PrimeCell® TrustZone
Protection Controller (BP147)
Cycle Model
Version 9.1.0

User Guide
PrimeCell TrustZone Protection Controller (BP147) Cycle Model
User Guide

Copyright © 2017 ARM Limited. All rights reserved.

Release Information

The following changes have been made to this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Issue</th>
<th>Confidentiality</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 2017</td>
<td>A</td>
<td>Non-Confidential</td>
<td>Restamp release.</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.

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. You must follow the ARM trademark usage guidelines http://www.arm.com/about/trademarks/guidelines/index.php.

Copyright © ARM Limited or its affiliates. All rights reserved.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

In this document, where the term ARM is used to refer to the company it means “ARM or any of its subsidiaries as appropriate”. 
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

Chapter 1. Using the Cycle Model in SoC Designer

TZPC BP147 Cycle Model Functionality ...........................................2
  Fully Functional Features ..................................................2
  Unsupported Hardware Features ..........................................2
  Features Additional to the Hardware ....................................3

Adding and Configuring the SoC Designer Component ....................3
  SoC Designer Component Files ..........................................3
  Adding the Cycle Model to the Component Library .....................4
  Adding the Component to the SoC Designer Canvas ...................4

Available Component ESL Ports ..............................................5
  APB Transaction Slave Interface ...........................................5
  Signal Master Interfaces ...................................................6

Setting Component Parameters ..............................................7

Debug Features .................................................................8
  Registers Information ....................................................8

Available Profiling Data .....................................................9
Preface

A Cycle Model component is a library developed from ARM intellectual property (IP) that is generated through Cycle Model Studio™. The Cycle Model then can be used within a virtual platform tool, for example, SoC Designer.

About This Guide

This guide provides all the information needed to configure and use the Cycle Model in SoC Designer.

Audience

This guide is intended for experienced hardware and software developers who create components for use with SoC Designer. You should be familiar with the following products and technology:

• SoC Designer
• Hardware design verification
• Verilog or SystemVerilog programming language
## Conventions

This guide uses the following conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>courier</strong></td>
<td>Commands, functions, variables, routines, and code examples that are set apart from ordinary text.</td>
<td><code>sparseMem_t SparseMemCreateNew();</code></td>
</tr>
<tr>
<td><em>italic</em></td>
<td>New or unusual words or phrases appearing for the first time.</td>
<td><em>Transactors</em> provide the entry and exit points for data ...</td>
</tr>
<tr>
<td><strong>bold</strong></td>
<td>Action that the user performs.</td>
<td>Click <em>Close</em> to close the dialog.</td>
</tr>
<tr>
<td><code>&lt;text&gt;</code></td>
<td>Values that you fill in, or that the system automatically supplies.</td>
<td><code>&lt;platform&gt;/</code> represents the name of various platforms.</td>
</tr>
<tr>
<td><code>[ text ]</code></td>
<td>Square brackets [ ] indicate optional text.</td>
<td><code>$CARBON_HOME/bin/modelstudio [ &lt;filename&gt; ]</code></td>
</tr>
<tr>
<td>`[ text1</td>
<td>text2 ]`</td>
<td>The vertical bar</td>
</tr>
</tbody>
</table>

Also note the following references:

- References to C code implicitly apply to C++ as well.
- File names ending in .cc, .cpp, or .cxx indicate a C++ source file.
Further reading

This section lists related publications. The following publications provide information that relate directly to SoC Designer:

- *SoC Designer Installation Guide*
- *SoC Designer User Guide*
- *SoC Designer Standard Component Library Reference Manual*

The following publications provide reference information about ARM® products:

- *AMBA 3 AHB-Lite Overview*
- *AMBA Specification (Rev 2.0)*
- *AMBA AHB Transaction Level Modeling Specification*
- *Architecture Reference Manual*

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

The following publications provide additional information on simulation:

Glossary

**AMBA**  *Advanced Microcontroller Bus Architecture.* The ARM open standard on-chip bus specification that describes a strategy for the interconnection and management of functional blocks that make up a System-on-Chip (SoC).

