DesignWare Synthesizable Components for AMBA 3 AXI, and AMBA 4 AXI

User Guide

Product Codes

Version 2020.03a
March 2020

Synopsys, Inc.
www.synopsys.com

Contents

Revision History

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

Date	Image Version	Description
March 2020	2020.03a	**Updated**: ■ All SolvNet links to SolvNetPlus ■ coreConsultant screen-shots ■ Figure 2-7 ■ Figure 2-14 **Added** ■ "hmaster Connectivity in DW_axi_a2x" on page 26
February 2018	2018.02a	The first version of the User Guide

Preface

This preface contains the following sections

- "User Guide Organization" on page 7
- "Related Product Information" on page 7
- "Web Resources" on page 8
- "Customer Support" on page 8
- "Product Codes" on page 9

This user guide provides information that you need to interface the, referred to as IP throughout the remainder of this user guide.

The information in this user guide includes Configuring the Core using the coreConsultant GUI and additional coreConsultant and coreAssembler information.

User Guide Organization

The chapters of this user guide are organized as follows:

- Chapter 1, "Product Overview" introduces the, 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 using the Synopsys coreConsultant tool.
- 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:

- Using DesignWare Library IP in coreAssembler. Contains information on getting started with using DesignWare SIP components for AMBA 3 AXI and AMBA 4 AXI components within coreTools
- coreAssembler User Guide. Contains information on using coreAssembler
- coreConsultant User Guide. Contains information on using coreConsultant

- Guide to Documentation for DesignWare Synthesizable Components for AMBA 2, AMBA 3 AXI, and AMBA 4 AXI - To see a complete listing of documentation within the DesignWare Synthesizable Components for AMBA 3 AXI and AMBA 4 AXI.
- Verification IP (VIP)

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

  - Synopsys AMBA VMT and AMBA SVT VIP - Download from the *SolvNet Download Center*

Web Resources

- DesignWare IP product information: https://www.synopsys.com/designware-ip.html
- Your custom DesignWare IP page: https://www.synopsys.com/dw/mydesignware.php
- Documentation through SolvNetPlus: https://solvnetplus.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): https://www.synopsys.com/keys

Customer Support

Synopsys provides the following various methods for contacting Customer Support:

- Prepare the following debug information, if applicable:
  - For environment set-up problems or failures with configuration, simulation, or synthesis that occur within coreConsultant or coreAssembler, select the following menu:

    File > Build Debug Tar-file

    Check all the boxes in the dialog box that apply to your issue. This option gathers all the Synopsys product data needed to begin debugging an issue and writes it to the <*core tool startup directory*>/debug.tar.gz file.
  - 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.
- *For the fastest response*, enter a case through SolvNetPlus:

  a. https://solvnetplus.synopsys.com

  > **Note** SolvNetPlus does not support Internet Explorer. Use a supported browser such as Microsoft Edge, Google Chrome, Mozilla Firefox, or Apple Safari.

  b. Click the **Cases** menu and then click **Create a New Case** (below the list of cases).

  c. Complete the mandatory fields that are marked with an asterisk and click **Save**.

  Ensure to include the following:

  - **Product L1:** DesignWare Library IP
  - **Product L2:** <name of L2>

d. After creating the case, attach any debug files you created.

For more information about general usage information, refer to the following article in SolvNetPlus:

https://solvnetplus.synopsys.com/s/article/SolvNetPlus-Usage-Help-Resources

- 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 L1 and Product L2 names, and Version number in your e-mail 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.
- Or, telephone your local support center:
  - North America:
    Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
  - All other countries:
    https://www.synopsys.com/support/global-support-centers.html

Product Codes

Following is the list of all product codes associated with DesignWare Synthesizable Components for AMBA 3 AXI, and AMBA 4 AXI.

- Product Code: 3768-0 - DesignWare AMBA Fabric
- Product Code: 2925-0 - DesignWare
- Product Code: A415-0 - DW_axi_dmac
- Product Code: E137-0 - DW_axi_dmac Safety Features

Note For the list of components associated with each product code, and Licenses Requirements, see the “Licenses” section of DesignWare Synthesizable Components for AMBA 2, AMBA 3 AXI Installation Guide.

1
Product Overview

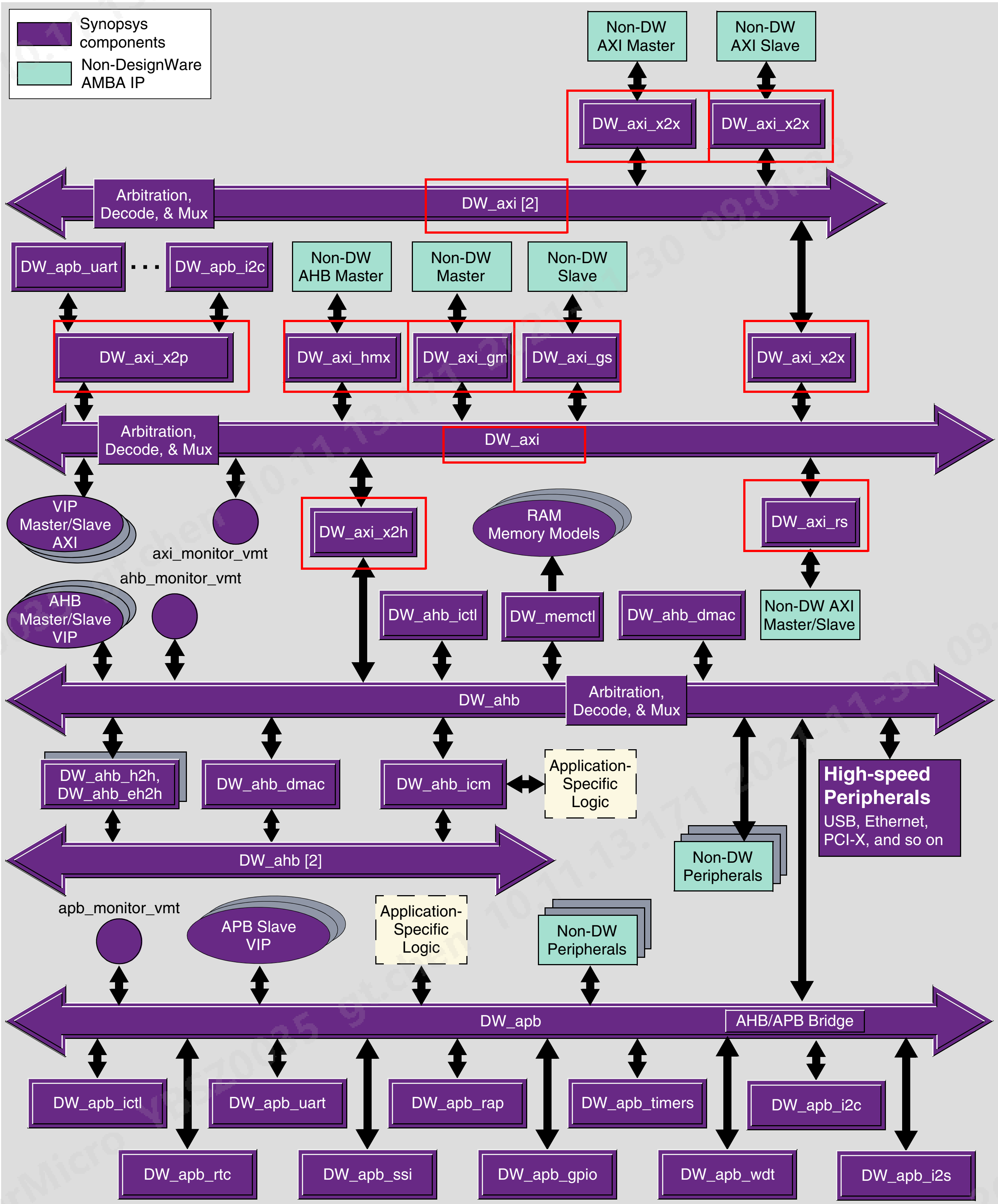
The DesignWare Library IP Cores are configurable, synthesizable, and programmable components. The DesignWare Library contains the following IP Cores of AXI family:

- **DW_axi** - The DW_axi component is a multilayer interconnect implementation of the AXI protocol, which is designed for high-performance, and high-frequency system designs.
- **DW_axi_a2x** - The DW_axi_a2x is a configurable bridge between an AHB or AXI bus protocol and an AXI bus protocol.
- **DW_axi_dmac** - The DW_axi_dmac is a highly configurable, highly programmable, with high performance, multi-master multichannel DMA controller with AXI as a bus interface for data transfer.
- **DW_axi_gm** - The DW_axi_gm is a configurable module between a generic interface (GIF, Master) and the AMBA AXI bus.
- **DW_axi_gs** - The DW_axi_gs is a configurable module between a generic interface (GIF, Slave) and the AMBA AXI bus.
- **DW_axi_hmx** - The DW_axi_hmx module connects a single AHB (Advanced High-performance Bus) master to the AXI bus.
- **DW_axi_rs** - The DW_axi_rs component is an AMBA 3 AXI/AMBA 4 AXI/ACE-Lite register slice (rs) component.
- **DW_axi_x2h** - The DW_axi_x2h is a configurable bridge between the ARM AMBA 3 AXI/AMBA 4 AXI protocol bus and the ARM AMBA 2.0 protocol AHB bus.
- **DW_axi_x2p** - The DW_axi_x2p is an AXI-to-APB bridge that connects an AMBA 3 AXI/AMBA 4 AXI-compliant master to an AMBA 2-compliant or AMBA 3-compliant APB slave.
- **DW_axi_x2x** - The DW_axi_x2x is a configurable bridge between an AXI master and an AXI slave with any combination of different data port widths, different clock frequencies, and different endianness.

Synopsys DesignWare Synthesizable Components environment is a parameterizable bus system that contains AMBA 2-compliant AHB (Advanced High-performance Bus), and APB (Advanced Peripheral Bus) components, and AMBA 3 AXI, AMBA 4 AXI (Advanced eXtensible Interface) components.

Figure 1-1 illustrates an example of this environment, including the AHB bus, the APB Bus (includes the APB Bridge), AHB multilayer interconnect IP, APB peripheral components, verification master/slave models, and bus monitors. To see the databook for DW_* component, click the corresponding component object in the illustration.

Figure 1-1 Complete System



You can connect, configure, synthesize, and verify the 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 of the, you might prefer to use coreConsultant, documentation for which is available in the *coreConsultant User Guide*.

2
Configuring the Core

DesignWare Synthesizable IPs (SIPs) are 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 3 AXI, and AMBA 4 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 the usage of coreConsultant.
- coreAssembler – Used for building and configuring a subsystem that connects multiple SIP titles with DesignWare Synthesizable IPs as well as AMBA 3, and AMBA 4 AXI library components, RTL generation, synthesis, and creation of a template subsystem testbench. The *coreAssembler User Guide* provides complete information on the usage of 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 DesignWare Synthesizable IPs 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:

- "Database Files" on page 69

2.1 Setting up Your Environment

The DesignWare Synthesizable IPs are included in a release of DesignWare SIP components. It is assumed that you have already downloaded and installed the release. If you have not, you can download and install the latest versions of required tools using the DesignWare Synthesizable Components for AMBA 2, AMBA 3 AXI Installation Guide.

You also need to set up your environment correctly using specific environment variables, such as DESIGNWARE_HOME, VERA_HOME, PATH, and SYNOPSYS. If you are not familiar with these requirements and the necessary licenses, see "Setting up Your Environment" in the DesignWare Synthesizable Components for AMBA 2, AMBA 3 AXI Installation Guide.

2.2 Overview of the coreConsultant Configuration and Integration Process

Once you have correctly downloaded and installed a release of DesignWare SIP components and then set up your environment, you can begin work on the DesignWare Synthesizable IPs using coreConsultant.

2.2.1 coreConsultant Usage

Figure 2-1 illustrates some general directories and files in a coreConsultant workspace.

Figure 2-1 coreConsultant Usage Flow

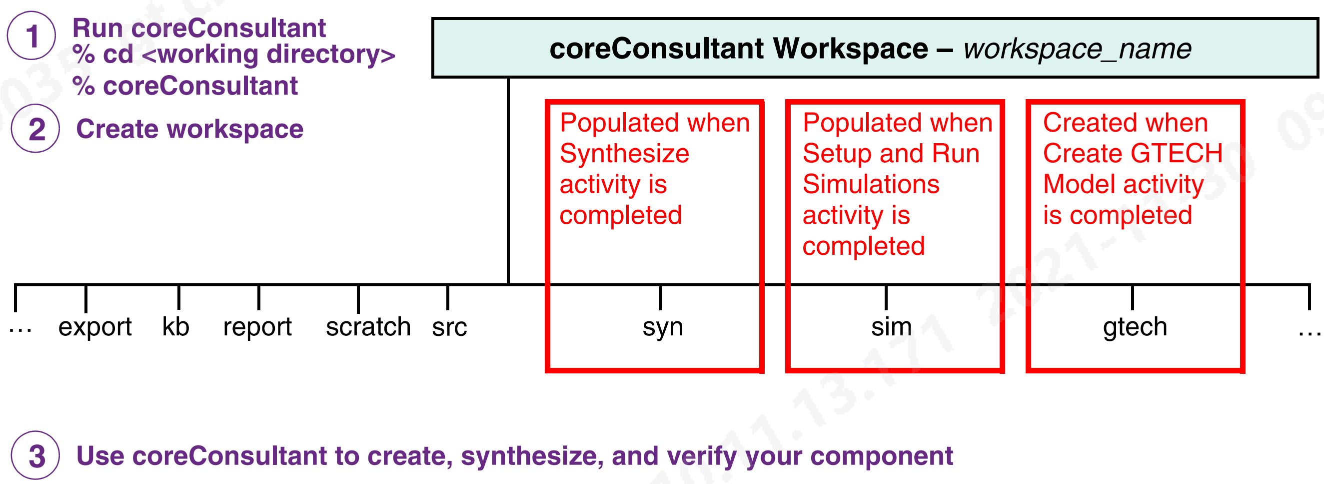


Table 2-1 provides a description of the implementation workspace directory and subdirectories.

Table 2-1 coreConsultant Implementation Workspace Directory Contents

Directory/Subdirectory	Description
auxiliary	Scripts and text files used by coreConsultant. Generated upon first creating workspace.
custom	Contains RTL preprocessor scripts. Generated during Specify Configuration activity.

Directory/Subdirectory	Description
doc	Contains local copies of component-specific databooks. 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 Specify Configuration activity.
gtech	Contains synthesis scripts and output netlists from gtech generation; also used for RTL simulation of encrypted source code. Generated during Generate GTECH Model activity.
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.
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 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 the top-level RTL file, *design_name*.v. If you have a source license, this contains plain-text RTL; if you only have a designware license, this contains encrypted RTL. Generated upon first creating workspace; populated during Specify Configuration activity.
syn	Contains synthesis files for the component. Generated upon first creating workspace; updated during Synthesis activity and Formal Verification activity.
tcl	Contains synthesis intent scripts. Generated upon first creating workspace.
xprop	Contains the files used for the VCS Xprop analysis activity. Generated upon running the "Run VCS XPROP Analyzer" activity under the "Verify Component" Tab in coreConsultant. This directory is created only when the VCS_HOME variable is set.

For details on some key files created during coreConsultant activities, see "Database Files" on page 69

For information on using coreConsultant, see the *coreConsultant User Guide*.

2.2.2 Configuring the DesignWare Synthesizable IPs within coreConsultan

The “Parameter Description” chapter in the databook of the respective DesignWare Synthesizable IPs describe the hardware configuration parameters that you configure using the coreConsultant GUI.

The “Creating the RTL View of a Core” chapter in the *coreConsultant User Guide* discusses how to specify a configuration for an individual DesignWare Synthesizable IPs.

2.2.3 Creating Gate-Level Netlists within coreConsultant

The “Creating the Gate-Level Netlist for a Core” chapter in the *coreConsultant User Guide* discusses how to create a translation of the RTL view into a technology-specific netlist for an individual DesignWare Synthesizable IPs.

2.2.4 Verifying the DesignWare Synthesizable IPs within coreConsultant

The “Verification” chapter in the databook of the respective DesignWare Synthesizable IPs provide an overview of the testbench available for DesignWare Synthesizable IPs verification using the coreConsultant GUI.

The “Verifying Your Implementation” chapter in the *coreConsultant User Guide* discusses how to simulate an individual component.

2.3 Running Low Power Static Verification Using VC LP

You can perform low power static verification using Synopsys VC Low Power (VC LP) tool. As shown in Figure 2-2 you can run VC LP in the coreConsultant GUI.

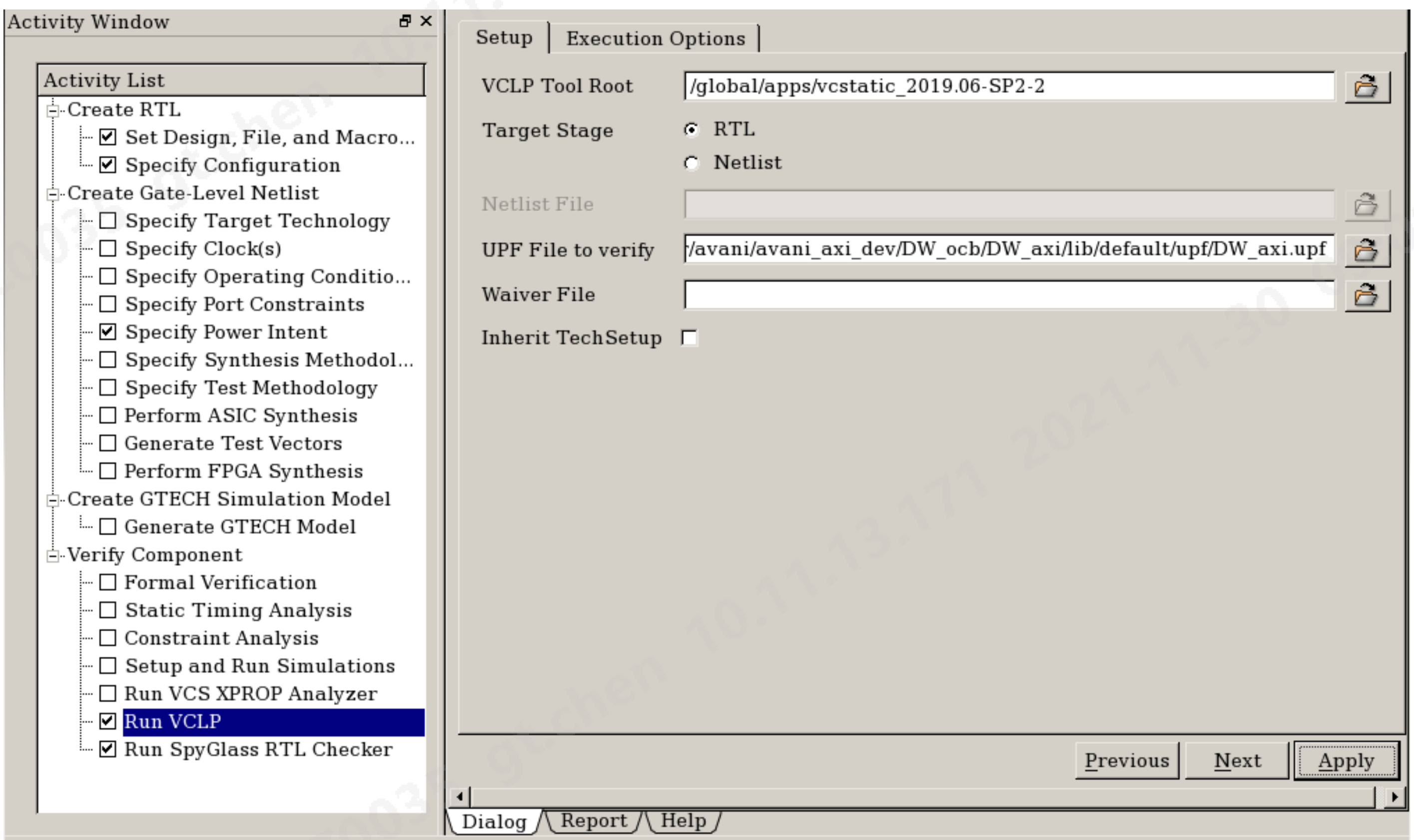
The VC LP flow in coreConsultant runs the following Low Power Checks either on the RTL or Netlist:

