Cortex-A72 Cycle Model
User Guide

Copyright © 2016 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 2016</td>
<td>A</td>
<td>Non-Confidential</td>
<td>Restamping/AXF Load update</td>
</tr>
<tr>
<td>April 2016</td>
<td>B</td>
<td>Non-Confidential</td>
<td>Release 8.2.0. r0p2.</td>
</tr>
<tr>
<td>August 2016</td>
<td>C</td>
<td>Non-Confidential</td>
<td>Update for semihosting support.</td>
</tr>
<tr>
<td>November 2016</td>
<td>D</td>
<td>Non-Confidential</td>
<td>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 or 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

Preface

About This Guide .................................................. 7
Audience ............................................................. 7
Conventions ............................................................ 8
Further reading ...................................................... 9
Glossary ............................................................... 9

Chapter 1.
Using the Cycle Model Component in SoC Designer Plus

Cortex-A72 Functionality .............................................. 11
Features Additional to the Hardware .............................. 12
Adding and Configuring the SoC Designer Plus Component .................................................. 13
SoC Designer Plus Component Files ......................... 13
Adding the Cycle Model to the Component Library .......... 14
Adding the Component to the SoC Designer Canvas .......... 14
ESL Ports ................................................................. 14
Available Component ESL Ports ................................. 14
Reset Behavior and Ports ........................................... 15
Tied Pins ................................................................. 15
Setting Component Parameters .................................. 16
Debug Features ........................................................ 22
Register Information .................................................. 22
AArch32 Core Registers ............................................. 23
AArch64 Core Registers ............................................. 24
AArch32 Control Registers ......................................... 24
AArch64 System Registers .......................................... 27
AArch32 Debug Registers ........................................... 29
External Debug Registers .......................................... 30
AArch64 Debug Registers ........................................... 31
AArch32 ID Registers .................................................. 32
AArch32 Normal World and Secure World Registers ........ 32
AArch32 VA to PA Registers ........................................ 34
AArch32 Performance Registers .................................. 35
AArch64 Performance Registers .................................. 36
AArch32 VFP/Neon Registers ....................................... 37
AArch64 AdvSIMD Registers ........................................ 37
VGIC Physical CPU Interface Register ....................... 38
VGIC VCPU Hypervisor Register ............................... 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGIC VCPU Virtual Registers</td>
<td>39</td>
</tr>
<tr>
<td>GICv3 CPU Interface CPU/Virtual Registers (System Registers)</td>
<td>39</td>
</tr>
<tr>
<td>GICv3 CPU Interface Hypervisor registers (System Registers)</td>
<td>40</td>
</tr>
<tr>
<td>Run To Debug Point</td>
<td>41</td>
</tr>
<tr>
<td>Memory Information</td>
<td>41</td>
</tr>
<tr>
<td>Disassembly View</td>
<td>41</td>
</tr>
<tr>
<td>Available Profiling Data</td>
<td>42</td>
</tr>
<tr>
<td>Hardware Profiling</td>
<td>42</td>
</tr>
<tr>
<td>Software Profiling</td>
<td>49</td>
</tr>
</tbody>
</table>
Preface

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

About This Guide

This guide provides all the information needed to configure and use the Cortex-A72 multi-processor Cycle Model in SoC Designer Plus.

Audience

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

- SoC Designer Plus
- 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><strong>italic</strong></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 publication provides information that relates directly to SoC Designer Plus:

- *SoC Designer Plus User Guide*

The following publications provide reference information about ARM® products:

- *Cortex-A72 Technical Reference Manual*
- *ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile*
- *AMBA AXI and ACE Protocol Specification, Issue E*
- *Large Physical Address Extensions Specification* (ARM Architecture Group)

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 Carbon Model Studio (or Carbon compiler) from an RTL design. The Cycle Model contains a cycle- and register-accurate model of the hardware design.

**Carbon 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 Plus, 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. |
| **CHI** | The AMBA® 5 Coherent Hub Interface specification. A bus protocol with coherency channels designed to support high frequency, non-blocking data transfers between multiple coherent processors. |
| **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** | The full name is *SoC Designer Plus*. A 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 Component in SoC Designer Plus

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

- Cortex-A72 Functionality
- Adding and Configuring the SoC Designer Plus Component
- ESL Ports
- Setting Component Parameters
- Debug Features
- Available Profiling Data

1.1 Cortex-A72 Functionality

In the multiprocessor configuration, up to four Cortex-A72 processors are available in a cache-coherent cluster, under the control of a Snoop Control Unit (SCU), which maintains L1 and L2 data cache coherency.

Most hardware features have been implemented. See the ARM Cortex-A72 Technical Reference Manual for more information.

The Cortex-A72 processor supports:

- Up to four Cortex-A72 processors.
- AArch32 (32-bit ISA) and AArch64 mode.
• An SCU responsible for maintaining coherency among caches.
• Variable ICache/Dcache sizes.
• A Global Interrupt Controller (GIC) with support for legacy ARM interrupts.
• A generic 64-bit timer per processor.
• Support for AMBA 4.0 AXI Coherency Extension (ACE) master port for both 32- and 64-bit modes, and AMBA 5 CHI (Coherent Hub Interface) master port for both 32- and 64-bit modes.
• An optional Cryptography engine.
• Support for Virtualization Extensions for the development of virtualized systems that enable the switching of guest operating systems.
• ACP Transactors.
• VFP Floating Point.
• Neon Advanced SIMD.
• Semihosting.

Note: In systems that use multi-clock domains with semihosting, the clk-in pins of the Cortex-A72 and CarbonSemihost components must be connected to the same CDIV component.

• Large Physical Address (LPA) Extension.

1.1.1 Features Additional to the Hardware

The following features that are implemented in the Cortex-A72 Cycle Model do not exist in the Cortex-A72 hardware. These features have been added to the Cycle Model for enhanced usability.

• Support for positive- and negative-level irq, virq, fiq, and vfiq signals. This is configurable using the negLogic parameter (see Table 1-3 on page 16).
• The “run to debug point” feature has been added. This feature forces the debugger to advance the processor to the debug state instead of having the Cycle Model get into a non-debuggable state. See “Run To Debug Point” on page 41 for more information.
• Waveform dumping using the waveform-related parameters described in Table 1-3 on page 16.
• Support for viewing memory spaces (see “Memory Information” on page 41).
• Support for viewing disassembly data (see “Disassembly View” on page 41).
1.2 Adding and Configuring the SoC Designer Plus Component

The *SoC Designer Plus User Guide* describes how to use the component. See that guide for more information.

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

1.2.1 SoC Designer Plus Component Files

The component files are the final output from the Carbon Model Studio compile and are the input to SoC Designer Plus. 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 Plus. 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 in Table 1-1 below:

**Table 1-1 SoC Designer Plus Component Files**

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

For more information on SoC Designer Canvas, see the SoC Designer Plus User Guide.

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. Depending on your configuration, ports may differ slightly from those listed in Table 1-2 (see “Available Component ESL Ports” on page 14).

1.3 ESL Ports

This section describes the differences between the pins listed in the ARM Cortex-A72 Technical Reference Manual (TRM) and those on the Cortex-A72 Cycle Model. Certain hardware pins have been converted to init-time Cycle Model parameters.

- **Available Component ESL Ports** — Describes ports that have been added to the Cycle Model, such as clocks and resets required by SoC Designer Plus, or those created by wrapping multiple hardware pins into transactors.

- **Reset Behavior and Ports** — Describes the default reset behavior of the Cycle Model and how to generate a reset sequence during simulation.

- **Tied Pins** — Describes pins that are tied under certain conditions.

1.3.1 Available Component ESL Ports

Table 1-2 describes the Cortex-A72 ESL transactor and special pins that are exposed in SoC Designer Plus. See the ARM Cortex-A72 Technical Reference Manual for more information.

Table 1-2 ESL Component Ports

<table>
<thead>
<tr>
<th>ESL Port</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE_Master</td>
<td>ACE Master S2T when configured in ACE mode. IP configurable.</td>
<td>Transaction Master</td>
</tr>
<tr>
<td>ACP_Slave</td>
<td>Optional Accelerator Coherency Port implemented as an AXI4 slave interface.</td>
<td>Transaction Slave</td>
</tr>
<tr>
<td>CHI_RNF_CHI_Master</td>
<td>CHI Master S2T when configured with the CHI interface. IP configurable.</td>
<td>Transactor Master</td>
</tr>
<tr>
<td>Debug_APB</td>
<td>APB3 Transactor Slave.</td>
<td>Transaction Slave</td>
</tr>
<tr>
<td>CLK</td>
<td>Main clock of the Cortex-A72 MPCore multiprocessor. All processors, the shared L2 memory system logic, the GIC, and the Generic Timer are clocked with a distributed version of CLK.</td>
<td>Main Clock Transactor (Clock Slave)</td>
</tr>
<tr>
<td>clk-in</td>
<td>Leave this port unconnected except in systems that use multi-clock domains with semihosting. In this case, the clk-in pin of the Cortex-A72 and CarbonSemihost components must be connected to the same CDIV component.</td>
<td>Clock Slave</td>
</tr>
</tbody>
</table>
Note: Most ESL component port values can be set using a component parameter. In these cases, the parameter value is used whenever the ESL port is not connected. If the port is connected, the connection value takes precedence over the parameter value.

1.3.2 Reset Behavior and Ports

The Cycle Model is reset internally each time SoC Designer Simulator is initialized. This behavior is standard and can not be changed. To view the internal reset sequence, set the Align Waveforms parameter to False (see Setting Component Parameters), and this data appears in the waveform.

At simulation time zero and while simulation is running, you can generate a reset sequence. To do so, drive the reset pins on the component using external signals (for example, using the MxSigDriver component).

For information about reset pin names, bit ordering (for multiple cores), and required reset sequence, refer to the Technical Reference Manual for your IP.

1.3.3 Tied Pins

The following signals are tied to a certain value depending on the CoherencyType parameter setting:

- CPUQREQn (high)
- L2QREQn (high)
- DBGEN (high)
- NIDEN (high)
- SPIDEN (high)
- SPNIDEN (high)
- CIHSBYPASS (low)
- CISBYPASS (low)
- CTICHIN (low)
- CTICHOUTACK (low)
- CTIIRQACK (low)
- DFTCLKBYPASS (low)
- DFTCRCLKDISABLE (low)
- DFTL2CLKDISABLE (low)
- DFTMCPHOLD (low)
- DFTRAMHOLD (low)
- DFTRSTDISABLE (low)
- DFTSE (low)
- nMBISTRESET (high)
- MBISTREQ (low)
• when "id('BUS_INTERFACE') = 'CHI,"" SCLKEN is tied high
• when "id('BUS_INTERFACE') = 'ACE,"" WIDM is tied low

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 component’s 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.

   The list of available parameters will be slightly different depending on the settings that you enabled in the configuration.

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 component parameters are described in Table 1-3.

