



Arm® CoreLink™ MMU-700 System Memory Management Unit

Revision: r1p0

Technical Reference Manual

Non-Confidential

Issue 06

Copyright © 2019–2021 Arm Limited (or its affiliates). 101542_0100_06_en
All rights reserved.



Arm® CoreLink™ MMU-700 System Memory Management Unit

Technical Reference Manual

Copyright © 2019–2021 Arm Limited (or its affiliates). All rights reserved.

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-01	25 October 2019	Confidential	First issue for r0p0 BET release
0000-02	30 March 2020	Confidential	First issue for r0p0 LAC release
0001-03	14 September 2020	Non-Confidential	First issue for r0p1 EAC release
0001-04	19 February 2021	Non-Confidential	Second issue for r0p1 EAC release
0100-05	26 March 2021	Non-Confidential	First issue for r1p0 EAC release
0100-06	10 December 2021	Non-Confidential	Second issue for r1p0 EAC release

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL,

INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>.

Copyright © 2019–2021 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

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.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

This document includes language that can be offensive. We will replace this language in a future issue of this document.

To report offensive language in this document, email terms@arm.com.

Contents

1 Introduction.....	12
1.1 Product revision status.....	12
1.2 Intended audience.....	12
1.3 Conventions.....	12
1.4 Additional reading.....	14
1.5 Feedback.....	15
2 Overview of the System Memory Management Unit.....	16
2.1 About the System Memory Management Unit.....	16
2.2 Compliance.....	17
2.2.1 Arm architecture.....	17
2.2.2 SMMU architecture.....	17
2.2.3 AMBA Distributed Translation Interface protocol.....	17
2.2.4 AMBA ACE5-Lite and AMBA AXI5 protocol.....	17
2.2.5 AMBA APB protocol.....	18
2.2.6 LTI protocol.....	18
2.2.7 LPI Q-Channel protocol.....	18
2.3 Features.....	18
2.4 Interfaces.....	20
2.5 Configurable options.....	21
2.6 Product documentation and design flow.....	22
2.6.1 Documentation.....	22
2.6.2 Design flow.....	22
2.7 Product revisions.....	23
3 Functional description.....	25
3.1 About the functions.....	25
3.1.1 Translation Buffer Unit.....	27
3.1.2 Translation Control Unit.....	31
3.1.3 DTI interconnect.....	33
3.2 Interfaces.....	34
3.2.1 TCU interfaces.....	34
3.2.2 TBU interfaces.....	38

3.2.3 Integration TBU.....	43
3.2.4 DTI interconnect interfaces.....	54
3.3 Operation.....	57
3.3.1 DTI overview.....	57
3.3.2 Performance Monitoring Unit.....	58
3.3.3 Multiple LTI interface TBU.....	64
3.3.4 Main TLB direct indexing and main TLB direct partitioning.....	65
3.3.5 RAS implementation.....	66
3.3.6 Quality of Service.....	69
3.3.7 Distributed Virtual Memory messages.....	69
3.3.8 TCU transaction handling.....	70
3.3.9 TCU prefetch.....	71
3.3.10 Error responses.....	73
3.3.11 Conversion between ACE-Lite and Armv8 attributes.....	73
3.3.12 AXI USER bits that MMU-700 TBU TBM defines.....	76
3.4 Constraints and limitations of use.....	77
3.4.1 SMMUv3 implementation.....	77
3.4.2 AMBA implementation.....	80
3.4.3 MPAM implementation.....	88
3.4.4 Local Translation Interface implementation.....	93
3.5 Configuration parameters and methodology.....	94
3.5.1 Translation Control Unit I/O configuration parameters.....	94
3.5.2 Translation Control Unit buffer configuration parameters.....	95
3.5.3 Translation Control Unit debug configuration parameters.....	97
3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters.....	97
3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters.....	99
3.5.6 ACE-Lite Translation Buffer Unit register slice configuration parameters.....	100
3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters.....	101
3.5.8 Local Translation Interface Translation Buffer Unit configuration parameters.....	102
3.5.9 Common Translation Buffer Unit debug configuration parameters.....	103
3.6 Debug capability.....	103
4 Programmers model.....	105
4.1 About the programmers model.....	105
4.1.1 Clearing ERRSTATUS registers.....	106

4.2 SMMU architectural registers.....	107
4.3 MMU-700 memory map.....	110
4.3.1 Main MMU-700 memory map.....	111
4.3.2 TCU memory map.....	111
4.3.3 TBU memory map.....	113
4.4 MMU-700 registers summary.....	113
4.4.1 TCU identification registers summary.....	114
4.4.2 TCU and TBU PMU identification registers summary.....	114
4.4.3 TCU Reliability, Availability, and Service registers summary.....	115
4.4.4 TCU microarchitectural registers summary.....	115
4.4.5 TCU system discovery registers summary.....	115
4.4.6 TCU integration registers summary.....	116
4.4.7 TBU identification registers summary.....	116
4.4.8 TBU Reliability, Availability, and Serviceability registers summary.....	117
4.4.9 TBU microarchitectural registers summary.....	117
4.4.10 TBU system discovery registers summary.....	118
4.4.11 TBU integration registers summary.....	118
4.5 TCU component and peripheral ID registers.....	119
4.6 TCU PMU registers.....	120
4.6.1 Registers.....	120
4.6.2 Events.....	121
4.6.3 SMMU_PMCG_CFGR.....	122
4.6.4 SMMU_PMCG_CEID{0-1} registers.....	122
4.6.5 PMU ID registers.....	122
4.7 TCU microarchitectural registers.....	123
4.7.1 TCU_CTRL register.....	124
4.7.2 TCU_QOS register.....	125
4.7.3 TCU_CFG register.....	126
4.7.4 TCU_STATUS register.....	127
4.7.5 TCU_NODE_CTRLn register.....	128
4.7.6 TCU_NODE_STATUSn register.....	130
4.7.7 TCU_SCR register.....	131
4.7.8 TCU_WC_SxLy_CMAX registers.....	133
4.8 TCU RAS registers.....	133
4.8.1 TCU_ERRFR register.....	134
4.8.2 TCU_ERRCTLR register.....	135

4.8.3 TCU_ERRSTATUS register.....	135
4.8.4 TCU_ERRGEN register.....	138
4.9 TCU system discovery registers.....	139
4.9.1 TCU_SYSDISC0 system discovery register.....	140
4.9.2 TCU_SYSDISC1 system discovery register.....	141
4.9.3 TCU_SYSDISC2 system discovery register.....	142
4.9.4 TCU_SYSDISC3 system discovery register.....	143
4.9.5 TCU_SYSDISC4 system discovery register.....	144
4.9.6 TCU_SYSDISC5 system discovery register.....	145
4.9.7 TCU_SYSDISC6 system discovery register.....	146
4.9.8 TCU_SYSDISC7 system discovery register.....	147
4.9.9 TCU_SYSDISC8 system discovery register.....	148
4.9.10 TCU_SYSDISC9 system discovery register.....	149
4.9.11 TCU_SYSDISC10 system discovery register.....	150
4.9.12 TCU_SYSDISC11 system discovery register.....	151
4.9.13 TCU_SYSDISC12 system discovery register.....	152
4.9.14 TCU_SYSDISC13 system discovery register.....	153
4.9.15 TCU_SYSDISC14 system discovery register.....	154
4.9.16 TCU_SYSDISC15 system discovery register.....	155
4.9.17 TCU_SYSDISC16 system discovery register.....	156
4.9.18 TCU_SYSDISC17 system discovery register.....	157
4.10 TCU PIU integration registers.....	158
4.10.1 ITEN register for the TCU.....	158
4.10.2 ITOP register for the TCU Programmer Interface Unit.....	159
4.11 TCU TMU integration registers.....	160
4.11.1 ITOP register for the TCU Translation Management Unit.....	161
4.11.2 ITIN register for the TCU Translation Management Unit.....	162
4.12 TBU component and peripheral ID registers.....	163
4.13 TBU PMU registers.....	163
4.13.1 Registers.....	163
4.13.2 Events.....	164
4.13.3 SMMU_PMCGR_CFGR.....	165
4.13.4 SMMU_PMCGR_CEID{0-1} registers.....	165
4.13.5 PMU ID registers.....	166
4.14 TBU microarchitectural registers.....	166
4.14.1 TBU_CTRL register.....	167

4.14.2 TBU_LTI_PORT_RESOURCE_LIMIT register.....	168
4.14.3 TBU_SCR register.....	171
4.15 TBU RAS registers.....	172
4.15.1 TBU_ERRFR register.....	173
4.15.2 TBU_ERRCTLR register.....	174
4.15.3 TBU_ERRSTATUS register.....	175
4.15.4 TBU_ERRGEN register.....	177
4.16 TBU system discovery registers.....	178
4.16.1 TBU_SYSDISC0 system discovery register.....	178
4.16.2 TBU_SYSDISC1 system discovery register.....	179
4.16.3 TBU_SYSDISC2 system discovery register.....	180
4.16.4 TBU_SYSDISC3 system discovery register.....	181
4.16.5 TBU_SYSDISC4 system discovery register.....	182
4.16.6 TBU_SYSDISC5 system discovery register.....	183
4.16.7 TBU_SYSDISC6 system discovery register.....	184
4.16.8 TBU_SYSDISC7 system discovery register.....	185
4.16.9 TBU_SYSDISC8 system discovery register.....	186
4.16.10 TBU_SYSDISC9 system discovery register.....	187
4.16.11 TBU_SYSDISC10 system discovery register.....	188
4.16.12 TBU_SYSDISC11 system discovery register.....	189
4.16.13 TBU_SYSDISC12 system discovery register.....	190
4.16.14 TBU_SYSDISC13 system discovery register.....	191
4.16.15 TBU_SYSDISC14 system discovery register.....	192
4.17 TBU integration registers.....	193
4.17.1 ITEN register for the TBU.....	193
4.17.2 ITOP_TBU register.....	194
4.17.3 ITIN_TBU register.....	195
A Signal descriptions.....	197
A.1 TCU signals.....	197
A.1.1 TCU clock and reset signals.....	197
A.1.2 TCU QTW/DVM interface signals.....	197
A.1.3 TCU programming interface signals.....	199
A.1.4 TCU SYSCO interface signals.....	200
A.1.5 TCU PMU snapshot interface signals.....	200
A.1.6 TCU LPI_PD interface signals.....	201

A.1.7 TCU LPI_CG interface signals.....	201
A.1.8 TCU DTI interface signals.....	201
A.1.9 TCU interrupt signals.....	202
A.1.10 TCU MSI interface signals.....	203
A.1.11 TCU event interface signal.....	203
A.1.12 TCU tie-off signals.....	205
A.1.13 TCU ELA debug signals.....	206
A.2 TBU signals.....	208
A.2.1 TBU clock and reset signals.....	208
A.2.2 TBU TBS interface signals.....	208
A.2.3 TBU TBM interface signals.....	212
A.2.4 TBU PMU snapshot interface signals.....	216
A.2.5 TBU LPI_PD interface signals.....	216
A.2.6 TBU LPI_CG interface signals.....	216
A.2.7 TBU DTI interface signals.....	217
A.2.8 TBU LTI interface signals.....	217
A.2.9 TBU interrupt signals.....	217
A.2.10 TBU tie-off signals.....	218
A.2.11 TBU ELA debug signals.....	220
A.3 TCU and TBU shared signals.....	221
A.3.1 TCU and TBU test and debug signals.....	221
A.4 DTI signals.....	222
A.4.1 DTI interconnect switch signals.....	222
A.4.2 DTI interconnect sizer signals.....	224
A.4.3 DTI interconnect register slice signals.....	226
B ELA signal descriptions.....	228
B.1 TCU observation interfaces.....	228
B.2 ACE-Lite TBU observation interfaces.....	231
B.3 LTI TBU observation interfaces.....	234
C Software initialization examples.....	237
C.1 Initializing the SMMU.....	237
C.1.1 Allocating the Command queue.....	237
C.1.2 Allocating the Event queue.....	238
C.1.3 Configuring the Stream table.....	238
C.1.4 Initializing the Command queue.....	239

C.1.5 Initializing the Event queue.....	239
C.1.6 Invalidating TLBs and configuration caches.....	239
C.1.7 Creating a basic Context Descriptor.....	240
C.1.8 Creating a Stream Table Entry.....	241
C.2 Enabling the SMMU.....	242
D Revisions.....	243
D.1 Revisions.....	243

1 Introduction

1.1 Product revision status

The r_xp_y identifier indicates the revision status of the product described in this manual, for example, $r1p2$, where:

- r_x** Identifies the major revision of the product, for example, $r1$.
- p_y** Identifies the minor revision or modification status of the product, for example, $p2$.

1.2 Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the MMU-700.

1.3 Conventions

The following subsections describe conventions used in Arm documents.







Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Convention	Use
<i>italic</i>	Introduces citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
monospace	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
monospace bold	Denotes language keywords when used outside example code.
monospace <u>underline</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <pre>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></pre>

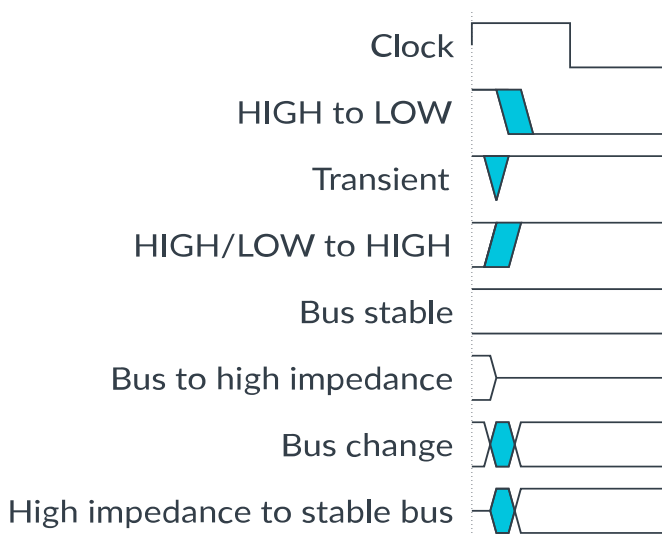
Convention	Use
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the <i>Arm® Glossary</i> . For example, IMPLEMENTATION DEFINED , IMPLEMENTATION SPECIFIC , UNKNOWN , and UNPREDICTABLE .
 Caution	This represents a recommendation which, if not followed, might lead to system failure or damage.
 Warning	This represents a requirement for the system that, if not followed, might result in system failure or damage.
 Danger	This represents a requirement for the system that, if not followed, will result in system failure or damage.
 Note	This represents an important piece of information that needs your attention.
 Tip	This represents a useful tip that might make it easier, better or faster to perform a task.
 Remember	This is a reminder of something important that relates to the information you are reading.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

Figure 1-1: Key to timing diagram conventions



Signals

The signal conventions are:

Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name, n denotes an active-LOW signal.

1.4 Additional reading

This document contains information that is specific to this product. See the following documents for other relevant information:

Table 1-2: Arm publications

Document name	Document ID	Licensee only
Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual	101543	Y
Arm® CoreLink™ MMU-700 System Memory Management Unit Release Note	PJDOC-1779577084-34669	Y
Arm® CoreLink™ LPD-500 Low Power Distributor Technical Reference Manual	100361	N
Arm® CoreLink™ LPD-500 Low Power Distributor Integration and Implementation Manual	100362	Y
Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual	100806	N
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101088	N
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Configuration and Integration Manual	101089	Y
Arm® CoreLink™ ADB-400 AMBA® Domain Bridge User Guide	DUI 0615	Y
Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2	IHI 0070C.a	N
Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile	DDI 0487E.a	N
AMBA® AXI and ACE Protocol Specification	IHI 0022H	N
AMBA® 4 AXI4-Stream Protocol Specification	IHI 0051A	N
AMBA® APB Protocol Specification	IHI 0024C	N
AMBA® DTI Protocol Specification	IHI 0088E	N
AMBA® LTI Protocol Specification	IHI 0089A	N
AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces	IHI 0068C	N
Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A	DDI 0598B.a	N
Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability (RAS), for Armv8-A	DDI 0587C.b	N

Document name	Document ID	Licensee only
Arm® Server Base System Architecture 7.0 Platform Design Document	DEN 0029	N
GIC MSI Delivery Interface	ARM AES 0019A (Beta)	Y

1.5 Feedback

Arm welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

Information about how to give feedback on the content.

If you have comments on content then send an e-mail to errata@arm.com. Give:

- The title Arm® CoreLink™ MMU-700 System Memory Management Unit Technical Reference Manual.
- The number 101542_0100_06_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.



Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

2 Overview of the System Memory Management Unit

This chapter provides an overview of the MMU-700 *System Memory Management Unit* (SMMU).

2.1 About the System Memory Management Unit

The MMU-700 is a *System-level Memory Management Unit* (SMMU) that translates an input address to an output address. This translation is based on address mapping and memory attribute information that is available in the MMU-700 internal registers and translation tables.

The MMU-700 implements the Arm® SMMU architecture version 3.2, SMMUv3.2, as the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) defines.

An address translation from an input address to an output address is described as a *stage* of address translation. The MMU-700 can perform:

- Stage 1 translations that translate an input *virtual address* (VA) to an output *physical address* (PA) or *intermediate physical address* (IPA)
- Stage 2 translations that translate an input IPA to an output PA
- Combined stage 1 and stage 2 translations that translate an input VA to an IPA, and then translate that IPA to an output PA. The MMU-700 performs translation table walks for each stage of the translation.

In addition to translating an input address to an output address, a stage of address translation also defines the memory attributes of the output address. With a two-stage translation, the stage 2 translation can modify the attributes that the stage 1 translation defines. A stage of address translation can be disabled or bypassed, and the MMU-700 can define memory attributes for disabled and bypassed stages of translation.

The MMU-700 uses inputs from the requesting master to identify a context. Configuration tables in memory define how the MMU-700 is to translate each context, such as which translation tables to use.

The MMU-700 can cache the result of a translation table lookup in a *Translation Lookaside Buffer* (TLB). It can also cache configuration tables in a configuration cache.

The MMU-700 contains the following key components:

- *Translation Buffer Units* (TBUs) that use a TLB to cache translation tables
- A *Translation Control Unit* (TCU) that controls and manages address translations
- *Distributed Translation Interface* (DTI) interconnect components that connect multiple TBUs to the TCU

Related information

[About the functions](#) on page 25

2.2 Compliance

The MMU-700 complies with, or implements, the specifications that this section describes. This *Technical Reference Manual* (TRM) complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

2.2.1 Arm architecture

The MMU-700 implements parts of the Armv8.5 *Virtual Memory System Architecture* (VMSA), as the [Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile](#) defines. The SMMUv3.2 architecture describes the parts of VMSA that apply to the MMU-700.

2.2.2 SMMU architecture

The MMU-700 implements the SMMUv3.2 architecture.

See the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#).

Related information

[SMMUv3 implementation](#) on page 77

2.2.3 AMBA Distributed Translation Interface protocol

The MMU-700 implements the *Distributed Translation Interface* (DTI) protocol, as the [AMBA® DTI Protocol Specification](#) defines.

The DTI interfaces use an AXI4-Stream interface, as the [AMBA® 4 AXI4-Stream Protocol Specification](#) defines.

Related information

[DTI overview](#) on page 57

2.2.4 AMBA ACE5-Lite and AMBA AXI5 protocol

The MMU-700 complies with the AMBA® ACE5-Lite protocol.

For more information, see the [AMBA® AXI and ACE Protocol Specification](#).

Related information

[AMBA implementation](#) on page 80

2.2.5 AMBA APB protocol

The MMU-700 complies with the AMBA APB4 protocol, as the [AMBA® APB Protocol Specification](#) defines.

2.2.6 LTI protocol

The MMU-700 complies with the LTI protocol, as the [AMBA® LTI Protocol Specification](#) defines.

Related information

[LTI TBU LTI interface](#) on page 41

2.2.7 LPI Q-Channel protocol

The MMU-700 complies with the LPI Q-Channel, as the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#) defines.

Related information

[TCU LPI_PD interface signals](#) on page 200

[TCU LPI_CG interface signals](#) on page 201

[TBU LPI_PD interface signals](#) on page 216

[TBU LPI_CG interface signals](#) on page 216

2.3 Features

The MMU-700 provides the following features:

Compliance with the SMMUv3.2 architecture

- Support for stage 1 translation, stage 2 translation, and stage 1 followed by stage 2 translation
- Support for Armv8 AArch32 and AArch64 translation table formats
- Support for 4KB, 16KB, and 64KB granule sizes in AArch64 format
- Support for *PCI Express* (PCIe) integration, including:
 - *Address Translation Services* (ATS), including full and split-stage ATS
 - *Process Address Space IDs* (PASIDs)
 - *Access Control Services* (ACS)

- Support for *Page Request Interface* (PRI), as SMMUv3 defines. PRI is an optional PCIe ATS extension that enables support for unpinned memory in PCIe.
- Support for MPAM
- Support for Secure-EL2
- Masters can be stalled while a processor handles translation faults, enabling software support for on-demand paging
- Configuration tables in memory can support more than a million active translation contexts
- Queues in memory perform MMU-700 management. There is no requirement to stall a processor when it accesses the MMU-700.
- A *Performance Monitoring Unit* (PMU) in each TBU and TCU that enables MMU-700 performance to be investigated
- *Reliability, Serviceability, and Availability* (RAS) features for RAM corruption detection and correction

Support for AMBA® interfaces

- ACE5-Lite TBU transaction interfaces that support cache stash transactions, deallocating transactions, and cache maintenance
- An architected AXI5 extension that communicates per-transaction translation stream information
- An ACE5-Lite+*Distributed Virtual Memory* (DVM) TCU table walk interface that enables Armv8.5 processors to perform shared TLB invalidate operations without accessing the MMU-700 directly
- An ACE5 Low-Power extension that enables the TCU to subscribe to DVM TLB invalidate requests on powerup and powerdown without reprogramming the DTI interconnect
- AMBA® DTI communication between the TCU and TBUs, enabling masters to request translations and implement TBU functionality internally
- Support for the AMBA® *Low-Power Interface* (LPI) Q-Channel so that standard controllers can control power and clock gating
- AXI5 **WAKEUP** signaling on all interfaces, including DTI and APB interfaces
- Support for ACE5-Lite atomic transactions in the ACE-Lite TBU
- Support for *Local Translation Interface* (LTI)
- Support for a dedicated *Generic Interrupt Controller* (GIC) integration, with *Message Signaled Interrupts* (MSIs) supported for common interrupt types

Support for flexible integration

- You can place a configurable number of TBUs close to the masters being translated
- Communication between TBU and TCU over AXI4-Stream is supported using the supplied DTI interconnect components, or any other AXI4-Stream interconnect

- DTI interconnect components support hierarchical topologies and control the tradeoff between the number of wires and the DTI bandwidth

Support for high-performance translation

- Scalable configurable MicroTLB and *Main TLB* (MTLB) in the TBU can reduce the number of translation requests to the TCU
- TBU direct indexing and MTLB partitioning enable the use of MTLB entries to be managed outside the TBU, improving real-time translation performance
- Optimization enables storage of all architecturally-defined page and block sizes, including contiguous page and block entries, as a single entry in the TBU and TCU TLBs (WCs)
- Per-TBU prioritization in the TCU enables high-priority transaction streams to be translated before low-priority streams
- TCU prefetch of translation tables, which can be enabled on a per-context basis, improves translation performance for real-time masters that access memory linearly
- *Hit-Under-Miss* (HUM) support in the TBU enables transactions with different AXI IDs to be propagated out of order, when a translation is available
- TBU detects multiple transactions that require the same translation so that only one TBU request to the TCU is required
- TCU detects multiple translations that require the same table in memory so that only one TCU memory request is required
- Multi-level, multi-stage walk caches in the TCU reduce translation cost by performing only part of the table walk process on a miss
- A configurable number of concurrent translations in the TBU and TCU promotes high translation throughput

Trace debugging

- Using a CoreSight™ ELA-600 Embedded Logic Analyzer

2.4 Interfaces

Both the TCU and TBU support the following common interfaces:

- Clocks and resets
- *Distributed Translation Interface* (DTI)
- Tie-offs
- Interrupts
- PMU snapshot
- Test and debug
- LPI clock gating
- LPI powerdown

The TCU also supports the following interfaces:

- Programming
- System coherency
- *Queue and Table Walk (QTW)/DVM*
- *Generic Interrupt Controller (GIC) Message Signaled Interrupt (MSI) interface*

The ACE-Lite TBU also supports the following interfaces:

- *Transaction slave (TBS)*
- *Transaction master (TBM)*

The LTI TBU also supports the *Local Translation Interface (LTI)*.

Related information

[Interfaces](#) on page 34

2.5 Configurable options

The MMU-700 is highly configurable and provides configuration options for each of the main components.

For the TCU, you can configure the following:

- Size of each cache
- Data width of the QTW/DVM interface
- Number of translations that can be performed at the same time
- Number of translation requests that can be accepted from all DTI masters

For the TBU, you can configure the following:

- Size of each cache
- Number of transactions that can be translated at the same time
- Register slices

For the ACE-Lite TBU, you can configure the following:

- Write data buffer depth
- Number of outstanding read and write transactions that the TBM interface supports
- Width of data, ID, User, StreamID, and SubstreamID signals on the TBS and TBM interfaces



Depths are specified as a discrete number of entries.

You can also configure the DTI interconnect components to meet your system requirements.

See [3.5 Configuration parameters and methodology](#) on page 94.

Related information

[Configuration parameters and methodology](#) on page 94

2.6 Product documentation and design flow

This section describes the MMU-700 documentation in relation to the design flow.

2.6.1 Documentation

The MMU-700 documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the MMU-700. It is required at all stages of the design flow. The choices that are made in the design flow can mean that some behaviors that are described in the TRM are not relevant. If you are programming the MMU-700, then contact:

- The implementer to determine:
 - The build configuration of the implementation
 - The integration, if any, that was performed before implementing the MMU-700
- The integrator to determine the pin configuration of the device that you are using.

Configuration and Integration Manual

The *Configuration and Integration Manual* (CIM) describes:

- The available build configuration options and related issues in selecting them.
- How to integrate the MMU-700 into an SoC. This section describes the pins that the integrator must tie off to configure the macrocells for the required integration.
- The processes to sign off on the configuration, integration, and implementation of the design.

The CIM is a confidential book that is only available to licensees.

2.6.2 Design flow

The MMU-700 is delivered as synthesizable RTL. Before it can be used in a product, it must go through the following processes:

Implementation

The implementer configures and synthesizes the RTL to produce a hard macrocell. This process might include integrating RAMs into the design.

Integration

The integrator connects the implemented design into an SoC. Integration includes connecting the design to a memory system and peripherals.

Programming

The system programmer develops the software to configure and initialize the MMU-700, and tests the required application software.

Each process is separate, and can include implementation and integration choices that affect the behavior and features of the MMU-700.

The operation of the final device depends on:

Build configuration

The implementer chooses the options that affect how the RTL source files are pre-processed. These options usually include or exclude logic that affects one or more of the following:

- Area
- Maximum frequency
- Features of the resulting macrocell

Configuration inputs

The integrator configures some features of the MMU-700 by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made.

Software configuration

The programmer configures the MMU-700 by programming particular values into registers. This configuration affects the behavior of the MMU-700.

Related information

[Configurable options](#) on page 21

[Configuration parameters and methodology](#) on page 94

[Compliance](#) on page 17

2.7 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

**r0p0-
r0p1**

The following changes apply to this release:

- New system discovery registers. See [4.9 TCU system discovery registers](#) on page 139 and [4.16 TBU system discovery registers](#) on page 178.
- New parameters. See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94 and [3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters](#) on page 98.
- New stitching flow
- New generate executable

**r0p1-
r1p0**

The following changes apply to this release:

- New multiple LTI interface TBU. See [3.2.2.3 LTI TBU LTI interface](#) on page 41.
- New integration TBU. See [3.2.3 Integration TBU](#) on page 43.
- PCIe CXL.IO support
- Performance improvement by changing the configuration of some RAMs
- Changes to registers. See [4 Programmers model](#) on page 105.
- Changes to parameters. See:
 - [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94
 - [3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters](#) on page 98

3 Functional description

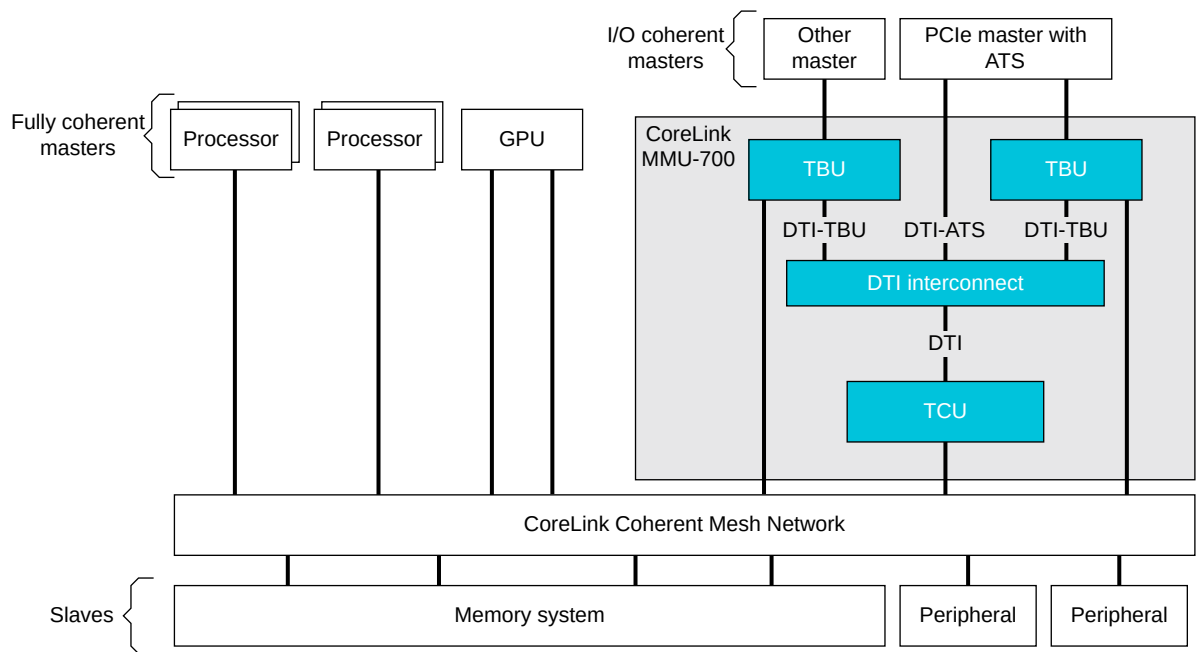
This chapter describes the functionality of the MMU-700.

3.1 About the functions

The major functional blocks of the MMU-700 are the *Translation Buffer Unit (TBU)*, *Translation Control Unit (TCU)*, and *Distributed Translation Interface (DTI)* interconnect.

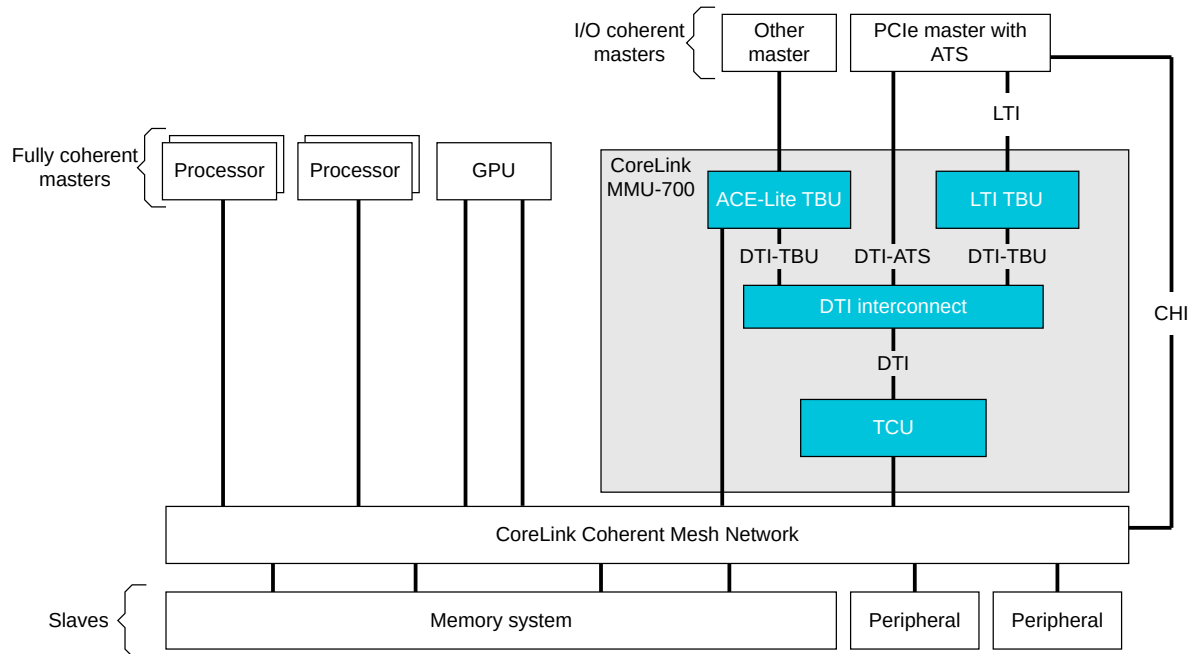
The following figure shows an example system that uses the MMU-700.

Figure 3-1: Example system with the MMU-700



The following figure shows an example system that uses the MMU-700 and includes a *Local Translation Interface (LTI)* TBU.

Figure 3-2: Example system with the MMU-700 and LTI TBU



The MMU-700 contains the following key components:

Translation Buffer Unit (TBU)

The TBU contains *Translation Lookaside Buffers* (TLBs) that cache translation tables. The MMU-700 implements a TBU that can be connected to single master or multiple masters. It is also possible to connect multiple TBUs to a single master to improve performance. These TBUs are local to the corresponding master and can be one of the following:

- ACE-Lite TBU
- LTI TBU

Translation Control Unit (TCU)

The TCU controls and manages the address translations. The MMU-700 implements a single TCU. In MMU-700-based systems, the AMBA® DTI protocol defines the standard for communicating with the TCU. See the [AMBA® DTI Protocol Specification](#).

DTI interconnect

The DTI interconnect connects multiple TBUs to the TCU.

When an MMU-700 TBU receives a transaction on the TBS or LA interface, it looks for a matching translation in its TLBs. If it has a matching translation, it uses it to translate the transaction and outputs the transaction on the TBM interface. If it does not have a matching translation, it requests a new translation from the TCU using the DTI interface.

When the TCU receives a DTI translation request, it uses the QTW interface to perform:

- Configuration table walks, which return configuration information for the translation context

- Translation table walks, that return translation information that is specific to the transaction address

The TCU contains caches that reduce the number of configuration and translation table walks that are to be performed. Sometimes no walks are required.

When the TBU receives the translation from the TCU, it stores it in its TLBs. If the translation was successful, the TBU uses it to translate the transaction, otherwise it terminates it.

A processor controls the TCU by:

- Writing commands to a Command queue in memory
- Receiving events from an Event queue in memory
- Writing to its configuration registers using the programming interface

See the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) for more information about the following:

- Translation
- How software communicates with the TCU

3.1.1 Translation Buffer Unit

A typical SMMUv3-based system includes multiple *Translation Buffer Units* (TBUs). Each TBU is located close to the component for which it provides address translation.

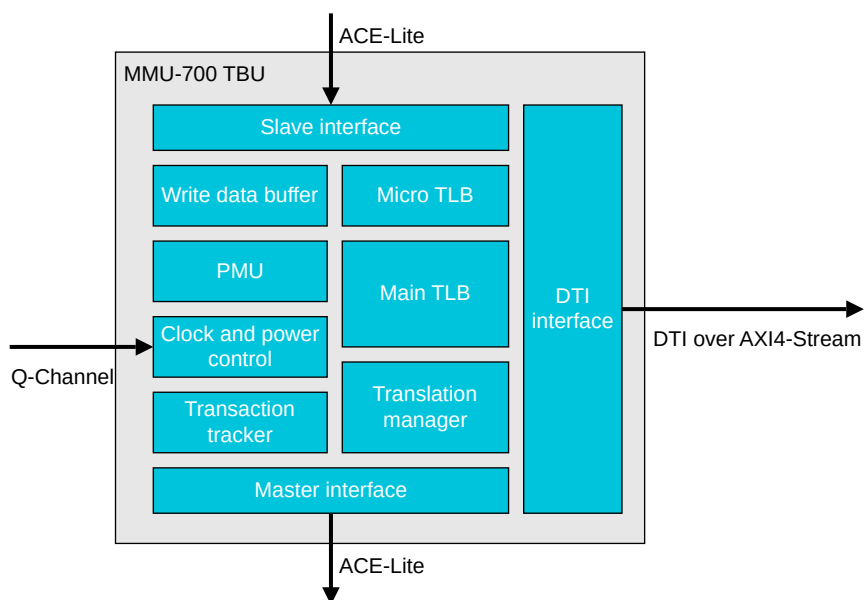
A TBU can be one of the following:

- ACE-Lite TBU
- *Local Translation Interface* (LTI) TBU

A TBU intercepts transactions and provides the required translation from a *Translation Lookaside Buffer* (TLB) if possible. If a TLB does not contain the required translation, the TBU requests translations from the TCU and then caches the translation in one of the TLBs.

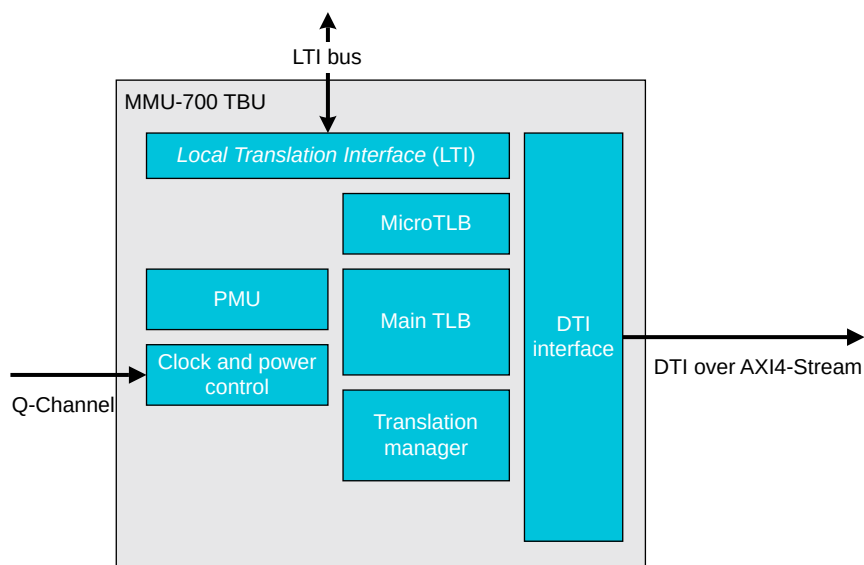
The following figure shows the ACE-Lite TBU.

Figure 3-3: MMU-700 ACE-Lite TBU



The following figure shows the LTI TBU.

Figure 3-4: MMU-700 LTI TBU



The TBU consists of:

Master and slave interfaces

ACE-Lite TBU

TBS interface

For receiving untranslated transactions from the requesting master

TBM interface

For issuing translated transactions to the rest of the system.

LTI TBU

For LTI translation requests and responses.

MicroTLB

The TBU compares incoming transactions with translations that are cached in the MicroTLB before looking in the *Main TLB* (MTLB). The MicroTLB provides end-to-end translation from an input address to an output address. You can use a tie-off signal to configure the cache replacement policy as either round-robin or *Pseudo Least Recently Used* (PLRU).

Main TLB

Each TBU includes an optional *Main TLB* (MTLB) that caches translation table walk entries from:

- Stage 1 translations
- Stage 2 translations
- Stage 1 combined with stage 2 translations

The MTLB is a set associative cache structure with a configurable number of ways and banks.

If multiple translation sizes are in use, a single transaction might require multiple lookups. Lookups are pipelined to permit a sustained rate of one lookup per cycle.

TBU direct indexing enables the MMU-700 to manage MTLB entries externally to the TBU. Direct indexing improves the predictability of TBU performance, for bus masters that have real-time performance requirements.

TBU hazarding

If the MicroTLB lookup results in a miss, the transaction checks whether there are any pending transactions from which it can use the translation. This method is called *forming a hazard*. If the transaction hazards on any pending transactions, then the transaction waits until the response is available for the hazarded transaction and uses that response. The number of unique addresses that form a hazard is limited. The `TBUCFG_HZRD_ENTRIES` parameter controls the number of unique addresses. For information about `TBUCFG_HZRD_ENTRIES`, see [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101 and [3.5.8 Local Translation Interface Translation Buffer Unit configuration parameters](#) on page 102.

For more information about TBU hazarding, see the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

Translation manager

The translation manager manages translation requests that are in progress. Each transaction occupies a translation slot until it is propagated downstream through the ACE-Lite TBM, or an LTI translation response is returned. All transactions are hazard-checked to reduce the possibility of duplicate translation requests being sent to the TCU.

There is no restriction on the ordering of transactions with different AXI IDs/LTI *Order Groups* (OGs). Transactions with different AXI IDs can be propagated downstream out-of-order.

All transactions with a given AXI ID/LTI OG value must remain ordered. The translation manager propagates such transactions when the translation is ready, provided no other transaction with the same AXI ID/LTI OG is already waiting.

For more information about AXI transaction identifiers, see the [AMBA® AXI and ACE Protocol Specification](#).

For more information about LTI OGs, see the [AMBA® LTI Protocol Specification](#).

Write data buffer

The write data buffer is available in the ACE-Lite TBU only. The optional write data buffer enables write transactions with different AXI IDs to progress through the TBU out-of-order. It reorders the data to match the downstream transaction order.

PMU

The PMU counts TBU performance-related events.

Clock and power control

The TBU has its own clock and power control, that the Q-Channels provide.

DTI interface

The master DTI interface uses the DTI protocol, typically over AXI4-Stream, to enable the TBU to communicate with a slave component. For the MMU-700, the slave component is the TCU. Although you can implement DTI over different transport protocols, the MMU-700 interfaces use AXI4-Stream.

Transaction tracker

The transaction trackers manage outstanding read and write transactions, permitting invalidation and synchronization to take place without stalling the AXI interfaces.



The transaction tracker is available in the ACE-Lite TBU only.