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

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

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

Figure 2-2 VC LP Options in coreConsultant

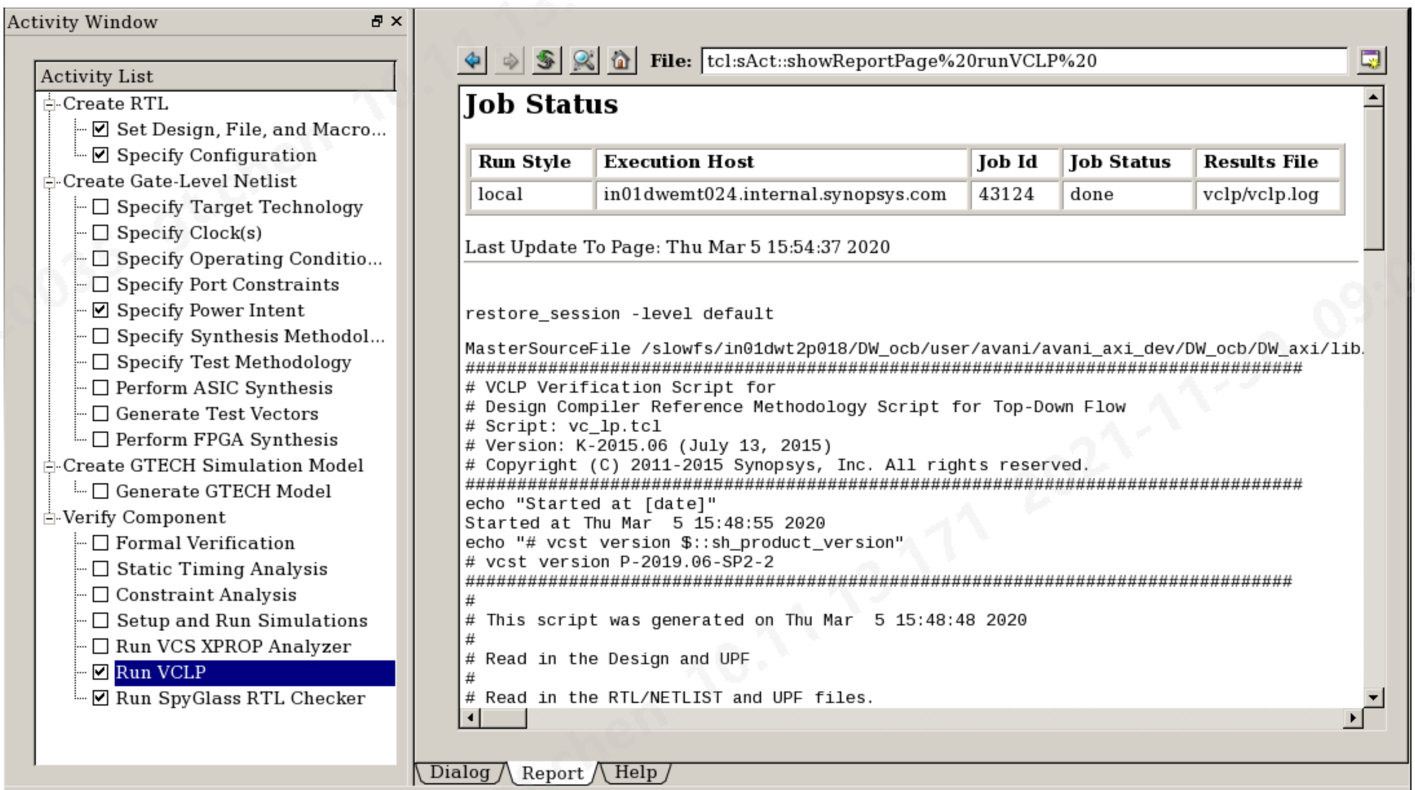


To run VC LP, complete the following steps:

1. Select **Verify Component> Run VCLP** activity.
2. In the **Setup** tab, complete the following fields:
   - **VCLP Tool Root**: Specify the complete path for the VC LP tool.
   - **Target Stage**: Select either RTL or Netlist stage.
     - For RTL stage, specify the location of the *.upf file.
     - For Netlist stage, specify the location of the netlist and the upf prime file (UPF output of Design Compiler).
3. Click **Apply**.

On completion, the results are available in the Report tab. Go to the report summary, and check for any errors/warnings as shown in Figure 2-3. For debugging the warnings/errors, refer to the *VC Low Power User Guide*.

Figure 2-3 VC LP – Sample Report Summary

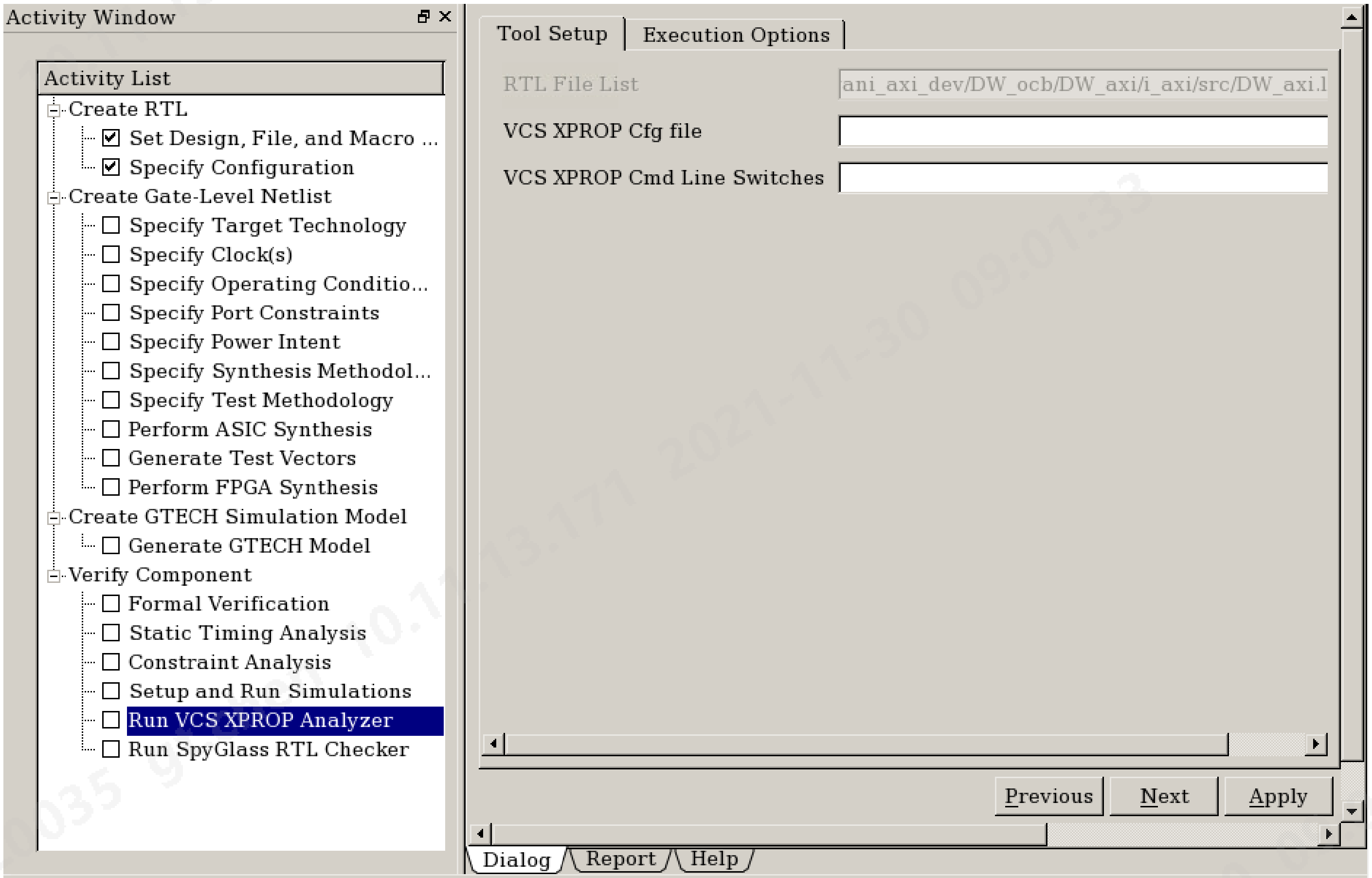


2.4 Running VCS XPROP Analyzer

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

Figure 2-4 shows the coreConsultant GUI in which you run the VCS XPROP analyzer.

Figure 2-4 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. The VCS XPROP Analyzer results are saved in the <workspace>/xprop/xprop.log file. The script <workpace>/xprop/sh.xprop contains the VCS XPROP configuration and switch settings. This can be used to run the VCS XPROP activity in batch mode.

From the "Tool Setup" tab in Figure 2-4, you can pass any command-line switches that can be supplied in addition to what is considered by default.

2.5 Overview of the coreAssembler Configuration and Integration Process

Once you have correctly downloaded and installed a release of DesignWare SIP components and then set up your environment, you can begin work on your DesignWare subsystem with coreAssembler.

2.5.1 coreAssembler Usage

Figure 2-5 illustrates some general directories and files in a coreAssembler workspace.

Figure 2-5 coreAssembler Usage Flow

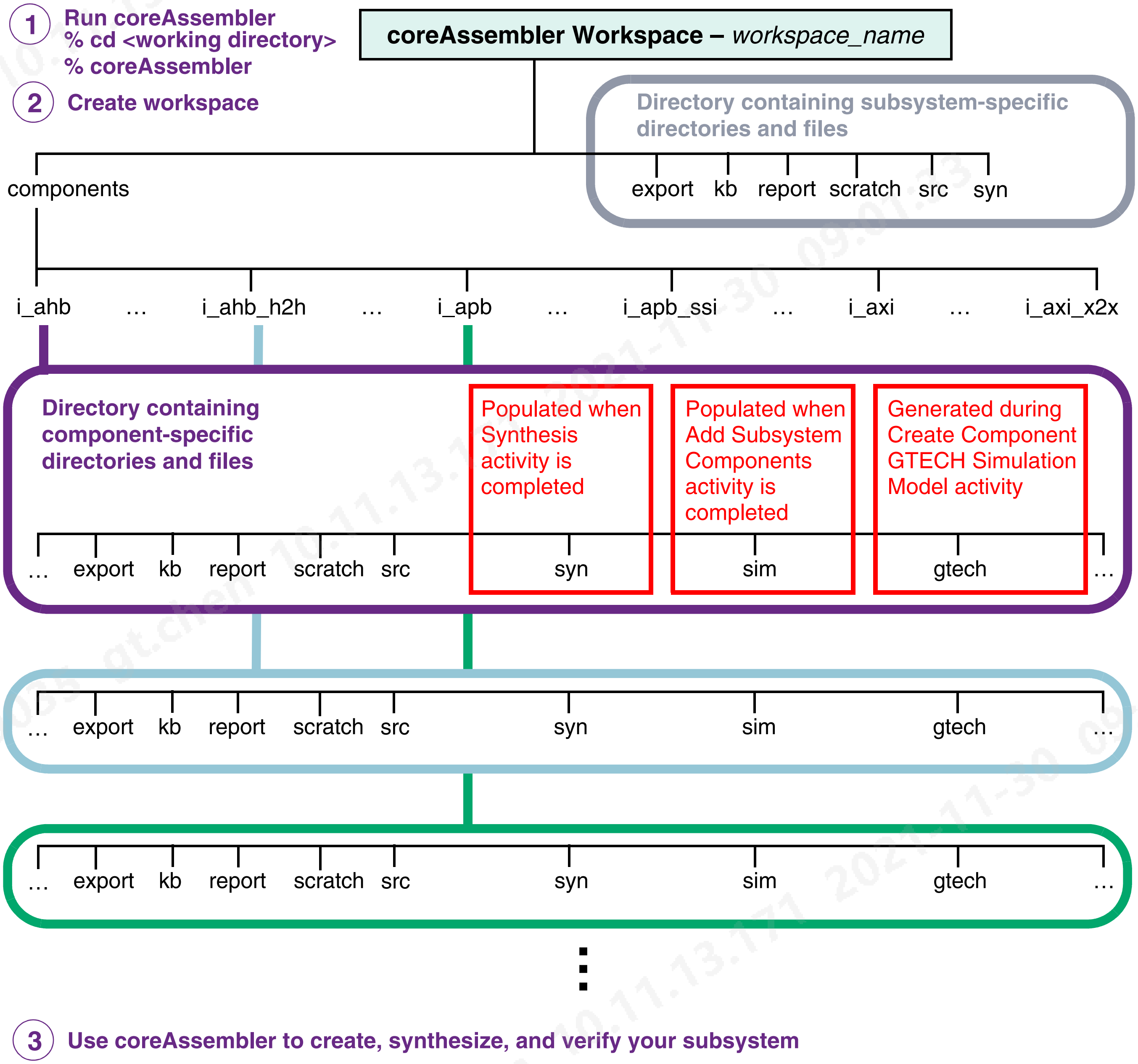


Table 2-2 provides a description of the implementation workspace directory and subdirectories.

Table 2-2 coreAssembler Implementation Workspace Directory Contents

Directory/Subdirectory	Description
components	Contains a directory for each IP component instance connected in the subsystem. Generated and populated with separate component directories upon first adding components; populated and updated throughout activities.
i_*component*/auxiliary	Scripts and text files used by coreAssembler. Generated during Add Subsystem Components activity.
i_*component*/custom	Contains RTL preprocessor scripts. Generated during Configure Components activity.
i_*component*/doc	Contains local copies of component-specific databooks. Generated during Add Subsystem Components activity.
i_*component*/export	Contains files used to integrate results from the completed source configuration and synthesis activities into your design (outside coreAssembler). Generated during Add Subsystem Components activity; populated during Configure Components activity.
i_*component*/gtech	Contains synthesis scripts and output netlists from gtech generation; also used for RTL simulation of encrypted source code. Generated during Create Component GTECH Simulation Model activity.
i_*component*/kb	Contains knowledge base information used by coreAssembler. These are binary files containing the state of the design. Generated during Add Subsystem Components activity; populated and updated throughout activities.
i_*component*/report	Contains all of the reports created by coreAssembler during build, configuration, test and synthesis phases. An index.html file in this directory links to many of these generated reports. Generated during Add Subsystem Components activity; populated and updated throughout activities.
i_*component*/scratch	Contains temp files used during the coreAssembler processes. Generated during Add Subsystem Components activity; populated and updated throughout activities.
i_*component*/sim	Contains test stimulus and output files. Generated during Add Subsystem Components activity; updated during Setup and Run Simulations (for /i_*component*) activity.
i_component/spyglass	Contains SpyGlass Lint and CDC configuration files for the component. Generated upon first SpyGlass run; updated during Run Spyglass RTL Checker activity.
i_*component*/src	Includes the top-level RTL file, *design_name*.v. If you have a source license, this contains plain-text RTL; if you only have a designware license, this contains encrypted RTL. Generated during Add Subsystem Components activity; populated during Specify Configuration activity.

Directory/Subdirectory	Description
i_*component*/syn	Contains synthesis files for the component. Generated during Add Subsystem Components activity; updated during Synthesis activity.
i_*component*/tcl	Contains synthesis intent scripts. Generated during Add Subsystem Components activity.
i_*component*/xprop	Contains the files used for the VCS Xprop analysis activity. Generated upon running the “Run VCS XPROP Analyzer” activity. This directory is created only when the VCS_HOME variable is set.
export	Contains subsystem files used to integrate the results from the completed source configuration and synthesis activities into your design (outside coreAssembler). Generated upon first creating workspace; populated starting with Memory Map Specification activity.
kb	Contains subsystem knowledge base information used by coreAssembler. These are binary files containing the state of the design. Generated upon first creating workspace; populated and updated throughout activities.
report	Contains subsystem reports created by coreAssembler 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 subsystem temp files used during the coreAssembler processes. Generated upon first creating workspace; populated and updated throughout activities.
src	Includes the RTL related to the subsystem. If you have a source license, this contains plain-text RTL; if you only have a designware license, this contains encrypted RTL. Generated upon first creating workspace; populated starting with Generate Subsystem RTL activity.
syn	Contains synthesis files for the subsystem. Generated upon first creating workspace; updated during Synthesize activity and Formal Verification activity.

- For details on some key files created during coreAssembler activities, see “Database Files” on page 69.
- For information on using coreAssembler, see the *coreAssembler User Guide*.
- For information on getting started with using DesignWare SIP components for AMBA 3 AXI, and AMBA 4 AXI components within coreTools, see *Using DesignWare Library IP in coreAssembler*.

2.5.2 DesignWare Synthesizable IPs within Simple Subsystems

“DW_axi in Simple Subsystem”

“DW_axi_a2x in Simple Subsystem”

"DW_axi_dmac in Simple Subsystem"

"DW_axi_gm in Simple Subsystem"

"DW_axi_gs in Simple Subsystem"

"DW_axi_hmx in Simple Subsystem"

"DW_axi_rs in Simple Subsystem"

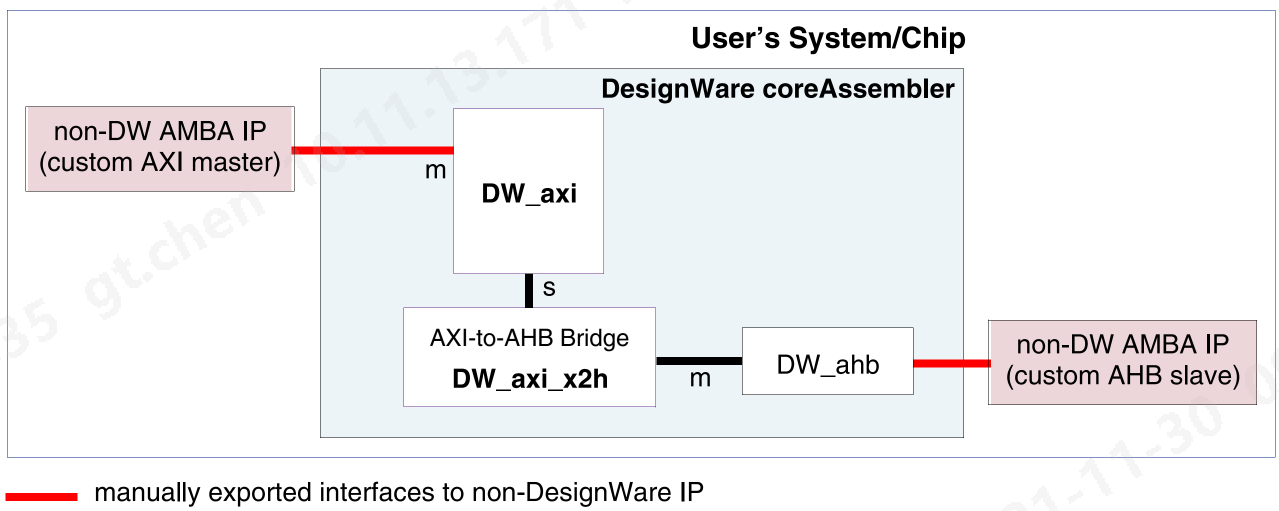
"DW_axi_x2h in Simple Subsystem"

"DW_axi_x2p in Simple Subsystem"

"DW_axi_x2x in Simple Subsystem"

Figure 2-6 DW_axi in Simple Subsystem

Figure 2-6 illustrates the DW_axi components in a simple subsystem.