<table>
<thead>
<tr>
<th>Table 1-3 Component Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>AA64nAA32</td>
</tr>
<tr>
<td>AA64nAA320</td>
</tr>
<tr>
<td>ACE_Master Enable Debug Messages</td>
</tr>
<tr>
<td>ACE_Master Protocol Variant</td>
</tr>
<tr>
<td>ACINACTM</td>
</tr>
<tr>
<td>ACLKENM</td>
</tr>
</tbody>
</table>
### Table 1-3 Component Parameters (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Init/Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACLKENS</td>
<td>ACP AXI Slave Input Clock Enable</td>
<td>0, 1</td>
<td>1</td>
<td>Runtime</td>
</tr>
<tr>
<td>ACP_Slave axi_size [0-5]°</td>
<td>These parameters should be left at their default values.</td>
<td>—</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>ACP_Slave axi_start [0-5]°</td>
<td>These parameters should be left at their default values.</td>
<td>—</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>ACP_Slave Enable Debug Messages</td>
<td>Enables ACP_Slave port debug.</td>
<td>true, false</td>
<td>false</td>
<td>Runtime</td>
</tr>
<tr>
<td>ACP_Slave Protocol Variant°</td>
<td>Protocol variant of the corresponding port. This is set by configuration choice at build time and can not be changed in SoC Designer Plus. Protocol choice is reflected in the parameter and port name; i.e., ACE_Lite yields ACE_LITE_Sn_NIDm while ACE_Lite+DVM yields ACE_LITE_DVM_Sn_NIDm.</td>
<td>ACE-Lite or ACELite+DVM</td>
<td>ACE-Lite or ACELite+DVM</td>
<td>Init</td>
</tr>
<tr>
<td>AFVALIDMx</td>
<td>Fifo flush request. This signal is part of the ATB interface.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>Align Waveforms</td>
<td>When set to true, waveforms dumped by the component are aligned with the SoC Designer Plus 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 SoC Designer Plus time.</td>
<td>true, false</td>
<td>true</td>
<td>Init</td>
</tr>
<tr>
<td>ATCLKEN</td>
<td>ATB clock enable.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>ATREADYMx</td>
<td>ATB device ready.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</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>Init</td>
</tr>
<tr>
<td>CFGEND</td>
<td>Endianness configuration. 1-bit wide for UP, 4 bits wide for MP. Automatically kept in sync with CFGENDn.</td>
<td>integer</td>
<td>0</td>
<td>Init</td>
</tr>
<tr>
<td>CFGENDn</td>
<td>Endianness configuration. Per-core value of CFGEND; automatically kept in sync with CFGEND.</td>
<td>bool</td>
<td>false</td>
<td>Init</td>
</tr>
<tr>
<td>CFGTE</td>
<td>Default exception handling state (ARM/Thumb). 1-bit wide for UP, 4 bits wide for MP. Automatically kept in sync with CFGTEn.</td>
<td>integer</td>
<td>0</td>
<td>Init</td>
</tr>
</tbody>
</table>
Table 1-3 Component Parameters (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Init/Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFGTEn</td>
<td>Default exception handling state (ARM/Thumb). Per-core value of CFGTE; automatically kept in sync with CFGTE.</td>
<td>bool</td>
<td>false</td>
<td>Init</td>
</tr>
<tr>
<td>CHI_RNF_CHI_Master Enable Debug Message</td>
<td>Whether debug messages are enabled on the CHI Master port.</td>
<td>True, False</td>
<td>False</td>
<td>Runtime</td>
</tr>
<tr>
<td>CHI_RNF_CHI_Master Protocol Variant</td>
<td>Protocol Variant in use for CHI.</td>
<td>CHI-RNF</td>
<td>CHI-RNF</td>
<td>Init</td>
</tr>
<tr>
<td>CLKEN</td>
<td>Clock enable.</td>
<td>0, 1</td>
<td>1</td>
<td>Runtime</td>
</tr>
<tr>
<td>CLUSTERIDAFF1</td>
<td>Value read in the Cluster ID Affinity Level-1 field, bits[15:8], of the Multi-processor Affinity Register (MPIDR).</td>
<td>integer</td>
<td>0</td>
<td>Init</td>
</tr>
<tr>
<td>CLUSTERIDAFF2</td>
<td>Individual processor register width state.</td>
<td>integer</td>
<td>0</td>
<td>Init</td>
</tr>
<tr>
<td>CNTCLKEN</td>
<td>Counter clock enable. This clock enable must be inserted one cycle before the CNTVALUEB bus.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>CoherencyType</td>
<td>Drives the Coherency signals; supported for ACE and CHI configurations. Supported values: NonCoherentNoL3² NonCoherentWithL3² OuterCoherentNoL3 OuterCoherentWithL3 InnerCoherentNoL3 InnerCoherentWithL3</td>
<td>String</td>
<td>OuterCoherent WithL3</td>
<td>Init</td>
</tr>
<tr>
<td>CP15DISABLE</td>
<td>Disable write access to some Secure CP15 registers. This is the aggregated bus [NUM_CPUS-1:0].</td>
<td>bool</td>
<td>false</td>
<td>Runtime</td>
</tr>
<tr>
<td>CP15DISABLEn</td>
<td>One-bit parameter. Disable write access to DBGL1RSTDISABLE.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>CRYPTODISABLE</td>
<td>Individual core Cryptography engine disable. Only available with the Cryptography Extension enabled.</td>
<td>0, 1</td>
<td>0</td>
<td>Init</td>
</tr>
<tr>
<td>CRYPTODISABLEn</td>
<td>One-bit parameter. Disable Cryptography engine.</td>
<td>bool</td>
<td>false</td>
<td>Init</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Allowed Values</td>
<td>Default Value</td>
<td>Init/Runtime</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>DBGL1RSTDISABLE</td>
<td>Disable L1 data cache automatic invalidate on reset functionality:</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td></td>
<td>0 - Enable automatic invalidation of L1 data cache on reset.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 - Disable automatic invalidation of L1 data cache on reset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBGPWRDUP</td>
<td>Processor powered-up.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td></td>
<td>0 - Processor is powered down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 - Processor is powered up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBGROMADDR</td>
<td>External debug device CoreSight system configuration. Specifies bits [31:12] of the ROM.</td>
<td>Table Physical Address</td>
<td>0-fffffff</td>
<td>Init</td>
</tr>
<tr>
<td>DBGROMADDRV</td>
<td>Valid signal for DBGROMADDR.</td>
<td>bool</td>
<td>False</td>
<td>Init</td>
</tr>
<tr>
<td>Debug_APB Base Address</td>
<td>Start of the Debug_APB port address region.</td>
<td>Address</td>
<td>0x0</td>
<td>Init</td>
</tr>
<tr>
<td>Debug_APB Enable Debug Messages</td>
<td>Enable Debug_APB port debug.</td>
<td>true, false</td>
<td>false</td>
<td>Runtime</td>
</tr>
<tr>
<td>Debug_APB Size</td>
<td>Size of the Debug_APB port address region.</td>
<td>Size</td>
<td>0x100000000</td>
<td>Init</td>
</tr>
<tr>
<td>Dump Waveforms</td>
<td>Whether SoC Designer Plus dumps waveforms for this component.</td>
<td>bool</td>
<td>false</td>
<td>Runtime</td>
</tr>
<tr>
<td>Enable Debug Messages</td>
<td>Whether debug messages are logged for the component.</td>
<td>bool</td>
<td>false</td>
<td>Runtime</td>
</tr>
<tr>
<td>Fast Application Load Support</td>
<td>Identifies that the component supports fast debug access for application load.</td>
<td>Not configurable</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Enable Fast Application Load</td>
<td>Controls fast debug access for application load. “True” setting loads multiple bytes at a time; “False” setting loads 1 byte at a time.</td>
<td>true/false</td>
<td>true</td>
<td>Init</td>
</tr>
<tr>
<td>GICCDISABLE</td>
<td>Disables the GIC CPU interface logic and routes the legacy nIRQ, nFIQ, nVIRQ, nVFIQ. Required to enable use of non-ARM interrupt controllers.</td>
<td>bool</td>
<td>If IP is configured with GIC present then the default is false, else default is true.</td>
<td>Init</td>
</tr>
<tr>
<td>ICCTREADY</td>
<td>Input AXI4 Stream Protocol signal. GIC CPU Interface to Distributor messages. TREADY indicates that the slave can accept a transfer in the current cycle.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Allowed Values</td>
<td>Default Value</td>
<td>Init/Runtime</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>ICDTDATA</td>
<td>Input AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. TDATA is the primary payload that is used to provide the data that is passing across the interface.</td>
<td>[0-0xFFFF]</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>ICDTDEST</td>
<td>Input AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. TDEST provides routing information for the data stream.</td>
<td>[0-3]</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>ICDTLAST</td>
<td>When HIGH, indicates the boundary of a packet.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>ICDTVALID</td>
<td>Input AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. TVALID indicates that the master is driving a valid transfer.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>L2RSTDISABLE</td>
<td>Controls automatic hardware invalidation of the L2 cache during reset. A setting of:  1— Disables the hardware L2 invalidation reset sequence (this setting is required for Swap &amp; Play using L2 cache restore).  0 — Enables the hardware L2 invalidation reset sequence.</td>
<td>1— Disables the reset sequence. 0 — Enables the reset sequence.</td>
<td>1</td>
<td>Init</td>
</tr>
<tr>
<td>negLogic</td>
<td>Enables active low interrupts</td>
<td>bool</td>
<td>false</td>
<td>Runtime</td>
</tr>
<tr>
<td>NODE ID</td>
<td>Cortex-A72 CHI Node Identifier</td>
<td>Cortex-A72 CHI Node Identifier</td>
<td>0x7</td>
<td>Init</td>
</tr>
<tr>
<td>PCLKENDBG</td>
<td>APB DBG Clock Enable.</td>
<td>0, 1</td>
<td>1</td>
<td>Runtime</td>
</tr>
<tr>
<td>PERIPHBASE</td>
<td>Peripheral base [39:0] (Bits 14 - 0 are ignored)</td>
<td>integer</td>
<td>0x0013000000</td>
<td>Init</td>
</tr>
<tr>
<td>PMUSNAPSHOTREQ</td>
<td>PMU snapshot trigger acknowledge.</td>
<td>integer (per-core value)</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>RVBARADDRx</td>
<td>Reset Vector Base Address for executing in AArch64 state.</td>
<td>integer</td>
<td>0</td>
<td>Init</td>
</tr>
<tr>
<td>SAMADDRMAP [0 - 19]</td>
<td>CHI Region Mapping.</td>
<td>integer</td>
<td>0x0</td>
<td>Init</td>
</tr>
<tr>
<td>SAMHNF [0 - 7]NODEID</td>
<td>HN-F node ID</td>
<td>integer</td>
<td>0x0</td>
<td>Init</td>
</tr>
<tr>
<td>SAMHNFMODE</td>
<td>HN-F interleaving module</td>
<td>integer</td>
<td>0x0</td>
<td>Init</td>
</tr>
<tr>
<td>SAMJNI [0, 1]NODEID</td>
<td>HN-I Node ID</td>
<td>integer</td>
<td>0x0</td>
<td>Init</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Allowed Values</td>
<td>Default Value</td>
<td>Init/Runtime</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------</td>
<td>-------------------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>SAMMNBASE</td>
<td>MN base address</td>
<td>integer</td>
<td>0x90</td>
<td>Init</td>
</tr>
<tr>
<td>SAMMNNODEID</td>
<td>MN node ID</td>
<td>integer</td>
<td>0x0</td>
<td>Init</td>
</tr>
<tr>
<td>SYNCREQM0</td>
<td>ETM Signal. Synchronization request from trace sink.</td>
<td>0, 1</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>SCLKEN</td>
<td>CHI interface bus clock enable.</td>
<td>0, 1</td>
<td>1</td>
<td>Init</td>
</tr>
<tr>
<td>SINACT</td>
<td>CHI snoop active</td>
<td>0, 1</td>
<td>0</td>
<td>Init</td>
</tr>
<tr>
<td>SYNCREQM[0, 1, 2, 3]</td>
<td>Synchronization request from trace sink</td>
<td>0, 1</td>
<td>0</td>
<td>Init</td>
</tr>
<tr>
<td>SYSBARDISABLE</td>
<td>Disable broadcasting of barriers onto the system bus.</td>
<td>0, 1</td>
<td>0</td>
<td>Init</td>
</tr>
<tr>
<td>TSVALUEB</td>
<td>ETM signal. Timestamp in binary encoding.</td>
<td>[0-0xFFFFFFFFFFFFFFFFFF]</td>
<td>0</td>
<td>Runtime</td>
</tr>
<tr>
<td>VINITHI</td>
<td>Use high vector addresses. 1-bit wide for UP, 4 bits wide for MP. Automatically kept in sync with VINITHI.</td>
<td>integer</td>
<td>0</td>
<td>Init</td>
</tr>
<tr>
<td>VINITHIn</td>
<td>Use high vector addresses. Per-core value of VINITHI; automatically kept in sync with VINITHI.</td>
<td>bool</td>
<td>false</td>
<td>Init</td>
</tr>
<tr>
<td>Waveform File³</td>
<td>Name of the waveform file.</td>
<td>string</td>
<td>carbon_CORT EXA72.vcd or carbon_CORT EXA72MP.vcd</td>
<td>Init</td>
</tr>
<tr>
<td>Waveform Format</td>
<td>The format of the waveform dump file.</td>
<td>VCD, FSDB</td>
<td>VCD</td>
<td>Init</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>Init</td>
</tr>
</tbody>
</table>

