller” ■ “Checking Simulation Status and Results” ■ “Performing Synthesis” ■ “Imported C code location” ■ “Imported Firmware Location” ■ Table B-1 “DWC_ddrctl to DesignWare Library Part Mapping” ■ “Synthesis Constraints” ■ “Automotive FIFO Controllers”
1.00a-lca01	June 2020	Initial release

**Note**

In some instances, documentation-only updates occur. The DesignWare IP product <https://www.synopsys.com/designware-ip.html> has the latest information.

Preface

This document describes the DesignWare® Cores LPDDR5/4/4X Memory Controller. The controller, combined with DWC LPDDR5/4/4X PHY, is a complete memory interface solution for DDR memory subsystems. The LPDDR5/4/4X Memory Controller (also referred to as DDRCTL or DWC_ddrctl) is delivered as configurable Verilog source and is compatible with various EDA environments.

This section contains the following topics:

- [“User Guide Organization”](#) on page 9
- [“Reference Documentation”](#) on page 10
- [“Web Resources”](#) on page 10
- [“Customer Support”](#) on page 10

User Guide Organization

The chapters of this user guide are organized as follows:

- Chapter 1, [“Design Flow”](#) on page 13, introduces you to the coreConsultant tool, and describes the design flow of the DDRCTL.
- Chapter 2, [“Configuring the Controller”](#) describes how to configure the DDRCTL using coreConsultant tool, generate reports and views. It also describes how to run Spyglass Link and CDC.
- Chapter 3, [“Simulating the Controller”](#) describes the provided Packaged Verification Environment (PVE) Testbench, and how to use it to verify the DDRCTL using coreConsultant and from command line.
- Chapter 4, [“Performing Synthesis and Post-Synthesis”](#) describes the synthesis flow for the DDRCTL. It also shows you how to run formal verification, use the coreConsultant tool to insert DFT, run automatic test pattern generation (ATPG), and perform static timing analysis (STA)
- Chapter 5, [“Integrating the Controller”](#) describes instantiating and integrating the DDRCTL Memory Controller into a design, and in particular how to integrate it with a Synopsys DWC DDR PHY.
- Appendix A, [“Additional coreConsultant Information”](#) provides additional information about the coreConsultant.
- Appendix B, [“CDC Methodology”](#) provides detailed information about the Clock Domain Crossing (CDC) strategy used in DDRCTL.
- Appendix C, [“OCCAP Synthesis Constraints”](#) provides information on special requirements for the duplicate modules used in On Chip Command and Address Path Protection (OCCAP) feature.

Reference Documentation

This document assumes familiarity with the basic terminology and concepts outlined in:

- AMBA Specification, Rev. 2.0
<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ih0011a/index.html>
- AMBA 3 AXI Specification
<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ih0011a/index.html>

**Note**

You need to have the ARM account to access these documents.

Web Resources

- DesignWare IP product information: <https://www.synopsys.com/designware-ip.html>
- Your custom DesignWare IP page: <https://www.synopsys.com/dw/mydesignware.php>
- Documentation through SolvNet: <http://solvnet.synopsys.com> (SolvNet password required)
- Synopsys Common Licensing (SCL): <https://www.synopsys.com/support/licensing-installation-computeplatforms/licensing.html>

STAR on the Web (SotW)

You must review all STARs on the Web (SotWs) associated with your product. SotWs are considered a part of the Synopsys documentation suite, and show critical information related to your product. To review product SotWs, refer to the DesignWare IP product information:

<https://www.synopsys.com/designware-ip.html>

Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

Customer Support

To obtain support for your product:

- First, prepare the following debug information, if applicable:
 - For environment setup problems or failures with configuration, simulation, or synthesis that occur within coreConsultant or coreAssembler, use the following menu entry:

File > Build Debug Tar-file

Check all the boxes in the dialog box that apply to your issue. This menu entry gathers all the Synopsys product data needed to begin debugging an issue and writes it to the file <coreTool startup directory>/debug.tar.gz.

- For simulation issues outside of coreConsultant or coreAssembler:
 - Create a waveforms file (such as VPD or FSDB)
 - Identify the hierarchy path to the DesignWare instance
 - Identify the timestamp of any signals or locations in the waveforms that are not understood.
- Then, contact the Support Center, with a description of your question and supplying the previous information, using one of the following methods:
 - For the fastest response, use the SolvNet website. If you fill in your information as explained below, your issue is automatically routed to a support engineer who is experienced with your product. The Sub Product entry is critical for correct routing.

Go to <http://solvnet.synopsys.com/EnterACall> and click on the link to open a support case. Provide the requested information, including:

- Product: DesignWare Cores
- Sub Product: Memory - Controller
- Tool Version: 1.10a-lca00
- Problem Type:
- Priority:
- Title: DWC LPDDR5/4/4X Controller: Provide a brief summary of the issue or list the error message you have encountered.
- Description: For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood.

After creating the case, attach any debug files you created in the previous step.

- Or, send an e-mail message to support_center@synopsys.com (your email will be queued and then, on a first-come, first-served basis, manually routed to the correct support engineer):
 - Include the Product name, Sub Product name, process, and Tool Version number in your e-mail (as identified previously) so it can be routed correctly.
 - For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood
 - Attach any debug files you created in the previous step.
- Or, telephone your local support center:
 - North America:
Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
 - All other countries:
<https://www.synopsys.com/support/global-support-centers.html>

1

Design Flow

This chapter provides an overview of the Synopsys coreConsultant tool. It also discusses the design flow of the LPDDR5/4/4X Memory Controller.

The topics in this chapter are as follows:

- [“Overview of coreConsultant”](#) on page 14
- [“Design Flow”](#) on page 15

1.1 Overview of coreConsultant

The Synopsys coreConsultant tool allows you to configure, simulate, synthesize, and export the DWC_ddrctl controller to your design flow. The final output from the coreConsultant design flow is a set of RTL files (the configured controller) and scripts to allow you to run various tools standalone within your own design flow. The coreConsultant features and tasks include:

- A graphical user interface (GUI) to guide you through design flow activities.
You can also perform all activities through a command line interface. For details, see the coreConsultant User Guide and Help menu in coreConsultant.
- Interactive parameter selection
The coreConsultant tool checks for consistent settings and automatically derives any dependent parameters from your choices.
- Testbench simulation and synthesis configuration consistent with your parameter choices.
- Docbook files documenting your final configured design.

This user guide gives more detailed information on how to configure your controller and generate RTL files, as well as running synthesis and simulation.



Note

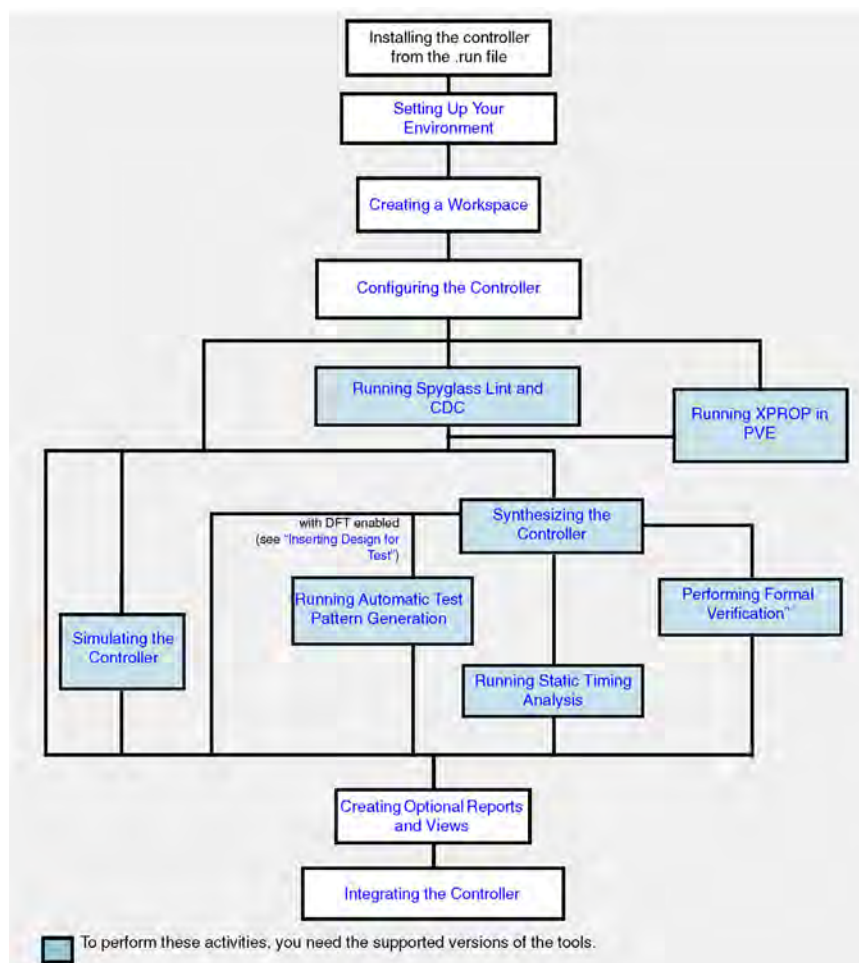
When you are using more than one DesignWare Cores, coreAssembler is recommended. The coreAssembler tool includes all of the features of coreConsultant but allows you to configure, automatically inter-connect, synthesize, and export a subsystem as opposed to a single component. For more information, see the coreAssembler documents mentioned in [“Help Information”](#) on page 75.

1.2 Design Flow

Figure 1-1 on page 15 gives the design flow for the DDRCTL. The coreConsultant GUI guides you through the controller design flow in your workspace. A workspace is a local Unix directory structure containing your configured copy of the DDRCTL (see “[Creating a Workspace](#)” on page 18).

Most coreConsultant activities are optional as indicated by the various paths shown in Figure 1-1 on page 15 and do not need to be completed before exporting the configured RTL to your chip design flow.

Figure 1-1 Design Flow



The links to corresponding sections of the User Guide are listed below:

- “[Setting Up Your Environment](#)” on page 18
- “[Creating a Workspace](#)” on page 18
- “[Configuring the Controller](#)” on page 17
- “[Running Spyglass Lint and CDC](#)” on page 28
- “[Important Simulation Files](#)” on page 36
- “[Synthesizing the Controller](#)” on page 40
- “[Running Automatic Test Pattern Generation](#)” on page 53

- [“Performing Formal Verification”](#) on page 61
- [“Simulating the Controller”](#) on page 33
- [“Running Static Timing Analysis”](#) on page 57
- [“Creating Optional Reports and Views”](#) on page 25
- [“Integrating the Controller”](#) on page 63

Configuring the Controller

This chapter provides an overview of the step-by-step process to configure, synthesize, simulate, and export the DWC_ddrctl controller using the Synopsys coreConsultant tool.

The final output from the coreConsultant design flow is a set of RTL files comprising of the configured controller or an optimized gate-level netlist for custom configuration in your target technology. The output also includes scripts that allow you to run various tools stand alone within your own custom design flow. This chapter also describes how to export the controller into your chip development environment.

The coreConsultant features and tasks include:

- A graphical user interface (GUI) to guide you through design flow activities. All activities may also be performed through a command line interface. For more information, see coreConsultant User Guide (see [“Help Information”](#) on page 75 in Appendix A [“Additional coreConsultant Information”](#)), and Help menu.
- Interactive parameter selection. coreConsultant checks for consistent settings and automatically derives any dependent parameters from your choices.
- Synthesis, verification, and test vector script generation.
- Testbench and test configuration consistent with your parameter choices.
- Docbook files documenting your final configured design.

The topics in this chapter are as follows:

- [“Prerequisites for Configuring the Controller”](#) on page 18
- [“Configuring the Controller”](#) on page 21
- [“Creating Optional Reports and Views”](#) on page 25
- [“Running Spyglass Lint and CDC”](#) on page 28
- [“Running VC SpyGlass Verification”](#) on page 31

2.1 Prerequisites for Configuring the Controller

This section describes how to set up your environment and create a workspace so that you can start configuring the controller.

2.1.1 Setting Up Your Environment

Before you install the DDRCTL, you must set the following environment variables:

1. Set the DESIGNWARE_HOME environment variable to your installation directory:

```
% setenv DESIGNWARE_HOME <path to c_ddrctl/DWC_ddrctl_Installation_Base_Directory>
```
2. Set either the SNPSLMD_LICENSE_FILE or the LM_LICENSE_FILE variable, if you have not already done so:

```
% setenv SNPSLMD_LICENSE_FILE ${SNPSLMD_LICENSE_FILE}:<my_license_file|port@host>
```

or

```
% setenv LM_LICENSE_FILE ${LM_LICENSE_FILE}:<my_license_file|port@host>
```
3. Include \$DESIGNWARE_HOME/bin in your PATH environment variable.

\$DESIGNWARE_HOME/bin is required in your PATH environment variable as it contains scripts that are necessary when using your DWC components. For more information about these scripts, see [Table A-1](#) on page 77.

For more information about setting up your environment, see the “Setting License File Environment Variable” and “Checking License Requirements” sections in the LPDDR5/4/4X Memory Controller Installation Guide.



Attention

Do not proceed unless you have completed all of these steps.

2.1.2 Creating a Workspace

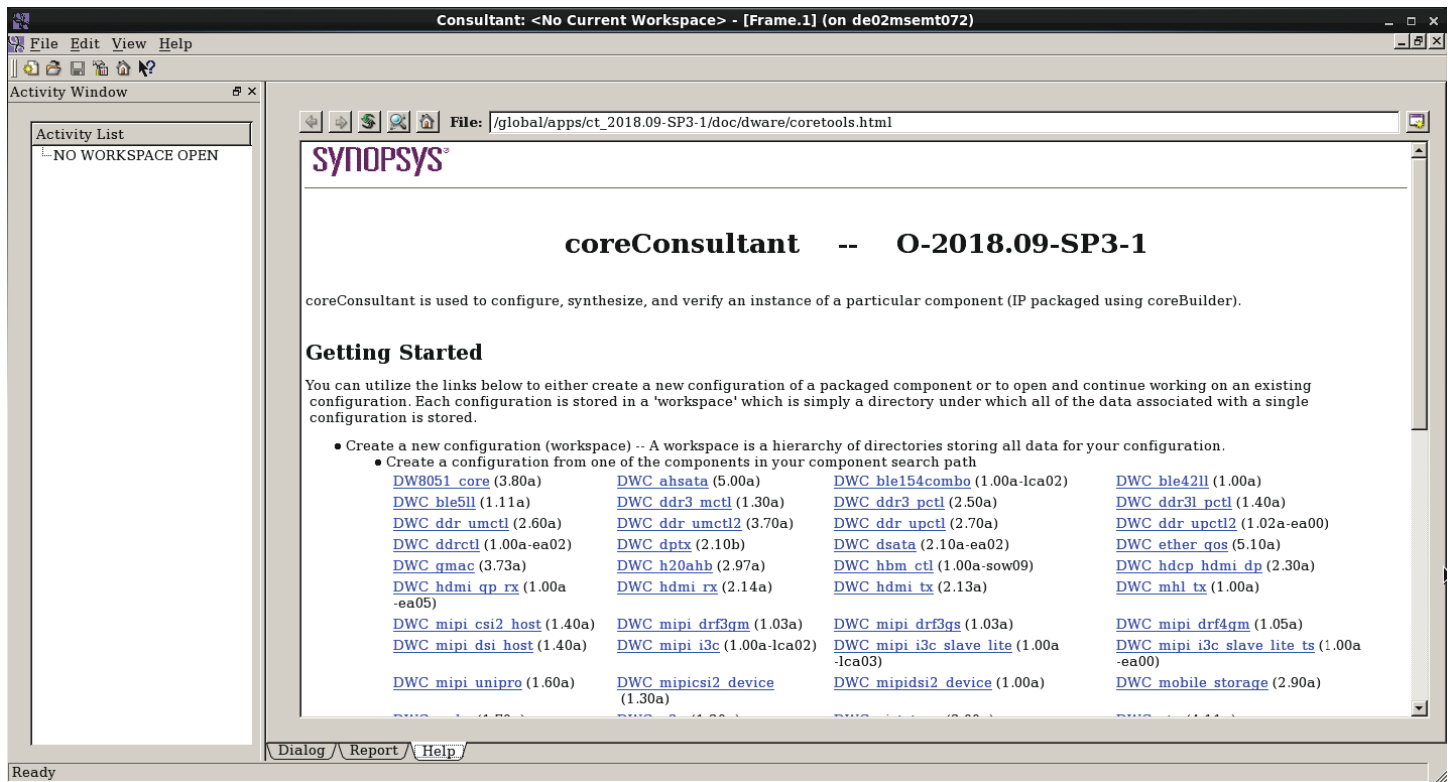
Use coreConsultant to create a workspace — your working version of the controller with a particular configuration in which you connect, configure, simulate, and synthesize your implementation of the subsystem. You can create several workspaces to experiment with different design alternatives.

To create a workspace, follow these steps:

1. In a UNIX shell, navigate to a directory where you plan to locate your component workspace.
2. Start the coreConsultant GUI:

```
% coreConsultant &
```

[Figure 2-1](#) shows an example of the coreConsultant Welcome Page.

Figure 2-1 coreConsultant Welcome Page

3. Create a workspace.

From the Welcome page (Figure 2-1), click on the name of the IP controller that you wish to configure.



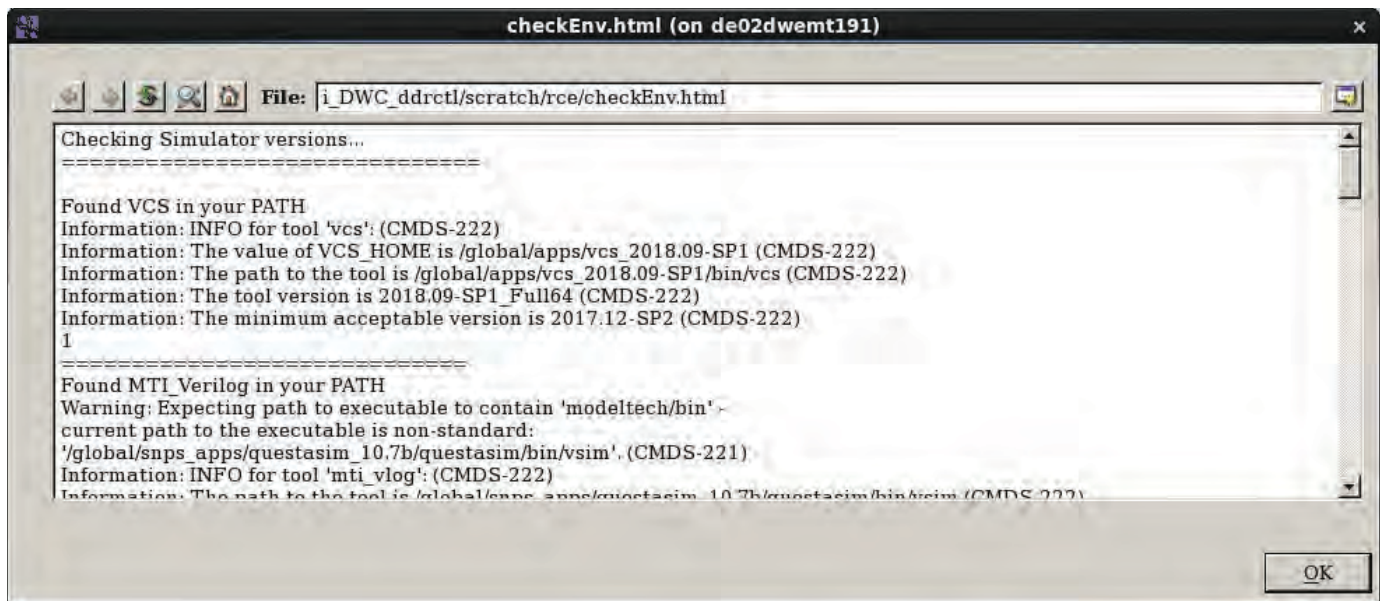
Hint

Here is a common problem that can occur:

- The DWC_ddrctl controller is not visible in the welcome page.

This issue is caused by not setting the \$DESIGNWARE_HOME environment variable (or by not setting it to your IP installation directory) in the shell from which coreConsultant was started. For information about setting \$DESIGNWARE_HOME correctly, refer to the LPDDR5/4/4X Memory Controller Installation Guide.

4. Validate your installation and environment by selecting Help->Check Environment from the menu bar. A report window shows the results (a sample is shown in Figure 2-2).

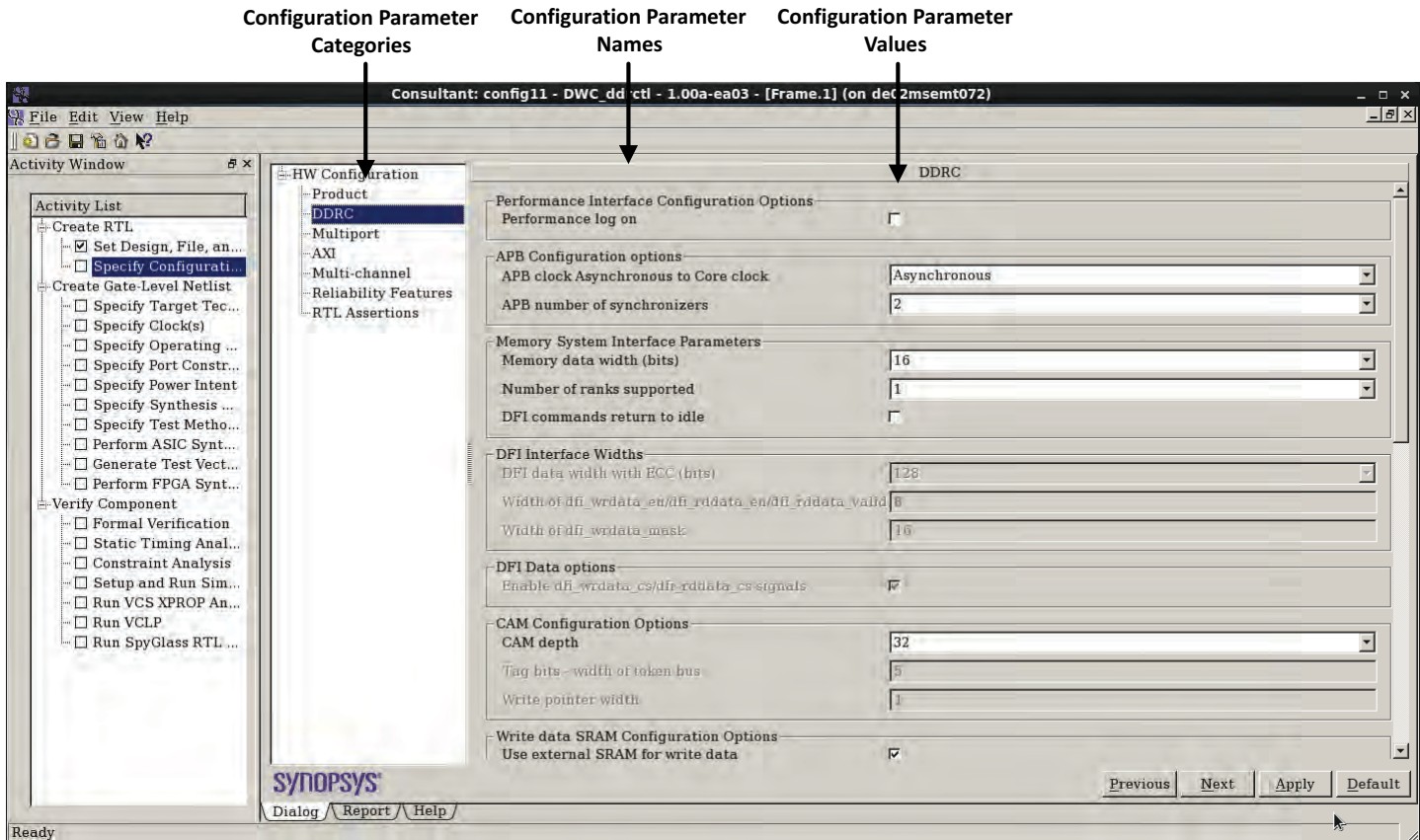
Figure 2-2 Sample Results of the Check Environment Activity

- ❑ Correct any reported errors by modifying your Unix environment setup or through the **Edit > Tool Installation Roots** menu.
- ❑ An error occurs if your \$DESIGNWARE_HOME environment variable is not set correctly. Failure to set up the \$DESIGNWARE_HOME environment variable properly before invoking coreConsultant could result in errors and unexpected IP behavior.
- ❑ To correct the \$DESIGNWARE_HOME environment variable, quit coreConsultant.
- ❑ For information about setting \$DESIGNWARE_HOME correctly, refer to the LPDDR5/4/4X Memory Controller Installation Guide.