**AHB**  *Advanced High-performance Bus.* A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol.

**APB**  *Advanced Peripheral Bus.* A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports.

**AXI**  *Advanced eXtensible Interface.* A bus protocol that is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.

**Cycle Model** A software object created by the Cycle Model Studio (or Cycle Model Compiler) from an RTL design. The Cycle Model contains a cycle- and register-accurate model of the hardware design.

**Cycle Model Studio** Graphical tool for generating, validating, and executing hardware-accurate software models. It creates a Cycle Model, and it also takes a Cycle Model as input and generates a component that can be used in SoC Designer, Platform Architect, or Accellera SystemC for simulation.

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

**CADI**  *ESL API Debug Interface,* enables reading and writing memory and register values and also provides the interface to external debuggers.

**CAPI**  *ESL API Profiling Interface,* enables collecting historical data from a component and displaying the results in various formats.

**Component** Building blocks used to create simulated systems. Components are connected together with unidirectional transaction-level or signal-level connections.

**ESL**  *Electronic System Level.* A type of design and verification methodology that models the behavior of an entire system using a high-level language such as C or C++.

**HDL**  *Hardware Description Language.* A language for formal description of electronic circuits, for example, Verilog.

**RTL**  *Register Transfer Level.* A high-level hardware description language (HDL) for defining digital circuits.

**SoC Designer** High-performance, cycle accurate simulation framework which is targeted at System-on-a-Chip hardware and software debug as well as architectural exploration.

**SystemC** SystemC is a single, unified design and verification language that enables verification at the system level, independent of any detailed hardware and software implementation, as well as enabling co-verification with RTL design.

**Transactor**  *Transaction adaptors.* You add transactors to your component to connect your component directly to transaction level interface ports for your particular platform.
Chapter 1

Using the Cycle Model in SoC Designer

This chapter describes the functionality of the Cycle Model component, and how to use it in SoC Designer. It contains the following sections:

• TZPC BP147 Cycle Model Functionality
• Adding and Configuring the SoC Designer Component
• Available Component ESL Ports
• Setting Component Parameters
• Debug Features
• Available Profiling Data
1.1 TZPC BP147 Cycle Model Functionality

The PrimeCell TrustZone Protection Controller (TZPC) is an AMBA compliant System-on-Chip peripheral. The TZPC provides a software interface to the protection bits in a secure system in a TrustZone design. It provides system flexibility to enable you to configure different areas of memory as secure or non-secure. The actual TrustZone information is configured via a set of component registers. The status of the registers in turn is reflected in the signal master ports of this component.

The BP147 can be connected in two ways:

1. The prot0, prot1, and prot2, signal master interface ports provide security information to an external decoder, e.g. the PL301.
2. The size signal master port provides a region size information that is used, for example, by the TrustZone Memory Adaptor (BP141).

This section provides a summary of the functionality of the Cycle Model compared to that of the hardware. For details of the functionality of the hardware that the Cycle Model simulates, refer to the ARM PrimeCell Infrastructure AMBA 3 TrustZone Protection Controller (BP147) Technical Overview.

- Fully Functional Features
- Unsupported Hardware Features
- Features Additional to the Hardware

1.1.1 Fully Functional Features

The following features of the TZPC BP147 hardware implementation are fully implemented in the BP147 TZPC Cycle Model.

- Protection bits to enable you to program up to 24 areas of memory as secure or non-secure.
- Secure region bits to enable you to split an area of internal RAM into both secure and non-secure regions.
- An AMBA APB system interface.
- Does not generate any APB wait states or a slave error response, and is therefore compatible with the AMBA 2 APB protocol.

1.1.2 Unsupported Hardware Features

The following features of the TZPC BP147 hardware are not implemented in the BP147 TZPC Cycle Model:

- SCAN control related signals are not supported in the Cycle Model: SCANENABLE, SCANINPCLK, and SCANOUTPCLK.
1.1.3 Features Additional to the Hardware

The following features that are implemented in the TZPC BP147 Cycle Model to enhance usability do not exist in the TrustZone Protection Controller (BP147) hardware:

- The Cycle Model supports enabling of debug information - See Debug Features for more information.