1. Available based on IP configuration.
2. When configured with the ACE main bus interface, this setting uses the ACE-Lite protocol.
3. When enabled, SoC Designer Plus 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 Cortex-A72 Cycle Model has a debug interface (CADI) that allows the user to view, manipulate, and control the registers and memory. A view can be accessed in SoC Designer Plus by right clicking on the Cycle Model and choosing the appropriate menu entry.

The following topics are discussed in this section:

- Register Information
- Run To Debug Point
- Memory Information
- Disassembly View

1.5.1 Register Information

This section describes the registers supported with this release. Registers are grouped into sets according to functional area, as described in the following sections:

- AArch32 Core Registers
- AArch64 Core Registers
- AArch32 Control Registers
- AArch64 System Registers
- AArch32 Debug Registers
- External Debug Registers
- AArch64 Debug Registers
- AArch32 ID Registers
- AArch32 Normal World and Secure World Registers
- AArch32 VA to PA Registers
- AArch32 Performance Registers
- AArch64 Performance Registers
- AArch32 VFP/Neon Registers
- AArch64 AdvSIMD Registers
- VGIC Physical CPU Interface Register
- VGIC VCPU Hypervisor Register
- VGIC VCPU Virtual Registers
- GICv3 CPU Interface CPU/Virtual Registers (System Registers)
- GICv3 CPU Interface Hypervisor registers (System Registers)
For detailed descriptions of these registers, refer to the *ARM® Cortex™-A72 MPCore Technical Reference Manual* for the appropriate processor core.

**Note:** Registers are accurate only at debuggable points. While SoC Designer Plus grays out the register view when the processor is not at a debuggable point, values are still visible. Due to the speculative nature of the processor pipeline, these values are not guaranteed to be accurate.

In general, you can write to a register only at a debuggable point. If a value is deposited at any other point, it may not be correctly propagated.

### 1.5.1.1 AArch32 Core Registers

Table 1-4 describes the AArch32 Core register group.

**Table 1-4  AArch32 Core Registers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtendedTargetFeatures</td>
<td>Pseudo register that describes additional processor features</td>
<td>Read-Only</td>
</tr>
<tr>
<td>PC_MEMSPACE</td>
<td>Pseudo register that indicates the current memory space</td>
<td>Read-Only</td>
</tr>
<tr>
<td>R0-R14</td>
<td>General purpose registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>R15</td>
<td>PC. Write at debug point only.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CPSR32</td>
<td>Current Program Status Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SPSR32</td>
<td>Current program status register in usr mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SPSR_irq</td>
<td>Saved program status register in IRQ mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SPSR_fiq</td>
<td>Saved program status register in FIQ mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SPSR_svc</td>
<td>Saved program status register in SVC mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SPSR_abt</td>
<td>Saved program status register in ABT mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SPSR_und</td>
<td>Saved program status register in UND mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SPSR_hyp</td>
<td>Saved program status register in HYP mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SPSR_mon</td>
<td>Saved program status register in MON mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>R13_svc, R14_svc</td>
<td>R13/R14 in SVC mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>R13_irq, R14_irq</td>
<td>R13/R14 in IRQ mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>R8_fiq, R14_fiq</td>
<td>R8-R14 in FIQ mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>R8_usr, R14_usr</td>
<td>R8-R14 in USR mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>R13_und, R14_und</td>
<td>R13/R14 in UND mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>R13_abt, R14_abt</td>
<td>R13/R14 in ABT mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>R13_mon, R14_mon</td>
<td>R13/R14 in MON mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>R13_hyp</td>
<td>R13 in HYP mode</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ELR_hyp</td>
<td>Exception Link Register in HYP mode</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
1.5.1.2 AArch64 Core Registers

Table 1-5 describes the AArch64 Core registers.

**Table 1-5 AArch64 Core Registers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0-X30</td>
<td>ID registers.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PC</td>
<td>Program Counter register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SP</td>
<td>Stack Pointer register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ELR</td>
<td>Exception Link register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CPSR</td>
<td>Current Program State register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SPSR</td>
<td>Saved Program Status Register for current mode</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>

1.5.1.3 AArch32 Control Registers

Table 1-6 describes the AArch32 Control register group.

**Table 1-6 AArch32 Control Registers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCTLR</td>
<td>System Control Register.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ACTLR</td>
<td>Auxiliary Control Register.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CPACR</td>
<td>Coprocessor Access Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>NSACR</td>
<td>Nonsecure Access Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TTBR0</td>
<td>Translation Table Base Register 0.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TTBR1</td>
<td>Translation Table Base Register 1.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TTBCR</td>
<td>Translation Table Base Control Register.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DACR</td>
<td>Domain Access Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DFSR</td>
<td>Data Fault Status</td>
<td>Read-Write</td>
</tr>
<tr>
<td>IFSR</td>
<td>Instruction Fault Status</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ADFSR</td>
<td>Auxiliary Data Fault Status</td>
<td>Read-Write</td>
</tr>
<tr>
<td>AIFSR</td>
<td>Auxiliary Instruction Fault Status</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DFAR</td>
<td>Data Fault Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>IFAR</td>
<td>Instruction Fault Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>L2CTLR</td>
<td>L2 Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>L2ECTLR</td>
<td>L2 Extended Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PRRR</td>
<td>Primary region remap</td>
<td>Read-Write</td>
</tr>
<tr>
<td>NMRR</td>
<td>Normal memory remap</td>
<td>Read-Write</td>
</tr>
<tr>
<td>MAIR0</td>
<td>Memory Attribute Indirection Register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>MAIR1</td>
<td>Memory Attribute Indirection Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>AMAIR0</td>
<td>Auxiliary Memory Attribute Indirection Register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>AMAIR1</td>
<td>Auxiliary Memory Attribute Indirection Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>VBAR</td>
<td>Vector Base Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ISR</td>
<td>Interrupt Status</td>
<td>Read-Only</td>
</tr>
<tr>
<td>FCSEIDR</td>
<td>FCSE Process ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CONTEXTIDR</td>
<td>Context ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TPIDRURW</td>
<td>User Read-Write Thread ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TPIDRURO</td>
<td>User Read-Only Thread ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TPIDRPRW</td>
<td>Privileged Only Thread ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTFRQ</td>
<td>Timer Clk Ticks Per Sec</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTKCTL</td>
<td>Timer Kernel Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTP_TVAL</td>
<td>Timer Phy Timer Val</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTP_CTL</td>
<td>Timer Phy Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTV_TVAL</td>
<td>Timer Virt Timer Val</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTV_CTL</td>
<td>Timer Virt Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTHCTL</td>
<td>Timer Hyp Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTHP_TVAL</td>
<td>Timer Hyp_and_S_Priv Timer Val</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTHP_CTL</td>
<td>Timer Hyp_and_S_Priv Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CBAR</td>
<td>Configuration Base Address</td>
<td>Read-Only</td>
</tr>
<tr>
<td>VPIDR</td>
<td>Virtualization Processor ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>VMPIDR</td>
<td>Virtualization Multiprocessor ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HSCTLR</td>
<td>Hypervisor System Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HACTLR</td>
<td>Hypervisor Auxiliary Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HCR</td>
<td>Hypervisor Configuration</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HCR2</td>
<td>Hypervisor Configuration 2</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HDCR</td>
<td>Hypervisor Debug Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HCPTR</td>
<td>Hypervisor Copro Trap</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HSTR</td>
<td>Hypervisor System Trap</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HTCR</td>
<td>NSHyp Translation Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>VTCR</td>
<td>Virt Translation Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HADFSR</td>
<td>Hyp Aux Data Fault Status</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HAIFSR</td>
<td>Hyp Aux Inst Fault Status</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HSR</td>
<td>Hyp Undef Except Syndrome</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HDFAR</td>
<td>Hyp Data Fault Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HIFAR</td>
<td>Hyp Inst Fault Address</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
Table 1-6  AArch32 Control Registers  (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPFAR</td>
<td>Hyp IPA Fault Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HMAIR0</td>
<td>Hyp Memory Attribute Indirection 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HMAIR1</td>
<td>Hyp Memory Attribute Indirection 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HAMAIR0</td>
<td>Hyp Auxiliary Memory Attribute Indirection 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HAMAIR1</td>
<td>Hyp Auxiliary Memory Attribute Indirection 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HVBAR</td>
<td>Hyp Vector Base Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HTPIDR</td>
<td>Hyp Thread Local Storage</td>
<td>Read-Write</td>
</tr>
<tr>
<td>RMR</td>
<td>Reset Management</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTPCT</td>
<td>Timer Physical Count</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTVCT</td>
<td>Timer Virt Count</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTP_CVAL</td>
<td>Timer Phy Compare Val</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTV_CVAL</td>
<td>Timer Virt Compare Val</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTVOFF</td>
<td>Timer Virt Offset</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CNTHP_CVAL</td>
<td>Timer Hyp and _S_Priv Compare Val</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CPUECTLR</td>
<td>CPU Extended Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HTTBR</td>
<td>NSHyp Translation Table Base</td>
<td>Read-Write</td>
</tr>
<tr>
<td>VTTBR</td>
<td>Virt Translation Base</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TTBR0_64</td>
<td>Translation Table Base Register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TTBR1_64</td>
<td>Translation Table Base Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PAR_64</td>
<td>Physical Address Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>IL1DATA0 — IL1DATA2</td>
<td>Instruction L1 Data0 Register - Instruction L1 Data2 Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DL1DATA0 — DL1DATA3</td>
<td>Data L1 Data1 Register - Data L1 Data3 Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CSSELR</td>
<td>Cache Size Select Register</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
### 1.5.1.4 AArch64 System Registers

Table 1-7 describes the AArch64 System registers.