2.2 Configuring the Controller

After you have created a workspace, configure the controller and create the RTL using the activities under **Specify Configuration** section in coreConsultant (illustrated in the following figures).

Figure 2-3 Screenshot of coreConsultant GUI dialog to configure the IP



Complete the following steps to generate the RTL:

1. Set Design and File Prefix

The Set Design Prefix activity is already checked. This setting is used so that each design in your component has a unique name. The setting is needed only when you have two or more versions of the controller. This only changes module names; parameter names are preserved, regardless of this setting.



For the remaining steps in this chapter, we recommend to apply the default values so that the directions and description in mentioned this chapter coincide with your display. After you have used DWC_ddrctl in coreConsultant, you can then go back through these steps and change values to see how they affect the design.

Only pre-verified configurations are supported.

2. Specify Configuration

This is used to enable or disable features, to select the protocols to support, to choose the correct data width, CAM depth, and so on. Built-in checks will ensure that you choose settings that are consistent with each other.

You can access the information about each parameter by right-clicking on the parameter label and selecting **What's This?** or by selecting the **Help** tab.

3. Generate RTL

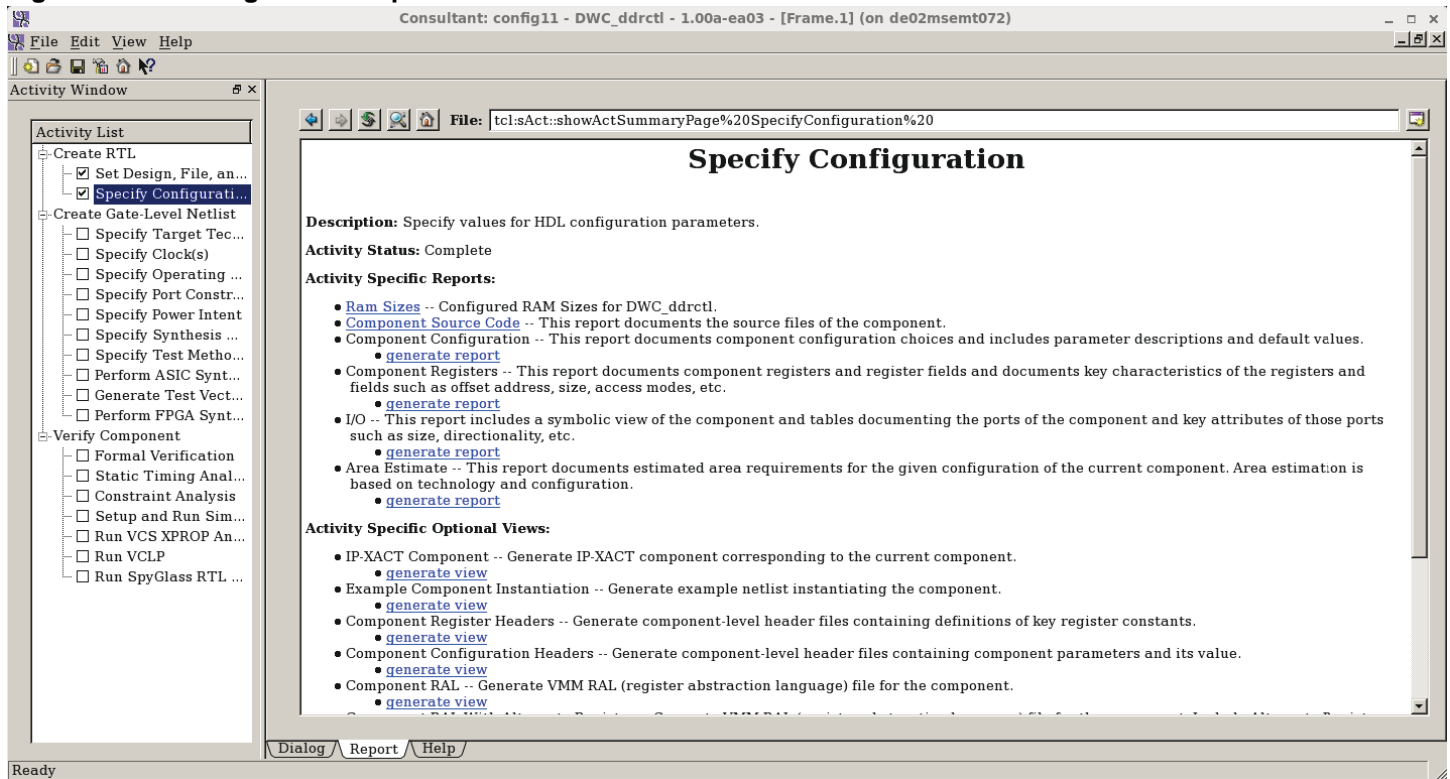
Click **Apply** to generate configured RTL code for the controller with the default values. The coreConsultant tool then checks your parameter values and generates configured RTL code in the <workspace>/src/ directory.

You can return to the Configuration activity to re-configure the controller (create new RTL) at any time. If you do so, you must complete any downstream activities again.

4. View Reports and Generate Optional Views/Reports

After you have configured the controller, coreConsultant generates several configuration reports. You can access these by clicking the **Report** tab under the **Specify Configuration** activity (see [Figure 2-4](#)).

Figure 2-4 Configuration Reports



Additionally, you can generate **Activity Specific Reports** and **Activity Specific Optional Views**. For more information, see [“Creating Optional Reports and Views”](#) on page 25. For example, to generate an IP-XACT component of the controller, click the generate view link as indicated in [Figure 2-4](#).

5. Create a Batch Script

In case you delete or overwrite your current workspace, you can create a batch script to recreate your exact configuration at a later stage. Choose the **File > Write Batch Script** menu item and enter a name for the file. This step is recommended as a batch script helps Synopsys Customer Support troubleshoot

problems you may encounter with your controller. For more information, see “[Creating a Batch Script](#)” on page [74](#).

**Hint**

Some common problems that can occur in the **Specify Configuration** activity:

- Encrypted source code gets generated. Typically this problem is caused by not providing a Project ID when installing the DWC_ddrctl controller. You must re-install the DWC_ddrctl using the Project ID number.
- The coreConsultant tool reports an error that it can not write certain files. Check the available storage. You may have too little storage on your server.

**Note**

At this point, you can also:

- Generate a gate-level netlist (“[Synthesizing the Controller](#)” on page [40](#))
- Verify the controller (“[Simulating the Controller](#)” on page [33](#))

2.3 Running VCS Xprop Analyzer

After generating the RTL in the Specify Configuration activity, you can run the Xprop feature of the VCS simulator in the coreConsultant environment to check for the instrumentation capabilities of Xprop on the generated RTL. The RTL is analyzed with relevant VCS switches. You can check the resulting log file to ensure the RTL is 100% instrumented. For more details on VCS Xprop, see the VCS documentation.

Running VCS Xprop Analyzer

You must configure the controller (see “[Configuring the Controller](#)” on page 21) before starting this task. To run the VCS Xprop analyzer on your controller, complete the following steps in the Run VCS Xprop Analyzer activity (see Figure 2-15).

- **Specify Command Line Switches**

The following command line switches can be passed to VCS. By default, you do not need to specify any argument; you can leave these fields blank.

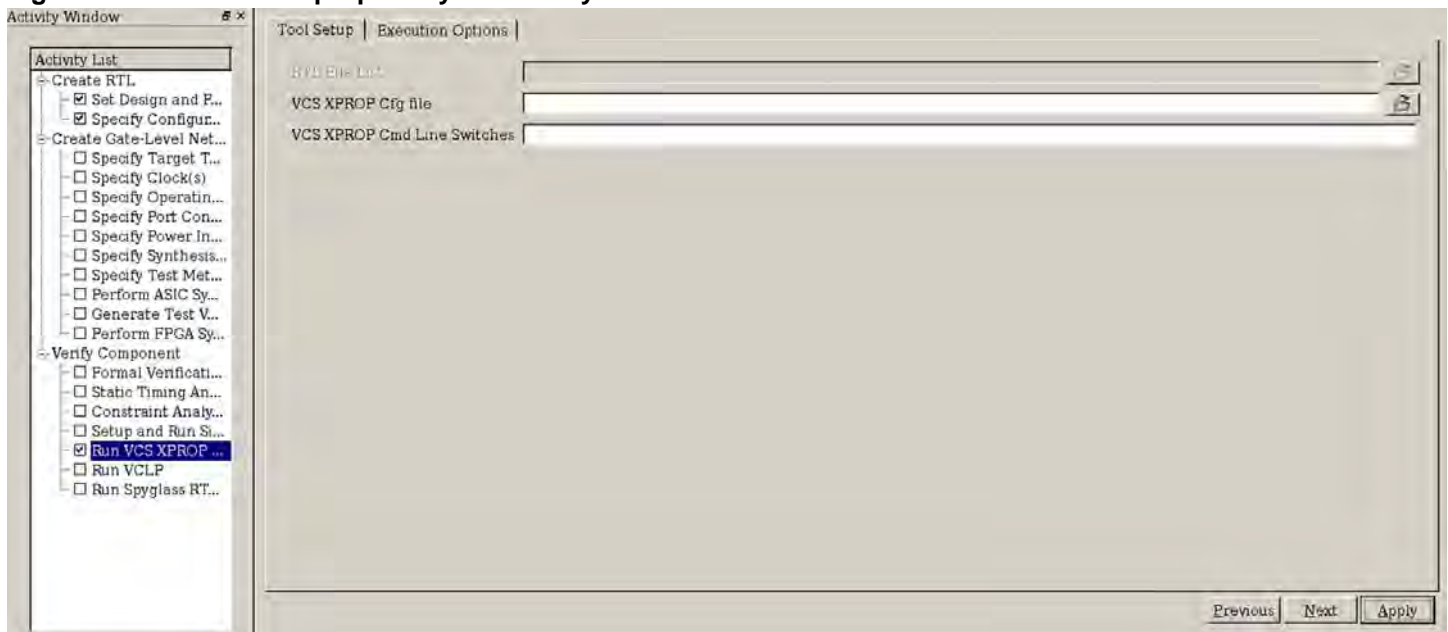
- **VCS Xprop Cfg File:** Specify the full path of the xprop config file that you want to pass to VCS as `-xprop=<CfgFile>`.

By default, you do not need to specify any file; you can leave this field blank.

- **VCS Xprop Cmd Line Switches:** You can also specify any additional switches that you want to pass to the VCS command line when running the VCS executable. By default, you do not need to specify any additional switches; you can leave this field blank.

- Click **Apply** to run the VCS Xprop Analyzer activity.

Figure 2-5 Run VCS Xprop Analyzer Activity



Checking VCS Xprop Analyzer Results

- Click the **Report** tab.
- Click the **Run VCS Xprop Analyzer Summary** link.

In the opened spreadsheet, the **Instrumentation** row should show 100%. If it is not 100%, then additional information will be provided for the non-instrumented statements.

2.4 Creating Optional Reports and Views

After configuring the controller, you can generate several configuration reports. The coreConsultant tool displays a list of activity-specific views you can optionally generate. You can access this list by clicking the **Report** tab, shown in [Figure 2-4](#) on page 22, in the **Specify Configuration** activity.



Attention

Some of the activity-specific report creation may not be fully functional with your controller. For assistance, contact Synopsys Customer Support.

2.4.1 Activity-Specific Reports and Views

You can generate reports that document the I/O signals, parameters, and register descriptions.

[Table 2-1](#) describes the activity-specific optional reports.

Table 2-1 Activity-Specific Reports

Report	File Names (in <workspace>/report)	Description
Component Configuration	ComponentConfiguration.html ComponentConfiguration.xml	Documents component configuration choices and includes parameter descriptions and default values
Component Registers	ComponentRegisters.html ComponentRegisters.xml	Documents component registers and register fields; documents key characteristics of the registers and fields such as offset address, size, and access mode
I/O	IO.html IO.xml	Includes a symbolic view of the component and tables documenting the ports of the component and key attributes of those ports such as size, direction

You can also generate other reports and views such as area estimates, IP-XACT (to use IP-XACT XML format), component level header files, example component instantiation, and RAL files. These reports can be viewed through the coreConsultant Reports tab, and they are saved in the <workspace>/export directory. [Table 2-2](#) describes the activity-specific optional views.

Table 2-2 Activity-Specific Optional Views

Optional Views	File Name (in <workspace>/export)	Description
IP-XACT Component	DWC_ddrctl.xml	Generates IP-XACT component corresponding to the current component.
Example Component Instantiation	DWC_ddrctl_inst.v	Generates an example netlist that instantiates the component.
Component Register Headers	./headers/*	Generates component-level header files containing definitions of key register constants.

**Note**

- To automatically generate these reports or views, select appropriate check boxes in the **File > Generate Reports or File > Generate Optional Views** dialog box.
- You can modify the schema version for IP-XACT output files by changing user's preferences as follows:

Edit -> Preferences -> IP-XACT -> Schema version for IP-XACT output files

2.4.2 Format of Activity-Specific Reports

The reports are generated in XML using the DocBook schema (an open source XML-based language). These reports are subsequently rendered in HTML applying an XSLT style sheet.

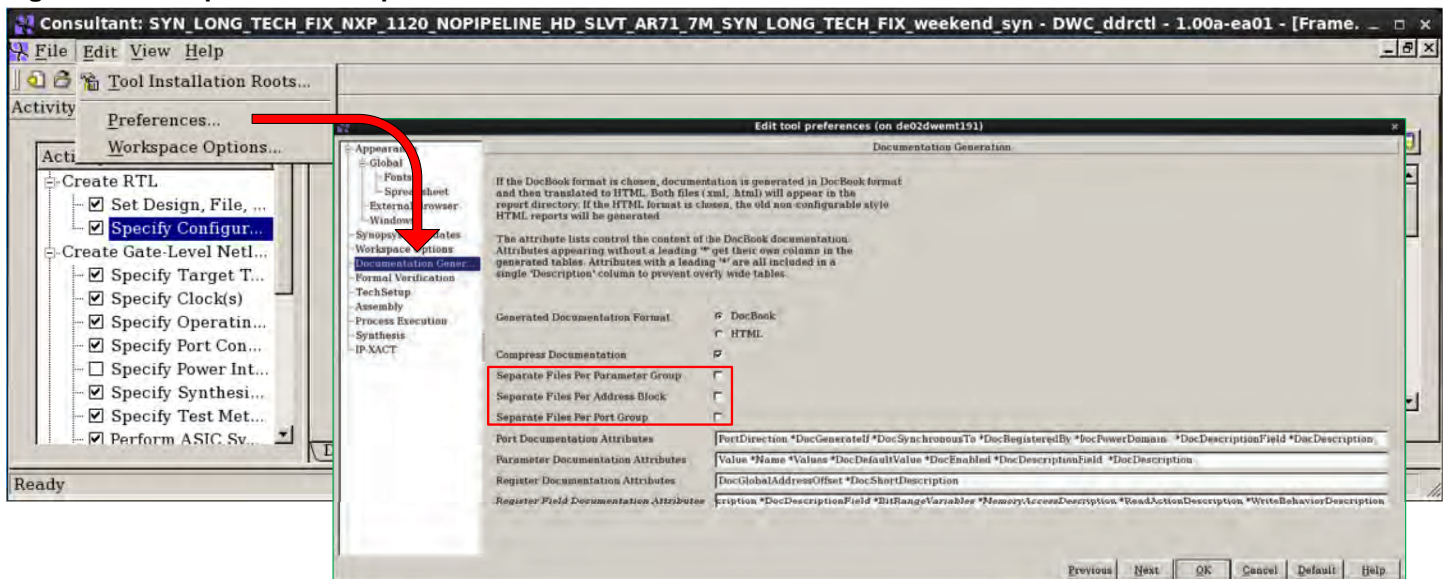
2.4.3 Setting Preferences for Report Generation

It is possible to generate reports as a single file or multiple files based on specific groups of signals and registers. To generate reports contained in a single file for easier navigation, set the preference as follows:

1. In the coreConsultant GUI, click **Edit > Preferences**.
2. In the **Edit tool preferences** window, select **Document Generation**.
3. Uncheck the "Separate Files Per ..." check boxes, as shown in [Figure 2-6](#).

You have to do this only once because these settings are saved in your `~/ .synopsys_rt_prefs.tcl` file. You must do this before you create the reports. If you have previously created reports with old settings, delete the workspace and start again.

Figure 2-6 Separate Files Options



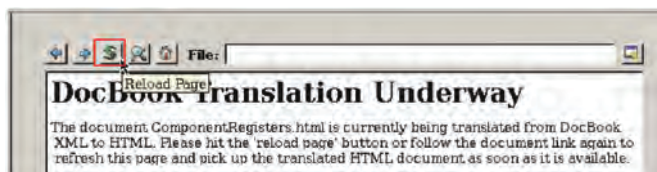
2.4.4 Generating Activity-Specific Reports

To generate the required report, click the respective **generate report**. For example, to generate component registers-related reports, click the one highlighted in [Figure 2-6](#).

2.4.5 Accessing Reports







To view the report, click **Reload Page** after clicking **generate report** as shown in [Figure 2-7](#).

Figure 2-7 Viewing Reports



You can also access these report files from the **<your workspace>/report** directory. The content of this directory is similar to that shown in [Figure 2-8](#).

Figure 2-8 Contents of the Report Directory

Name	Type
 ComponentConfiguration.html	HTML Document
 ComponentConfiguration.xml	XML Document
 ComponentRegisters.html	HTML Document
 ComponentRegisters.xml	XML Document
 IO.html	HTML Document
 IO.xml	XML Document

2.4.6 Generating and Accessing Activity-Specific Views

To generate the required view, click the respective **generate view** in the **Report** tab.

Access the generated views from the **<your workspace>/export** directory.

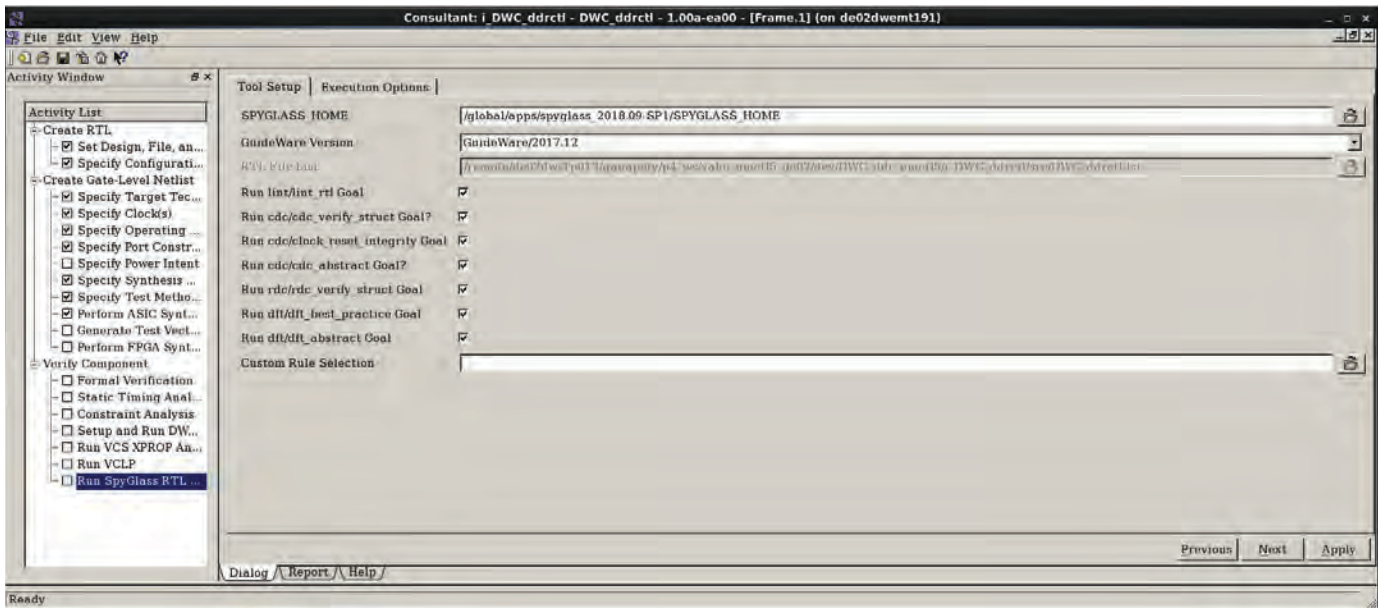
2.5 Running Spyglass Lint and CDC

The Spyglass flow in coreConsultant runs Guidewares rules for block/RTL handoff. Within the block/RTL handoff, only lint/lint_rtl and cdc/cdc_verify_struct goals are run.lc

Complete the following steps to run Spyglass Lint and CDC.

