SYNOPSYS®

DesignWare Cores Synchronous Serial Interface (SSI)

User Guide

DWC_ssi – Product Code: B858-0

1.01a
August 2017

Copyright Notice and Proprietary Information

Destination Control Statement

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

Disclaimer

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

Trademarks

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

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

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.
690 E. Middlefield Road
Mountain View, CA 94043

www.synopsys.com

Contents

Revision History

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

Date	Release	Description
August 2017	1.01a	**Added:** ■ “eXecute In Place (XIP) Transfers” on page 71 ■ “Concurrent XIP and non-XIP Transfers” on page 72 **Updated:** ■ Updated coreConsultant GUI screen-shots
August 2016	1.00a	First version of the document.

Preface

This preface contains the following sections

Product Description

This databook provides information that you need to interface the DesignWare Synchronous Serial Interface (SSI), referred to as DWC_ssi throughout the remainder of this databook.

The information in this databook includes a functional description, signal and parameter descriptions, and a memory map. Also provided are an overview of the component testbench, a description of the tests that are run to verify the coreKit, and synthesis information for the coreKit.

User Guide Organization

The chapters of this user guide are organized as follows:

- Chapter 1, "Product Overview" introduces the DWC_ssi features, supported standards, and architecture.
- Chapter 2, "Configuring the Core" provides an overview of the step-by-step process you use to configure, synthesize, simulate, and export the DWC_ssi core using the Synopsys coreConsultant tool.
- Chapter 3, "Verification" provides information on verifying the configured DWC_ssi.
- Chapter 4, "Programming" discusses the programming sequences for operating the host controller.
- Appendix A, "Additional coreConsultant Information", provides miscellaneous information related to coreConsultant, such as writing a batch script, accessing help information, and workspace directory contents.

Related Product Information

Refer to the following documentation:

- Verification IP (VIP)

  To run simulations in coreConsultant, the testbench uses the following:

  - Synopsys AMBA VMT VIP
    - Download from the *SolvNet Download Center*
- Synopsys Tools
  - *coreConsultant User's Guide*

Web Resources

- DesignWare IP product information: http://www.designware.com
- Your custom DesignWare IP page: http://www.mydesignware.com
- Documentation through SolvNet: http://solvnet.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): http://www.synopsys.com/keys

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 <*core tool startup directory*>/debug.tar.gz.
  - For simulation issues outside of coreConsultant or coreAssembler:
    - Create a waveforms file (such as VPD or VCD)
    - 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 Support Center, with a description of your question and supplying the previous information, using one of the following methods:
  - *For 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 enter a call.
Provide the requested information, including:

- **Customer Tracking Number:** Enter your project name. Use the name for cases related to the same project.
- **Product:** DesignWare Cores
- **Sub Product 1:** AMBA
- **Sub Product 2:** DWC_ssi
- **Product Version:**1.01a
- **Problem Type:**
- **Issue Severity:**
- **Problem Title:** Provide a short summary of the issue or list the error message you have encountered
- **Problem 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, and Tool Version number in your e-mail (as identified above) 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:

    http://www.synopsys.com/Support/GlobalSupportCenters

1 Product Overview

The DWC_ssi is a configurable, synthesizable, and programmable component that is a full-duplex master or slave synchronous serial interface (SSI). This component is an AMBA version 2.0-compliant AHB (Advanced High-performance Bus) slave device and is part of the family of DesignWare Synthesizable Components.

1.1 DesignWare System Overview

The Synopsys DesignWare Synthesizable Components environment is a parameterizable bus system containing AMBA version 2.0-compliant AHB (Advanced High-performance Bus) and APB (Advanced Peripheral Bus) components, and AMBA version 3.0-compliant AXI (Advanced eXtensible Interface) components.

The host processor accesses data, control, and status information on the DWC_ssi through the AHB interface. The DWC_ssi may also interface with a DMA Controller using an optional set of DMA signals, which can be selected during configuration.

The DWC_ssi can be configured in one of these modes of operations:

- Serial master
- Serial slave

You can connect, configure, synthesize, and verify the DWC_ssi within a DesignWare subsystem using coreAssembler, documentation for which is available on the web in the *coreAssembler User Guide*.

If you want to configure, synthesize, and verify a single component such as the DWC_ssi component, you might prefer to use coreConsultant, documentation for which is available in the *coreConsultant User Guide*.

2 Configuring the Core

DesignWare Synthesizable IP (SIP) component for DWC_ssi is packaged using Synopsys coreTools, which enable you to configure, synthesize, and run simulations on a single SIP title, or to build a configured subsystem using DesignWare components for AMBA 2/AMBA 3 AXI library. You can do this by generating a workspace view using one of the following coreTools applications:

- coreConsultant – Used for configuration, RTL generation, synthesis, and execution of packaged verification for a single SIP title. The *coreConsultant User Guide* provides complete information on using coreConsultant.
- coreAssembler – Used for building and configuring a subsystem that connects multiple SIP titles with DWC_ssi and AMBA 2/AMBA 3 library components, RTL generation, synthesis, and creation of a template subsystem testbench. The *coreAssembler User Guide* provides complete information on using coreAssembler.

A workspace is your working version of a DesignWare SIP component or subsystem. In fact, you can create several workspaces to experiment with different design alternatives.

Hint If you are unfamiliar with coreTools—which is comprised of the coreAssembler, coreConsultant, and coreBuilder tools—you can go to *Using DesignWare Library IP in coreAssembler* to "get started" learning how to work with DesignWare SIP components.

This chapter shows you how to use coreConsultant and coreAssembler tools to configure, simulate, synthesize, and export the DWC_ssi core to your design flow.

The final output from coreConsultant design flow is a set of RTL files (the configured RTL core) and scripts to allow you to run various tools standalone within your own design flow.

The topics in this chapter are as follows:

2.1 Configuring the Core Using coreConsultant

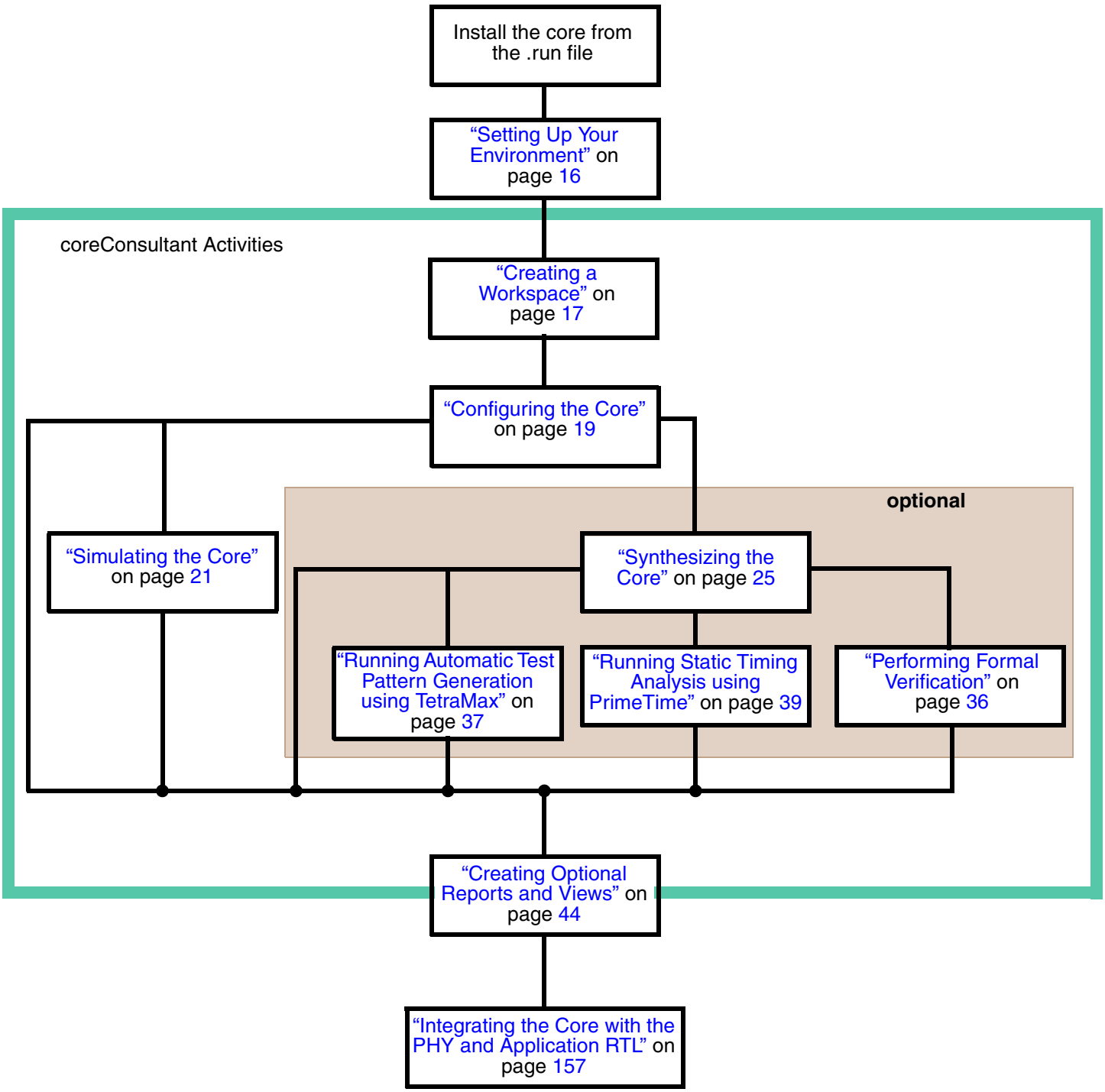
After you have correctly downloaded and installed a release of DesignWare SIP components and then setup your environment, you can begin to work on DWC_ssi using coreConsultant. This section describes how to configure, simulate and synthesize the core using coreConsultant.

This section discuses the following topics:

2.1.1 Design Flow

The coreConsultant GUI guides you through the core design flow. Most coreConsultant activities are optional as indicated by the various paths shown in Figure 2-1 and do not need to be completed before exporting the configured RTL to your chip design flow.

Figure 2-1 Design Flow



2.1.2 Setting Up Your Environment

Note Correctly setting up your installation directory and environment ensures that you can quickly configure the core.

To confirm that you have a correct installation directory and environment setup, perform these steps:

- Confirm that you have installed the core design files and fully set up your environment as described in the *DWC_ssi Installation Guide*.
- Confirm that you are using the required versions of coreConsultant and your preferred synthesis and simulation tools as specified in the Installation Guide.
- Check that $DESIGNWARE_HOME points to your installation base directory. Executing the following command in a Unix terminal shell lists a directory structure similar to that described in the "DESIGNWARE_HOME Directory Structure" appendix of the Installation Guide.

  ```
  % ls $DESIGNWARE_HOME
  ```

- When you are using Synopsys Design Compiler, check that $SYNOPSYS points to the Synopsys tools tree.
- Check that the DWC_ssi licenses are installed correctly on your license server by using the `lmstat` command. For example, enter:

  ```
  % lmstat -a -c $LM_LICENSE_FILE | grep <PRODUCT_LICENCE_FILE>
  ```

  For more information, see the "Setting License File Environment Variable" and "Checking License Requirements" sections of the *Installation Guide*.

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

2.1.3 Creating a Workspace

A workspace is a local Unix directory structure containing your configured copy of the DWC_ssi core. For more details about this directory structure, refer to "Workspace Directory Structure Overview" on page 77. You can create several workspaces to experiment with different design alternatives.

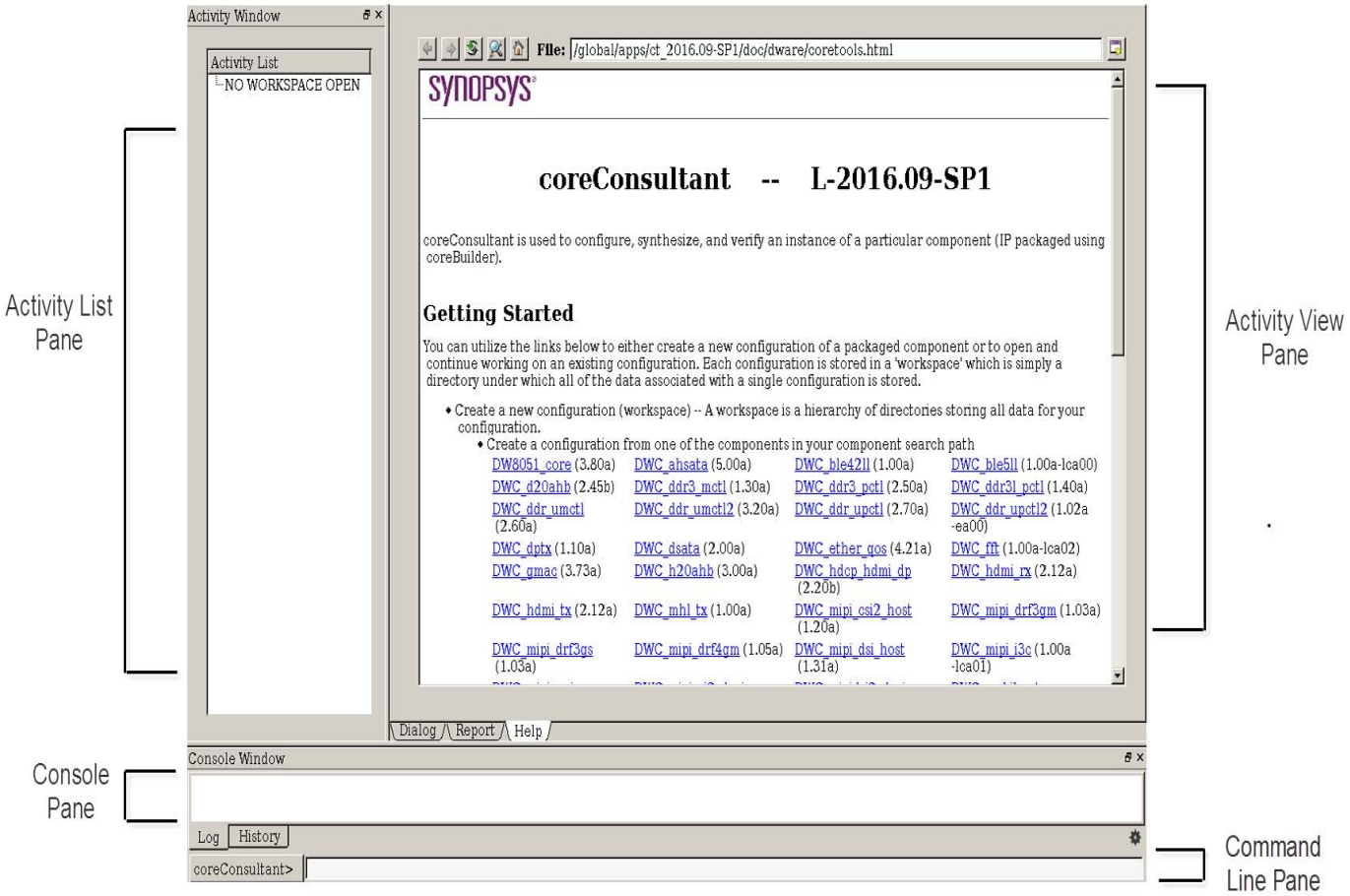
To create a workspace, follow these steps:

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

   ```
   % coreConsultant &
   ```

   Figure 2-2 shows the initial coreConsultant screen.

Figure 2-2 Initial coreConsultant Screen



1. Create a workspace.

From this screen, click on the name of the IP core that you would like to configure.

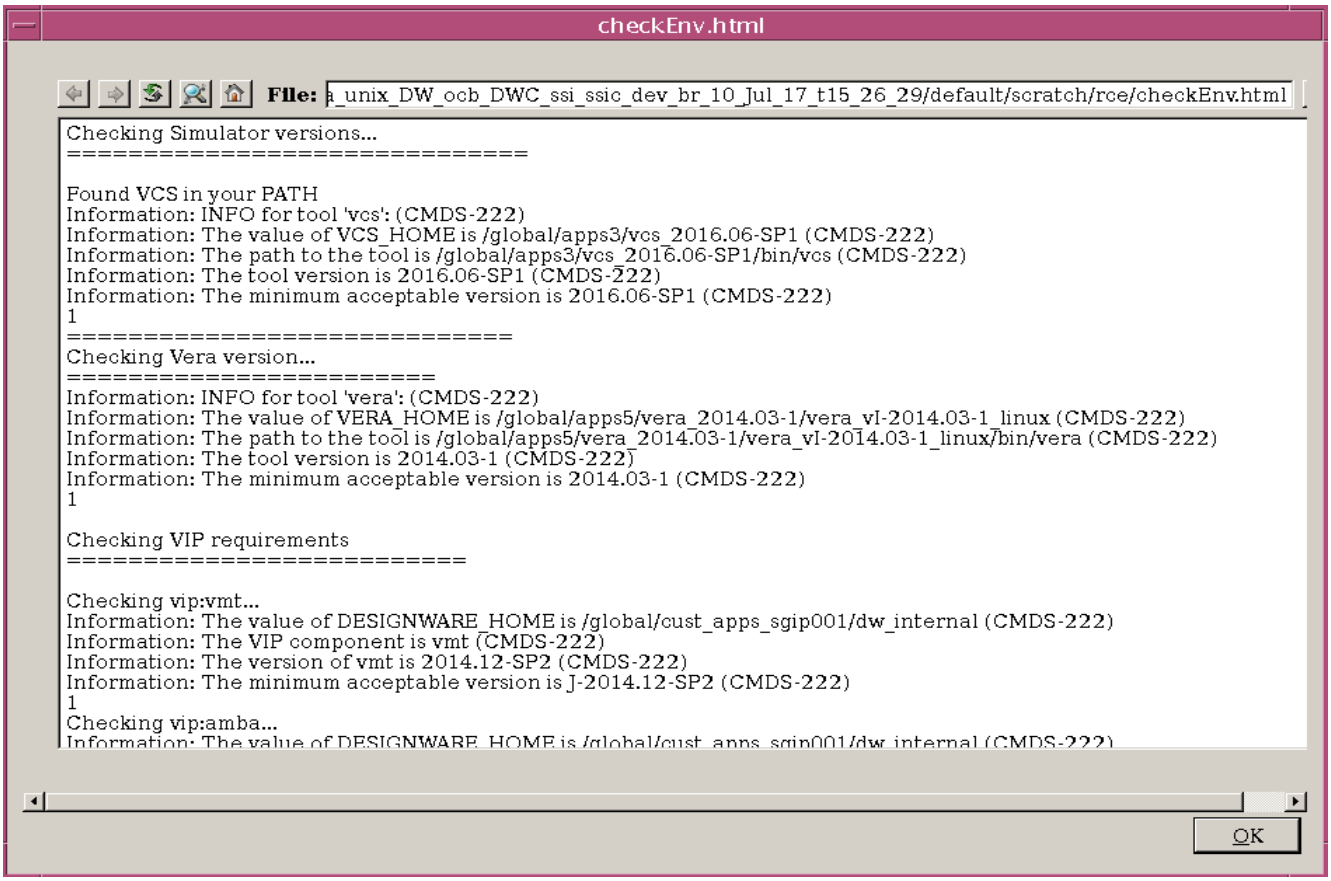
Hint

Here is a common problem that can occur:

- The DWC_ssi IP core is not visible in the initial coreConsultant page. The $DESIGNWARE_HOME environmental variable is not set (or is not pointing to your IP installation directory) in the shell from which coreConsultant was started. For more details, refer to "Configuring the Core Using coreConsultant" on page 14

2. Validate your installation and environment by selecting **Help->Check Environment** from the menu bar. A report window (Figure 2-3) shows the results.

Figure 2-3 Sample Results of the Check Environment Activity

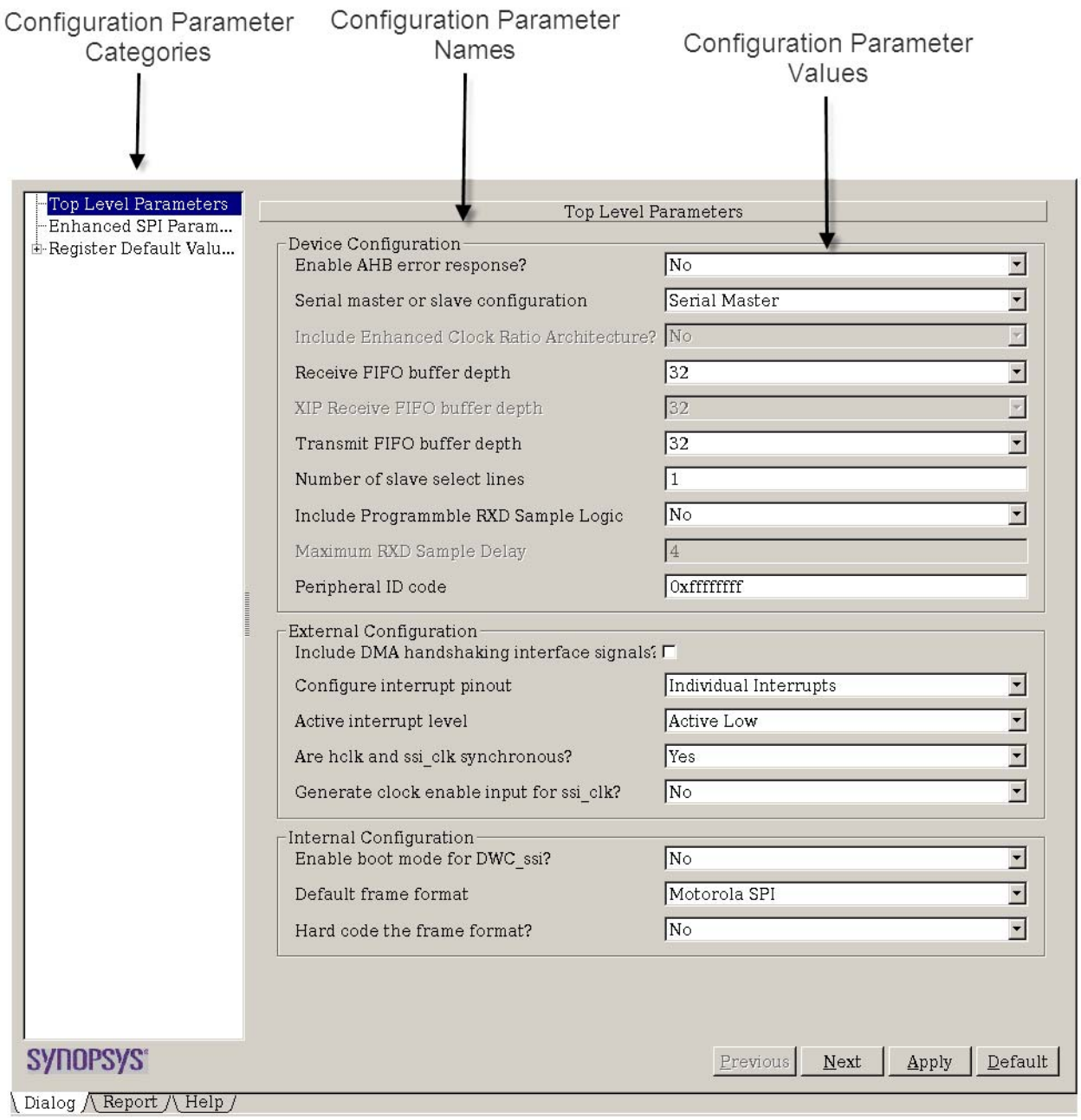


Correct any reported errors by modifying your UNIX environment setup (see "Configuring the Core Using coreConsultant" on page 14) or through the **Edit > Tool Installation Roots** menu.