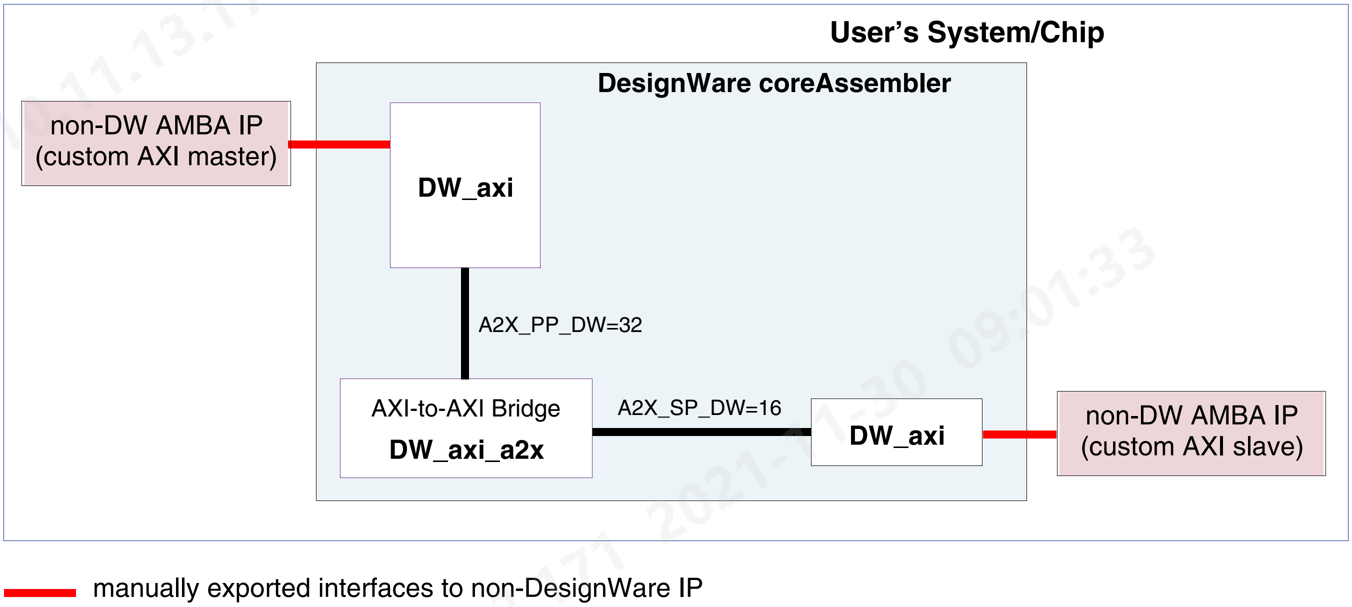
The subsystem in Figure 2-6 contains the following components that you may want to use as you learn to use coreAssembler:

- DW_axi
- DW_axi_x2h
- DW_ahb
- AXI Master
- AHB Slave

The AXI Master and AHB Slave are meant to be exported out of the design and then replaced by a real AXI Master—such as a CPU—and a real AHB Slave—such as an interrupt controller or UART—later in the design process; at least one exported AXI master is required in a subsystem if you intend to do a basic simulation that tests connections.

Figure 2-7 DW_axi_a2x in Simple Subsystem

Figure 2-7 illustrates the DW_axi_a2x in another subsystem.



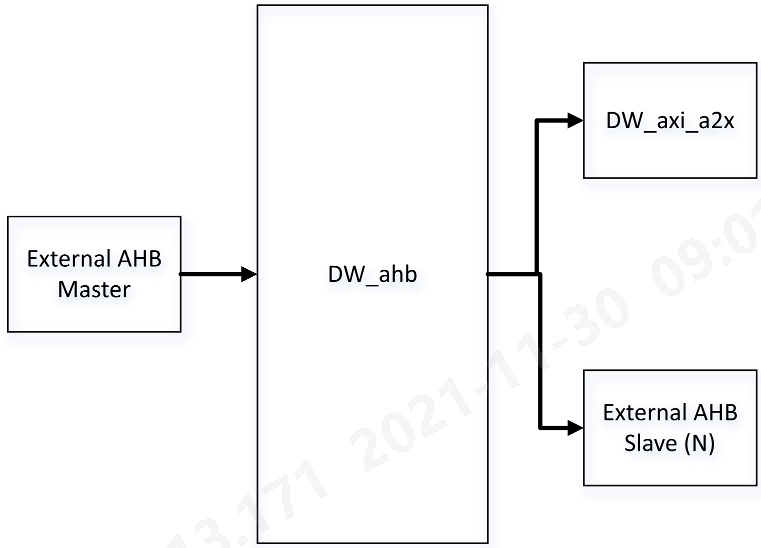
The subsystem in Figure 2-7 contains the following components:

- DW_axi_a2x
- Two DW_axi components
- AXI Master
- AXI Slave

hmaster Connectivity in DW_axi_a2x

For the following sub-system:

Figure 2-8 DW_ahb and DW_axi_a2x Configured for AHB Lite Mode Operation

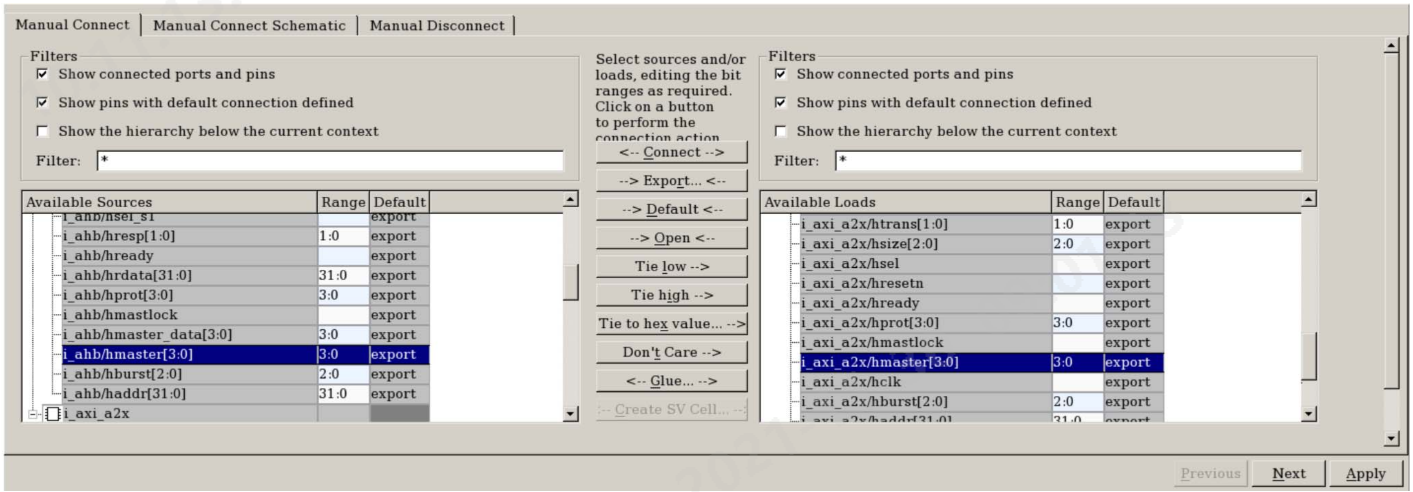


If DW_ahb and DW_axi_a2x are configured for AHB Lite mode operation, then the SplitCapable in the Configure Interfaces Tab must be set to 0 to match the expectation of the IP. By default, coreAssembler exports the i_ahb/hmaster output unconnected and i_axi_a2x/hmaster input to be exported at the top level.

If you want to connect these two signals, then during Complete Connections activity, connect them manually using the following steps:

1. Instantiate the DW_ahb and DW_axi_a2x, and connect them in the Specify Subsystem sub-activity under the Generate RTL activity.
2. As DW_ahb/hmaster are not connected to DW_axi_a2x/hmaster because SplitCapable is set to 0, you need to visit the Complete Connections sub-activity and explicitly create connection based on your requirement.

Figure 2-9 Connections Sub-activity

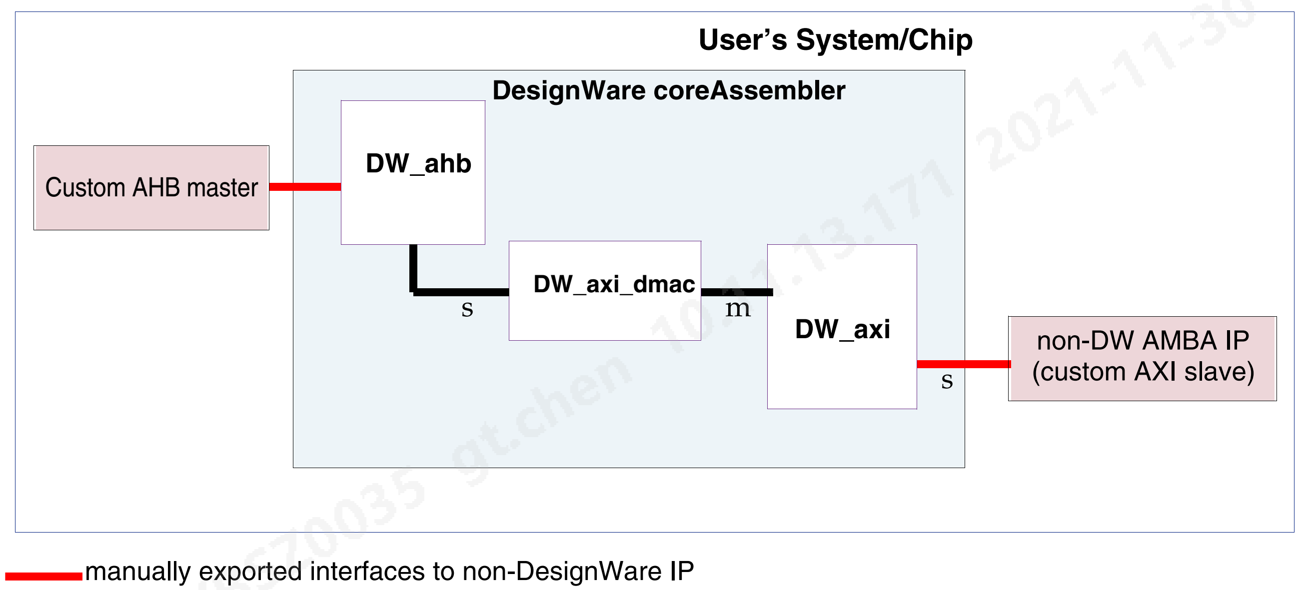


3. In the batch mode perform the following commands:

   - ❑ `autocomplete_activity SpecifySubsystem`
   - ❑ `set_current_component /`
   - ❑ `create_connection {{i_ahb/hmaster[3:0]} {i_axi_a2x/hmaster[3:0]}}`
   - ❑ `autocomplete_activity CompleteConnections`

Figure 2-10 DW_axi_dmac in Simple Subsystem

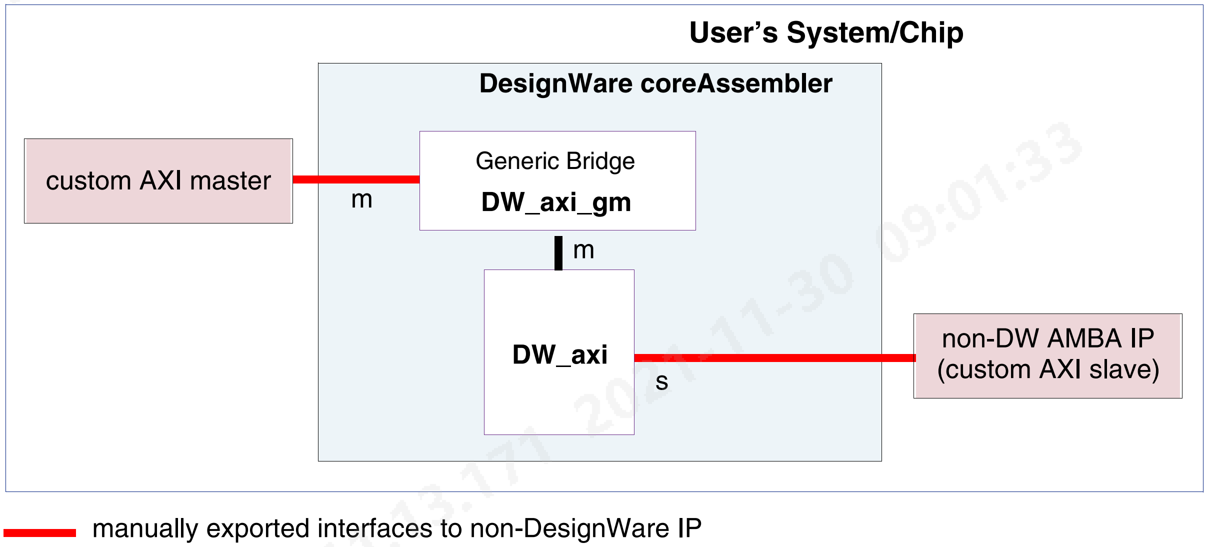
Figure 2-10 illustrates the DW_axi_gm in a simple subsystem.



The subsystem in Figure 2-10 contains the following components that you may want to use as you learn to use coreAssembler:

Figure 2-11 DW_axi_gm in Simple Subsystem

Figure 2-11 illustrates the DW_axi_gm in a simple subsystem.



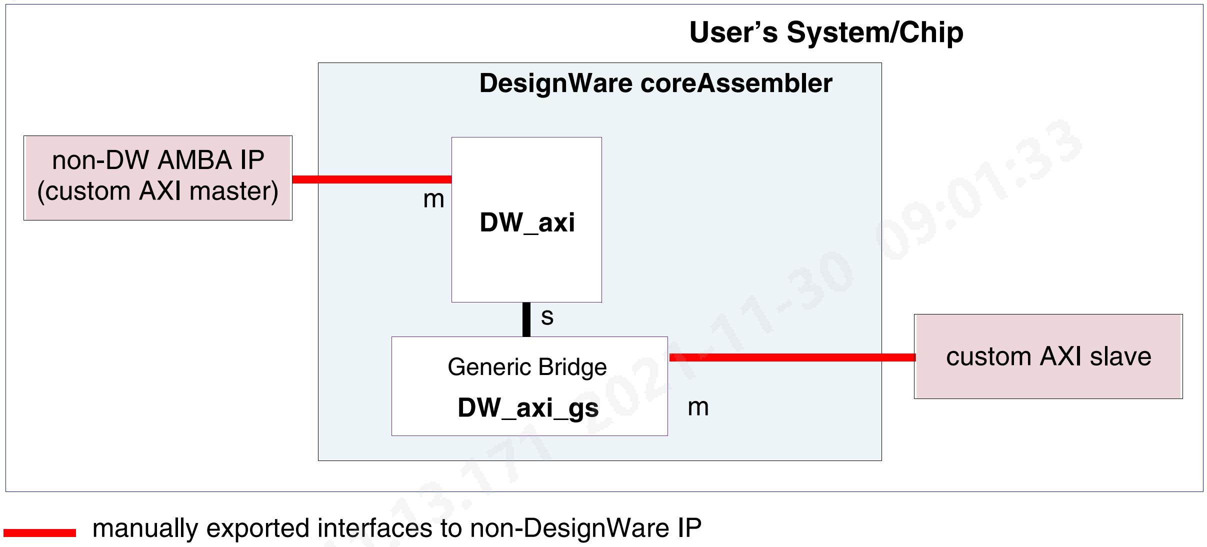
The subsystem in Figure 2-11 contains the following components that you may want to use as you learn to use coreAssembler:

- DW_axi_gm
- DW_axi
- GIF master
- AXI slave

The GIF master and AXI slave are meant to be exported out of the design and then replaced by a real AXI slave — such as a bridge — and a real AXI master — such as a CPU — later in the design process; at least one exported AXI master is required in a subsystem if you intend to do a basic simulation that tests connections.

Figure 2-12 DW_axi_gs in Simple Subsystem

Figure 2-12 illustrates the DW_axi_gs in a simple subsystem.



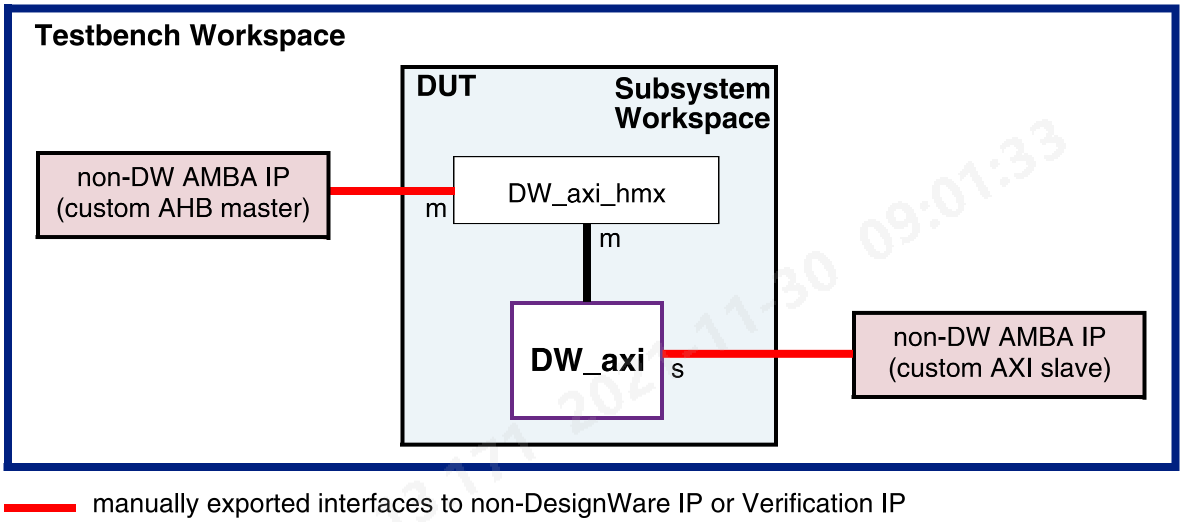
The subsystem in Figure 2-12 contains the following components that you may want to use as you learn to use coreAssembler:

- DW_axi
- AXI master
- GIF slave

The AXI master and the GIF slave are meant to be exported out of the design and then replaced by a real AXI master — such as a CPU — and a real AXI slave — such as an interrupt controller or UART — later in the design process; at least one exported AXI master is required in a subsystem if you intend to do a basic simulation that tests connections.

Figure 2-13 DW_axi_hmx in Simple Subsystem

Figure 2-13 illustrates the DW_axi_hmx in a simple subsystem.



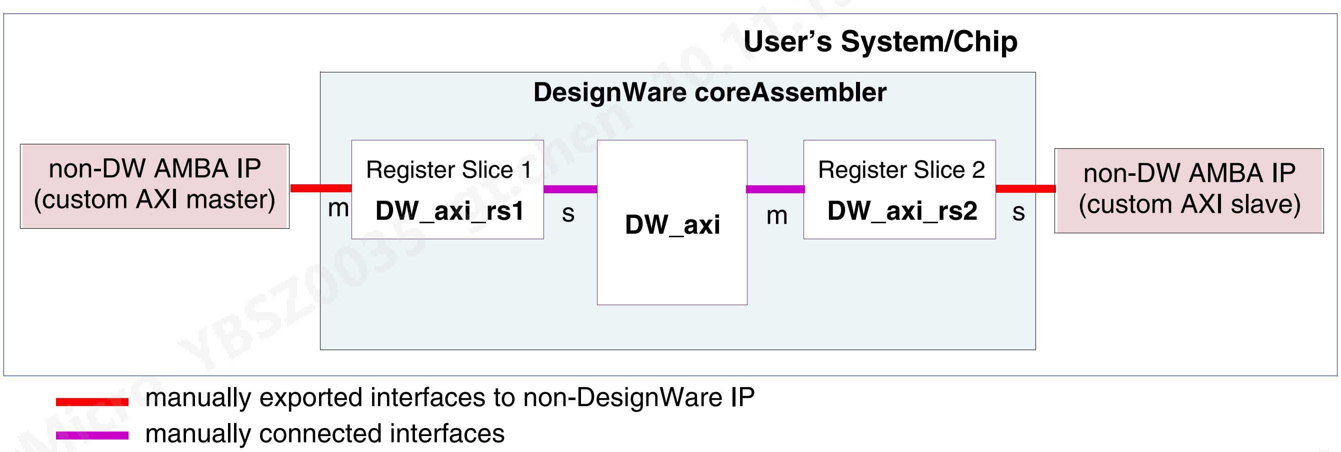
The subsystem in Figure 2-13 contains the following components that you may want to use as you learn to use coreAssembler:

- DW_axi_hmx
- DW_axi
- AHB master
- AXI slave

The AHB master and AXI slave are meant to be exported out of the design and then replaced by a real AHB master — such as a CPU — and a real AXI slave later in the design process; at least one exported AHB master is required in a subsystem if you intend to do a basic simulation that tests connections.

Figure 2-14 DW_axi_rs in Simple Subsystem

Figure 2-14 illustrates the DW_axi_rs in a simple subsystem.



