DesignWare DW_ahb Tutorial

Release 2004.06

June 2, 2004

Contents

Preface

About This Manual

This tutorial provides information about the DesignWare Advanced High-performance Bus (DesignWare AMBA On-chip Bus). The information in this document includes an introduction, a tutorial using the DesignWare AMBA On-chip Bus component and coreConsultant, and a glossary.

Related Documents

To see a complete listing of documentation related to the DesignWare AMBA On-chip Bus within the DesignWare AMBA On-chip Bus platform, refer to the *Guide to DesignWare AMBA OCB Documentation*.

Manual Overview

This manual contains the following chapters and appendixes:

Chapter 1 "Overview of the DesignWare AMBA On-chip Bus"	Provides a DesignWare AMBA OCB System Overview, a tutorial block diagram, and information on setting up your environment.
Chapter 2 "Configuring a Synthesizable Component"	Gives a step-by-step tutorial to walk through the process of using the DesignWare AMBA On-chip Bus. This tutorial includes configuration, simulation, and basic synthesis flows.
Appendix A "Glossary"	Provides a glossary of general DesignWare AMBA terms.

Typographical and Symbol Conventions

The following conventions are used throughout this document:

Table 1: Documentation Conventions

Convention	Description and Example
%	Represents the UNIX prompt.
Bold	User input (text entered by the user). `% cd $LMC_HOME/hdl`
`Monospace`	System-generated text (prompts, messages, files, reports). `No Mismatches: 66 Vectors processed: 66 Possible"`

Table 1: Documentation Conventions (Continued)

Convention	Description and Example
Italic or `Italic`	Variables for which you supply a specific value. As a command line example: `% setenv LMC_HOME prod_dir` In body text: In the previous example, *prod_dir* is the directory where your product must be installed.
| (Vertical rule)	Choice among alternatives, as in the following syntax example: `-effort_level low	medium	high`
[ ] (Square brackets)	Enclose optional parameters: `pin1 [pin2 ... pinN]` In this example, you must enter at least one pin name (*pin1*), but others are optional ([*pin2* … *pinN*]).
TopMenu > SubMenu	Pulldown menu paths, such as: **File > Save As …**

Getting Help

If you have a question about using Synopsys products, please consult product documentation that is installed on your network or located at the root level of your Synopsys product CD-ROM (if available). You can also access documentation for DesignWare products on the Web:

- Product documentation for many DesignWare products:

  http://www.synopsys.com/products/designware/docs

- Datasheets for individual DesignWare IP components:

  http://www.synopsys.com/products/designware/ipdir

You can also contact a Synopsys Support Center online or by phone:

- http://www.synopsys.com/support/support.html
- United States:
  Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific Time, Mon—Fri.
- Canada:
  Call 1-650-584-4200 from 7 AM to 5:30 PM Pacific Time, Mon—Fri.
- All other countries:
  Find other local support center telephone numbers at the following URL:

  http://www.synopsys.com/support/support_ctr

Additional Information

General information about Synopsys and its products is available at this URL:

http://www.synopsys.com

Comments?

To report errors or make suggestions, please send e-mail to:

doc@synopsys.com

To report an error that occurs on a specific page, select the entire page (including headers and footers), and copy to the buffer. Then paste the buffer to the body of your e-mail message. This will provide us with information to identify the source of the problem.

1 Overview of the DesignWare AMBA On-chip Bus

The DesignWare AMBA On-chip Bus provides synthesizable components and verification models in a technology-independent bus system that you can configure with an easy-to-use tool interface. This chapter provides a brief overview of what is included in a DesignWare AMBA On-chip Bus release. The topics include the following:

- "Synthesizable Components"
- "Building Block (DWF) Library" on page 10
- "Verification Models" on page 10

Synthesizable Components

The DesignWare AMBA On-chip Bus Family supports configuration, synthesis and verification for the following synthesizable components:

DW_ahb	AHB bus decoder, arbiter, default slave, and muxes
DW_ahb_dmac	AHB master/slave: AHB Direct Memory Access controller
DW_ahb_h2h	AHB master/slave: AHB to AHB bridge
DW_ahb_eh2h	AHB master/slave: AHB to AHB bridge
DW_ahb_icm	AHB Multi-layer Interconnection Matrix
DW_ahb_ictl	AHB slave: interrupt controller
DW_apb	APB bridge and interconnect for APB peripherals to AHB bus
DW_apb_gpio	APB slave: General Purpose I/O interface
DW_apb_i2c	APB slave: I2C bus (two-wire serial interface)
DW_apb_ictl	APB slave: interrupt controller
DW_apb_rap	APB slave: remap and pause control, ID register, and a reset status register
DW_apb_rtc	APB slave: Real Time Clock
DW_apb_ssi	APB slave: Synchronous Serial Interface
DW_apb_timers	APB slave: 8 programmable timers

DW_apb_uart	APB slave: Universal Asynchronous Receiver/Transmitter
DW_apb_wdt	APB slave: Watch Dog Timer

Building Block (DWF) Library

Many of the synthesizable components that are included in the DesignWare AMBA On-chip Bus release contain smaller synthesizable building blocks. These blocks are delivered in a DWF Building Block library and are needed when you synthesize the DesignWare AMBA On-chip Bus synthesizable components. See the “Installation and Setup” chapter in the *DesignWare DW_ahb Release Notes* for information on installing the DWF Building Block library.

Verification Models

The DesignWare AMBA On-chip Bus Family of components also includes verification models for both the AHB and APB bus environments. For more information, see the *Guide to DesignWare AMBA Verification IP Documentation* and *Guide to DesignWare VMT Documentation.*

ACT Certification

In this release of DesignWare AMBA On-Chip-Bus, the DW_ahb_ictl and DW_ahb_dmac components include AMBA Compliance Tool (ACT) certification, which allows you to run ACT certification on the component provided you have the required license (DesignWare-ACT-VIP) and the data width of the AHB bus is 32, 64, or 128 bits. For more information about DesignWare ACT certification, refer to the DesignWare AMBA Compliance Tool data sheet at:

www.synopsys.com/products/designware/docs/ds/v/ahb_act_monitor_vmt.pdf

If you are interested in purchasing the DesignWare AMBA Compliance Tool verification IP product, please contact your local Synopsys Sales Office.

2 Configuring a Synthesizable Component

Tutorial Overview

This tutorial is designed to help you learn about the DesignWare AMBA On-chip Bus environment. You will learn to use the configuration tool coreConsultant to configure, synthesize, and simulate any DW AMBA synthesizable component, using the DW_ahb as an example. The following topics are contained in this tutorial:

Design Content

This tutorial uses the following synthesizable component:

- DW_ahb – AMBA-compliant configurable AHB bus with arbitration.

Tutorial Block Diagram

Figure 1 shows the DW_ahb component used in the AMBA On-chip Bus Tutorial.

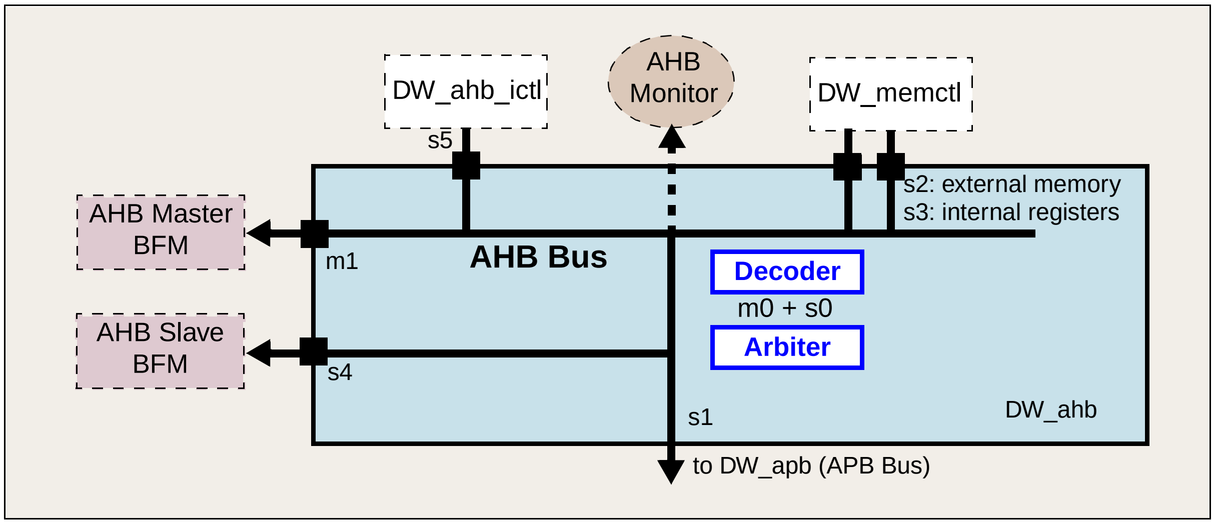


Figure 1: AMBA On-chip Bus Tutorial Diagram

In this tutorial, the DW_ahb is configured for use in a simple subsystem, which has one master and four slave components as follows:

- Master Component:
    - AHB Master BFM. This is the verification model and represents an external master, such as a CPU. This connects to DW_ahb master port 1.
- Slave Components:
    - APB bridge—DW_apb—connects to Slave port 1.
    - Memory Controller—DW_memctl—has two AHB slave ports: one for registers and one for external memory. It connects to two DW_ahb slave ports 2 and 3.
    - AHB Slave BFM—Verification model—connects to port 4 (an example of an external slave).
    - Interrupt controller on AHB —DW_ahb_ictl—connects to slave port 5.

In addition to the master and slave ports used in the previous description, the DW_ahb also has an internal master, the dummy master, and an internal slave, the arbiter. These take up master port 0 and slave port 0, respectively. For examples of this component configured in more complex subsystems, refer to the *DW_AMBA QuickStart SingleLayer Example Guide* and the *DW_AMBA QuickStart MultiLayer Example Guide*.

Configuration Information

In this tutorial, you configure the DW_ahb component as it is configured in the QuickStart_SingleLayer example subsystem (see *DW_AMBA QuickStart SingleLayer Example Guide*).

Tutorial Conventions

Each of the procedures included in this tutorial has the following parts:

- A description of what is going to happen, or what will be accomplished, in the procedure. This allows you to determine if you need to complete the procedure as some of the procedures are optional.
- The steps necessary to complete the procedure. These are numbered steps with detailed actions, file names, commands, or instructions. Some steps within a procedure may be optional or environment specific, so read the procedure fully before completing any steps.
- Some kind of verification that you have completed the procedure correctly, and possibly some helpful hints about commonly encountered problems. This includes completed dialog boxes, waveforms, transcripts, and error or warning messages.

It is useful to check-off completed steps, and put an X next to steps not followed. Also, fill in any information that is asked for in a step, as it will be useful for future review and may be needed in subsequent procedures.

Configuring the Component

Setting up Your AMBA On-chip Bus Tutorial Environment

This tutorial uses the same environment as all synthesizable components in the DesignWare AMBA On-chip Bus family. A discussion of the necessary environment variables is given in the section “ Setting Up Your Environment” of the *DesignWare DW_ahb Release Notes*.

Note

These variables and settings may already be set in your .cshrc or equivalent file. If you have questions, see your site administrator.

General coreConsultant Terminology

Activity	An item in the activity list such as “Specify Configuration.”
Previous/Next	Move to another configuration window for the same component.
Apply	The coreConsultant tool checks the configuration entries and saves them into the workspace. This is the last step of each major activity.

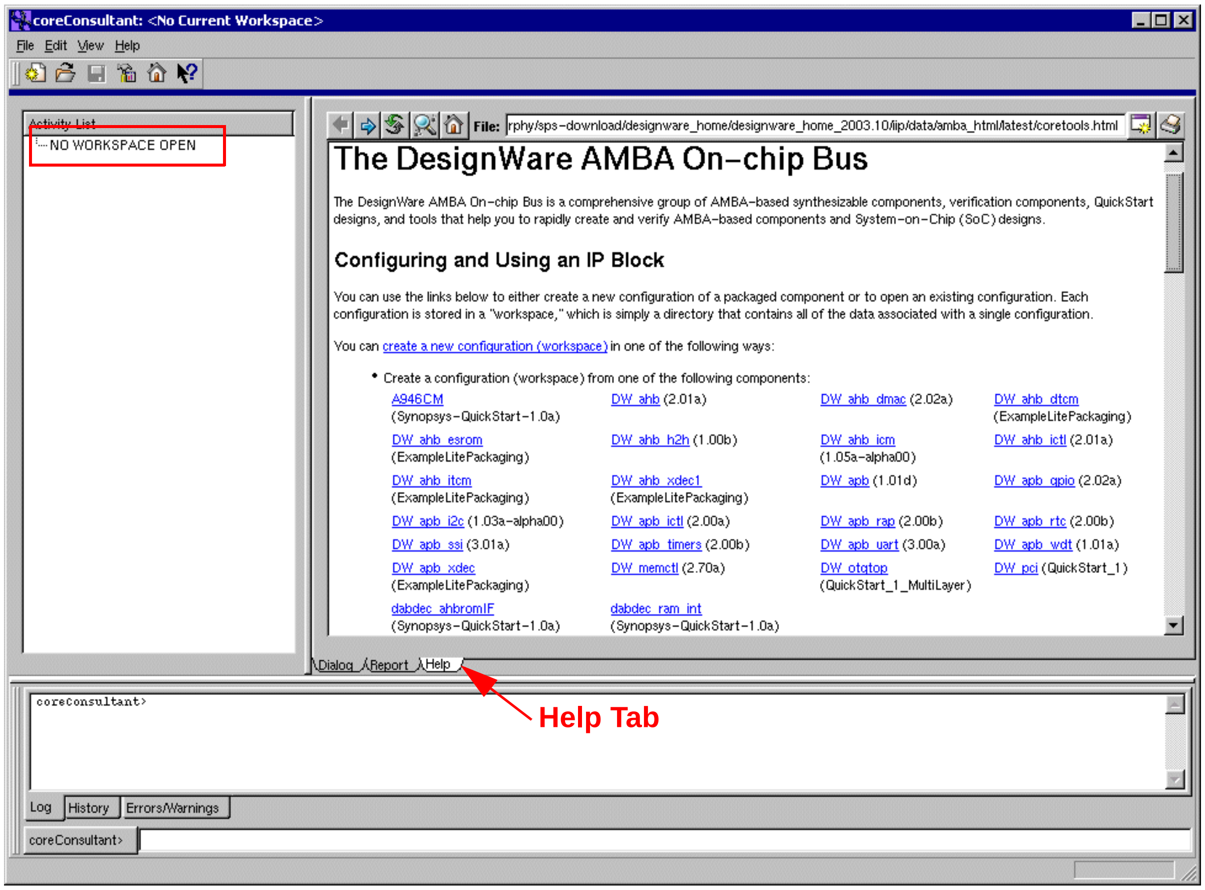
Invoking coreConsultant

In this procedure, you will use coreConsultant to set up a workspace that will eventually contain configuration files, RTL design, netlists, and test results. You also use coreConsultant to check paths to the tools (Design Compiler, VCS simulator, and so on) needed to complete the DesignWare AMBA On-chip Bus creation process.

1. **Navigate to the directory where you want to create your workspace (usually a project work area or a project area within your home account).**
2. **Invoke coreConsultant.**

```
% coreConsultant
```

The coreConsultant console appears showing “NO WORKSPACE OPEN.”



NO WORKSPACE OPEN

File: rphy/sps-download/designware_home/designware_home_2003.10/ip/data/amba_html/latest/coretools.html

The DesignWare AMBA On-chip Bus

The DesignWare AMBA On-chip Bus is a comprehensive group of AMBA-based synthesizable components, verification components, QuickStart designs, and tools that help you to rapidly create and verify AMBA-based components and System-on-Chip (SoC) designs.

Configuring and Using an IP Block

You can use the links below to either create a new configuration of a packaged component or to open an existing configuration. Each configuration is stored in a "workspace," which is simply a directory that contains all of the data associated with a single configuration.