**Table 1-7 AArch64 System Registers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDR_EL1</td>
<td>Main ID</td>
<td>Read-Only</td>
</tr>
<tr>
<td>CTR_EL0</td>
<td>Cache Type</td>
<td>Read-Only</td>
</tr>
<tr>
<td>MIPIDR_EL1</td>
<td>Multiprocessor Affinity</td>
<td>Read-Only</td>
</tr>
<tr>
<td>REVIDR_EL1</td>
<td>Revision ID</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_PFR0_EL1</td>
<td>Processor Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_PFR1_EL1</td>
<td>Processor Features 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_DFR0_EL1</td>
<td>Debug Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AFR0_EL1</td>
<td>Auxiliary Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_MMFR0_EL1 — ID_MMFR3_EL1</td>
<td>Memory Model Features 0 - 3</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_ISAR0_EL1</td>
<td>Instruction Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_ISAR1_EL1 — ID_ISAR5_EL1</td>
<td>Instruction Features 1 - 5</td>
<td>Read-Only</td>
</tr>
<tr>
<td>MVFR0_EL1 — MVFR2_EL1</td>
<td>Media and VFP Feature 0 - 2</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64PFR0_EL1</td>
<td>AARCH64 Processor Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64PFR1_EL1</td>
<td>AARCH64 Processor Features 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64DFR0_EL1</td>
<td>AARCH64 Debug Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64DFR1_EL1</td>
<td>AARCH64 Debug Features 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64AFR0_EL1</td>
<td>AARCH64 Auxiliary Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64AFR1_EL1</td>
<td>AARCH64 Auxiliary Features 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64ISR0_EL1</td>
<td>AARCH64 Instruction Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64ISR1_EL1</td>
<td>AARCH64 Instruction Features 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64MMFR0_EL1</td>
<td>AARCH64 Memory Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AA64MMFR1_EL1</td>
<td>AARCH64 Memory Features 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>SCTLR_EL1</td>
<td>Control EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DCZID_EL0</td>
<td>Data Cache Zero ID</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ACTLR_EL1</td>
<td>Auxiliary Control EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CPACR_EL1</td>
<td>Coproc Access Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TTBR0_EL1</td>
<td>Translation Table Base Register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TTBR1_EL1</td>
<td>Translation Table Base Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TCR_EL1</td>
<td>Translation Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>AFSR0_EL1</td>
<td>Auxiliary Fault Status Register 0</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
Table 1-7  AArch64 System Registers  (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFSR1_EL1</td>
<td>Auxiliary Fault Status Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ESR_EL1</td>
<td>Exception Syndrome Register, EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>FAR_EL1</td>
<td>Fault Address Register, EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PAR_EL1</td>
<td>Physical Address Register, EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>MAIR_EL1</td>
<td>Memory Attribute Indirection Register, EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>VBAR_EL1</td>
<td>Vector Base Address Register, EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>AMAIR_EL1</td>
<td>Auxiliary Memory Attribute Indirection Regis-</td>
<td>Read-Write</td>
</tr>
<tr>
<td></td>
<td>ter, EL1</td>
<td></td>
</tr>
<tr>
<td>CPUECTLR_EL1</td>
<td>CPU Extended Control Register, EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CBAR_EL1</td>
<td>Configuration Base Address Register, EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ISR_EL1</td>
<td>Interrupt Status Register, EL1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>CONTEXTIDR_EL1</td>
<td>Context ID Register, EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TPIDR_EL0</td>
<td>Thread Pointer/ID Register, EL0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TPIDRRO_EL0</td>
<td>Thread Pointer/ID Register, Read-Only, EL0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TPIDR_EL1</td>
<td>Thread Pointer/ID Register, EL1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>VPIDR_EL2</td>
<td>Virtualization Processor ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>VMPIDR_EL2</td>
<td>Virtualization Multiprocessor ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SCTLR_EL2</td>
<td>Control EL2</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ACTLR_EL2</td>
<td>Auxiliary Control EL2</td>
<td>Read-Write</td>
</tr>
<tr>
<td>CPTR_EL2</td>
<td>Hyp CoPro Trap</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HCR_EL2</td>
<td>Hyp Configuration</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HSTR_EL2</td>
<td>Hyp System Trap</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HACR_EL2</td>
<td>Hyp Auxiliary Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TTBR0_EL2</td>
<td>Translation Table Base Address Register 0, EL2</td>
<td>Read-Write</td>
</tr>
<tr>
<td>VTTBR_EL2</td>
<td>Virtualization Translation Table Base Address Register, EL2</td>
<td>Read-Write</td>
</tr>
<tr>
<td>TCR_EL2</td>
<td>Translation Control Register, EL2</td>
<td>Read-Write</td>
</tr>
<tr>
<td>VTCR_EL2</td>
<td>Virtualization Translation Control Register, EL2</td>
<td>Read-Write</td>
</tr>
<tr>
<td>AFSR0_EL2</td>
<td>Auxiliary Fault Status Register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>AFSR1_EL2</td>
<td>Auxiliary Fault Status Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ESR_EL2</td>
<td>Exception Syndrome Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>FAR_EL2</td>
<td>Fault Address Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>HPFAR_EL2</td>
<td>Hypervisor IPA Fault Address Register</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
Table 1-8  AArch32 Debug Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBGDSCRint</td>
<td>Debug Status and Control Register (Internal)</td>
<td>Read-Only</td>
</tr>
<tr>
<td>DSPSR</td>
<td>Debug Saved Processor Status Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>DBGDIDR</td>
<td>Debug ID Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>DCCINT</td>
<td>Debug Comms Channel Interrupt Enable Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGWFAR</td>
<td>Watchpoint Fault Address Register, RES0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>EDECR</td>
<td>External Debug Execution Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGDTRRXext</td>
<td>Debug Data Transfer Register, Receive, External View</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGDTRXText</td>
<td>Debug Data Transfer Register, Transmit, External View</td>
<td>Read-Write</td>
</tr>
<tr>
<td>EDECCR</td>
<td>External Debug Exception Catch Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGBV0 - DBGBV5</td>
<td>Debug Breakpoint Value Register 0 - Debug Breakpoint Value Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGBCR0 - DBGBCR5</td>
<td>Debug Breakpoint Control Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGWVR0 - DBGWVR3</td>
<td>Debug Watchpoint Value Register 0 - Debug Watchpoint Value Register 3</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGWCR0 — DBGWCR3</td>
<td>Debug Watchpoint Control Registers</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
### 1.5.1.6 External Debug Registers

Table 1-9 describes the External Debug registers group.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBGDRAR</td>
<td>Debug ROM Address Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGBXVR4</td>
<td>Breakpoint Extended Value 4</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGBXVR5</td>
<td>Breakpoint Extended Value 5</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGWXVR0</td>
<td>Watchpoint eXtended Value 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGWXVR1</td>
<td>Watchpoint eXtended Value 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGWXVR2</td>
<td>Watchpoint eXtended Value 2</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGWXVR3</td>
<td>Watchpoint eXtended Value 3</td>
<td>Read-Write</td>
</tr>
<tr>
<td>OSLSR_EL1</td>
<td>OS Lock Status</td>
<td>Read-Only</td>
</tr>
<tr>
<td>DBGOSDLR</td>
<td>OS Double Lock</td>
<td>Read-Write</td>
</tr>
<tr>
<td>EDPRCR</td>
<td>Device Powerdown and Reset Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>EDPRSR</td>
<td>Device Powerdown and Reset Status</td>
<td>Read-Only</td>
</tr>
<tr>
<td>DBGDSAR</td>
<td>Debug Self Address Offset</td>
<td>Read-Only</td>
</tr>
<tr>
<td>DBGAUTHSTATUS</td>
<td>Authentication Status</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDDEVID0</td>
<td>Debug Device ID register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>EDDEVID1</td>
<td>Debug Device ID register 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDDEVTYPE</td>
<td>Debug Device Type</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EPID0 - EPID4</td>
<td>Peripheral ID Register 0-4</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ECID0 - ECID3</td>
<td>Component ID Register 0-3</td>
<td>Read-Write</td>
</tr>
<tr>
<td>EDESR</td>
<td>External Debug Event Status Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>EDWARLO</td>
<td>External Debug Watchpoint Address Register lo</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDWARHI</td>
<td>External Debug Watchpoint Address Register hi</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDACR</td>
<td>External Debug Auxiliary Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>EDPCSRLO</td>
<td>External Debug Program Counter Sample Register, low word</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDPCSRHI</td>
<td>External Debug Program Counter Sample Register, high word</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDCIDSR</td>
<td>External Debug Context ID Sample Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDVDISR</td>
<td>External Debug Virtual Context Sample Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>DBGBXVR0 - DBGBXVR3</td>
<td>Breakpoint Extended Value Registers</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
### Table 1-9 External Debug Registers (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDDEVARCH</td>
<td>External Debug Device Architecture Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDDEVAFF0</td>
<td>Multiprocessor Affinity Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDDEVAFF1</td>
<td>External Debug Device Affinity Register 1, RES0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>EDLSR</td>
<td>External Debug Lock Status Register</td>
<td>Read-Only</td>
</tr>
</tbody>
</table>

### 1.5.1.7 AArch64 Debug Registers

Table 1-10 describes the AArch64 Debug registers group.