The subsystem in Figure 2-14 contains the following components that you may want to use as you learn to use coreAssembler:

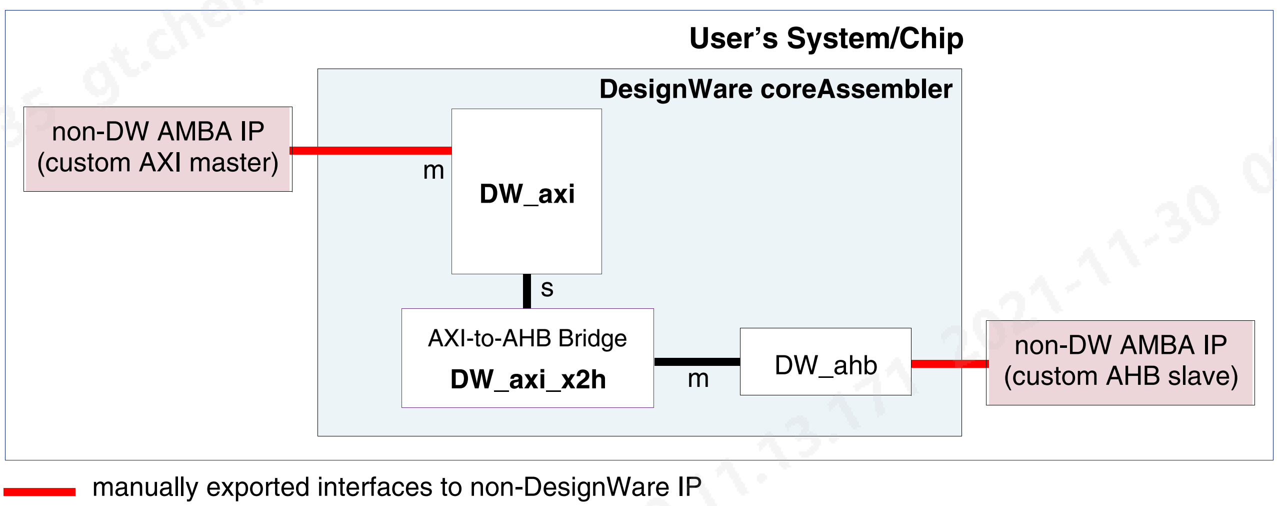
- DW_axi_rs
- DW_axi
- AXI master
- AXI slave

The AXI master and AXI slave are meant to be exported out of the design and then replaced by a real AXI master and AXI slave—such as a CPU and an AXI slave, respectively—later in the design process; at least one exported AXI master is required in a subsystem if you intend to do a basic simulation that tests connections.

Note It is recommended to use DW_axi_rs between two AXI components to ease the timing, and increase the circuit reliability. DW_axi_rs gates the payload of the respective AXI channel by considering the corresponding AXI valid signal.

Figure 2-15 DW_axi_x2h in Simple Subsystem

Figure 2-15 illustrates the DW_axi_x2h in a simple subsystem.



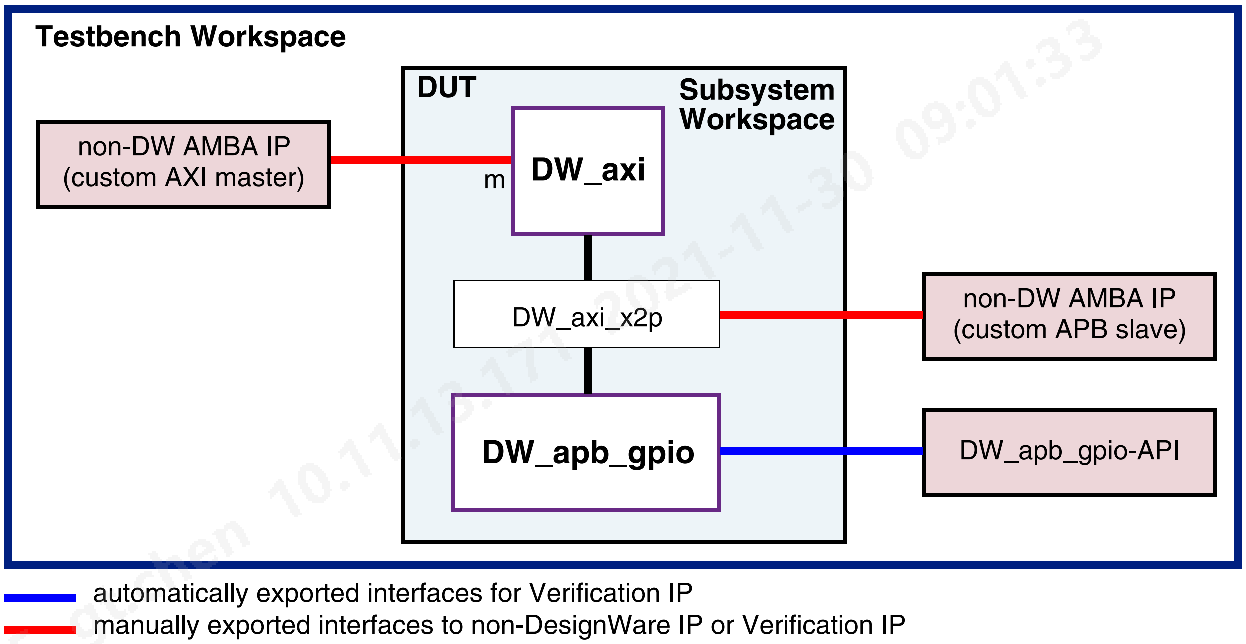
The subsystem in Figure 2-15 contains the following components that you may want to use as you learn to use coreAssembler:

- DW_axi_x2h
- DW_axi
- DW_ahb
- AXI master
- AHB slave

The AXI master and AHB slave are meant to be exported out of the design and then replaced by a real AXI master — such as a CPU — and a real AHB slave — such as an interrupt controller or UART — later in the design process; at least one exported AXI master is required in a subsystem if you intend to do a basic simulation that tests connections.

Figure 2-16 DW_axi_x2p in Simple Subsystem

Figure 2-16 illustrates the DW_axi_x2p in a simple subsystem.



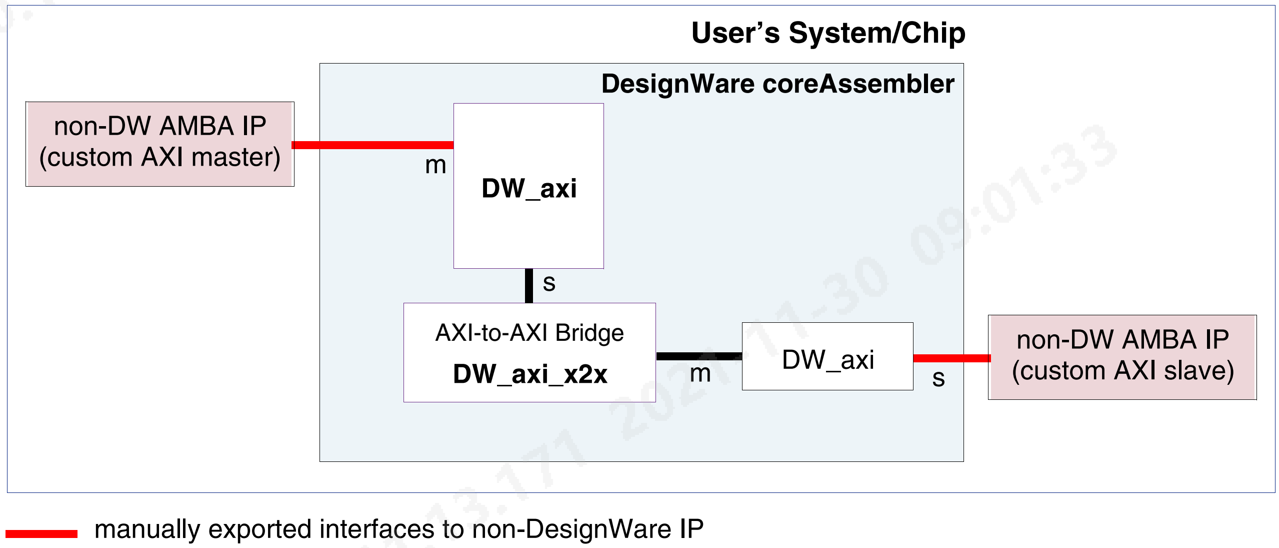
The subsystem in Figure 2-16 contains the following components that you may want to use as you learn to use coreAssembler:

- DW_axi_x2p
- DW_axi
- DW_apb_gpio
- APB slave
- AXI master

The AXI master is meant to be exported out of the design and then replaced by a real AXI master — such as a CPU — later in the design process; at least one exported AXI master is required in a subsystem if you intend to do a basic simulation that tests connections.

Figure 2-17 DW_axi_x2x in Simple Subsystem

Figure 2-17 illustrates the DW_axi_x2x in a simple subsystem.



The subsystem in Figure 2-17 contains the following components that you may want to use as you learn to use coreAssembler:

- DW_axi_x2x
- Two DW_axi components
- AXI Master
- AXI Slave

The AXI Master and AXI Slave are meant to be exported out of the design and then replaced by a real AXI Master—such as a CPU—and a real AXI Slave later in the design process; at least one exported AXI master is required in a subsystem if you intend to do a basic simulation that tests connections.

2.5.3 Configuring the DesignWare Synthesizable IPs within a Subsystem

The "Parameter Description" chapter in the databook of the respective DesignWare Synthesizable IPs describe the hardware configuration parameters that you configure using the coreAssembler GUI.

The "Creating the RTL View of a Subsystem" chapter in the *coreAssembler User Guide* discusses how to configure subsystem components and automatically connect them using the coreAssembler GUI.

2.5.4 Creating Gate-Level Netlists within coreAssembler

The "Creating the Gate-Level Netlist for a Subsystem" chapter in the *coreAssembler User Guide* discusses how to create a translation of the RTL view into a technology-specific netlist for a subsystem.

2.5.5 Verifying the DesignWare Synthesizable IPs within coreAssembler

The "Verification" chapter in the databook of the respective DesignWare Synthesizable IPs provides an overview of the testbench available for verification using the coreAssembler GUI.

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

2.5.6 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.

2.6 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 DesignWare Synthesizable IPs using coreConsultant. This section describes how to configure, simulate and synthesize the core using coreConsultant.

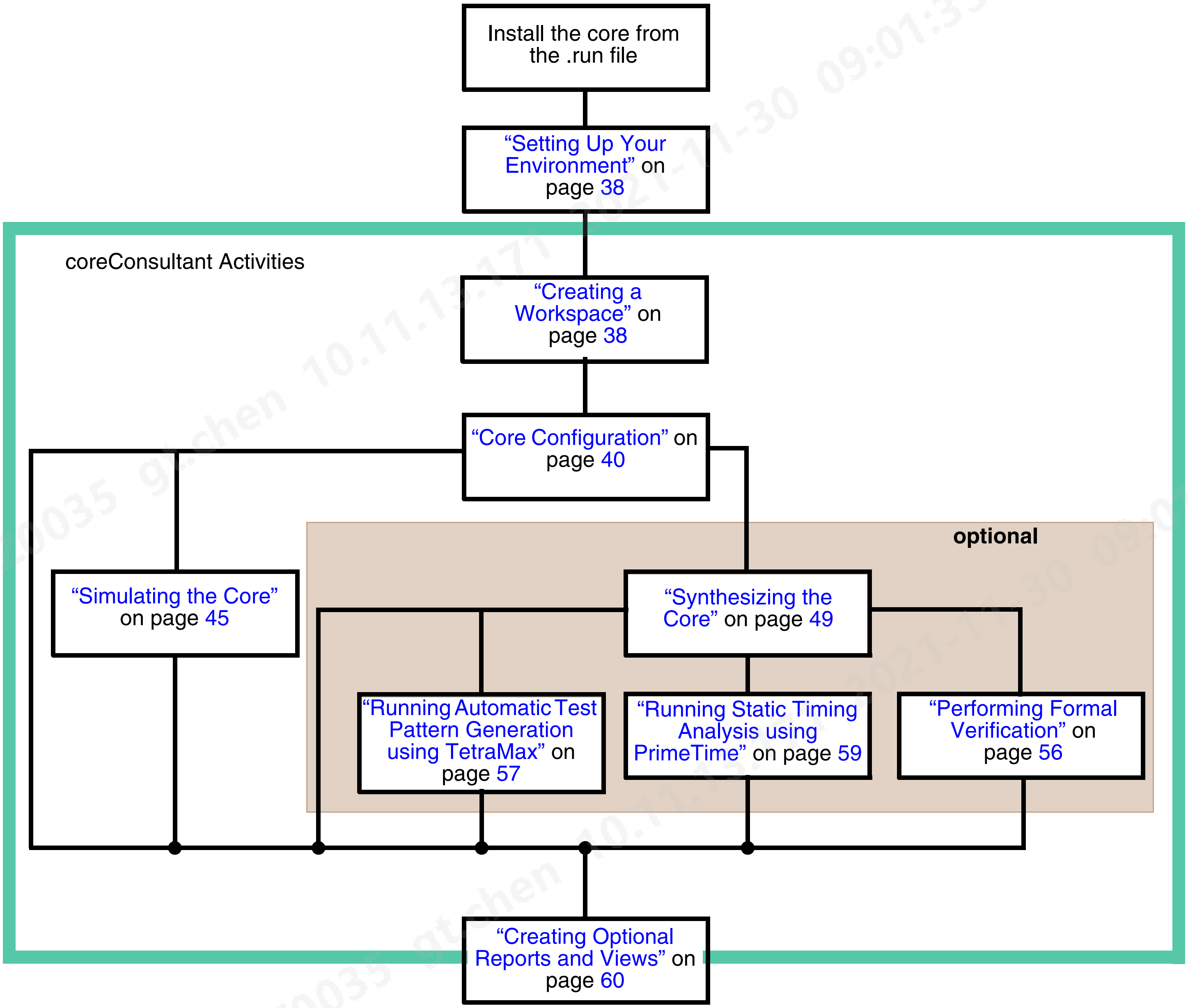
This section discuses the following topics:

- "Design Flow" on page 37
- "Setting Up Your Environment" on page 38
- "Creating a Workspace" on page 38
- "Core Configuration" on page 40
- "Running SpyGlass® Lint and SpyGlass® CDC" on page 43
- "Simulating the Core" on page 45
- "Synthesizing the Core" on page 49
- "Performing Formal Verification" on page 56
- "Inserting Design For Test Using Design Compiler" on page 57
- "Running Automatic Test Pattern Generation using TetraMax" on page 57
- "Running Static Timing Analysis using PrimeTime" on page 59
- "Creating Optional Reports and Views" on page 60
- "Exporting a Core to Your Chip Design Database" on page 63

2.6.1 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 2-18 and do not need to be completed before exporting the configured RTL to your chip design flow.

Figure 2-18 Design Flow



2.6.2 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 DesignWare Synthesizable Components for AMBA 2, AMBA 3 AXI 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 DesignWare Synthesizable IPs-specific 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.

2.6.3 Creating a Workspace

A workspace is a local Unix directory structure containing your configured copy of the DesignWare Synthesizable IPs. For more details about this directory structure, see "coreConsultant Usage" on page 16. You can create several workspaces to experiment with different design alternatives.

To create a workspace, follow these steps:

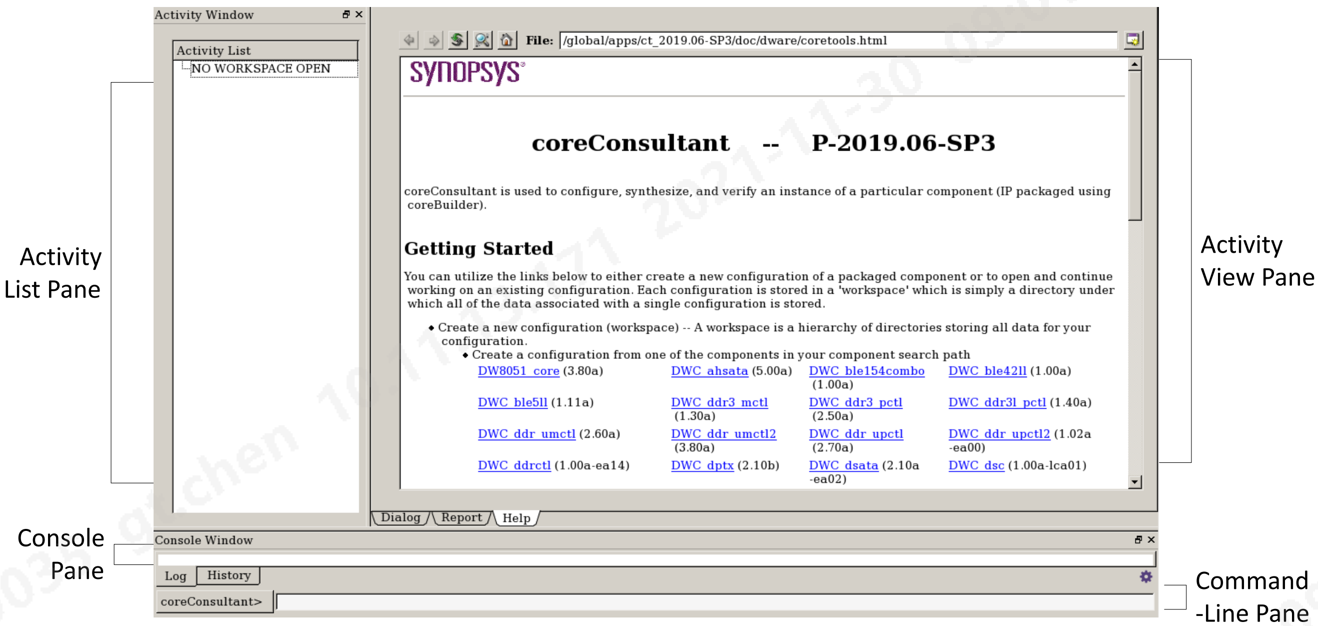
Attention All screens in this section are for the illustration purpose only. They may differ from the actual coreConsultant screens depending on the selected DesignWare Component.

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

```
% coreConsultant &
```

Figure 2-19 shows the initial coreConsultant screen.

Figure 2-19 Initial coreConsultant Screen



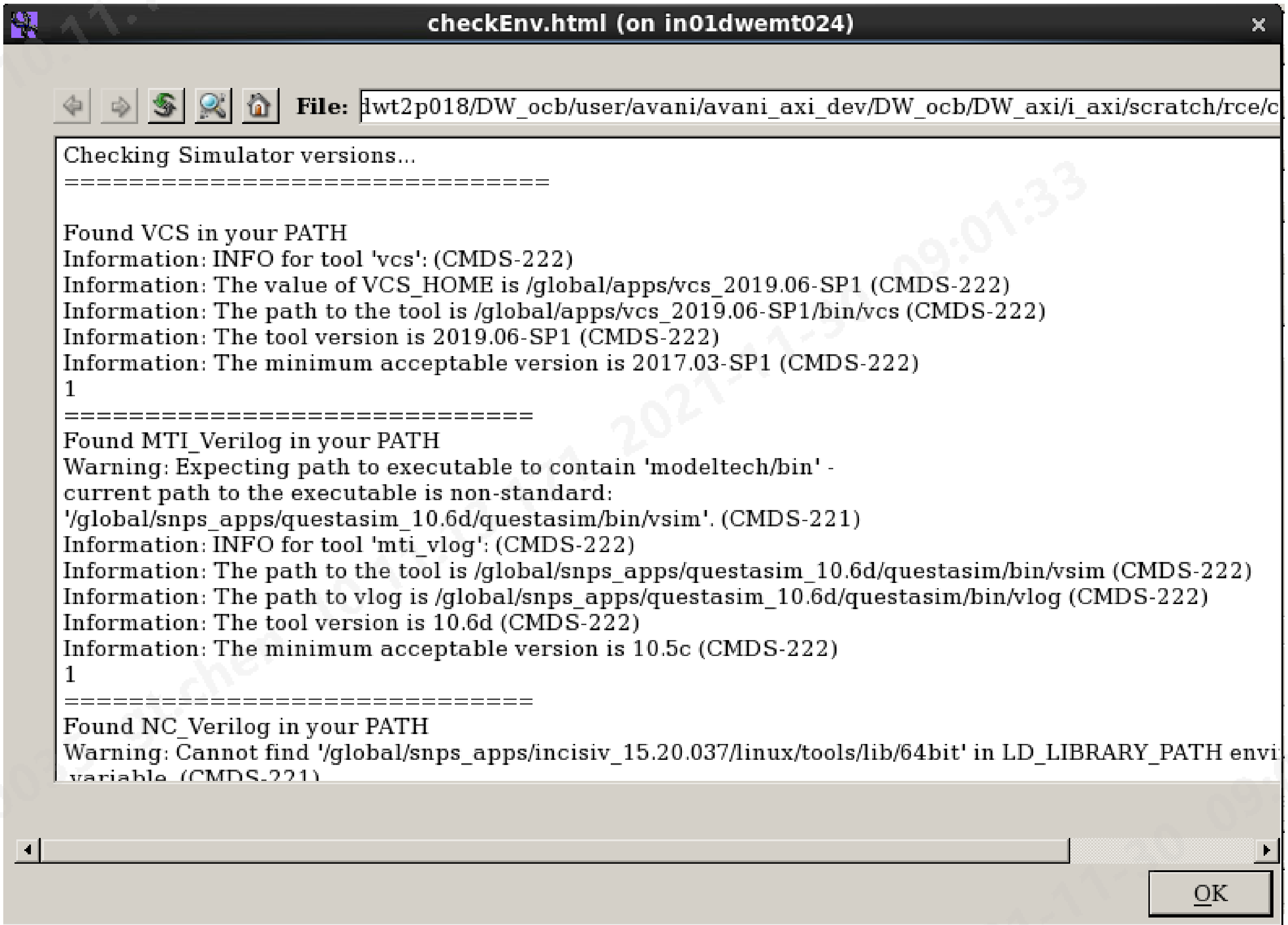
3. Create a workspace.