Related information

[Main TLB direct indexing and main TLB direct partitioning](#) on page 65

[SMMU architectural registers](#) on page 106

[Operation](#) on page 57

3.1.2 Translation Control Unit

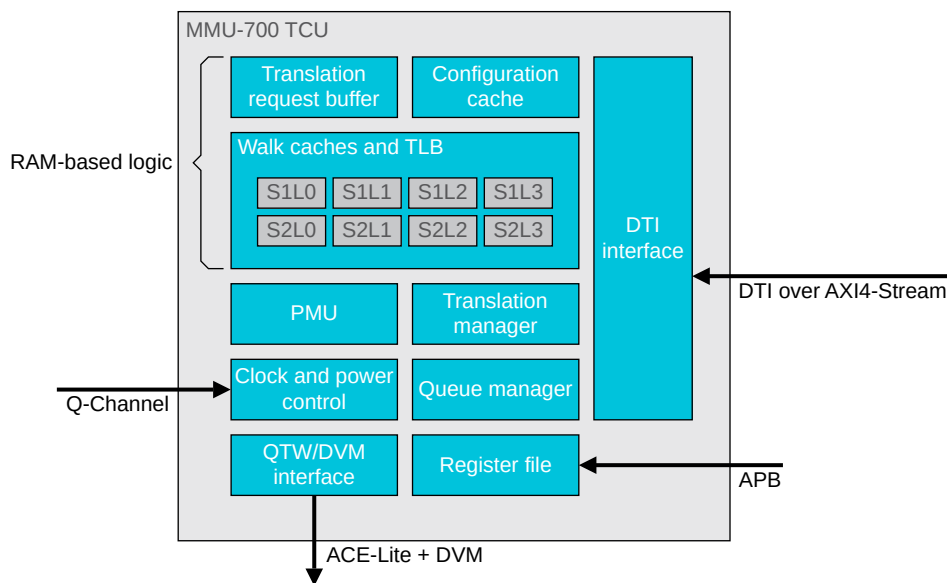
A typical SMMUv3-based system includes a single *Translation Control Unit* (TCU). The TCU is usually the largest block in the system, and performs several roles.

The TCU:

- Manages the memory queues
- Performs translation table walks
- Performs configuration table walks
- Implements backup caching structures
- Implements the SMMU programmers model

The following figure shows the TCU with its main interfaces.

Figure 3-5: MMU-700 TCU



The TCU consists of:

Walk cache

The TCU is a set-associative walk cache that has a configurable number of banks and ways and stores the results of translation table walks. During MMU-700 configuration, the cache line entries can be programmatically split to create separate walk caches that are reserved for:

- Stage 1 level 0 table entries
- Stage 1 level 1 table and block entries
- Stage 1 level 2 table and block entries

- Stage 1 level 3 block entries
- Stage 2 level 0 table entries
- Stage 2 level 1 table and block entries
- Stage 2 level 2 table and block entries
- Stage 2 level 3 block entries

To enable and disable the walk cache for a particular stage and level of translation, use the [4.7.1 TCU_CTRL register](#) on page 123. If an error occurs for a cache line entry, the [4.8.3 TCU_ERRSTATUS register](#) on page 135 identifies the affected entry.

The walk cache is useful in cases where a translation request results in a miss in other TCU caches. A subsequent hit in the walk cache requires only a single memory access to complete the translation table walk and fetch the required descriptor.

Configuration cache

The configuration caches are 4-way set-associative cache structures that store configuration information. Each entry stores the *Context Descriptor* (CD) and *Stream Table Entry* (STE) contents for a translation context.



The configuration cache does not cache the contents of intermediate configuration tables.

Translation manager

The translation manager manages translation requests that are in progress. All translation table walks and configuration table walks are hazard-checked to reduce the possibility of multiple transactions requesting duplicate walks.

Translation request buffer

The translation request buffer stores translation requests from TBUs when all translation manager slots are full. The translation request buffer supports more slots than the translation manager. When correctly configured, this buffer has enough space to store all translation requests that TBUs can issue simultaneously. This buffer therefore prevents the DTI interface from becoming blocked.

PMU

The PMU counts TCU performance-related events and has a configurable number of counters to count the events.

Clock and power control

The TCU has its own clock and power control, that the Q-Channels provide.

Queue manager

The queue manager manages all SMMUv3 queues that are stored in memory:

- Command queues, Secure and Non-secure
- Event queues, Secure and Non-secure
- PRI queue, Non-secure

QTW/DVM interface

The *Queue and Table Walk (QTW)/Distributed Virtual Memory (DVM)* interface is an ACE-Lite +DVM master interface.

Register file

The register file implements the SMMUv3 programmers model, as the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) defines.

DTI interface

The slave DTI interface uses the DTI protocol, typically over AXI4-Stream, to enable the TCU to communicate with a master component. For the MMU-700, the master component is either a TBU or a PCIe master.

Related information

[Interfaces](#) on page 34

[TCU transaction handling](#) on page 70

[TCU prefetch](#) on page 71

[SMMU architectural registers](#) on page 106

[Operation](#) on page 57

3.1.3 DTI interconnect

The TCU and TBUs use a DTI interface to communicate. The DTI interconnect enables the DTI interface to use the AXI4-Stream transport protocol.

The DTI interconnect can connect any components that conform to the AXI4-Stream protocol, as the [AMBA® DTI Protocol Specification](#) defines.

The DTI interconnect contains internal components that are hierarchically composable, that is, they can be connected in different ways to suit your system requirements. For example, within an MMU-700 system, you can use the switch component to combine the DTI interfaces of multiple TBUs into a single DTI interface. You can then connect the combined DTI interface to another DTI interconnect that is closer to the TCU.

The DTI interconnect includes switch, sizer, and register slice components.

Switch

The switch connects multiple DTI masters, such as TBUs, to a DTI slave such as a TCU. The switch implements the following parallel networks:

- For TBU to TCU traffic, a network that connects multiple AXI4-Stream slave interfaces to a single AXI4-Stream master interface
- For TCU to TBU traffic, a network that connects a single AXI4-Stream slave interface to multiple AXI4-Stream master interfaces



The switch does not store any data, and therefore does not require a Q-Channel clock gating interface.

Sizer

The sizer connects channels that have different data widths, enabling different tradeoffs of bandwidth to area. The sizer supports conversion between any of the supported AXI4-Stream data widths:

- 1 byte
- 4 bytes
- 10 bytes
- 20 bytes

The sizer includes a Q-Channel interface to provide clock gating control.

Register slice

Use the register slice to improve timing. The register slice includes a Q-Channel interface to provide clock gating control.

The MMU-700 DTI interconnect components do not include a component to connect different clock and power domains. You can connect DTI interfaces in different clock and power domains by using the *Bidirectional AXI4-Stream* (BAS) configuration of the ADB-400 AMBA® Domain Bridge.

Related information

[Operation](#) on page 57

3.2 Interfaces

The MMU-700 includes interfaces for each of the TCU, TBU, and DTI interconnect components.

The DTI interconnect consists of switch, sizer, and register slice components that you can connect separately, and these components therefore have their own interfaces.

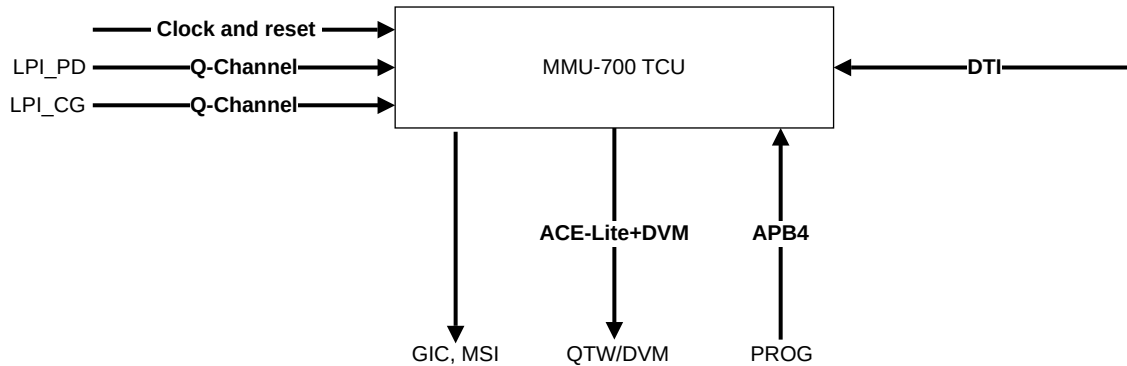
The PMU snapshot interface is common to both TCU and TBU.

3.2.1 TCU interfaces

The MMU-700 TCU includes several master and slave interfaces.

The following figure shows the TCU interfaces.

Figure 3-6: TCU interfaces



Related information

[Distributed Virtual Memory messages](#) on page 69

[Error responses](#) on page 73

[AMBA implementation](#) on page 80

[TCU QTW/DVM interface signals](#) on page 197

3.2.1.1 TCU Queue and Table Walk/Distributed Virtual Memory interface

The *Queue and Table Walk/Distributed Virtual Memory* (QTW/DVM) interface is an ACE-Lite+DVM master interface.

The QTW/DVM interface can issue the following transaction types:

- ReadNoSnoop
- WriteNoSnoop
- ReadOnce
- WriteUnique
- DVM Complete

The QTW/DVM interface uses the write address transaction ID signal **awid_qtw**, and the read address transaction ID signal, **arid_qtw**.

External ID Width = $TCU_ID_WIDTH = \text{MAX}(4, \text{ceil}(\log_2(TCUCFG_PTW_SLOTS)) + 2)$.

The smallest possible TCU_ID_WIDTH value is 4.

See [3.5 Configuration parameters and methodology](#) on page 94.

The following table shows the possible values of **arid_qtw**.

Table 3-1: arid_qtw assignment

Transaction type	arid_qtw[TCU_ID_WIDTH-1:2]	arid_qtw[1:0]
Command Queue walk	Bits [3:2] = 2'b00. If <i>TCU_ID_WIDTH</i> > 4, bits { <i>TCU_ID_WIDTH</i> - 1 :4} are 0.	2'b00
DVM Complete	Bits [3:2] = 2'b01. If <i>TCU_ID_WIDTH</i> > 4, bits { <i>TCU_ID_WIDTH</i> - 1 :4} are 0.	2'b00
Configuration table walk	Indicates the configuration table walk slot that is requesting the configuration table walk	2'b01
Page table walk	Indicates the page table walk slot that is requesting the page table walk	2'b10

The following table shows the possible values of **arid_qtw**.

Table 3-2: awid_qtw assignment

Transaction type	awid_qtw[TCU_ID_WIDTH-1:2]	awid_qtw[1:0]
PRI Queue Write	Bits [3:2] = 2'b01. If <i>TCU_ID_WIDTH</i> > 4, bits { <i>TCU_ID_WIDTH</i> - 1:4} are 0.	2'b00
Event Queue write	Bits [3:2] = 2'b10. If <i>TCU_ID_WIDTH</i> > 4, bits { <i>TCU_ID_WIDTH</i> - 1:4} are 0.	2'b00
MSI write	Bits [3:2] = 2'b11. If <i>TCU_ID_WIDTH</i> > 4, bits { <i>TCU_ID_WIDTH</i> - 1:4} are 0.	2'b00
HTTU Write	Indicates the page table walk slot requesting the HTTU write	2'b11

To support 16-bit *Virtual Machine Identifiers* (VMIDs), the interface provides DVMv8.4 support.

The interface does not issue cache maintenance operations or exclusive accesses.

3.2.1.2 TCU PROG interface

The PROG interface is an AMBA APB4 slave interface. It enables software to program the MMU-700 internal registers and read the *Performance Monitoring Unit* (PMU) registers and the Debug registers.

This interface runs synchronously with the other TCU interfaces.

The applicable address width for this interface depends on the value of *TCUCFG_NUM_TBU*:

- When *TCUCFG_NUM_TBU* = 14, the address width is 21 bits
- When *TCUCFG_NUM_TBU* = 62, the address width is 23 bits

Transactions are *Read-As-Zero, Writes Ignored* (RAZ/WI) when any of the following apply:

- An unimplemented register is accessed
- **PSTRB[3:0]** is not 0b1111 for write transfers
- **PPROT[1]** is not set to 0 for Secure register accesses

For more information, see the [AMBA® APB Protocol Specification](#).

Related information

[TCU programming interface signals](#) on page 199

3.2.1.3 TCU LPI_PD interface

This Q-Channel slave interface manages LPI powerdown for the TCU.

For more information, see the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

Related information

[TCU LPI_PD interface signals](#) on page 200

3.2.1.4 TCU LPI_CG interface

This Q-Channel slave interface enables LPI clock gating for the TCU.

For more information, see the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

Related information

[TCU LPI_CG interface signals](#) on page 201

3.2.1.5 TCU DTI interface

The DTI interface manages communication between the TBUs and the TCU, using the DTI protocol. The DTI protocol can be conveyed over different transport layer mediums, including AXI4-Stream.

The TCU includes a slave DTI interface and each TBU includes a master DTI interface. To permit bidirectional communication, each DTI interface includes one AXI4-Stream master interface and one AXI4-Stream slave interface.

For more information, see the [AMBA® DTI Protocol Specification](#) and the [AMBA® 4 AXI4-Stream Protocol Specification](#).

Related information

[DTI overview](#) on page 57

[TCU DTI interface signals](#) on page 201

3.2.1.6 TCU MSI interface

This interface provides global, per-context, and performance interrupts. A direct *Message Signaled Interrupt* (MSI) connection to a *Generic Interrupt Controller* (GIC) is supported, to avoid complex dependencies in the system.

Related information

[TCU MSI interface signals](#) on page 203

3.2.1.7 TCU SYSCO signaling

The MMU-700 provides a hardware system coherency interface. This master interface permits the TCU to remove itself from a coherency domain in response to an LPI request.

The SYSCO signals include the **syscoreq_qtw** and **syscoack_qtw** handshake signals to enter or exit a coherency domain.

If the **sup_btm** signal is tied LOW, the **syscoreq_qtw** signal is always driven LOW and **syscoack_qtw** is ignored.

Related information

[TCU ELA debug signals](#) on page 206

3.2.1.8 TCU tie-off signals

The TCU tie-off signals enable you to initialize various operating parameters on exit from reset state.

Related information

[TCU tie-off signals](#) on page 205

3.2.1.9 TCU ELA observation interface

This *Embedded Logic Analyzer* (ELA) observation master interface drives the signal group, signal qualifier, and signal clock enable ELA signals to the on-chip ELA module, if present.

When *TCUCFG_USE_ELA_DEBUG* is 0, these signals are tied to 0. See [3.5.3 Translation Control Unit debug configuration parameters](#) on page 97.

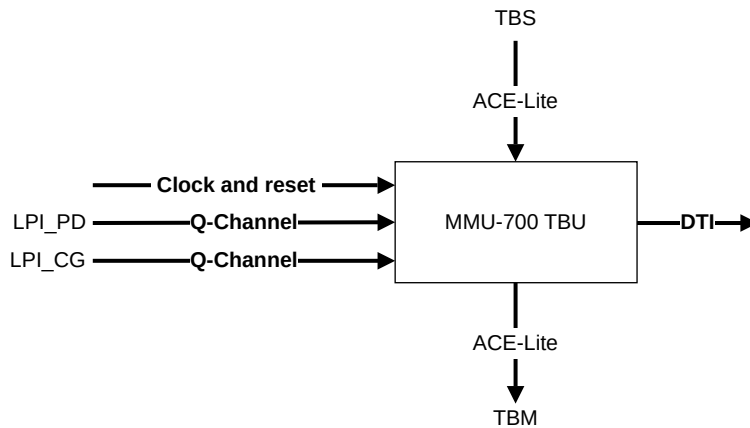
For more information about the interface signals, see [B.1 TCU observation interfaces](#) on page 228.

3.2.2 TBU interfaces

Each MMU-700 TBU includes several master and slave interfaces.

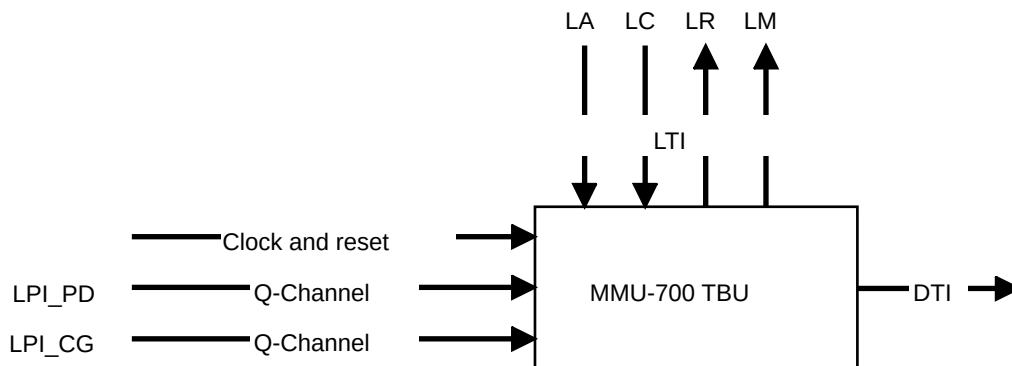
The following figure shows the ACE-Lite TBU interfaces.

Figure 3-7: ACE-Lite TBU interfaces



The following figure shows the LTI TBU interfaces.

Figure 3-8: LTI TBU interfaces



LTI TBUs can have variants with 1, 2, 4, and 8 LTI interfaces. The figure shows a TBU with one LTI interface.

3.2.2.1 ACE-Lite TBU TBS interface

The transaction slave interface, TBS, is an ACE5-Lite interface on which the ACE-Lite TBU receives incoming untranslated memory accesses.

This interface supports a 64-bit address width.

The interface implements optional signals to support the following AXI5 extensions:

- Wakeup_Signals

- Untranslated_Transactions v2
- Cache_Stash_Transactions
- DeAllocation_Transactions
- Atomic_Transactions
- Loopback_Signals
- Poison
- Unique_ID_Support
- Read_Data_Chunking
- CMO_On_Read, Persist_CMO

For more information, see [3.4.2 AMBA implementation](#) on page 80.

The TBS interface supports ACE Exclusive accesses.

If a transaction is terminated in the TBU, the transaction tracker returns the transaction with the user-defined AXI **RUSER** and **BUSER** bits set to 0.

Related information

[Error responses](#) on page 73

[TBU TBS interface signals](#) on page 208

3.2.2.2 ACE-Lite TBU TBM interface

The transaction master interface, TBM, is an ACE5-Lite interface on which the ACE-Lite TBU sends outgoing translated memory accesses.

The AXI ID of a transaction on this interface is the same as the AXI ID of the corresponding transaction on the TBS interface.

This interface supports a 52-bit address width, and *TBUCFG_DATA_WIDTH* defines the data width. See:

- [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101
- [3.5.8 Local Translation Interface Translation Buffer Unit configuration parameters](#) on page 102

This interface can issue read and write transactions until the outstanding transaction limit is reached. The MMU-700 provides parameters that permit you to configure:

- The outstanding read transactions limit
- The outstanding write transactions limit
- The total outstanding read and write transactions limit

The interface implements optional signals to support the following AXI5 extensions:

- Wakeup_Signals

- `Untranslated_Transactions` v2¹
- `Cache_Stash_Transactions`
- `DeAllocation_Transactions`
- `Atomic_Transactions`
- `Loopback_Signals`
- `Ordered Write Observation`
- `Poison`
- `Unique_ID_Support`
- `Read_Data_Chunking`
- `Read_Interleaving_Disabled`²
- `CMO_On_Read`, `Persist_CMO`
- `MPAM_Support`

For more information, see [3.4.2 AMBA implementation](#) on page 80.

When receiving an SLVERR or DECERR response to a downstream transaction, the TBM interface propagates the same response to the TBS interface.

Related information

[Error responses](#) on page 73

[TBU TBM interface signals](#) on page 212

[AMBA implementation](#) on page 80

3.2.2.3 LTI TBU LTI interface

There are four LTI TBU variants, with 1, 2, 4, and 8 LTI interfaces. Each LTI interface is a complete interface, but most of the parameters that you can use to configure an LTI interface are shared between all of them on the same TBU.

The exception to this is the register slice modes on the LA and LR channels. For more information, see the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

The interface contains the following channels:

- | | |
|-----------|---|
| LA | Request channel. Address and attributes that require translation are sent to the TBU. |
| LR | Response channel. Provides the translated address and attributes to the LTI device. |

¹ The TBM interface does not support the *Untranslated_Transactions* property. The TBM contains the **axmmusecsid** and **axmmusid** signals for backwards-compatibility with other SMMU products. These signals are not required for normal operation of the MMU-700 and you can ignore them.

² The BIU supports the *Read_Interleaving_Disabled* property provided that terminated transaction responses are not interleaved.

- LC** Completion channel. LTI devices must provide information about completion to the TBU.
- LM** Link Management channel. Contains:
- **LMOPENREQ**
 - **LMOPENACK**
 - **LMASKCLOSE**
 - **LMACTIVE**

For more information, see the following:

- [3.4.4 Local Translation Interface implementation](#) on page 93
- [3.5 Configuration parameters and methodology](#) on page 94
- [AMBA® LTI Protocol Specification](#)

3.2.2.4 TBU LPI_PD interface

This Q-Channel slave interface manages LPI powerdown for the TBU.

For more information, see the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

Related information

[TBU LPI_PD interface signals](#) on page 216

3.2.2.5 TBU LPI_CG interface

This Q-Channel slave interface enables LPI clock gating for the TBU.

For more information, see the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

Related information

[TBU LPI_CG interface signals](#) on page 216

3.2.2.6 TBU DTI interface

The TBU DTI interface enables the TBU to request translations from the TCU. This interface uses the DTI-TBU protocol for communication between the TBU and the TCU.

The TCU includes a slave DTI interface and each TBU includes a master DTI interface. To permit bidirectional communication, each DTI interface includes one AXI4-Stream master interface and one AXI4-Stream slave interface.

For more information, see the [AMBA® DTI Protocol Specification](#) and the [AMBA® 4 AXI4-Stream Protocol Specification](#).

Related information

[DTI overview](#) on page 57

[TBU DTI interface signals](#) on page 217

3.2.2.7 TBU interrupt interfaces

This interface provides global, per-context, and performance interrupts.

Related information

[TBU tie-off signals](#) on page 43

[TBU interrupt signals](#) on page 217

3.2.2.8 TBU tie-off signals

The TBU tie-off signals enable you to initialize various operating parameters on exit from reset state.

3.2.2.9 TBU ELA observation interface

This *Embedded Logic Analyzer* (ELA) observation master interface drives the signal group, signal qualifier, and signal clock enable ELA signals to the on-chip ELA module, if present.

When `TBUCFG_USE_ELA_DEBUG` is 0, these signals are tied to 0. See [3.5.9 Common Translation Buffer Unit debug configuration parameters](#) on page 103.

For more information about the interface signals, see:

- [B.2 ACE-Lite TBU observation interfaces](#) on page 231
- [B.3 LTI TBU observation interfaces](#) on page 234

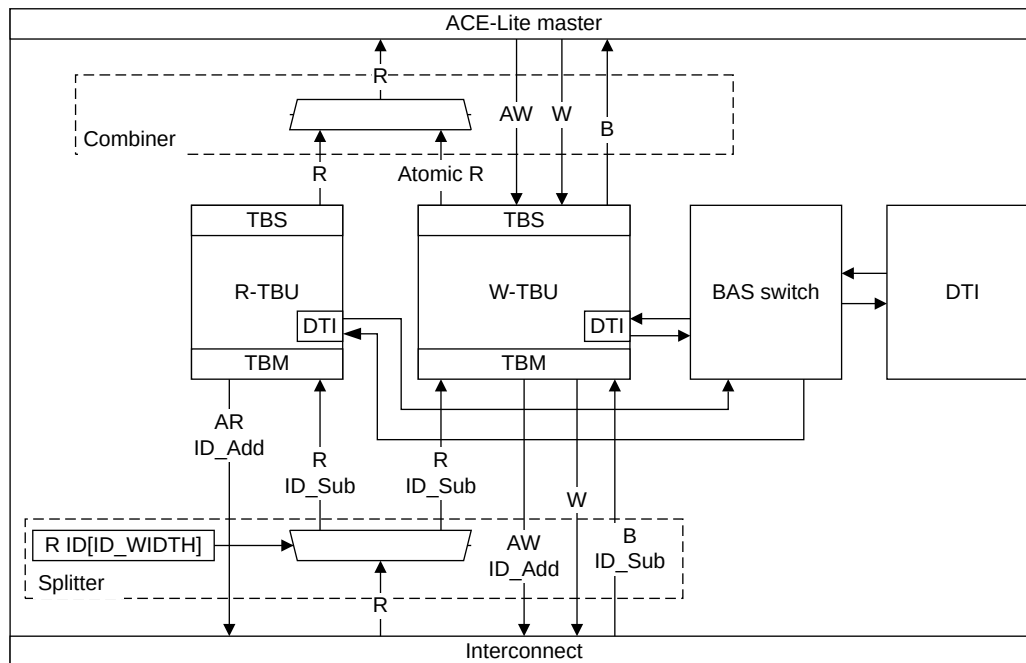
3.2.3 Integration TBU

In the ACE-Lite TBU, either of the address channels, AW and AR, can use the aggregate bandwidth through the TBU individually, meaning that it is not possible to achieve full bandwidth through both channels simultaneously.

The Integration TBU module enables you to achieve greater bandwidth by implementing separate MMU-700 TBU ACE-Lite instances for read transactions (R-TBU) and write transactions (W-TBU). Atomic transactions, which can have responses on both the B and R channels, are routed through the W-TBU.

The following figure shows how the ACE-Lite and DTI channels are connected in the Integration TBU.

Figure 3-9: ACE-Lite and DTI channel connections in the Integration TBU



The figure is not intended to convey detailed schematic information. In particular, it omits information on the LPD-500 instances required for clock and power management and the handling of the PMU and RAS signaling.

3.2.3.1 ACE-Lite transactions

The Integration TBU has a single ACE-Lite slave interface, and a single ACE-Lite master interface between these interfaces.

The behavior is as follows:

- AR channel is routed through the R-TBU
- AW, W, and B channels are routed through the W-TBU
- R channel might be routed through the R-TBU or W-TBU, depending on whether the transaction is atomic or not.

The Integration TBU issues translated ACE-Lite transactions downstream as normal with an extra bit appended to each AxID value to signify whether the transaction is atomic or not.

However, responses on the B channel are routed through the W-TBU, responses on the R channel can be destined for either the R-TBU or the W-TBU. This is because AtomicLoad, AtomicSwap, and AtomicCompare transactions return responses on both the B and the R channels.

The R responses for these atomic transactions must therefore be routed through the W-TBU, alongside the corresponding B response. R responses for other transactions must be routed through the R-TBU. The extra bit of the RID value indicates whether a response on the R channel is routed through the W-TBU or the R-TBU.

3.2.3.2 DTI transactions

Both the R-TBU and W-TBU can issue and receive DTI transactions.

Therefore, an MMU-700 BAS Switch arbitrates between the R-TBU and W-TBU to provide a single DTI interface on the MMU-700 Integration TBU.

3.2.3.3 Interrupts and PMU snapshot interface

Both the R-TBU and W-TBU have their own RAS interrupts, PMU interrupts, and PMU snapshot interfaces.

The Integration TBU includes logic to combine these signals to form a single RAS interrupt, PMU interrupt, and PMU snapshot interface on the Integration TBU.

3.2.3.4 Configuration options

Because only the R-TBU and the W-TBU are configurable, all the configuration parameters of the MMU-700 Integration TBU are for the R-TBU and W-TBU only.

The following table shows the configuration parameters for the R-TBU and the W-TBU.

Table 3-3: R-TBU and the W-TBU configuration parameters

Integration TBU parameter	TBUs that use parameter		R-TBU/W-TBU parameter
	R-TBU	W-TBU	
<i>TBUCFG_SID_WIDTH</i>	Yes	Yes	<i>TBUCFG_SID_WIDTH</i>
<i>TBUCFG_SSID_WIDTH</i>	Yes	Yes	<i>TBUCFG_SSID_WIDTH</i>
<i>TBUCFG_ID_WIDTH</i>	Yes	Yes	<i>TBUCFG_ID_WIDTH</i>
<i>TBUCFG_LOOP_WIDTH</i>	Yes	Yes	<i>TBUCFG_LOOP_WIDTH</i>
<i>TBUCFG_CACHERAM_TYPE</i>	Yes	Yes	<i>TBUCFG_CACHERAM_TYPE</i>
<i>TBUCFG_SLOTRAM_TYPE</i>	Yes	Yes	<i>TBUCFG_SLOTRAM_TYPE</i>
<i>TBUCFG_DATARAM_TYPE</i>	Yes	Yes	<i>TBUCFG_DATARAM_TYPE</i>
<i>TBUCFG_USE_ELA_DEBUG</i>	Yes	Yes	<i>TBUCFG_USE_ELA_DEBUG</i>
<i>TBUCFG_LTI_OG_WIDTH_R</i>	Yes	_3	<i>TBUCFG_LTI_OG_WIDTH</i>
<i>TBUCFG_LTI_OG_WIDTH_W</i>	_4	Yes	<i>TBUCFG_LTI_OG_WIDTH</i>
<i>TBUCFG_XLATE_SLOTS_R</i>	Yes	_3	<i>TBUCFG_XLATE_SLOTS</i>
<i>TBUCFG_XLATE_SLOTS_W</i>	_4	Yes	<i>TBUCFG_XLATE_SLOTS</i>
<i>TBUCFG_DIRECT_IDX</i>	Yes	Yes	<i>TBUCFG_DIRECT_IDX</i>
<i>TBUCFG_MTLB_PARTS</i>	Yes	Yes	<i>TBUCFG_MTLB_PARTS</i>

³ W-TBU is not configurable.

⁴ R-TBU is not configurable.

Integration TBU parameter	TBUs that use parameter		R-TBU/W-TBU parameter
	R-TBU	W-TBU	
<i>TBUCFG_UTLB_DEPTH_R</i>	Yes	_3	<i>TBUCFG_UTLB_DEPTH</i>
<i>TBUCFG_UTLB_DEPTH_W</i>	_4	Yes	<i>TBUCFG_UTLB_DEPTH</i>
<i>TBUCFG_MTLB_DEPTH</i>	Yes	Yes	<i>TBUCFG_MTLB_DEPTH</i>
<i>TBUCFG_MTLB_WAYS</i>	Yes	Yes	<i>TBUCFG_MTLB_WAYS</i>
<i>TBUCFG_MTLB_BANKS</i>	Yes	Yes	<i>TBUCFG_MTLB_BANKS</i>
<i>TBUCFG_HZRD_ENTRIES_R</i>	Yes	_3	<i>TBUCFG_HZRD_ENTRIES</i>
<i>TBUCFG_HZRD_ENTRIES_W</i>	_4	Yes	<i>TBUCFG_HZRD_ENTRIES</i>
<i>TBUCFG_PARTID_WIDTH</i>	Yes	Yes	<i>TBUCFG_PARTID_WIDTH</i>
<i>TBUCFG_MTLB_LKP_SLOTS_R</i>	Yes	_3	<i>TBUCFG_MTLB_LKP_SLOTS</i>
<i>TBUCFG_MTLB_LKP_SLOTS_W</i>	_4	Yes	<i>TBUCFG_MTLB_LKP_SLOTS</i>
<i>TBUCFG_PMU_COUNTERS_R</i>	Yes	_3	<i>TBUCFG_PMU_COUNTERS</i>
<i>TBUCFG_PMU_COUNTERS_W</i>	_4	Yes	<i>TBUCFG_PMU_COUNTERS</i>
<i>TBUCFG_LA_HNDSHK_MODE</i>	Yes	Yes	<i>TBUCFG_LA_HNDSHK_MODE</i>
<i>TBUCFG_LR_HNDSHK_MODE</i>	Yes	Yes	<i>TBUCFG_LR_HNDSHK_MODE</i>
<i>TBUCFG_DATA_WIDTH</i>	Yes	Yes	<i>TBUCFG_DATA_WIDTH</i>
<i>TBUCFG_AWUSER_WIDTH</i>	Yes	Yes	<i>TBUCFG_AWUSER_WIDTH</i>
<i>TBUCFG_WUSER_WIDTH</i>	Yes	Yes	<i>TBUCFG_WUSER_WIDTH</i>
<i>TBUCFG_BUSER_WIDTH</i>	Yes	Yes	<i>TBUCFG_BUSER_WIDTH</i>
<i>TBUCFG_ARUSER_WIDTH</i>	Yes	Yes	<i>TBUCFG_ARUSER_WIDTH</i>
<i>TBUCFG_RUSER_WIDTH</i>	Yes	Yes	<i>TBUCFG_RUSER_WIDTH</i>
<i>TBUCFG_STASH_SUPPORT</i>	_3	Yes	<i>TBUCFG_STASH_SUPPORT</i>
<i>TBUCFG_WBUF_DEPTH</i>	_4	Yes	<i>TBUCFG_WBUF_DEPTH</i>
<i>TBUCFG_LFIFO_DEPTH_R</i> ⁵	Yes	_3	<i>TBUCFG_LFIFO_DEPTH</i>
<i>TBUCFG_LFIFO_DEPTH_W</i>	_4	Yes	<i>TBUCFG_LFIFO_DEPTH</i>
<i>TBUCFG_WOT_DEPTH_W</i>	_4	Yes	<i>TBUCFG_WOT_DEPTH</i>
<i>TBUCFG_ROT_DEPTH_R</i>	Yes	_3	<i>TBUCFG_ROT_DEPTH</i>
<i>TBUCFG_ROT_DEPTH_W</i>	_4	Yes	<i>TBUCFG_ROT_DEPTH</i>
<i>TBUCFG_OT_TRACKER_TYPE</i>	Yes	Yes	<i>TBUCFG_OT_TRACKER_TYPE</i>
<i>TBUCFG_SI_AW_HNDSHK_MODE</i>	_4	Yes	<i>TBUCFG_SI_AW_HNDSHK_MODE</i>
<i>TBUCFG_SI_W_HNDSHK_MODE</i>	_4	Yes	<i>TBUCFG_SI_W_HNDSHK_MODE</i>
<i>TBUCFG_SI_B_HNDSHK_MODE</i>	_4	Yes	<i>TBUCFG_SI_B_HNDSHK_MODE</i>
<i>TBUCFG_SI_AR_HNDSHK_MODE</i>	Yes	_3	<i>TBUCFG_SI_AR_HNDSHK_MODE</i>
<i>TBUCFG_SI_R_HNDSHK_MODE_R</i>	Yes	_3	<i>TBUCFG_SI_R_HNDSHK_MODE</i>
<i>TBUCFG_SI_R_HNDSHK_MODE_W</i>	_4	Yes	<i>TBUCFG_SI_R_HNDSHK_MODE</i>

⁵ We recommend that you always set this parameter to 0

Integration TBU parameter	TBUs that use parameter		R-TBU/W-TBU parameter
	R-TBU	W-TBU	
<i>TBUCFG_MI_AW_HNDSHK_MODE</i>	_4	Yes	<i>TBUCFG_MI_AW_HNDSHK_MODE</i>
<i>TBUCFG_MI_W_HNDSHK_MODE</i>	_4	Yes	<i>TBUCFG_MI_W_HNDSHK_MODE</i>
<i>TBUCFG_MI_B_HNDSHK_MODE</i>	_4	Yes	<i>TBUCFG_MI_B_HNDSHK_MODE</i>
<i>TBUCFG_MI_AR_HNDSHK_MODE</i>	Yes	_3	<i>TBUCFG_MI_AR_HNDSHK_MODE</i>
<i>TBUCFG_MI_R_HNDSHK_MODE_R</i>	Yes	_3	<i>TBUCFG_MI_R_HNDSHK_MODE</i>
<i>TBUCFG_MI_R_HNDSHK_MODE_W</i>	_4	Yes	<i>TBUCFG_MI_R_HNDSHK_MODE</i>

3.2.3.5 Signal descriptions

The following sections list the module interfaces on the block and their purpose.

3.2.3.5.1 Clock and reset

The Integration TBU implements a single clock and a single reset domain. These signals follow the standard Arm® clock and reset guidelines.

The following table shows the clock and reset signals.

Table 3-4: Clock and reset signals

Name	Direction	Width	Description
clk	Input	1	Global clock
resetn	Input	1	Global reset, active-LOW



Clock and reset might logically form part of the buses used for the interfaces in other sections. Because they are physically shared signals, they are not included in the signal tables.

3.2.3.5.2 LPI PD interface

The Integration TBU implements a single LPI Q-Channel for power control. Because the two TBUs each have an LPI Q-Channel for power control, an LPD-500 is used in expander mode to control the power of the two TBUs from the single LPI Q-Channel interface on the Integration TBU.

The following table shows the interface directions and the agents.

Table 3-5: Interface directions and agents

Interface direction	Agent
Producer	External power controller

Interface direction	Agent
Consumer	Integration TBU LPD-500 Q-Channel Expander PD

This interface implements the LPI Q-Channel protocol. See the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

The following table shows the LPI PD interface signals.

Table 3-6: LPI PD interface signals

Name	Direction	Width	Description
qreqn	Input	1	Active-LOW, controller requesting quiescence
qacceptn	Output	1	Active-LOW, device accepting quiescence
qdeny	Output	1	Active-HIGH, device denying quiescence
qactive	Output	1	Active-HIGH, hint that power is required

3.2.3.5.3 LPI CG interface

The Integration TBU implements a single LPI Q-Channel for clock control. Because the two TBUs each have an LPI Q-Channel for clock control, an LPD-500 is used in expander mode to control the two clocks of the two TBUs from the single LPI Q-Channel interface on the Integration TBU.

The following table shows the interface directions and the agents.

Table 3-7: Interface directions and agents

Interface direction	Agent
Producer	External power controller
Consumer	Integration TBU LPD-500 Q-Channel Expander CG

This interface implements the LPI Q-Channel protocol. See the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

The following table shows the LPI CG interface signals.

Table 3-8: LPI CG interface signals

Name	Direction	Width	Description
qreqn	Input	1	Active-LOW, controller requesting quiescence
qacceptn	Output	1	Active-LOW, device accepting quiescence
qdeny	Output	1	Active-HIGH, device denying quiescence
qactive	Output	1	Active-HIGH, hint that power is required

3.2.3.5.4 Slave interface

The Integration TBU implements an AMBA ACE5-Lite interface which receives both read and write transactions from an upstream client device requiring translation services.

See the [AMBA® AXI and ACE Protocol Specification](#).

The following table shows the interface directions and the agents.

Table 3-9: Interface directions and agents

Interface direction	Agent
Producer	Upstream client device
Consumer	Integration TBU Combiner

Several ACE-Lite properties are supported as [3.4.2 AMBA implementation](#) on page 80 describes.

3.2.3.5.5 Master interface

The Integration TBU implements an AMBA ACE5-Lite interface to send the transactions received on the slave interface to the downstream memory system after the SMMU has translated the transactions.

See the [AMBA® AXI and ACE Protocol Specification](#).

The following table shows the interface directions and the agents.

Table 3-10: Interface directions and agents

Interface direction	Agent
Producer	Integration TBU Splitter
Consumer	Downstream memory component

Several ACE-Lite properties are supported as [3.4.2 AMBA implementation](#) on page 80 describes.. The following are minor differences to the port listing:

- AWID is $(TBU_{CFG_ID_WIDTH} + 1)$ bits wide
- ARID is $(TBU_{CFG_ID_WIDTH} + 1)$ bits wide
- BID is $(TBU_{CFG_ID_WIDTH} + 1)$ bits wide
- RID is $(TBU_{CFG_ID_WIDTH} + 1)$ bits wide

See [A Signal descriptions](#) on page 197.

3.2.3.5.6 DTI interface

The Integration TBU implements a single DTI interface to enable communication with the MMU-700 TCU. As both TBUs each have a DTI interface, a BAS Switch is used to multiplex between the two DTI interfaces of the TBU.

See the [AMBA® DTI Protocol Specification](#).

The following table shows the interface directions and the agents.

Table 3-11: Interface directions and agents

Interface direction	Agent
Producer	Integration TBU BAS Switch
Consumer	TCU

The following table shows the DTI interface signals.

Table 3-12: DTI interface signals

Name	Direction	Width	Description
tvalid_dti_up	Input	1	UP payload contains valid data
tready_dti_up	Output	1	UP consumer is ready to receive payload
tdata_dti_up	Input	160	UP interface payload
tkeep_dti_up	Input	20	Indicates which bytes of tdata contain valid data
tlast_dti_up	Input	1	UP payload transfer is complete
tdest_dti_up	Input	6	Indicates the TBU for which the transaction is intended
twakeup_dti_up	Input	1	DTI slave is sending, or attempting to send, something
tvalid_dti_dn	Output	1	DN payload contains valid data
tready_dti_dn	Input	1	DN consumer is ready to receive payload
tdata_dti_dn	Output	160	DN interface payload
tkeep_dti_dn	Output	20	Indicates which bytes of tdata contain valid data
tlast_dti_dn	Output	1	DN payload transfer is complete
tid_dti_dn	Output	6	Indicates the TBU from which the DTI message originates
twakeup_dti_dn	Output	1	A TBU wants to, or is sending, a DTI DN transaction

The Integration TBU contains only two TBUs, but the **tdest/tid** width on the Integration TBU DTI interface is fixed at 6 bits.

3.2.3.5.7 Interrupts

The Integration TBU implements positive edge triggered interrupts. This section describes these interrupts.

The following table shows the interface directions and the agents.

Table 3-13: Interface directions and agents

Interface direction	Agent
Producer	R-TBU, W-TBU
Consumer	Integration TBU

Both the R-TBU and the W-TBU produce their own set of interrupts. These interrupts are OR'd together to produce the external interrupt signals that the following table shows.

Table 3-14: Interrupt signals

Name	Direction	Width	Description
ras_fhi	Output	1	Fault handling RAS interrupt for a contained error from either R-TBU or W-TBU
ras_eri	Output	1	Error recovery RAS interrupt for an uncontained error from either R-TBU or W-TBU
ras_cri	Output	1	Critical error RAS interrupt for an uncontained uncorrected error from either R-TBU or W-TBU
pmu_irpt	Output	1	PMU counter overflow interrupt from either R-TBU or W-TBU

3.2.3.5.8 Tie-offs

The Integration TBU has some configuration options that the static tie-off signals determine. The values of these signals are sampled after reset of the Integration TBU, and so providing the configuration state.

The following table shows the interface directions and the agents.

Table 3-15: Interface directions and agents

Interface direction	Agent
Producer	System integration layer
Consumer	R-TBU, W-TBU

Signals that are appended with *_r* are directed to R-TBU and signals appended with *_w* are directed to W-TBU. Signals without either *_r* or *_w* appended are shared between R-TBU and W-TBU.