You can create a new configuration (workspace) in one of the following ways:

- Create a configuration (workspace) from one of the following components:

A946CM (Synopsys-QuickStart-1.0a)	DW_ahb (2.01a)	DW_ahb_dmac (2.02a)	DW_ahb_dtcm (ExampleLitePackaging)
DW_ahb_esrom (ExampleLitePackaging)	DW_ahb_h2h (1.00b)	DW_ahb_icm (1.05a-alpha00)	DW_ahb_ictl (2.01a)
DW_ahb_itcm (ExampleLitePackaging)	DW_ahb_xdec1 (ExampleLitePackaging)	DW_apb (1.01d)	DW_apb_gpio (2.02a)
DW_apb_i2c (1.03a-alpha00)	DW_apb_ictl (2.00a)	DW_apb_rap (2.00b)	DW_apb_rtc (2.00b)
DW_apb_ssi (3.01a)	DW_apb_timers (2.00b)	DW_apb_uart (3.00a)	DW_apb_wdt (1.01a)
DW_apb_xdec (ExampleLitePackaging)	DW_memctl (2.70a)	DW_otgtop (QuickStart_1_MultiLayer)	DW_pci (QuickStart_1)
dabdec_ahbromIF (Synopsys-QuickStart-1.0a)	dabdec_ram_int (Synopsys-QuickStart-1.0a)		

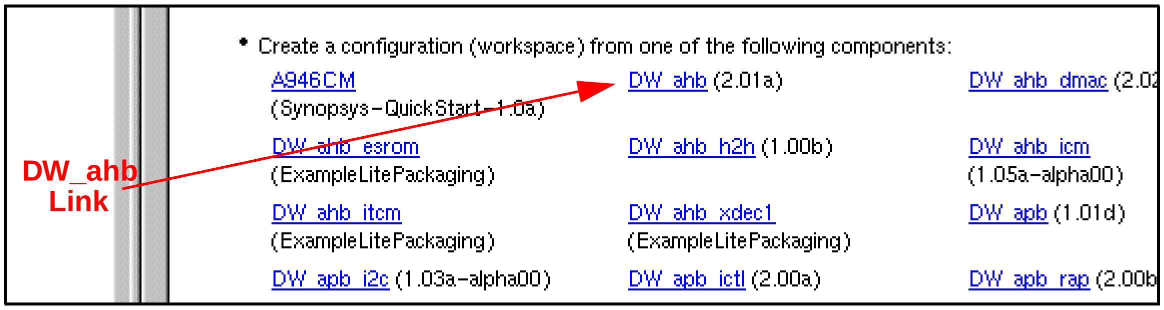
```
coreConsultant>
```

The “Help” tab for the right window is active, showing a “Getting Started” help option. This window helps you choose a component to open in your workspace.

Creating a New Configuration (workspace)

In this procedure, you will create a new workspace in your working directory that is where your copy of the DW_ahb component and custom configuration will be built. The Getting Started window lists all the components and versions from which to select.

1. Position your pointer on the "DW_ahb" link beneath "Configuring and Using and IP Block" in the Help window.



• Create a configuration (workspace) from one of the following components:

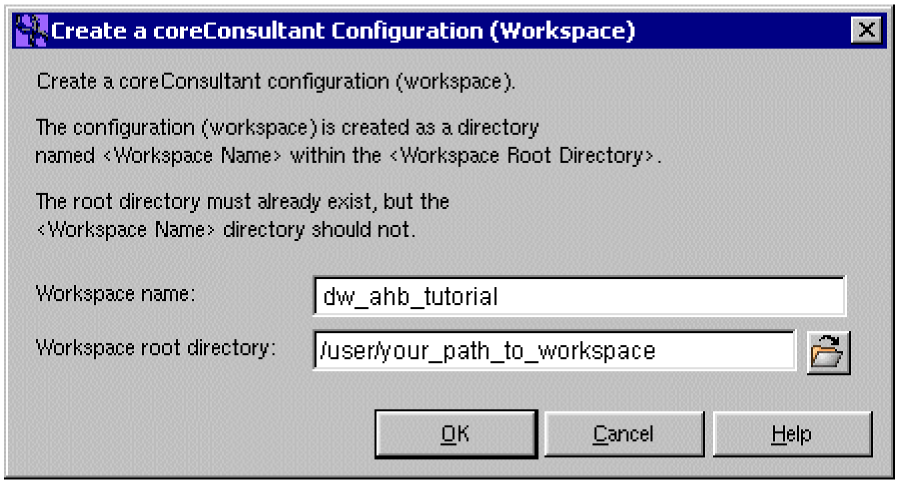
A946CM (Synopsys-QuickStart-1.0a)	DW_ahb (2.01a)	DW_ahb_dmac (2.0
DW_ahb_esrom (ExampleLitePackaging)	DW_ahb_h2h (1.00b)	DW_ahb_icm (1.05a-alpha00)
DW_ahb_itcm (ExampleLitePackaging)	DW_ahb_xdec1 (ExampleLitePackaging)	DW_apb (1.01d)
DW_apb_i2c (1.03a-alpha00)	DW_apb_ictl (2.00a)	DW_apb_rap (2.00b

Notice that in the bottom of the window, the command to create a workspace is shown, and will be executed when you click on this link:

```
cgi://AMBA::create_amba_workspace /path/to/your/DW_ahb DW_ahb
```

2. Click on the "DW_ahb" link text.

The "Create a coreConsultant Configuration (Workspace)" dialog box appears.



Create a coreConsultant configuration (workspace).

The configuration (workspace) is created as a directory named <Workspace Name> within the <Workspace Root Directory>.

The root directory must already exist, but the <Workspace Name> directory should not.

Workspace name: dw_ahb_tutorial

Workspace root directory: /user/your_path_to_workspace

Workspace Name: dw_ahb_tutorial.

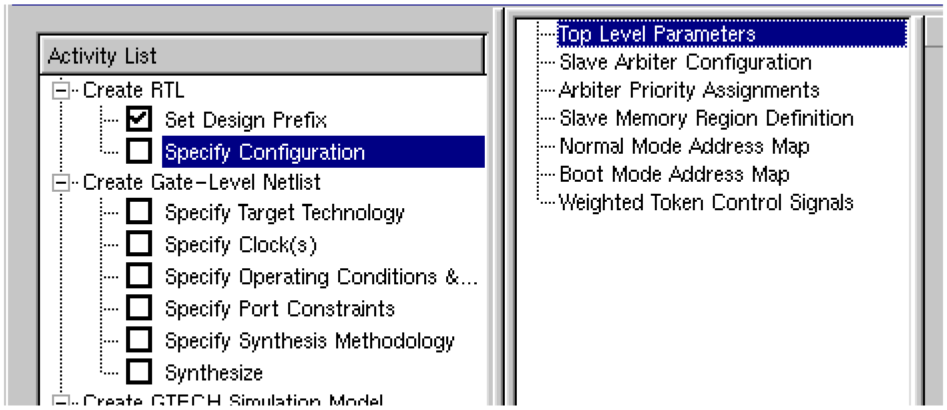
Workspace Root Directory: (a writable directory to contain the workspace)

The folder button can be used to browse your directory structure.

3. When the fields are completed, click OK.

Using the Activity List

The DW_ahb component workspace is created in the “Workspace Root Directory” path, and the Activity List appears with the “Set Design Prefix” activity checked (complete) and the “Specify Configuration” activity selected.



Activity List
- Create RTL
  - ☑ Set Design Prefix
  - ☐ Specify Configuration
- Create Gate–Level Netlist
  - ☐ Specify Target Technology
  - ☐ Specify Clock(s)
  - ☐ Specify Operating Conditions &...
  - ☐ Specify Port Constraints
  - ☐ Specify Synthesis Methodology
  - ☐ Synthesize
- Create GTECH Simulation Model

- Top Level Parameters
- Slave Arbiter Configuration
- Arbiter Priority Assignments
- Slave Memory Region Definition
- Normal Mode Address Map
- Boot Mode Address Map
- Weighted Token Control Signals

Figure 2: Consultant Activity List - Setup for DW_ahb

Notice also that the first Configuration view “Top Level Parameters” appears in the right window. Each component type (for example DW_apb_gpio) has a unique set of configuration parameter views and parameters pre-defined for that component. The next section takes you through the configuration process.

The following lists some of the features of the Activity List:

- **Checkbox** – Shows whether this activity has been completed.
- **Auto-complete** – By clicking on a future activity or checking a checkbox, the tool will attempt to auto-complete all uncompleted steps to that point, using defaults.
- **What’s This?** – Right-clicking on an Activity List item gives you the “What’s This” help selection box. Click on “What’s This” to view the help information.

Now lets begin the Specify Configuration activity.

Setting Top-level Parameters

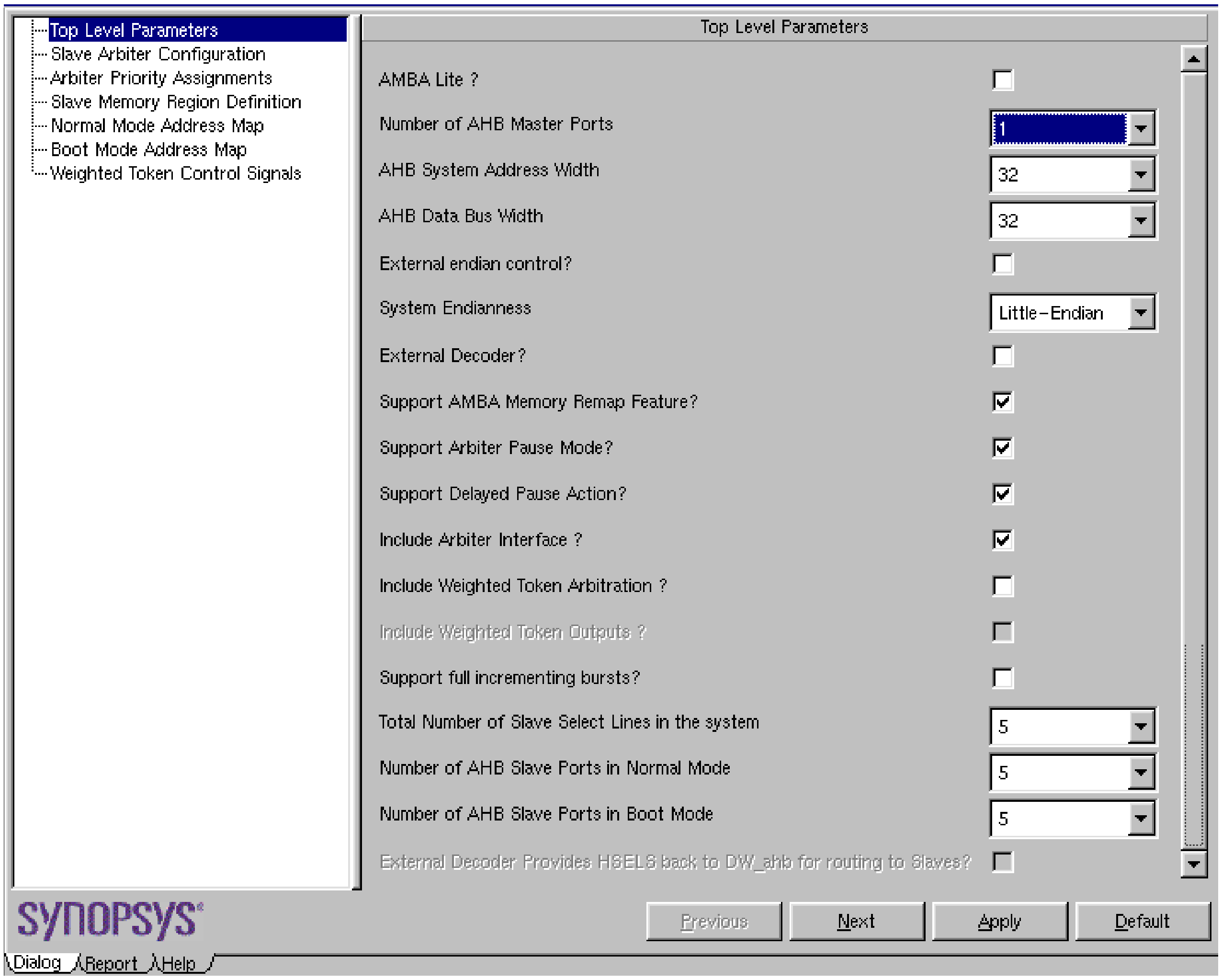
Table 2 shows the parameters for the “Top Level Parameters.”

Table 2: Top Level Parameters

Item	Value
AMBA Lite?	Unchecked
Number of AHB Master Ports	1
AHB System Address Width	32
AHB Data Bus Width	32
External endian control?	Unchecked
System Endianness (big or little endian)	Little-Endian
External Decoder?	Unchecked
Support AMBA Memory Remap Feature?	Checked
Support Arbiter Pause Mode?	Checked
Support Delayed Pause Action?	Checked
Include Arbiter Interface?	Checked
Include Weighted Token Arbitration?	Unchecked
Include Weighted Token Outputs?	(ignored)
Support full incrementing bursts?	Unchecked
Total number of slave select lines in the system	5
Number of AHB slave ports in Normal Mode	5
Number of AHB slave ports in Boot Mode	5

1. If not already highlighted, click on “Top Level Parameters” and use the values in Table 2 to complete the dialog box, then click the “Next” button

The following shows where to enter these values in “Top Level Parameters.”.


Top Level Parameters
Slave Arbiter Configuration
Arbiter Priority Assignments
Slave Memory Region Definition
Normal Mode Address Map
Boot Mode Address Map
Weighted Token Control Signals
SYNOPSYS
Previous
Next
Apply
Default
Dialog
Report
Help

Top Level Parameters

AMBA Lite ?	☐
Number of AHB Master Ports	1
AHB System Address Width	32
AHB Data Bus Width	32
External endian control?	☐
System Endianness	Little-Endian
External Decoder?	☐
Support AMBA Memory Remap Feature?	✓
Support Arbiter Pause Mode?	✓
Support Delayed Pause Action?	✓
Include Arbiter Interface ?	✓
Include Weighted Token Arbitration ?	☐
Include Weighted Token Outputs ?	☐
Support full incrementing bursts?	☐
Total Number of Slave Select Lines in the system	5
Number of AHB Slave Ports in Normal Mode	5
Number of AHB Slave Ports in Boot Mode	5
External Decoder Provides HSELS back to DW_ahb for routing to Slaves?	☐

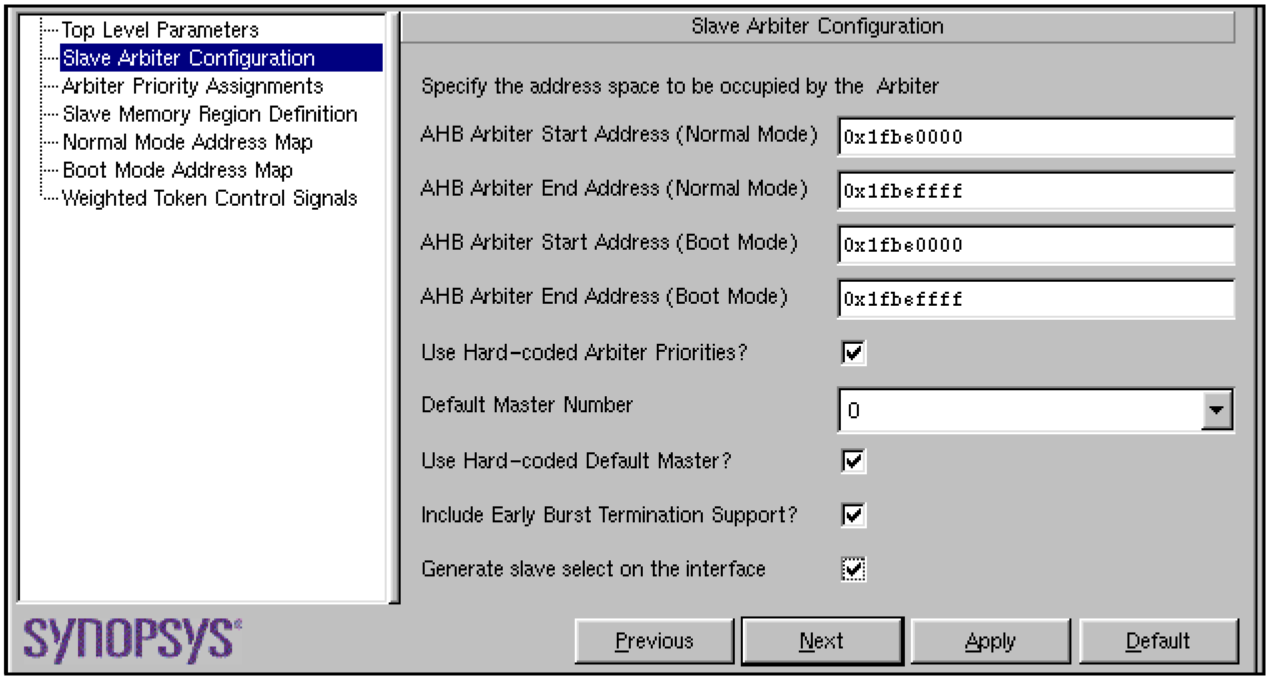
Setting Slave Arbiter Configuration – DW_ahb

The arbiter slave interface is an optional AHB slave over which the internal registers of DW_ahb may be read from and written to by any master in the system. Table 3 shows the parameters that you can set for the Slave Arbiter Configuration.

