Using DesignChecker with HDL Designer Series

Software Version 2010.3

June, 2011

Contractor/manufacturer is:
Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/
Send Feedback on Documentation: supportnet.mentor.com/doc_feedback_form

Table of Contents

List of Tables

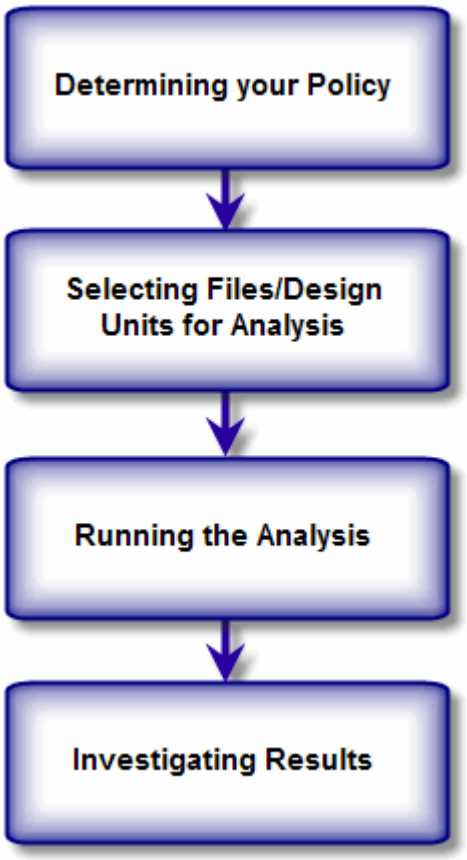
Chapter 1
Introduction

Design Checker

HDL Designer Series integrates with DesignChecker which is a design checking tool that ensures interoperability and consistency across design styles and coding practices, for developers and development teams creating complex designs. It is a powerful solution for individual engineers, and can be used in situations where a number of individuals and/or teams are involved in the design and development processes.

DesignChecker can operate in batch mode, or with a full featured Graphical User Interface (GUI). The user interface supports quick and easy rule configuration, with powerful results cross-referencing between DesignChecker, the DesignPad text editor and any of the HDL Designer graphical editing tools.

The basic flow of DesignChecker involves configuring and running checks on your design code. The flow starts by ensuring that you have access to the appropriate ruleset(s) and policies which have been previously configured by the project manager to match your coding standards. To run an analysis, you have to specify the policy you intend to use, and then select the files/design units to be analyzed. After running the analysis and viewing the results, you can cross-reference your results to the source code to view the exact lines of code where violations occur, or you can cross-reference your results to the source graphical views. Note that before running an analysis, you have the option to set exclusions on specific files or design units if needed.



In the following sections, it is assumed that you are familiar with the basic operation of DesignChecker and HDL Designer Series. For information, refer to *DesignChecker User Guide* and *HDL Designer Series User Manual*.

Chapter 2
Invoking DesignChecker

DesignChecker can be invoked in several ways from the Design Manager of HDL Designer Series. DesignChecker can be invoked for two reasons:

1. **To Manage Rules and Policies:** DesignChecker enables the user to create the desired rulesets and policies that are to be applied to files/ design units before running an analysis. It is possible to manage rules and polices by invoking DesignChecker directly from HDL Designer Series Design Manager.
2. **To Run an Analysis:** DesignChecker integrates with HDL Designer Series, such that it is possible to run an analysis using DesignChecker directly through HDL Designer Series Design Manager.

Managing RuleSets and Policies

You can invoke DesignChecker through HDL Designer Series for the purpose of configuring rulesets and policies. You can do that through:

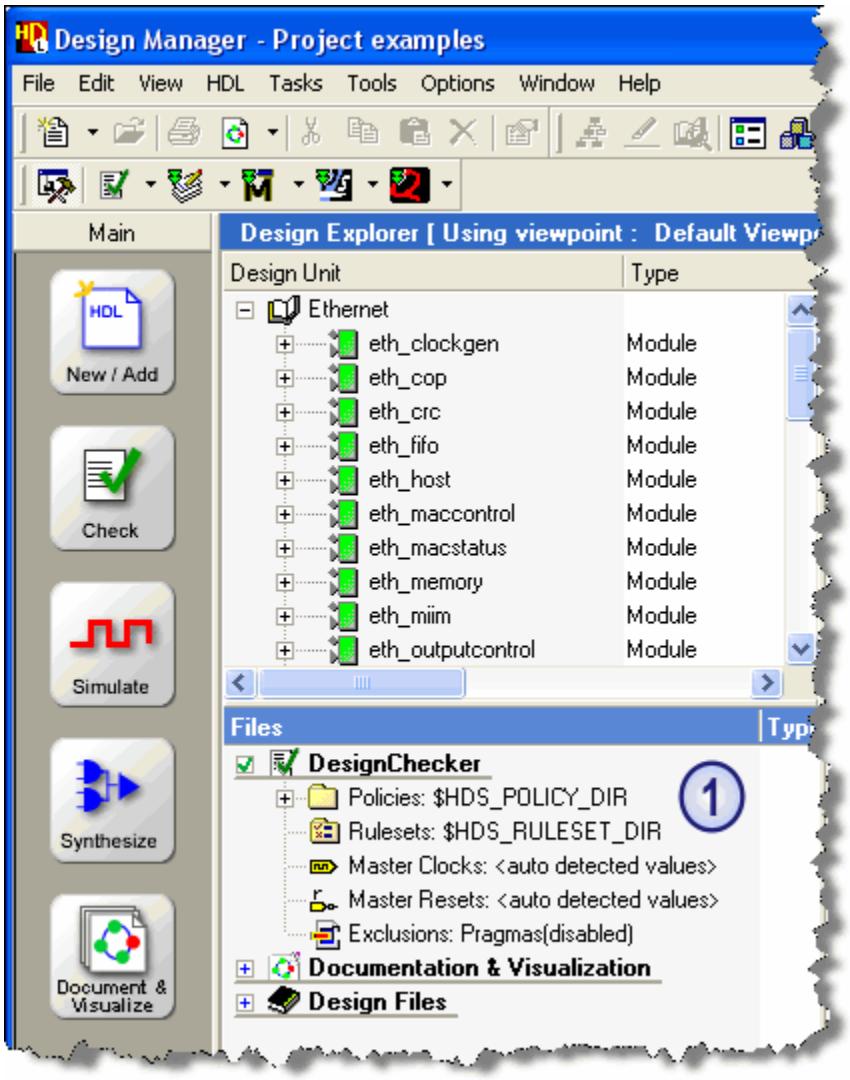
- The Files Pane
- The Tools Menu

Note

The above is applicable to Windows users. Unix and Linux users should first choose the appropriate script from the *bin* directory of their HDL Designer Series installation to invoke the HDL Designer Series tool, that is, the script titled *hdl_designer*. Subsequently, DesignChecker can be run from the HDL Designer Series tool in exactly the same way as Windows users (that is through the Files pane or Tools menu).

The Files Pane

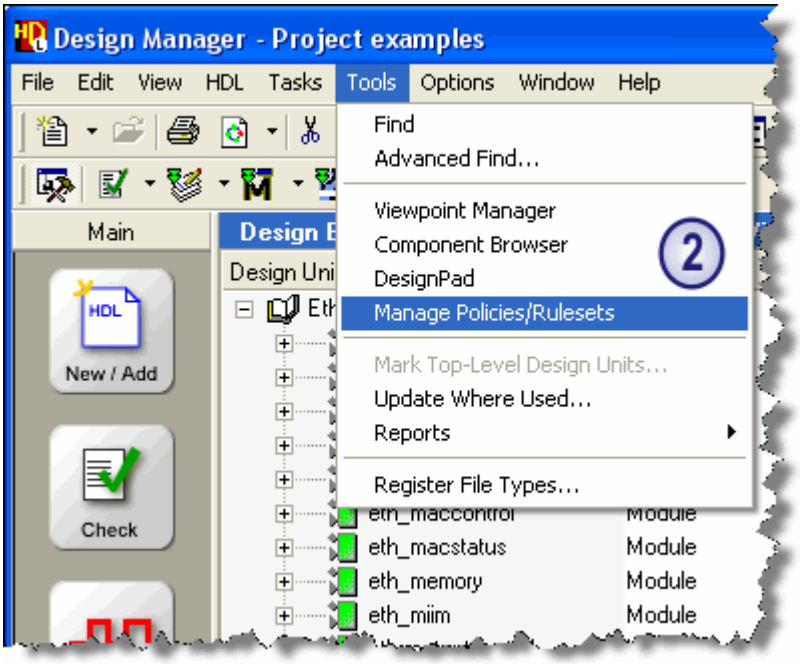
In the Files pane, you can use the ⊞ sign to expand the DesignChecker tree. You can right-click on the Policies or Rulesets nodes and select **Manage Policies/Rulesets** from the popup menu so that DesignChecker would be invoked.



Note that you can also perform a number of DesignChecker operations through the Files pane: you can set a shared location for policies or rulesets (refer to "DesignChecker for Teams" on page 55), you can manually specify master clocks and resets, or you can enable/disable pragma exclusions and manually edit the constraints file if needed (refer to "DesignChecker Exclusions" on page 27).

The Tools Menu

You can invoke DesignChecker by selecting **Tools > Manage Policies/Rulesets**.



Running an Analysis

To run an analysis through HDL Designer Series, you must first set the default policy and select the desired file/design unit for analysis. For more information on how to run the analysis directly through HDL Designer Series, refer to "Running the Analysis" on page 15.

Chapter 3
The DesignChecker Flow

This chapter describes how you can validate the correctness of the designs available in HDL Designer Series by using DesignChecker.

Before using DesignChecker to validate your code, make sure you have a design in HDL Designer Series, whether an already existing design that you have imported in HDL Designer Series or a new design that you have created. Refer to *HDL Designer Series User Manual* for more information.

The basic DesignChecker flow is as follows:

1. Determining your Policy
2. Selecting Files/Design Units for Analysis
3. Running the Analysis
4. Investigating Results

Note

Before selecting files/design units and running an analysis, you have the ability to set exclusions. The exclusions feature allows you to exclude specific code blocks or entire files from analysis, to exclude certain rules/rulesets from being applied to specific design units or files, and so on. There are various types of DesignChecker exclusions that you can use with HDL Designer Series. For more information on exclusions, refer to "DesignChecker Exclusions" on page 27.

Determining your Policy

To run an analysis using Design checker, you must first set a default policy. You may have multiple policies defined in DesignChecker each referencing a different ruleset, so you have to set a default policy to use in the analysis.

Prerequisites

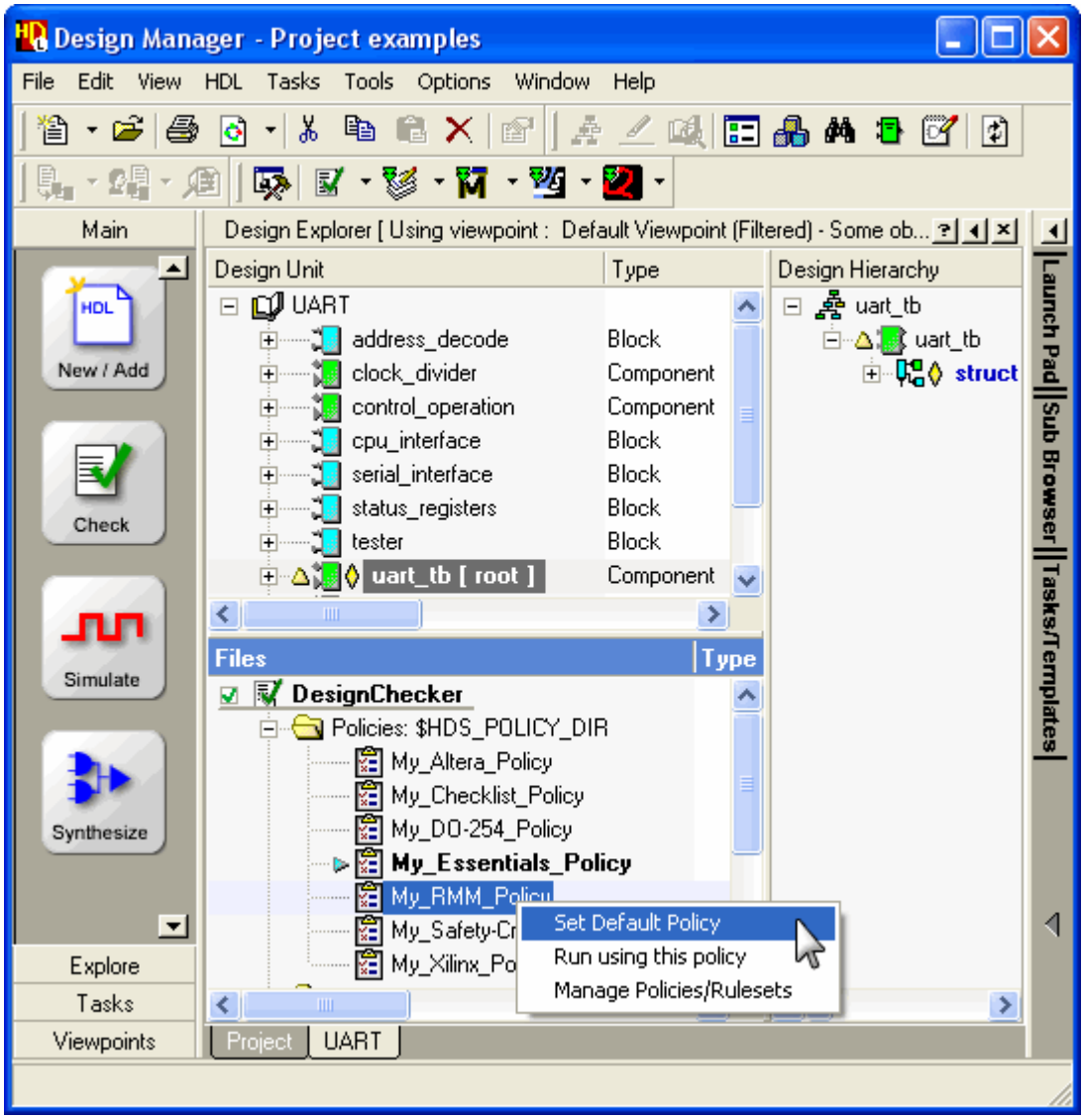
In order to set a default policy, you must first configure your desired rulesets and policies. Refer to "Working with RuleSets" and "Working with Policies" in the *DesignChecker User Guide* for information.

Procedure

You can set a default policy either through HDL Designer Series or through DesignChecker:

Through HDL Designer Series:

1. In the Files pane, click on the ⊞ sign to expand the DesignChecker tree.
2. Expand the *Policies* folder; consequently, all policies predefined through DesignChecker are displayed.



3. Right-click on the required policy and then select **Set Default Policy**.

Through DesignChecker:

Refer to the *DesignChecker User Guide* for information on setting a default policy through DesignChecker.

Results

The selected policy is set as the default policy; it will be indicated in both HDL Designer Series and DesignChecker by the ▶marker.

Selecting Files/Design Units for Analysis

Before running an analysis, use the design explorer in HDL Designer Series to select files, design units, or an entire library for analysis.

File Level and Design Unit Level Checking

Note that when running a DesignChecker analysis, you have the ability to do that on one of two levels: File level or Design Unit level. Determining which level to use in your analysis depends on the level of depth you are targeting which is consequently reflected on the analysis results.

If you run DesignChecker on the level of a source file, the tool analyzes the selected file along with all the design units that pertain to that file. On the other hand, if you run DesignChecker on the level of a design unit, the tool analyzes the specified design unit individually. Refer to "Design Checking Levels" on page 65 for details on file level and design unit level analysis.

You can select design units through the Design Units pane of the design explorer, and you can select files through the Files pane.

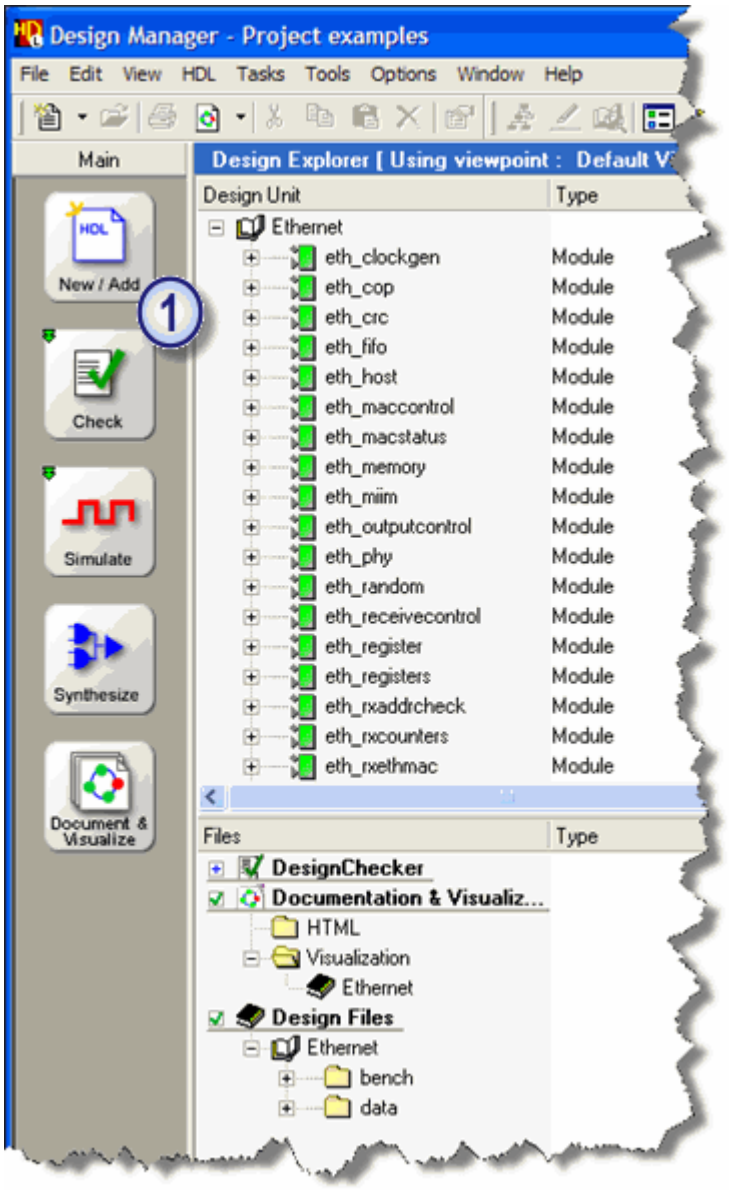
Running the Analysis

When running an analysis using DesignChecker, you need to first determine the desired files/design units for analysis. You also have to determine the hierarchical depth of the analysis by selecting the desired run options. This is done through HDL Designer Series from:

- The Main Shortcut Bar
- The Main Menu
- The Tasks Pane
- The Design Manager Toolbar

The Main Shortcut Bar

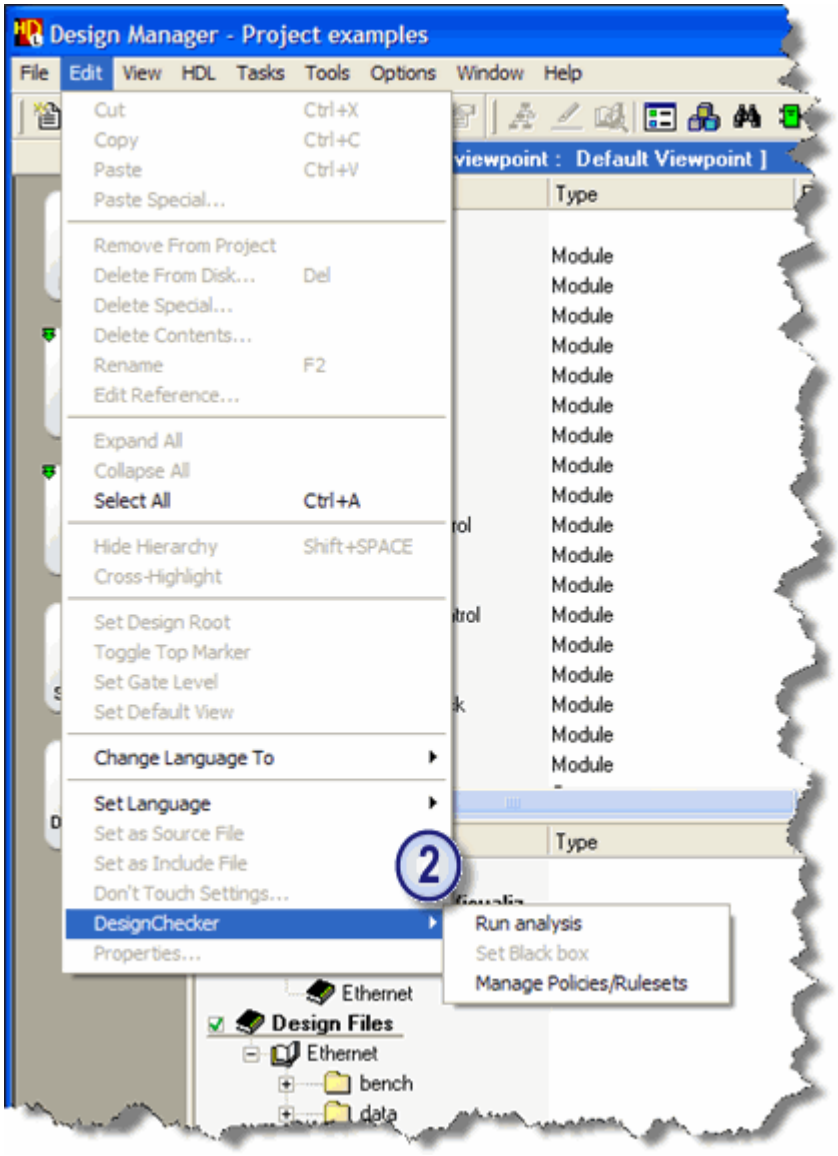
HDL Designer Series has a Main Shortcut bar in the Design Manager, from which many tasks can be performed. It is possible to run a DesignChecker analysis on the selected file/design unit from the Main Shortcut bar by clicking the **Check** button.