The following table shows the tie-off signals.

Table 3-16: Tie-off signals

Integration TBU signal name	Direction	Width	Corresponding R-TBU/W-TBU signal name
ns_sid_high_r	Input	31 - <i>TBUCFG_SID_WIDTH</i>	ns_sid_high
ns_sid_high_w	Input	31 - <i>TBUCFG_SID_WIDTH</i>	ns_sid_high
s_sid_high_r	Input	31 - <i>TBUCFG_SID_WIDTH</i>	s_sid_high
s_sid_high_w	Input	31 - <i>TBUCFG_SID_WIDTH</i>	s_sid_high
max_tok_trans_r	Input	$\log_2(TBUCFG_XLATE_SLOTS_R)$	max_tok_trans
max_tok_trans_w	Input	$\log_2(TBUCFG_XLATE_SLOTS_R)$	max_tok_trans
sec_override	Input	1	sec_override
ecorevnum	Input	4	ecorevnum
utlb_roundrobin_r	Input	1	utlb_roundrobin

Integration TBU signal name	Direction	Width	Corresponding R-TBU/W-TBU signal name
utlb_roundrobin_w	Input	1	utlb_roundrobin
pcie_mode	Input	1	pcie_mode
poison_support	Input	1	poison_support

See [A.2.10 TBU tie-off signals](#) on page 218.

3.2.3.5.9 PMU snapshot interface

The Integration TBU implements a single PMU snapshot interface. The Integration TBU includes logic to merge together the separate PMU snapshot interfaces of the R-TBU and W-TBU.

The following table shows the interface directions and the agents.

Table 3-17: Interface directions and agents

Interface direction	Agent
Producer	System integration layer
Consumer	R-TBU, W-TBU

The following table shows the PMU snapshot interface signals.

Table 3-18: PMU snapshot interface signals

Name	Direction	Width	Description
pmusnapshot_req	Input	1	Request to both R-TBU and W-TBU to capture PMU data.
pmusnapshot_ack	Input	1	Acknowledge that either both R-TBU and W-TBU have captured PMU data or that one of them has.

3.2.3.5.10 DFT interface

The Integration TBU implements a single *Design for Test* (DFT) interface. There is no glue logic required on this interface despite both TBUs having their own interfaces for DFT as the signals used are the same and when in DFT mode, the test controller wants to test all logic simultaneously.

The following table shows the interface directions and the agents.

Table 3-19: Interface directions and agents

Interface direction	Agent
Producer	External Test Controller
Consumer	R-TBU and W-TBU, LPD-500 PD, LPD-500 CG

The following table shows the DFT interface signals.

Table 3-20: DFT interface signals

Name	Direction	Width
dftcgen	Input	1
dftrstdisable	Input	1
dftramhold	Input	1

3.2.3.5.11 MBIST interface

The Integration TBU R-TBU and W-TBU contain RAM instances. A *Memory Built-In Self Test* (MBIST) interface is provided for in-silicon testing of these RAMs.

The following table shows the interface directions and the agents.

Table 3-21: Interface directions and agents

Interface direction	Agent
Producer	External MBIST controller
Consumer	R-TBU, W-TBU

The following table shows the MBIST interface signals.

Table 3-22: MBIST interface signals

Name	Direction	Width
mbistresetn	Input	1
mbistreq	Input	1

3.2.3.5.12 ELA interface

The Integration TBU separate ELA interfaces for R-TBU and W-TBU and then, unlike other interfaces in this block, has separate external interfaces for R-TBU and W-TBU.

The following table shows the interface directions and the agents.

Table 3-23: Interface directions and agents

Interface direction	Agent
Producer	External system
Consumer	R-TBU, W-TBU

The following table shows the ELA interface signals.

Table 3-24: ELA interface signals

Name	Direction	Width
ela_enable_wtbu	Input	1
signalgrp0_wtbu	Output	128

Name	Direction	Width
sigqual0_wtbu	Output	4
sigclken0_wtbu	Output	1
signalgrp1_wtbu	Output	128
sigqual1_wtbu	Output	4
sigclken1_wtbu	Output	1
signalgrp2_wtbu	Output	128
sigqual2_wtbu	Output	4
sigclken2_wtbu	Output	1
signalgrp3_wtbu	Output	128
sigqual3_wtbu	Output	4
sigclken3_wtbu	Output	1
signalgrp4_wtbu	Output	128
sigqual4_wtbu	Output	4
sigclken4_wtbu	Output	1
ela_enable_rtbu	Input	1
signalgrp0_rtbu	Output	128
sigqual0_rtbu	Output	4
sigclken0_rtbu	Output	1
signalgrp1_rtbu	Output	128
sigqual1_rtbu	Output	4
sigclken1_rtbu	Output	1
signalgrp2_rtbu	Output	128
sigqual2_rtbu	Output	4
sigclken2_rtbu	Output	1
signalgrp3_rtbu	Output	128
sigqual3_rtbu	Output	4
sigclken3_rtbu	Output	1
signalgrp4_rtbu	Output	128
sigqual4_rtbu	Output	4
sigclken4_rtbu	Output	1

3.2.4 DTI interconnect interfaces

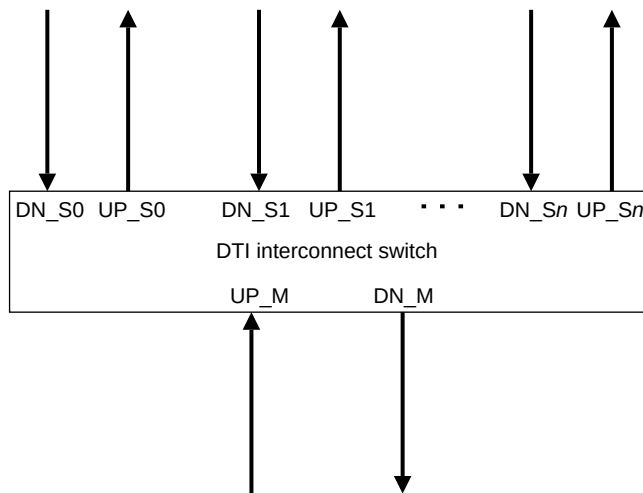
The DTI interconnect includes interfaces for each of the switch, sizer, and register slice components.

3.2.4.1 DTI interconnect switch interfaces

The DTI interconnect switch component includes dedicated interfaces.

The following figure shows the DTI interconnect switch interfaces.

Figure 3-10: DTI interconnect switch interfaces



The following table provides more information about the switch interfaces.

Table 3-25: DTI interconnect switch interfaces

Interface	Interface type	Protocol	Description
DN_Sn	Slave	AXI4-Stream	Slave downstream interface. One DN_Sn interface is present for each slave interface.
UP_Sn	Master		Slave upstream interface. One UP_Sn interface is present for each slave interface.
DN_M	Master		Master downstream interface
UP_M	Slave		Master upstream interface



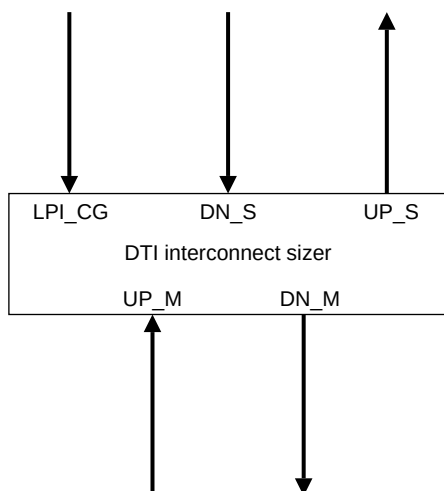
The interconnect switch does not store any data, and therefore does not require a Q-Channel clock-gating interface.

3.2.4.2 DTI interconnect sizer interfaces

The DTI interconnect sizer component includes dedicated interfaces.

The following figure shows the DTI interconnect sizer interfaces.

Figure 3-11: DTI interconnect sizer interfaces



The following table provides more information about the sizer interfaces.

Table 3-26: DTI interconnect sizer interfaces

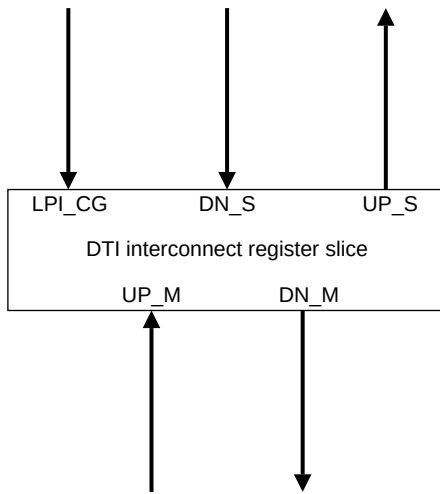
Interface	Interface type	Protocol	Description
LPI_CG	Slave	Q-Channel	Clock gating interface
DN_S	Slave	AXI4-Stream	Slave downstream interface
UP_S	Master		Slave upstream interface
DN_M	Master		Master downstream interface
UP_M	Slave		Master upstream interface

3.2.4.3 DTI interconnect register slice interfaces

The DTI interconnect register slice component includes dedicated interfaces.

The following figure shows the DTI interconnect register slice interfaces.

Figure 3-12: DTI interconnect register slice interfaces



The following table provides more information about the register slice interfaces.

Table 3-27: DTI interconnect register slice interfaces

Interface	Interface type	Protocol	Description
LPI_CG	Slave	Q-Channel	Clock gating interface
DN_S	Slave	AXI4-Stream	Slave downstream interface
UP_S	Master		Slave upstream interface
DN_M	Master		Master downstream interface
UP_M	Slave		Master upstream interface

3.3 Operation

This section provides information about the operation of the MMU-700 features.

3.3.1 DTI overview

In an MMU-700-based system, the AMBA® DTI protocol defines the standard for communicating with a TCU.

The AMBA® DTI protocol includes both:

- DTI-TBU protocol, for communication between a TBU and a TCU
- DTI-ATS protocol, for communication between a PCIe Root Complex and a TCU

The DTI protocol is a point-to-point protocol. Each channel consists of a link, a DTI master, and a DTI slave. The DTI masters in the respective protocols are:

- The TBU, in the DTI-TBU protocol
- The PCIe Root Complex, in the DTI-ATS protocol

The DTI slave in both DTI-TBU and DTI-ATS is the TCU.

DTI masters and slaves communicate using defined DTI messages. The DTI protocol defines the following message groups:

- Page request
- Register access
- Translation request
- Connection and disconnection
- Invalidation and synchronization

The DTI_TBU_CONDIS_REQ message initiates a TBU connection or disconnection handshake. The TBU uses this message to connect to the TCU. During connection, the TBU can specify the number of requested translation tokens. The DTI master uses the TOK_TRANS_REQ field to request translation tokens. For the TBU, the **max_tok_trans** signal defines the number of translation tokens that the TBU requests.

The TBU uses the TOK_INV_GNT field to grant invalidation tokens. The TBU grants only one invalidation token, and the TCU is only capable of issuing one invalidate message at a time.

A DTI master uses a DTI_TBU_CONDIS_REQ or a DTI_ATS_CONDIS_REQ message to initiate a connection handshake. If the master provides a **TID** value that is greater than the maximum supported **TID** that *TCUCFG_NUM_TBU* defines, the slave sends a Connect Deny message.

A translation request to the TCU where $\text{StreamID} \geq 2^{24}$ results in a fault and an SMMUv3 C_BAD_STREAMID event. If the TBU receives an invalidation request where $\text{StreamID} \geq 2^{24}$, any comparisons with a StreamID value fail. No TLB entries are invalidated, but other effects that do not consider the supplied StreamID occur as normal.



- The TBU never generates translation requests with $\text{StreamID} \geq 2^{24}$
 - The TCU never generates invalidation requests with $\text{StreamID} \geq 2^{24}$
-

For more information, see the [AMBA® DTI Protocol Specification](#).

3.3.2 Performance Monitoring Unit

The MMU-700 includes a PMU for the TCU and a PMU for each TBU. The PMU events and counters indicate the runtime performance of the MMU-700.

The MMU-700 includes logic to gather various statistics on the operation of the MMU during runtime, using events and counters. These events, which the SMMUv3 architecture defines,

provide useful information about the behavior of the MMU. You can use this information when debugging or profiling traffic.

3.3.2.1 SMMUv3 architectural performance events

Both the TCU and the TBU implement performance events that the SMMUv3 Performance Monitor extension defines.

The SMMU_PMCG_SMRO register can filter some events so that only events with a particular StreamID are counted. This event filtering includes:

- Speculative transactions and translations
- Transactions and translations that result in a terminated transaction or a translation fault

The following table shows the architecturally defined MMU-700 TCU performance events.

Table 3-28: SMMUv3 performance events for the TCU

Event	Event ID	SMMU_PMCG_SMRO filterable	Description
Clock cycle	0x0	No	Counts clock cycles. Cycles where the clock is gated after a clock Q-Channel handshake are not counted.
Transaction	0x1	Yes	Counts translation requests that originate from a DTI-TBU or DTI-ATS master
TLB miss caused by incoming transaction or translation request	0x2	Yes	Counts translation requests where the translation walks new translation table entries
Configuration cache miss caused by transaction or translation request	0x3	Yes	Counts translation requests where the translation walks new configuration table entries
Translation table walk access	0x4	Yes	Counts translation table walk accesses
Configuration structure access	0x5	Yes	Counts configuration table walk accesses
PCIe ATS Translation Request received	0x6	Yes	Counts translation requests that originate from a DTI-ATS master

The following table shows the architecturally defined MMU-700 TBU performance events.

Table 3-29: SMMUv3 performance events for the TBU

Event	Event ID	SMMU_PMCG_SMRO filterable	Description
Clock cycle	0x0	No	Counts clock cycles. Cycles where the clock is gated after a clock Q-Channel handshake are not counted.
Transaction	0x1	Yes	Counts transactions that are issued on the TBM interface
TLB miss caused by incoming transaction or translation request	0x2	Yes	Counts non-speculative translation requests that are issued to the TCU
PCIe ATS Translation Request received	0x7	Yes	Counts ATS-translated transactions that are issued on the TBM interface

For more information, see the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#).

3.3.2.2 MMU-700 TCU events

The MMU-700 PMU can be configured to monitor a range of **IMPLEMENTATION DEFINED** TCU performance events.

The SMMU_PMCG_SMRO register can filter some TCU performance events so that only events with a particular StreamID are counted. This event filtering includes:

- Speculative transactions and translations
- Transactions and translations that result in a terminated transaction or a translation fault

The following table shows the TCU performance events.

Table 3-30: MMU-700 TCU performance events

Event	Event ID	SMMU_PMCG_SMRO filterable	Description
S1LOWC lookup	0x80	Yes	Counts translation requests that access the S1LOWC walk cache
S1LOWC miss	0x81	Yes	Counts translation requests that access the S1LOWC walk cache and do not result in a hit
S1L1WC lookup	0x82	Yes	Counts translation requests that access the S1L1WC walk cache
S1L1WC miss	0x83	Yes	Counts translation requests that access the S1L1WC walk cache and do not result in a hit
S1L2WC lookup	0x84	Yes	Counts translation requests that access the S1L2WC walk cache
S1L2WC miss	0x85	Yes	Counts translation requests that access the S1L2WC walk cache and do not result in a hit
S1L3WC lookup	0x86	Yes	Counts translation requests that access the S1L3WC walk cache
S1L3WC miss	0x87	Yes	Counts translation requests that access the S1L3WC walk cache and do not result in a hit
S2LOWC lookup	0x88	Yes	Counts translation requests that access the S2LOWC walk cache
S2LOWC miss	0x89	Yes	Counts translation requests that access the S2LOWC walk cache and do not result in a hit
S2L1WC lookup	0x8A	Yes	Counts translation requests that access the S2L1WC walk cache
S2L1WC miss	0x8B	Yes	Counts translation requests that access the S2L1WC walk cache and do not result in a hit
S2L2WC lookup	0x8C	Yes	Counts translation requests that access the S2L2WC walk cache
S2L2WC miss	0x8D	Yes	Counts translation requests that access the S2L2WC walk cache and do not result in a hit

Event	Event ID	SMMU_PMCGR_SMR0 filterable	Description
S2L3WC lookup	0x8E	Yes	Counts translation requests that access the S2L3WC walk cache
S2L3WC miss	0x8F	Yes	Counts translation requests that access the S2L3WC walk cache and do not result in a hit
WC read	0x90	Yes	Counts reads from the walk cache RAMs, excluding reads that invalidation requests cause Note: A single walk cache lookup might result in multiple RAM reads. This behavior permits contiguous entries to be located.
Buffered translation	0x91	Yes	Counts translations that are written to the translation request buffer because either all the configuration table walk slots or all the page table walk slots are occupied
CC lookup	0x92	Yes	Counts lookups into the configuration cache
CC read	0x93	Yes	Counts reads from the configuration cache RAMs, excluding reads that invalidation requests cause Note: A single cache lookup might result in multiple RAM reads. This behavior permits contiguous entries to be located.
CC miss	0x94	Yes	Counts lookups into the configuration cache that result in a miss
Speculative translation	0xA0	Yes	Counts translation requests that are marked as Speculative



Note

A single DTI translation request might correspond to multiple translation request events in either of the following circumstances:

- A translation results in a stall fault event and is restarted
- If a translation results in a stall fault event, because the Event queue is full, the translation is retried when an Event queue slot becomes available

3.3.2.3 MMU-700 TBU events

The MMU-700 PMU can be configured to monitor a range of **IMPLEMENTATION DEFINED** TBU performance events.

The SMMU_PMCGR_SMR0 register can filter the TBU performance events so that only events with a particular StreamID are counted. This event filtering includes:

- Speculative transactions and translations
- Transactions and translations that result in a terminated transaction or a translation fault

The following table shows the TBU performance events.

Table 3-31: MMU-700 TBU performance events

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
Main TLB lookup	0x80	Yes	Counts Main TLB lookups
Main TLB miss	0x81	Yes	Counts translation requests that miss in the Main TLB
Main TLB read	0x82	Yes	Counts once per access to the Main TLB RAMs, excluding reads that invalidation requests cause Note: A transaction might access the Main TLB multiple times to look for different page sizes.
MicroTLB lookup	0x83	Yes	Counts MicroTLB lookups
MicroTLB miss	0x84	Yes	Counts translation requests that miss in the MicroTLB
Slots full	0x85	No	Counts once per cycle when all slots are occupied and not ready to issue transactions downstream. This Secure event is visible only when the SMMU_PMCG_SCR.SO bit is set to 1.
Out of translation tokens	0x86	No	Counts once per cycle when a translation request cannot be issued because all translation tokens are in use. This Secure event is visible only when the SMMU_PMCG_SCR.SO bit is set to 1.
Write data buffer full	0x87	No	Counts once per cycle when a transaction is blocked because the write data buffer is full. This Secure event is visible only when the SMMU_PMCG_SCR.SO bit is set to 1.
DCMO downgrade	0x8B	Yes	For the ACE-Lite TBU, counts when either: <ul style="list-style-type: none"> A MakeInvalid transaction on the TBS interface is output as CleanInvalid on the TBM interface A ReadOnceMakeInvalid transaction on the TBS interface is output as ReadOnceCleanInvalid on the TBM interface For the LTI TBU, counts once per cycle when an LTI DCMO or R-DCMO transaction on the LA channel is responded to with a downgrade on the LR channel

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
Stash fail	0x8C	Yes	<p>For the ACE-Lite TBU, counts when either:</p> <ul style="list-style-type: none"> A WriteUniquePtlStash or WriteUniqueFullStash transaction on TBS is output as a WriteNoSnoop or WriteUnique transaction on the TBM interface A StashOnceShared or StashOnceUnique transaction on the TBS interface has a valid translation, but is terminated in the TBU <p>For the LTI TBU, counts once whenever either an:</p> <ul style="list-style-type: none"> LTI WDCP transaction on the LA channel is downgraded as W on the LR channel. LTI DCP transaction on the LA channel that is responded to as FaultRAZWI on the LR channel is counted. This can be because of: <ul style="list-style-type: none"> Memory attributes or DCP, R, W, or X permission check failure in the <i>Translation Lookaside Buffer Unit</i> (TLBU) DTI fault response with Non-Abort <p>The transaction that is responded to with FaultAbort because of DTI StreamDisable or GlobalDisable is not counted</p> <p>Note: A StashOnceShared or StashOnceUnique transaction that is terminated because of a StreamDisable or GlobalDisable translation response does not cause this event to count</p>
LTI port slots full	0xD0-0xD7	Yes	<p>LTI port event (0xD0 + N) corresponds to LTI port N.</p> <p>Counts once per cycle when the slots that are allocated to the LTI port are all occupied and not ready to issue downstream.</p>
LTI port out of translation tokens	0xE0-0xE7	Yes	<p>LTI port event (0xD0 + N) corresponds to LTI port N.</p> <p>Counts once per cycle when a translation request cannot be issued for an LTI port because all its allocated translation tokens are in use.</p>

3.3.2.4 SMMUv3 PMU register architectural options

The SMMUv3 architecture defines the *Performance Monitor Counter Group* (PMCG) configuration register, SMMU_PMCG_CFGR. An MMU-700 implementation assumes fixed values for SMMU_PMCG_CFGR, and these values define behavioral aspects of the implementation.

The following table shows the SMMU_PMCG_CFGR register options that the MMU-700 TCU and TBU use.

Table 3-32: MMU-700 SMMU_PMCG_CFGR register architectural options

Field	Default value	Description for default value
SID_FILTER_TYPE	1	A single StreamID filter applies to all PMCG counters
CAPTURE	1	Capture of counter values into SVRn registers is supported
MSI	0	The counter group does not support <i>Message Signaled Interrupts</i> (MSIs)
RELOC_CTRS	1	The PMCG registers are relocated to page 1 of the PMU address map
SIZE	0x31	The counter group implements 32-bit counters
MPAM	0	<i>Memory System Resource Partitioning and Monitoring</i> (MPAM)

Field		Default value	Description for default value
NCTR	NCTR for the TCU	<i>TCUCFG_PMU_COUNTERS</i> - 1	The counter group includes <i>TCUCFG_PMU_COUNTERS</i> counters. See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.
	TCTR for the TBU	<i>TBUCFG_PMU_COUNTERS</i> - 1	The counter group includes <i>TBUCFG_PMU_COUNTERS</i> counters. See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Related information

[MMU-700 memory map](#) on page 110

3.3.2.5 PMU snapshot interface

The *Performance Monitoring Unit* (PMU) snapshot interface is included on the TCU and on each TBU. You can use this asynchronous interface to initiate a PMU snapshot. A simultaneous snapshot of each counter register is created and copied to the respective SMMU_PMCG_SVRn register.

The PMU snapshot sequence is a 4-phase handshake. Both **pmusnapshot_req** and **pmusnapshot_ack** are LOW after reset. A snapshot occurs on the rising edge of **pmusnapshot_req**, and is equivalent to writing the value 1 to SMMU_PMCG_CAPR.CAPTURE.

The **pmusnapshot_req** signal is sampled using synchronizing registers. A register drives **pmusnapshot_ack** so that the connected component can sample the signal asynchronously.

Related information

[RAS implementation](#) on page 66

[TCU PMU snapshot interface signals](#) on page 200

[TBU PMU snapshot interface signals](#) on page 215

3.3.3 Multiple LTI interface TBU

There are variants of the LTI TBU that have 1, 2, 4, or 8 LTI interfaces. The traffic through these interfaces is multiplexed together and they share a single translation manager, TLB, and DTI interface.

Registers are provided to enable you to set limits on the maximum proportion of these shared resources any individual LTI interface can use. This limiting applies to entries in the translation managing structure and DTI translation tokens. See the [4 Programmers model](#) on page 105.

A given LTI interface is not permitted to issue more translation requests into the TBU if it already uses the limit or more than the limit of any of the types of managed resource. If the limits are changed while an interface has outstanding translation requests, and this results in the interface using more than the new permitted proportion, it is unable to issue translation requests until the resource usage returns to below the permitted maximum. To avoid deadlock, each LTI interface must be guaranteed to be able to have at least 2 outstanding DTI requests and 2 outstanding LTI translation requests (one on each virtual channel on the LTI interface).

As a result, when there is more than one LTI interface, the maximum number of translation requests that any individual LTI interface might have outstanding is less than the total number possible across all interfaces. An LTI transaction is considered to be outstanding for these purposes for its full life according to the [AMBA® LTI Protocol Specification](#). An LTI transaction is only considered to be using a DTI translation token until the translation manager determines that it is not required to perform a DTI translation request in the future. This determination can be because it has completed a DTI translation request or because it is not necessary to perform one because it gets a hit response to its TLB lookup or it receives a hazard response from another transaction ahead of it.

In all cases, a DTI translation token is required for a transaction to enter the LTI TBU because at the point that it enters the TBU, it cannot be known whether it requires a DTI translation request or not. Because the LTI TBU cannot guarantee deadlock freedom unless it receives at least 2 DTI credits per LTI interface, it fails to exit the Q_STOPPED LPI state if the **max_trans_tok** tie off signal does not indicate at least that number of tokens should be requested at connection. Similarly, on connection, if the TCU does not grant the minimum number of credits, then the TBU does not unfence its LA interface, and instead initiates a DTI disconnection when this connection condition is detected.

3.3.4 Main TLB direct indexing and main TLB direct partitioning

Main TLB direct indexing can help your system to meet real-time translation requirements by enabling the MMU-700 to manage *Main TLB* (MTLB) entries externally to the TBU.



If you use the Main TLB direct indexing and Main TLB direct partitioning features, MPAM is not valid.

Direct indexing enables real-time translation requirements to be met, as follows:

- It can be guaranteed that different streams do not overwrite prefetched entries
- The MTLB can be partitioned into different sets of entries that different streams use

If you configure your system to not use MTLB direct indexing, you can select MTLB direct partitioning. MTLB direct partitioning has similar behavior, but only the most significant TLB index bits are provided, and the other bits are generated internally.

Direct indexing is enabled for a TBU when `TBUCFG_DIRECT_IDX = 1`.

When `TBUCFG_DIRECT_IDX = 1`, or when an MTLB is partitioned, the width of the **AxUSER** signals on the TBS interface is extended to convey the indexing information that is required for MTLB direct indexing or MTLB direct partitioning.

Indexing information is sent using the **latlbloc** signal for the LTI TBU. See [B.3 LTI TBU observation interfaces](#) on page 234.



The table shows the extended bits in the order MSB first.

Table 3-33: Extended aruser_s and awuser_s bits for MTLB partitioning

Field name	Width	Description
mtlbidx	When direct indexing is enabled, the width of this field is $\log_2(TBUCFG_MTLB_DEPTH) - \log_2(T \setminus BUCFG_MTLB_WAYS)$. When direct indexing is not enabled, the width of this field is 0.	MTLB index
mtlbway	When direct indexing is enabled, the width of this field is $\log_2(TBUCFG_MTLB_WAYS)$. When direct indexing is not enabled, the width of this field is 0.	MTLB way
mtlbpart	$\log_2(TBUCFG_MTLB_PARTS)$	MTLB partition
-	$TBUCFG_AWUSER_WIDTH$ for awuser_s . $TBUCFG_ARUSER_WIDTH$ for aruser_s .	Regular AxUSER signals

If an MTLB is partitioned:

- The MTLB size is multiplied by $TBUCFG_MTLB_PARTS$
- The mtlbpart field defines the $\log_2(TBUCFG_MTLB_PARTS)$ most significant index bits

When direct indexing is enabled for a TBU:

- Lookups and updates to the MTLB use the mtlbidx field
- Updates to the MTLB use the way that mtlbway specifies
- Lookups to the MTLB operate on all ways simultaneously

To maintain system performance, we recommend that you disable DVM invalidation on TBUs on which direct indexing is enabled. Disable DVM invalidation by setting the appropriate TCU_NODE_CTRLn.DIS_DVM bit. See [4.7.1 TCU_CTRL register](#) on page 123.

3.3.5 RAS implementation

Reliability, Availability, and Serviceability (RAS) features enable SRAM corruption to be detected and corrected, optionally generating interrupts into the system. All MMU-700 RAM-based caches support RAS error detection and correction. This section describes RAS implementation in the CoreLink™ MMU-700 System Memory Management Unit.

MMU-700 implements a combination of *Single-Error-Correct-Double-Error-Detect* (SECEDED) and *Double-Error-Detect* (DED) error correction mechanisms.

SECDED is used in RAMs where a double error usually means that the SMMU cannot contain this error and must raise a *Critical Error Interrupt*. DED is used in TLB TAGS or DATA, where a single or double error can be recovered by fetching data from System Memory.

Also, the SMMU raises *Fault Handling Interrupts* (FHIs), *Error Recovery Interrupts* (ERIs), and *CRITICAL Error Interrupts* (CRIs) based on a contained error or uncontained error respectively.

The following table shows the RAMs in MMU-700, and actions that are taken when errors occur.

Table 3-34: RAM RAS error sources

RAM name	Error correction and detection mechanism	RAS error triggered		RAS interrupts triggered
BIU WDB ROBUFF_D	SEC	CE		FHI
	DED	Poison supported:	DE	FHI
		Poison not supported:	UE (UC)	ERI, FHI, and CRI
BIU WDB ROBUFF_C	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
BIU WDB ROBUFF_P	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU OGQ	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU UOQ	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU DTIQ	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU REQ	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU RSP	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU LB	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU HLB_ENTRY LEFT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU HLB_ENTRY RIGHT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU HLB_PTR LEFT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU HLB_PTR RIGHT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU DCU MTLB PLIM	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU DCU MTLB PCNT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU DCU MTLB REPL	SEC	CE		FHI

RAM name	Error correction and detection mechanism	RAS error triggered	RAS interrupts triggered
	DED	UE (UC)	FHI, ERI, and CRI
TLBU DCU MTLB TAGS	SED	CE	FHI
	DED	CE	FHI
TLBU DCU MTLB DATA	SED	CE	FHI
	DED	CE	FHI
TMU TWB BSU	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU HZU PTR	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU TWB WMB LKP STATUS	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU TWB WMB WLK STATUS	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU TWB WMB SCRATCH	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU HTTU RAM	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC PLIM	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC PCNT	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC REPL	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC TAGS	SED	CE	FHI
	DED	CE	FHI
TMU WCB MWC DATA	SED	CE	FHI
	DED	CE	FHI
TMU CCB MCC PLIM	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU CCB MCC PCNT	SEC	CE	FHI, ERI, and FHI on DED
	DED	UE (UC)	FHI, ERI, and CRI
TMU CCB MCC REPL	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU CCB MCC TAGS	SED	CE	FHI
	DED	CE	FHI
TMU CCB MCC DATA	SED	CE	FHI
	DED	CE	FHI

3.3.6 Quality of Service

You can program the TCU with a priority level for each LTI TBU interface. The priority level is applied to every translation from that TBU interface.

The TCU uses this priority level to:

- Arbitrate between translations that are waiting in the translation request buffer when translation manager slots become available
- Determine the AXI **AxQOS** value for translation table walks and configuration table walks that the TCU issues on the QTW/DVM interface

The arbiters contain starvation avoidance mechanisms to prevent transactions from being stalled indefinitely.

The TBU does not implement any prioritization between transactions. However, if a TBU has multiple LTI ports, the resources that a given port uses can be capped using the [4.14.2 TBU_LTI_PORT_RESOURCE_LIMIT register](#) on page 168 . In most use cases, this limiting functionality, combined with the per-LTI interface priority level provides the required QoS management functionality. However, sometimes it might be necessary to use separate TBUs for masters with different QoS requirements.

Related information

[TCU_NODE_CTRLn register](#) on page 128

[TCU_QOS register](#) on page 125

3.3.7 Distributed Virtual Memory messages

The QTW/DVM interface supports *Distributed Virtual Memory* (DVM) messages. The MMU-700 supports DVMv8.4.

The interface supports DVM transactions of message types TLB Invalidate and Synchronization. The interface accepts all other DVM transaction message types, and sends a snoop response, but otherwise ignores such transactions.

Tie the **sup_btm** input signal HIGH when the system supports Broadcast TLB Maintenance.

When Broadcast TLB Maintenance is supported, you can use SMMU_CR2 and SMMU_S_CR2 to control how the SMMU handles TLB Invalidate operations as follows:

SMMU_CR2.PTM = 0	Non-secure TLB Invalidate operations are applied to the TLBs
SMMU_CR2.PTM = 1	Non-secure TLB Invalidate operations have no effect
SMMU_S_CR2.PTM = 0	Secure TLB Invalidate operations are applied to the TLBs
SMMU_S_CR2.PTM = 1	Secure TLB Invalidate operations have no effect



When **sup_btm** is tied HIGH, the reset value of SMMU_CR2.PTM and SMMU_S_CR2.PTM is 1.



Although TLB Invalidate operations have no effect when PTM = 1, the QTW/DVM interface still returns the appropriate response.

The QTW/DVM interface might receive DVM Sync transactions without receiving a DVM TLB Invalidate transaction, or when the PTM bits have masked a TLB Invalidate. If no DVM TLB Invalidate operations have occurred since the most recent DVM Sync transaction, subsequent DVM Sync transactions result in an immediate DVM Complete transaction. This behavior ensures that the TCU does not affect system DVM performance unless TLB Invalidate operations are performed.

The ACE-Lite interface allocates the access permissions and shareability of DVM Complete transactions as follows:

- **ARPROT** = 0b000, indicating Unprivileged, Secure, Data access
- **ARDOMAIN** = 0b10, indicating Outer Shareable

For a DVM Operation or DVM Sync request on the **AC** channel, the snoop response signal **CRRESP[4:0]** is always set to 0b00000.

Related information

[SMMU architectural registers](#) on page 106

3.3.8 TCU transaction handling

The transaction width, burst length, and transfer size that the TCU supports depend on the transaction type.

The following table shows the TCU support for read transactions.

Table 3-35: TCU support for read transactions

Transaction type	Transaction width	ARID[n:2]	ARID[1:0]
Level 1 Stream table or Level 1 Context Descriptor table lookup	64-bit	Config slot number	2'b01
Stream table or Context Descriptor table lookup	512-bit	Config slot number	2'b01
Translation table lookup	64-bit	PTW slot number	2'b10
Command queue read	128-bit	All 0	2'b00
DVM Complete	-	Bit 2 is 1 and all other bits are 0	2'b00

DVM Complete transactions are always one beat of full data width.

Command queue reads and DVM Complete transactions are independent of translation slots. Therefore, the maximum number of read transactions that the TCU can issue at any time is $TCUCFG_PTW_SLOTS + 2$.

The following table shows the TCU support for write transactions.

Table 3-36: TCU support for write transactions

Transaction type	Transaction width	AWID[n:2]	AWID[1:0]
Event queue write	256-bit	Bits [3:2] = 2'b10. If $TCU_ID_WIDTH > 4$, bits { $TCU_ID_WIDTH - 1:4$ } are 0.	2'b00
PRI queue write	128-bit	Bits [3:2] = 2'b01. If $TCU_ID_WIDTH > 4$, bits { $TCU_ID_WIDTH - 1:4$ } are 0.	2'b00
Message Signaled Interrupt (MSI)	32-bit	Bits [3:2] = 2'b11. If $TCU_ID_WIDTH > 4$, bits { $TCU_ID_WIDTH - 1:4$ } are 0.	2'b00
HTTU write	128-bit	Indicates the page table walk slot requesting the HTTU write	2'b11

Only one write transaction can be outstanding at a time.

All read and write transactions are aligned to the transaction size.

3.3.9 TCU prefetch

The TCU can prefetch translations on a per-context basis to improve translation performance for real-time masters that access memory linearly. If TCU prefetch is enabled, a second translation request occurs after the original request, and is initiated and terminated entirely within the TCU. This second translation request is regarded as the *prefetch* because it is an advance request of the next translation that is expected to be requested. This second request is Speculative and is used to allocate into the caches of the TCU.

Software can enable TCU prefetch for a particular translation context by programming the *Stream Table Entry* (STE). Bits [121:120] are **IMPLEMENTATION DEFINED** in the SMMUv3 architecture. See the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#).

The MMU-700 uses these bits for the PF field as follows:

PF, bits [121:120]

This field determines whether prefetch is enabled or disabled for the translation context that this STE defines as follows:

- 0b00** Prefetching disabled
- 0b01** Reserved
- 0b10** Forward prefetching
- 0b11** Backward prefetching

Prefetching disabled

TCU prefetch does not occur

Reserved

Reserved values must not be used

Forward prefetching

The address to be prefetched is the first address following the end of the translation range, as DTI_TBU_TRANS_RESP.TRANS_RNG indicates

Backward prefetching

The address to be prefetched is the last address before the beginning of the translation range, as DTI_TBU_TRANS_RESP.TRANS_RNG indicates

Whenever a miss occurs in the MicroTLB and Main TLB of the TBU, the TBU sends a translation request to the TCU. If the STE for the translation is programmed to enable prefetch, each translation request to the TCU can also potentially result in a prefetch that occurs after the original request is complete. When each incoming translation request completes its translation in the TCU, the STE.PF field indicates whether TCU prefetch is enabled. If TCU prefetch is enabled, a second translation request, the prefetch request, is then issued into the same TCU translation slot. This prefetch request is Speculative, and only allocates into the TCU walk caches. A translation response for the prefetch is not returned to the TBU.

When the TCU handles each incoming translation request from the TBU, translation table walks might or might not occur depending on whether there is a hit in each level of walk cache that is looked up. Translation table walks also might or might not occur for the subsequent prefetch request. The number of memory accesses that are performed for this prefetch are unrelated to the number of memory accesses that are performed for the original translation request.

Consider the following examples:

1. An incoming translation request might hit in the lowest level of walk cache, but the subsequent prefetch request might still require at least one translation table walk to memory.
2. The original translation request might require multiple translation table walks, but the subsequent prefetch request might hit in the lowest level of walk cache and not require any memory accesses. If the prefetch request hits in the lowest level of walk cache, then the walk caches are not updated and no memory accesses are performed.



The walk cache uses a round-robin replacement policy.

The prefetch can only occur when the original request is complete irrespective of whether translation table walks were required. Waiting for completion of the original request means that by the time it becomes possible for the prefetch to be initiated, the TCU might have already received a non-speculative request for the next translation and begun to handle this request using a separate translation slot. Therefore, TCU prefetch results in a performance advantage only if the number of cycles between each sequential translation request from the TBU is greater than the number of cycles that is taken for the TCU to handle the original translation request and to start the subsequent prefetch.

Even if TCU prefetch is enabled, a prefetch does not occur if one of the following caused the original request:

- A Speculative translation request, that is, **DTI_TBU_TRANS_REQ.PERM[1:0] = 2'b11**, because a TBU receives a StashOnceShared, StashOnceUnique, or StashTranslation transaction
- A translation request for an atomic transaction that provides a data response, that is, **DTI_TBU_TRANS_REQ.PERM[1:0] = 2'b10**, because a TBU receives an AtomicLoad, AtomicSwap, or AtomicCompare transaction

If the original translation request returns one the following, prefetch also does not occur:

- Fault response
- Global bypass response
- Stream bypass response



Prefetches can only occur with non-ATS translation requests because ATS itself is already a prefetch mechanism.

3.3.10 Error responses

AMBA defines external AXI slave error, SLVERR, and external AXI decode error, DECERR, and TRANSFAULT. The MMU-700 error response behavior depends on the interface.

The TCU ACE-Lite interface treats SLVERR and DECERR identically, as an abort.

When terminating a transaction, the TBS interface generates an OKAY, SLVERR, or TRANSFAULT response depending on the reason for the termination.

If the TBU TBM interface receives a DECERR or SLVERR response to a downstream transaction, it propagates the same abort type to the TBS interface.

3.3.11 Conversion between ACE-Lite and Armv8 attributes

The SMMUv3 architecture defines attributes in terms of the Arm®v8 architecture. See the [Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile](#). The MMU-700 components are therefore required to perform conversion between ACE-Lite and Arm®v8 attributes.

The TBU must convert:

- ACE-Lite attributes to Arm®v8 attributes when it receives transactions on the *Transaction Slave* (TBS) interface
- Arm®v8 attributes to ACE-Lite attributes when it outputs transactions on the *Transaction Master* (TBM) interface

The TCU must convert Arm®v8 attributes to ACE-Lite attributes when it outputs transactions on the QTW/DVM interface.

3.3.11.1 Slave interface memory type attribute handling

The **AxCACHE** and **AxDOMAIN** signals contain the memory attributes that apply to the TBS interface.

The following table shows the ACE-Lite to Armv8 attribute conversions that the TBU TBS interface performs.

Table 3-37: MMU-700 ACE-Lite to Armv8 memory attribute conversions

AxCACHE attribute	AxDOMAIN attribute	Armv8 memory attribute	Armv8 Shareability
Device Non-bufferable	System	Device-nGnRnE	Outer Shareable
Device Bufferable	System	Device-nGnRE	Outer Shareable
Normal Non-cacheable Bufferable	Any	Normal Inner Non-cacheable Outer Non-cacheable	Outer Shareable
Normal Non-cacheable Non-bufferable			
Write-Through No Allocate			
Write-Through Read-Allocate			
Write-Through Write-Allocate			
Write-Through Read and Write-Allocate			
Write-Back No Allocate	Non-shareable	Normal Inner Write-Back Outer Write-Back	Non-shareable
Write-Back Read-Allocate	Inner Shareable		Non-shareable
Write-Back Write-Allocate	Outer Shareable		Outer Shareable
Write-Back Read-Allocate Write-Allocate			



- Write-Back transactions are always treated as non-transient
- The Armv8-A Read-Allocate and Write-Allocate hints are the same as the hints that the **AxCACHE** Write-Back type provides
- The TBU TBS interface converts instruction writes into data writes, that is, it treats **awprot_s[2]** as 0

3.3.11.2 Master interface memory type attribute handling

The **AxCACHE** and **AxDOMAIN** signals contain the memory attributes that apply to the TBM and the QTW/DVM interfaces.

The TBU TBM interface can also use the **AxLOCK** signal to indicate an Exclusive access. The QTW/DVM interface does not use the **AxLOCK** signal.

On the TBU TBM interface, a bit on **AxUSER** indicates whether the memory type before the conversion is Outer Cacheable.

The following table shows the Armv8 to ACE-Lite attribute conversions that the master interfaces perform.