Table 3: Slave Arbiter Configuration

Item	Value
AHB Arbiter Start Address (Normal Mode)	`0x1fbe0000`
AHB Arbiter End Address (Normal Mode)	`0x1fbeffff`
AHB Arbiter Start Address (Boot Mode)	`0x1fbe0000`
AHB Arbiter End Address (Boot Mode)	`0x1fbeffff`
Use hard-coded arbiter priorities?	Checked
Default master number	0
Use hard-coded default master	Checked
Include early burst termination support	Checked
Generate slave select on the interface	Checked

1. If not highlighted, click on "Slave Arbiter Configuration" and use the values in Table 3 to complete the dialog box, then click the "Next" button.


Top Level Parameters
Slave Arbiter Configuration
Arbiter Priority Assignments
Slave Memory Region Definition
Normal Mode Address Map
Boot Mode Address Map
Weighted Token Control Signals
SYNOPSYS
Previous
Next
Apply
Default

Slave Arbiter Configuration

Specify the address space to be occupied by the Arbiter

AHB Arbiter Start Address (Normal Mode) 0x1fbe0000
AHB Arbiter End Address (Normal Mode) 0x1fbeffff
AHB Arbiter Start Address (Boot Mode) 0x1fbe0000
AHB Arbiter End Address (Boot Mode) 0x1fbeffff
Use Hard-coded Arbiter Priorities? ✓
Default Master Number 0
Use Hard-coded Default Master? ✓
Include Early Burst Termination Support? ✓
Generate slave select on the interface ✓

Figure 3: DW_ahb Configuration – Slave Arbiter

Making Arbiter Priority Assignments – DW_ahb

Because there is only one master specified, no arbitration is required, and coreConsultant skips this activity.

Setting Slave Configurations – DW_ahb

The setup of the slaves is split into two forms. The Slave Memory Region form determines memory regions and data/response paths. The Address Map form sets the addresses for all slaves. Table 4 shows the Slave Memory Region parameters.

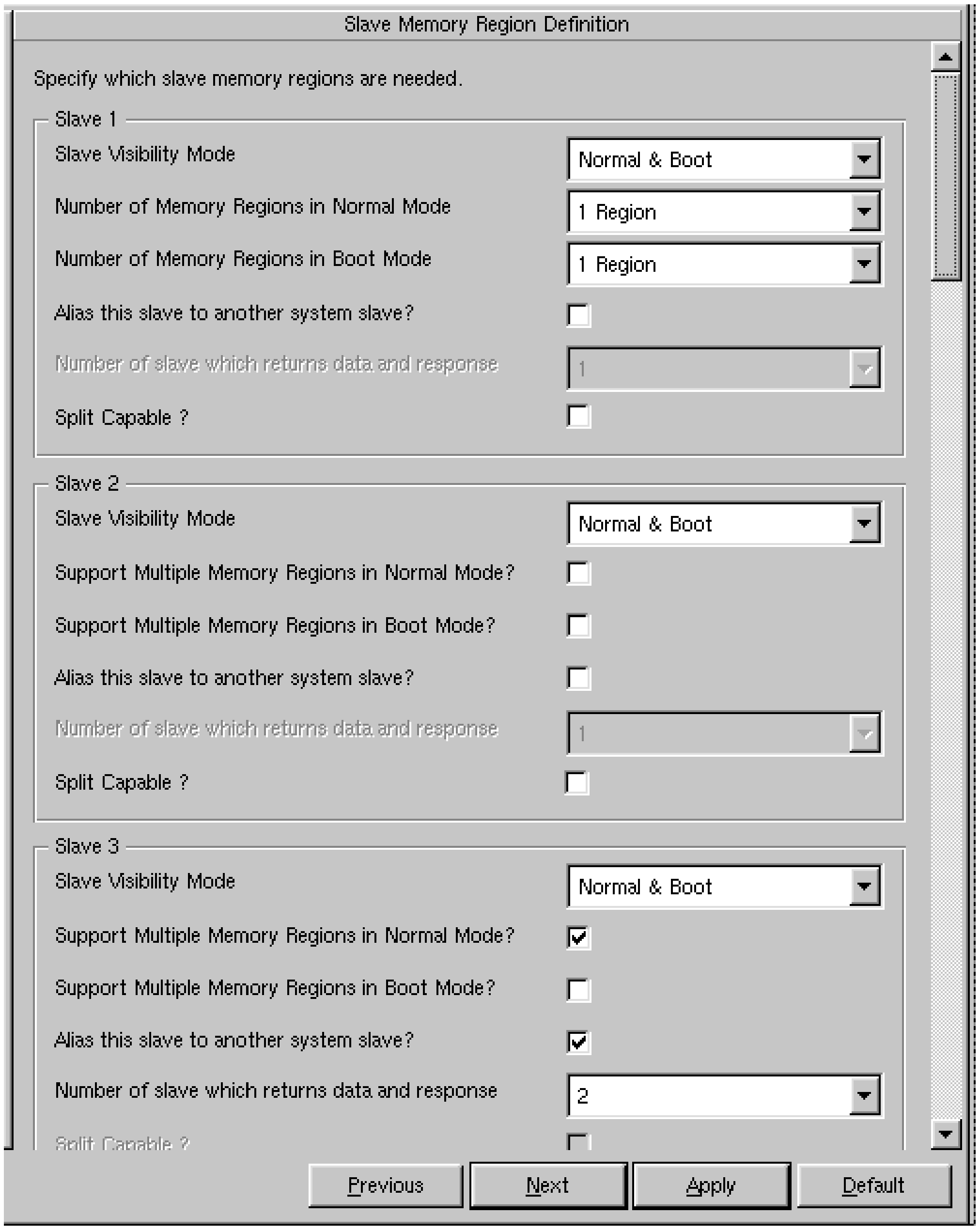
Table 4: Slave Memory Region Description

Slave #	Item	Value
Slave 1 (DW_apb)	Slave Visibility Mode	Normal & Boot
	Number of memory regions in normal mode	1 Region
	Number of memory regions in boot mode	1 Region
	Alias this slave to another system slave?	Unchecked
	Number of slave which return data and responsed	(ingored)
	Split capable?	Unchecked
Slave 2 (external memory port of DW_memctl)	Slave Visibility Mode	Normal & Boot
	Support multiple memory regions in normal mode?	Unchecked
	Support multiple memory regions in boot mode?	Unchecked
	Alias this slave to another system slave?	Unchecked
	Number of slave which return data and responsed	(ingored)
	Split capable?	Unchecked
Slave 3 (register port of DW_memctl)	Slave Visibility Mode	Normal & Boot
	Support multiple memory regions in normal mode?	Checked
	Support multiple memory regions in boot mode?	Unchecked
	Alias this slave to another system slave?	Checked
	Number of slave which returns data and response	2
	Split capable?	(ignored)

Table 4: Slave Memory Region Description (Continued)

Slave #	Item	Value
Slave 4 (BFM)	Slave Visibility Mode	Normal & Boot
	Support multiple memory regions in normal mode?	Unchecked
	Support multiple memory regions in boot mode?	Unchecked
	Alias this slave to another system slave?	Unchecked
	Number of slave which returns data and response	(ignored)
	Split capable?	Unchecked
Slave 5 (DW_ahb_ictl)	Slave Visibility Mode	Normal & Boot
	Support multiple memory regions in normal mode?	Unchecked
	Support multiple memory regions in normal mode?	Unchecked
	Alias this slave to another system slave?	Unchecked
	Number of slave which returns data and response	(ignored)
	Split capable?	Unchecked

1. If unhighlighted, click on "Slave Memory Region Definition" and use the values in Table 4 to complete the dialog box, then click the "Next" button.



Slave Memory Region Definition

Specify which slave memory regions are needed.

Slave 1

Slave Visibility Mode	Normal & Boot
Number of Memory Regions in Normal Mode	1 Region
Number of Memory Regions in Boot Mode	1 Region
Alias this slave to another system slave?	☐
Number of slave which returns data and response	1
Split Capable ?	☐

Slave 2

Slave Visibility Mode	Normal & Boot
Support Multiple Memory Regions in Normal Mode?	☐
Support Multiple Memory Regions in Boot Mode?	☐
Alias this slave to another system slave?	☐
Number of slave which returns data and response	1
Split Capable ?	☐

Slave 3

Slave Visibility Mode	Normal & Boot
Support Multiple Memory Regions in Normal Mode?	✓
Support Multiple Memory Regions in Boot Mode?	☐
Alias this slave to another system slave?	✓
Number of slave which returns data and response	2
Split Capable ?	☐

Figure 4: Slave Memory Region Definition

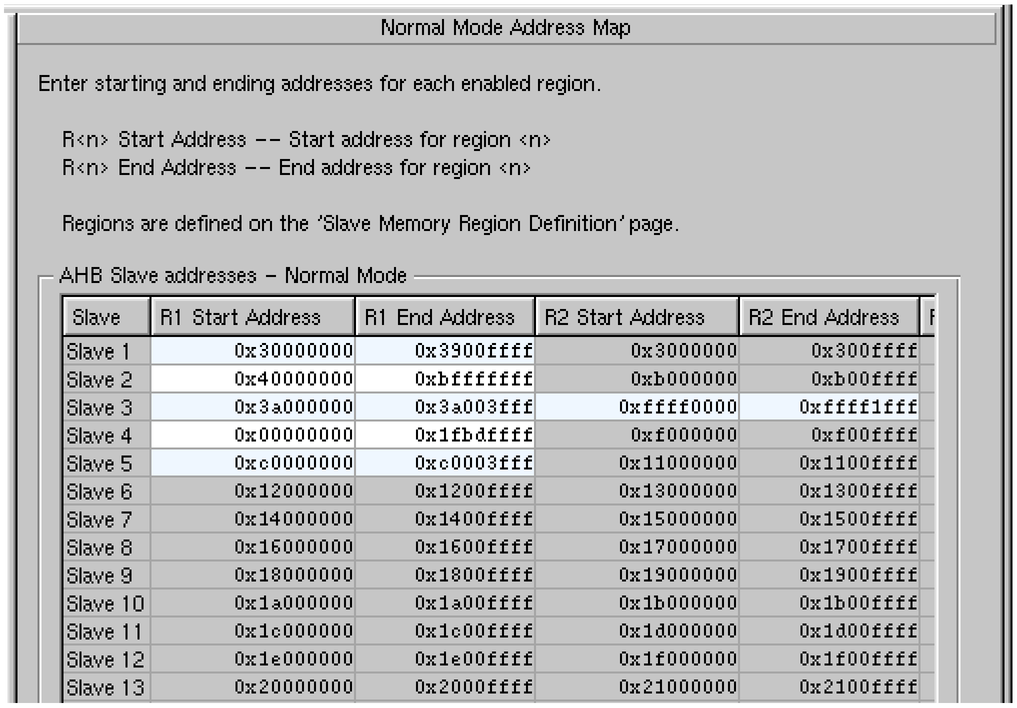
Setting the Memory Address Map – DW_ahb

Use this form to specify the address space configuration of all Slaves. Table 5 shows the parameters that you set for this item.

Table 5: Normal Mode Slave Address Map

Slave	R1 Start Address	R1 End Address	R2 Start Address	R2 End Address
Slave 1	`0x30000000`	`0x3900ffff`	`N/A`	`N/A`
Slave 2	`0x40000000`	`0xbfffffff`	`N/A`	`N/A`
Slave 3	`0x3a000000`	`0x3a003fff`	`0xffff0000`	`0xffff1fff`
Slave 4	`0x00000000`	`0x1fbdffff`	`N/A`	`N/A`
Slave 5	`0xc0000000`	`0xc0003fff`	`N/A`	`N/A`

1. **If unhighlighted, click on “Address Map” and use the values in Table 5 to complete the dialog box, then click the “Next” button.**



Normal Mode Address Map

Enter starting and ending addresses for each enabled region.

R<n> Start Address -- Start address for region <n>
R<n> End Address -- End address for region <n>

Regions are defined on the 'Slave Memory Region Definition' page.

AHB Slave addresses - Normal Mode

Slave	R1 Start Address	R1 End Address	R2 Start Address	R2 End Address
Slave 1	0x30000000	0x3900ffff	0x3000000	0x300ffff
Slave 2	0x40000000	0xbfffffff	0xb000000	0xb00ffff
Slave 3	0x3a000000	0x3a003fff	0xffff0000	0xffff1fff
Slave 4	0x00000000	0x1fbdffff	0xf000000	0xf00ffff
Slave 5	0xc0000000	0xc0003fff	0x11000000	0x1100ffff
Slave 6	0x12000000	0x1200ffff	0x13000000	0x1300ffff
Slave 7	0x14000000	0x1400ffff	0x15000000	0x1500ffff
Slave 8	0x16000000	0x1600ffff	0x17000000	0x1700ffff
Slave 9	0x18000000	0x1800ffff	0x19000000	0x1900ffff
Slave 10	0x1a000000	0x1a00ffff	0x1b000000	0x1b00ffff
Slave 11	0x1c000000	0x1c00ffff	0x1d000000	0x1d00ffff
Slave 12	0x1e000000	0x1e00ffff	0x1f000000	0x1f00ffff
Slave 13	0x20000000	0x2000ffff	0x21000000	0x2100ffff

Figure 5: Address Map Dialog Box

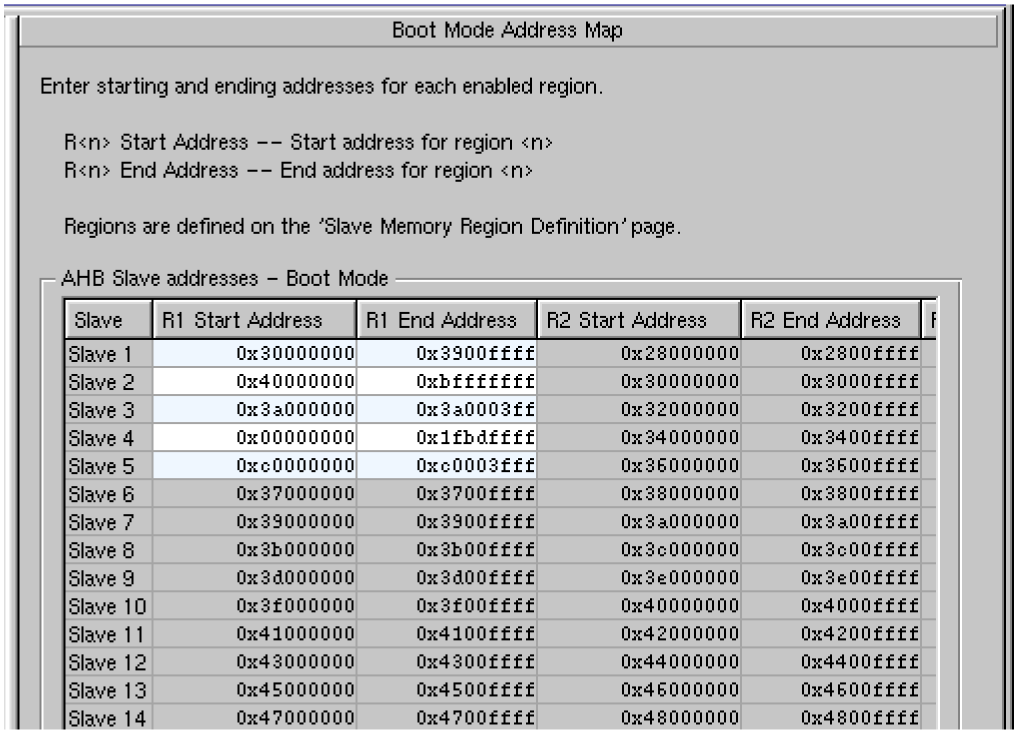
Boot Mode Address Map – DW_ahb

The Boot Mode Address Map appears, showing regions that are available to enter boot mode addresses.