2.1.4 Configuring the Core

After you have created a workspace, you configure the core and create the RTL using the Create RTL activity dialog boxes as illustrated in Figure 2-4.

Figure 2-4 Specify Configuration Activity



1. **Specify your configuration.**

Select options to enable or disable features.

- You can use default values for an initial simulation and synthesis trial. However, you must select or enter specific values required to implement your design.
- Make sure that you understand the definition of each parameter and change the default value only when it is not suitable for your application.

  You can access detailed information about each parameter by right-clicking on the parameter label and selecting *What's This* or by selecting the Help tab.
- The coreConsultant tool enforces the parameter interdependencies interactively.
- For more information about the configuration parameters, refer to the "Parameters" chapter in the *DesignWare Cores Synchronous Serial Interface Databook*.

Use the **Set Design and File Prefix** activity when you plan to instantiate the core more than once in your design. It is used to create a unique name for each design in your component.

2. **Generate RTL.**

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

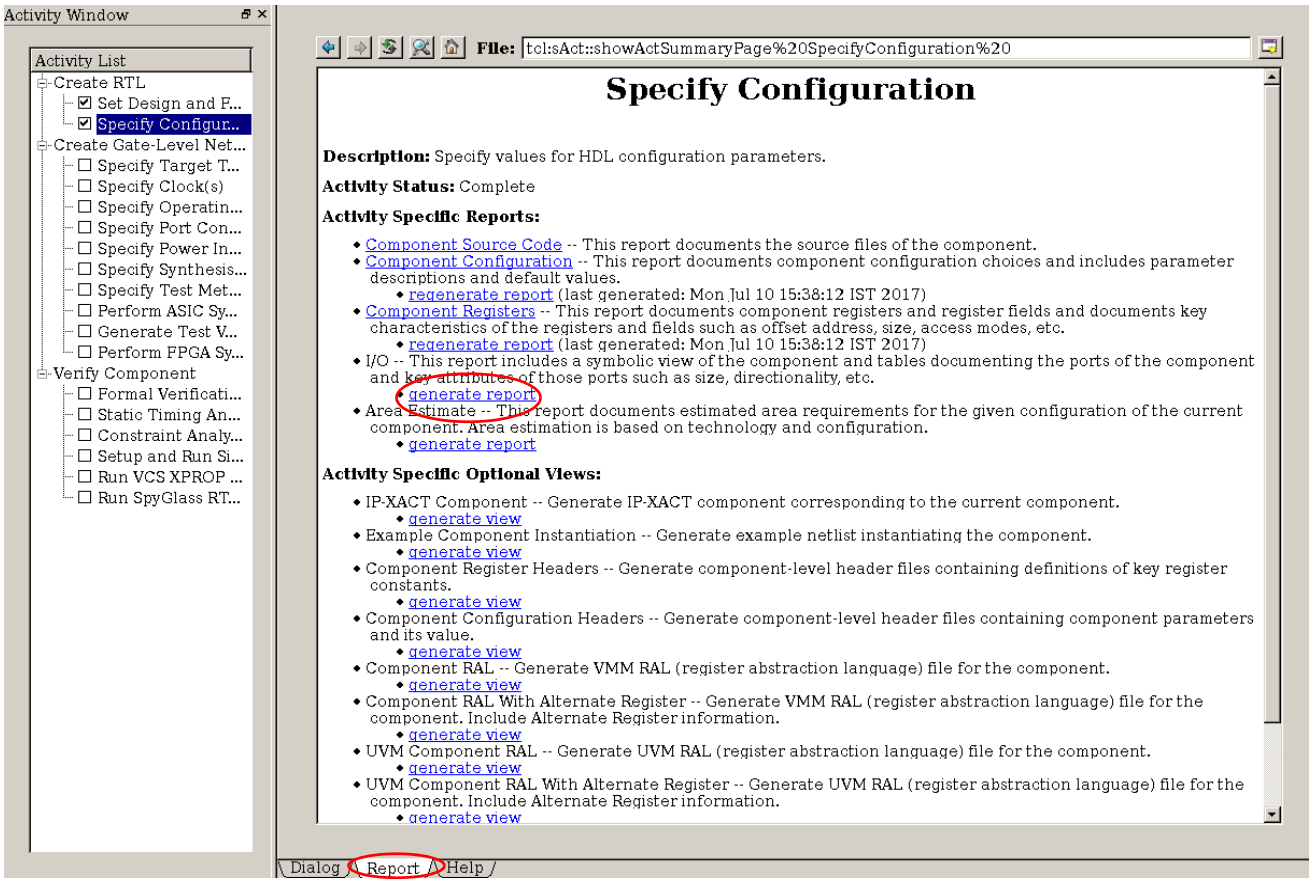
You can return to the Configuration activity to configure the core again (create new RTL) at any time.

Create a batch script so that you can recreate your exact configuration at a later stage in case you delete or overwrite your current workspace. For more details, see Creating a Batch Script.

3. **View reports and generate optional views/reports.**

After you have configured the core, coreConsultant generates several configuration reports. You can access these from the Report tab (shown in Figure 2-5 on page 21).

Figure 2-5 Configuration Reports



Also, you can generate Activity Specific Reports and Activity Specific Optional Views. For more information, refer to "Creating Optional Reports and Views" on page 44. For example, to generate an example component instantiation, click the **generate view** link as indicated in Figure 2-5.

Note

- If any problems occur during or after the Specify Configuration activity, refer to "Troubleshooting" on page 74.
- At this point, you can verify the core ("Simulating the Core" on page 21) or generate a gate-level netlist ("Synthesizing the Core" on page 25).

2.1.5 Simulating the Core

This section shows you how to simulate the DWC_ssi core in the coreConsultant environment. You can simulate the core RTL in the supplied Vera testbench test environment.

Before You Start

Check that you have the latest tool versions installed and your environment variables are set up correctly (see "Configuring the Core Using coreConsultant" on page 14). 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.

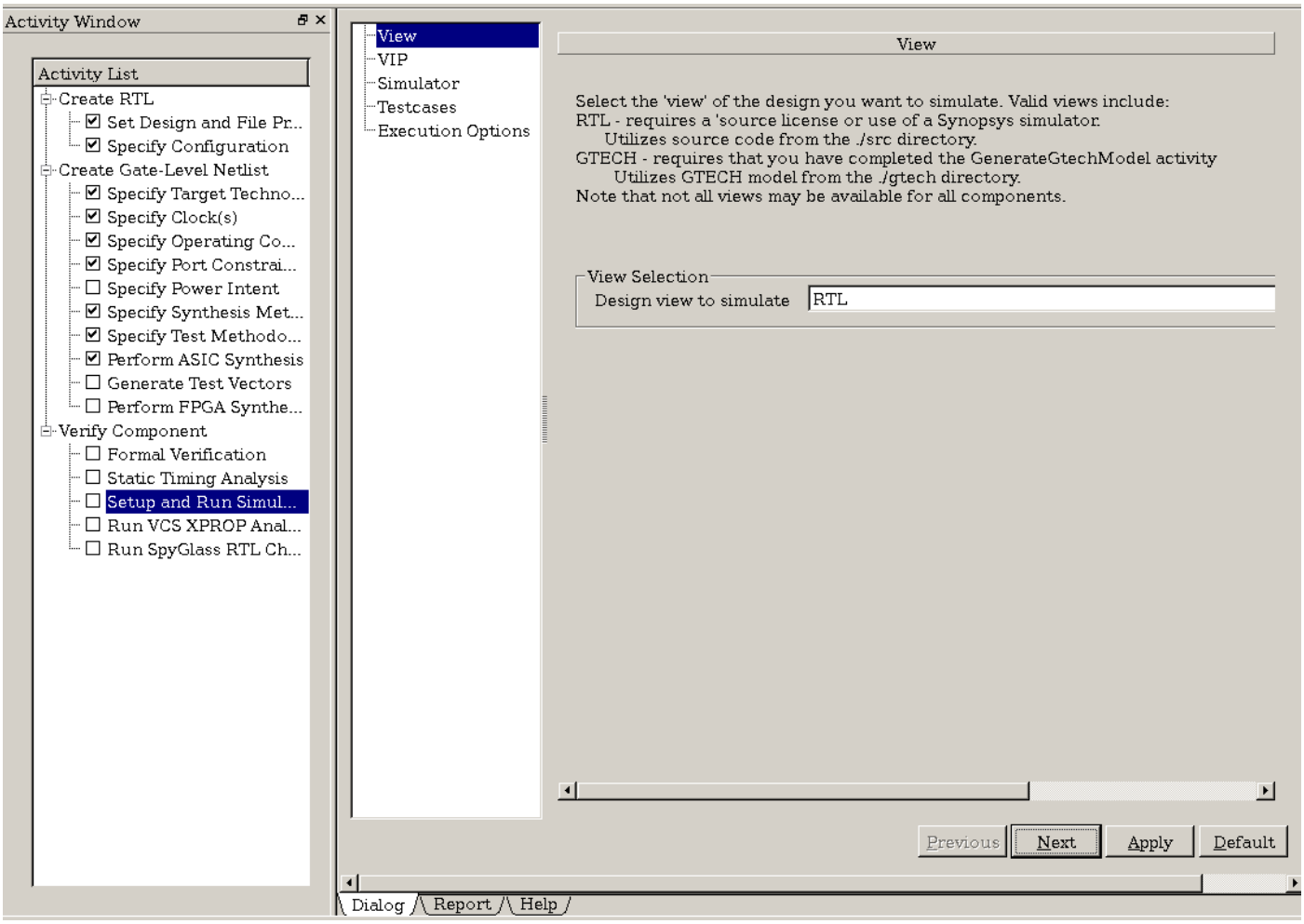
The steps involved in simulating the DWC_ssi core are:

- "Running the Simulation"
- "Checking Simulation Status and Results" on page 23

2.1.5.1 Running the Simulation

The Simulate activity is where you select your simulation options and run the simulation. When the simulation is complete, the results are available in the Report tab of the Simulate dialog box.

Figure 2-6 Setup and Run Simulation Activity



A new workspace has all the default verification values set. You can choose to modify these values based on your application requirements or click **Default** in Figure 2-6 to reset all DWC_ssi verification attributes to their default values.

To modify values based on your application requirements, complete the following steps:

1. In the Verify Component activity, select **Setup and Run Simulations**.

   The View page, shown in Figure 2-6, is displayed.

2. Specify the **Simulation View**.

   a. In View Selection, select the view to simulate:

      - RTL

   b. Click **Next**.

3. Specify the VIP versions in the **VIP** area, and click **Next**. You can keep the default values in these fields.

4. Select the simulator.

   a. In the **Simulator** field, select the simulator you wish to use. Only Synopsys VCS simulator is supported in this release.

   b. In the **Waves Setup** field, select whether you want waveform files to be generated for each test run.

   c. In the **Depth of waves to be recorded**, specify the wave depth.

   d. Click **Next**.

5. Select the testcase in the **Testcase** field.

6. Specify Execution Options.

   a. Select the **Do Not Launch Simulation** check box to generate only a simulation script and not launch the simulation process.

   b. Select one of the following **Run Style** selections.

      You can retain the default value in this field.

   c. Add command-line options to the **Run Style Options** box as required. For options separated by the pipe(|) symbol, include the options within double-quotes as shown in the following example:

      ```
      "os_version="WS5.0|WS6.0",os_distribution=redhat"
      ```

   d. Select the **Send e-mail** check box if you want to receive an e-mail when the Simulation activity is complete, and enter the e-mail address in the **To** field.

   e. Click **Next**.

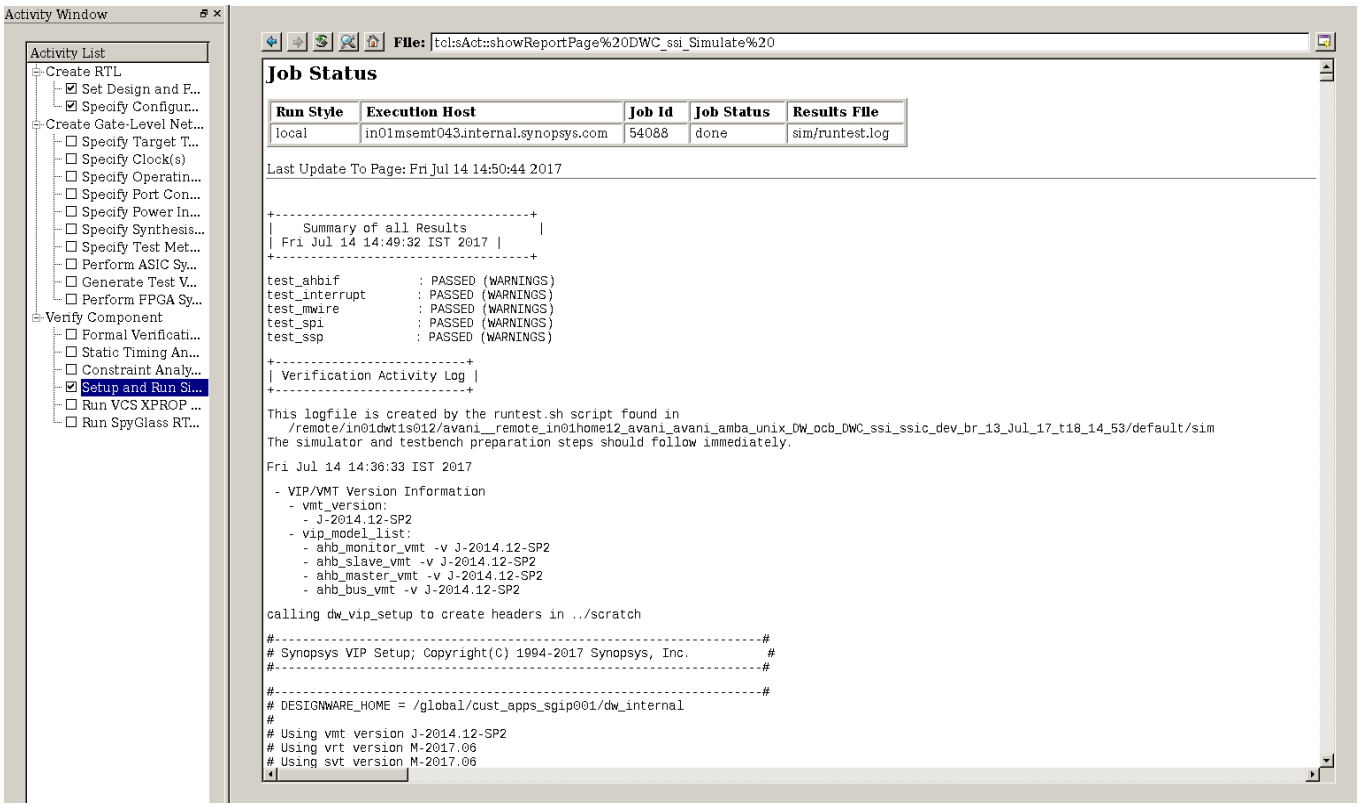
7. Click **Apply** to start the simulation.

2.1.5.2 Checking Simulation Status and Results

To check simulation status and results:

1. Click the **Report** tab.
2. Click the **Setup and Run Simulations Summary** link.

Figure 2-7 Example Simulate Summary Report



Note

- When you select the “LSF/GRD” option for the Run Style in the ‘Execution Options’ of the Simulate dialog box in Figure 2-6 on page 22, the status of the simulation jobs (running or complete) is incorrect. After all of the simulation jobs are submitted to the LSF/GRD queue, the status indicates “complete.” You should use “bjobs/qstatus” to check if all of the jobs are completed.

2.1.5.3 Location of Simulation Files

After simulations are completed, you can also check simulation files in the workspace/sim directory

Table 2-1 *workspace*/sim Contents

File	Description
test.log	Simulation fully detailed test log file
Passed or Failed	Pass or fail notification folder for the simulation
test.result	Test result detail (pass with/without warnings or test fail)
workspace/sim/<*testpattern*>	
test.startsim	Command script to launch simulation
test.sim_command	Simulation Control file

2.1.5.4 Running Simulations from UNIX Shell

After you run a simulation through the coreConsultant GUI, you can re-run the simulation directly from the UNIX command line, as follows:

1. Go to the <workspace>/sim directory:

   ```
   % cd <workspace>/sim
   ```

2. Execute the run.scr script.

You can run the test with waveform dumps by editing the test.sim_command file and uncomment the waveform dump switches for the simulator (the following example is for VCS):

```
+vpdfile+test.vpd
```

```
+define+DUMP_DEPTH=0
```

You can add more required switches to be passed to the simulator by editing the test.sim_command.

You can edit the Makefile and comment out the tests you do not want to include in the simulation.

If you build your own test environment, you can use the `-f ./src/DWC_ssi.lst` option during compilation to read in the file list, including both `DWC_ssi_cc_constants.v` and other RTL files.

2.1.6 Synthesizing the Core

The coreConsultant tool is your interface to Synopsys synthesis tools for ASIC or FPGA synthesis of the core.

If you want to access the synthesis scripts for exporting to your chip database, or if you are using non-Synopsys synthesis tools, refer to "Creating Optional Reports and Views" on page 44.

The topics in this section are as follows:

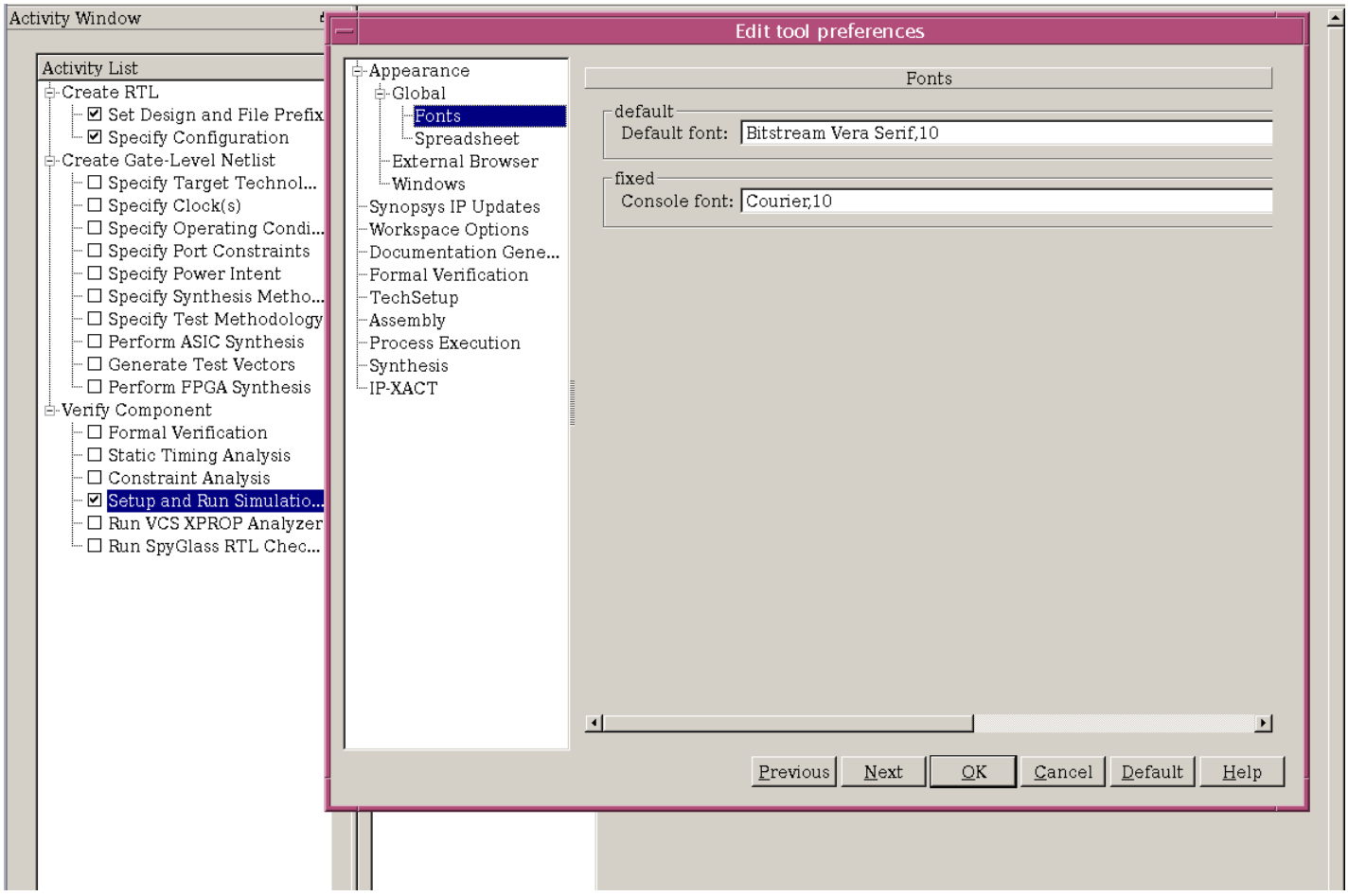
Before You Start

Check that you have the latest tool versions installed and your environment variables set up correctly (see "Configuring the Core Using coreConsultant" on page 14). 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. You can access global synthesis options through **Edit -> Preferences -> Synthesis** menu.

