Cortex™-A72 Model

Version 8.0.0

User Guide

Non-Confidential

ARM®

ARM DUI 0947 (ID030116)

Cortex-A72 Model
User Guide

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
February 2016	A	Non-Confidential	Restamping/AXF Load update

Non-Confidential Proprietary Notice

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

A Model component is a library developed from ARM intellectual property (IP) that is generated through Carbon Model Studio™. The 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 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:

Convention	Description	Example
`courier`	Commands, functions, variables, routines, and code examples that are set apart from ordinary text.	`sparseMem_t SparseMemCreate-New();`
italic	New or unusual words or phrases appearing for the first time.	*Transactors* provide the entry and exit points for data ...
bold	Action that the user performs.	Click **Close** to close the dialog.
<text>	Values that you fill in, or that the system automatically supplies.	<platform>/ represents the name of various platforms.
[text]	Square brackets [ ] indicate optional text.	`$CARBON_HOME/bin/modelstudio [ <filename> ]`
[text1 | text2]	The vertical bar | indicates "OR," meaning that you can supply text1 or text 2.	`$CARBON_HOME/bin/modelstudio [<name>.symtab.db | <name>.ccfg ]`

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 for access to ARM documentation.

The following publications provide additional information on simulation:

- IEEE 1666™ SystemC Language Reference Manual, (IEEE Standards Association)
- SPIRIT User Guide, Revision 1.2, SPIRIT Consortium.

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.
Model	A software object created by the Carbon Model Studio (or *Carbon compiler*) from an RTL design. The 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 *Model*, and it also takes a Model as input and generates a component that can be used in SoC Designer Plus, Platform Architect, or OSCI 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 Model Kit Component in SoC Designer Plus

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

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.
- Large Physical Address (LPA) Extension.

1.1.1 Features Additional to the Hardware

The following features that are implemented in the Cortex-A72 model do not exist in the Cortex-A72 hardware. These features have been added to the 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 1-17).
- 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 model get into a non-debuggable state. See “Run To Debug Point” on page 1-42 for more information.
- Waveform dumping using the waveform-related parameters described in Table 1-3 on page 1-9.
- Support for viewing memory spaces (see “Memory Information” on page 1-42).
- Support for viewing disassembly data (see “Disassembly View” on page 1-42).

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.

- Carbon SoC Designer Plus Component Files
- Adding the 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

Platform	File	Description
Linux	maxlib.lib<*model_name*>.conf	SoC Designer Plus configuration file
	lib<*component_name*>.mx.so	SoC Designer Plus component runtime file
	lib<*component_name*>.mx_DBG.so	SoC Designer Plus component debug file
Windows	maxlib.lib<*model_name*>.windows.conf	SoC Designer Plus configuration file
	lib<*component_name*>.mx.dll	SoC Designer Plus component runtime file
	lib<*component_name*>.mx_DBG.dll	SoC Designer Plus component debug file

Additionally, this User Guide PDF file is provided with the component.

1.2.2 Adding the Model to the Component Library

The compiled Model component is provided as a configuration file (*.conf*). To make the component available in the Component Window in SoC Designer Canvas, 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 1-15).

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 model. Certain hardware pins have been converted to init-time model parameters.

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

ESL Port	Description	Type
ACE_Master	ACE Master S2T when configured in ACE mode. IP configurable.	Transaction Master
ACP_Slave	Optional Accelerator Coherency Port implemented as an AXI4 slave interface. IP configurable.	Transaction Slave
CHI_RNF_CHI_Master	CHI Master S2T when configured with the CHI interface. IP configurable.	Transactor Master
Debug_APB	APB3 Transactor Slave.	Transaction Slave
CLK	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.	Main Clock Transactor (Clock Slave)
clk_in	This port is used internally. Leave unconnected.	Clock Slave

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 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 1-3 Component Parameters

Name	Description	Allowed Values	Default Value	Init/ Runtime
AA64nAA32	Determines whether processor boots in 32-bit or 64-bit mode.	0 – Boots processor in 32-bit mode. 1 – Boots processor in 64-bit mode.	0	Init
AA64nAA320	Register width state: 0 - AArch32. 1 - AArch64.	bool	0	Init
ACE_Master Enable Debug Messages	Enables ACE_Master port debug.	true, false	false	Runtime
ACE_Master Protocol Variant	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_S*n*_NID*m* while ACE_Lite+DVM yields ACE_LITE_DVM_S*n*_NID*m*.	ACE-Lite or ACELite+DVM	ACE-Lite or ACELite+DVM	Init
ACINACTM	Snoop interface is inactive and no longer accepting requests.	0, 1	0	Runtime
ACLKENM	ACE Master Input Clock Enable	0, 1	1	Runtime
ACLKENS	ACP AXI Slave Input Clock Enable	0, 1	1	Runtime
ACP_Slave axi_size [0-5][1]	These parameters should be left at their default values.	—	0	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
ACP_Slave axi_start [0-5][1]	These parameters should be left at their default values.	—	0	Runtime
ACP_Slave Enable Debug Messages [1]	Enables ACP_Slave port debug.	true, false	false	Runtime
ACP_Slave Protocol Variant[1]	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_S*n*_NID*m* while ACE_Lite+DVM yields ACE_LITE_DVM_S*n*_NID*m*.	ACE-Lite or ACELite+DVM	ACE-Lite or ACELite+DVM	Init
AFVALIDMx	Fifo flush request. This signal is part of the ATB interface.	0, 1	0	Runtime
Align Waveforms	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.	true, false	true	Init
ATCLKEN	ATB clock enable.	0, 1	0	Runtime
ATREADYMx	ATB device ready.	0, 1	0	Runtime
Carbon DB Path	Sets the directory path to the database file.	Not Used	empty	Init
CFGEND	Endianess configuration. 1-bit wide for UP, 4 bits wide for MP. Automatically kept in sync with CFGEND*n*.	integer	0	Init
CFGEND*n*	Endianess configuration. Per-core value of CFGEND; automatically kept in sync with CFGEND.	bool	false	Init
CFGTE	Default exception handling state (ARM/Thumb). 1-bit wide for UP, 4 bits wide for MP. Automatically kept in sync with CFGTE*n*.	integer	0	Init
CFGTE*n*	Default exception handling state (ARM/Thumb). Per-core value of CFGTE; automatically kept in sync with CFGTE.	bool	false	Init

