Cortex®-A53 Cycle Model

Version 9.2.0

User Guide

Non-Confidential

ARM®

ARM 101013_0902_00 (ID050417)

Cortex-A53 Cycle Model
User Guide

Release Information

The following changes have been made to this document.

Change History

Issue	Date	Confidentiality	Change
0902-00-01	May 2017	Non-Confidential	Release with 9.2.

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

About This Guide

This guide provides all the information needed to configure and use the Cortex-A53 multi-processor Cycle Model in SoC Designer.

Audience

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

- SoC Designer
- Hardware design verification
- Verilog or SystemVerilog programming language

Conventions

This guide uses the following conventions:

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

The following publication provides information that relates directly to SoC Designer:

- *SoC Designer User Guide*

The following publications provide reference information about ARM® products:

- *Cortex-A53 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.
Cycle Model	A software object created by the Cycle Model Studio (or Carbon compiler) from an RTL design. The Cycle Model contains a cycle- and register-accurate Cycle Model of the hardware design.
Cycle Model Studio	Graphical tool for generating, validating, and executing hardware-accurate software models. It creates a Cycle Model, and it also takes a Cycle Model as input and generates a component that can be used in SoC Designer, Platform Architect, or Accellera SystemC for simulation.
CASI	*ESL API Simulation Interface*, is based on the SystemC communication library and manages the interconnection of components and communication between components.
CADI	*ESL API Debug Interface*, enables reading and writing memory and register values and also provides the interface to external debuggers.
CAPI	*ESL API Profiling Interface*, enables collecting historical data from a component and displaying the results in various formats.
Component	Building blocks used to create simulated systems. Components are connected together with unidirectional transaction-level or signal-level connections.
ESL	*Electronic System Level*. A type of design and verification methodology that models the behavior of an entire system using a high-level language such as C or C++.
HDL	*Hardware Description Language*. A language for formal description of electronic circuits, for example, Verilog.
RTL	*Register Transfer Level*. A high-level hardware description language (HDL) for defining digital circuits.
SoC Designer	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. It contains the following sections:

1.1 Cortex-A53 Functionality

In the multiprocessor configuration, up to four Cortex-A53 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.

The Cortex-A53 multiprocessor supports:

- Up to four Cortex-A53 processors.
- The AArch32 and AArch64 versions of the ARMv8-A architecture instruction set.
- In-order pipeline with symmetric dual-issue of most instructions.
- Harvard Level 1 (L1) memory system with a Memory Management Unit (MMU).
- Level 2 (L2) memory system providing multiprocessor memory coherency, optionally including an L2 cache.
- Per-core Global Interrupt Controllers (GICs).
- Variable ICache/DCache sizes.
- A generic 64-bit timer per processor.
- Support for AMBA 5 CHI, AMBA 4.0 AXI Coherency Extension (ACE) and AXI3 master ports.
- VFP Floating Point.
- Neon Advanced SIMD.
- Cryptography Engine.
- ACP Transactors.
- Semihosting.
- Hardware profiling supported for all CPUs in a system.

This section provides a summary of the functionality of the Cycle Model compared to that of the hardware, and the performance and accuracy of the Cycle Model.

- Implemented Hardware Features
- Unsupported Hardware Features
- Features Additional to the Hardware

1.1.1 Implemented Hardware Features

Most hardware features have been implemented. Some functionality and register differences are listed in the next section.

See the *ARM Cortex-A53 Technical Reference Manual* for more information.

1.1.2 Unsupported Hardware Features

The following features of the Cortex-A53 hardware are *not* implemented in the Cycle Model:

- SCU cache protection.
- CPU cache protection.
- The full set of registers. Only the registers listed in "Register Information" on page 25 are available to be read/written via debug transactions — for example, in the SoC Designer Registers window.

  The functionality of all registers, however, does exist and can be accessed by software running on the virtual platform.

1.1.3 Features Additional to the Hardware

The following features that are implemented in the Cortex-A53 Cycle Model do not exist in the Cortex-A53 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 19).
- 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 45 for more information.
- Waveform dumping using the waveform-related parameters described in Table 1-3 on page 19.
- Support for viewing memory spaces (see "Memory Information" on page 45).
- Support for viewing disassembly data (see "Disassembly View" on page 45).