2.1.6.1 Setting Synthesis Preferences

You can set synthesis-specific preferences by choosing the **Edit > Preferences > Synthesis** in coreConsultant a shown in Figure 2-8.

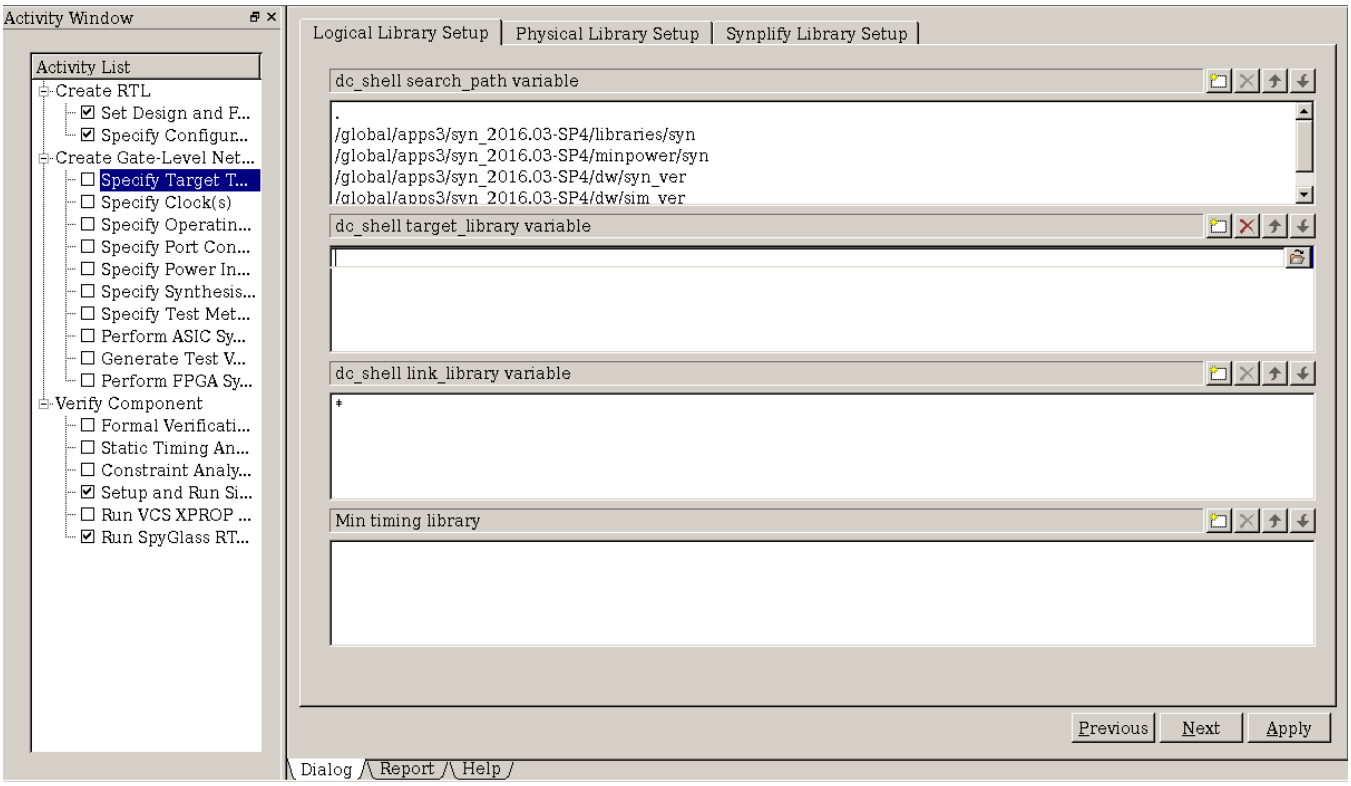
Figure 2-8 Setting Synthesis Preferences



2.1.6.2 Synthesizing Your Core

To synthesize your core, select the ‘Create Gate-Level Netlist’ activity as shown in Figure 2-9.

Figure 2-9 Create Gate-Level Netlist (Synthesis) Activity



2.1.6.3 Synthesizing the Core for an ASIC

When you want to map and synthesize the core to an ASIC using the Synopsys DC tool within coreConsultant, follow the process outlined in this section. Otherwise, you must export the synthesis scripts from coreConsultant as outlined in "Synthesizing to a Device Outside of coreConsultant" on page 49 so that you can synthesize the core in your own design flow.

The topics in this section are as follows:

2.1.6.3.1 Performing ASIC Synthesis

You must configure the core (see "Configuring the Core" on page 19) before starting this task. To synthesize your core, complete the following steps in the **Create Gate-Level Netlist** activity (see Figure 2-10).

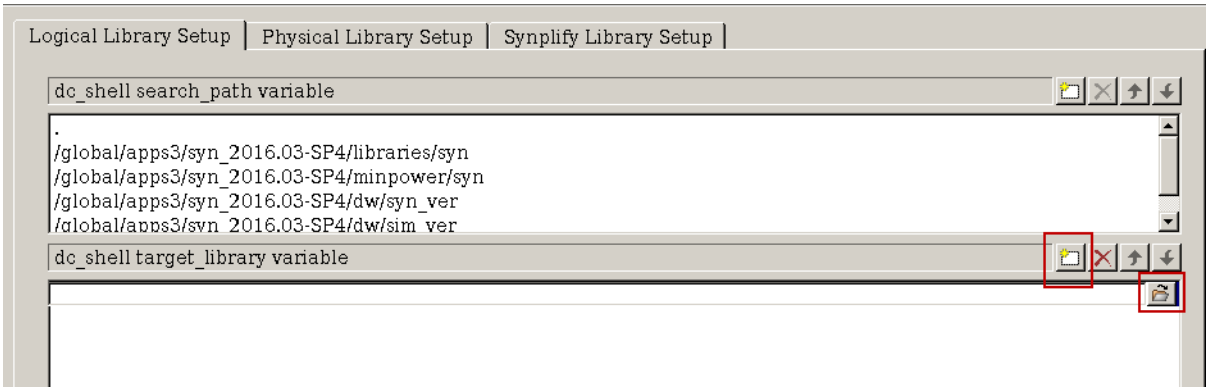
1. **Specify Target Technology.**

   A target library must be specified, otherwise, errors occur in coreConsultant. For ASIC synthesis, use a target technology of .18 microns or less. To add a target technology library, click the folder icon (A in Figure 2-10) and use the navigation tool (B in the figure).

   When you are running Physical Synthesis, then you must specify the physical library under the "**Physical Library Setup**" tab.

The default values for the search_path and link library do not normally need to be changed.

Figure 2-10 Specify Target Technology – Logical Library Setup



2. **Specify Clock(s).**

   The default CycleTime for the input clocks gets populated in the initial screen. You can obtain the values for default CycleTime for all the clocks from coreConsultant. The default values for other clock-related synthesis attributes provide acceptable synthesis results in most libraries.

   To obtain all the clocks, go to "Specify clocks" and note down all the clock names and default cycle time.

3. **Specify Operating Conditions and Wireloads.**

   a. When you do not see a value beside “OperatingConditionsWorst”, select an appropriate value from the drop-down list. If there is no value for this attribute, you may get an error message.

   b. You should also set the various “WireLoad” attributes, unless you are using 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 as shown in Figure 2-11.

Figure 2-11 Specify Operating Conditions and Wireloads Window

Attributes	DWC_ssi
ParentWireLoad	
WireLoadMode	
WireLoadGroup	WireAreaForZero
WireLoadMinBlockSize (libArea)	
WireLoad	
OperatingConditionsWorst	WCCOM
OperatingConditionsBest	

4. **Specify Port Constraints.**

Port input and output delay constraints are specified using actual clocks. However, asynchronous inputs use virtual clock as a reference.

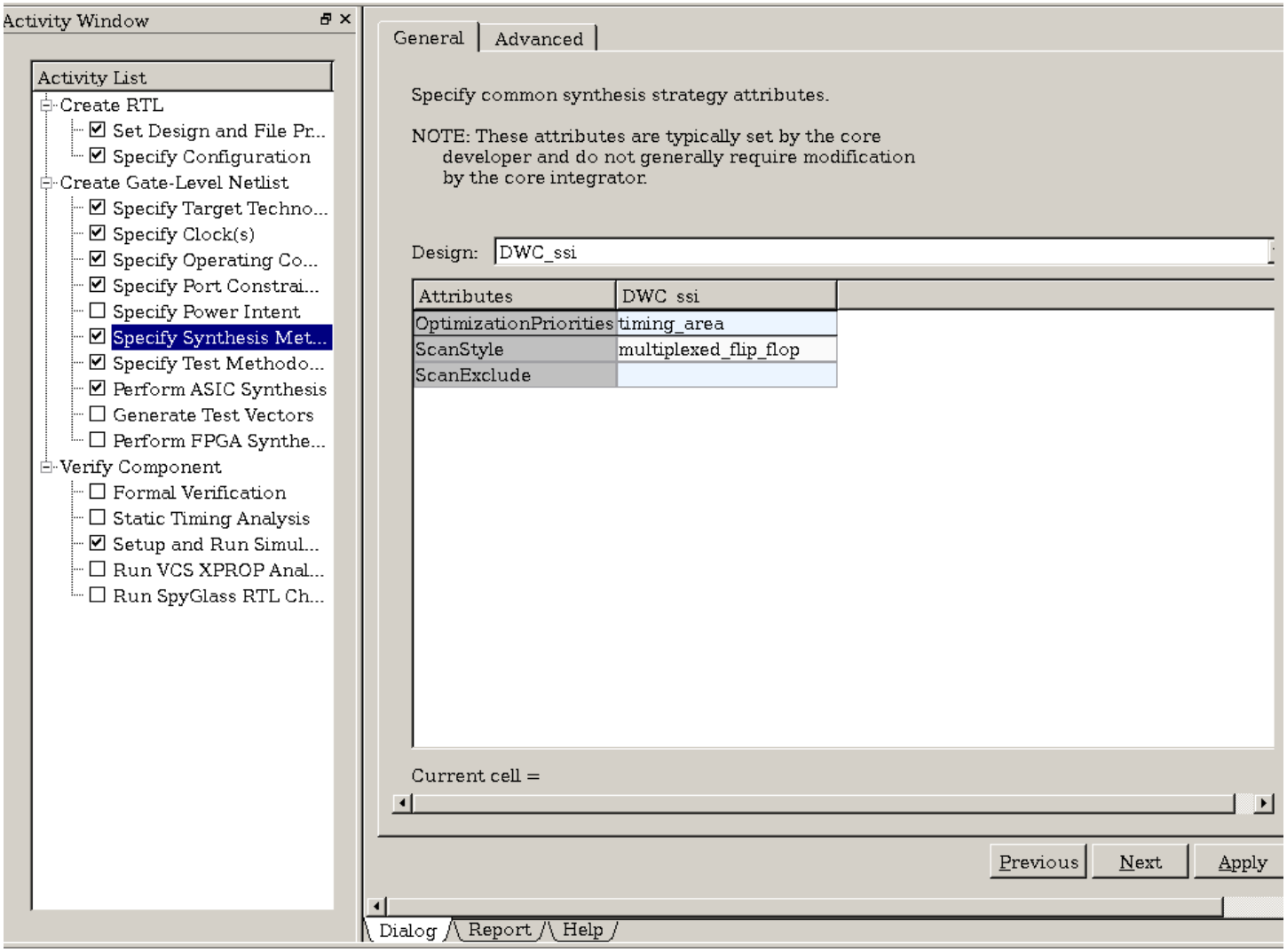
The default input and output delay constraints are specified as a percentage of the clk associated with the port.

5. **Specify Synthesis Methodology.**

In the "Specify Synthesis Methodology" activity, review the synthesis strategy attributes. These attributes are set by the core developer and can be optionally modified by you. The default setting of these attributes provide acceptable synthesis results in most libraries.

To run Physical Synthesis, click on the "Physical Synthesis" tab (Figure 2-12) and specify information in the related fields. The physical synthesis flow uses the Synopsys DC-Topographical engine, which is part of Design Compiler (dc_shell).

Figure 2-12 Specify Synthesis Methodology – Physical Synthesis Tab



6. **Specify Test Methodology.**

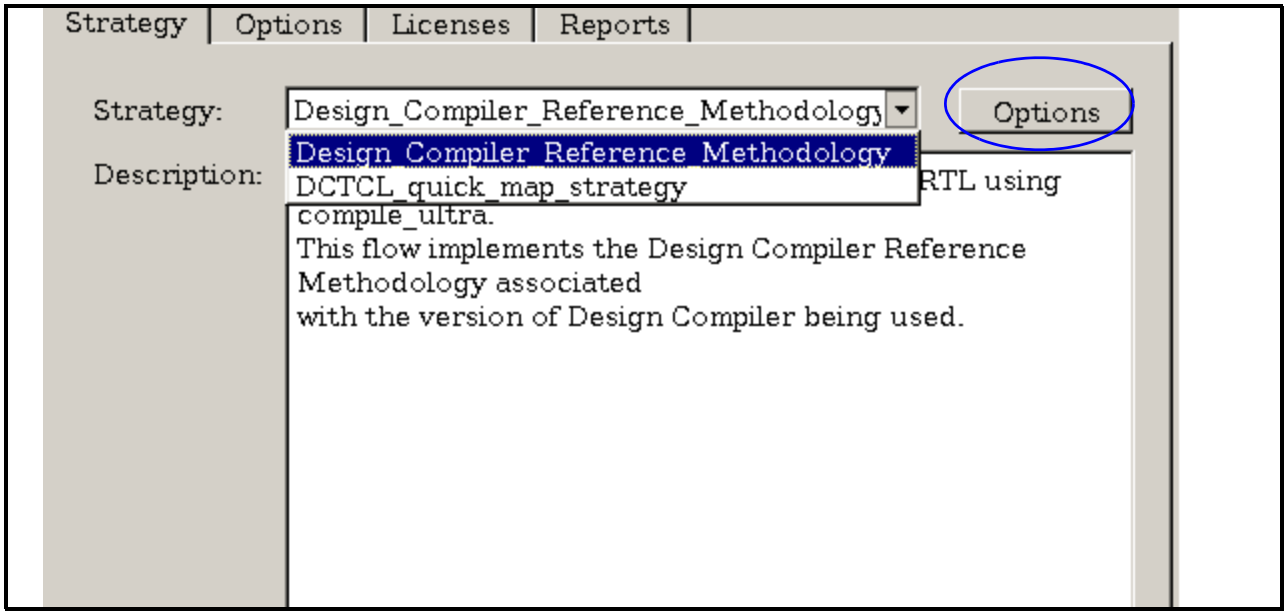
These attributes are set by the core developer and can be optionally modified by you. For more details, refer to "Inserting Design For Test Using Design Compiler" on page 37.

7. **Perform ASIC Synthesis.**

You can select the synthesis strategy on the Strategy tab shown in Figure 2-13 on page 30. The default flow is "Design_Compiler_Reference_Methodology". A detailed explanation of each strategy is provided in the GUI.

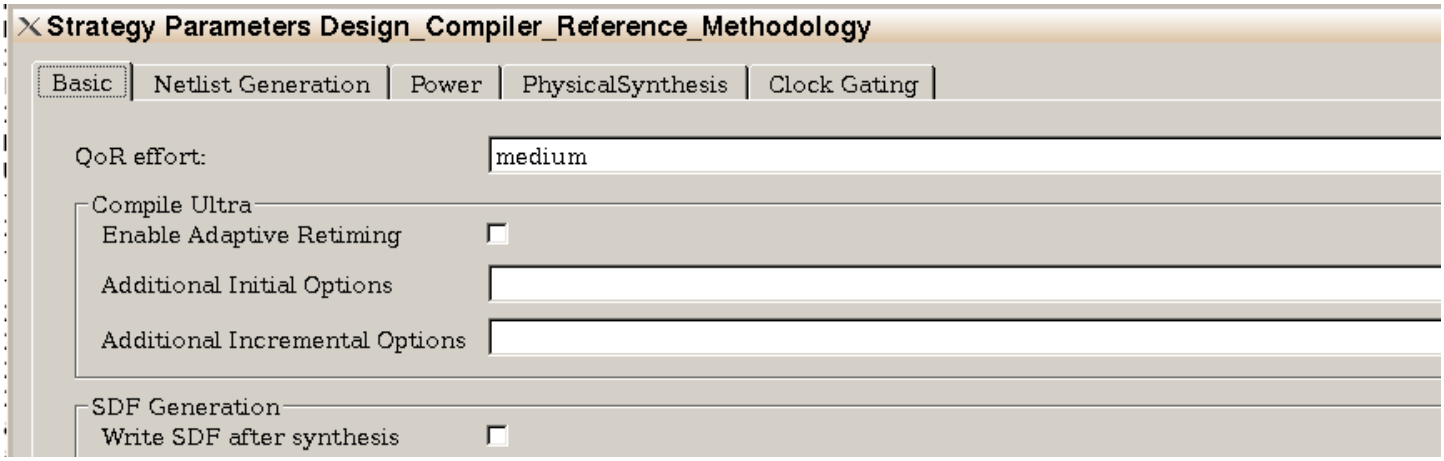
> **Hint** One other strategy that is of interest is the DCTCL_one_pass_compile_ultra, which uses a simple TCL script based on the Synopsys Design Compiler Reference Methodology at https://solvnet.synopsys.com/retrieve/021023.html. This synthesis strategy is quicker and does not generate intermediate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

Figure 2-13 Synthesize Activity -> Strategy Tab



Clicking on the **Options** button (highlighted in Figure 2-13) beside the selected strategy lists a series of other options to specify other options for that particular strategy. More information for all these options is available in the *coreConsultant User Guide* (also see "Help Information" on page 76) or by right-clicking on any dialog text.

Figure 2-14 Synthesize Activity -> Options Button



These are the most important options.

- **Design for Test**

  Here you specify whether to add the -scan option to the initial compile call ("Test Ready Compile") and/or insert DFT circuitry ("Insert Dft"). See "Inserting Design For Test Using Design Compiler" on page 37.

- **Physical Synthesis**

  When you are running Physical Synthesis, tick the "Physical Synthesis" box and complete the other relevant fields. Setting this parameter causes the physical synthesis placement engine to be used during optimization. Net loads and delays are estimated during placement. The flow may be used with a physical library in the .ddc format or Milkyway reference library. Note that no placement data is saved with the ddc or Milkyway database.

8. When you have finished specifying the remaining synthesis options, click **Apply**.

Note If you experience any problems during synthesis, refer to "Troubleshooting" on page 74.

2.1.6.3.2 Checking ASIC Synthesis Results

After you have run synthesis, coreConsultant generates several reports. You can access these by clicking the Report tab in Figure 2-9 on page 27.

All the synthesis results and log files are created under the syn directory in your workspace. 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, incr2, or incr2) depend on the QoR effort that you chose for your synthesis (default is Medium) or whether you chose to insert DFT (see "Inserting Design For Test Using Design Compiler" on page 37):

- Low effort – initial
- Medium effort – incr1

- High effort – incr2

The QoR effort is selected through the **Synthesis -> Options**[1] **-> Basic** tab. There are also options here (amongst many) to:

- Generate reports for all stages[2]
- Generate netlist for all stages

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

Table 2-2 ASIC Synthesis Output Files

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

When you are run Physical Synthesis, the final .ddc file is stored in the <workspace>/syn/final/db directory.

Note 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.

2.1.6.4 Synthesizing the Core for an FPGA

When you want to map and synthesize the core to an FPGA using the Synopsys Synplify tool (only) within coreConsultant, then follow the process outlined in this section. Otherwise, you must export the synthesis scripts from coreConsultant as outlined in "Synthesizing to a Device Outside of coreConsultant" on page 49 so that you can synthesize the core in your own design flow. The topics in this section are as follows:

- "Performing FPGA Synthesis" on page 33
- "Checking FPGA Synthesis Results" on page 35

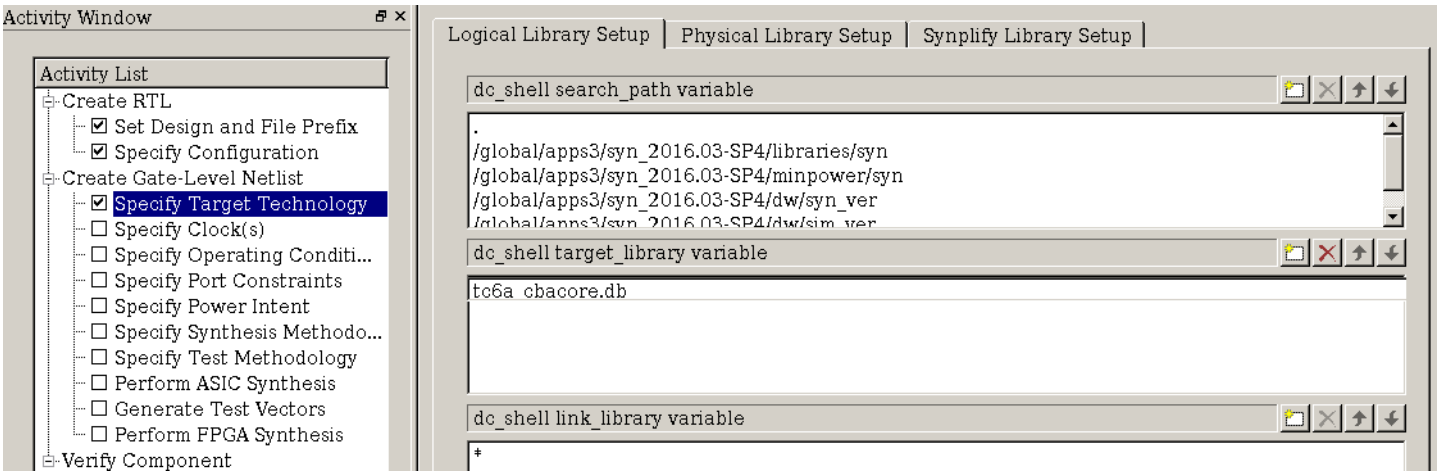
Note

- The PHY is outside the core and is not involved in the coreConsultant synthesis activity.

To synthesize your core, select the "Create Gate-Level Netlist" from the "Activity List" in the coreConsultant GUI Activity Window as shown in Figure 2-15.

1. BUTTON as circled in Figure 2-13 on page 30 and not TAB.
2. 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.

Figure 2-15 Create Gate-Level Netlist (Synthesis) Activity



2.1.6.4.1 Performing FPGA Synthesis

Attention You must first define the location of your Synplify tool installation (**Edit -> Tool Installation roots -> Synplify FPGA** menu) or you cannot proceed with the following steps.

