SoC Designer
Runtime User Guide

Copyright © 2016 ARM Limited. All rights reserved.

Release Information

The following changes have been made to this document.

Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Issue</th>
<th>Confidentiality</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2016</td>
<td>A</td>
<td>Non-Confidential</td>
<td>Rebrand update for 9.0.0</td>
</tr>
</tbody>
</table>

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of ARM Limited (“ARM”). **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any patents.

**THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT.** For the avoidance of doubt, ARM makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version shall prevail.

**TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.**

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to ARM’s customers is not intended to create or refer to any partnership relationship with any other company. ARM may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any signed written agreement specifically covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms.
Contents

Preface
About this document ........................................................................................................ viii

Chapter 1  Introduction
1.1 Introduction to SoC Designer ............................................................................... 1-2
1.2 SoC Designer Simulator ..................................................................................... 1-3

Chapter 2  SoC Designer Runtime Simulator Reference
2.1 Starting SoC Designer Simulator ........................................................................ 2-2
2.2 Application window .......................................................................................... 2-11
2.3 Tools windows .................................................................................................. 2-29
2.4 Preferences dialog ............................................................................................ 2-34
2.5 Memory Map Editor .......................................................................................... 2-42
2.6 Breakpoint windows ......................................................................................... 2-43
2.7 Monitor windows ............................................................................................... 2-48
2.8 Trace Manager and the Waveform Viewer windows ..................................... 2-51
2.9 Profiling windows ............................................................................................ 2-62
2.10 Software analysis ............................................................................................. 2-84

Chapter 3  Batch Mode Simulation
3.1 Introduction ........................................................................................................ 3-2
3.2 Scripted batch mode simulation ........................................................................ 3-3

Chapter 4  Debugging
4.1 Introduction ........................................................................................................ 4-2
4.2 Component level debugging ............................................................................. 4-3
4.3 System level debugging and probes ............................................................. 4-14

Appendix A  Keyboard Shortcuts
A.1 SoC Designer Simulator shortcuts ................................................................... A-2
Appendix B  Third-Party Software
B.1  Verilog Parser ................................................................. B-2

Appendix C  SoC Designer Component Library Configuration File
C.1  Configuration file contents .............................................. C-2

Glossary
Preface

This preface introduces the SoC Designer Runtime. It contains the following sections:

- *About this document* on page viii
About this document

This document describes how to set up and use the SoC Designer Runtime.

Intended audience

This document has been written for experienced hardware and software developers to create SoC Designer systems from components delivered by ARM (for example ARM cores, peripherals, or buses) or third parties (for example, DSP cores) that have been previously created by modeling development engineers.

SoC Designer is SystemC-based, but strong SystemC or C++ expertise is not required to use SoC Designer. Users must be familiar with the basic concepts of SystemC (such as `sc_module` and `sc_port`) and basic concepts of C++ (such as classes and inheritance).

Note

This document focuses only on the general use of the SoC Designer Runtime to run simulations on the system and the debugging of systems.

The *SoC Designer User Guide* (ARM DUI 0956) supplied with the full SoC Designer suite describes the simulation methodology and the model creation tools.

Organization

This document is organized into the following chapters:

**Chapter 1 Introduction**

This chapter gives an introduction to SoC Designer Runtime.

**Chapter 2 SoC Designer Runtime Simulator Reference**

This chapter describes the SoC Designer Runtime Simulator application and runtime windows.

**Chapter 3 Batch Mode Simulation**

This chapter is the reference for batch mode simulation and RTL Generation.

**Chapter 4 Debugging**

Read this chapter for additional information on how to debug components and systems.

**Appendix A Keyboard Shortcuts**

Read this appendix for a list of keyboard shortcuts.
Appendix B *Third-Party Software*

Read this appendix for a list of third-party software used by the SoC Designer Runtime.

Appendix C *SoC Designer Component Library Configuration File*

Read this appendix for additional information on the configuration file.

**Glossary** on page Glossary-1

Read this glossary for a list of special terms that apply to products made by ARM.
Conventions

Conventions that this guide can use are described in:
• **Typographical**
• **Signals**
• **Numbering** on page xi

**Typographical**

The typographical conventions are:

- **italic** Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
- **bold** Highlights interface elements, such as menu names. Denotes processor signal names. Also used for terms in descriptive lists, where appropriate.
- **monospace** Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.
- **monospace** Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
- **monospace italic** Denotes arguments to commands and functions where the argument is to be replaced by a specific value.
- **monospace bold** Denotes language keywords when used outside example code.
- `< and >` Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example:

  ```
  MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>
  ```

**Signals**

The signal conventions are:

- **Signal level** The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
  • **HIGH** for active-HIGH signals
  • **LOW** for active-LOW signals.
- **Lower-case n** At the start or end of a signal name denotes an active-LOW signal.
- **Prefix A** Denotes global *Advanced eXtensible Interface* (AXI) signals.
Prefix AR  Denotes AXI read address channel signals.
Prefix AW  Denotes AXI write address channel signals.
Prefix B   Denotes AXI write response channel signals.
Prefix C   Denotes AXI low-power interface signals.
Prefix H   Denotes Advanced High-performance Bus (AHB) signals.
Prefix P   Denotes Advanced Peripheral Bus (APB) signals.
Prefix R   Denotes AXI read data channel signals.
Prefix W   Denotes AXI write data channel signals.

Numbering

The numbering convention is:

<size in bits>'<base>'<number>

This is a Verilog method of abbreviating constant numbers. For example:
- 'h7B4 is an unsized hexadecimal value.
- 'o7654 is an unsized octal value.
- 8'd9 is an eight-bit wide decimal value of 9.
- 8'h3F is an eight-bit wide hexadecimal value of 0x3F. This is equivalent to b00111111.
- 8'b1111 is an eight-bit wide binary value of b00001111.

Terminology

The table below lists the equivalent SoC Designer terms corresponding to the SystemC terms used in the SystemC environment:

<table>
<thead>
<tr>
<th>SystemC term</th>
<th>SoC Designer term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>component</td>
<td>The models for individual devices. For example, CPU core, memory, bus interface, and I/O.</td>
</tr>
<tr>
<td>port</td>
<td>master port</td>
<td>A port that generates transactions or signals.</td>
</tr>
<tr>
<td>channel</td>
<td>slave port</td>
<td>SystemC channels are also known as sc_export.</td>
</tr>
</tbody>
</table>
Further reading

This section lists related publications:

- SoC Designer User Guide
- SoC Designer Tools API Reference Manual
- SoC Designer Runtime User Guide
- SoC Designer Installation Guide
- ESL API Developer's Guide
- SoC Designer SystemC Linking Guide
- MxScript Reference Manual
- SoC Designer CDP HDL Cosimulation Guide

The following publications provide reference information about ARM*-related architecture:

  See http://infocenter.arm.com/help/index.jsp for access to ARM documentation.

The following publications provide additional information on simulation:

Chapter 1

Introduction

This chapter provides an overview of the SoC Designer Runtime. It contains the following sections:

- *Introduction to SoC Designer* on page 1-2
- *SoC Designer Simulator* on page 1-3
1.1 Introduction to SoC Designer

SoC Designer products provide a simulation environment for fast simulation of integrated systems-on-chip with multiple cores, peripherals and memories using C++. The cycle-based scheduler and the transaction-based component interfaces enable high simulation speed while retaining full accuracy.

SoC Designer systems can be used as standalone simulation models or integrated into HW simulation or HW/SW co-simulation tools. Third-party debuggers are easily attached to one or more targets within the SoC Designer systems for full visibility of component details.

SoC Designer Runtime is a graphical application that enables the user to simulate, analyze, and debug SoC Designer systems. It contains a debug server for attaching and interacting with third-party debuggers.
1.2 SoC Designer Simulator

SoC Designer Simulator is a graphical application that enables you to simulate and debug SoC Designer systems. It also provides the ability to look into the hierarchy of the system and modify parameters for the sub-components of a system. Simulated systems can be saved off to a file for repetitive use. SoC Designer Simulator also provides extensive system and component level debugging features.

1.2.1 Simulation Features

SoC Designer Simulator supports the following simulation features:
• controlling the simulation: run, step, stop, reset
• viewing the system structure and hierarchy
• viewing and modifying the parameters of a system
• viewing registers, memories, and disassembly through the CADI interface
• setting breakpoints on registers, memories, disassembly, transactions, and signals
• monitoring of transactions and signals
• viewing signal or transaction activity with the waveform viewer
• collecting statistical information with the profiling analysis windows
• launching and controlling of external debuggers
• automating a simulation in batch mode with scripting support

The basic steps to run a simulation are:
1. Load a simulation file (see Opening a simulation on page 2-6).
2. Optionally attach monitor probes to the system (see System level debugging and probes on page 4-14).
3. Run the simulation and observe the results in the runtime windows (see Monitor windows on page 2-48, Trace Manager and the Waveform Viewer windows on page 2-51, and Profiling windows on page 2-62).
1.2.2 Window layout

Table 1-1 describes the graphical user interface elements in the layout of the SoC Designer Simulator:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Menu</td>
<td>The main menu contains the available options with their corresponding keyboard short cuts.</td>
</tr>
<tr>
<td>Toolbar</td>
<td>The toolbar contains buttons to frequently used features.</td>
</tr>
<tr>
<td>Output window</td>
<td>The output window displays information about the state of the simulation and messages sent from the system and its sub-components.</td>
</tr>
<tr>
<td>Command Line</td>
<td>The command line enables the user to enter simulation commands from the keyboard.</td>
</tr>
<tr>
<td>Task Bar</td>
<td>The task bar appears as soon as one or more debug or profiling windows are opened. It shows icons for all open windows.</td>
</tr>
<tr>
<td>Status Bar</td>
<td>The status bar displays information about menu items, commands, buttons, and component information.</td>
</tr>
</tbody>
</table>

Figure 1-1 on page 1-5 shows the SoC Designer Simulator application window.
1.2.3 The runtime windows

There are many debug and profile windows available for the different aspects of simulation and system analysis. These windows provide a high degree of visibility into system details and enable debugging and profiling of systems.
Switching between the diagram and runtime windows

Use the SoC Designer task bar to manage the open windows.

All component-related debug windows are color-coded to identify the component that the window belongs to. The i field of a component is assigned a color when the debug or profiling window opens.
To temporarily minimize all runtime windows, use the context menu of the taskbar and select **Show Diagram** or press **Ctrl-D** on the keyboard. Repeating the action restores the runtime windows.

To switch between the open runtime windows and select a different window, click on the window or type **Ctrl-Tab**.

**Saving/restoring windows between simulation sessions**

If a simulation is saved into an MXS file (by using the **Save** feature in SoC Designer Simulator, the state of the open runtimes windows is recorded and the windows are reopened in the same state when the simulation is reloaded.
This chapter provides a reference for SoC Designer Runtime Simulator. It contains the following sections:

- *Starting SoC Designer Simulator* on page 2-2
- *Application window* on page 2-11
- *Tools windows* on page 2-29
- *Preferences dialog* on page 2-34
- *Memory Map Editor* on page 2-42
- *Breakpoint windows* on page 2-43
- *Monitor windows* on page 2-48
- *Trace Manager and the Waveform Viewer windows* on page 2-51
- *Profiling windows* on page 2-62
- *Software analysis* on page 2-84
2.1 Starting SoC Designer Simulator

You can invoke SoC Designer Simulator directly to load and simulate an existing system:

**Windows**

There are several ways to invoke SoC Designer Simulator on Windows:

- Click the **Start** menu, navigate to the SoC Designer menu, and choose the Release version or the Debug version of SoC Designer.
- Double-click the **Release** or **Debug** SoC Designer Simulator icons.
- Drag a SoC Designer file type over one of the icons.
- Double-click an **MXS**, **MXR**, or **MXE** file, after setting up file associations (release versions only).
- Select **Run** from the **Start** menu and type the path and name in the displayed dialog.

For example, for SoC Designer version 9.0.0, enter:

```plaintext
C:\Program Files\ARM\SoCDesigner9.0.0\Bin\Release\SDRuntime.bat
```

If the **MAXSIM_HOME** variable is set, you can also enter this as:

```plaintext
%MAXSIM_HOME%\Bin\Release\SDRuntime.bat
```

**Linux**

On Linux platforms you can invoke SoC Designer Simulator from the console with the following command line:

```plaintext
> sdsimruntime options filename
```

*filename*, if used, can see an **MXP** or **MXE** system, an **MXS** or **MXR** simulation, or an **MXSCR** script file.
2.1.1 Command line options

SoC Designer Simulator supports the command line options as listed in Table 2-1.

In cases where the option can be used on both Linux and Windows, the Windows command `SDRuntime.bat` is used. In cases where the option can be used only on Linux, the Linux `sdsimruntime` command is used.

Note: Both the `sdsimruntime` and `SDRuntime.bat` commands invoke `sdsim -X`, so these commands and `sdsim -X` can be used interchangeably.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-u / --plus</td>
<td><code>sdsimruntime -u / --plus</code></td>
<td>Invoke SoCD Runtime Only with the plus features and libraries enabled</td>
</tr>
</tbody>
</table>
| filename, loaded with options | `sdsimruntime <options>`
root_filename.mxe
sdsimruntime <options>
root_filename.mxp
sdsimruntime <options>
root_filename.mxr
sdsimruntime <options>
root_filename.mxs
sdsimruntime <options>
root_filename.mxscr | Name of file to load. The file must be one of the following file types: `mxe / mxp / mxr / mxs / mxscr` |
| -dbg               | `sdsimruntime -dbg`                      | This option specifies that you want to run the Debug version of the tool. Using this command without the `-debugger=` option prompts you to enter the command to start the debugger that you wish to use. You may also run the debug version without a debugger. (Supported in Linux, only.) |
| -debugger=         | `sdsimruntime -dbg -debugger=none`
`sdsimruntime -dbg -debugger=ddd` | This option is to be used in conjunction with the above option. Use this option to specify a debugger on the command line (in the example shown, the debugger name is ddd), or to specify that you want to use the debug version without a debugger. (Supported in Linux, only.) |
| -h                 | `sdsimruntime -h / --help`               | This prints the command line help with a list of options.                                                                                     |
This option causes SoC Designer to enter batch mode and execute the supplied script file in GUI mode. The GUI is under control of a script, but the program remains in batch mode.

**Note**
This command must be run in the path where the mxp/mxs system resides.

This option can be used along with specifying a simulation file to run the simulation for the specified number of cycles. SoC Designer Simulator starts in batch mode (no GUI) and all messages are directed to stdout.

**Note**
These command line options cannot be combined with the `--script` option that specifies a script file.

This option has the same effect as `--s`, except that the simulation is run indefinitely.