Table 3-38: MMU-700 Armv8 to ACE-Lite memory attribute conversions

Armv8 memory attribute	AxCACHE attribute	AxDOMAIN attribute	AxLOCK attribute	AxUSER Outer Cacheable
Device-nGnRnE	Device Non-bufferable	System	As Transaction Slave (TBS) AxLOCK value	0
Device-GRE	Device Bufferable	System	As TBS AxLOCK value	0
Device-nGRE				
Device-nGnRE				
Normal Inner Non-cacheable Outer Non-cacheable	Normal Non-cacheable Bufferable	System	As TBS AxLOCK value	0
Normal Inner Write-Through Outer Non-cacheable				
Normal Inner Write-Back Outer Non-cacheable				
Normal Inner Non-cacheable Outer Write-Through	Normal Non-cacheable Bufferable	System	As TBS AxLOCK value	1
Normal Inner Write-Through Outer Write-Through				
Normal Inner Write-Back Outer Write-Through				
Normal Inner Non-cacheable Outer Write-Back				
Normal Inner Write-Through Outer Write-Back				

Armv8 memory attribute	AxCACHE attribute	AxDOMAIN attribute	AxLOCK attribute	AxUSER Outer Cacheable
Normal Inner Write-Back Outer Write-Back	Write-Back No Allocate Write-Back Read-Allocate Write-Back Write-Allocate Write-Back Read and Write-Allocate	If AxBURST == FIXED, Non-shareable. If AxBURST != FIXED, the attribute reflects the Armv8 Shareability: <ul style="list-style-type: none"> Non-shareable Outer Shareable 	0	1

3.3.12 AXI USER bits that MMU-700 TBU TBM defines

The TBU TBM interface **AxUSER** signals, **aruser_m** and **awuser_m**, have 5 bits more than *TBUCFG_AxUSER_WIDTH* defines. These extra bits are output in higher-order bits of **aruser_m** and **awuser_m**.

The following table shows the MMU-700-defined **aruser_m** and **awuser_m** bits, where *w* represents the AXI USER bus width that *TBUCFG_AxUSER_WIDTH* defines.

Table 3-39: MMU-700-defined aruser_m and awuser_m bits

Bit position	Value
[<i>w</i> +4]	Outer Cacheable
[<i>w</i> +3: <i>w</i>]	The IMPLEMENTATION DEFINED page-based hardware attributes

3.3.12.1 Page Based Hardware Attribute (PBHA) in SMMUs

The Arm® architecture defines that 4 bits in both stage 1 and stage 2 leaf page table entry formats are reserved for software use. Arm®v8.5 and SMMUv3.2 define a mechanism where software can declare that it does not require them, on a per bit basis.

See the following:

- [Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile](#)
- [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#)

The *Page Based Hardware Attribute* (PBHA) mechanism effectively hands over control of those bits to **IMPLEMENTATION DEFINED** hardware purposes. These bits are called PBHA bits.

For more information about PBHA bits, see the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

3.4 Constraints and limitations of use

Certain usage constraints and limitations apply to the MMU-700.

Unless otherwise specified, an **IMPLEMENTATION DEFINED** field in a structure that the MMU-700:

- Generates is 0
- Reads is ignored

3.4.1 SMMUv3 implementation

This section describes SMMUv3 implementation in the CoreLink™ MMU-700 System Memory Management Unit.

Related information

[SMMU architectural registers](#) on page 106

3.4.1.1 ID register architectural options

This section describes ID register architectural options in the CoreLink™ MMU-700 System Memory Management Unit.

The following table shows the architectural options for MMU-700 from the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) that the SMMUv3 ID registers expose.

Table 3-40: SMMUv3 ID registers options

Register	Field	Value	Description
SMMU_IDR0	S2P	1	Stage 2 translation supported
	S1P	1	Stage 1 translation supported
	TTF	11	AArch64 and AArch32 translation supported
	COHACC	sup_cohacc	Coherent accesses supported, system configuration option
	BTM	sup_btm	Broadcast TLB maintenance, system configuration option
	HTTU[1:0]	{ sup_httu , 1'b0}	Access and dirty flag update supported
	DORMHINT	0	Dormant hint is not supported
	Hyp	1	EL2-E2H is supported
	ATS	1	ATS is supported
	NS1ATS	0	Stage 1-only ATS is supported
	ASID16	1	16-bit ASID is supported
	MSI	1	<i>Message Signaled Interrupts</i> (MSIs) are supported
	SEV	sup_sev	Send event is supported, system configuration option
	ATOS	0	ATOS is not supported
	PRI	1	PRI is supported
	VMW	1	VMID wildcard matching supported

Register	Field	Value	Description
	VMID16	1	16-bit VMIDs are supported
	CD2L	1	2-level context descriptor tables are supported
	VATOS	0	Virtual ATOS is not supported
	TTENDIAN	2'b00	Mixed-endian translation walks are supported
	STALL_MODEL	{1'b0, SMMU_S_CRO.NSSTALLD}	Stall and terminate models that are supported unless the Secure world disables Non-secure stalling
	TERM_MODEL	0	Terminating a transaction with RAZ/WI is supported
	ST_LEVEL	01	2-level stream table is supported
SMMU_IDR1	SIDSIZE	32	32-bit stream IDs are supported
	SSIDSIZE	20	20-bit substream IDs are supported
	PRIQS	5'b10011	2 ¹⁹ PRI queue entries are supported
	EVENTQS	5'b10011	2 ¹⁹ Event queue entries are supported
	CMDQS	5'b10011	2 ¹⁹ Command queue entries are supported
	ATTR_PERMS_OVR	1	Incoming permission attributes can be overridden
	ATTR_TYPES_OVR	1	Incoming memory attributes can be overridden
	REL	0	N/A, not fixed base addresses
	QUEUES_PRESET	0	Not fixed queue base addresses
	TABLES_PRESET	0	Not fixed table base addresses
SMMU_IDR2	BA_VATOS	0	N/A, no VATOS support
SMMU_IDR3	HAD	1	Hierarchical attribute disable supported
	PBHA	1	Page-based hardware attributes are supported
	XNX	1	ELO/EL1 stage 2 execute control is supported
	PPS	1	PASID, when present always used in PRI auto-generated response
	MPAM	1	MPAM is supported
	FWB	1	S2 control of memory type is supported
	STT	1	Small translation tables are supported
	RIL	1	Range-based invalidation and Level hint are supported
	BBML	2	Break before Make level 2 is supported
SMMU_IDR4	IMPDEF	0	No IMPLEMENTATION DEFINED features
SMMU_IDR5	OAS	sup_oas	Output address size, system configuration option
	GRAN4K	1	4K translation granule is supported
	GRAN16K	1	16K translation granule is supported
	GRAN64K	1	64K translation granule is supported
	VAX	01	Virtual addresses of 52 bits per CD.TTBx are supported
	STALL_MAX	TCUCFG_XLATE_SLOTS	Maximum number of outstanding stalled transactions
SMMU_IIDR	Implementor	0x43B	Arm® implementation
	Revision	MAX(p_level, ecorevnum)	Where p_level is 0 for p0
	Variant	1	Product variant, or major revision is r1
	ProductID	0x487	Arm® ID
SMMU_AIDR	ArchMinorRev	2	Architectural minor revision is SMMUv3.2
	ArchMajorRev	0	Architectural Major Revision is SMMUv3

Register	Field	Value	Description
SMMU_S_IDR0	MSI	1	Secure MSIs are supported
	STALL_MODEL	2'b00	Stall and terminate model is supported
SMMU_S_IDR1	S_SIDSIZE	24	24-bit Secure stream IDs are supported
	SEL2	1	Secure EL2 is supported
	SECURE_IMPL	1	Two Security states are implemented
SMMU_S_IDR3	SAMS	1	Secure ATS maintenance is not implemented
SMMU_S_IDR4	IMPDEF	0	No IMPLEMENTATION DEFINED features

3.4.1.2 Non-implemented commands and events

This section describes the non-implemented commands and events in the CoreLink™ MMU-700 System Memory Management Unit.

Event queue

MMU-700 does not generate the following events:

- F_UUT
- F_TLB_CONFLICT
- F_CFG_CONFLICT
- E_PAGE_REQUEST
- IMPDEF_EVENTn

Command queue

The following commands are accepted but silently ignored:

- CMD_PREFETCH_CONFIG
- CMD_PREFETCH_ADDR
- CMD_CFGI_VMS_PIDM

The CMD_ATC_INV command is supported for the Non-secure Command queue only. If the TCU encounters this command in the Secure Command queue, it results in a Secure Command queue error with reason code CERROR_ILL.

3.4.1.3 IMPLEMENTATION DEFINED fields

This section describes the **IMPLEMENTATION DEFINED** fields in the CoreLink™ MMU-700 System Memory Management Unit.

Unless otherwise specified, **IMPLEMENTATION DEFINED** fields in structures that MMU-700:

- Generates are 0
- Reads are ignored

3.4.1.4 Non-implemented registers

This section describes the non-implemented registers in the CoreLink™ MMU-700 System Memory Management Unit.

The following optional registers are not implemented and are RAZ/WI:

- SMMU_IDR4
- SMMU_STATUSR
- SMMU_GATOS_*
- SMMU_S_GATOS_*
- SMMU_VATOS_*

The following PMCG registers are not implemented and are RAZ/WI:

- SMMU_PMCG_IRQ_CFG0
- SMMU_PMCG_IRQ_CFG1
- SMMU_PMCG_IRQ_CFG2

3.4.2 AMBA implementation

This section describes AMBA implementation in the CoreLink™ MMU-700 System Memory Management Unit.

3.4.2.1 ACE-Lite feature support

The CoreLink™ MMU-700 System Memory Management Unit supports many ACE-Lite features.

The following table shows the ACE-Lite features that the CoreLink™ MMU-700 System Memory Management Unit supports. The table also shows the version of the [AMBA® AXI and ACE Protocol Specification](#), that is, E, F, G, or H, to which the particular feature was first added.

Table 3-41: ACE-Lite feature support

Specification issue	Interface properties	ACE-Lite TBU		TCU
		TBS	TBM	
E	DVM_v8	-	-	Y
	Ordered_Write_Observation	Y	Y	N
	WriteEvict_Transaction	N	N	N
F	Atomic_Transactions	Y	Y	Y
	Barrier_Transactions	N	N	N
	Cache_Stash_Transactions	Y	Y	N
	Check_Type (Odd_Parity_Byte_All)	N	N	N
	Coherency_Connection_Signals	-	-	Y

Specification issue	Interface properties	ACE-Lite TBU		TCU
		TBS	TBM	
	DeAllocation_Transactions	Y	Y	N
	DVM_v8.1	N	N	Y
	Loopback_Signals	Y	Y	N
	NSAccess_Identifiers	N	N	N
	Persist_CMO	Y	Y	N
	Poison	Y	Y	Y
	QoS_Accept	N	N	N
	Trace_Signals	N	N	N
	Untranslated_Transactions	Y	Y	N
	Wakeup_Signals	Y	Y	Y
G	CMO_On_Read	Y	Y	N
	CMO_On_Write	N	N	N
	MPAM_Support	Y	Y	Y
	Read_Data_Chunking	Y	Y	N
	Read_Interleaving_Disabled	N	N	Y
	Unique_ID_Support	Y	Y	Y
H	Consistent_DECERR	N	N	N
	DVM_Message_Support	Y	Y	Y
	DVM_v8.4	N	N	Y
	Exclusive_Accesses	Y	Y	N
	Max_Transaction_Bytes	4096	4096	64
	MTE_Support	N	N	N
	Prefetch_Transaction	N	N	N
	Regular_Transactions_Only	N	N	N
	Shareable_Transactions	Y	Y	Y
	Untranslated_Transactions v2	Y, v2 only	Y, v2 only ⁶	N
	Write_Plus_CMO	N	N	N
	WriteZero_Transaction	N	N	N

3.4.2.2 SLVERR and DECERR

This section describes SLVERR and DECERR in the CoreLink™ MMU-700 System Memory Management Unit.

The TCU QTW interface treats SLVERR and DECERR identically, as an abort.

The TBU TBS interface generates SLVERR when terminating a transaction that requires an abort response.

⁶ The TBM implements only certain signals of this property.

If the TBU TBM interface receives an SLVERR or DECERR response to a downstream transaction, the same abort type is propagated to the TBS interface.

3.4.2.3 Attribute handling

This section describes attribute handling in the CoreLink™ MMU-700 System Memory Management Unit.

When translation is enabled and a PCIe Root Complex issues transactions to a TBU, the following apply, depending on the type of transaction:

Untranslated (non-ATS) transaction

The SMMU applies attributes that a combination of the input attributes, STE overrides, and translation table descriptors determine.

Fully-translated (full ATS) transaction

The SMMU does not modify the attributes that are encoded in the fully translated transaction. The unmodified attributes are used as the output attributes.

Partially-translated (Split-stage ATS) transaction

The ATS translation response from the TCU to the PCIe Root Complex includes Stage 1 and Stage 2 attributes. The Stage-1-translated transaction to the TBU encodes these Stage 1 and Stage 2 attributes. The SMMU performs Stage 2 translation and combines the Stage 2 attributes a second time, but this does not affect the output attributes. The output attributes remain the same as the attributes that the TBU receives for the Stage-1-translated transaction.

For information about the preceding transactions and their attributes, see the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#).



TBUs that are connected to a PCIe Root Complex must have the **pcie_mode** input signal tied HIGH, as the table in [A.2.10 TBU tie-off signals](#) on page 218 describes.

3.4.2.3.1 Slave interface attribute handling

This section describes the slave interface attribute handling in the CoreLink™ MMU-700 System Memory Management Unit.

The TBU TBS interface converts the incoming ACE-Lite attributes into Armv8 attributes.

The following table shows the slave interface attribute handling.

Table 3-42: Slave interface attribute handling

ACE-Lite AxCACHE	ACE-Lite AxDOMAIN	Armv8 memory type	Armv8 shareability	Description
Device Non-bufferable	SY	Device-nGnRnE	OSH	-
Device Bufferable	SY	Device-nGnRE	OSH	-
Normal Non-cacheable Bufferable, Normal Non-cacheable Non-bufferable, Write-Through No-allocate, Write-Through Read-Allocate, Write-Through Write-Allocate, Write-Through Read and Write-Allocate	Any	Normal-iNC-oNC	OSH	Normal Non-cacheable Non-bufferable is a deprecated AxCACHE type and is converted to Normal Non-cacheable Bufferable. Write-Through types are converted to Non-cacheable on input to match the normalization step on output.
Write-back No-allocate, Write-back Read-Allocate, Write-back Write-Allocate, Write-back Read and Write-Allocate	NSH/ISH/ OSH	Normal-iWB-oWB	NSH/OSH	The Armv8 RA and WA hints depend on the Write-Back type. The transaction is always treated as non-transient. All ISH Shareability types are converted to OSH.

Slave interface AxPROT handling

Instruction writes are converted into data writes on the TBU TBS interface. In effect, **AWPROT[2]** is ignored and always treated as 0.

3.4.2.3.2 Master interface attribute handling

This section describes the master interface attribute handling in the CoreLink™ MMU-700 System Memory Management Unit.

Normalization

Both AMBA master interfaces, TBU TBM interface and TCU QTW interface, carry normalized attributes using the standard Cortex Armv8 scheme:

- Memory that is marked as Inner Write-Back Cacheable and Outer Write-Back Cacheable is output as Write-Back Cacheable
- Memory that is marked as Inner Non-cacheable or Write-Through Cacheable, or Outer Non-cacheable or Write-Through Cacheable, is output as Non-cacheable, Outer Shareable

On the TBU TBM interface, a bit on **AxUSER** indicates whether the output memory type before this conversion is outer cacheable.

The following table shows how the Armv8 transaction types are translated into AMBA ACE-Lite signals.

Table 3-43: Armv8 transaction types translated into AMBA ACE-Lite signals

Armv8 Memory Type	AxCACHE (TBM, QTW)	AxDOMAIN (TBM, QTW)	AxLOCK (TBM)	AxUSER outer cacheable bit (TBM)
Device-nGnRnE	Device No-bufferable	SY	TBS AxLOCK value	0
Device-GRE, Device-nGRE, Device-nGnRE	Device Bufferable	SY	TBS AxLOCK value	0
Normal-iNC-oNC, Normal-iWT-oNC, Normal-iWB-oNC	Normal Non-cacheable Bufferable	SY	TBS AxLOCK value	0
Normal-iNC-oWT, Normal-iWT-oWT, Normal-iWB-oWT, Normal-iNC-oWB, Normal-iWT-oWB	Normal Non-cacheable Bufferable	SY	TBS AxLOCK value	1
Normal-iWB-oWB	Write-back No-allocate / Write-back Read-Allocate / Write-back Write-Allocate / Write-back Read and Write-Allocate, depending on the Armv8 outer allocate hints.	AxBURST = FIXED: NSH AxBURST != FIXED: NSH or OSH, depending on the Armv8 Shareability	0	1

AxCACHE encodings

Where there are multiple legal values for **AxCACHE** as the [AMBA® AXI and ACE Protocol Specification](#) describes, the canonical, non-bracketed, one is used. Therefore, only the **AxCACHE** encodings that the following table shows are used.

Table 3-44: AxCACHE encodings

AMBA memory type	ARCACHE	AWCACHE
Device Non-bufferable	0000	0000
Device Bufferable	0001	0001
Normal Non-cacheable Bufferable	0011	0011
Write-back No-allocate	1011	0111
Write-back Read-Allocate	1111	0111
Write-back Write-Allocate	1011	1111
Write-back Read and Write-Allocate	1111	1111

3.4.2.4 AxREGION

This section describes the **AxREGION** signals in the CoreLink™ MMU-700 System Memory Management Unit.

The **AxREGION** signals are not required because:

- On QTW, they are driven as 0
- On TBM, they reflect the values of the corresponding TBS transaction

3.4.2.5 DVM interface

This section describes the DVM interface in the CoreLink™ MMU-700 System Memory Management Unit.

Supported DVM operations

In response to an **ACSNOOP** request, the CRRESP field is always driven as 0b00000.

All DVM operations are handled in a protocol-compliant manner, because the interconnect does not know that the TCU does not need DVM operations other than TLB invalidate. Any DVM operation with a DVM Message Type in **ACADDR[14:12]** other than TLB Invalidate or Synchronization is accepted and responded to on the CR channel but otherwise ignored.

DVM complete

DVM Complete messages are presented with:

- **ARPROT[2:0]** = 0b000, that is, Unprivileged Secure Data
- **ARDOMAIN[1:0]** = 0b10, that is, Outer Shareable

3.4.2.6 Internally terminated transactions

This section describes the internally terminated transactions in the CoreLink™ MMU-700 System Memory Management Unit.

Transactions that are terminated inside the TBU are returned with all **RUSER** and **BUSER** bits zero.

3.4.2.7 Transaction types

This section describes the transaction types in the CoreLink™ MMU-700 System Memory Management Unit.

MMU-700 supports several special transaction types, distinguished by a nonzero encoding of **AxSNOOP**. This section describes how each transaction type is handled.

Unless otherwise specified, transactions are propagated on TBM with the same transaction type that was presented on TBS.

An *ordinary read* or *ordinary write* is one with **AxSNOOP** = 0b0000, that is, depending on **AxDOMAIN**, and whether it is a read or a write, one of the following:

- ReadNoSnoop
- ReadOnce
- WriteNoSnoop
- WriteUnique

In the [AMBA® LTI Protocol Specification](#), see:

- *Section 5.2* for information about the mapping of LTI transactions to AXI types
- *Section 6* for information about handling LTI responses to convert them to AXI responses

3.4.2.8 Transactions that can result in a translation fault

In an MMU-700 system, some transactions can result in a translation fault, and certain behavior is associated with such transactions.

The MMU-700 treats the following transactions as ordinary reads when calculating translation faults:

- CleanShared
- CleanInvalid
- MakeInvalid
- CleanSharedPersist
- ReadOnceMakeInvalid
- ReadOnceCleanInvalid

Therefore, these transactions might require either read permission or execute permission at the appropriate privilege level.

The MMU-700 treats the following transactions as ordinary writes when calculating translation faults:

- WriteUniquePtlStash
- WriteUniqueFullStash

Therefore, these transactions require write permission at the appropriate privilege level.

CleanShared, CleanInvalid, MakeInvalid, and CleanSharedPersist transactions do not have a memory type. The input transaction and output transaction memory type and allocation hints are ignored and replaced by Normal, Inner Write-Back, Outer Write-Back, Read Allocate, Write Allocate. This behavior means that the **ARDOMAIN** output on the TBM interface is never System Shareable for these transactions, because they are never Non-cacheable or Device.

The MMU-700 treats transactions that pass the translation fault check as follows:

MakeInvalid transactions

The MMU-700 converts MakeInvalid transactions to CleanInvalid transactions, unless the translation also grants write permission and *Destructive Read Enable* (DRE) permission.

ReadOnceMakeInvalid and ReadOnceCleanInvalid transactions

The MMU-700 outputs ReadOnceMakeInvalid transactions as ReadOnceCleanInvalid transactions, unless the translation also granted write permission and DRE permission.

If the final transaction attributes on the TBU TBM interface are not Outer Shareable Write-Back, the MMU-700 converts ReadOnceMakeInvalid and ReadOnceCleanInvalid transactions into ordinary reads.

WriteUniquePtlStash and WriteUniqueFullStash transactions

If they pass the translation fault check, the MMU-700 converts WriteUniquePtlStash and WriteUniqueFullStash transactions to ordinary write transactions if either:

- The translation did not grant *Directed Cache Prefetch* (DCP) permission
- The final transaction attributes on the TBU TBM interface are not Outer Shareable Write-Back

If such a conversion occurs, **AWSTASH*** is driven as 0.

3.4.2.9 Transactions that cannot result in a translation fault

In an MMU-700 system, certain transactions cannot result in a translation fault, and certain behavior is associated with such transactions.

The following transactions never result in a translation fault:

- StashOnceShared
- StashOnceUnique
- StashTranslation

If any of these transactions require a translation request to the TCU, the MMU-700 issues a Speculative translation request on the DTI interconnect. StashOnceShared and StashOnceUnique transactions are terminated in the TBU, with a **BRESP** value of OKAY, when any of the following cases apply:

- The translation did not grant *Directed Cache Prefetch* (DCP) permission
- The final transaction attributes on the TBM interface are not Outer Shareable Write-Back
- The translation did not grant any of read, write, or execute permission at the appropriate privilege level



Only one of these permissions is required for the stash transaction to be permitted.



A **BRESP** value of OKAY indicates transaction success. The MMU-700 always generates this value when a StashOnceShared or a StashOnceUnique transaction is terminated in the TBU. This behavior applies even when a StreamDisable or GlobalDisable translation response causes the transaction to be terminated.

The MMU-700 never propagates StashTranslation transactions downstream, and uses StashTranslation only to prefetch Main TLB contents. MMU-700 always terminates StashTranslation transactions with a **BRESP** value of OKAY, even if no translation could be stored in the Main TLB.

The TBU ignores **AWPROT[0]** and **AWPROT[2]** for StashTranslation transactions, because they do not affect Speculative translation requests.



A StashTranslation transaction can be used to prefetch translations into the Main TLB of the MMU-700. However, for this prefetching to be useful, any subsequent transactions that intend to take advantage of the translations that have been prefetched into the Main TLB must use the same StreamID as the original prefetch. The StreamID identifies a translation context. Using a different StreamID for a subsequent transaction means that this subsequent transaction uses a different translation context to the translation that has been prefetched into the Main TLB and might lead to a TLB miss.

3.4.3 MPAM implementation

This section describes *Memory System Resource Partitioning and Monitoring* (MPAM) implementation in the CoreLink™ MMU-700 System Memory Management Unit.

This section describes the MPAM architectural options that MMU-700 uses from the [Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring \(MPAM\), for Armv8-A](#).

Certain MPAM registers are implemented.

Registers that are not described are not implemented.

MPAM capacity partitioning manages the following:

- TBU MTLB
- TCU configuration cache
- Walk caches

No other mechanism from the [Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring \(MPAM\), for Armv8-A](#) is implemented.

3.4.3.1 TCU MPAM

This section describes TCU *Memory System Resource Partitioning and Monitoring* (MPAM) implementation in the CoreLink™ MMU-700 System Memory Management Unit.

Internally, RIS is truncated to 1 bit, but externally it is the normal 4 bits.

RIS 0 = WCB

RIS 1 = CCB

Because RIS is internally truncated to 1 bit, there is no illegal RIS. It is therefore not necessary to report fewer controls in the ID registers for illegal RIS. It is also not necessary to RAZ/WI accesses to the control registers for illegal RIS.

The following table shows the TCU MPAM registers that are implemented.

Table 3-45: TCU MPAM registers implemented

Register	Field	Value used ⁷	Value not used ⁸	Description
MPAMF_IDR_LO (0x0000, Shared)	PARTID_MAX	1, 63, 511	1, 63, 511	In the TBU, set to: $(2^{TBUCFG_PARTID_WIDTH} - 1)$ In the TCU, set to $(2^{TCUCFG_PARTID_WIDTH} - 1)$
	PMG_MAX	1	1	Two Non-secure Performance Monitoring groups supported per PARTID
	HAS_CCAP_PART	1	0	Supports cache maximum capacity partitioning
	HAS_CPOR_PART	0	0	Cache portion partitioning not supported

⁷ Indicates the values when Resource is in use.

⁸ Indicates the values when Resource is not in use.

Register	Field	Value used ⁷	Value not used ⁸	Description
	HAS_MBW_PART	0	0	Memory Bandwidth partitioning not supported
	HAS_PRI_PART	0	0	Priority partitioning not supported
	EXT	1	1	EXTended MPAMF_IDR
	HAS_IMPL_IDR	0	0	Does not have IMPLEMENTATION-SPECIFIC partitioning features
	HAS_MSMON	1	0	Supports performance monitoring by matching a combination of PARTID and PMG
	HAS_PARTID_NRW	0	0	Does not have MPAMF_PARTID_NRW_IDR, MPAMCFG_INTPARTID or intPARTID mapping support
MPAMF_IDR_HI (0x0004, Shared)	HAS_RIS	1	0	Has Resource Instance Selector
	NO_IMPL_PART	1	0	There are no IMPLEMENTATION DEFINED resource controls that MPAMF_IMPL_IDR defines
	NO_IMPL_MSMON	1	0	There are no IMPLEMENTATION DEFINED resource monitors that MPAMF_IMPL_IDR defines
	HAS_EXTD_ESR	0	1	MPAMF_ESR is 64-bits. Not relevant because HAS_ESR is 0.
	HAS_ESR	0	1	MPAMF_ESR and MPAMF_ECR are not implemented
	RIS_MAX	0 or 1	0	Maximum RIS value used in the MSC: In the TBU, set to 0 In the TCU, set to 1
MPAMF_SIDR (0x0008, S-Only)	S_PARTID_MAX	1, 63, 511	1, 63, 511	In the TBU, set to $(2^{TBUCFG_PARTID_WIDTH} - 1)$ In the TCU, set to $(2^{TCUCFG_PARTID_WIDTH} - 1)$
	S_PMG_MAX	1	-	Two Secure Performance Monitoring groups supported per PARTID
MPAMF_MSMON_IDR (0x0080, Shared)	MSMON_CSU	1	-	Performance monitor supported for Cache Storage Usage by PARTID and PMG
	MSMON_MBWU	0	-	No performance monitor for Memory Bandwidth Usage by PARTID and PMG
	HAS_LOCAL_CAPT_EVNT	1	-	Has the local capture event generator and the MSMON_CAPT_EVNT register

⁷ Indicates the values when Resource is in use.

⁸ Indicates the values when Resource is not in use.

Register	Field	Value used ⁷	Value not used ⁸	Description
MPAMF_CCAP_IDR (0x0038, Shared)	CMAW_WD	8	-	256 fractions are supported
MPAMF_CSUMON_IDR (0x0088, Shared)	NUM_MON	4	-	Four monitoring counters are implemented
	HAS_CAPTURE	1	-	Has an MSMON_CSU_CAPTURE register for every MSMON_CSU and supports the capture event behavior
MPAMF_IIDR (0x0018, Shared)	All fields	All fields valid	-	Implementation ID Register
MPAMF_AIDR (0x0020, Shared)	ArchMajorRev	0x1	-	MPAM architecture v1.10
	ArchMinorRev	0x10	-	
MPAMCFG_PART_SEL (0x0100, Banked)	PARTID_SEL	Bits [TCUCFG_PARTID_WIDTH - 1:0] or [TBUCFG_PARTID_WIDTH - 1:0] valid	-	Can select up to 512 partitions to configure, based on <i>TBUCFG_PARTID_WIDTH</i> , <i>TCUCFG_PARTID_WIDTH</i> . When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	RIS	Bits [27:24] valid	-	Resource Instance Selector. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MPAMCFG_CMAX (0x0108, Banked)	CMAW	Bits [15:8] valid	-	Can choose up to 256 fractions. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MSMON_CFG_MON_SEL (0x0800, Banked)	MON_SEL	Bits [1:0] valid	-	Selects the monitor to configure. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	RIS	Bits [27:24] valid	-	Resource Instance Selector. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MSMON_CFG_CSU_FLT (0x0810, Banked)	PARTID	Bits [TCUCFG_PARTID_WIDTH - 1:0] or [TBUCFG_PARTID_WIDTH - 1:0] valid	-	Can select up to 512 partitions to configure, based on <i>TCUCFG_PARTID_WIDTH</i> , <i>TBUCFG_PARTID_WIDTH</i> . When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.

⁷ Indicates the values when Resource is in use.

⁸ Indicates the values when Resource is not in use.

Register	Field	Value used ⁷	Value not used ⁸	Description
	PMG	0	-	Can select up to PMG number 1. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (0x0818, Banked)	EN	Valid field.	-	The monitor instance is enabled or disabled to collect information. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	CAPT_EVNT	3'b111	-	Capture occurs when a MSMON_CAPT_EVNT register is written. All other values are not supported. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	CAPT_RESET	RES0.	-	There is no reason to ever reset a CSU monitor. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	OFLOW_STATUS	RES0	-	Overflow is not possible for a CSU monitor. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (0x0818, Banked)	OFLOW_INTR	RES0	-	This MPAM implementation does not support OFLOW_INTR. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	OFLOW_FRZ	RES0	-	Overflow is not possible for a CSU monitor. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	SUBTYPE	RES0	-	This field is reserved for future use. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.

⁷ Indicates the values when Resource is in use.

⁸ Indicates the values when Resource is not in use.

Register	Field	Value used ⁷	Value not used ⁸	Description
MSMON_CSU (0x0840, Banked)	All fields	All fields valid	-	Cache storage usage value. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MON_CSU_CAPTURE (0x0848, Banked)	All fields	All fields valid	-	Capture cache storage usage. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MSMON_CAPT_EVNT (0x0808, Banked)	All fields	All fields valid	-	Capture event. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.

3.4.3.2 TBU MPAM

This section describes *TBU Memory System Resource Partitioning and Monitoring* (MPAM) implementation in the CoreLink™ MMU-700 System Memory Management Unit.

When *TBUCFG_MTLB_DEPTH* == 0, the resource is not present, and when *TBUCFG_DIRECT_IDX* == 1, the resource is present but does not have *Memory System Resource Partitioning and Monitoring* (MPAM) controls. The associated ID Registers must report values of limited control under these circumstances. Therefore, many non-ID control registers are RAZ/WI in such circumstances.

For information about the TBU MPAM registers that are implemented, see the table in [3.4.3.1 TCU MPAM](#) on page 89.

3.4.4 Local Translation Interface implementation

This section describes *Local Translation Interface* (LTI) implementation in the CoreLink™ MMU-700 System Memory Management Unit.

The following table shows the values of the LTI properties.

Table 3-46: LTI properties

Name	Value	Description
<i>LTI_VC_COUNT</i>	2	Two LTI Virtual channels are chosen, one for read and one for write
<i>LTI_ID_WIDTH</i>	<i>TBUCFG_ID_WIDTH</i>	Equal to ID width of incoming transaction
<i>LTI_SID_WIDTH</i>	<i>TBUCFG_SID_WIDTH</i>	Equal to width of incoming SID
<i>LTI_OG_WIDTH</i>	<i>TBUCFG_LTI_OG_WIDTH</i>	Equal to width of incoming ordering groups

⁷ Indicates the values when Resource is in use.

⁸ Indicates the values when Resource is not in use.

Name	Value	Description
<i>LTI_TLBLOC_WIDTH</i>	⁹	Width of TLB location, in bits
<i>LTI_LOOP_WIDTH</i>	¹⁰	Width the LTI loopback signals, in bits
<i>LTI_LAUSER_WIDTH</i>	0	The LRUSER single bit indicates whether that transaction has been filtered for performance monitoring purposes
<i>LTI_LRUSER_WIDTH</i>	0	
<i>LTI_LCUSER_WIDTH</i>	0	

3.5 Configuration parameters and methodology

The TBU, TCU, and BAS components in the CoreLink™ MMU-700 System Memory Management Unit are delivered as SystemVerilog that you can parameterize. Use the *Generate* script to configure these components. There are several versions of the switch component, part of the BAS components that are delivered, to accommodate different numbers of slave interfaces.

For more information about the *Generate* script and detailed descriptions of parameters, see the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

3.5.1 Translation Control Unit I/O configuration parameters

You can configure the *Translation Control Unit* (TCU) I/O.



For more detailed descriptions of these configuration parameters, see the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

The following table shows the TCU I/O configuration parameter.

Table 3-47: TCU I/O configuration parameter

Interface and module name	Parameter name	Values	Description
QTW	<i>TCUCFG_QTW_DATA_WIDTH</i>	64, 128, 256, 512	ACE-Lite_DVM interface data width.

⁹ The value of the *LTI_TLBLOC_WIDTH* parameter, which is a local parameter, is as follows:

$$LTI_TLBLOC_WIDTH = \text{MAX}(1, ((TBUCFG_DIRECT_IDX == 1) ? ((TBUCFG_MTLB_DEPTH > 0) ? \log_2(TBUCFG_MTLB_DEPTH) : \log_2(4)) : \log_2(TBUCFG_MTLB_PARTS)))$$

¹⁰ For the LTI TBU, the value is equal to *TBUCFG_LTI_LOOP_WIDTH*.

For the ACE-Lite TBU, the value is calculated automatically.

3.5.2 Translation Control Unit buffer configuration parameters

You can configure the *Translation Control Unit* (TCU) buffer.



For more detailed descriptions of these configuration parameters, see the Arm® *CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

The following table shows the TCU buffer configuration parameters.

Table 3-48: TCU buffer configuration parameters

Parameter name	Values	Description
<i>TCUCFG_CC_DEPTH</i>	4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096	Configuration cache depth, in entries
<i>TCUCFG_WC_DEPTH</i>	8, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536	Walk cache depth, in entries Note: (<i>TCUCFG_WC_DEPTH</i> / <i>TCUCFG_WC_BANKS</i>)/ <i>TCUCFG_WC_WAYS</i>) must be > 1
<i>TCUCFG_WC_BANKS</i>	1, 2, 4	Number of banks in Walk Cache Note: (<i>TCUCFG_WC_DEPTH</i> / <i>TCUCFG_WC_BANKS</i>)/ <i>TCUCFG_WC_WAYS</i>) must be > 1
<i>TCUCFG_WC_WAYS</i>	4, 8, 16	Number of ways in walk cache Note: (<i>TCUCFG_WC_DEPTH</i> / <i>TCUCFG_WC_BANKS</i>)/ <i>TCUCFG_WC_WAYS</i>) must be > 1
<i>TCUCFG_NUM_TBU</i>	14, 62	Maximum number of DTI masters, that is, DTI-TBU and DTI-ATS masters, that the TCU supports. The value is two less than 16/64 to better fit into system memory maps. Note: The ACE-Lite TBU and LTI TBU are both examples of DTI-TBU masters. Integration TBU components count as two DTI masters because they contain two ACE-Lite TBUs.
<i>TCUCFG_XLATE_SLOTS</i>	4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096	Total permitted translation requests from all DTI masters Note: This value must be greater than or equal to <i>TCUCFG_PTW_SLOTS</i> .
<i>TCUCFG_PTW_SLOTS</i>	2, 4, 8, 16, 32, 64, 128, 256, 512	Number of parallel translation table walks
<i>TCUCFG_CTW_SLOTS</i>	1, 2, 4. Note: This value must not be greater than <i>TCUCFG_PTW_SLOTS</i> .	Number of parallel configuration table walks

Parameter name	Values	Description
<i>TCUCFG_WC_LKP_SLOTS</i>	2-28	<p>Walk Cache Lookup slots.</p> <p>The number of lookup slots that the walk cache uses.</p> <p>If you do not specify a value, the default value is used to provide the best performance when one page size is active in the walk cache.</p> <p>Reduce the value if TCU performance is not critical and increase the value if more than one page size is active in the walk cache.</p> <p>Make sure that the value is not greater than <i>TCUCFG_PTW_SLOTS</i>.</p> <p>You can:</p> <ul style="list-style-type: none"> • Increase the value of this parameter to improve performance, but with greater area • Decrease the value of this parameter to save area but with reduced performance
<i>TCUCFG_CC_IDXGEN_MODE</i>	0, 1	<p>Index generation mode for the configuration cache:</p> <p>0 Polynomial. Polynomial is the recommended setting for most systems</p> <p>1 Simple</p>
<i>TCUCFG_DTI_ATS</i>	0, 1, 2, 3, 4, 5, 6, 7, 8	<p>Number of DTI-ATS masters.</p> <p>Note: <i>TCUCFG_NUM_TBU</i> is the total number of DTI-TBU and DTI-ATS masters. <i>TCUCFG_DTI_ATS</i> is the total number of DTI-ATS masters.</p>
<i>TCUCFG_PMU_COUNTERS</i>	4, 16, 32	Number of PMU counters
<i>TCUCFG_PARTID_WIDTH</i>	1, 6, 9	<p>Width of PARTID that is supported:</p> <p>1 When set to 1, PARTID[8:1] sent on the DTI interface is set to 0</p> <p>6 When set to 6, PARTID[8:6] sent on the DTI interface is set to 0</p>
<i>TCUCFG_HZU_DEPTH</i>	2, 4, 8, 16, 32, 64.	Number of hazard cache entries. The number of hazard cache entries determines how many hazard lists the hazard cache can actively maintain in parallel. Choose the number of entries by considering the number of independent addresses that the TCU might access in parallel.
<i>TCUCFG_PREFETCH_SUPPORTED</i>	0, 1	Specifies whether prefetch is supported
<i>TCUCFG_DATARAM_TYPE</i>	0, 1, 2	<p>RAM type for data group of RAMs:</p> <p>0 Two ports, that is, one port is for reads and one port is for writes</p> <p>1 One port, that is, one port for both reads and writes</p> <p>2 2 × one port, that is, banked configuration</p> <p>See the <i>Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual</i>.</p> <p>Note: If this parameter is set to 2 and the depth of any particular RAM is 1 or 2, then the type is automatically set to 0. We recommend that you implement the RAM as registers in these cases.</p>

Parameter name	Values	Description
<i>TCUCFG_SLOTRAM_TYPE</i>	0, 1	RAM type for slot group of RAMs: 0 Two ports. One port is for reads and one port is for writes 1 One port, that is, one port for both reads and writes See the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.
<i>TCUCFG_CACHERAM_TYPE</i>	0, 1	RAM type for cache group of RAMs: 0 Two ports. One port is for reads and one port is for writes 1 One port, that is, one port for both reads and writes

3.5.3 Translation Control Unit debug configuration parameters

You can configure the *Translation Control Unit* (TCU) debug parameters.



For more detailed descriptions of these configuration parameters, see the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.

The following table shows the TCU debug configuration parameters.

Table 3-49: TCU debug configuration parameters

Parameter name	Values	Description
<i>TCUCFG_USE_ELA_DEBUG</i>	0, 1	Set the <i>TCUCFG_USE_ELA_DEBUG</i> parameter as follows: 0 The SIGCLKEN<n> , SIGNALGRP<n> , and SIGQUAL<n> signals are driven to 0. 1 The SIGCLKEN<n> , SIGNALGRP<n> , and SIGQUAL<n> signals are driven to according to B.1 TCU observation interfaces on page 228.

3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters

You can configure the common *Local Translation Interface* (LTI) *Translation Buffer Unit* (TBU) and ACE-Lite TBU parameters.



For more detailed descriptions of these configuration parameters, see the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.

The following table shows the common LTI TBU and ACE-Lite TBU configuration parameters.

Table 3-50: Common LTI TBU and ACE-Lite TBU configuration parameters

Parameter name	Values	Description
<i>TBUCFG_SID_WIDTH</i>	8, 16, 20, 24	Stream ID width
<i>TBUCFG_SSID_WIDTH</i>	1, 8, 20	SubstreamID width
<i>TBUCFG_DIRECT_IDX</i>	0, 1	Direct indexing. Note: Must be 0 if <i>TBUCFG_MTLB_DEPTH</i> = 0.
<i>TBUCFG_MTLB_PARTS</i>	1, 2, 4, 8, 16	Number of main TLB partitions. Note: Must be 1 if <i>TBUCFG_MTLB_DEPTH</i> = 0. Must be 1 if <i>TBUCFG_DIRECT_IDX</i> = 1. <i>TBUCFG_MTLB_PARTS</i> × <i>TBUCFG_MTLB_DEPTH</i> must not exceed 65536.
<i>TBUCFG_LTI_OG_WIDTH</i>	1-5	LTI ordering groups width. Number of ordering groups = $2^{\textit{Tbucfg_Lti_og_width}}$. Note: The top-level block being used implicitly sets the number of LTI ports. If the number of LTI ports is greater than 1, then <i>TBUCFG_LTI_OG_WIDTH</i> cannot have a value that is greater than 3.
<i>TBUCFG_LA_HNDSHK_MODE</i>	0, 1, 2, 3	Handshake mode on address channel before TLB lookup. Supported values are as follows: 0 FWD. Registered on the forward path only, that is, the direction that LAVALID on the corresponding interface indicates. 1 FWD. Registered on the forward path only, that is, the direction that LAVALID on the corresponding interface indicates. 2 BP. Bypass register slice, not registered. 3 BP. Bypass register slice, not registered.
<i>TBUCFG_LR_HNDSHK_MODE</i>	0, 1, 2, 3	Handshake mode on translation response path. Supported values are as follows: 0 or 1 FWD: Registered on the forward path only, that is, the direction that LRVALID on the corresponding interface indicates. 2 or 3 BP: Bypass register slice. Note: If the <i>TBUCFG_LR_HNDSHK_MODE</i> parameter is set to: 0 or 1 The LTI master must provide at least three LR credits to achieve full utilization of the LTI interface 2 or 3 The LTI master must provide at least two LR credits to achieve full utilization of the LTI interface

3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters

You can configure the *Translation Buffer Unit* (TBU) buffer.



For more detailed descriptions of these configuration parameters, see the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.