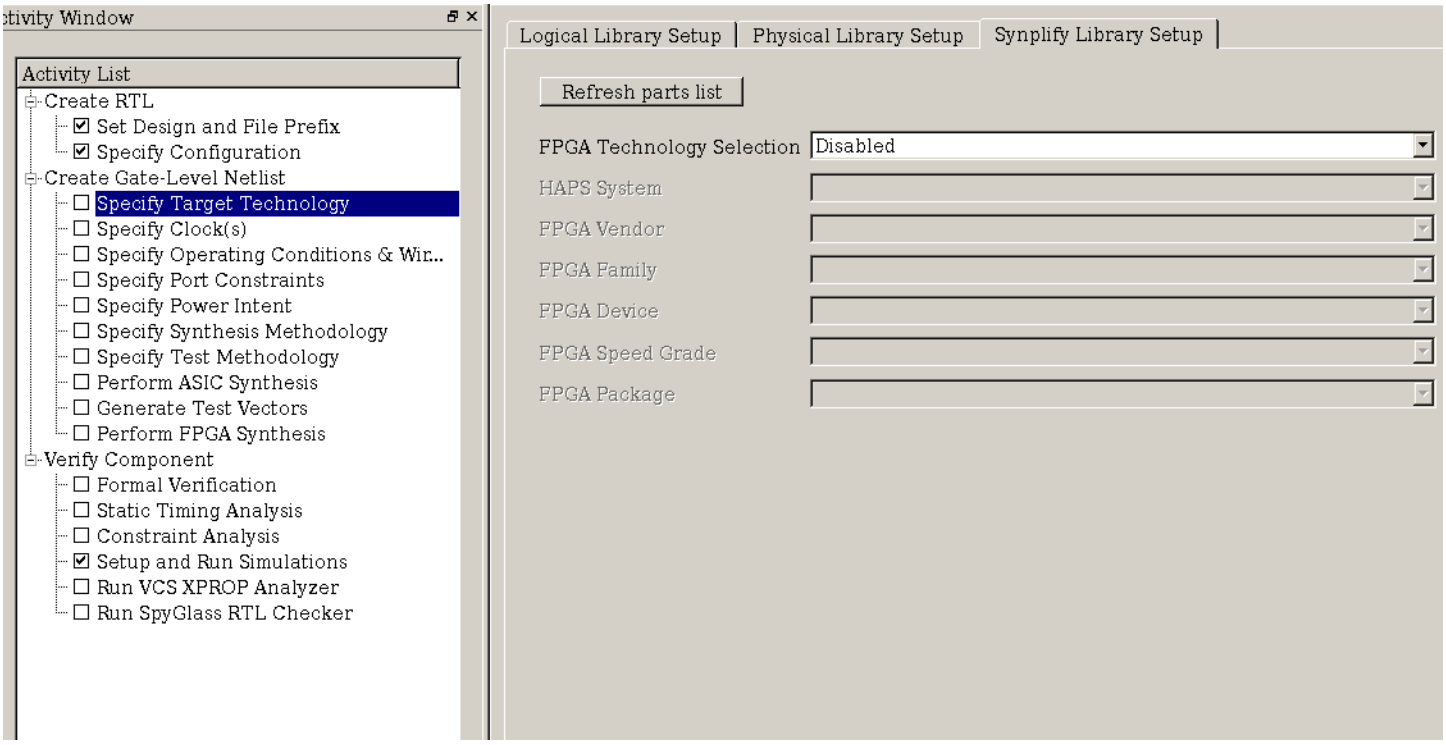
You must configure the core (see "Configuring the Core" on page 19) before starting this task. To synthesize your core, complete the following steps in the "Create Gate-Level Netlist" activity:

1. **Specify Target Technology.**

   Specify your FPGA device in the window under the "Synplicity Library Setup" tab.

   Click on the "FPGA Technology Selection" drop-down list (circled in Figure 2-16), choose **Select Vendor/Family/Device**, and select your target FPGA vendor, family, device, speed, and package.

Figure 2-16 Specify Target Technology Window



Hint For more information about running synthesis for an FPGA device, refer to the coreConsultant User Guide (see "Help Information" on page 76 of Appendix A, "Additional coreConsultant Information").

1. **Specify Clocks (in the Specify Clock Activity).**

   The default CycleTime for the input clocks is 8ns to match the default clock rate configuration of 125 MHz for FPGAs. The default set of the other clock-related synthesis attributes provide acceptable synthesis results for most typical devices.

2. **Specify Operating Conditions and Wireloads.**

   This step is not required for FPGA synthesis.

3. **Specify Port Constraints.**

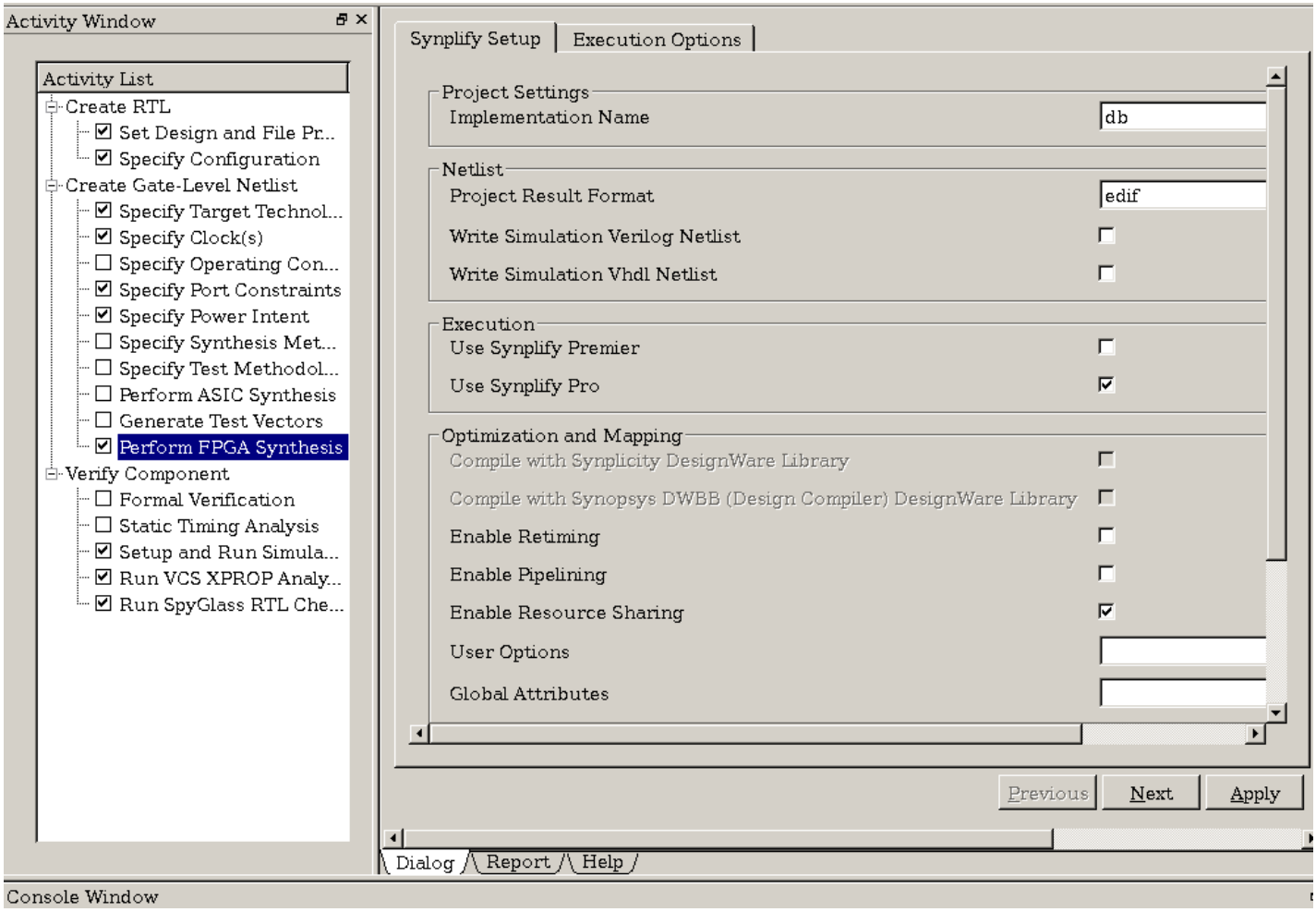
   Port input and output delay constraints are specified using actual clocks. However, asynchronous inputs use virtual clock as a reference.

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

4. **Perform FPGA synthesis.**

   Click **Apply** in **Figure 2-17** on page 35.

Figure 2-17 Perform FPGA Synthesis Activity



For more information regarding all of these options, refer to the *coreConsultant User Guide* (see also "Help Information" on page 76 in Appendix A, "Additional coreConsultant Information").

2.1.6.4.2 Checking FPGA Synthesis Results

After you have started synthesis, coreConsultant generates several reports. You can access these by clicking the Report tab in Figure 2-9 on page 27.

All the synthesis results and log files are created under the syn/synplify directory in your workspace. The final log file is <workspace>/syn/run.log.

Table 2-3 includes the other files of interest that are generated.

Table 2-3 FPGA Synthesis Output Files

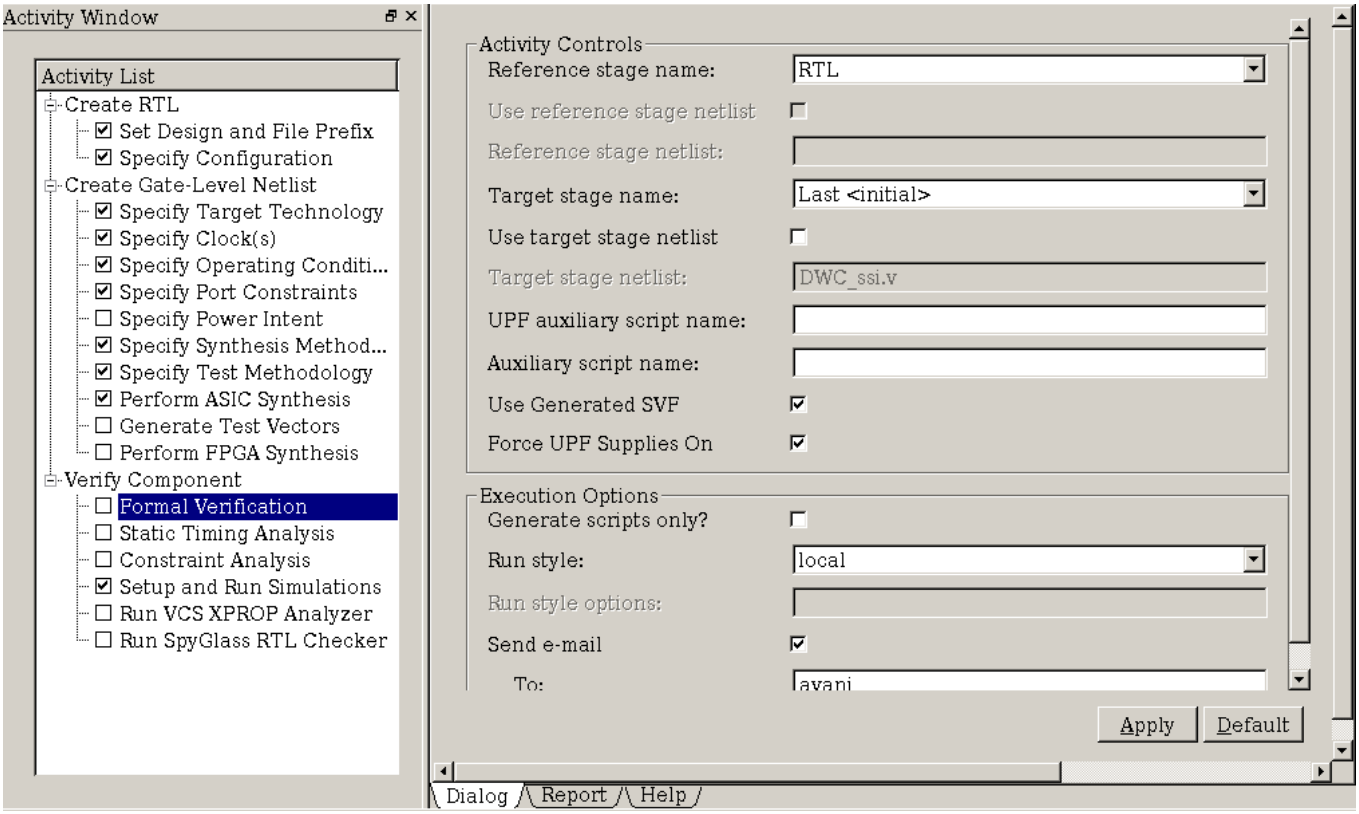
Files	Purpose
./syn/synplify/db/DWC_ssi.*	Synopsys database files (gate level) that can be into Synplify for further synthesis, if desired.
./syn/synplify/report/*.*	Synthesis report files.

2.1.7 Performing Formal Verification

You use the Formal Verification activity (in Figure 2-18) for RTL-to-gate comparisons in coreConsultant. This activity is not applicable to FPGA implementations.

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. For more information, refer to the following site: http://www.synopsys.com/Tools/Verification/FormalEquivalence/Pages/default.aspx

Figure 2-18 Formal Verification Activity



Choose **Formal Verification** from the Activity List and complete the following fields:

- **Reference stage name**: RTL.
- **Target stage name**: Select last synthesis stage from the pull-down menu.
- **UPF auxiliary script name:** Provide a script to be used for UPF flow. It can be used to specify the retention register models that match in both the reference and implementation designs.
- **Auxiliary script name:** Specify the name of any auxiliary fm_shell script that you want to execute before the fm_shell verify command. For example, you may want to insert commands to set or remove compare points.

- **Generate scripts only:** When selected, coreConsultant generates the necessary scripts for formal verification in the workspace but it does not execute them. You can invoke these manually.
- **Use Generated SVF**: Enabled.

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.

2.1.8 Inserting Design For Test Using Design Compiler

This activity is not applicable to FPGA implementations.

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

1. Check that the core input clocks are selected as "Test Clock" in the "Specify Clocks" window. This is normally selected by default. See page 28.
2. 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".
3. 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 (Figure 2-13 on page 30). 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 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.

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

2.1.9 Running Automatic Test Pattern Generation using TetraMax

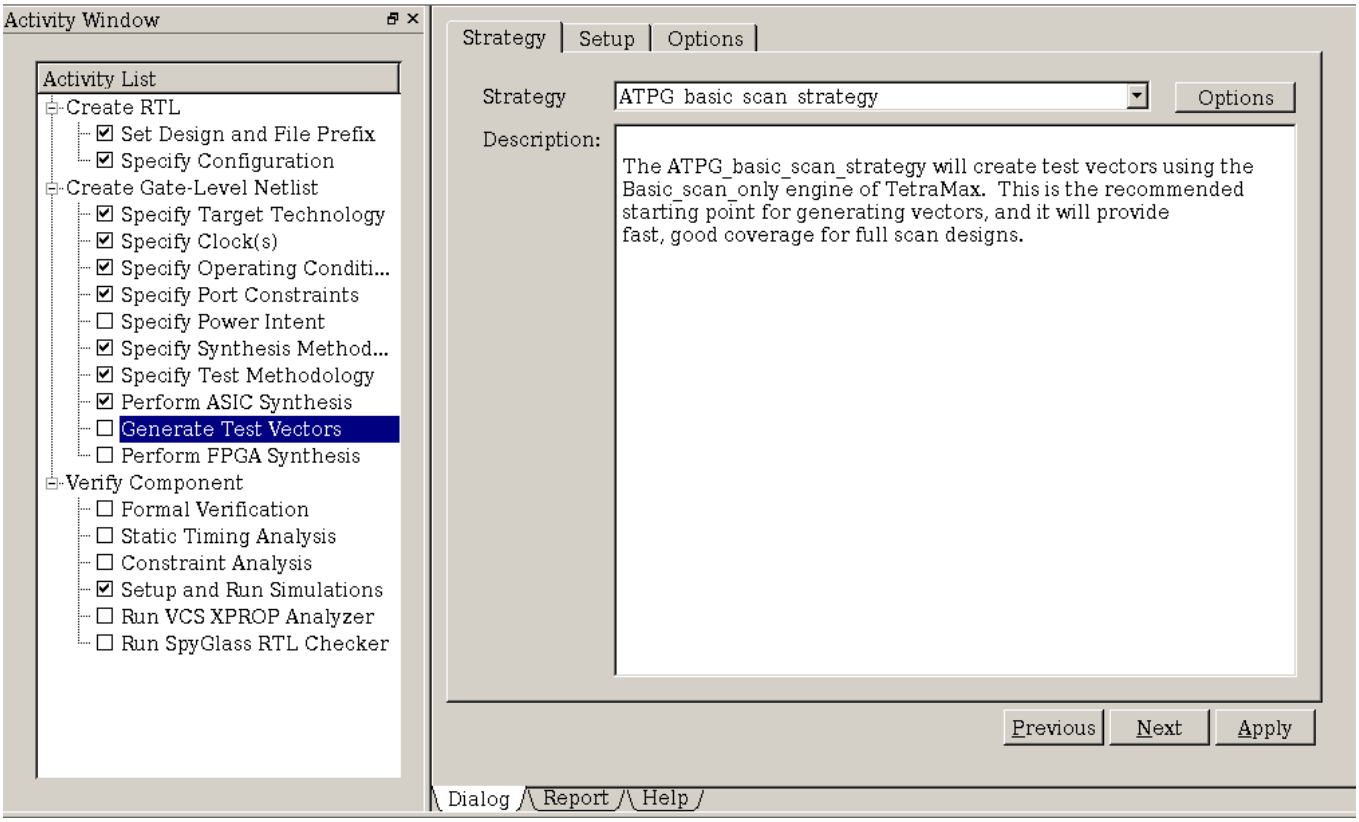
You perform automatic test pattern generation (ATPG) by using the Synopsys TetraMax tool. This activity is not applicable to FPGA implementations.

Attention

- ATPG is an activity which takes the output of synthesis as input.
- Synthesis must be completed (with DFT as described in "Inserting Design For Test Using Design Compiler" on page 37) before a setup can exist in the ATPG window.

To run ATPG, you must complete the Generate Test Vectors activity (Figure 2-19) in coreConsultant.

Figure 2-19 Generate Test Vectors Activity



After you have synthesized the core with the Insert Dft option (see "Inserting Design For Test Using Design Compiler" on page 37), select the **Generate Test Vectors -> Setup** tab and complete the following fields in the dialog box.

- **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 will be created in your workspace directory. For example, *atpg*.
- **Test Libraries**: Specify the library Verilog file name(s).

  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 be each standard cell instantiated.
- **Test Pattern Format**: Specify your test format.

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

The following files in `<workspace>/syn/<Output stage>` can help you understand the ATPG flow:

- `atpg/script/DWC_ssi.tcl`
- `Makefile`

For more information about running ATPG, refer to the *coreConsultant User Guide* (also see "Help Information" on page 76).

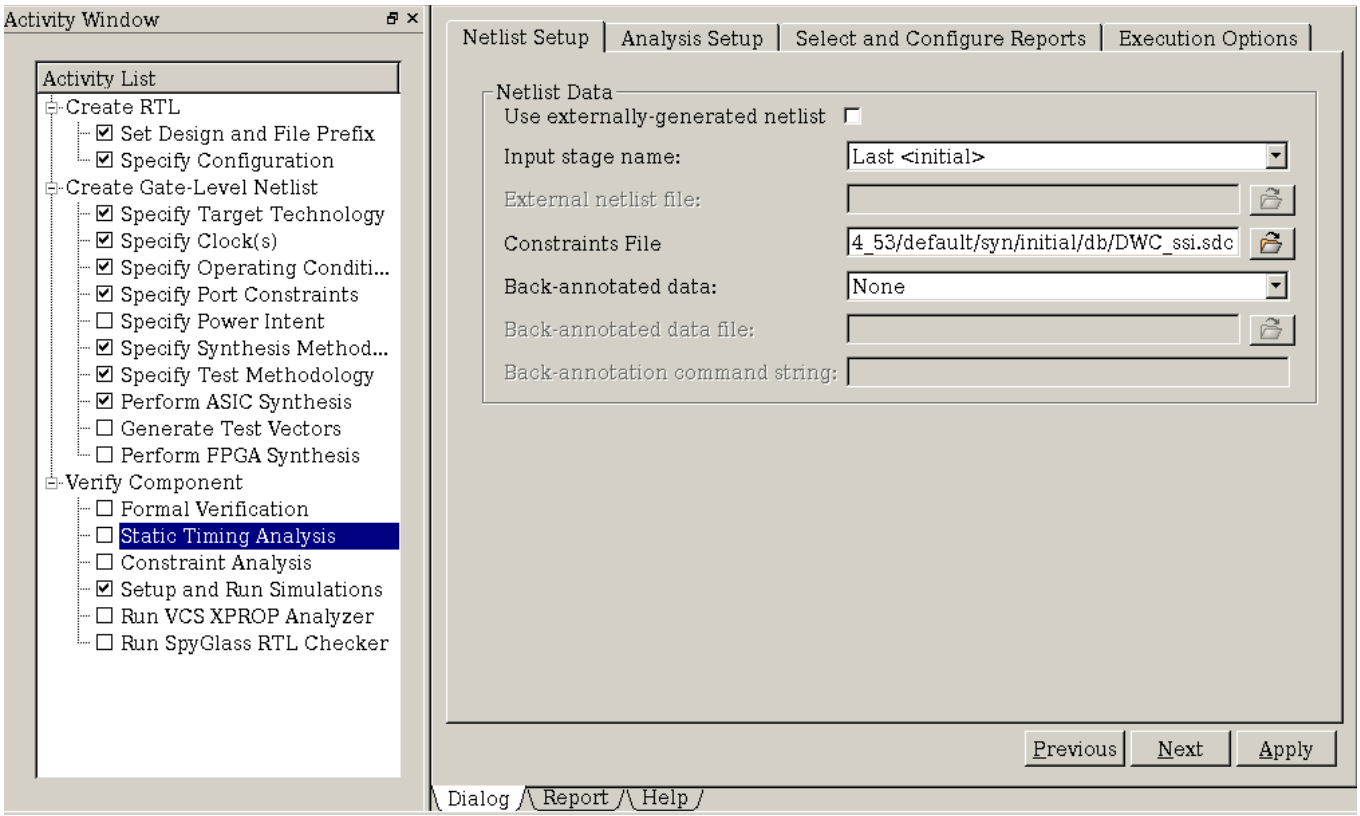
2.1.10 Running Static Timing Analysis using PrimeTime

You perform static timing analysis (STA) by using the Synopsys PrimeTime tool. This activity is not applicable to FPGA implementations.