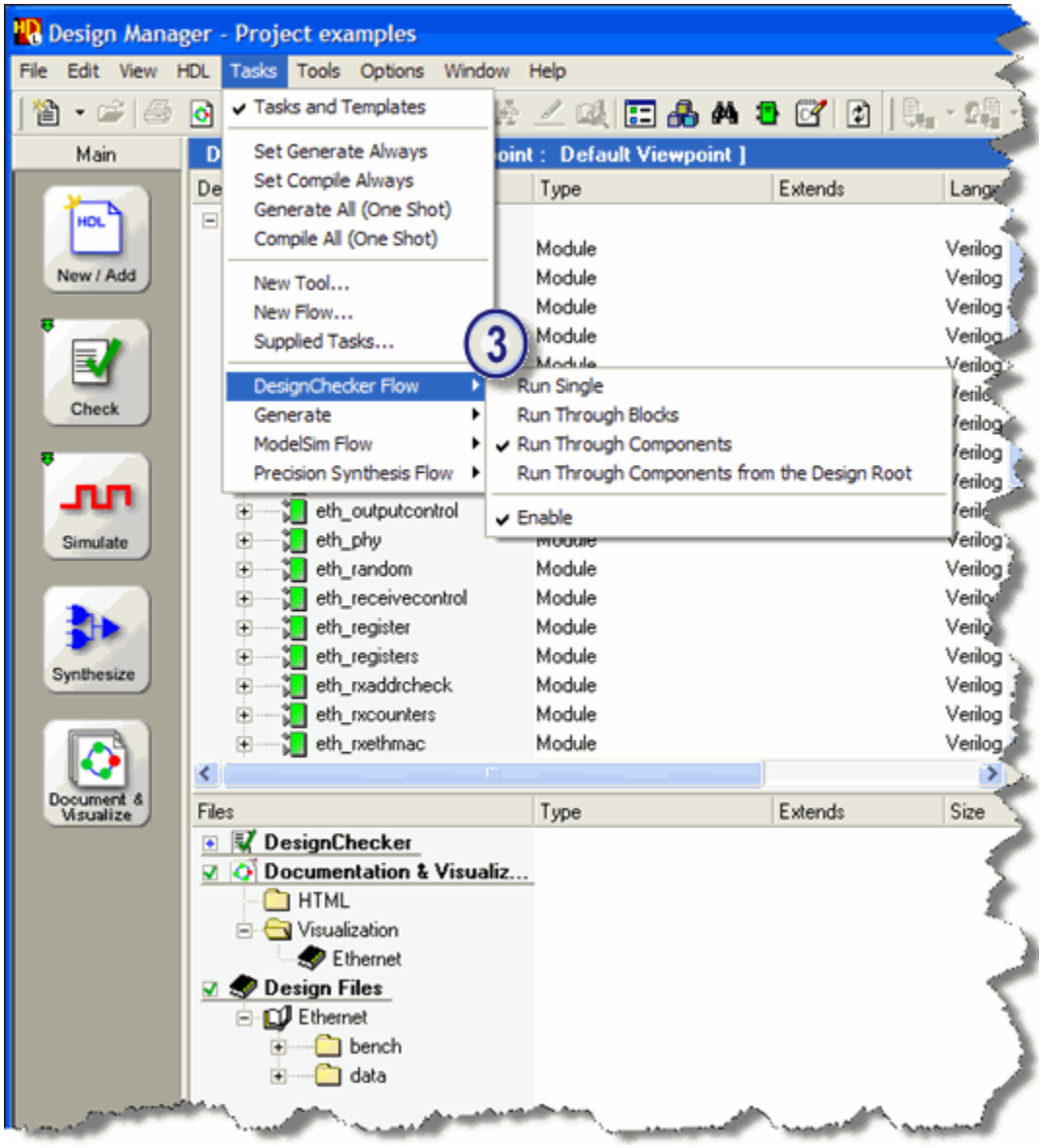
You can right-click on the **Check** button and select whether you want to **Run Single**, **Run Through Blocks**, **Run Through Components**, or **Run Through Design Root**.

The Main Menu

It is possible to run a DesignChecker analysis on the selected file/design unit by choosing **Edit > DesignChecker > Run Analysis**.



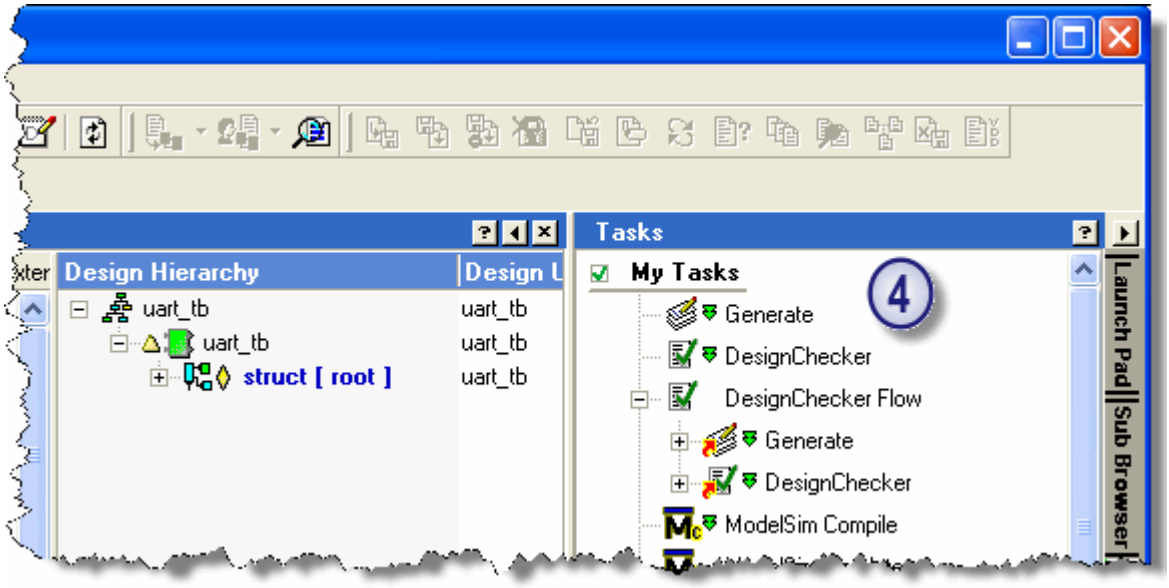
It is also possible to run an analysis by selecting **Tasks > DesignChecker Flow** and then selecting **Run Single**, **Run Through Blocks**, **Run Through Components**, or **Run Through Components from Design Root**. This can also be achieved through The Tasks Pane.



Tip: You can also run DesignChecker as a task from within DesignPad or an appropriate graphical editor.

The Tasks Pane

The DesignChecker tasks in the **Tasks** pane allow the user to run an analysis on a selected file/design unit.

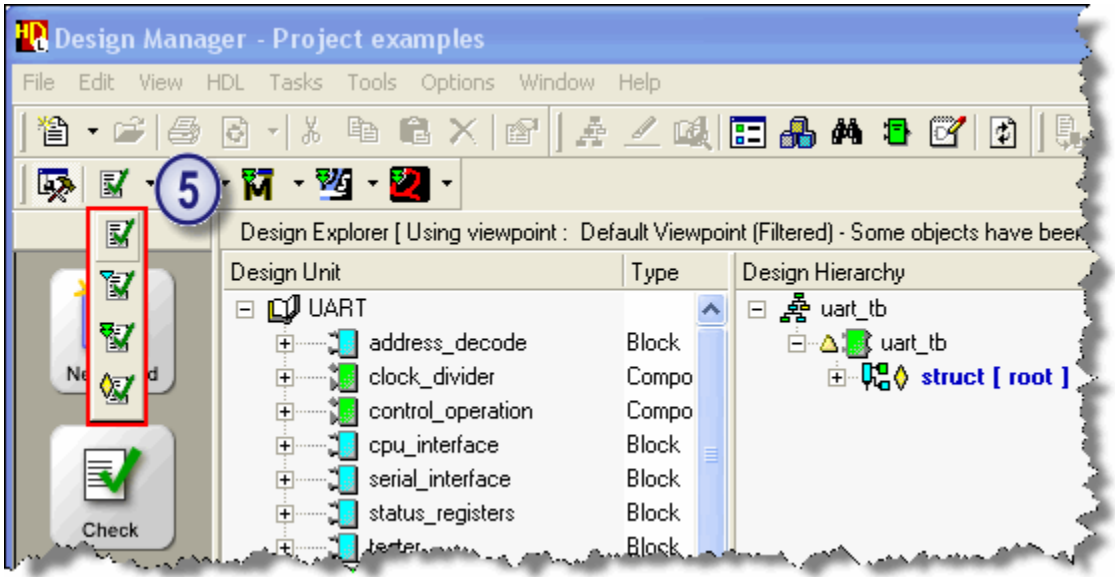


The DesignChecker tool and flow tasks are available in the HDL Designer Series design manager. The “DesignChecker Flow” automatically generates any graphical views (also available from the toolbar or menu). The “DesignChecker” tool task is available in the Tasks pane if you want to run an analysis without generating HDL.

You can optionally incorporate the DesignChecker task as a step in a new or existing design flow. Refer to the “Tasks, Tools and Flows” chapter in the *HDL Designer Series User Manual* for more information.

The Design Manager Toolbar

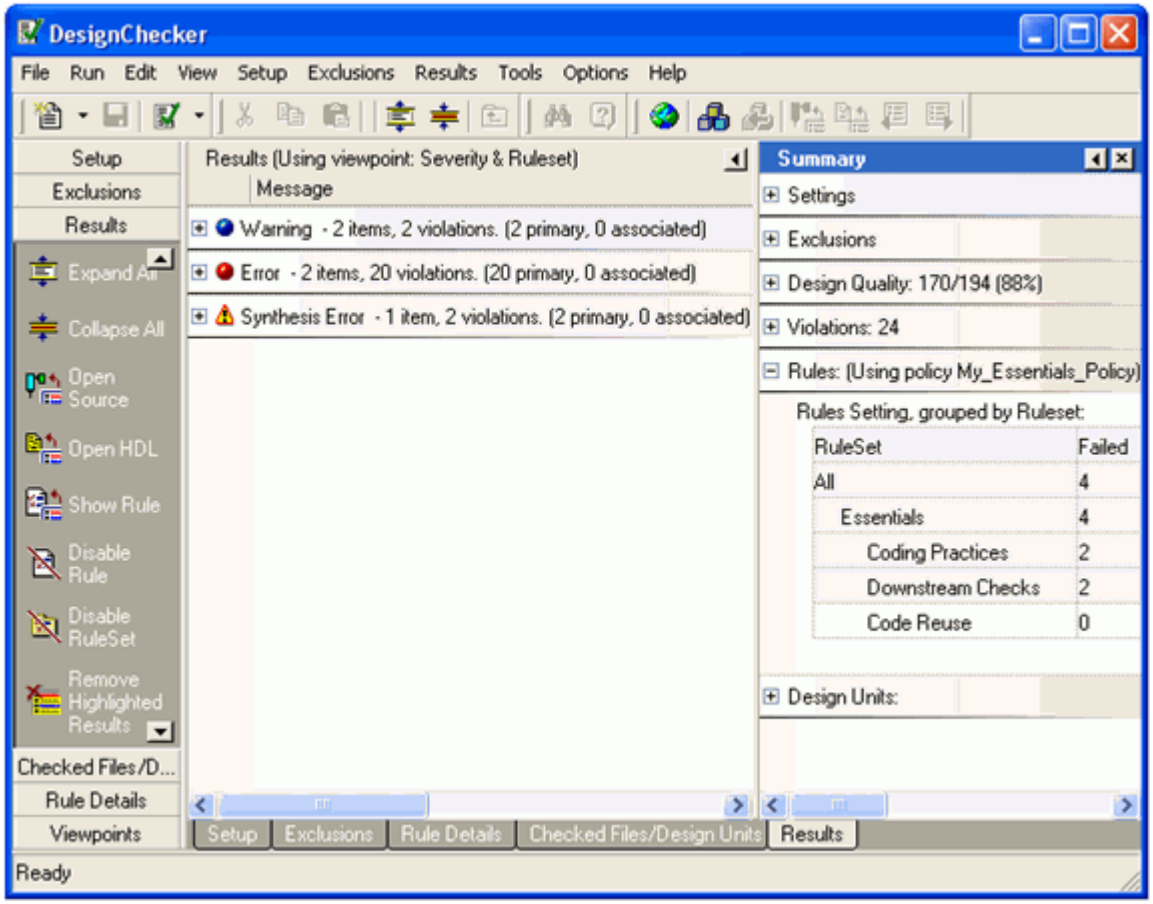
The DesignChecker drop-down palette in the Design Manager toolbar allows the user to run an analysis on a selected file/design unit.



You can choose to generate and run DesignChecker either on a single level, through blocks, through components, or through design root.

Investigating Results

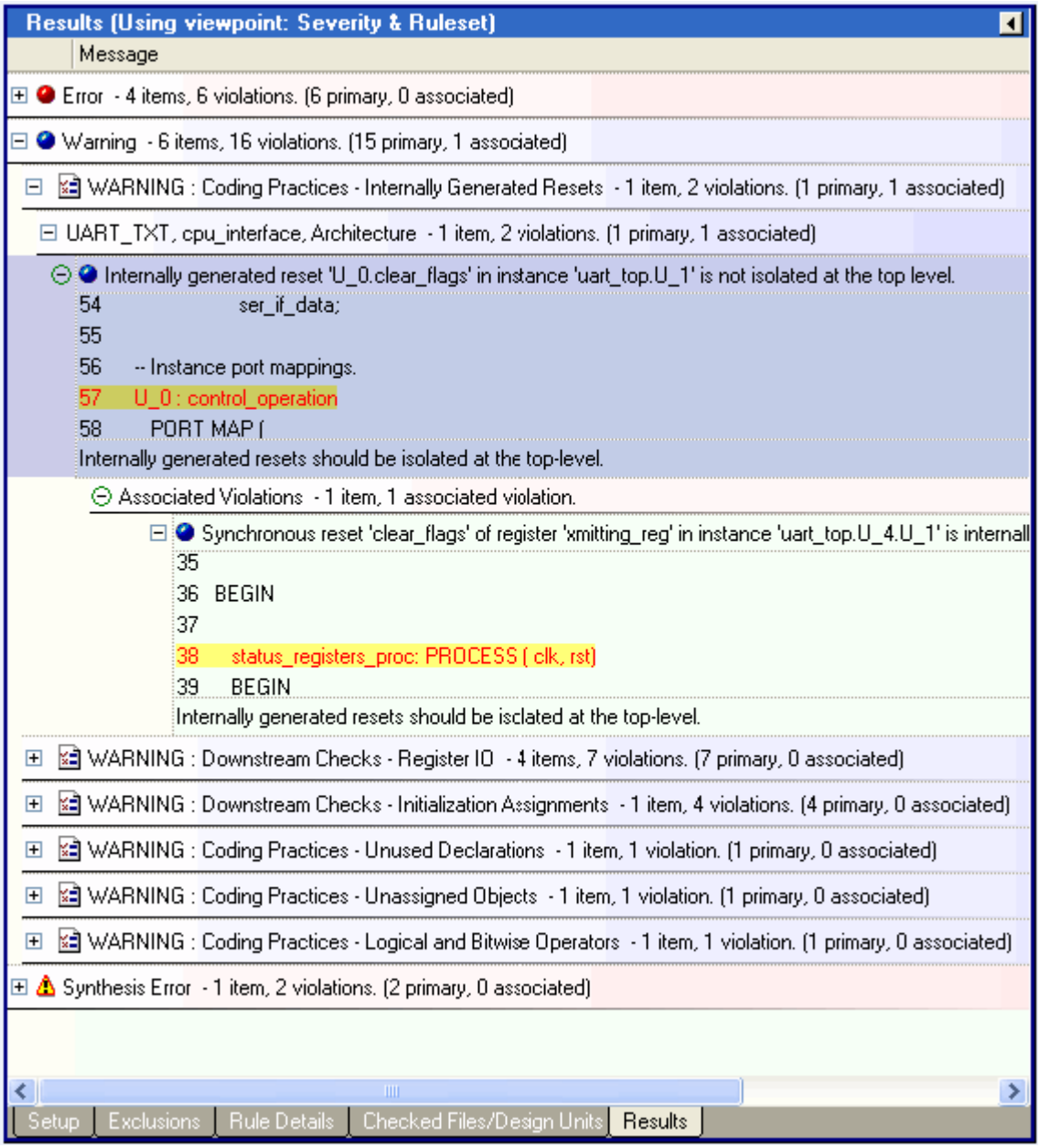
Once you run an analysis through HDL Designer Series, DesignChecker is invoked with the Results tab opened and all the detected violations are displayed. Refer to *DesignChecker User Guide* for more information on the Results tab.



The Results tab is divided into a central results pane in which all the violations are listed, and a summary pane in which you will find high level information about the analysis.

By default, the Results tab shows the results grouped by severity, and then within each severity, the results are grouped by the violated rule (you can control the display of results by changing viewpoints; refer to the *DesignChecker User Guide* for more information).

You can use the plus sign to keep traversing down from the severity group until you reach the violation message and the code snippet highlighting the exact line of code having that violation.



As shown in the figure, each violation is given a certain severity level depending on the prior configurations of the ruleset. For example, if the ruleset is using the severity set titled *Default SeveritySet*, the violations will have the severity levels *Error*, *Note* or *Warning*, and if the ruleset is using the severity set titled *RMM_SeveritySet*, the violations will have the severity levels *Rule* or *Guideline*. Refer to “Configuring Rule SeveritySets” in the *DesignChecker User Guide* for further information.

In addition to the current severity set used in your analysis, you may find some violations having different titles such as *Syntax Error*, *Synthesis Error* or *Elaboration Error*. These types of violations can occur when running certain rules on a design that is not complete or synthesizable. For more information, refer to "Supporting Synthesizable Designs" and "Further Understanding Design Checking Rule Behavior" in the *DesignChecker User Guide*.

Cross-Referencing Results with DesignPad

HDL Designer Series includes the DesignPad text editor. You can directly cross-reference between your DesignChecker results and the DesignPad text editor.

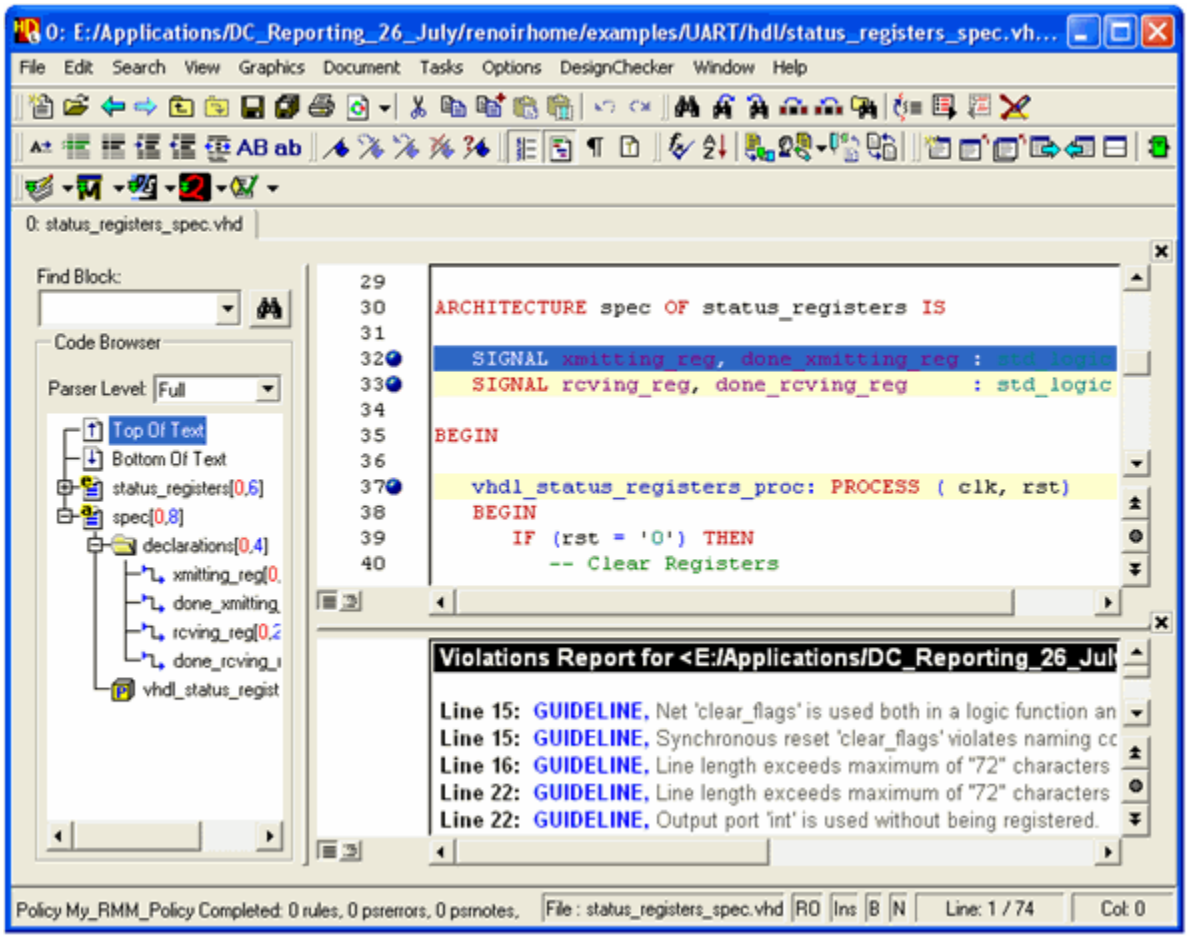
DesignPad is a fully featured text editor. It supports normal text editing procedures including file creation, read-only or edit modes, print and HTML export. Advanced features include, for example, multiple undo or redo operations, indenting, case changing and automatic keyword completion.

