

ARM® Design Simulation Model

Revision: r1p0

User Guide



ARM Design Simulation Model

User Guide

Copyright © 2005, 2013, 2015 ARM. All rights reserved.

Release Information

The following changes have been made to this book.

Change history			
Date	Issue	Confidentiality	Change
20 January 2005	A	Non-Confidential	First release for r0p0
30 August 2013	B	Non-Confidential	Second release for r1p0
30 September 2013	C	Non-Confidential	Third release for r1p0
12 June 2015	D	Non-Confidential	Fourth release for r1p0

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. 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 third party 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.

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 covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms. 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 of the Agreement shall prevail.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow ARM’s trademark usage guidelines at <http://www.arm.com/about/trademark-usage-guidelines.php>

Copyright © 2005, 2013, 2015 ARM. All rights reserved. ARM Limited or its affiliates.

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20348

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

<http://www.arm.com>

Contents

ARM Design Simulation Model User Guide

	Preface	
	About this book	vi
	Feedback	viii
Chapter 1	Introduction	
	1.1 About Design Simulation Models	1-2
	1.2 DSM package contents	1-4
	1.3 Simulation with DSMs	1-5
Chapter 2	Initial DSM Configuration	
	2.1 How to extract the DSM	2-2
	2.2 How to test the DSM	2-3
	2.3 Simulator interfacing	2-4
Chapter 3	Using a Model	
	3.1 Limitations of use	3-2
Appendix A	Revisions	

Preface

This preface introduces the *ARM® Design Simulation Model (DSM) User Guide*. It contains the following sections:

- *About this book on page vi.*
- *Feedback on page viii.*

About this book

This book is for the *ARM Design Simulation Model* (DSM).

Product revision status

The *rm**pn* identifier indicates the revision status of the product described in this book, for example, r1p0, where:

<i>rm</i>	Identifies the major revision of the product, for example, r1.
<i>pn</i>	Identifies the minor revision or modification status of the product, for example, p0.

Intended audience

This book is written for experienced hardware engineers, software engineers and System-on-Chip (SoC) designers who might have experience of ARM products. You are expected to have experience of Verilog.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for a high-level description of DSMs and how they are used in simulations. This chapter describes the contents of the package and potential simulation inaccuracies when using DSMs.

Chapter 2 *Initial DSM Configuration*

Read this for a description of how to install, set up, and test a DSM.

Chapter 3 *Using a Model*

Read this for a description of the timing issues for synthesizable and non-synthesizable cores, the top-level registers, and how to control instruction stream tracing. This chapter also describes some of the limitations of DSMs.

Appendix A *Revisions*

Read this for a description of the technical changes between released issues of this book.

Glossary

The *ARM® Glossary* is a list of terms used in ARM documentation, together with definitions for those terms. The *ARM® Glossary* does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

The *ARM® Glossary* is available on the ARM Infocenter at <http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html>.

Conventions

Conventions that this book can use are described in:

- *Typographical conventions on page vii.*

Typographical conventions

The following table describes the typographical conventions:

Style	Purpose
<i>italic</i>	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
monospace	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
<u>monospace</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
monospace <i>italic</i>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
monospace bold	Denotes language keywords when used outside example code.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the <i>ARM® glossary</i> . For example, IMPLEMENTATION DEFINED, UNKNOWN, and UNPREDICTABLE.

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter <http://infocenter.arm.com>, for access to ARM documentation.

See onARM <http://www.onarm.com>, for embedded software development resources including the *Cortex Microcontroller Software Interface Standard (CMSIS)*.

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to errata@arm.com. Give:

- The title.
- The number, ARM DUI 0302D.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

———— **Note** —————

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1

Introduction

This chapter introduces the *ARM Design Simulation Model* (DSM). It contains the following sections:

- *About Design Simulation Models on page 1-2.*
- *DSM package contents on page 1-4.*
- *Simulation with DSMs on page 1-5.*

1.1 About Design Simulation Models

DSMs are cycle accurate, simulation models that you can include in a range of target HDL simulators. Each DSM is specific to a host platform. The DSM fully matches the architecture and functionality of the RTL model.

The DSMs are derived directly from the RTL model. DSMs can function with a wide-range of industry-standard Verilog simulators. DSM execution speeds are in the range of 5-500 cycles per second, depending on:

- The simulator interface efficiency.
- The complexity of the design in which it is instantiated.
- The complexity of the original design.

The DSM consists of:

- A functional core block.
- A Verilog wrapper.

The wrapper uses the foreign language interface of the host simulator to instantiate the functional model. The DSM is generally derived from the RTL source of the ARM design using the *Verilator* compiler. For some ARM products, this might be augmented with extra functionality, such as Tarmac trace, added by ARM.

The DSM interfaces to the wrapper using technology developed by ARM to enable a single compiled model to function with a variety of logic simulators. For some products, the DSMs might include behavioral debug facilities, such as Tarmac trace. When you use compiled models, it enables distribution of models without compromising the intellectual property that they embody.