The STA 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 for back-annotation.

You must first complete the "Performing ASIC Synthesis" on page 27 step or have an externally generated netlist available before starting this task. To perform STA, select the **Static Timing Analysis** activity.

Figure 2-20 Static Timing Analysis Activity



To run STA, complete the following set-up options:

1. **Netlist Setup**

   Click the Netlist Setup tab and complete the following fields:

- **Input stage name:** This field should be automatically filled in by coreConsultant if you have successfully completed ASIC Synthesis. Otherwise, specify the last synthesis stage from the pulldown menu. Typically it is Last<initial>.
- **Constraints file:** This field should be automatically filled in by coreConsultant if you have successfully completed ASIC Synthesis.

2. **Analysis Setup**

   Next, click on the Analysis Setup tab, which allows you to select various STA setup options. You can accept most of the default settings here.

3. **Select and Configure Reports**

   In this tab, you can choose what reports you would like PrimeTime to produce. You can accept most of the default settings here.

4. Click **Apply**.

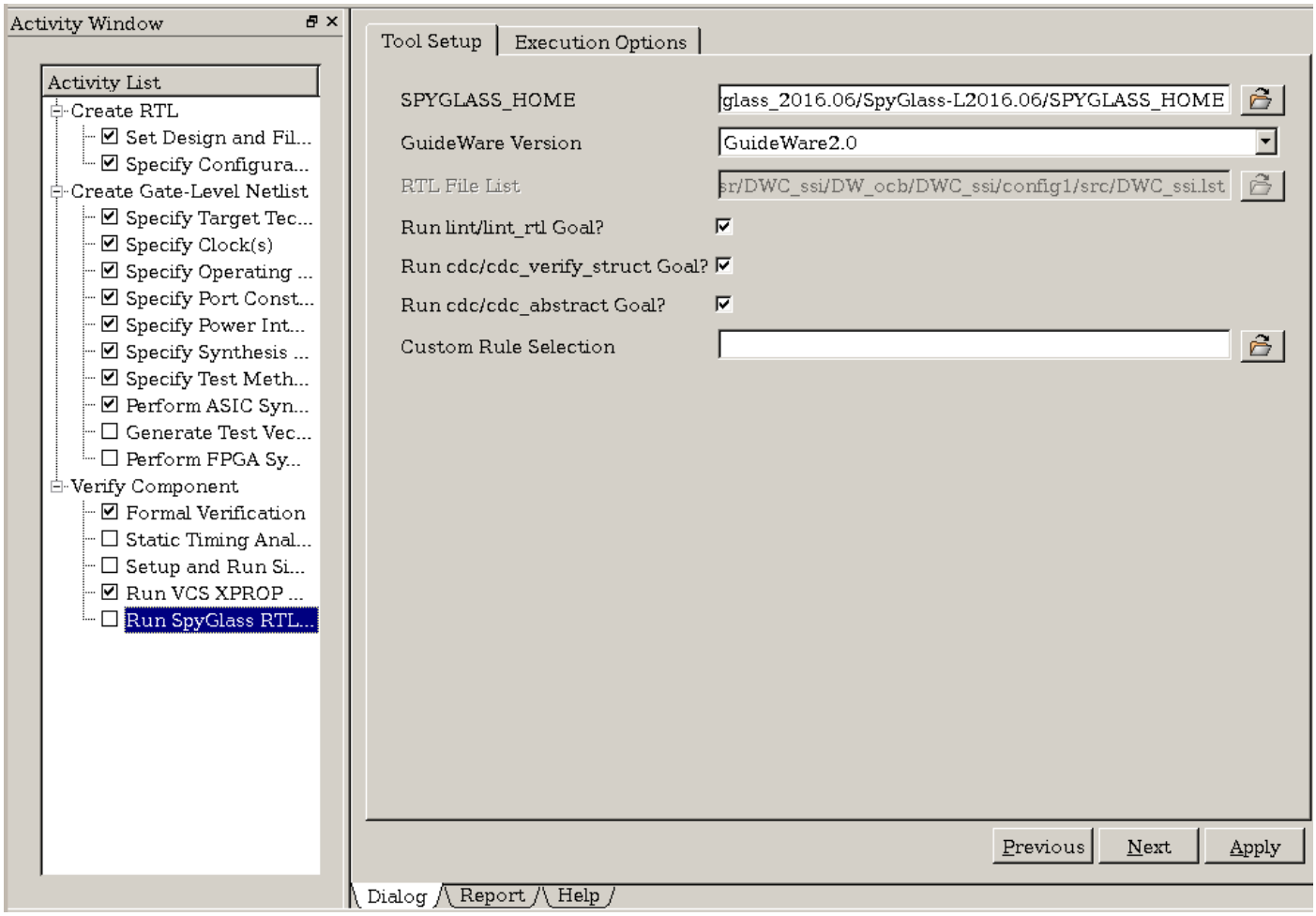
The STA results are saved in the <workspace>/syn/final/report/sta directory.

2.1.11 Running SpyGlass® Lint and SpyGlass® CDC

This section discusses the procedure to run SpyGlass Lint and SpyGlass CDC.

Figure 2-21 shows the coreConsultant GUI in which you run Lint and CDC goals.

Figure 2-21 SpyGlass Options in coreConsultant

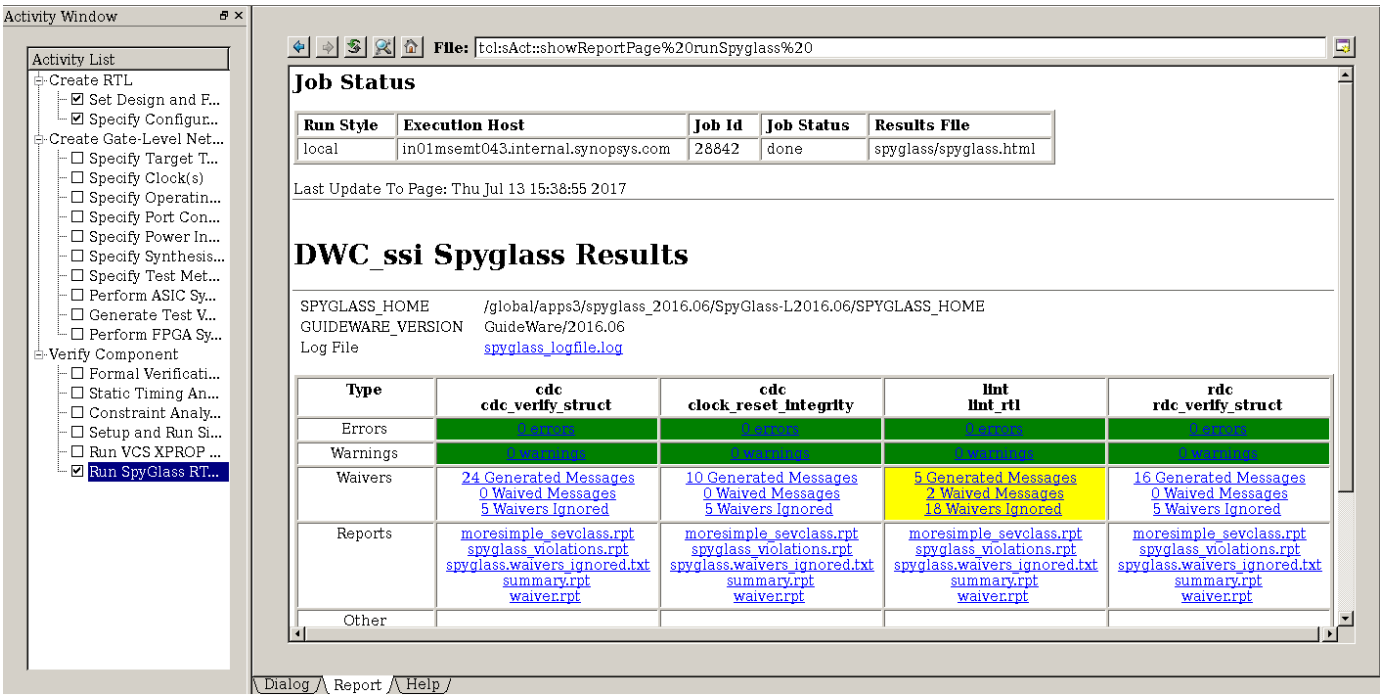


The SpyGlass flow in coreConsultant runs Guideware 2.0 rules for block/rtl_handoff. Within the block/rtl_handoff, only lint/lint_rtl and cdc/cdc_verify_struct goals are run.

In Figure 2-21, select the type of run goals. You can select either Lint run goal or CDC run goal, or both Lint and CDC run goals. By default, both Lint and CDC are selected.

When the Lint and/or CDC is run, the results are available in the Report tab. Errors (if any) are displayed with a red colored cell and warnings (if any) are displayed in yellow colored cell, as shown in Figure 2-22.

Figure 2-22 coreConsultant SpyGlass Report Summary



2.1.11.1 Fixed Settings

The settings are fixed (hardcoded) when you run SpyGlass in coreConsultant.

2.1.11.2 SpyGlass Lint

Table 2-4 lists the SpyGlass Lint waiver files that are used by the coreConsultant tool.

Table 2-4 Waiver Files for SypGlass Lint

File Name	Description
<configured_workspace>/spyglass/spyglass_design_specific_waivers.swl	These are DWC_ssi design-specific rule waivers. The reason for each of the waivers are included as comments in the file.
<configured_workspace>/spyglass/spyglass_engineering_council_waivers.swl	This file contains rules that Synopsys waives for its IPs.

2.1.11.3 SpyGlass CDC

To define the SpyGlass CDC constraints, it is important to understand the reset and clock logic used in DWC_ssi.

2.1.11.3.1 CDC Files

Table 2-5 summarizes files for SpyGlass CDC used by coreConsultant.

Table 2-5 Waiver and Constraint Files for SpyGlass CDC

File Name	Description
<configured_workspace>/spyglass/manual.sgdc	These are the constraints pertaining to a given mode.
<configured_workspace>/spyglass/ports.sgdc	These are the list of I/O signals and their respective clocks.
<configured_workspace>/spyglass/spyglass_design_specific_waivers.swl	These are DWC_ssi design-specific rule waivers. The reason for each of the waivers are included as comments in the file.
<configured_workspace>/spyglass/spyglass_engineering_council_waivers.swl	These are rules that Synopsys waives for its IPs.

2.1.11.3.2 CDC Path Debug Using the SpyGlass GUI

For debugging the CDC path, it is necessary to run SpyGlass in interactive mode in the configured workspace. To invoke the SpyGlass GUI and to run CDC, complete the following steps:

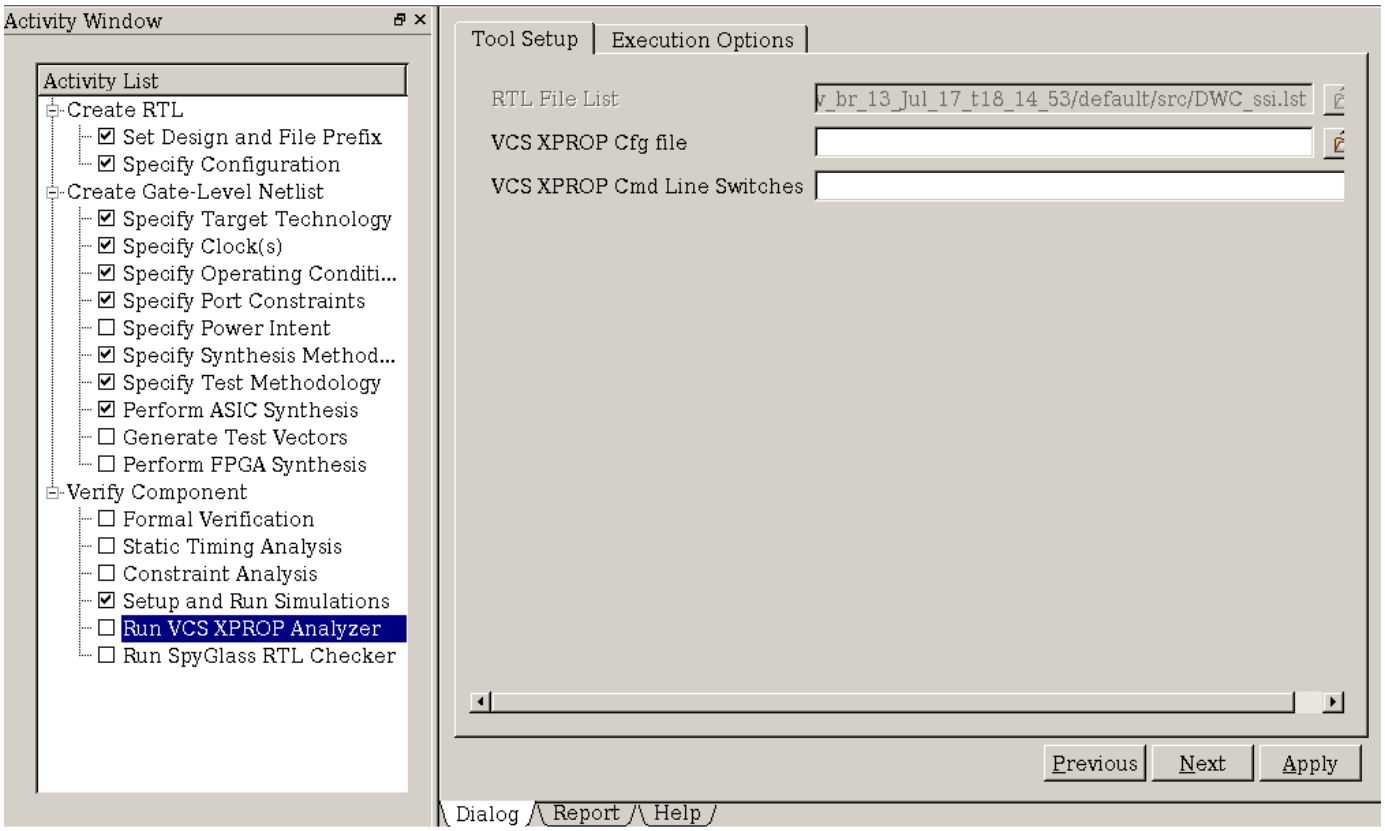
1. Go to the <*configured_workspace*>/spyglass directory.
2. Issue `./sh.spyglass` to start the SpyGlass GUI or issue `./sh.spyglass -batch` to start the SpyGlass in batch mode.
3. In the SpyGlass GUI, the Goal Setup window opens by default.
4. Uncheck the lint_rtl option and click the **Selected Goal (s)** button.
5. After the CDC run is complete, the Analyze Results window displays the results.
6. Navigate to and select the relevant errors to open a schematic for analysis.

2.1.12 Running VCS XPROP Analyzer

This section discusses the procedure to run the VCS XPROP analyzer activity.

Figure 2-23 shows the coreConsultant GUI in which you run the VCS XPROP analyzer.

Figure 2-23 VCS Xprop Option in coreConsultant



This activity runs the XPROP analysis on the configured RTL files. It checks the code for any potential instrumentation issues and reports the same.

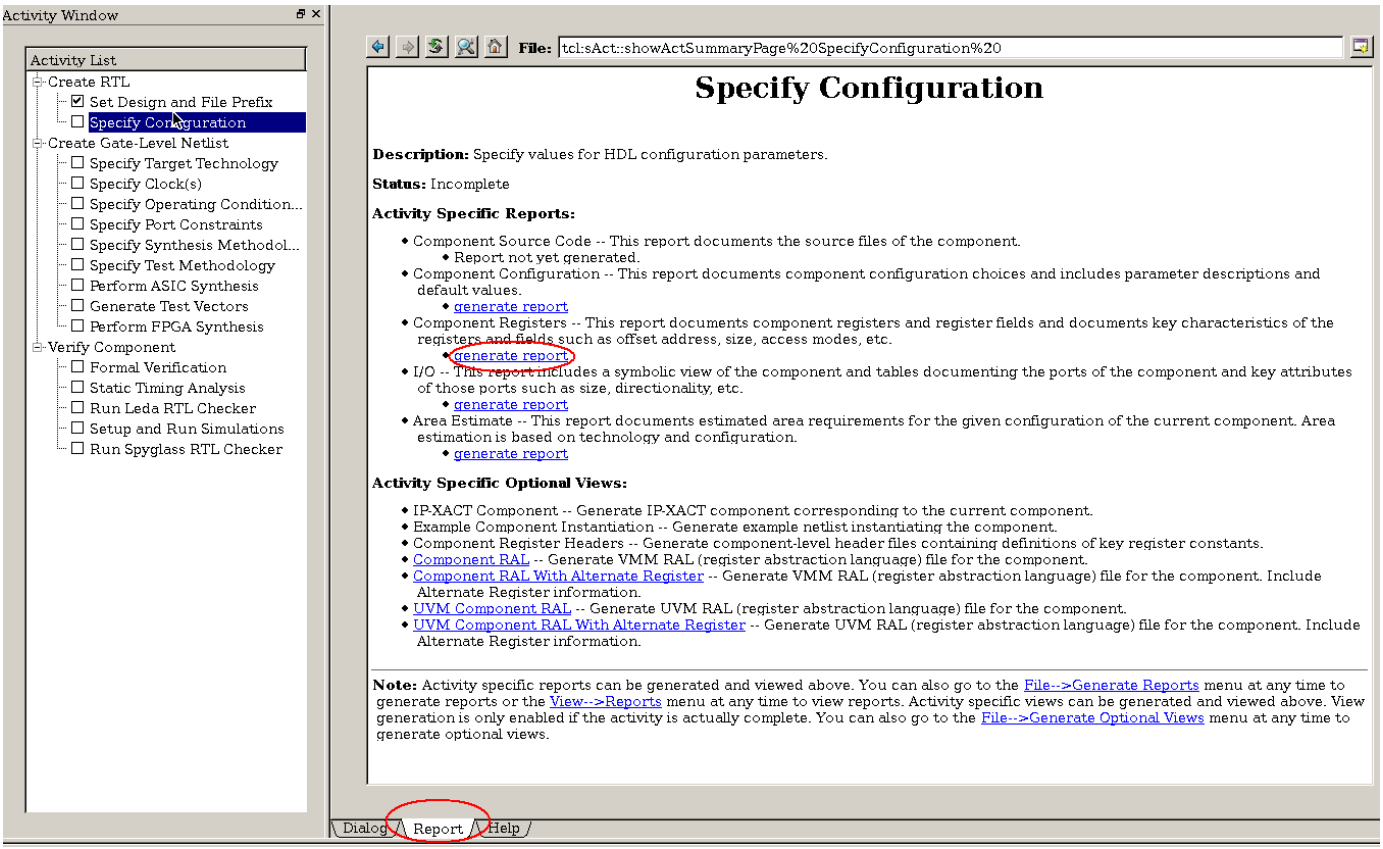
From the "Tool Setup" tab in Figure 2-23, you can pass any command-line switches that can be supplied in addition to what is considered by default.

2.1.13 Creating Optional Reports and Views

After configuring the core, 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 2-24, in the **Specify Configuration** activity.

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

Figure 2-24 Generating Activity-Specific Reports and Views



2.1.13.1 Activity-Specific Reports and Views

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

Table 2-6 describes the activity-specific optional reports.

Table 2-6 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 2-7 describes the activity-specific optional views.

Table 2-7 Activity-Specific Optional Views

Optional Views	File Name (in <workspace>/export)	Description
IP-XACT Component	DWC_ssi.xml	Generates IP-XACT component corresponding to the current component.
Example Component Instantiation	DWC_ssi_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.
Component RAL	DWC_ssi.ralf	Generates RAL (register abstraction language) file for the component.

Note To automatically generate these reports or views, select appropriate check boxes in the **File > Generate Reports** or **File > Generate Optional Views** dialog box.

2.1.13.2 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.

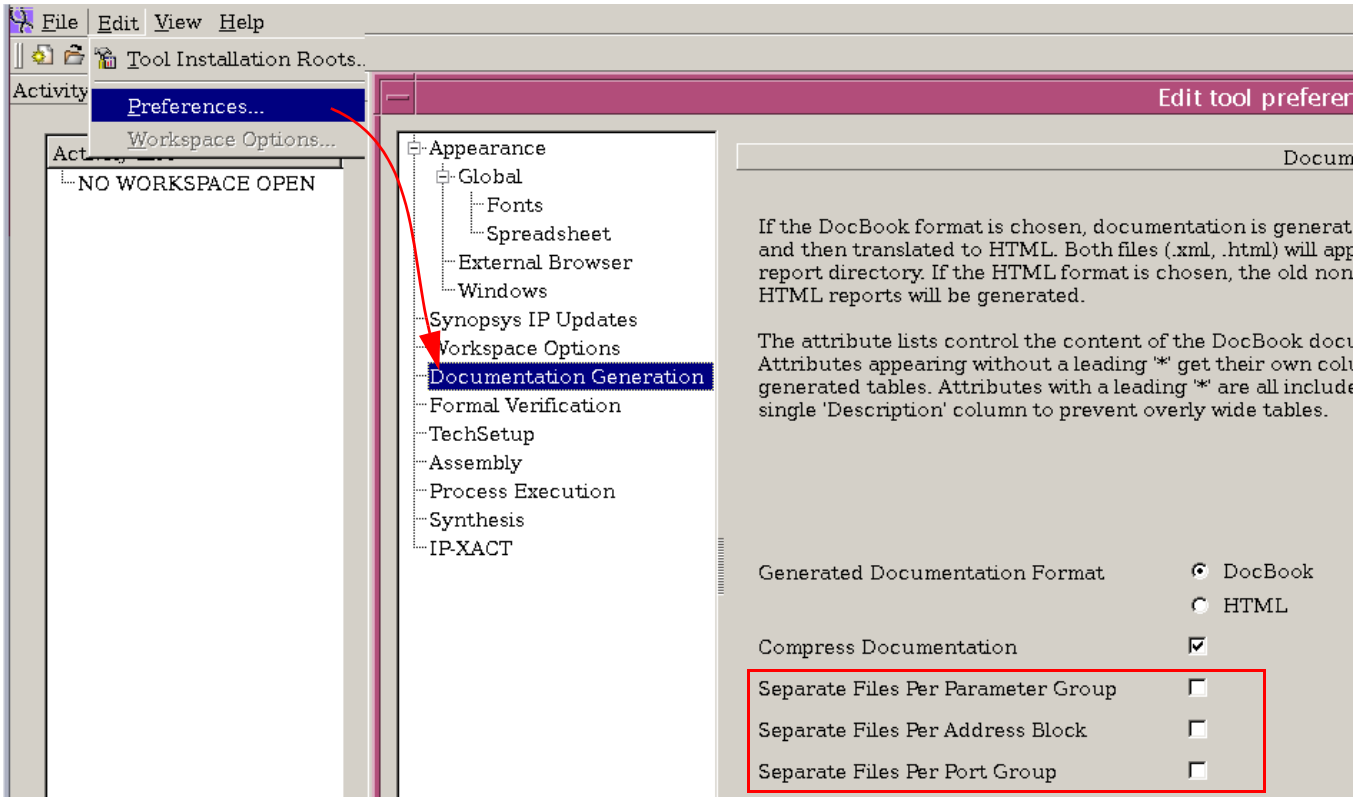
2.1.13.3 Setting Preferences for Report Generation

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 2-25.