**Caution**
If you use this option, the only way to shutdown the simulation is to kill the process.

This option is for batch mode only. Specifying a log file causes all output to stdout and stderr to be redirected to that file. This option is most useful on Windows where output redirection is not possible.
### Table 2-1 Command line options (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-t</code> or <code>-statefile filename</code></td>
<td><code>sdsimruntime mxSys.mxs -r -t statefile.bin</code></td>
<td>This option is for batch mode only. Specifying a state file causes SoC Designer Simulator to save the simulation state (MXR) file to the name specified, overwriting a potentially existing file with the same name. The <code>.mxr</code> file ending is added if omitted.</td>
</tr>
<tr>
<td><code>-l</code> or <code>-script filename.mxscr</code></td>
<td><code>sdsimruntime --script myscript.mxscr</code></td>
<td>Starts the simulator and executes it in batch mode. <strong>Note</strong> These command line options cannot be combined with command-line options like <code>-s 1000</code> that control simulation execution.</td>
</tr>
<tr>
<td><code>-b</code> or <code>-maxlib filename</code></td>
<td><code>sdsimruntime mxSys.mxs --maxlib mylib.conf</code></td>
<td>Starts the simulator with a specific component configuration file loaded.</td>
</tr>
<tr>
<td><code>-n</code> or <code>-nomaxlib</code></td>
<td><code>sdsimruntime --nomaxlib</code></td>
<td>Starts the simulator with no component configuration files loaded.</td>
</tr>
<tr>
<td><code>-e</code> or <code>-silent</code></td>
<td><code>sdsimruntime --script myscript.mxscr -l logfile.txt --silent</code></td>
<td>Directs Simulator to not open a console terminal window, but run silently instead (Windows only). This can be used, for example, in combination with the <code>-l logfile</code> command line option to run batches of multiple simulations.</td>
</tr>
<tr>
<td><code>--display new_display</code></td>
<td><code>sdsimruntime --display $DISPLAY</code></td>
<td>Use this option to set the X display (Supported in Linux, only.). The default is <code>$DISPLAY</code>.</td>
</tr>
<tr>
<td><code>--geometry new_geometry</code></td>
<td><code>sdsimruntime --geometry new_geometry</code></td>
<td>Use this option to set the client geometry of the main window (Supported in Linux, only.).</td>
</tr>
<tr>
<td><code>--font new_font</code></td>
<td><code>sdsimruntime --font arial</code></td>
<td>Use this option to set the application font (Supported in Linux, only.).</td>
</tr>
<tr>
<td><code>--background color</code></td>
<td><code>sdsimruntime --background 8000</code></td>
<td>Use this option to set the default background color and an application palette (Supported in Linux, only.). Light and dark shades are calculated based on the new palette.</td>
</tr>
<tr>
<td><code>--foreground color</code></td>
<td><code>sdsimruntime --foreground 8000</code></td>
<td>Use this option to set the default foreground color (Supported in Linux, only.).</td>
</tr>
</tbody>
</table>
### Table 2-1 Command line options (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--button color</td>
<td>sdsimruntime --button 8000</td>
<td>Use this option to set the default button color (Supported in Linux, only.).</td>
</tr>
<tr>
<td>--visual Truecolor</td>
<td>sdsimruntime --visual Truecolor</td>
<td>Use this option to force the application to use TrueColor on an eight-bit display (Supported in Linux, only.).</td>
</tr>
<tr>
<td>--ncols count</td>
<td>sdsimruntime --ncols 216</td>
<td>Use this option to limit the number of colors allocated in the color cube on an eight-bit display (Supported in Linux, only.). If the count is 216, a 6x6x6 color cube is used (that is, six levels of red, blue, and green). For other values, a cube approximately proportional to a 2x3x1 cube is used.</td>
</tr>
<tr>
<td>--cmap</td>
<td>sdsimruntime --cmap</td>
<td>Use this option to install a private color map on an eight-bit display (Supported in Linux, only.).</td>
</tr>
</tbody>
</table>

#### 2.1.2 Opening a simulation

If SoC Designer Simulator is invoked from SoC Designer Canvas, the selected system is automatically loaded. To simulate a different system (or if SoC Designer Simulator was invoked directly), select **File → Open** and select the file from the list. (See Figure 2-1 on page 2-7.)
1. Some systems (for example, the ARM926-EJS system example) contain components that require an application file to be loaded for the core. If so, the Select Application Files dialog is displayed.
2. Select a component from the list and click on the **Select File** button to bring up a standard file-browsing dialog to select the input file for the core.

![Select The Application Code dialog](image)

**Figure 2-3 Selecting the input file**
2.1.3 Saving a simulation

You can save a loaded simulation at any time.

If the simulation was created from a SoC Designer project file (MXP), you are prompted to choose the type of simulation file to save (see Figure 2-4). Select one of the following options:

**MXS file**

The MXS file contains information about the simulation (including any attached probes or parameters you might have modified).

This is the default file type.

**MXR file**

The MXR file contains the same details as the MXS file, but also saves additional state information. (Not all components support this advanced feature. If one component does not support saving, the file can only be saved as an MXS file.) This state information enables you to then restore the simulation at exactly the same cycle count as when the simulation was stopped.

--- Note ---

The MXR file stores state information, but previously recorded profile or monitor information is discarded. Any Profile windows that are displayed are empty. You can of course continue to run the system and record new profile information to the windows.

To save collected profiling information for later analysis, use the **Export Raw Data** menu option on the Profile window context menu or the **Dump to CSV** checkbox on the Profiling Manager window. You cannot reload previously saved data back into a Profile window.

![Figure 2-4 Save As dialog](image)

Clicking the **Save** button displays the File Save dialog (see Figure 2-5 on page 2-10).
**Note**

MXS and MXR files contain the path to the original MXP file that actually contains the system. It is not possible to use an MXS or MXR file if the original MXP file is not present.
2.2 Application window

After starting SoC Designer Simulator, the application window shown in Figure 2-6 is displayed.

Figure 2-6 SoC Designer Simulator main window

The application window has the following sections:

**Toolbar** The Toolbar (see *Main menu and Toolbar commands* on page 2-16) provides a fast intuitive way to perform common actions on the simulation. (See *The Toolbar* on page 2-17.)
Cycle counter

The cycle counter shows how many cycles have elapsed in the current simulation run.

Diagram window

This window shows a graphical view of the system under simulation. By selecting components or connections in this window, it is possible to set breakpoints, add monitors, perform waveform tracing, and collect profile information. (See Diagram window on page 2-13.)

Only one simulation at a time can be loaded, but you can open a sub-system of the simulation (if it contains any). A new subsystem appears in a new tab, at the bottom of the diagram window.

Tool windows

The System Configuration and Parameters windows display information about the system and the selected component. (See Tools windows on page 2-29.)

Output window

The Output window displays log messages from the simulation. (See Output window on page 2-28.)

Command line

The command line is used to enter commands as a text string. (See Command Line interface on page 2-29.)

Task bar and status bar

The task bar and status bar display additional information about the system. See Task bar on page 2-29 and Status bar on page 2-28.

During a typical simulation run, several additional windows might be displayed:

Breakpoint Manager

Once a simulation is started, it runs until stopped by the user or until a breakpoint is reached. The Breakpoint Manager window displays all breakpoints that have been set in the system and controls whether or not they are enabled. (See Breakpoint windows on page 2-43.)

Trace Manager

The Trace Manager window controls whether activity on a connection or bus is logged. The recorded information can be displayed in a Waveform Viewer window. (See Trace Manager and the Waveform Viewer windows on page 2-51.)
Profiling Manager

The Profiling Manager window shows all system objects that can record profile information and controls whether a profile log file is created. The recorded information can be displayed in a profiling window. (See Profiling windows on page 2-62.)

Probes and monitors

Connecting a monitor probe to a component or connection displays information about the current state of the object. The Monitor window can be changed to history view to show previous activity on the object. The following types of monitor probes are available:

- Connection monitors (transaction, signal, or bus)
- Register or memory monitors (for example, for a cache)
- Disassembly monitors (for example, for a CPU component)
- Communication probes are used to test interfaces.

For more information, see Monitor windows on page 2-48.

Debuggers

You can connect an external debugger to enable source-code level debugging of an application running on the simulated system. This is called Target debugging. See Component level debugging on page 4-3.

You can also connect an external debugger to enable source-code level debugging of a component design. This is called host level debugging.

2.2.1 Diagram window

The Diagram window contains a graphical representation of the system model. The following sections describe the actions that can be controlled from the window:

- Context menu for a component in the Diagram window
- Context menu for connections in the Diagram window on page 2-14
- Context menu for Diagram window on page 2-15
- Diagram window appearance on page 2-15

Context menu for a component in the Diagram window

Select a component in the system and then right-click to display the context menu for the component. The menu contains the following entries:

View Subsystem…

Opens the components in the subsystem and displays them in the Diagram window. (Disabled if the component is not a system.)
View Registers…
Displays the Registers window for this component. (Disabled if the component does not contain registers.)

View Memory…
Displays the Memory window for this component. (Disabled if the component does not contain memory.)

View Disassembly…
Displays the Disassembly window for this component. (Disabled if the component is not a processor.)

View Dependency Graph…
Displays dependencies for components that use static scheduling.

Profiling
This entry contains a submenu of streams that can be enabled for profiling for this component. If the CAPI interface is not implemented for the component, this entry is not present.

Launch RealView Debugger…
The debugger is launched for the processor component. The debugger to launch is determined by the type of component. If the component is not a processor, this entry is not present.

Attach Host Level Debugger (Visual C++)…
Launches the Host level debugger for that platform: Microsoft Visual C++ for Windows, and gdb/ddd for Linux). This selection also loads design code for the component.

Edit Parameters…
Displays the Edit Parameters dialog for the component.

For more information on manipulating components, see The Object menu on page 2-22 and The Debug menu on page 2-25.

Context menu for connections in the Diagram window
Select a connection between components in the system and then right-click to display the context menu for connections.
• Insert/Remove Breakpoint
• Enable/Disable Breakpoint (disabled if no breakpoint has been set)
• Edit Breakpoint Properties (disabled if no breakpoint has been set)
• Insert/Remove Monitor
• Enable/Disable Tracer
• Profiler
  Profiler has two submenu components, to Enable and to Display.
For more information on manipulating connections, see The Object menu on page 2-22, Breakpoint windows on page 2-43, and Monitor windows on page 2-48.

Note
Use the View menu to hide or display connections between components (see The View menu on page 2-20).

Context menu for Diagram window
Right-click on a blank portion of the Diagram window to display its context menu:

Remove All Connection Breakpoints
Select this entry to remove all breakpoints on signals, registers, memory, and code.

Remove All Probes
Select this entry to remove all monitor, communication, and tracing probes. Any windows relating to the probes close.

Zoom In, Zoom Out, Zoom to Fit, and Zoom to 100%
Select this entry to zoom the system model shown in the Diagram window.

View Parent System…
Select this entry to return to the parent system after you have finished examining a component within a subsystem. If you are not currently viewing a subsystem, this entry is disabled.

Diagram window appearance
Use the View menu to control the appearance of the Diagram window. From this menu you can hide connections, set grid options, and zoom the view of the model in the window.

Select View → Grid Options to display the Grid Settings dialog shown in Figure 2-7 on page 2-16.
The dialog has the following controls:

**Show Grid [default = off]:**
Show the grid in the diagram with the specific options.

**Snap to Grid [default = on]:**
Moving or resizing objects snap the object to the grid boundary.

**Grid Horizontal Spacing / Grid Vertical Spacing [default = 10]:**
Set the horizontal/vertical distance (in pixels) between grid points to use if the window is zoomed to 100%.

**Grid Point Diameter [default = 1]:**
The diameter of the grid points in pixels with a zoom of 100%.

**Select Point Color… [default = gray]:**
The color of the grid points.

There is also a button to reset all the values back to the factory default settings.

### 2.2.2 Main menu and Toolbar commands

The following sections describe the toolbar buttons and the menus:

- *The Toolbar* on page 2-17
- *The File menu* on page 2-18
- *The View menu* on page 2-20
- *The Object menu* on page 2-22
- *The Control menu* on page 2-23
• The Debug menu on page 2-25
• The Window menu on page 2-27
• The Help menu on page 2-28

The Toolbar

The Toolbar buttons provide a fast way to access functionality that is available from the menu items. The Toolbar status indicators display the current system state.

![Figure 2-8 Toolbar](image)

The Simulation Toolbar contains the following items:

**Open, Save, and Close**

Click on these buttons to open, save, or close a simulation model.

**Brkpts, Profile, and Trace**

Click on these buttons to toggle displaying the Breakpoint Manager window, the Profiling Manager window, and the Trace Manager window.

**MemMap**

Click on this button to display a read-only version of the Memory Map Editor dialog and view the memory map configuration for the bus masters.

**Wave**

Click on this button to display the Waveform Viewer window and view data for tracing-enabled ports.

**Run, Stop, Step, Step n**

Click on these buttons to run and stop the simulation.

**Reset**

Click on this button to reset the simulation and set the cycle count to 0.

**Sync All**

Click on this button to synchronize open viewer windows for software and hardware profiling, and waveform features.

--- Note ---

Segment Size, Zoom Size, and Start Cycle data is not always possible to synchronize. For Segment Size, ensure that all open windows have the same x-axis display format.
System state

The state indicator shows whether the system is running or stopped.

To display the Simulation Toolbar, select **Simulation ToolBar** from the **Window** menu.

The following additional controls can be displayed in the Toolbar area:

**Cycle counter**

The cycle counter displays the current processor cycle. To display this button, select **Cycle Counter** from the **Window** menu.

**Zoom %**

The Zoom text box displays the current magnification of the system in the Diagram window. Use the controls at the side of the display to increase or decrease the zoom factor. To display this button, select **Zoom ToolBar** from the **Window** menu.

**Step N**

The Step N control displays the currently specified step size. Click on the button below the step size value to advance the simulation that number of steps. To display this button, select **Step N ToolBar** from the **Window** menu.

**The File menu**

Use the **File** menu to open, save, and exit simulations. The menu has the following entries:

**Open…**

Select this item to open a dialog that enables you to select a system project (.mxp or .mxe) file, a batch file (.mxscr), or an existing simulation (.mxs or .mxr) file.

--- **Note** ---

If your models are linked dynamically to external libraries, such as a high precision math library, you must make sure that the required library is in a place that the system’s dynamic loader can find it. On Linux systems this means putting the path of the library in your LD_LIBRARY_PATH. On Windows, the path to the location of the library must be in the PATH environment variable.

--- ---

If a system project file containing programmable cores is loaded the Select Applications dialog is displayed, where you can pick application files to be loaded into the core(s).

**Close**

Select this item to close the currently loaded simulation. If the simulation is a new or modified simulation, you are prompted to save the simulation.
Save  
If the current simulation was loaded from an existing file, selecting Save writes the simulation to the same file (and in the same format as it was read). The .mxs and .mxr files have different formats (see Save As below). If the simulation is newly created, clicking Save displays the Save As dialog.

Save As...  
Select this item to save a newly created simulation or to save an existing simulation to a new file. Selecting Save As displays a dialog shown in Figure 2-9 and gives you two choices:

- without state saves a simulation as an .mxs file
- with state saves an .mxr file that contains the simulation with state information. The simulation context is also saved. This includes monitor, tracing, or profiling windows that were open when the file is saved.