Language support includes automatic syntax highlighting, checking and comment insertion for several languages such as C, C++, Tcl, Verilog, VHDL and XML. In addition, language templates are provided for fast insertion and expansion of Verilog or VHDL code constructs.

You can find detailed information about using the DesignPad text editor in the *DesignPad Text Editor User Guide*, which you can access from the InfoHub of HDL Designer Series by selecting **Help and Manuals** from the **Help** menu.

Procedure

1. Right-click on a code snippet in the Results tab and select **Open HDL** from the popup menu. This will launch the DesignPad text editor.

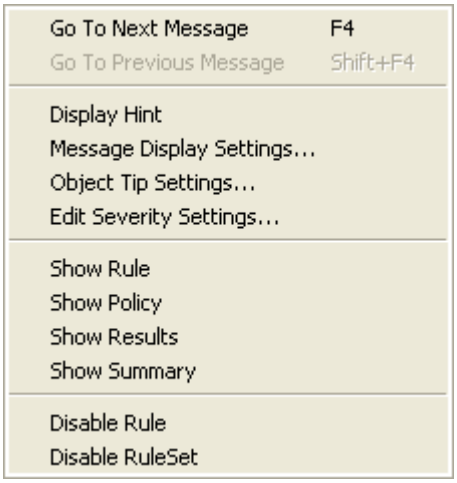


   Notice that the DesignPad window is split into three areas:

   - A Code Browser. This displays a hierarchical view of the code structure in the active file. Please refer to the *DesignPad Text Editor User Guide* for further information.
   - A Code Editing pane. This displays the source code, with the lines containing violations highlighted.
   - A Report pane which displays the violations DesignChecker has detected.

2. Move around the code by clicking on the different violations highlighted in the Report pane. The Code Editing pane automatically changes to display the appropriate line of code, and the Code Browser updates itself to display the correct location in your file structure.

DesignPad contains a **DesignChecker** menu. This contains commands specific to DesignChecker.



- You can display the correction hint message corresponding to a highlighted line of code by choosing **Display Hint**.
- You can display the configured rule or policy corresponding to a highlighted line of code in the DesignChecker Setup tab by choosing **Show Rule** or **Show Policy**.
- You can display the expanded results row corresponding to a highlighted line of code in the DesignChecker Results tab by choosing **Show Results** from the **DesignChecker** menu.
- You can display a message including summary information on the analysis by choosing **Show Summary** from the **DesignChecker** menu.
- You can disable the rule or ruleset applied to a highlighted line of code in the DesignChecker Setup tab by choosing **Disable Rule** or **Disable RuleSet** from the **DesignChecker** menu.

3. Move to the next rule violation message in DesignPad by using the  button or by choosing **Go To Next Message** from the **DesignChecker** menu.

   Similarly you can move to the previous rule violation message in DesignPad by using the  button or by choosing **Go To Previous Message** from the **DesignChecker** or the **Search** menu.

Note

You will also have the DesignChecker task available within the **Tasks** menu.

4. Investigate the other DesignChecker menu options. When you have finished you can shut down DesignPad by choosing **Exit** from the **File** menu.

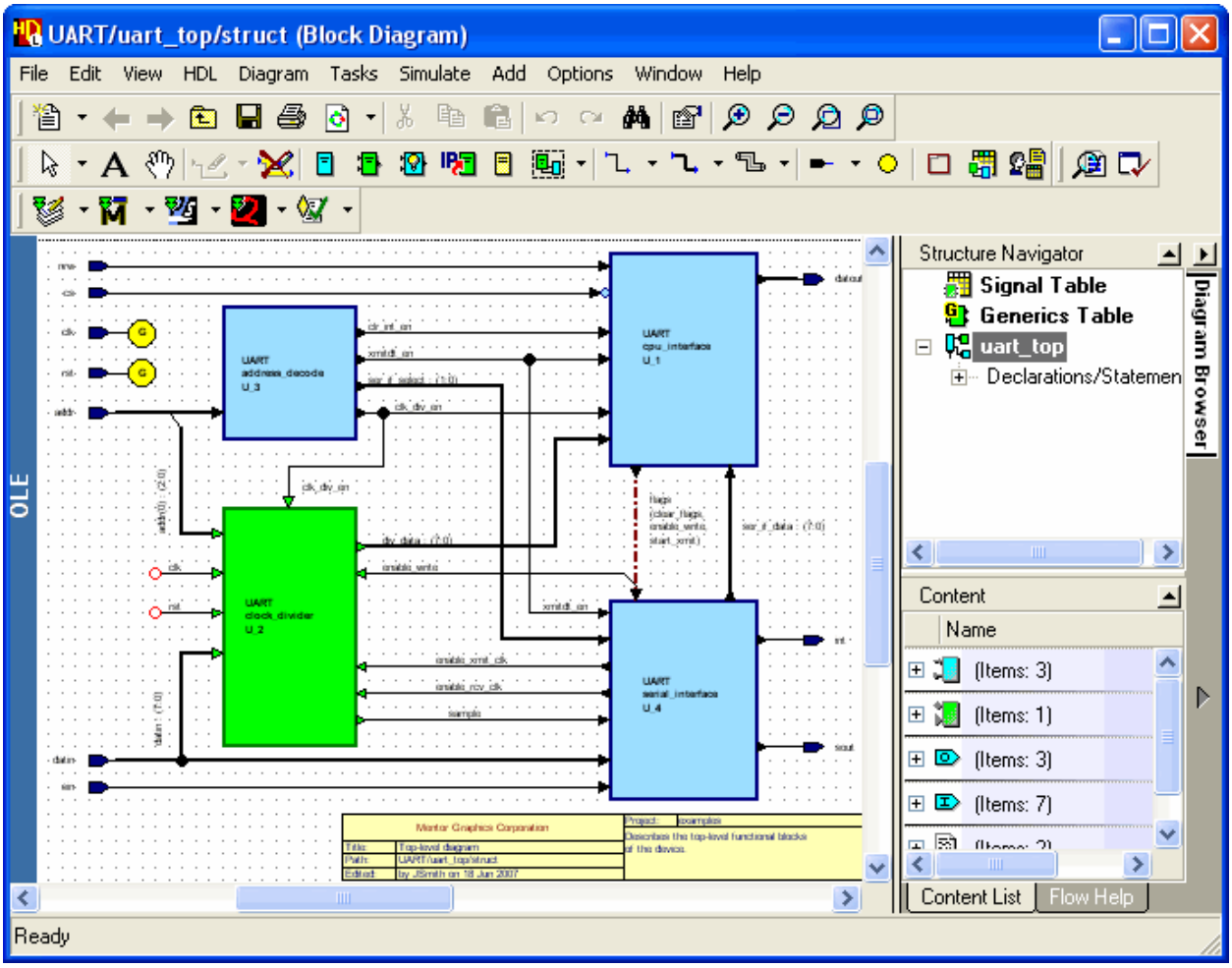
Cross-Referencing Results with Graphics

You can cross-reference from a rule violation in DesignChecker to a graphical source view created in any of the HDL Designer Series graphical editors.

Procedure

1. Right-click on a code snippet in the Results tab of DesignChecker and then choose **Open Source** from the popup menu.

   This invokes the graphical editor from which the code was generated.



2. Investigate the graphical view and then select **File > Exit**.

Investigating the Results Summary Pane

The Summary pane provides high level information on the results of the analysis. For more information on the results summary pane, refer to *DesignChecker User Guide*.

Chapter 4
DesignChecker Exclusions

Before running an analysis, you can use the exclusions feature which enables you to exclude certain areas of your code from being analyzed or more specifically exclude certain rules or rulesets from being run on specific design units or files within your code.

Excluding certain areas of your code from being generally checked, or more specifically from being checked by certain rules, saves you the hassle of having to go through the violations that may be generated by this code, which could be in fact behaving as designed or could be third-party code which does not comply with your coding standards.

Note that on running an analysis, an Exclusions tab is displayed showing all the exclusion settings applied in the analysis.

Setting Exclusions

DesignChecker has the following types of exclusions that can be used with HDL Designer Series:

Table 4-1. Exclusion Types

Exclusion Type	Description
Code/Rule Exclusion	Excludes specific checks from being performed on specific parts of the design.
Pragma Exclusion	Allows you to skip specific RTL code blocks.
Black Box Exclusion	Ensures that DesignChecker recognizes that a component is present, but does not apply any checks to it.
Don’t Touch exclusion	Ensures that a file is not loaded or analyzed by DesignChecker.

Setting Code/Rule Exclusions

Exclusion via Code/Rule exclusions enables you to force DesignChecker to skip selected design units or files when running a specific rule or ruleset. You can set your Code/Rule exclusions visually through the DesignChecker Add Code/Rule Exclusions dialog box or you can set them manually by editing the *dc_constraints* file. The *dc_constraints* file can be opened from the HDS Design Explorer Files’ pane.

Caution

It is strongly recommended not to edit the *dc_constraints* file when running DesignChecker. Also, attempting to edit the *dc_constraints* file while in the process of adding code/rule exclusions visually may lead to the loss of your manual edits.

Add Code/Rule Exclusions Dialog Box

The Add Code/Rule Exclusions dialog box enables you to define code/rule exclusions for libraries on which an analysis was previously run. That is, you can exclude specific checks from being performed on specific parts of the design on the next DesignChecker run.

Accessing the Dialog Box

Do one of the following:

- From the DesignChecker menu, choose **Exclusions > Add Code/Rule Exclusion**.
- In the Code/Rule Exclusions pane of the Exclusions tab, right-click and choose **Add Code/Rule Exclusion**.
- Select Exclusions in the shortcut bar on the left and click **Add Code/Rule Exclusion**.

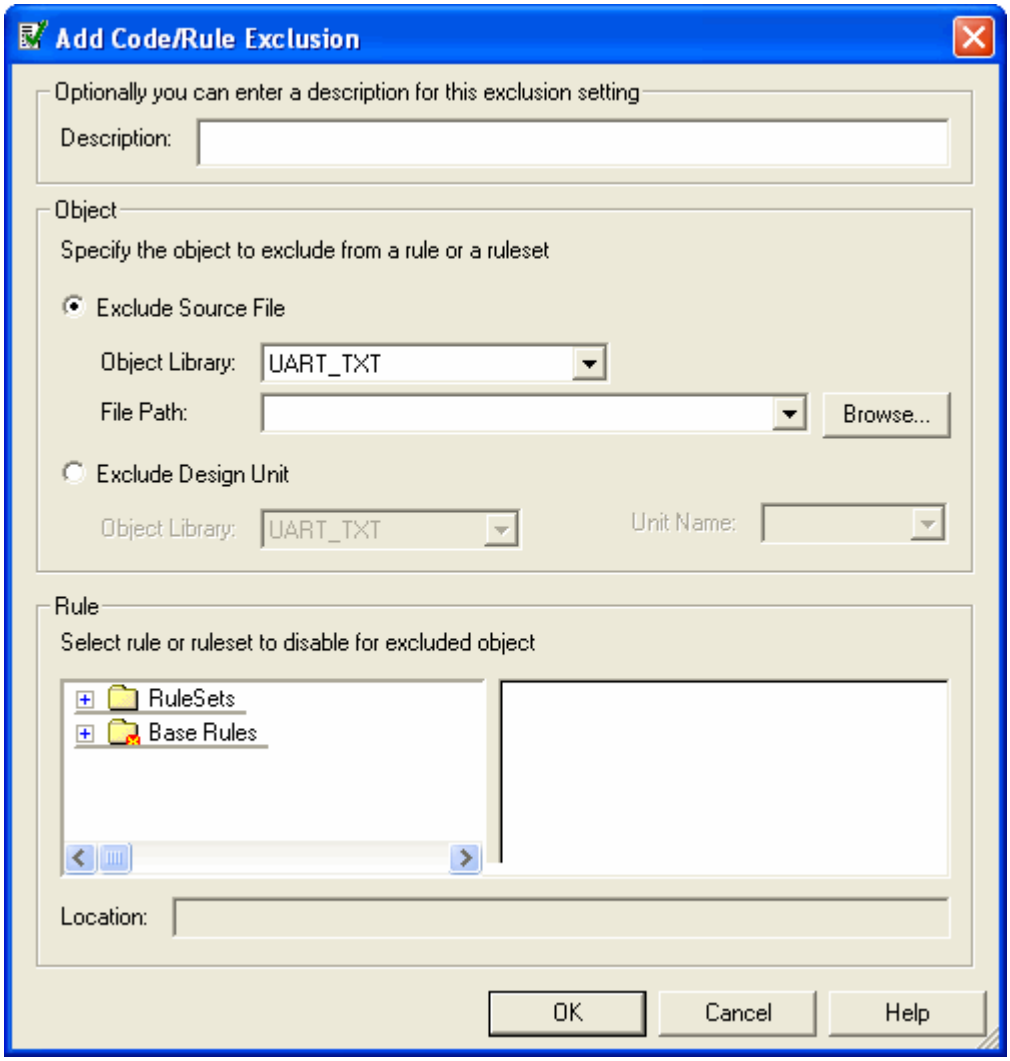


Table 4-2. Add Code/Rule Exclusion Dialog Box Controls

Control	Description
Description Field	Lets you enter a short description of your defined exclusion.
Object Group Box	Allows you to define whether you want to exclude a file or a design unit.
Object Library Dropdown list	Allows you to specify the library you wish to exclude a file/design unit from.The list displays the libraries on which you had run your last analysis.
File Path Field	Lets you enter or browse for the file you wish to exclude from checking.
Unit Name Dropdown list	Allows you to select a design unit from the specified library to exclude from checking.

Table 4-2. Add Code/Rule Exclusion Dialog Box Controls (cont.)

Control	Description
Rule Box	Allows you to browse for rules or rulesets you wish to disable for the objects defined in the Object Group box.
Location field	Displays the path to the chosen rules/rulesets

DC Constraints File

The DC constraints file is a Tcl file in which code/rule exclusions set for a given library are saved. The file is located under the library's HDS mapping and is given the name '*dc_constraints.tcl*'.

It can be opened for viewing or editing by selecting the **Edit Code/Rule Exclusions File** from the popup menu of the Exclusions node in the HDS Files' browser. Consequently, the file will be opened in DesignPad.

The content of the file is derived from the code/rule exclusions set visually through DesignChecker or set manually by directly editing the file using the *dc_exclude* command. Refer to "dc_exclude API" on page 30.

Note

Do not attempt to edit the *dc_constraints* file when running DesignChecker to avoid losing your manual edits.

dc_exclude API

Exclude selected library design unit or file from being checked by the specified rule or ruleset. Exclusion settings are associated to the library under which the *dc_constraints* file holding the *dc_exclude* command resides.

Syntax

dc_exclude -design_unit <design unit name> | -source_file <full file path> **-check** <hierarchical check path> [comment<comment text>]

Arguments

- **-design_unit <design unit name>**

  Specifies the design unit name to be excluded from the check. You must specify either **-design_unit** or **-source_file**. Check examples below.

- **-source_file <full file path>**

  Specifies the path to the file to be excluded from the check.You must specify either **-design_unit** or **-source_file**. Check examples below.

- **-check < check path>**

  Specifies the path to the rule or ruleset name to be disabled for a specific design unit or file.

- **-comment<comment text>**

  Specifies comment text that you can associate with the defined exclusion.

Examples

- Exclude the design unit **serial_interface** from being checked by the **Unused Declarations** configured rule.

```
dc_exclude -design_unit {serial_interface) -check
{RuleSets\Essentials\Coding Practices\Unused Declarations}
```

- Exclude the file **control_operation_fsm.vhd** from being checked by the **Downstream Checks** ruleset.

```
dc_exclude -source_file
{R:\HDS_2008.1\Examples\uart_txt\hdl\control_operation_fsm.vhd} -
check {RuleSets\Essentials\Downstream Checks}
```

- Exclude the design unit **address_decode** from being checked by the **Allowed Constructs** base rule and all its configured rules.

```
dc_exclude -design_unit {address_decode) -check {Base
Rules\Allow\Allowed Constructs}
```

Setting Code/Rule Exclusions Visually

Code/Rule Exclusions can be set visually through the Add Code/Rule Exclusions dialog box or through working directly with the analysis results in the Results tab.

Method 1

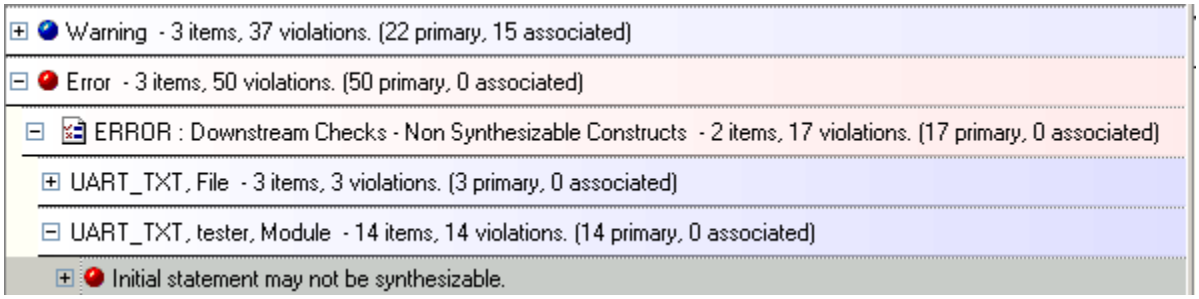
1. Open the Add Code/Rule Exclusions dialog box. Refer to “Add Code/Rule Exclusions Dialog Box” on page 28.
2. In the Object Group Box define the design object to be excluded by selecting one of the following:
   - Exclude Source File: This excludes a design file from being checked.
     - From the Object library dropdown list specify the source file library.
     - Enter or browse to the design file path.
   - Exclude Design Unit: This excludes a design unit from being checked.
     - Specify the design unit library from the Object library dropdown list.
     - Specify the design unit name from the Design unit dropdown list.

3. In the Rule Box select the rulesets or rules you wish to disable for the design objects specified in step two. The path to the selected rule/ruleset is displayed in the Location field.
4. Click **OK** to close the Add Code/Rule Exclusions dialog and run your analysis.

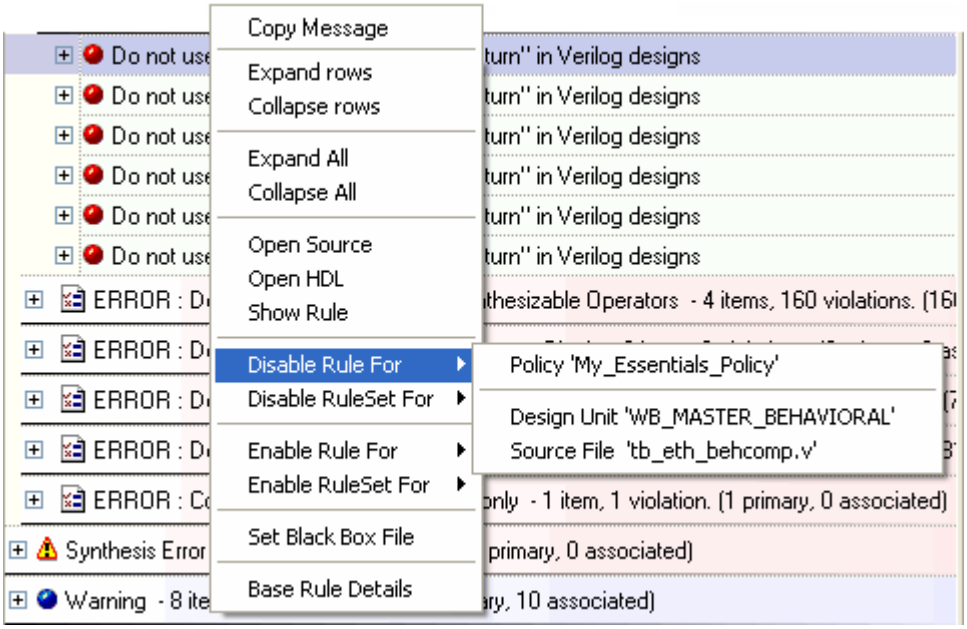
Method 2

In the Results tab, expand the Warning or Error tree.

1. Select a violation for a rule/ruleset and design unit/file you wish to disable. E.g. In the snapshot below we have chosen a violation for Non-synthesizable Constructs rule and tester module.



2. Right-click and choose **Disable Rule For** to display a cascade menu from which you can select one of the following:
   - Policy < policy used in last run>
   - Design Unit <violating design unit name>
   - Source File <violating source file name>.