1.2 Adding and Configuring the SoC Designer Component

The following topics briefly describe how to use the component. See the SoC Designer User Guide for more information.

- SoC Designer Component Files
- Adding the Cycle Model to the Component Library
- Adding the Component to the SoC Designer Canvas

1.2.1 SoC Designer Component Files

The component files are the final output from the Cycle Model Studio compile and are the input to SoC Designer. There are two versions of the component; an optimized release version for normal operation, and a debug version.

On Linux the debug version of the component is compiled without optimizations and includes debug symbols for use with gdb. The release version is compiled without debug information and is optimized for performance.

On Windows the debug version of the component is compiled referencing the debug runtime libraries, so it can be linked with the debug version of SoC Designer. The release version is compiled referencing the release runtime library. Both release and debug versions generate debug symbols for use with the Visual C++ debugger on Windows.

The provided component files are listed below:

<table>
<thead>
<tr>
<th>Platform</th>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>maxlib.lib&lt;model_name&gt;.conf</td>
<td>SoC Designer configuration file</td>
</tr>
<tr>
<td></td>
<td>lib&lt;component_name&gt;.mx.so</td>
<td>SoC Designer component runtime file</td>
</tr>
<tr>
<td></td>
<td>lib&lt;component_name&gt;.mx_DBG.so</td>
<td>SoC Designer component debug file</td>
</tr>
<tr>
<td>Windows</td>
<td>maxlib.lib&lt;model_name&gt;.windows.conf</td>
<td>SoC Designer configuration file</td>
</tr>
<tr>
<td></td>
<td>lib&lt;component_name&gt;.mx.dll</td>
<td>SoC Designer component runtime file</td>
</tr>
<tr>
<td></td>
<td>lib&lt;component_name&gt;.mx_DBG.dll</td>
<td>SoC Designer component debug file</td>
</tr>
</tbody>
</table>

Additionally, this User Guide PDF file is provided with the component.
1.2.2 Adding the Cycle Model to the Component Library

The compiled Cycle Model component is provided as a configuration file (.conf). To make the component available in the Component Window in SoC Designer Canvas, perform the following steps:

1. Launch SoC Designer Canvas.
2. From the File menu, select Preferences.
3. Click on Component Library in the list on the left.
4. Under the Additional Component Configuration Files window, click Add.
5. Browse to the location where the SoC Designer Cycle Model is located and select the component configuration file:
   - maxlib.lib<model_name>.conf (for Linux)
   - maxlib.lib<model_name>.windows.conf (for Windows)
6. Click OK.
7. To save the preferences permanently, click the OK & Save button.

The component is now available from the SoC Designer Component Window.

1.2.3 Adding the Component to the SoC Designer Canvas

Locate the component in the Component Window and drag it out to the Canvas.
1.3 Available Component ESL Ports

Table 1-2 describes the ESL ports that are exposed in SoC Designer. See the *ARM PrimeCell TrustZone Protection Controller (BP147) Technical Overview* for more information.

**Table 1-2 ESL Component Ports**

<table>
<thead>
<tr>
<th>ESL Port</th>
<th>Description</th>
<th>Direction</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>apb</td>
<td>APB3 slave port for accessing the TZPC memory mapped registers. See section 1.3.1 below.</td>
<td>Input</td>
<td>Transaction slave</td>
</tr>
<tr>
<td>reset</td>
<td>Input reset. Reset port for receiving reset signal.</td>
<td>Input</td>
<td>Signal slave</td>
</tr>
<tr>
<td>clk-in</td>
<td>Input clock. This component can be connected to the clock master. If it is unconnected, it is implicitly connected to the system master clock.</td>
<td>Input</td>
<td>Clock slave</td>
</tr>
<tr>
<td>prot0</td>
<td>Protection bits. The output should be connected to an external bus decoder or bridge component to provide security settings (for example, a PL301). See section 1.3.2 below.</td>
<td>Output</td>
<td>Signal master</td>
</tr>
<tr>
<td>prot1</td>
<td>Protection bits. The output should be connected to an external bus decoder or bridge component to provide security settings (for example, a PL301). See section 1.3.2 below.</td>
<td>Output</td>
<td>Signal master</td>
</tr>
<tr>
<td>prot2</td>
<td>Protection bits. The output should be connected to an external bus decoder or bridge component to provide security settings (for example, a PL301). See section 1.3.2 below.</td>
<td>Output</td>
<td>Signal master</td>
</tr>
<tr>
<td>size</td>
<td>R0, secure RAM size. The output should be connected to a PL141 TrustZone Memory Adaptor. See section 1.3.2 below.</td>
<td>Output</td>
<td>Signal master</td>
</tr>
</tbody>
</table>