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
CHI_RNF_CHI_Master Enable Debug Message	Whether debug messages are enabled on the CHI Master port.	True, False	False	Runtime
CHI_RNF_CHI_Master Protocol Variant	Protocol Variant in use for CHI.	CHI-RNF	CHI-RNF	Init
CLKEN	Clock enable.	0, 1	1	Runtime
CLUSTERIDAFF1	Value read in the Cluster ID Affinity Level-1 field, bits[15:8], of the Multi-processor Affinity Register (MPIDR).	integer	0	Init
CLUSTERIDAFF2	Individual processor register width state.	integer	0	Init
CNTCLKEN	Counter clock enable. This clock enable must be inserted one cycle before the CNTVALUEB bus.	0, 1	0	Runtime
CoherencyType	Drives the Coherency signals; supported for ACE and CHI configurations. Supported values: NonCoherentNoL3[2] NonCoherentWithL3[2] OuterCoherentNoL3 OuterCoherentWithL3 InnerCoherentNoL3 InnerCoherentWithL3	String	OuterCoherent WithL3	Init
CP15DISABLE	Disable write access to some Secure CP15 registers. This is the aggregated bus [NUM_CPUS-1:0].	bool	false	Runtime
CP15DISABLE*n*	One-bit parameter. Disable write access to DBGL1RSTDISABLE.	0, 1	0	Runtime
CRYPTODISABLE	Individual core Cryptography engine disable. Only available with the Cryptography Extension enabled.	0, 1	0	Init
CRYPTODISABLEn	One-bit parameter. Disable Cryptography engine.	bool	false	Init
DBGL1RSTDISABLE	Disable L1 data cache automatic invalidate on reset functionality: 0 - Enable automatic invalidation of L1 data cache on reset. 1 - Disable automatic invalidation of L1 data cache on reset	0, 1	0	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
DBGPWRDUP	Processor powered-up. 0 - Processor is powered down 1 - Processor is powered up	0, 1	0	Runtime
DBGROMADDR	External debug device CoreSight system configuration. Specifies bits [31:12] of the ROM.	Table Physical Address	0-ffffffff	Init
DBGROMADDRV	Valid signal for DBGROMADDR.	bool	False	Init
Debug_APB Base Address	Start of the Debug_APB port address region.	Address	0x0	Init
Debug_APB Enable Debug Messages	Enable Debug_APB port debug.	true, false	false	Runtime
Debug_APB Size	Size of the Debug_APB port address region.	Size	0x100000000	Init
Dump Waveforms	Whether SoC Designer Plus dumps waveforms for this component.	bool	false	Runtime
Enable Debug Messages	Whether debug messages are logged for the component.	bool	false	Runtime
Fast Application Load Support	Identifies that the component supports fast debug access for application load.	Not configurable	Yes	N/A
Enable Fast Application Load	Controls fast debug access for application load. “True” setting loads multiple bytes at a time; “False” setting loads 1 byte at a time.	true/false	true	Init
GICCDISABLE	Disables the GIC CPU interface logic and routes the legacy nIRQ, nFIQ, nVIRQ, and nVFIQ. Required to enable use of non-ARM interrupt controllers.	bool	If IP is configured with GIC present then the default is false, else default is true.	Init
ICCTREADY	Input AXI4 Stream Protocol signal. GIC CPU Interface to Distributor messages. TREADY indicates that the slave can accept a transfer in the current cycle.	0, 1	0	Runtime
ICDTDATA	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.	[0-0xFFFF]	0	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
ICDTDEST	Input AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. TDEST provides routing information for the data stream.	[0-3]	0	Runtime
ICDTLAST	When HIGH, indicates the boundary of a packet.	0, 1	0	Runtime
ICDTVALID	Input AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. TVALID indicates that the master is driving a valid transfer.	0, 1	0	Runtime
L2RSTDISABLE	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 & Play using L2 cache restore). 0 — Enables the hardware L2 invalidation reset sequence.	1— Disables the reset sequence. 0 — Enables the reset sequence.	1	Init
negLogic	Enables active low interrupts	bool	false	Runtime
NODE ID	Cortex-A72 CHI Node Identifier	Cortex-A72 CHI Node Identifier	0x7	Init
PCLKENDBG	APB DBG Clock Enable.	0,1	1	Runtime
PERIPHBASE	Peripheral base [39:0] (Bits 14 - 0 are ignored)	integer	0x0013000000	Init
PMUSNAPSHOTREQ	PMU snapshot trigger acknowledge.	integer (per-core value)	0	Runtime
RVBARADDRx	Reset Vector Base Address for executing in AArch64 state.	integer	0	Init
SAMADDRMAP [0 - 19]	CHI Region Mapping.	integer	0x0	Init
SAMHNF [0 - 7]NODEID	HN-F node ID	integer	0x0	Init
SAMHNFMODE	HN-F interleaving module	integer	0x0	Init
SAMHNI [0, 1]NODEID	HN-I Node ID	integer	0x0	Init
SAMMNBASE	MN base address	integer	0x90	Init
SAMMNNODEID	MN node ID	integer	0x0	Init
SYNCREQM0	ETM Signal. Synchronization request from trace sink.	0, 1	0	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
SCLKEN	CHI interface bus clock enable.	0, 1	1	Init
SINACT	CHI snoop active	0, 1	0	Init
SYNCREQM[0, 1, 2, 3]	Synchronization request from trace sink	0, 1	0	Init
SYSBARDISABLE	Disable broadcasting of barriers onto the system bus.	0, 1	0	Init
TSVALUEB	ETM signal. Timestamp in binary encoding.	[0-0xFFFFFFFFFFFFFFF]	0	Runtime
VINITHI	Use high vector addresses. 1-bit wide for UP, 4 bits wide for MP. Automatically kept in sync with VINITHI*n*.	integer	0	Init
VINITHI*n*	Use high vector addresses. Per-core value of VINITHI; automatically kept in sync with VINITHI.	bool	false	Init
Waveform File[3]	Name of the waveform file.	*string*	carbon_CORTEXA53.vcd or carbon_CORTEXA53MP.vcd	Init
Waveform Format	The format of the waveform dump file.	VCD, FSDB	VCD	Init
Waveform Timescale	Sets the timescale to be used in the waveform.	Many values in drop-down	1 ns	Init

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 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 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