3. Select the required design unit or file and run your analysis.

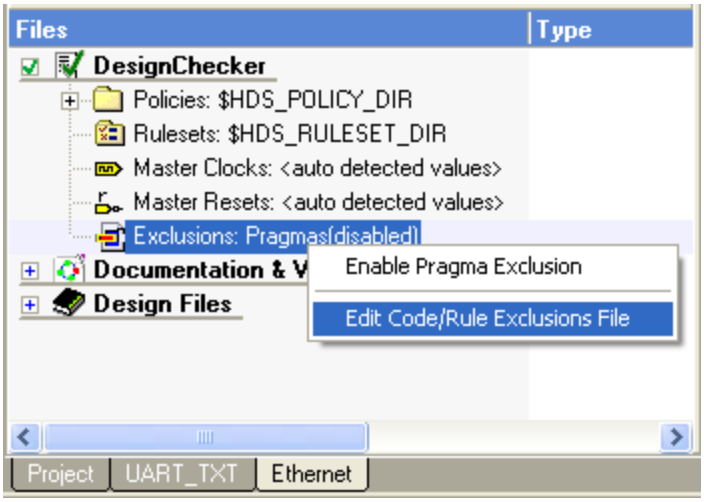
Note

You can turn on or off certain checks through the Results tab. You can right-click on a violation message in the Results tab and select **Disable Rule/Ruleset For** from the popup menu and from the cascade menu you can choose whether to disable the rule/ruleset on the level of the policy, design unit or source file. You can re-enable checks in the same way by right-clicking on the violation message and selecting **Enable Rule/Ruleset For** from the popup menu and from the cascade menu you can choose whether to enable the rule/ruleset on the level of the policy, design unit or source file.

Any rules/rulesets you disable for design units or source files are displayed in the Exclusions tab; refer to "Editing Exclusions" on page 40.

Setting Code/Rule Exclusions Manually

1. In the Files pane of the HDS Design Manager, use the plus ⊞ sign to expand the DesignChecker tree.
2. Right-click on the Exclusion node and click on **Edit Code/Rule Exclusions File**. DesignPad is invoked displaying the *dc_constraints* file for the active library.
3. Add your exclusions using the *dc_exclude* command, save your file and then run your analysis. Refer to **"DC Constraints File"** on page 30.



Enabling Pragma Exclusion

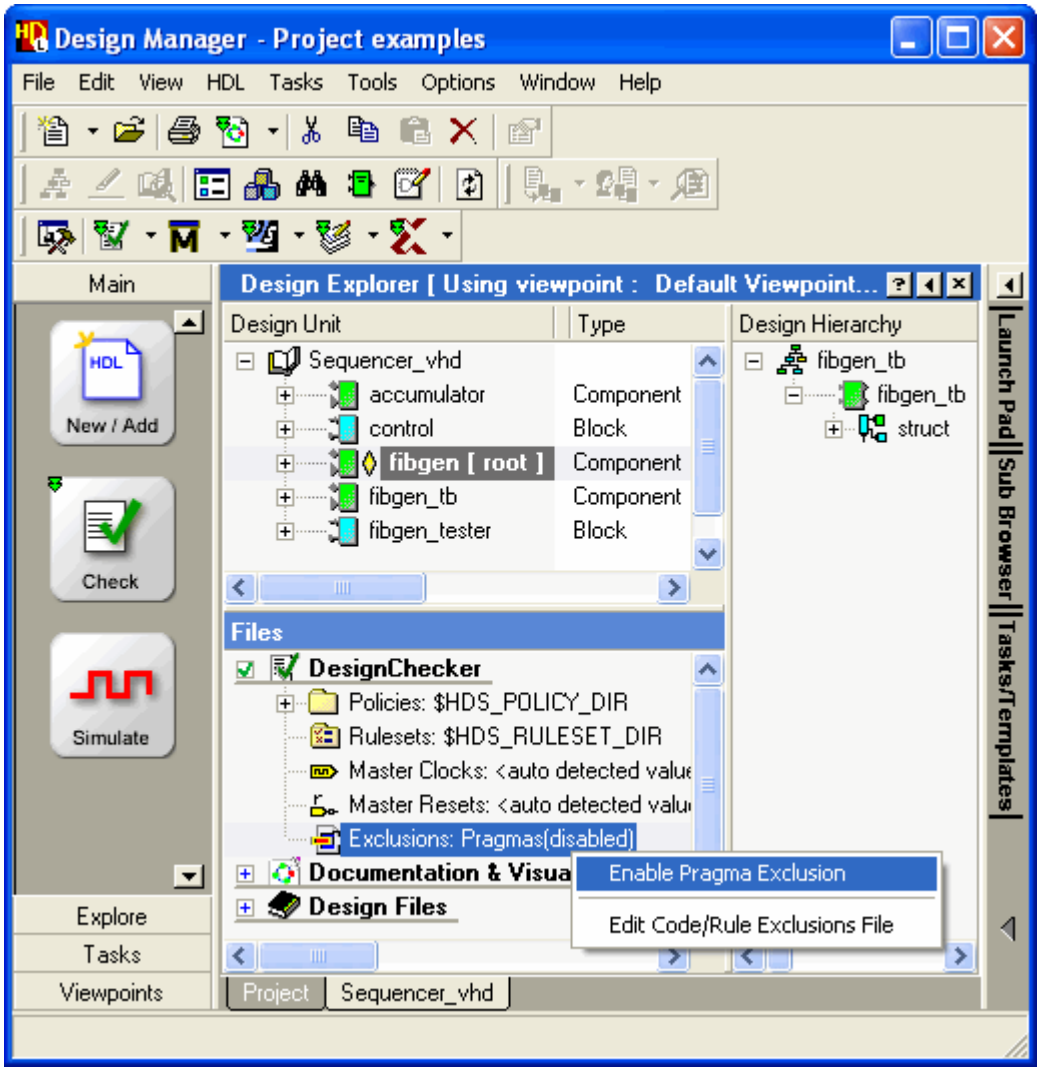
Exclusion via Pragma Pairs enables you to force DesignChecker to skip specific RTL code blocks while running an analysis. For example, parts of your code may be used only for debugging purposes, and hence, it may not be necessary to have these parts checked. In this case, you can enable the Exclusion via Pragma Pairs feature, and then apply the pragmas to your

code. This feature relies basically on embedding the pragmas in your HDL code in such a way to enclose the sections that shall be excluded by DesignChecker.

Procedure

1. In the Files pane of the HDS Design Manager, use the plus ⊞ sign to expand the DesignChecker tree.
2. Right-click on the Exclusions node and click on **Enable Pragma Exclusion**.

   By doing this step, the pragma *checking_off/on* is enabled.



Note

Note that synthesis pragmas are always enabled (*translate_off/on*, *synthesis_off/on*, *dc_script_begin/end*). You can only enable or disable the DesignChecker-specific pragma *checking_off/on* by selecting **Enable/Disable Pragma Exclusion** from the popup menu of the Exclusions node. The pragma *checking_off/on* cannot exclude syntax errors.

Caution

Double commenting a pragma disables the pragma.
It's recommended to write the pragma in a separate line to ensure it works properly.

3. Within your source code, use the exclusion pragma pair *checking_off/on* to encase the excluded code.

   The pragma pair constitutes of a start pragma and an end pragma. The start pragma is placed as a comment before the beginning of the code block as an indication for DesignChecker to stop analysis at this point; similarly, the end pragma is placed as a comment after the end of the code block as an indication to resume analysis. Note that excluded code is highlighted in gray by default in DesignPad.

   For example, you can exclude RTL code sections (such as processes, concurrent assignments, etc.), or you can exclude a whole design unit (entity, architecture, etc.). If an entire VHDL entity is excluded, all the relevant architectures will not be checked even if they were not explicitly excluded. According to the used scope, the Results tab is affected.

   As an example of pragma pairs used in different scopes, the following Verilog example shows the exclusion of an entire module:

   ```
   // pragma checking_off
   module pragma_exclusion_du1 (i1, o1);
   input i1;
   output o1;
   assign o1 = i1;
   endmodule
   // pragma checking_on
   ```

   The following Verilog example excludes only an always block:

   ```
   module pragma_exclusion_du1 (i1, o1);
   input i1;
   output o1;
   // pragma checking_off
   always @(i1) begin
   o1 <= i1;
   end
   // pragma checking_on
   endmodule
   ```

4. Run the analysis. For details on running an analysis, refer to "Running the Analysis" on page 15.

   The Results tab opens displaying the outcome of the analysis; the details of the exclusion settings are displayed in the Summary pane. For more details on how exclusion affects your results, refer to "Reporting Exclusions in the Results Summary Pane" on page 46.

DesignChecker Predefined Exclusion Pragmas

In addition to the DesignChecker-specific pragma *checking_off/on*, DesignChecker provides a predefined set of synthesis-specific pragmas which you can use for code exclusion. Each pragma pair is defined by a start pragma name and an end pragma name. While the DesignChecker *checking_off/on* pragma can be optionally enabled or disabled through HDS Design Manager, all the synthesis pragmas (listed in Table 4-3) are always enabled by default. The supported pragma pairs are as follows:

Table 4-3. Supported Exclusion Pragma Pairs (Synthesis-Specific)

Short Name	Start Pragma	End Pragma
dc_script_begin/end	dc_script_begin	dc_script_end
synthesis_off/on	synthesis_off	synthesis_on
translate_off/on	translate_off	translate_on

Table 4-4. Supported Exclusion Pragma Pairs (DesignChecker-Specific)

Short Name	Start Pragma	End Pragma
checking_off/on	checking_off	checking_on

You insert the start pragma before the excluded code, and then insert the end pragma after. Thus, this pragma-enclosed section, which is highlighted in gray by default in DesignPad, is skipped during the DesignChecker analysis.

Exclusion Settings Batch Commands

Exclusions can be enabled through Tcl commands before invoking DesignChecker.

Table 4-5. Tcl Commands for Configuring Exclusion Settings

Command	Arguments
enablePragmaExclusions	“1” to enable, or “0” to disable

The following example enables pragma exclusion, and runs the DesignChecker on *uart_txt* library.

```
enablePragmaExclusions 1
runTask {DesignChecker\ Flow DesignChecker} UART_TXT uart_tb
```

For more details, refer to “The Batch Process” on page 51.

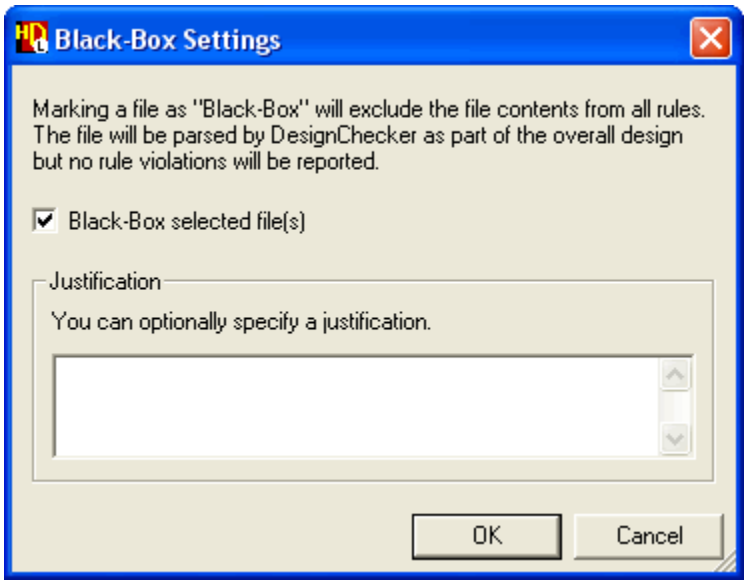
Setting Black Box Exclusions

Marking a design component as a “Black Box” ensures that DesignChecker recognizes that component is present, but does not apply any checks to it. For example, you may wish to mark third party components used in your designs as black boxes as they may not comply with your coding standards.

Procedure

To set black box exclusions, do one of the following:

- You can set the source view corresponding to the selected result row as a black box for DesignChecker by choosing **Set Black Box File** from the popup menu of the message in the results tab.
- Alternatively, select a design view and choose **DesignChecker > Edit Black-Box Settings** from the **Edit** or popup menu in the HDL Designer Series design manager. A dialog box opens as a result, set the black box option and then provide a justification in the text box.



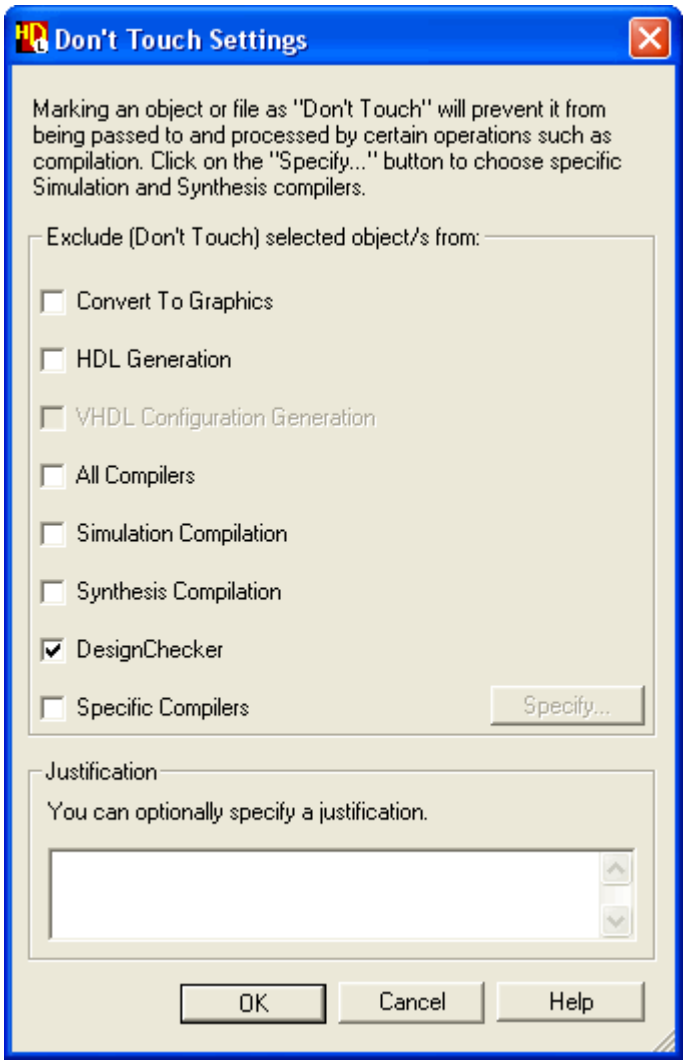
When a file has been set as a black box for DesignChecker, the file is parsed but no checks are performed. Files that have been marked as black boxes can be identified in the design manager window by the presence of the icon.

Setting Don’t Touch Exclusions

The Don’t Touch setting ensures that a file is not loaded or analyzed by DesignChecker.

Procedure

HDL Designer Series users can disable checking for a design view by using the **Don't Touch Settings** command from the **Edit** or popup menu in the HDL Designer Series design manager. This opens the Don't Touch Settings dialog box. Choose the DesignChecker option as shown in the figure.



You will be able to enter a justification for setting a view as Don't Touch and this justification will be displayed later in DesignChecker's Exclusion tab, in the Don't Touch Files pane.

When a Don't Touch property has been set, the view will be ignored for subsequent analysis until the property is unset. For example, you may want to disable checking for the non-synthesizable views in a test bench but keep the synthesis checks enabled for all other views. When set, the ! icon appears against the file in the design manager. You may also want to refer to the "Disabling Downstream Operations" section in the *HDL Designer Series User Manual* for more information.

Adding Justification for Black Boxed Files, Don't Touch Files and Disabled Rules

You have the ability to keep record of why you have files marked as black box, why you have files marked as Don't Touch, and why you have disabled rules. It is useful to have this information stored for reference.

Black Boxed Files

As mentioned in "Setting Black Box Exclusions" on page 37, you can mark a file as black box either through HDL Designer Series or through DesignChecker.

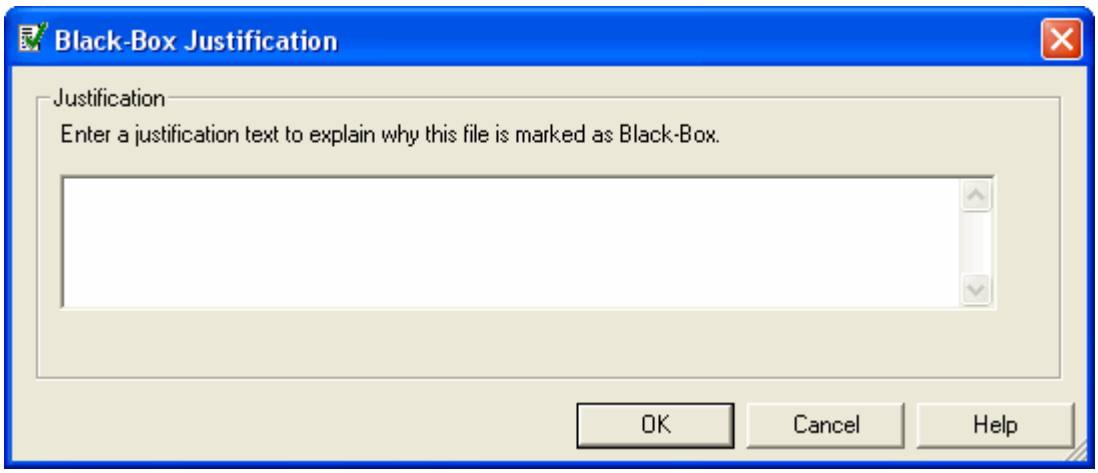
You can add (or edit) justifications through HDL Designer Series using the Black Box Settings dialog box:

1. Select a design unit in the Design Manager and then select **DesignChecker > Edit Black-Box Settings** from the **Edit** or popup menu.
2. In the text box type the justification.
3. Click **OK**.

You can also add (or edit) justifications through DesignChecker as follows:

1. Open the Exclusions tab in DesignChecker.
2. Right-click on a entry in the Black Boxed Files pane, and select **Edit Justification** from the popup menu.

   The Black-Box Justification dialog box opens.



3. In the text box type the justification.
4. Click **OK**.

Note that if you edit a justification through HDL Designer Series, then DesignChecker will be updated (and vice-versa).

The Black Boxed Files pane in the Exclusions tab shows the justification related to each file in a separate column.

Don't Touch Files

As mentioned in "Setting Don't Touch Exclusions" on page 37, you can set a file as Don't Touch through HDL Designer Series and specify a justification. This justification will be displayed in the Exclusions tab, namely in the Don't Touch Files pane.

Disabled Rules

You can add justifications for disabling rules on policy level through DesignChecker. You can disable a rule within a policy through the setup tab of DesignChecker by expanding the policy, selecting the required rule in the Content pane on the right hand side, and then unsetting the check box corresponding to that rule (or by right-clicking on the rule and selecting **Disable** from the popup menu).

If you disable a rule for a policy through the Results tab, by right-clicking on a violation and selecting **Disable Rule For > Policy <policy_name>** from the popup menu, you can also add a justification for disabling that rule using the same steps below.

To add justification for the disabled rule, you can do the following:

1. In the setup tab, expand the policy and select the disabled rule in the Content pane.
2. Right-click on the disabled rule and choose **Edit Justification** from the popup menu.

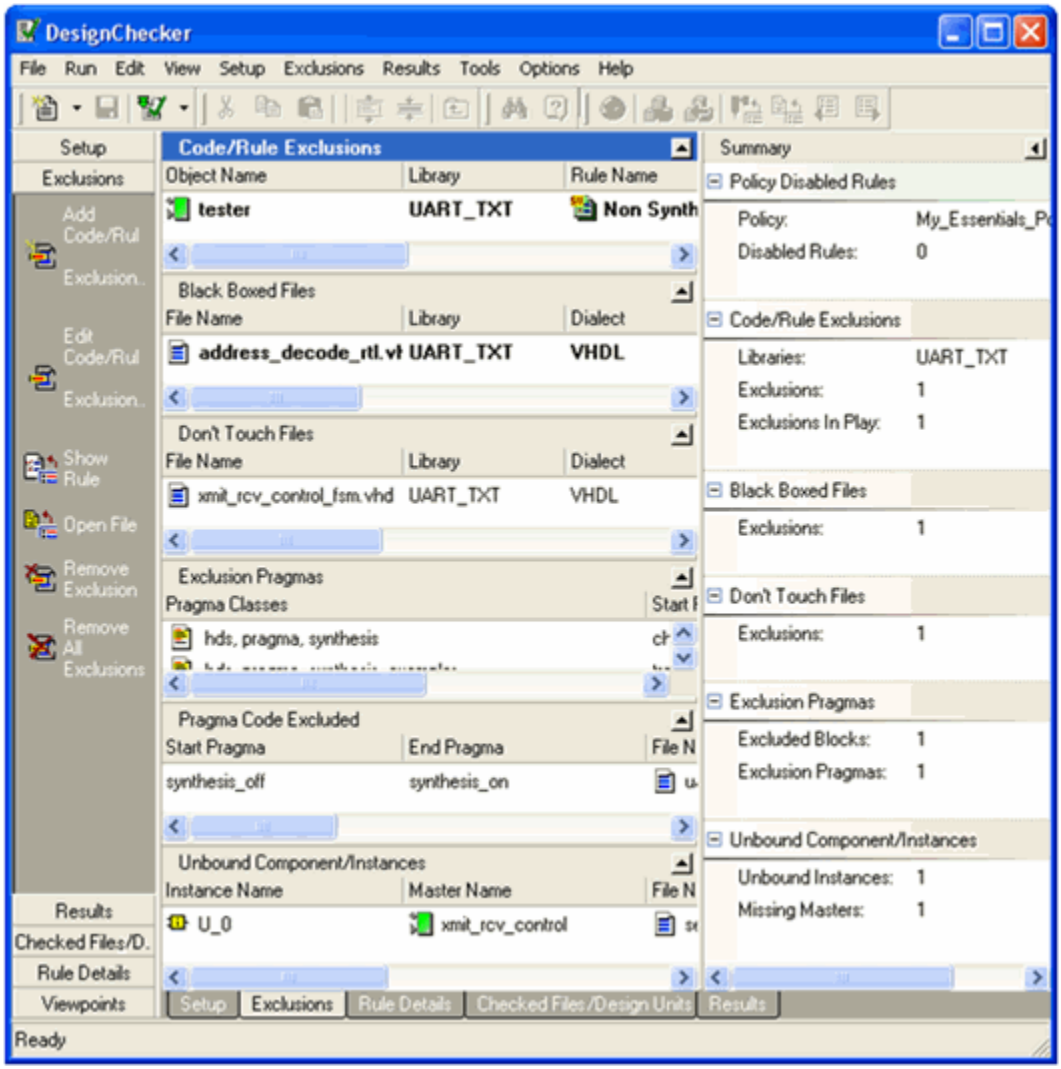
   The Edit Justification dialog box opens.
3. Use the text box to type the justification. Note that you can write a multi-line justification, and that this justification will be retained by the tool even if you enable the rule and disable it again.
4. Click **OK**.