The following table shows the TBU buffer configuration parameters.

Table 3-51: TBU buffer configuration parameters

Parameter name	Values	Description
<i>TBUCFG_XLATE_SLOTS</i>	2, 4, 8, 16, 32, 64, 128, 256, 512, 1024	Number of translation slots, controlling the Hit-Under-Miss capability of the TBU
<i>TBUCFG_MTLB_LKP_SLOTS</i>	2-28	<p>Number of MTLB lookup slots.</p> <p>Use the default value to provide the best performance when one page size is active in the MTLB.</p> <p>You can:</p> <ul style="list-style-type: none"> • Increase the value of this parameter if more than one page size is active in the MTLB • Decrease the value of this parameter if the TBU performance is not critical <p>Ensure that the value of <i>TBUCFG_MTLB_LKP_SLOTS</i> is not greater than <i>TBUCFG_XLATE_SLOTS</i></p>
<i>TBUCFG_UTLB_DEPTH</i>	4, 8, 12, 16, 32, 64	MicroTLB depth, in entries
<i>TBUCFG_MTLB_DEPTH</i>	0, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536	Main TLB depth, in entries
<i>TBUCFG_MTLB_WAYS</i>	4, 8, 16	Number of ways in the MTLB
<i>TBUCFG_MTLB_BANKS</i>	1, 2, 4	Number of banks in the MTLB
<i>TBUCFG_PMU_COUNTERS</i>	4, 16, 32	Number of PMU counters
<i>TBUCFG_PARTID_WIDTH</i>	1, 6, 9	<p>Width of PARTID supported.</p> <p>When set to 1, PARTID[8:1] that the DTI interface receives is ignored.</p> <p>When set to 6, PARTID[8:6] received on the DTI interface is ignored.</p>
<i>TBUCFG_HZRD_ENTRIES</i>	0, 4, 8, 16, 32, 64	Number of hazard entries

Parameter name	Values	Description
<i>TBUCFG_SLOTRAM_TYPE</i>	0, 1	<p>RAM type for slot group of RAMs:</p> <p>0 Two ports, that is, one port for reads and one port for writes</p> <p>1 One port, that is, one port for both reads and writes</p> <p>See the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.</p>
<i>TBUCFG_CACHERAM_TYPE</i>	0, 1	<p>RAM type for cache group of RAMs:</p> <p>0 Two ports, that is, one port for reads and one port for writes</p> <p>1 One port, that is, one port for both reads and writes</p>
<i>BUCFG_DATARAM_TYPE</i>	0, 1, 2	<p>RAM type for data group of RAMs.</p> <p>0 Two ports, that is, one port for reads and one port for writes</p> <p>1 One port, that is, one port for both reads and writes</p> <p>2 2 × one port, that is, banked configuration</p> <p>See the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.</p> <p>Note: If <i>BUCFG_DATARAM_TYPE</i> is set to 2 and the depth of any particular RAM is 1 or 2, then the type is automatically set to 0. We recommend that you implement the RAM as registers in these cases.</p>

3.5.6 ACE-Lite Translation Buffer Unit register slice configuration parameters

You can configure the *Translation Buffer Unit* (TBU) register slice.



For more detailed descriptions of these configuration parameters, see the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.

The following table shows the TBU register slice configuration parameters.

Table 3-52: TBU register slice configuration parameters

Parameter name	Values
<i>TBUCFG_SI_AR_HNDSHK_MODE</i>	<p>Supported values are as follows:</p> <p>0 FULL: Fully registered, double-buffered register slice.</p> <p>1 FWD: Registered on the forward path only, that is, the direction that xVALID on the corresponding interface indicates.</p> <p>2 REV: Registered on the reverse path only, that is, the direction that xREADY on the corresponding interface indicates.</p> <p>3 BP: Bypass register slice.</p>
<i>TBUCFG_SI_R_HNDSHK_MODE</i>	
<i>TBUCFG_SI_AW_HNDSHK_MODE</i>	
<i>TBUCFG_SI_W_HNDSHK_MODE</i>	
<i>TBUCFG_SI_B_HNDSHK_MODE</i>	
<i>TBUCFG_MI_AR_HNDSHK_MODE</i>	
<i>TBUCFG_MI_R_HNDSHK_MODE</i>	
<i>TBUCFG_MI_AW_HNDSHK_MODE</i>	
<i>TBUCFG_MI_W_HNDSHK_MODE</i>	
<i>TBUCFG_MI_B_HNDSHK_MODE</i>	

3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters

You can configure the ACE-Lite *Translation Buffer Unit* (TBU) I/O.



For more detailed descriptions of these configuration parameters, see the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.

The following table shows the ACE-Lite TBU I/O configuration parameters.

Table 3-53: ACE-Lite TBU I/O configuration parameters

Interface and module name	Parameter name	Values	Description
TBS, TBM	<i>TBUCFG_ID_WIDTH</i>	1-32	AXI ID width
TBS, TBM	<i>TBUCFG_DATA_WIDTH</i>	64, 128, 256, 512	AXI data width
TBS, TBM	<i>TBUCFG_ARUSER_WIDTH</i> <i>TBUCFG_AWUSER_WIDTH</i> <i>TBUCFG_RUSER_WIDTH</i> <i>TBUCFG_WUSER_WIDTH</i> <i>TBUCFG_BUSER_WIDTH</i>	1-128	AXI USER bus widths
TBS, TBM	<i>TBUCFG_STASH_SUPPORT</i>	0, 1	Include stash ID signals
TBS, TBM	<i>TBUCFG_LOOP_WIDTH</i>	1-8	AXI loopback signal width

Interface and module name	Parameter name	Values	Description
TBS, TBM	<i>TBUCFG_WBUF_DEPTH</i>	0, 8, 16, 32, 64, 128, 256, 512, 1024, 2048	Write buffer depth. This parameter selects the maximum number of beats that can be stored in the write buffer. A value of 0 causes the write buffer not to be implemented. For example, for masters where most transactions are reads.
TBS, TBM	<i>TBUCFG_LFIFO_DEPTH</i>	0, 4	Latency FIFO depth. Supported values are as follows: 0, 4.
TBS, TBM	<i>TBUCFG_OT_TRACKER_TYPE</i>	0, 1	Type of the outstanding transaction tracker used. 0 Table. 1 Loopback. Loopback signals to track outstanding transactions. This setting increases the loopback signal width by 2 on the TBM. When using this mode, 4095 outstanding transactions are supported.
TBS, TBM	<i>TBUCFG_ROT_DEPTH</i>	4, 8, 16, 32, 64, 128, 256, 512	Number of outstanding read transactions. This configuration is valid only when <i>TBUCFG_OT_TRACKER_TYPE</i> is 0.
TBS, TBM	<i>TBUCFG_WOT_DEPTH</i>	4, 8, 16, 32, 64, 128, 256, 512	Number of outstanding write transactions. This configuration is valid only when <i>TBUCFG_OT_TRACKER_TYPE</i> is 0.
TBS, TBM	<i>TBUCFG_DATARAM_TYPE</i>	0, 1, 2	RAM type for data group of RAMs. 0 Two ports, that is, one port for reads and one port for writes 1 One port, that is, one port for both reads and writes 2 2 × one port, that is, banked configuration. See the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual. Note: If this parameter is set to 2 and the depth of any particular RAM is 1 or 2, then the type is automatically set to 0. We recommend that you implement the RAM as registers in these cases.

3.5.8 Local Translation Interface Translation Buffer Unit configuration parameters

You can configure the *Local Translation Interface (LTI) Translation Buffer Unit (TBU)*.



For more detailed descriptions of these configuration parameters, see the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.

The following table shows the LTI TBU configuration parameters.

Table 3-54: LTI TBU configuration parameters

Parameter name	Values	Description
<i>TBUCFG_LTI_ID_WIDTH</i>	1-32	LTI ID width. The top-level block being used implicitly sets the number of LTI ports. Note: If the number of LTI ports is greater than 1, then <i>TBUCFG_LTI_ID_WIDTH</i> >= <i>TBUCFG_LTI_OG_WIDTH</i> . See 3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters on page 97.
<i>TBUCFG_LTI_LOOP_WIDTH</i>	1-256	LTI loop width

3.5.9 Common Translation Buffer Unit debug configuration parameters

You can configure the *Translation Buffer Unit* (TBU) debug parameters.



For more detailed descriptions of these configuration parameters, see the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.

The following table shows the TBU debug configuration parameters.

Table 3-55: TBU debug configuration parameters

Parameter name	Values	Description
<i>TBUCFG_USE_ELA_DEBUG</i>	0, 1	Set the <i>TBUCFG_USE_ELA_DEBUG</i> parameter as follows: 0 The SIGCLKEN<n> , SIGNALGRP<n> , and SIGQUAL<n> signals are driven to 0 1 The SIGCLKEN<n> , SIGNALGRP<n> , and SIGQUAL<n> signals are driven according to: <ul style="list-style-type: none"> B.2 ACE-Lite TBU observation interfaces on page 231 B.3 LTI TBU observation interfaces on page 234

3.6 Debug capability

The CoreLink™ MMU-700 System Memory Management Unit provides debug functionality using the CoreSight™ ELA-600 Embedded Logic Analyzer.

For more information about debug capability, see the Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual.



The CoreSight™ ELA-600 Embedded Logic Analyzer is a separate licensed product that is not included with the CoreLink™ MMU-700 System Memory Management Unit.

Configuration options

For TCU configuration options, see [3.5.3 Translation Control Unit debug configuration parameters](#) on page 97.

For TBU configuration options, see [3.5.9 Common Translation Buffer Unit debug configuration parameters](#) on page 103.

Signals

For TCU observation signals, see [B.1 TCU observation interfaces](#) on page 228.

For ACE-Lite TBU observation signals, see [B.2 ACE-Lite TBU observation interfaces](#) on page 231.

For LTI TBU observation signals, see [B.3 LTI TBU observation interfaces](#) on page 234.

4 Programmers model

This chapter describes the MMU-700 programmers model.

4.1 About the programmers model

This section provides general information about the MMU-700 register properties.

The following information applies to the MMU-700 registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Access type is described as follows:

RW	Read and write
RO	Read-only
WO	Write-only
RAZ	Read-As-Zero
WI	Writes ignored
- Do not attempt to access reserved or unused address locations. Reading these locations results in RAZ and writing to these locations results in WI.
- Unless otherwise stated in the accompanying text:
 - Do-Not-Modify **UNDEFINED** register bits
 - Ignore **UNDEFINED** register bits on reads
 - All register bits are reset to 0 by a system or Cold reset
- Bit positions that are described as reserved are:
 - In an RW register, RAZ/WI
 - In an RO register, RAZ
 - In a WO register, WI

The MMU-700 registers are accessed using the PROG APB4 slave interface on the TCU, and cannot be accessed directly through any other slave interfaces.

Some registers are 64 bits, but the PROG APB4 interface is 32 bits. Because software accesses 64-bit registers 32 bits at a time, such accesses are not guaranteed to be 64-bit atomic. This behavior does not cause problems for software, because the SMMUv3 architecture does not require 64-bit atomic access to any registers.

The programmer's model contains separate TBU and TCU regions for internal control, RAS, and identification registers. Writes to unmapped or reserved registers are ignored, and reads SBZ. Non-secure accesses to Secure registers are RAZ/WI. The MMU-700 implements the identification register scheme that the SMMUv3 architecture defines.

The MMU-700 implements all the *Performance Monitor Counter Group* (PMCG) registers that the SMMUv3 architecture defines, except for:

- SMMU_PMCG_IRQ_CFG0
- SMMU_PMCG_IRQ_CFG1
- SMMU_PMCG_IRQ_CFG2

The MMU-700 does not implement the following SMMUv3 architectural registers, and accesses to these locations are RAZ/WI:

- SMMU_IDR4
- SMMU_STATUSR
- SMMU_GATOS_*
- SMMU_S_GATOS_*
- SMMU_VATOS_*

For more information about the SMMU architectural registers, see the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#).

4.1.1 Clearing ERRSTATUS registers

Software can clear the TCU_ERRSTATUS and TBU_ERRSTATUS registers by writing ones to fields that are set.

For more information about these registers, see the following:

- [4.8.3 TCU_ERRSTATUS register](#) on page 135
- [4.15.3 TBU_ERRSTATUS register](#) on page 175

If both of the following are true, a write to the register is ignored:

- Any of the V, UE, OF, CE, DE, fields are nonzero before the write.
- The write does not clear the nonzero V, UE, OF, CE, DE fields to zero by writing ones to the applicable field or fields.



CE must be cleared by writing 2'b11 to the field.

If a valid clearing write reaches the ERRSTATUS register on the same cycle as a new error, the new record is applied as though no previous error existed:

ERRSTATUS.V = 0.

4.2 SMMU architectural registers

The MMU-700 implements many of the SMMU architectural registers, that the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) defines.

The following table shows the SMMUv3 architectural registers that the MMU-700 implements.



All writable register fields reset to 0 unless the SMMU architecture specifies otherwise.

Table 4-1: SMMUv3 architectural registers

Register	Name	Description
SMMU_S_IDR0 - SMMU_S_IDR3	SMMU Secure feature Identification Registers	Provides information about the Secure features that the SMMU implementation supports
SMMU_S_CR0	Secure global Control Register 0	Provides global configuration of the Secure SMMU
SMMU_S_CR0ACK	Secure global Control Register 0 update Acknowledge	Provides acknowledgment of completion of updates to SMMU_S_CR0
SMMU_S_CR1	Secure global Control Registers	Provides the controls for Secure table and queue access attributes
SMMU_S_CR2		
SMMU_S_INIT	Secure Initialization control register	Provides a control to invalidate all Secure SMMU caching on system initialization
SMMU_S_GBPA	Secure Global Bypass Attribute register	Controls the global bypass attributes that are used for transactions from Secure streams when the MMU is disabled
SMMU_S_IRQ_CTRL	Secure Interrupt Control register	Contains enables for SMMU interrupts
SMMU_S_IRQ_CTRLACK	Secure Interrupt Control register update Acknowledge	Provides acknowledgment of the completion of updates to SMMU_S_IRQ_CTRL
SMMU_S_GERROR	Secure Global Error status register	Provides information on Secure global programming interface errors
SMMU_S_GERRORN	Secure Global Error Acknowledgment register	Contains the acknowledgment fields for SMMU_S_GERROR errors
SMMU_S_GERROR_IRQ_CFG0 - SMMU_S_GERROR_IRQ_CFG2	Secure Global Error IRQ Configuration register	Contains the Secure MSI address configuration for the GERROR IRQ
SMMU_S_STRTAB_BASE	Secure Stream Table Base address register	Contains the base address and attributes for the Secure Stream table
SMMU_S_STRTAB_BASE_CFG	Secure Stream Table Base Configuration register	Contains configuration fields for the Secure Stream table
SMMU_S_CMDQ_BASE	Secure Command queue Base address register	Contains the base address and attributes for the Secure Command queue
SMMU_S_CMDQ_PROD	Secure Command queue Producer index register	Contains the Secure Command queue index for writes by the producer
SMMU_S_CMDQ_CONS	Secure Command queue Consumer index register	Contains the Secure Command queue index for reads by the consumer

Register	Name	Description
SMMU_S_EVENTQ_BASE	Secure Event queue Base address register	Contains the base address and attributes for the Secure Event queue
SMMU_S_EVENTQ_PROD	Secure Event queue Producer index register	Contains the Secure Event queue index for writes by the producer
SMMU_S_EVENTQ_CONS	Secure Event queue Consumer index register	Contains the Secure Event queue index for reads by the consumer
SMMU_S_EVENTQ_IRQ_CFG0 - SMMU_S_EVENTQ_IRQ_CFG2	Secure Event queue IRQ Configuration registers	Contains the MSI address configuration for the Secure Event queue IRQ
SMMU_IDR0 - SMMU_IDR3 SMMU_IDR5	SMMU feature Identification Registers	Provides information about the features that the SMMU implementation supports
SMMU_IIDR	Implementation Identification Register	Provides implementer, part, and revision information for the SMMU implementation
SMMU_AIDR	Architecture Identification Register	Identifies the SMMU architecture version to which the implementation conforms
SMMU_CR0	Non-secure global Control Register 0	Provides the controls for the global configuration of the Non-secure SMMU
SMMU_CR0ACK	Non-secure global Control Register 0 update Acknowledge register	Provides acknowledgment of completion of updates to SMMU_CR0
SMMU_CR1	Non-secure global Control Register 1	Provides the controls for Non-secure table and queue access attributes
SMMU_CR2	Non-secure global Control Register 2	Provides the controls for the configuration of the global Non-secure features
SMMU_GBPA	Non-secure Global Bypass Attribute register	Controls the global bypass attributes that are used for transactions from Non-secure streams when the MMU is disabled
SMMU_IRQ_CTRL	Non-secure Interrupt Control register	Provides IRQ enable flags for edge-triggered wired outputs, if implemented, and MSI writes, if implemented
SMMU_IRQ_CTRLACK	Non-secure Interrupt Control register update Acknowledge register	Provides acknowledgment of the completion of updates to SMMU_IRQ_CTRL
SMMU_GERROR	Non-secure Global Error status register	Provides information about Non-secure global programming interface errors
SMMU_GERRORN	Non-secure Global Error acknowledgment register	Contains the acknowledgment fields for SMMU_GERROR errors
SMMU_GERROR_IRQ_CFG0	Non-secure Global Error IRQ Configuration register 0	Contains the MSI address configuration for the GERROR IRQ
SMMU_GERROR_IRQ_CFG1	Non-secure Global Error IRQ Configuration register 1	Contains the MSI payload configuration for the GERROR IRQ
SMMU_GERROR_IRQ_CFG2	Non-secure Global Error IRQ Configuration register 2	Contains the MSI attribute configuration for the GERROR IRQ
SMMU_STRTAB_BASE	Non-secure Stream Table Base address register	Contains the base address and attributes for the Non-secure Stream table
SMMU_STRTAB_BASE_CFG	Non-secure Stream Table Configuration register	Contains configuration fields for the Non-secure Stream table
SMMU_CMDQ_BASE	Non-secure Command queue Base address register	Contains the base address and attributes for the Non-secure Command queue

Register	Name	Description
SMMU_CMDQ_PROD	Non-secure Command queue Producer index register	Contains the Non-secure Command queue index for writes by the producer
SMMU_CMDQ_CONS	Non-secure Command queue Consumer index register	Contains the Non-secure Command queue index for reads by the consumer
SMMU_EVENTQ_BASE	Non-secure Event queue Base address register	Contains the base address and attributes for the Non-secure Event queue
SMMU_EVENTQ_PROD	Non-secure Event queue Producer index register	Contains the Non-secure Event queue index for writes by the producer
SMMU_EVENTQ_CONS	Non-secure Event queue Consumer index register	Contains the Non-secure Event queue index for reads by the consumer
SMMU_EVENTQ_IRQ_CFG0	Non-secure Event queue IRQ Configuration register 0	Contains the MSI address configuration for the Event queue IRQ
SMMU_EVENTQ_IRQ_CFG1	Non-secure Event queue IRQ Configuration register 1	Contains the MSI payload configuration for the Event queue IRQ
SMMU_EVENTQ_IRQ_CFG2	Non-secure Event queue IRQ Configuration register 2	Contains the MSI attribute configuration for the Event queue IRQ
SMMU_PRIQ_BASE	Non-secure PRI queue Base address register	Contains the base address and attributes for the Non-secure PRI queue
SMMU_PRIQ_PROD	Non-secure PRI queue Producer index register	Contains the Non-secure PRI queue index for writes by the producer
SMMU_PRIQ_CONS	Non-secure PRI queue Consumer index register	Contains the Non-secure PRI queue index for reads by the consumer
SMMU_PRIQ_IRQ_CFG0	Non-secure PRI queue IRQ Configuration register 0	Contains the MSI address configuration for the PRI queue IRQ
SMMU_PRIQ_IRQ_CFG1	Non-secure PRI queue IRQ Configuration register 1	Contains the MSI payload configuration for the PRI queue IRQ
SMMU_PRIQ_IRQ_CFG2	Non-secure PRI queue IRQ Configuration register 2	Contains the MSI attribute configuration for the PRI queue IRQ

The MMU-700 implements an SMMUv3 *Performance Monitor Counter Group* (PMCG) in the TCU and in each TBU. The following table lists the registers that the MMU-700 implements in each PMCG.

Table 4-2: SMMUv3 PMCG registers

Register	Name	Description
SMMU_PMCG_EVCNTR0 - SMMU_PMCG_EVCNTR3	SMMU PMCG Event Counter registers	Contains the values of the event counters
SMMU_PMCG_EVTYPER0 - SMMU_PMCG_EVTYPER3	SMMU PMCG Event Type configuration registers	Configures the events that the corresponding counter counts
SMMU_PMCG_SVR0 - SMMU_PMCG_SVR3	SMMU PMCG Shadow Value Registers	Contains the shadow value of the corresponding event counter
SMMU_PMCG_SMRO	SMMU PMCG Stream Match filter Register	Configures the stream match filter for the corresponding event counter
SMMU_PMCG_CNTENSET0	SMMU PMCG Counter Enable Set register	Provides the set mechanism for the counter enables

Register	Name	Description
SMMU_PMC_G_CNTENCLR0	SMMU PMCG Counter Enable Clear register	Provides the clear mechanism for the counter enables
SMMU_PMC_G_INTENSETO	SMMU PMCG Interrupt contribution Enable Set register	Provides the set mechanism for the counter interrupt contribution enables
SMMU_PMC_G_INTENCLR0	SMMU PMCG Interrupt contribution Enable Clear register	Provides the clear mechanism for the counter interrupt enables
SMMU_PMC_G_OVSCLR0	SMMU PMCG Overflow Status Clear register	Provides the clear mechanism for the overflow status bits and provides read access to the overflow status bit values
SMMU_PMC_G_OVSSETO	SMMU PMCG Overflow Status Set register	Provides the set mechanism for the overflow status bits and provides read access to the overflow status bit values
SMMU_PMC_G_CAPR	SMMU PMCG Counter shadow value Capture Register	Controls the counter shadow value capture mechanism
SMMU_PMC_G_SCR	SMMU PMCG Secure Control Register	Secure Control Register
SMMU_PMC_G_CFGR	SMMU PMCG Configuration information Register	Provides information about the PMCG implementation
SMMU_PMC_G_CR	SMMU PMCG Control Register	Contains the Performance Monitor control flags
SMMU_PMC_G_CEID0 - SMMU_PMC_G_CEID1	SMMU PMCG Common Event ID registers	Contains the lower and upper 64 bits of the Common Event identification bitmap
SMMU_PMC_G_IRQ_CTRL	SMMU PMCG IRQ enable register	Contains the Performance Monitors IRQ enable
SMMU_PMC_G_IRQ_CTRLACK	SMMU PMCG IRQ enable Acknowledge register	Provides acknowledgment of the completion of updates to SMMU_PMC_G_IRQ_CTRL
SMMU_PMC_G_AIDR	SMMU PMCG Architecture Identification Register	Provides the Performance Monitor Architecture Identification
SMMU_PMC_G_ID_REGS	ID registers	IMPLEMENTATION DEFINED
SMMU_PMC_G_PMAUTHSTATUS	PMU Authentication Status register	Performance Monitor authentication status
SMMU_PMC_G_PMDEVARCH	PMU Device Architecture register	Performance Monitor architecture identifier
SMMU_PMC_G_PMDEVTYPE	PMU Device Type register	Performance Monitor device type

Related information

[SMMUv3 implementation](#) on page 77

4.3 MMU-700 memory map

The MMU-700 memory map contains all registers.

4.3.1 Main MMU-700 memory map

The main MMU-700 memory map includes the TCU and all TBUs, and the maximum number of implemented TBUs.

The following table shows the full memory map.

Table 4-3: Main MMU-700 memory map

Address range	Description
0x000000 - 0x03FFFC	TCU registers
0x040000 - 0x05FFFC	TBU0 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.
0x060000 - 0x07FFFC	TBU1 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.
0x080000 - 0x09FFFC	TBU2 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.
...	...
0x7C0000 - 0x7DFFFC	TBU60 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.
0x7E0000 - 0x7FFFC	TBU61 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.



This document describes all TBU and TCU register addresses relative to the base address for that component.



Where a multi-port LTI TBU is implemented, it occupies the same address map space as if it were a single-port LTI TBU. The addressing is per TBU, not per initiating interface.

4.3.2 TCU memory map

The TCU memory map contains various categories of registers.

The TCU **IMPLEMENTATION DEFINED** registers include the following:

- [4.7 TCU microarchitectural registers](#) on page 123 for controlling microarchitectural features
- [4.9 TCU system discovery registers](#) on page 139
- [4.8 TCU RAS registers](#) on page 133

- [4.6 TCU PMU registers](#) on page 119

The following registers are also included:

- [4.10 TCU PIU integration registers](#) on page 157.
- Walk cache stage and level *Memory System Resource Partitioning and Monitoring* (MPAM) maximum capacity registers.
- MPAM memory-mapped registers.

The following table shows the MMU-700 TCU memory map.

Table 4-4: MMU-700 TCU memory map

Address range	Description
0x00000-0x0FFFC	TCU registers, page 0, including: <ul style="list-style-type: none"> • SMMUv3 registers, page 0 • TCU <i>Performance Monitor Counter Group</i> (PMCG) registers, page 0, starting at offset 0x02000 • TCU microarchitectural registers • TCU system discovery registers • TCU MPAM registers
0x10000-0x1FFFC	TCU registers, page 1. This address range contains the SMMUv3 registers, page 1.
0x20000-0x2FFFC	TCU registers, page 2. This address range contains the TCU PMCG registers, page 1, starting at offset 0x22000.
0x30000-0x3FFFC	Reserved.

The following table shows how the TCU **IMPLEMENTATION DEFINED** PMCG, and MPAM registers are allocated to regions of the TCU address space. Other regions are reserved.

Table 4-5: TCU PMCG, RAS, and MPAM register allocation to regions of TCU address space

Address range	Description
0x00FD0-0x00FFC	SMMU ID registers
0x02000-0x02FFC	Performance Monitor, page 0
0x03000-0x03FFC	MPAM Non-secure registers
0x08E00-0x08E7C	Microarchitectural features and integration registers
0x08E34-0x08E78	System discovery registers
0x08E80-0x08EFC	<i>Reliability, Availability, and Serviceability</i> (RAS) registers
0x09000-0x097FC	TCU node microarchitecture registers
0x09800-0x0981C	Walk cache stage and level MPAM maximum capacity registers
0x0B000-0x0BFFC	MPAM Secure registers
0x22000-0x22FFC	Performance Monitor, page 1

4.3.3 TBU memory map

The TBU memory map contains various categories of registers.

The TBU registers contain the following:

- **IMPLEMENTATION DEFINED** [4.14 TBU microarchitectural registers](#) on page 166 for controlling microarchitectural features
- [4.16 TBU system discovery registers](#) on page 178
- [4.15 TBU RAS registers](#) on page 172
- Direct access to cache state
- [4.13 TBU PMU registers](#) on page 163
- Performance Monitor counter registers, on a separate 64KB page to enable it to be paged for direct access from a Guest OS

The following table shows the TBU memory map.

Table 4-6: TBU memory map

Address range	Description
0x08E00-0x08E7C	Microarchitectural registers System discovery registers Integration registers
0x08E80-0x08EFC	RAS
0x00FD0-0x00FFC	ID registers
0x02000-0x02FFC	Performance Monitor page 0
0x12000-0x12FFC	Performance Monitor page 1
0x03000-0x03FFC	TBU MPAM Non-secure registers
0x0B000-0x0BFFC	TBU MPAM Secure registers



Any regions that the table does not show are reserved.

4.4 MMU-700 registers summary

The register summary describes the MMU-700 registers and some key characteristics.

4.4.1 TCU identification registers summary

The MMU-700 contains TCU identification registers.

The following table shows the TCU identification registers in offset order from the base memory address.

Table 4-7: TCU identification registers summary

Name	Offset	Type	Description
SMMU_CIDR3	0x00FFC	RO	4.5 TCU component and peripheral ID registers on page 119
SMMU_CIDR2	0x00FF8	RO	
SMMU_CIDR1	0x00FF4	RO	
SMMU_CIDR0	0x00FF0	RO	
SMMU_PIDR3	0x00FEC	RO	
SMMU_PIDR2	0x00FE8	RO	
SMMU_PIDR1	0x00FE4	RO	
SMMU_PIDR0	0x00FE0	RO	
SMMU_PIDR7	0x00FDC	RO	
SMMU_PIDR6	0x00FD8	RO	
SMMU_PIDR5	0x00FD4	RO	
SMMU_PIDR4	0x00FD0	RO	

4.4.2 TCU and TBU PMU identification registers summary

The TCU and the TBU use the same PMU identification registers.

The following table shows the TCU and TBU PMU identification registers in offset order from the base memory address.

Table 4-8: TCU and TBU PMU identification registers summary

Name	Offset	Type	Description
SMMU_PMCG_PMAUTHSTATUS	0x00FB8	RO	4.6 TCU PMU registers on page 119 4.13 TBU PMU registers on page 163
SMMU_PMCG_PIDR4	0x02FD0	RO	
SMMU_PMCG_PIDR5	0x02FD4	RO	
SMMU_PMCG_PIDR6	0x02FD8	RO	
SMMU_PMCG_PIDR7	0x02FDC	RO	
SMMU_PMCG_PIDR0	0x02FE0	RO	
SMMU_PMCG_PIDR1	0x02FE4	RO	
SMMU_PMCG_PIDR2	0x02FE8	RO	
SMMU_PMCG_PIDR3	0x02FEC	RO	
SMMU_PMCG_CIDR0	0x02FF0	RO	
SMMU_PMCG_CIDR1	0x02FF4	RO	

Name	Offset	Type	Description
SMMU_PMCGR_CIDR2	0x02FF8	RO	
SMMU_PMCGR_CIDR3	0x02FFC	RO	

4.4.3 TCU Reliability, Availability, and Service registers summary

The MMU-700 contains TCU *Reliability, Availability, and Service* (RAS) registers.

The following table shows the TCU RAS registers in offset order from the base memory address.

Table 4-9: TCU RAS registers summary

Name	Offset	Type	Width	Description
TCU_ERRFR	0x08E80	RO, Secure	64-bit	4.8.1 TCU_ERRFR register on page 133
TCU_ERRCTLR	0x08E88	RW, Secure	64-bit	4.8.2 TCU_ERRCTLR register on page 135
TCU_ERRSTATUS	0x08E90	RW, Secure	64-bit	4.8.3 TCU_ERRSTATUS register on page 135
TCU_ERRGEN	0x08EC0	RW, Secure	64-bit	4.8.4 TCU_ERRGEN register on page 138

4.4.4 TCU microarchitectural registers summary

The MMU-700 contains TCU microarchitectural registers.

The following table shows the TCU microarchitectural registers in offset order from the base memory address.

Table 4-10: TCU microarchitectural registers summary

Name	Offset	Type	Width	Description
TCU_CTRL	0x08E00	RW	32-bit	4.7.1 TCU_CTRL register on page 123
TCU_QOS	0x08E04	RW	32-bit	4.7.2 TCU_QOS register on page 125
TCU_CFG	0x08E08	RO	32-bit	4.7.3 TCU_CFG register on page 126
TCU_STATUS	0x08E10	RO	32-bit	4.7.4 TCU_STATUS register on page 127
TCU_SCR	0x08E18	RW, Secure	32-bit	4.7.7 TCU_SCR register on page 131
TCU_NODE_CTRL _n	0x09000-0x093FC	RW	32-bit	4.7.5 TCU_NODE_CTRL_n register on page 128
TCU_NODE_STATUS _n	0x09400-0x097FC	RO	32-bit	4.7.6 TCU_NODE_STATUS_n register on page 130
TCU_WC_SxLy_CMAX	0x09800-0x0981C	RW	32-bit	4.7.8 TCU_WC_SxLy_CMAX registers on page 132

4.4.5 TCU system discovery registers summary

The MMU-700 contains TCU system discovery registers.

The following table shows the TCU system discovery registers in offset order from the base memory address.

Table 4-11: TCU system discovery registers summary

Name	Offset	Type	Width	Description
TCU_SYSDISC0	0x08E34	RO	32-bit	4.9.1 TCU_SYSDISC0 system discovery register on page 139
TCU_SYSDISC1	0x08E38	RO	32-bit	4.9.2 TCU_SYSDISC1 system discovery register on page 140
TCU_SYSDISC2	0x08E3C	RO	32-bit	4.9.3 TCU_SYSDISC2 system discovery register on page 141
TCU_SYSDISC3	0x08E40	RO	32-bit	4.9.4 TCU_SYSDISC3 system discovery register on page 142
TCU_SYSDISC4	0x08E44	RO	32-bit	4.9.5 TCU_SYSDISC4 system discovery register on page 143
TCU_SYSDISC5	0x08E48	RO	32-bit	4.9.6 TCU_SYSDISC5 system discovery register on page 144
TCU_SYSDISC6	0x08E4C	RO	32-bit	4.9.7 TCU_SYSDISC6 system discovery register on page 145
TCU_SYSDISC7	0x08E50	RO	32-bit	4.9.8 TCU_SYSDISC7 system discovery register on page 146
TCU_SYSDISC8	0x08E54	RO	32-bit	4.9.9 TCU_SYSDISC8 system discovery register on page 147
TCU_SYSDISC9	0x08E58	RO	32-bit	4.9.10 TCU_SYSDISC9 system discovery register on page 148
TCU_SYSDISC10	0x08E5C	RO	32-bit	4.9.11 TCU_SYSDISC10 system discovery register on page 149
TCU_SYSDISC11	0x08E60	RO	32-bit	4.9.12 TCU_SYSDISC11 system discovery register on page 150
TCU_SYSDISC12	0x08E64	RO	32-bit	4.9.13 TCU_SYSDISC12 system discovery register on page 151
TCU_SYSDISC13	0x08E68	RO	32-bit	4.9.14 TCU_SYSDISC13 system discovery register on page 152
TCU_SYSDISC14	0x08E6C	RO	32-bit	4.9.15 TCU_SYSDISC14 system discovery register on page 153
TCU_SYSDISC15	0x08E70	RO	32-bit	4.9.16 TCU_SYSDISC15 system discovery register on page 154
TCU_SYSDISC16	0x08E74	RO	32-bit	4.9.17 TCU_SYSDISC16 system discovery register on page 155
TCU_SYSDISC17	0x08E78	RO	32-bit	4.9.18 TCU_SYSDISC17 system discovery register on page 156

4.4.6 TCU integration registers summary

The MMU-700 contains TCU integration registers.

The following table shows the TCU integration registers in offset order from the base memory address.

Table 4-12: TCU integration registers summary

Name	Offset	Type	Width	Description
ITEN	0x08E20	RW	32-bit	4.10.1 ITEN register for the TCU on page 158
ITOP_PIU	0x08E24	RW	32-bit	4.10.2 ITOP register for the TCU Programmer Interface Unit on page 159
ITOP_TMU	0x08E2C	RW	32-bit	4.11.1 ITOP register for the TCU Translation Management Unit on page 160
ITIN_TMU	0x08E30	RO	32-bit	4.11.2 ITIN register for the TCU Translation Management Unit on page 162

4.4.7 TBU identification registers summary

The MMU-700 contains TBU identification registers.

The following table shows the TBU identification registers in offset order from the base memory address.

Table 4-13: TBU identification registers summary

Name	Offset	Type	Description
SMMU_CIDR3	0x00FFC	RO	4.12 TBU component and peripheral ID registers on page 162
SMMU_CIDR2	0x00FF8	RO	
SMMU_CIDR1	0x00FF4	RO	
SMMU_CIDR0	0x00FF0	RO	
SMMU_PIDR3	0x00FEC	RO	
SMMU_PIDR2	0x00FE8	RO	
SMMU_PIDR1	0x00FE4	RO	
SMMU_PIDR0	0x00FE0	RO	
SMMU_PIDR7	0x00FDC	RO	
SMMU_PIDR6	0x00FD8	RO	
SMMU_PIDR5	0x00FD4	RO	
SMMU_PIDR4	0x00FD0	RO	

4.4.8 TBU Reliability, Availability, and Serviceability registers summary

The MMU-700 contains TBU *Reliability, Availability, and Serviceability* (RAS) registers.

The following table shows the TBU RAS registers in offset order from the base memory address.

Table 4-14: TBU RAS registers summary

Name	Offset	Width	Type	Description
TBU_ERRFR	0x08E80	64-bit	RO, Secure	4.15.1 TBU_ERRFR register on page 173
TBU_ERRCTLR	0x08E88	64-bit	RW, Secure	4.15.2 TBU_ERRCTLR register on page 174
TBU_ERRSTATUS	0x08E90	64-bit	RW, Secure	4.15.3 TBU_ERRSTATUS register on page 175
TBU_ERRGEN	0x08EC0	64-bit	RW, Secure	4.15.4 TBU_ERRGEN register on page 177

RAS error reporting

When a *Correctable Error* (CE) occurs:

A CE is reported in 4.15.3 TBU_ERRSTATUS register on page 175.

If TBU_ERRCTLR.FI is set, an interrupt is raised on **ras_fhi**. See 3.2.2.7 TBU interrupt interfaces on page 43.

4.4.9 TBU microarchitectural registers summary

The MMU-700 contains TBU microarchitectural registers.

The following table shows the TBU microarchitectural registers in offset order from the base memory address.

Table 4-15: TBU microarchitectural registers summary

Name	Offset	Type	Width	Description
TBU_CTRL	0x08E00	RW	32-bit	4.14.1 TBU_CTRL register on page 166
TBU_LTI_PORT_RESOURCE_LIMIT	0x08E04	RW	32-bit	4.14.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 168
TBU_SCR	0x08E18	RW, Secure	32-bit	4.14.3 TBU_SCR register on page 171

4.4.10 TBU system discovery registers summary

The MMU-700 contains TBU system discovery registers.

The following table shows the TBU system discovery registers in offset order from the base memory address.

Table 4-16: TBU system discovery registers summary

Name	Offset	Type	Width	Description
TBU_SYSDISCO	0x08E30	RO	32-bit	4.16.1 TBU_SYSDISCO system discovery register on page 178
TBU_SYSDISC1	0x08E34	RO	32-bit	4.16.2 TBU_SYSDISC1 system discovery register on page 179
TBU_SYSDISC2	0x08E38	RO	32-bit	4.16.3 TBU_SYSDISC2 system discovery register on page 180
TBU_SYSDISC3	0x08E3C	RO	32-bit	4.16.4 TBU_SYSDISC3 system discovery register on page 181
TBU_SYSDISC4	0x08E40	RO	32-bit	4.16.5 TBU_SYSDISC4 system discovery register on page 182
TBU_SYSDISC5	0x08E44	RO	32-bit	4.16.6 TBU_SYSDISC5 system discovery register on page 183
TBU_SYSDISC6	0x08E48	RO	32-bit	4.16.7 TBU_SYSDISC6 system discovery register on page 184
TBU_SYSDISC7	0x08E4C	RO	32-bit	4.16.8 TBU_SYSDISC7 system discovery register on page 185
TBU_SYSDISC8	0x08E50	RO	32-bit	4.16.9 TBU_SYSDISC8 system discovery register on page 186
TBU_SYSDISC9	0x08E54	RO	32-bit	4.16.10 TBU_SYSDISC9 system discovery register on page 187
TBU_SYSDISC10	0x08E58	RO	32-bit	4.16.11 TBU_SYSDISC10 system discovery register on page 188
TBU_SYSDISC11	0x08E5C	RO	32-bit	4.16.12 TBU_SYSDISC11 system discovery register on page 189
TBU_SYSDISC12	0x08E60	RO	32-bit	4.16.13 TBU_SYSDISC12 system discovery register on page 190
TBU_SYSDISC13	0x08E64	RO	32-bit	4.16.14 TBU_SYSDISC13 system discovery register on page 191
TBU_SYSDISC14	0x08E68	RO	32-bit	4.16.15 TBU_SYSDISC14 system discovery register on page 192

4.4.11 TBU integration registers summary

The MMU-700 contains TBU integration registers.

The following table shows the TBU integration registers in offset order from the base memory address.

Table 4-17: TBU integration registers summary

Name	Offset	Type	Width	Description
ITEN	0x08E20	RW	32-bit	4.17.1 ITEN register for the TBU on page 193
ITOP_TBU	0x08E24	RW	32-bit	4.17.2 ITOP_TBU register on page 194

Name	Offset	Type	Width	Description
ITIN_TBU	0x08E28	RW	32-bit	4.17.3 ITIN_TBU register on page 195

4.5 TCU component and peripheral ID registers

This section describes the TCU component and peripheral ID registers.

The following table shows the TCU component and peripheral ID registers.

Table 4-18: TCU component and peripheral ID registers

Name	Offset	Field	Value	Description
SMMU_CIDR3, Component ID3	0x00FFC	[7:0]	0xB1	Preamble
SMMU_CIDR2, Component ID2	0x00FF8	[7:0]	0x05	Preamble
SMMU_CIDR1, Component ID1	0x00FF4	[7:0]	0xF0	Preamble
SMMU_CIDR0, Component ID0	0x00FF0	[7:0]	0x0D	Preamble
SMMU_PIDR3, Peripheral ID3	0x00FEC	[7:4]	MAX(<i>p_level</i> , ecorevnum)	REVAND, minor revision, where <i>p_level</i> is 0 for p0.
		[3:0]	0x00	CMOD
SMMU_PIDR2, Peripheral ID2	0x00FE8	[7:4]	0x01	REVISION, major revision
		[3]	1	JEDEC-assigned value for DES always used
		[2:0]	3	DES_1: bits [6:4] bits of the JEP106 Designer code
SMMU_PIDR1, Peripheral ID1	0x00FE4	[7:4]	0xB	DES_0: bits [3:0] of the JEP106 Designer code
		[3:0]	0x4	PART_1: bits [11:8] of the Part number
SMMU_PIDR0, Peripheral ID0	0x00FE0	[7:0]	0x87	PART_0: bits [7:0] of the Part number
SMMU_PIDR7, Peripheral ID7	0x00FDC	-	RES0	Reserved
SMMU_PIDR6, Peripheral ID6	0x00FD8			
SMMU_PIDR5, Peripheral ID5	0x00FD4			
SMMU_PIDR4, Peripheral ID4	0x00FD0	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code

4.6 TCU PMU registers

This section describes the *Performance Monitor Unit* (PMU) registers. The Performance Monitor counter registers, on a separate 64KB page, enable it to be paged for direct access from a Guest OS.

4.6.1 Registers

The TBU and TCU support the same PMCG registers.