Name	Description	Type
ExtendedTargetFeatures	Pseudo register that describes additional processor features	Read-Only
PC_MEMSPACE	Pseudo register that indicates the current memory space (0=secure,1=normal)	Read-Only
R0-R14	General purpose registers	Read-Write
R15	PC. Write at debug point only.	Read-Write
CPSR32	Current Program Status Register	Read-Write
SPSR32	Current program status register in usr mode	Read-Write
SPSR_irq	Saved program status register in IRQ mode	Read-Write
SPSR_fiq	Saved program status register in FIQ mode	Read-Write
SPSR_svc	Saved program status register in SVC mode	Read-Write
SPSR_abt	Saved program status register in ABT mode	Read-Write
SPSR_und	Saved program status register in UND mode	Read-Write
SPSR_hyp	Saved program status register in HYP mode	Read-Write
SPSR_mon	Saved program status register in MON mode	Read-Write
R13_svc, R14_svc	R13/R14 in SVC mode	Read-Write
R13_irq, R14_irq	R13/R14 in IRQ mode	Read-Write
R8_fiq — R14_fiq	R8-R14 in FIQ mode	Read-Write
R8_usr — R14_usr	R8-R14 in USR mode	Read-Write
R13_und, R14_und	R13/R14 in UND mode	Read-Write
R13_abt, R14_abt	R13/R14 in ABT mode	Read-Write
R13_mon, R14_mon	R13/R14 in MON mode	Read-Write
R13_hyp	R13 in HYP mode	Read-Write
ELR_hyp	Exception Link Register in HYP mode	Read-Write

1.5.1.2 AArch64 Core Registers

Table 1-5 describes the AArch64 Core registers.

Table 1-5 AArch64 Core Registers

Name	Description	Type
X0-X30	ID registers.	Read-Write
PC	Program Counter register	Read-Write
SP	Stack Pointer register	Read-Write
ELR	Exception Link register	Read-Write
CPSR	Current Program State register	Read-Write
SPSR	Saved Program Status Register for current mode	Read-Write

1.5.1.3 AArch32 Control Registers

Table 1-6 describes the AArch32 Control register group.

Table 1-6 AArch32 Control Registers

Name	Description	Type
SCTLR	System Control Register.	Read-Write
ACTLR	Auxiliary Control Register.	Read-Write
CPACR	Coprocessor Access Control	Read-Write
NSACR	Nonsecure Access Control	Read-Write
TTBR0	Translation Table Base Register 0.	Read-Write
TTBR1	Translation Table Base Register 1.	Read-Write
TTBCR	Translation Table Base Control Register.	Read-Write
DACR	Domain Access Control Register	Read-Write
DFSR	Data Fault Status	Read-Write
IFSR	Instruction Fault Status	Read-Write
ADFSR	Auxiliary Data Fault Status	Read-Write
AIFSR	Auxiliary Instruction Fault Status	Read-Write
DFAR	Data Fault Address	Read-Write
IFAR	Instruction Fault Address	Read-Write
L2CTLR	L2 Control Register	Read-Write
L2ECTLR	L2 Extended Control Register	Read-Write
PRRR	Primary region remap	Read-Write
NMRR	Normal memory remap	Read-Write
MAIR0	Memory Attribute Indirection Register 0	Read-Write
MAIR1	Memory Attribute Indirection Register 1	Read-Write
AMAIR0	Auxiliary Memory Attribute Indirection Register 0	Read-Write

Table 1-6 AArch32 Control Registers (continued)

Name	Description	Type
AMAIR1	Auxiliary Memory Attribute Indirection Register 1	Read-Write
VBAR	Vector Base Address	Read-Write
ISR	Interrupt Status	Read-Only
FCSEIDR	FCSE Process ID	Read-Write
CONTEXTIDR	Context ID	Read-Write
TPIDRURW	User Read-Write Thread ID	Read-Write
TPIDRURO	User Read-Only Thread ID	Read-Write
TPIDRPRW	Privileged Only Thread ID	Read-Write
CNTFRQ	Timer Clk Ticks Per Sec	Read-Write
CNTKCTL	Timer Kernel Control	Read-Write
CNTP_TVAL	Timer Phy Timer Val	Read-Write
CNTP_CTL	Timer Phy Control	Read-Write
CNTV_TVAL	Timer Virt Timer Val	Read-Write
CNTV_CTL	Timer Virt Control	Read-Write
CNTHCTL	Timer Hyp Control	Read-Write
CNTHP_TVAL	Timer Hyp_and_S_Priv Timer Val	Read-Write
CNTHP_CTL	Timer Hyp_and_S_Priv Control	Read-Write
CBAR	Configuration Base Address	Read-Only
VPIDR	Virtualization Processor ID	Read-Write
VMPIDR	Virtualization Multiprocessor ID	Read-Write
HSCTLR	Hypervisor System Control	Read-Write
HACTLR	Hypervisor Auxiliary Control	Read-Write
HCR	Hypervisor Configuration	Read-Write
HCR2	Hypervisor Configuration 2	Read-Write
HDCR	Hypervisor Debug Control	Read-Write
HCPTR	Hypervisor Copro Trap	Read-Write
HSTR	Hypervisor System Trap	Read-Write
HTCR	NSHyp Translation Control	Read-Write
VTCR	Virt Translation Control	Read-Write
HADFSR	Hyp Aux Data Fault Status	Read-Write
HAIFSR	Hyp Aux Inst Fault Status	Read-Write
HSR	Hyp Undef Except Syndrome	Read-Write
HDFAR	Hyp Data Fault Address	Read-Write
HIFAR	Hyp Inst Fault Address	Read-Write

Table 1-6 AArch32 Control Registers (continued)

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

1.5.1.4 AArch64 System Registers

Table 1-7 describes the Aarch64 System registers.

Table 1-7 AArch64 System Registers

Name	Description	Type
MIDR_EL1	Main ID	Read-Only
CTR_EL0	Cache Type	Read-Only
MPIDR_EL1	Multiprocessor Affinity	Read-Only
REVIDR_EL1	Revision ID	Read-Only
ID_PFR0_EL1	Processor Features 0	Read-Only
ID_PFR1_EL1	Processor Features 1	Read-Only
ID_DFR0_EL1	Debug Features 0	Read-Only
ID_AFR0_EL1	Auxiliary Features 0	Read-Only
ID_MMFR0_EL1 — ID_MMFR3_EL1	Memory Model Features 0 - 3	Read-Only
ID_ISAR0_EL1	Instruction Features 0	Read-Only
ID_ISAR1_EL1 — ID_ISAR5_EL1	Instruction Features 1 - 5	Read-Only
MVFR0_EL1 — MVFR2_EL1	Media and VFP Feature 0 - 2	Read-Only
ID_AA64PFR0_EL1	AARCH64 Processor Features 0	Read-Only
ID_AA64PFR1_EL1	AARCH64 Processor Features 1	Read-Only
ID_AA64DFR0_EL1	AARCH64 Debug Features 0	Read-Only
ID_AA64DFR1_EL1	AARCH64 Debug Features 1	Read-Only
ID_AA64AFR0_EL1	AARCH64 Auxiliary Features 0	Read-Only
ID_AA64AFR1_EL1	AARCH64 Auxiliary Features 1	Read-Only
ID_AA64ISR0_EL1	AARCH64 Instruction Features 0	Read-Only
ID_AA64ISR1_EL1	AARCH64 Instruction Features 1	Read-Only
ID_AA64MMFR0_EL1	AARCH64 Memory Features 0	Read-Only
ID_AA64MMFR1_EL1	AARCH64 Memory Features 1	Read-Only
SCTLR_EL1	Control EL1	Read-Write
DCZID_EL0	Data Cache Zero ID	Read-Only
ACTLR_EL1	Auxiliary Control EL1	Read-Write
CPACR_EL1	Coproc Access Control	Read-Write
TTBR0_EL1	Translation Table Base Register 0	Read-Write
TTBR1_EL1	Translation Table Base Register 1	Read-Write
TCR_EL1	Translation Control Register	Read-Write
AFSR0_EL1	Auxiliary Fault Status Register 0	Read-Write