![Figure 2-9 Save as dialog](image)

If you load or create a simulation and run it for x number of cycles, you can then save the simulation as an .mxr file. If you reopen the file, the simulation is in exactly the same state as when you saved it. If you had previously run it for 1000 cycles, for example, after loading it indicates that it has already run 1000 cycles.

—— Note ———
Saving with state only works if all the components of the system have implemented the save state feature.

Recorded profiling information is not reloaded if an .mxr file is loaded. The profiling windows are displayed, but there is no profile information. Run the simulation to collect new profile information.

After you have chosen your file type, clicking Save displays a standard file browser.

Print...  
Select this item to print the system as displayed in the Diagram window. The Print dialog is displayed to enable you to select a printer.
Preferences…

Select this item to change the look and feel of SoC Designer Simulator. There are three sections for SoC Designer Simulator preferences.

Selecting OK and Save saves all of the currently set preferences.

Restore debug view…

Select this item to display a dialog for loading the debug context from a previously saved version of the MXS simulation file. Any breakpoints or probes that were present when the simulation was saved to file are restored when the file is opened. The simulation, however, is not reset.

See also the Save As menu item for the File menu.

File list

A list of all recently opened files is displayed near the bottom of the file menu. Selecting a file from the list opens the file.

Exit

Select this item to exit SoC Designer Simulator. If a simulation is running, it is stopped before closing.

If you have made any modification to the simulation, a prompt asks if you want to save before closing the simulation.

If the simulation windows has been launched from SoC Designer Canvas, only the simulation window is closed. If SoC Designer Simulator has been started directly, the application exits.

The View menu

The View menu is similar to the one in the SoC Designer Canvas window. The following menu items are present:

Simulation Hierarchy…

Select this item to display the Hierarchy Of dialog shown in Figure 2-10 on page 2-21. Use the Properties, Ports, Parameters, and Debuggers tabs to display detailed information on a selected component in the System list.
Clock Connections, Signal Connections, and Transaction Connections

Select these items to toggle the display of the specific connection type. If off, the ports and connections are not displayed.

Grid

Select this item to toggle display of the grid.

Note

Movement of objects is bound by the snap to grid flag not the display of the grid.

Grid Options…

Select this item to display the dialog box described in Diagram window appearance on page 2-15.
Zoom In, Zoom Out

Select these items to zoom the diagram in/out. The step size changes depending on the current zoom level.

Set Zoom to 100%

Select this item to reset the zoom level to the default of 100% (this is the normal zoom).

Zoom to Fit

Select this item to set the zoom level so that the entire diagram fits into the Diagram window.

Set Zoom…

Select this item to display a dialog box that enables to set the amount of zoom. The zoom can be in the range of 10% - 500%.

Center on Selected Object(s)

Select this item to center the Diagram window over the selected object or objects.

Center Diagram

Select this item to place the center of the system diagram at the center of the Diagram window.

Clear Output Window

Select this item to clear the output window. All output is lost after you use this item, unless logging to file is enabled.

The Object menu

The **Object** menu contains items that are typically found in context menus for components or connections:

Edit Parameters…

Select this item to display a dialog box that contains the following sections:

- The first section contains all the read-only properties of the component: type, version, load file extension, description, and debuggers supported.
- The second section contains a list of all the ports of the component.
- The third section contains all the editable parameters of the component.

To change the value of a parameter:

1. Double-click the left mouse button in the **value** column under the specific row of the parameter to be edited.
2. Type in the new value.
3. Click in the background of the list box to accept the change.

Note

Notice that the button to the right of the value field has changed to reset. Clicking this button resets the value to the factory set default value.

Insert/Remove Breakpoint
Select this item to insert a breakpoint on the selected connection. If the breakpoint already exists, selecting this menu option removes it.

Insert/Remove Monitor
Select this item to insert a monitor on the selected connections. If the probe already exists, selecting this menu option removes it.

Enable/Disable Tracing
Select this item to enable or disable tracing on the selected connections.

The Control menu
Use the Control menu to start, run, or stop the simulation. The menu has the following menu items:

Restart Simulation…
Select this item to cause a hard reset in the simulation.
If application files are associated with any of the components, the Select Application Files dialog is displayed. The default files to load are the currently loaded application files. Change the files if required and click Proceed. (If no application files are associated with any components, the dialog is not displayed.)

Reset
Select this item to cause a soft reset in the simulation. It resets the cycle count to zero and resets all models in the system. Unlike Restart Simulation, the application files are not reloaded.

Run
Select this item to start the simulation running and continues running until you stop the simulation or a breakpoint is hit.

Stop
Select this item to stop the simulation if it is currently running.

Step
Select this item to simulate one system cycle.
Note
One simulation cycle may not be equivalent to one instruction in the instruction set simulators. For instruction stepping, use the step button in the debugger attached to the processor.

Step n …
Select this item to display the dialog shown in Figure 2-11 and step the simulation a fixed number of cycles:

1. Enter the number of cycles to step in the Cycles text box.
2. Optionally, specify a delay between cycles by entering a value between 1 and 99 in the Delay text box. The delay is in 1/10ths of a second and delay time therefore varies between 0.1 and 9.9 seconds.
3. Click Step to start the simulation for the specified number of cycles. Click on the Iterate button to continuously run the specified number of cycles, stop for the specified delay and then run again.
4. Use the Stop button in the toolbar (or the Stop entry on the Debug menu) to stop the simulation before the specified number of cycles has been reached (for Step mode).

Cosimulation Turbo Control
This item is shown only when a system is configured for HDL cosimulation with the CoDesign Package. For specific information, see the SoC Designer CDP HDL CoSimulation Guide.
The Debug menu

Use the Debug menu to control how debug information for components is displayed. The menu has the following entries:

**Display Messages**

Select this item to toggle displaying of messages generated by the components in the currently running simulation. The components still generate the messages however. If this option is turned off, the simulation performance is improved at the expense of diagnostic information.

**Log Messages To File**

Select this item to send all output to a file in the preferences directory:

- On Linux this is usually a `${HOME}/.maxsim/mxexplorer/logs` directory.
- On Windows, the `.maxsim/mxexplorer/logs` directory branch is within your profile directory.

If this option is turned off, the simulation performance is improved at the expense of diagnostic information.

**Remove All Probes**

Select this item to remove all probes from the current diagram.

**Breakpoint Manager…**

Select this item to display the Breakpoint Manager window that is used to administer all breakpoints (including component level breakpoints and system level breakpoints).

**Profiling Manager…**

Select this item to open a dialog that enables you to control the collection of profiling data from components that implement the CAPI interface. See *Profiling windows* on page 2-62.

**Trace Manager…**

Select this item to open a dialog that enables you to administrate all traces in the current simulation.

**Memory Map Editor**

Select this item to display the Memory Map Editor dialog and view the memory map configuration for the bus components.
Waveform Viewer…

Select this item to open the Waveform Viewer. The viewer enables you to display traces without leaving the SoC Designer Simulator application. See Waveform Viewer on page 2-55 for additional information.

View Host Level Performance Profiling…

Select this item to open the Host Level Performance Profiling Viewer. The viewer enables you to display accumulated run time for a module, a function or a source code line.

Reset Host Level Performance Profiling…

Forces the accumulated time that displays in the Host Level Performance Viewer to reset to 0.

Launch Debugger for …

Select this item to display a submenu containing all the components that have support for at least one of the debuggers supported by SoC Designer. Clicking on one of the components in this sub menu launches the default debugger associated with that component. The debugger to be launched is specified in brackets next to the component’s instance name.

View Registers for …

Select this item to display a submenu listing all components that implement an CADI interface. These components enable you to view and modify the component’s registers. Selecting an item from this menu opens a dialog displaying the registers and their current values for the selected component.

For more details see Component level debugging on page 4-3.

View Memory for …

Select this item to display a submenu listing all components that implement a CADI interface. The components permit you to view and modify the component’s memory. Selecting an item from this menu opens a dialog displaying the current values held in the memory for the selected component.

For more details on using this window see Component level debugging on page 4-3.

View Disassembly for …

Select this item to display a submenu listing all the components that implement a CADI disassembly interface. These components enable you to view the disassembly and to step through the instructions.

For more details on using this window see Component level debugging on page 4-3.
Refresh External Debuggers
Select this item to send synchronization information to all attached debuggers.

The Window menu
The Windows menu has the following entries:

Simulation Toolbar
Select this item to toggle display of the Simulation Toolbar.

Cycle Counter
Select this item to toggle display of the cycle counter.

Zoom Toolbar
Select this item to toggle display of the zoom control.

Step N Toolbar
Select this item to toggle display of step n control.

Tools Window
Select this item to toggle display of the System Configuration and Parameter tools windows.

System Configuration Window
Select this item to toggle display of the System Configuration window.

Parameter Window
Select this item to toggle display of the Parameter window.

Output Window
Select this item to toggle display of the Output window. If the Output window is not displayed, the taskbar and command line are also not displayed.

Command Line
Select this item to toggle display of the command line.

Taskbar
Select this item to toggle display of the taskbar.

The Help menu
The Help menu has the following entries:

Help… Select this item to open the user documentation using Acrobat Reader.
Tip of the Day…
Select this item to display helpful hints about using SoC Designer Simulator.

About…
This displays the standard About dialog box displaying SoC Designer version information. A Plugins button enables you to display the list of currently installed Plugins.

About System…
Select this item to display detailed system and session setup information. This also displays the currently loaded component libraries.

What’s this
Select this item to change the cursor to a question mark. Clicking on a control or window in SoC Designer Simulator displays general information about that item.

2.2.3 Status bar
Move the mouse cursor over an menu item, toolbar button, or component to display additional information about the object.

2.2.4 Output window
Most of the menu items and buttons log information in the Output window.
Right-click in the Output window to display the context menu:
Copy to clipboard
Select this item to copy the contents of the Output window to the clipboard. The information can then be pasted into a text file for later analysis.
Clear
Select this item to delete all of the text from the Output window.
Select Font…
Select this item to display the Select Font dialog that enables you to select the font used to display messages. Select the new font and style from the dialog and click OK.

2.2.5 Command Line interface
Use the Command Line text box (located at the bottom of the Output window) to enter most of the commands that are available from the graphic user interface. For example, typing step into the Command Line and pressing Enter advances the simulation by one step.
If you are writing a SoC Designer script, you can use the Command Line to try out the commands before including them in the script. For more information on scripting, see the MxScript Reference Manual.

Right-click inside the text box to display the context menu that enables you to edit, copy, or paste text.

### 2.2.6 Title bar

The title bar of SoC Designer Simulator contains the name of the application in addition to the current system being simulated.

If there is no system loaded, the title bar contains the application name plus **No Simulation Loaded**. If the simulation has been modified and not yet saved, an asterisk is placed to the right of the system name.

### 2.2.7 Task bar

If user windows (such as monitor windows for example) are open, a task bar is displayed above the status bar and displays the name of each of the user windows. Click on the icon in the task bar to bring the selected window to the front.

### 2.2.8 System menu and controls

SoC Designer Simulator supports the standard system menu and controls: move, size, minimize, maximize, restore, and close.

The appearance and function of these controls vary depending on the operating system. See the operating system help for more information.

### 2.3 Tools windows

The System Configuration and Parameters windows are collectively called the Tools windows. These windows display information about the system and its components. To display the windows, use the **Window** menu (see *The Window menu* on page 2-27).

To adjust the area occupied by the windows, move the dividing bar between the Diagram window and the Tools windows (see Figure 2-12 on page 2-30).
Figure 2-12 Changing the size of the Tools windows

The Tools windows are typically displayed as part of the main application window. The windows can be undocked by selecting the bar at the top of the windows and dragging the panel off the application window (see Figure 2-13 on page 2-31) or by selecting Undock from the context menu.
Figure 2-13 Undocking the Tools windows

Left-click over a blank part of one of the tools windows to display a context menu. You can use the context menu to toggle display of some or all of the tools windows.
2.3 Tools windows

2.3.1 Parameter window

The Parameter window contains all the editable parameters of the currently selected component. If more than one object is selected, or the object is not a component, the Parameter window is blank.

![Parameter window](image)

The window contains the following columns:

- The **Parameter** column lists all the parameters available to be edited.
- The **Value** column contains the current values for all the components.
- The default column contains a button to reset the value back to the default value. If the value currently is the default, the button is disabled. Otherwise, the button displays the reset icon.
- Parameters that are editable are indicated by a check mark in the fourth column.
Double-click the left mouse button over the editable value that you wish to modify. This invokes an in-place text edit field. Modify the value and then press the Enter key (or click elsewhere in the application) to accept the value. Press the Esc key to cancel the change.

- The Type column indicates whether the parameter is a Boolean or numerical variable.

2.3.2 System Configuration window

The System Configuration window contains a list of all the components that have been used to create the displayed system model.

Selecting an object in the System Configuration window also selects the object in the Diagram window.

Click on the + symbol next to a component to display its contents:

- A system component in the list expands to show all the components that it is constructed from.
- A basic component expands to show all the ports that are present on the component.

The context menu displayed when you right-click on an entry in the System Configuration window is the same context menu displayed when you right-click on a component or port in the Diagram window.
2.4 Preferences dialog

This section describes the configuration settings present in SoC Designer Simulator sections of the Preferences dialog.

2.4.1 General Simulator preferences

Use the text boxes in the Application Options panel to change:

- the number of recently loaded simulations to remember
- the number of commands issued on the command line to keep in the history
- the number of output lines to keep in the main window.

The Restore Window Geometry For Simulator check box enables you to turn on/off the restoring of the simulation geometry for each simulation session. It restores the window size and position of SoC Designer Simulator to the details described in the simulation file when it is reloaded. These details are written to the simulation file every time you save it.

Use the radio buttons in the Double Click on Components panel to select how a component reacts to a mouse double-click. Select Edits Parameter to open the Parameter window. Select Opens Sub-System to open the subsystem (sub system only applies to hierarchical components).

Use the check boxes in the Remember panel to select whether previously opened application files and previously set breakpoints are used when an .mxp file is reloaded. Use the Clear All buttons to clear the list of remembered files and breakpoints.

Use the Clear Recent File History List button to empty the list of files that have recently been opened.
2.4.2 Simulator preferences for appearance

The options in this section affect the appearance of the SoC Designer Simulator.

- The entries in the Miscellaneous group control the display of windows or controls as follows:
  - The Show Cycle Counter option toggles the cycle counter that appears in the toolbar.
  - Select the Show Commandline box to enable or disable the command line window that is displayed at the bottom of the SoC Designer Simulator window.
  - Select the Display external ports check box to display external ports in the Diagram window.

- The Font size group selects the font sizes for the register, memory and disassembly windows and enables increasing/decreasing the font used in the CADI register, memory and disassembly view windows.

- Use the Windows Management section to select whether minimizing of windows is managed by the Taskbar or by explicitly minimizing the windows.

- Click the Advanced button to display the Advanced Preferences dialog to choose a different font for text in CADI probe windows. The default font for text associated with probes is 10-point Courier New. Click the Select Font button on the Advanced Preferences dialog to replace this font with another from the displayed Select Font dialog.
2.4.3 Simulator preferences for general simulation behavior

The preferences in this section only apply to the SoC Designer Simulator.