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 DesignWare Synthesizable IPs 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, see "Configuring the Core Using coreConsultant" on page 36

4. Validate your installation and environment by selecting **Help->Check Environment** from the menu bar. A report window (Figure 2-20) shows the results.

Figure 2-20 Sample Results of the Check Environment Activity



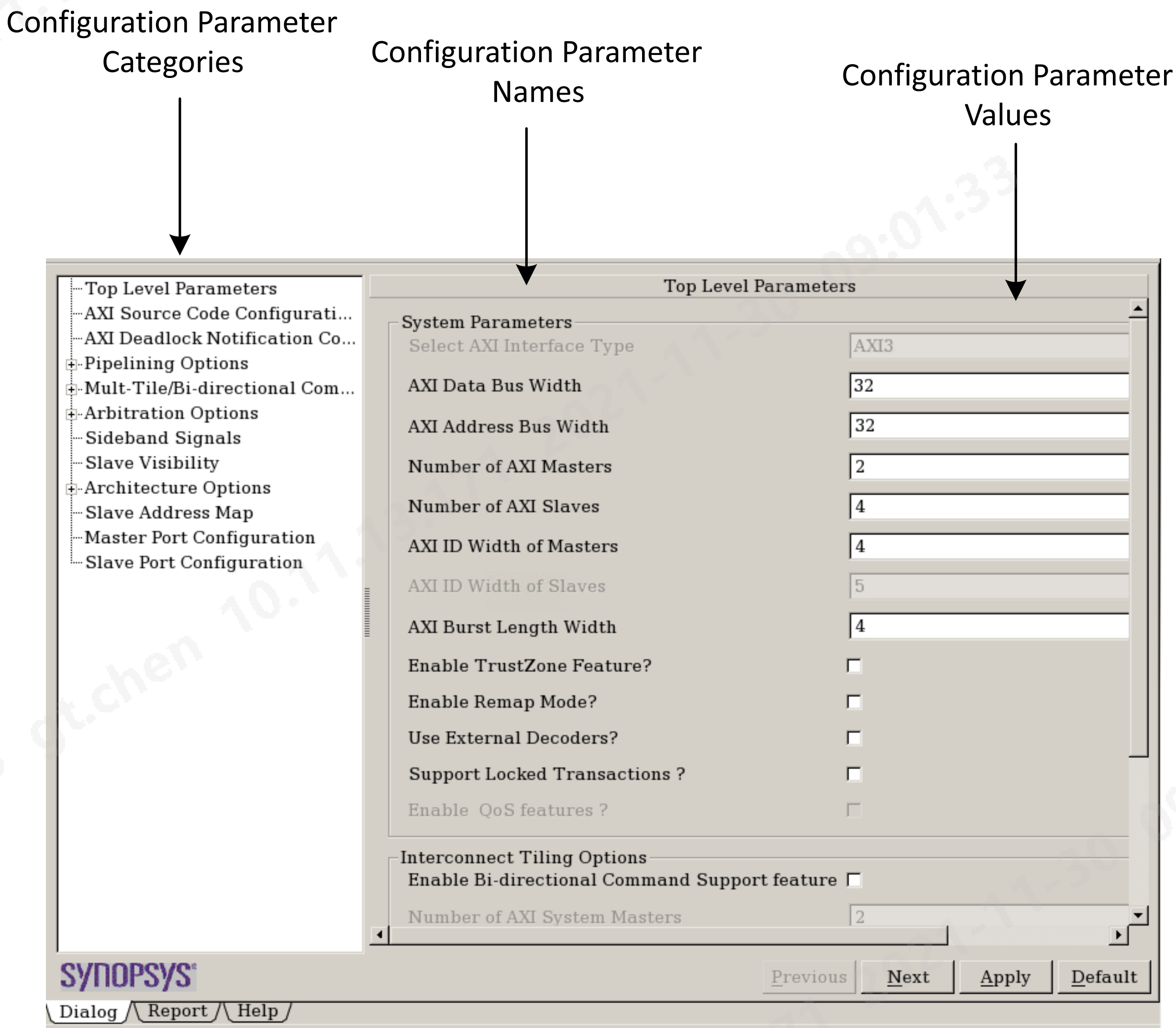
Correct any reported errors by modifying your UNIX environment setup (see "Configuring the Core Using coreConsultant" on page 36) or through the **Edit > Tool Installation Roots** menu.

2.6.4 Core Configuration

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 2-21.

Attention	All screens in this section are for the illustration purpose only. They may differ from the actual coreConsultant screens depending on the selected DesignWare Component.

Figure 2-21 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, see the “Parameters” chapter in the *DesignWare Cores Synchronous Serial Interface Databook*.

> **Note** 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.

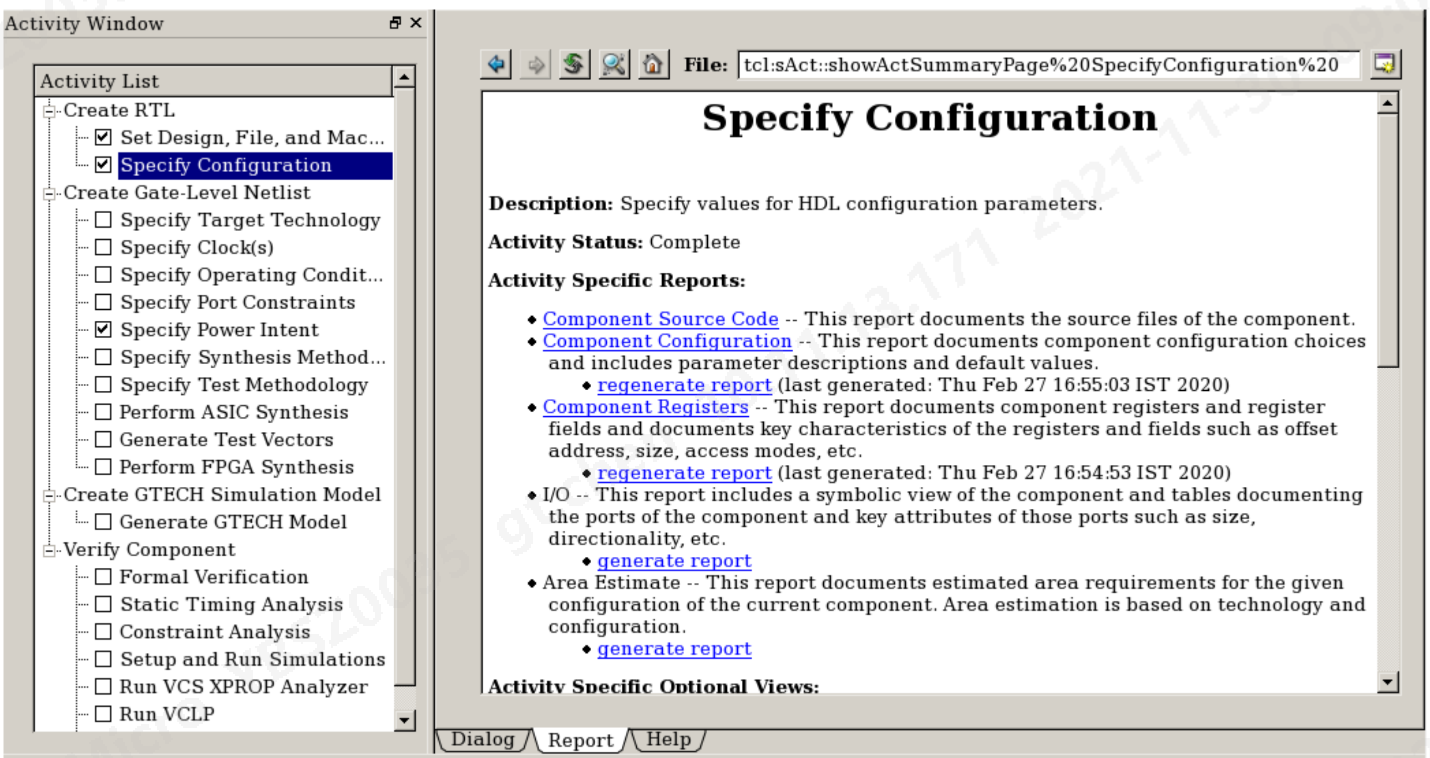
   You can return to the Configuration activity to configure the core again (create new RTL) at any time.

> **Hint** 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 2-22 on page 42).

Figure 2-22 Configuration Reports



Also, you can generate Activity Specific Reports and Activity Specific Optional Views. For more information, see "Creating Optional Reports and Views" on page 60. For example, to generate an example component instantiation, click the **generate view** link as indicated in Figure 2-22.

Note

- If any problems occur during or after the Specify Configuration activity, see "Troubleshooting" on page 74.
- At this point, you can verify the core ("Simulating the Core" on page 45) or generate a gate-level netlist ("Synthesizing the Core" on page 49).

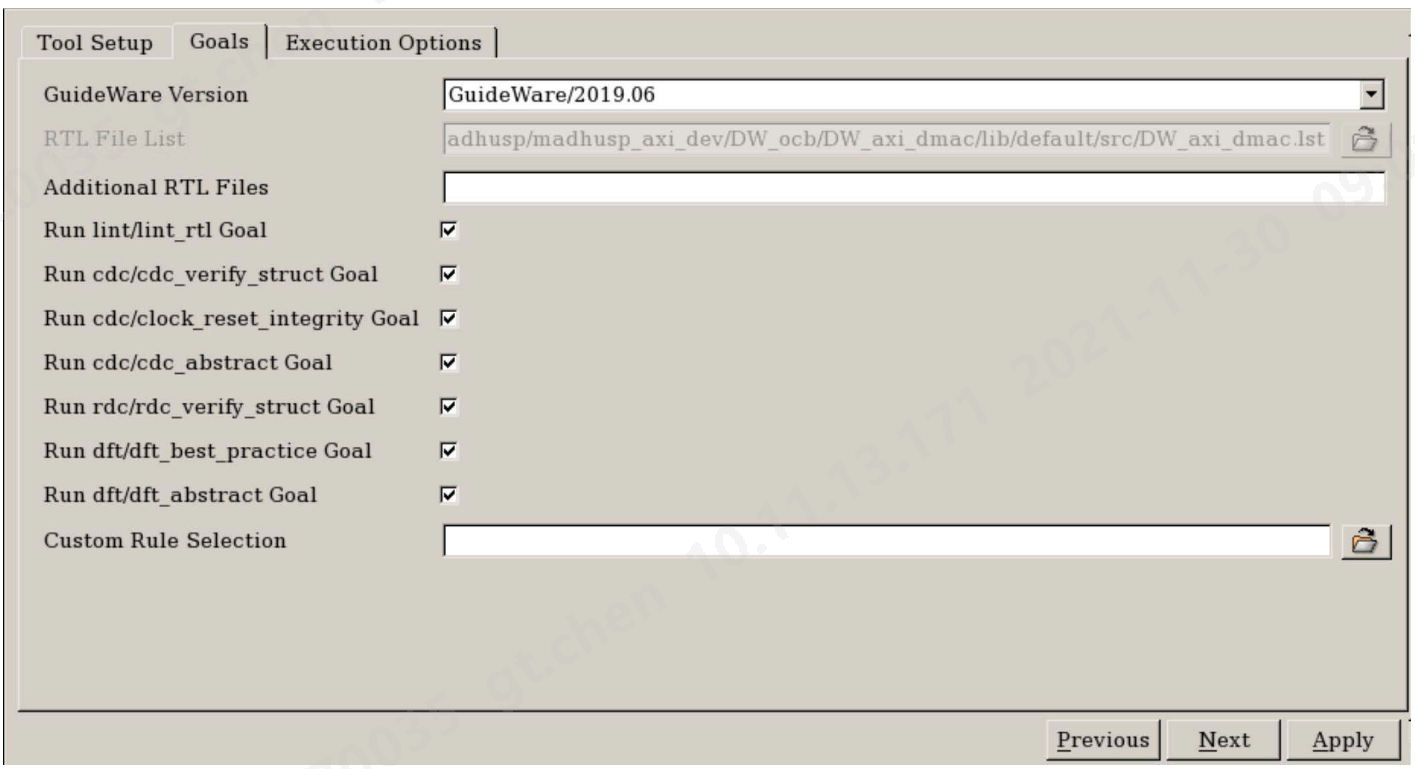
2.6.5 Running SpyGlass® Lint and SpyGlass® CDC

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

Attention All screens in this section are for the illustration purpose only. They may differ from the actual coreConsultant screens depending on the selected DesignWare Component.

Figure 2-23 shows the sample screen of coreConsultant GUI in which you run Lint and CDC goals.

Figure 2-23 SpyGlass Options in coreConsultant

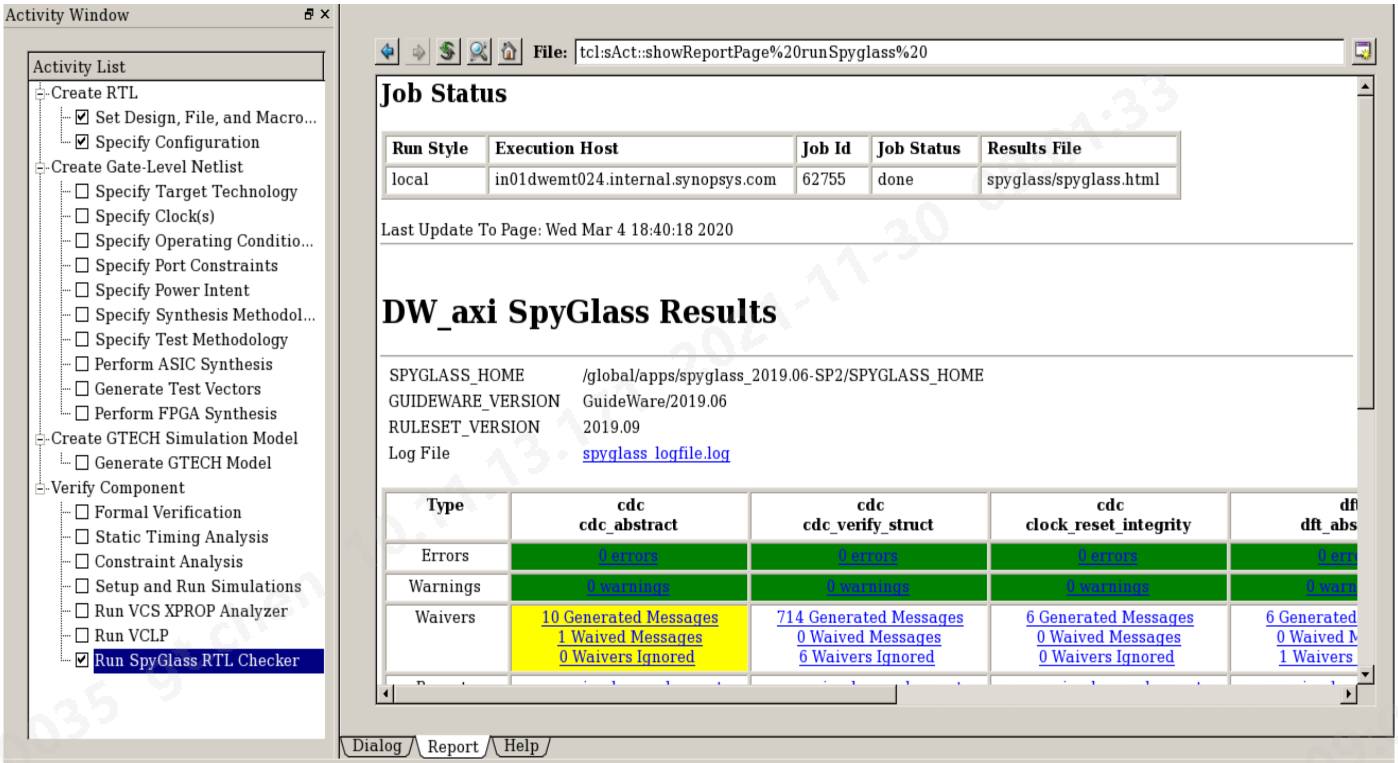


Within the block/rtl_handoff, only lint/lint_rtl and cdc/cdc_verify_struct goals are run.

In Figure 2-23, 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 2-24.

Figure 2-24 coreConsultant SpyGlass Sample Report Summary



2.6.5.1 Fixed Settings

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

2.6.5.2 SpyGlass Lint

Table 2-3 lists the SpyGlass Link waiver files that are used by the coreConsultant tool.

Table 2-3 Waiver Files for Sypglass Lint

File Name	Description
<configured_workspace>/spyglass/spyglass_design_specific_waivers.swl	These are DesignWare Synthesizable IPs design-specific rule waivers. This file contains Lint waivers for DesignWare Synthesizable IPs (if applicable). The reason for each of the waivers (if any) are included as comments in the file.
<configured_workspace>/spyglass/spyglass_engineering_council_rules.tcl	This file contains rules that Synopsys waives for its IPs.

2.6.5.3 SpyGlass CDC

To define the SpyGlass CDC constraints, it is important to understand the reset and clock logic used in DesignWare Cores. For information on reset and clock logic, see <component name> databook.

2.6.5.3.1 CDC Files

Table 2-4 summarizes files for SpyGlass CDC used by coreConsultant.

Table 2-4 Constraint and Waiver Files for Sypglass CDC

File Name	Description
<configured_workspace>/spyglass/manual.sgdc	These are the constraints pertaining to a given configuration.
<configured_workspace>/spyglass/ports.sgdc	These are the list of I/O signals.
<configured_workspace>/spyglass/clocks.sgdc	These are the list of respective clocks.
<configured_workspace>/spyglass/resets.sgdc	These are the list of respective resets.
<configured_workspace>/spyglass/spyglass_design_specific_waivers.swl	These are DesignWare Synthesizable IPs design-specific rule waivers. This file contains CDC waivers for DesignWare Synthesizable IPs (if applicable). The reason for each of the waivers (if any) are included as comments in the file.
<configured_workspace>/spyglass/spyglass_engineering_council_rules.tcl	These are rules that Synopsys waives for its IPs.

2.6.5.3.2 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.

Navigate to and select the relevant errors to open a schematic for analysis.

2.6.6 Simulating the Core

This section shows you how to simulate the DesignWare Synthesizable IPs in the coreConsultant environment. You can simulate the core RTL in the supplied 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 36). 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.