These registers follow the register layout that the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) Performance Monitor Extension describes.

The following PMCG registers, that the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) defines, are implemented:

- SMMU_PMCG_EVCNTR{0-(TCUCFG_PMU_COUNTERS-1)}
- SMMU_PMCG_EVTYPER{0-(TCUCFG_PMU_COUNTERS-1)}
- SMMU_PMCG_SVR{0-(TCUCFG_PMU_COUNTERS-1)}
- SMMU_PMCG_SMRO
 - All counters share this mask register
 - The mask is 24 bits because the TCU uses 24-bit StreamIDs
- SMMU_PMCG_CNTENSET0
- SMMU_PMCG_CNTENCLR0
- SMMU_PMCG_INTENSET0
- SMMU_PMCG_INTSENCLR0
- SMMU_PMCG_OVSCLR0
- SMMU_PMCG_OVSSET0
- SMMU_PMCG_CAPR
- SMMU_PMCG_SCR
- SMMU_PMCG_CFGR. See [4.6.3 SMMU_PMCG_CFGR](#) on page 121.
- SMMU_PMCG_CR
- SMMU_PMCG_CEID{0-1}. See [4.6.4 SMMU_PMCG_CEID{0-1} registers](#) on page 122.
- SMMU_PMCG_IRQ_CTRL
- SMMU_PMCG_IRQ_CTRLACK
- SMMU_PMCG_AIDR, indicates SMMUV3.2
- SMMU_PMCG_ID_REGS

The following registers are not implemented, because the PMCG does not support MSIs:

- SMMU_PMCG_IRQ_CFG0
- SMMU_PMCG_IRQ_CFG1
- SMMU_PMCG_IRQ_CFG2
- SMMU_PMCG_IRQ_STATUS

The following registers are not implemented, because the PMCG implementation does not support MPAM:

- SMMU_PMCG_GMPAM
- SMMU_PMCG_MPAMIDR
- SMMU_PMCG_S_MPAMIDR

4.6.2 Events

In this description, a translation request corresponds to a translation slot allocation.

A single DTI translation request might correspond to multiple translation request events if:

- A translation results in a stall fault event and is restarted
- A translation results in a stall fault event when the Event queue is full, and is later retried when the Event queue becomes non-full

Each event indicates:

- Whether the SMMU_PMCG_SMR0 register can filter it
- For events that cannot be filtered, whether they are only visible when Secure events are visible by SMMU_PMCG_SCR.SO = 1

For more information about the architectural and **IMPLEMENTATION DEFINED** events that are implemented, see [3.3.2.1 SMMUv3 architectural performance events](#) on page 59.

The following events are also counted for prefetch accesses:

0x80-0x90

Walk cache events.

0x92-0x94

Configuration cache events.

0xC0-0xC8

RAS events.

4.6.3 SMMU_PMCG_CFGR

An MMU-700 implementation assumes fixed values for SMMU_PMCG_CFGR, and these values define behavioral aspects of the implementation.

For information about the SMMU_PMCG_CFGR field values, see [3.3.2.4 SMMUv3 PMU register architectural options](#) on page 63.

4.6.4 SMMU_PMCG_CEID{0-1} registers

The SMMU_PMCG_CEID{0-1} registers indicate the architectural events that are supported. They are described as 64-bit registers, but are accessed 32 bits at a time through the 32-bit PROG interface.

The following table shows the SMMU_PMCG_CEID{0-1} registers.

Table 4-19: SMMU_PMCG_CEID{0-1} registers

Address	Register	Value
0x02E20	SMMU_PMCG_CEID0	0x0000007F
0x02E28	SMMU_PMCG_CEID1	0x00000000

4.6.5 PMU ID registers

The PMU ID registers appear only in Performance Monitor Page 0. Page 1 does not contain any ID registers.

The following table shows the PMU ID registers.

Table 4-20: PMU ID registers

Address	Name	Field	Value	Description
0x02FFC	SMMU_PMCG_CIDR3, Component ID3	[7:0]	0xB1	Preamble
0x02FF8	SMMU_PMCG_CIDR2, Component ID2	[7:0]	0x05	Preamble
0x02FF4	SMMU_PMCG_CIDR1, Component ID1	[7:0]	0x90	Preamble
0x02FF0	SMMU_PMCG_CIDR0, Component ID0	[7:0]	0x0D	Preamble
0x02FEC	SMMU_PMCG_PIDR3, Peripheral ID3	[7:4]	MAX(<i>p_level</i> , ecorevnum)	REVAND, minor revision, where <i>p_level</i> is: 0 For p0 1 For p1
		[3:0]	0x00	CMOD
0x02FE8	SMMU_PMCG_PIDR2, Peripheral ID2	[7:4]	0x00	REVISION, major revision
		[3]	1	JEDEC-assigned value for DES always used
		[2:0]	3	DES_1: bits [6:4] bits of the JEP106 Designer code
0x02FE4	SMMU_PMCG_PIDR1, Peripheral ID1	[7:4]	0xB	DES_0: bits [3:0] of the JEP106 Designer code
		[3:0]	0x4	PART_1: bits [11:8] of the Part number

Address	Name	Field	Value	Description
0x02FE0	SMMU_PMC_G_PIDR0, Peripheral ID0	[7:0]	0x87	PART_0: bits [7:0] of the Part number
0x02FDC	SMMU_PMC_G_PIDR7, Peripheral ID7	-	RES0	Reserved
0x02FD8	SMMU_PMC_G_PIDR6, Peripheral ID6	-	RES0	Reserved
0x02FD4	SMMU_PMC_G_PIDR5, Peripheral ID5	-	RES0	Reserved
0x02FD0	SMMU_PMC_G_PIDR4, Peripheral ID4	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code
0x00FB8	SMMU_PMC_G_PMAUTHSTATUS	[7:0]	0x00	No authentication interface is implemented

The PMDEVARCH and PMDEVTYPE registers are implemented as the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) defines.

4.7 TCU microarchitectural registers

You can set the TCU microarchitectural registers at boot time to optimize TCU behavior for your system. We recommend that you use the default values for most systems.

The [4.7.7 TCU_SCR register](#) on page 131 is Secure-only. Non-secure access to this register is *Read-As-Zero* (RAZ)/*Write-Ignored* (WI).

TCU_SCR.NS_UARCH controls Non-secure access to registers in this section other than TCU_SCR. Non-secure accesses to these registers, when TCU_SCR.NS_UARCH = 0, are RAZ and WI.

The following registers:

- [4.7.1 TCU_CTRL register](#) on page 123
- [4.7.2 TCU_QOS register](#) on page 125
- [4.7.5 TCU_NODE_CTRLn register](#) on page 128
- [4.7.8 TCU_WC_SxLy_CMAX registers](#) on page 132

Can only be written when the following occur:

- SMMU_CR0.SMMUEN = 0.
- SMMU_CR0ACK.SMMUEN = 0.
- SMMU_S_CR0.SMMUEN = 0.
- SMMU_S_CR0ACK.SMMUEN = 0.

After modifying these registers, software must issue an INV_ALL operation using the SMMU_S_INIT register, before it sets SMMUEN to 1. Failure to issue the operation results in **UNPREDICTABLE** behavior.

4.7.1 TCU_CTRL register

The TCU Control register disables TCU features. If the hit rate of the individual walk cache is too low, you can disable individual walk caches to improve performance in some systems. Do not modify the AUX bits unless directed to do so by Arm®.

Configurations

This register is available in all configurations.

Attributes

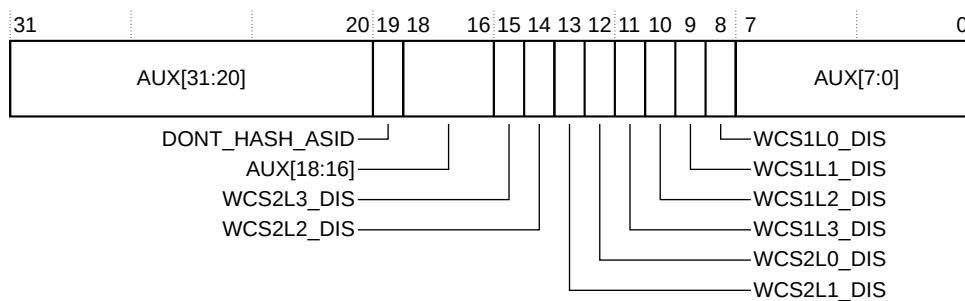
The TCU_CTRL register attributes are as follows:

Width	32-bit
Functional group	TCU microarchitectural features. See 4.7 TCU microarchitectural registers on page 123.
Address	0x08E00
offset	
Type	RW
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-1: TCU_CTRL register bit assignments



The following table shows the bit descriptions.

Table 4-21: TCU_CTRL register bit descriptions

Bits	Name	Description
[31:20]	AUX[31:20]	Reads the value that is written, but has no other effect
[19]	DONT_HASH_ASID	When set to 1, ASID is not used in the hash to create walk cache indices
[18:16]	AUX[18:16]	Reads the value that is written, but has no other effect

Bits	Name	Description
[15]	WCS2L3_DIS	Walk cache disable. When a bit of this field is set, it disables the corresponding stage and level of walk cache. WCS2L3_DIS is in bit [15], through to WCS1L0_DIS that is in bit [8].
[14]	WCS2L2_DIS	
[13]	WCS2L1_DIS	
[12]	WCS2L0_DIS	
[11]	WCS1L3_DIS	
[10]	WCS1L2_DIS	
[9]	WCS1L1_DIS	
[8]	WCS1L0_DIS	
[7:0]	AUX[7:0]	Reads the value written, but has no other effect

4.7.2 TCU_QOS register

This register selects the QoS value to attach to transactions issued from the TCU.



The QoS value that is associated with each transaction does not take account of other transactions that are blocked behind it. For example, although translations with a high priority setting in TCU_NODE_CTRLn are normally progressed before translations with a lower priority, it is possible for a low-priority page table walk to block a higher priority page table walk from being issued from the TCU.

Configurations

This register is available in all configurations.

Attributes

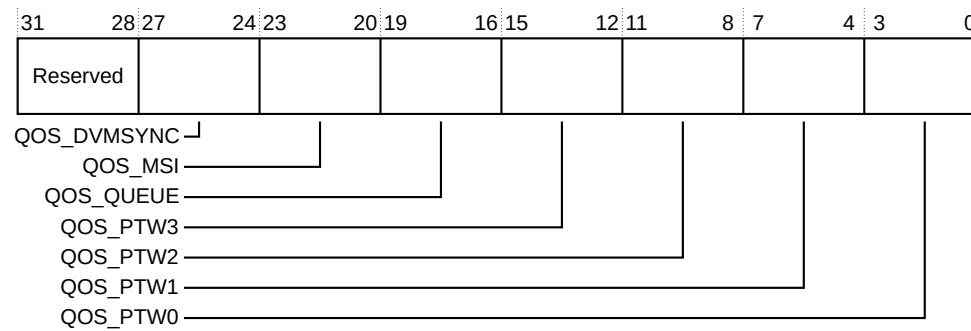
The TCU_QOS register attributes are as follows:

Width	32-bit
Functional group	TCU microarchitectural features. See 4.7 TCU microarchitectural registers on page 123.
Address offset	0x08E04
Type	RW
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-2: TCU_QOS register bit assignments



The following table shows the bit descriptions.

Table 4-22: TCU_QOS register bit descriptions

Bits	Name	Description
[31:28]	-	Reserved
[27:24]	QOS_DVMSYNC	QoS level to use for DVM Sync Completion messages
[23:20]	QOS_MSI	QoS level to use for MSIs
[19:16]	QOS_QUEUE	QoS level to use for queue accesses
[15:12]	QOS_PTW3	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRLn.PRIORITY = 3
[11:8]	QOS_PTW2	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRLn.PRIORITY = 2
[7:4]	QOS_PTW1	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRLn.PRIORITY = 1
[3:0]	QOS_PTW0	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRLn.PRIORITY = 0

4.7.3 TCU_CFG register

This section describes the TCU Configuration Information register.

Configurations

This register is available in all configurations.

Attributes

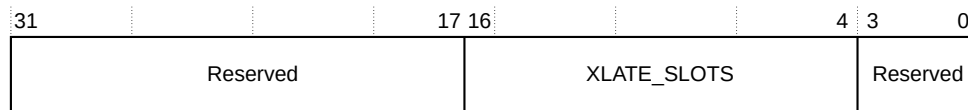
The TCU_CFG register attributes are as follows:

Width	32-bit
Functional group	TCU microarchitectural features. See 4.7 TCU microarchitectural registers on page 123.
Address offset	0x08E08
Type	RO

Bit descriptions

The following figure shows the bit assignments.

Figure 4-3: TCU_CFG register bit assignments



The following table shows the bit descriptions.

Table 4-23: TCU_CFG register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved
[16:4]	XLATE_SLOTS	Number of translation slots that are available to be shared between all nodes. The value is <i>TCUCFG_XLATE_SLOTS</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.
[3:0]	-	Reserved

4.7.4 TCU_STATUS register

This section describes the TCU Status Information register.

Configurations

This register is available in all configurations.

Attributes

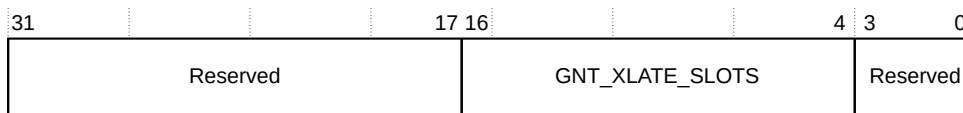
The TCU_STATUS register attributes are as follows:

Width	32-bit
Functional group	TCU microarchitectural features. See 4.7 TCU microarchitectural registers on page 123.
Address	0x08E10
offset	
Type	RO
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-4: TCU_STATUS register bit assignments



The following table shows the bit descriptions.

Table 4-24: TCU_STATUS register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved
[16:4]	GNT_XLATE_SLOTS	Total number of translation slots that are currently allocated to all connected nodes. This information can be useful for debugging purposes.
[3:0]	-	Reserved

4.7.5 TCU_NODE_CTRLn register

The TCU_NODE_CTRLn register controls how the TCU communicates with a single DTI master, either a TBU or a PCIe Root Complex implementing ATS.

Each DTI master has a node ID, with the control register for:

Node 0

At address 0x09000

Node 1

At address 0x09004

The number of registers that are implemented corresponds to the value of *TCUCFG_NUM_TBU*. See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94.

All bits [31:16] are implemented and are readable and writable, regardless of the number of ports the attached DTI node has. The following expression determines the priority that is associated with a transaction:

PRIORITY = (TCU_NODE_CTRLx.PRIORITY_SEL ?
TCU_NODE_CTRLx[((DTI_x_TRANS_REQ.QOS[2:0] × 2) + 16) + :2] :
TCU_NODE_CTRLx.DEFAULT_PRIORITY)

If a DTI node sends a translation request with an incorrect QoS value, the programmed value for the LTI port indicated in the QoS field is used because it is not possible for the TCU to determine what is a valid or invalid port number.



Note

When the priority level is established, these are translated into QoS values by selection of the appropriate QOS_PTW* field from the TCU_QOS register, which override the value in the DTI *trans_req* message. This means that the transaction has its QoS value overridden but no additional information is required to be associated with the transaction. If the PRI level association is updated, the rest of the mechanism requires no alteration.

Configurations

This register is available in all configurations.

Attributes

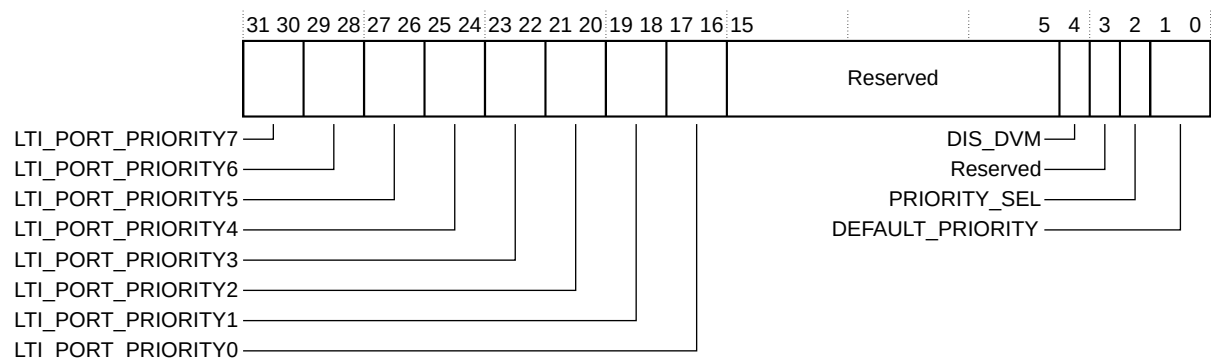
The TCU_NODE_CTRLn register attributes are as follows:

Width	32-bit
Functional group	TCU microarchitectural features. See 4.7 TCU microarchitectural registers on page 123.
Address offset	0x09000-0x093FC
Type	RW
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-5: TCU_NODE_CTRLn register bit assignments



The following table shows the bit descriptions.

Table 4-25: TCU_NODE_CTRLn register bit descriptions

Bits	Name	Description
[31:30]	LTI_PORT_PRIORITY7	Priority for LTI Port 7 for this node, if the port exists
[29:28]	LTI_PORT_PRIORITY6	Priority for LTI Port 6 for this node, if the port exists
[27:26]	LTI_PORT_PRIORITY5	Priority for LTI Port 5 for this node, if the port exists
[25:24]	LTI_PORT_PRIORITY4	Priority for LTI Port 4 for this node, if the port exists

Bits	Name	Description
[23:22]	LTI_PORT_PRIORITY3	Priority for LTI Port 3 for this node, if the port exists
[21:20]	LTI_PORT_PRIORITY2	Priority for LTI Port 2 for this node, if the port exists
[19:18]	LTI_PORT_PRIORITY1	Priority for LTI Port 1 for this node, if the port exists
[17:16]	LTI_PORT_PRIORITY0	Priority for LTI Port 0 for this node
[15:5]	-	Reserved
[4]	DIS_DVM	Disable DVM. When this bit is set, the node does not participate in DVM invalidation. This setting can improve performance if the node can be slow to respond to invalidations issued over DTI. This bit is only used for TBU nodes. It is ignored for ATS nodes.
[3]	-	Reserved
[2]	PRIORITY_SEL	Select priority between DTI node ID (default) and LTI port: 0 When this bit is set to 0, the priority of all translation requests from the DTI master are given the priority that the DEFAULT_PRIORITY field specifies. 1 When this bit is set to 1, transactions from the DTI master are given the priority that is set in the LTI_PORT_PRIORITY _m field, where the QOS[2:0] bits in the DTI translation request provide the value of <i>m</i> .
[1:0]	PRI_LEVEL	Default Priority level for this DTI master. Translation requests from a node with a higher priority level are normally progressed before translation requests from a node with a lower priority level.

4.7.6 TCU_NODE_STATUS_n register

The TCU_NODE_STATUS_n register provides status for each node, similarly to TCU_NODE_CTRL_n. Each node has a single status register.

The number of registers that are implemented corresponds to the value of *TCUCFG_NUM_TBU*. See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94.

Configurations

This register is available in all configurations.

Attributes

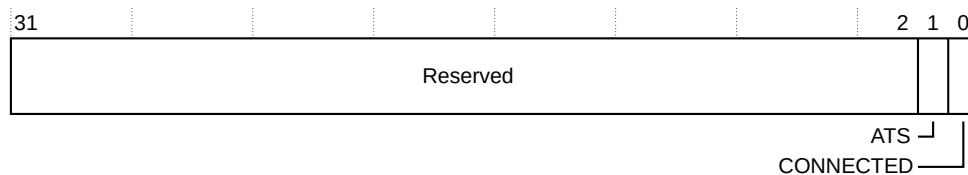
The TCU_NODE_STATUS_n register attributes are as follows:

Width	32-bit
Functional group	TCU microarchitectural features. See 4.7 TCU microarchitectural registers on page 123.
Address offset	0x09400-0x097FC
Type	RO

Bit descriptions

The following figure shows the bit assignments.

Figure 4-6: TCU_NODE_STATUS register bit assignments



The following table shows the bit descriptions.

Table 4-26: TCU_NODE_STATUSn register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1]	ATS	Indicates whether the node implements ATS: 0 The node is a TBU connected using DTI-TBU 1 The node is a PCIe Root Complex supporting ATS, connected using DTI-ATS This bit is only valid when CONNECTED = 1. When CONNECTED = 0, this bit is 0.
[0]	CONNECTED	Indicates whether the DTI link for this node is in the connected state: 0 Node currently not in the connected state, including the states transitioning to and from connected state 1 Node currently in the connected state When not connected, write accesses to TBU registers are ignored and read accesses return 0. However, the state might change between reading this register and attempting to access the TBU.

4.7.7 TCU_SCR register

The TCU Secure Control register controls whether Non-secure software is permitted to access each TCU register group.

This register does not control Secure access to the Performance Monitor registers. The SMMU_PMCGR_SCR register controls Secure access to these registers as the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) defines.

Configurations

This register is available in all configurations.

Attributes

The TCU_SCR register attributes are as follows:

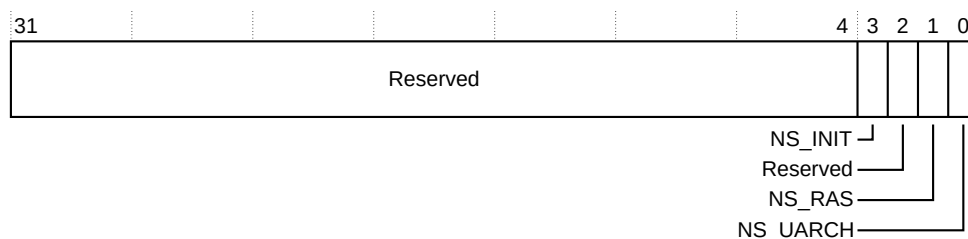
Width	32-bit
Functional group	TCU microarchitectural features. See 4.7 TCU microarchitectural registers on page 123.
Address offset	0x08E18
Type	Secure, RW

Reset value **sec_override**. See [A.1.12 TCU tie-off signals](#) on page 205.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-7: TCU_SCR register bit assignments



The following table shows the bit descriptions.

Table 4-27: TCU_SCR register bit descriptions

Bits	Name	Description
[31:4]	–	Reserved
[3]	NS_INIT	Non-secure register access that is permitted to the SMMU_S_INIT register
[2]	–	Reserved
[1]	NS_RAS	Non-secure register access that is permitted for RAS registers. When this bit is 0, Non-secure writes to the following register addresses are ignored, and Non-secure reads return zero: 0x08E80-0x08EC0. The sec_override input sets the reset value of this signal. See A.1.12 TCU tie-off signals on page 205.
[0]	NS_UARCH	Non-secure register access is permitted for microarchitectural registers When this bit is 0, Non-secure writes to the following register addresses are ignored, and Non-secure reads return zero: 0x08E00-0x08E7C 0x09000-0x093FC The sec_override input sets the reset value of this signal. See A.1.12 TCU tie-off signals on page 205. If Secure translation might be used, we recommend that software does not set this bit.

4.7.8 TCU_WC_SxLy_CMAX registers

TCU_WC_SxLy_CMAX registers enable you to set maximum capacities for the TCU walk cache RAMs, per stage and level.

The encoding of the TCU_WC_SxLy_CMAX registers is the same as the encoding for the MPAMCFG_CMAX registers that the [Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring \(MPAM\), for Armv8-A](#) defines. These registers are readable and writable registers.

The following table describes the TCU_WC_SxLy_CMAX registers.

Table 4-28: TCU_WC_SxLy_CMAX registers

Address	Name	Field	Position	Meaning
0x09800	TCU_WC_S1L0_CMAX	CMAX	[15:0]	Maximum capacity for TCU Walk Cache stage 1 level 0
0x09804	TCU_WC_S1L1_CMAX	CMAX	[15:0]	Maximum capacity for TCU Walk Cache stage 1 level 1
0x09808	TCU_WC_S1L2_CMAX	CMAX	[15:0]	Maximum capacity for TCU Walk Cache stage 1 level 2
0x0980C	TCU_WC_S1L3_CMAX	CMAX	[15:0]	Maximum capacity for TCU Walk Cache stage 1 level 3
0x09810	TCU_WC_S2L0_CMAX	CMAX	[15:0]	Maximum capacity for TCU Walk Cache stage 2 level 0
0x09814	TCU_WC_S2L1_CMAX	CMAX	[15:0]	Maximum capacity for TCU Walk Cache stage 2 level 1
0x09818	TCU_WC_S2L2_CMAX	CMAX	[15:0]	Maximum capacity for TCU Walk Cache stage 2 level 2
0x0981C	TCU_WC_S2L3_CMAX	CMAX	[15:0]	Maximum capacity for TCU Walk Cache stage 2 level 3



The implementation defines how many bits are used. For MMU-700, the number of bits is 8b. However, because the value represents a fixed-point binary fraction, it is the MSB of those 16 bits that are significant, so only bits [15:8] have any impact.

4.8 TCU RAS registers

This section describes *Reliability, Availability, and Serviceability* (RAS).

The RAS registers implement the RAS Extension registers, single record format.

Non-secure accesses to these registers, when `tcu_scr.ns_ras = 0`, are RAZ/WI. See [4.7.7 TCU_SCR register](#) on page 131.

The RAS registers enable software to monitor the following classes of error:

- *Corrected Errors* (CEs) in the RAMs used by the configuration cache
- CEs in the RAMs used by the walk caches

4.8.1 TCU_ERRFR register

Use the TCU Error Feature register to discover how the TCU handles errors.

Configurations

This register is available in all configurations.

Attributes

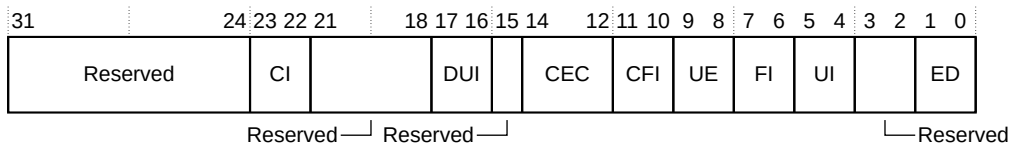
The TCU_ERRFR register attributes are as follows:

Width	32-bit
Functional group	<i>Reliability, Availability, and Serviceability (RAS)</i> . See 4.8 TCU RAS registers on page 133.
Address	0x08E80
offset	
Type	S, RO

Bit descriptions

The following figure shows the bit assignments.

Figure 4-8: TCU_ERRFR register bit assignments



The following table shows the bit descriptions.

Table 4-29: TCU_ERRFR register bit descriptions

Bits	Name	Description	Value
[31:24]	-	Reserved	-
[23:22]	CI	Critical Error Interrupt is always enabled	01b
[21:18]	-	Reserved	-
[17:16]	DUI	Does not support this feature	00b
[15]	-	Reserved	-
[14:12]	CEC	Does not implement the standard corrected error counter model	000b
[11:10]	CFI	Does not support this feature	00b
[9:8]	UE	In-band error signaling feature is always enabled	01b
[7:6]	FI	Fault handling interrupt is controllable	10b
[5:4]	UI	Error Recovery Interrupt always enabled for UE	01b
[3:2]	-	Reserved	-
[1:0]	ED	Error detection is always enabled	01b

4.8.2 TCU_ERRCTLR register

Use the TCU Error Control register to enable fault handling interrupts.

Configurations

This register is available in all configurations.

Attributes

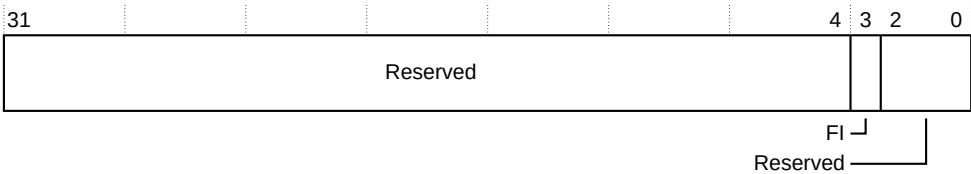
The TCU_ERRCTLR register attributes are as follows:

Width	32-bit
Functional group	Reliability, Availability, and Serviceability (RAS). See 4.8 TCU RAS registers on page 133.
Address offset	0x08E88
Type	S, RW
Reset value	1

Bit descriptions

The following figure shows the bit assignments.

Figure 4-9: TCU_ERRCTLR register bit assignments



The following table shows the bit descriptions.

Table 4-30: TCU_ERRCTLR register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3]	FI	Fault handling interrupt enable
[2:0]	-	Reserved

4.8.3 TCU_ERRSTATUS register

Use the TCU Error Control register to enable fault handling interrupts.

Certain bits in this register are cleared by writing a 1 to their bit position. These writes are ignored in certain circumstances to avoid race conditions where a new error has occurred which software has not yet observed.

Configurations

This register is available in all configurations.

Attributes

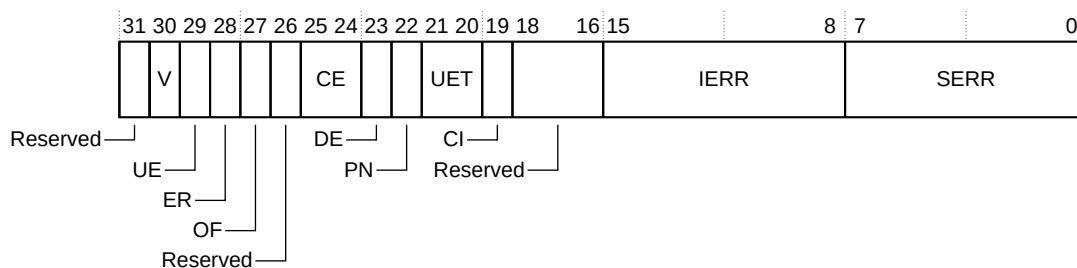
The TCU_ERRSTATUS register attributes are as follows:

Width	32-bit
Functional group	<i>Reliability, Availability, and Serviceability (RAS)</i> . See 4.8 TCU RAS registers on page 133.
Address offset	0x08E90
Type	Secure, RW
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-10: TCU_ERRSTATUS register bit assignments



The following table shows the bit descriptions.

Table 4-31: TCU_ERRSTATUS register bit descriptions

Bits	Name	Description
[31]	-	Reserved
[30]	V	<p>Status Register valid. The possible values of this bit are:</p> <p>0 ERRSTATUS is not valid</p> <p>1 ERRSTATUS is valid. At least one error has been recorded</p> <p>If any of the UE, DE, or CE bits are set to 1, and are not being cleared to 0 in the same write, direct writes to this bit are ignored. This bit is read/write-one-to-clear.</p> <p>This bit resets to zero on a reset.</p>

Bits	Name	Description
[29]	UE	<p>Uncorrected error, or errors. The possible values of this bit are:</p> <p>0 No errors that could not be corrected or deferred</p> <p>1 At least one error detected that has not been corrected or deferred</p> <p>Direct writes to this bit are ignored if the OF bit is set to 1 and is not being cleared to zero in the same write. This bit is read/write-one-to-clear.</p>
[28]	ER	<p>Error Reported. The possible values of this bit are:</p> <p>0 No in-band error (External abort) is reported</p> <p>1 The node to the master making the access or other transaction signaled an External abort</p> <p>Writes to this field are ignored.</p>
[27]	OF	<p>Overflow.</p> <p>Multiple errors are detected. This bit is set to 1 when:</p> <ul style="list-style-type: none"> • An Uncorrected error is detected and the previous error syndrome is kept because UE == 1 • A Deferred error is detected and the previous error syndrome is discarded because DE == 1 • A Corrected error is detected and the CE field might be updated for the new Corrected error • A Deferred error is detected and UE == 1 • A Corrected error is detected and either or both the DE or UE bits are set to 1 <p>This bit is cleared by writing a 1 to it. A write of 0 is ignored.</p>
[26]	-	Reserved
[25:24]	CE	<p>Correctable Error, or errors.</p> <p>00b No correctable errors recorded</p> <p>10b At least one Corrected error recorded</p> <p>Other values are Reserved.</p> <p>This field is cleared by writing 11b to it. If OF is set and not being cleared, the write is ignored. A write of any value other than 11b is ignored.</p>
[23]	DE	<p>Deferred error, or errors. The possible values of this bit are:</p> <p>0 No errors were deferred</p> <p>1 At least one error was not corrected and deferred</p> <p>This error is raised when wpoison is set in BIU.</p> <p>If the OF bit is set to 1 and is not being cleared to 0 in the same write, direct writes to this bit are ignored.</p> <p>This bit is read/write-one-to-clear.</p>
[22]	PN	<p>Poison. The possible values of this bit are:</p> <p>0 Uncorrected error or deferred error is recorded because a corrupt value was detected, for example, by an <i>Error Detection Code</i> (EDC)</p> <p>1 Uncorrected error or deferred error is recorded because a poison value was detected</p> <p>Writes to this field are ignored.</p>

Bits	Name	Description
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. The possible values of this field are: 0b00 Uncorrected error, Uncontainable error (UC) 0b11 Uncorrected error, Signaled or Recoverable error (UER)
[19]	CI	Indicates whether a critical error condition has been recorded. The possible values of this bit are: 0 No critical error condition 1 Critical error condition Writes to this field are ignored.
[18:16]	-	Reserved
[15:8]	IERR	IMPLEMENTATION DEFINED error code. When SERR≠0, this field indicates the source of the error: 12h PIU CMD RPOISON 11h TMU CCB MCC DATA 10h TMU CCB MCC TAGS 0Fh TMU WCB MWC DATA 0Eh TMU WCB MWC TAGS 0Dh TMU CCB MCC REPL 0Ch TMU CCB MCC PCNT 0Bh TMU CCB MCC PLIM 0Ah TMU WCB MWC REPL 09h TMU WCB MWC PCNT 08h TMU WCB MWC PLIM 07h Reserved 06h Reserved 05h TMU HTTU RAM 04h TMU TWB WMB SCRATCH 03h TMU TWB WMB WLK STATUS 02h TMU TWB WMB LKP STATUS 01h TMU HZU PTR 00h TMU TWB BSU Writes to this field are ignored.
[7:0]	SERR	The error code provides information about the earliest unacknowledged error. It can contain the following values: 2 Single or double error from RAMs that are not CCB or WCB TAGS or DATA 8 Single or double error from CCB or WCB data 9 Single or double error from CCB or WCB tags 21 Poisoned data read from downstream All other values are reserved. Writes to this field are ignored.

4.8.4 TCU_ERRGEN register

Use the TCU Error Generation Register to test software for when a RAS error occurs in the RAM.

The field locations are the same as for [4.8.3 TCU_ERRSTATUS register](#) on page 135.

When this register is updated, the TCU_ERRSTATUS register is also updated with the same value, as long as the write data generates a valid error record.

A write to ERRGEN is valid if all the following is true:

- ERRGEN.V is set
- At least one of the following is true (CE is legal if $CE == 2'b00$ or $CE == 2'b10$):
 - ERRGEN.UE is set and CE is legal
 - ERRGEN.DE is set and CE is legal
 - ERRGEN.CE is set to $2'b10$
- One of the following is true:
 - $UET == 2'b00$
 - $UET == 2'b11$ and $UE == 1$
- If a valid error record is written, then the appropriate interrupt, or interrupts, are also generated.



This register has identical fields to [4.8.3 TCU_ERRSTATUS register](#) on page 135.

Configurations

This register is available in all configurations.

Attributes

The TCU_ERRGEN register attributes are as follows:

Width	64-bit
Functional group	<i>Reliability, Availability, and Serviceability (RAS).</i> See 4.8 TCU RAS registers on page 133.
Address offset	0x08EC0
Type	Secure, RW
Reset value	0

4.9 TCU system discovery registers

This section describes the TCU system discovery registers.

4.9.1 TCU_SYSDISC0 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

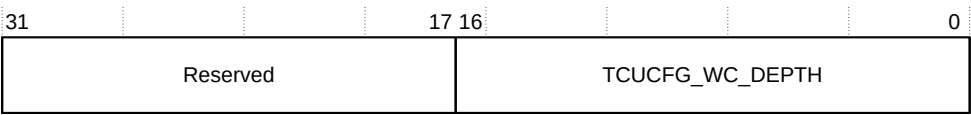
The TCU_SYSDISC0 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E34
offset	
Type	RO
Reset value	<i>TCUCFG_WC_DEPTH</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-11: TCU_SYSDISC0 register bit assignments



The following table shows the bit descriptions.

Table 4-32: TCU_SYSDISC0 register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved
[16:0]	TCUCFG_WC_DEPTH	The read data reflects the chosen parameter value, for example: 17'h0_0008 : 8 17'h1_0000 : 65536

4.9.2 TCU_SYSDISC1 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

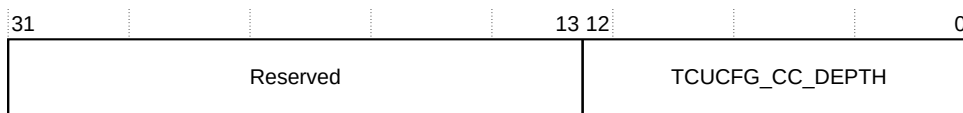
The TCU_SYSDISC1 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address offset	0x08E38
Type	RO
Reset value	<i>TCUCFG_CC_DEPTH</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-12: TCU_SYSDISC1 register bit assignments



The following table shows the bit descriptions.

Table 4-33: TCU_SYSDISC1 register bit descriptions

Bits	Name	Description
[31:13]	-	Reserved
[12:0]	TCUCFG_CC_DEPTH	<p>The read data reflects the chosen parameter value, for example:</p> <p>13'h0004 : 4</p> <p>....</p> <p>13'h1000 : 4096</p>

4.9.3 TCU_SYSDISC2 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

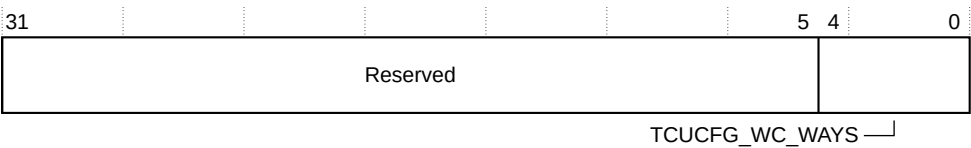
The TCU_SYSDISC2 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E3C
offset	
Type	RO
Reset value	<i>TCUCFG_WC_WAYS</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-13: TCU_SYSDISC2 register bit assignments



The following table shows the bit descriptions.

Table 4-34: TCU_SYSDISC2 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved
[4:0]	TCUCFG_WC_WAYS	The read data reflects the chosen parameter value, for example: 5'h04 : 4 5'h10 : 16

4.9.4 TCU_SYSDISC3 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

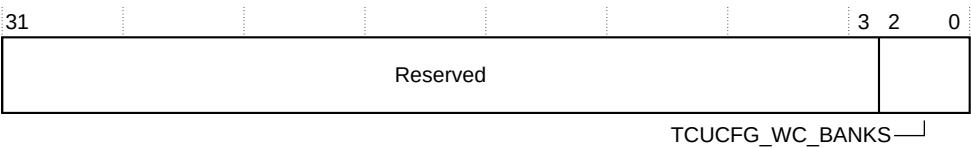
The TCU_SYSDISC3 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E40
offset	
Type	RO
Reset value	TCUCFG_WC_BANKS. See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-14: TCU_SYSDISC3 register bit assignments



The following table shows the bit descriptions.

Table 4-35: TCU_SYSDISC3 register bit descriptions

Bits	Name	Description
[31:3]	-	Reserved
[2:0]	TCUCFG_WC_BANKS	The read data reflects the chosen parameter value, for example: 3'b001 : 1 3'b100 : 4

4.9.5 TCU_SYSDISC4 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

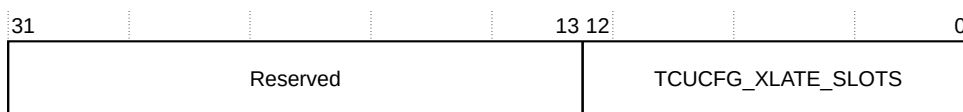
The TCU_SYSDISC4 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E44
offset	
Type	RO
Reset value	<i>TCUCFG_XLATE_SLOTS</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-15: TCU_SYSDISC4 register bit assignments



The following table shows the bit descriptions.

Table 4-36: TCU_SYSDISC4 register bit descriptions

Bits	Name	Description
[31:13]	-	Reserved
[12:0]	TCUCFG_XLATE_SLOTS	The read data reflects the chosen parameter value, for example: 13'h0004 : 4 13'h1000 : 4096

4.9.6 TCU_SYSDISC5 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

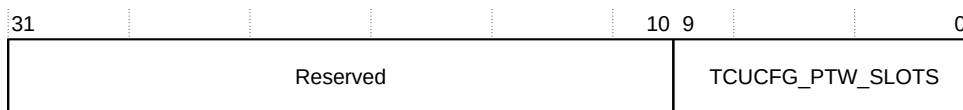
The TCU_SYSDISC5 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E48
offset	
Type	RO
Reset value	TCUCFG_PTW_SLOTS. See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-16: TCU_SYSDISC5 register bit assignments



The following table shows the bit descriptions.

Table 4-37: TCU_SYSDISC5 register bit descriptions

Bits	Name	Description
[31:10]	-	Reserved
[9:0]	TCUCFG_PTW_SLOTS	The read data reflects the chosen parameter value, for example: 9'h002 : 2 9'h200 : 512

4.9.7 TCU_SYSDISC6 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

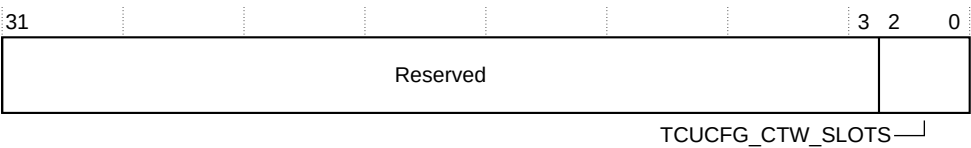
The TCU_SYSDISC6 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E4C
offset	
Type	RO
Reset value	TCUCFG_CTW_SLOTS. See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-17: TCU_SYSDISC6 register bit assignments



The following table shows the bit descriptions.