![SoC Designer Preferences dialog](image)

**Figure 2-18 Preferences dialog for general simulation behavior**

The Simulation options in this dialog are as follows:

- The cycle count display format can be controlled with the Mode option.
- The Display Performance check box enables display of the performance data in the output window when the simulation is stopped. This requires the execution of more than 1000 cycles.
- The Unconnected Ports group controls how warning messages are displayed.
- The Verbosity group controls the level of any errors that are displayed.

2.4.4 Simulator preferences for Tracer features

_____ Note _____

The features of this preference dialog are identical to the features of the Trace Manager dialog. For more complete information on the dialogs in this section, see the Trace Manager dialog, Settings tab on page 2-53.

You can set preferences for Tracer Simulation features using the Preferences dialog for Tracer features (see Figure 2-19 on page 2-37), as follows:
The sections in this dialog are as follows:

**Memory Usage**

Memory Usage displays in this section. To set memory usage, click Usage Settings and set the Max Memory Usage and Traces specifications in the dialog (see Figure 2-20).

![Memory Usage dialog](image1)

**Export to VCD panel**

- Generate “value” wire for registers — Option for backward compatibility
- Set all values to 0 on initialization — Option for backward compatibility

![Export to VCD panel](image2)
Flush the VCD trace data after every write—When enabled, VCD fails are flushed to disk after every simulation cycle. When disabled (this is the default), VCD output is buffered for maximum performance.

2.4.5 Simulator preferences for profiling configuration

Use the Profiling Manager Preferences Settings dialog (shown in Figure 2-21) to set the profile buffer characteristics, as follows:

1. Select Profile preference option from the left-side tree menu to configure the buffer size and display options for profiles.

![Figure 2-21 Profiling Manager preference settings](image)

2. A default buffer size is allocated for the profiling streams. To change the buffer size (for example, if you are recording a large number of streams), enter the new value in the Total raw data buffer size text box and click Apply.

You can also use the Total data buffer per view and Data display range text boxes to change these profiling characteristics. The default values for these settings, however are generally adequate and typically do not require changing.

3. To save the Profile settings for the current session, click OK. To save the Profile settings permanently, click OK & Save.

2.4.6 Simulator preferences debug behavior

This section explains component debugging options. The page is divided into the panels Debug Messages, Monitor, Logging, Component Calling Order, and Exceptions.
The **Debug Messages** panel contains several check boxes grouped below the **Enable Messages Before** check box. Unchecking **Enable Messages Before** turns off the display of all messages and disables the related check boxes. Messages are still generated by the components but not displayed and logging still occurs if logging is enabled. If **Enable Messages Before** is checked, you can check the boxes for various types of event and force the simulation to print out a debug message. For example, checking `communicate()` results in a debug message being displayed before the `communicate()` function is called for each of the clocked components in the simulation for every step executed. Use these debug options to identify a problem in a model.

If **Interactive Mode** is checked, a prompt asking the user whether to continue is displayed after each debug message.

If **Stop Simulation on Warning** is checked, the simulation stops if a warning message is generated.

Check **Show debug transactions in monitors** check box in the **Monitor** panel to include lines for **ReadDbg** and **WriteDbg** events in monitor history dialogs.
• Use the controls in the **Logging** panel to change the settings for logging:
  — The **Enable Logging to File** box is used to turn logging off or on.
  — If logging is enabled, the **Clear Log File on Startup** is enabled. If checked, the log file is cleared on startup of the tool. This option is on by default and a log file from a previous run is lost the next time you start the tool.
  — Enter the path for logging in the **Log File** text box. If a new file is entered instead of the default file, the new file is used after the **OK** (or **OK and Save**) button is pressed.

• Use the controls in the **Component Calling Order** panel to modify the order that components communicate and updates are called in during a simulation. This can be used to verify that components do not have order dependency problems. The system must behave in the same way, regardless of the order that the components communicate and updates are called:
  — Selecting **Default Calling Order** causes communicate and update to be called on each component in the default order for the tool.
  
  ____________
  **Note**
  ____________
  The order that the tool calls the components is not guaranteed. The components must be developed in such a way that they are not order-dependent.

  — Selecting **Reverse Calling Order** causes the components to be called in reverse order instead of the default order that is normally used for the tools.
  
  — Selecting **Progressive Permutation Ordering** changes the order that the components are called in for every cycle. (A progressive permutation algorithm is used so that every possible order of calling the clocked components is used, one per cycle. After all possible combinations have been exhausted, the order restarts at the beginning.)

These options can be used to verify that a system behaves correctly under different scenarios and is not dependent on the calling order.

• Check **Enable Crash Handling** to cause the system to perform the following when an exception occurs in the code of a model:
  — stop the simulation
  — record system information relevant to the reproduction and analysis of the erroneous session.
  — record system information relevant to the reproduction and analysis of the erroneous session
  — display an error message (see Figure 2-23 on page 2-41) advising the user where to locate and send the recorded file and other information related to the crash.
If **Enable Crash Handling** is unchecked, SoC Designer Simulator does not catch the exception. The exception is passed to the external debugger.

On Windows, for example, the Microsoft Visual Studio debugger catches the exception and, if debug information is available, point to the line in the source code that triggered the exception.
2.5 Memory Map Editor

The Memory Map Editor lists all memory regions for all slave components. If the system contains hierarchical elements, the memory regions for all slaves in the subsystem are also displayed.

——— Note ————

The memory address space can be viewed from SoC Designer Simulator if a memory map exists.

Use the Memory Map Editor from within SoC Designer Canvas to create and edit new memory maps.
2.6 Breakpoint windows

This section describes the windows related to setting or viewing breakpoints.

It is possible to enable, disable and remove existing breakpoints and define breakpoint conditions.

To set a breakpoint on a transaction or signal connection, perform one of the following:

- Double-click on the connection to set or clear a breakpoint.
- Right-click on a connection and select Insert/Remove Breakpoint from the context menu (see Figure 2-24).

![Figure 2-24 AXI component context menu](image)

- Select the connection and select Insert/Remove Breakpoint from the Object menu.

To set a breakpoint on a memory or register window:

1. Select View Registers or View Memory from the component context menu.
2. Click on the location.
3. Right-click to display the context menu.
4. Select Insert/Remove Breakpoint from the context menu.

To set a breakpoint on a disassembly window:

1. Select View Disassembly from the processor component context menu.
2. Double-click on the memory address.

An alternate procedure to set a breakpoint on a disassembly window is as follows:

1. Select View Disassembly from the processor component context menu.
2. Click on the location.
3. Right-click to display the context menu.
4. Select Insert/Remove Breakpoint from the context menu.
2.6.1 Breakpoint Condition dialog

To set a specific breakpoint condition:
1. Select the breakpoint.
2. Right-click to display the context menu.
3. Select Edit Breakpoint Properties from the context menu.
4. Use the dialog displayed to edit the properties (see Memory or register breakpoints and Set conditions on signal or transaction breakpoints on page 2-45).

Memory or register breakpoints

Possible breakpoint conditions for a memory location or register are:

changes stop whenever the register/memory value changes
equals stop whenever the register/memory equals a specific value
not equals stop whenever the register/memory differs from a specific value
greater than stop whenever the register/memory is greater than a specific value
greater than or equal stop whenever the register/memory is greater than or equal to a specific value
less than stop whenever the register/memory is less than a specific value
less than or equal to stop whenever the register/memory is less than or equal to a specific value
within stop whenever the register/memory is between the specified values.

Note

The CADI interfaces do not enable breakpoints on read/write of a register or memory location.

---

Figure 2-25 Breakpoint dialog for memory
Set conditions on signal or transaction breakpoints

Click on Debug → Breakpoint Manager… to open the Breakpoint Manager dialog (see Figure 2-27).
To set a Signal or Transaction breakpoint to break on specified conditions, select a signal or transaction breakpoint in the **Breakpoint Manager** dialog and click the **Properties** button to display the **Breakpoint Condition** dialog shown in Figure 2-28.

To see a more complete description of the **Breakpoint Manager dialog**, see *Breakpoint Manager*.

![Breakpoint Condition dialog](image)

Figure 2-28 Breakpoint conditions dialog for a signal or transaction

Using the **Breakpoint Condition** dialog, you can set simple conditions based on AXI properties such as address, data, beat, using the check-boxes to select the property and setting the conditions by using the pull-down menus.

---

**Note**

You can set breakpoint condition in batch mode, using MxScript. For information on using MxScript, see the `bpConnSetCond(…)` function in the *MxScript Reference Manual*.

---

### 2.6.2 Breakpoint Manager

Click on the **Brkpts** button or select **Breakpoint Manager** from the **Debug** menu to display the Breakpoint Manager dialog.
Use the controls on the Breakpoint Manager dialog to remove, enable, or disable breakpoints. The breakpoints can be managed individually (for example by selecting a breakpoint in the list and then pressing the **Enable** button) or as a group (for example by pressing the **Enable All** button).

Select a breakpoint and click **Properties** to display the breakpoint conditions that have been set for the breakpoint (see *Breakpoint Condition dialog* on page 2-44).

Select a breakpoint and click **Locate** to display the window that contains the code, register, or memory that has a breakpoint set. If a signal or transaction has a breakpoint set, the connection is selected in the Diagram window (see *Diagram window* on page 2-13).
2.7 Monitor windows

Attach a monitor to a connection or bus to view the activity each cycle. The default dialog displayed shows the current cycle. If there was a change in the current cycle then the information is displayed in red, otherwise it is gray. The cycle number (for the last change) is displayed in the upper left corner of the dialog.

To insert a monitor on a connection, select the connection and use the context menu (see Figure 2-30).

![Figure 2-30 Selecting Insert Monitor from the context menu](image)

A tracing symbol is displayed on the two ends of the connection, as shown in Figure 2-31 on page 2-49.
Use the **History** button to enlarge the monitor dialog to view a history of the transactions (see Figure 2-32).

![Figure 2-31 A monitor on a connection](image)

**Figure 2-31 A monitor on a connection**

![Figure 2-32 Monitor window for transaction](image)

**Figure 2-32 Monitor window for transaction**
The monitor for a signal is similar to the transaction monitor, but there is not a Return column.

![Monitor window for signal](image)

**Figure 2-33 Monitor window for signal**

For additional information on connecting a monitor to buses, see *Monitor probes* on page 4-18.
2.8 Trace Manager and the Waveform Viewer windows

Tracing records changes that occur in a register or are transmitted over a connector. Recorded trace information can be displayed in a Waveform Viewer window.

For information on configuring a Trace probe for changes that transmit over a connector, see the following subsections:

- **Trace Manager dialog, General tab**
- **Trace Manager dialog, Settings tab on page 2-53**
- **Waveform Viewer on page 2-55**

--- **Note** ---

For information on viewing recorded trace information with a Waveform Viewer, see **Waveform Viewer on page 2-55**

For information on tracing registers, see **Tracing register contents on page 2-91**.

2.8.1 Trace Manager dialog, General tab

To display the Trace Manager General tab, click **Debug → Trace Manager** in the **Simulator** window.

Use the **General** tab of the Trace Manager (Figure 2-34 on page 2-52) for the following purposes:

- to remove existing traces
- to enable or disable existing traces
- to control the waveform display.

Items can be traced with or without display in the Waveform Viewer.

Select the check box under **Enable** to record trace information for a stream.

If a trace in the **Name** list is selected, the buttons at the right of the dialog are enabled:

- **Remove** Click this button to remove the selected trace from the list and stop recording trace information for this stream.

**Enable/Disable Tracing**

Click these button to enable or disable tracing for the stream. If the stream was previously enabled, the **Disable** button is enabled and the **Enable** button is grayed out (and the reverse if the stream is not enabled).
Locate  Click this button to move the connector matching this stream to the center of the Diagram window.

Properties  Click this button to display the properties dialog for this trace. The contents of the dialog depend on the type of trace selected.

![Trace Manager dialog](image)

Figure 2-34 Trace Manager dialog

Use the buttons at the bottom of the dialog to manage all of the traces:

**Remove All**  Click this button to remove all traces from the list and stop recording trace information for the streams.

**Enable All/Disable All**

Click these buttons to enable or disable tracing for all of the streams in the list.

Click the **Display Wave** button to display the Waveform Viewer window. The window contains all the traces that are currently enabled.
2.8.2 Trace Manager dialog, Settings tab

To display the Trace Manager, click Debug->Trace Manager in the Simulator window. Use the Settings tab of the Trace Manager (Figure 2-35) to enable viewing and modifying Memory Usage and Usage Settings for tracing.

Memory Usage panel

Click on the Usage Settings button to open the Memory Usage dialog (Figure 2-36 on page 2-54).

The bar at the bottom of the dialog indicates how much of the allocated memory has been used.

Use the Memory Usage dialog to adjust the maximum memory used for tracing and configuring the page size used for each trace.

Specify the total memory available for all traces in the Max Memory Usage text box.
Use the controls in the **Page Size** panel to specify how much memory is allocated to each added trace channel. If the buffer for this page becomes full, the buffer size is increased by the amount specified (subject to the total limit imposed by the value in **Max Memory Usage**). The page size values determine the compromise between tracing performance and memory consumption:

- The default values are chosen to be suitable for standard computer configurations in the most common cases, for example, 256MB of RAM.
- If a very large number of trace channels are required or if a lot of simulation time is to be covered by the trace (for example, much greater than one million cycles) it might be desirable to reduce the page size.
- If you are using a workstation that has a lot of available memory, you can sometimes improve performance by increasing page size.

![Memory Usage dialog](image)

**Figure 2-36 Memory Usage dialog**

**Export to VCD panel**

The options in this section of the **Trace Manager dialog** are for backward compatibility.
2.8.3 Waveform Viewer

Select Waveform Viewer from the Debug menu to display the window for viewing and analyzing the events of connections and the values of registers and memories over time (see Figure 2-37).

The Waveform Viewer provides powerful features for analyzing the traced waveforms:

- choice between cycle mode and time mode (not available on all traces)
- zooming and zoom history
- dynamic auto-scrolling at runtime
- transaction awareness, including display of access type (read or write)

Note

Transactions that are split across multiple cycles, such as AHB or AXI bus transactions, have the address, data, and status information correctly represented in the appropriate cycle.

The Channel column shows the names of each channel and the active channel
- the Value column shows the values of each channel for the cycle under the cursor
- hierarchical channels enable the display of single bits of any value
- a cursor and left and right locators for detailed event analysis
- placing the mouse cursor over a waveform displays the value at that point
- keyboard shortcuts for all features.
Using the cursors and locators

The waveform viewer cursors enable you to conveniently locate events and measure the time between events. The following cursors are used:

- main cursor (blue)
- left cursor (green), also called a locator
- right cursor (red), also called a locator

When the Waveform Viewer window is first displayed, none of the cursors are visible and the channel in the top row is the active channel. Left-click with the mouse to set the current cursor in the viewer.

To set the left and right cursors, press the Shift key and left or right-click. (You can alternatively use the L and R keys.) Figure 2-38 shows the window and cursors.