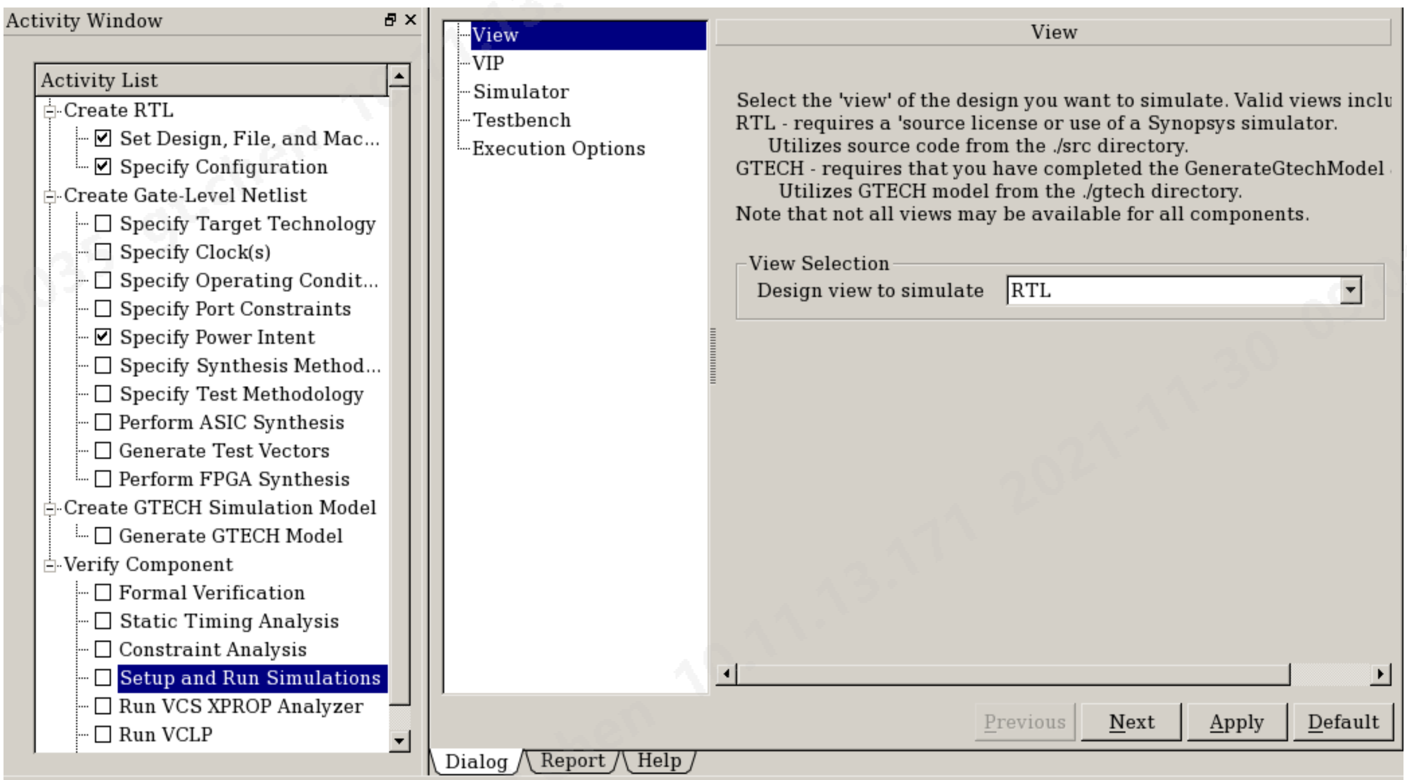
The steps involved in simulating the DesignWare Synthesizable IPs are:

- "Running the Simulation"
- "Checking Simulation Status and Results" on page 47

2.6.6.1 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 2-25 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 2-25 to reset all DesignWare Synthesizable IPs 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 2-25, 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="WS6.0|WS7.0",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.

2.6.6.2 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 2-26 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 2-25 on page 46, 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.

2.6.6.3 Location of Simulation Files

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

Table 2-5 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

2.6.6.4 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.

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

2.6.7 Synthesizing the Core

The coreConsultant tool is your interface to Synopsys synthesis tools for ASIC 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 60.

The topics in this section are as follows:

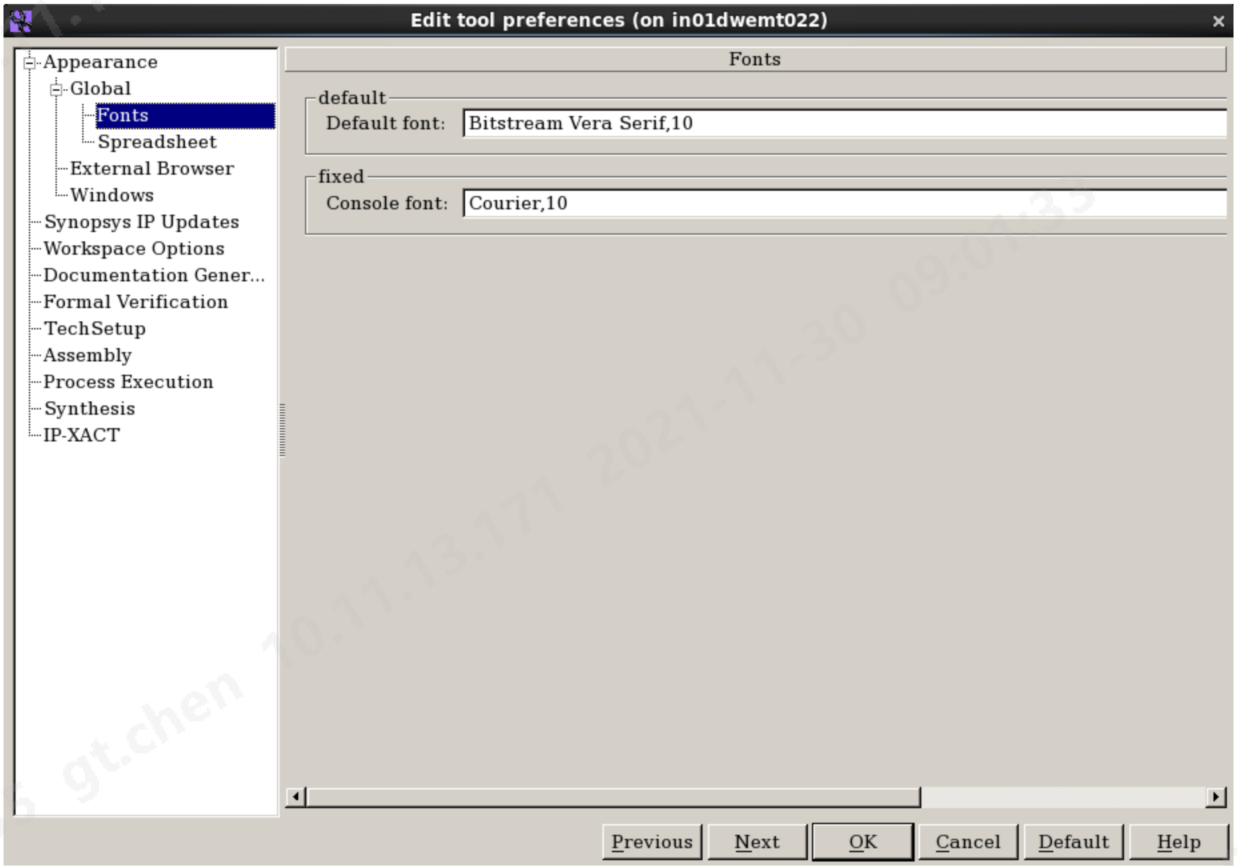
Before You Start

Check that you have the latest tool versions installed and your environment variables set up correctly (see "Configuring the Core Using coreConsultant" on page 36). 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.

2.6.7.1 Setting Synthesis Preferences

You can set synthesis-specific preferences by choosing the **Edit > Preferences > Synthesis** in coreConsultant a shown in Figure 2-27.

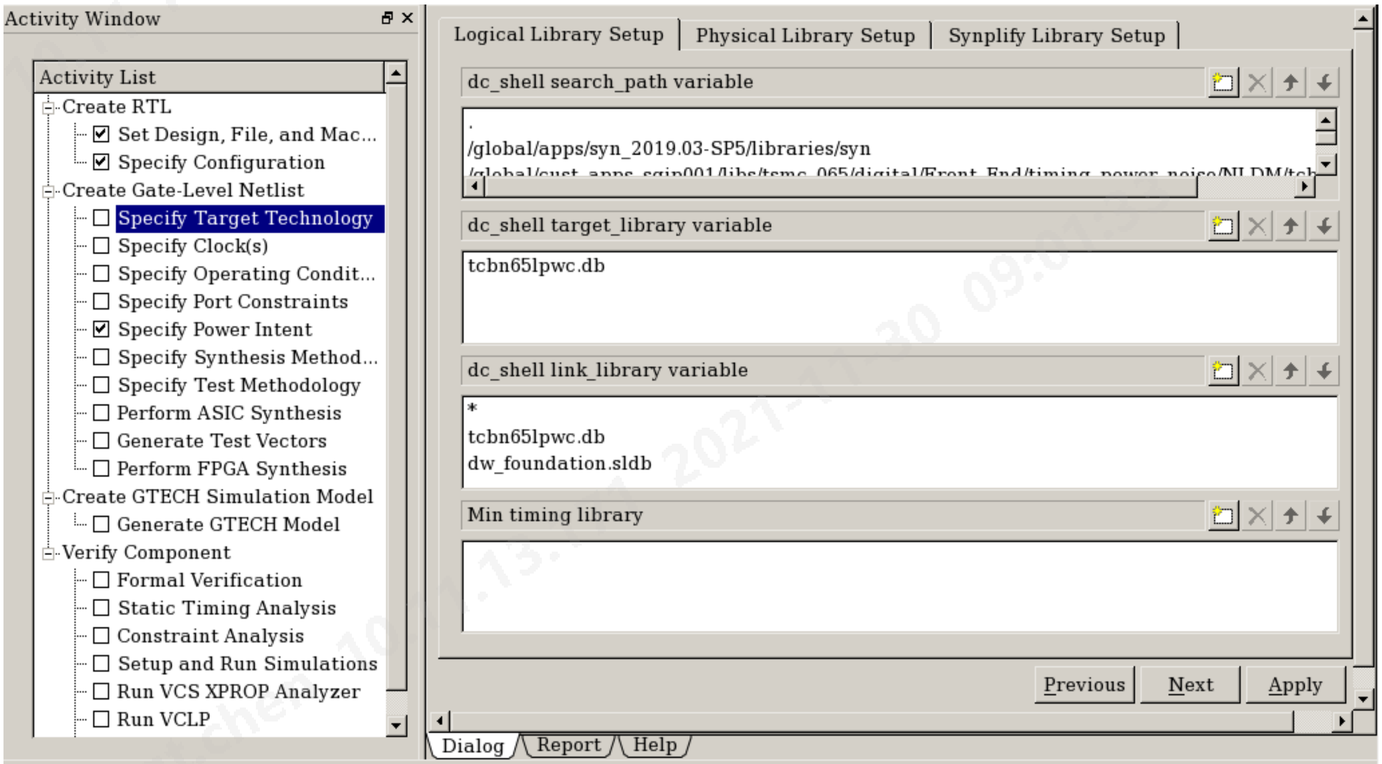
Figure 2-27 Setting Synthesis Preferences



2.6.7.2 Synthesizing Your Core

To synthesize your core, select the 'Create Gate-Level Netlist' activity as shown in Figure 2-28.

Figure 2-28 Create Gate-Level Netlist (Synthesis) Activity



2.6.7.3 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 64 so that you can synthesize the core in your own design flow.

The topics in this section are as follows:

- "Performing ASIC Synthesis" on page 51
- "Checking ASIC Synthesis Results" on page 55

2.6.7.3.1 Performing ASIC Synthesis

You must configure the core (see "Core Configuration" on page 40) before starting this task. To synthesize your core, complete the following steps in the **Create Gate-Level Netlist** activity (see Figure 2-29).

1. **Specify Target Technology.**

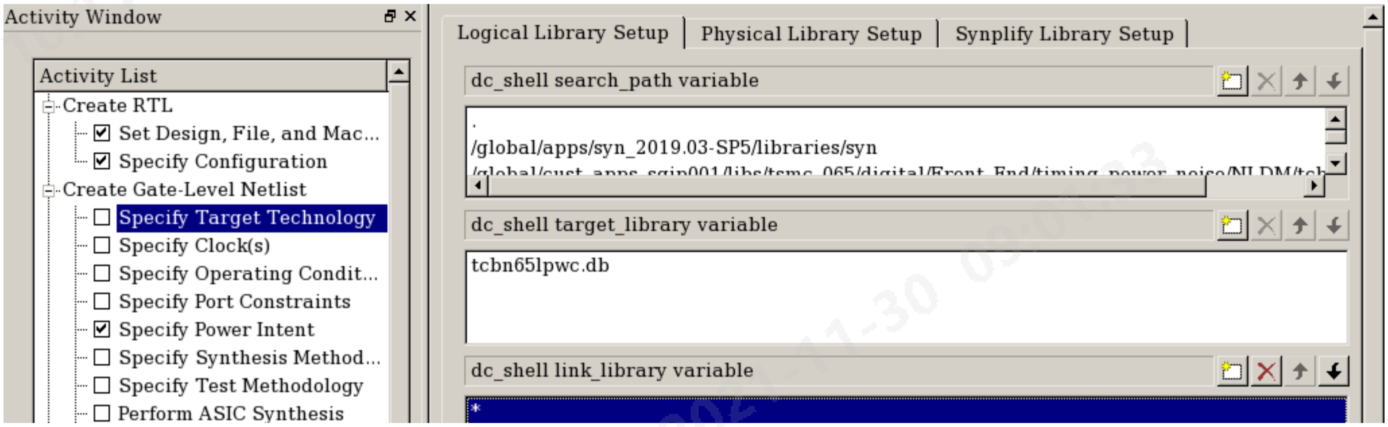
   A target library must be specified, otherwise, errors occur in coreConsultant. For ASIC synthesis, use a target technology of .18 microns or less. To add a target technology library, click the folder icon (A in Figure 2-29) and use the navigation tool (B in the figure).

   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 2-29 Specify Target Technology – Logical Library Setup

2. **Specify Clocks.**



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 2-30.

Figure 2-30 Specify Operating Conditions and Wireloads Window

Attributes	DW_axi
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.

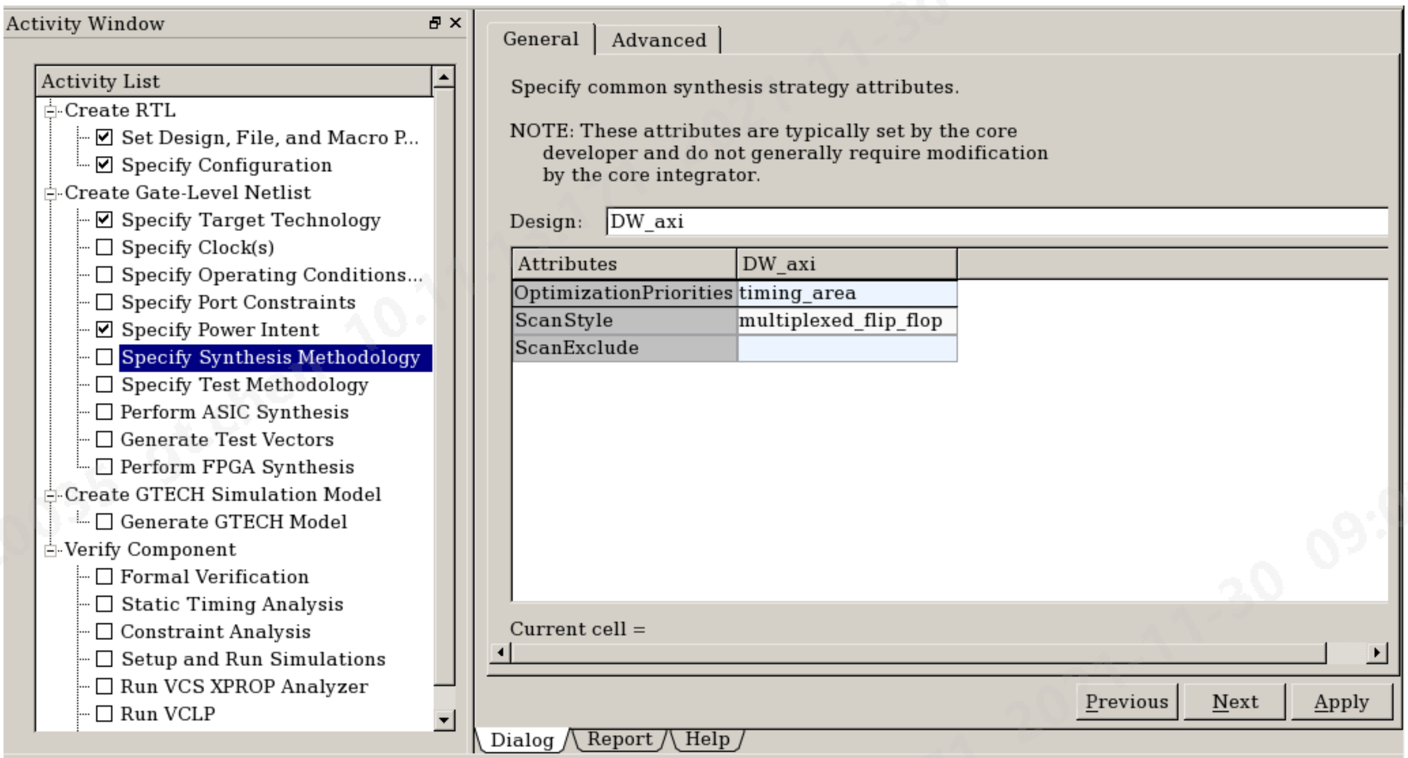
The default input and output delay constraints are specified as a percentage of the clk associated with the port.

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 2-31) 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 2-31 Specify Synthesis Methodology – Physical Synthesis Tab



6. **Specify Test Methodology.**

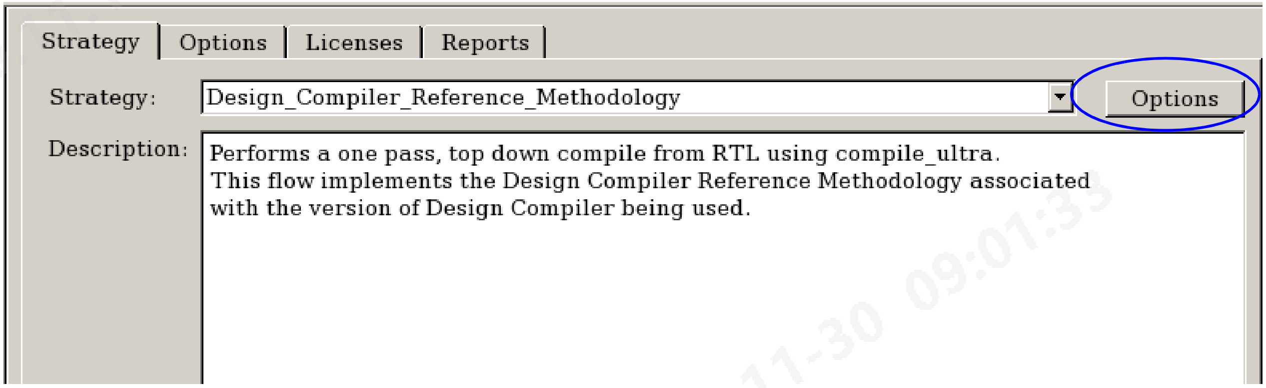
   These attributes are set by the core developer and can be optionally modified by you. For more details, refer to “Inserting Design For Test Using Design Compiler” on page 57.

7. **Perform ASIC Synthesis.**

   You can select the synthesis strategy on the Strategy tab shown in Figure 2-32 on page 54. The default flow is “Design_Compiler_Reference_Methodology”. A detailed explanation of each strategy is provided in the GUI.