You can edit an existing justification using the same steps above.

Editing Exclusions

Once you have run DesignChecker, the exclusions tab is available from which you can review the exclusions related to the last run. The tab contains facilities to add/edit code/rule exclusions,

remove exclusions or link to excluded files or rules. The Exclusions tab is divided into an exclusions viewpoint and a summary pane, as shown in the picture below.



The summary pane (shown on the right) contains high-level information about the policy disabled rules, code/rule exclusions, black boxed files, don't touch files, exclusion pragmas and unbound component/instances.

The viewpoint (shown on the left) displays code/rule exclusions, black-boxed files, don't touch files, exclusion pragmas, pragma excluded code and unbound component/instances each in a separate pane.

Editing Code/Rule Exclusions

The code/rule exclusions pane displays the code/rule exclusions related to design units/files that are part of the library on which DesignChecker was last run. Exclusions related to the last

analysis are shown in bold while those that are not are listed in normal font. (See Using the Code/Rule Exclusions Pane.)

Example

This is an example to demonstrate how exclusions related to the last DesignChecker analysis results are highlighted.

Prerequisites

- UART.TXT library of the Examples Project should be opened.
- DesignChecker should be configured to use My_Essentials_Policy.

Steps

1. In the Design Explorer of HDL Designer Series, select the “tester” design unit and choose **Run Single** from the popup menu of the **Check** button in the shortcut bar on the left hand side.
2. In the DesignChecker Results tab, select a violation for “Non Synthesizable Constructs” rule and “tester” design unit. Choose **Disable Rule For > Design unit ‘tester’** from the popup menu.
3. Open the Exclusions tab and note that a row has been added to the Code/Rule Exclusions pane showing the new exclusion added. The bold font indicates that the new exclusion will affect the results if applied to the last analysis run by DesignChecker.
4. Return back to the Design Explorer window and select the “clock_divider” design unit. Choose **Run Single** from the **Check** button’s popup menu.
5. In the DesignChecker Results tab, select a violation for “Coding Practices” rule and “clock_divider.v” file. Choose **Disable Rule For > Source File ‘clock_divider.v’** from the popup menu.
6. Open the Exclusions tab. A new clock_divider/Coding Practices exclusion is added in bold. Note that the Tester/Non-synthesizable Constructs exclusion is now displayed in normal font as it would not affect the results of the last analysis.

Using the Code/Rule Exclusions Pane

The Code/Rule Exclusions pane displays a list of all the library code/rule exclusions set for the libraries on which DesignChecker was last run. Refer to “Editing Code/Rule Exclusions” on page 41.

You can view the following information on each exclusion in the Code/Rule Exclusions pane: the Object Name (the design unit or source file that is being excluded in conjunction with a rule), the Library to which the object belongs, the Rule Name (the check that should not be applied to the object), the Object Type (whether it is a Design Unit or Source File), the

Description of the exclusion, and the Rule Path (for example, RuleSets\Essentials\Downstream Checks\Non synthesizable Constructs).

Tip: You can right-click on any column title in the Code/Rule Exclusion panes and add more columns from the **Select Columns** cascade such as Affects Results (the value of this column is Yes or No depending on whether the exclusion affected the last analysis or not) and Object Full Name (which shows the full name of the excluded object if it is a design unit or the physical path if it is a source file).

You can also select one of the listed exclusions and right click to display a popup menu with the following options:

Table 4-6. Code/Rule Exclusion Pane Options - Exclusion Selected

Menu Item	Description
Edit Code/Rule Exclusions	Displays the Edit Code/Rule Exclusions dialog to enable you to change the information related to the selected exclusion. Refer to "Add Code/Rule Exclusions Dialog Box" on page 28.
Show Disabled Rule	Displays the Setup tab with the disabled rule highlighted.
Open Source File(s)	Opens DesignPad showing the excluded file. Notice that this option is disabled if a rule was disabled for a design unit.
Remove Exclusion	Deletes the selected exclusion.

If you right click in the Code/Rule Exclusions pane without selecting an exclusion a popup menu with the following options is displayed.

Table 4-7. Code/Rule Exclusion Pane Options - No Exclusion Selected

Menu Item	Description
Add Code/Rule Exclusion	Displays the Add Code/Rule Exclusions dialog to enable you to add a new exclusion.
Remove all Exclusions	Deletes all the exclusions listed in the Code/Rule Exclusions pane.

Reviewing Black Box Exclusions

The Black Boxed Files pane displays the black box exclusions set for the code on which DesignChecker was last run. Refer to "Setting Black Box Exclusions" on page 37.

You will be able to view the name of the black boxed file, the library to which it belongs, the dialect of the file, the location of the file on the hard disk, whether or not the file affects results, and the justification for being marked as a black box.

You can right-click on any black box exclusion and select **Open Source File(s)** from the popup menu to open the source file whether in a graphical editor or in DesignPad. You can also select **Edit Justification** which opens the Black-Box Justification dialog box in which you can edit or add an explanation for reference.

You can cancel the black box by right-clicking on the required black box exclusion and selecting **Remove Exclusion** from the popup menu.

Reviewing Don’t Touch Exclusions

The Don’t Touch Files pane displays the “don’t touch” exclusions applied in the latest DesignChecker analysis. Refer to “Setting Don’t Touch Exclusions” on page 37.

You will be able to view the name of the file, the library to which it belongs, the dialect of the file, the location of the file on the hard disk, and the justification for being set as “don’t touch”.

You can right-click on any don’t touch exclusion and select **Open Source File(s)** from the popup menu to open the source file whether in a graphical editor or in DesignPad.

Reviewing Exclusion Pragmas

The Exclusion Pragmas pane displays the pragmas set for the project that includes the design units/files on which DesignChecker was last run. Exclusions related to design files that were part of the last analysis are shown in bold while those related to files that were not are listed in normal font.

Reviewing Pragma Code Excluded

The Pragma Code Excluded pane shows all the code blocks that have been excluded from the latest DesignChecker analysis due to using exclusion pragmas.

The pane shows the Start Pragma and End Pragma used in the exclusion, the name of the file in which the pragma pair was used, the number of the Start Line and End Line of the excluded code block, and the location of the file on the hard disk.

Note that double-clicking on an entry in this pane opens the corresponding file in DesignPad.

Reviewing Unbound Component/Instances

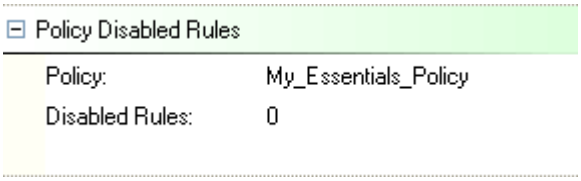
The Unbound Component/Instances pane displays a list of the instances which are not bound to masters as found in the latest DesignChecker analysis.

You will be able to view the name of the unbound instance, the name of its missing master, the name of the relevant file, the line number of the instance, and location of the file on the hard disk.

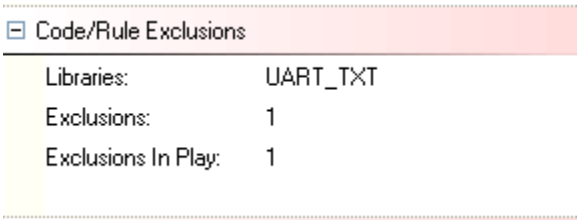
The Exclusions Summary Pane

The Exclusions summary pane displays an overview of the set exclusions in tabular form under the following headings:

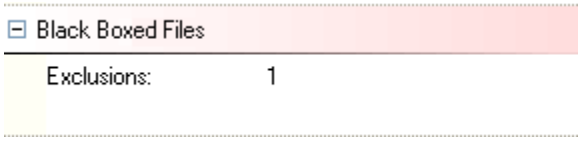
- **Policy Disabled Rules:** This displays information about the rules that have been disabled from a specific policy. For example:



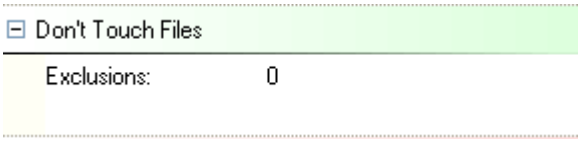
- **Code/Rule Exclusions:** This displays the names of the libraries that were part of the last DesignChecker analysis, the number of the set code/rule exclusions, and the number of exclusions that were actually used in the last run.



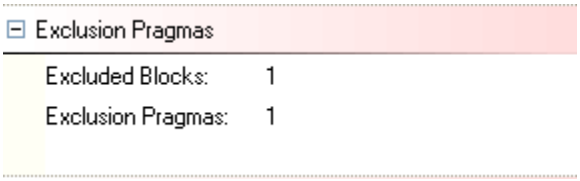
- **Black Boxed Files:** This displays the number of black box exclusions. For example:



- **Don't Touch Files:** This shows the number of excluded files that are set as "don't touch". Refer to "Setting Don't Touch Exclusions" on page 37.

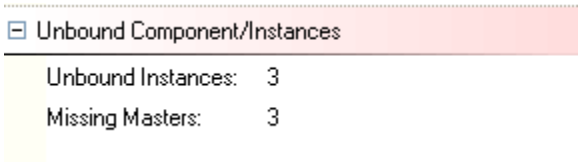


- **Exclusion Pragmas:** This shows the number of excluded code blocks and the number of exclusion pragmas. Refer to "Enabling Pragma Exclusion" on page 33 for information.



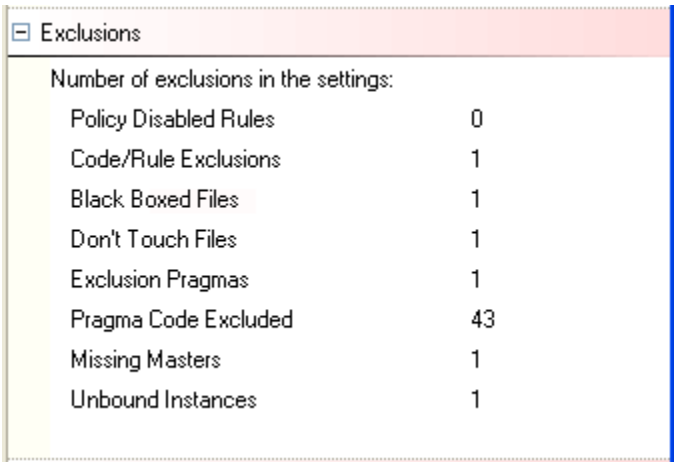
  Note that the number of exclusion pragmas reflects the actual number of pragma types encountered during the analysis (for example: checking_off/on, dc_script_begin/end, and so on).

- **Unbound Component/Instances:** This displays the number of unbound component/instances and the number of missing masters in the latest run.



Reporting Exclusions in the Results Summary Pane

The Summary pane of the Results tab contains a section titled Exclusions, in which data on the exclusion settings is displayed. Refer to "Setting Exclusions" on page 27 for more information on how to set exclusions.



It is important to note that the data shown in the Exclusions section includes data on the exclusion settings that affected the last performed analysis only. For example, you may have set four files as black boxed files, whereas only two are reported in the Exclusions pane of the Results Summary pane; this means that only two black boxed files of the four were involved in the scope of analysis.

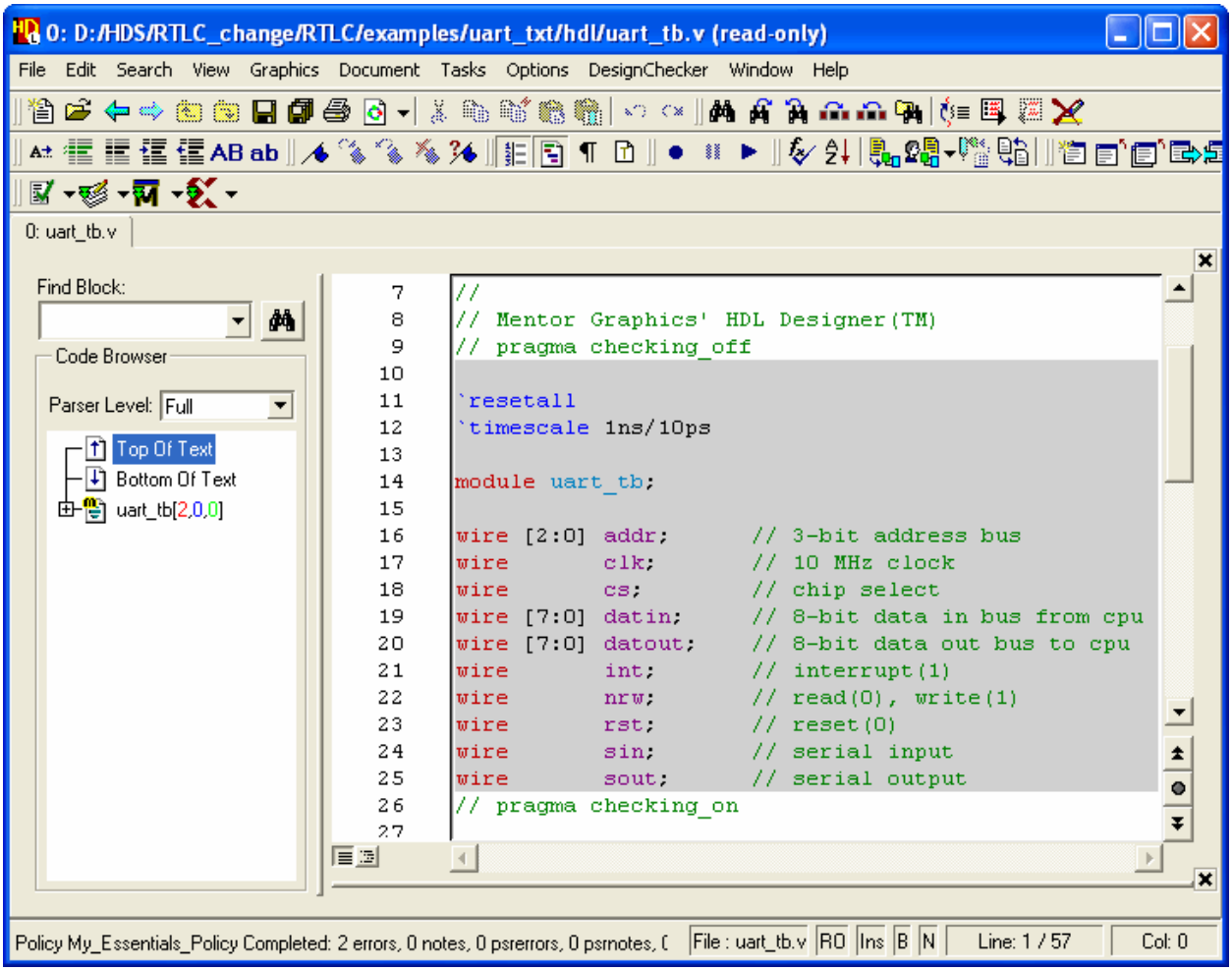
The following data is displayed in the Exclusions summary section:

- **Policy Disabled Rules** — shows the number of rules that have been disabled in the policy and hence were not applied to the analyzed code. If you right-click on this entry and select **Show Exclusions** from popup menu, the Rule Details tab is opened displaying the policy. Refer to "The Rule Details Tab" in the *DesignChecker User Guide* for information on the tab's content
- **Code/Rule Exclusions** — shows the number of code/rule exclusions that have affected the analysis results. If you right-click on this entry and select **Show Exclusions** from the popup menu, the Exclusions tab is opened displaying the Code/Rule Exclusions pane.
- **Black Boxed Files** — shows the number of black boxed files that have been excluded from the analyzed code. If you right-click on this entry and select **Show Exclusions** from the popup menu, the Exclusions tab is opened displaying the Black Boxed Files pane.
- **Don't Touch Files** — shows the number of "don't touch" files that have been excluded from the analyzed code. If you right-click on this entry and select **Show Exclusions** from the popup menu, the Exclusions tab is opened displaying the Don't Touch Files pane.
- **Exclusion Pragmas** — shows the number of exclusion pragma types that have affected the analyzed code. On running a DesignChecker analysis while having enabled exclusion pragmas, DesignChecker will skip the HDL blocks wrapped by the pragma pairs. If you right-click on this entry and select **Show Exclusions** from popup menu, the Exclusions tab is opened displaying the Exclusion Pragmas pane.

  Enabling exclusion pragmas is reflected on your results as follows:

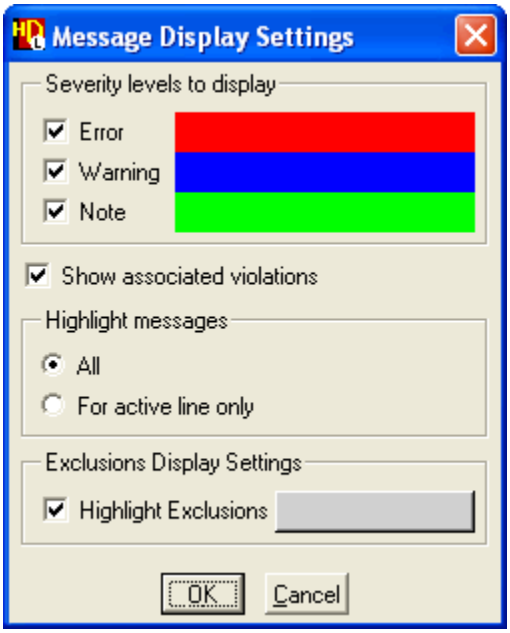
  - If an entire design unit is excluded, it will not be counted in the total number of analyzed design units which is shown in the Design Units section of the Results Summary pane.

- In the DesignPad, all the excluded blocks are highlighted in gray in addition to the corresponding objects in the Code Browser.



The default highlighting color for the excluded code is gray; however, you can change the color through DesignPad by selecting **Message Display Settings** from

the **DesignChecker** menu. In the Exclusion Display Settings section, select the required highlighting color.



- **Pragma Code Excluded** — shows the number of code blocks that have been discarded from analysis using exclusion pragmas. If you right-click on this entry and select **Show Exclusions** from the popup menu, the Exclusions tab is opened displaying the Pragma Code Excluded pane.
- **Missing Masters** — shows the number of missing masters in the analyzed design. If you right-click on this entry and select **Show Exclusions** from the popup menu, the Exclusions tab is opened displaying the Unbound Component/Instances pane.
- **Unbound Instances** — shows the number of instances that are not bound to masters. If you right-click on this entry and select **Show Exclusions** from the popup menu, the Exclusions tab is opened displaying the Unbound Component/Instances pane.

Appendix A
The Batch Process

You can run DesignChecker in batch mode from a command line without any need for interaction with the user interface. This is achieved using a Tcl script. Your Tcl scripts can consist solely of DesignChecker commands, or can be more complex and include commands defining arguments to be passed into the script, checking arguments, verifying pathnames or checking for existing filenames.

Running in Batch Mode

You can run DesignChecker in batch mode using the DesignChecker task. A task is a method of integrating scripts, tools and flows. Tasks accept setup commands and then use a run command to launch the task itself. You can use these commands in a script file, or embed them within a Tcl script to perform a variety of operations. You invoke the DesignChecker task using the *-tcl* switch to load a Tcl script containing API commands. For example:

```
hdldesigner -tcl my_analysis_script.tcl
```

You can run DesignChecker and output the results to your screen and a file using the following command:

```
hdldesigner -tcl my_analysis_script.tcl 2>&1 | tee log.txt
```

DesignChecker batch mode is driven using the normal HDS task mechanism, with the standard HDS task commands and switches.

The following tool setting parameters are supported by the DesignChecker task:

Table A-1. Tool Setting Parameters

Description	Parameter Name	Values	Default
Master clock	masterClock	any valid signal name	clk
Master reset	masterReset	any valid signal name	rst
Output file	OutputFile	filename of your choice	none
Output format	OutputFormat	CSV, TSV, HTML	TSV
Summary file	SummaryFile	filename of your choice	none
Summary format	SummaryFormat	CSV, TSV, HTML	TSV

Table A-1. Tool Setting Parameters (cont.)

Description	Parameter Name	Values	Default
Summary Report Content	SummaryReportContent	Settings, Design Units, Quality, Rules, Violations	All
Output Report Content	OutputReportContent	Rule Severity, Rule Category, Ruleset, Rule Name, Library, Design Unit Name, Scope, Filename, Line Number, Message, Hint	All
Run Quality Analysis	RunQualityAnalysis	On | Off	On
Include Disabled Rules	IncludeDisabledRules	on | Off	Off
Enable Pragma Exclusions	enablePragmaExclusions	1 | 0	0
Results Report Configuration File	ConfigFile	filename of your choice	none
Run Level	RunLevel	FILE_LEVEL, UNIT_LEVEL	FILE_LEVEL
Run Mode	RunMode	INTERACTIVE, BATCH	INTERACTIVE

For example, the following script runs the DesignChecker tool task on the Ethernet example design, through components with the master clock set to *Clk1*, the master reset set to *Rst1*. The results are saved as HTML files:

```
setupTask {DesignChecker} -throughCpt
setupTask {DesignChecker} -setting SummaryFile
c:/batch_results/results_summary.htm
setupTask {DesignChecker} -setting OutputFile
c:/batch_results/results_output.htm
setupTask {DesignChecker} -setting masterClock Clk1
setupTask {DesignChecker} -setting masterReset Rst1
setupTask {DesignChecker} -setting OutputFormat HTML
setupTask {DesignChecker} -setting SummaryFormat HTML
runTask {DesignChecker} Ethernet tb_cop
```

Refer to “Batch Command Language” in the *HDL Designer Series User Manual* for more information about running tasks from a command file.

