Cortex®-A32 Cycle Model

Version 9.7.0

User Guide

Non-Confidential

arm

101008_0907_00 (ID011118)

Cortex-A32 Cycle Model
User Guide

Release Information

The following changes have been made to this document.

Change History

Issue	Date	Confidentiality	Change
0907-00	February 2018	Non-Confidential	Release with 9.7
0906-00	November 2017	Non-Confidential	Release with 9.6.
0902-00-01	May 2017	Non-Confidential	Release with 9.2.

Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

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-A32 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

This section lists related publications.

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

- *SoC Designer User Guide* (100996)

The following publications provide reference information about Arm products:

- *Arm Cortex-A32 Processor Technical Reference Manual* (100241)
- *Arm Architecture Reference Manual Armv8* (DDI0487)
- *AMBA AXI and ACE Protocol Specification, Issue E* (IHI0022)
- *Large Physical Address Extensions Specification* (Arm Architecture Group) (DDI0438)

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 *Cycle Model Compiler*) from an RTL design. The Cycle Model contains a cycle- and register-accurate model of the hardware design.
Cycle Model Studio	Graphical tool for generating, validating, and executing hardware-accurate software models. It creates a Cycle Model, and it also takes a Cycle Model as input and generates a component that can be used in SoC Designer, Platform Architect, or Accellera SystemC for simulation.
CASI	*ESL API Simulation Interface*, is based on the SystemC communication library and manages the interconnection of components and communication between components.
CADI	*ESL API Debug Interface*, enables reading and writing memory and register values and also provides the interface to external debuggers.
CAPI	*ESL API Profiling Interface*, enables collecting historical data from a component and displaying the results in various formats.

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*. 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 in SoC Designer

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

1.1 Cortex-A32 functionality

In the multiprocessor configuration, up to four Cortex-A32 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-A32 processor supports:

- Up to four Cortex-A32 processors.
- AArch32 (32-bit ISA) mode.
- 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) and AMBA 4.0 AXI4 master ports for both 32- and 64-bit modes.
- VFP Floating Point.
- Support for AXI4-Stream interface.
- Support for CHI Bus interface.

See the *Arm Cortex-A32 Technical Reference Manual* (Arm 100241) for more information.

1.1.1 Hardware Features not supported by the Cycle Model

The following Cortex-A32 features are not currently supported by the Cortex-A32 Cycle Model:

- ACP Slave Port
- SCU Cache Protection
- Semihosting
- Cryptography engine

1.1.2 Features additional to the hardware

The following features that are implemented in the Cortex-A32 Cycle Model do not exist in the Cortex-A32 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 17).
- Waveform dumping using the waveform-related parameters described in Table 1-3 on page 17.

1.2 Adding and configuring the SoC Designer component

The *SoC Designer User Guide* (Arm 100996) 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
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

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

1.2.2 Adding the Cycle Model to the Component Library

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

For more information on SoC Designer Canvas, see the *SoC Designer User Guide* (Arm 100996).

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 15).

1.3 ESL ports

This section describes the differences between the pins listed in the *Arm Cortex-A32 Technical Reference Manual* (100372) and those on the Cortex-A32 Cycle Model. Certain hardware pins have been converted to init-time Cycle Model parameters.

- Available component ESL ports — Describes ports that have been added to the Cycle Model, such as clocks and resets required by SoC Designer, 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-A32 Cycle Model ESL transactor and special pins that are exposed in SoC Designer. See the *Arm Cortex-A32 Technical Reference Manual* (100372) for more information.

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.

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
APB_slave_Debug	APB Transactor Slave.	Transaction Slave
CLKIN	Main clock of the Cortex-A32 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)
AXI4Stream_slave_ Distributor	AXI4-Stream Slave transactor.	Transaction Slave
AXI4Stream_master_ Processor	AXI4-Stream Master transactor.	Transaction Master
clk_in	This port is used internally. Leave unconnected.	Clock Slave

1.3.2 Tied pins

The following signals are tied to a certain value:

- ACINACTM (high)
- CIHSBYPASS (low)
- CISBYPASS (low)
- CPUQREQn (high)
- CTICHIN (low)
- CTICHOUTACK (low)
- CTIIRQACK (low)
- DBGEN (high)
- DFTCGEN (low)
- DFTMCPHOLD (low)
- DFTRAMHOLD (low)
- DFTRSTDISABLE (low)
- L2QREQn (high)
- NIDEN (high)
- MBISTREQ (low)
- SPIDEN (high)
- SPNIDEN (high)
- nMBISTRESET (high)
- CLREXMONREQ (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 differs slightly 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
ACE_master Enable Debug Messages [1]	Enables ACE_Master port debug.	true, false	false	Runtime
ACE_master 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. 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
ACLKENM	ACE Master Input Clock Enable When CHI is enabled, this port is not shown.	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
APB_slave_Debug Base Address[1]	Start of the Debug_APB port address region.	Address	0x0	Init

Table 1-3 Component parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
APB_slave_Debug Enable Debug Messages[1]	Enable Debug_APB port debug.	true, false	false	Runtime
APB_Slave_Debug 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. 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
APB_slave_Debug Size[1]	Size of the Debug_APB port address region.	Size	0x100000000	Init
ARM CycleModels DB Path	Sets the directory path to the database file.	Not Used	empty	Init
ATCLKEN	ATB clock enable.	0, 1	0	Runtime
ATREADYMx	ATB device ready.	0, 1	0	Runtime
AXI4Stream_master_ Processor Enable Debug Messages[1]	Whether debug messages are enabled on the AXI4 Master port.	true, false	false	Runtime
AXI4Stream_slave_ Distributor Enable Debug Messages[1]	Whether debug messages are enabled on the AXI4 Slave port.	true, false	false	Runtime
BROADCASTCACHE-MAINT	Enables broadcasting of cache maintenance operations to downstream caches: 0 — Cache maintenance operations are not broadcast to downstream caches. 1 — Cache maintenance operations are broadcast to downstream caches.	0, 1	0	Runtime
BROADCASTINNER	Enable broadcasting of Inner Shareable transactions: 0 — Inner Shareable transactions are not broadcast externally. 1 — Inner Shareable transactions are broadcast externally.	0, 1	0	Runtime