1.2 Adding and Configuring the SoC Designer Component

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

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

1.2.1 SoC Designer Component Files

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

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

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

The provided component files are listed in Table 1-1 below:

Table 1-1 SoC Designer Component Files

Platform	File	Description
Linux	maxlib.lib<*model_name*>.conf	SoC Designer configuration file
	lib<*component_name*>.mx.so	SoC Designer component runtime file
	lib<*component_name*>.mx_DBG.so	SoC Designer component debug file
	libA53_save_arch_handler.a	Library file for Architectural cache/restore support.
Windows	maxlib.lib<*model_name*>.windows.conf	SoC Designer configuration file
	lib<*component_name*>.mx.dll	SoC Designer component runtime file
	lib<*component_name*>.mx_DBG.dll	SoC Designer component debug file
	A53_save_arch_handler.lib	Library file for Architectural cache/restore support.

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

1.2.2 Adding the Cycle Model to the Component Library

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

For more information on SoC Designer Canvas, see the *SoC Designer 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 16).

1.3 ESL Ports

This section describes the differences between the pins listed in the *ARM Cortex-A53 Technical Reference Manual* (TRM) and those on the Cortex-A53 Cycle Model. For each pin listed in the TRM, the Cycle Model includes a corresponding port of the same name, *except* as detailed in the following sections:

- Available Component ESL Ports — Describes ports that have been added to the Cycle Model, such as clocks and resets required by SoC Designer, 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.
- Unsupported Pins — Describes hardware pins that are unsupported in the Cycle Model.
- Port Name and Polarity Differences — Describes ports that differ in name and polarity from their RTL counterparts.

1.3.1 Available Component ESL Ports

Table 1-2 describes the Cortex-A53 transactors and special pins that are exposed in SoC Designer Plus. See the *ARM Cortex-A53 Technical Reference Manual* for more information.

Table 1-2 ESL Component Ports

ESL Port	Description	Type
CHI_RNF_CHI_Master	CHI Master S2T when configured in CHI mode.	Transactor Master
ACE_Master	ACE Master S2T when configured in ACE mode.	Transactor Master
axi_m	AXI3 S2T when configured in AXI3 mode.	Transactor Master
Debug_APB	APB3 Transactor Slave.	Transactor Slave
CLKIN	Main clock of the Cortex-A53 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
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 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

Certain pins in the Cortex-A53 hardware have been either tied or disconnected in the Cycle Model for performance reasons. These include:

- DBGEN
- NIDEN
- SPIDEN
- SPNIDEN
- DFTSE
- nMBISTRESET
- MBISTREQ
- CIHSBYPASS
- CISBYPASS
- CTICHIN
- CTICHOUTACK
- CTIIRQACK

In AXI3 mode, the following ACE-specific signals are tied:

- SYSBARDISABLE
- ACSNOOPM
- ACADDRM
- ACPROTM
- ACVALIDM
- ACREADYM
- ARBARM
- ARDOMAINM
- ARSNOOPM

- AWBARM
- AWDOMAINM
- AWSNOOPM
- CRVALIDM
- CRREADYM
- CRRESPM
- CDVALIDM
- CDREADYM
- CDDATAM
- CDLASTM
- RACKM
- WACKM

In ACE mode, the following AXI3-specific signal is tied:

- WIDM

1.3.4 Unsupported Pins

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

- BROADCASTINNER
- BROADCASTOUTER
- BROADCASTCACHEMAINT

Refer to the *Cortex-A53 Technical Reference Manual* for details on how the broadcast pins are driven.

1.3.5 Port Name and Polarity Differences

The following ports are named differently than their counterparts in the RTL; ports in the RTL that begin with the prefix nCNT* are called CNT* in the Cycle Model:

- nCNTHPIRQ
- nCNTPNSIRQ
- nCNTPSIRQ
- nCNTVIRQ

In addition, these ports are Active Low in the RTL, but Active High in the Cortex-A53 Cycle Model. You can use the Cycle Model parameter negLogic to invert the polarity of these ports if necessary.