### Table 1-10 AArch64 Debug Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDCR_EL3</td>
<td>Monitor Debug Configuration Register, EL3</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SDER32_EL3</td>
<td>Secure Debug Enable Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>MDCR_EL2</td>
<td>Hyp Debug Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGDTRRX_EL0</td>
<td>Debug Data Transfer Register, Receive</td>
<td>Read-Write</td>
</tr>
<tr>
<td>MDRAR_EL1</td>
<td>Monitor Debug ROM Address Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>MDCCSR_EL0</td>
<td>Monitor DCC Status Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>MDCCINT_EL1</td>
<td>Monitor DCC Interrupt Enable Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGBVR0_EL1 —</td>
<td>Debug Breakpoint Value Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGBVR5_EL1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBGBCR0_EL1 —</td>
<td>Debug Breakpoint Control Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGBCR3_EL1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBGWVR0_EL1 —</td>
<td>Debug Watchpoint Value Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGWVR3_EL1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBGWCR0_EL1 —</td>
<td>Debug Watchpoint Control Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>DBGWCR3_EL1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1.5.1.8 AArch32 ID Registers

Table 1-11 describes the AArch32 ID registers group.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDR</td>
<td>Main ID</td>
<td>Read-Only</td>
</tr>
<tr>
<td>CTR</td>
<td>Cache Type</td>
<td>Read-Only</td>
</tr>
<tr>
<td>TCMTR</td>
<td>TCM_TYPE</td>
<td>Read-Only</td>
</tr>
<tr>
<td>TLBTR</td>
<td>TLB Type</td>
<td>Read-Only</td>
</tr>
<tr>
<td>MPIDR</td>
<td>Multiprocessor Affinity</td>
<td>Read-Only</td>
</tr>
<tr>
<td>REVIDR</td>
<td>Revision ID</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_PFR0</td>
<td>Processor Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_PFR1</td>
<td>Processor Features 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_DFR0</td>
<td>Debug Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_AFR0</td>
<td>Auxiliary Features 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_MMFR0 —</td>
<td>Memory Model Features 0 - 3</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_MMFR3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID_ISAR0 —</td>
<td>Instruction Features 0 - 5</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ID_ISAR5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCSIDR</td>
<td>Cache Size ID</td>
<td>Read-Only</td>
</tr>
<tr>
<td>CLIDR</td>
<td>Cache Level ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>AIDR</td>
<td>Auxiliary ID</td>
<td>Read-Only</td>
</tr>
<tr>
<td>JIDR</td>
<td>Jazelle ID Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>JOSCR</td>
<td>Jazelle OS Control Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>JMCR</td>
<td>Jazelle Main Configuration Register</td>
<td>Read-Only</td>
</tr>
</tbody>
</table>

1.5.1.9 AArch32 Normal World and Secure World Registers

Table 1-12 describes the AArch32 Normal World and Secure World registers. The Normal World group contains register that are only accessible in non-secure mode. The Secure group contains registers that are only accessible in secure mode.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_CSSELV, S_CSSELV</td>
<td>Cache Size Selection</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_SCTLV, S_SCTLV</td>
<td>Secure and nonsecure views of System Control Register.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_ACTLV, S_ACTLV</td>
<td>Secure and nonsecure views of Auxiliary Control Register.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SCR</td>
<td>Secure Configuration</td>
<td>Read-Write</td>
</tr>
<tr>
<td>SDER</td>
<td>Secure Debug Enable</td>
<td>Read-Write</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>SDCR</td>
<td>Secure Debug Configuration</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_TTBR0, S_TTBR0</td>
<td>Secure and nonsecure views of Translation Table Base Address Register 0.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_TTBR1, S_TTBR1</td>
<td>Secure and nonsecure views of Translation Table Base Address Register 1.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_TTBCR, S_TTBCR</td>
<td>Secure and nonsecure views of Translation Table Base Control Register.</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_DACR, S_DACR</td>
<td>Secure and nonsecure DACR</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_DFSR , S_DFSR</td>
<td>Secure and nonsecure DFSR</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_IFSR, S_IFSR</td>
<td>Secure And Nonsecure IFSR</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_ADFS R , S_ADFS R</td>
<td>Secure And Nonsecure ADFS R</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_AIFSR, S_AIFSR</td>
<td>Secure And Nonsecure AIFSR</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_DFAR, S_DFAR</td>
<td>Secure And Nonsecure Data Fault Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_IFAR, S_IFAR</td>
<td>Secure And Nonsecure Instruction Fault Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_PAR, S_PAR</td>
<td>Secure And Nonsecure Physical Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_PRRR, S_PRRR</td>
<td>Secure And Nonsecure Primary region remap</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_NMRR, S_NMRR</td>
<td>Secure And Nonsecure Normal memory remap</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_MAIR0, S_MAIR0</td>
<td>Secure And Nonsecure Memory Attribute Indirection Register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_MAIR1, S_MAIR1</td>
<td>Secure And Nonsecure Memory Attribute Indirection Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_AMAIR0, S_AMAIR0</td>
<td>Secure And Nonsecure Auxiliary Memory Attribute Indirection Register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_AMAIR1, S_AMAIR1</td>
<td>Secure And Nonsecure Auxiliary Memory Attribute Indirection Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_VBAR, S_VBAR</td>
<td>Secure And Nonsecure Vector Base Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>MVBAR</td>
<td>Monitor Vector Base Address</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_FCSEIDR, S_FCSEIDR</td>
<td>Secure And Nonsecure FCSE Process ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_CONTEXTIDR, S_CONTEXTIDR</td>
<td>Secure And Nonsecure Context ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_TPIDRURW, S_TPIDRURW</td>
<td>Secure And Nonsecure User Read-Write Thread ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_TPIDRUDO, S_TPIDRUDO</td>
<td>User Read-Only Thread ID</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_TPIDRP RW, S_TPIDRP RW</td>
<td>Secure and nonsecure Privileged Only Thread ID</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
### Table 1-13  AArch32 VA to PA Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_CNTP_TVAL, S_CNTP_TVAL</td>
<td>Secure And Nonsecure Timer Phy Timer Val</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_CNTP_CTL, S_CNTP_CTL</td>
<td>Secure And Nonsecure Timer Phy Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_TTBR0_64, S_TTBR0_64</td>
<td>Secure And Nonsecure Translation Table Base Register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_TTBR1_64, S_TTBR1_64</td>
<td>Secure And Nonsecure Translation Table Base Register 1</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_PAR_64, S_PAR_64</td>
<td>Secure And Nonsecure Physical Address Reg-</td>
<td>Read-Write</td>
</tr>
<tr>
<td>N_CNTP_CVAL, S_CNTP_CVAL</td>
<td>Secure And Nonsecure Timer Phy Compare Val</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>

### 1.5.1.10  AArch32 VA to PA Registers

Table 1-13 describes the AArch32 VA to PA registers group.

### Table 1-13  AArch32 VA to PA Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAR</td>
<td>Physical address</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
1.5.1.11 AArch32 Performance Registers

Table 1-14 describes the AArch32 Performance Registers group.

**Note:** The registers PMXEVCNTRn and PMXEVTYPEERn allow direct access to the event count and event type registers without requiring that the specific register number be programmed into PMSELR. For example, if PMSELR contains the value 2 then PMXEVCNTR2 and PMXEVCNTR will display the contents of the same register.

**Table 1-14 AArch32 Perf Registers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMCR</td>
<td>Performance Monitor Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMCNTENSET</td>
<td>Count Enable Set</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMCNTENCLR</td>
<td>Count Enable Clear</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMOVSR</td>
<td>Overflow Flag Status Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMSELR</td>
<td>Event Counter Selection Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMCEID0</td>
<td>Performance Monitor Common Event Identification Register 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>PMCEID1</td>
<td>Performance Monitor Common Event Identification Register 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>PMCCNTR</td>
<td>Cycle count</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMXEVTYPEER</td>
<td>Event Type Select</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMXEV CNTR</td>
<td>Event Count</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMUSERENR</td>
<td>User Enable Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMINTENSET</td>
<td>Interrupt Enable Set</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMINTENCLR</td>
<td>Interrupt Enable Clear</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMOVSSSET</td>
<td>Overflow Flag Status Set</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
### 1.5.1.12 AArch64 Performance Registers

Table 1-15 describes the AArch64 Performance registers group.

#### Table 1-15 AArch64 Performance Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMCR_EL0</td>
<td>Performance Monitors Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMCNTENSET_EL0</td>
<td>Performance Monitors Count Enable Set Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMCNTENCLR_EL0</td>
<td>Performance Monitors Count Enable Clear Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PSEL_EL0</td>
<td>Performance Monitors Event Counter Selection Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMCEID0_EL0</td>
<td>Performance Monitors Common Event Identification Register 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>PMCEID1_EL0</td>
<td>Performance Monitors Common Event Identification Register 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>PMCCNTR_EL0</td>
<td>Performance Monitors Cycle Count Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PXEVTYPE0</td>
<td>Performance Monitors Selected Event Type Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PXEVCNTR_EL0</td>
<td>Performance Monitors Selected Event Count Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMUSERENR_EL0</td>
<td>Performance Monitors User Enable Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMINTNSET_EL1</td>
<td>Performance Monitors Interrupt Enable Set Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMINTENCLR_EL1</td>
<td>Performance Monitors Interrupt Enable Clear Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>PMOVSET_EL0</td>
<td>Performance Monitors Overflow Flag Status Set Register</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
1.5.1.13 AArch32 VFP/Neon Registers
Table 1-16 describes the AArch32 VFP/Neon registers group.

**Table 1-16 AArch32 VFP/Neon Registers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0 — S31</td>
<td>Advanced SIMD and VFP Extension Single Word Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>D0 — D31</td>
<td>Advanced SIMD and VFP Extension Double Word Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>Q0-Q15</td>
<td>Advanced SIMD and VFP Extension Quad Word Registers Neon only</td>
<td>Read-Write</td>
</tr>
<tr>
<td>FPSID</td>
<td>Floating Point System ID</td>
<td>Read-Only</td>
</tr>
<tr>
<td>FPSCR</td>
<td>Floating-Point Status and Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>FPEXC</td>
<td>Floating-point Exception Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>MVFR0 — MVFR1</td>
<td>Media and VFP Feature Register 0 - Media and VFP Feature Register 1</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>

1.5.1.14 AArch64 AdvSIMD Registers
Table 1-17 describes the AArch64 AdvSIMD registers group.

**Table 1-17 AArch64 AdvSIMD Registers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>S64_0 — S64_31</td>
<td>Advanced SIMD and VFP Extension Single Word Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>D64_0 — D64_31</td>
<td>Advanced SIMD and VFP Extension Double Word Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>V0 — V31</td>
<td>SIMD and floating-point registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>FPCR</td>
<td>Floating-Point Control</td>
<td>Read-Write</td>
</tr>
<tr>
<td>FPSR</td>
<td>Floating-Point Status</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
### 1.5.1.15 VGIC Physical CPU Interface Register

This group contains VGIC Physical CPU interface registers.

**Table 1-18 VGIC Physical CPU Interface Registers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GICC_CTLR_S</td>
<td>CPU Interface Control Register (secure)</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICC_CTLR_N</td>
<td>CPU Interface Control Register (non-secure)</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICC_PMR</td>
<td>Interrupt Priority Mask Registers</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICC_BPR_S</td>
<td>Binary Point Register (secure)</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICC_BPR_N</td>
<td>Binary Point Register (non-secure)</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICC_RPR</td>
<td>Running Priority Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICC_HPPIR</td>
<td>Highest Pending Interrupt Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICC_APR0</td>
<td>Active Priority Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICC_NSAPR0</td>
<td>Non Secure Active Priority Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICC_IIDR</td>
<td>CPU Interface Identification Register</td>
<td>Read-Only</td>
</tr>
</tbody>
</table>

### 1.5.1.16 VGIC VCPU Hypervisor Register

This group contains the VGIC VCPU Hypervisor registers.

**Table 1-19 VGIC VCPU Hypervisor Registers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GICH_HCR</td>
<td>Hypervisor Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICH_VTR</td>
<td>VGIC Type Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICH_VMCR</td>
<td>Virtual Machine Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICH_MISR</td>
<td>Maintenance Interface Status Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICH_EISR</td>
<td>End of Interrupt Status Registers (0-1)</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICH_ELRSR</td>
<td>Empty List Register Status Registers (0-1)</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICH_APR0</td>
<td>Active Priorities Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICH_LR</td>
<td>List Registers (0-4)</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
1.5.1.17 VGIC VCPU Virtual Registers

This group contains VGIC VCPU Virtual Machine registers.