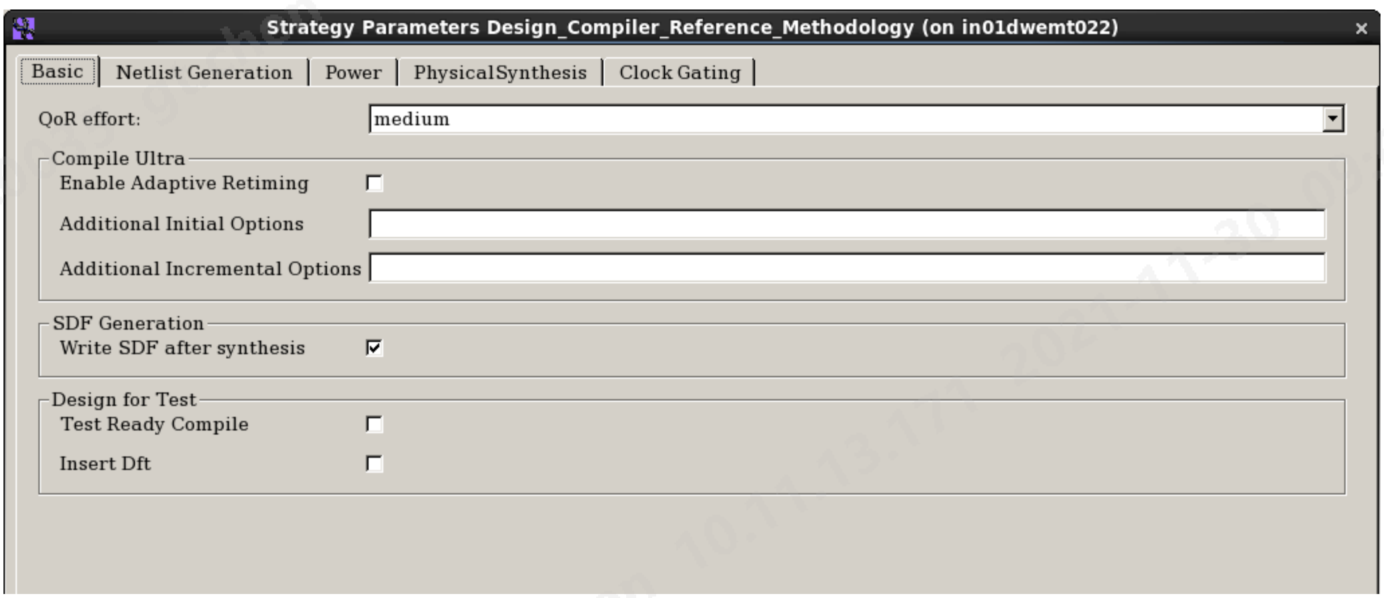
Hint One other strategy that is of interest is the DCTCL_one_pass_compile_ultra, which uses a simple TCL script based on the Synopsys Design Compiler Reference Methodology at https://solvnet.synopsys.com/retrieve/021023.html. This synthesis strategy is quicker and does not generate intermediate stages.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

Figure 2-32 Synthesize Activity -> Strategy Tab



Clicking on the **Options** button (highlighted in Figure 2-32) 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 2-33 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 57.

- **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.

2.6.7.3.2 Checking ASIC Synthesis Results

After you have run synthesis, coreConsultant generates several reports. You can access these by clicking the Report tab in Figure 2-28 on page 51.

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 57):

- 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 2-6 includes the other files that are generated after synthesis.

Table 2-6 ASIC Synthesis Output Files

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

1. BUTTON as circled in Figure 2-32 on page 54 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.
3. component indicates the name of the IP that you are using. For example, DW_axi.

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

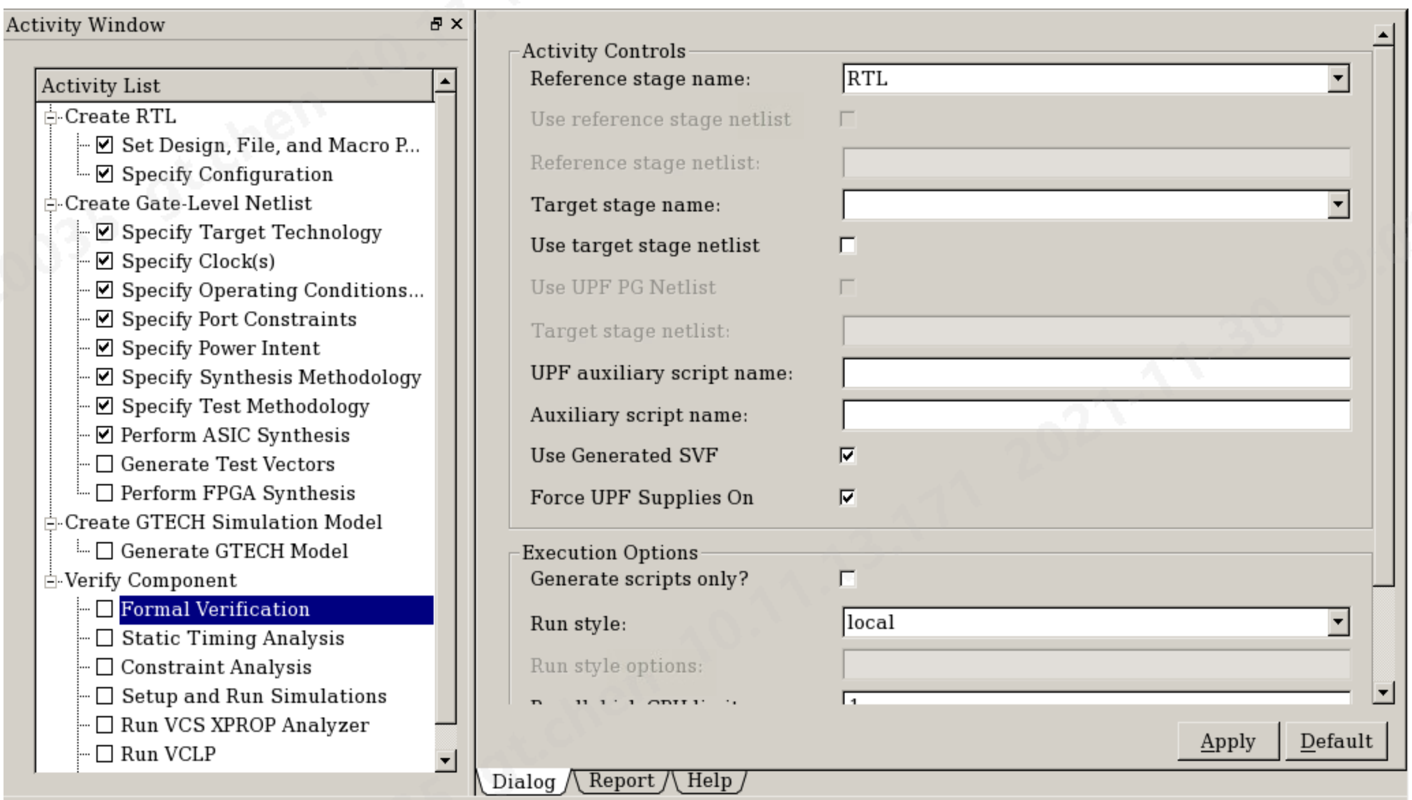
Note In the <workspace>/syn directory, run.scr, Makefile, and final are symbolic links to the current synthesis stage files. These links are also used and set to different locations during ATPG and Formal Verification.

2.6.8 Performing Formal Verification

You use the Formal Verification activity (in Figure 2-34) for RTL-to-gate comparisons in coreConsultant.

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 2-34 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.

2.6.9 Inserting Design For Test Using Design Compiler

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.
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 2-32 on page 54). You must select one of these two options:
   - "Test Ready Compile": Design" Compiler uses scan flip-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 flip-flops and scan chain insertion.

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

2.6.10 Running Automatic Test Pattern Generation using TetraMax

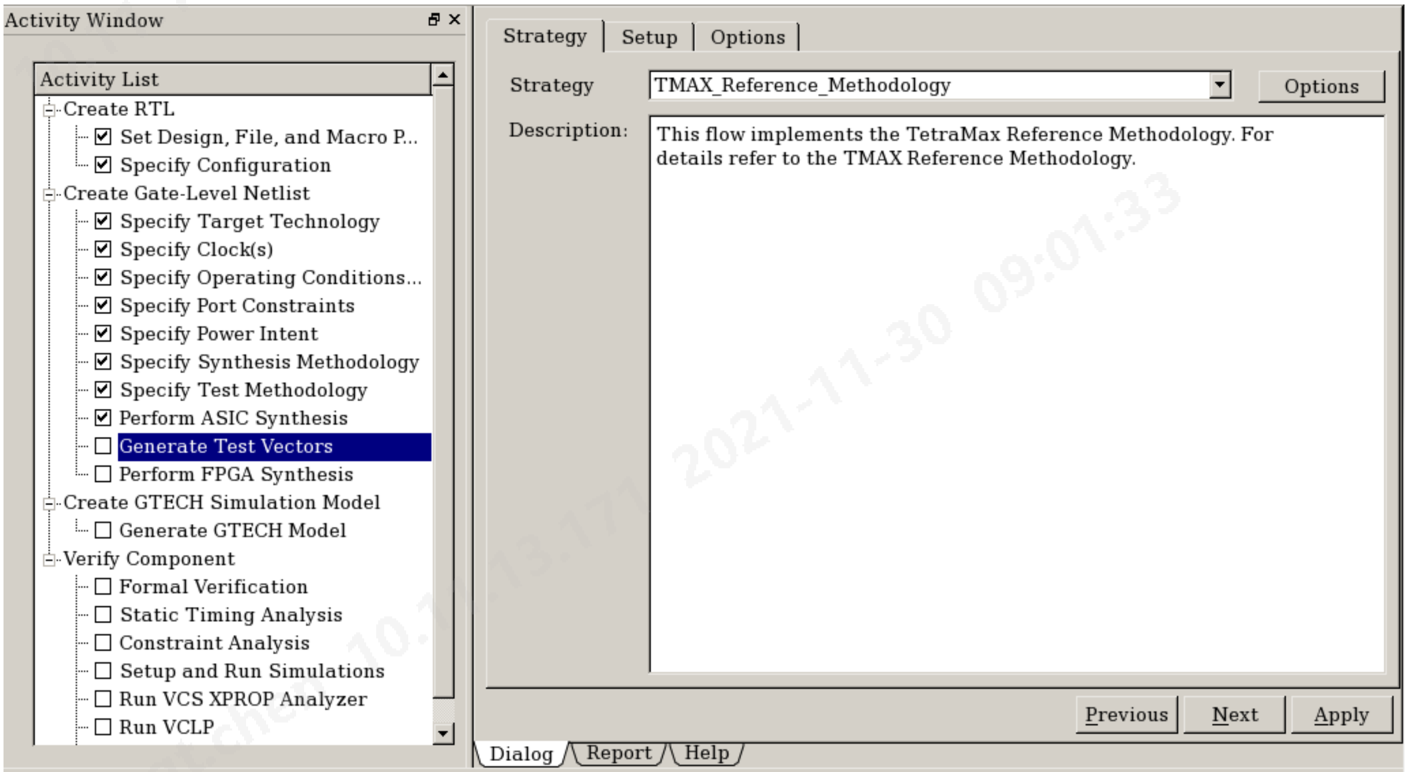
You perform automatic test pattern generation (ATPG) by using the Synopsys TetraMax tool.

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 57) before a setup can exist in the ATPG window.

To run ATPG, you must complete the Generate Test Vectors activity (Figure 2-35) in coreConsultant.

Figure 2-35 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 57), 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 to identify the output files that will be created in your workspace directory. For example, *atpg*.
- **Test Libraries**: Specify the library Verilog file names.

  These are the Verilog simulation models for the target libraries used during synthesis. They provide the description of the logical functionality of all the cells in the standard cell library. They are needed for the ATPG tool to comprehend the logic function implemented 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/component.tcl`
- `Makefile`

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

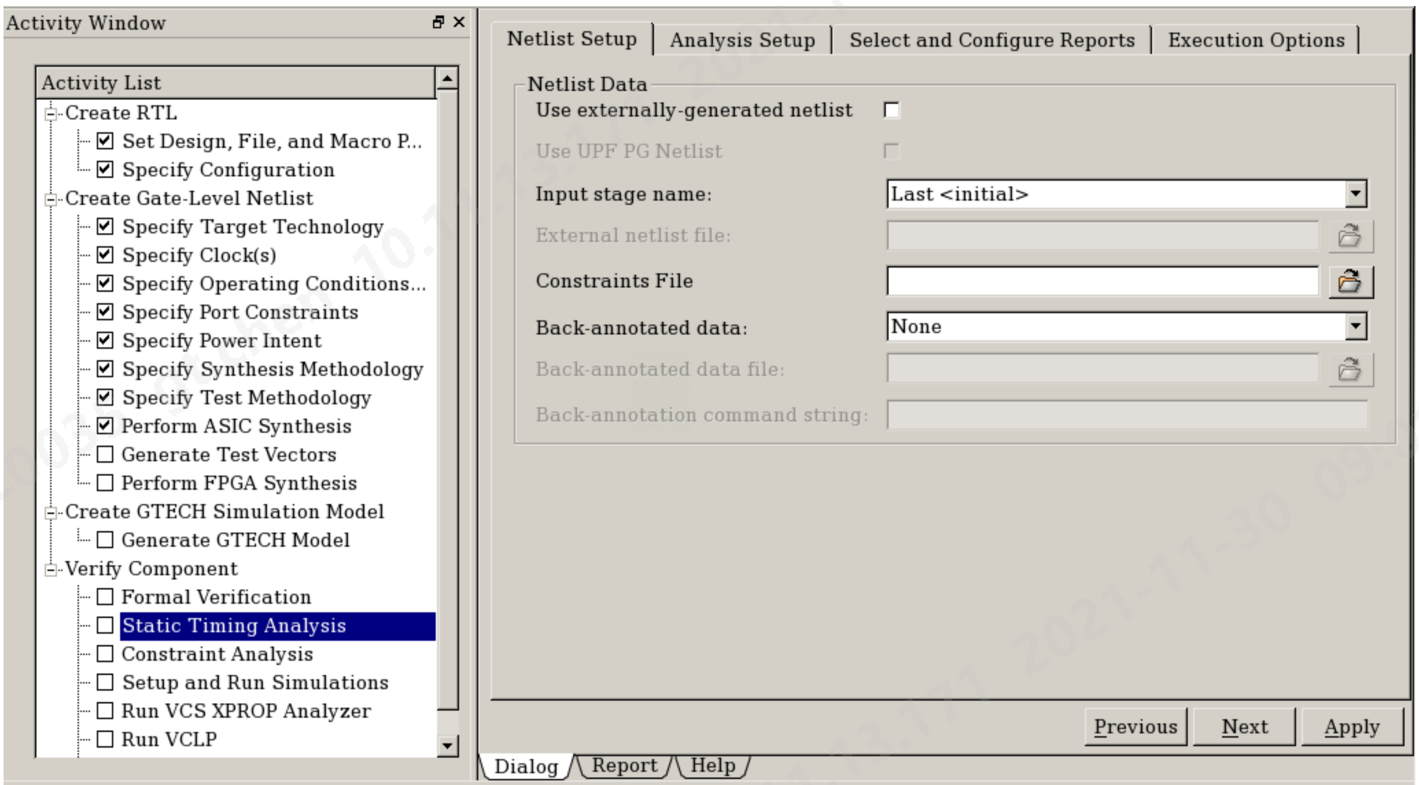
2.6.11 Running Static Timing Analysis using PrimeTime

You perform static timing analysis (STA) by using the Synopsys PrimeTime tool.

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 51 step or have an externally generated netlist available before starting this task. To perform STA, select the **Static Timing Analysis** activity.

Figure 2-36 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**.

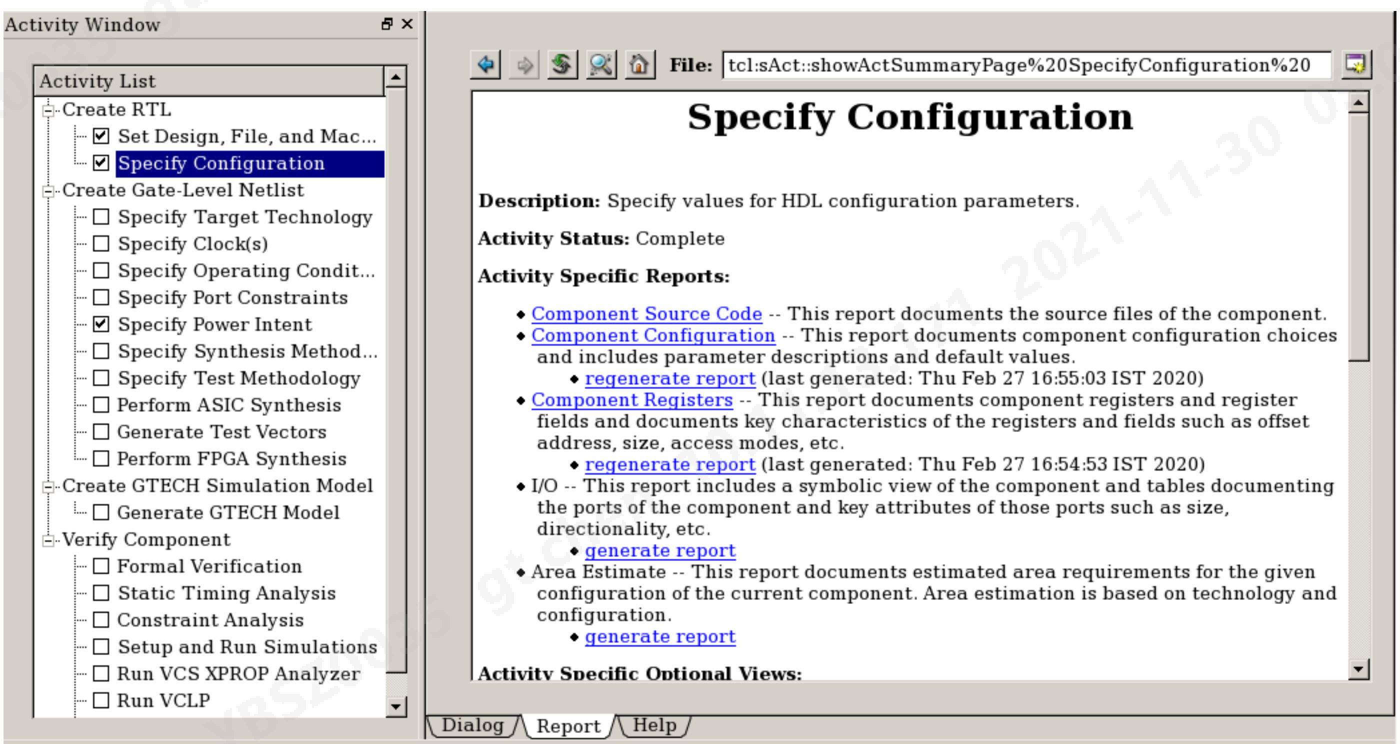
The STA results are saved in the <workspace>/syn/final/report/sta directory.

2.6.12 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 2-37, 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 2-37 Generating Activity-Specific Reports and Views



2.6.12.1 Activity-Specific Reports and Views

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

Table 2-7 describes the activity-specific optional reports.

Table 2-7 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 2-8 describes the activity-specific optional views.

Table 2-8 Activity-Specific Optional Views

Optional Views	File Name (in <workspace>/export)	Description
IP-XACT Component	component.xml	Generates IP-XACT component corresponding to the current component.
Example Component Instantiation	component_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	component.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.

2.6.12.2 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.

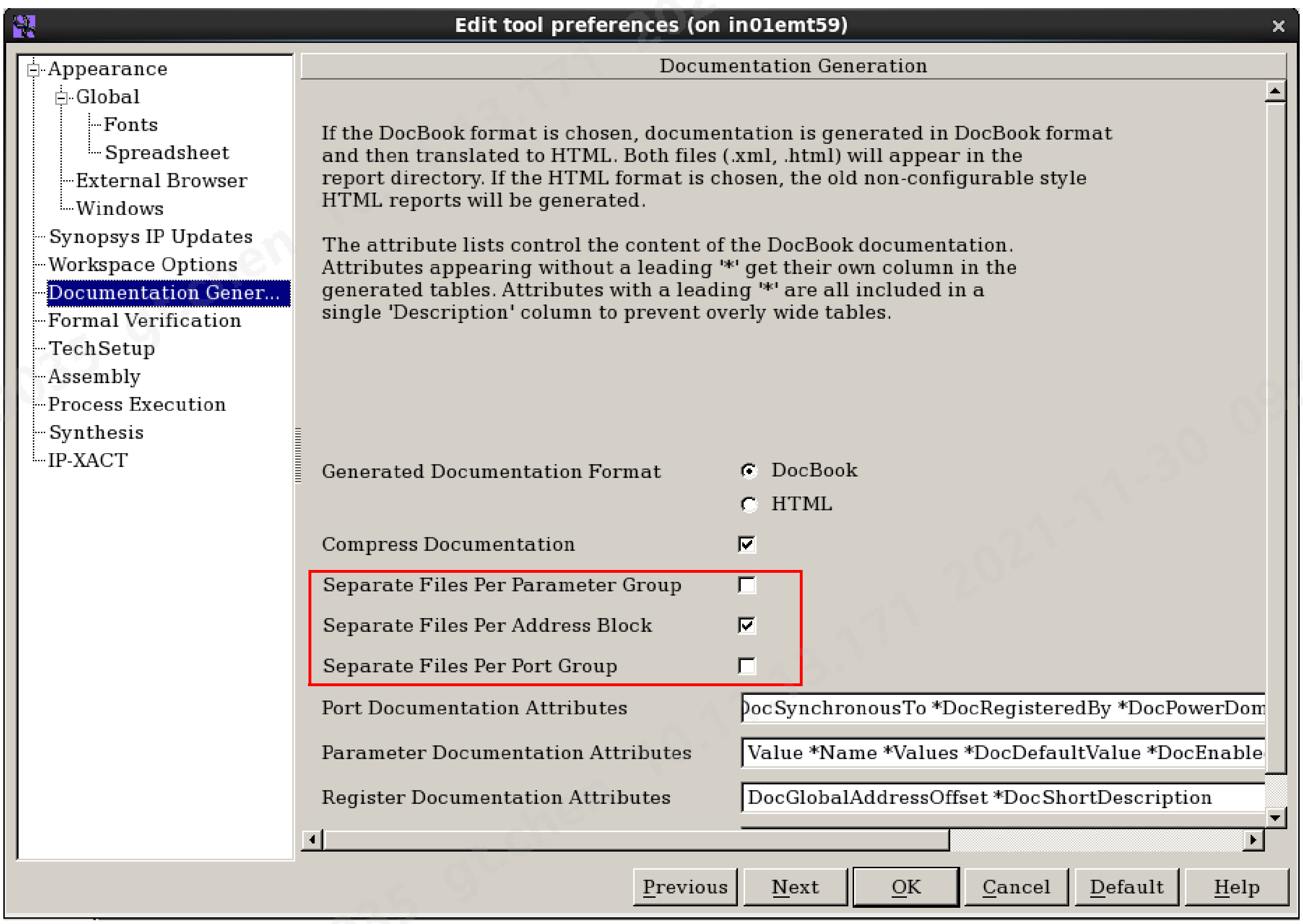
2.6.12.3 Setting Preferences for Report Generation

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 2-38.