1. Select either Lint, CDC, or both run goals as shown in [Figure 2-9](#). By default, both Lint and CDC are enabled.

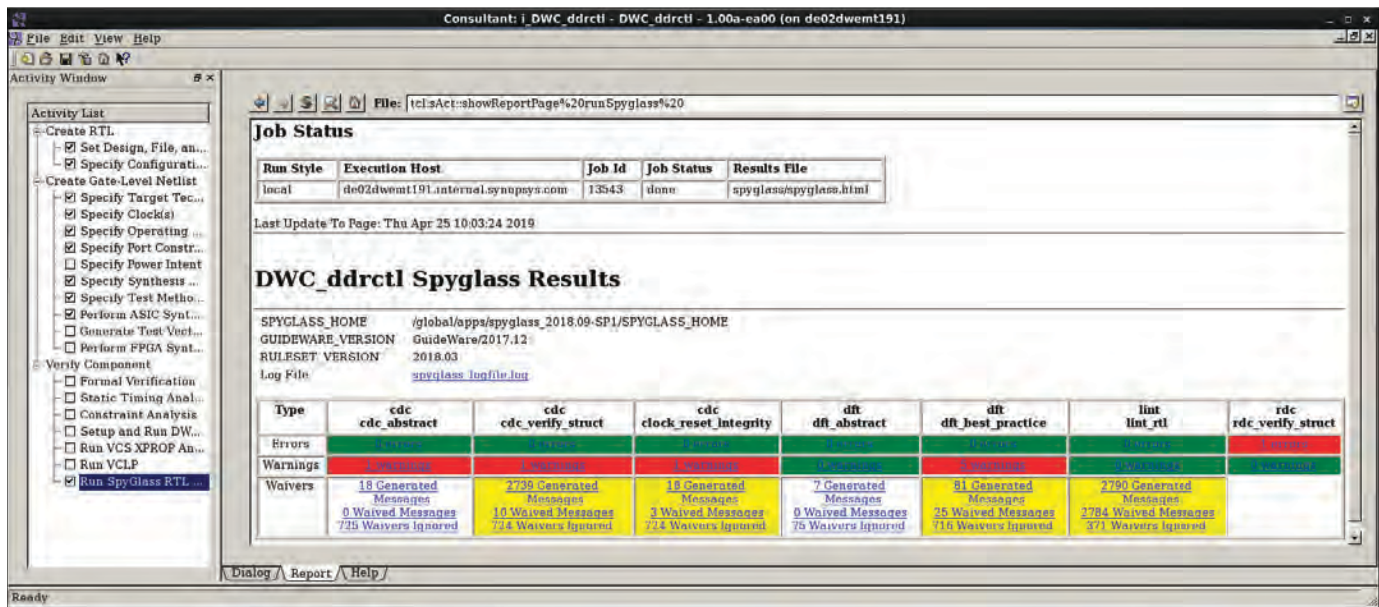
Figure 2-9 Spyglass Options in coreConsultant



2. Click **Apply** to run Spyglass.

When the Lint and/or CDC is run, the results are available in the **Report** tab. Any error is displayed with a red colored cell and any warning is displayed in a yellow colored cell as shown in [Figure 2-10](#).

Figure 2-10 Spyglass Results



2.5.1 Errors and Warnings

You may encounter errors or warnings when you run Spyglass using the coreConsultant flow because of the large number of possible configurations. For all Spyglass-related issues, open a support case to Synopsys.

Inspect the following files when the Spyglass Lint run fails with a warning:

<configured_workspace>/spyglass/work/core/consolidated_reports/<design_prefix>DWC_ddrctl_lint_1_int_rtl/moresimple_sevclass.rpt file

2.5.2 Lint

The coreConsultant tool uses the waiver files listed in Table 2-3 for Spyglass Lint.

Table 2-3 Waiver Files for Spyglass Lint

File Name	Description
<configured_workspace>/spyglass/spyglass_engineering_council_waivers.tcl	These are DWC_ddrctl design specific rule waivers. There are two sections in this waiver file. The first portion is Lint waivers. The second portion is CDC waivers. The reason for each of the waivers are included as comments in the file.
<configured_workspace>/spyglass/spyglass_engineering_council_waivers.tcl	These are rules which Synopsys waives for its IPs.

2.5.3 CDC

To understand the Spyglass CDC constraints, you must understand the reset and clock logic used in DWC_ddrctl.

For more information, see:

- The Clocks section in the Product Overview chapter of the LPDDR5/4/4X Memory Controller Data-book.
- The ‘Clocks, Resets, and Registers’ section in the LPDDR5/4/4X Memory Controller Programming Guide.

2.5.3.1 CDC Files

The coreConsultant tool uses the files listed in [Table 2-4](#) for Spyglass CDC.

Table 2-4 Waiver Files for Spyglass CDC

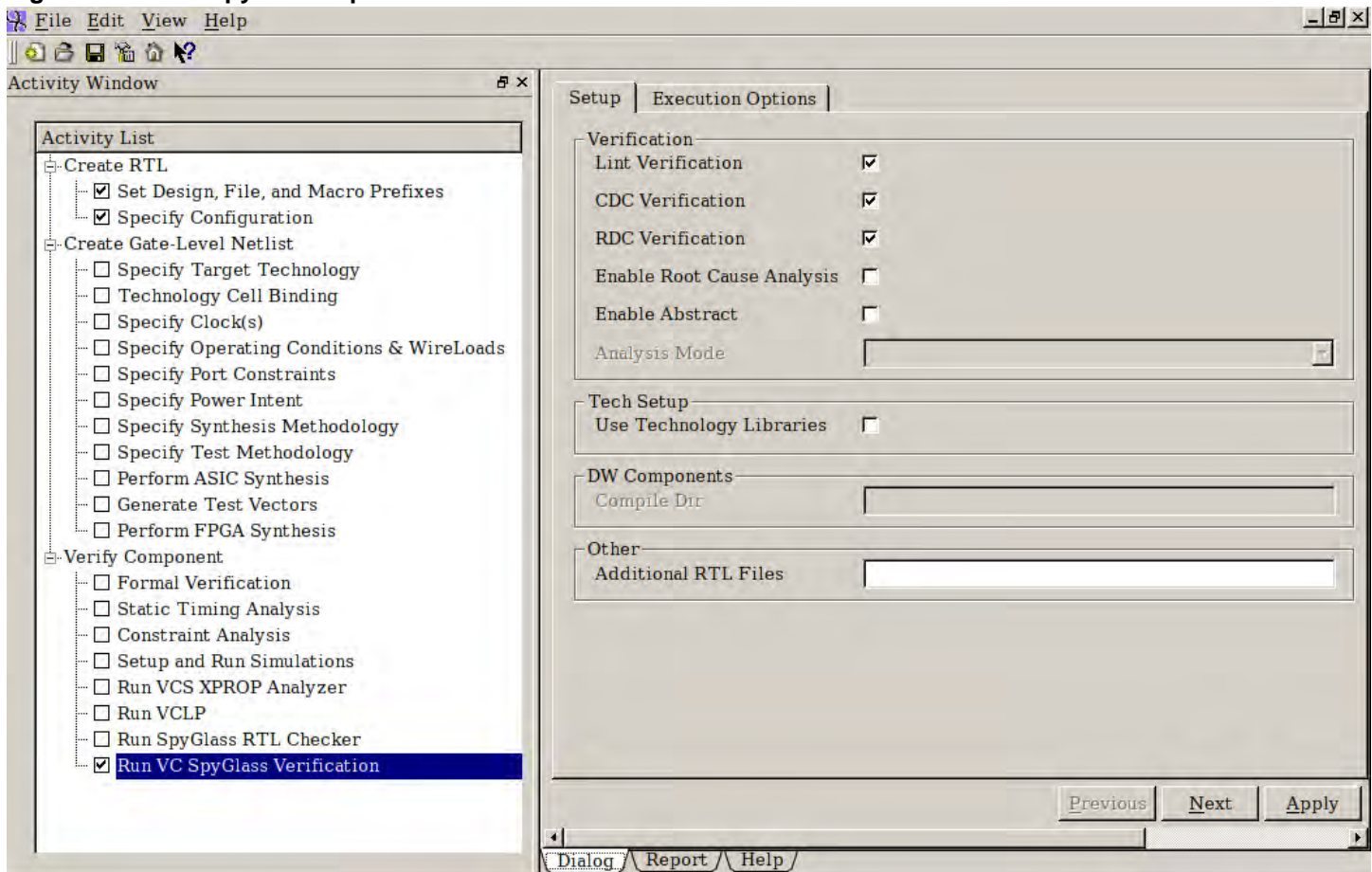
File Name	Description
<configured_workspace>/spyglass/manual.sgcd	These are the constraints pertaining to a given mode.
<configured_workspace>/spyglass/ports.sgcd	These are the list of I/O signals and their respective clocks.
<configured_workspace>/spyglass/spyglass_design_specific_waivers.swl	<p>These are DWC_ddrctl design specific rule waivers. The following sections are present in this waiver file:</p> <ul style="list-style-type: none"> ■ Lint waivers ■ CDC waivers <p>The reasons for each waiver are included as comments in the file.</p>
<configured_workspace>/spyglass/spyglass_engineering_council_waivers.tcl	These are rules which Synopsys waives for its IPs.

2.6 Running VC SpyGlass Verification

The VC SpyGlass flow in coreConsultant runs Lint, CDC and RDC verifications. Follow these steps to run VC SpyGlass Lint, CDC and RDC verification.

1. Select all the required verification options (that is, Lint, CDC or RDC).

Figure 2-11 VC SpyGlass Options in coreConsultant

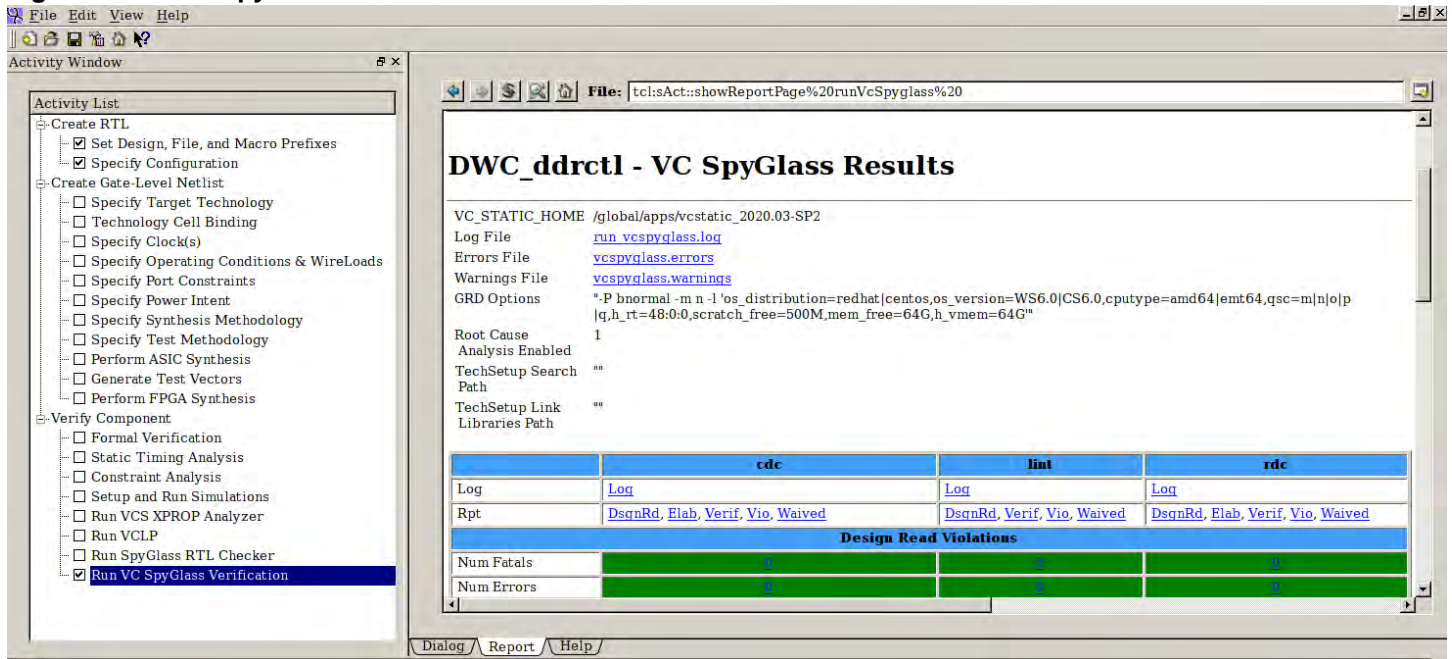


2. Click Apply to run the selected verification goal.



Note

'Enable Root Cause Analysis', 'Enable Abstract', 'Use Technology Libraries', and 'Additional RTL Files' options in VC SpyGlass Verification section are not fully tested in the coreConsultant environment.

Figure 2-12 VC SpyGlass Results

2.6.1 Errors and Warnings

Inspect the following files when the VC SpyGlass run fails with an error/warning.

Lint: <configured_workspace>/vcspyglass/lint/verification.violated.rpt

CDC: <configured_workspace>/vcspyglass/cdc/verification.violated.rpt

RDC: <configured_workspace>/vcspyglass/rdc/verification.violated.rpt

2.6.2 Waivers and Constraints Used

The following waiver and constraint files are used by VC SpyGlass for running the verification.

Waivers:

- vcspyglass/constraints/waivers.tcl

Constraints:

- vcspyglass/engineering_council/vcspyglass_bcm_constraints.tcl
- vcspyglass/constraints/DWC_ddrctl.sdc
- vcspyglass/constraints/resets.tcl
- vcspyglass/constraints/constraints.tcl

3

Simulating the Controller

This chapter provides information about the UVM-based Packaged Verification Environment (PVE) Testbench that is provided with the DWC_ddrctl IP. The PVE provides a starting point for understanding how to use DesignWare Verification IP (VIP) and the configured DWC_ddrctl controller together in your DDRCTL verification environment.

This chapter discusses the following topics:

- [“PVE Testbench”](#) on page 34
- [“Running PVE Tests Using coreConsultant”](#) on page 36
- [“Running PVE Tests from Command Line”](#) on page 38

3.1 PVE Testbench

The PVE is a testbench written using UVM 1.2 base classes, making use of latest best in class verification IP to provide a scalable test vehicle for DDR products from Synopsys. It is an environment that simulates both DWC DDR Controllers and DDR PHYs together in a memory subsystem using Synopsys SVT memory models and other verification IPs. The PVE makes use of C-reference code for both controller and PHY initialization.



Note

Third-party simulators and third-party memory models are not supported.

3.1.1 PVE Testbench Details

The information in this section is a summary; more detail is provided in HTML reference pages which are auto-extracted from PVE source code at the following location: `<ws>/doc/html/pve`.

3.1.2 PVE Testbench Files

[Table 3-1](#) provides the locations and short descriptions for PVE Testbench files.



Note

The sim directory is not fully populated until applying the Simulate Activity.

Table 3-1 PVE Testbench Files Locations and Descriptions

Directory	Description
sim/testbench	Top-level test-bench directory
sim/testbench/module	All test-bench Verilog modules including the PVE top-level wrapper file <code>ddr_uvm_pve_tb_top.sv</code>
sim/testbench/tests	All UVM test-cases
sim/testbench/env	UVM top-level environment files
sim/testbench/macros	All test environment compile macro
sim/sw_utilities/cinit	Controller Initialization C files
sim/sw_utilities/phy	PHY initialization C files

3.1.3 List of PVE Tests

The test suite provided with the PVE is for demonstration purposes only. The controller IP is fully verified internally by Synopsys R&D teams. The tests provided are just a subset to help you in understanding certain features and as a guide for integration.

The list of tests can be found in HTML pages packaged with the controller IP.

3.1.4 PVE Test Case Details

Test case details can be found in HTML pages packaged with the controller (<ws>/doc/html/pve).

3.2 Running PVE Tests Using coreConsultant

To run the PVE in the coreConsultant GUI, select the *Verify Component/Setup and Run DWC_ddrctl Simulation*.

3.2.1 Running the Simulation

You can select the PHY type, memory model, and bus width, tests in the 'Testbench' options panel. Click Apply to start simulation jobs. A SIMV executable is compiled for each protocol; each test runs in a separate test directory. The directory name will be suffixed by the protocol in operation automatically.

3.2.2 Test Cases

Test case details can be found in HTML reference pages packaged with the controller.

3.2.3 Important Simulation Files

The log files produced by simulation are post-processed by runtest (PERL script) and a pass/fail status is determined. Some important files in the test directory are as follows:

Table 3-2 Important Simulation Files

Filenames	Description
compile.log	VCS compilation log file
test.log	VCS Simulation log file
dwc_cinit.log	CINIT debug log file
phyinit.log	PHYINIT output log file
test.error	If the test fails this will contain the first error
test.startsim	The main invocation script to begin the simulation
test.simcommand	VCS compile options
test.plusarg	VCS simulation runtime arguments, check this files for optional debug switches. For example +VIP_ENABLE_APB_TRACING=1 will enable APB log file containing CINIT and PHYINIT programming commands

3.2.4 Checking Simulation Status and Results

To check simulation status and results, click the Report tab; coreConsultant displays a dialog that indicates:

- Your selected Run Style (local, lsf, grd, or remote)
- The full path to the HTML file that contains your simulation results
- The name of the host on which the simulation is running
- The process ID (Job Id) of the simulation
- The status of the simulation job (running or done)

The Results dialog also enables you to kill the simulation (Kill Job) and to refresh the status display in the results dialog (refresh status).

The Results information includes:

- Simulation execution messages
- A pass or fail result

The `runtest.html` file in `workspace/sim` includes all of the results of the simulations. When multiple different tests are selected, this file contains the concatenated results of all these simulations (tests). Individually, each test's unique output can be found in the `workspace/sim/test_testname/test.log` file. The `runtest.html` file, presents the information in the following categories:

- Verification Activity Log: A log of the simulation activity
- Testbench Preparation: A list of `runtest` options that were executed during the simulation
- Simulation Execution: Provides the output of each simulation (test); this information is also saved to `test.log` in `workspace/sim/test_testname`
- Simulation Results: Includes the time the simulation completed, the path to `test.log`, how many errors were encountered and the overall result of each simulation

The `workspace/sim/test_testname` directory includes the various log files that are unique to a given test/simulation. The following output files are included at this level:

- `test.log`: Output of the testbench and includes specifics about the simulators used, transaction monitor and self checker messages, including error messages
- `test.result`: Summary report indicating whether the test PASSED or FAILED and when it failed, details about the failure (such as timestamp, expected response, actual response)
- `uvm_ddr_pve_tb_top.fsdb` or `uvm_ddr_pve_tb_top.vpd`: Waveform dump file when enabled through the simulation options
- `dwc_ddrctl_cinit.log.gz`: Log file of all APB accesses to the DDRCTL and all configuration register accesses to the PUB which have occurred during the simulation
- `dwc_ddrctl_apb.log.gz`: The register settings logged in this file can be used as a guide to how to program the DDRCTL and PUB registers. While the registers are correct for the selected configuration and speed grade, they are not guaranteed to give optimum utilization and latency performance. To achieve optimum performance requires an analysis of the configuration and traffic patterns.

3.3 Running PVE Tests from Command Line

To run PVE test cases from the Linux command line, the workspace must be fully populated first. This means that the RTL and testbench files are written out to a workspace ready for simulation. This can be done using the GUI or by running a coreConsultant batch file. You can prevent any simulations from running in the GUI by setting "DryRun=1" simulate activity parameter.

When the workspace directory is populated with a configured design, each test can be run from the command line by using the following simple steps:

```
% cd <path_to_cC_workspace>/sim/<test_name>
% ./test.startsim -recompile -fsdb
```

**Note**

Your environment must be configured correctly in order for these commands to be successful.

4

Performing Synthesis and Post-Synthesis

This chapter provides information about the UVM-based Packaged Verification Environment (PVE) Testbench that is provided with the LPDDR5/4/4X Memory Controller. The PVE provides a starting point for understanding how to use DesignWare Verification IP (VIP) and the configured DWC_ddrctl controller together in your DDRCTL verification environment.

This chapter discusses the following topics:

- Performing Synthesis Activities
 - [“Synthesizing the Controller”](#) on page 40
 - [“Inserting Design for Test”](#) on page 52
- Performing Post-Synthesis Activities
 - [“Running Automatic Test Pattern Generation”](#) on page 53
 - [“Running Static Timing Analysis”](#) on page 57
 - [“Performing Formal Verification”](#) on page 61

4.1 Synthesizing the Controller

The coreConsultant tool is your interface to Synopsys synthesis tools for ASIC synthesis of the controller. If you want to access the synthesis scripts for exporting to your chip database, then refer to “[Exporting Controller to Your Chip Design Database](#)” on page 64.

During the **Create Gate-Level Netlist** activity in coreConsultant, you can also perform Physical Synthesis, and Design For Test. Instructions for these tasks are provided in this section.

The topics in this section are as follows:

- “[Performing Synthesis](#)” on page 40
- “[Checking Synthesis Results](#)” on page 50
- “[Running Synthesis from a Unix Shell](#)” on page 51
- “[Troubleshooting: Common Problems in this Step](#)” on page 51



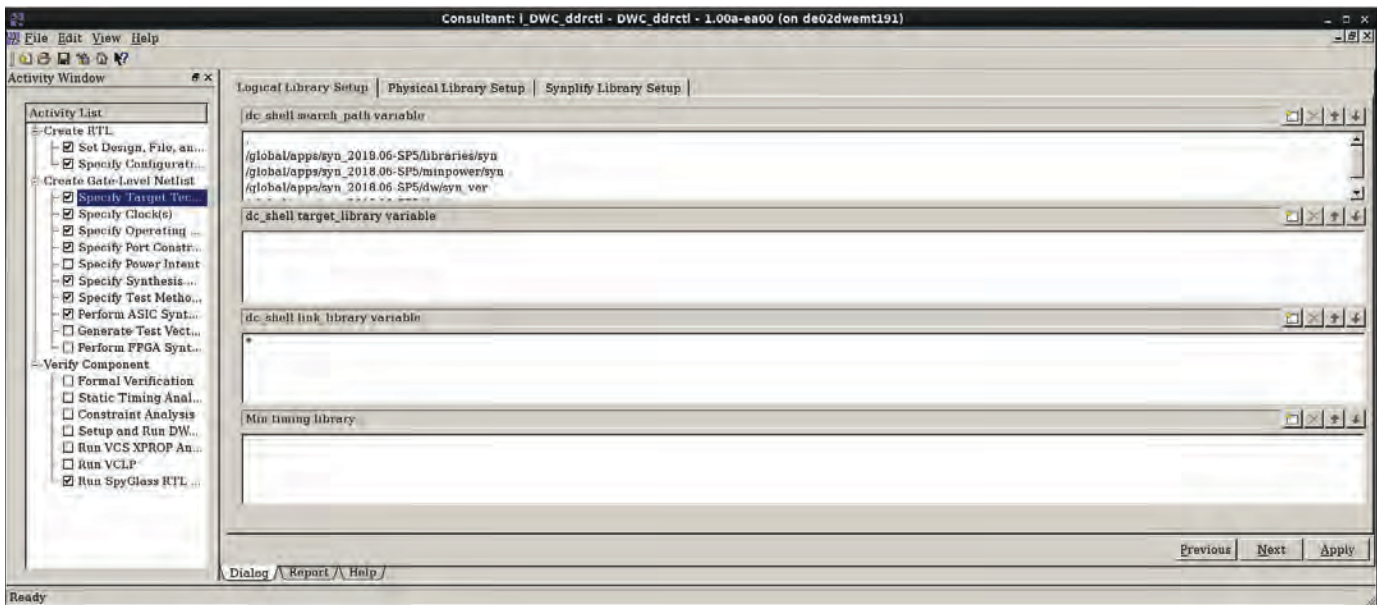
Hint

Before you begin the **Create Gate-Level Netlist** activity, check that you have the latest tool versions installed and your environment variables set up correctly as detailed in the LPDDR5/4/4X Memory Controller Installation Guide.

To check your tools, click the **Help > Check Environment** menu item. Correct any reported errors by modifying your Unix environment setup or through the **Edit > Tool Installation Roots** menu.

4.1.1 Performing Synthesis

Configure the controller (see “[Configuring the Controller](#)” on page 17) before starting this task. To synthesize your DWC_ddrctl controller and create a gate-level netlist, select the **Create Gate-Level Netlist** activity as shown in [Figure 4-1](#) on page 41.