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
AA64nAA32[0, 1, 2, 3]	Register width state; CPU number is appended.	bool: 0 - AArch32 1 - AArch64	0	Init
ACE_Master Enable Debug Messages	Enables ACE_Master port debug.	true, false	false	Runtime
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
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 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 time.	true, false	true	Init

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
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
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
CFGTE*n*	Default exception handling state (ARM/Thumb). Per-core value of CFGTE; automatically kept in sync with CFGTE.	bool	false	Init
CLUSTERIDAFF1	Value read in the Cluster ID Affinity Level-1 field, bits[15:8], of the Multiprocessor 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 configurations. Supported values: NonCoherentNoL3[1] NonCoherentWithL3[1] OuterCoherentNoL3 OuterCoherentWithL3 InnerCoherentNoL3 InnerCoherentWithL3	String	OuterCoherentWithL3	Init

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
CP15DISABLE	Disable write access to some Secure CP15 registers. This is the aggregated bus [NUM_CPUS-1:0].	0, 1	0	Runtime
CP15DISABLE*n*	One-bit parameter. Disable write access to DBGL1RSTDISABLE.	0, 1	0	Runtime
DBGL1RSTDIS-ABLE	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
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 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.	0, 1	0	Init

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
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
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	If set to 1, inverts the values of the following ports (these ports are Active High by default): • fiq • irq • virq • vfiq • sei • vsei • rei • CNTHPIRQ • CNTPNSIRQ • CNTPSIRQ • CNTVIRQ	bool	false	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
NODE ID	Cortex-A53 CHI Node Identifier	Cortex-A53 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
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
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 - Barriers are broadcast onto system bus, this requires an AMBA4 interconnect. 1 - Barriers are not broadcast onto the system bus. This is compatible with an AXI3 interconnect. This pin is sampled only during reset of the Cortex-A53 MPCore multiprocessor.	0, 1	0	Init
TSVALUEB	ETM signal. Timestamp in binary encoding.	[0-0xFFFFFFFFFFFFFFFF]	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

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
VINITHI*n*	Use high vector addresses. Per-core value of VINITHI; automatically kept in sync with VINITHI.	bool	false	Init
Waveform File[2]	Name of the waveform file.	*string*	arm_cm_CORTEXA53.vcd or arm_cm_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. When configured with the ACE main bus interface, this setting uses the ACE-Lite protocol.
2. When enabled, SoC Designer writes accumulated waveforms to the waveform file in the following situations: when the waveform buffer fills, when validation is paused and when validation finishes, and at the end of each validation run.

1.5 Debug Features

The Cortex-A53 Cycle Model has a debug interface (CADI) that allows the user to view, manipulate, and control the registers and memory. You can access a view in SoC Designer 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
- GICv3 CPU Interface Registers (Memory-Mapped)
- GICv3 Hypervisor Registers (Memory-Mapped)
- GICv3 Virtual Registers (Memory-Mapped)
- GICv3 CPU/Virtual Registers (System Registers)
- GICv3 Hypervisor Registers (System Registers)

For detailed descriptions of these registers, refer to the *ARM® Cortex™-A53 MPCore Technical Reference Manual* or the *ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile*.

Note: Registers are accurate only at debuggable points. While SoC Designer 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

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	
SP	Stack Pointer register	
ELR	Exception Link register	
CPSR	Current Program State register	
SP_EL0 — SP_EL3	Stack Pointer, EL0 - EL3	
ELR_EL1 — ELR_EL3	Exception Link Registers, EL1 - EL3	
SPSR_EL1 — SPSR_EL3	Saved Program Status Registers, EL1 - EL3	
SPSR	Saved Program Status Register for current mode	

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
PRRR	Primary region remap	Read-Write
NMRR	Normal memory remap	Read-Write
MAIR0	Memory Attribute Indirection Register 0	Read-Write

Table 1-6 AArch32 Control Registers (continued)

Name	Description	Type
MAIR1	Memory Attribute Indirection Register 1	Read-Write
AMAIR0	Auxiliary Memory Attribute Indirection Register 0	Read-Write
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

Table 1-6 AArch32 Control Registers (continued)