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 2-38 Separate Files Options



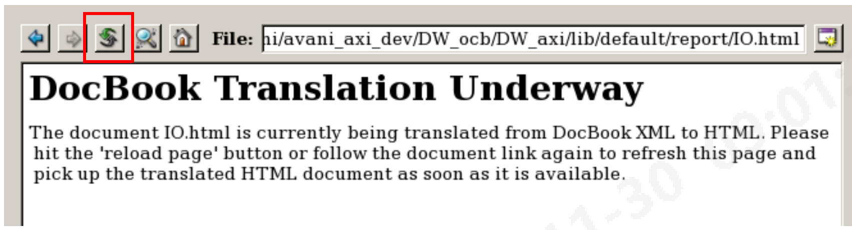
2.6.12.4 Generating Activity-Specific Reports

To generate the required report, click the respective **generate report**. For example, to generate component registers-related reports, click the one highlighted in Figure 2-37.

2.6.12.5 Accessing Reports

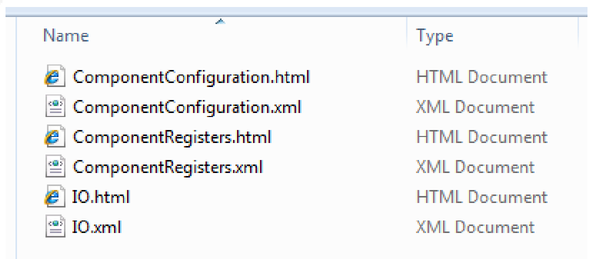
To view the report, click **Reload Page** after clicking **generate report** as shown in Figure 2-39.

Figure 2-39 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 2-40:

Figure 2-40 Contents of the report Directory



2.6.12.6 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.

2.6.13 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.

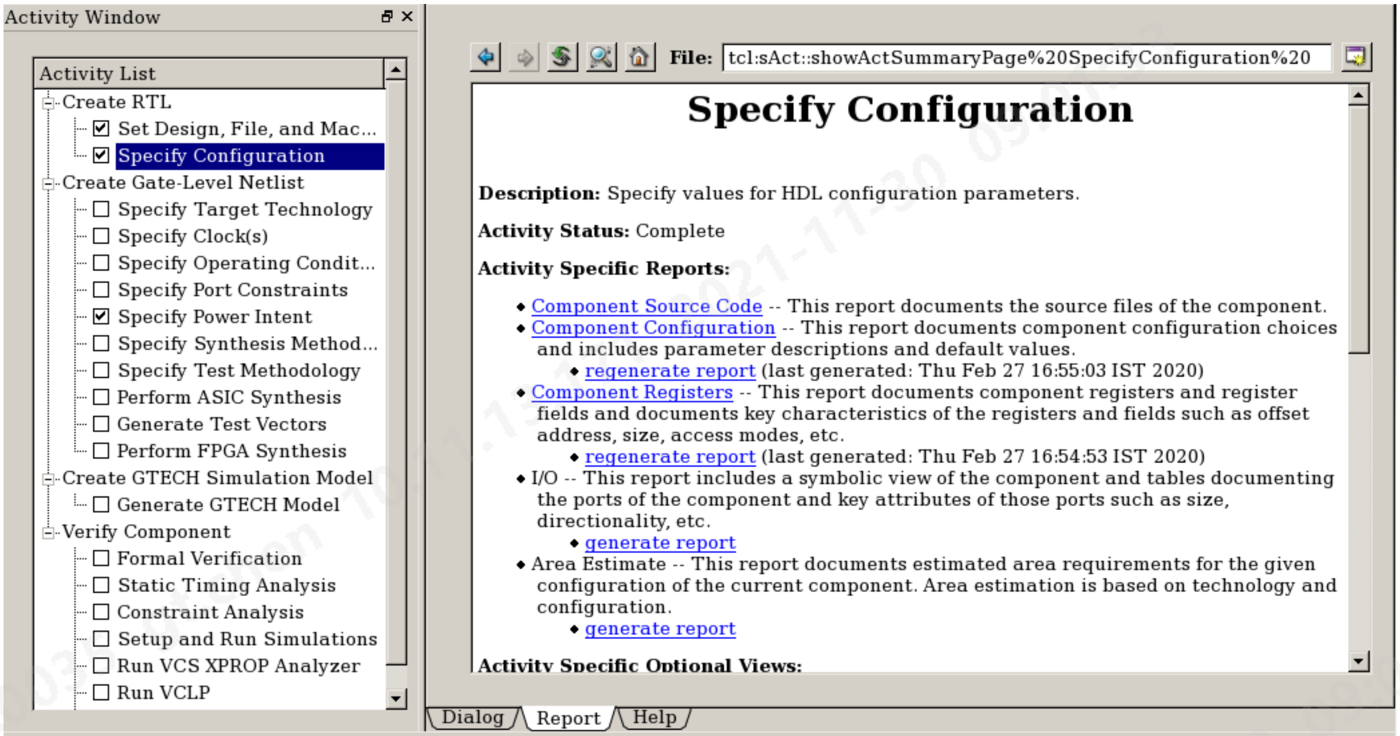
The topics in this section are as follows:

- "Generating an Example Core Instantiation"
- "Synthesizing to a Device Outside of coreConsultant" on page 64
- "Exporting Views and Reports" on page 67

2.6.13.1 Generating an Example Core Instantiation

After you have configured your DesignWare Synthesizable IPs, you can generate an example Component Instantiation.

Figure 2-41 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 2-37.

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

2.6.13.2 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 49. 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.

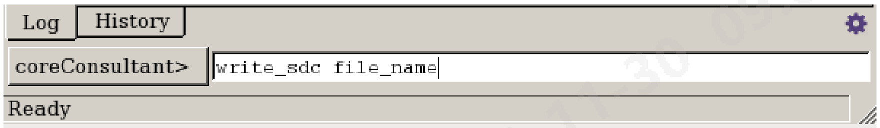
Method 1: Using write_sdc Command

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 2-42 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**, 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 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 ASIC Synthesis" on page 51.
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/component.sdc

Table 2-9 ASIC Synthesis Flow Files in <workspace>/syn

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

2.6.13.3 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 56 or "Running Automatic Test Pattern Generation using TetraMax" on page 57. 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. The topics in this section are as follows:

- "Exporting Formality Scripts"
- "Exporting DFT Scripts"
- "Exporting ATPG Scripts"
- "Exporting Views and Reports" on page 67

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 56. 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 57.

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 57. 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 57)

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.

2.6.13.3.1 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 60. 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).

2.7 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 DesignWare Synthesizable IPs using coreAssembler.

This section discuses the following topics:

- "Configuring DesignWare Synthesizable IPs within a Subsystem" on page 67
- "Creating Gate-Level Netlists within coreAssembler" on page 69
- "Verifying the DesignWare Synthesizable IPs within coreAssembler" on page 69

Note While adding components into the coreAssembler GUI, you must add the DesignWare Synthesizable IPs 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.

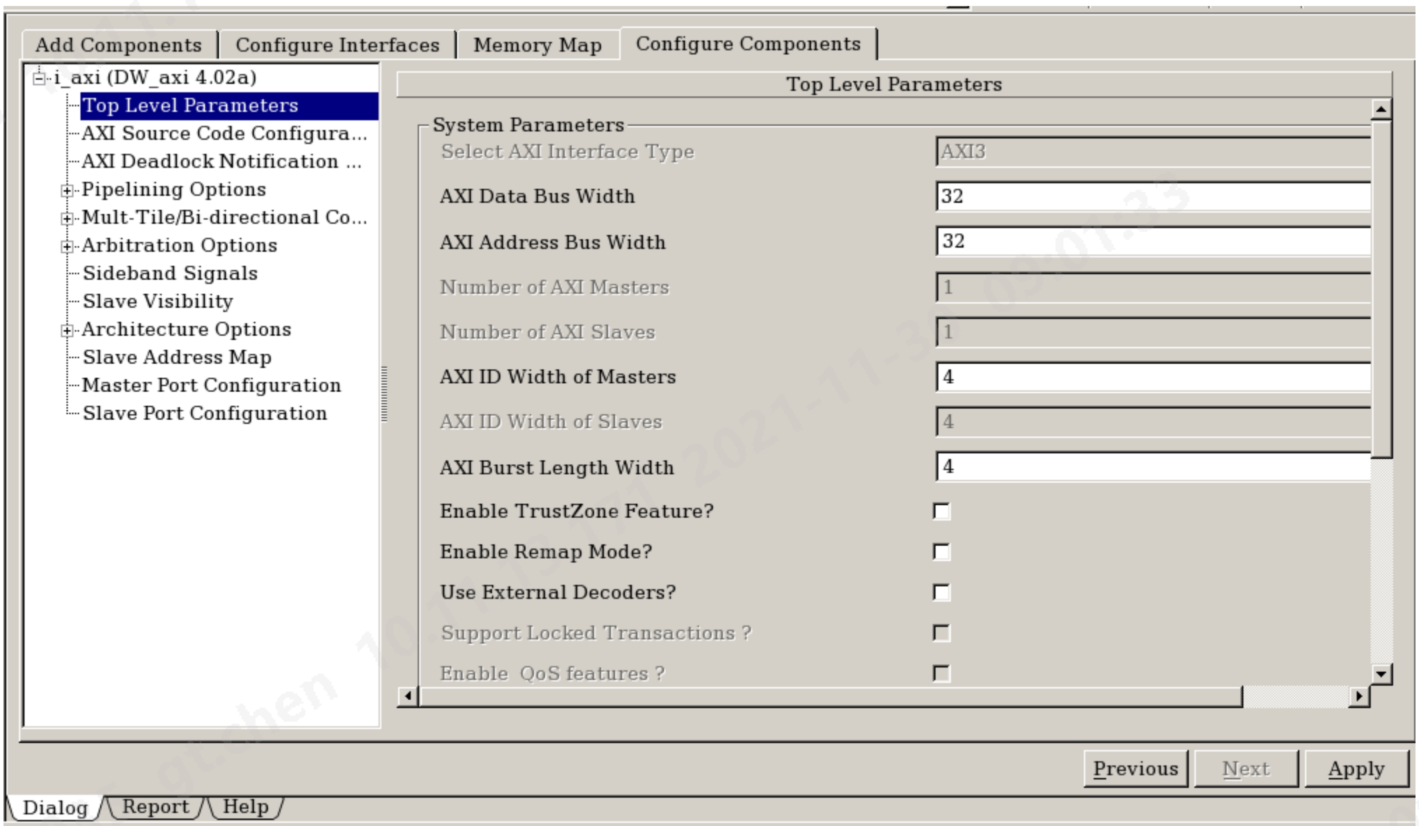
2.7.1 Configuring DesignWare Synthesizable IPs within a Subsystem

Figure 2-43 represents the coreAssembler GUI to configure DesignWare Synthesizable IPs in a subsystem.

The "Parameter Descriptions" chapter in the databook of the respective component describes the DesignWare Synthesizable IPs hardware configuration parameters that you configure using the coreAssembler GUI. Corresponding databooks for the other components in a subsystem contain "Parameters" chapters that describe their respective configuration parameters.

The "Creating the RTL View of a Subsystem" chapter in the *coreAssembler User Guide* discusses how to configure subsystem components and automatically connect them using the coreAssembler GUI.

Figure 2-43 Configuring DesignWare Synthesizable IPs in a Subsystem



1. **Specify your configuration for DesignWare Synthesizable IPs.**

   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. **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.

2.7.2 Creating Gate-Level Netlists within coreAssembler

The "Creating the Gate-Level Netlist for a Subsystem" chapter in the *coreAssembler User Guide* discusses how to create a translation of the RTL view into a technology-specific netlist for a subsystem.

2.7.3 Verifying the DesignWare Synthesizable IPs within coreAssembler

The "Verification" chapter in the databook of the respective component provides an overview of the testbench verification using the coreAssembler GUI.

2.7.3.1 coreAssembler Subsystem Simulation Guidelines

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

2.7.4 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.

2.8 Database Files

The following subsections describe some key files created in coreConsultant and coreAssembler activities.

2.8.1 Design/HDL Files

The following sections describe the design and HDL files that are produced by coreConsultant and coreAssembler when configuring and verifying a DesignWare Synthesizable Component. The following files are created in different directories by coreConsultant and coreAssembler:

- coreConsultant – *workspace*/ directory
- coreAssembler – *workspace*/components/i_*component*/ directory

2.8.1.1 RTL-Level Files

The following table describes the RTL files that are generated by the Create RTL activity. They are encrypted except where otherwise noted. Any Synopsys synthesis tool or simulator can read encrypted RTL files.

Table 2-10 RTL-Level Files

Files	Encrypted?	Purpose
./src/*component*_cc_constants.v	No	Includes definitions and values of all configuration parameters that you have specified for the component.

./src/*component*.v	No	Top-level HDL file. Include the DesignWare libraries by using the following options in your simulator invocation:
		`+libext+.v+.V`
		`-y ${SYNOPSYS}/packages/gtech/src_ver`
		`-y ${SYNOPSYS}/dw/sim_ver`

./src/*component_submodule*.v	Yes	Sub-modules of component
./src/*component*_constants.v	No	Includes the constants used internally in the design.
./src/*component*_undef.v		Includes an undef for each of the definitions found in the *component*_cc_constants.v file; compiled in after the last file listed in ./src/*components*.lst when compiling multiple instances of the same IP.
./src/*component*.lst	No	Lists the order in which the RTL files should be read into tools, such as simulators or dc_shell. For example, use the following option to read the design into VCS: `vcs +v2k -f component.lst`

2.8.1.2 Simulation Model Files

The following table includes files generated for the component during the Generate GTECH Simulation activity. These files are needed when you are using a non-Synopsys simulator (when you can not use the encrypted RTL).

Table 2-11 Simulation Model Files

Files	Encrypted?	Purpose

./gtech/final/db/*component*.v	No	Simulation model of the component for use with non-Synopsys simulators. A technology-independent, gate-level netlist; VHDL and Verilog versions are generated. Include the DesignWare libraries by using the following options in your simulator invocation:
		`+libext+.v+.V`
		`-y ${SYNOPSYS}/packages/gtech/src_ver`
		`-y ${SYNOPSYS}/dw/sim_ver`

2.8.2 Synthesis Files

The following table includes files generated after the Create Gate-Level Netlist activity is performed on a component.

Table 2-12 Synthesis Files

Files	Encrypted?	Purpose
./syn/auxScripts	No	Auxiliary files for synthesis.
./syn/final/db/*component*.db	Binary format	Synopsys .db files (gate level) that can be read into dc_shell for further synthesis, if desired.
./syn/final/db/*component*.v	No	Gate-level netlist that is mapped to technology libraries that you specify.
./syn/constrain/script/*.*	No	Constraint files for the components.
./syn/final/report/*.*	No	Synthesis result files.

2.8.3 Verification Reference Files

Files described in the following table include information pertaining to the operation of the component so that you can verify installation and configuration of the component has been successful. These files are not for re-use during system-level verification.

Table 2-13 Verification Reference Files

Files	Encrypted?	Purpose
./sim/runtest	No	Perl script that runs the Setup and Run Simulations activity from the command line.
./sim/runtest.log	No	The overall result of simulation, including pass/fail results.
./sim/test_*testname*/test.result	No	Pass/fail of individual test.
./sim/test_*testname*/test.log	No	Log file for individual test.

A
Additional coreConsultant Information

The topics in this section are as follows:

A.1 Troubleshooting

This section describes some common problems you may encounter while performing activities in the 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 while installing the core. You must re-install the core using the Project ID number. For more information, see 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 or SVT VIP is not installed in the DesignWare home directory.

A.1.3 Create Gate-Level Netlist Activity

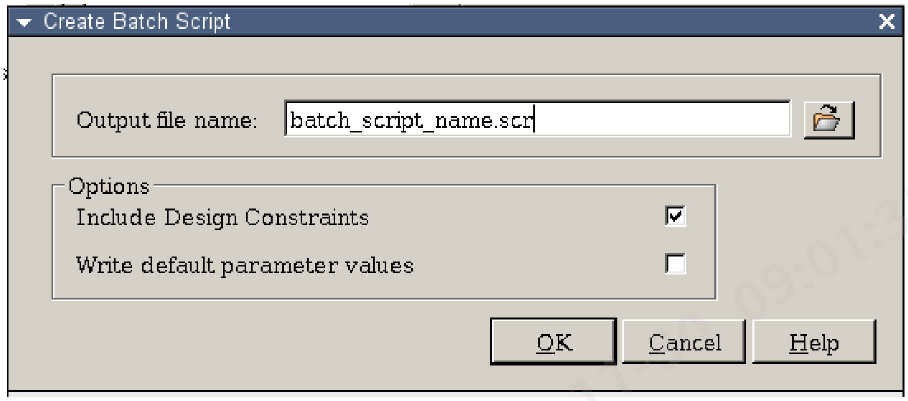
Here are some common problems that can occur after you perform synthesis (ASIC) for your configured core:

- 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
  ```

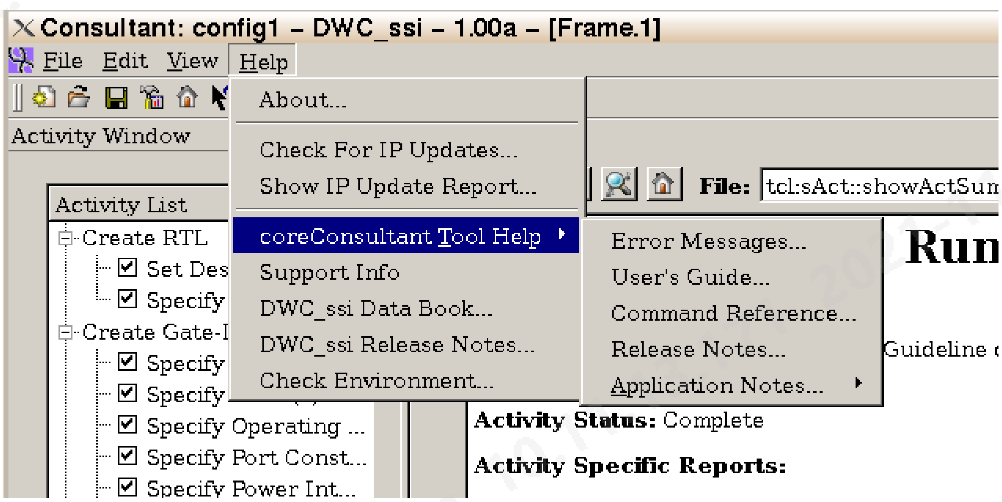
- **"What's This?" Quick Help**

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

Figure A-2 coreConsultant Help Menu Pull-down with 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 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).