Table 4-38: TCU_SYSDISC6 register bit descriptions

Bits	Name	Description
[31:3]	-	Reserved
[2:0]	TCUCFG_CTW_SLOTS	The read data reflects the chosen parameter value, for example: 3'b001 : 1 3'b100 : 4

4.9.8 TCU_SYSDISC7 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

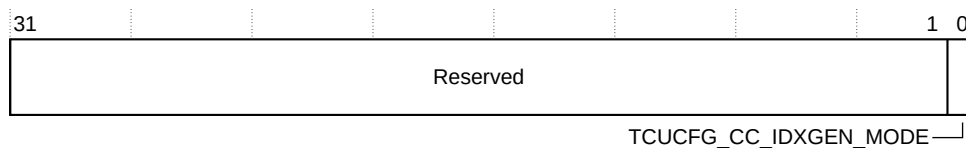
The TCU_SYSDISC7 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E50
offset	
Type	RO
Reset value	<i>TCUCFG_CC_IDXGEN_MODE</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-18: TCU_SYSDISC7 register bit assignments



The following table shows the bit descriptions.

Table 4-39: TCU_SYSDISC7 register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	TCUCFG_CC_IDXGEN_MODE	The read data reflects the chosen parameter value, for example: 1'b0 : 0 1'b1 : 1

4.9.9 TCU_SYSDISC8 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

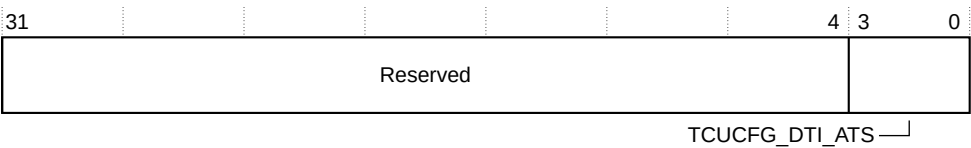
The TCU_SYSDISC8 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E54
offset	
Type	RO
Reset value	<i>TCUCFG_DTI_ATS</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-19: TCU_SYSDISC8 register bit assignments



The following table shows the bit descriptions.

Table 4-40: TCU_SYSDISC8 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3:0]	TCUCFG_DTI_ATS	The read data reflects the chosen parameter value, for example: 4'b0000 : 0 4'b1000 : 8

4.9.10 TCU_SYSDISC9 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

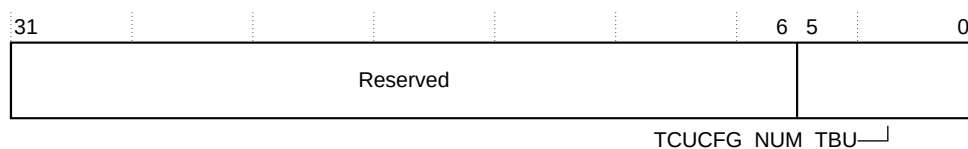
The TCU_SYSDISC9 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E58
offset	
Type	RO
Reset value	<i>TCUCFG_NUM_TBU</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-20: TCU_SYSDISC9 register bit assignments



The following table shows the bit descriptions.

Table 4-41: TCU_SYSDISC9 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved
[5:0]	TCUCFG_NUM_TBU	<p>The read data reflects the chosen parameter value, for example:</p> <p>6'h01 : 1</p> <p>....</p> <p>6'h3E : 62</p> <p>Note: Legal values are 14 and 62.</p>

4.9.11 TCU_SYSDISC10 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

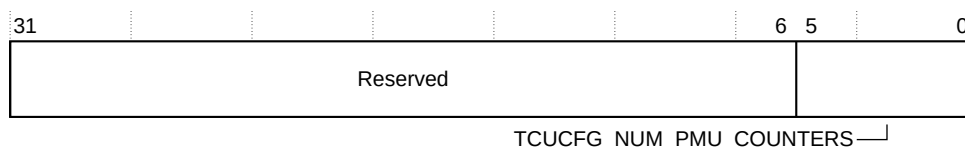
The TCU_SYSDISC10 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E5C
offset	
Type	RO
Reset value	TCUCFG_NUM_PMU_COUNTERS. See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-21: TCU_SYSDISC10 register bit assignments



The following table shows the bit descriptions.

Table 4-42: TCU_SYSDISC10 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved
[5:0]	TCUCFG_NUM_PMU_COUNTERS	The read data reflects the chosen parameter value, for example: 6'h04 : 4 6'h20 : 32

4.9.12 TCU_SYSDISC11 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

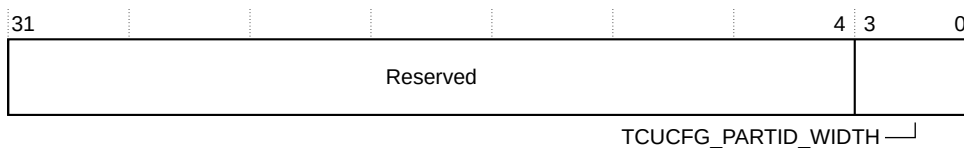
The TCU_SYSDISC11 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address offset	0x08E60
Type	RO
Reset value	<i>TCUCFG_PARTID_WIDTH</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-22: TCU_SYSDISC11 register bit assignments



The following table shows the bit descriptions.

Table 4-43: TCU_SYSDISC11 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3:0]	TCUCFG_PARTID_WIDTH	<p>The read data reflects the chosen parameter value, for example:</p> <p>4'b0001 : 1</p> <p>....</p> <p>4'b1001 : 9</p>

4.9.13 TCU_SYSDISC12 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

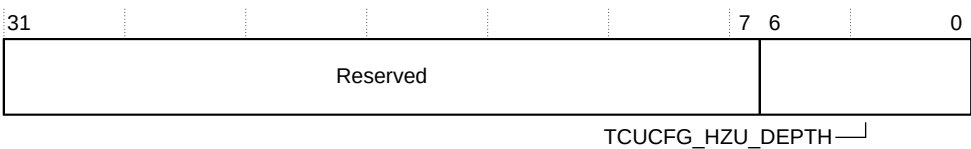
The TCU_SYSDISC12 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E64
offset	
Type	RO
Reset value	TCUCFG_HZU_DEPTH. See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-23: TCU_SYSDISC12 register bit assignments



The following table shows the bit descriptions.

Table 4-44: TCU_SYSDISC12 register bit descriptions

Bits	Name	Description
[31:7]	-	Reserved
[6:0]	TCUCFG_HZU_DEPTH	The read data reflects the chosen parameter value, for example: 7'h02 : 2 7'h40 : 64

4.9.14 TCU_SYSDISC13 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

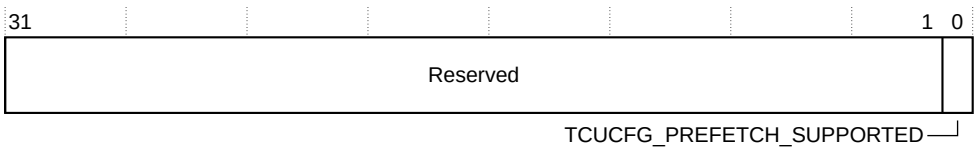
The TCU_SYSDISC13 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E68
offset	
Type	RO
Reset value	<i>TCUCFG_PREFETCH_SUPPORTED</i> . See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-24: TCU_SYSDISC13 register bit assignments



The following table shows the bit descriptions.

Table 4-45: TCU_SYSDISC13 register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	TCUCFG_PREFETCH_SUPPORTED	The read data reflects the chosen parameter value, for example: 1'b0 : 0 1'b1 : 1

4.9.16 TCU_SYSDISC15 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

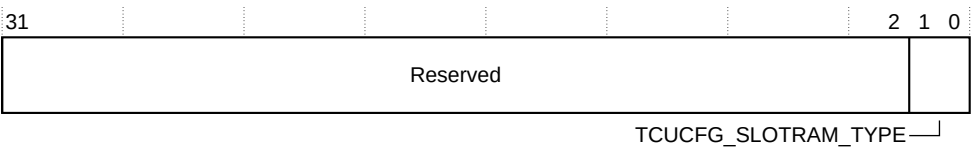
The TCU_SYSDISC15 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E70
offset	
Type	RO
Reset value	TCUCFG_SLOTRAM_TYPE. See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-26: TCU_SYSDISC15 register bit assignments



The following table shows the bit descriptions.

Table 4-47: TCU_SYSDISC15 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1:0]	TCUCFG_SLOTRAM_TYPE	The read data reflects the chosen parameter value, for example: 2'b00 : 0 2'b01 : 1

4.9.17 TCU_SYSDISC16 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

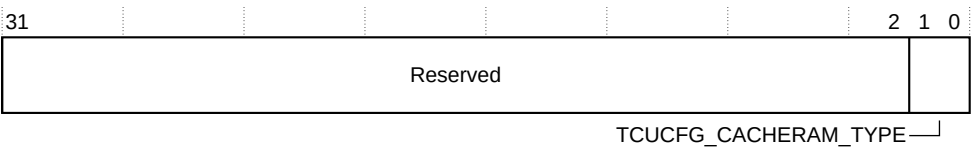
The TCU_SYSDISC16 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E74
offset	
Type	RO
Reset value	TCUCFG_CACHERAM_TYPE. See 3.5.2 Translation Control Unit buffer configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-27: TCU_SYSDISC16 register bit assignments



The following table shows the bit descriptions.

Table 4-48: TCU_SYSDISC16 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1:0]	TCUCFG_CACHERAM_TYPE	The read data reflects the chosen parameter value, for example: 2'b00 : 0 2'b01 : 1

4.9.18 TCU_SYSDISC17 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

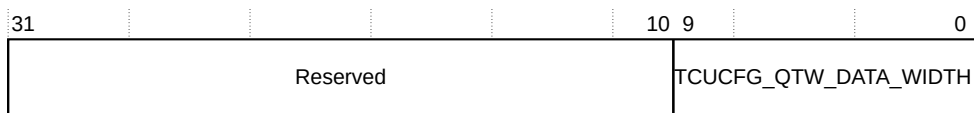
The TCU_SYSDISC17 register attributes are as follows:

Width	32-bit
Functional group	TCU system discovery registers. See 4.9 TCU system discovery registers on page 139.
Address	0x08E78
offset	
Type	RO
Reset value	TCUCFG_QTW_DATA_WIDTH. See 3.5.1 Translation Control Unit I/O configuration parameters on page 94.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-28: TCU_SYSDISC17 register bit assignments



The following table shows the bit descriptions.

Table 4-49: TCU_SYSDISC17 register bit descriptions

Bits	Name	Description
[31:10]	-	Reserved
[9:0]	TCUCFG_QTW_DATA_WIDTH	<p>The read data reflects the chosen parameter value, for example:</p> <p>10'h040 : 64</p> <p>....</p> <p>10'h200 : 512</p>

4.10 TCU PIU integration registers

This section describes the *Programmer Interface Unit* (PIU) integration registers.

4.10.1 ITEN register for the TCU

Integration mode register for the TCU. When integration mode is enabled, the values of certain TCU input pins are made visible in the ITIN register for the TCU.

The values that are written to the ITOP for the TCU control the values of certain TCU output pins to help system integrators to integrate the SMMU into the system and perform basic connectivity checks.

Configurations

This register is available in all configurations.

Attributes

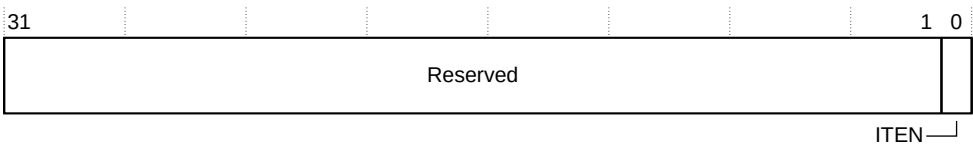
The ITEN register attributes are as follows:

Width	32-bit
Functional group	Performance monitor. See 4.8 TCU RAS registers on page 133.
Address offset	0x08E20
Type	RW
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-29: ITEN register bit assignments



The following table shows the bit descriptions.

Table 4-50: ITEN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved

Bits	Name	Description
[10]	event_q_irpt_ns	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to event_q_irpt_ns
[9]	event_q_irpt_s	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to event_q_irpt_s
[8]	pri_q_irpt_ns	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to pri_q_irpt_ns
[7]	cmd_sync_irpt_ns	0 When ITEN.ITEN == 0, SBZ. 1 When ITEN.ITEN == 1, the value driven to cmd_sync_irpt_ns
[6]	cmd_sync_irpt_s	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to cmd_sync_irpt_s
[5]	global_irpt_ns	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to global_irpt_ns
[4]	global_irpt_s	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to global_irpt_s
[3]	evento	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to evento
[2]	ras_fhi	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to ras_fhi
[1]	ras_eri	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to ras_eri
[0]	ras_cri	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to ras_cri

See:

- [A.1.9 TCU interrupt signals](#) on page 202
- [A.1.11 TCU event interface signal](#) on page 203

4.11 TCU TMU integration registers

This section describes the TCU *Translation Management Unit* (TMU) integration registers.

4.11.1 ITOP register for the TCU Translation Management Unit

This section describes the ITOP register for the TCU *Translation Management Unit* (TMU).

Configurations

This register is available in all configurations.

Attributes

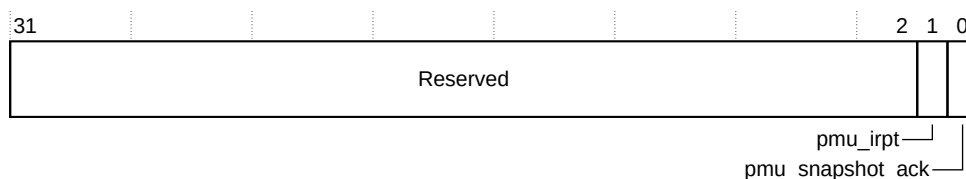
The ITOP register attributes are as follows:

Width	32-bit
Functional group	TCU <i>Translation Management Unit</i> (TMU) integration registers. See 4.11 TCU TMU integration registers on page 160.
Address	0x08E2C
offset	
Type	RW
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-31: ITOP register bit assignments



The following table shows the bit descriptions.

Table 4-52: ITOP register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved, SBZ
[1]	pmu_irpt	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to pmu_irpt
[0]	pmu_snapshot_ack	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to pmusnapshot_ack

See:

- [A.1.9 TCU interrupt signals](#) on page 202
- [A.1.5 TCU PMU snapshot interface signals](#) on page 200

4.11.2 ITIN register for the TCU Translation Management Unit

This section describes the ITIN register for the TCU *Translation Management Unit* (TMU).

Configurations

This register is available in all configurations.

Attributes

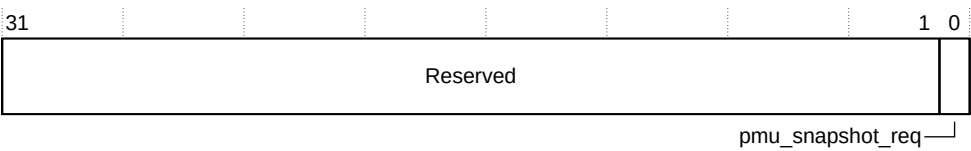
The ITIN register attributes are as follows:

Width	32-bit
Functional group	TCU <i>Translation Management Unit</i> (TMU) integration registers. See 4.11 TCU TMU integration registers on page 160.
Address offset	0x08E30
Type	RO
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-32: ITIN register bit assignments



The following table shows the bit descriptions.

Table 4-53: ITIN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved, SBZ
[0]	pmu_snapshot_req	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, reflects pmusnapshot_req

See [A.1.5 TCU PMU snapshot interface signals](#) on page 200.

4.12 TBU component and peripheral ID registers

This section describes the TBU component and peripheral ID registers.

The following table shows the TBU component and peripheral ID.

Table 4-54: TBU component and peripheral ID registers

Name	Offset	Field	Value	Description
SMMU_CIDR3, Component ID3	0x00FFC	[7:0]	0xB1	Preamble
SMMU_CIDR2, Component ID2	0x00FF8	[7:0]	0x05	Preamble
SMMU_CIDR1, Component ID1	0x00FF4	[7:0]	0xF0	Preamble
SMMU_CIDR0, Component ID0	0x00FF0	[7:0]	0x0D	Preamble
SMMU_PIDR3, Peripheral ID3	0x00FEC	[7:4]	MAX(<i>p_level</i> , ecorevnum)	REVRAND, minor revision. Where <i>p_level</i> is 0 for p0.
		[3:0]	0x00	CMOD
SMMU_PIDR2, Peripheral ID2	0x00FE8	[7:4]	0x01	REVISION, major revision
		[3]	1	JEDEC-assigned value for DES always used
		[2:0]	3	DES_1: bits [6:4] bits of the JEP106 Designer code
SMMU_PIDR1, Peripheral ID1	0x00FE4	[7:4]	0xB	DES_0: bits [3:0] of the JEP106 Designer code
		[3:0]	0x4	PART_1: bits [11:8] of the Part number
SMMU_PIDR0, Peripheral ID0	0x00FE0	[7:0]	0x88	PART_0: bits [7:0] of the Part number
SMMU_PIDR7, Peripheral ID7	0x00FDC	-	RES0	Reserved
SMMU_PIDR6, Peripheral ID6	0x00FD8			
SMMU_PIDR5, Peripheral ID5	0x00FD4			
SMMU_PIDR4, Peripheral ID4	0x00FD0	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code

4.13 TBU PMU registers

This section describes the *Performance Monitor Unit* (PMU).

The TBU PMU registers follow the register layout that the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) Performance Monitor Extension describes.

4.13.1 Registers

The TBU and TCU support the same PMCG registers.

See [4.6 TCU PMU registers](#) on page 119.

SMMU_PMCG_SMR0 is 24 bits, because the TBU uses 24-bit StreamIDs architecturally, even though a static tie-off sets either 8 bits or 16 bits.

4.13.2 Events

Each event indicates whether the SMMU_PMCGR_SMR0 register can filter it. For events that cannot be filtered, whether they are visible only when Secure events are visible by SMMU_PMCGR_SCR.SO = 1.

For more information, see [3.3.2 Performance Monitoring Unit](#) on page 58.

The following table shows the architectural events that are implemented.

Table 4-55: Architectural events

Event ID	Description	Filterable	Secure only	Description
0	Clock cycle	No	No	Counts every clock cycle. Does not count cycles where the clock is gated after a clock Q-Channel handshake.
1	Transaction	Yes	-	Counts once per transaction issued downstream into the system
2	TLB miss that an incoming transaction or translation request causes	Yes	-	Counts once per non-speculative TCU translation request
7	PCIe ATS Translated Transaction passed through SMMU	Yes	-	Counts once per ATS transaction that is issued into the system

The following table shows the **IMPLEMENTATION DEFINED** events that are implemented.

Table 4-56: IMPLEMENTATION DEFINED events

Event ID	Description	Filterable	Secure only	Description
0x80	Main TLB lookup	Yes	-	Counts once per transaction that accesses the Main TLB
0x81	Main TLB miss	Yes	-	Counts once per transaction that accesses the Main TLB and does not hit
0x82	Main TLB read	Yes	-	Counts once per access to the Main TLB RAMs. A transaction might access the Main TLB multiple times to look for different page sizes.
0x83	MicroTLB lookup	Yes	-	Counts once per lookup in the MicroTLB
0x84	MicroTLB miss	Yes	-	Counts once per miss in the MicroTLB
0x85	Slots full	No	Yes	Counts once per cycle when all slots are occupied and not ready to issue downstream. This applies across all LTI ports, so if capacity limits make it impossible to completely fill the TLBU, this event cannot occur.
0x86	Out of translation tokens	No	Yes	Counts once per cycle when a translation request cannot be issued because all translation tokens are in use. This applies across all LTI ports, so if capacity limits make it impossible to completely fill the TLBU, this event cannot occur.
0x87	Write data buffer full	No	Yes	Counts once per cycle when a transaction is blocked because the write data buffer is full

Event ID	Description	Filterable	Secure only	Description
0x8B	DCMO downgrade	Yes	-	Counts once whenever either: <ul style="list-style-type: none"> A MakeInvalid transaction on TBS is output as CleanInvalid on TBM A ReadOnceMakeInvalid transaction on TBS is output as ReadOnceCleanInvalid on TBM
0x8C	DCP fail	Yes	-	Counts once whenever either: <ul style="list-style-type: none"> An LTI WDCP transaction on the LA channel is downgraded as W on the LR channel. An LTI DCP transaction on the LA channel is responded to as FaultRAZWI on the LR channel is counted. This response can be because of memory attributes or DCP, R, W, X permission check failure in the TLBU or the DTI fault response with Non-Absort. The transaction responded with FaultAbort because of DTI StreamDisable, GlobalDisable is not counted.
0xD0-0xD7	LTI port slots full	No	Yes	LTI port event (0xD0 + N) corresponds to LTI port N. Counts once per cycle when the slots that are allocated to the LTI port are all occupied and not ready to issue downstream.
0xE0-0xEF	LTI port out of translation tokens	No	Yes	LTI port event (0xD0 + N) corresponds to LTI port N. Counts once per cycle when a translation request cannot be issued for an LTI port because all its allocated translation tokens are in use.

4.13.3 SMMU_PMCG_CFGR

An MMU-700 implementation assumes fixed values for SMMU_PMCG_CFGR, and these values define behavioral aspects of the implementation.

For information about the SMMU_PMCG_CFGR fields values, see [3.3.2.4 SMMUv3 PMU register architectural options](#) on page 63.

See also [3.5 Configuration parameters and methodology](#) on page 94.

4.13.4 SMMU_PMCG_CEID{0-1} registers

The SMMU_PMCG_CEID{0-1} registers indicate the architectural events that are supported. They are described as 64-bit registers, but they are accessed 32 bits at a time through the 32-bit DTI register access messages.

The following table shows the SMMU_PMCG_CEID{0-1} registers.

Table 4-57: SMMU_PMCG_CEID{0-1} registers

Name	Offset	Value
SMMU_PMCG_CEID0	0x02e20	0x00000087
SMMU_PMCG_CEID1	0x02e28	0x00000000

4.13.5 PMU ID registers

The PMU ID registers are defined as follows. The PMU ID registers appear only in Performance Monitor Page 0. Page 1 does not contain any ID registers.

The following table shows the PMU ID registers.

Table 4-58: PMU ID registers

Name	Offset	Field	Value	Description
SMMU_PMCGR_CIDR3, Component ID3	0x02FFC	[7:0]	0xB1	Preamble
SMMU_PMCGR_CIDR2, Component ID2	0x02FF8	[7:0]	0x05	Preamble
SMMU_PMCGR_CIDR1, Component ID1	0x02FF4	[7:0]	0x90	Preamble
SMMU_PMCGR_CIDR0, Component ID0	0x02FF0	[7:0]	0x0D	Preamble
SMMU_PMCGR_PIDR3, Peripheral ID3	0x02FEC	[7:4]	MAX(<i>p_level</i> , ecorevnum)	REVAND, minor revision, where <i>p_level</i> is 0 for p0.
		[3:0]	0x00	CMOD
SMMU_PMCGR_PIDR2, Peripheral ID2	0x02FE8	[7:4]	0x01	REVISION, major revision
		[3]	1	JEDEC-assigned value for DES always used
		[2:0]	3	DES_1: bits [6:4] bits of the JEP106 Designer code
SMMU_PMCGR_PIDR1, Peripheral ID1	0x02FE4	[7:4]	0xB	DES_0: bits [3:0] of the JEP106 Designer code
		[3:0]	0x4	PART_1: bits [11:8] of the Part number
SMMU_PMCGR_PIDR0, Peripheral ID0	0x02FE0	[7:0]	0x88	PART_0: bits [7:0] of the Part number
SMMU_PMCGR_PIDR7, Peripheral ID7	0x02FDC	-	RES0	Reserved
SMMU_PMCGR_PIDR6, Peripheral ID6	0x02FDB			Reserved
SMMU_PMCGR_PIDR5, Peripheral ID5	0x02FDA			Reserved
SMMU_PMCGR_PIDR4, Peripheral ID4	0x02FD0	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code
SMMU_PMCGR_PMAUTHSTATUS	0x02FB8	[7:0]	0x00	No authentication interface is implemented

The PMAUTHSTATUS, PMDEVARCH, and PMDEVTYPE registers are implemented as the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#) defines.

4.14 TBU microarchitectural registers

You can set the microarchitectural registers at boot time to optimize TBU behavior for your system. We recommend that you use the default values for most systems.

The [4.14.3 TBU_SCR register](#) on page 171 is Secure-only. Non-secure access to this register is Read-As-Zero (RAZ)/Writes-Ignored (WI).

Non-secure access to the [4.14.1 TBU_CTRL register](#) on page 166 when TBU_SCR.NS_UARCH = 0 is RAZ/WI.

4.14.1 TBU_CTRL register

The TBU_CTRL register disables TBU features. Do not modify the bits in this register unless Arm® instructs you to do so.

Configurations

This register is available in all configurations.

Attributes

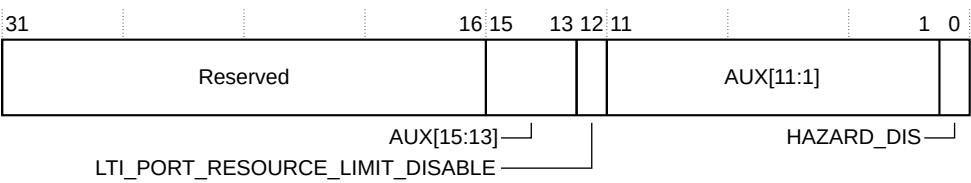
The TBU_CTRL register attributes are as follows:

Width	32-bit
Functional group	TBU microarchitectural features. See 4.14 TBU microarchitectural registers on page 166.
Address	0x08E00
offset	
Type	RW
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-33: TBU_CTRL register bit assignments



The following table shows the bit descriptions.

Table 4-59: TBU_CTRL register bit descriptions

Bits	Name	Description
[31:16]	-	Reserved. If writing other bits of this register, ensure that you write the current value for these bits to them, for example, by performing a read-modify-write sequence.
[15:13]	[AUX15:13]	Reserved. If writing other bits of this register, ensure that you write the current value for these bits to them, for example, by performing a read-modify-write sequence.

Bits	Name	Description
[12]	LTI_PORT_RESOURCE_LIMIT_DISABLE	<p>This bit is used only when there is more than one LTI port configured. An ACE-Lite TBU only has one LTI port internally.</p> <p>0</p> <p>When 0, the 4.14.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 168 is used to control the usage of translation slots and DTI credits by LTI ports.</p> <p>1</p> <p>When 1, all LTI ports are permitted to use the maximum resource.</p> <p>For deadlock and starvation avoidance reasons, a few slots and DTI credits are reserved for each LTI port regardless of the value of this register field and the 4.14.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 168, so if multiple LTI ports are present, it is not possible for any individual port to use all the slots or DTI credits.</p>
[11:1]	[AUX11:1]	Reserved. If writing other bits of this register, ensure that you write the current value for these bits to them, for example, by performing a read-modify-write sequence.
[0]	HAZARD_DIS	<p>0</p> <p>When this bit is clear, and multiple outstanding transactions access the same page, the TBU sends a single translation request and uses that for all the affected transactions.</p> <p>1</p> <p>When this bit is set, disables hazarding between translation requests from transactions in the same page. Post reset, this bit can be set to 1 once, but cannot be cleared again without a reset.</p>

4.14.2 TBU_LTI_PORT_RESOURCE_LIMIT register

The TBU_LTI_PORT_RESOURCE_LIMIT register limits the resource usage for each LTI port.

Configurations

This register is available in all configurations.

Attributes

The TBU_CTRL register attributes are as follows:

Width	32-bit
Functional group	TBU microarchitectural features. See 4.14 TBU microarchitectural registers on page 166.
Address	0x08E04
offset	
Type	RW
Reset value	4'hF

Bit descriptions

The following figure shows the bit assignments.

Figure 4-34: TBU_LTI_PORT_RESOURCE_LIMIT register bit assignments

Non-secure access to TBU_LTI_PORT_RESOURCE_LIMIT when TBU_SCR.NS_UARCH = 0 is RAZ/WI. See [4.14.3 TBU_SCR register](#) on page 171. Each 4-bit field of this register indicates the resource usage limit fraction for the LTI port number that is indicated.

If the current usage is greater than or equal to the specified fraction of total resource, an LTI port is not permitted to use extra resource.

This allocation affects the resources as follows:

- Translation slots, the total number of which the *TBUCFG_XLATE_SLOTS* parameter provides, that transactions from that LTI port can occupy. See [3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters](#) on page 98.
- DTI translation tokens that the TCU returns in the CONDISE_ACK message. A transaction is considered to use a DTI translation credit from the point that it is accepted into a translation slot until it is known that it no longer requires a DTI request, other than a retry. This includes no longer requiring a translation because it has already performed one.



This is different from occupancy of a translation slot because a translation that has received a cache hit, or otherwise determines that it will not require a DTI translation, is not counted in this fraction. Similarly, a transaction that has received a DTI response, but has not returned the LR is no longer considered to use a DTI credit.

The usage of the port of each of the controlled resources is tracked separately because their usage varies separately, but the limit applies equally to all of them.

The register value expresses the number of sixteenths of the available resource that can be used. Therefore, the values encode to the fractions that the following table shows.

Table 4-60: Value encodings

Register value	Fraction
4'b0000	0
4'b0001	1/16
4'b0010	1/8
4'b0011	3/16
4'b0100	1/4
4'b0101	5/16
4'b0110	3/8
4'b0111	7/16
4'b1000	1/2
4'b1001	9/16
4'b1010	5/8
4'b1011	11/16
4'b1100	3/4

Register value	Fraction
4'b1101	13/16
4'b1110	7/8
4'b1111	15/16

The greater-than-or-equal-to constraint permits a fractional credit to be used when the register fraction value multiplied by the total resource available is not a whole number.

If the sum of the allocated resources is more than 1, then a port might not achieve its maximum allocated resources. Resources are allocated on a first case first served basis for that LTI port.

To guarantee freedom from starvation and deadlock, the TLBU must receive at least $(2 \times \text{NUM_LTI_PORTS})$ DTI translation request tokens.

The minimum value of tokens that is required is considered to be part of the usage fraction when in use, but those 2 credits are not available to any other port when not in use. Two credits must be reserved for each port to use. The reserved credits are included in the computation of whether extra resource can be used.

If the minimum allocation, 2, is greater than the fractional allocation, the limit is 2.

Combining the fractional maximum and per-port reservation requirements means that the maximum number of translation slots that transactions can occupy from a given port is:

$$\min(\max(2, (\text{TBUCFG_XLATE_SLOTS} * \text{lti_port_resource_limit})), (\text{TBUCFG_XLATE_SLOTS} - (2 * (\text{NUM_LTI_PORTS} - 1))))$$

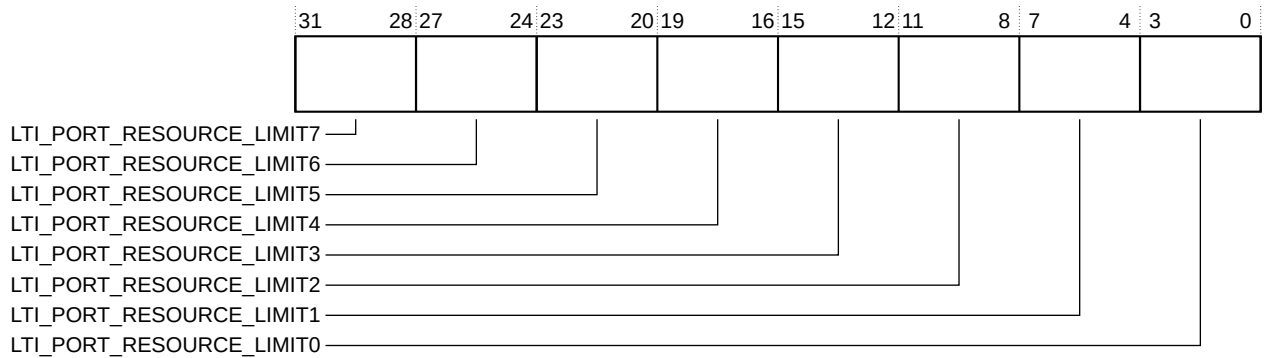


If the TBUCFG.LTI_PORT_RESOURCE_LIMIT_DISABLE register field is set to 1, this register value does not limit the usage per port. Instead, only the availability of resource once other reservations of ports are considered limits usage.

When only one port is present, this register is RAZ/WI but is treated internally as 4'hF. This register has no effect on behavior because its effective value permits the single port to use all the available credits.

The following figure shows the bit assignments.

Figure 4-35: TBU_LTI_PORT_RESOURCE_LIMIT register bit assignments



The following table shows the bit descriptions.

Table 4-61: TBU_LTI_PORT_RESOURCE_LIMIT register bit descriptions

Bits	Name	Description
[31:28]	LTI_PORT_RESOURCE_LIMIT7	Resource limit for LTI Port 7
[27:24]	LTI_PORT_RESOURCE_LIMIT6	Resource limit for LTI Port 6
[23:20]	LTI_PORT_RESOURCE_LIMIT5	Resource limit for LTI Port 5
[19:16]	LTI_PORT_RESOURCE_LIMIT4	Resource limit for LTI Port 4
[15:12]	LTI_PORT_RESOURCE_LIMIT3	Resource limit for LTI Port 3
[11:8]	LTI_PORT_RESOURCE_LIMIT2	Resource limit for LTI Port 2
[7:4]	LTI_PORT_RESOURCE_LIMIT1	Resource limit for LTI Port 1
[3:0]	LTI_PORT_RESOURCE_LIMIT0	Resource limit for LTI Port 0

4.14.3 TBU_SCR register

This section describes the TBU Secure Control register.

Configurations

This register is available in all configurations.

Attributes

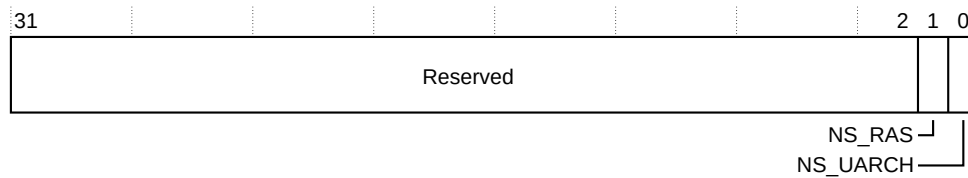
The TBU_SCR register attributes are as follows:

Width	32-bit
Functional group	<i>Reliability, Availability, and Serviceability (RAS)</i> . See 4.8 TCU RAS registers on page 133.
Address offset	0x08EC0
Type	RW
Reset value	sec_override . See A.2.10 TBU tie-off signals on page 218.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-36: TBU_SCR register bit assignments



The following table shows the bit descriptions.

Table 4-62: TBU_SCR register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1]	NS_RAS	<p>Non-secure register access that is permitted for microarchitectural registers.</p> <p>When this bit is 0, Non-secure writes to the following register addresses are ignored, and Non-secure reads return zero:</p> <p>0x08E80-0x08EC0</p> <p>The sec_override input sets the reset value of this signal. See A.2.10 TBU tie-off signals on page 218.</p>
[0]	NS_UARCH	<p>Non-secure register access that is permitted for TBU_CTRL.</p> <p>When this bit is 0, Non-secure writes to TBU_CTRL is ignored, and Non-secure reads return zero.</p> <p>The sec_override input sets reset value of this signal. See A.2.10 TBU tie-off signals on page 218.</p> <p>If Secure translation might be used, we recommend that software does not set this bit.</p>

4.15 TBU RAS registers

This section describes *Reliability, Availability, and Serviceability* (RAS) registers.

These registers implement the RAS Extension registers, single record format.

Non-secure accesses to these registers, when TBU_SCR.NS_RAS = 0, are RAZ/WI.

The RAS registers enable software to monitor the following classes of error:

- *Corrected Errors* (CEs) in the RAMs that the Main TLB uses
- CEs in the RAMs, that the Write Data Buffer uses

RAS error reporting

When a CE occurs:

A CE is reported in [4.15.3 TBU_ERRSTATUS register](#) on page 175.

If TBU_ERRCTL.R.FI is set, an interrupt is raised on **ras_fhi**. See [3.2.2.7 TBU interrupt interfaces](#) on page 43.

4.15.1 TBU_ERRFR register

Error Feature Register.

Configurations

This register is available in all configurations.

Attributes

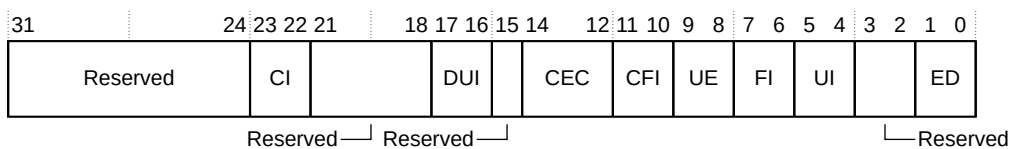
The TBU_ERRFR register attributes are as follows:

Width	32-bit
Functional group	TBU <i>Reliability, Availability, and Serviceability</i> (RAS). See 4.15 TBU RAS registers on page 172.
Address offset	0x08E80
Type	Secure, RO
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-37: TBU_ERRFR register bit assignments



The following table shows the bit descriptions.

Table 4-63: TBU_ERRFR register bit descriptions

Bits	Name	Description	Value
[31:24]	-	Reserved	-
[23:22]	CI	Critical Error Interrupt is always enabled	01b
[21:18]	-	Reserved	-
[17:16]	DUI	Does not support feature	00b
[15]	-	Reserved	-
[14:12]	CEC	Does not implement the standard corrected error counter model	000b

Bits	Name	Description	Value
[11:10]	CFI	Does not support feature	00b
[9:8]	UE	In-band error signaling feature is not enabled	00b
[7:6]	FI	Fault handling interrupt is controllable	10b
[5:4]	UI	Error Recovery Interrupt always enabled for UE	01b
[3:2]	-	Reserved	-
[1:0]	ED	Error detection is always enabled	01b

4.15.2 TBU_ERRCTL register

Use the TBU Error Control register to enable fault handling interrupts.

Configurations

This register is available in all configurations.

Attributes

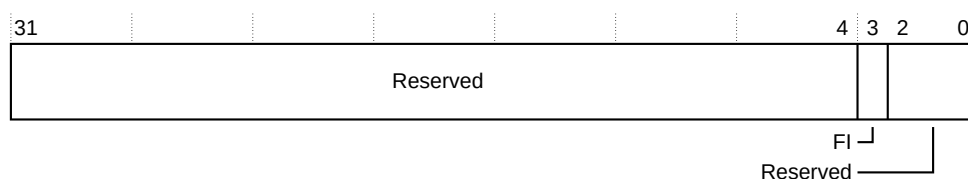
The TBU_ERRCTL register attributes are as follows:

Width	32-bit
Functional group	TBU <i>Reliability, Availability, and Serviceability</i> (RAS). See 4.15 TBU RAS registers on page 172.
Address offset	0x08E88
Type	S, RW
Reset value	1

Bit descriptions

The following figure shows the bit assignments.

Figure 4-38: TBU_ERRCTL register bit assignments



The following table shows the bit descriptions.

Table 4-64: TBU_ERRCTL register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3]	FI	Fault handling interrupt enable
[2:0]	-	Reserved

4.15.3 TBU_ERRSTATUS register

This section describes the TBU_ERRSTATUS register.

Certain bits in this register are cleared by writing a 1 to their bit position. These writes are ignored in certain circumstances to avoid race conditions where a new error has occurred that software has not yet observed.

Configurations

This register is available in all configurations.

Attributes

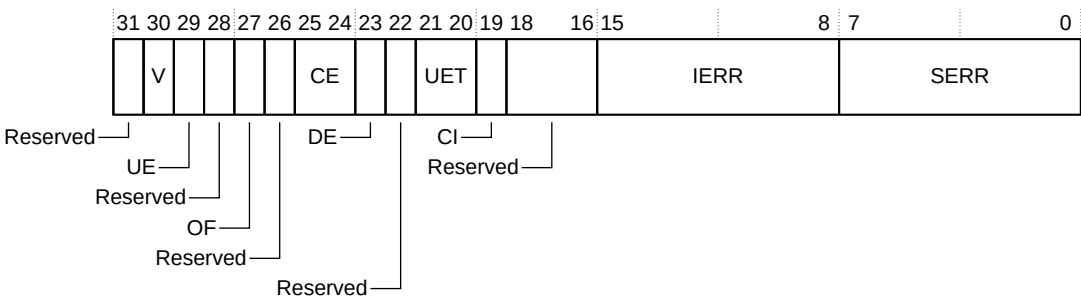
The TBU_ERRSTATUS register attributes are as follows:

Width	32-bit
Functional group	TBU Reliability, Availability, and Serviceability (RAS). See 4.15 TBU RAS registers on page 172.
Address offset	0x08E90
Type	Secure, RW
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-39: TBU_ERRSTATUS register bit assignments



The following table shows the bit descriptions.