Table 1-20 VGIC VCPU Virtual Machine Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GICV_CTLR</td>
<td>VM Control Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICV_PMR</td>
<td>VM Priority Mask Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICV_BPR</td>
<td>VM Binary Point Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICV_IAR</td>
<td>VM Interrupt Acknowledge Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICV_RPR</td>
<td>VM Running Priority Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICV_HPPIR</td>
<td>VM Highest Priority Pending Interrupt Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICV_ABPR</td>
<td>VM Aliased Binary Point Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICV_AIAR</td>
<td>VM Aliased Interrupt Acknowledge Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICV_AHPPIR</td>
<td>VM Aliased Highest Priority Pending Interrupt Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICV_APR0</td>
<td>VM Active Priority Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>GICV_NSAPR0</td>
<td>Non Secure Active Priority Register 0</td>
<td>Read-Write</td>
</tr>
<tr>
<td>GICV_IIDR</td>
<td>VM Cpu Interface Identification Register</td>
<td>Read-Only</td>
</tr>
</tbody>
</table>

1. See GICC_APR0 or GICH_APR0 to write this register.

1.5.1.18 GICv3 CPU Interface CPU/Virtual Registers (System Registers)

This group contains GICv3 CPU Interface CPU/Virtual registers.

Table 1-21 GICv3 CPU Interface CPU/Virtual registers (System Registers)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICC_AP0R0_EL1</td>
<td>Active Priority Group0 Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ICC_PMR_EL1</td>
<td>Priority Mask Register</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ICC_BPR0_EL1</td>
<td>Group0 Binary Pointer Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICC_HPPIR0_EL1</td>
<td>Group0 Highest Priority Pending Interrupt Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICC_HPPIR1_EL1</td>
<td>Group1 Highest Priority Pending Interrupt Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICC_IAR0_EL1</td>
<td>Group0 Interrupt Acknowledge Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICC_IAR1_EL1</td>
<td>Group1 Interrupt Acknowledge Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICC_IGRPEN0_EL1</td>
<td>Group0 Interrupt Group Enable Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICC_IGRPEN1_EL1</td>
<td>Group1 Interrupt Group Enable</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICC_IGRPEN1_EL3</td>
<td>EL3 Group1 Interrupt Group Enable</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICC_RPR_EL1</td>
<td>Running Priority Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICC_SRE_EL3</td>
<td>EL3 System Register Enable</td>
<td>Read-Write</td>
</tr>
<tr>
<td>ICC_CTLR_EL3</td>
<td>EL3 Control Register</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
### 1.5.1.19 GICv3 CPU Interface Hypervisor registers (System Registers)

This group contains the GICv3 CPU Interface Hypervisor registers.

**Table 1-22  GICv3 CPU Interface Hypervisor registers (System Registers)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICH_APR0_EL2</td>
<td>Hypervisor Active Priority Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_EISR_EL2</td>
<td>End of Interrupt Status Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_ELSR_EL2</td>
<td>Empty List Register Status Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_HCR_EL2</td>
<td>Hypervisor Control Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_LR0_EL2</td>
<td>List Register 0</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_LR1_EL2</td>
<td>List Register 1</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_LR2_EL2</td>
<td>List Register 2</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_LR3_EL2</td>
<td>List Register 3</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_MISR_EL2</td>
<td>Maintenance Interrupt Status Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_VTR_EL2</td>
<td>VGIC Type Register</td>
<td>Read-Only</td>
</tr>
<tr>
<td>ICH_VMCR_EL2</td>
<td>Virtual Machine Control Register</td>
<td>Read-Write</td>
</tr>
</tbody>
</table>
1.5.2 Run To Debug Point

The “run to debug point” feature has been added to enhance Cycle Model debugging. The Cortex-A72 processor is a dual issue out of order completion machine. This means that while the processor is running it does not present a coherent programmer’s view state; instructions in the pipeline may be in different execution states.

This feature forces the processor into a coherent state called “run to debug point”. When debugging, the Cycle 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 the system is advanced by cycles within SoC Designer Plus, the Cycle Model can get to a non-debuggable state. In this event, the *run to debug point* will advance the processor to the debug state. It does this by stalling the instruction within the decode stage and allowing all earlier instructions to complete. Once that has been accomplished, the Cycle Model will cause the system to stop simulating.

The run to debug point is available as a context menu item (*Run to Debuggable Point*) for the component within SoC Designer Simulator. It is also available in the disassembler view.

1.5.3 Memory Information

Each memory space represents a different view of memory using a page table. The Cortex-A72 processor memory spaces are selectable using the Space: pulldown menu in the Memory view, and the Memory space pulldown menu in the Disassembly view (see the *SoC Designer Plus User Guide* for more information).

From the top-level component view, one memory space is visible:

- **axi_m** — Main memory space visible from the Memory view.

At the subcomponent level, (e.g., CortexA72[0].cpu0), the following memory spaces are supported:

- **Secure Monitor** — Uses the S_TTBR0, S_TTBR1, and S_TTBCR registers to translate from VA to PA. This space is active when (SCR.NS == 0) or (CPSR.M == MON).
- **NS Hyp** — Uses the HTTBR and HTCCR registers to translate from VA to PA. This space is active when (SCR.NS == 1) and (CPSR.M == HYP).
- **Guest** — Uses the N_TTBR0, N_TTBR1, and N_TTBCR registers to translate from VA to PA. This space is active when (SCR.NS == 1) and (CPSR.M != HYP) and (CPSR.M != MON).
- **Physical Memory (Secure)** — Main memory space visible from the Memory view.

1.5.4 Disassembly View

SoC Designer Simulator supports a disassembly view of a program running on the Cortex-A72 Cycle Model. To display the disassembly view in the SoC Designer Simulator, right-click on the Cortex-A72 Cycle Model and select *View Disassembly…* from the context menu. Refer to the *SoC Designer Plus User Guide* for more information.
1.6 Available Profiling Data

Profiling data is enabled, and can be viewed using the Profiling Manager, which is accessible via the Debug menu in the SoC Designer Simulator. Both hardware and software based profiling are available.

1.6.1 Hardware Profiling

Hardware events are uniquely identified by their Event Number as defined in the Cortex-A72 TRM. The event names that appear in the Profiling Manager view are a concatenation of the event number and a shortened form of the event name. If architecture mnemonics have been defined by ARM then that name has been used; otherwise, a short form of the name has been created.

Hardware profiling includes the streams and events shown in Table 1-23.