![Waveform Viewer window in time mode](image)

**Figure 2-38 Waveform Viewer window in time mode**

Use the left/right arrow keys to move the cursor to the previous/next event in the active channel. The active channel is highlighted in red and can be changed using the arrow up/down keys or by clicking on an entry in the Channel or Value columns.

Synchronization

You can synchronize multiple hardware (CAPI) profiling, software profiling and waveform viewer windows by clicking the SyncAll button on the toolbar.

To exclude individual windows from resynchronization, click the Sync button on the title bar of the windows.
Using the Zoom functions

The SoC Designer waveform viewer supports a wide range of zoom features and dedicated buttons.

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Zoom in or out" /></td>
<td>Zoom in or out (factor is 2 or 2.5)</td>
</tr>
<tr>
<td><img src="image" alt="Zoom to fit" /></td>
<td>Zoom to fit (zooms all the way out)</td>
</tr>
<tr>
<td><img src="image" alt="Zoom to normal" /></td>
<td>Zoom to normal (zooms to a level where 1 pixel represents 1 cycle)</td>
</tr>
<tr>
<td><img src="image" alt="Zoom to cursors" /></td>
<td>Zoom to cursors (zoom to area defined by the left and right locators)</td>
</tr>
<tr>
<td><img src="image" alt="Zoom history undo/redo" /></td>
<td>Zoom history undo/redo (a circular buffer remembers the previous five zoom levels)</td>
</tr>
<tr>
<td>-</td>
<td>Zoom in/out at mouse cursor (Ctrl-key + left/right mouse button)</td>
</tr>
<tr>
<td>-</td>
<td>Zoom using the mouse wheel (while holding down the Ctrl-key)</td>
</tr>
</tbody>
</table>

All buttons have tool-tips that appear when the mouse pointer hovers over the button for more than a second.

Viewing options

The waveform of each channel can be displayed in one of the following ways:

- values
- boolean
- graph (with discrete or interpolated values).

The waveform viewer supports graph displays a graph with connected event-dots instead of a printed value. The standard graph displays a line at the level of the last event until the next event occurs. The interpolated graph simply interconnects consecutive events with a straight line.
To select one of the waveform options right-click on a trace in the Channel list and choose the desired option from View waveform as item on the context menu.

The values in the displayed waveforms can be displayed in the following ways:

- hexadecimal
- full hexadecimal (with leading zeros)
- unsigned decimal
- signed decimal
- floating point (only for floating point CADI registers)

To select one of the view options for values, right-click on a trace in the Channel list and choose the desired option from the context menu View values in item.

When choosing the graph view option, the Y-axis needs to be configured. By default the channel is 64 pixels high and the scale shows a range from 0 to the maximum value permitted by the bitwidth of the channel (for example in many cases 32bit). This can be changed by selecting Configure Graph View... from the context menu.

### Context menu for the Waveform Viewer window

Right-click on a Waveform Viewer window to display the general context menu or right-click on an individual channel to display the context menu for that channel. The menu has the following entries:

**Add trace to view**

Select one of the available traces from the list in the Trace Manager window to add it to the Waveform Viewer window.

**Remove trace from view**

Removes the trace from the list.

This option is disabled on the general context menu.

**Move trace**

Select up, down, top, or bottom to move the position of the trace in the trace list.

This option is not present on the general context menu.

**Show waveform for channel**

Select a channel from the list to toggle display.

This option is not present on the general context menu for the waveform display area.
Hide/Show SubChannels…

Select the Show/Hide SubChannels dialog. Check or uncheck the individual channels from the dialog and click OK. Double-click on a subchannel name to edit the name as shown in Figure 2-39.

This option is not present on the general context menu.

![Hide/Show SubChannels dialog](Image)

**Figure 2-39 Hide/Show SubChannels dialog**

**View waveform as**

Select a display type (Values, Boolean, Graph, or Graph Interpolated) from the list.

This option is not present on the general context menu.

**View values in**

Select a numeric display type from the list.

This option is not present on the general context menu.
Simulate to cursor

Runs the simulation until the cycle count indicated by the cursor is reached. If a breakpoint occurs before the cursor is reached, the simulation stops. This option is only present on the context menu for the cursor.

View values in

Select a display type (Values, Boolean, Graph, or Graph Interpolated) from the list.

Configure Graph View…

Displays the dialog specifying the scale and value range for the graph associated with the channel. Channel Height determines the amount of height available for this channel. If the height is too small, the name of the channel might not be visible. If values are outside of the Minimum Value and Maximum Value range, the graph uses the range value instead. This option is only present on the context menu displayed for a channel that is currently in graph view.

Open new window…

Displays a new Waveform Viewer window with no traces displayed.

Duplicate this window…

Displays a new Waveform Viewer window with the same traces displayed as the original.

Open trace manager…

Displays the Trace Manager (see Trace Manager dialog on page 2-52).
Configure waveform view…

Displays a dialog for configuring the Waveform Viewer refresh rate and the X axis scale (see Figure 2-41). If Autoscroll to Simulation Cycle is checked, the window always displays the current cycle.

Figure 2-41 Configure Waveform Viewer
2.9 Profiling windows

SoC Designer Simulator provides support for profiling and visualization at runtime. Components use the CAPI interface to provide profiling streams that the user can choose to activate and display.

This section contains the following subsections:
- Profiling Manager dialog
- Profile window on page 2-64
- Displaying a profile from the component context menu on page 2-68
- Profile Settings Colors dialog on page 2-69
- Profiling Settings Groups dialog on page 2-70
- Profiling register values and transaction connections on page 2-73
- Latency profiling on page 2-74

2.9.1 Profiling Manager dialog

Use the Profiling Manager dialog (shown in Figure 2-42 on page 2-63) to view the profiling streams that are available and to activate the streams of interest:

1. Select the Streams tab to display the streams that can be profiled.
2. Select the Enable checkbox next to the desired streams to record profile information for that stream.
3. Some streams can be displayed with different parameters for the X and Y axis. If the X axis or Y axis entries contain a dropdown list, select the values for the axis from the list.
4. To save the stream to a CSV file (commas separate the values in the file), click the checkbox under Dump to CSV for the stream. This feature is present for hardware streams only. Use the Export raw data menu option on the Profile window context menu to save software profile data.

Note

Saving a simulation as an MXR file stores state information, but previously recorded profile or monitor information is discarded. Any Profile windows that are displayed when an MXR file is reloaded are empty. You can of course continue to run the system and record new profile information to the windows.

You cannot reload previously saved data back into a Profile window.

5. Profile information is typically recorded to a buffer in memory. If the buffer becomes full, older information is discarded to make room for the new data. To avoid having the older information discarded, click the checkbox under Buffer to disk for the stream. If the memory buffer fills, the hard disk is used to hold the overflow data. This feature is present for hardware streams only.

Note

If the Buffer to disk checkbox is unchecked, any files previously used to record profile data are deleted.
6. Click on Display to display the profiling windows for the selected streams. Each stream is opened in a separate window. (See Profile window below.)

7. Select the Settings tab to configure the buffer size and display options for profiles.

8. A default buffer size is allocated for the profiling streams. To change the buffer size (for example, if you are recording a large number of streams), enter the new value in the Total raw data buffer size text box and click Apply.

9. You can also use the Total data buffer per view and Data display range text boxes to change these profiling characteristics. The default values for these settings, however, are generally adequate and typically do not require changing.

   --- Note ---
   You can also specify Profiling Manager Settings in the Profile sub-dialog of the SoC Designer Preferences menu (see Simulator preferences for profiling configuration on page 2-38).

2.9.2 Profile window

The Profiling window shows the profiled information of the selected profiling stream in a graphical form. Each profiling stream can provide multiple channels (represented by different colors within the same diagram). Each channel represents an event type, for example reads or writes. The display of each channel can be controlled individually by using the appropriate check box in the Legend list.
Segment type and size

The settings for the Segment type and Segment size drop-down lists determine how hardware profiling data are summed up in each bar of the Profiling window diagram.

There are the following options:

Fixed size segments

Fixed size controls how many cycles are summed for each bar. If the value in the Segment size drop-down list is 1000 for example then the vertical size of the color box inside each bar shows how many times the event has occurred within the 1000 cycles that are represented by the bar.

Select the segment type you wish to record and then select the size from the Segment size list.

Function based segments

This displays the profiling on a per-function basis. The Segment size list contains the core components that support software profiling. Each segment has the width of the corresponding function call from the selected core’s profiling. This permits analyzing the events occurring inside each separate function call. Use this feature to, for example, locate the functions the result in significant conflicts on the bus or cache misses.
--- **Note** ---

This is enabled only if software profiling has been enabled for a core. (See *Software function profiling* on page 2-94.)

--- **Note** ---

**Clock based segments**

This displays the profiling based on a component clock (other than the main simulation clock). The **Segment size** list contains the clock connections that are enabled for profiling.

--- **Note** ---

To enable profiling, right-click on a clock connection and select **Enable profiling** from the context menu.

--- **Note** ---

Enabling software, clock, or AHB/AXI latency profiling must be done before the profiling window is displayed. If you enable profiling after the window is displayed, you must close and reopen the window to have the additional options displayed.

For software profiles, the **Segment** lists are not displayed.

---

**Zooming**

You can zoom in, out, zoom to fit, and zoom all the way in. Figure 2-45 shows the zoom controls on a hardware profile window.

---

**Figure 2-45 Zoom buttons**
Note

Software profiles use a window similar to the Waveform Viewer window. See Using the Zoom functions on page 2-57.

You can also use the mouse to zoom profiling and waveform viewer windows:

- Press the Ctrl key and click in the profiling diagram to center the view on the cursor position.
- Press the Ctrl key and use the mouse to select a rectangle in the profiling diagram. The viewer zooms to the selected area.

Synchronization

You can synchronize multiple hardware (CAPI) profiling, software profiling and waveform viewer windows by clicking the SyncAll button on the toolbar.

To exclude individual windows from resynchronization, click the Sync button on the title bar of the windows.

Profile window context menu

![Profile window context menu](image)

Figure 2-46 Profile window context menu
Right-click on the graph area of a Profile window to display its context menu (see Figure 2-46 on page 2-67). The menu entries available depend on the type of window.

**Export raw data…**

A Save As dialog is displayed to enable you to save the displayed profile information to a text file.

**Set colors…**

The Profile Color Settings dialog is displayed to enable you to configure the colors used for the different bar graph items (see Profile Settings Colors dialog on page 2-69).

**Set groups…**

The Profile Groups dialog is displayed to enable you to create new display categories (see Profiling Settings Groups dialog on page 2-70).

**x Axis Radix…**

The popup selection menu displays the current Radix.

To change the current Radix, select View in decimal format or View in hexadecimal format.

**y Axis Radix…**

The popup selection menu displays the current Radix.

To change the current Radix, select View in decimal format or View in hexadecimal format.

### 2.9.3 Displaying a profile from the component context menu

If a component supports profiling, you can right-click on it to display a **Profiling** entry in the context menu.

You can use the context menu to quickly display a profile for the selected component:

1. Right-click on the component to display the context menu.
2. Select **Profiling**, the stream to profile, and **Enable**. See Figure 2-47 on page 2-69.
   
   ____ Note _______
   
   AHB or AXI bus latency profiling must be enabled from the context menu before the Profiling Manager window for the profile streams to be displayed in the Profiling Manager window.
   
   _____

3. Run the simulation to record profiling information.
4. Right-click on the component and select the following from the context menu:
   - Profiling
   - the stream to profile
   - Display

The default X and Y axis values are used. To display other values, use the Profiling Manager dialog (see Figure 2-42 on page 2-63) or the Segment type and Segment size controls (if present) on the Profile window.

--- Note ---

Selecting **Profile → Channel name → Display** from the context menu automatically enables the channel if it was not already enabled.

![Figure 2-47 Profiling entry on the context menu](image)

### 2.9.4 Profile Settings Colors dialog

To change the colors for a Profile window, right-click on the window and select **Set colors** from the context menu. The Profile Settings Colors dialog (see Figure 2-48 on page 2-70) is displayed. Double-click on a color to select a new color from the color dialog.

Select **Apply to this stream** to use the colors on the selected Profile window. Select **Apply to all streams** to create a new color group.
2.9.5 Profiling Settings Groups dialog

The Profile Settings Groups dialog is used to specify recording specific events as a group (for example, recording all of the cache hits in one channel).

——— Note ————

Use the Profile Settings Global Groups dialog to specify how software profiling is done (see Software function profiling on page 2-94).

To create a new profiling group:

1. Display the Profile window for the component by right-clicking on the display area and selecting Set groups from the context menu.
2. Click on the **Add** group button to display the Add Group dialog.

3. Enter the name of the group and select a color for displaying the group.

4. Click **OK** to return to the Profile Settings Groups dialog and select the name of the new group from the **Groups** list.
5. Select an event from the column on the left of the dialog and click the $\rightarrow$ button to add the event to the group. Repeat until the required events are displayed in the **Group Events** list.

![Profile Settings Groups](image)

**Figure 2-51 Profile Settings Groups with events added**

6. Click **Close** to close the dialog and return to the Profile window.

7. Click on the group names to select the streams to display (see Figure 2-52 on page 2-73).
2.9.6 Profiling register values and transaction connections

You can profile the values assigned to a particular register by enabling tracing for that register. To profile the register values, open the CADI register view, right-click on the register and select **Enable/Disable Tracing**. After running the simulation, open the profiling manager, and select the profiling stream corresponding to that register to be displayed.

It is possible to trace the read/write events for a particular transaction connection:

1. Enable profiling for a transaction connection by right-clicking on the connection and selecting **Enable/Disable Tracing**.

2. Run the simulation to collect data until the simulation finishes.

3. Open the profiling manager.

4. Display the stream corresponding to that connection

**Note**

For AMBA® transaction connections for AXI, AHB, and APB, you can enable round-trip profiling for the connection by right-clicking on the connection and selecting **Profiler** from the context menu.
2.9.7 Latency profiling

To display the latency:

1. Right-click on a connection and select Profiler from the context menu.
2. The bus latency icon appears on the connection.

![Bus latency profile icon](image)

3. Right-click on the connection and select Profiler from the context menu.

![AHB latency profile](image)

--- Note ---

The dots displayed indicate transaction endings or any step of a transaction ending.

The positions of the dots in the graph area (see Figure 2-55 on page 2-75) indicate the type of transaction and latency by color and position, as follows:

**Dot color**  The dot color indicates the activity type. If two activities occur at the same point, the dot is split into two differently colored sections.
Dot position

The dot height indicates the length of a transaction or a transaction step. The position on the x axis indicates where a transaction or transaction step ended. The gap between dots shows that no transactions or transaction steps occurred between the two points.

**Figure 2-55 Dot position indicates AHB latency**

The Legend entries have the following meaning for AHB buses:

**Write Trans**

The total length of the write transaction, starting with `requestAccess` (if arbitration is required) or `address` (if arbitration is not required) until the last beat of data is done.

**Write-Initial**

The length of the first beat of write data, starting with `requestAccess` (if arbitration is required) or `address` (if arbitration is not required) until the beat of data is done.