Table 1-7 AArch64 System Registers (continued)

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

Table 1-7 AArch64 System Registers (continued)

Name	Description	Type
MAIR_EL2	Memory Attribute Indirection Register, EL2	Read-Write
AMAIR_EL2	Auxiliary Memory Attribute Indirection Register, EL2	Read-Write
VBAR_EL2	Vector Base Address Register, EL2	Read-Write
TPIDR_EL2	Thread Pointer/ID Register, EL2	Read-Write
DACR32_EL2	Domain Access Control Register, EL2	Read-Write
IFSR32_EL2	Instruction Fault Status Register, EL2	Read-Write
FPEXC32_EL2	Floating Point Exception Register	Read-Write
SCTLR_EL3	Control EL3	Read-Write
ACTLR_EL3	Auxiliary Control EL3	Read-Write
CPTR_EL3	Coprocessor trap register EL3	Read-Write
SCR_EL3	Secure Configuration Register, EL3	Read-Write

1.5.1.5 AArch32 Debug Registers

Table 1-8 describes the AArch32 Debug registers group.

Table 1-8 AArch32 Debug Registers

Name	Description	Access
DBGDSCRint	Debug Status and Control Register (Internal)	Read-Only
DSPSR	Debug Saved Processor Status Register	Read-Only
DBGDIDR	Debug ID Register	Read-Only
DCCINT	Debug Comms Channel Interrupt Enable Register	Read-Write
DBGWFAR	Watchpoint Fault Address Register, RES0	Read-Write
EDECR	External Debug Execution Control Register	Read-Write
DBGDTRRXext	Debug Data Transfer Register, Receive, External View	Read-Write
DBGDTRTXext	Debug Data Transfer Register, Transmit, External View	Read-Write
EDECCR	External Debug Exception Catch Control Register	Read-Write
DBGBVR0 - DBGBVR5	Debug Breakpoint Value Register 0 - Debug Breakpoint Value Register 1	Read-Write
DBGBCR0 - DBGBCR5	Debug Breakpoint Control Registers	Read-Write
DBGWVR0 - DBGWVR3	Debug Watchpoint Value Register 0 - Debug Watchpoint Value Register 3	Read-Write
DBGWCR0 — DBGWCR3	Debug Watchpoint Control Registers	Read-Write

Table 1-8 AArch32 Debug Registers (continued)

Name	Description	Access
DBGDRAR	Debug ROM Address Register	Read-Write
DBGBXVR4	Breakpoint Extended Value 4	Read-Write
DBGBXVR5	Breakpoint Extended Value 5	Read-Write
DBGWXVR0	Watchpoint eXtended Value 0	Read-Write
DBGWXVR1	Watchpoint eXtended Value 1	Read-Write
DBGWXVR2	Watchpoint eXtended Value 2	Read-Write
DBGWXVR3	Watchpoint eXtended Value 3	Read-Write
OSLSR_EL1	OS Lock Status	Read-Only
DBGOSDLR	OS Double Lock	Read-Write
EDPRCR	Device Powerdown and Reset Control	Read-Write
EDPRSR	Device Powerdown and Reset Status	Read-Only
DBGDSAR	Debug Self Address Offset	Read-Only
DBGAUTHSTATUS	Authentication Status	Read-Only
EDDEVID0	Debug Device ID register 0	Read-Write
EDDEVID1	Debug Device ID register 1	Read-Only
EDDEVTYPE	Debug Device Type	Read-Only
EPID0 - EPID4	Peripheral ID Register 0-4	Read-Write
ECID0 - ECID3	Component ID Register 0-3	Read-Write

1.5.1.6 External Debug Registers

Table 1-9 describes the External Debug registers group.

Table 1-9 External Debug Registers

Name	Description	Access
EDESR	External Debug Event Status Register	Read-Write
EDWARLO	External Debug Watchpoint Address Register lo	Read-Only
EDWARHI	External Debug Watchpoint Address Register hi	Read-Only
EDACR	External Debug Auxiliary Control Register	Read-Write
EDPCSRLO	External Debug Program Counter Sample Register, low word	Read-Only
EDPCSRHI	External Debug Program Counter Sample Register, high word	Read-Only
EDCIDSR	External Debug Context ID Sample Register	Read-Only
EDVIDSR	External Debug Virtual Context Sample Register	Read-Only
DBGBXVR0 - DBGBXVR3	Breakpoint Extended Value Registers	Read-Write

Table 1-9 External Debug Registers (continued)

Name	Description	Access
EDDEVARCH	External Debug Device Architecture Register	Read-Only
EDDEVAFF0	Multiprocessor Affinity Register	Read-Only
EDDEVAFF1	External Debug Device Affinity Register 1, RES0	Read-Only
EDLSR	External Debug Lock Status Register	Read-Only

1.5.1.7 AArch64 Debug Registers

Table 1-10 describes the AArch64 Debug registers group.

Table 1-10 AArch64 Debug Registers

Name	Description	Access
MDCR_EL3	Monitor Debug Configuration Register, EL3	Read-Write
SDER32_EL3	Secure Debug Enable Register	Read-Write
MDCR_EL2	Hyp Debug Control Register	Read-Write
DBGDTRRX_EL0	Debug Data Transfer Register, Receive	Read-Write
MDRAR_EL1	Monitor Debug ROM Address Register	Read-Write
MDCCSR_EL0	Monitor DCC Status Register	Read-Write
MDCCINT_EL1	Monitor DCC Interrupt Enable Register	Read-Write
DBGBVR0_EL1 — DBGBVR5_EL1	Debug Breakpoint Value Registers	Read-Write
DBGBCR0_EL1 — DBGBCR3_EL1	Debug Breakpoint Control Registers	Read-Write
DBGWVR0_EL1 — DBGWVR3_EL1	Debug Watchpoint Value Registers	Read-Write
DBGWCR0_EL1 — DBGWCR3_EL1	Debug Watchpoint Control Registers	Read-Write

1.5.1.8 AArch32 ID Registers

Table 1-11 describes the AArch32 ID registers group.