Table 6: Boot Mode Slave Address Map

Slave	R1 Start Address	R1 End Address	R2 Start Address	R2 End Address
Slave 1	`0x30000000`	`0x3900ffff`	N/A	N/A
Slave 2	`0x40000000`	`0xbfffffff`	N/A	N/A
Slave 3	`0x3a000000`	`0x3a0003ff`	N/A	N/A
Slave 4	`0x00000000`	`0x1fbdffff`	N/A	N/A
Slave 5	`0xc0000000`	`0xc0003fff`	N/A	N/A

1. **Using the values in Table 6, complete the Boot Mode Address Map dialog box.**



Boot Mode Address Map

Enter starting and ending addresses for each enabled region.

R<n> Start Address -- Start address for region <n>
R<n> End Address -- End address for region <n>

Regions are defined on the 'Slave Memory Region Definition' page.

AHB Slave addresses - Boot Mode

Slave	R1 Start Address	R1 End Address	R2 Start Address	R2 End Address
Slave 1	0x30000000	0x3900ffff	0x28000000	0x2800ffff
Slave 2	0x40000000	0xbfffffff	0x30000000	0x3000ffff
Slave 3	0x3a000000	0x3a0003ff	0x32000000	0x3200ffff
Slave 4	0x00000000	0x1fbdffff	0x34000000	0x3400ffff
Slave 5	0xc0000000	0xc0003fff	0x36000000	0x3600ffff
Slave 6	0x37000000	0x3700ffff	0x38000000	0x3800ffff
Slave 7	0x39000000	0x3900ffff	0x3a000000	0x3a00ffff
Slave 8	0x3b000000	0x3b00ffff	0x3c000000	0x3c00ffff
Slave 9	0x3d000000	0x3d00ffff	0x3e000000	0x3e00ffff
Slave 10	0x3f000000	0x3f00ffff	0x40000000	0x4000ffff
Slave 11	0x41000000	0x4100ffff	0x42000000	0x4200ffff
Slave 12	0x43000000	0x4300ffff	0x44000000	0x4400ffff
Slave 13	0x45000000	0x4500ffff	0x46000000	0x4600ffff
Slave 14	0x47000000	0x4700ffff	0x48000000	0x4800ffff

2. **When completed, click the "Next" button.**

   Notice that you receive a message that there are "No remaining page with enabled controls." The "Weighted Token Control Signals" is not needed, since you did not check the weighted token box.

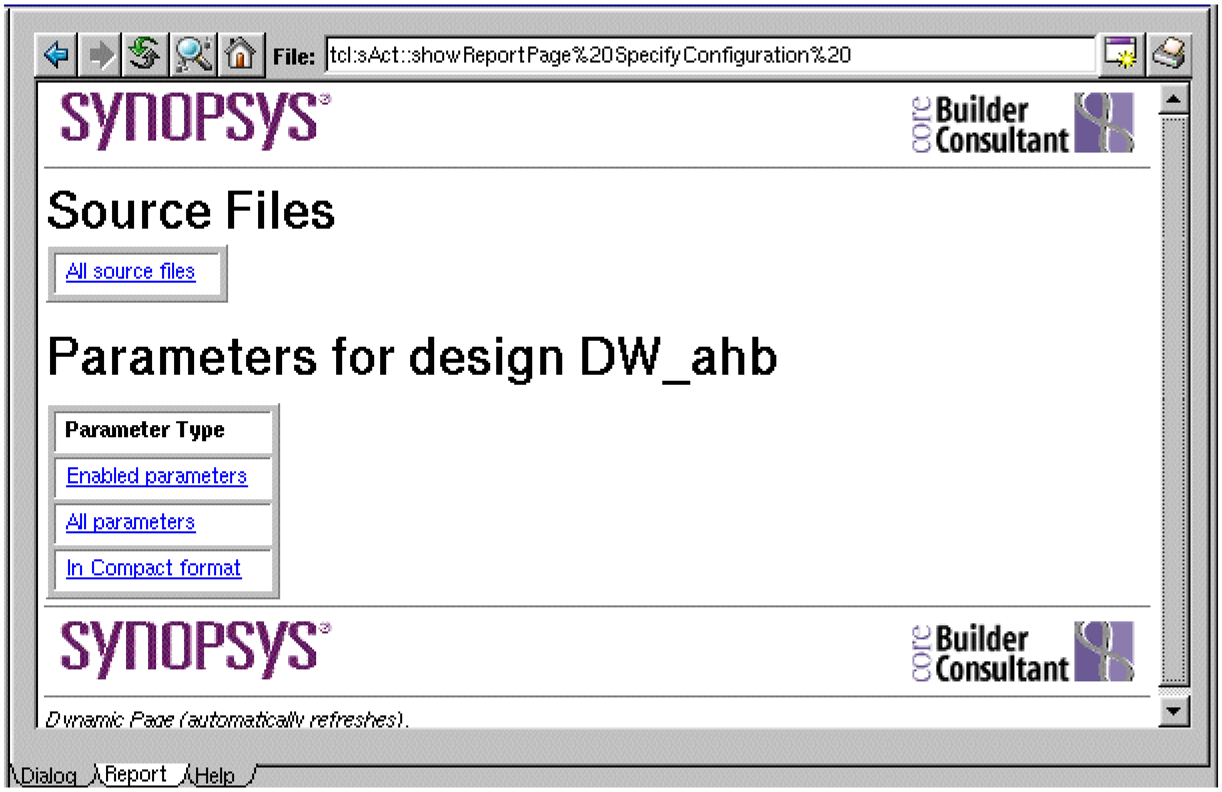
3. **OK the message dialog box.**

Checking and Writing the Configuration – DW_ahb

1. **When you have completed all of the needed configuration activities, click the "Apply" button at the bottom of the dialog box.**

   coreConsultant checks the configuration settings and writes a number of configuration files for this DW_ahb component. While processing, coreConsultant shows its progress in the bottom Status window.

   The "Specify Configuration" activity completes, and the parameter report displays in the right window pane.



File: tcl:sAct::showReportPage%20SpecifyConfiguration%20

SYNOPSYS

core Builder Consultant

Source Files

All source files

Parameters for design DW_ahb

Parameter Type
Enabled parameters
All parameters
In Compact format

SYNOPSYS

core Builder Consultant

Dynamic Page (automatically refreshes).

   The basic configuration information is at the top of the report, and specific master and slave information is below the Basic section.

   Notice that the "Report" tab below this window is highlighted. You can access this report by clicking on this tab.

2. **Click on the "All source files" link to view the source files that have been created. Files that are not linked are encrypted.**

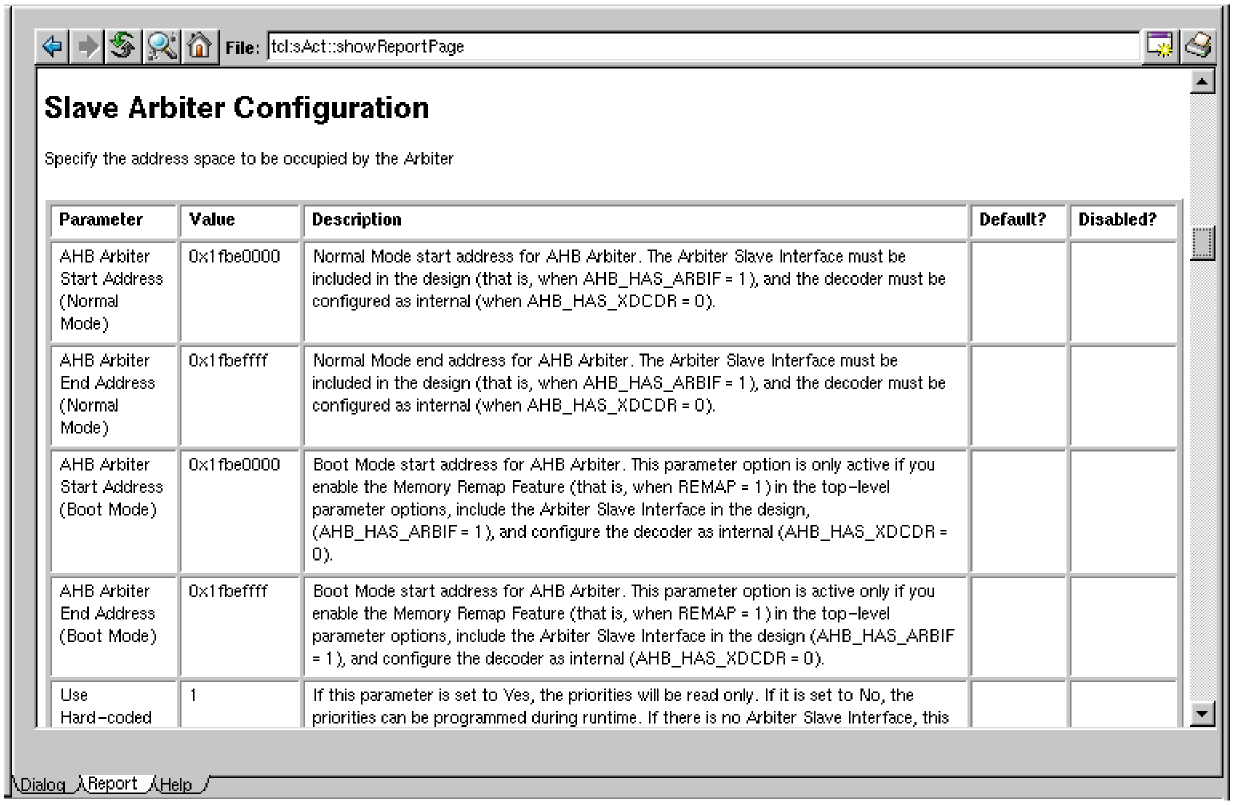
3. **Click on the "DW_amba_constants.v" file at the bottom of the table to view these global AMBA constants.**

   You can use these constants in your testbench stimulus.

4. Click the back arrow twice to return to the first report page; then click on the "All Parameters" link

The parameters report page appears with current parameter information.

5. Scroll down within this window to view the Slave Arbiter Configuration.



Slave Arbiter Configuration

Specify the address space to be occupied by the Arbiter

Parameter	Value	Description	Default?	Disabled?
AHB Arbiter Start Address (Normal Mode)	0x1fbe0000	Normal Mode start address for AHB Arbiter. The Arbiter Slave Interface must be included in the design (that is, when AHB_HAS_ARBIF = 1), and the decoder must be configured as internal (when AHB_HAS_XDCDR = 0).		
AHB Arbiter End Address (Normal Mode)	0x1fbeffff	Normal Mode end address for AHB Arbiter. The Arbiter Slave Interface must be included in the design (that is, when AHB_HAS_ARBIF = 1), and the decoder must be configured as internal (when AHB_HAS_XDCDR = 0).		
AHB Arbiter Start Address (Boot Mode)	0x1fbe0000	Boot Mode start address for AHB Arbiter. This parameter option is only active if you enable the Memory Remap Feature (that is, when REMAP = 1) in the top-level parameter options, include the Arbiter Slave Interface in the design, (AHB_HAS_ARBIF = 1), and configure the decoder as internal (AHB_HAS_XDCDR = 0).		
AHB Arbiter End Address (Boot Mode)	0x1fbeffff	Boot Mode start address for AHB Arbiter. This parameter option is active only if you enable the Memory Remap Feature (that is, when REMAP = 1) in the top-level parameter options, include the Arbiter Slave Interface in the design (AHB_HAS_ARBIF = 1), and configure the decoder as internal (AHB_HAS_XDCDR = 0).		
Use Hard-coded	1	If this parameter is set to Yes, the priorities will be read only. If it is set to No, the priorities can be programmed during runtime. If there is no Arbiter Slave Interface, this		

Each parameter text and value is accompanied by a full description of the parameter and an explanation of dependencies.

6. View the Activity list again.

Notice that the "Specify Configuration" activity item is now checked (completed).

- ❍ The synthesis activity uses the encrypted RTL that has been written.
- ❍ Verilog and C header files, which contain constants and register map definitions used by testbench developers are also written. See *workspace*/c_headers and *workspace*/verilog_headers for these unencrypted files.

Verification Activity

There are two types of verification you can perform:

- **Formal Verification** – you can run formal verification (Formality) from coreConsultant to compare the RTL generated with your post-synthesis netlist generated during the Synthesis activity phase
- **Setup and Run Simulation** – simulates configured RTL (encrypted) or the configured GTECH model in the verification environment supplied with the component.
  - If you are using the Synopsys VCS simulator, you can set up and run your simulation directly on the encrypted RTL. The Synopsys VCS simulator can read encrypted RTL code directly without requiring a GTECH model.
  - If you are using another simulator, you must first generate a GTECH netlist of the RTL and use this netlist in your simulation.

Generate GTECH Model

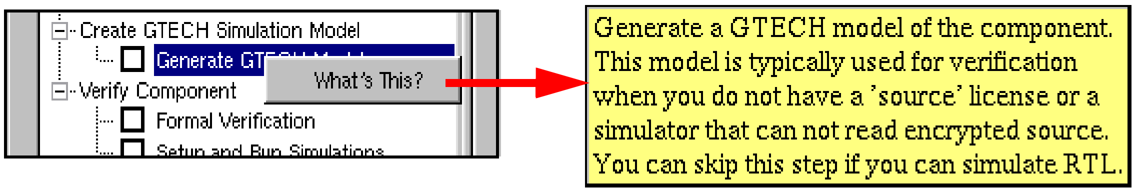
GTECH is a technology independent mapped netlist that is used for simulation with non-Synopsys simulators, which cannot read encrypted RTL. If you have VCS, you can skip this section.

The GTECH netlist is created by the “Generate GTECH Model” activity.

Verification tests are included with the DW_ahb to test the features you have configured.

1. **Right-click on the “Generate GTECH Model” text and then click on the “What’s This?” box to obtain help information about this activity.**

   The help information appears in a overlay window, as shown.



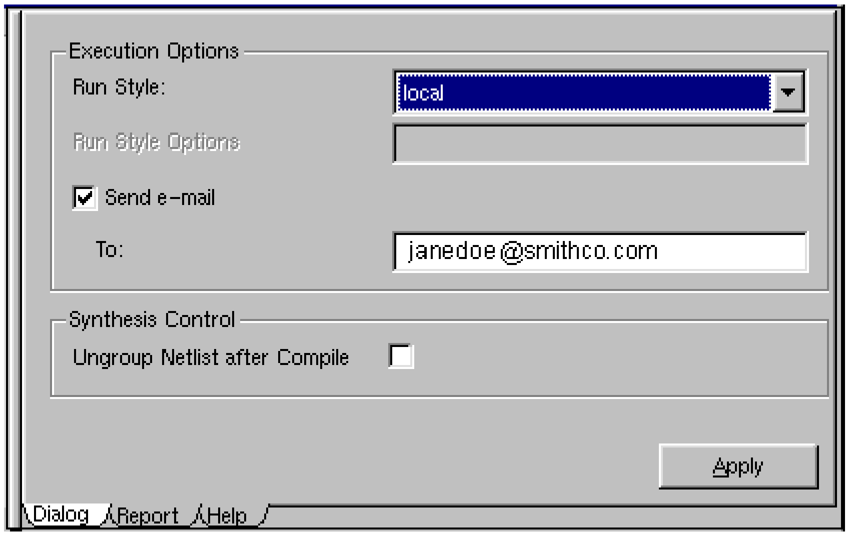
Generate a GTECH model of the component. This model is typically used for verification when you do not have a ’source’ license or a simulator that can not read encrypted source. You can skip this step if you can simulate RTL.