Name	Description	Type
HDFAR	Hyp Data Fault Address	Read-Write
HIFAR	Hyp Inst Fault Address	Read-Write
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
CSSELR	Cache Size Select Register	Read-Write
CPUACTLR	CPU Auxiliary Control 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
CCSIDR_EL1	Cache Size ID	Read-Only
CLIDR_EL1	Cache Level ID	Read-Write
AIDR_EL1	Auxiliary ID	Read-Only
CSSELR_EL1	Cache Size Selection	Read-Write
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

Table 1-7 AArch64 System Registers (continued)

Name	Description	Type
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
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

Table 1-7 AArch64 System Registers (continued)

Name	Description	Type
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
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
TTBR0_EL3	Translation Table Base Address Register 0, EL3	Read-Write
TCR_EL3	Translation Control Register, EL3	Read-Write
AFSR0_EL3	Auxiliary Fault Status Register 0, EL3	Read-Write
AFSR1_EL3	Auxiliary Fault Status Register 1, EL3	Read-Write
ESR_EL3	Exception Syndrome Register, EL3	Read-Write
FAR_EL3	Fault Address Register, EL3	Read-Write
MAIR_EL3	Memory Attribute Indirection Register, EL3	Read-Write
AMAIR_EL3	Auxiliary Memory Attribute Indirection Register, EL3	Read-Write
VBAR_EL3	Vector Base Address Register, EL3	Read-Write
TPIDR_EL3	Thread Pointer/ID Register, EL3	Read-Write
RVBAR_EL3	Reset Vector Base Address Register, EL3	Read-Only
RMR_EL3	Reset Management Register	Read-Write
CNTFRQ_EL0	Counter-timer Frequency register, EL0	Read-Write
CNTPCT_EL0	Counter-timer Physical Count register, EL0	Read-Write
CNTVCT_EL0	Counter-timer Virtual Count register	Read-Write
CNTVOFF_EL2	Counter-timer Virtual Offset register	Read-Write
CNTKCTL_EL1	Counter-timer Kernel Control register	Read-Write

Table 1-7 AArch64 System Registers (continued)

Name	Description	Type
CNTHCTL_EL2	Counter-timer Hypervisor Control register	Read-Write
CNTP_TVAL_EL0	Counter-timer Physical Timer TimerValue register	Read-Write
CNTP_CTL_EL0	Counter-timer Physical Timer Control register	Read-Write
CNTP_CVAL_EL0	Counter-timer Physical Timer Compare Value register	Read-Write
CNTV_TVAL_EL0	Counter-timer Virtual Timer Timer Value register	Read-Write
CNTV_CTL_EL0	Counter-timer Virtual Timer Control register	Read-Write
CNTV_CVAL_EL0	Counter-timer Virtual Timer Compare Value register	Read-Write
CNTHP_TVAL_EL2	Counter-timer Hypervisor Physical Timer TimerValue register	Read-Write
CNTHP_CTL_EL2	Counter-timer Hypervisor Physical Timer Control register	Read-Write
CNTHP_CVAL_EL2	Counter-timer Hypervisor Physical Timer CompareValue register	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
DBGVCR	Debug Vector Catch Register	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

Table 1-8 AArch32 Debug Registers (continued)

Name	Description	Access
DBGWVR0 - DBGWVR3	Debug Watchpoint Value Register 0 - Debug Watchpoint Value Register 3	Read-Write
DBGWCR0 - DBGWCR3	Debug Watchpoint Control Registers	Read-Write
DBGDRAR	Debug ROM Address Register	Read-Write
DBGDSAR	Debug Self Address Register RES0	Read-Only

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
EDVIDSR	External Debug Virtual Context Sample Register	Read-Only
DBGBXVR0 - DBGBXVR5	Breakpoint Extended Value Registers	Read-Write
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

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_ISAR7	Instruction Features 0 - 6	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
PMXEVCNTR0 — PMXEVCNTR5	Event Count Register 0 — Event Count Register 5	Read-Write
PMXEVTYPER0 — PMXEVTYPER5	Event Type Select Register 0 — Event Type Select Register 0	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
PMOVSR_EL0	Performance Monitors Overflow Flag Status 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
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
PMEVCNTR0_EL0	Performance Monitors Event Count Registers	Read-Write
PMEVTYPER0_EL0 — PMEVTYPER5_EL0	Performance Monitors Event Type Registers	Read-Write
PMEVCNTR1_EL0 — PMEVCNTR5_EL0	Performance Monitor Event Count Registers 1 - 5	Read-Write
PMXEVCNTR_EL0	Performance Monitors Cycle Count Register	Read/Write
PMXEVTYPER_EL0	Performance Monitors Selected Event Type and Filter Register	Read/Write

1.5.1.13 AArch32 VFP/Neon Registers

Table 1-16 describes the AArch32 VFP/Neon registers group.