**Write-Burst**

The length of any beat of write data, except the first one, starting with the cycle when the previous beat of data was done.

**Read Trans**

The total length of the read transaction, starting with `requestAccess` (if arbitration is required) or `address` (if arbitration is not required) until the last beat of data is done.
Read-Initial
The length of the first beat of read data, starting with requestAccess (if arbitration is required) or address (if arbitration is not required) until the beat of data is done.

Read-Burst
The length of any beat of read data, except the first one, starting with the cycle when the previous beat of data was done.

The Legend entries have the following meaning for AXI buses:

Write Trans
The total length of the write transaction, starting with accessing the address channel until the response phase is done.

Write-Initial
The length of the first beat of write data, starting with requestAccess (if arbitration is required) or address (if arbitration is not required) until the beat of data is done.

Write-Burst
The length of any beat of write data, except the first one, starting with the cycle when the previous beat of data was done.

Write Response
The length of response for a write transaction, starting with the cycle when the last beat of data was done.

Read Trans
The total length of the read transaction, starting with accessing the address channel until the last beat of data is done.

Read-Initial
The length of the first beat of read data, starting with accessing the address channel until the beat of data is done.

Read-Burst
The length of any beat of read data, except the first one, starting with the cycle when the previous beat of data was done.
AHB/AXI profiling views

Buses can be profiled by:
- latency
- events
- operation type
- address

Figure 2-56 shows the Profile Manager entries for a bus that has **Latency** and **Operation Type** options.

Click on the Profile icon of a context menu for a component, or select **Profiling Manager** from the **Debug** menu to display the **Profiling Manager**.

**Note**

You must enable latency profiling before you open the Profiling Manager for the latency streams to be included in the list of available streams. If you have already opened the Profiling Manager, close and reopen it to refresh the list.

---

**Figure 2-56 Profiling Manager dialog with an AHB component for latency streams**

The same profiler specification procedures enable profiling for different components, but different component protocols can offer different options for a profiler probe. For example, an AXI latency profiler probe can be inserted on any AXI connection by selecting **Profiler** from the context menu for the connection (see Figure 2-24 on page 2-43).
There are different profiling options for an AXI component. In the Profiling Manager dialog in Figure 2-57, the AXI profiler streams are selected, and the X-axis is being configured for Address.

![Profiling Manager dialog with an AXI component for latency streams](image)

**Figure 2-57 Profiling Manager dialog with an AXI component for latency streams**

- The Profile window for an AXI Latency stream profile displayed by Cycle and Latency is shown in Figure 2-58.

![Profiling AXI latency](image)

**Figure 2-58 Profiling AXI latency**
- The Profile window for an AHB **Latency** stream profile displayed by **Cycle** and **Latency** is shown in Figure 2-58 on page 2-78.

![Figure 2-59 Profiling AHB latency](image)

- Selecting the **Latency** stream with **Cycle** for the X axis and **Operation Type** for the Y axis with an AHB component results in the profile shown in Figure 2-60. The profile is displaying the operation types related to latency.

![Figure 2-60 Profiling AHB latency by number of operations](image)

- Selecting the AHB **Events** stream with **Cycle** for the X axis and **Operation Type** for the Y axis results in the profile shown in Figure 2-61 on page 2-80. The profile is displaying different bus events.
Figure 2-61 Profiling AHB bus events by number of operation types

- The profile for AXI Channel activity by Cycle and Channel event is shown in Figure 2-62.

Figure 2-62 Profiling AXI channel events

- Selecting the Events stream with Address for the X axis and Operation Type for the Y axis results in the profile shown in Figure 2-63 on page 2-81. The profile displays the operations that occur in different memory ranges.
Figure 2-63 Profiling AHB bus events by address location

- Figure 2-64 shows profiling of **Latency** by **Address**.

Figure 2-64 Profiling AXI Latency by Address

- Figure 2-65 on page 2-82 shows profiling of an AXI **Channel** stream profiled **Cycle** and **Address**.
Figure 2-65 Profiling AXI channels by cycle and address

Zooming in and changing the segment size (for the AHB bus events) results in dialogs similar to those in Figure 2-66 and Figure 2-67 on page 2-83.

Figure 2-66 Profiling AHB Latency (zoomed in)
Figure 2-67 Profiling AHB bus events (zoomed in)
2.10 Software analysis

This section describes the tools used to analyze software performance:

- Viewing memory contents
- Viewing registers on page 2-88
- Tracing register contents on page 2-91
- Viewing disassembled code on page 2-92
- Software function profiling on page 2-94
- Combined software and hardware profiling on page 2-104

2.10.1 Viewing memory contents

This section describes how to view and modify memory contents.

Setting a breakpoint on a memory location

To set a breakpoint on a memory location:

1. Left-click on a memory location to select it.
2. Right-click and select Insert/Remove Breakpoint from the context menu (see Figure 2-68).

![Figure 2-68 Insert breakpoint](image)

The cell with a breakpoint is highlighted with a red frame (see Figure 2-69 on page 2-85).
When the breakpoint is hit, the memory is highlighted with an orange background (see Figure 2-70).

If a breakpoint was hit while the Memory window was closed or in the background, the window is re-opened and brought to the foreground.

To keep a breakpoint location, but disable stopping if the location is hit, select **Enable/Disable Breakpoint** from the context menu. A disabled breakpoint is displayed with a thick black border (see Figure 2-71 on page 2-86).
The default breakpoint property is to break whenever the value changes. To change the breakpoint condition, select Edit Breakpoint Properties... from the context menu (see Figure 2-72).

The Memory window enables you to see and edit the contents of memory. The address area is synchronized with the Address Watch window and is updated to reflect the current address. The default memory view is the instruction memory, which is the same as the area of physical memory used for instructions.

Saving and loading a memory image

To save the current contents of the memory to a file, select Write memory image from the context menu (see Figure 2-73 on page 2-87).
To load the component memory from a file, select **Read memory image** from the context menu (see Figure 2-74).

**Note**

If you are reading a file and loading its contents into the component, the file characteristics must match. For example:

- If the file was saved as ASCII, it must be selected as ASCII for reloading.
• If the file was saved with the Space setting of PROG, the Memory window must have PROG selected when the file is reloaded.

Typically, the Block setting must also match unless you are deliberately moving memory from one block to another.

2.10.2 Viewing registers

Components that implement a CADI interface permit you to view and modify the registers (see Figure 2-75). These are displayed in the menu item, Debug → View Registers for…. You can also access the register view from the context menu.

![Register view](image)

Use the tabs on the bottom of the Register window to change the register bank displayed. The list shows values currently held in the selected register bank.

Note

Not all components have multiple register banks.

Registers are colored red if the value changes during simulation steps.
To change the value of a register, click on a register value and edit the contents (see Figure 2-76). You can enter a value in hexadecimal (with prefix 0x) or decimal.

--- Note ---

You can use the underline character “_” to separate groups of digits. For example, 0xabcd_1234 is the same as 0xabcd1234.

![Figure 2-76 Changing the register value](image)

To save the current values of the registers, select Copy to Clipboard or Save to File from the context menu for the Register window.

To set a breakpoint on a register value, click on the register and select Insert/Remove Breakpoint from the context menu (see Figure 2-77 on page 2-90). A register with a breakpoint set displays a red dot in front of the register name. When the breakpoint is hit the register is highlighted with an orange background.

If a breakpoint was hit while the Register window was closed or in the background, the window is re-opened and brought to the foreground.
Figure 2-77 Register window context menu

Figure 2-78 Register breakpoint
2.10.3 Tracing register contents

To trace the contents of a register,

1. Display the registers for the component.
2. Use the context menu to enable tracing for a single register (or all registers) as shown in Figure 2-79.

![Figure 2-79 Trace register value](image)

3. The register entry is added to the list of channels displayed in the Trace Manager (see Trace Manager and the Waveform Viewer windows on page 2-51).

   A tracing icon is displayed next to the register to indicate that it is currently being traced.

4. Select Waveform Viewer from the Debug menu (or click on the Wave icon).
5. The Waveform Viewer window is displayed with the register added under the Channel list.
2.10.4 Viewing disassembled code

Components that implement a CADI interface for disassembly enable you to view the disassembly at the current program counter (PC) or at any specified address (see Figure 2-80).

To view the disassembly for a processor, select the processor and then select one of the following options:

- Select **Debug → View Disassembly**.
- Right-click on the component and select **View Disassembly** from the context menu.

![Disassembly view](image)

**Figure 2-80 Disassembly view**

The green arrow always displays the current execution address or the program address of the instruction of the reference stage in the core model.

Symbol information for address labels is also displayed in the code.

Use the control buttons in the Disassembly window to run, stop, and reset the simulation. Use the **Step into** and **Step over** buttons to instruction step the simulation.

--- **Note** ---

The instruction step may translate into more than one simulation cycle.
Check the **Duration** box to display the number of cycles used for the instructions. If an instruction has not yet been reached, the number of cycles is not displayed.

To set a breakpoint:

1. Click on a location and then right-click to display the context menu.
2. Select the option **Insert/Remove Breakpoint** or **Continue Until Here**.

![Breakpoint context menu for code](image)

3. The simulation runs until the breakpoint activates.
4. The current PC (green arrow) is shown at the selected address when the breakpoint occurs (see Figure 2-82 on page 2-94).
2.10.5 Software function profiling

SoC Designer Simulator provides the capability to profile the software running on programmable components. Features include displaying of the calling graph over time in addition to generation of statistics for each function.

To take advantage of this powerful feature, components must support the ELF format with debug info. Components supporting this feature have the menu item Profiling/Software in their context menu. Check the Enable submenu item to activate the feature, then open the Profiling window from the Display context sub-menu.

——— Note ————

To support software profiling, the ELF/AXF application must be loaded into the processor core through SoC Designer Simulator. This can be done in addition to loading the application through a software debugger.

To display the Function Profiling window for a core perform one of the following options:

- Right-click the core component in the Diagram window and select Profiling → Software → Display from the context menu.
- Click the Profile button to display the Profile Manager window and enable the Software stream for the core.
The Diagram panel (see Figure 2-83) of the Function Profiling window contains two columns:

- On the left the function names of the loaded object files are listed.
- On the right hand side the calling graph is displayed in the time domain.

The Summary panel of the Function Profiling window contains major column groups:

- On the left the function names of the loaded object files are listed.
- On the right hand side the profile information for each category is displayed in a separate column within the panel.
Diagram view of the Function Profiling window

Click on a function to select it. You can then use the keyboard arrow keys to navigate to each call.

The selected function is displayed in red (see Figure 2-84) and the cursor can be navigated to the beginning or end of each function using the arrow keys. The cursor can also be placed with the mouse by clicking on the diagram.

By default, the function in the summary row is selected. If the cursor is moved with the keyboard arrow keys, the simulation stops at every function.

You can control which functions are displayed in the window:

- To move a function from the bottom of the window to the top, select the function and press and hold the left mouse button and move the function to a new location in the list.

- To use a filter list to control which functions are displayed, right-click in the Function Profiling window and select Configure View from the context menu. Use the Diagram tab of the Configuration dialog to create filter functions. (See Software Profiling Configuration dialog on page 2-99.)
Right-clicking on a function name displays the context menu:

**Insert/Remove Breakpoint**

Selecting this entry toggles a breakpoint for the selected function. The breakpoint appears in the Disassembly window.

You can also set or reset a breakpoint directly by double-clicking on an entry in the **Function** column.

**Enable/Disable Breakpoint**

Selecting this entry toggles between enabling or disabling an existing breakpoint for the selected function. A disabled breakpoint is indicated by a gray disk instead of a red disk.

**Export Raw Data**

Selecting this entry displays a dialog for saving the function trace information to a text file.

**Configure View**

Selecting this entry displays the configuration dialog (See *Software Profiling Configuration dialog* on page 2-99.

**Summary view of the Function Profiling window**

Click the **Summary** tab in the Function Profiling window to display a table showing totals for various categories by function. The left panel lists the functions and the right panel contains statistics for the software or hardware activities related to that function.

--- **Note** ---

By default, only software statistics (such as Average Duration) are displayed in the **Summary** view. To also display hardware statistics for the component, use the **Summary** tab of the Software Profiling Configuration dialog to select hardware categories.

The row and column order and size can be changed to make the summary more readable:

**Selected rows**

To change the functions that are displayed:

1. Click on the Diagram tab in the Profiling window to display the trace view.
2. Click **Configure View** from the context menu to display the Software Profiling Configuration dialog.
3. Click on the **Diagram** tab and create a filter group.
Row sort order

To change the order of rows in the table, click on the column title. Repeated clicking switches between sorting by function and sorting by column value.

Selected columns

To change the categories that are displayed:
1. Click **Configure View** from the context menu to display the Software Profiling Configuration dialog.
2. Click on the **Summary** tab to display the current software and hardware streams.
3. Check the streams to record. Streams that are not checked do not have a column present in the Profile window.

Column order

To change the column order for the listed statistics, select a column title and press and hold the left mouse key while moving the cursor to the left or right.

Column size

To change the column size, move the mouse cursor over the column separators in the title bar. The mouse cursor changes into the resize symbol as shown in Figure 2-85. Press and hold the left mouse button and drag the column border to change the width.

![Function Profiling for Core0](image)

**Figure 2-85 Changing the width of the major column**

Right-clicking on a row in the window displays the context menu:

**Insert/Remove Breakpoint**

Selecting this entry toggles a breakpoint for the selected function. The breakpoint appears in the Disassembly window.
Enable/Disable Breakpoint

Selecting this entry toggles between enabling or disabling an existing breakpoint for the selected function. A disabled breakpoint is indicated by a gray disk instead of a red disk.

Copy Summary to Clipboard

Select this entry displays a dialog for copying the summary information to a paste buffer. (This option is only visible from the Summary tab context menu.) The copied data is separated by tabs and line feeds so that it can easily be pasted into a spreadsheet application such as Excel.

Save Summary to File

Selecting this entry displays a dialog for saving the summary information to a .csv file. (This option is only visible from the Summary tab context menu.) The saved data is separated by commas and line feeds so that it can loaded directly by database tools such as Excel or by a custom application.

Configure View

Selecting this entry displays the configuration dialog (See Software Profiling Configuration dialog.)

Software Profiling Configuration dialog

To configure software profiling:

1. Select a function and right-click to display the context menu.
2. Select Configure view… from the menu.

The Software Profiling Configuration dialog is displayed (see Figure 2-86 on page 2-100).
3. Select the display mode as **cycles** or **time** from the **Scale** drop-down list.

4. Use the check boxes in the **Function Name Filter** to select how the functions are displayed in the Profile window.

5. The Filter Function Calls panel enables suppressing the display of functions. This might be useful if, for example, you did not want to see library or initialization functions displayed in the Profile window.
6. The functions to be ignored are entered by clicking on the New button in the Configure dialog. The Create New Filter Group dialog is displayed.