Figure 6: Generate GTECH Model Help Window

2. **Click on the help information box to dismiss it.**
3. **Click on the “Generate GTECH Model” text in the Activity List**

   The “Parameters for Generate GTECH Model” dialog box appears.

4. View the Parameters for Generate GTECH Model dialog box.



Execution Options

Run Style: local

Run Style Options

[✓] Send e-mail

To: janedoe@smithco.com

Synthesis Control

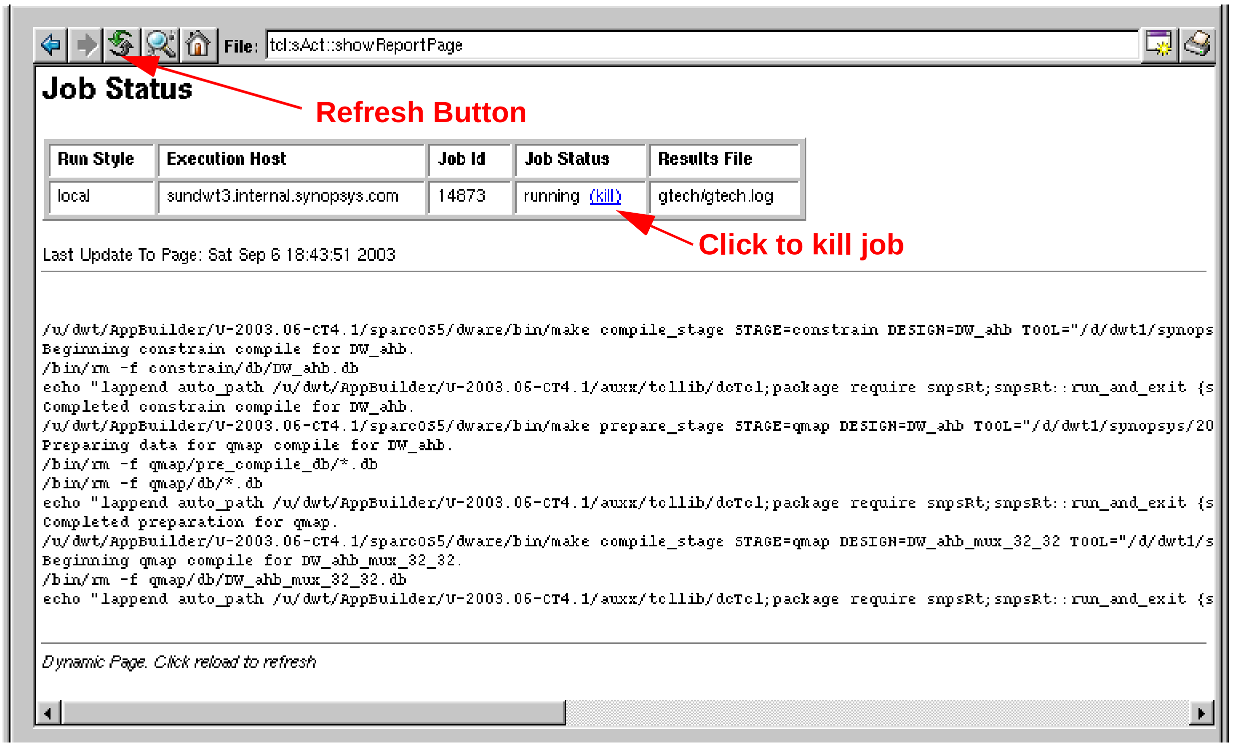
Ungroup Netlist after Compile [ ]

Here you can specify your “Run Style” (default: local) and whether to send email when the generation process has completed. From the previous help window, you learned that VCS does not need a GTECH netlist, and can simulate encrypted RTL directly.

5. Verify that the Run Style is “local,” enter your e-mail address, and “Apply” the dialog box.

The default is to send you email when the generate process has completed. The Job Status report appears. This is a dynamic report, so use the Refresh button to frequently refresh the report.

When finished, a Verilog and a VHDL version of the GTECH model are created. These objects can be found at <workspace>/gtech/qmap/db.



Job Status

Run Style	Execution Host	Job Id	Job Status	Results File
local	sundwt3.internal.synopsys.com	14873	running (kill)	gtech/gtech.log

Last Update To Page: Sat Sep 6 18:43:51 2003

```
/u/dwt/AppBuilder/U-2003.06-CT4.1/sparcOS5/dware/bin/make compile_stage STAGE=constrain DESIGN=DW_ahb TOOL="/d/dwt1/synops
Beginning constrain compile for DW_ahb.
/bin/rm -f constrain/db/DW_ahb.db
echo "lappend auto_path /u/dwt/AppBuilder/U-2003.06-CT4.1/auxx/tcllib/dcTcl;package require snpsRt;snpsRt::run_and_exit {s
Completed constrain compile for DW_ahb.
/u/dwt/AppBuilder/U-2003.06-CT4.1/sparcOS5/dware/bin/make prepare_stage STAGE=qmap DESIGN=DW_ahb TOOL="/d/dwt1/synopsys/20
Preparing data for qmap compile for DW_ahb.
/bin/rm -f qmap/pre_compile_db/*.db
/bin/rm -f qmap/db/*.db
echo "lappend auto_path /u/dwt/AppBuilder/U-2003.06-CT4.1/auxx/tcllib/dcTcl;package require snpsRt;snpsRt::run_and_exit {s
Completed preparation for qmap.
/u/dwt/AppBuilder/U-2003.06-CT4.1/sparcOS5/dware/bin/make compile_stage STAGE=qmap DESIGN=DW_ahb_mux_32_32 TOOL="/d/dwt1/s
Beginning qmap compile for DW_ahb_mux_32_32.
/bin/rm -f qmap/db/DW_ahb_mux_32_32.db
echo "lappend auto_path /u/dwt/AppBuilder/U-2003.06-CT4.1/auxx/tcllib/dcTcl;package require snpsRt;snpsRt::run_and_exit {s
```

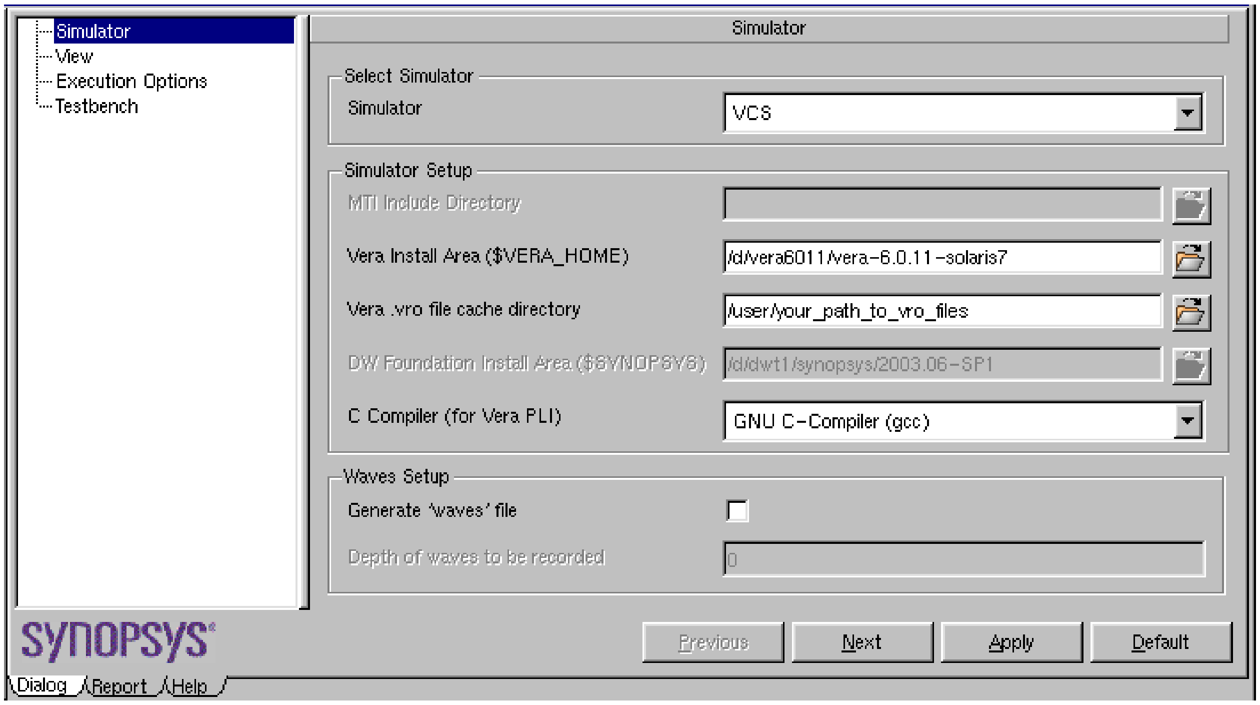
Dynamic Page. Click reload to refresh

Verify Component (Setup and Run Simulations)

This activity is where you choose and run the simulator you will use to perform DW_ahb tests. It also allows you to configure the testbench and select the tests to run.

1. **Click on the "Setup and Run Simulations" activity beneath the Verify Component category.**

   A view appears, showing the detail for the "Simulator" item.



Simulator

Select Simulator

Simulator	VCS

Simulator Setup

MTI Include Directory	
Vera Install Area ($VERA_HOME)	/d/vera6011/vera-6.0.11-solaris7
Vera .vro file cache directory	/user/your_path_to_vro_files
DW Foundation Install Area ($SYNOPSYS)	/d/dwt1/synopsys/2003.06-SP1
C Compiler (for Vera PLI)	GNU C-Compiler (gcc)

Waves Setup

Generate 'waves' file	☐
Depth of waves to be recorded	0

Figure 7: Simulate Activity – Simulation Setup

   The figure shows the Selected Simulator as VCS.

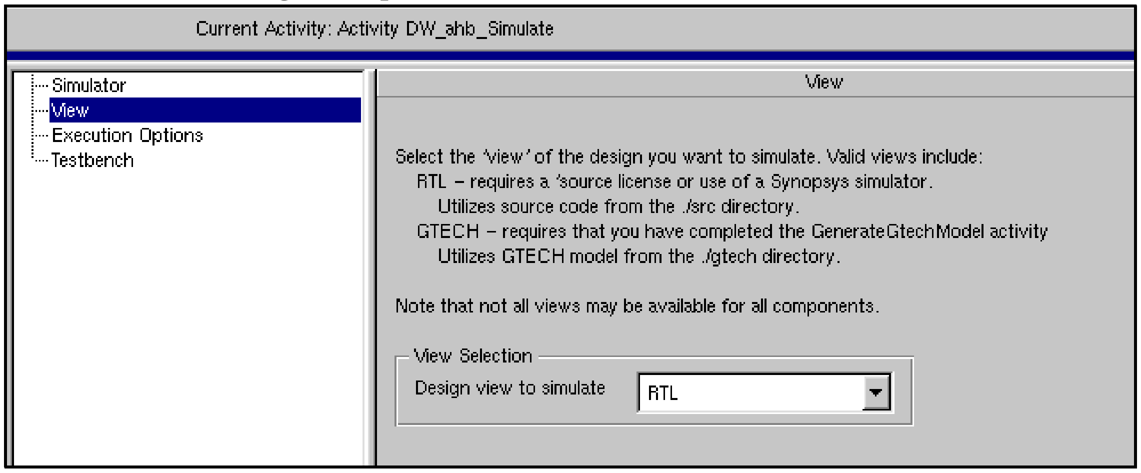
2. **If you are using a different simulator, choose it from the pull-down list, and enter any unique VERA or compiler requirements. Otherwise, you should be able to use all the defaults.**

   You can choose to generate a wave file of the simulation run which will dump the output waveforms from the simulation.

   Under your *workspace*/sim directory, a separate directory is created for each test in the testbench setup which is discussed in Step 5 to follow. A description of each of the tests can be found in the *DesignWare DW_ahb Databook*.

3. Click on the “View” text or the “Next” button to go to the “View” screen.

The “View” dialog box options are shown.



Select the 'view' of the design you want to simulate. Valid views include:
RTL – requires a 'source license or use of a Synopsys simulator.
Utilizes source code from the ./src directory.
GTECH – requires that you have completed the GenerateGtechModel activity
Utilizes GTECH model from the ./gtech directory.

Note that not all views may be available for all components.

View Selection
Design view to simulate: RTL

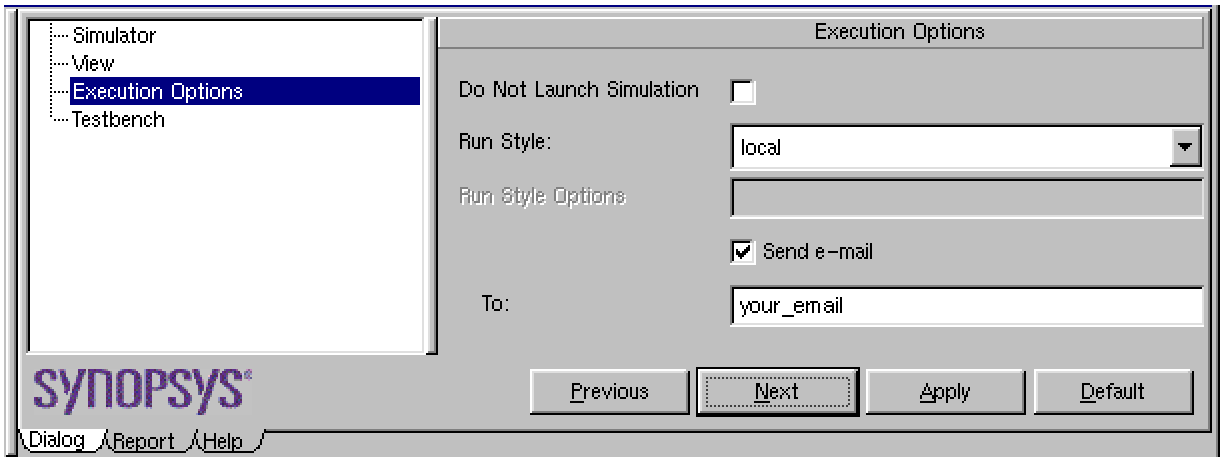
The figure shows that the RTL design is selected for a VCS simulation. A non-Synopsys simulator may not simulate encrypted RTL directly and thus have a GTECH view shown.

Note
To select “GTECH” and run a simulation, you must have previously performed the “Create GTECH Model” activity.

4. Click on Execution Options” text (or click the “Next” button).

The “Execution Options” dialog box appears. It allows you to customize the way the simulation runs (local or remote) and to send you email when finished.