———— Note ————

For synthesizable cores, the DSM is a pre-implementation model and not a sign-off model.

1.1.1 Features of ARM DSMs

[Table 1-1](#) shows the main features of ARM DSMs.

Table 1-1 ARM DSM features

Feature	Description
Full device functionality	The DSM fully matches the architecture and functionality of the RTL model.
Phase accuracy	You can expect the DSM to exhibit the same intra-cycle timing as the RTL model.

Table 1-1 ARM DSM features (continued)

Feature	Description
Register visibility	<p>The DSM might provide debug visibility of the registers within models of processors, depending on the processor. The register set of a core for all modes represented in the architecture might be visible in a special layer of Verilog hierarchy inside the DSM, depending on the processor. These registers are available for tracing in your simulation.</p> <p>———— Note ————</p> <ul style="list-style-type: none"> • Some out-of-order processors might not offer any visibility. • This feature does not apply if your ARM product is not a processor, or does not include a processor.
Cache and memory size configuration	<p>You can configure the size of the cache, or TCM, for each particular DSM instance, where applicable.</p> <p>———— Note ————</p> <p>This feature does not apply if your ARM product has no such configuration, or where your DSM generation flow does not support multiple configurations.</p>
Disassembler	<p>Some DSMs also provide a built-in disassembler. The availability of the disassembler varies from core to core and depends on the availability of a suitable execution tracer built into the RTL of the core from which the DSM is derived.</p> <p>———— Note ————</p> <p>This feature does not apply if your ARM product does not have a disassembler.</p>

1.2 DSM package contents

Each DSM contains the following components:

- DSM Verilog and SystemVerilog wrappers.
- DSM library, as a dynamic .so file and static .a file.
- One or more implementation libraries as .so files.
- A testbench file to check the DSM setup.
- Documentation.

1.3 Simulation with DSMs

Figure 1-1 shows the integration of DSMs into an HDL simulation. During simulation, the DSM interfaces to the simulator by using a simulator-specific model manager supplied with the DSM. The model manager is a dynamically loaded library that uses the simulator API to interface between the simulator and the DSMs in the design. The simulator invokes the model manager through the HDL wrapper. This presents a module that consists of the connections to the device, as the appropriate device *Technical Reference Manual* (TRM) describes, into the logic simulator.

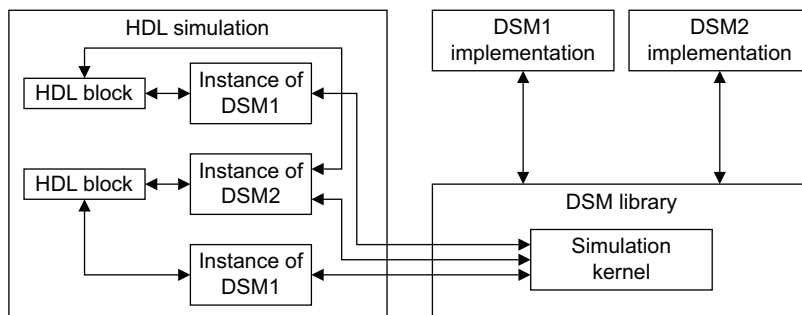


Figure 1-1 DSM integration

Note

For synthesizable cores, the DSM is a pre-implementation, pre-synthesis model. It does not contain any scan test insertion or BIST and is not a *Sign-Off Model* (SOM).

Chapter 2

Initial DSM Configuration

This chapter describes the functionality of the *ARM Design Simulation Model* (DSM).

It contains the following sections:

- [*How to extract the DSM on page 2-2.*](#)
- [*How to test the DSM on page 2-3.*](#)
- [*Simulator interfacing on page 2-4.*](#)

2.1 How to extract the DSM

To extract the DSM package, execute the `.sh` executable and agree to the licence terms, if applicable.

2.2 How to test the DSM

Execute the `_TESTBENCH.sh` executable within the expanded package to check that operation is valid. The DSM is contained within a testbench that is supplied with the package. The script requires that you select a system-installed simulator. You can specify the simulator to use by setting one of the following parameters:

<code>vcs</code>	Selects the VCS simulator.
<code>ius</code>	Selects the Cadence Incisive simulator.
<code>mti</code>	Selects the QuestaSim simulator.

A test that executes correctly prints the following message at the end of the simulation:

```
DSMINATOR: *** TEST PASSED ***
```


2.3 Simulator interfacing

You can use the DSM by instantiating an instance of the DSM module in the simulated Verilog. The interface of the DSM is the same as the RTL module it replaces. You can instantiate the DSM using either a Verilog PLI or a SystemVerilog DPI interface.