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 2-25 Separate Files Options



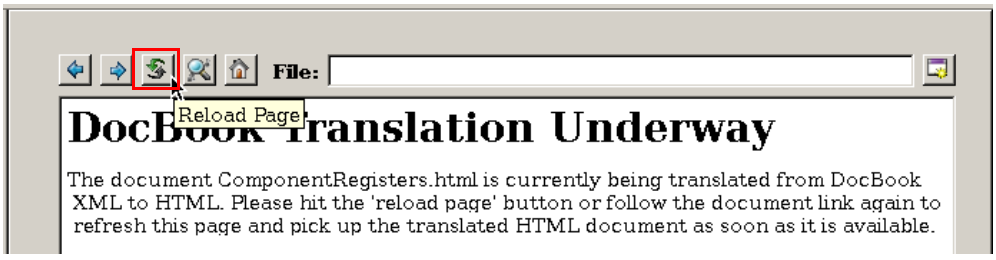
2.1.13.4 Generating Activity-Specific Reports

To generate the required report, click the respective **generate report**. For example, to generate component registers-related reports, click the one highlighted in Figure 2-24.

2.1.13.5 Accessing Reports

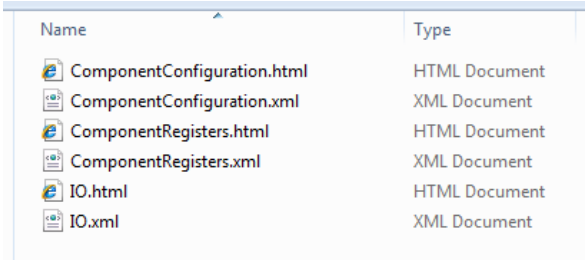
To view the report, click **Reload Page** after clicking **generate report** as shown in Figure 2-26.

Figure 2-26 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 2-27:

Figure 2-27 Contents of the report Directory



2.1.13.6 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.

2.1.14 Exporting a Core to Your Chip Design Database

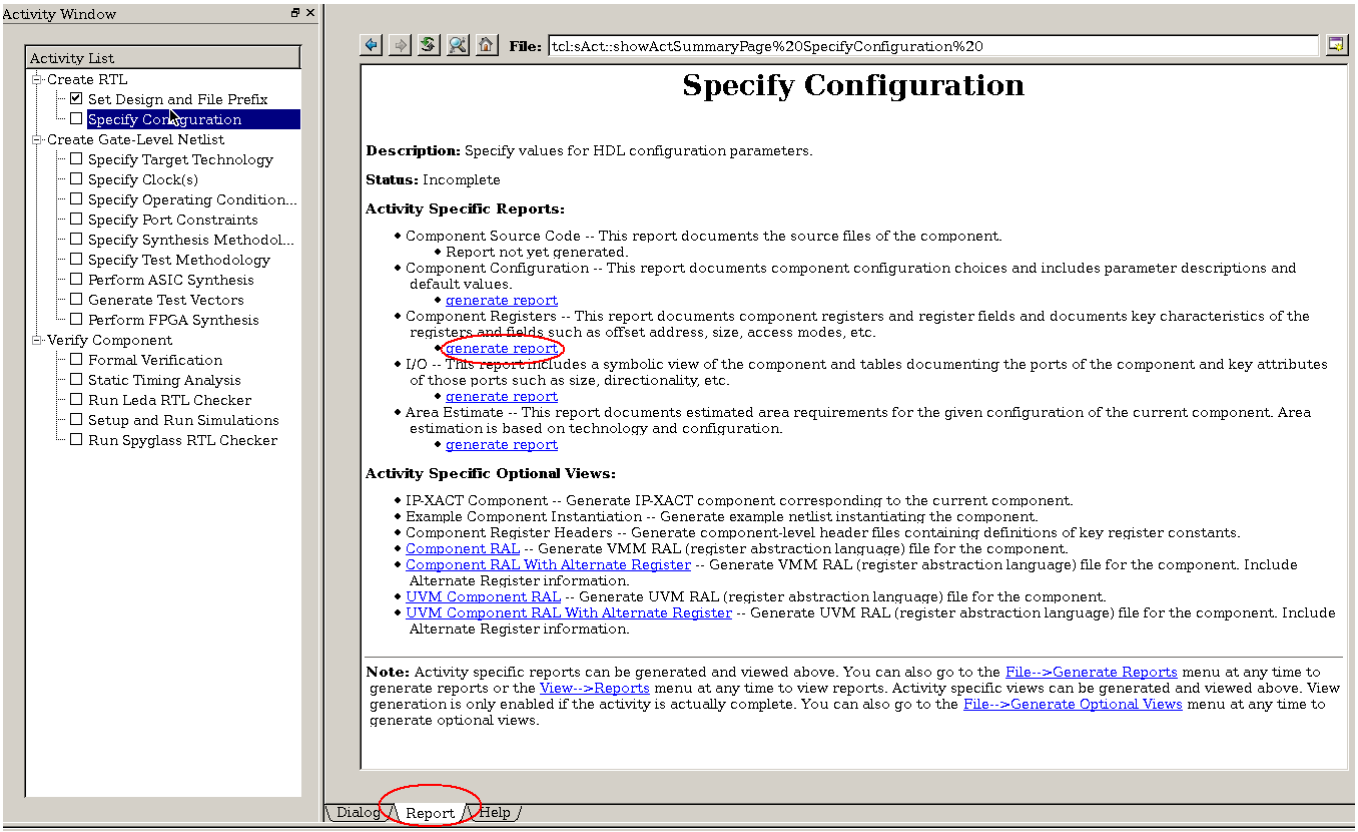
At a certain point you may want to transfer (export) a configured core into your own custom design flow. The coreConsultant tool has a number of features to accomplish this task. This section shows you how to access the relevant files, scripts, and information for your configured core. You can then transfer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

The topics in this section are as follows:

2.1.14.1 Generating an Example Core Instantiation

After you have configured your DWC_ssi core, you can generate an example Component Instantiation.

Figure 2-28 Generating an Example Core Instantiation



To create an example instantiation of your core, follow these steps:

1. Select the Report tab in the Specify Configuration activity.
2. Click the **generate view** link as indicated in Figure 2-24 on page 45.

The core instantiation is now available in `export/DWC_ssi_inst.v`.

2.1.14.2 Synthesizing to a Device Outside of coreConsultant

If you want to map and synthesize the core to a device using the Synopsys synthesis tools within coreConsultant, then follow the process as outlined in "Synthesizing the Core" on page 25. Otherwise, you must export the synthesis scripts from coreConsultant as outlined here so that you can synthesize the core in your own design flow. This section shows you how to access the relevant synthesis scripts and information for your configured core. You can then transfer these to your chip design database.

Hint To get a detailed description of how to integrate the IP at chip level. enter `man Synthesis_API` in the coreConsultant command line.

Method 1: Using 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.

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

```
write_sdc file_name
```

Figure 2-29 Using write_sdc command on coreConsultant Command Line

This file resides in the directory where you invoked coreConsultant.

Attention

When 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), 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.

If a technology library is not loaded, 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 an) FPGA has not been completed:

```
write_sdc [-force] file_name
```

Method 2: Accessing Synopsys Design Compiler Scripts

To access Design Compiler scripts:

1. Complete all the set-up steps in "Performing FPGA Synthesis" on page 33 or "Performing FPGA Synthesis" on page 33.
2. In the **Options** tab of the Synthesis activity, select the **Generate scripts only?** check box.
3. Click **Apply**.

The location of the constraints in an unpackaged core is at <workspace>/syn/constrain/script/

Table 2-8 ASIC and FPGA Synthesis Flow Files in <workspace>/syn

Synthesis Type	File	Description
ASIC	constrain/script/DWC_ssi.cscr	Primary constraints
	run.scr	Runs synthesis
	Makefile	Creates synthesis scripts
FPGA	synplify/script/DWC_ssi.sdc	Primary constraints
	synplify.tcl	Creates synthesis scripts
	synplify.scr	Runs synthesis

DWC_ssi.sdc.

Synthesizing for Xilinx FPGA

You must have access to Synopsys Synplify to synthesize and map the DWC_ssi core to an FPGA device using coreConsultant. Otherwise, you must run your third-party tools outside of coreConsultant.

Synthesizing for Altera FPGA

You must have access to Synopsys Synplify to synthesize and map the DWC_ssi core to an FPGA device using coreConsultant. Otherwise you must run your third-party tools outside of coreConsultant.

2.1.14.3 Exporting Formality, DFT, and ATPG Scripts

When you want run Formality and Tetramax (for ATPG) using the Synopsys synthesis tools within coreConsultant, then follow the process as outlined in "Performing Formal Verification" on page 36 or "Running Automatic Test Pattern Generation using TetraMax" on page 37. Otherwise, you must export the relevant scripts from coreConsultant as outlined here so that you can run Formality or TetraMax on the core in your own design flow.

This section shows you how to access the relevant scripts and information for your configured core. You can then transfer these to your chip design database. You must have access to Synopsys Design Compiler and Formality to generate and export the relevant scripts. These activities are not applicable to FPGA implementations. The topics in this section are as follows:

- "Exporting Formality Scripts"
- "Exporting DFT Scripts"
- "Exporting ATPG Scripts"
- "Exporting Views and Reports" on page 52

Exporting Formality Scripts

To access the Formality scripts you must first complete synthesis in coreConsultant. For details on generating and accessing the Formality scripts, see "Performing Formal Verification" on page 36. When you select the DCTCL_one_pass_compile_ultra flow (under **Synthesis -> Strategy**), synthesis runs quicker as it does not generate intermediate stages. The generated Formality script does not contain any core-specific directives. It is only useful to keep track of the .svf files (for all synthesis stages) and pass them all to Formality.

A template for that script may be inspected at <workspace>/export/fm.tcl. You must use the Svf flow for RTL-to-gate comparison. To disable non-applicable warning messages in Formality, ensure that your .synopsys_fm.setup file contains the following line:

```
set hdlin_warn_on_mismatch_message {FMR_ELAB-146 FMR_ELAB-147 FMR_VLOG-116}
```

Exporting DFT Scripts

See "Inserting Design For Test Using Design Compiler" on page 37.

Exporting ATPG Scripts

To access the ATPG scripts you must first complete synthesis in coreConsultant. For details on generating and accessing the TetraMax scripts, see "Running Automatic Test Pattern Generation using TetraMax" on page 37. When you select the DCTCL_one_pass_compile_ultra flow (under **Synthesis -> Strategy**),

synthesis runs quicker as it does not generate intermediate stages. The generated ATPG script does not contain any core-specific directives. It is only useful to keep track of the following and to pass them to TetraMax:

- .spf file (from the synthesis DFT stage)
- Technology library netlist
- Design netlist
- snps_phy_atpg_aux.tcl ("Running Automatic Test Pattern Generation using TetraMax" on page 37)

A template for that script may be inspected at <workspace>/export/atpg.tcl. A template for the Makefile may be inspected at <workspace>/export/Makefile.atpg.

2.1.14.3.1 Exporting Views and Reports

As part of the export process, you may want to generate documentation for your configured core to be used as part of the documentation for an entire chip. The report and view generation process is based on DocBook, which 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 44. All reports can be accessed in the `<workspace>/report` directory. The optional views are located in the `<workspace>/export` directory.

Attention The Generating Optional Reports and Views feature is still under development within coreConsultant and may not be fully functional with your core. For assistance, contact Synopsys Customer Support (see "Help Information" on page 76).

2.2 Creating a Subsystem Using coreAssembler

After you have correctly downloaded and installed a release of DesignWare SIP components and then setup your environment, you can begin to work on a subsystem using coreAssembler.

This section describes how to build a subsystem with DWC_ssi and AMBA 2/AMBA 3 AXI components using coreAssembler.

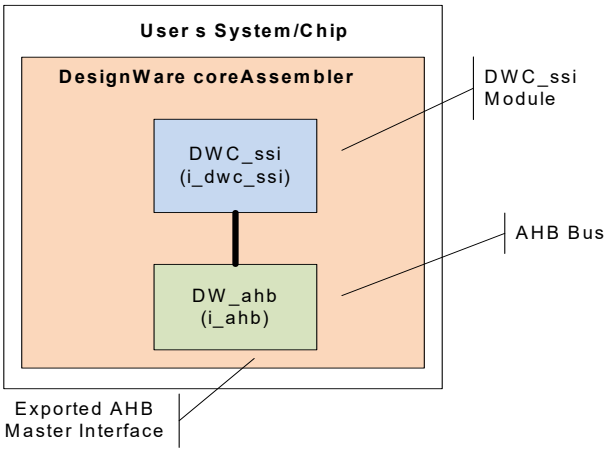
This section discuses the following topics:

Note While adding components into the coreAssembler GUI, you must add the DWC_ssi component after adding all other components in the GUI. This requirement is due to a limitation in coreAssembler, which will be fixed in a subsequent release.

2.2.1 Simple Subsystem

Figure 2-30 illustrates DWC_ssi in a simple subsystem.

Figure 2-30 DWC_ssi Simple Subsystem

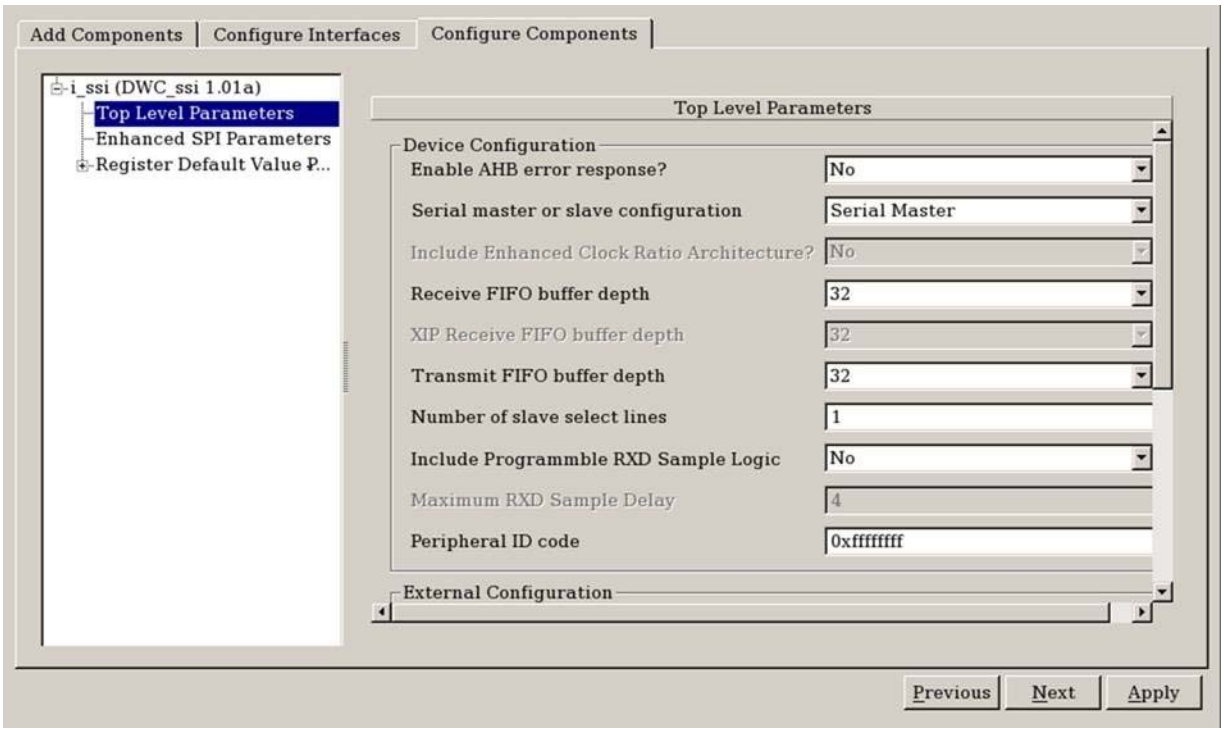


The subsystem in Figure 2-30 contains the DW_ahb (i_ahb) component that you may want to use as you learn to use coreAssembler.

2.2.2 Configuring DWC_ssi within a Subsystem

Figure 2-31 represents the coreAssembler GUI to configure DWC_ssi in a subsystem.

Figure 2-31 Configuring DWC_ssi in a Subsystem



1. **Specify your configuration for DWC_ssi.**

   Select options to enable or disable features.

   - You can use default values for an initial trial subsystem. However, you must select or enter specific values required to implement your design.
   - Make sure that you understand the definition of each parameter and change the default value only when it is not suitable for your application.

     You can access detailed information about each parameter by right-clicking on the parameter label and selecting *What's This* or by selecting the Help tab.
   - The coreAssembler tool enforces the parameter interdependencies interactively. For more information about the configuration parameters, refer to the "Parameters" chapter in the *DesignWare Cores Synchronous Serial Interface Databook*.

2. You must specify the configuration for DW_axi and DW_ahb similar to the previous step.

3. **Generate Subsystem RTL**

   Click **Apply** to generate configured RTL code for the subsystem. The coreAssembler tool then checks your parameter values and generates configured subsystem RTL code in the <workspace>/component/src/ directory.

2.2.3 Verifying the DWC_ssi within coreAssembler

"Verification" on page 57 provides an overview of the testbench available for DWC_ssi verification using the coreAssembler GUI.

The "Verifying Subsystems and Components" chapter in the *coreAssembler User Guide* discusses how to simulate a subsystem.

2.2.3.1 coreAssembler Subsystem Simulation Guidelines

The hardware reset test and bit-bash test can be performed using the coreAssembler.

2.2.4 Running SpyGlass on Generated Code with coreAssembler

When you select **Verify Component > Run SpyGlass RTL Checker for */i_component*** from the Activity List, the corresponding Activity View appears. In this Activity View, you can select to run SpyGlass Lint and SpyGlass CDC.

3 Verification

This chapter provides an overview of the testbench available for DWC_ssi verification. Once you have configured DWC_ssi in either coreAssembler or coreConsultant and set up the verification environment, you can run simulations automatically.

3.1 Overview of Vera Tests

The DWC_ssi verification testbench performs the set of tests in this section, which have been written to exhaustively verify the functionality and have also achieved maximum RTL code coverage. All tests use the AHB Interface to dynamically program memory-mapped registers during tests.

3.1.1 AHB Interface

This suite of tests is run to verify that the AHB interface functions correctly by checking the following:

- All address locations are written to with valid data
- Configured MacroCell is AMBA-compliant
- Read/write coherent
- Reset value of all registers
- Functionality of all registers

3.1.2 DWC_ssi as Master

This suite of tests is run only when the DWC_ssi is configured as a master, during which time all transfers are initiated by the DWC_ssi. When a master, the DWC_ssi must generate the clock sclk_out.

3.1.3 Interrupts

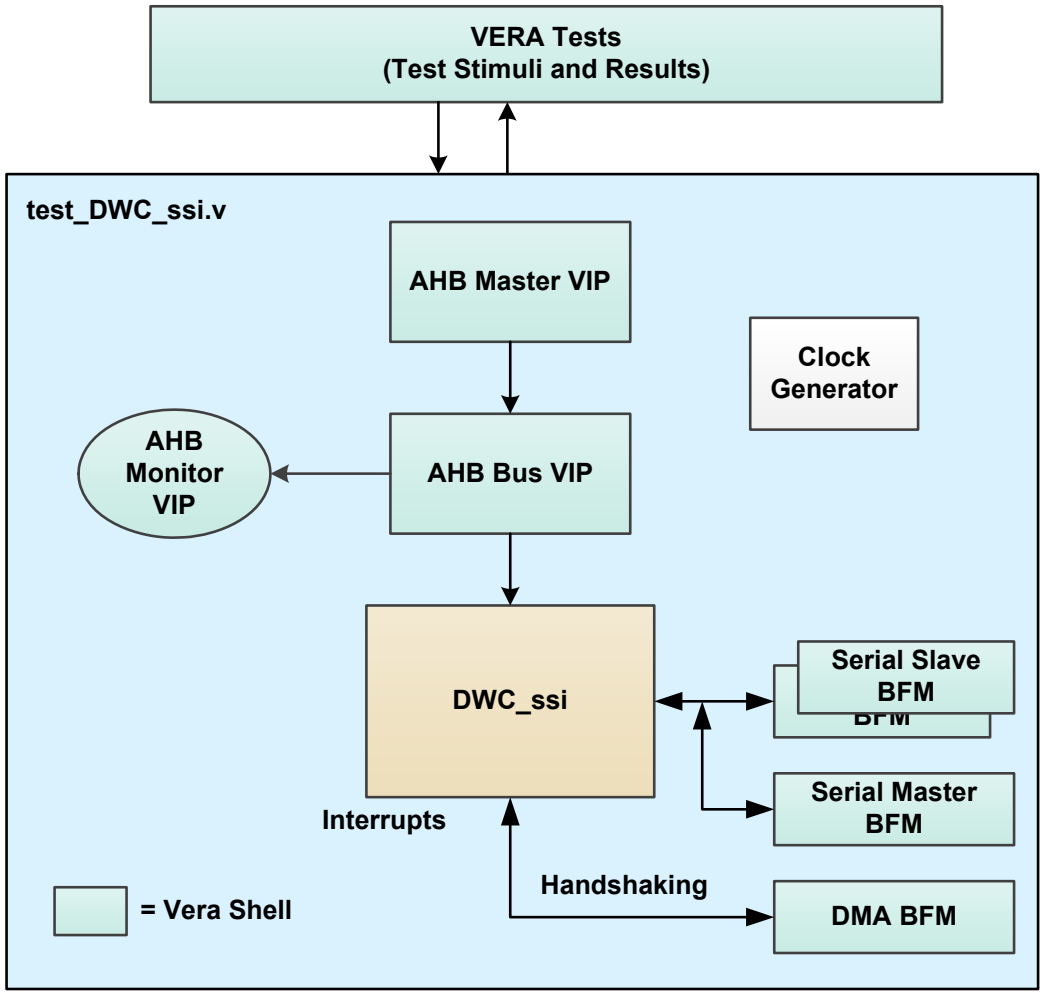
These tests verify the following:

- The Transmit FIFO Empty interrupt is not active when the DWC_ssi is returned from reset.
- The Receive FIFO Full interrupt is not active when the DWC_ssi is returned from reset.
- The Transmit FIFO Overflow interrupt is not active when the DWC_ssi is returned from reset.
- The Receive FIFO Overflow interrupt is not active when the DWC_ssi is returned from reset.
- The Receive FIFO Underflow interrupt is not active when the DWC_ssi is returned from reset.
- The MultiMaster Contention interrupt is not active when the DWC_ssi is returned from reset.