You can use the *ConfigFile* option as shown below to call a Tcl script containing the API commands required to configure results reports. The API commands produce outcome

equivalent to that of the Export Results dialog box in DesignChecker (refer to “Exporting Results” in the *DesignChecker User Guide*).

```
setupTask {DesignChecker} –setting ConfigFile {/home/username/config.tcl}
```

Below is an example of a Tcl file with API commands configuring the reports required to be generated. Refer to “DesignChecker Commands” in the *HDL Designer Series Tcl Reference Manual* for information about the APIs that can be used in the report configuration Tcl file.

```
enableDumpCodeSnippet
setSnippetLinesBefore 2
setSnippetLinesAfter 1
enableFilterAssocViolation
setExclusionReport
{"/home/msyousuf/DC_Reports_2/Formal_Exclusion_report.csv"
"/home/msyousuf/DC_Reports_2/Formal_Exclusion_report.htm"} {"CSV" "HTML"}
setExclusionReportContents {{Black Boxed Files} {Don't Touch Files}}
setRulesCheckedReport
{"/home/msyousuf/DC_Reports_2/Formal_RulesChecked_report.csv"
"/home/msyousuf/DC_Reports_2/Formal_RulesChecked_report.htm"} {"CSV"
"HTML"}
setCheckedFileDUsReport
{"/home/msyousuf/DC_Reports_2/Formal_CheckedFileDUs_report.csv"
"/home/msyousuf/DC_Reports_2/Formal_CheckedFileDUs_report.htm"} {"CSV"
"HTML"}
```

The parameter *RunLevel* allows you to run either a design unit level check or a file level check. The parameter *RunMode* configures DesignChecker to run either interactively or in batch mode. It should be noted that this parameter gives the ability to run DesignChecker in batch mode even if invoked from an interactive HDL Designer Series session.

Also, you can store your rulesets and policies on a central server, so that every team member would have access to them. The following commands enable team members to set the shared ruleset and policy location:

```
setRulesetLocation <ruleset path>
setPolicyLocation <policy path>
```

Refer to “Sharing RuleSets” on page 57 and “Sharing Policies” on page 60 for more information on these two commands.

Note

Batch mode does not support the configuration of rulesets or user policies. To configure these for use in batch mode you must run DesignChecker interactively, configure your rulesets and policies, and set this as your default DesignChecker configuration before running any batch process.

General Notes:

- In case you are using more than one value within the same command, you have to enclose the values between curly braces { } or double quotation marks “ ” as follows:

```
setupTask {DesignChecker} -setting SummaryReportContent {Settings
Quality Rules}
setupTask {DesignChecker} -setting SummaryReportContent “Settings
Quality Rules”
```

- In case you are using a value that constitutes of more than one word separated by spaces, enclose this value within secondary curly braces as follows:

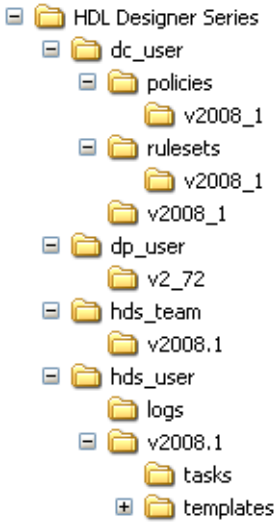
```
setupTask {DesignChecker} -setting SummaryReportContent {Settings
Quality Rules {Design Units}}
setupTask {DesignChecker} -setting SummaryReportContent “Settings
Quality Rules {Design Units}”
setupTask {DesignChecker} -setting SummaryReportContent {{Design
Units}}
setupTask {DesignChecker} -setting SummaryReportContent “{Design
Units}”
```

Appendix B
DesignChecker for Teams

It is often necessary to ensure that all members of a specific design team or all teams within the company are checking designs against the same set of rules. This can easily be achieved by sharing the rulesets and optionally sharing policies as well. Refer to "Sharing RuleSets" on page 57 and "Sharing Policies" on page 60 for details.

Default RuleSet and Policy Location

By default, DesignChecker uses the policies, rulesets and preferences in the *dc_user* folder within the directory structure of the HDL Designer Series preferences (illustrated in the below picture):



The default location for the *dc_user* folder on each platform is as follows:

```
Windows: Documents and Settings\<user_or_login_name>\Application Data\
HDL Designer Series\dc_user

UNIX and Linux: $HOME/hdl_designer_series/dc_user
```

By default, the ruleset and policy locations are set to use the $HDS_RULESET_DIR and $HDS_POLICY_DIR environment variables respectively. If you do not explicitly set these environment variables, they will default to the appropriate locations within the HDL Designer Preferences directory (equivalent to the following examples: *$HDS_USER_HOME/../dc_user/rulesets* and *$HDS_USER_HOME/../dc_user/policies*).

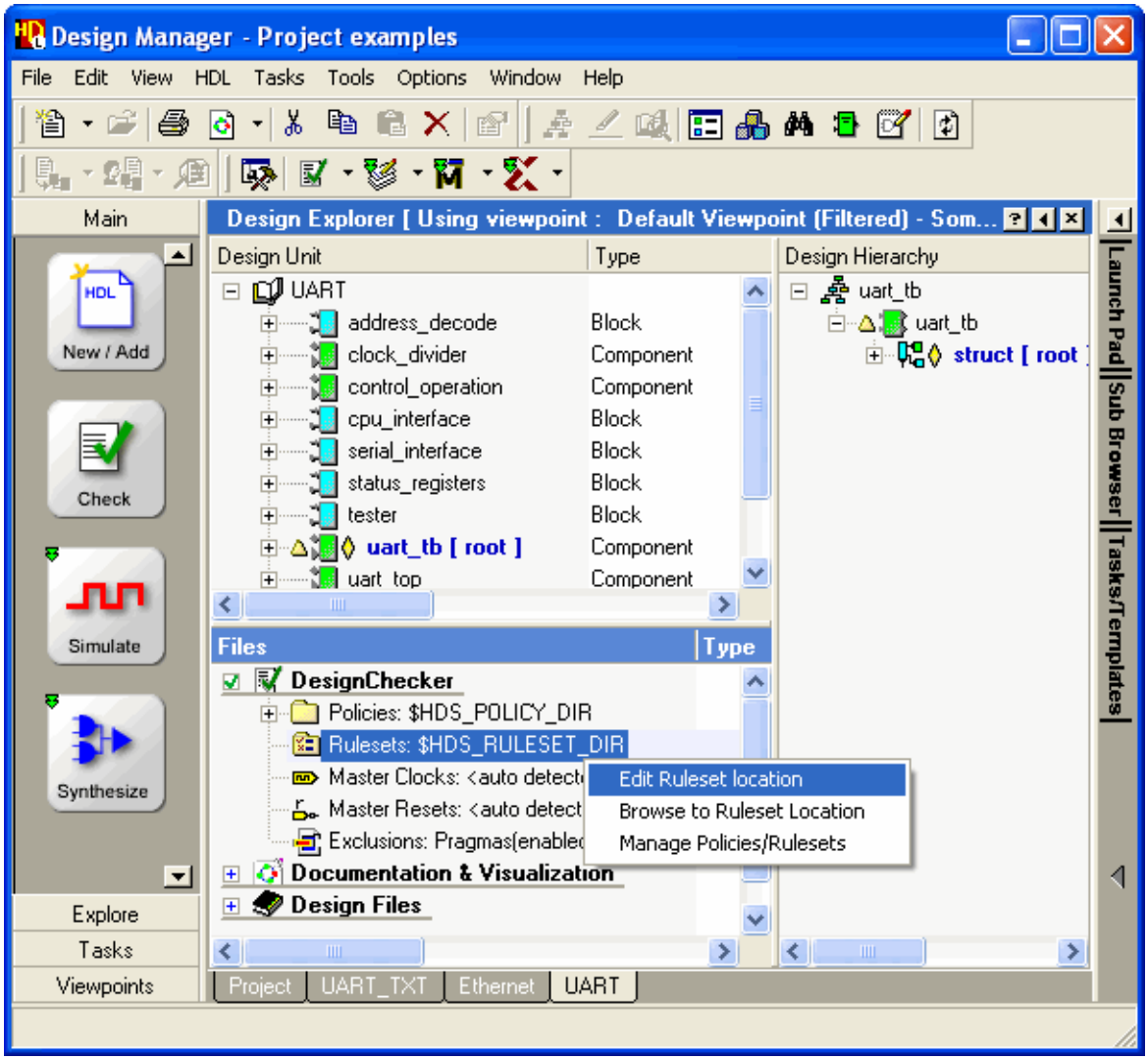
The location of the HDS user preferences (*hds_user*) can be changed using the HDS_USER_HOME environment variable or the *-user_home <path>* command line option. Note that the location of the DesignPad preferences (*dp_user*) and DesignChecker preferences (*dc_user*) are relative to this location, unless otherwise specified.

Sharing RuleSets

The first step to sharing rules within a team is for the project manager to create the required ruleset(s) in his/ her default location and then copy the rulesets directory to a central server. It is a good practice to change the access rights to these files to read-only to avoid accidental changes.

All members of the design team can then point to the shared rulesets directory instead of the default. The ruleset location can be set in two ways:

1. You can set the ruleset location directly through the Design Manager of HDL Designer Series:
   a. In the Files view, click on the plus sign to expand the DesignChecker tree.
   b. Right-click on the *Rulesets* folder and select **Edit Ruleset Location**.



c. Enter the path to the shared ruleset directory (you can use environment variables in the pathname, for example, *$SHARED/rulesets*).

   Alternatively, specify the path as $HDS_RULESET_DIR. Each team member can reference the shared ruleset directory by setting the $HDS_RULESET_DIR environment variable; refer to "Environment Variables" on page 63 for information.

   Note that instead of step 1.b and 1.c, you can right-click on the *Rulesets* folder and select **Browse to Ruleset Location**.

2. You can also set the new location using Tcl commands within a tcl script referenced by the "-do" or "-tcl" command line arguments:

   ```
   setRulesetLocation <ruleset path>
   ```

   When you invoke HDS with a Tcl script using "-tcl" batch mode, no preferences are saved; on the other hand, on using "-do", the preferences are saved since the script is run before opening in full interactive mode.

Having created and shared rulesets, you can reference them using user-specific or shared policies:

- Each member in the team can have their own policies which reference the shared rulesets. This allows individual control in terms of enabling or disabling certain rules referenced by the policies. In this case, each member will have to set their own default policy and link it to the shared rulesets; for information, refer to "Linking Shared Rulesets to your Own Policies" on page 58.
- Alternatively, all the members within the team have the ability to share the same policies in a similar way to rulesets. For further information, refer to "Sharing Policies" on page 60. In this case, shared policies will be made read-only; hence, members in the team will be prevented from enabling or disabling rules referenced by the policies.

Linking Shared Rulesets to your Own Policies

You can reference shared rulesets from your own policies. This allows you to refer to a common set of rules, but still be able to enable/disable specific rules within your policy. You can create references to shared rulesets in your own policy in the same way as you would for your own rulesets.

Each team member can take these simple steps to link their own policy to shared rulesets:

1. Set the ruleset path through HDL Designer Series (refer to "Sharing RuleSets" on page 57) and then invoke DesignChecker.
2. In the Setup tab, select the *Policies* folder in the Folders pane.
3. Choose **New Policy** from the popup menu.
4. Provide a name for the policy.

5. Click on the top level of the team ruleset hierarchy and then drag and drop it into your new policy.
6. Select the new policy and choose **Set as Default** from the popup menu. This makes your new policy the active policy.

Policies simply contain links to the rulesets. Thus, if anything changes in the rulesets, those changes will be automatically reflected in the policy. For more information on policies, refer to "Working with Policies" in the *DesignChecker User Guide*.

Sharing Policies

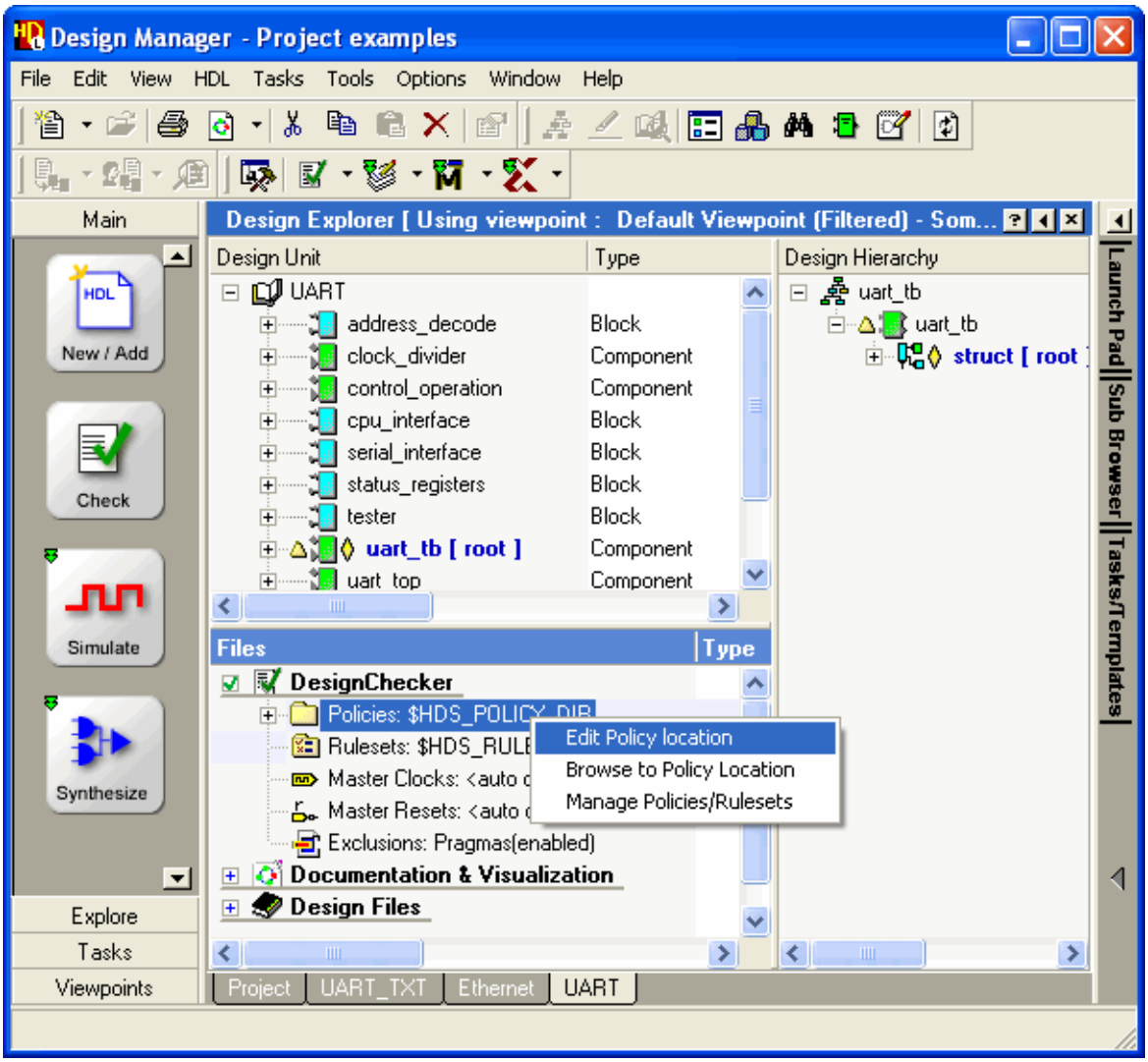
In addition to sharing rulesets, you can also share policies among team members. Like rulesets, the project manager can create policies, set them as read-only, and then copy the policies directory to a central server so that each individual in the team has access to these shared policies. Subsequently, the team members can reference the policies by setting the appropriate path.

All members of the design team can then point to the shared policies directory instead of the default. The policy location can be set in two ways:

1. You can set the policy location directly through the Design Manager, by doing the following:

   a. In the Files view, click on the plus sign to expand the DesignChecker tree.

   b. Right-click on the *Policies* folder and select **Edit Policy Location**.



c. Enter the path to the shared policy directory (you can use environment variables in the pathname, for example, *$SHARED/policies*).

   Alternatively, specify the path as $HDS_POLICY_DIR. Each team member can reference the shared policy directory by setting the $HDS_POLICY_DIR environment variable; refer to "Environment Variables" on page 63 for information.

   Having set the path of the Policies folder, all the policies on that path are loaded and displayed in the DesignChecker tree.

   Note that instead of step 1.b and 1.c, you can right-click on the *Policies* folder and select **Browse to Policy Location**.

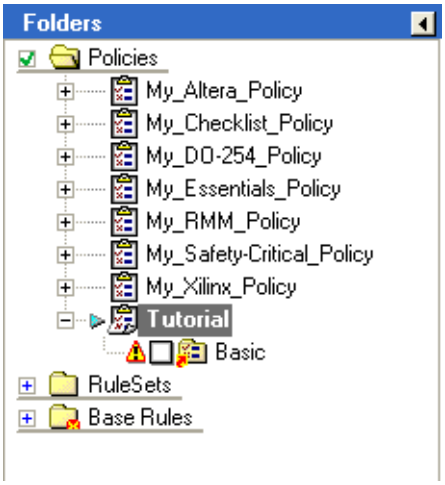
2. You can also set the new location using Tcl commands within a tcl script referenced by the "-do" or "-tcl" command line arguments:

   ```
   setPolicyLocation <policy path>
   ```

   When you invoke HDS with a Tcl script using "-tcl" batch mode, no preferences are saved; on the other hand, on using "-do", the preferences are saved since the script is run before opening in full interactive mode.

It is important to note that to be able to share policies, the team has to keep the locations of both the ruleset and policy in-sync. This is because policies refer to rulesets by name only and expect to find the ruleset within the rulesets' directory.

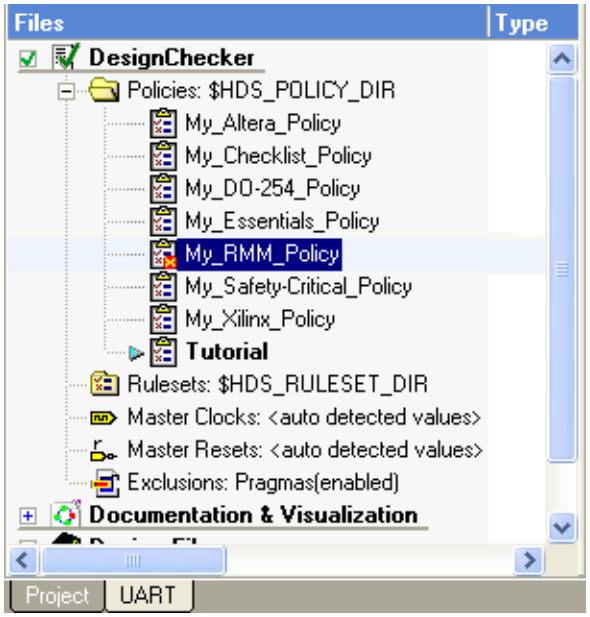
If a policy refers to a ruleset which is not available in the current ruleset directory, the ruleset is flagged with a warning sign in DesignChecker as illustrated below.



Read-Only Policies

DesignChecker supports read-only policies as follows:

- If a policy is read-only, it is marked with the read-only overlay as shown in the below picture; this indicates that no modifications can be applied to this policy. For instance, you cannot change the name of the policy, the names of the rulesets referenced by the policy, or the rulesets referenced by the policy, and so forth.



- If the directory of policies is read-only, a team member becomes unable to create a new policy since it would not be possible to save the new policy.

  On the other hand, if the directory of the policy is writable, each team member has the ability to apply their own modifications to the shared policies.

Environment Variables

HDL Designer Series provides a number of built-in environment variables for several purposes. Refer to the “Environment Variables” chapter in the *HDL Designer Series User Manual* for more information.

- HDS_RULESET_DIR: This environment variable specifies the location of the DesignChecker ruleset directory. This variable can either be explicitly set to force the use of a specific ruleset directory, or it can be used to reference the default ruleset directory which is located in the HDS preferences directory.
  - If you explicitly set this variable, then its value for the new ruleset directory will override the one specified in the preferences. Subsequently, if you unset this variable, the ruleset location will revert to the value stored in the preferences.
  - If you do not explicitly set this variable, then its value will automatically default to the ruleset directory within the HDS preferences directory upon invoking HDL Designer Series.
- HDS_POLICY_DIR: This environment variable specifies the location of the DesignChecker policy directory. This variable can either be explicitly set to force the use of a specific policy directory, or it can be used to reference the default policy directory which is located in the HDS preferences directory.
  - If you explicitly set this variable, then its value for the new policy directory will override the one specified in the preferences. Subsequently, if you unset this variable, the policy location will revert to the value stored in the preferences.
  - If you do not explicitly set this variable, then its value will automatically default to the policy directory within the HDS preferences directory upon invoking HDL Designer Series.
- HDS_USER_HOME: This variable specifies the location of the hds_user directory which contains the user resource files. The default location is in the Application Data folder (Windows) or user directory (UNIX/LINUX) or the location in which user preferences have been stored for the previous release. This variable is ignored if an existing location is specified using the -user_home command line switch.

  The dc_user and dp_user directories are both relative to the location of the hds_user directory. They both follow the hds_user directory and are automatically created in the same directory.

Search Order of RuleSet and Policy Directory

DesignChecker goes through a specific search sequence in order to locate rulesets and policies.