Table 1-23 Cortex-A72 Profiling Events

<table>
<thead>
<tr>
<th>Stream</th>
<th>Event Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions</td>
<td>0x00_SW_INCR</td>
<td>Instruction architecturally executed</td>
</tr>
<tr>
<td></td>
<td>0x08_INST_RETIRED</td>
<td>Instructions architecturally executed</td>
</tr>
<tr>
<td></td>
<td>0x09_EXC_TAKEN</td>
<td>Exception taken</td>
</tr>
<tr>
<td></td>
<td>0x0A_EXC_RETURN</td>
<td>Exception return architecturally executed</td>
</tr>
<tr>
<td></td>
<td>0x0B_CID_WRITE_RETIRED</td>
<td>Counts the number of instructions architecturally executed writing into the ContextID Register</td>
</tr>
<tr>
<td></td>
<td>0x1B_INST_SPEC</td>
<td>Operation speculatively executed</td>
</tr>
<tr>
<td></td>
<td>0x81_EXC_UNDEF</td>
<td>Exception taken: other synchronous</td>
</tr>
<tr>
<td></td>
<td>0x82_EXC_SVC</td>
<td>Exception taken: Supervisor Call</td>
</tr>
<tr>
<td></td>
<td>0x83_EXC_PABORT</td>
<td>Exception taken: Instruction Abort</td>
</tr>
<tr>
<td></td>
<td>0x84_EXC_DABORT</td>
<td>Exception taken: Data Abort or SError</td>
</tr>
<tr>
<td></td>
<td>0x86_EXC_IRQ</td>
<td>Exception taken: IRQ</td>
</tr>
<tr>
<td></td>
<td>0x87_EXC_FIQ</td>
<td>Exception taken: FIQ</td>
</tr>
<tr>
<td></td>
<td>0x88_EXC_SMC</td>
<td>Exception taken: Secure Monitor Call</td>
</tr>
<tr>
<td></td>
<td>0x8A_EXC_HVC</td>
<td>Exception taken: Hypervisor Call</td>
</tr>
<tr>
<td></td>
<td>0x8B_EXC_TRAP_P</td>
<td>Exception taken: Instruction Abort not taken locally</td>
</tr>
<tr>
<td></td>
<td>0x8C_EXC_TRAP_D</td>
<td>Exception taken: Data Abort, or SError not taken locally</td>
</tr>
<tr>
<td></td>
<td>0x8D_EXC_TRAP_O</td>
<td>Exception taken: Other traps not taken locally</td>
</tr>
<tr>
<td>Stream</td>
<td>Event Name</td>
<td>Comments</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Instructions (continued)</td>
<td>0x8E_EXC_TRAP_I</td>
<td>Exception taken: IRQ not taken locally</td>
</tr>
<tr>
<td></td>
<td>0x8F_EXC_TRAP_F</td>
<td>Exception taken: FIQ not taken locally</td>
</tr>
<tr>
<td></td>
<td>0x90_RC_LD_SPEC</td>
<td>Release consistency instruction</td>
</tr>
<tr>
<td></td>
<td>0x91_RC_ST_SPEC</td>
<td>Release consistency instruction</td>
</tr>
<tr>
<td>Pipeline</td>
<td>0x76_PC_WRITE_S</td>
<td>Operation speculatively executed - Software change of the PC</td>
</tr>
<tr>
<td></td>
<td>0x10_BR_MIS_PRED</td>
<td>Mispredicted or not predicted branch speculatively executed</td>
</tr>
<tr>
<td></td>
<td>0x12_BR_PRED</td>
<td>Predictable branch speculatively executed</td>
</tr>
<tr>
<td></td>
<td>0x6C_LDREX_SPEC</td>
<td>Exclusive operation speculatively</td>
</tr>
<tr>
<td></td>
<td>0x6D_STREX_PASS</td>
<td>Exclusive instruction speculatively</td>
</tr>
<tr>
<td></td>
<td>0x6E_STREX_FAIL</td>
<td>Exclusive operation speculatively</td>
</tr>
<tr>
<td></td>
<td>0x70_LD_SPEC</td>
<td>Operation speculatively executed - Load</td>
</tr>
<tr>
<td></td>
<td>0x71_ST_SPEC</td>
<td>Operation speculatively executed - Store</td>
</tr>
<tr>
<td></td>
<td>0x72_LDST_SPEC</td>
<td>Operation speculatively executed - Load or store</td>
</tr>
<tr>
<td></td>
<td>0x73_DP_SPEC</td>
<td>Operation speculatively executed - Integer data processing</td>
</tr>
<tr>
<td></td>
<td>0x74_ASE_SPEC</td>
<td>Operation speculatively executed - Advanced SIMD</td>
</tr>
<tr>
<td></td>
<td>0x75_VFP_SPEC</td>
<td>Operation speculatively executed - VFP</td>
</tr>
<tr>
<td></td>
<td>0x77_CRYPTO_SPE</td>
<td>Operation speculatively executed, crypto data processing</td>
</tr>
<tr>
<td></td>
<td>0x78_BR_IMMED_S</td>
<td>Branch speculatively executed - Immediate branch</td>
</tr>
<tr>
<td></td>
<td>0x79_BR_RETURN_</td>
<td>Branch speculatively executed - Procedure return</td>
</tr>
<tr>
<td></td>
<td>0x7A_BR_INDIREC</td>
<td>Branch speculatively executed - Indirect branch</td>
</tr>
<tr>
<td></td>
<td>0x7C_ISB_SPEC</td>
<td>Barrier speculatively executed - ISB</td>
</tr>
<tr>
<td></td>
<td>0x7D_DSB_SPEC</td>
<td>Barrier speculatively executed - DSB</td>
</tr>
<tr>
<td></td>
<td>0x7E_DMB_SPEC</td>
<td>Barrier speculatively executed - DMB</td>
</tr>
<tr>
<td>I-Cache</td>
<td>0x01_L1I_CACHE_REFILL</td>
<td>Level 1 instruction cache refill</td>
</tr>
<tr>
<td></td>
<td>0x14_L1I_CACHE</td>
<td>Level 1 instruction cache access</td>
</tr>
<tr>
<td>Stream</td>
<td>Event Name</td>
<td>Comments</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>D-Cache</td>
<td>0x03_L1D_CACHE_MISS</td>
<td>Data cache miss</td>
</tr>
<tr>
<td></td>
<td>0x04_L1D_CACHE_ACCESS</td>
<td>Data read or write operation that causes a cache access at (at least) the lowest level of data or unified cache</td>
</tr>
<tr>
<td></td>
<td>0x15_L1D_CACHE_WB</td>
<td>Level 1 data cache write back</td>
</tr>
<tr>
<td></td>
<td>0x40_L1D_CACHE_LD</td>
<td>Level 1 data cache access - Read</td>
</tr>
<tr>
<td></td>
<td>0x41_L1D_CACHE_ST</td>
<td>Level 1 data cache access - Write</td>
</tr>
<tr>
<td></td>
<td>0x42_L1D_CACHE_REFILL_LD</td>
<td>Level 1 data cache refill - Read</td>
</tr>
<tr>
<td></td>
<td>0x43_L1D_CACHE_REFILL_ST</td>
<td>Level 1 data cache refill - Write</td>
</tr>
<tr>
<td></td>
<td>0x46_L1D_CACHE_WB_VICTIM</td>
<td>Level 1 data cache Write-back - Victim</td>
</tr>
<tr>
<td></td>
<td>0x47_L1D_CACHE_WB_CLEAN</td>
<td>Level 1 data cache Write-back - Cleaning</td>
</tr>
<tr>
<td></td>
<td>0x48_L1D_CACHE_INVAL</td>
<td>Level 1 data cache invalidate</td>
</tr>
<tr>
<td></td>
<td>0x16_L2D_CACHE</td>
<td>Level 2 data cache access</td>
</tr>
<tr>
<td></td>
<td>0x17_L2D_CACHE_REFILL</td>
<td>Level 2 data cache refill</td>
</tr>
<tr>
<td></td>
<td>0x18_L2D_CACHE_WB</td>
<td>Level 2 data cache Write-Back</td>
</tr>
<tr>
<td></td>
<td>0x50_L2D_CACHE_LD</td>
<td>Level 2 data cache access - Read</td>
</tr>
<tr>
<td></td>
<td>0x51_L2D_CACHE_ST</td>
<td>Level 2 data cache access - Write</td>
</tr>
<tr>
<td></td>
<td>0x52_L2D_CACHE_REFILL_LD</td>
<td>Level 2 data cache refill - Read</td>
</tr>
<tr>
<td></td>
<td>0x53_L2D_CACHE_REFILL_ST</td>
<td>Level 2 data cache refill - Write</td>
</tr>
<tr>
<td></td>
<td>0x56_L2D_CACHE_WB_VICTIM</td>
<td>Level 2 data cache Write-back - Victim</td>
</tr>
<tr>
<td></td>
<td>0x57_L2D_CACHE_WB_CLEAN</td>
<td>Level 2 data cache Write-back - Cleaning</td>
</tr>
<tr>
<td></td>
<td>0x58_L2D_CACHE_INVAL</td>
<td>Level 2 data cache invalidate</td>
</tr>
<tr>
<td>Cycle</td>
<td>0x11_CPU_CYCLES</td>
<td>Cycle</td>
</tr>
<tr>
<td></td>
<td>0x1D_BUS_CYCLES</td>
<td>Bus cycle</td>
</tr>
</tbody>
</table>
Table 1-23  Cortex-A72 Profiling Events  (continued)

<table>
<thead>
<tr>
<th>Stream</th>
<th>Event Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microarchitecture</td>
<td>0x02_L1I_TLB_REFIL</td>
<td>Instruction TLB refill</td>
</tr>
<tr>
<td></td>
<td>0x05_L1D_TLB_REFILL</td>
<td>Data TLB refill</td>
</tr>
<tr>
<td></td>
<td>0x13_MEM_ACCESS</td>
<td>Data memory access</td>
</tr>
<tr>
<td></td>
<td>0x19_BUS_ACCESS</td>
<td>Bus access</td>
</tr>
<tr>
<td></td>
<td>0x1A_MEMORY_ERR</td>
<td>Local memory error</td>
</tr>
<tr>
<td></td>
<td>0x1C_TTBR_WRITE</td>
<td>Instruction architecturally executed</td>
</tr>
<tr>
<td></td>
<td>0x1E_CHAIN</td>
<td>Performance counter chain mode</td>
</tr>
<tr>
<td></td>
<td>0x4C_L1D_TLB_REFILL_LD</td>
<td>Level 1 data TLB refill - Read</td>
</tr>
<tr>
<td></td>
<td>0x4D_L1D_TLB_REFILL_ST</td>
<td>Level 1 data TLB refill - Write</td>
</tr>
<tr>
<td></td>
<td>0x60_BUS_ACCESS_L</td>
<td>Bus access - Read</td>
</tr>
<tr>
<td></td>
<td>0x61_BUS_ACCESS_ST</td>
<td>Bus access - Write</td>
</tr>
<tr>
<td></td>
<td>0x62_BUS_ACCESS_SHARED</td>
<td>Bus access - Normal</td>
</tr>
<tr>
<td></td>
<td>0x63_BUS_ACCESS_NOT_SHARED</td>
<td>Bus access - Not normal</td>
</tr>
<tr>
<td></td>
<td>0x64_BUS_ACCESS_NORMAL</td>
<td>Bus access - Normal</td>
</tr>
<tr>
<td></td>
<td>0x65_BUS_ACCESS_PERIPH</td>
<td>Bus access - Peripheral</td>
</tr>
<tr>
<td></td>
<td>0x66_MEM_ACCESS_L</td>
<td>Data memory access - Read</td>
</tr>
<tr>
<td></td>
<td>0x67_MEM_ACCESS_ST</td>
<td>Data memory access - Write</td>
</tr>
<tr>
<td></td>
<td>0x68_UNALIGNED_LD_SPEC</td>
<td>Unaligned access - Read</td>
</tr>
<tr>
<td></td>
<td>0x69_UNALIGNED_ST_SPEC</td>
<td>Unaligned access - Write</td>
</tr>
<tr>
<td></td>
<td>0x6A_UNALIGNED_LDST_SPEC</td>
<td>Unaligned access</td>
</tr>
<tr>
<td>IMPL_ BranchPrediction</td>
<td>0x100_BR_PRED_BTB</td>
<td>Number of uBTB or BTB predicted branches</td>
</tr>
<tr>
<td></td>
<td>0x101_BR_PRED_STATIC</td>
<td>Number of predictable branches predicted by static predictor</td>
</tr>
<tr>
<td></td>
<td>0x102_BR_PRED_BTB_INDIR</td>
<td>Number of predictable branches predicted by BTB/uBTB that are indirect</td>
</tr>
<tr>
<td></td>
<td>0x103_BR_PRED_BTB_COND</td>
<td>Number of uBTB/ BTB predicted conditional branches</td>
</tr>
<tr>
<td></td>
<td>0x104_BR_COND_SPEC</td>
<td>Number of predictable branches that are conditional</td>
</tr>
<tr>
<td></td>
<td>0x105_BR_TAKEN_SPEC</td>
<td>Number of predictable branches that are taken</td>
</tr>
<tr>
<td></td>
<td>0x106_BR_MIS_PRED_BTB_DIR</td>
<td>Number of directional mispredicted uBTB/ BTB predicted branches</td>
</tr>
<tr>
<td></td>
<td>0x107_BR_MIS_PRED_BTB</td>
<td>Number of mispredicted branches predicted by BTB/uBTB</td>
</tr>
<tr>
<td>Stream</td>
<td>Event Name</td>
<td>Comments</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>IMPL_ BranchPrediction (continued)</td>
<td>0x108_BR_MIS_PRED_BTB_INDIR</td>
<td>Number of mispredicted indirect branch predicted by BTB/uBTB</td>
</tr>
<tr>
<td></td>
<td>0x109_BR_MIS_PRED_RETURN</td>
<td>Mispredicted procedure returns</td>
</tr>
<tr>
<td></td>
<td>0x10A_BR_MIS_PRED_INDIRECT</td>
<td>Number of mispredicted indirect branches</td>
</tr>
<tr>
<td></td>
<td>0x10B_BR_UNPRED_NT_OK</td>
<td>Number of unpredicted NT branches</td>
</tr>
<tr>
<td>IMPL_ QueuesRenameMisc</td>
<td>0x10C_FETCHQ_EMPTY_CYCLES</td>
<td>Number of cycles that the fetch queue contains no instructions</td>
</tr>
<tr>
<td></td>
<td>0x110_SINGLE_UOP_SEQUENCE</td>
<td>cycle count: single uop decoded/sequenced this cycle.</td>
</tr>
<tr>
<td></td>
<td>0x111_RENAMEQ_EMPTY_CYCLES</td>
<td>cycle count: Rename queue empty</td>
</tr>
<tr>
<td></td>
<td>0x112_RENAME_FLAG_STALL</td>
<td>cycle count: rename stall due to flags</td>
</tr>
<tr>
<td></td>
<td>0x113_RENAME_REG_STALL</td>
<td>cycle count: rename stall due to registers</td>
</tr>
<tr>
<td></td>
<td>0x118_UOP_COUNT</td>
<td>uop count: total number of micro-ops renamed.</td>
</tr>
<tr>
<td></td>
<td>0x120_DISPQ_EMPTY_CYCLES</td>
<td>cycle count: Dispatch queue empty</td>
</tr>
<tr>
<td></td>
<td>0x121_ISSUEQ_FULL_BX_STALL</td>
<td>Dispatch stall due to issue queue full BX</td>
</tr>
<tr>
<td></td>
<td>0x122_ISSUEQ_FULL_CX_STALL</td>
<td>Dispatch stall due to issue queue full CX1 or CX2</td>
</tr>
<tr>
<td></td>
<td>0x123_ISSUEQ_FULL_LS_STALL</td>
<td>Dispatch stall due to issue queue full LS</td>
</tr>
<tr>
<td></td>
<td>0x124_ISSUEQ_FULL_MX_STALL</td>
<td>Dispatch stall due to issue queue full MX</td>
</tr>
<tr>
<td></td>
<td>0x125_ISSUEQ_FULL_SP_STALL</td>
<td>Dispatch stall due to issue queue full SP</td>
</tr>
<tr>
<td></td>
<td>0x126_ISSUEQ_FULL_SX_STALL</td>
<td>Dispatch stall due to issue queue full SX1 or SX2</td>
</tr>
<tr>
<td></td>
<td>0x127_FLUSH_BAD_BRANCH</td>
<td>Bad branch flush</td>
</tr>
<tr>
<td></td>
<td>0x128_FLUSH_MEM</td>
<td>MEM NUKE Flush (i.e ECC flush + RAR flush)</td>
</tr>
<tr>
<td></td>
<td>0x129_FLUSH_RARE</td>
<td>Rare flushes</td>
</tr>
<tr>
<td></td>
<td>0x12A_FLUSH_SWDW</td>
<td>SWDW_NUKE</td>
</tr>
</tbody>
</table>