All pins that are not listed in this table have been either tied or disconnected for performance reasons.

1.3.1 APB Transaction Slave Interface

The APB interface allows you to program the memory mapped registers. This interface is expected to be connected to an AXI compliant interface via some AXI2APB bridge. The transaction interface uses the CASI methods *read*, *write*, *readDbg*, and *writeDbg* for communicating the APB interface signals. This is defined in the SoC Designer documentation for the MxAPB component that is shipped as part of the AMBA2 model archive.
1.3.2 Signal Master Interfaces

The three signal master interfaces (prot0, prot1, and prot2) have a bit-width of 8. Each of the bit signals correspond to the TZPCDECPROT signals of the IP block and can be connected to an external bus decoder component to provide security settings. A signal of 0 configures the corresponding decoder address region to “secure”, whereas an asserted value is considered “non-secure”. The reset state of all pins is 0 (deasserted). The status of this port is communicated as a single value via the first parameter of the driveSignal port method. The second parameter of the driveSignal method is unused. The method is called whenever at least one bit of the interface status gets changed.

The size interface corresponds to the TZPCR0SIZE signal and is used to set the secure RAM size in the TrustZone Memory Adaptor. The bit-width of this signal is 10. The status of this port is communicated as a single value via the first parameter of the driveSignal port method. The second parameter of the driveSignal method is unused. The method is called whenever at least one bit of the interface status gets changed.

For further information, refer to the ARM PrimeCell TrustZone Protection Controller Technical Overview.
1.4 Setting Component Parameters

You can change the settings of all the component parameters in SoC Designer Canvas, and of some of the parameters in SoC Designer Simulator. To modify the Cycle Model parameters:

1. In the Canvas, right-click on the component and select **Edit Parameters...** You can also double-click the component. The **Edit Parameters** dialog box appears.

2. In the **Parameters** window, double-click the **Value** field of the parameter that you want to modify.

3. If it is a text field, type a new value in the **Value** field. If a menu choice is offered, select the desired option. The parameters are described in Table 1-3.

**Table 1-3 Component Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Runtime¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>Align Waveforms</td>
<td>When set to true, waveforms dumped from the component are aligned with the SoC Designer simulation time. The reset sequence, however, is not included in the dumped data. When set to false, the reset sequence is dumped to the waveform data, however, the component time is not aligned with the SoC Designer time.</td>
<td>true, false</td>
<td>true</td>
<td>No</td>
</tr>
<tr>
<td>apb Base Address</td>
<td>Base address of the non-secure APB interface.</td>
<td>0x0 - 0xffffffff</td>
<td>0x0</td>
<td>No</td>
</tr>
<tr>
<td>apb Enable Debug Messages</td>
<td>Whether debug messages are logged for the apb port.</td>
<td>true, false</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>apb Size</td>
<td>Region size of the non-secure APB interface.</td>
<td>0x0-0xfffffffff</td>
<td>0x100000000</td>
<td>No</td>
</tr>
<tr>
<td>Carbon DB Path</td>
<td>Sets the directory path to the database file.</td>
<td>Not Used</td>
<td>empty</td>
<td>No</td>
</tr>
<tr>
<td>Dump Waveforms</td>
<td>Whether SoC Designer dumps waveforms for this component.</td>
<td>true, false</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>Enable Debug Messages</td>
<td>Enable or disable the capture of debug messages.</td>
<td>true, false</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>Waveform File²</td>
<td>Name of the waveform file.</td>
<td>string</td>
<td>arm_cm_Tzpc_bp147.vcd</td>
<td>No</td>
</tr>
<tr>
<td>Waveform Timescale</td>
<td>Sets the timescale to be used in the waveform.</td>
<td>Many values in drop-down</td>
<td>1 ns</td>
<td>No</td>
</tr>
</tbody>
</table>