Table 1-3 Component parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
BROADCASTOUTER	Enable broadcasting of outer shareable transactions: 0 — Outer Shareable transactions are not broadcast externally. 1 — Outer Shareable transactions are broadcast externally.	0, 1	0	Runtime
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.	true, false	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
CNTVALUEB	Global system counter value in binary format.	0 to $(2^{64} - 1)$	0	Runtime
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
DBGPWRDUP	Processor powered-up. 0 - Processor is powered down 1 - Processor is powered up	0, 1	0	Runtime

Table 1-3 Component parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
DBGROMADDR	External debug device CoreSight system configuration. Specifies bits [31:12] of the ROM.	Table Physical Address	0-fffffff	Init
DBGROMADDRV	Valid signal for DBGROMADDR.	bool	false	Init
Dump Waveforms	Whether SoC Designer dumps waveforms for this component.	true, false	false	Runtime
EDBGRQ	External debug request: 0 — No external debug request. 1 — External debug request.	0, 1	0	Runtime
Enable Debug Messages	Whether debug messages are logged for the component.	bool	false	Runtime
EVENT1	External debug request: 0 — No external debug request. 1 — External debug request.	0, 1	0	Runtime
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
L2FLUSHREQ	L2 hardware flush request.	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.	true, false	true	Runtime
nFIQ	FIQ request. Active-LOW, level sensitive, asynchronous FIQ interrupt request: 0 — Activate FIQ interrupt. 1 — Do not activate FIQ interrupt.	0, 1	0	Runtime
nIRQ	IRQ request input lines. Active-LOW, level sensitive, asynchronous interrupt request: 0 — Activate interrupt. 1 — Do not activate interrupt.	0, 1	0	Runtime

Table 1-3 Component parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
nREI	RAM Error Interrupt request. Active-LOW, edge sensitive: 0 — Activate REI request. Reports an asynchronous RAM error in the system. 1 — Do not activate REI request.	0, 1	0	Runtime
nSEI	System Error Interrupt request. Active-LOW, edge sensitive: 0 — Activate SEI request. 1 — Do not activate SEI request.	0, 1	0	Runtime
nVFIQ	Virtual FIQ request. Active-LOW, level sensitive, asynchronous FIQ interrupt request: 0 — Activate FIQ interrupt. 1 — Do not activate FIQ interrupt.	0, 1	0	Runtime
nVIRQ	Virtual IRQ request. Active-LOW, level sensitive, asynchronous interrupt request: 0 — Activate interrupt. 1 — Do not activate interrupt.	0, 1	0	Runtime
nVSEI	Virtual System Error Interrupt request. Active-LOW, edge sensitive: 0 — Activate virtual SEI request. 1 — Do not activate virtual SEI request.	0, 1	0	Runtime
NODEID	CHI Node Identifier.	Integer	0x7	Init
PADDRDBG31	APB address bus bit[31]: 0 — Not an external debugger access. 1 — External debugger access.	0, 1	0	Runtime
PCLKENDBG	APB DBG Clock Enable.	0,1	1	Runtime
PERIPHBASE	Peripheral base [39:0] (Bits 14 - 0 are ignored).	integer	0x0013000000	Init
SAMADDRMAP[0-15]	CHI Region Mapping. Refer to the *Cortex-A32 Processor Integration Manual* (Arm 100245) for details.	integer	0	Init
SAMMNBASE	MN base address.	integer	0x90	Init
SAMMNNODEID	MN Node ID.	integer	0	Init
SAMHNI[1,0]NODEID	HN-I Node ID.	integer	0	Init
SAMHNF[0-7]NODEID	HN-F Node ID.	integer	0	Init

Table 1-3 Component parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
SAMHNFMODE	HN-F interleaving module.	integer	0	Init
SCLKEN	CHI interface bus clock enable (shown when CHI is enabled).	0, 1	1	Init
SYNCREQM0	ETM Signal. Synchronization request from trace sink.	0, 1	0	Runtime
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[2]	Name of the waveform file.	*string*	arm_cm_CORTEXA32x1.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 menu	1 ns	Init

1. The presence of this parameter is configuration-dependent.
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-A32 Cycle Model has a debug interface (CADI) that allows the user to view, manipulate, and control the registers and memory.

Note: CPUs are modeled as masters that issue debug access downstream to other components. Upstream debug access into CPU models through slave ports is not supported. Waveforms and monitoring can be used to track transaction data.

1.5.1 Register information

This release of the Cortex-A32 Cycle Model does not support Register views.

1.5.2 Run to debug point

This release of the Cortex-A32 Cycle Model does not support Run to Debug Point functionality.

1.5.3 Memory information

This release of the Cortex-A32 Cycle Model does not support Memory views.

1.5.4 Disassembly view

This release of the Cortex-A32 Cycle Model does not support Disassembly views.

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.

1.6.1 Hardware profiling

Hardware events are uniquely identified by their Event Number as defined in the Cortex-A32 Technical Reference Manual. 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 1-4.

Table 1-4 Cortex-A32 Cycle Model 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 1-4 Cortex-A32 Cycle Model 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 1-4 Cortex-A32 Cycle Model 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

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.