Table 1-16 AArch64 VFP/Neon Registers

Name	Description	Access
S0_AARCH32 — S31_AARCH32	Advanced SIMD and VFP Extension Single Word Registers	Read-Write
D0_AARCH32 — D31_AARCH32	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 — MVFR2	Media and VFP Feature Register 0 - Media and VFP Feature Register 2	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
S0 — S31	Advanced SIMD and VFP Extension Single Word Registers	Read-Write
D0 — D31	Advanced SIMD and VFP Extension Double Word Registers	Read-Write
V0 — V15	SIMD and floating-point registers	Read-Write
FPCR	Floating-Point Control	Read-Write
FPSR	Floating-Point Status	Read-Write

1.5.1.15 GICv3 CPU Interface Registers (Memory-Mapped)

Table 1-18 describes the CPU interface registers group.

Note: Access shown is in accordance with the Cortex-A53 TRM. The CADI access of the Cycle Model implementation may differ; CADI access to the ICC an dICH registers more closely matches the TRM,

Table 1-18 CPU Interface Registers (Memory-Mapped)

Name	Description	Access
GICC_CTLR_NS	CPU Interface Control Register, non-secore	Read-Write
GICC_CTLR_S	CPU Interface Control Register, secure	Read-Write
GICC_NSAPRO	Non-secure Active Priority Register	Read-Write
GICC_APR0_NS	Active Priority Register, non-secure	Read-Write
GICC_APR0_S	Active Priority Register, secure	Read-Write
GICC_BPR	Binary Point Register	Read-Write
GICC_PMR	Interrupt Priority Mask Register	Read-Write
GICC_HPPIR	Highest Priority Pending Interrupt Register	Read Only
GICC_IAR	Interrupt Acknowledge Register	Read Only
GICC_ABPR	Aliased Binary Point Register	Read-Write
GICC_AEOIR	Aliased End of Interrupt Register	Write Only
GICC_AHPPIR	Aliased Highest Priority Pending Interrupt Register	Read Only
GICC_AIAR	Aliased Interrupt Acknowledge Register	Read Only
GICC_DIR	Deactivate Interrupt Register	Read-Write
GICC_EOIR	End Of Interrupt Register	Write Only
GICC_IIDR	CPU Interface Identification Register	Read Only
GICC_RPR	Running Priority Register	Read Only

1.5.1.16 GICv3 Hypervisor Registers (Memory-Mapped)

Table 1-19 describes the memory-mapped Hypervisor registers group.

Note: Access shown is in accordance with the Cortex-A53 TRM. The CADI access of the Cycle Model implementation may differ; CADI access to the ICC an dICH registers more closely matches the TRM

Table 1-19 Hypervisor Registers (Memory-Mapped)

Name	Description	Access
GICH_APR0	Active Priorities Register	Read-Write
GICH_EISR0	End of Interrupt Status Registers	Read Only
GICH_ELSR0	Empty List Register Status Registers	Read Only
GICH_HCR	Hypervisor Control Register	Read-Write
GICH_LR[0-3]	List Registers	Read-Write
GICH_MISR	Maintenance Interrupt Status Register	Read Only
GICH_VMCR	Virtual Machine Control Register	Read-Write
GICH_VTR	VGIC Type Register	Read Only

1.5.1.17 GICv3 Virtual Registers (Memory-Mapped)

Table 1-20 describes the memory-mapped Virtual registers group.