Rulesets are located using the following search order:

1. The location specified by the command *setRulesetLocation <ruleset path>*.
2. The value of the $HDS_RULESET_DIR environment variable, if set.
3. The location in the user preferences, in case it is different from the default. You can use the $HDS_RULESET_DIR environment variable to reference defaults.
4. The default location which is the subdirectory within the HDS user preferences directory ($HDS_USER_HOME/../dc_user/rulesets).

Policies are located using the following search order:

1. The location specified by the command *setPolicyLocation <policy path>*.
2. The value of the $HDS_POLICY_DIR environment variable, if set.
3. The location in the user preferences, in case it is different from the default. You can use the $HDS_POLICY_DIR environment variable to reference defaults.
4. The default location which is the subdirectory within the HDS user preferences directory ($HDS_USER_HOME/../dc_user/policies).

Appendix C
Design Checking Levels

When running a DesignChecker analysis, you have the ability to do that on one of two levels: File level or Design Unit level. Determining which level to use in your analysis depends on the level of depth you are targeting which is consequently reflected on the analysis results.

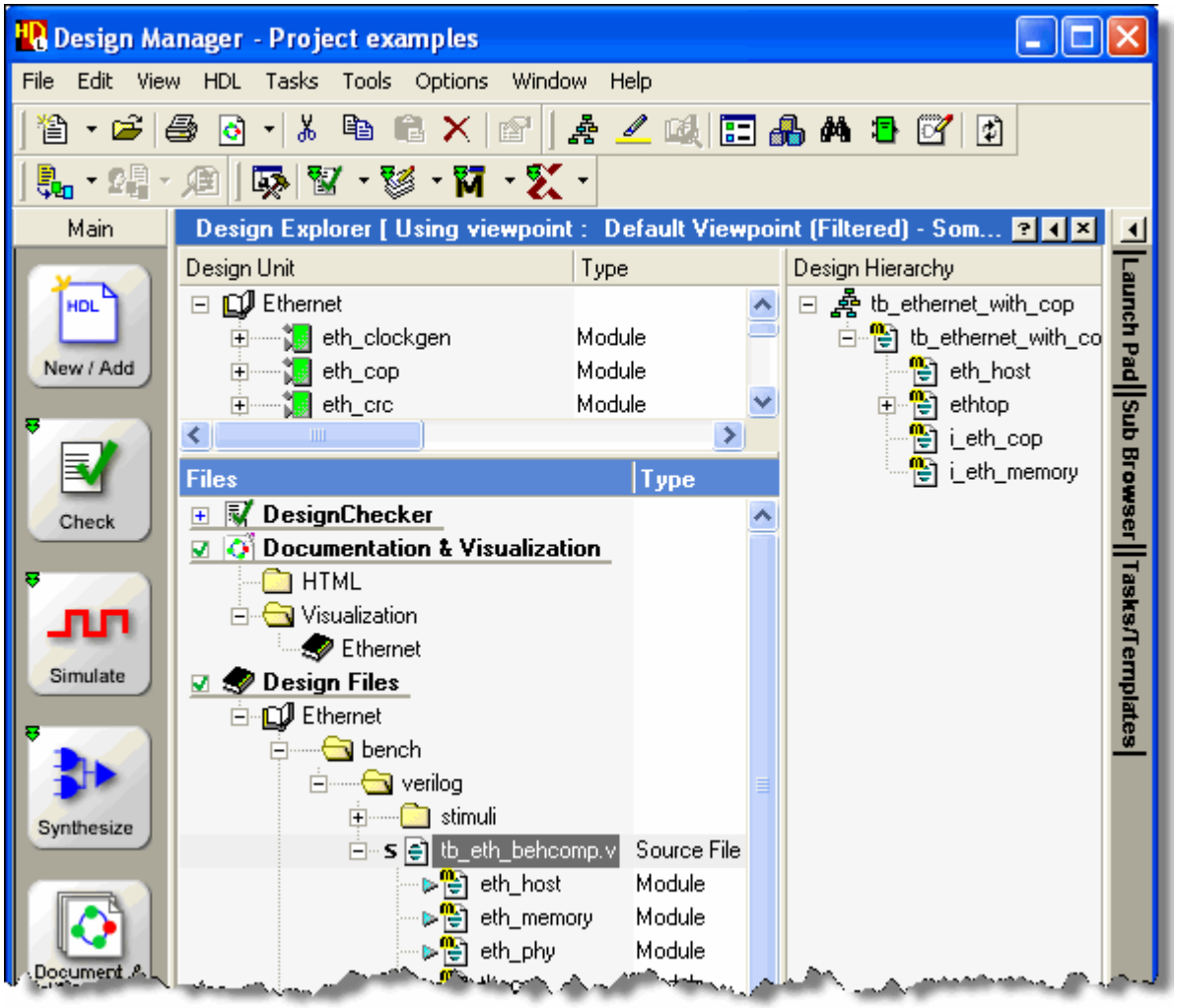
If you run DesignChecker on the level of a source file, the tool analyzes the selected file along with all the design units that pertain to that file. On the other hand, if you run DesignChecker on the level of a design unit, the tool analyzes the specified design unit individually.

File Level Checking

As mentioned earlier, file level checking is performed on a specified file and all the design units included in that file. Follow this procedure to run a file level check.

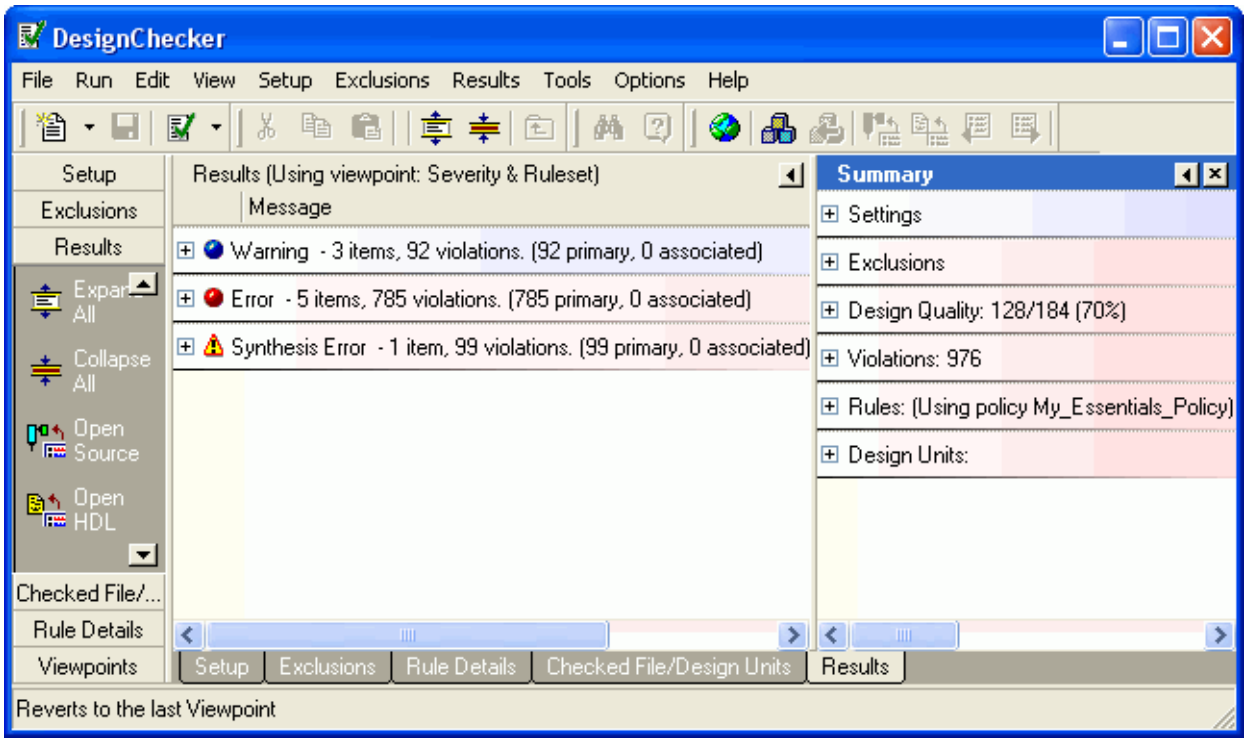
Procedure

1. In the HDL Designer Series design manager, select the file you need to analyze in the Files pane under the Design Files node. For example, you can select *tb_eth_behcomp.v* in the *Ethernet* library.

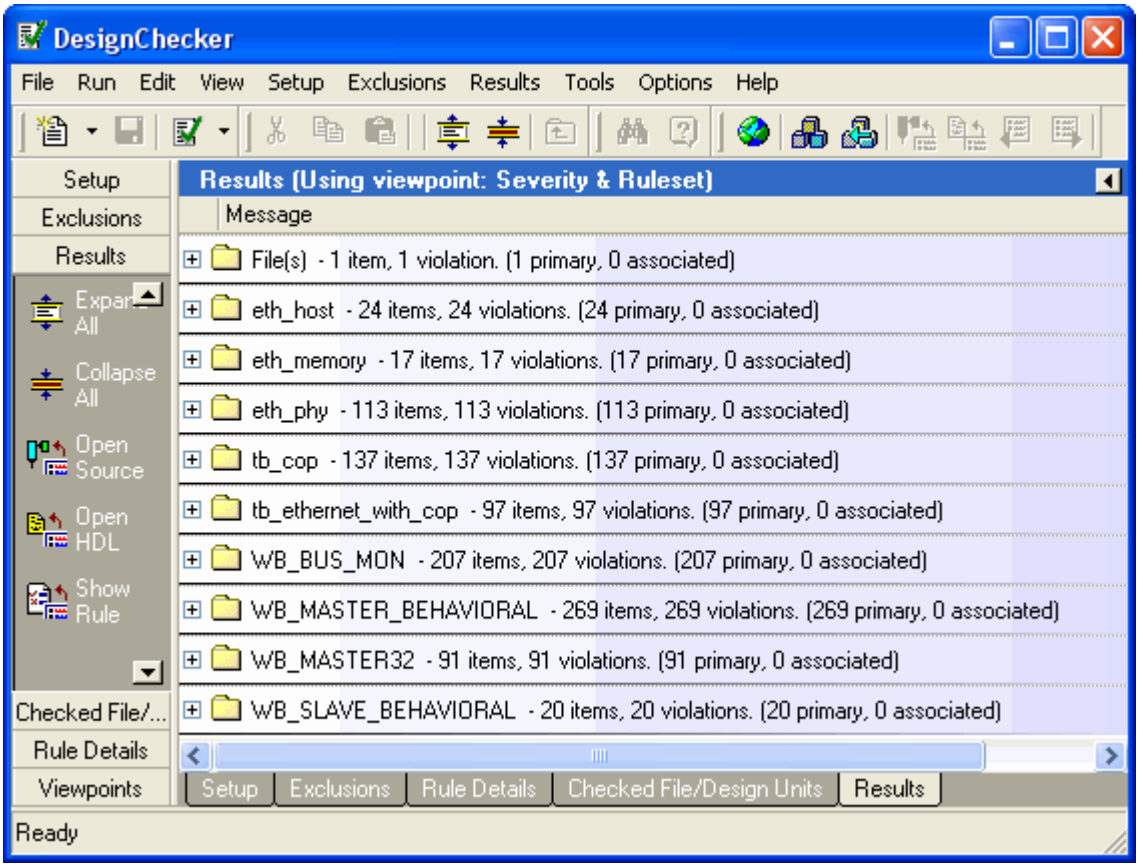


2. In the Main shortcut bar, right-click on the **Check** button and select **Run Single** from the popup menu.

   The DesignChecker runs the analysis on the specified file and the results are displayed as shown in the following figure.

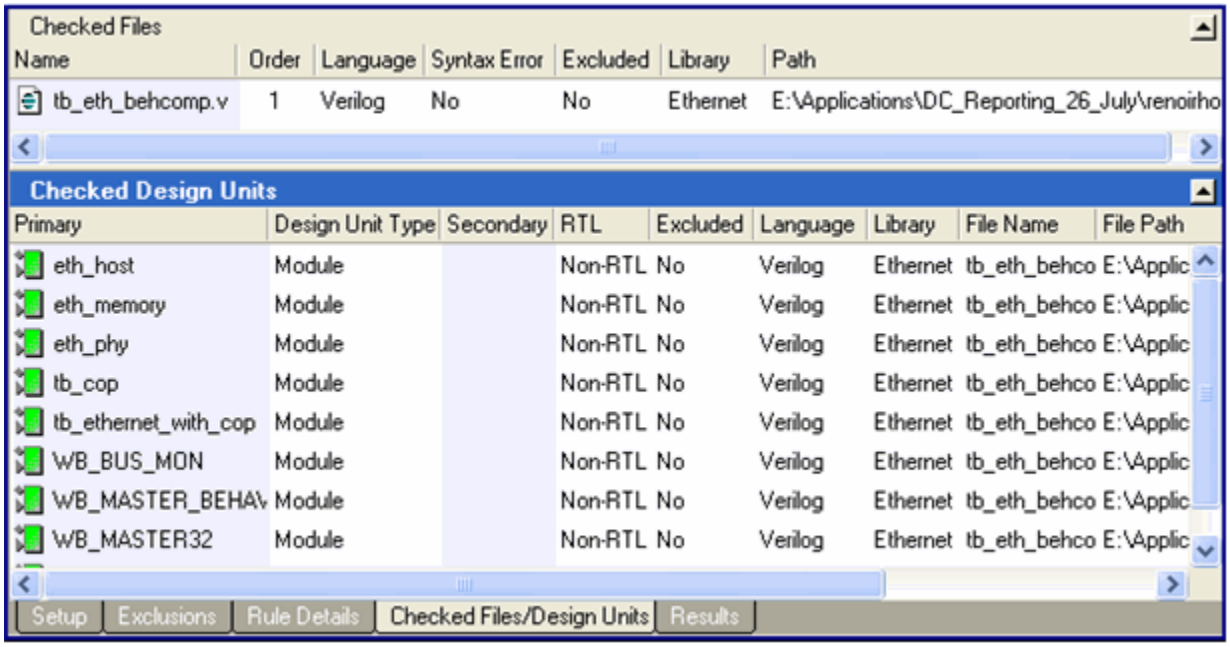


The following figure displays the results grouped by the design units analyzed.



3. Open the **Checked Files/Design Units** tab.

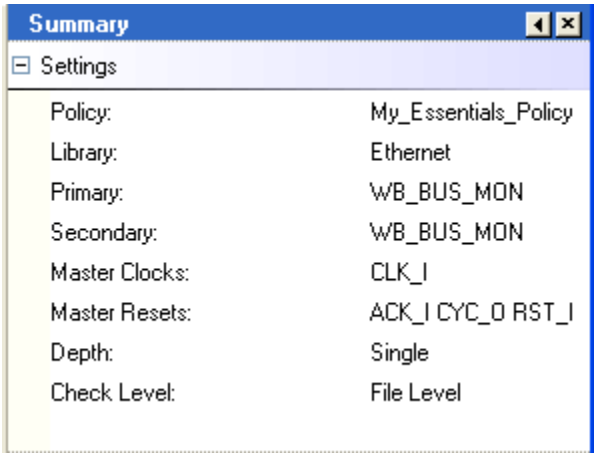
   The Checked Files pane displays information on the file which has undergone the DesignChecker analysis. You will find the file you previously selected in the Design Manager.



   In the Checked Design Units pane, you will find all the design units related to the file as shown in the above figure; these design units have undergone the DesignChecker analysis along with the file.

   As shown in the figure, all the design units belonging to the file *tb_eth_behcomp.v* have been checked: *eth_host*, *eth_memory*, *eth_phy*, *tb_cop*, *tb_ethernet_with_cop*, *WB_BUS_MON*, *WB_MASTER32*, *WB_MASTER_BEHAVIORAL* and *WB_SLAVE_BEHAVIORAL*.

You can identify the type of check level you performed through the Results tab. In the Settings section of the Summary pane, you will find that the Check Level of the analysis is “File Level” as shown in the below figure.

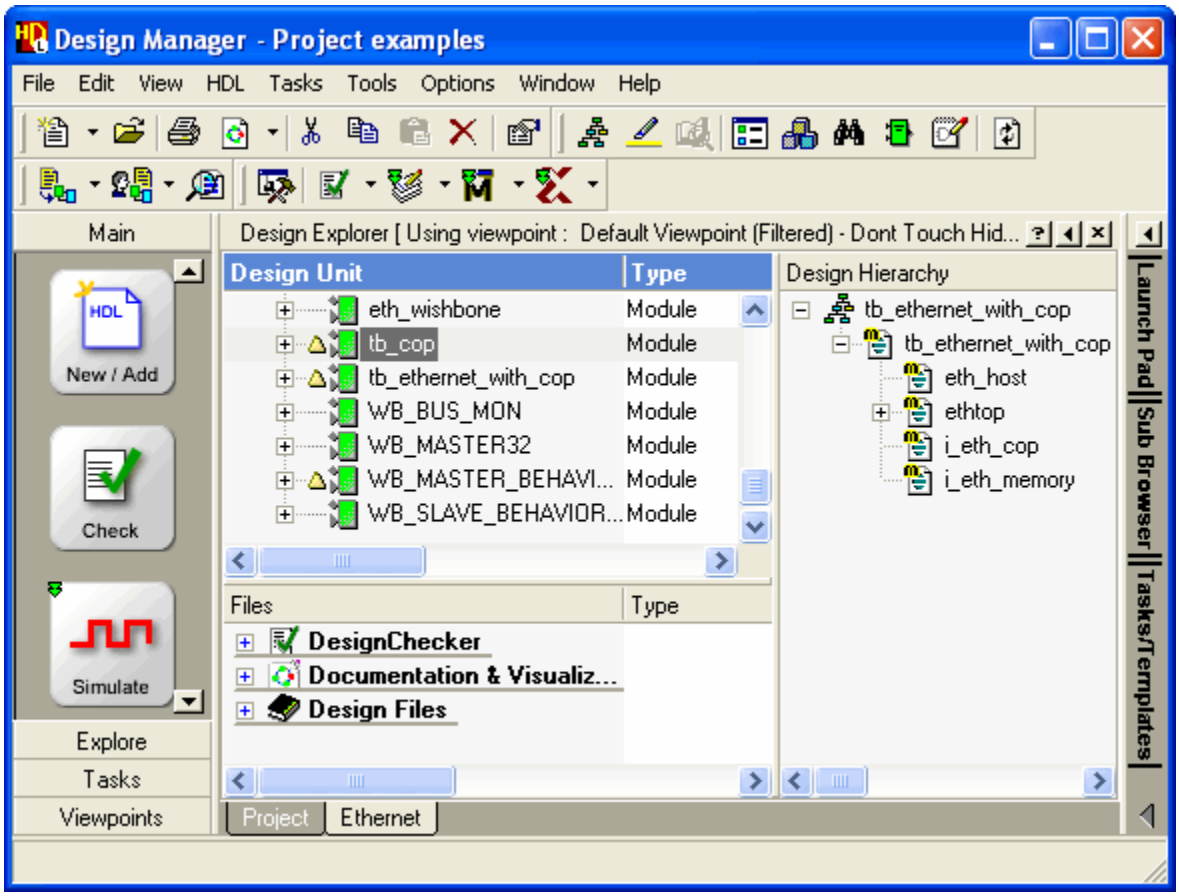


Design Unit Level Checking

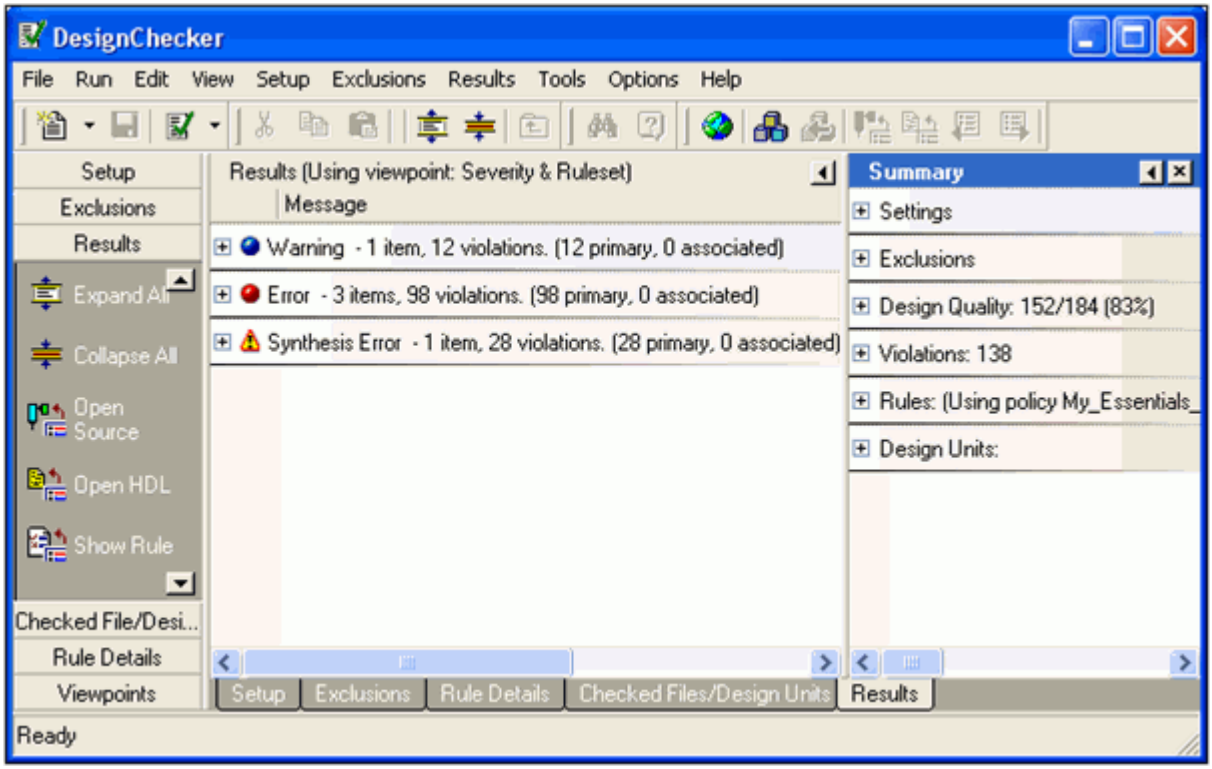
Design unit level checking grants you a deeper level of granularity as it allows you to perform the DesignChecker analysis on an individual design unit. Follow this procedure to run a design unit level check.