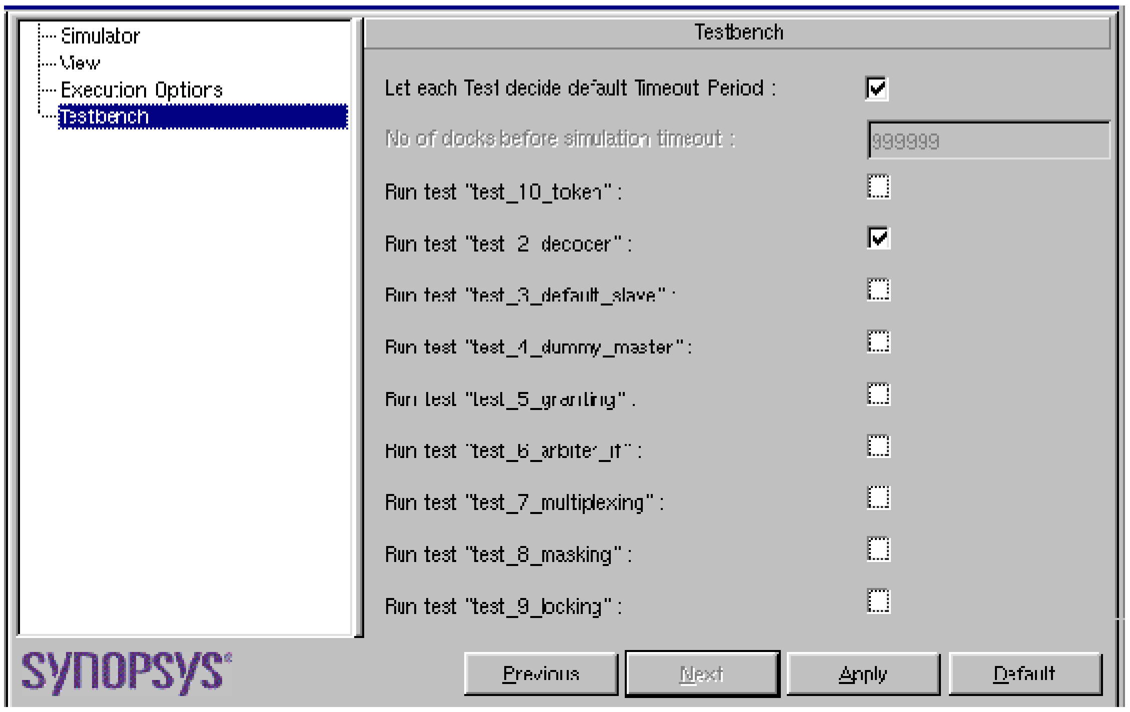
Use the defaults for this tutorial. Depending on the speed of your machine, and which tests you run, the simulation time could take up to an hour. Enter your email address to be notified when the simulation run finishes.



Do Not Launch Simulation ☐
Run Style: local
Run Style Options
☑ Send e-mail
To: your_email

5. **Click on the “Testbench” text (or click the “Next” button).**

   The detail for Testbench setup appears.



Testbench

Let each Test decide default Timeout Period : ☑

No of clocks before simulation timeout : 999999

Run test "test_10_token" : ☐

Run test "test_2_decoder" : ☑

Run test "test_3_default_slave" : ☐

Run test "test_4_dummy_master" : ☐

Run test "test_5_granting" : ☐

Run test "test_6_arbiter_if" : ☐

Run test "test_7_multiplexing" : ☐

Run test "test_8_masking" : ☐

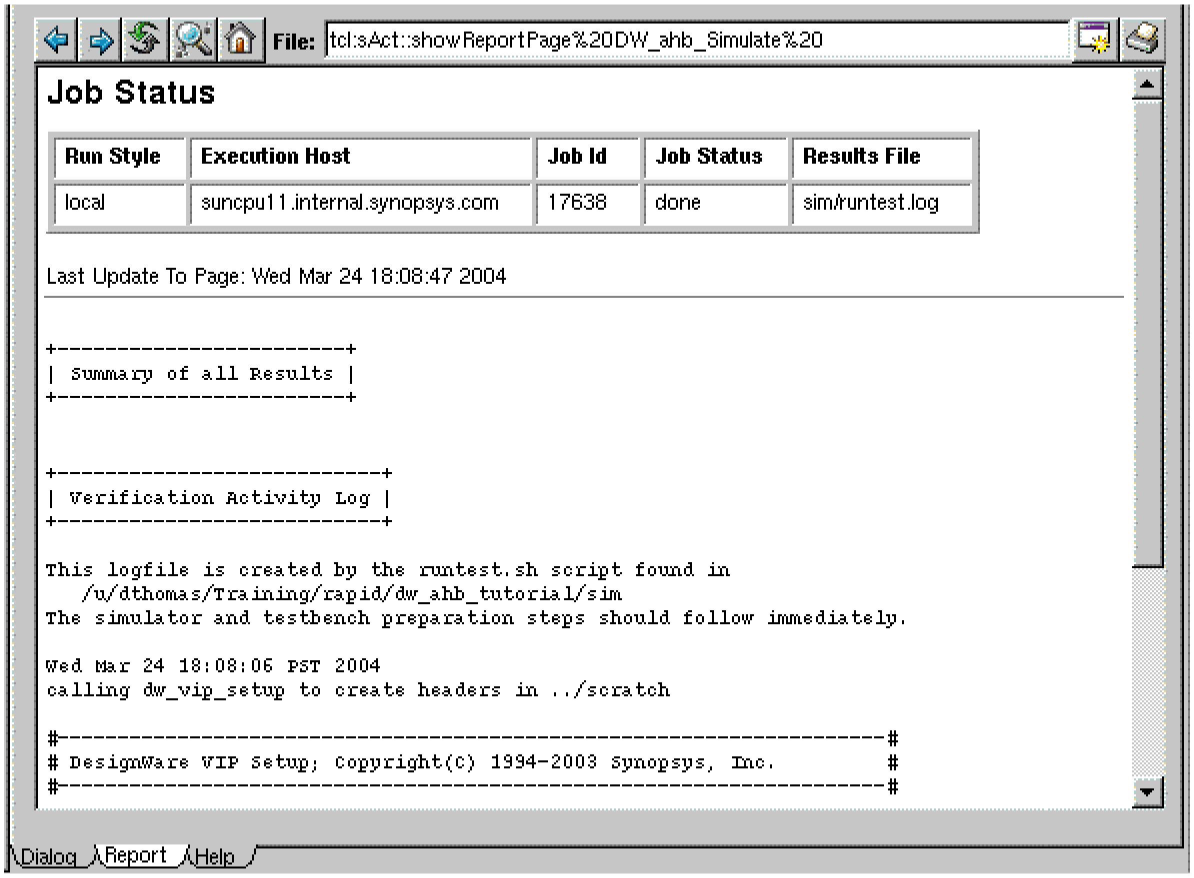
Run test "test_9_locking" : ☐

   This dialog box allows you to change some of the operating conditions of the testbench, such as clock period. This is also where you choose which tests are to be run on the DW_ahb.

6. **Make sure the “Let the testbench decide default Clock Period” option is checked.**

7. **Uncheck all but the ‘Run test “test_2_decoder”’ test to reduce the length of the simulation run, and “Apply” the dialog box.**

The simulation is launched and the “test_2_decoder” test runs. When the test is launched, the dialog box view changes to the “Report” view.



Job Status

Run Style	Execution Host	Job Id	Job Status	Results File
local	suncpu11.internal.synopsys.com	17638	done	sim/runtest.log

Last Update To Page: Wed Mar 24 18:08:47 2004

```
+-------------------------+
| Summary of all Results |
+-------------------------+



+----------------------------+
| Verification Activity Log |
+----------------------------+

This logfile is created by the runtest.sh script found in
   /u/dthomas/Training/rapid/dw_ahb_tutorial/sim
The simulator and testbench preparation steps should follow immediately.

Wed Mar 24 18:08:06 PST 2004
calling dw_vip_setup to create headers in ../scratch

#---------------------------------------------------------------------#
# DesignWare VIP Setup; Copyright(C) 1994-2003 Synopsys, Inc.         #
#---------------------------------------------------------------------#
```

The Job Status gives you information about the progress of the simulation. While the simulation is running, the “Refresh” button allows you to update the status. The “Job Status” indicates “Done” when the simulation run has finished.

Attention

The coreConsultant tool will display “Simulation Activity Completed” in the transcript window, but that does not mean the simulation is finished, just that it’s been launched.

The results are displayed below the Job Status. The results are also written to a sim/runtest.log file. There is also a log file for each test under each test directory (have a look). Note the summary of all tests appears first in the window, followed by details of the test run, and results of each test.

Synthesis Activities – DW_ahb

The Synthesis activities are beneath “Create Gate-level Netlist” in the Activity List. This is where you specify the rules and guidelines that the Design Compiler (DC) tool uses to synthesize the design.

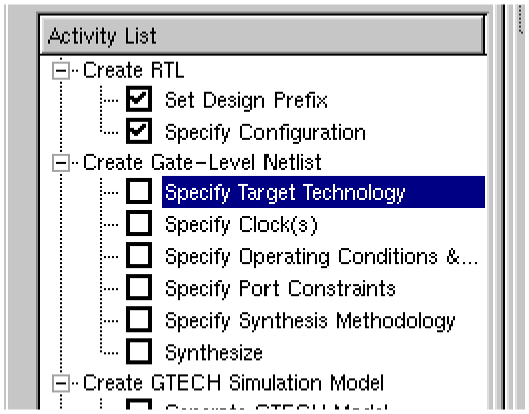

Activity List
Create RTL
Set Design Prefix
Specify Configuration
Create Gate–Level Netlist
Specify Target Technology
Specify Clock(s)
Specify Operating Conditions &...
Specify Port Constraints
Specify Synthesis Methodology
Synthesize
Create GTECH Simulation Model

Figure 8: Consultant Activity List – Synthesis

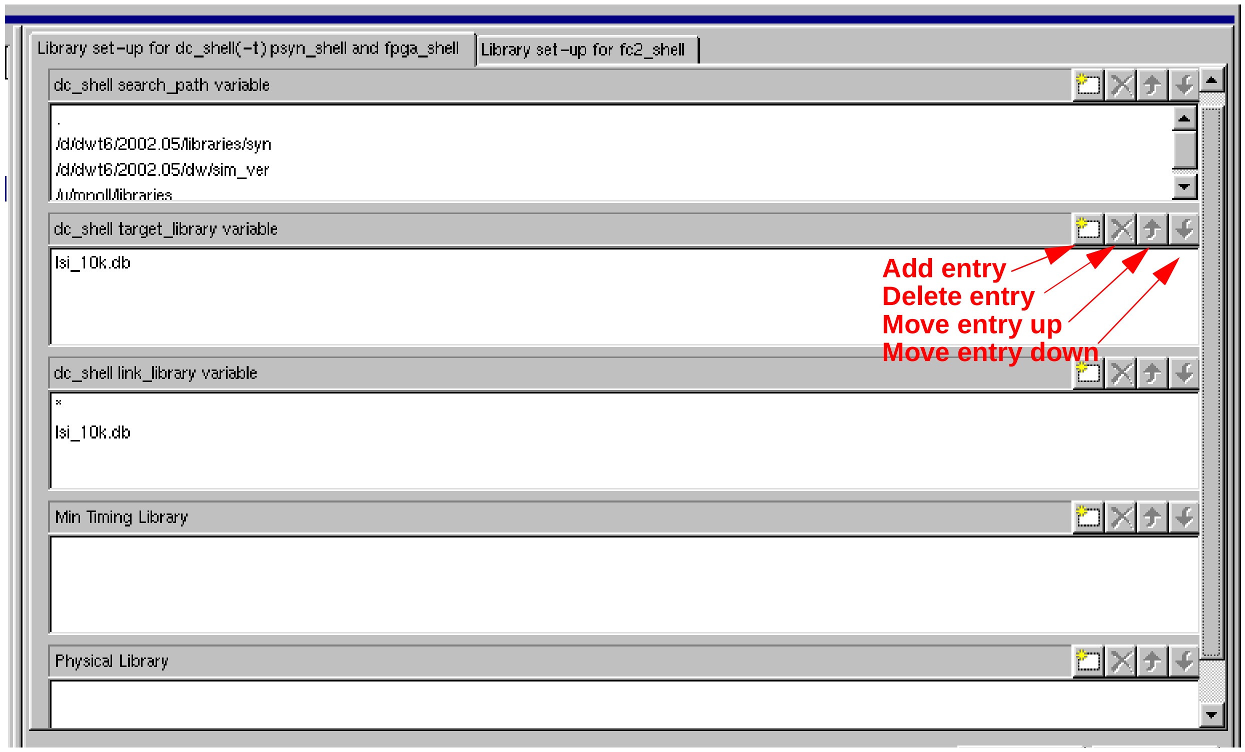
1. **Click on the “Specify Target Technology” item.**

   The coreConsultant tool will invoke and check the Design Compiler paths and variables to initialize them. This may take about a minute.

   When the checks have completed, the Setup Technology dialog box appears.

Specifying the Target Technology

The Setup Technology dialog box appears with paths to your synthesis libraries. Here is where you can add or change libraries according to your target technology.


Library set-up for dc_shell(-t) psyn_shell and fpga_shell
Library set-up for fc2_shell
dc_shell search_path variable
dc_shell target_library variable
dc_shell link_library variable
Min Timing Library
Physical Library
Add entry
Delete entry
Move entry up
Move entry down

```
.
/d/dwt6/2002.05/libraries/syn
/d/dwt6/2002.05/dw/sim_ver
/u/mnoll/libraries
```

target_library: lsi_10k.db

link_library:

```
*
lsi_10k.db
```

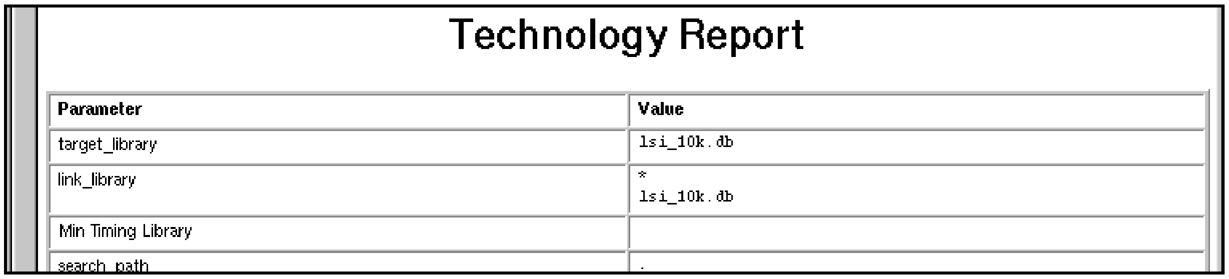
1. **To add your Technology Library, click on the new entry icon for the "cc_shell search_path variable" field to add a new entry.**

Note

You must include the path to your DWF Foundation or DWF FPGA library in the search_path, since this library is needed for synthesis.

2. **Click the "Apply" button to accept all of the paths on this dialog box.**

You are then presented with the "Technology Report" page (Report tab highlighted) giving you information about the process of loading the libraries:

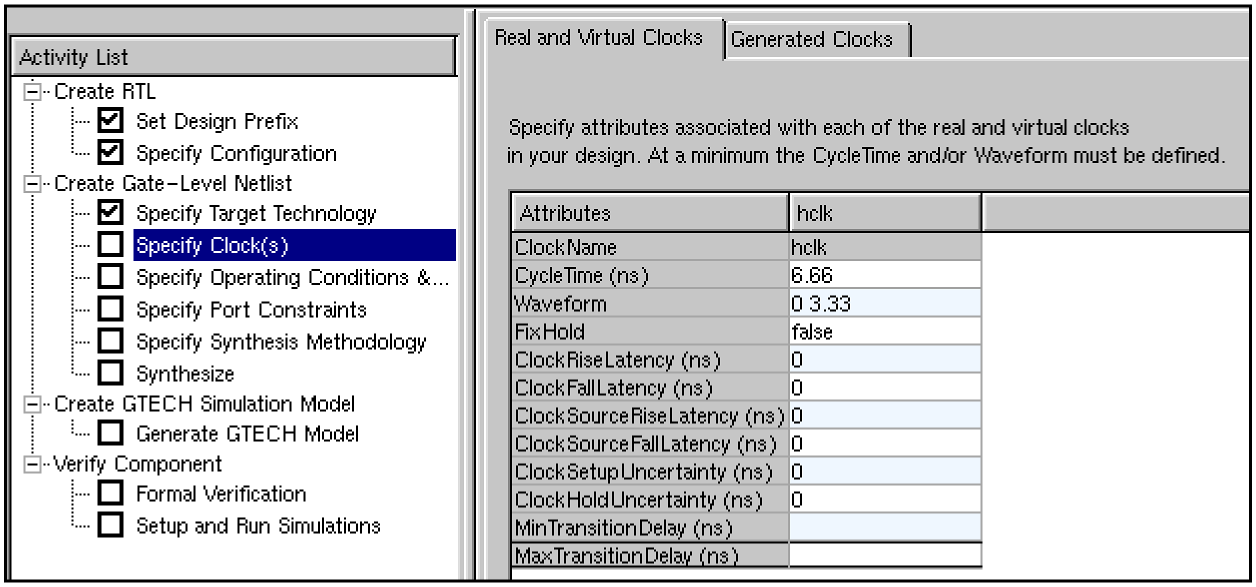


Technology Report

Parameter	Value
target_library	lsi_10k.db
link_library	* lsi_10k.db
Min Timing Library	
search_path	.