Table 1-11 AArch32 ID Registers

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

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 1-12 AArch32 Normal World and Secure World Registers

Name	Description	Type
N_CSSELR, S_CSSELR	Cache Size Selection	Read-Write
N_SCTLR, S_SCTLR	Secure and nonsecure views of System Control Register.	Read-Write
N_ACTLR, S_ACTLR	Secure and nonsecure views of Auxiliary Control Register.	Read-Write
SCR	Secure Configuration	Read-Write
SDER	Secure Debug Enable	Read-Write

Table 1-12 AArch32 Normal World and Secure World Registers (continued)

Name	Description	Type
SDCR	Secure Debug Configuration	Read-Write
N_TTBR0, S_TTBR0	Secure and nonsecure views of Translation Table Base Address Register 0.	Read-Write
N_TTBR1, S_TTBR1	Secure and nonsecure views of Translation Table Base Address Register 1.	Read-Write
N_TTBCR, S_TTBCR	Secure and nonsecure views of Translation Table Base Control Register.	Read-Write
N_DACR, S_DACR	Secure and nonsecure DACR	Read-Write
N_DFSR , S_DFSR	Secure and nonsecure DFSR	Read-Write
N_IFSR, S_IFSR	Secure And Nonsecure IFSR	Read-Write
N_ADFSR, S_ADFSR	Secure And Nonsecure ADFSR	Read-Write
N_AIFSR, S_AIFSR	Secure And Nonsecure AIFSR	Read-Write
N_DFAR, S_DFAR	Secure And Nonsecure Data Fault Address	Read-Write
N_IFAR, S_IFAR	Secure And Nonsecure Instruction Fault Address	Read-Write
N_PAR, S_PAR	Secure And Nonsecure Physical Address	Read-Write
N_PRRR, S_PRRR	Secure And Nonsecure Primary region remap	Read-Write
N_NMRR, S_NMRR	Secure And Nonsecure Normal memory remap	Read-Write
N_MAIR0, S_MAIR0	Secure And Nonsecure Memory Attribute Indirection Register 0	Read-Write
N_MAIR1, S_MAIR1	Secure And Nonsecure Memory Attribute Indirection Register 1	Read-Write
N_AMAIR0, S_AMAIR0	Secure And Nonsecure Auxiliary Memory Attribute Indirection Register 0	Read-Write
N_AMAIR1, S_AMAIR1	Secure And Nonsecure Auxiliary Memory Attribute Indirection Register 1	Read-Write
N_VBAR, S_VBAR	Secure And Nonsecure Vector Base Address	Read-Write
MVBAR	Monitor Vector Base Address	Read-Write
N_FCSEIDR, S_FCSEIDR	Secure And Nonsecure FCSE Process ID	Read-Write
N_CONTEXTIDR, S_CONTEXTIDR	Secure And Nonsecure Context ID	Read-Write
N_TPIDRURW, S_TPIDRURW	Secure And Nonsecure User Read-Write Thread ID	Read-Write
N_TPIDRURO, S_TPIDRURO	User Read-Only Thread ID	Read-Write
N_TPIDRPRW, S_TPIDRPRW	Secure and nonsecure Privileged Only Thread ID	Read-Write

Table 1-12 AArch32 Normal World and Secure World Registers (continued)

Name	Description	Type
N_CNTP_TVAL, S_CNTP_TVAL	Secure And Nonsecure Timer Phy Timer Val	Read-Write
N_CNTP_CTL, S_CNTP_CTL	Secure And Nonsecure Timer Phy Control	Read-Write
N_TTBR0_64, S_TTBR0_64	Secure And Nonsecure Translation Table Base Register 0	Read-Write
N_TTBR1_64, S_TTBR1_64	Secure And Nonsecure Translation Table Base Register 1	Read-Write
N_PAR_64, S_PAR_64	Secure And Nonsecure Physical Address Register	Read-Write
N_CNTP_CVAL, S_CNTP_CVAL	Secure And Nonsecure Timer Phy Compare Val	Read-Write

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

Name	Description	Access
PAR	Physical address	Read-Write

1.5.1.11 AArch32 Performance Registers

Table 1-14 describes the AArch32 Performance Registers group.

Note: The registers PMXEVCNTRn and PMXEVTYPERn 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

Name	Description	Access
PMCR	Performance Monitor Control Register	Read-Write
PMCNTENSET	Count Enable Set	Read-Write
PMCNTENCLR	Count Enable Clear	Read-Write
PMOVSR	Overflow Flag Status Register	Read-Write
PMSELR	Event Counter Selection Register	Read-Write
PMCEID0	Performance Monitor Common Event Identification Register 0	Read-Only
PMCEID1	Performance Monitor Common Event Identification Register 1	Read-Only
PMCCNTR	Cycle count	Read-Write
PMXEVTYPER	Event Type Select	Read-Write
PMXEVCNTR	Event Count	Read-Write
PMUSERENR	User Enable Register	Read-Write
PMINTENSET	Interrupt Enable Set	Read-Write
PMINTENCLR	Interrupt Enable Clear	Read-Write
PMOVSSET	Overflow Flag Status Set	Read-Write

1.5.1.12 AArch64 Performance Registers

Table 1-15 descirbes the AArch64 Performance registers group.

Table 1-15 AArch64 Performance Registers

Name	Description	Access
PMCR_EL0	Performance Monitors Control Register	Read-Write
PMCNTENSET_EL0	Performance Monitors Count Enable Set Register	Read-Write
PMCNTENCLR_EL0	Performance Monitors Count Enable Clear Register	Read-Write
PMSELR_EL0	Performance Monitors Event Counter Selection Register	Read-Write
PMCEID0_EL0	Performance Monitors Common Event Identification Register 0	Read-Only
PMCEID1_EL0	Performance Monitors Common Event Identification Register 1	Read-Only
PMCCNTR_EL0	Performance Monitors Cycle Count Register	Read-Write
PMXEVTYPER	Performance Monitors Selected Event Type Register	Read-Write
PMXEVCNTR_EL0	Performance Monitors Selected Event Count Register	Read-Write
PMUSERENR_EL0	Performance Monitors User Enable Register	Read-Write
PMINTENSET_EL1	Performance Monitors Interrupt Enable Set Register	Read-Write
PMINTENCLR_EL1	Performance Monitors Interrupt Enable Clear Register	Read-Write
PMOVSSET_EL0	Performance Monitors Overflow Flag Status Set Register	Read-Write

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

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

1.5.1.14 AArch64 AdvSIMD Registers

Table 1-17 describes the AArch64 AdvSIMD registers group.