Procedure

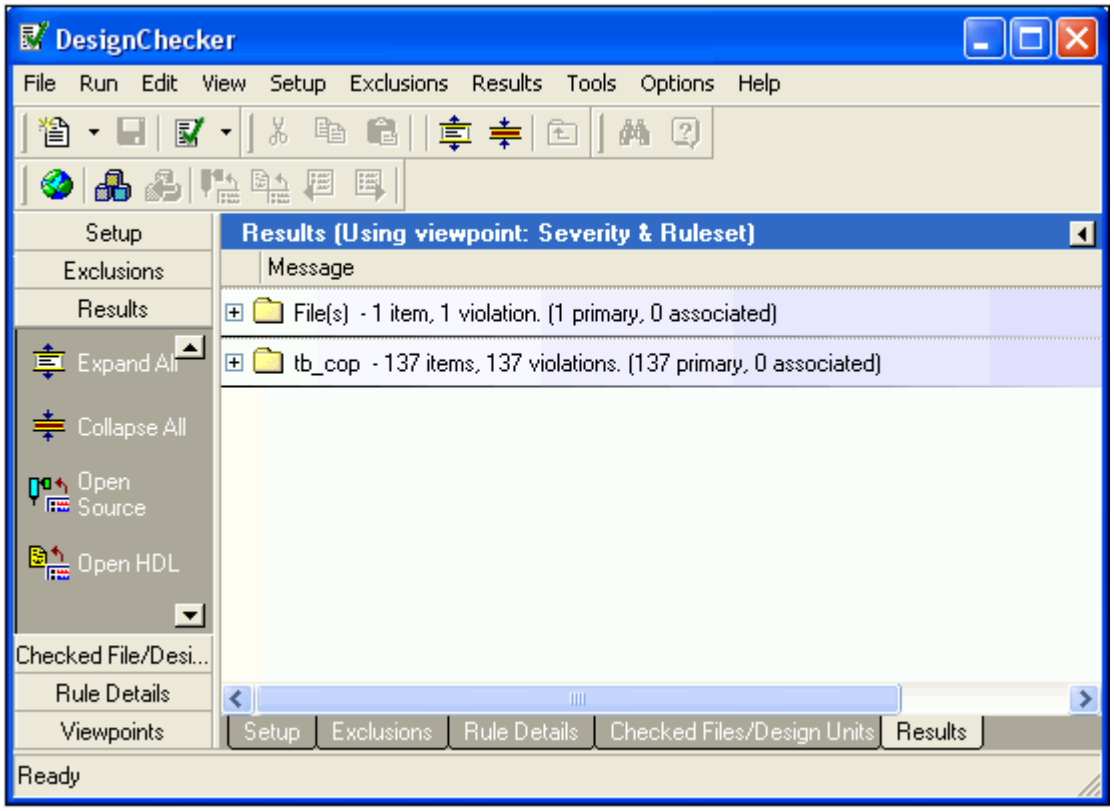
1. In the HDL Designer Series design manager, select the design unit you need to analyze in the Design Units pane. For example, you can select *tb_cop* in the *Ethernet* library.



2. In the Main shortcut bar, right-click on the **Check** button and select **Run Single** from the popup menu.



The following figure displays the results grouped by the design units analyzed.

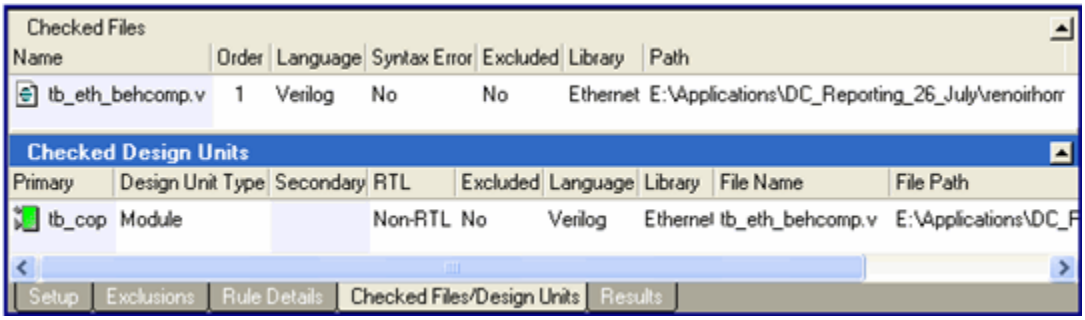


Note

Note that the violations that resulted from analyzing the design unit *tb_cop* are much less than the violations that resulted from analyzing the file *tb_eth_behcomp.v* (refer to "File Level Checking" on page 65). This is because the design unit *tb_cop* is part of the file *tb_eth_behcomp.v*; hence, design level checking allows you to obtain more specific and minimal results. This can be very useful in case of large designs.

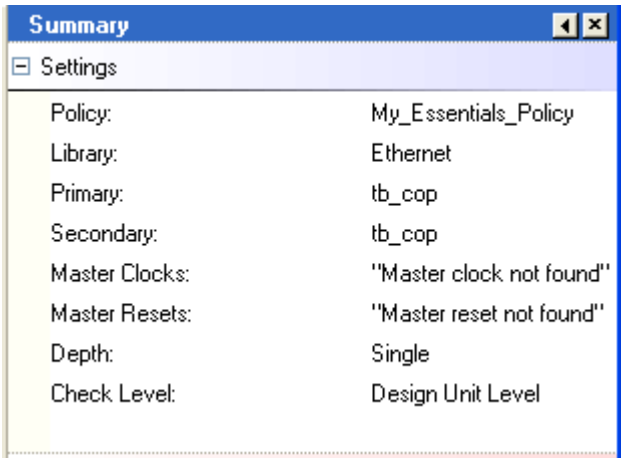
3. Open the **Checked Files/Design Units** tab.

   The Checked Design Units pane displays information on the design unit which has undergone the DesignChecker analysis. You will find the design unit you previously selected in the Design Manager.



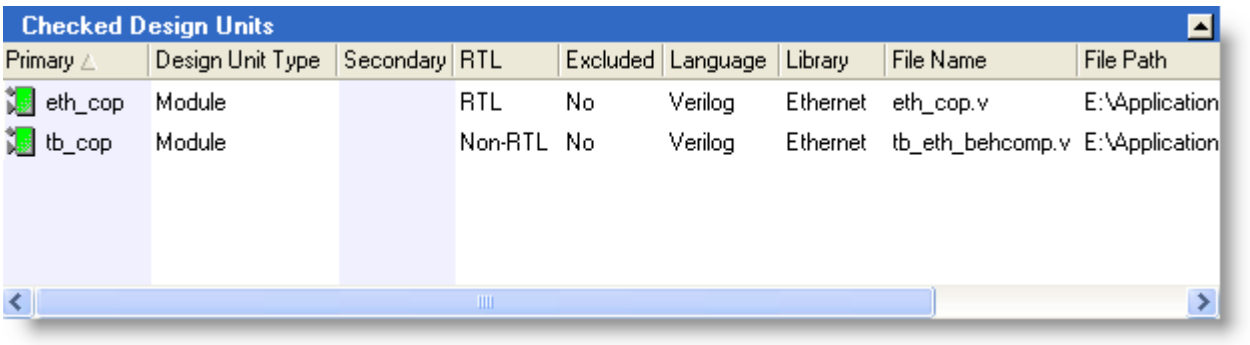
   The Checked Files pane displays the file to which the analyzed design unit belongs. Note that none of the other design units that belong to that file is analyzed.

4. You can identify the type of check level you performed through the Results tab. In the Settings section of the Summary pane, you will find that the Check Level of the analysis is "Design Unit Level" as shown in the below figure.

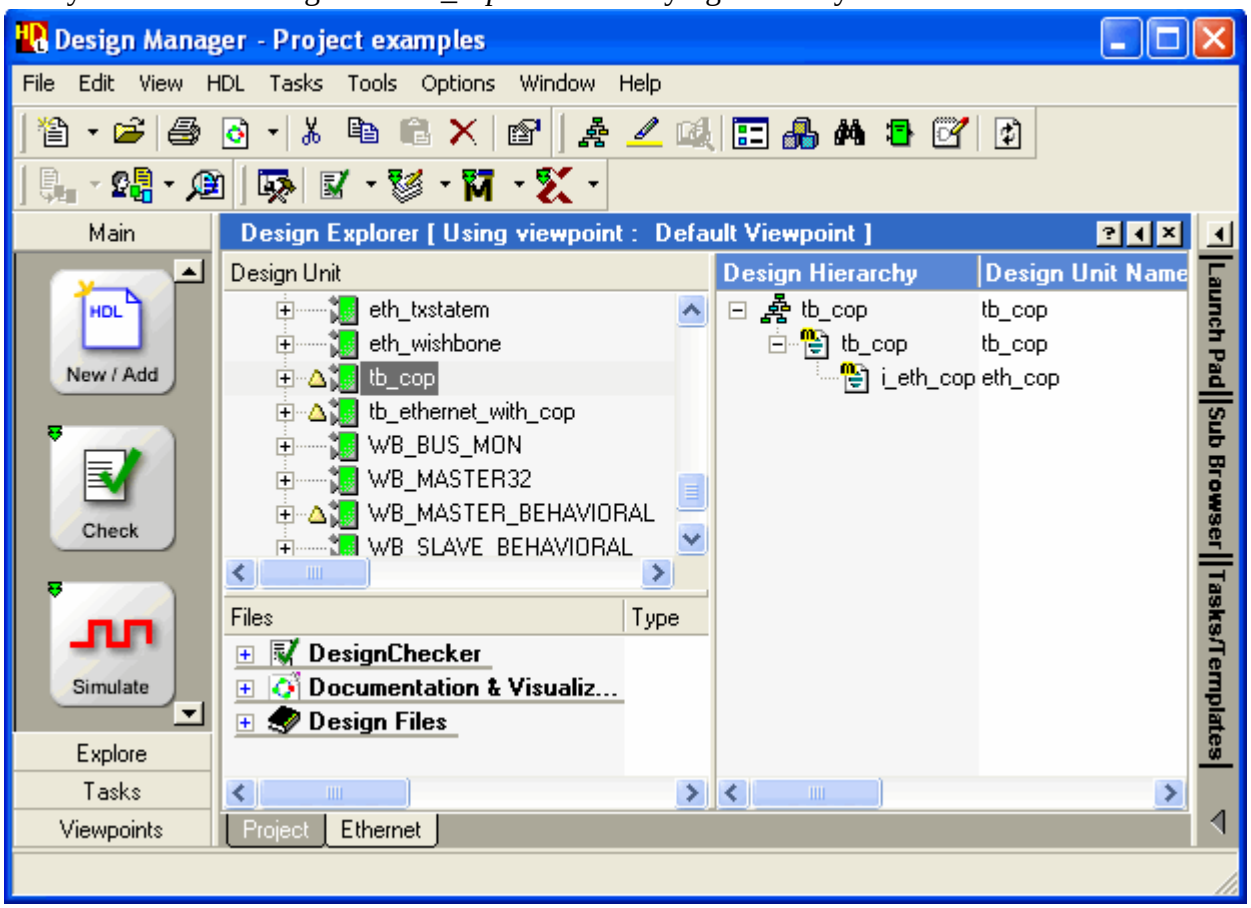


It is important to note that if you run design unit level checking through hierarchy, only the selected design unit in addition to the design units in the underlying hierarchy will be included in the analysis.

For example, if you analyze *tb_cop* through components, only the designs that come below *tb_cop* in the design hierarchy are checked, which is *eth_cop* in that case as shown in the Checked Files/Design Units tab.



Also, by checking the design hierarchy of *tb_cop* in the design manager, you will find that it only includes the design unit *eth_cop* in its underlying hierarchy.



Appendix D
DesignChecker Features Available in HDL Designer Series

Below is a list of the DesignChecker-related features that can be performed directly through HDL Designer Series:

- Invoking DesignChecker
- Setting the Default Policy
- Selecting Files for Analysis (File Level/Design Unit Level)
- Running DesignChecker
- Accessing the DC_constraints file
- Setting Pragma Exclusions
- Setting Black Box Exclusions
- Setting Don’t Touch Exclusions
- Sharing Rulesets and Policies

Index

— S —

End-User License Agreement

The latest version of the End-User License Agreement is available on-line at:
www.mentor.com/eula

IMPORTANT INFORMATION

USE OF ALL SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF SOFTWARE INDICATES CUSTOMER'S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

END-USER LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Software (as defined in Section 2) and hardware (collectively "Products") between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Software received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

1. **ORDERS, FEES AND PAYMENT.**

   1.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement ("Order(s)"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement, any applicable addenda and the applicable quotation, whether or not these documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order will not be effective unless agreed in writing by an authorized representative of Customer and Mentor Graphics.

   1.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice(s). Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

   1.3. All Products are delivered FCA factory (Incoterms 2000), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics retains a security interest in all Products delivered under this Agreement, to secure payment of the purchase price of such Products, and Customer agrees to sign any documents that Mentor Graphics determines to be necessary or convenient for use in filing or perfecting such security interest. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

2. **GRANT OF LICENSE.** The software installed, downloaded, or otherwise acquired by Customer under this Agreement, including any updates, modifications, revisions, copies, documentation and design data ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form (except as provided in Subsection 5.2); (b) for Customer's internal business purposes; (c) for the term of the license; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Customer may have Software temporarily used by an employee for telecommuting purposes from locations other than a Customer office, such as the employee's residence, an airport or hotel, provided that such employee's primary place of employment is the site where the Software is authorized for use. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions. For the avoidance of doubt, if Customer requests any change or enhancement to Software, whether in the course of

receiving support or consulting services, evaluating Software, performing beta testing or otherwise, any inventions, product improvements, modifications or developments made by Mentor Graphics (at Mentor Graphics' sole discretion) will be the exclusive property of Mentor Graphics.

3. **ESC SOFTWARE.** If Customer purchases a license to use development or prototyping tools of Mentor Graphics' Embedded Software Channel ("ESC"), Mentor Graphics grants to Customer a nontransferable, nonexclusive license to reproduce and distribute executable files created using ESC compilers, including the ESC run-time libraries distributed with ESC C and C++ compiler Software that are linked into a composite program as an integral part of Customer's compiled computer program, provided that Customer distributes these files only in conjunction with Customer's compiled computer program. Mentor Graphics does NOT grant Customer any right to duplicate, incorporate or embed copies of Mentor Graphics' real-time operating systems or other embedded software products into Customer's products or applications without first signing or otherwise agreeing to a separate agreement with Mentor Graphics for such purpose.

4. **BETA CODE.**

4.1. Portions or all of certain Software may contain code for experimental testing and evaluation ("Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and Customer's use of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form.

4.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer's use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer's evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

4.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer's feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 4.3 shall survive termination of this Agreement.

5. **RESTRICTIONS ON USE.**

5.1. Customer may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Customer shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer's employees and on-site contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use it except as permitted by this Agreement. Customer shall give Mentor Graphics written notice of any unauthorized disclosure or use of the Products as soon as Customer learns or becomes aware of such unauthorized disclosure or use. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive any source code from Software. Log files, data files, rule files and script files generated by or for the Software (collectively "Files"), including without limitation files containing Standard Verification Rule Format ("SVRF") and Tcl Verification Format ("TVF") which are Mentor Graphics' proprietary syntaxes for expressing process rules, constitute or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Customer may use Files containing SVRF or TVF only with Mentor Graphics products. Under no circumstances shall Customer use Software or Files or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Software, or disclose to any third party the results of, or information pertaining to, any benchmark.

5.2. If any Software or portions thereof are provided in source code form, Customer will use the source code only to correct software errors and enhance or modify the Software for the authorized use. Customer shall not disclose or permit disclosure of source code, in whole or in part, including any of its methods or concepts, to anyone except Customer's employees or contractors, excluding Mentor Graphics competitors, with a need to know. Customer shall not copy or compile source code in any manner except to support this authorized use.

5.3. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense or otherwise transfer the Products, whether by operation of law or otherwise ("Attempted Transfer"), without Mentor Graphics' prior written consent and payment of Mentor Graphics' then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms

of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer's permitted successors in interest and assigns.

5.4. The provisions of this Section 5 shall survive the termination of this Agreement.

6. **SUPPORT SERVICES.** To the extent Customer purchases support services, Mentor Graphics will provide Customer updates and technical support for the Products, at the Customer site(s) for which support is purchased, in accordance with Mentor Graphics' then current End-User Support Terms located at http://supportnet.mentor.com/about/legal/.

7. **AUTOMATIC CHECK FOR UPDATES; PRIVACY.** Technological measures in Software may communicate with servers of Mentor Graphics or its contractors for the purpose of checking for and notifying the user of updates and to ensure that the Software in use is licensed in compliance with this Agreement. Mentor Graphics will not collect any personally identifiable data in this process and will not disclose any data collected to any third party without the prior written consent of Customer, except to Mentor Graphics' outside attorneys or as may be required by a court of competent jurisdiction.

8. **LIMITED WARRANTY.**

8.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Software under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes or alternate Software under a transaction involving Software re-mix. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification or improper installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY, PROVIDED CUSTOMER HAS OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) PRODUCTS PROVIDED AT NO CHARGE; OR (C) BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

8.2. THE WARRANTIES SET FORTH IN THIS SECTION 8 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO PRODUCTS PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

9. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT RECEIVED FROM CUSTOMER FOR THE HARDWARE, SOFTWARE LICENSE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 9 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

10. **HAZARDOUS APPLICATIONS.** CUSTOMER ACKNOWLEDGES IT IS SOLELY RESPONSIBLE FOR TESTING ITS PRODUCTS USED IN APPLICATIONS WHERE THE FAILURE OR INACCURACY OF ITS PRODUCTS MIGHT RESULT IN DEATH OR PERSONAL INJURY ("HAZARDOUS APPLICATIONS"). NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF MENTOR GRAPHICS PRODUCTS IN OR FOR HAZARDOUS APPLICATIONS. THE PROVISIONS OF THIS SECTION 10 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

11. **INDEMNIFICATION.** CUSTOMER AGREES TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH THE USE OF PRODUCTS AS DESCRIBED IN SECTION 10. THE PROVISIONS OF THIS SECTION 11 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

12. **INFRINGEMENT.**

12.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay costs and damages finally awarded against Customer that are attributable to the action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance

to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

12.2. If a claim is made under Subsection 12.1 Mentor Graphics may, at its option and expense, (a) replace or modify the Product so that it becomes noninfringing; (b) procure for Customer the right to continue using the Product; or (c) require the return of the Product and refund to Customer any purchase price or license fee paid, less a reasonable allowance for use.

12.3. Mentor Graphics has no liability to Customer if the action is based upon: (a) the combination of Software or hardware with any product not furnished by Mentor Graphics; (b) the modification of the Product other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells; (f) any Beta Code or Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by Customer that is deemed willful. In the case of (h), Customer shall reimburse Mentor Graphics for its reasonable attorney fees and other costs related to the action.

12.4. THIS SECTION 12 IS SUBJECT TO SECTION 9 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS FOR DEFENSE, SETTLEMENT AND DAMAGES, AND CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

13. **TERMINATION AND EFFECT OF TERMINATION.** If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized term.

13.1. Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement immediately upon written notice if Customer: (a) exceeds the scope of the license or otherwise fails to comply with the licensing or confidentiality provisions of this Agreement, or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. For any other material breach of any provision of this Agreement, Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement upon 30 days written notice if Customer fails to cure the breach within the 30 day notice period. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination.

13.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination, Customer shall ensure that all use of the affected Products ceases, and shall return hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form.

14. **EXPORT.** The Products provided hereunder are subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products and information about the products to certain countries and certain persons. Customer agrees that it will not export Products in any manner without first obtaining all necessary approval from appropriate local and United States government agencies.

15. **U.S. GOVERNMENT LICENSE RIGHTS.** Software was developed entirely at private expense. All Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to US FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in this Agreement, except for provisions which are contrary to applicable mandatory federal laws.

16. **THIRD PARTY BENEFICIARY.** Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Microsoft Corporation and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

17. **REVIEW OF LICENSE USAGE.** Customer will monitor the access to and use of Software. With prior written notice and during Customer's normal business hours, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system and records deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FLEXlm or FLEXnet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. The provisions of this Section 17 shall survive the termination of this Agreement.

18. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** The owners of certain Mentor Graphics intellectual property licensed under this Agreement are located in Ireland and the United States. To promote consistency around the world, disputes shall be resolved as follows: excluding conflict of laws rules, this Agreement shall be governed by and construed under the laws of the State of Oregon, USA, if Customer is located in North or South America, and the laws of Ireland if Customer is located outside of North or South America. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the courts of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply. Notwithstanding the foregoing, all disputes in Asia arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the chairman of the Singapore International

Arbitration Centre ("SIAC") to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section. This section shall not restrict Mentor Graphics' right to bring an action against Customer in the jurisdiction where Customer's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

19. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

20. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements, including but not limited to any purchase order terms and conditions. Some Software may contain code distributed under a third party license agreement that may provide additional rights to Customer. Please see the applicable Software documentation for details. This Agreement may only be modified in writing by authorized representatives of the parties. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 100615, Part No. 246066