3.2 Overview of Vera Testbench

As illustrated in Figure 3-1, the DWC_ssi testbench is a Verilog testbench that includes an instantiation of the design under test (DUT) and a Vera shell.

Figure 3-1 DWC_ssi Testbench



The Vera shell consists of a number of serial-slave BFMs, as master BFM, and a DMA BFM to simulate and stimulate traffic to and from the DWC_ssi.

The test_DWC_ssi.v file shows the instantiation of the top-level MacroCell in a testbench and resides in the workspace/sim directory. The testbench checks your configuration selected in the Specify Configuration task of coreConsultant. The testbench also determines if the component is AMBA-compliant and includes a self-checking mechanism. When a coreKit has been unpacked and configured, the verification environment is stored in workspace/sim. Files in workspace/sim/test_xxx form the actual testbench for DWC_ssi.

4 Programming the DWC_ssi

This chapter describes the programmable features of the DWC_ssi.

4.1 Programming Considerations

You should program the following features during the configuration setup:

- AHB data bus width
- Type of device configuration; that is, serial master or serial slave
- Depth of receive and transmit FIFO buffers
- Peripheral ID code
- Whether to include DMA handshaking interface signals
- Whether the interrupt level is active high or active low
- Whether interrupts are individual or combined
- Whether to generate a clock enable input for the ssi_clk
- Whether or not pclk and ssi_clk are synchronous
- Whether to hardcode the frame format, and what type of frame format:
  - Motorola SPI – Requires setting the serial clock polarity and phase
  - Texas Instruments Synchronous Serial Protocol
  - National Semiconductor Microwire
- Whether to use Enhanced SPI mode of operation and advanced features

4.2 Master SPI and SSP Serial Transfers

The sections “Motorola Serial Peripheral Interface (SPI)” and “Texas Instruments Synchronous Serial Protocol (SSP)” in the DWC_ssi Databook describe the SPI and SSP serial protocols, respectively. They include timing diagrams and provide information as to how data are structured in the transmit and receive the FIFOs before and after the serial transfer.

When the transfer mode is “transmit and receive” or “transmit only” (TMOD = 2'b00 or TMOD = 2'b01, respectively), transfers are terminated by the shift control logic when the transmit FIFO is empty. For

continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level (TXFTLR) can be used to early interrupt (ssi_txe_intr) the processor indicating that the transmit FIFO buffer is nearly empty.

When a DMA is used for AHB accesses, the transmit data level (DMATDLR) can be used to early request (dma_tx_req) the DMA Controller, indicating that the transmit FIFO is nearly empty. The FIFO can then be refilled with data to continue the serial transfer. The user may also write a block of data (at least two FIFO entries) into the transmit FIFO before enabling a serial slave. This ensures that serial transmission does not begin until the number of data-frames that make up the continuous transfer are present in the transmit FIFO.

When the transfer mode is “receive only” (TMOD = 2'b10), a serial transfer is started by writing one “dummy” data word into the transmit FIFO when a serial slave is selected. The txd output from the DWC_ssi is held at a constant logic level for the duration of the serial transfer. The transmit FIFO is popped only once at the beginning and may remain empty for the duration of the serial transfer. The end of the serial transfer is controlled by the “number of data frames” (NDF) field in control register 1 (CTRLR1).

If, for example, you want to receive 24 data frames from a serial-slave peripheral, you should program the NDF field with the value 23; the receive logic terminates the serial transfer when the number of frames received is equal to the NDF value + 1. This transfer mode increases the bandwidth of the AHB bus as the transmit FIFO never needs to be serviced during the transfer. The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow.

When the transfer mode is “eeprom_read” (TMOD = 2‘b11), a serial transfer is started by writing the opcode and/or address into the transmit FIFO when a serial slave (EEPROM) is selected. The opcode and address are transmitted to the EEPROM device, after which read data is received from the EEPROM device and stored in the receive FIFO. The end of the serial transfer is controlled by the NDF field in the control register 1 (CTRLR1).

EEPROM read mode is not supported when the DWC_ssi is configured to be in the SSP mode.

The receive FIFO threshold level (RXFTLR) can be used to give early indication that the receive FIFO is nearly full. When a DMA is used for AHB accesses, the receive data level (DMARDLR) can be used to early request (dma_rx_req) the DMA Controller, indicating that the receive FIFO is nearly full.

A typical software flow for completing an SPI or SSP serial transfer from the DWC_ssi serial master is outlined as follows:

1. If the DWC_ssi is enabled, disable it by writing 0 to the SSI Enable register (SSIENR).
2. Set up the DWC_ssi control registers for the transfer; these registers can be set in any order.
   - Write Control Register 0 (CTRLR0). For SPI transfers, the serial clock polarity and serial clock phase parameters must be set identical to target slave device.
   - If the transfer mode is *receive only*, write CTRLR1 (Control Register 1) with the number of frames in the transfer minus 1; for example, if you want to receive four data frames, write this register with 3.
   - Write the Baud Rate Select Register (BAUDR) to set the baud rate for the transfer.

- Write the Transmit and Receive FIFO Threshold Level registers (TXFTLR and RXFTLR, respectively) to set FIFO threshold levels.
- Write the TXFTLR.TXFTHR (TX FIFO Start Transfer Threshold) accordingly.
- Write the IMR register to set up interrupt masks.
- The Slave Enable Register (SER) register can be written here to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the Data Register (DR), the transfer does not begin until a slave is enabled.

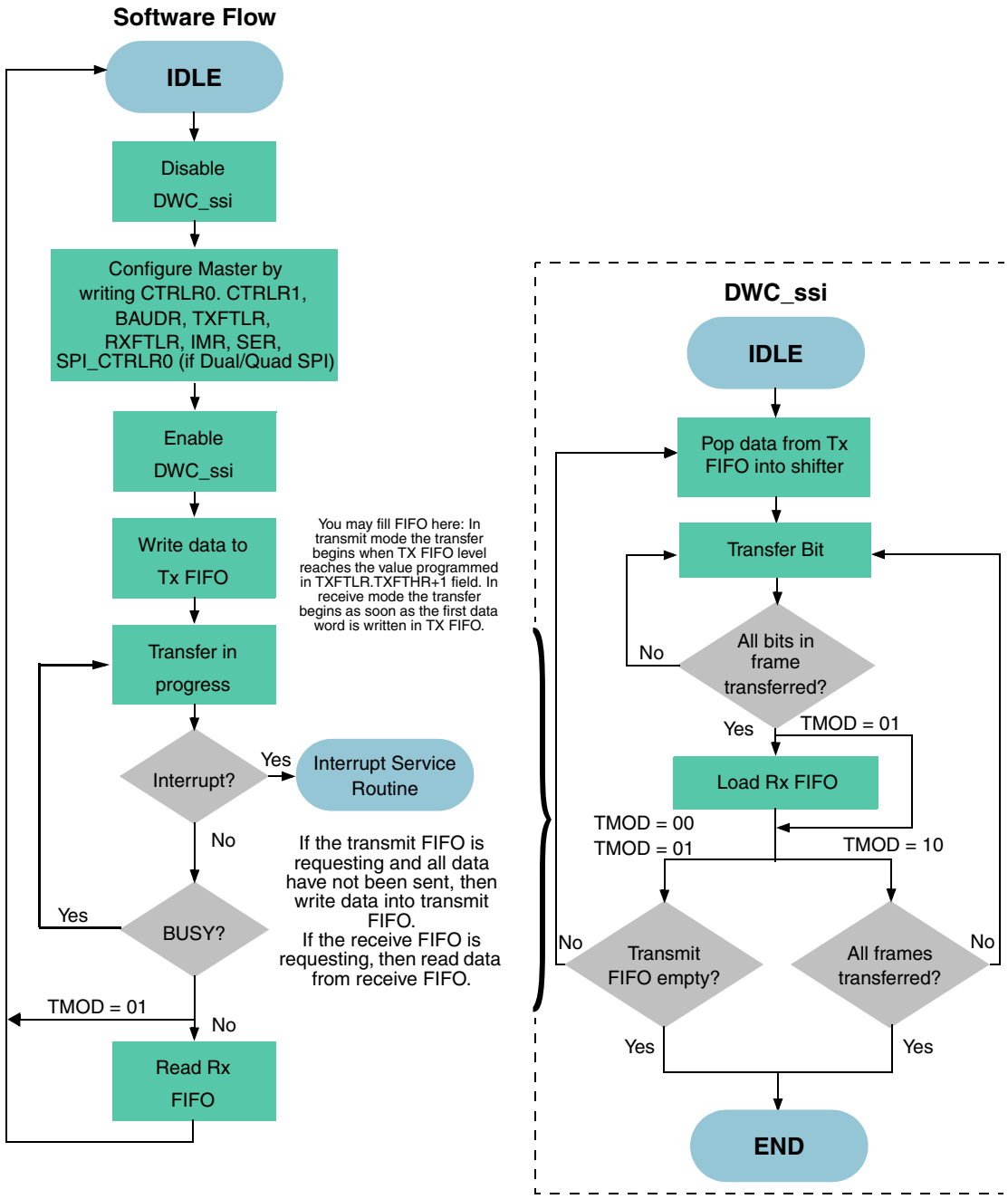
3. Enable the DWC_ssi by writing 1 to the SSIENR register.
4. Write data for transmission to the target slave into the transmit FIFO (write DR).

   If no slaves were enabled in the SER register at this point, enable it now to begin the transfer.
5. Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately.

   If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
6. The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is *receive only* (TMOD = 2'b10), the transfer is stopped by the shift control logic when the specified number of frames have been received. When the transfer is done, the BUSY status is reset to 0.
7. If the transfer mode is not *transmit only* (TMOD != 01), read the receive FIFO until it is empty.
8. Disable the DWC_ssi by writing 0 to SSIENR.

Figure 4-1 shows a typical software flow for starting a DWC_ssi master SPI/SSP serial transfer. The diagram also shows the hardware flow inside the serial-master component.

Figure 4-1 DWC_ssi Master SPI/SSP Transfer Flow



4.3 Slave SPI and SSP Serial Transfers

The sections on “Motorola Serial Peripheral Interface (SPI)” and “Texas Instruments Synchronous Serial Protocol (SSP)” in the DWC_ssi Databook contain a description of the SPI and SSP serial protocols,

respectively. The sections also provide timing diagrams and information on how data are structured in the transmit and receive FIFOs before and after the serial transfer.

If the DWC_ssi slave is *receive only* (TMOD=10), the transmit FIFO need not contain valid data because the data currently in the transmit shift register is resent each time the slave device is selected. The TXE error flag in the status register (SR) is not set when TMOD=01. You should mask the transmit FIFO empty interrupt when this mode is used.

If the DWC_ssi slave transmits data to the master, you must ensure that data exists in the transmit FIFO before a transfer is initiated by the serial-master device. If the master initiates a transfer to the DWC_ssi slave when no data exists in the transmit FIFO, an error flag (TXE) is set in the DWC_ssi status register, and the previously transmitted data frame is resent on txd. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level register (TXFTLR) can be used to early interrupt (ssi_txe_intr) the processor, indicating that the transmit FIFO buffer is nearly empty. When a DMA Controller is used for AHB accesses, the DMA transmit data level register (DMATDLR) can be used to early request (dma_tx_req) the DMA Controller, indicating that the transmit FIFO is nearly empty. The FIFO can then be refilled with data to continue the serial transfer.

The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow. The receive FIFO threshold level register (RXFTLR) can be used to give early indication that the receive FIFO is nearly full. When a DMA Controller is used for AHB accesses, the DMA receive data level register (DMARDLDR) can be used to early request (dma_rx_req) the DMA controller, indicating that the receive FIFO is nearly full.

A typical software flow for completing a continuous serial transfer from a serial master to the DWC_ssi slave is described as follows:

1. If the DWC_ssi is enabled, disable it by writing 0 to SSIENR.
2. Set up the DWC_ssi control registers for the transfer. These registers can be set in any order.
   a. Write CTRLR0 (for SPI transfers SCPH and SCPOL must be set identical to the master device).
   b. Write TXFTLR and RXFTLR to set FIFO threshold levels.
   c. Write the IMR register to set up interrupt masks.
3. Enable the DWC_ssi by writing 1 to the SSIENR register.
4. If the transfer mode is *transmit and receive* (TMOD=2'b00) or *transmit only* (TMOD=2'b01), write data for transmission to the master into the transmit FIFO (Write DR).

   If the transfer mode is *receive only* (TMOD=2'b10), there is no need to write data into the transmit FIFO; the current value in the transmit shift register is retransmitted.
5. The DWC_ssi slave is now ready for the serial transfer. The transfer begins when the DWC_ssi slave is selected by a serial-master device.
6. When the transfer is underway, the BUSY status can be polled to return the transfer status. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
7. The transfer ends when the serial master removes the select input to the DWC_ssi slave. When the transfer is completed, the BUSY status is reset to 0.

8. If the transfer mode is not *transmit only* (TMOD != 01), read the receive FIFO until empty.
9. Disable the DWC_ssi by writing 0 to SSIENR.

Figure 4-2 shows a typical software flow for a DWC_ssi slave SPI or SSP serial transfer. The diagram also shows the hardware flow inside the serial-slave component.

Figure 4-2 DWC_ssi Slave SPI/SSP Transfer Flow

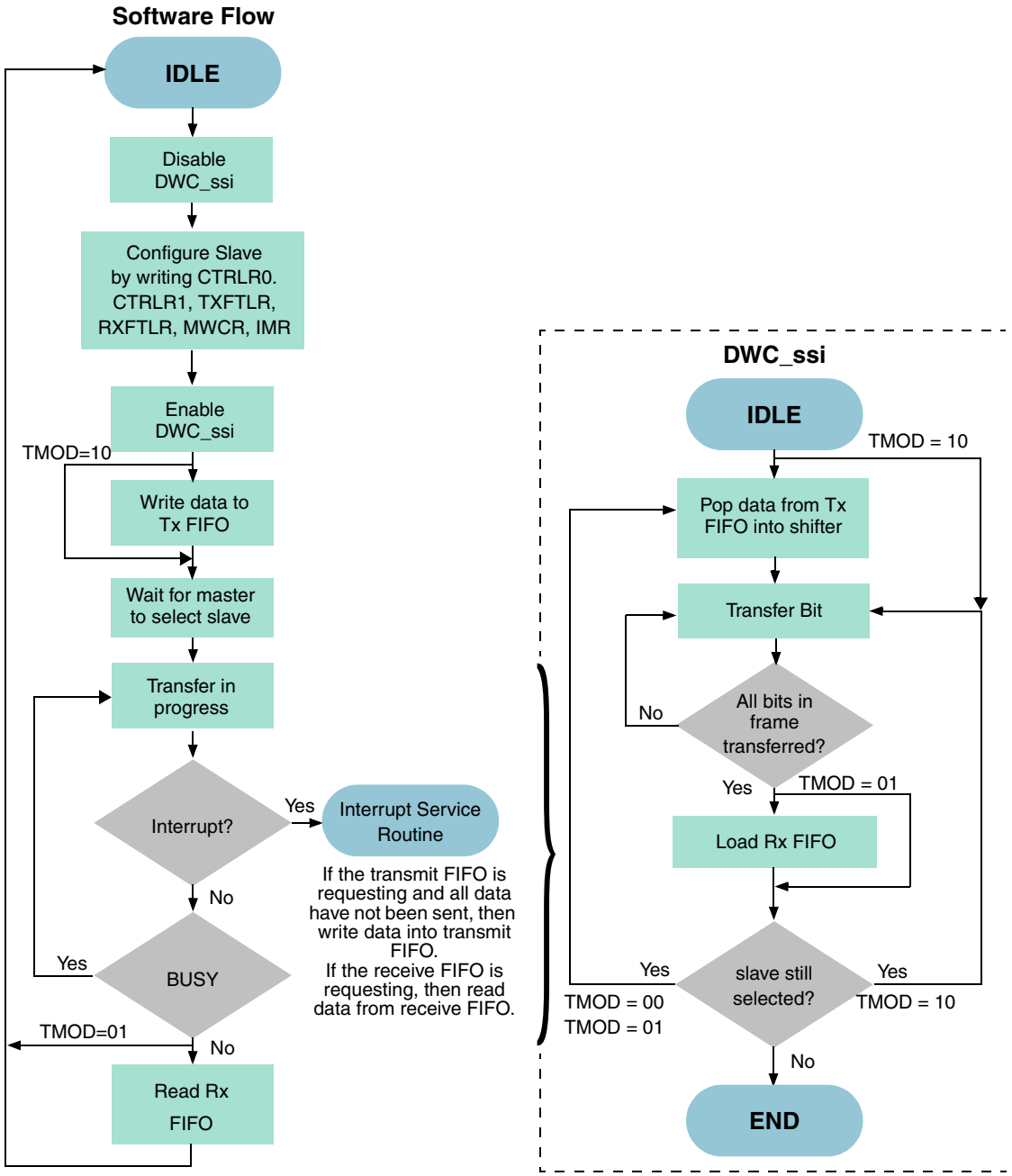
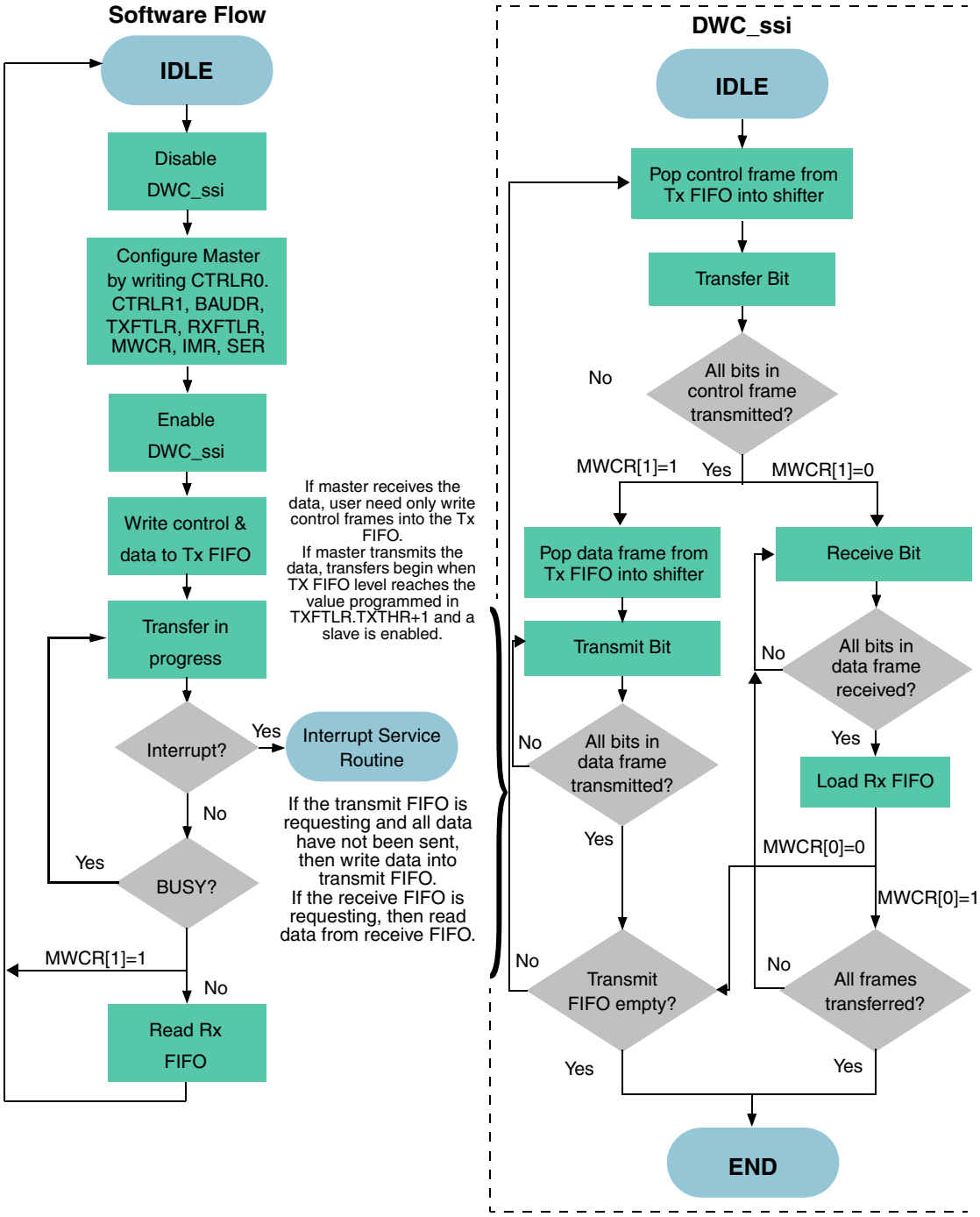


Figure 4-3 shows a typical software flow for starting a DWC_ssi master Microwire serial transfer. The diagram also shows the hardware flow inside the serial-master component.