7. Enter the file name for the text tile and click OK to open the standard editor on the respective platform.

8. Enter the functions you want to include or exclude in a text file (see Figure 2-87) and save the file.

![Figure 2-87 List of functions in text editor](image)

9. After creating one or more groups, each group can be selected or deselected using the Enable/Disable buttons.

10. Use the Exclude or Include radio buttons in the Filter Action panel to select whether the functions in the group file are excluded from display or whether only the functions in the group files are displayed.

11. Click on the Summary tab of the dialog to display the Summary Columns information shown in Figure 2-88 on page 2-102.
12. Select the details to record for the software and hardware by checking the appropriate boxes:

- If, for example, all hardware streams and two software profile entries are chosen from the configuration dialog list, the window displays the two columns for the software (for Figure 2-89 on page 2-103, this is No Calls and Duration) and a column for each of the hardware streams.

**Figure 2-88 Software Profiling Configuration dialog, Summary information**

- configure hardware data collection:
  - Collect summary data only for selected hardware streams - slightly faster simulation
  - Collect summary data for all enabled hardware streams
If only the core and two software profile entries are chosen from the summary configuration dialog list, the window displays the two columns as shown in Figure 2-90 on page 2-104.
13. Use the Hardware panel of the Software Profiling Configuration dialog to profile hardware accesses and software functions calls together (see Combined software and hardware profiling).

Select the components (or one or more activities for a component) from the list.

14. From the radio buttons at the bottom dialog, select whether you want to record all details for the hardware streams or only record the selected details.

15. Click OK to close the dialog and return to the Profile windows.

### 2.10.6 Combined software and hardware profiling

SoC Designer Simulator can display hardware and software profiles that enable you to relate hardware activity to specific function calls.

To display the profiles:

1. Enable software profiling for the core and enable profiling for the selected component. Use the Profile Manager or right-click to display the context menus for the processor and the individual components.

2. Display the Function Profiling window and the Profiling windows for the hardware activities.

---

**Note**

Set the Segment type in the component’s Profile window to **Function based**.
See also the general setup information in *Software Profiling Configuration dialog* on page 2-99.

3. Adjust the zoom and start cycle for each window so that the displays cover the same range (see Figure 2-91 on page 2-106).

4. Click the **Sync All** button on the toolbar to synchronize the hardware profiling windows with the software profiling and waveform viewer windows.

5. Examine the windows to find relationships between the two profiles.
   For the windows in Figure 2-91 on page 2-106, the cache activity is much higher for `memcpy()` than for `wait_for_trigger()`.
Figure 2-91 Hardware and Function Profiling windows
Chapter 3
Batch Mode Simulation

This chapter describes batch mode simulation. It contains the following sections:

• Introduction on page 3-2
• Scripted batch mode simulation on page 3-3
3.1 Introduction

SoC Designer provides a batch mode to run simulations in the background or run regression tests without any user interaction. SoC Designer additionally supports extensive scripting features that permit the control of all aspects of simulation and verification of SoC Designer systems.

There are three ways to run a simulation in batch mode (using SDRuntime.bat for Windows and sdsimruntime on Linux):

**run for a specific number of cycles:**

```
> sdsimruntime -s 10000 simfile.mxs
```

**run indefinitely**

```
> sdsimruntime -r simfile.mxs
```

**run from a script**

From within the SoC Designer Simulator, open an MXSCR script file from the File menu or toolbar. This displays a Script dialog (see Figure 3-1 on page 3-3) with Step, Run, Stop, and Reset button controls to operate the script.

--- Note ---

To toggle breakpoints in a script, double-click on a line.

---

Alternatively, specify a script file to run from the command line as follows:

```
> sdsimruntime --script example.mxscr
```

--- Note ---

If running SoC Designer indefinitely, the simulation ends only if any of the components calls the halt() command or if an error (or fatal error) occurs. If the option Stop on Warning is set in the SoC Designer preferences, a warning message also causes the simulation to be aborted.

To capture output in batch mode, regular output is directed to stdout. Warnings, errors, and fatal errors go to stderr.
3.2 Scripted batch mode simulation

When you launch a batch mode script, SoC Designer Simulator executes commands from a text file that resembles source code file (see Figure 3-1).

The name of the scripting language is MxScript, which is based on the C language. For information, see the MxScript Reference Manual.

--- Note ---

When you launch a script from the command line, you cannot combine script options, command options, and runfile options in the same line.

---

![Figure 3-1 MxScript dialog](image-url)
3.2.1 Output

Use the -l option to redirect output to a log file:

sdsimruntime -l logfile.txt

This option is for batch mode only. Specifying a log file causes all output to stdout and stderr to be redirected to that file. This option is particularly useful on Windows because other methods of output redirection are not possible.
Chapter 4
Debugging

This chapter describes debugging components and applications in a C++ environment. It contains the following sections:

• *Introduction* on page 4-2
• *Component level debugging* on page 4-3
• *System level debugging and probes* on page 4-14
4.1 Introduction

Debugging a simulation can have the following meanings:

- The embedded software developer must debug the application code that is running on the system. Debugging application code is known as target-level debugging. An example of a target-level debugger is the ARM® RealView® Debugger.

  ARM® RealView® Debugger, if used, must be connected externally or put into a startup script for the SoC Designer Simulator. RealView Debugger is only available from ARM.

  In this case the designer is debugging the (simulated) contents of a CPU or memory component. (See Component level debugging on page 4-3.)

- The system designer must examine signals and transactions between components to ensure correct system operation. This is known as System Level Debugging. For information, see System level debugging and probes on page 4-14.

  This type of target-level debugging is typically done using the built-in debug features of SoC Designer.

- An engineer developing a component must debug component behavior and code to ensure that the component operates correctly.
4.2 Component level debugging

SoC Designer offers built-in debug windows for registers and memories of components based on the CADI debug interface. This makes it possible for any component to have its own register and memory windows. Any component with CADI interfaces appears in the list of debuggable components provided by the SoC Designer Simulator. To see this list, click \textbf{Debug} \rightarrow \textbf{View registers / memory for} … You can also right-click on a component instance in the SoC Designer Simulator diagram window, to see that component’s supported debugging options.

See the following sections for more information on the windows that are available for debugging at the component level:

- \textit{Register windows}
- \textit{Memory windows} on page 4-8
- \textit{Disassembly window} on page 4-11
- \textit{Synchronization of external debuggers} on page 4-13

4.2.1 Register windows

If a component defines registers in its CADI interface, they can be viewed in the SoC Designer Simulator (see Figure 4-1).

![Figure 4-1 Viewing CADI defined registers](image-url)
Registers are organized in groups that can be selected using the tabs at the bottom of the register window. Each register can have a different bitwidth. Use the **Register Radix** submenu to select how the register contents are displayed.

If a register is not read-only, you can modify its value by selecting the current value and entering the new one. You can enter new values in hex, decimal, binary, or octal:

- hex values must be preceded with `0x`.
- binary values must be preceded with `0b`.
- octal values must be preceded with `0`.

**Note**

You can use the underline character “_” to separate groups of hexadecimal digits. For example, `0xabcd_1234` is the same as `0xabcd1234`.

The display format of each register is specified by the debug interface of the component itself. Registers may be displayed as hex or decimal values, floating values, symbolic values, booleans, or strings. By default the display format is chosen according to the definition of the registers in the component. However it is possible to force the display to hex or decimal for all registers by selecting the desired option in the context menu.

**Register breakpoints**

Breakpoints can be set on a register by double-clicking on its name or from the context menu. A red dot in front of the register name indicates that the breakpoint is set (see Figure 4-2 on page 4-5). A default breakpoint causes the simulation to stop if the respective value changed. From the breakpoint properties dialog (context menu) the breakpoint condition can be modified to stop the simulation if the register is equal to, greater than, or less than a specified value.
Register tracing

Each register can be traced by selecting **enable/disable tracing** in the context menu (see Figure 4-3 on page 4-6). The name of the trace consists of the component name, followed by the register group name and the register name. The generated trace can then be viewed in the waveform viewer or exported to VCD.

It is also possible to enable/disable the tracing for an entire register group by selecting the option **trace all registers in group** in the context menu.
After selecting the registers and signals to display, you can select View waveform as → Graph to display one or more of the channels as a graph. This can be used to record and display combinations of register statistical information and hardware activity in the same window (see Figure 4-5 on page 4-7).
If the graph is not readable with the size selected, select the channel and then select Configure Graph View from the context menu and enter the range in the Minimum Value and Maximum Value fields. The Channel Height field controls the height of the channel graph in the Waveform Viewer window.

![Graph View Settings](image)
4.2.2 Memory windows

If a component defines memories in its CADI interface then they can be viewed in SoC Designer Simulator (see Figure 4-7).

![Memory window](image)

Figure 4-7 Viewing CADI defined memories

A component can have multiple memory spaces that can each have different properties, such as size and bitwidth. Components can define memory blocks that can be used to quickly navigate to a certain address in the memory space.

If a memory is not read-only, its contents can be edited by selecting the desired memory cell and entering the new value. Values are always displayed and entered in hexadecimal format.

**Loading files into memories**

For debug purposes, it is possible to upload files into a memory space using the component’s CADI interface. The files are treated as memory images.
The context menu of the memory window features an item called Read memory image that opens the Read memory dialog shown in Figure 4-8.

Use this dialog to specify the file name to load and the following options:

**Start Address**
The address where the image is to be loaded (default is 0)

**Addresses to Load**
How many words to load (default is given by the file size)

**Load File as Binary**
The file is read in binary mode

**Load File as ASCII**
The file is interpreted as containing only ASCII coding of hex values.

Pressing OK loads to the memory area defined by the start address. The number of bytes loaded is whichever is smaller of the file size or the value in Bytes to Load.

---

**Note**
You can use the underline character “_” to separate groups of digits. For example, 0xabcd_1234 is the same as 0xabcd1234.
The binary mode must be used for loading binary images. The ASCII mode provides an easy way to write custom images by hand. The ASCII reader reads the hex values in a file and loads them into the memory as bytes.

**Sample ASCII file**

Loading a file with a contents as specified above into an empty memory (initialized with zeroes) results in the following memory contents:

```
0x0000 00 12 34 55 0a 0b 0c 04 0x0008
05 10 20 30 00 00 00 00
```

**Note**

This feature is intended for debugging purposes only. Applications must be loaded at simulation start from the Load Applications dialog. This is why it is necessary for simulated systems to include a component that contains an object loader.

**Dumping memory images to a file**

It is possible to store the contents of a memory window to a file in binary or ASCII format. Right-clicking on a memory window that supports the **Write memory image** feature opens a dialog that enables you to select the file and dump options.

**Start Address**

Start address of the memory range to write (default is 0).

**End Address**

End address of the memory range to write (default is 65535 – for 64k).

**Write File as Binary**

The file is written in binary mode.

**Write File as ASCII**

The file is written as ASCII.

If the dialog is closed using the **OK** button, the file is written from the start address to the end address as specified in the dialog.
Memory breakpoints

Breakpoints on memory locations can be set by double-clicking on the desired memory location or by right-clicking and selecting from the context menu. A red box around the value indicates that the breakpoint is set. From the context menu it is also possible to define the breakpoint properties that define the break condition. The default break condition is `break on change`. Other options are break if equal to, not equal to, greater than, or less than a specified value.

4.2.3 Disassembly window

If a component implements the disassembly API, the disassembly can be viewed directly in SoC Designer Simulator.

![Disassembly for CoreG (Instruction Count: 150)](image)

Figure 4-9 Disassembly for a component

The green arrow shown in Figure 4-9 always displays the current execution address or the program address of the instruction of the reference stage in the core model. The control buttons in the disassembly window enable you to run, stop, or reset the simulation and instruction step the simulation.

--- Note ---

The instruction step may translate into more than one simulation cycle. Check the Cycles box to display the number of cycles for the instruction step.
The following features are supported:

- Run to debuggable point
- Locate current instruction
- Run
- Stop
- Step into
- Step over (if supported by the model)
- Reset

Also it is possible to let the simulation run until a certain point by right-clicking on an instruction and selecting **Continue Until Here** from the context menu.

Breakpoints can be set on a disassembly line by double-clicking on the first column of the window. If you scroll through code, it may not be convenient to determine the correct disassembly mode (for example, 32bit or 16bit mode). You can select a mode from a drop down list box. As soon as the simulation continues however, the mode switches back to the current mode of simulation (as reported by the model) and all instructions below the current instruction are displayed in the current mode.

The “run to debuggable point” feature, the first button on the Disassembler window, forces the processor into a coherent state. When debugging with the ARM RealView Development Suite (RVDS), the model is brought to the debug point automatically whenever a software breakpoint is hit (including single stepping). However, if a hardware breakpoint is reached, or if the system is advanced by cycles within SoC Designer, the model can get to a non-debuggable state. In this case, the run to debug point will advance the processor to the debug state. Once that has been accomplished, the model will cause the system to stop simulating. The run to debug point feature is also available as a context menu item (Run to Debuggable Point) from the component within SoC Designer Simulator.

### 4.2.4 Debuggers

To debug the application source for a processor, select the **Launch Debugger** option from the context menu of the processor component or from the **Debug** menu. Use the **Step** or **Run** buttons in SoC Designer Simulator to advance the code pointer in the attached debugger. If your system uses an ARM processor, you can attach the ARM® RealView® Debugger shown in Figure 4-10 on page 4-13.

The debugger that is started depends on the processor component:
4.2.5 Synchronization of external debuggers

All connected debuggers are automatically synchronized by the debug server built into SoC Designer Simulator. The debuggers are synchronized every time the simulation starts, stops, or steps. In some cases, however, it is desirable to establish an additional synchronization. For example, if one debugger has changed a resource and it has not been noticed by the other debugger. In that case, it is possible to select the Refresh External Debuggers from the Debug menu or use the keyboard shortcut Ctrl+Shift+F12.
4.3 System level debugging and probes

Debug probes can be attached to any existing connection between two components. SoC Designer offers several built-in probes and also enables the usage of user-defined probes. To attach a probe to a connection, right-click on a connection and select an option from the popup-menu as shown in Figure 4-11.

The following sections describe the probe types:

- **Breakpoint probes**
- **Monitor probes** on page 4-18
- **Tracing probes** on page 4-21

### 4.3.1 Breakpoint probes

Use breakpoint probes to let the simulation run until a transaction takes place on a connection. Breakpoints can be configured to set a break condition (for example, you can break to stop if the address equals 0).

The red dots in front of the two ports represent connection breakpoints (see Figure 4-12 on page 4-15).
Figure 4-12 Breakpoint on a connection

Right-click on a red dot to enable modification of breakpoint properties (see Figure 4-13 on page 4-16).
All transaction and signal breakpoints are displayed in the Breakpoint Manager dialog, opened from the Debug menu or the Brkpts button.
To set breakpoint conditions, right-click on a breakpoint and select **Edit Breakpoint Properties** from the context menu (see Figure 4-15).

Figure 4-14 Breakpoint Manager dialog

Figure 4-15 Breakpoint conditions dialog for a signal or transaction
4 Debugging

4.3 System level debugging and probes

——— Note ————

For information about signal, transaction, or register breakpoint conditions, see Breakpoint Condition dialog on page 2-44.

For information about software breakpoints, see Software analysis on page 2-84.