Figure 4-1 Create Gate-Level Netlist Activity

1. Specify Target Technology

When you click on **Create Gate-Level Netlist**, the first step in this activity is **Specify Target Technology**. A target library must be specified, otherwise, errors occur in coreConsultant.

□ ASIC Synthesis

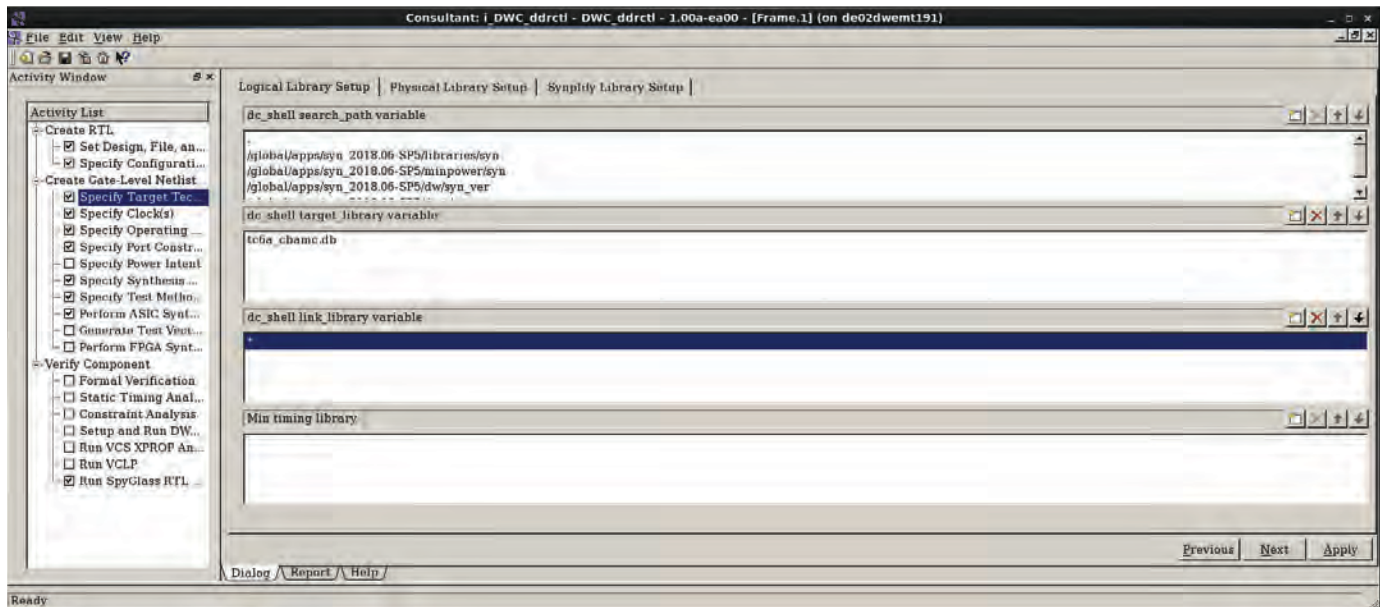
For ASIC synthesis, use a target technology of .18 microns or less. To add variables and libraries to the dialog fields, click the folder icon (as shown in [Figure 4-2](#) on page 42) and use the navigation tool. The default values for the search_path and link_library do not normally need to be changed.



Note

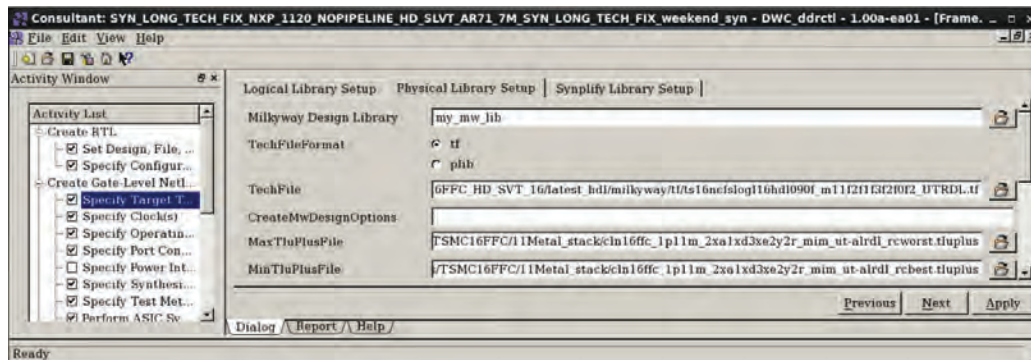
link_library should always contain '*' and all the specified files.

If you intend to use the Fusion_Compiler_Reference_Methodology as Strategy in the "Perform ASIC Synthesis" step, libraries that are compatible with FC need to be specified in this step. Otherwise, the option to select Fusion_Compiler_Reference_Methodology will not be available.

Figure 4-2 Specify Target Technology – Logical Library Setup

■ Physical Synthesis

If you are running Physical Synthesis, then you must specify the physical library under the **Physical Library Setup** tab (Figure 4-3 on page 42). You specify a Milkyway Reference Library and Milkyway Technology File when you enable Physical Synthesis during the Specify Target Technology activity. This may be .tf or .plib file. You may also provide optional files (TLUPlus), which allow the tools to model Deep Submicron effects with greater accuracy.

Figure 4-3 Specify Target Technology – Physical Library Setup

2. Specify Clock(s)

In **Create Gate-Level Netlist -> Specify Clock(s)** activity, the default Cycle Time for the input clocks is provided in Table 4-1. These times match the DWC_ddrctl default clock-rate configuration (400 MHz).

Table 4-1 Default Cycle Time for DWC_ddrctl

Clock	Cycle Time
core_ddrc_core_clk	2.5

- ❑ Design for Test (DFT)

These clocks are also selected as 'Test Clock'.

- ❑ ASIC/Physical

The default values for other clock-related synthesis attributes provide acceptable synthesis results in most libraries.

3. Specify Operating Conditions and Wire Loads

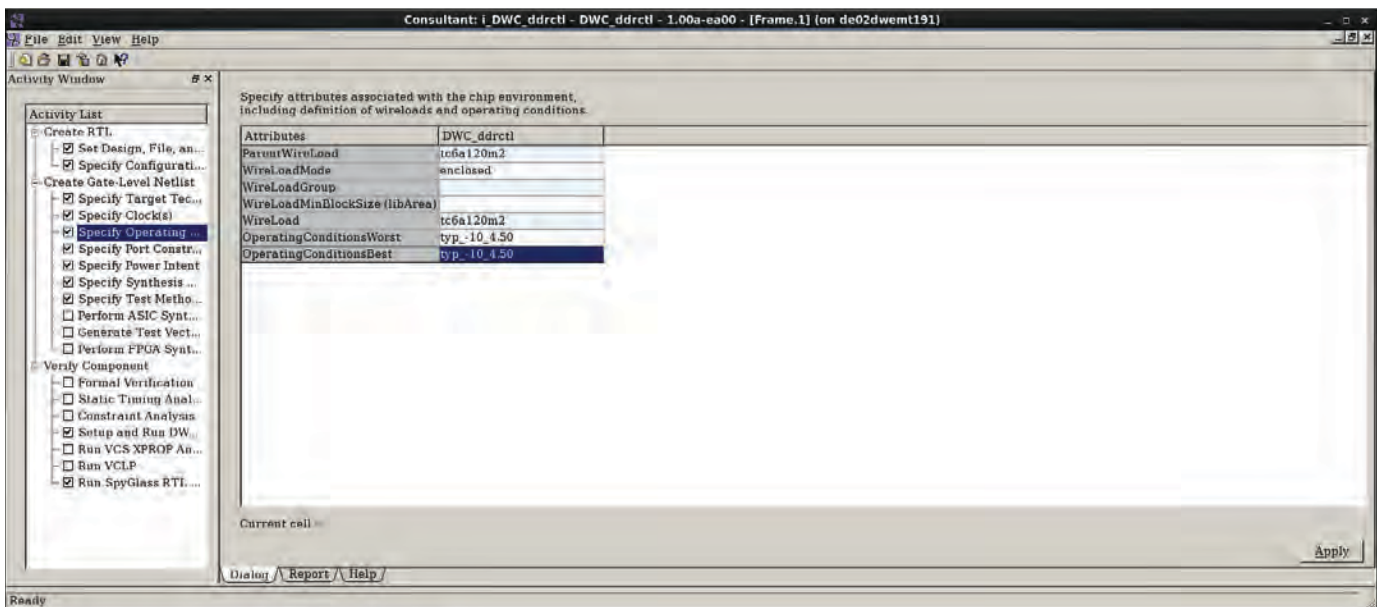
a. Select **Create Gate-Level Netlist -> Specify Operating Conditions and Wire Loads** activity.

b. Set the various "WireLoad" attributes, unless you run the Physical Synthesis.

For detailed help on any of the options, right-click and select **Help on this Row**.

c. Click **Apply** and look at the report, which gives the operating conditions and wireload information.

Figure 4-4 Specify Operating Conditions and Wireloads Window



4. Specify Port Constraints

In **Create Gate-Level Netlist -> Specify Port Constraints** task, port input and output delay constraints are specified relative to a virtual clock (clk_ideal), which has the same CycleTime as the real clocks specified in [Table 4-1](#) on page 42.

The default input and output delay constraint values are specified in nanoseconds and are configuration-dependent. The default set of these synthesis attributes provide acceptable synthesis results in most libraries.

Figure 4-5 Specify Port Constraints - Input Delay

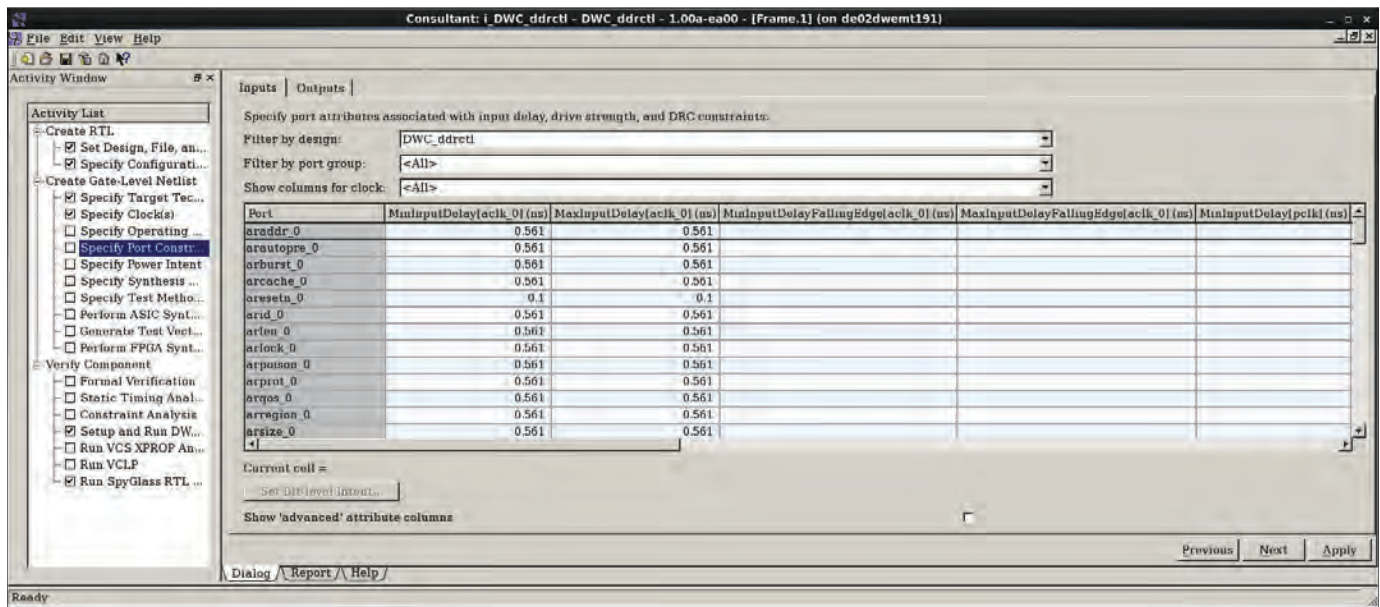
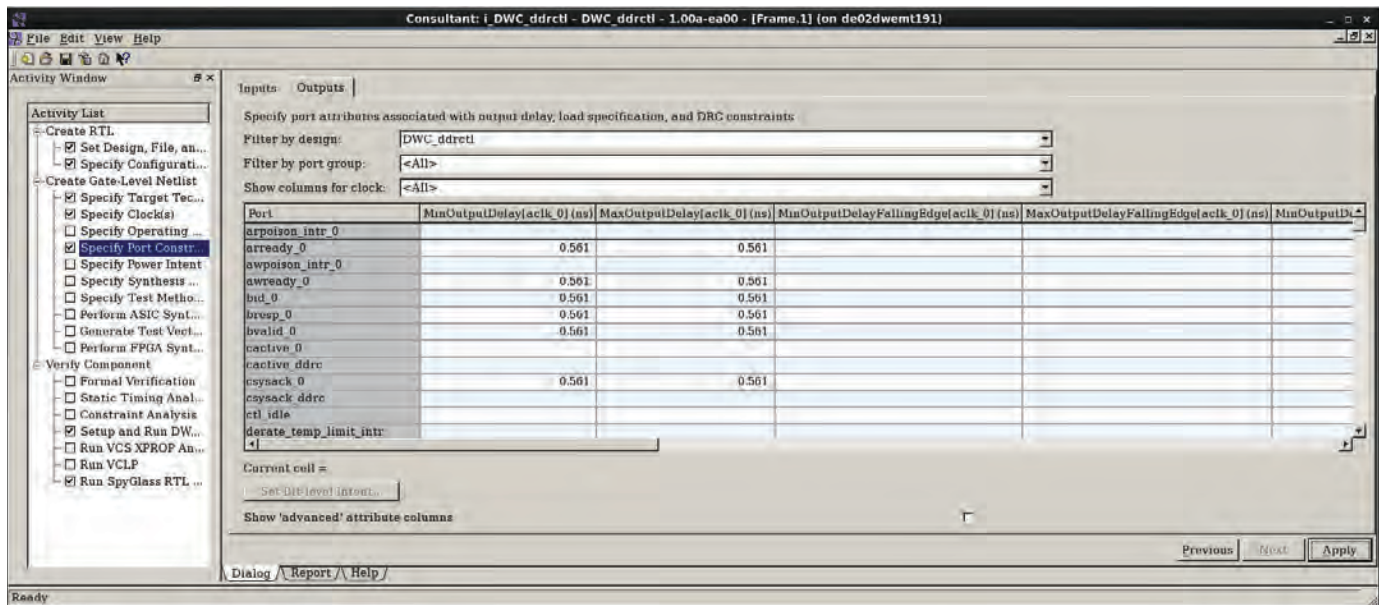


Figure 4-6 Specify Port Constraints - Output Delay



5. Specify Synthesis Methodology

□ ASIC

In **Create Gate-Level Netlist -> Specify Synthesis Methodology** activity, review the synthesis strategy attributes. These attributes are set by the controller developer and can be optionally modified by you. The default setting of these attributes provide acceptable synthesis results in most libraries.

Physical Synthesis

If you are running Physical Synthesis, then click on the **Physical Synthesis** tab (Figure 4-12 on page 49) and enter information in the related fields. The physical synthesis flow uses the Synopsys DC-Topographical engine, which is part of Design Compiler (dc_shell).

Figure 4-7 Specify Synthesis Methodology - Common Synthesis Strategy Attributes

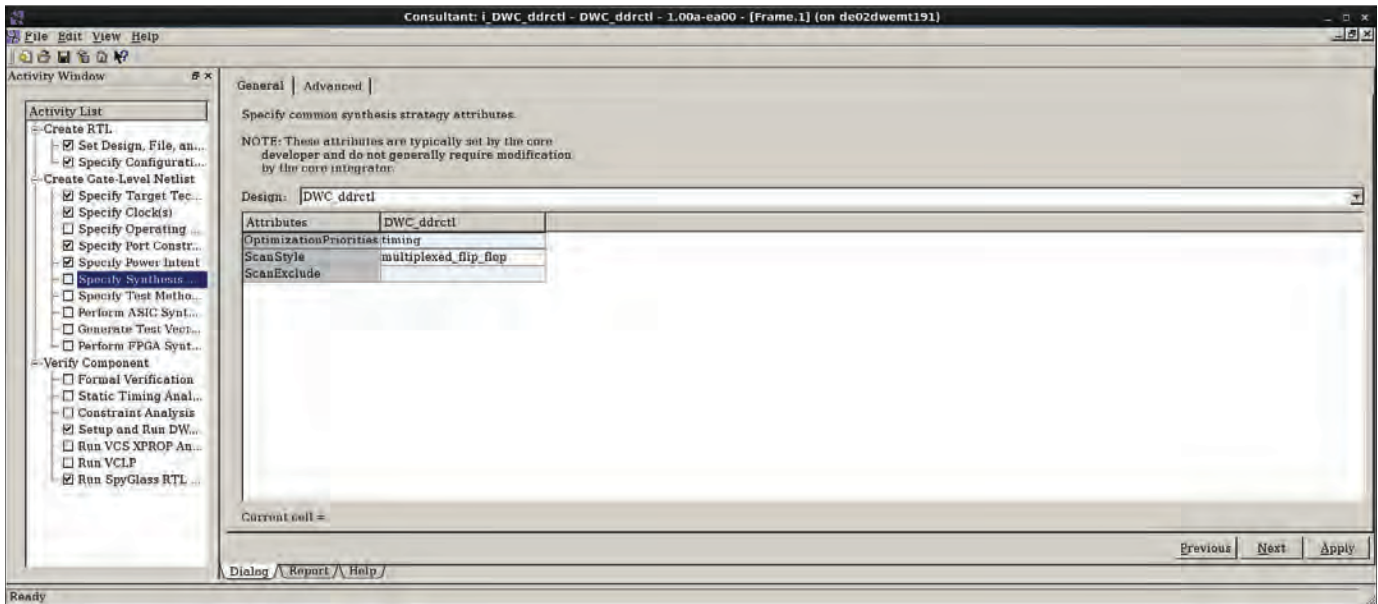
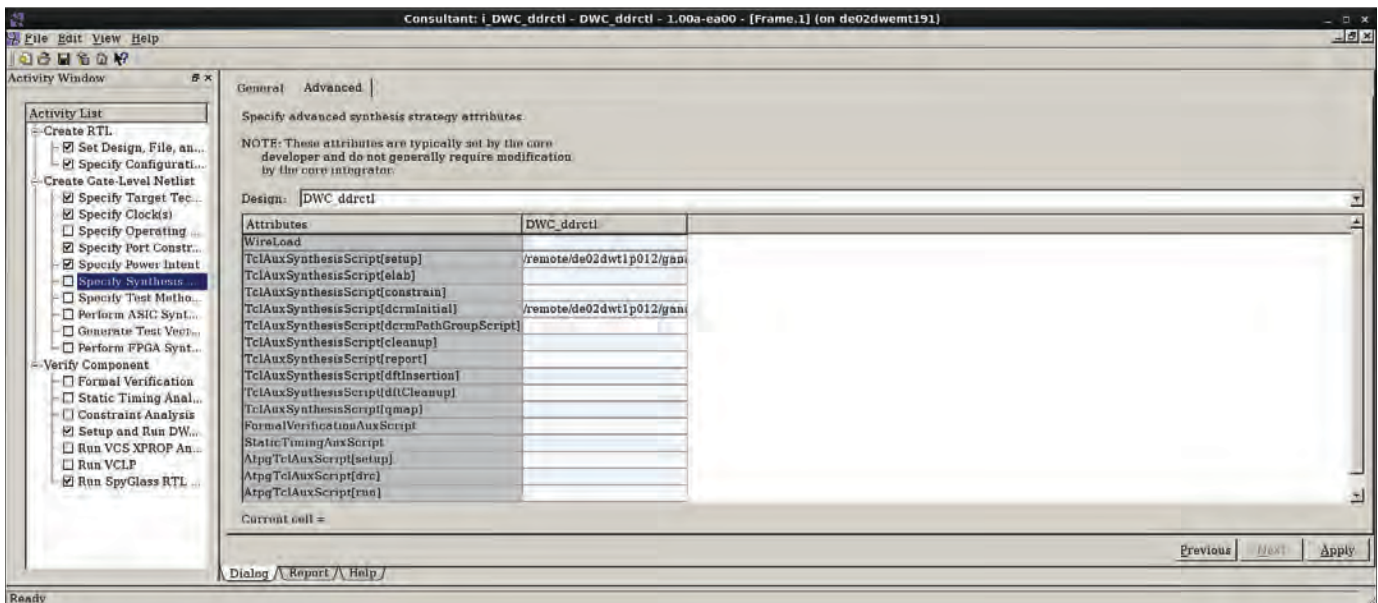


Figure 4-8 Specify Synthesis Methodology - Advanced Synthesis Strategy Attributes



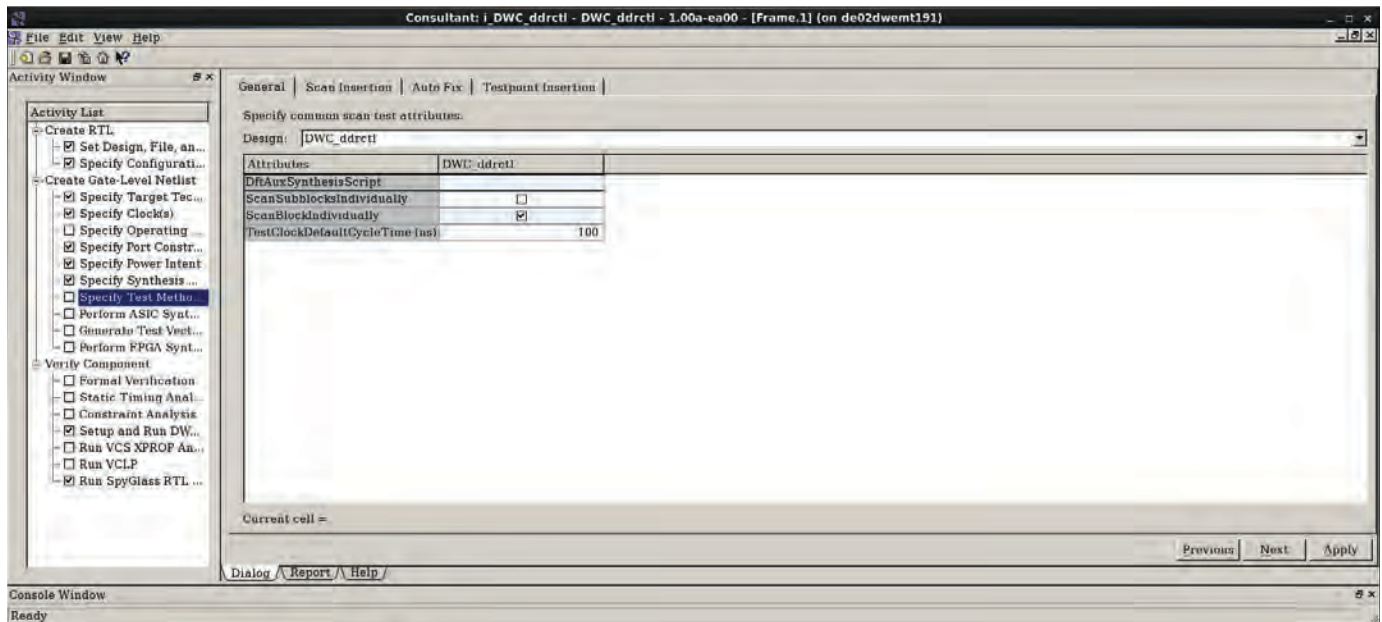
6. Specify Test Methodology

In **Create Gate-Level Netlist -> Specify Test Methodology** activity, these attributes are set by the controller developer and can be optionally modified by you.