Specifying Clocks

1. Click on the Specify Clock(s) activity.



Specify attributes associated with each of the real and virtual clocks in your design. At a minimum the CycleTime and/or Waveform must be defined.

Attributes	hclk
ClockName	hclk
CycleTime (ns)	6.66
Waveform	0 3.33
FixHold	false
ClockRiseLatency (ns)	0
ClockFallLatency (ns)	0
ClockSourceRiseLatency (ns)	0
ClockSourceFallLatency (ns)	0
ClockSetupUncertainty (ns)	0
ClockHoldUncertainty (ns)	0
MinTransitionDelay (ns)	
MaxTransitionDelay (ns)	

For Synthesis Intent, Synopsys has already assigned defaults in the component knowledge base to each core's:

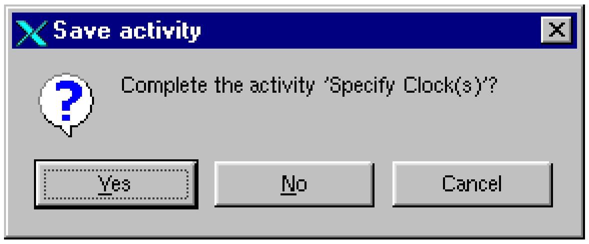
- Top-level and its ports
- Subblocks (and for those subblocks designated for individual compilation, the subblock ports)
- Clocks

In general, the default synthesis strategy for a component has been chosen carefuly and reflects the IP developers knowledge of that component. You should only need to add information specific to your environment, for example, your clock and I/O constraint information. Unique synthesis concerns for a synthesizable component are described in the databook for that component. For general synthesis strategies, refer to the Design Compiler documentation.

This view is where you specify your clocks; these are the clocks on the AHB bus (hclk) and so on. Although we are using defaults in this tutorial, you would normally enter your clock values here.

2. Click on the "Synthesize" activity to complete all of the synthesis steps up to the "Synthesize" activity.

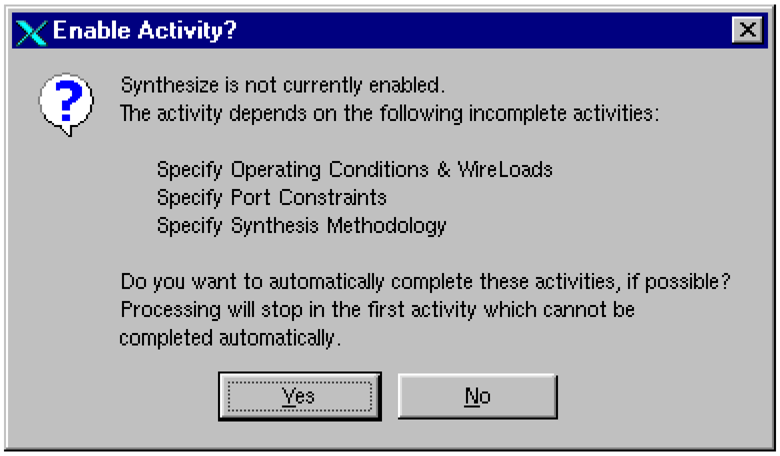
The “Save activity” dialog box for the “Specify Clock(s)” activity appears.



Complete the activity 'Specify Clock(s)'?

3. **Click “Yes” to finish this activity using the defaults.**

   The “Enable Activity?” dialog box appears to allow you to finish the other remaining activities between “Specify Clock(s)” and “Synthesis.”



Synthesize is not currently enabled.
The activity depends on the following incomplete activities:

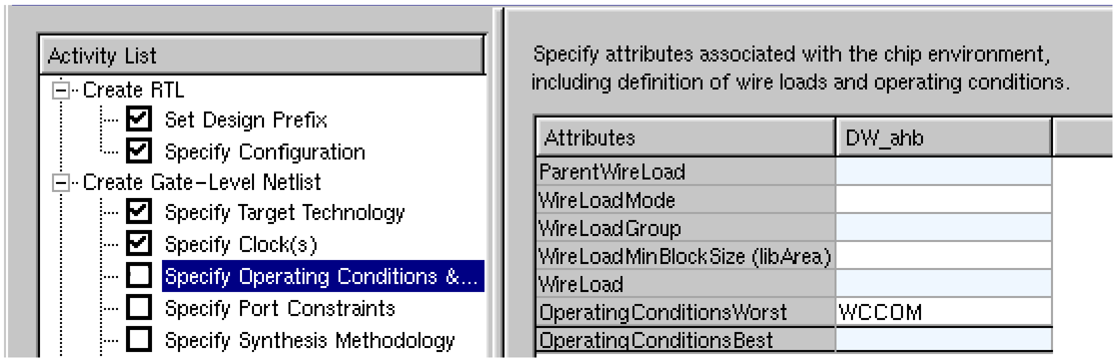
Specify Operating Conditions & WireLoads
Specify Port Constraints
Specify Synthesis Methodology

Do you want to automatically complete these activities, if possible? Processing will stop in the first activity which cannot be completed automatically.

4. **Click “Yes” to auto-complete these listed activities.**

   Depending on the Technology used, you may get an error message telling you that you need to provide a value for “OperatingConditionsWorst.”

5. **For example, choose “WCCOM” from the “OperatingConditionsWorst” field as shown below. and then click on the “Synthesize” activity again.**


Activity List
Create RTL
Set Design Prefix
Specify Configuration
Create Gate-Level Netlist
Specify Target Technology
Specify Clock(s)
Specify Operating Conditions &...
Specify Port Constraints
Specify Synthesis Methodology

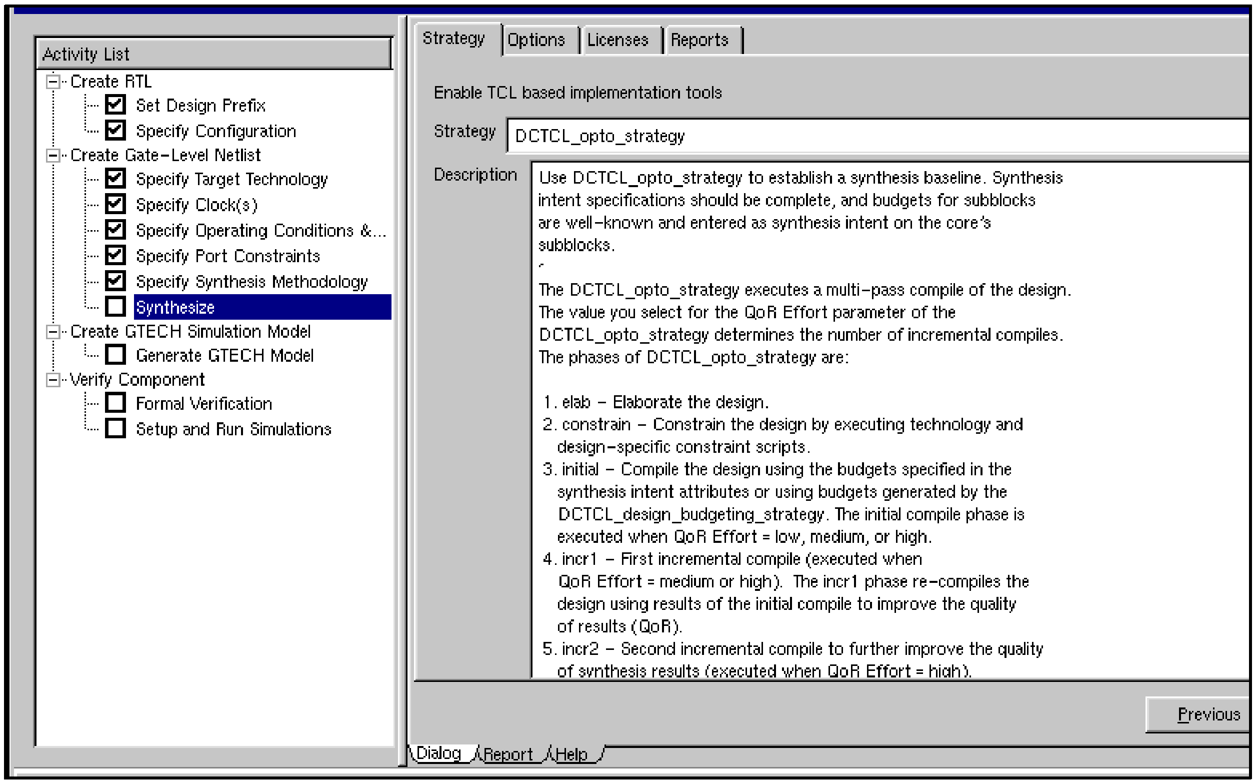
Specify attributes associated with the chip environment, including definition of wire loads and operating conditions.

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

6. **Click “Yes” for the remaining “Save activity” dialog boxes.**

Synthesizing – DW_ahb

All of the activities prior to “Synthesize” should have auto-completed and the right window should show the Synthesize dialog box with the Strategy tab revealed. The default strategy is DCTCL_opto_strategy, as shown.



- Create RTL
  - ✓ Set Design Prefix
  - ✓ Specify Configuration
- Create Gate-Level Netlist
  - ✓ Specify Target Technology
  - ✓ Specify Clock(s)
  - ✓ Specify Operating Conditions &...
  - ✓ Specify Port Constraints
  - ✓ Specify Synthesis Methodology
  - ☐ Synthesize
- Create GTECH Simulation Model
  - ☐ Generate GTECH Model
- Verify Component
  - ☐ Formal Verification
  - ☐ Setup and Run Simulations

Enable TCL based implementation tools

Strategy: DCTCL_opto_strategy

Description:

```
Use DCTCL_opto_strategy to establish a synthesis baseline. Synthesis
intent specifications should be complete, and budgets for subblocks
are well-known and entered as synthesis intent on the core's
subblocks.
.
The DCTCL_opto_strategy executes a multi-pass compile of the design.
The value you select for the QoR Effort parameter of the
DCTCL_opto_strategy determines the number of incremental compiles.
The phases of DCTCL_opto_strategy are:

 1. elab - Elaborate the design.
 2. constrain - Constrain the design by executing technology and
    design-specific constraint scripts.
 3. initial - Compile the design using the budgets specified in the
    synthesis intent attributes or using budgets generated by the
    DCTCL_design_budgeting_strategy. The initial compile phase is
    executed when QoR Effort = low, medium, or high.
 4. incr1 - First incremental compile (executed when
    QoR Effort = medium or high). The incr1 phase re-compiles the
    design using results of the initial compile to improve the quality
    of results (QoR).
 5. incr2 - Second incremental compile to further improve the quality
    of synthesis results (executed when QoR Effort = high).
```

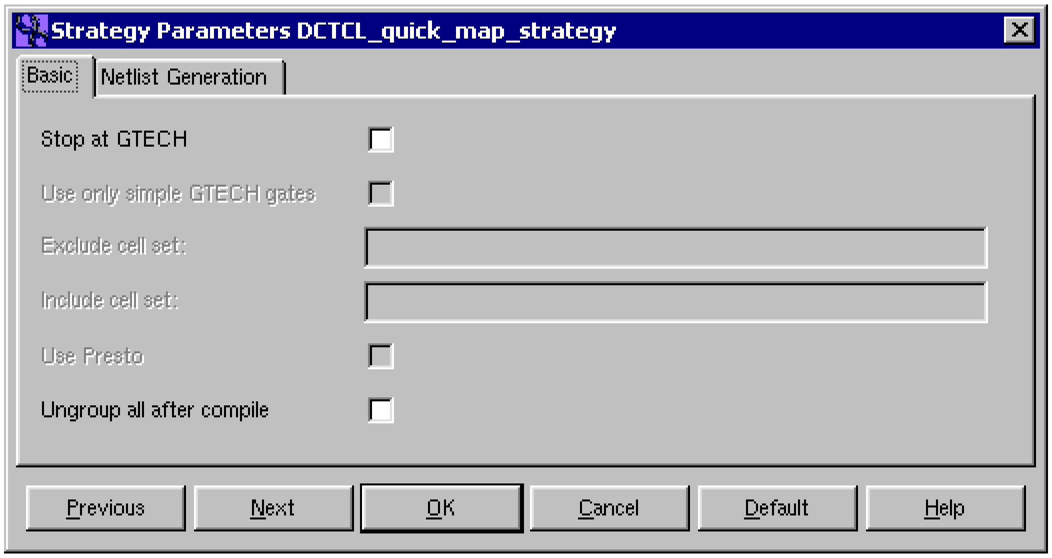
1. **Change the Strategy field to “DCTCL_quick_map_strategy”.**

   This strategy performs a quick, low-effort compile to generate a gate-level netlist of the component for analysis purposes only. Use DCTCL_quick_map_strategy to perform a quick feasibility check on a design that you will later synthesize using one of the other synthesis strategies. Do not use DCTCL_quick_map_strategy simply to reduce compile time, as it is not rigorous enough for production.

2. **Read the Description field and then click on the “Options” button to the right of the Strategy field (not the Options tab).**

   The Strategy Parameters for the DC_quick_map_strategy appear.

3. Fill out the dialog box as shown below and OK it.


Strategy Parameters DCTCL_quick_map_strategy

Basic | Netlist Generation

Stop at GTECH [ ]

Use only simple GTECH gates [ ]

Exclude cell set:

Include cell set:

Use Presto [ ]

Ungroup all after compile [ ]

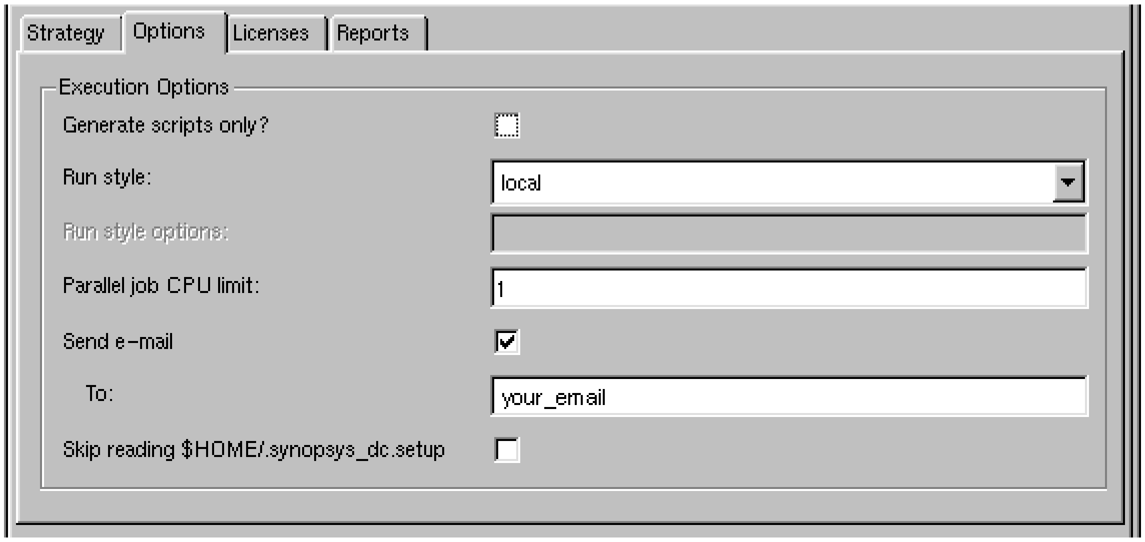
The following synthesis strategy occurs:

- coreConsultant builds only GTECH logic (but does not map to “lsi_10k” logic).
- coreConsultant uses only simple gates from the GTECH library.
- coreConsultant uses Presto.
- coreConsultant writes the final netlist for a Verilog simulator.

Performing the Synthesis Run

1. **Click on "Options" tab in the activity list.**

   The "Options" view in the dialog box appears.