Note: Access shown is in accordance with the Cortex-A53 TRM. The CADI access of the Cycle Model implementation may differ; CADI access to the ICC an dICH registers more closely matches the TRM

Table 1-20 Virtual Registers

Name	Description	Access
GICV_AEOIR	VM Aliased End of Interrupt Register	Write Only
GICV_AHPPIR	VM Aliased Highest Priority Pending Interrupt Register	Read Only
GICV_AIAR	VM Aliased Interrupt Acknowledge Register	Read-Only
GICV_APR0	VM Active Priority Register	Read Write
GICV_BPR	Binary Point Register	Read Write
GICV_CTLR	Control Register	Read Write
GICV_DIR	VM Deactivate Interrupt Register	Write Only
GICV_EOIR	VM End Of Interrupt Register	Write Only
GICV_HPPIR	VM Highest Priority Pending Interrupt Register	Read-Only
GICV_IAR	-VM Interrupt Acknowledge Register	Read-Only
GICV_IIDR	VM CPU Interface Identification Register	Read-Only

Table 1-20 Virtual Registers (continued)

Name	Description	Access
GICV_PMR	Priority Mask Register	Read Write
GICV_RPR	VM Running Priority Register	Read-Only

1.5.1.18 GICv3 CPU/Virtual Registers (System Registers)

Table 1-21 describes the CPU/Virtual registers group.

Table 1-21 CPU/Virtual Registers

Name	Description	Access
ICC_AP0P0_EL1	Active Priorities 0 Register 0	Read-Write
ICC_BPR0_EL1	Binary Point Register 0	Read Write
ICC_CTLR_EL3	Interrupt Control Register for EL3	Read Write
ICC_CTLR_EL1	Interrupt Control Register for EL1	Read Write
ICC_HPPIR0_EL1	Highest Priority Pending Interrupt Register 0	Read-Only
ICC_HPPIR1_EL1	Highest Priority Pending Interrupt Register 1	Read-Only
ICC_IAR0_EL1	Interrupt Acknowledge Register 0	Read-Only
ICC_IAR1_EL1	Interrupt Acknowledge Register 1	Read-Only
ICC_IGRPEN0_EL1	Interrupt Group Enable Register 0	Read Write
ICC_IGRPEN1_EL1	Interrupt Group Enable Register 1	Read Write
ICC_PMR_EL1	Priority Mask Register	Read-Write
ICC_RPR_EL1	Running Priority Register	Read-Only
ICC_SRE_EL[1-3]	System Register Enable Register for EL[1-3]	Read Write

1.5.1.19 GICv3 Hypervisor Registers (System Registers)

Table 1-22 describes the System Hypervisor registers group.

Table 1-22 Interface Hypervisor Registers (System Registers)

Name	Description	Access
ICH_AP0R0_EL2	Interrupt Controller Hyp Active Priorities Register (0,0)	Read Write
ICH_AP1R0_EL2	Interrupt Controller Hyp Active Priorities Register (1,0)	Read Write
ICH_EISR_EL2	Interrupt Controller End of Interrupt Status Register	Read-Only
ICH_ELSR_EL2	Interrupt Controller Empty List Register Status Register	Read-Only
ICH_HCR_EL2	Interrupt Controller Hyp Control Register	Read Write
ICH_LR0_EL2	Interrupt Controller List Register 0	Read Write

Table 1-22 Interface Hypervisor Registers (System Registers) (continued)

Name	Description	Access
ICH_LR1_EL2	Interrupt Controller List Register 1	Read Write
ICH_LR2_EL2	Interrupt Controller List Register 2	Read Write
ICH_LR3_EL2	Interrupt Controller List Register 3	Read Write
ICH_MISR_EL2	Interrupt Controller Maintenance Interrupt State Register	Read-Only
ICH_VMCR_EL2	Interrupt Controller Virtual Machine Control Register	Read Write
ICH_VTR_EL2	Interrupt Controller VGIC Typ	Read-Only

1.5.2 Run To Debug Point

The “run to debug point” feature has been added to enhance Cycle Model debugging. The Cortex-A53 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, 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 of the Cortex-A53 Cycle Model represents a different view of memory using a page table. The Cortex-A53 processor supports two memory spaces, which are selectable using the Space: pulldown menu in the Memory view (see the *SoC Designer User Guide* for more information):

- **Physical Memory (secure)** — Main memory space visible from the Memory view.
- **Secure Monitor** — CPU virtual memory visible from the Memory view.
- **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).