Design for Test

Modify options according to your chip's DFT scheme. For more information on any of these options, right-click and select **Help on this Row**.

Figure 4-9 Specify Test Methodology



7. Synthesize

The next step in the **Create Gate-Level Netlist** is to Perform ASIC Synthesis.

□ ASIC Synthesis

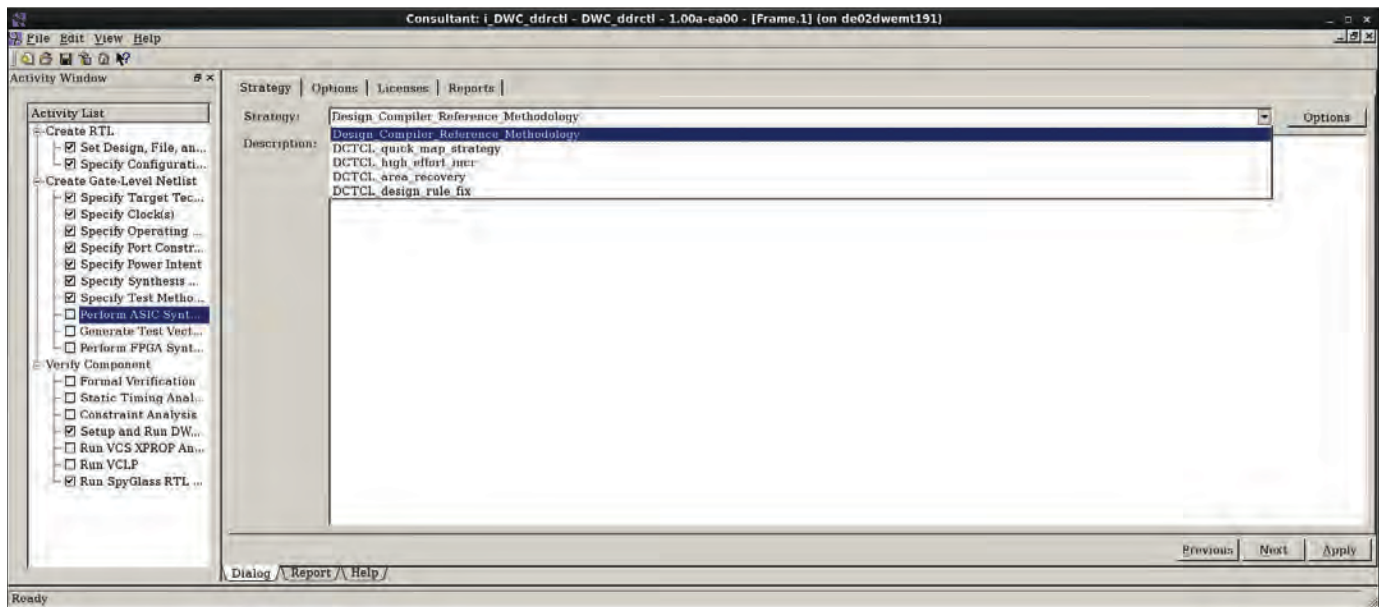
In the **Perform ASIC Synthesis activity -> Strategy** tab, you can select the synthesis strategy/flow as shown in [Figure 4-10](#) on page 47. The default flow is "Design Compiler Reference Methodology". A detailed explanation of each strategy is provided in the GUI.



Hint

Another strategy that is of interest is the DCTCL_one_pass_compile_ultra, which uses a simple TCL script based on the Synopsys Design Compiler Reference Methodology at <https://solvnet.synopsys.com/retrieve/021023.html>. This synthesis strategy is quicker and does not generate intermediate stages. Therefore, it is much easier to trace the steps in the flow.

When you select a strategy, you can click on the **Options** button next to the selected strategy drop-down list (highlighted in [Figure 4-10](#) on page 47) to specify other options for that particular strategy. For instance, [Figure 4-10](#) on page 47 provides the options for the DCTCL_opto_strategy.

Figure 4-10 Perform ASIC Synthesis Activity -> Strategy Tab

For more information, right-click on any text field in the dialog box or refer to the coreConsultant User Guide (also see [“Help Information”](#) on page 75).

Figure 4-11 Perform ASIC Synthesis Activity -> Design_Compiler_Reference_Methodology -> Options Button

The screenshot shows the 'Strategy Parameters Design_Compiler_Reference_Methodology (on de02dwemt191)' dialog box. The 'Basic' tab is active, displaying the following settings:

- QoR effort:** medium
- Optimization flow:** Disabled
- EnableOptimizeNetlist:** ☒
- OptimizeNetlistArgs:** -area
- Compile Ultra:**
 - Enable Adaptive Retiming:** ☐
 - Additional Initial Options:** (empty text field)
 - Additional Incremental Options:** (empty text field)
- SDC Generation:**
 - Options for write_sdc:** (empty text field)
- SDF Generation:**
 - Write SDF after synthesis:** ☐
- ICC2 Write Files:**
 - Write ICC2 files after synthesis:** ☐
- Design for Test:**
 - Test Ready Compile:** ☒
 - Insert Dft:** ☒

At the bottom of the dialog are buttons for **Previous**, **Next**, **OK**, **Cancel**, **Default**, and **Help**.



Note The 'Generate Test Vectors Activity' fails if, 'Insert Dft' option is not selected.

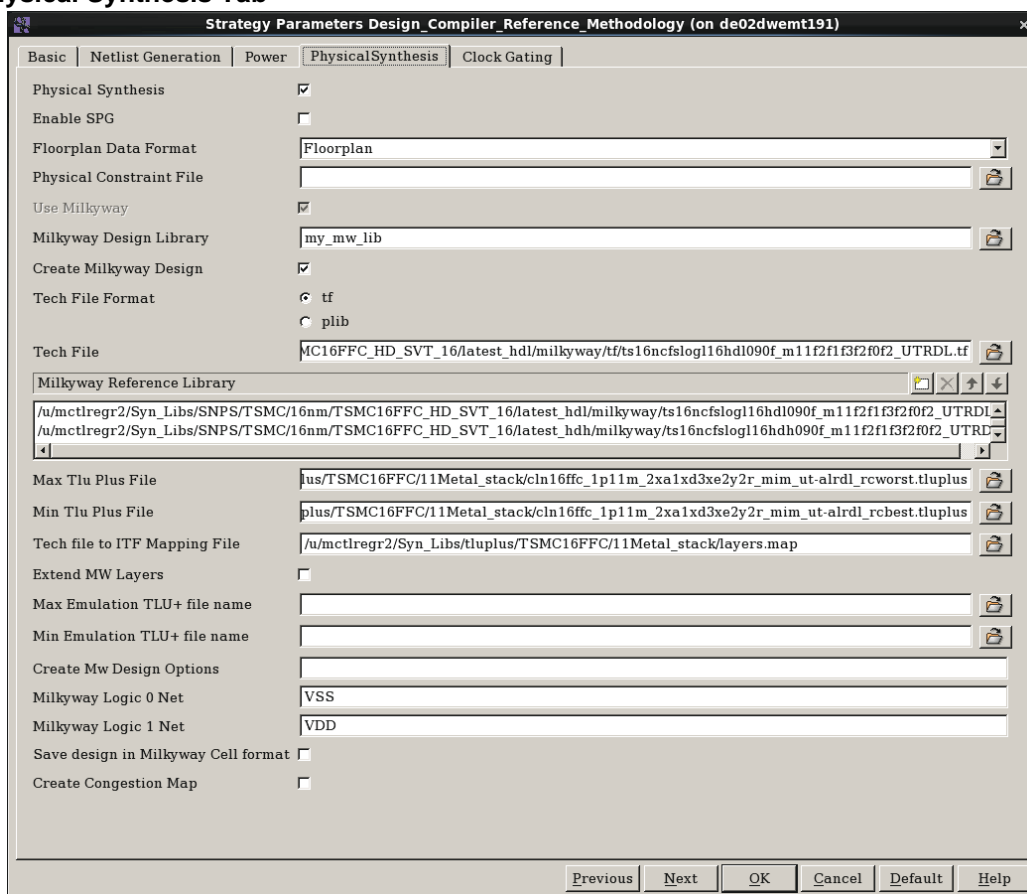
□ Physical Synthesis

- i. In the **Create Gate-Level Netlist -> Perform ASIC Synthesis** activity, select a strategy from the drop-down box or accept the default value (Design_Compiler_Reference_Methodology).
- ii. Click the Options button and then select the **Physical Synthesis** tab (see [Figure 4-12](#) on page 49).
- iii. Select **Physical Synthesis** check box and enter information in the related fields.

Setting this parameter causes the physical synthesis placement engine to be used during optimization. Net loads and delays are estimated during placement. This flow may be used with either a physical library in the .ddc format or Milkyway reference library. Note that no placement data is saved with the ddc or Milkyway database.

- iv. Click OK.

Figure 4-12 Perform ASIC Synthesis Activity -> Design_Compiler_Reference_Methodology -> Options Button -> Physical Synthesis Tab

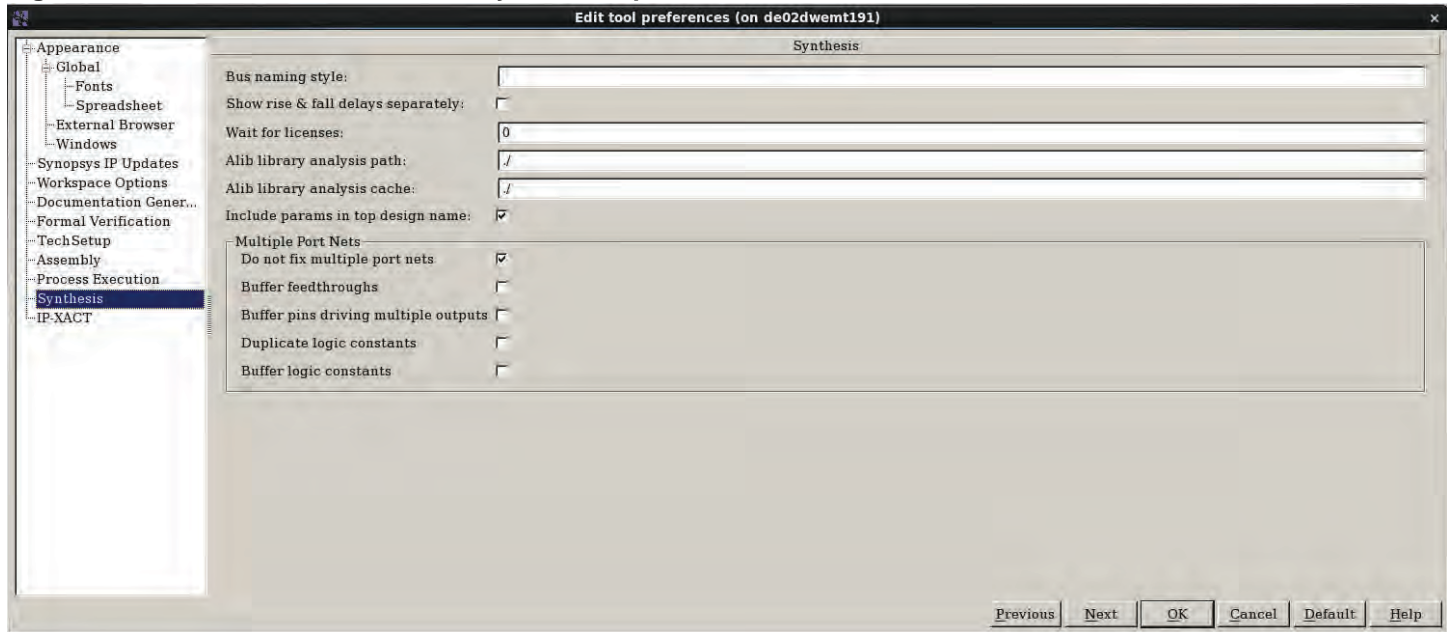


4.1.2 Setting Additional Synthesis Options

You can access more synthesis options in the Edit Tool Preferences dialog box accessed from the coreConsultant Edit Menu.

To set these additional options:

- Select **Edit -> Preferences**. The Edit Tool Preference dialog box is displayed.
- Click on **Synthesis** to display the list of options, as shown in [Figure 4-13](#) on page 50.

Figure 4-13 Edit Tool Preferences – Synthesis Options

4.1.3 Checking Synthesis Results

After you have run synthesis, you can check the results by clicking the **Report** tab. All the synthesis results and log files are created under your <workspace>/syn directory. The final symbolic link points to the current synthesis stage directory. The final log file is written to <workspace>/syn/run.log.

Except in the case of the DCTCL_one_pass_compile_ultra strategy, your final netlist and report directories (initial, incr1, or incr2) depend on the QoR effort that you chose for your synthesis (default is Medium) or whether you chose to insert DFT:

- Low effort – initial
- Medium effort – incr1
- High effort – incr2

The QoR effort is selected through the **Perform ASIC Synthesis -> Options¹ -> Basic** tab. There are also options here to:

- Generate reports for all stages
- Generate netlist for all stages

The synthesis process runs in stages or steps, which are normally initial, incr1, or incr2, depending on the value of the QoR Effort synthesis strategy parameter.



- In the <workspace>/syn directory, run.scr, Makefile, and final are symbolic links to the current synthesis stage files.
- These links are also used and set to different locations during ATPG and Formal Verification.

Table 4-2 shows the other files that are generated after synthesis.

1. BUTTON as circled in Figure 4-10 on page 47 and not TAB.

Table 4-2 Synthesis Output Files

Synthesis Type	Files	Purpose
ASIC	./syn/final/db/DWC_ddrctl.ddc	Synopsys database files (gate level) that can be read into dc_shell for further synthesis, if desired.
	./syn/final/db/DWC_ddrctl.v	Gate-level netlist that is mapped to technology libraries that you specify.
	./syn/final/report/*.*	Synthesis report files.
Physical	./syn/final/db/DWC_ddrctl.ddc	Synopsys database files (gate level) that can be read into dc_shell for further synthesis, if desired.
DFT	./syn/dft/*.*	Results of DFT insertion.

4.1.4 Running Synthesis from a Unix Shell

To run ASIC, FPGA, or Physical synthesis from Unix shell (before running from coreConsultant):

1. Complete all the setup steps as outlined in “[Synthesizing the Controller](#)” on page 40.
2. Select **Perform ASIC Synthesis -> Options -> Execution Options -> Generate scripts only?**
3. Click **Apply**.
4. Enter the following commands:

```
% cd <workspace>/syn
% run.scr
```

To run synthesis again from a Unix shell (after having run it from coreConsultant), enter the following commands:

```
% cd <workspace>/syn
% run.scr
```



Caution

- In the <workspace>/syn directory, `run.scr`, `Makefile`, and `final` are symbolic links to the current synthesis stage files.
- These links are also used/set to different locations during ATPG, Static Timing Analysis and Formal Verification. Therefore, ensure they are pointing to the correct location if you are attempting to run Synthesis from the Unix shell AFTER having just run ATPG, Static Timing Analysis or Formal Verification.

4.1.5 Troubleshooting: Common Problems in this Step

- coreConsultant cannot invoke the synthesis tool after a crash. The activity has detected the presence of a synthesis “guard” file from the previous run. Usually this indicates that a synthesis run is in progress in this workspace. The guard file is removed when the synthesis run is completed. This activity cannot continue since files written by this activity can adversely affect the outcome of the synthesis run. Check the Console Pane for any messages which called out for the guard file name.
- A user-defined strategy file does not exist. Go the Console Pane and scroll to the message complaining about the file. Create that file.
- Specified cell name could not be found in the loaded technology libraries. Review the technology and link libraries being used and select a cell that exists within one of the libraries.

4.2 Inserting Design for Test

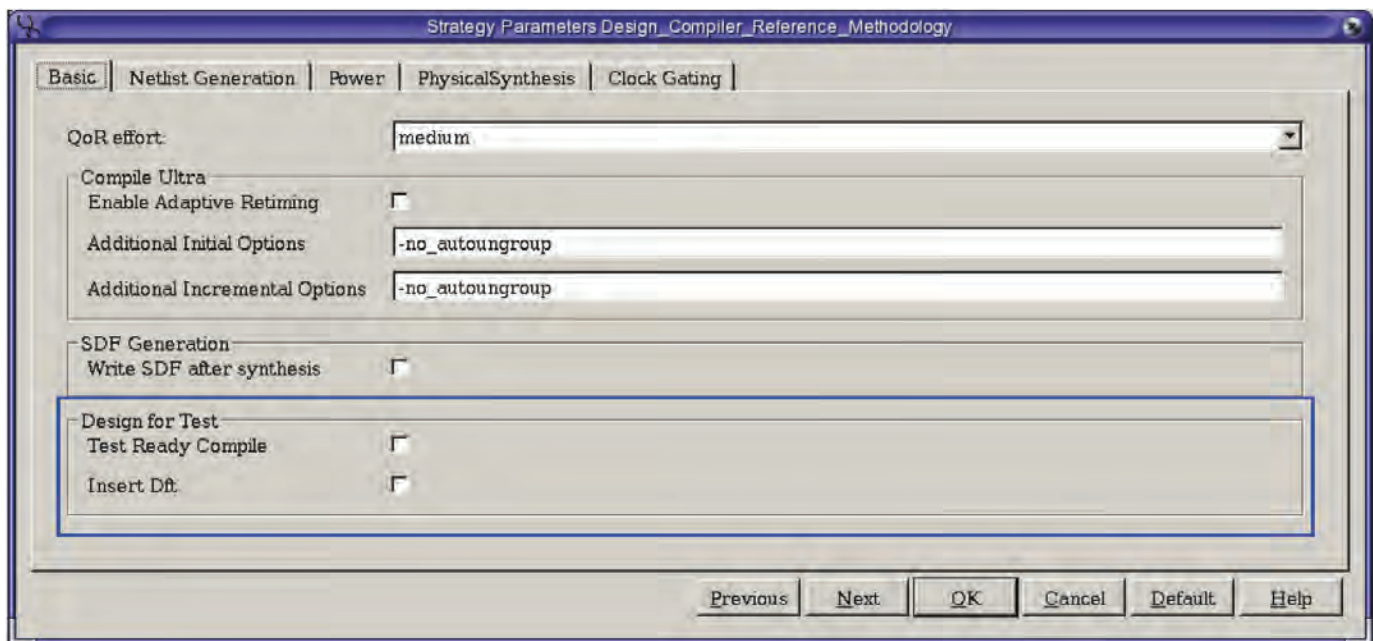
This section shows you how to use the coreConsultant tool to insert DFT during synthesis of the DDRCTL. This activity is not applicable to FPGA implementations.

You perform Design For Test (DFT) insertion during synthesis.

To insert DFT, perform the following steps:

- Check that the controller input clocks are selected as “Test Clock” in the “Specify Clocks” window. This is normally selected by default. See “Specify Clocks”.
- In the “Specify Test Methodology” window, modify the options according to your chip DFT scheme. For detailed help on any of the options, right-click and select “Help on this Row”.
- DFT insertion is controlled in the “Design For Test” tab that is displayed by clicking the “Options” button (NOT the Options tab) in the “Synthesis” window. You must select one of these two options:
 - “Test Ready Compile”: Design Compiler uses scan flops in synthesis, but it does not insert and route the chains. The scan option is added to the initial compile call.
 - “Insert DFT”: Design Compiler completes DFT insertion, that is, it performs synthesis with scan flops and scan chain insertion.

Figure 4-14 Design For Test Tab



The DFT results are saved in the *workspace/syn/dft/* directory.

4.3 Running Automatic Test Pattern Generation

This section shows you how to run automatic test pattern generation (ATPG) on the DWC_ddrctl using Synopsys TetraMax tool. The topics in this section are as follows:

- [Running ATPG using TetraMax](#)

4.3.1 Running ATPG using TetraMax

ATPG is performed using TetraMax.



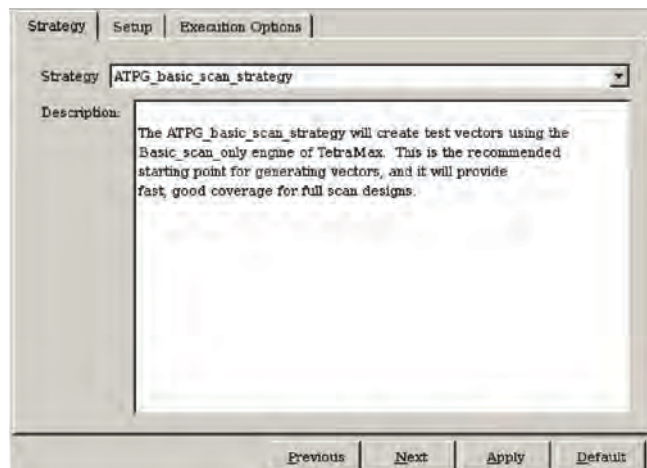
Attention

- ATPG is an activity which takes the output of synthesis as input.
- Synthesis must be completed before a meaningful setup can exist in the ATPG window.

After you have synthesized the controller with the Insert Dft option, complete the following procedure:

1. Select **Create Gate-Level Netlist -> Generate Test Vectors** activity. The following dialog box (see [Figure 4-15](#)) is displayed.

Figure 4-15 Create Gate-Level Netlist -> Generate Test Vectors Activity



2. Select the Setup tab and complete the following fields.
 - **Input stage:** Specify the last synthesis stage from the pull-down menu.
 - **Output stage:** Set to any name that you want to use identify the output files that are created in your workspace directory. For example, atpg.
 - **Test Libraries:** Specify the library Verilog file names.
 These are the Verilog simulation models for the target libraries used during synthesis. They provide the description of the logical functionality of all the cells in the standard cell library. They are needed for the ATPG tool to comprehend the logic function implemented by each standard cell instantiated.
 - **Test Pattern Format:** Specify your test format.
3. Click **Apply** when you have finished specifying your options.

4.3.1.1 Checking ATPG Output Files

The ATPG results are saved in the `<workspace>/syn/<Output stage>` directory.

The following files in `<workspace>/syn/<Output stage>` may be inspected to help you understand the ATPG flow.

- `script/DWC_ddrctl.tcl`
- `Makefile`

4.3.1.2 Running ATPG from the Unix shell

To run ATPG from a Unix shell (before running it from coreConsultant), you must first complete all the setup steps as outlined previously and then select **Generate Test Vectors -> Execution Options -> Generate scripts only?** before clicking **Apply**. Then enter the following commands:

```
% cd <workspace>/syn
% run.scr
```

To re-run ATPG from a Unix shell (after having run it from coreConsultant), enter the following commands:

```
% cd <workspace>/syn
% run.scr
```



Caution

- In the `<workspace>/syn` directory, `run.scr`, `Makefile`, and `final` are symbolic links to the ATPG input stage files, which are identified by the label `<'Output stage'>`
- These links are also used/set to different locations during Synthesis and Formal Verification. Therefore, ensure they are pointing to the correct location if you are attempting to run ATPG from the Unix shell AFTER having just run Synthesis or Formal Verification.