Table 1-17 AArch64 AdvSIMD Registers

Name	Description	Access
S64_0 — S64_31	Advanced SIMD and VFP Extension Single Word Registers	Read-Write
D64_0 — D64_31	Advanced SIMD and VFP Extension Double Word Registers	Read-Write
V0 — V31	SIMD and floating-point registers	Read-Write
FPCR	Floating-Point Control	Read-Write
FPSR	Floating-Point Status	Read-Write

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

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

1.5.1.16 VGIC VCPU Hypervisor Register

This group contains the VGIC VCPU Hypervisor registers.

Table 1-19 VGIC VCPU Hypervisor Registers

Name	Description	Access
GICH_HCR	Hypervisor Control Register	Read-Write
GICH_VTR	VGIC Type Register	Read-Only
GICH_VMCR	Virtual Machine Control Register	Read-Write
GICH_MISR	Maintenance Interface Status Register	Read-Only
GICH_EISR*n*	End of Interrupt Status Registers (0-1)	Read-Only
GICH_ELRSR*n*	Empty List Register Status Registers (0-1)	Read-Only
GICH_APR0	Active Priorities Register	Read-Write
GICH_LR*n*	List Registers (0-4)	Read-Write

1.5.1.17 VGIC VCPU Virtual Registers

This group contains VGIC VCPU Virtual Machine registers.

Table 1-20 VGIC VCPU Virtual Machine Registers

Name	Description	Access
GICV_CTLR	VM Control Register	Read-Write
GICV_PMR	VM Priority Mask Register	Read-Write
GICV_BPR	VM Binary Point Register	Read-Write
GICV_IAR	VM Interrupt Acknowledge Register	Read-Only
GICV_RPR	VM Running Priority Register	Read-Only
GICV_HPPIR	VM Highest Priority Pending Interrupt Register	Read-Only
GICV_ABPR	VM Aliased Binary Point Register	Read-Write
GICV_AIAR	VM Aliased Interrupt Acknowledge Register	Read-Only
GICV_AHPPIR	VM Aliased Highest Priority Pending Interrupt Register	Read-Only
GICV_APR0	VM Active Priority Register	Read-Only[1]
GICV_NSAPR0	Non Secure Active Priority Register 0	Read-Write
GICV_IIDR	VM Cpu Interface Identification Register	Read-Only

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)

Name	Description	Access
ICC_AP0R0_EL1	Active Priority Group0 Register	Read-Write
ICC_PMR_EL1	Priority Mask Register	Read-Write
ICC_BPR0_EL1	Group0 Binary Pointer Register	Read-Only
ICC_HPPIR0_EL1	Group0 Highest Priority Pending Interrupt Register	Read-Only
ICC_HPPIR1_EL1	Group1 Highest Priority Pending Interrupt Register	Read-Only
ICC_IAR0_EL1	Group0 Interrupt Acknowledge Register	Read-Only
ICC_IAR1_EL1	Group1 Interrupt Acknowledge Register	Read-Only
ICC_IGRPEN0_EL1	Group0 Interrupt Group Enable Register	Read-Only
ICC_IGRPEN1_EL1	Group1 Interrupt Group Enable	Read-Only
ICC_IGRPEN1_EL3	EL3 Group1 Interrupt Group Enable	Read-Only
ICC_RPR_EL1	Running Priority Register	Read-Only
ICC_SRE_EL3	EL3 System Register Enable	Read-Write
ICC_CTLR_EL3	EL3 Control Register	Read-Write

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

Name	Description	Access
ICC_CTLR_EL1	Control Register	Read-Write
ICC_SRE_EL2	Hypervisor System Register Enable	Read-Only
ICC_SRE_EL1	System Register Enable	Read-Only

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)

Name	Description	Access
ICH_APR0_EL2	Hypervisor Active Priority Register	Read-Only
ICH_EISR_EL2	End of Interrupt Status Register	Read-Only
ICH_ELSR_EL2	Empty List Register Status Register	Read-Only
ICH_HCR_EL2	Hypervisor Control Register	Read-Only
ICH_LR0_EL2	List Register 0	Read-Only
ICH_LR1_EL2	List Register 1	Read-Only
ICH_LR2_EL2	List Register 2	Read-Only
ICH_LR3_EL2	List Register 3	Read-Only
ICH_MISR_EL2	Maintenance Interrupt Status Register	Read-Only
ICH_VTR_EL2	VGIC Type Register	Read-Only
ICH_VMCR_EL2	Virtual Machine Control Register	Read-Write

1.5.2 Run To Debug Point

The “run to debug point” feature has been added to enhance 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 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 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 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 HTCR 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 model. To display the disassembly view in the SoC Designer Simulator, right-click on the Cortex-A72 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

Stream	Event Name	Comments
Instructions	0x00_SW_INCR	Instruction architecturally executed
	0x08_INST_RETIRED	Instructions architecturally executed
	0x09_EXC_TAKEN	Exception taken
	0x0A_EXC_RETURN	Exception return architecturally executed
	0x0B_CID_WRITE_RETIRED	Counts the number of instructions architecturally executed writing into th ContextID Register
	0x1B_INST_SPEC	Operation speculatively executed
	0x81_EXC_UNDEF	Exception taken: other synchronous
	0x82_EXC_SVC	Exception taken: Supervisor Call
	0x83_EXC_PABORT	Exception taken: Instruction Abort
	0x84_EXC_DABORT	Exception taken: Data Abort or SError
	0x86_EXC_IRQ	Exception taken: IRQ
	0x87_EXC_FIQ	Exception taken: FIQ
	0x88_EXC_SMC	Exception taken: Secure Monitor Call
	0x8A_EXC_HVC	Exception taken: Hypervisor Call
	0x8B_EXC_TRAP_P	Exception taken: Instruction Abort not taken locally
	0x8C_EXC_TRAP_D	Exception taken: Data Abort, or SError not taken :locally
	0x8D_EXC_TRAP_O	Exception taken: Other traps not taken locally