1. Yes means the parameter can be dynamically changed during simulation, No means it can be changed only when building the system, Reset means it can be changed during simulation, but its new value will be taken into account only at the next reset.

2. When enabled, SoC Designer writes accumulated waveforms to the waveform file in the following situations: when the waveform buffer fills, when validation is paused and when validation finishes, and at the end of each validation run.
1.5 Debug Features

The TZPC BP147 Cycle Model has a debug interface (CADI) that allows the user to view, manipulate, and control the registers in the SoC Designer Simulator, or any simulator that supports CADI.

1.5.1 Registers Information

This section lists the register views available for the TZPC BP147 model in the SoC Designer Simulator. The TZPC BP147 model has two sets of registers that are accessible via the debug interface. Registers are grouped into sets according to functional area.

- APB Registers
- PrimeCell ID Registers

See the *ARM PrimeCell TrustZone Protection Controller (BP147) Technical Overview* for detailed descriptions of these registers.

1.5.1.1 APB Registers

Table 1-4 shows the APB registers.

**Table 1-4 APB Registers**

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TZPCR0SIZE</td>
<td>Secure RAM Region Size register read-write</td>
<td></td>
</tr>
<tr>
<td>TZPCDECPROT$x$Stat</td>
<td>Decode Protection Status registers read-only</td>
<td></td>
</tr>
<tr>
<td>TZPCDECPROT$x$Set</td>
<td>Decode Protection Set registers write-only</td>
<td></td>
</tr>
<tr>
<td>TZPCDECPROT$x$Clr</td>
<td>Decode Protection Clear registers write-only</td>
<td></td>
</tr>
</tbody>
</table>

1. There are three sets of these registers - where $x = 0, 1, 2$ - for each master interface.

1.5.1.2 PrimeCell ID Registers

Table 1-5 shows the PrimeCell Identification and Peripheral Identification registers.

**Table 1-5 PrimeCell ID Registers**

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>periph_id_0</td>
<td>Peripheral Identification register 0 (bits 7:0). Identifies the part number of the peripheral.</td>
<td>read-only</td>
</tr>
<tr>
<td>periph_id_1</td>
<td>Peripheral Identification register 1 (bits 15:8). Identifies the part number and designer of the peripheral.</td>
<td>read-only</td>
</tr>
<tr>
<td>periph_id_2</td>
<td>Peripheral Identification register 2 (bits 23:16). Identifies the revision and designer of the peripheral.</td>
<td>read-only</td>
</tr>
<tr>
<td>periph_id_3</td>
<td>Peripheral Identification register 3 (bits 31:24). Identifies the configuration of the peripheral.</td>
<td>read-only</td>
</tr>
<tr>
<td>cell_id_0</td>
<td>TZPC Identification register 0 (bits 7:0). Determines the reset value.</td>
<td>read-only</td>
</tr>
</tbody>
</table>
1.6 Available Profiling Data

The TZPC BP147 Cycle Model component has no profiling capabilities.

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell_id_1</td>
<td>TZPC Identification register 1 (bits 15:8). Determines the reset value.</td>
<td>read-only</td>
</tr>
<tr>
<td>cell_id_2</td>
<td>TZPC Identification register 2 (bits 23:16). Determines the reset value.</td>
<td>read-only</td>
</tr>
<tr>
<td>cell_id_3</td>
<td>TZPC Identification register 3 (bits 31:24). Determines the reset value.</td>
<td>read-only</td>
</tr>
</tbody>
</table>