For more information about running ATPG, see coreConsultant User Guide (also see [“Help Information”](#) on page 75).

You have to do this only once because these settings are saved in your `~/ .synopsys_rt_prefs.tcl` file. You must do this before you create the reports. If you have previously created reports with old settings, delete the workspace and start again.

4.4 Running Low Power Static Verification Using VC LP

Low power static verification can be performed using Synopsys VC Low Power (VC LP) tool. You can run VC LP in the coreConsultant GUI (see [Figure 4-16](#)).

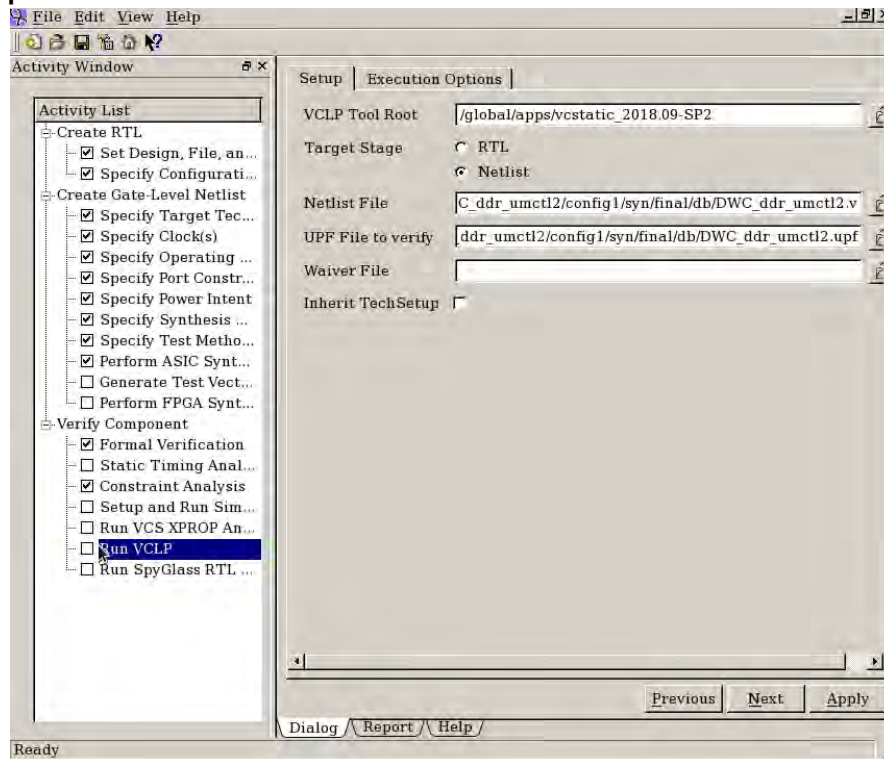
The VC LP flow in coreConsultant runs the following Low Power Checks either on the RTL or netlist:

- Power Intent Consistency Checks
- Signal Corruption Checks
- Structural Checks
- Power and Ground (PG)
- Checks and Functional Checks

For details on each of the checks, refer to the VC Low Power User Guide, which is available at the following location:

https://solvet.synopsys.com/dow_retrieve/latest/ni/vc_static.html

Figure 4-16 VC LP Options in coreConsultant



To run VC LP, complete the following steps:

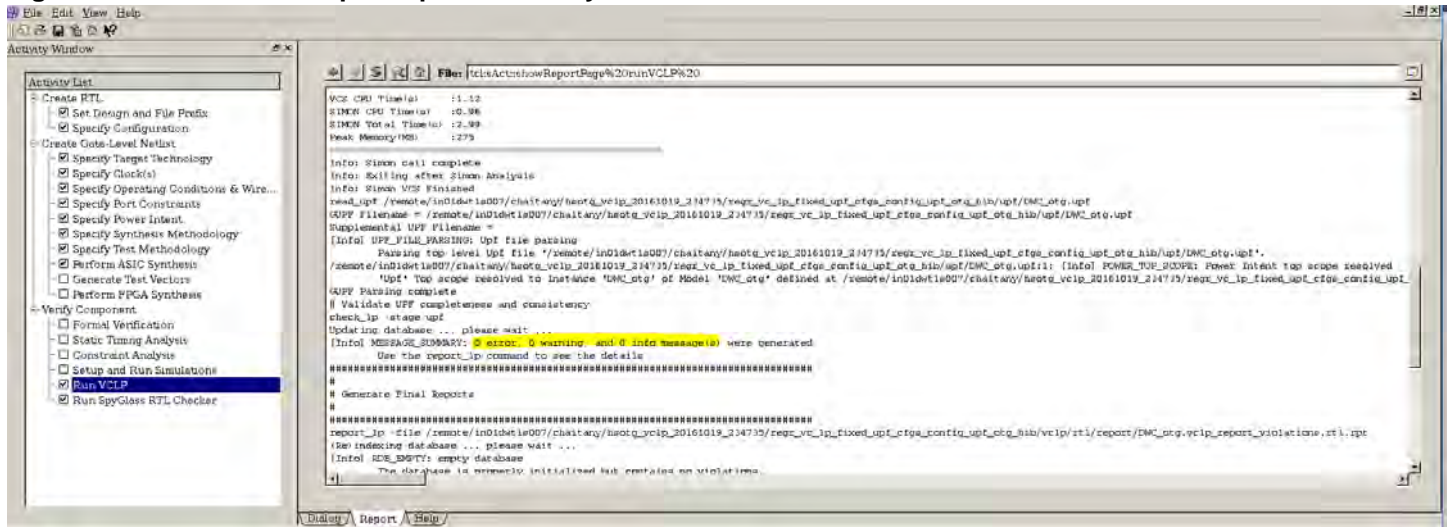
1. Select Verify Component> Run VCLP activity.
2. In the Setup tab, complete the following fields:
 - VCLP Tool Root: Specify the complete path for the VC LP tool.
 - Target Stage: Select either RTL or Netlist stage.
 - For RTL stage, specify the location of the *.upf file.

- For Netlist stage, specify the location of the netlist and the upf prime file (UPF output of Design Compiler).

3. Click Apply.

On completion, the results are available in the Report tab. Go to the report summary, and check for any errors/warnings as shown in [Figure 4-17](#). For debugging the warnings/errors, refer to the *VC Low Power User Guide*.

Figure 4-17 VC LP – Sample Report Summary



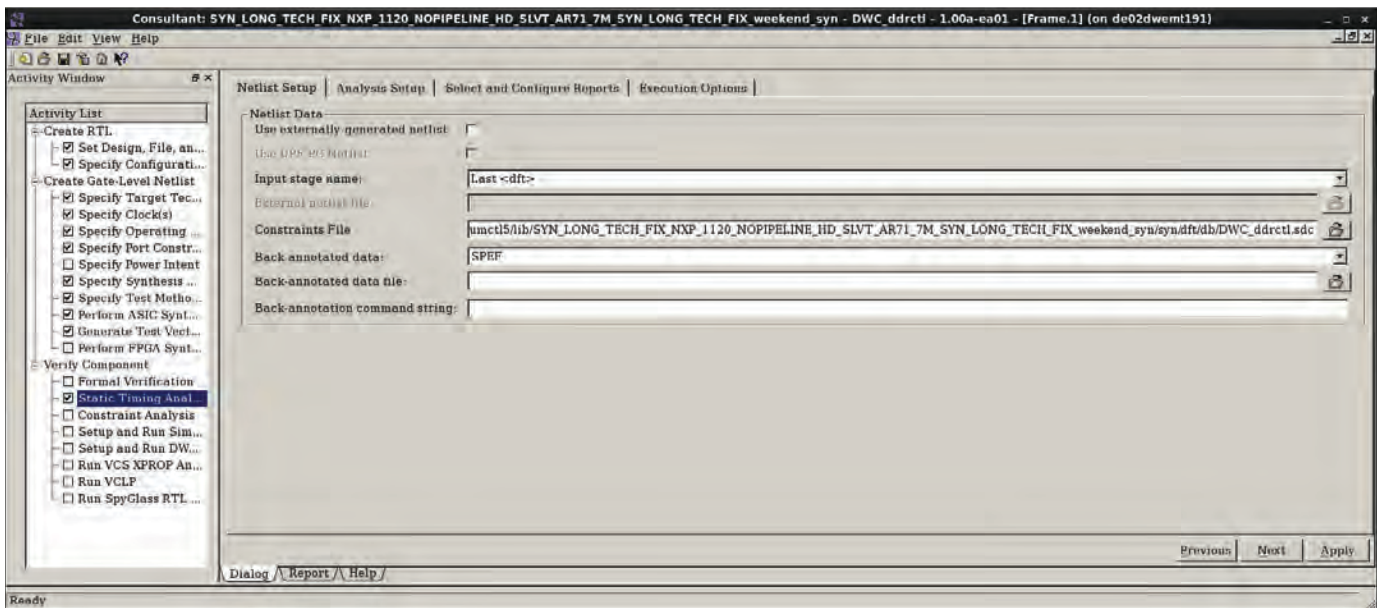
4.5 Running Static Timing Analysis

The **Verify Component -> Static Timing Analysis** activity allows you to run PrimeTime static timing analysis, using constraints generated from coreConsultant. This can be run either on the netlist generated from synthesis, or on an externally generated netlist (for example, from a layout tool). An externally generated SPEF or SDF file can also be read in for back-annotation.

4.5.1 Performing Static Timing Analysis

You must have completed the **Create Gate-Level Netlist** activity (see “[Synthesizing the Controller](#)” on page 40), or have an externally generated netlist available before starting this task. To perform STA on your DWC_ddrctl controller, select the **Static Timing Analysis** activity as shown in [Figure 4-1](#) on page 41.

Figure 4-18 Static Timing Analysis Activity-Netlist Setup Tab



1. Netlist Setup

The first step is to specify what netlist and back-annotated data you wish to use as inputs to the Static Timing Analysis (STA) activity.

□ Netlist

If you choose to use a netlist from the **Create Gate-Level Netlist** activity, you should choose to use the netlist generated from any of the synthesis stages used during that activity. For more information on the synthesis stages, see “[Checking Synthesis Results](#)” on page 50.

If you choose to use an external netlist, specify its location in the **External netlist** file field.

□ Constraints

You can specify what constraints to use. By default, this is populated with the constraints generated from the final synthesis stage from the **Create Gate-Level Netlist** activity, and you will not normally need to change this.

□ Back-annotated data

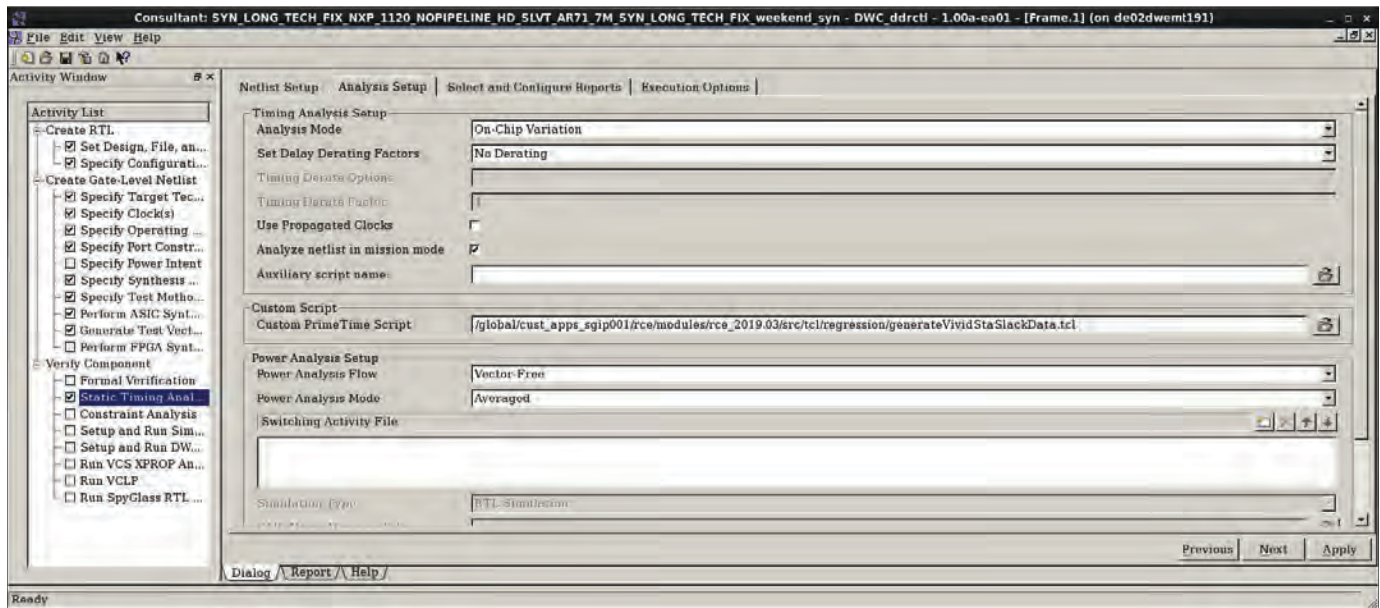
You can choose to use SPEF or SDF annotation, or to run STA without annotation, if none is available. If you choose to annotate, you should specify the SPEF or SDF file in the **Back-annotated data**

file. If you need to provide options to the read_spef or read_sdf command, they can be entered in the **Back-annotation command string**.

2. Analysis Setup

Click on the **Analysis Setup** tab (Figure 4-19 on page 58), which allows you to select various STA setup options.

Figure 4-19 Static Timing Analysis Activity: Analysis Setup Tab



□ Timing Analysis Setup

Here you can choose whether to run in on-chip variation (OCV) or best-case/worst-case mode, and whether to include derating or not. There is also an option to use propagated clocks. For more information, right-click on the individual options and select **What's This?**.

□ Custom Script

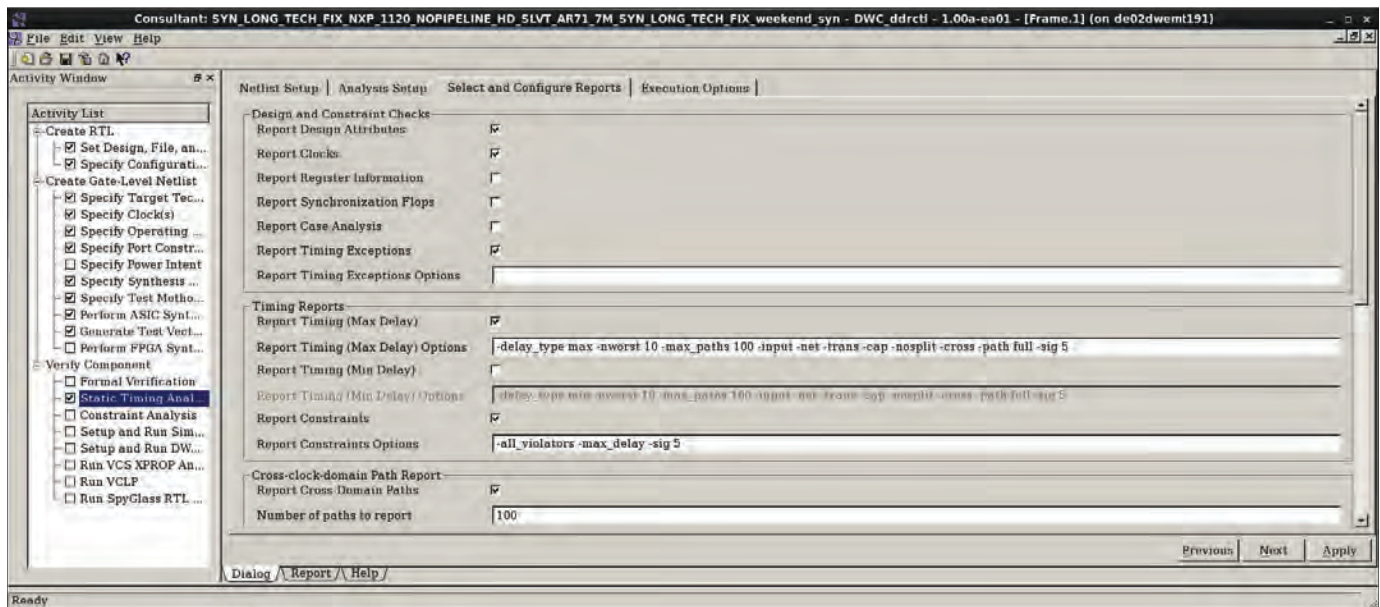
There is an option to select a **Custom Prime Time Script**, which runs at the end of the PrimeTime run (after all reports have been generated).

□ Power Analysis Setup

You can also use PrimeTime to generate power estimates. These can either be vector-free (not based on any external switching information), or based on an SAIF or simulation dump file. Fields are provided to specify the switching activity file, to map names between RTL and gates, and to specify what parts (hierarchical path, and simulation time) of the simulation dump file should be used for the power calculations. For more information, right-click on the individual options and select **What's This?**.

3. Select and Configure Reports

In this tab (see Figure 4-20 on page 59), you can choose what reports you would like PrimeTime to produce.

Figure 4-20 Static Timing Analysis Activity: Select and Configure Reports Tab

- Design and Constraint Checks

You can choose whether to generate reports on design attributes, clocks, case analysis and timing exceptions. For timing exceptions, you can specify options to be appended to the PrimeTime report_exceptions command.

- Timing Reports

You can choose to generate max_delay, min_delay and constraints reports. In each case, options can be added to the appropriate PrimeTime command.

- Cross-clock-domain Path Reports

For detailed information, see “[CDC Methodology](#)” on page 81

- Clock Reports

You can choose to generate clock_timing, min_pulse_width and clock_gating_check reports, and provide options for the last two.

- Power Analysis

This section performs power analysis using PrimeTime-PX. For more information, right-click on the individual options and select **What's This?**.

4. Execution Options

You can choose relevant options in the **Execution Options** tab. As with other coreConsultant activities, you can choose whether to generate the scripts only without running them, or the method of running them.

When you have chosen all relevant options for your Static Timing Analysis run, click **Apply**.

4.5.2 Checking STA Results

After you have run STA, you can check the results by clicking the **Report** tab. The location of all STA results and reports can be seen from the **Report** tab, and is typically in <workspace>/syn/incr1/report/sta. Links to all reports are also provided in the **Report** tab.

After you have run synthesis, you can check the results by clicking the **Report** tab. All the synthesis results and log files are created under your <workspace>/syn directory. The `final` symbolic link points to the current synthesis stage directory. The final log file is written to <workspace>/syn/run.log.

4.5.3 Running STA from Unix Shell

To run Static Timing Analysis from a Unix shell (before running it from coreConsultant):

1. Complete all the setup steps as outlined in “[Performing Static Timing Analysis](#)” on page 57.
2. Select **Static Timing Analysis-> Execution Options -> Generate scripts only?**
3. Click **Apply**.
4. Enter the following commands:

```
% cd <workspace>/syn
% run.scr
```

To run STA again from a Unix shell (after having run it from coreConsultant), enter the following commands:

```
% cd <workspace>/syn
% run.scr
```



Caution

- In the <workspace>/syn directory, `run.scr`, `Makefile`, and `final` are symbolic links to the current synthesis stage or STA files.
- These links are also used/set to different locations during Synthesis, ATPG and Formal Verification. Therefore, ensure they are pointing to the correct location if you are attempting to run STA from the Unix shell AFTER having just run Synthesis, ATPG or Formal Verification.

4.6 Performing Formal Verification



Attention

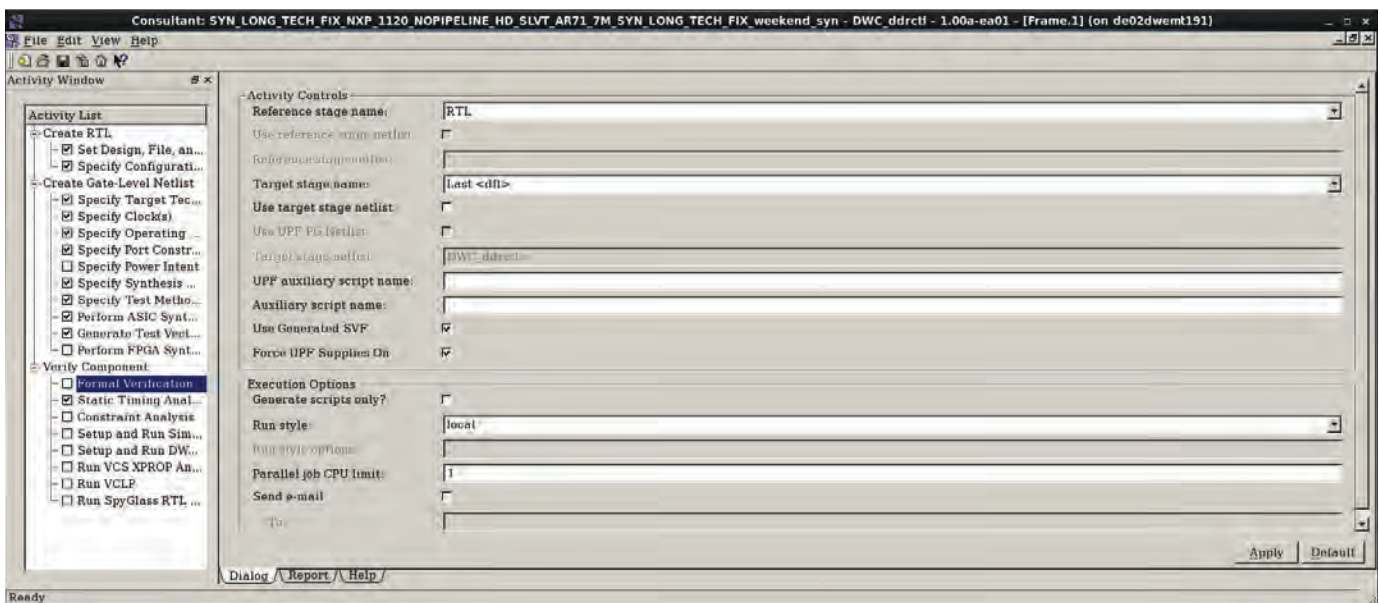
Formal Verification (also referred to as formal equivalence checking) is an activity which takes the output of synthesis as input. Synthesis must be completed before a meaningful set up can exist in the Formality window.

The **Verify Component -> Formal Verification** activity in coreConsultant uses Synopsys Formality (fm_shell) to check two designs for functional equivalence. You can check the gate-level design from a selected phase of a previously executed synthesis strategy against either the RTL version of the design or the gate-level design from another stage of synthesis.

To perform formal verification:

1. Choose the **Verify Component -> Formal Verification** activity. The following dialog box is displayed.

Figure 4-21 Verify Component -> Formal Verification Activity



2. Complete the following fields:
 - ☐ **Reference stage name:** RTL
 - ☐ **Target stage name:** Select last synthesis stage from the pull-down menu.
 - ☐ **Use Generated SVF:** Enabled
3. Click **Apply**.