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

Stream	Event Name	Comments
Instructions (continued)	0x8E_EXC_TRAP_I	Exception taken: IRQ not taken locally
	0x8F_EXC_TRAP_F	Exception taken: FIQ not taken locally
	0x90_RC_LD_SPEC	Release consistency instruction
	0x91_RC_ST_SPEC	Release consistency instruction
Pipeline	0x76_PC_WRITE_S	Operation speculatively executed -Software change of the PC
	0x10_BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed
	0x12_BR_PRED	Predictable branch speculatively executed
	0x6C_LDREX_SPEC	Exclusive operation speculatively
	0x6D_STREX_PASS	Exclusive instruction speculatively
	0x6E_STREX_FAIL	Exclusive operation speculatively
	0x70_LD_SPEC	Operation speculatively executed - Load
	0x71_ST_SPEC	Operation speculatively executed - Store
	0x72_LDST_SPEC	Operation speculatively executed - Load or store
	0x73_DP_SPEC	Operation speculatively executed - Integer data processing
	0x74_ASE_SPEC	Operation speculatively executed - Advanced SIMD
	0x75_VFP_SPEC	Operation speculatively executed - VFP
	0x77_CRYPTO_SPE	Operation speculatively executed, crypto data processing
	0x78_BR_IMMED_S	Branch speculatively executed - Immediate branch
	0x79_BR_RETURN_	Branch speculatively executed - Procedure return
	0x7A_BR_INDIREC	Branch speculatively executed - Indirect branch
	0x7C_ISB_SPEC	Barrier speculatively executed - ISB
	0x7D_DSB_SPEC	Barrier speculatively executed - DSB
	0x7E_DMB_SPEC	Barrier speculatively executed - DMB
I-Cache	0x01_L1I_CACHE_REFILL	Level 1 instruction cache refill
	0x14_L1I_CACHE	Level 1 instruction cache access

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

Stream	Event Name	Comments
D-Cache	0x03_L1D_CACHE_MISS	Data cache miss
	0x04_L1D_CACHE_ACCESS	Data read or write operation that causes a cache access at (at least) the lowest level of data or unified cache
	0x15_L1D_CACHE_WB	Level 1 data cache write back
	0x40_L1D_CACHE_LD	Level 1 data cache access - Read
	0x41_L1D_CACHE_ST	Level 1 data cache access - Write
	0x42_L1D_CACHE_REFILL_LD	Level 1 data cache refill - Read
	0x43_L1D_CACHE_REFILL_ST	Level 1 data cache refill - Write
	0x46_L1D_CACHE_WB_VICTIM	Level 1 data cache Write-back - Victim
	0x47_L1D_CACHE_WB_CLEAN	Level 1 data cache Write-back - Cleaning
	0x48_L1D_CACHE_INVAL	Level 1 data cache invalidate
	0x16_L2D_CACHE	Level 2 data cache acces
	0x17_L2D_CACHE_REFILL	Level 2 data cache refill
	0x18_L2D_CACHE_WB	Level 2 data cache Write-Back
	0x50_L2D_CACHE_LD	Level 2 data cache access - Read
	0x51_L2D_CACHE_ST	Level 2 data cache access - Write
	0x52_L2D_CACHE_REFILL_LD	Level 2 data cache refill - Read
	0x53_L2D_CACHE_REFILL_ST	Level 2 data cache refill - Write
	0x56_L2D_CACHE_WB_VICTIM	Level 2 data cache Write-back - Victim
	0x57_L2D_CACHE_WB_CLEAN	Level 2 data cache Write-back - Cleaning
	0x58_L2D_CACHE_INVAL	Level 2 data cache invalidate
Cycle	0x11_CPU_CYCLES	Cycle
	0x1D_BUS_CYCLES	Bus cycle

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

Stream	Event Name	Comments
Microarchi-tecture	0x02_L1I_TLB_REFIL	Instruction TLB refill
	0x05_L1D_TLB_REFILL	Data TLB refill
	0x13_MEM_ACCESS	Data memory access
	0x19_BUS_ACCESS	Bus access
	0x1A_MEMORY_ERR	Local memory error
	0x1C_TTBR_WRITE	Instruction architecturally executed
	0x1E_CHAIN	Performance counter chain mode
	0x4C_L1D_TLB_REFILL_LD	Level 1 data TLB refill - Read
	0x4D_L1D_TLB_REFILL_ST	Level 1 data TLB refill - Write
	0x60_BUS_ACCESS_LD	Bus access - Read
	0x61_BUS_ACCESS_ST	Bus access - Write
	0x62_BUS_ACCESS_SHARED	Bus access - Normal
	0x63_BUS_ACCESS_NOT_SHARED	Bus access - Not normal
	0x64_BUS_ACCESS_NORMAL	Bus access - Normal
	0x65_BUS_ACCESS_PERIPH	Bus access - Peripheral
	0x66_MEM_ACCESS_LD	Data memory access - Read
	0x67_MEM_ACCESS_ST	Data memory access - Write
	0x68_UNALIGNED_LD_SPEC	Unaligned access - Read
	0x69_UNALIGNED_ST_SPEC	Unaligned access - Write
	0x6A_UNALIGNED_LDST_SPEC	Unaligned access
IMPL_ BranchPre-diction	0x100_BR_PRED_BTB	Number of uBTB or BTB predicted branches
	0x101_BR_PRED_STATIC	Number of predictable branches predicted by static predictor
	0x102_BR_PRED_BTB_INDIR	Number of predictbale branches predicted by BTB/uBTB that are indirect
	0x103_BR_PRED_BTB_COND	Number of uBTB/BTB predicted conditional branches
	0x104_BR_COND_SPEC	Number of preditable branches that are conditional
	0x105_BR_TAKEN_SPEC	Number of predictable branches that are taken
	0x106_BR_MIS_PRED_BTB_DIR	Number of directional mispredicted uBTB/BTB predicted branches
	0x107_BR_MIS_PRED_BTB	Number of mispredicted branches predicted by BTB/uBTB

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

Stream	Event Name	Comments
IMPL_ BranchPre-diction (continued)	0x108_BR_MIS_PRED_BTB_INDIR	Number of mispredicted indirect branch predicted by BTB/uBTB
	0x109_BR_MIS_PRED_RETURN	Mispredicted procedure returns
	0x10A_BR_MIS_PRED_INDIRECT	Number of mispredicted indirect branches
	0x10B_BR_UNPRED_NT_OK	Number of unpredicted NT branches
IMPL_ QueuesRena-meMisc	0x10C_FETCHQ_EMPTY_CYCLES	Number of cycles that the fetch queue contains no instructions
	0x110_SINGLE_UOP_SEQUENCED	cycle count: single uop decoded/ sequenced this cycle.
	0x111_RENAMEQ_EMPTY_CYCLES	cycle count: Rename queue empty
	0x112_RENAME_FLAG_STALL	cycle count: rename stall due to flags
	0x113_RENAME_REG_STALL	cycle count: rename stall due to regis-ters
	0x118_UOP_COUNT	uop count: total number of micro-ops renamed.
	0x120_DISPQ_EMPTY_CYCLES	cycle count: Dispatch queue empty
	0x121_ISSUEQ_FULL_BX_STALL	Dispatch stall due to issue queue full BX
	0x122_ISSUEQ_FULL_CX_STALL	Dispatch stall due to issue queue full CX1 or CX2
	0x123_ISSUEQ_FULL_LS_STALL	Dispatch stall due to issue queue full LS
	0x124_ISSUEQ_FULL_MX_STALL	Dispatch stall due to issue qeueu full MX
	0x125_ISSUEQ_FULL_SP_STALL	Dispatch stall due to issue queue full SP
	0x126_ISSUEQ_FULL_SX_STALL	Dispatch stall due to issue queue full SX1 or SX2
	0x127_FLUSH_BAD_BRANCH	Bad branch flush
	0x128_FLUSH_MEM	MEM NUKE Flush (i.e ECC flush + RAR flush)
	0x129_FLUSH_RARE	Rare flushes
	0x12A_FLUSH_SWDW	SWDW_NUKE