Figure 4-3 DWC_ssi Master Microwire Transfer Flow



4.4 Master Microwire Serial Transfers

Microwire serial transfers from the DWC_ssi serial master are controlled by the Microwire Control Register (MWCR). The MWHS bit field enables and disables the Microwire handshaking interface. The MDD bit field controls the direction of the data frame (the control frame is always transmitted by the master and received by the slave). The MWMOD bit field defines whether the transfer is sequential or nonsequential.

All Microwire transfers are started by the DWC_ssi serial master when there is at least one control word in the transmit FIFO and a slave is enabled. When the DWC_ssi master transmits the data frame (MDD = 1), the transfer is terminated by the shift logic when the transmit FIFO is empty. When the DWC_ssi master receives the data frame (MDD = 1), the termination of the transfer depends on the setting of the MWMOD bit field. If the transfer is nonsequential (MWMOD = 0), it is terminated when the transmit FIFO is empty after shifting in the data frame from the slave. When the transfer is sequential (MWMOD = 1), it is terminated by the shift logic when the number of data frames received is equal to the value in the CTRLR1 register + 1.

When the handshaking interface on the DWC_ssi master is enabled (MWHS =1), the status of the target slave is polled after transmission. Only when the slave reports a *ready status* does the DWC_ssi master complete the transfer and clear its BUSY status. If the transfer is continuous, the next control/data frame is not sent until the slave device returns a *ready status*.

A typical software flow for completing a Microwire serial transfer from the DWC_ssi serial master is outlined as follows:

1. If the DWC_ssi is enabled, disable it by writing 0 to SSIENR.
2. Set up the DWC_ssi control registers for the transfer. These registers can be set in any order. Write CTRLR0 to set transfer parameters.
   - If the transfer is sequential and the DWC_ssi master receives data, write CTRLR1 with the number of frames in the transfer minus 1; for instance, if you want to receive four data frames, write this register with 3.
   - Write BAUDR to set the baud rate for the transfer.
   - Write TXFTLR and RXFTLR to set FIFO threshold levels.
   - Write the IMR register to set up interrupt masks.

   You can write the SER register to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the DR register, the transfer does not begin until a slave is enabled.
3. Enable the DWC_ssi by writing 1 to the SSIENR register.
4. If the DWC_ssi master transmits data, write the control and data words into the transmit FIFO (write DR). If the DWC_ssi master receives data, write the control word(s) into the transmit FIFO.

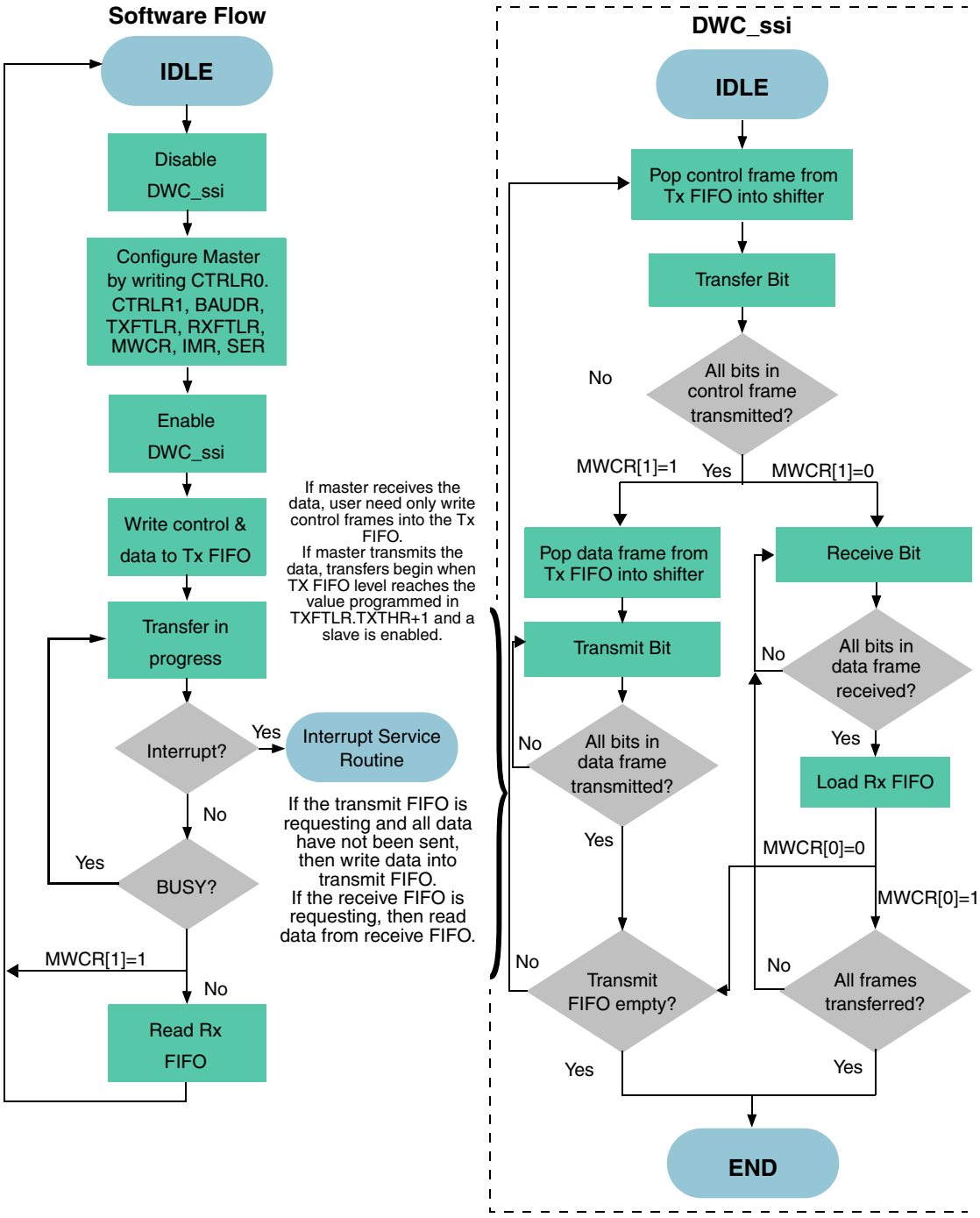
   If no slaves were enabled in the SER register at this point, enable now to begin the transfer.
5. Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately.

   If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).

6. The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is sequential and the DWC_ssi master receives data, the transfer is stopped by the shift control logic when the specified number of data frames is received. When the transfer is done, the BUSY status is reset to 0.
7. If the DWC_ssi master receives data, read the receive FIFO until it is empty.
8. Disable the DWC_ssi by writing 0 to SSIENR.

Figure 4-4 shows a typical software flow for starting a DWC_ssi master Microwire serial transfer. The diagram also shows the hardware flow inside the serial-master component.

Figure 4-4 DWC_ssi Master Microwire Transfer Flow



4.5 Slave Microwire Serial Transfers

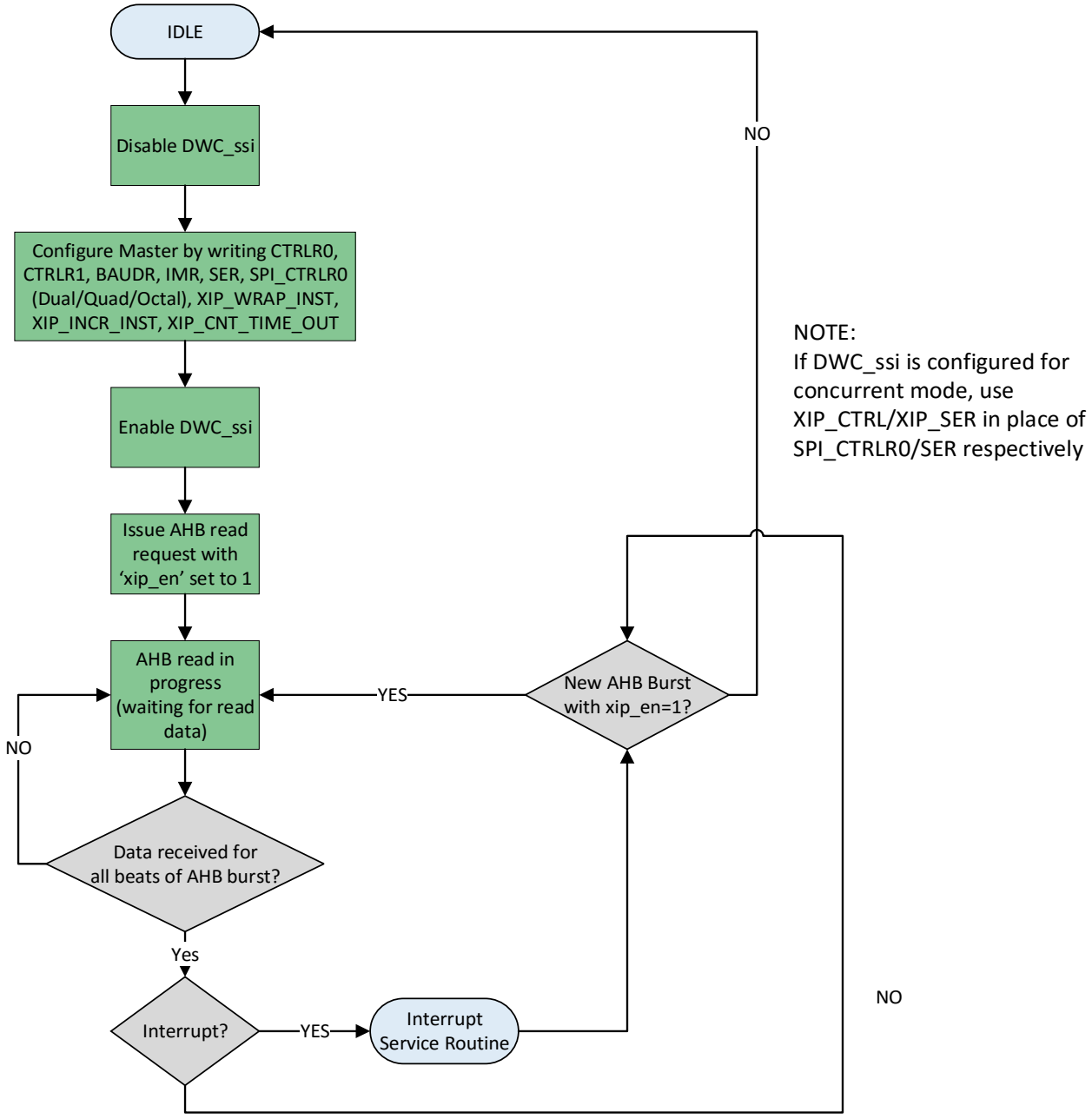
The “National Semiconductor Microwire” section in the DWC_ssi Databook describes the Microwire serial protocol in detail, including timing diagrams and information on how data are structured in the transmit and receive FIFOs before and after a serial transfer. When the DWC_ssi is configured as a slave device, the

Microwire protocol operates in the same way as the SPI protocol. There is no decode of the control frame by the DWC_ssi slave device.

4.6 eXecute In Place (XIP) Transfers

The “eXecute In Place (XIP) Mode” section in the DWC_ssi Databook describes the XIP transfers in detail. The section also includes timing diagrams and usage models. Below diagram explains the transfer flow of XIP transfer.

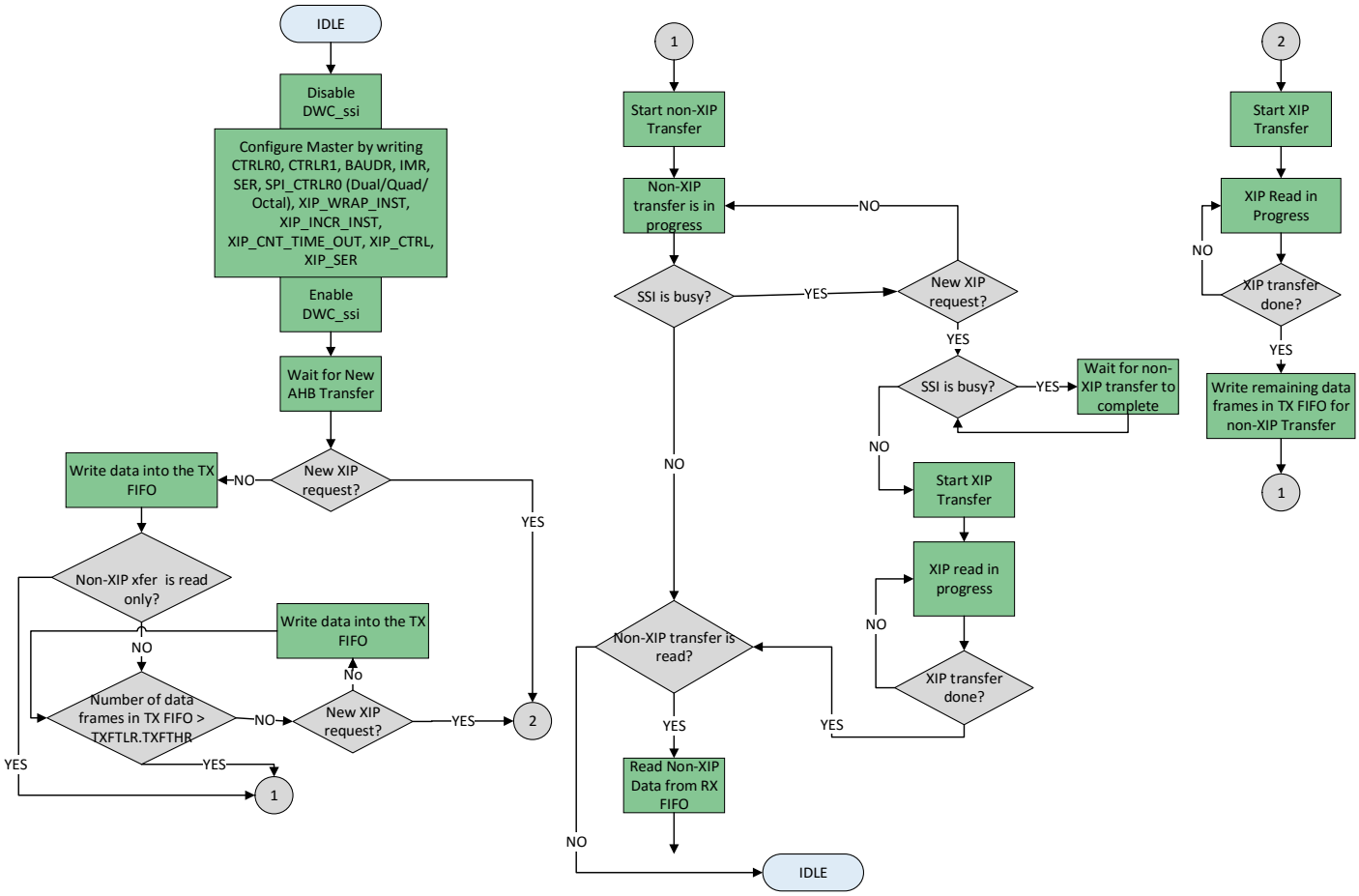
Figure 4-5 eXecute In Place (XIP) Transfer Flow



4.7 Concurrent XIP and non-XIP Transfers

The “Concurrent XIP and Non-XIP Operation” section in the DWC_ssi Databook describes the Concurrent XIP and non XIP transfers in detail. Below diagram explains the transfer flow of Concurrent XIP and non XIP transfer.

Figure 4-6 Concurrent XIP and non-XIP Transfer flow



A

Additional coreConsultant Information

The topics in this section are as follows:

A.1 Troubleshooting

This section provides some common problems you may encounter when performing activities in coreConsultant.

A.1.1 Specify Configuration Activity

Here are some common problems that can occur after you generate RTL for your configured core:

- Encrypted source code is generated. Typically this problem is caused by not providing a Project ID when installing the core. You must re-install the core using the Project ID number. For more information, refer to the <*full product name*> *Installation Guide*.
- The coreConsultant tool issues a message that it cannot write to certain files. Check your available storage. You may have too little storage on your server.

A.1.2 Setup and Run Simulation Activity

An error occurs if AMBA VMT VIP is not installed in the DesignWare home directory.

A.1.3 Create Gate-Level Netlist Activity

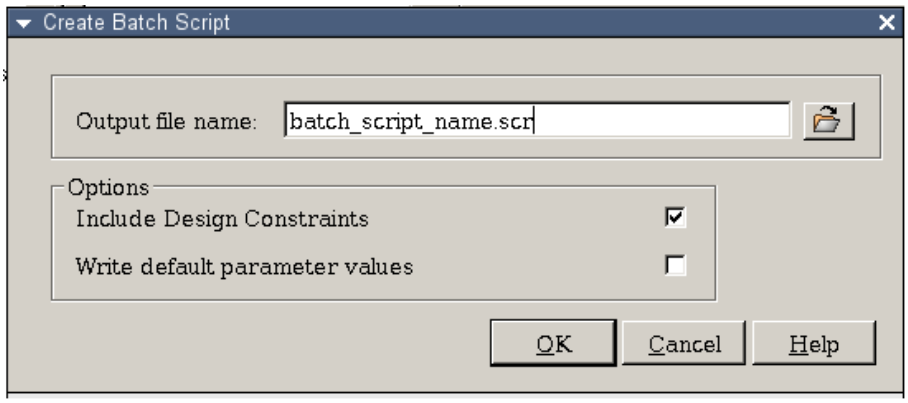
Here are some common problems that can occur after you perform synthesis (ASIC or FPGA) for your configured core:

- 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 because files) written by this activity can adversely affect the outcome of the synthesis run. Check the Console Pane for any messages that are called out for the guard file name.
- A user defined strategy file does not exist. Go the Console Pane and scroll to the message about the file not being created. Create that file.
- A cell name was specified that could not be found in the currently loaded technology libraries. Review the technology and link libraries being used and select a cell that exists within one of the libraries.
- When you cannot achieve timing closure with the default settings, then

A.2 Creating a Batch Script

Batch mode allows you to execute a series of coreConsultant commands from a batch file. You create a batch script after you configure the core, so that you can recreate your exact configuration at a later stage in case you delete or overwrite your current workspace. To create a batch file, choose the **File > Write Batch Script** menu item and enter a name for the file, as illustrated 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.

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
  ```

- Source the batch file from the coreConsultant command line:

  ```
  source <batch_file_name>
  ```

A.3 Help Information

Several types of online help are available through coreConsultant:

- **coreConsultant User Manual 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
  ```

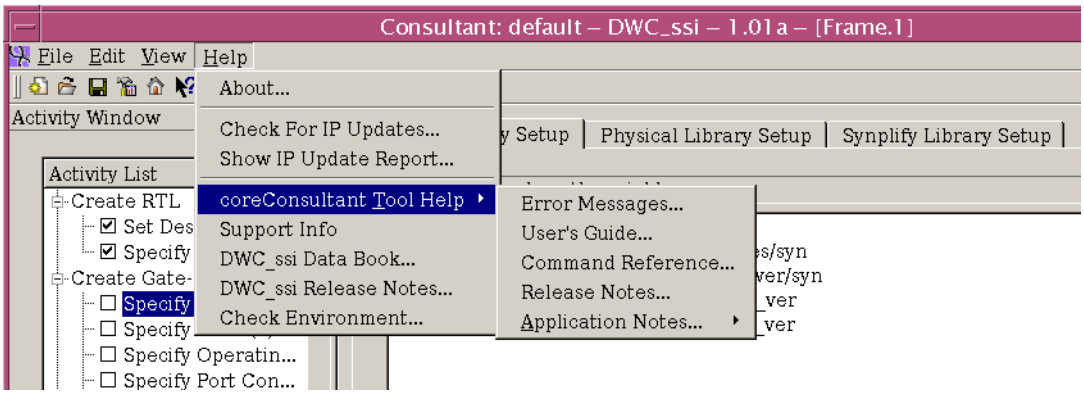
- **"What's This?" Quick Help**

  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-2shows the Help pull-down with the available manuals for both the DWC_ssi core and the coreConsultant tool.

Figure A-2 coreConsultant Help Menu Pull-down with DWC_ssi Core 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.4 Workspace Directory Structure Overview

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

Figure A-3 coreConsultant Workspace Directory

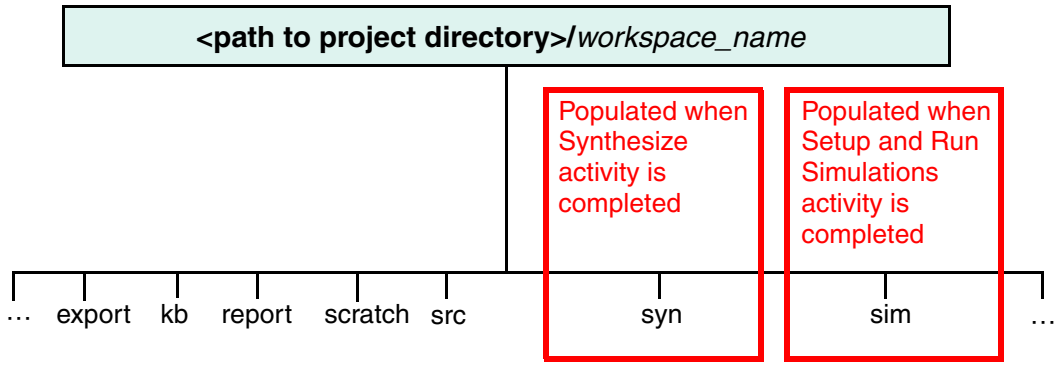


Table A-1 provides a description of the workspace directory and subdirectories.

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_ssi 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.
fpga	Contains FPGA implementation example files and Synopsys PHY application note.
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.

Table A-1 Workspace Directory Contents (Continued)

Directory/Subdirectory	Description
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.
sim	Contains test stimulus and output files. Generated upon first creating workspace; updated during the Setup and Run Simulations activity.
spyglass	Contains SpyGlass Lint and CDC configuration files for the component. Generated upon first SpyGlass run; updated during Run SpyGlass RTL Checker activity.
src	Includes all the RTL files. Generated upon first creating workspace; populated during the Specify Configuration activity.
syn	Contains synthesis files for the DWC_ssi. Generated upon first creating workspace; updated during Create Gate-Level Netlist activity and Formal Verification activity.
tcl	Contains synthesis intent scripts. Generated upon first creating workspace.
xprop	Contains the files used for VCS Xprop analysis activity. Generated upon running the Run VCS XPROP Analyzer activity under Verify Component Tab. It is applicable only when you have set the VCS_HOME directory.

A.5 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).