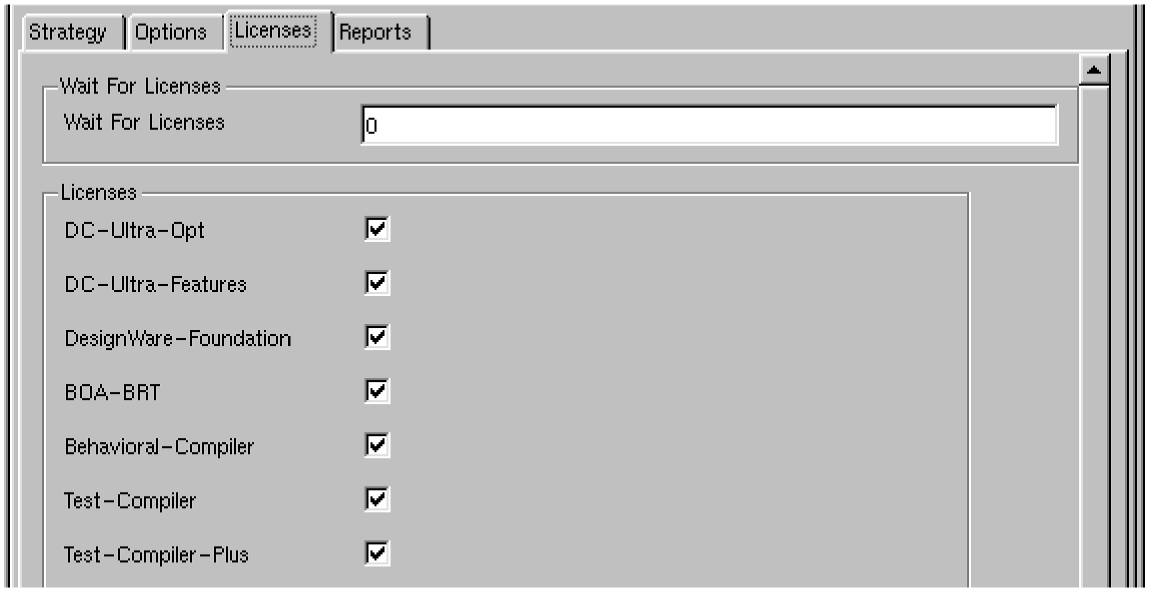
Execution Options

Generate scripts only?	☐
Run style:	local
Run style options:	
Parallel job CPU limit:	1
Send e-mail	✓
To:	your_email
Skip reading $HOME/.synopsys_dc.setup	☐

2. **Make sure that "Generate Scripts Only?" is *not* checked to actually launch the synthesis run; otherwise the scripts will not be executed.**

   If you want e-mail sent when DC completes the generation, check the "Send e-mail" box and enter your email address.

3. **Click on the "Licenses" tab to reveal the licensing detail.**



Wait For Licenses

Wait For Licenses	0

Licenses

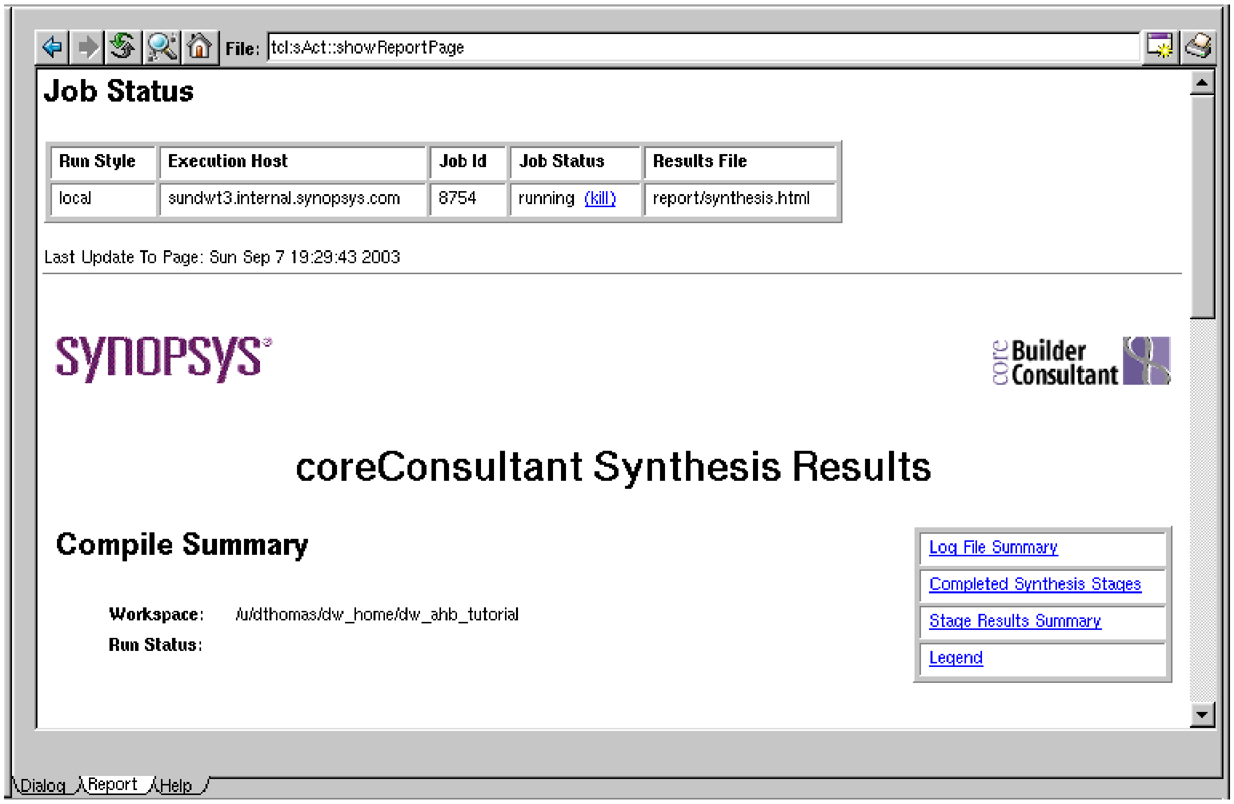
DC-Ultra-Opt	✓
DC-Ultra-Features	✓
DesignWare-Foundation	✓
BOA-BRT	✓
Behavioral-Compiler	✓
Test-Compiler	✓
Test-Compiler-Plus	✓

The Licenses boxes are checked if you have this licence feature available.

The “DesignWare-Foundation” license feature must be checked in order for synthesis to use this license; it is needed to obtain sub-components from the DesignWare library (adders, muxes, for example) and for coreConsultant.

4. Click “Apply” to use all the parameters for the Synthesize activity.

The coreConsultant tool builds the synthesis scripts, and launches a synthesis run. The “Results” are displayed including the details for this activity.



Job Status

Run Style	Execution Host	Job Id	Job Status	Results File
local	sundwt3.internal.synopsys.com	8754	running (kill)	report/synthesis.html

Last Update To Page: Sun Sep 7 19:29:43 2003

coreConsultant Synthesis Results

Compile Summary

Workspace: /u/dthomas/dw_home/dw_ahb_tutorial
Run Status:

Note

It will says “Synthesis Activity Completed” in the transcript window, but that doesn't mean the synthesis run finished, just that it’s been launched.

5. Click on the Refresh Button frequently to update this report window and to determine when the run has completed.

When the synthesis run completes, you can find the scripts and files in the “syn” directory for the DW_ahb component in your workspace.

Saving Your Configured Design

You have now completed all of the necessary steps on the DW_ahb component that initially prepares it for inclusion into your design. Before you close your work session, you should save it, if not already saved.

1. **From the coreConsultant File menu, choose "Save Workspace."**

Note

This item is not available (grayed out) because at the end of "Applying" an activity, the workspace is automatically saved for you.
If you change parameters but don't apply them (by clicking the "Apply" button or "auto-complete" to a next step), this menu item will be available for you to save your workspace.

2. **Close the workspace using the File > Close Workspace menu item.**

   This allows you to close the current workspace and exit or open another workspace.

Design Views

During this tutorial, the coreConsultant process you used created many design objects, as described in "Design/HDL Files" in the *DesignWare DW_ahb Databook*.

This completes the DesignWare DW_ahb Tutorial. You can continue learning about coreConsultant or choose the **File** > **Exit** menu item to exit.

Tutorial Summary

You have completed the tutorial procedures, and have performed the activities necessary to create, synthesize and verify an DesignWare AMBA On-chip Bus design. These activities include:

- Setting up your DesignWare AMBA On-chip Bus environment
- Opening the DW_ahb component in a new workspace in coreConsultant
- Using coreConsultant to configure the DW_ahb component
- Setting up technology information for synthesis and verification
- Creating a GTECH model of the DW_ahb for non-Synopsys simulators
- Verifying the DW_ahb configured component prior using it in your testbench
- Applying default parameters for synthesis and synthesizing the DW_ahb
- Viewing reports on configuration, verification and synthesis activities
- Writing out all the views needed to integrate the part in your design

The information in this tutorial should help you configure, synthesize and verify a single component to be used in your design.

What's Next?

Once you have configured, tested, and synthesized the design's core with the coreConsultant flow, you can integrate generated output files that are described in this chapter into your design environment. For information on the design views and files that you integrate, refer to "Integrating DW AMBA Components" in the DW_ahb Databook.

For an example showing how to integrate various IP blocks in an AMBA On-Chip Bus system, refer to the *DW_AMBA QuickStart SingleLayer Example Guide* and the *DW_AMBA QuickStart MultiLayer Example Guide*.

A
Glossary

active command queue	Command queue from which a model is currently taking commands; see also command queue.
activity	A set of functions in coreConsultant that step you through configuration, verification, and synthesis of a selected core.
AHB	Advanced High-performance Bus — high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces (ARM Limited specification).
AMBA	Advanced Microcontroller Bus Architecture — a trademarked name by ARM Limited that defines an on-chip communication standard for high speed microcontrollers.
APB	Advanced Peripheral Bus — optimized for minimal power consumption and reduced interface complexity to support peripheral functions (ARM Limited specification).
APB bridge	DW_apb submodule that converts protocol between the AHB bus and APB bus.
application design	Overall chip-level design into which a subsystem or subsystems are integrated.
arbiter	AMBA bus submodule that arbitrates bus activity between masters and slaves.
BFM	Bus-Functional Model — A simulation model used for early hardware debug. A BFM simulates the bus cycles of a device and models device pins, as well as certain on-chip functions. See also Full-Functional Model.
big-endian	Data format in which most significant byte comes first; normal order of bytes in a word.
blocked command stream	A command stream that is blocked due to a blocking command issued to that stream; see also command stream, blocking command, and non-blocking command.

blocking command	A command that prevents a testbench from advancing to next testbench statement until this command executes in model. Blocking commands typically return data to the testbench from the model.
bus bridge	Logic that handles the interface and transactions between two bus standards, such as AHB and APB. See APB bridge.
command channel	Manages command streams. Models with multiple command channels execute command streams independently of each other to provide full-duplex mode function.
command stream	The communication channel between the testbench and the model.
component	A generic term that can refer to any synthesizable IP or verification IP in the DesignWare Library. In the context of synthesizable IP, this is a configurable block that can be instantiated as a single entity (VHDL) or module (Verilog) in a design.
configuration	The act of specifying parameters for a core prior to synthesis; can also be used in the context of VIP.
configuration intent	Range of values allowed for each parameter associated with a reusable core.
core	Any configurable block of synthesizable IP that can be instantiated as a single entity (VHDL) or module (Verilog) in a design. Core is the preferred term for a big piece of IIP. Anything that requires coreConsultant for configuration, as well as anything in the DesignWare Cores library, is a core.
core developer	Person or company who creates or packages a reusable core. All the cores in the DesignWare Library are developed by Synopsys.
core integrator	Person who uses coreConsultant or coreAssembler to incorporate reusable cores into a system-level design.
coreAssembler	Synopsys product that enables automatic connection of a group of cores into a subsystem. Generates RTL and gate-level views of the entire subsystem.
coreConsultant	A Synopsys product that lets you configure a core and generate the design views and synthesis views you need to integrate the core into your design. Can also synthesize the core and run the unit-level testbench supplied with the core.
coreKit	An unconfigured core and associated files, including the core itself, a specified synthesis methodology, interfaces definitions, and optional items such as verification environment files and core-specific documentation.
cycle command	A command that executes and causes HDL simulation time to advance.
decoder	Software or hardware subsystem that translates from and “encoded” format back to standard format.
design context	Aspects of a component or subsystem target environment that affect the synthesis of the component or subsystem.
design creation	The process of capturing a design as parameterized RTL.
Design View	A simulation model for a core generated by coreConsultant.

DesignWare AMBA On-chip Bus	The Synopsys name for the collection of AMBA-compliant coreKits and verification models delivered with DesignWare and used with coreConsultant or coreAssembler to quickly build DesignWare AMBA On-chip Bus designs.
DesignWare cores	A specific collection of synthesizable cores that are licensed individually. Products include Blue IQ and all former inSilicon cores.
DesignWare Library	A collection of synthesizable IP and verification IP components that is authorized by a single DesignWare license. Products include SmartModels, VMT model suites, DesignWare Memory Models, Building Block IP, and the DesignWare AMBA OCB synthesizable components.
dual role device	Device having the capabilities of function and host (limited).
endian	Ordering of bytes in a multi-byte word; see also little-endian and big-endian.
Full-Functional Mode	A simulation model that describes the complete range of device behavior, including code execution. See also BFM.
GPIO	General Purpose Input Output.
GTECH	A generic technology view used for RTL simulation of encrypted source code by non-Synopsys simulators.
hard IP	Non-synthesizable implementation IP.
HDL	Hardware Description Language – examples include Verilog and VHDL.
HNP	Host Negotiation Protocol.
IIP	Implementation Intellectual Property — A generic term for synthesizable HDL and non-synthesizable "hard" IP in all of its forms (coreKit, component, core, MacroCell, and so on).
implementation view	The RTL for a core. You can simulate, synthesize, and implement this view of a core in a real chip.
instantiate	The act of placing a core or model into a design.
interface	Set of ports and parameters that defines a connection point to a component.
IP	Intellectual property — A term that encompasses simulation models and synthesizable blocks of HDL code.
little-endian	Data format in which the least-significant byte comes first.
MacroCell	Bigger IP blocks (6811, 8051, memory controller) available in the DesignWare Library and delivered with coreConsultant.
master	Device or model that initiates and controls another device or peripheral.
model	A Verification IP component or a Design View of a core.
monitor	A device or model that gathers performance statistics of a system.

non-blocking command	A testbench command that advances to the next testbench statement without waiting for the command to complete.
non-OTG device	The Device model configured as USB 2.0 standard function and does not contain OTG features.
OTG	On-The-Go.
peripheral	Generally refers to a small core that has a bus connection, specifically an APB interface.
RTL	Register Transfer Level. A higher level of abstraction that implies a certain gate-level structure. Synthesis of RTL code yields a gate-level design.
SDRAM	Synchronous Dynamic Random Access Memory; high-speed DRAM adds a separate clock signal to control signals.
SDRAM controller	A memory controller with specific connections for SDRAMs.
slave	Device or model that is controlled by and responds to a master.
SoC	System on a chip.
soft IP	Any implementation IP that is configurable. Generally referred to as synthesizable IP.
SRP	Session Request Protocol.
static controller	Memory controller with specific connections for Static memories such as asynchronous SRAMs, Flash memory, and ROMs.
subsystem	In relation to coreAssembler, highest level of RTL that is automatically generated.
synthesis intent	Attributes that a core developer applies to a top-level design, ports, and core.
synthesizable IP	A type of Implementation IP that can be mapped to a target technology through synthesis. Sometimes referred to as Soft IP.
technology-independent	Design that allows the technology (that is, the library that implements the gate and via widths for gates) to be specified later during synthesis.
Testsuite Regression Environment (TRE)	A collection of files for stand-alone verification of the configured component. The files, tests, and functionality vary from component to component.
VIP	Verification Intellectual Property — A generic term for a simulation model in any form, including a Design View.
workspace	A network location that contains a personal copy of a component or subsystem. After you configure the component or subsystem (using coreConsultant or coreAssembler), the workspace contains the configured component/subsystem and generated views needed for integration of the component/subsystem at the top level.

wrap, wrapper	Code, usually VHDL or Verilog, that surrounds a design or model, allowing easier interfacing. Usually requires an extra, sometimes automated, step to create the wrapper.
zero-cycle command	A command that executes without HDL simulation time advancing.