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

Stream	Event Name	Comments
IMPL_ LoadStore-Cache	0x12B_FLUSH_NEON_COND	Neon conditional flush
	0x12C_DISP_SWDW_STALL	Dispatch stall due to single word producer double word consumer issue
	0x140_GRP_COMMIT_COUNT	Number of groups tally
	0x141_ETM_EXTOUT_0	ETM EXTOUT 0
	0x142_ETM_EXTOUT_1	ETM EXTOUT 1
	0x143_ETM_EXTOUT_2	ETM EXTOUT 2
	0x144_ETM_EXTOUT_3	ETM EXTOUT 3
	0x150_LS_LD_RESTART_CNT	Load Store Load Restart Count
	0x151_LS_LD_ORDER_HAZARD	Loadstore Load Pipe Hazard
	0x152_LS_ST_ORDER_HAZARD	Loadstore Store Pipe Hazard
	0x153_LS_PM_L1_PF_REQ_GEN_PAGE	L1 prefetch request generated in page mode
	0x154_LS_PM_L1_PF_REQ_GEN_STRIDE	L1 prefetch request generated by the load stride prefetcher
	0x155_LS_PM_L2_PF_REQ_GEN_LS_LD	L2 prefetch request generated by a load instruction
	0x156_LS_PM_L2_PF_REQ_GEN_LS_ST	L2 prefetch request generated by a store instruction
	0x157_LS_PM_PF_TRAIN_TABLE_ALLOC	New train table entry allocated by a load
	0x158_LS_PF_PHT_LOOKUP	Lookup generated for the PF pattern history ram
	0x159_LS_PM_PF_GEN_TABLE_ALLOC	New generation table allocation for L2 prefetch
	0x15A_LS_PM_PF_GEN_TABLE_ALLOC_PEND	New generation table allocation for L2 prefetch replacing an entry that still had pending prefetches
	0x15B_LS_PMU_L1_DTLB_REFILL_PF	L1 dtlb refill's that are initiated by PF
	0x15C_LS_PMU_L1_DCACHE_REFILL_PF	L1 dcache refill's that are initiated by PF
	0x160_BUS_TRANS_RD	Bus read transaction
	0x161_BUS_TRANS_WR	Bus write transaction
	0x162_BUS_ACCESS_SNOOP	Bus access - snoop
	0x163_L2_PF_REQ_BUS_TRANS	L2 prefetch initiated by bus transaction
	0x164_L2_ACP_MAS_RD_DATA	ACP based AXI master port read data
	0x165_L2_ACP_MAS_WR_DATA	ACP based AXI master port write data

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

Stream	Event Name	Comments
IMPL_ LoadStore-Cache (continued)	0x166_L2_PF_REQ_GENERATED_IF	L2 Prefetch request generated due to an instr access
	0x167_L2_PF_REQ_GENERATED_TW	L2 Prefetch request generated due to a TBW desc fetch
	0x168_L2_PF_REQ_BUS_TRANS_LD	Prefetch initiated by a load access issued on the bus
	0x169_L2_PF_REQ_BUS_TRANS_ST	Prefetch initiated by a store access issued on the bus
	0x16A_L2_FEQ_STALL	L2 Stalled due to no FEQ's available
	0x16B_L2_CPU_INCOMING_CCB_XFER	L2 CCB transfer from another cpu's L1D cache to this cpu
	0x170_L2_TBW_DESC_ACCESS	L2 TBW descriptor read access
	0x171_L2_TBW_IPA_PA_CACHE_ACCESS	L2 IPA-PA cache accesses.
	0x172_L2_TBW_IPA_PA_CACHE_HIT	L2 IPA-PA cache hit
	0x173_L2_TBW_S1_L2_PA_CACHE_HIT	L2 S1 L2 PA Cache hit
	0x174_L2_TBW_S1_WLK_CACHE_HIT	L2 S1 wlk cache hit
	0x175_L2_TLB_ACCESS_ARRAY	L2 TLB No of accesses made to the array during lookup.
	0x176_L2_TLB_MAINT_OPS	L2 TLB shootdowns
	0x177_L2_TLB_ACCESS	L2 TLB accesses (lookups)
	0x178_L2_TLB_ACCESS_PF	L2 TLB accesses for table prefetches
	0x179_L2_TLB_REFILL	L2 TLB refills
	0x17A_L2_TLB_REFILL_INST	L2 TLB refills for inst fetches
	0x17B_L2_TLB_REFILL_LDST	L2 TLB refills for loads and stores
	0x17C_L2_TLB_REFILL_PF	L2 TLB refills for tablewalk prefetches
	0x17D_L2_PF_REQ_BUS_TRANS_IF	Prefetch initiated by an instr access issued on the bus
	0x17E_PF_REQ_BUS_TRANS_TW	Prefetch initiated by a TBW desc fetch issued on the bus
	0x17F_PF_FEQ_LATE	L2 Prefetch request late (demand hit outstanding PF request in FEQ)
	0x180_L2_LD_PF_REQ_HIT_L2	Prefetch initiated by a load instruction hit in the L2
	0x181_L2_ST_PF_REQ_HIT_L2	Prefetch initiated by a store instruction hit in the L2

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

Stream	Event Name	Comments
IMPL_ LoadStore-Cache (continued)	0x182_L2_IF_PF_REQ_HIT_L2	Prefetch initiated by an instruction fetch hit in the L2
	0x183_L2_TW_PF_REQ_HIT_L2	Prefetch initiated by a TBW desc fetch hit in the L2
	0x184_L2_PF_REQ_GENERATED	L2 prefetch generated

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.

Third Party Software Acknowledgement

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

- **ELF (Executable and Linking Format) Tool Chain Product**

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.