Hint

For more information about diagnosing and fixing failing points, see Formality User Guide. If you need to work with Formality directly to diagnose failing points, you can load the saved session file (workspace/syn/stage/final/db/Design.fss) directly into Formality.

4.6.1 Checking Formal Verification Results

The Formality results are saved in the `<workspace>/syn/<Output stage>` directory. The file `<workspace>/syn/<Target stage name>.tcl` may be inspected to help you understand the Formal Verification flow.

4.6.2 Running Formality from a Unix Shell

To run Formality from a Unix shell (before running it from coreConsultant), you must first complete all the setup steps as outlined previously, select **Formal Verification -> Execution Options -> Generate scripts only?** and then click **Apply**. Then enter the following commands:

```
% cd <workspace>/syn
% <*>_fm.scr
```

**Note**

The exact name of this script depends on the name of the 'last' executed stage of synthesis.

To re-run Formality from a Unix shell (after having run it from coreConsultant), enter the following commands:

```
% cd <workspace>/syn
% <*>_fm.scr
```

**Caution**

- In the `<workspace>/syn` directory, `run.scr` is a symbolic link to the current Formal Verification run file.
- This link is also used/set to different during Synthesis, Static Timing Analysis and ATPG. Therefore, ensure it is pointing to the correct location if you are attempting to run Formal Verification from the Unix shell AFTER having just run ATPG, Static Timing Analysis or Synthesis.

5

Integrating the Controller

This chapter contains the following sections:

- [“Exporting Controller to Your Chip Design Database”](#) on page 64
- [“Host Interface \(HIF\) Integration”](#) on page 67
- [“PHY Integration”](#) on page 70
- [“External RAM Integration”](#) on page 72

5.1 Exporting Controller to Your Chip Design Database

At a certain point you may want to transfer (export) a configured controller into your own custom design flow. The coreConsultant tool has a number of features to accomplish this task. This section shows you how to access the relevant files, scripts, and information for your configured controller. You can then transfer these to your chip design database.

The topics in this section are as follows:

- [“Instantiating Your Controller”](#) on page 64
- [“Exporting RTL Code”](#) on page 64
- [“Synthesizing to a Device Outside of coreConsultant”](#) on page 64
- [“Exporting Views and Reports”](#) on page 66

5.1.1 Instantiating Your Controller

After you have configured your DWC_ddrctl controller, you can generate an example Component Instantiation.

To instantiate the controller, follow these steps:

1. Select the **Report** tab in the **Specify Configuration** activity.
2. Click the **generate view** link.

The controller instantiation is now available in `export/DWC_ddrctl_inst.v`.

5.1.2 Exporting RTL Code

There is an `export` directory in your workspace. The coreConsultant tool populates this directory with various links and files for export use. After you have configured the DWC_ddrctl controller, the files in [Table 5-1](#) are created in the `<workspace>/export` directory.

Table 5-1 Files (of interest) Created in export Directory After Specify Configuration Activity

File Name	Description
<code><workspace>/export/src/</code>	Configured source code
<code><workspace>/export/DWC_ddrctl.lst</code>	List of source files in proper analysis order

5.1.3 Synthesizing to a Device Outside of coreConsultant

If you want to map and synthesize the controller to a device using the Synopsys synthesis tools within coreConsultant, then follow the process as outlined in [“Synthesizing the Controller”](#) on page 40. Otherwise, you must export the synthesis scripts from coreConsultant as outlined here so that you can synthesize the controller in your own design flow. This section shows you how to access the relevant synthesis scripts and information for your configured controller. You can then transfer these to your chip design database.

To get a detailed description of how to integrate the IP at chip level, enter the following command in the coreConsultant command line:

```
coreConsultant> man Synthesis_API
```

The following pop-up window is displayed.

Figure 5-1 Synthesis_API Pop-Up Window

There are two methods used to export the synthesis scripts. If you have access to Synopsys Design Compiler, then you can use either method. If you not have access to Synopsys Design Compiler, then you must use the `write_sdc` method.

5.1.3.1 Method 1: Access Synopsys Design Compiler Scripts

To access Design Compiler scripts:

1. First complete all the set-up steps in “[Synthesizing the Controller](#)” on page 40.
2. Before clicking **Apply** to complete the **Create Gate-Level Netlist** activity, select the **Synthesize -> Options tab -> Execution Options -> Generate scripts only?** parameter.

[Table 5-2](#) lists files that are relevant to the DWC_ddrctl synthesis flow.

Table 5-2 Files Related to Synthesis Flow

Synthesis Type	File	Description
ASIC	constrain/script/DWC_ddrctl.cscr	Primary constraints
	run.scr	Runs synthesis
	Makefile	Creates synthesis scripts

5.1.3.2 Method 2: Use `write_sdc` Command

Use the `write_sdc` command to write out a script in a Synopsys Design Constraints (SDC) format. The SDC files are TCL scripts that use a subset of the commands supported by PrimeTime and Design Compiler. This file generation process does not require a Design Compiler license.



Attention

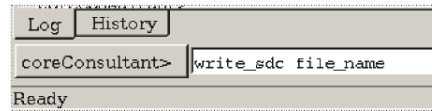
- If a technology library is not loaded in the **Specify Target Technology -> dc_shell target_library variable** (or Synplicity Library Setup in the case of an FPGA), then the generated SDC file contains constraints relative to a generic technology library. You need to modify these constraints to match your target library cell names.

You invoke this command on the coreConsultant command line as follows:

```
coreConsultant> write_sdc [-force] [-version 2.1] file_name
```

If you use the `-force` option to force the writing of the SDC file even if the Specify Target Technology activity (or Synplicity Library Setup in the case of FPGA) is not complete. The `-version` option must be used when writing out the SDC file without completing the Specify Target Technology activity. It will specify the SDC version to be used.

Figure 5-2 `write_sdc` Command



5.1.4 Exporting Views and Reports

As part of the export process, you may want to generate documentation for your configured controller to be used as part of the documentation for an entire chip. The report and view generation process is based on DocBook instead of HTML. This allows documentation for I/O definitions, parameter definitions, and memory maps to be generated in HTML, RTF, and MIF formats from the DocBook source.

For more information, see “[Creating Optional Reports and Views](#)” on page 25. All reports can be accessed in the `<workspace>/report` directory. The optional views are located in the `<workspace>/export` directory.

5.1.5 Simulating Outside coreConsultant

When simulating the PHY and the Controller outside coreConsultant (user simulation environment), ensure that the interrelated SDRAM, Controller and PHY registers (for example, SDRAM timing parameters, SDRAM mode registers, and so on) are programmed to be in sync. For example, if you adapt the `apb.log` (controller register settings) from coreConsultant environment, then the PHY CSRs as well as your DRAM model settings must also be configured likewise. Otherwise, it is most likely the simulation may fail due to errors in timing/MR parameter mismatch.

5.2 Host Interface (HIF) Integration

The HIF on DWC_ddrctl is the command and data interface. Typically, you need to bridge this interface to an AXI or other system bus. This section provides guidelines for this. For more information about this interface, refer to the following sections of LPDDR5/4/4X Memory Controller Databook:

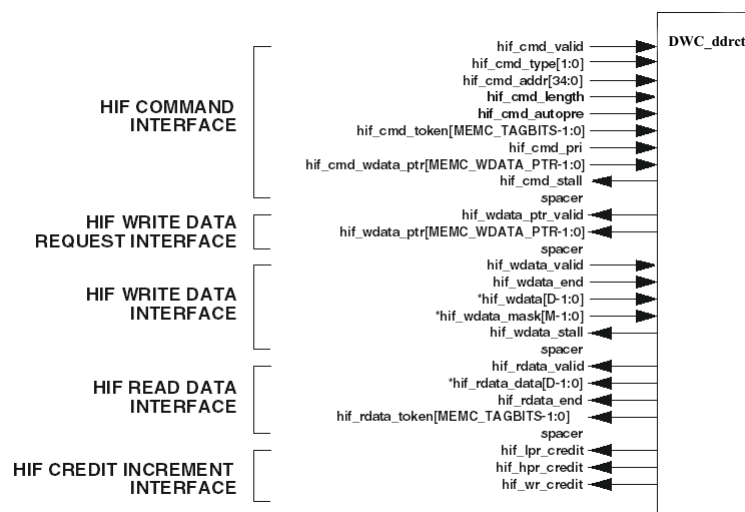
- Credit Requests
- Common Response
- Signals

The SoC core is required to maintain three credit counters - one each for the following queues:

- Low priority read
- High priority read
- Write command

A counter is decremented each time a command is sent to the DWC_ddrctl, and incremented each time a `hif_*_credit` signal from the DWC_ddrctl is asserted. For a read-modify-write (RMW) command, the low priority read and the write credit counters should both be decremented. These counters allow the SoC core to track the number of spaces available in the command queues. If a command queue is full (in other words, the credit counter is full), the SoC core must not send a command to that queue.

Figure 5-3 HIF interface



The width of the data and mask signals depends on the configuration of the DDRCTL. For 1:1 mode configurations (`MEMC_FREQ_RATIO=1`), $D=MEMC_DRAM_DATA_WIDTH*2$. For 1:2 mode configurations (`MEMC_FREQ_RATIO=2`), $D=MEMC_DRAM_DATA_WIDTH*4$. In both cases, $M=D/8$.

5.2.1 HIF Command Interface

The HIF command interface consists of nine signals as shown in [Figure 5-3](#). These are summarized here:

- `hif_cmd_valid` qualifies all the command interface signals. The remaining command interface signals are ignored if this signal is de-asserted. The `hif_cmd_valid` signal must be asserted for one cycle for each command sent to the DWC_ddrctl (assuming that `hif_cmd_stall` is de-asserted).
- `hif_cmd_type` determines whether a write (00), read (01), or read-modify-write (RMW) (10) is sent.
- `hif_cmd_addr` is the word address (not the byte address).

- `hif_cmd_length` (for reads only) determines whether the read burst is of normal length (0) or partial length (in other words, half the normal length) (1).
- `hif_cmd_pri` (for reads only) determines whether the read is low priority (0) or high priority (1).
- `hif_cmd_autopre` determines whether the command should be issued to memory without (0) or with (1) an auto precharge. This can improve the efficiency in cases where it is known that the following command will or will not be a page hit.
- `hif_cmd_token` (for reads only) is a token for the command being sent. This must be generated by the SoC core and is returned from the `DWC_ddrctl` with the read data, so that out-of-order reads can be matched with their commands. The SoC core must keep track of which tokens are pending read responses, and must ensure that the token sent with each command does not match any pending token.
- `hif_cmd_wdata_ptr` (for writes and RMW only) is a pointer to the SoC core's own buffers, which is returned by the `DWC_ddrctl` when it is requesting the write data for this command. This field is not strictly required, as the write data is always requested by the `DWC_ddrctl` in the same order that the commands were sent.

5.2.2 HIF Write Data Request Interface

The write data request interface consists of two output signals from the `DWC_ddrctl` controller and is used to request that the SoC core sends the write data associated with a write or RMW command that has already been sent:

- `hif_wdata_ptr_valid` indicates that the `hif_wdata_ptr` signal is valid. If `hif_wdata_ptr_valid` is de-asserted, then `hif_wdata_ptr` should be ignored.
- `hif_wdata_ptr` is a pointer to the SoC core's own buffers, which matches the previously sent pointer by the SoC core as `hif_cmd_wdata_ptr`. It indicates the write command for which the `DWC_ddrctl` requires write data.

5.2.3 HIF Write Data Interface

The HIF write data interface conveys the write data from the SoC core to the `DWC_ddrctl`, and consists of five signals:

- `hif_wdata_valid` qualifies the other write data interface signals. If `hif_wdata_valid` is de-asserted, the other write data interface signals are ignored. It should be asserted for all cycles of the write data being sent to the `DWC_ddrctl`.
- `hif_wdata_end` indicates that the last cycle of data for the current transaction is sent. It should be asserted on the last cycle of data, and de-asserted on the other cycles.
- `hif_wdata` is the write data for the transaction.
- `hif_wdata_mask` is the byte mask for the transaction.
- `hif_wdata_stall` indicates that no write data can be accepted. If this is asserted, all other write data interface signals are ignored by the `DWC_ddrctl`.

5.2.4 HIF Read Data Interface

The HIF read data interface provides read data from the `DWC_ddrctl` to the SoC core. It consists of the following four signals:

- `hif_rdata_valid` qualifies the other read data interface signals. If this is de-asserted, all other read data interface signals should be ignored. It is asserted for all cycles in which the read data is valid.

- `hif_rdata_data` is the read data being returned from the `DWC_ddrctl`
- `hif_rdata_end` indicates that the current cycle is the last cycle of valid data for the current transaction.
- `hif_rdata_token` is a token which matches the `hif_cmd_token` that has been sent to the `DWC_ddrctl` with the command, and allows the SoC core to identify the command with which this read data is associated.

5.2.5 HIF Credit Increment Interface

The `DWC_ddrctl` indicates to the SoC core each time a command from one of the three command queues is scheduled to the SDRAM. This means that the SoC core should increment its credit count for that queue. The interface consists of one signal for each of the queues:

- `hif_lpr_credit` indicates that a low priority read command has been scheduled.
- `hif_hpr_credit` indicates that a high priority read command has been scheduled.
- `hif_wr_credit` indicates that a write command has been scheduled.

5.3 PHY Integration

The following sections detail how the DWC_ddrctl controller can be integrated with the Synopsys DWC LPDDR5/4/4X PHY, using the testbench options available in the “Testbench Options” tab of “Setup and Run Simulations” in coreConsultant.

This section contains the following sub-sections, which follow the structure of the coreConsultant options.

- “PHY Type” on page 70
- “PHY Simulation Model” on page 70
- “Imported PHY Location” on page 70
- “Imported C code location” on page 71

5.3.1 PHY Type

This parameter selects the PHY type to be used in the simulation. Options are:

- Synopsys DWC LPDDR5/4/4X PHY

5.3.2 PHY Simulation Model

The DWC_ddrctl controller allows users to simulate with different options for the PHY model used. The model is selected in coreConsultant (Setup and Run Simulations -> Testbench Options -> PHY Simulation Models). The options are as follows:

- Included (encrypted) Synopsys PHY model. This option simulates with the pre-installed (encrypted) versions of the Synopsys PHY model.
- Imported Synopsys PHY model. This option provides you the ability to simulate with a PHY model you import yourself. The intention here is to enable you to import a new version of your Synopsys PHY, and continue to simulate it with your existing DWC_ddrctl core. This is not guaranteed to be possible with all new PHY versions.

5.3.3 Imported PHY Location

This option is only relevant if the “Imported Synopsys DWC DDR PHY model” option is chosen, otherwise, it is ignored.

This option must be set to point to the root directory (for example, synopsys/dwc_lpddr54_phy_tsmc7ff18) of the installation of the Synopsys DDR PHY release. coreConsultant then searches the standard PHY directory structure to find the relevant simulation models for the PHY.

**Note**

The imported PHY must correspond to the PHY type selected during the “Setup and Run Simulations” activity.

If the Imported PHY option is chosen, a top-level PHY wrapper module called dwc_ddrphy_top must be included in the above directories. This PHY wrapper is instantiated in the provided ddr_chip module (in <workspace>/sim/testbench/tb/ddr_chip.v), and its ports must match that instantiation. All other Verilog modules required for the PHY must also be included in the above directories.

5.3.4 Imported C code location

This option is only relevant if the "Imported Synopsys DWC DDR PHY model" option is chosen, otherwise, this option is ignored.

For running simulations it is necessary to use C code to initialize the PHY. This option should point to the location of the C code, which is then read in and executed by the simulation. This should point to the phyinit directory of the PHY being simulated. For example,
<<PHY_installation_path>>/latest/phyinit/Latest/software/ path.

5.3.5 Imported Firmware Location

This option is relevant only if the "Imported Synopsys DWC DDR PHY model" option is chosen, otherwise, this option is ignored. For running simulations it is necessary to use firmware when the PHY is performing training/SDRAM initialization. This option must point to the location of the firmware files, which is then read in and executed by the simulation. This should point to the firmware directory of the PHY being simulated. For example, <<PHY_installation_path>>/Latest/firmware/Latest path.

5.3.6 PHY Parameters that can be Controlled from coreConsultant

When using simulating with a PHY in the DDRCTL environment, the PHY is automatically configured by coreConsultant. Some of this configuration is performed based on the configuration of the LPDDR5/4/4X Memory Controller (for example, to ensure matching data widths between the LPDDR5/4/4X Memory Controller and PHY), while other configuration is performed under the control of Testbench Options presented in the coreConsultant GUI.

Table 5-3 on page 71 lists the PHY parameters that are configured to match the DDRCTL Memory Controller configuration.

For more information on all these Synopsys PHY parameters, refer to the appropriate PHY databook.

Table 5-3 List of Macros in PHY VDEFINES file modified / checked by coreConsultant for the DWC LPDDR5/4/4X PHY

Description	VDEFINES macro	Equivalent DWC_ddrctl Testbench/Configuration Parameter
Number of channels	DWC_DDRPHY_NUM_CHANNELS_<n>	DFI1_EXISTS
Number of bytes per channels	DWC_DDRPHY_DBYTES_PER_CHANNEL_<n>	MEMC_DRAM_DATA_WIDTH/8 - divide value by 2 if DFI1_EXISTS=1
Number of Ranks	DWC_DDRPHY_NUM_RANKS_<n>	MEMC_NUM_RANKS
DMI/DBI support enabled	DWC_DDRPHY_DBYTE_DMI_ENABLED	Always enabled
LPDDR5 support enabled	DWC_DDRPHY_LPDDR5_ENABLED	Always enabled

5.4 External RAM Integration

The external RAM depth and width values depend on the values of other configured hardware parameters. It is not possible to configure these values during controller configuration. For the functions based on which the external RAM depth and width are calculated, see External RAM Interface Widths (information only) parameters in the HW/Multiport Parameters section of the “Parameter Descriptions” chapter in the LPDDR5/4/4X Memory Controller Databook.

The exact configured values of the external RAM depth and width for your configuration can be obtained through the coreConsultant tool. See the generated `DWC_ddrctl_cc_constants.v` file after the controller configuration. You can also generate the `ComponentConfiguration.html` that provides these values. For information about generating reports, see “[Creating Optional Reports and Views](#)” on page 25.

For an understanding of the signals used and the related timing diagrams, see the “Write Data” and “Read Reorder Buffer” sections in the “System Interfaces” chapter of the DesignWare® Cores LPDDR5/4/4X Memory Controller Databook.

After you configure the controller in the coreConsultant tool, a report that gives the configured memory sizes and depths is available by clicking the Report tab under the Specify Configuration activity as shown in [Figure 2-4](#) on page 22. These can be used as input to a memory generator to produce the necessary memories that would be used in the Memory SubSystem. This report is also available in the text form at: `<workspace>/export/mem.txt`

A

Additional coreConsultant Information

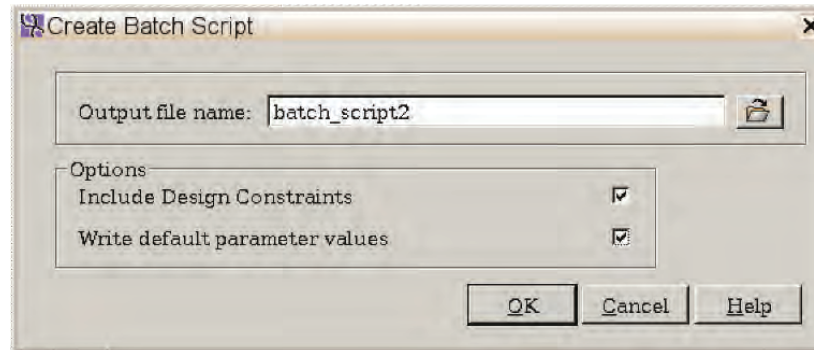
The topics in this chapter are as follows:

- [“Creating a Batch Script”](#) on page 74
- [“Help Information”](#) on page 75
- [“Workspace Directory Structure Overview”](#) on page 77
- [“Dumping Debug Information when Problems Occur”](#) on page 79

A.1 Creating a Batch Script

Batch mode allows you to execute a series of coreConsultant commands from a batch file. To create a batch file, choose the **File > Write Batch Script** menu item and enter a name for the file (shown in [Figure A-1](#)).

Figure A-1 Create Batch Script



You can review and edit the batch file by looking at the file in an ASCII editor.

- By default, the 'parameter configuration' section of the batch script only specifies those parameters whose values have been changed by you from their default values.
- If you have not modified any of the default information in the Synthesis section, it is possible to remove the very verbose synthesis intent commands from the batch script.
- Depending on how you plan to use the batch script (see next paragraph), you may be able to remove all the initial statements from the batch script that deal with creating the workspace. You can prefer to create a batch script that is only used to set configuration parameters and to control all other activities manually.

You can use the batch script to reproduce the workspace using any of the following methods:

- To run in non-GUI batch mode:

```
% coreConsultant -shell -f <batch_file_name>
```

- To run in GUI mode:

```
% coreConsultant
```

If your batch script does not have any commands for creating a workspace, then you must create a workspace first. For more information on creating a workspace, see "[Creating a Workspace](#)" on page 18.

- Source the batch file from the coreConsultant command line:

```
source <batch_file_name>
```

Your source the batch file before, or after, you have created a workspace, depending on whether the script includes commands to create the workspace.

A.2 Help Information

Several types of online help are available through coreConsultant:

- coreConsultant User Guide and Command Reference

These are available through the Help menu (see [Figure A-2](#)) and are also available at `<cc_tool_root>/../doc/dware` where `<cc_tool_root>` is the path to your coreConsultant executable as returned by typing the following command in a Unix shell:

```
% which coreConsultant
```

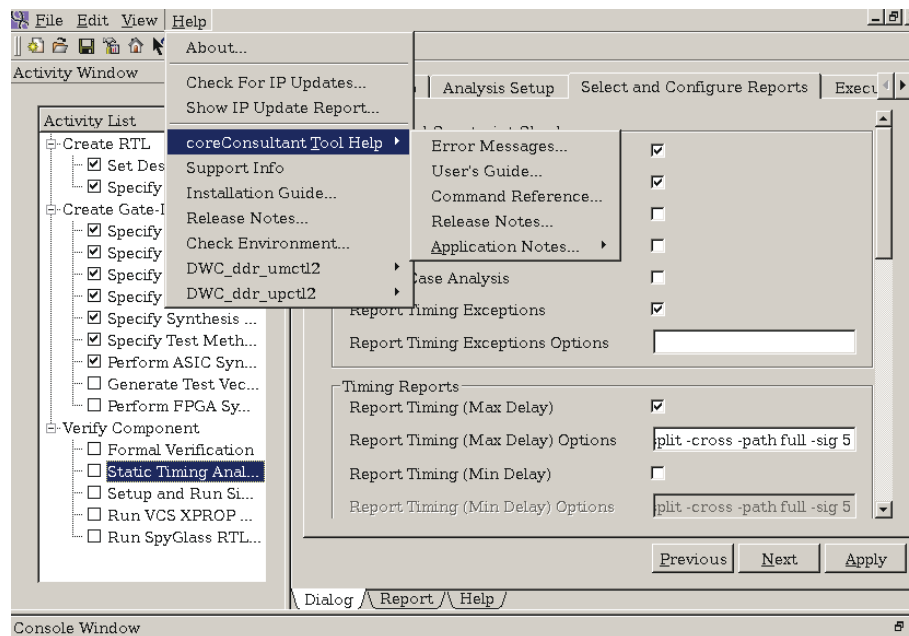
- “What’s This?” Quick Help

When you position your mouse pointer over a GUI item and right-click, coreConsultant displays a menu with the **What’s This?** option. Clicking **What’s This?** displays a brief help message for the selected item. **What’s This?** can also be accessed by left-clicking the Question Pointer on the toolbar and then left-clicking on a GUI item.