Table 1-23 Cortex-A72 Profiling Events (continued)
Table 1-23 Cortex-A72 Profiling Events (continued)

<table>
<thead>
<tr>
<th>Stream</th>
<th>Event Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPL_LoadStore-Cache</td>
<td>0x12B_FLUSH_NEON_COND</td>
<td>Neon conditional flush</td>
</tr>
<tr>
<td></td>
<td>0x12C_DISP_SWDWSTALL</td>
<td>Dispatch stall due to single word producer double word consumer issue</td>
</tr>
<tr>
<td></td>
<td>0x140_GRP_COMMIT_COUNT</td>
<td>Number of groups tally</td>
</tr>
<tr>
<td></td>
<td>0x141_ETM_EXTOUT_0</td>
<td>ETM EXTOUT 0</td>
</tr>
<tr>
<td></td>
<td>0x142_ETM_EXTOUT_1</td>
<td>ETM EXTOUT 1</td>
</tr>
<tr>
<td></td>
<td>0x143_ETM_EXTOUT_2</td>
<td>ETM EXTOUT 2</td>
</tr>
<tr>
<td></td>
<td>0x144_ETM_EXTOUT_3</td>
<td>ETM EXTOUT 3</td>
</tr>
<tr>
<td></td>
<td>0x150_LS_LD_RESTART_CNT</td>
<td>Load Store Load Restart Count</td>
</tr>
<tr>
<td></td>
<td>0x151_LS_LD_ORDER_HAZARD</td>
<td>Loadstore Load Pipe Hazard</td>
</tr>
<tr>
<td></td>
<td>0x152_LS_ST_ORDER_HAZARD</td>
<td>Loadstore Store Pipe Hazard</td>
</tr>
<tr>
<td></td>
<td>0x153_LS_PM_L1_PF_REQ_GEN_PAGE</td>
<td>L1 prefetch request generated in page mode</td>
</tr>
<tr>
<td></td>
<td>0x154_LS_PM_L1_PF_REQ_GEN_STRIDE</td>
<td>L1 prefetch request generated by the load stride prefetcher</td>
</tr>
<tr>
<td></td>
<td>0x155_LS_PM_L2_PF_REQ_GEN_LSLD</td>
<td>L2 prefetch request generated by a load instruction</td>
</tr>
<tr>
<td></td>
<td>0x156_LS_PM_L2_PF_REQ_GEN_LST</td>
<td>L2 prefetch request generated by a store instruction</td>
</tr>
<tr>
<td></td>
<td>0x157_LS_PM_PF_TRAIN_TABLE_ALLOC</td>
<td>New train table entry allocated by a load</td>
</tr>
<tr>
<td></td>
<td>0x158_LS_PF_PHT_LOOKUP</td>
<td>Lookup generated for the PF pattern history ram</td>
</tr>
<tr>
<td></td>
<td>0x159_LS_PM_PF_GEN_TABLE_ALLOC</td>
<td>New generation table allocation for L2 prefetch</td>
</tr>
<tr>
<td></td>
<td>0x15A_LS_PM_PF_GEN_TABLE_ALLOC_PEND</td>
<td>New generation table allocation for L2 prefetch replacing an entry that still had pending prefetches</td>
</tr>
<tr>
<td></td>
<td>0x15B_LS_PMU_L1_DTLB_REFIL_L_PF</td>
<td>L1 dtlb refill's that are initiated by PF</td>
</tr>
<tr>
<td></td>
<td>0x15C_LS_PMU_L1_DCACHE_REFILL_P</td>
<td>L1 dcache refill's that are initiated by PF</td>
</tr>
<tr>
<td></td>
<td>0x160_BUS_TRANS_RD</td>
<td>Bus read transaction</td>
</tr>
<tr>
<td></td>
<td>0x161_BUS_TRANS_WR</td>
<td>Bus write transaction</td>
</tr>
<tr>
<td></td>
<td>0x162_BUS_ACCESS_SNOOP</td>
<td>Bus access - snoop</td>
</tr>
<tr>
<td></td>
<td>0x163_L2_PF_REQ_BUS_TRANS</td>
<td>L2 prefetch initiated by bus transaction</td>
</tr>
<tr>
<td></td>
<td>0x164_L2_ACP_MAS_RD_DATA</td>
<td>ACP based AXI master port read data</td>
</tr>
<tr>
<td></td>
<td>0x165_L2_ACP_MAS_WR_DATA</td>
<td>ACP based AXI master port write data</td>
</tr>
</tbody>
</table>
Table 1-23 Cortex-A72 Profiling Events  (continued)

<table>
<thead>
<tr>
<th>Stream</th>
<th>Event Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPL_LoadStore-Cache (continued)</td>
<td>0x166_L2_PF_REQ_GENERATED_IF</td>
<td>L2 Prefetch request generated due to an instr access</td>
</tr>
<tr>
<td></td>
<td>0x167_L2_PF_REQ_GENERATED_TW</td>
<td>L2 Prefetch request generated due to a TBW desc fetch</td>
</tr>
<tr>
<td></td>
<td>0x168_L2_PF_REQ_BUS_TRANS_LD</td>
<td>Prefetch initiated by a load access issued on the bus</td>
</tr>
<tr>
<td></td>
<td>0x169_L2_PF_REQ_BUS_TRANS_ST</td>
<td>Prefetch initiated by a store access issued on the bus</td>
</tr>
<tr>
<td></td>
<td>0x16A_L2_FEQSTALL</td>
<td>L2 Stalled due to no FEQ's available</td>
</tr>
<tr>
<td></td>
<td>0x16B_L2_CPU_INCOMING_CCB_XFER</td>
<td>L2 CCB transfer from another cpu's L1D cache to this cpu</td>
</tr>
<tr>
<td></td>
<td>0x170_L2_TBW_DESC_ACCESS</td>
<td>L2 TBW descriptor read access</td>
</tr>
<tr>
<td></td>
<td>0x171_L2_TBW_IPA_PA_CACHE_ACCESS</td>
<td>L2 IPA-PA cache accesses.</td>
</tr>
<tr>
<td></td>
<td>0x172_L2_TBW_IPA_PA_CACHE_HIT</td>
<td>L2 IPA-PA cache hit</td>
</tr>
<tr>
<td></td>
<td>0x173_L2_TBW_S1_L2_PA_CACHE_E_HIT</td>
<td>L2 S1 L2 PA Cache hit</td>
</tr>
<tr>
<td></td>
<td>0x174_L2_TBW_S1_WLK_CACHE_HIT</td>
<td>L2 S1 wlk cache hit</td>
</tr>
<tr>
<td></td>
<td>0x175_L2_TLB_ACCESS_ARRAY</td>
<td>L2 TLB No of accesses made to the array during lookup.</td>
</tr>
<tr>
<td></td>
<td>0x176_L2_TLB_MAINT_OPS</td>
<td>L2 TLB shootdowns</td>
</tr>
<tr>
<td></td>
<td>0x177_L2_TLB_ACCESS</td>
<td>L2 TLB accesses (lookups)</td>
</tr>
<tr>
<td></td>
<td>0x178_L2_TLB_ACCESS_PF</td>
<td>L2 TLB accesses for table prefetces</td>
</tr>
<tr>
<td></td>
<td>0x179_L2_TLB_REFILL</td>
<td>L2 TLB refills</td>
</tr>
<tr>
<td></td>
<td>0x17A_L2_TLB_REFILL_STRUCTURE</td>
<td>L2 TLB refills for inst fetches</td>
</tr>
<tr>
<td></td>
<td>0x17B_L2_TLB_REFILL_LDST</td>
<td>L2 TLB refills for loads and stores</td>
</tr>
<tr>
<td></td>
<td>0x17C_L2_TLB_REFILL_PF</td>
<td>L2 TLB refills for tablewalk prefetces</td>
</tr>
<tr>
<td></td>
<td>0x17D_L2_PF_REQ_BUS_TRANS_IF</td>
<td>Prefetch initiated by an instr access issued on the bus</td>
</tr>
<tr>
<td></td>
<td>0x17E_PF_REQ_BUS_TRANS_TW</td>
<td>Prefetch initiated by a TBW desc fetch issued on the bus</td>
</tr>
<tr>
<td></td>
<td>0x17F_PF_FEQ_LATE</td>
<td>L2 Prefetch request late (demand hit outstanding PF request in FEQ)</td>
</tr>
<tr>
<td></td>
<td>0x180_L2_LD_PF_REQ_HIT_L2</td>
<td>Prefetch initiated by a load instruction hit in the L2</td>
</tr>
<tr>
<td></td>
<td>0x181_L2_ST_PF_REQ_HIT_L2</td>
<td>Prefetch initiated by a store instruction hit in the L2</td>
</tr>
</tbody>
</table>
1.6.2 Software Profiling

Software-based profiling is provided by SoC Designer Plus. Profiling information is also available in the SoC Designer Profiler. See the user guide for SoC Designer Plus or SoC Designer Profiler for more information.

<table>
<thead>
<tr>
<th>Stream</th>
<th>Event Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPL_LoadStore-Cache (continued)</td>
<td>0x182_L2_IF_PF_REQ_HIT_L2</td>
<td>Prefetch initiated by an instruction fetch hit in the L2</td>
</tr>
<tr>
<td></td>
<td>0x183_L2_TW_PF_REQ_HIT_L2</td>
<td>Prefetch initiated by a TBW desc fetch hit in the L2</td>
</tr>
<tr>
<td></td>
<td>0x184_L2_PF_REQ GENERATED</td>
<td>L2 prefetch generated</td>
</tr>
</tbody>
</table>
Third Party Software Acknowledgement

ARM acknowledges and thanks the respective owners for the following software that is used by our product:

- ELF (Executable and Linking Format) Tool Chain Product

Copyright (c) 2006, 2008-2015 Joseph Koshy

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.