This component supports full system coherent memory views. Refer to the *SoC Designer User Guide* for details.

1.5.4 Disassembly View

SoC Designer Simulator supports a disassembly view of a program running on the Cortex-A53 Cycle Model. To display the disassembly view in the SoC Designer Simulator, right-click on the Cortex-A53 Cycle Model and select **View Disassembly…** from the context menu. Refer to the *SoC Designer 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-A53 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-A53 Profiling Events

Stream	Event Name	Comments
Instructions	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 the ContextID Register
	0x06_LD_RETIRED	Data read architecturally executed
	0x07_ST_RETIRED	Data write architecturally executed
Pipeline	0x0C_PC_WRITE_RETIRED	Instruction speculatively executed - Software change of the PC
	0x0D_BR_IMMED_RETIRED	Immediate branch architecturally executed
	0x0E_BR_RETURN_RETIRED	Procedure return (other than exception returns) architecturally executed
	0xOF_UNALIGNED_LDST_RETIRED	Unaligned load-store
	0x10_BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed
	0x12_BR_PRED	Predictable branch speculatively executed
	0x7A_BR_INDIRECT_SPEC	Predictable branch speculatively executed - indirect branch
I-Cache	0x01_L1I_CACHE_REFILL	Level 1 instruction cache refill
	0x02_L1I_TLB_REFILL	Level 1 instruction TLB refill
	0x14_L1I_CACHE	Level 1 instruction cache access

Table 1-23 Cortex-A53 Profiling Events (continued)

Stream	Event Name	Comments
D-Cache	0x03_D_CACHE_REFILL", "Level 1 data cache refill"	Level 1 data cache refill
	0x04_L1D_CACHE_ACCESS	Level 1 data cache access
	0x05_L1D_TLB_REFILL	Level 1 data TLB refill
	0x15_L1D_CACHE_WB	Level 1 data cache write-back
Cycle	0x11_CPU_CYCLES	Cycle
	0x86_EXC_IRQ	0x86_EXC_IRQ
	0x86_EXC_FIQ	FIQ Exception Taken
Memory	0x13_MEM_ACCESS	Data memory access
	0x1A_MEM_ERROR	Local Data memory error
	0xC0_EXT_MEM_REQ	External Memory Request
	0xC1_NC_EXT_MEM_REQ	Non cacheable external memory request
	0xC2_Linefill_Prefetch	Linefill because of prefetch
	0xC3_Throttle	Instruction Cache Throttle occurred
	0xC4_Entering_Read_Alloc	Entering read allocation mode
	0xC5_Read_Alloc_Mode	Read Allocate Mode
L2-Cache	0x16_L2D_CACHE	Level 2 data cache access
	0x17_L2D_CACHE_REFILL	Level 2 data cache refill
	0x18_L2D_CACHE_WB	Level 2 data cache write-back
Bus	0x19_BUS_ACCESS	Bus access
	0x1D_BUS_CYCLES	Bus cycles
	0x60_BUS_ACCESS_LD	Bus access - Read
	0x60_BUS_ACCESS_ST	Bus access - Write

Table 1-23 Cortex-A53 Profiling Events (continued)

Stream	Event Name	Comments
Microarchi-tecture	0x1E_CHAIN	Odd performance counter chain mode
	0xC6_PREDECODE_ERR	Predecode error
	0xC7_DATA_WR_STALL	Data Write operation that stalls the pipeline because of the store buffer being full
	0xC8_SCU_SNOOPED	SCU Snooped data from another CPU for this CPU
	0xC9_COND_BR	Conditional branch executed
	0xCA_INDIRECT_BR_MIS	Indirect branch mis-predicted
	0xCB_INDIRECT_BR_MIS_MISC	Indirect branch mis-predicted because of address mis-compare
	0xCC_COND_BR_MIS	Conditional branch mis-predicted
	0xE0_attr_evnt_e0 — 0xE8_attr_evnt_e8	attr_evnt_e0 — attr_evnt_e8

1.6.2 Software Profiling

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

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**

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.