- The Toolbar Help Menu

Click the toolbar Help button to show a list of help topics. When you click a topic, the corresponding help information appears. [Figure A-2](#) shows the Help pull-down with the available manuals for both the DWC_ddrctl controller and the coreConsultant tool.

Figure A-2 coreConsultant Help Menu Pull-down Showing DWC_ddrctl Controller and coreConsultant Manuals



- Activity View Help Tab

The Activity View pane features a **Help** tab that displays a detailed, context-specific help page, with additional links to the online command reference and other appropriate references.

■ Help on coreConsultant Commands

The Synopsys coreTools Online Command Reference Index is available through the Help menu. It contains a collection of man pages for all coreConsultant commands, attributes, variables, and item types.

You can also access coreConsultant command help by entering one of the following commands at the coreConsultant prompt in the Console pane:

```
coreConsultant> help [cmd] [-verbose]
coreConsultant> cmd -help
coreConsultant> man [cmd]
```

A.3 Workspace Directory Structure Overview

Figure A-3 illustrates some general directories and files in a coreConsultant workspace.

Figure A-3 coreConsultant Workspace Directory

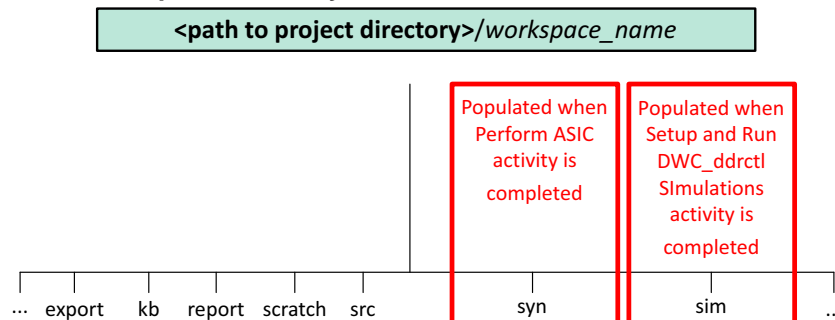


Table A-1 provides a description of the workspace directory and subdirectories.

Table A-1 Workspace Directory Contents

Directory/Subdirectory	Description
auxiliary	Scripts and text files used by coreConsultant. Generated upon first creating workspace.
doc	Contains local copies of DWC LPDDR5/4 Memory Controller documentation (pdf files). Generated upon first creating workspace.
export	Contains files used to integrate results from the completed source configuration and synthesis activities into your design (outside coreConsultant). Generated upon first creating workspace; populated during the Specify Configuration and other activities.
kb	Contains knowledge base information used by coreConsultant. These are binary files containing the state of the design. Generated upon first creating workspace; populated and updated throughout activities.
pkg	Contains RTL preprocessor scripts. Generated during the Specify Configuration activity.
report	Contains all of the reports created by coreConsultant during build, configuration, test and synthesis phases. An index.html file in this directory links to many of these generated reports. Generated upon first creating workspace; populated and updated throughout activities.
scratch	Contains temp files used during the coreConsultant processes. Generated upon first creating workspace; populated and updated throughout activities.

Directory/Subdirectory	Description
sim	Contains test stimulus and output files. Generated upon first creating workspace; updated during the Setup and Run Simulations activity.
src	Includes the top-level RTL file, design_name.sv. Generated upon first creating workspace; populated during the Specify Configuration activity.
syn	Contains synthesis files for the DWC_ddrctl controller. Generated upon first creating workspace; updated during Create Gate-Level Netlist activity and Formal Verification activity.

A.4 Dumping Debug Information when Problems Occur

The menu entry, **File > Build Debug Tar-file**, is used to capture debug information for the Synopsys Support Center. This menu item creates the file `<working_dir>/debug.tar.gz`.

This debug file includes:

- Batch script for recreating the workspace
- Output from the existing `debug_info` command
- Synthesis and simulation log files (if available)
- Results of an environment check (if available)

B

CDC Methodology

This appendix provides detailed information about the Clock Domain Crossing (CDC) strategy used in DWC_ddrctl. In addition, refer to [“Running Spyglass Lint and CDC”](#) on page 28.

- [“Clocks, Resets, and Clocking Scheme”](#) on page 82
- [“Synchronizers Used in DWC_ddrctl”](#) on page 83
- [“Synthesis Constraints”](#) on page 85
- [“Synchronization Method”](#) on page 87
- [“CDC Constraints and Report”](#) on page 89

B.1 Clocks, Resets, and Clocking Scheme

See the following sections:

- “Clock and Reset Requirements” section in the LPDDR5/4/4X Memory Controller Databook.
- Clocks and Resets Signals in the Signal Descriptions chapter of the LPDDR5/4/4X Memory Controller Databook.

Also see the “Clock and Reset Requirements” section in the LPDDR5/4/4X Memory Controller Databook.

Figure 6-1 in the Programming chapter of the LPDDR5/4/4X Memory Controller Programming Guide illustrates the DWC_ddrctl clocking domain.

Main boundaries are:

- `pclk - aclk_n/core_ddrc_core_clk`: Crossing for configuration registers coming from the APB and used in the main logic, or for status/interrupt signals coming from the main logic and written to a status register. Crossing is done inside the APB device. Different methods for different type of registers are used. See “[Synchronization Method](#)” on page 87 for details. Whenever crossing is not necessary, no synchronizer is instantiated and value is simply passed through.
- `hclk_n/aclk_n - core_ddrc_core_clk`: Crossing between port n and the main logic. This is done inside the XPI block, primarily to transfer read/write command information and write data from the input port to the main logic and read data and read/write responses from the main logic to the output. Crossing is done through dual port FIFOs, see Data Bus Signals for details. If crossing is not necessary, a single clock dual-port FIFO is used instead. See Data FIFO and Replaceable Components for details.

B.2 Synchronizers Used in DWC_ddrctl

A synchronizer is instantiated for every signal, register or data-bus crossing between different clock domains, with the exception of APB static and quasi-dynamic registers. For details, see “[APB Static Registers](#)” on page 87.

DWC_ddrctl uses proven blocks, called Basic Core Modules (BCMs), to implement CDC. It uses a library of design-specific CDC parts that are mapped to the standard DesignWare library parts in <http://www.synopsys.com/dw/buildingblock.php> as follows:

Table B-1 DWC_ddrctl to DesignWare Library Part Mapping

DWC_ddrctl Part Name	DesignWare Library Part Name	Type of Synchronizer
AXI Interface Crossing		
DWC_ddrctl_bcm21.v	DW_sync	Single Clock Data Bus Synchronizer
APB Interface Crossing		
DWC_ddrctl_bcm25.v	DW_data_sync (x2 DW_sync plus flipflop)	Data Bus Synchronizer with Acknowledge
DWC_ddrctl_bcm23.v	DW_pulseack_sync	Pulse Synchronizer with Acknowledge
AXI Clock Domain Crossing		
DWC_ddrctl_bcm02.v	DW_mux_any	Universal Multiplexer
DWC_ddrctl_bcm05.v	DW_fifoclt_if	Submodule of DWbb_bcm07
DWC_ddrctl_bcm06.v	DW_fifoclt_s1_df	Synchronous (Single Clock) FIFO Controller with Dynamic Flags
DWC_ddrctl_bcm07.v	DW_fifoclt_s2_sf	Synchronous (Dual-Clock) FIFO Controller with Static Flags
DWC_ddrctl_bcm50.v		Memory read data multiplexer
DWC_ddrctl_bcm56.v		Memory array, used together with DWC_ddrctl_bcm50.v
DWC_ddrctl_bcm57.v	DW_ram_r_w_s_dff	Synchronous Write-Port, Asynchronous Read-Port RAM (Flip-Flop Based)
DWC_ddrctl_bcm58.v		Dual clock two port RAM with re-timing registers (Flip-Flop Based)
DWC_ddrctl_bcm65.v	DW_fifo_s1_sf	Synchronous (Single Clock) FIFO with Static Flags
DWC_ddrctl_bcm66_wae.v	DW_fifo_s2_sf	Synchronous (Dual-Clock) FIFO with Static Flags with output ports for write address and write enable
Automotive Crossing		
DWC_ddrctl_bcm65_atv.v		Automotive version of DWC_ddrctl_bcm65.v

DWC_ddrctl Part Name	DesignWare Library Part Name	Type of Synchronizer
DWC_ddrctl_bcm66_wae_atv.sv	DW_fifo_s2_sf	Synchronous (Dual-Clock) FIFO with Static Flags for Automotive IP with output ports for write address and write enable
DWC_ddrctl_bcm05_atv.v		Automotive version of DWC_ddrctl_bcm05.v
DWC_ddrctl_bcm06_atv.v		Automotive version of DWC_ddrctl_bcm06.v
DWC_ddrctl_bcm07_atv.v		Automotive version of DWC_ddrctl_bcm07.v
DWC_ddrctl_bcm21_atv.v		Automotive version of DWC_ddrctl_bcm21.v
DWC_ddrctl_bcm95_i.v		Triple Module Replication (TMR) Register used for Automotive support

B.3 Synthesis Constraints

The synthesis constraints applied to the design obey to a specific CDC Methodology, adding timing exceptions which override the default flop-to-flop timing paths that are defined for each individual clock domain. The `set_max_delay` constraint overrides the setup timing checks into the destination flip-flop while the `set_min_delay` constraint overrides the hold timing checks into the destination flip-flop. The `"-ignore_clock_latency"` switch in the `set_max_delay/set_min_delay` constraints is required in order to remove the clock tree latency from the timing calculations.

Note that `set_false_path` constraint has higher priority than `set_max_delay/set_min_delay` constraints, therefore these must not be used to constraint the same paths, otherwise `set_max_delay/set_min_delay` constraints will be ignored.

The CDC Methodology obeys to the following rules:

1. Global constraints are applied between asynchronous clocks. Every path between two different clock domains is constrained with a maximum delay (`set_max_delay`) equal to 150% of the destination clock period.

This ensures that the qualified data is always ready when the qualifier arrives. This constraint may be too strict in some cases (previous constraint always considers the worst case, that is when using a double register synchronizer).

If you encounter any difficulties meeting timing at the clock boundaries, the constraint can be relaxed by setting the maximum delay to $(\text{number of synchronizer stages} - 0.5) \times \text{destination clock periods}$ for qualified data signals.

These global constraints will be overridden if any specific path constraints are applied.

2. Use `"-allow_paths"` switch with `set_clock_groups`

The `set_clock_groups` constraint can be used to provide clock group information required in Signal Integrity analysis during P&R only if switch `"-allow_paths"` is included, otherwise `set_max_delay / set_min_delay` constraints will be ignored.

3. Define `set_max_delay` of 1 source clock period and `set_min_delay` of 0 constraints for Gray-coded signals

A `"set_max_delay -ignore_clock_latency"` constraint equivalent to 1 period (or less) of the source clock domain shall be applied in all Gray-coded signals reaching the first synchronization flip-flops in the destination clock domain. A `"set_min_delay -ignore_clock_latency"` constraint with value 0 shall be applied to the same paths. These constraints will ensure that only valid Gray codes will be sampled in the destination clock domain.

4. Define `set_max_delay` of $(\text{Number of Sync stages} - 0.5) \times \text{destination clock period}$ and `set_min_delay` of 0 constraints for Qualifier-based Data Bus signals

A `"set_max_delay -ignore_clock_latency"` constraint equivalent to $(\text{Number of Sync stages} - 0.5) \times \text{period}$ (or less) of the destination clock domain shall be applied in Qualifier-based Data Bus signals reaching the first synchronization flip-flops in the destination clock domain. A `"set_min_delay -ignore_clock_latency"` constraint with value 0 shall be applied to the same paths. These constraints will ensure that the Data Bus signals are stable when sampled in the destination clock domain.

5. Define `set_max_delay` of 1 destination clock period and `set_min_delay` of 0 constraints for all CDC crossings (excluding Gray-coded and Qualifier-based Data Bus signals)

A `"set_max_delay -ignore_clock_latency"` constraint equivalent to 1 period (or less) of the destination clock domain shall be applied in all CDC crossings (excluding Gray-coded and Qualifier-based Data Bus signals) reaching the first synchronization flip-flops in the destination clock domain. A `"set_min_delay -ignore_clock_latency"` constraint with value 0 shall be applied to the same paths. These

constraints limit the separation between the flip-flops connecting the source and destination clock domains, therefore maintaining the assumptions for a safe CDC implementation.

6. Define "set_false_path -through" constraint for quasi-static signals at the output of the Bus Delay components

Since quasi-static signals are not required to meet timing, the "set_false_path -through" constraint will remove the timing checks on those paths. The "set_false_path -through" constraint shall be applied to the output pin of Bus Delay components (DWC_ddrctl_umctl2_bcm36_nhs).

A script runs an exhaustive search for the CDC cells instantiated through the design and applies the necessary constraints automatically.

In some configurations, synthesizer might be able to simplify the logic at the FIFO output. Doing so, it may get rid of some of the AND gates which protect against glitch propagation in the destination domain when the FIFO is empty.

To avoid unwanted optimization at the FIFO outputs, it is required to use the set_size_only constraint applied to the leaf cells of all the FIFO qualifiers. This allows the respective AND gates to be resized while preserving the AND gate functionality.

The following synthesis constraint applies the size_only attribute to the leaf cells of all U_AND_2TO1_SIN instances:

- foreach_in_collection cell [get_cells -hierarchical U_AND_2TO1_SIN*] {set_size_only -all_instances [get_cells [get_object_name \$cell]/*]}

Both the CDC methodology and "set_size_only" constraints are applied during the synthesis constrain phase, by automatically writing these constraints into the design constraints file, which can be found in <path_to_cC_workspace>/syn/constrain/script/DWC_ddrctl.tcl.

B.4 Synchronization Method

The DDRCTL uses different types of synchronizers depending on type of signal to be crossed.

All synchronizers internally implement missampling models for verification purposes. These models are set to introduce missampling errors randomly from 0 to 1 clock cycle. Models can be enabled by defining the macro `DW_MODEL_MISSAMPLES`.

B.4.1 APB Static Registers

No synchronization is done in this case.

Registers defined as static must not be reprogrammed outside reset. Their value in `pclk` is simply sampled directly in the destination domains (`core_ddrc_core_clk/aclk` domains) without synchronizing circuits.

Registers defined as quasi dynamic are crossed the same way (that is, without synchronizers), but they can be changed outside reset, for details, see “Quasi Dynamic Registers” in the LPDDR5/4/4X Memory Controller Programming Guide.

The naming convention used in the APB device logic makes it easy to identify these registers. Their instances are denoted by the prefix “`cfgs_ff`” within the `apb_slvif` module.

B.4.2 Dynamic Single Bit Signals

For single bit signals that are assured to keep their value for at least a guaranteed number of destination clock cycles, a standard multiple register synchronizer is used. Depth of synchronizer can be programmed to 2, 3 or 4, depending on the needs.

Such signals can be easily identified in the logic. In a CDC report generated by a CDC checking tool (Spyglass CDC/Meridian CDC), destination points which contain instance of `bcm21`, are signals synchronized using this method. For details about how to generate a report with Spyglass, see “[CDC Constraints and Report](#)” on page 89.

For the list of APB registers that fall into this category, see “Dynamic Registers”.

Figure B-1 Standard Double Register Synchronizer



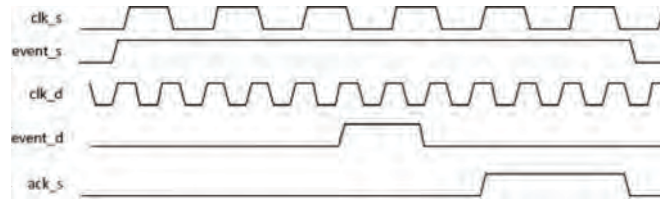
B.4.3 Pulse Signals

Pulses are synchronized through a dual clock pulse synchronizer.

This is the case for one-to-set/one-to-clear type registers in the APB logic. These registers can be easily identified in a CDC report as a destination point whenever there is an instance of `bcm23`.

Figure B-2 shows the synchronizer behavior: event_s is crossed to a single clock signal event_d in the destination domain. Acknowledge is then sent back to the source domain and at this stage input can be cleared.

Figure B-2 Dual Clock Pulse Synchronizer

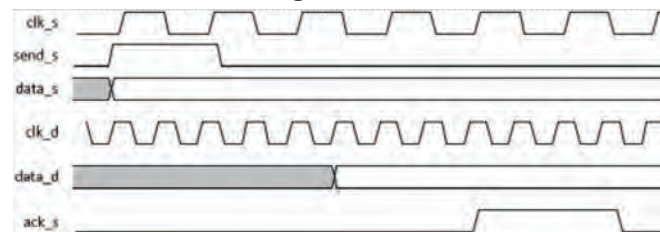


B.4.4 Data Bus Signals

Data bus signals such as multi bit dynamic configuration registers are crossed through a data bus synchronizer with acknowledge. A one clock cycle signal on the source domain indicates the end of the transfer.

Such signals can be identified in a CDC report as destination point whenever there is an instance of bcm25.

Figure B-3 Data Bus Synchronizer with Acknowledge



Data buses that are required to stream are synchronized through FIFOs, which internally implement a dual-clock FIFO synchronizer, which can be recognized when an instance of bcm07 is seen in the destination point. This is the case for data, address and response paths inside the XPI.

Every FIFO is equipped with empty/full signals for handshake purposes.

B.4.5 Data FIFO and Replaceable Components

Data RAM inside dual-clock FIFOs are implemented as two separate blocks:

- Memory array (bcm56), which provides an array of registers in the destination clock domain
- Output stage (bcm50) with replaceable data multiplexer (bcm02)

The output multiplexer can be easily replaced, in case, for example, a glitch-free multiplexer is needed for a particular application.

If no crossing is necessary (inputs/outputs are synchronous), a single clock FIFO is used instead. This is implemented using a standard dual port memory array (bcm57) and a single clock FIFO controller (bcm06).

B.5 CDC Constraints and Report

You can use the coreConsultant tool to generate a report of the clock domain crossing paths. This is useful when compiling a list of registers where timing checks can be ignored in gate-level simulations. For more information, see “[Running Spyglass Lint and CDC](#)” on page 28. To generate the CDC report only cdc/cdc_verify_struct Goal is needed.

Spyglass CDC constraints for your specific configuration are provided with the coreConsultant tool. After configuring the controller, execute the Spyglass flow as explained in the section “[Running Spyglass Lint and CDC](#)” on page 28. You can find the CDC constraints and waiver files in your workspace as documented in [Table 2-4](#) on page 30.

Two main reports are generated in directory

`<workspace>/spyglass/work/core/<top_name>/cdc/cdc_verify_struct/spyglass_reports/clock-reset/`

- `Ar_sync01.csv`: Spreadsheet containing all the reset synchronizers
- `Ac_sync01.csv`: Spreadsheet containing register crossings for scalar signals within the IP with source and destination point
- `Ac_sync02.csv`: Spreadsheet containing register crossings for vector signals within the IP with source and destination point

Each line of the `Ar_sync01` spreadsheet refers to a specific reset, information provided are:

- `RESET`: Reset signal source path
- `PINTYPE`: Reset type (clear/set)
- `INSTANCE`: Destination register
- `CLOCK`: Destination clock
- `SYNCHRONIZER`: path to the reset synchronizer

Each line of `Ac_sync02` spreadsheet refers to a specific register which is synchronized between a source and a destination clock. Information provided are:

- `SOURCE`: Start point
- `SOURCE CLOCK`: Clock start point
- `DEST`: End point
- `DEST CLOCK`: Clock end point
- `METHOD`: Synchronization method (synchronizer cell name)
- `CSV FILE`: Spreadsheet for the specific crossing

More information about each crossing can be found in the csv file linked in `Ac_sync02.csv`.

OCCAP Synthesis Constraints

If you are using the OCCAP feature (see “On Chip Command and Address Path Protection (OCCAP)” in the LPDDR5/4/4X Memory Controller Databook), special care is needed with respect to synthesizer setup for the following:

- Duplicated modules (See ‘Duplication with Comparison’ section in the LPDDR5/4/4X Memory Controller Databook.)
- Automotive FIFO controllers (See ‘Automotive FIFO Controllers’ section of the LPDDR5/4/4X Memory Controller Databook).
- Triple Module Replication (TMR) of specific registers (See ‘Triple Module Replication (TMR) of Registers’ section in the LPDDR5/4/4X Memory Controller Databook).

Synthesizer might be able to “simplify” the logic of these features which use redundant logic for protection. For example, the duplicated modules may be incorrectly simplified by combining some of the registers in one instance of the module with the other instance of same module.

The topics in this chapter are as follows:

- [“Duplicated Modules” on page 92](#)
- [“Automotive FIFO Controllers” on page 93](#)
- [“Triple Module Replication \(TMR\) of Specific Registers” on page 94](#)

C.1 Duplicated Modules

To avoid unwanted optimization on this logic, it is recommended to use the "set hierCells" attribute for the cells of the duplicated modules. An example of this is as follows:

- `set hierCells [get_cells -quiet -hierarchical { *ddrc_ctrl_inst* *mr_inst* *rd_inst* *RP_inst* *WP_inst* *WS_inst* *AU_inst* }]`
- `if {[sizeof_collection $hierCells]} { set_ungroup $hierCells false}`

**Note**

If synthesis runs in cC environment, the above constraints are part of the coreTools customizations that are automatically done. Otherwise, you must ensure that these constraints are part of your SDC.

C.2 Automotive FIFO Controllers

To avoid unwanted optimization on this logic, it is recommended to use the “set size_only” attribute for the cells of the triplicated registers/modules.

For all configurations, `DWC_ddrctl_bcm95_i` is used and its internal register has a unique name when triplicated: “dw_so_reg”. Hence, an example for this is as follows:

- `foreach_in_collection cell [get_cells -hierarchical *dw_so_reg*] {set_size_only -all_instances [get_cells [get_object_name $cell]]}`

Furthermore, when using Asynchronous AXI ports, the `DWC_ddrctl_bcm21_atv` is used inside the `DWC_ddrctl_bcm05_atv` (synchronous (Dual Clock) FIFO controller). Hence, it is recommended to use the following:

- `set_size_only -all_instances [get_cells -hierarchical *sample_meta*]`
- `set_size_only -all_instances [get_cells -hierarchical *sample_sync*]`

C.3 Triple Module Replication (TMR) of Specific Registers

To avoid unwanted optimization on this logic, it is recommended to use the “set size_only” attribute for the cells of the triplicated registers inside the `DWC_ddrctl_bcm95_i` module:

- `foreach_in_collection cell [get_cells -hierarchical *dw_so_reg*] {set_size_only -all_instances [get_cells [get_object_name $cell]]}`

**Note**

This is already covered by the Automotive FIFO controller synthesis requirements.