4.3.2 Monitor probes

This section describes the steps required to connect monitor probes:

- Monitors for synchronous transactions
- Bus-Master port monitors on page 4-20
- Monitors for signals on page 4-21

Monitors for synchronous transactions

Attach a monitor to a connection to view the transactions that take place each cycle.

The default dialog (displayed when a monitor is created) shows the last transaction. If there was a transaction in the current cycle, the information is displayed in red, otherwise it is gray. The cycle number of the last transaction is displayed in the upper left corner of the dialog.
Figure 4-16 Monitor attached to a connection

A small symbol is displayed on the two ends of the connection. Use the **History** button to enlarge the monitor dialog to view a history of the transactions.

Figure 4-17 Monitor for synchronous transactions

The **Elapsed Cycles** count is the number of cycles that have been executed since the monitor probe was attached. The **Activity/Cycle** displays the percentage of transactions per cycle.
**Bus-Master port monitors**

You can attach monitors to bus-master ports.

A bus-master port can be connected to multiple slaves. However, in general the bus permits only one slave to be accessed in each cycle (through arbitration). As a result, adding only one monitor for the bus master can replace the requirement for the multiple monitors attached to each connection to the slaves.

Bus-master port monitors are very similar to the transaction monitors. The bus-master monitor collects the transactions going to all the slaves of that bus-master port. This collection contains the transactions that would have been collected by multiple individual connection-based monitors on all of the attached slaves. The only difference between the bus-master monitor and the connection-based transaction monitor is that the bus-master monitor shows also the target slave that the current transaction accesses.

![Figure 4-18 Attaching a bus-master port monitor](image)

To attach a bus-master monitor, right-click on the bus-master port (for example, the bmaster port of the Bus component as shown in Figure 4-18), and select **Insert/Remove Monitor**.
Monitors for signals

For signals, the monitors display the passed value only. Similar to transaction monitors, a history view is available that keeps a history of signal events.

Signal monitors, similarly to their transaction counterparts, are capable of generating or injecting a signal. Right-clicking on the monitor dialog displays this option. This results in a call to `driveSignal()`.

Note
Check the Log transactions to a file box to enable transaction dumping to a file.

4.3.3 Tracing probes

Use tracing probes to log selected transactions to memory for display in the waveform window or for storage in a VCD or similar type of transaction recording file. VCD files can be viewed with a standard GTKWave tool or other standard, publicly available viewers. Set a log file with the Settings page of the trace manager. This is initially set to the system name with `.vcd` as the extension.

If a tracing probe is attached to a connection, it is represented by the trace symbol.

Each time a tracing probe is created, a dialog opens that you can use to define the bit-widths to be traced.

![Tracer Properties](image)

Figure 4-19 Add trace dialog for signal traces
You can use current settings as a default for the next trace you add. It is also possible to turn off the prompting and use only the defaults so that no dialog is displayed when a trace is added. By default, each added trace is automatically displayed in the waveform viewer (see Trace Manager and the Waveform Viewer windows on page 2-51).

--- Note ---

SoC Designer permits the insertion of tracing probes at any time. Logging only begins after the insertion of a probe. The state before the insertion is displayed as unknown.
Appendix A

Keyboard Shortcuts

This appendix lists keyboard shortcuts. It has the following section:

• SoC Designer Simulator shortcuts on page A-2
# A.1 SoC Designer Simulator shortcuts

Table A-1 lists keyboard shortcuts for SoC Designer Runtime.

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + O</td>
<td>File → Open existing System</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + S</td>
<td>File → Save the current system</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + F4</td>
<td>File → Close the currently open system</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + Q</td>
<td>File → Exit</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + H</td>
<td>View → Simulation Hierarchy</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + G</td>
<td>View → Toggle Grid Display</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + ‘+’</td>
<td>View → Zoom in</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + ‘-’</td>
<td>View → Zoom out</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + 1</td>
<td>View → Zoom to 100%</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + F</td>
<td>View → Zoom to Fit</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + R</td>
<td>View → Center on Selected Objects</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + I</td>
<td>View → Center Diagram</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + Shift + F5</td>
<td>Control → Restart Simulation (Hard Reset)</td>
<td>Any window</td>
</tr>
<tr>
<td>Ctrl + Shift + R</td>
<td>Control → Reset Simulation</td>
<td>Any window</td>
</tr>
<tr>
<td>F5</td>
<td>Control → Run</td>
<td>Any window</td>
</tr>
<tr>
<td>Shift F5</td>
<td>Control → Stop</td>
<td>Any window</td>
</tr>
<tr>
<td>Ctrl + F11</td>
<td>Control → Step one Cycle</td>
<td>Any window</td>
</tr>
<tr>
<td>Ctrl + Shift + F11</td>
<td>Control → Step n Cycles</td>
<td>Any window</td>
</tr>
<tr>
<td>Ctrl + Shift + A</td>
<td>Debug → Animate All</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + B</td>
<td>Debug → Breakpoint Manager</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + W</td>
<td>Debug → Toggle Waveform Viewer</td>
<td>Main window</td>
</tr>
<tr>
<td>Ctrl + Shift + F12</td>
<td>Debug → Refresh External Debuggers</td>
<td>Any window</td>
</tr>
<tr>
<td>Ctrl + A</td>
<td>Select All</td>
<td>Main window</td>
</tr>
</tbody>
</table>
## Table A-1 SoC Designer Simulator shortcuts (continued)

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + D</td>
<td>Show Diagram/Show Open Windows</td>
<td>Any window</td>
</tr>
<tr>
<td>Ctrl + E</td>
<td>Bring Focus To Main Window</td>
<td>Any window</td>
</tr>
<tr>
<td>Ctrl + Tab</td>
<td>Bring next Runtime Window in Foreground</td>
<td>Any window</td>
</tr>
<tr>
<td>F10</td>
<td>Instruction Step Over (skipping calls)</td>
<td>Disassembly window</td>
</tr>
<tr>
<td>F11</td>
<td>Instruction Step Into</td>
<td>Disassembly window</td>
</tr>
<tr>
<td>Ctrl + F5</td>
<td>Run to Debuggable Point</td>
<td>Disassembly window</td>
</tr>
<tr>
<td>Left/Right Arrow</td>
<td>Move cursor to previous/next event</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Home/End</td>
<td>Move cursor to first/last event</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Ctrl + Left/Right</td>
<td>Move cursor to previous/next cycle</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Ctrl + Home/End</td>
<td>Move cursor to first/last cycle</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Up/Down Arrow</td>
<td>Select previous/next channel</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Alt + Up/Down Arrow</td>
<td>Move selected channel up/down one position</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Plus/Minus</td>
<td>Show/Hide sub-channels (of selected channel)</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Del</td>
<td>Remove current channel or sub-channel from waveform viewer</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>L/R</td>
<td>Insert left/right locator at cursor position</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Shift + L/R</td>
<td>Remove left/right locator</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Ctrl + L/R</td>
<td>Move cursor to left/right locator</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Ctrl + Plus/Minus</td>
<td>Zoom in/out</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Ctrl+Shift + Plus/Minus</td>
<td>Zoom to next/previous setting (zoom redo/undo)</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Ctrl + F</td>
<td>Zoom to fit</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Ctrl + A</td>
<td>Zoom to range defined by L/R locators</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>Ctrl + 1</td>
<td>Reset zoom to 1 cycle per pixel</td>
<td>Waveform Viewer</td>
</tr>
<tr>
<td>C</td>
<td>Center view on cursor</td>
<td>Waveform Viewer</td>
</tr>
</tbody>
</table>
ARM acknowledges and thanks the respective owners for the following software that is used by our product:

- *Verilog Parser* on page B-2
B.1 Verilog Parser

/*
 * Copyright (c) 1992, Bakul Shah <bvs@BitBlocks.com>
 * All Rights Reserved.
 *
 * Permission to use, copy, modify, distribute or sell this software and
 * its documentation for any purpose is hereby granted without fee, subject
 * to the following conditions.
 *
 * 1. The above copyright, this list of conditions and the following
 *    disclaimer must appear UNCHANGED in all copies of the software and
 *    related documentation, and their derivative works or modified
 *    versions
 *
 * 2. Binary distribution must be accompanied by documentation that
 *    reproduces the above copyright, this list of conditions and the
 *    following disclaimer.
 *
 * 3. Any modifications to the source code must be clearly marked as such.
 *
 * 4. The author's name may not be used to endorse or promote products
 *    derived from this software without specific prior written permission.
 *
 * This software is provided by the author AS IS. The author DISCLAIMS
 * any and all warranties of merchantability and fitness for a particular
 * purpose. In NO event shall the author be LIABLE for any damages
 * whatsoever arising in any way out of the use of this software.
 */
This chapter describes the SoC Designer Model Library configuration file. It contains the following section:

- *Configuration file contents* on page C-2
C.1 Configuration file contents

The SoC Designer Model Library configuration file (CONF file) contains the information about the models that are accessible from the SoC Designer Developer Suite.

`maxlib.conf` is a legacy name for a library configuration file. A library configuration file can be renamed. For example, `my_core_comp.conf` is also a valid name for configuration file.

To include the contents of another configuration file, use the `@include` directive in one of the following formats:

- `@include <conf file installation path>/<file-name>.conf`
- `@include "<conf file installation path>/<file-name>.conf"

For each component, the configuration file that can make it available to SoC Designer contains an entry of the following type:

```plaintext
component "MyModel"
{
    type = "Other";
    version="1.0.0";
    path_linux = ".../lib/Linux/libMyModel.so";
    release_with_debug_path_linux = ".../lib/Linux/libMyModel.so";
    path_win32 = ".../lib/Win32/Release/libMyModel.dll";
    release_with_debug_path_win32 = ".../lib/Win32/Debug/libMyModel.dll";
}
```

--- Note ---

A configuration file can contain paths to components that are available on more than one platform, where needed. The illustrated path variables are not required in cases where components for to the corresponding platforms are not available.

---

The component `type` must reflect the component type defined in the model. If none is defined, the type `Other` is assumed.

--- Caution ---

The name of the model must be identical to the name (including case) that is returned by the component factory and by the `getName()` function of the component. If it is different, SoC Designer cannot recognize and load the component.

---

To distinguish different versions in SoC Designer, you must also include this information in the `version` entries of the library configuration file. If this is not done, the correct version is not found when the simulation is started. If there is only one version listed in the file, the `version` entries are optional.
Note
The paths are to files with extension .dll for Windows or .so for Linux.

The path name can be absolute or relative. A relative path is always relative to the location of the library configuration file. Environment variables, enclosed by ${ and }, are also supported.

Examples for legal paths (Windows and/or Linux):
- \path = "/home/user/tmp/MyModel.so";
- \path = "C:\MyModels\MyModel.dll";
- \path = "../lib/MyModel.so";
- \path = "${MYLOCATION}/lib/MyModel.so";

If more than one entry with the same model name and same version number is found in the configuration files, only the first entry is used. The others are ignored and SoC Designer issues a warning in its output window.

There exist other optional fields that you can add:
- **documentation_file**. Use this field to specify the document that is opened by the “Documentation” button in the Component Properties window. SoC Designer determines which application will open the documentation by its filename extension. “.txt” extension files are opened with the text editor of choice (set with the File -> Preferences -> General -> Text Editor option). “.pdf” files are opened by the Adobe Acrobat Reader. “.htm” files are opened by the browser specified in the File -> Preferences -> General -> Browser option (Linux only). On Windows, the default web browser is used.
  For example: documentation_file = "my_component.pdf";

- **alias**. Use this field to specify any aliases for the component. This is useful when the user wants to change the name of the component after the component had been used in other projects already. Existing .mxp files using the old component name may still be used after the component name is changed if the component .conf file specifies the old name as an alias to itself.
  For example: alias = “old_name”;

This section contains a list of terms and their definitions. The majority of the terms are common engineering terms, so the definitions tend to define the terms with respect to developer suite applications.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AHB</strong></td>
<td>Advanced High-performance Bus, a bus interface defined by the AMBA\textsuperscript{®} 2 Specification.</td>
</tr>
<tr>
<td><strong>AMBA</strong></td>
<td>Advanced Microcontroller Bus Architecture, the ARM open standard for multi-master on-chip buses that are capable of running with multiple masters and slaves.</td>
</tr>
<tr>
<td><strong>APB</strong></td>
<td>Advanced Peripheral Bus, a simplified AMBA bus interface designed for use with ancillary or general-purpose peripherals.</td>
</tr>
<tr>
<td><strong>AXI</strong></td>
<td>Advanced eXtensible Interface, a bus interface defined by the AMBA\textsuperscript{®} 3 Specification.</td>
</tr>
<tr>
<td><strong>Component</strong></td>
<td>An individual core, memory, bus, etc. A component can also be an entire subsystem. In this case, the subsystem is placed into another system as a component.</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>A link between two components by using one master and one slave port. There are three types of connections: clock, signal, and transaction.</td>
</tr>
<tr>
<td><strong>External Port</strong></td>
<td>A port that is used to connect the current system within another system.</td>
</tr>
<tr>
<td><strong>Internal Port</strong></td>
<td>A port of an individual component, used to establish connections between the component and other components/external ports within the system.</td>
</tr>
<tr>
<td><strong>IP-XACT</strong></td>
<td>A standard provided by the SPIRIT consortium. IP-XACT-compliant IP descriptions store configuration and interconnection information for the IP block in XML files.</td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>An annotation in the designer diagram window.</td>
</tr>
<tr>
<td><strong>Master</strong></td>
<td>A component or port that initiates a transaction or drives a signal.</td>
</tr>
<tr>
<td><strong>CADI</strong></td>
<td>ESL API Debug Interface, enables reading and writing memory and register values and also provides the interface to external debuggers.</td>
</tr>
<tr>
<td><strong>CASIMMI</strong></td>
<td>SoC Designer Memory Map Interface, used to define and use memory maps for bus master components.</td>
</tr>
</tbody>
</table>
CAPI  ESL API Profiling Interface, enables collecting historical data from a component and displaying the results in various formats.

CASI  ESL API Simulation Interface, is based on the SystemC communication library and manages the interconnection of components and communication between components.

SoC Designer Model Library (MaxLib)  
The repository of SoC Designer components.

Mega Model  A SoC Designer system consisting of one or more components.

Object  An object is one of the following: a component, a connection, an external port, or a label.

Probe  Object inserted on a connection for the purpose of, for example, tracing, monitoring, breakpoints.

Runtime  Plus Runtime is an environment for running binary versions of system models. These models cannot be modified by the user and the source for the component libraries is not provided.

Signal connection  The signal-based interface is very close to hardware simulators in that it simulates every signal independently.

Slave  A component or port that responds to a transaction or signal.

Simulation Run  This term is used for the combination of loading, running, and closing a system.

SPIRIT  (Structure for Packaging, Integrating, and Re-using IP), a consortium that promotes the IP-XACT standard for IP interconnection.

System  A system is a collection of components connected to collaborate in a simulation.

Transaction connection  The transaction-based interface encapsulates a group of signals into one read or write transaction.