This section contains the following subsections:

- [Verilog PLI interface.](#)
- [SystemVerilog DPI Interface.](#)
- [Running a simulation on page 2-5.](#)

2.3.1 Verilog PLI interface

For all simulators, the module search path must include the path to the DSM Verilog wrapper. In all the supported simulators, perform this by adding to the command line `-y` followed by the path to the directory containing the Verilog wrapper. Make sure that Verilog is the preferred libext extension.

VCS

Add the following to the VCS command line:

```
+acc -P [PATH TO]/[DSM MODULE NAME].tab [PATH TO]/[DSM MODULE NAME].so
```

Cadence Incisive simulator

Add the following to the `irun` command line:

```
+loadvpi=[PATH TO]/[DSM MODULE NAME].so:registerVPI
```

ModelSim

Add the following to the `vsim` command line:

```
-pli [PATH TO]/[DSM MODULE NAME].so
```

2.3.2 SystemVerilog DPI Interface

Because most simulators use the Verilog version of a module instead of a SystemVerilog version, if found, you must copy the `.sv` file to a separate directory. In the same way as with the Verilog version, add this directory to the search path using the `-y` command line option.

VCS

Add the following to the VCS command line:

```
[PATH TO]/[DSM MODULE NAME].so
```

Cadence Incisive simulator

Add the following to the `irun` command line:

```
-sv_lib [PATH TO]/[DSM MODULE NAME].so
```

ModelSim

Add the following to the `vsim` command line:

```
-sv_lib [PATH TO]/[DSM MODULE NAME]
```

2.3.3 Running a simulation

You must set the `DSM_MODEL_PATH` or `DSM_MODEL_PATH_[DSMNAME]` environment variable to point to the directory that contains the DSM implementation libraries. Not doing so results in a simulation exit with the following error:

```
DSM ERROR: DSM_MODEL_PATH_[DSMNAME] and DSM_MODEL_PATH environment variables have not been set
```

If no acceptable DSM implementations are found in this directory, the simulation exits with the following error:

```
DSM ERROR: Could not find an appropriate model ([DSM NAME])  
DSM ERROR: Generate a new model with config:[CONFIG PARAMETERS]
```

The value of the configuration parameters must be fed back to the DSM supplier to generate an appropriate DSM library for the configuration options used in the instantiation.

If the system has been set up correctly, the following message is issued:

```
DSMINATOR: Configured a module [DSM NAME] ([VERSION]) at [HIERARCHICAL PATH]
```

Chapter 3

Using a Model

This chapter describes how to use the model.

It contains the following sections:

- [*Limitations of use on page 3-2.*](#)

3.1 Limitations of use

Although DSMs match the architecture and functionality of the appropriate core designs, they are subject to the limitations that the following sections describe:

- *Unsupported simulator functions.*
- *Internal scan chain modeling.*
- *Caches and registers.*

3.1.1 Unsupported simulator functions

The following simulator function is not supported:

Save and Restore, also known as checkpointing

Save the simulation at a determined point of time, also known as a snapshot, and restore the simulation to that point of time.

Power-aware simulation

Power-aware simulation.

3.1.2 Internal scan chain modeling

DSMs are derived from the RTL description of the core that they model. The final netlist for the core might contain internal scan chains that were added during synthesis. It is not possible to use DSMs to model these scan chains because they do not exist in the device RTL. However, the *Sign-Off Model* (SOM) of a device models the scan chains.

3.1.3 Caches and registers

Although it might be possible to view the register values contained in the DSM simulation, depending on the processor, it is not possible to introduce any test data directly into the caches or registers because this cannot be performed in the RTL from which the DSM is derived.

———— Note ————

Some out-of-order processor might not offer any visibility.

Appendix A

Revisions

This appendix describes the technical changes between released issues of this book.

Table A-1 Issue A

Change	Location	Affects
First release	-	-

Table A-2 Differences between issue A and issue B

Change	Location	Affects
Deleted Chapter 3, Timing Issues	Chapter 3	r1p0
Removed descriptions relating to VHDL	Throughout book	r1p0

Table A-3 Differences between issue B and issue C

Change	Location	Affects
Removed ‘ARM publications’ section because it incorrectly referred to two documents that do not yet exist.	<i>Additional reading on page vii</i>	r1p0
Changed description for ‘Register visibility’ to indicate that visibility might be possible, but that it is not guaranteed. Also changed the description to indicate that for some out-of-order processors, visibility might not be possible at all.	<i>Features of ARM DSMs on page 1-2</i>	r1p0
	<i>Caches and registers on page 3-2</i>	r1p0

Table A-4 Differences between issue C and issue D

Change	Location	Affects
Revised description to clarify that certain functionality, such as Tarmac trace, does not apply for all ARM products.	<i>Features of ARM DSMs on page 1-2</i>	r1p0
Added notes to some features to clarify that these do not always apply.	<i>Features of ARM DSMs on page 1-2</i>	r1p0
Clarified that this process might not apply.	<i>How to extract the DSM on page 2-2</i>	r1p0
Modified search paths.	<i>SystemVerilog DPI Interface on page 2-4</i>	r1p0