Table 4-65: TBU_ERRSTATUS register bit descriptions

Bits	Name	Description
[31]	-	Reserved

Bits	Name	Description
[30]	V	<p>Status Register valid. The possible values of this bit are as follows:</p> <p>0 ERRSTATUS is not valid</p> <p>1 ERRSTATUS is valid, meaning that at least one error has been recorded</p> <p>This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See 4.1.1 Clearing ERRSTATUS registers on page 106.</p> <p>This bit resets to zero on a reset.</p>
[29]	UE	<p>Uncorrected errors. The possible values of this bit are:</p> <p>0 No errors that could not be corrected or deferred</p> <p>1 At least one error is detected that has not been corrected or deferred</p> <p>This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See 4.1.1 Clearing ERRSTATUS registers on page 106.</p>
[28]	-	Reserved
[27]	OF	<p>Overflow. Multiple errors detected. This bit is set to 1 when:</p> <ul style="list-style-type: none"> Any error is received and TBU_ERRSTATUS.V is already set, and not being cleared on the same cycle Multiple errors are received on the same cycle <p>This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See 4.1.1 Clearing ERRSTATUS registers on page 106.</p>
[26]	-	Reserved
[25:24]	CE	<p>Correctable Errors:</p> <p>00b No correctable errors recorded</p> <p>10b At least one corrected error recorded</p> <p>Other values are Reserved.</p> <p>Clearing depends on other ERRSTATUS fields. See 4.1.1 Clearing ERRSTATUS registers on page 106.</p>
[23]	DE	<p>Deferred errors. The possible values of this bit are as follows:</p> <p>0 No errors were deferred</p> <p>1 At least one error was not corrected and deferred</p> <p>This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See 4.1.1 Clearing ERRSTATUS registers on page 106.</p>
[22]	-	Reserved
[21:20]	UET	<p>Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. The possible values of this field are as follows:</p> <p>0b00 Uncorrected error, UnContainable error (UC)</p> <p>Writes to this field are ignored.</p>
[19]	CI	<p>Indicates whether a critical error condition has been recorded. The possible values of this bit are as follows:</p> <p>0 No critical error condition</p> <p>1 Critical error condition</p> <p>Writes to this field are ignored.</p>
[18:16]	-	Reserved

Bits	Name	Description																																												
[15:8]	IERR	<p>IMPLEMENTATION DEFINED error code. This field indicates the source of the error as follows:</p> <table><tr><td>15h</td><td>BIU WDB ROBUFF_P</td></tr><tr><td>14h</td><td>BIU WDB ROBUFF_C</td></tr><tr><td>13h</td><td>BIU WDB ROBUFF_D</td></tr><tr><td>12h</td><td>Reserved</td></tr><tr><td>11h</td><td>Reserved</td></tr><tr><td>10h</td><td>TLBU DCU MTLB DATA</td></tr><tr><td>0Fh</td><td>TLBU DCU MTLB TAGS</td></tr><tr><td>0Eh</td><td>TLBU DCU MTLB REPL</td></tr><tr><td>0Dh</td><td>TLBU DCU MTLB PCNT</td></tr><tr><td>0Ch</td><td>TLBU DCU MTLB PLIM</td></tr><tr><td>0Bh</td><td>TLBU TOU HLB_ENTRY RIGHT</td></tr><tr><td>0Ah</td><td>TLBU TOU HLB_ENTRY LEFT</td></tr><tr><td>09h</td><td>TLBU TOU HLB PTR RIGHT</td></tr><tr><td>08h</td><td>TLBU TOU HLB PTR LEFT</td></tr><tr><td>07h</td><td>Reserved</td></tr><tr><td>06h</td><td>Reserved</td></tr><tr><td>05h</td><td>TLBU TOU DTIQ</td></tr><tr><td>04h</td><td>TLBU TOU UOQ</td></tr><tr><td>03h</td><td>TLBU TOU OGQ</td></tr><tr><td>02h</td><td>TLBU TOU LB</td></tr><tr><td>01h</td><td>TLBU TOU RSP</td></tr><tr><td>00h</td><td>TLBU TOU REQ.</td></tr></table> <p>Writes to this field are ignored.</p>	15h	BIU WDB ROBUFF_P	14h	BIU WDB ROBUFF_C	13h	BIU WDB ROBUFF_D	12h	Reserved	11h	Reserved	10h	TLBU DCU MTLB DATA	0Fh	TLBU DCU MTLB TAGS	0Eh	TLBU DCU MTLB REPL	0Dh	TLBU DCU MTLB PCNT	0Ch	TLBU DCU MTLB PLIM	0Bh	TLBU TOU HLB_ENTRY RIGHT	0Ah	TLBU TOU HLB_ENTRY LEFT	09h	TLBU TOU HLB PTR RIGHT	08h	TLBU TOU HLB PTR LEFT	07h	Reserved	06h	Reserved	05h	TLBU TOU DTIQ	04h	TLBU TOU UOQ	03h	TLBU TOU OGQ	02h	TLBU TOU LB	01h	TLBU TOU RSP	00h	TLBU TOU REQ.
15h	BIU WDB ROBUFF_P																																													
14h	BIU WDB ROBUFF_C																																													
13h	BIU WDB ROBUFF_D																																													
12h	Reserved																																													
11h	Reserved																																													
10h	TLBU DCU MTLB DATA																																													
0Fh	TLBU DCU MTLB TAGS																																													
0Eh	TLBU DCU MTLB REPL																																													
0Dh	TLBU DCU MTLB PCNT																																													
0Ch	TLBU DCU MTLB PLIM																																													
0Bh	TLBU TOU HLB_ENTRY RIGHT																																													
0Ah	TLBU TOU HLB_ENTRY LEFT																																													
09h	TLBU TOU HLB PTR RIGHT																																													
08h	TLBU TOU HLB PTR LEFT																																													
07h	Reserved																																													
06h	Reserved																																													
05h	TLBU TOU DTIQ																																													
04h	TLBU TOU UOQ																																													
03h	TLBU TOU OGQ																																													
02h	TLBU TOU LB																																													
01h	TLBU TOU RSP																																													
00h	TLBU TOU REQ.																																													
[7:0]	SERR	<p>Error code.</p> <p>This provides information about the earliest unacknowledged Error.</p> <p>It can contain the following values:</p> <table><tr><td>0</td><td>No error</td></tr><tr><td>2</td><td>Single or double error from RAMs that are not MTLB TAGS or DATA</td></tr><tr><td>8</td><td>Single or double error from MTLB Data</td></tr><tr><td>9</td><td>Single or double error from MTLB Tags</td></tr></table> <p>All other values are reserved.</p> <p>Writes to this field are ignored.</p>	0	No error	2	Single or double error from RAMs that are not MTLB TAGS or DATA	8	Single or double error from MTLB Data	9	Single or double error from MTLB Tags																																				
0	No error																																													
2	Single or double error from RAMs that are not MTLB TAGS or DATA																																													
8	Single or double error from MTLB Data																																													
9	Single or double error from MTLB Tags																																													

4.15.4 TBU_ERRGEN register

Error Generation Register. Use the TBU_ERRGEN register to test software for when a RAS error occurs.

The field locations are same as for the [4.15.3 TBU_ERRSTATUS register](#) on page 175.

When this register is updated, the [4.15.3 TBU_ERRSTATUS register](#) on page 175 is also updated with the same value, as long as the write data generates a valid error record.

A write to ERRGEN is valid if all the following is true:

- ERRGEN.V is set

- At least one of the following is true (CE is legal if CE == 2'b00 or CE == 2'b10):
 - ERRGEN.UE is set and CE is legal
 - ERRGEN.DE is set and CE is legal
 - ERRGEN.CE is set to 2'b10
- UET must be 2'b00

If a valid error record is written, then the appropriate interrupt, or interrupts, are also generated.

This register has identical fields to [4.15.3 TBU_ERRSTATUS register](#) on page 175.

Configurations

This register is available in all configurations.

Attributes

The TBU_ERRGEN register attributes are as follows:

Width	32-bit
Functional group	TBU <i>Reliability, Availability, and Serviceability</i> (RAS). See 4.15 TBU RAS registers on page 172.
Address	0x08EC0
offset	
Type	S, RW
Reset value	0

Bit descriptions

See the bit descriptions in [4.15.3 TBU_ERRSTATUS register](#) on page 175.

4.16 TBU system discovery registers

This section describes the TBU system discovery registers.

4.16.1 TBU_SYSDISC0 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC0 register attributes are as follows:

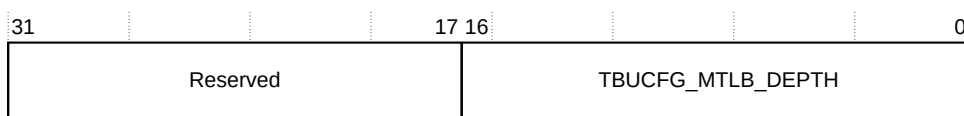
Width	32-bit
--------------	--------

Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E30
Type	RO
Reset value	<i>TBUCFG_MTLB_DEPTH</i> . See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-40: TBU_SYSDISC0 register bit assignments



The following table shows the bit descriptions.

Table 4-66: TBU_SYSDISC0 register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved
[16:0]	TBUCFG_MTLB_DEPTH	<p>The read data reflects the chosen parameter value, for example:</p> <p>17'h0_0008 : 8</p> <p>....</p> <p>17'h1_0000 : 65536</p>

4.16.2 TBU_SYSDISC1 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC1 register attributes are as follows:

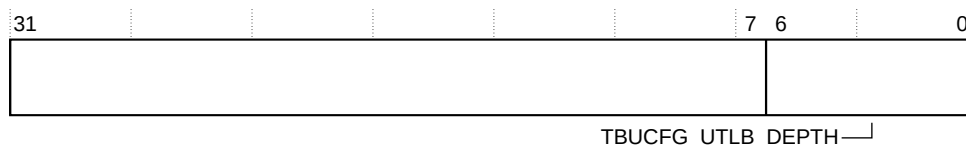
Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E34
Type	RO

Reset value *TBUCFG_UTLB_DEPTH*. See [3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters](#) on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-41: TBU_SYSDISC1 register bit assignments



The following table shows the bit descriptions.

Table 4-67: TBU_SYSDISC1 register bit descriptions

Bits	Name	Description
[31:7]	-	Reserved
[6:0]	TBUCFG_UTLB_DEPTH	The read data reflects the chosen parameter value, for example: 7'h04 : 4 7'h40 : 64

4.16.3 TBU_SYSDISC2 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

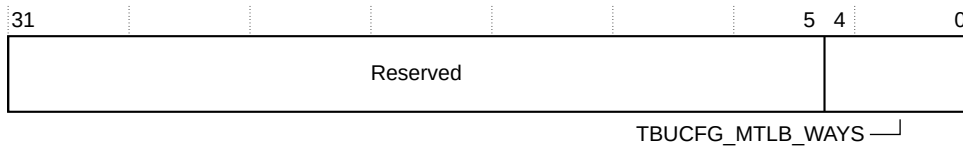
The TBU_SYSDISC2 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E38
Type	RO
Reset value	<i>TBUCFG_MTLB_WAYS</i> . See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-42: TBU_SYSDISC2 register bit assignments



The following table shows the bit descriptions.

Table 4-68: TBU_SYSDISC2 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved
[4:0]	TBUCFG_MTLB_WAYS	The read data reflects the chosen parameter value, for example: 5'h04 : 4 5'h10 : 16

4.16.4 TBU_SYSDISC3 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

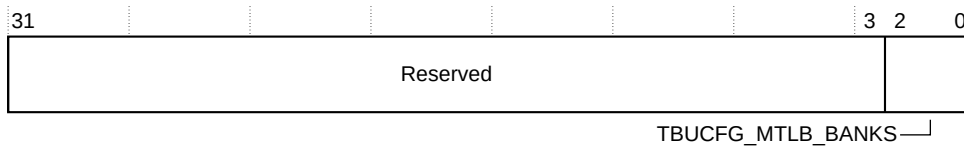
The TBU_SYSDISC3 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address	0x08E3C
offset	
Type	RO
Reset value	<i>TBUCFG_MTLB_BANKS</i> . See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-43: TBU_SYSDISC3 register bit assignments



The following table shows the bit descriptions.

Table 4-69: TBU_SYSDISC3 register bit descriptions

Bits	Name	Description
[31:3]	-	Reserved
[2:0]	TBUCFG_MTLB_BANKS	The read data reflects the chosen parameter value, for example: 3'b001 : 1 3'b100 : 4

4.16.5 TBU_SYSDISC4 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

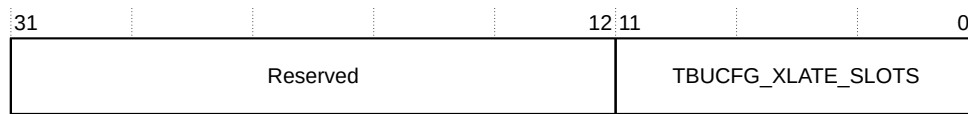
The TBU_SYSDISC4 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E40
Type	RO
Reset value	<i>TBUCFG_XLATE_SLOTS</i> . See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-44: TBU_SYSDISC4 register bit assignments



The following table shows the bit descriptions.

Table 4-70: TBU_SYSDISC4 register bit descriptions

Bits	Name	Description
[31:12]	-	Reserved
[11:0]	TBUCFG_XLATE_SLOTS	<p>The read data reflects the chosen parameter value, for example: 11'h0004 : 4</p> <p>....</p> <p>11'h400 : 1024</p>

4.16.6 TBU_SYSDISC5 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

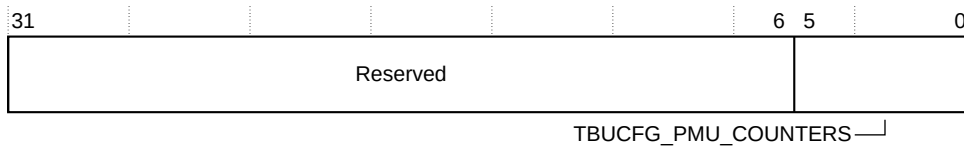
The TBU_SYSDISC5 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E44
Type	RO
Reset value	<i>TBUCFG_PMU_COUNTERS</i> . See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-45: TBU_SYSDISC5 register bit assignments



The following table shows the bit descriptions.

Table 4-71: TBU_SYSDISC5 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved
[5:0]	TBU_CFG_PMU_COUNTERS	The read data reflects the chosen parameter value, for example: 6'h04 : 4 6'h20 : 32

4.16.7 TBU_SYSDISC6 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

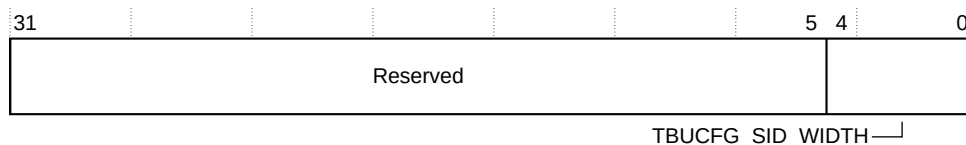
The TBU_SYSDISC6 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E48
Type	RO
Reset value	<i>TBU_CFG_SID_WIDTH</i> . See 3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters on page 97.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-46: TBU_SYSDISC6 register bit assignments



The following table shows the bit descriptions.

Table 4-72: TBU_SYSDISC6 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved
[4:0]	TBUCFG_SID_WIDTH	The read data reflects the chosen parameter value, for example: 5'h08 : 8 5'h18 : 24

4.16.8 TBU_SYSDISC7 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

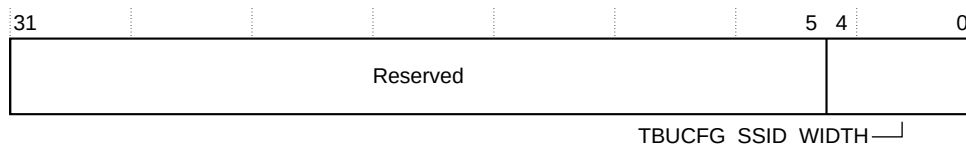
The TBU_SYSDISC7 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address	0x08E4C
offset	
Type	RO
Reset value	<i>TBUCFG_SSID_WIDTH</i> . See 3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters on page 97.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-47: TBU_SYSDISC7 register bit assignments



The following table shows the bit descriptions.

Table 4-73: TBU_SYSDISC7 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved
[4:0]	TBUCFG_SSID_WIDTH	The read data reflects the chosen parameter value, for example: 5'h01 : 1 5'h14 : 20

4.16.9 TBU_SYSDISC8 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

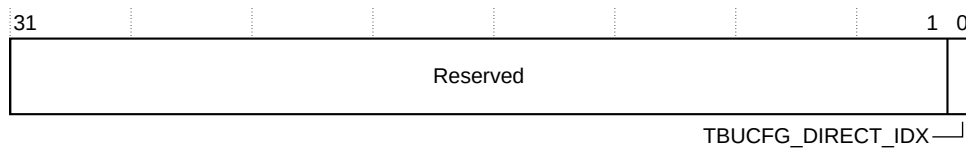
The TBU_SYSDISC8 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E50
Type	RO
Reset value	<i>TBUCFG_DIRECT_IDX</i> . See 3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters on page 97.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-48: TBU_SYSDISC8 register bit assignments



The following table shows the bit descriptions.

Table 4-74: TBU_SYSDISC8 register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	TBUCFG_DIRECT_IDX	The read data reflects the chosen parameter value, for example: 1'b0 : 0 1'b1 : 1

4.16.10 TBU_SYSDISC9 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

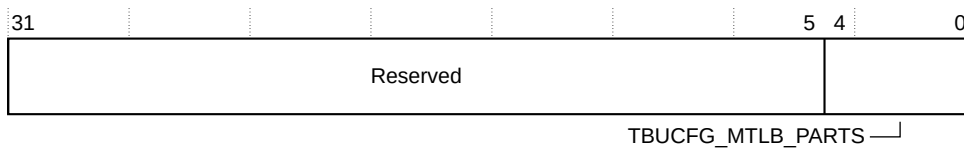
The TBU_SYSDISC9 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E54
Type	RO
Reset value	<i>TBUCFG_MTLB_PARTS</i> . See 3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters on page 97.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-49: TBU_SYSDISC9 register bit assignments



The following table shows the bit descriptions.

Table 4-75: TBU_SYSDISC9 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved
[4:0]	TBUCFG_MTLB_PARTS	The read data reflects the chosen parameter value, for example: 5'h00 : 0 5'h10 : 16

4.16.11 TBU_SYSDISC10 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

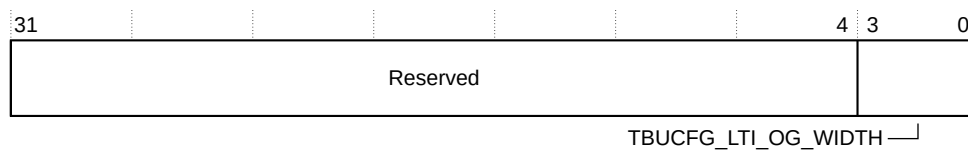
The TBU_SYSDISC10 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E58
Type	RO
Reset value	<i>TBUCFG_LTI_OG_WIDTH</i> . See 3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters on page 97.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-50: TBU_SYSDISC10 register bit assignments



The following table shows the bit descriptions.

Table 4-76: TBU_SYSDISC10 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3:0]	TBU_CFG_LTI_OG_WIDTH	The read data reflects the chosen parameter value, for example: 3'h00 : 0 3'h20 : 05

4.16.12 TBU_SYSDISC11 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

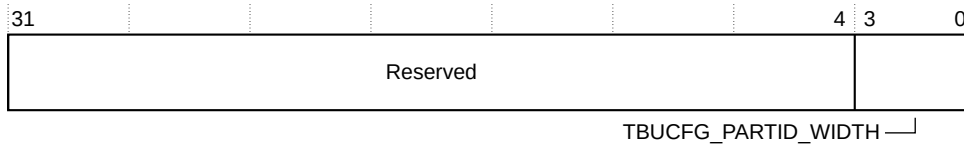
The TBU_SYSDISC11 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E5C
Type	RO
Reset value	TBU_CFG_PARTID_WIDTH. See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-51: TBU_SYSDISC11 register bit assignments



The following table shows the bit descriptions.

Table 4-77: TBU_SYSDISC11 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3:0]	TBUCFG_PARTID_WIDTH	The read data reflects the chosen parameter value, for example: 4'b0001 : 1 4'b1001 : 9

4.16.13 TBU_SYSDISC12 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

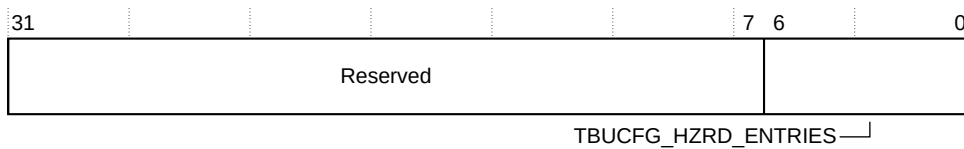
The TBU_SYSDISC12 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E60
Type	RO
Reset value	<i>TBUCFG_HZRD_ENTRIES</i> . See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-52: TBU_SYSDISC12 register bit assignments



The following table shows the bit descriptions.

Table 4-78: TBU_SYSDISC12 register bit descriptions

Bits	Name	Description
[31:7]	-	Reserved
[6:0]	TBU_CFG_HZRD_ENTRIES	The read data reflects the chosen parameter value, for example: 7'h00 : 0 7'h40 : 64

4.16.14 TBU_SYSDISC13 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

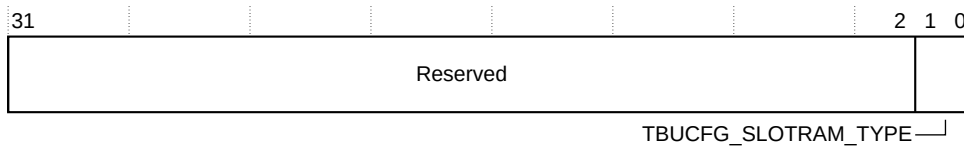
The TBU_SYSDISC13 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E64
Type	RO
Reset value	<i>TBU_CFG_SLOTRAM_TYPE</i> . See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-53: TBU_SYSDISC13 register bit assignments



The following table shows the bit descriptions.

Table 4-79: TBU_SYSDISC13 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1:0]	TBU_CFG_SLOTRAM_TYPE	The read data reflects the chosen parameter value, for example: 2'b00 : 0 2'b01 : 1

4.16.15 TBU_SYSDISC14 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

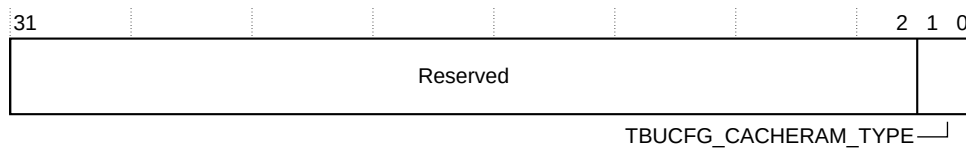
The TBU_SYSDISC14 register attributes are as follows:

Width	32-bit
Functional group	TBU system discovery registers. See 4.16 TBU system discovery registers on page 178.
Address offset	0x08E68
Type	RO
Reset value	TBU_CFG_CACHERAM_TYPE. See 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98.

Bit descriptions

The following figure shows the bit assignments.

Figure 4-54: TBU_SYSDISC14 register bit assignments



The following table shows the bit descriptions.

Table 4-80: TBU_SYSDISC14 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1:0]	TBUCFG_CACHERAM_TYPE	The read data reflects the chosen parameter value, for example: 2'b00 : 0 2'b01 : 1

4.17 TBU integration registers

This section describes the TBU integration registers.

4.17.1 ITEN register for the TBU

Integration mode register for the TBU. When integration mode is enabled, the values of certain TBU input pins are made visible in the ITIN register of the TBU.

The values that are written to the ITOP register of the TBU control the values of certain TBU output pins to help system integrators to integrate the SMMU into the system and perform basic connectivity checks.

Configurations

This register is available in all configurations.

Attributes

The ITEN register attributes are as follows:

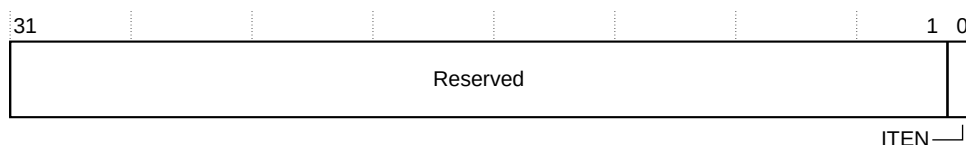
Width	32-bit
Functional group	TBU integration registers. See 4.4.11 TBU integration registers summary on page 118.
Address offset	0x08E20
Type	RW

Reset value 0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-55: ITEN register bit assignments



The following table shows the bit descriptions.

Table 4-81: ITEN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	ITEN	0 Integration mode is disabled 1 Integration mode is enabled

4.17.2 ITOP_TBU register

This section describes the ITOP register for the TBU.

Configurations

This register is available in all configurations.

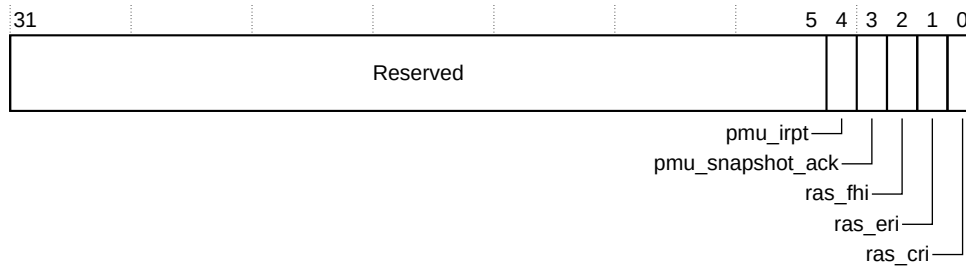
Attributes

The ITOP_TBU register attributes are as follows:

Width	32-bit
Functional group	TBU integration registers. See 4.4.11 TBU integration registers summary on page 118.
Address offset	0x08E24
Type	RW
Reset value	0

Bit descriptions

Figure 4-56: ITOP_TBU register bit assignments



The following table shows the bit descriptions.

Table 4-82: ITOP_TBU register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, SBZ
[4]	pmu_irpt	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to pmu_irpt
[3]	pmu_snapshot_ack	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to pmusnapshot_ack
[2]	ras_fhi	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to ras_fhi
[1]	ras_eri	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to ras_eri
[0]	ras_cri	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, the value driven to ras_cri

See:

- [A.2.9 TBU interrupt signals](#) on page 217
- [A.1.5 TCU PMU snapshot interface signals](#) on page 200

4.17.3 ITIN_TBU register

This section describes the ITIN register for the TBU.

Configurations

This register is available in all configurations.

Attributes

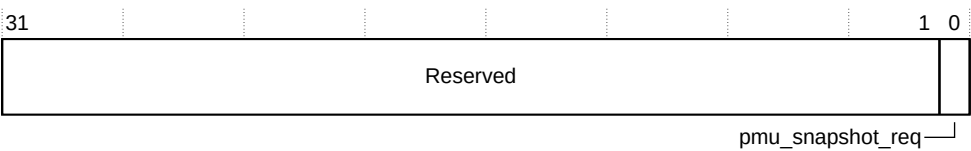
The ITIN_TBU register attributes are as follows:

Width	32-bit
Functional group	TBU integration registers. See 4.4.11 TBU integration registers summary on page 118.
Address	0x08E28
offset	
Type	RO
Reset value	0

Bit descriptions

The following figure shows the bit assignments.

Figure 4-57: ITIN_TBU register bit assignments



The following table shows the bit descriptions.

Table 4-83: ITIN_TBU register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved, SBZ
[0]	pmu_snapshot_req	0 When ITEN.ITEN == 0, SBZ 1 When ITEN.ITEN == 1, reflects pmusnapshot_req

See [A.1.5 TCU PMU snapshot interface signals](#) on page 200.

Appendix A Signal descriptions

This appendix describes the MMU-700 external signals.

A.1 TCU signals

This section describes the MMU-700 TCU signals.

A.1.1 TCU clock and reset signals

The TCU uses a single set of standard clock and reset signals.

The following table shows the clock and reset signals.

Table A-1: Clock and reset signals

Signal	Width, in bits	Direction	Description
clk	1	Input	Global clock
resetsn	1	Input	Global reset

A.1.2 TCU QTW/DVM interface signals

The TCU QTW/DVM interface signals are based on the AMBA ACE5-Lite signals.

The following table shows the TCU QTW/DVM interface signals.

Table A-2: TCU QTW/DVM interface signals

Signal	Width, in bits	Direction	Description
acaddr_qtw	52	Input	Snoop address
acprot_qtw	3	Input	Snoop protection type
acready_qtw	1	Output	Snoop address ready
acsnoop_qtw	4	Input	Snoop transaction type
acvalid_qtw	1	Input	Snoop address valid
arid_qtw	$Q_{TW_ID_WIDTH}$. See ¹¹	Output	Read address ID
araddr_qtw	52	Output	Read address
arburst_qtw	2	Output	Burst type
arcache_qtw	4	Output	Memory type

¹¹ $Q_{TW_ID_WIDTH}$ is calculated as follows:

$((\log_2(TCUCFG_PTW_SLOTS) + 2) > 4) ? (\log_2(TCUCFG_PTW_SLOTS) + 2) : 4$; See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94.

Signal	Width, in bits	Direction	Description
ardomain_qtw	2	Output	Shareability domain
arlen_qtw	8	Output	Burst length
arlock_qtw	1	Output	Lock type
arprot_qtw	3	Output	Protection type
arqos_qtw	4	Output	QoS identifier
arready_qtw	1	Input	Read address ready
arsize_qtw	3	Output	Burst size
arsnoop_qtw	4	Output	Transaction type
aruser_qtw	4	Output	Hardware attribute information
arvalid_qtw	1	Output	Read address valid
awid_qtw	<i>QTW_ID_WIDTH</i> . See ¹²	Output	Write address ID
awaddr_qtw	52	Output	Write address
awburst_qtw	2	Output	Burst type
awcache_qtw	4	Output	Memory type
awdomain_qtw	2	Output	Shareability domain
awlen_qtw	8	Output	Burst length
awlock_qtw	1	Output	Lock type
awprot_qtw	3	Output	Protection type
awqos_qtw	4	Output	QoS identifier
awready_qtw	1	Input	Write address ready
awsize_qtw	3	Output	Burst size
awsnoop_qtw	4	Output	Transaction type
awuser_qtw	4	Output	Hardware attribute information
awvalid_qtw	1	Output	Write address valid
crready_qtw	1	Input	Snoop response ready
crresp_qtw	5	Output	Snoop response
crvalid_qtw	1	Output	Snoop response valid
rid_qtw	<i>QTW_ID_WIDTH</i> . See ¹³	Input	Read data ID
rdata_qtw	<i>TCUCFG_QTW_DATA_WIDTH</i> . See ¹⁴	Input	Read data
rlast_qtw	1	Input	Read last
rready_qtw	1	Output	Read ready

¹² *QTW_ID_WIDTH* is calculated as follows:

$((\log_2(TCUCFG_PTW_SLOTS) + 2) > 4) ? (\log_2(TCUCFG_PTW_SLOTS) + 2) : 4$; See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94.

¹³ *QTW_ID_WIDTH* is calculated as follows:

$((\log_2(TCUCFG_PTW_SLOTS) + 2) > 4) ? (\log_2(TCUCFG_PTW_SLOTS) + 2) : 4$; See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94.

¹⁴ *TCUCFG_QTW_DATA_WIDTH*-bit. See [3.5.1 Translation Control Unit I/O configuration parameters](#) on page 94.

Signal	Width, in bits	Direction	Description
rresp_qtw	2	Input	Read response
rvalid_qtw	1	Input	Read valid
wdata_qtw	<i>TCUCFG_QTW_DATA_WIDTH</i> . See ¹⁵	Output	Write data
wlast_qtw	1	Output	Write last
wready_qtw	1	Input	Write ready
wstrb_qtw	<i>TCUCFG_QTW_DATA_WIDTH</i> /8. See ¹⁶	Output	Write strobe
wvalid_qtw	1	Output	Write valid
bid_qtw	<i>QTW_ID_WIDTH</i> . See ¹⁷	Input	Response ID
bready_qtw	1	Output	Response ready
bresp_qtw	2	Input	Write response
bvalid_qtw	1	Input	Write response valid
awakeup_qtw	1	Output	Wakeup
acwakeup_qtw	1	Input	Snoop wakeup
acvmidext_qtw	4	Input	Snoop Extended <i>Virtual Machine Identifier</i> (VMID)

For more information about these signals, see the [AMBA® AXI and ACE Protocol Specification](#).

A.1.3 TCU programming interface signals

The TCU programming interface signals are based on the AMBA APB4 signals.

The following table shows the TCU programming interface signals.

Table A-3: TCU programming interface signals

Signal	Width, in bits	Direction	Description
paddr_prog	21 or 23. See ¹⁸	Input	Peripheral address
psel_prog	1	Input	Peripheral select
penable_prog	1	Input	Enable for transfer
pwrite_prog	1	Input	Write transaction indicator
pprot_prog	3	Input	Protection type
pwwdata_prog	32	Input	Write data
pstrb_prog	4	Input	Write data strobe
pslverr_prog	1	Output	Error response

¹⁵ *TCUCFG_QTW_DATA_WIDTH*-bit. See [3.5.1 Translation Control Unit I/O configuration parameters](#) on page 94.

¹⁶ (*TCUCFG_QTW_DATA_WIDTH*/8)-bit. See [3.5.1 Translation Control Unit I/O configuration parameters](#) on page 94.

¹⁷ *QTW_ID_WIDTH* is calculated as follows:

$((\log_2(\text{TCUCFG_PTW_SLOTS}) + 2) > 4) ? (\log_2(\text{TCUCFG_PTW_SLOTS}) + 2) : 4$; See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94.

¹⁸ If *TCUCFG_NUM_TBU* is 62, the width of **paddr_prog** is 23-bit. Otherwise, the width of **paddr_prog** is 21-bit. See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94.

Signal	Width, in bits	Direction	Description
prdata_prog	32	Output	Read data
pready_prog	1	Output	Transfer ready
pwakeup_prog	1	Input	Interface wakeup

For more information about these signals, see the [AMBA® APB Protocol Specification](#).

A.1.4 TCU SYSCO interface signals

The following table shows the TCU SYSCO interface signals.

Table A-4: TCU SYSCO interface signals

Signal	Width, in bits	Direction	Description
syscoreq_qtw	1	Output	System coherency request. This output transitions: HIGH To indicate that the master is requesting to enter the coherency domain. LOW To indicate that the master is requesting to exit the coherency domain.
syscoack_qtw	1	Input	System coherency acknowledge. This input transitions to the same level as syscoreq_qtw when the request to enter or exit the coherency domain is complete.

For more information about these signals, see the [AMBA® AXI and ACE Protocol Specification](#).

A.1.5 TCU PMU snapshot interface signals

The following table shows the TCU PMU snapshot interface signals.

Table A-5: TCU PMU snapshot interface signals

Signal	Width, in bits	Direction	Description
pmusnapshot_req	1	Input	PMU snapshot request. The PMU snapshot occurs on the rising edge of pmusnapshot_req . Note: Connect to the debug infrastructure of your SoC.
pmusnapshot_ack	1	Output	PMU snapshot acknowledge. The TCU uses this signal to acknowledge that the PMU snapshot has occurred. This signal is LOW after reset. Note: Connect to the debug infrastructure of your SoC.

A.1.6 TCU LPI_PD interface signals

The following table shows the TCU LPI_PD interface signals.

Table A-6: TCU LPI_PD interface signals

Signal	Width, in bits	Direction	Description
qactive_pd	1	Output	Component active
qreqn_pd	1	Input	Quiescence request
qacceptn_pd	1	Output	Quiescence accept
qdeny_pd	1	Output	Quiescence deny

For more information about these signals, see the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

A.1.7 TCU LPI_CG interface signals

The following table shows the TCU LPI_CG interface signals.

Table A-7: TCU LPI_CG interface signals

Signal	Width, in bits	Direction	Description
qactive_cg	1	Output	Component active
qreqn_cg	1	Input	Quiescence request
qacceptn_cg	1	Output	Quiescence accept
qdeny_cg	1	Output	Quiescence deny

For more information about these signals, see the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

A.1.8 TCU DTI interface signals

The following table shows the TCU DTI interface signals.

Table A-8: TCU DTI interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_dn	1	Master to slave	Flow control signal
tready_dti_dn	1	Slave to master	Flow control signal
tdata_dti_dn	20	Master to slave	Message data signal
tid_dti_dn	4 or 6. See ¹⁹	Master to slave	Identifies the master that initiated the message
tlast_dti_dn	1	Master to slave	Indicates the last cycle of a message
tkeep_dti_dn	20	Master to slave	This signal indicates valid bytes

¹⁹ If *TCUCFG_NUM_TBU* is 62, the width of **tid_dti_dn** is 6-bit. Otherwise, the width of **tid_dti_dn** is 4-bit. See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94.

Signal	Width, in bits	Direction	Description
tvalid_dti_up	1	Slave to master	Flow control signal
tready_dti_up	1	Master to slave	Flow control signal
tdata_dti_up	160	Slave to master	Message data signal
tdest_dti_up	4 or 6. See ²⁰	Slave to master	Identifies the master that is receiving the message
tlast_dti_up	1	Slave to master	Indicates the last cycle of a message
tkeep_dti_up	20	Slave to master	Indicates valid bytes
twakeup_dti_up	1	Slave to master	Wakeup signal
twakeup_dti_dn	1	Master to slave	Wakeup signal

For more information about the DTI signals, see the [AMBA® 4 AXI4-Stream Protocol Specification](#).

For more information about DTI protocol messages, see the [AMBA® DTI Protocol Specification](#).

A.1.9 TCU interrupt signals

The TCU interrupt signals are edge-triggered. The interrupt controller must detect the rising edge of these signals.

The TCU can also output the Secure and Non-secure Event queue, SYNC complete commands, and global interrupts as *Message Signaled Interrupts* (MSIs) on the QTW/DVM interface. If the system supports capturing MSIs from the TCU, there is no requirement to connect the corresponding interrupt signals in this interface.

The following table shows the TCU interrupt signals.

Table A-9: TCU interrupt interface signals

Signal	Width, in bits	Direction	Description
event_q_irpt_s	1	Output	Event queue, Secure interrupt. Asserts a Secure interrupt to indicate that the Event queue is not empty or has overflowed.
event_q_irpt_ns	1	Output	Event queue, Non-secure interrupt. Asserts a Non-secure interrupt to indicate that the Event queue is not empty or has overflowed.
cmd_sync_irpt_ns	1	Output	SYNC complete, Non-secure interrupt. Asserts a Non-secure interrupt to indicate that the CMD_SYNC command is complete.
cmd_sync_irpt_s	1	Output	SYNC complete, Secure interrupt. Asserts a Secure interrupt to indicate that the CMD_SYNC command is complete.
global_irpt_ns	1	Output	Asserts a global Non-secure interrupt
global_irpt_s	1	Output	Asserts a global Secure interrupt
ras_fhi	1	Output	Fault handling RAS interrupt for a contained error
ras_eri	1	Output	Error recovery RAS interrupt for an uncontained error
ras_cri	1	Output	Critical error interrupt, for an uncontrollable uncorrected error

²⁰ If *TCUCFG_NUM_TBU* is 62, the width of **tdest_dti_up** is 6-bit. Otherwise, the width of **tdest_dti_up** is 4-bit. See [3.5.2 Translation Control Unit buffer configuration parameters](#) on page 94.

Signal	Width, in bits	Direction	Description
pmu_irpt	1	Output	Asserts a PMU interrupt. Note: The MMU-700 cannot output PMU interrupts as MSIs. You must connect this output to an interrupt controller.
pri_q_irpt_ns	1	Output	Asserts a <i>Page Request Interface</i> (PRI) queue interrupt

A.1.10 TCU MSI interface signals

This section describes the TCU *Message Signaled Interrupt* (MSI) interface.

The interface follows the AXI4-Stream protocol and uses the signals in the following table to send MSIs.

The following table shows the TCU MSI interface signals.

Table A-10: TCU MSI interface signals

Signal	AXI4-Stream signal	Width, in bits	Direction	Description
msitvalid	TVALID	1	Output	Indicates valid data to the GIC
msitready	TREADY	1	Input	Indicates acceptance by the GIC
msitdata	TDATA	64	Output	Data being passed to the GIC
msitwakeup	TWAKEUP, AMBA extension	1	Output	Indicates that a transaction is ongoing
msirtvalid	TVALID	1	Input	Indicates that the GIC has accepted an MSI
msirtready	TREADY	1	Output	Indicates that the device has accepted the response packet
msirtwakeup	TWAKEUP, AMBA extension	1	Input	Indicates that a transaction is ongoing

For more information about these signals, see the *GIC MSI Delivery Interface* document.

A.1.11 TCU event interface signal

The TCU event interface signal is an event output for connection to processors.

The following table shows the TCU event interface signal.

Table A-11: TCU event interface signal

Signal	Width, in bits	Direction	Description
evento	1	Output	<p>The evento signal is asserted for one cycle to indicate an event that enables processors to wake up from the <i>Wait For Event</i> (WFE) low-power state.</p> <p>Connect the evento signal of the TCU to the event interface of Arm® processors. Processors that use the <i>DynamiQ Shared Unit</i> (DSU) have a different event handshake mechanism.</p> <p>The mechanism that the DSU uses is the successor to the mechanism that some MMUs use.</p> <p>Arm® processors can use the following event mechanisms:</p> <ul style="list-style-type: none"> Some processors have an eventi input to connect directly to the evento output from the MMU Some processors, including DSU-based systems, have a req/ack handshake mechanism that requires the evento signal from the MMU to be converted and uses the eventiack, eventireq, eventoack, and eventoreq signals <p>Note: You can also route the evento signal through other interconnects such as the Arm® CoreLink™ CMN-600 Coherent Mesh Network instead of connecting evento directly to the processor. These interconnects, like the DSU, support only the newer event mechanism. If the rest of your system uses the newer event mechanism, you must add logic to convert events that the MMU-700 generates, which uses the older event mechanism.</p> <p>In both mechanisms, in the signal names:</p> <p>i Represents events that are inputs to a particular component o Represents events that are outputs from a particular component</p> <p>Note: For the signals, the handshake mechanism uses one input and one output in each direction. This is because the acknowledgment of the request operates in the opposite direction to the original request. The MMU-700 has an event output and therefore only has the evento signal. The processor has an input interface to receive the event from the MMU-700, and other devices. This input interface uses the eventiack and eventireq signals, if the processor uses the newer mechanism.</p> <p>The required conversion is from the older mechanism, eventi and evento signals, to the newer mechanism, eventiack, eventireq, eventoack, and eventoreq signals.</p> <p>When connecting the MMU-700 to a DSU, the only signals to consider are the following:</p> <ul style="list-style-type: none"> evento signal of the MMU-700 eventiack and eventireq signals of the DSU <p>Some processors have an eventi input instead.</p> <p>You can use the <i>Event Pulse to Event adapter</i> that is provided in the CoreSight™ System-on-Chip SoC-600. For more information about this component, see Section 6.5 in the Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual.</p> <p>Note: To use the <i>Event Pulse to Event adapter</i> from CoreSight™ System-on-Chip SoC-600, you must be a licensee of the SoC-600 product. If you are not a licensee of SoC-600, you must add your own logic. For guidance on how to add your own logic, see the CMN-600AE Event Interface Connections Application Note.</p>

For more information, see your processor or DSU documentation.

A.1.12 TCU tie-off signals

The TCU tie-off signals are sampled between exiting reset and the LPI_PD interface first entering the Q_RUN state. Ensure that the value of these signals does not change when the LPI_PD interface is in the Q_STOPPED or Q_EXIT state for the first time after exiting reset.

The following table shows the TCU tie-off signals.

Table A-12: TCU tie-off signals

Signal	Width, in bits	Direction	Description
sup_cohacc	1	Input	This signal indicates whether the QTW interface is I/O-coherent. Tie HIGH when the TCU is connected to a coherent interconnect.
sup_btm	1	Input	This signal indicates whether the Broadcast TLB Maintenance is supported. Tie HIGH when the TCU is connected to an interconnect that supports DVM.
sup_sev	1	Input	This signal indicates whether the Send Event mechanism is supported. Tie HIGH when evento is connected.
sup_oas[2:0]	3	Input	Output address size supported. The encodings for this input are as follows: <div> 0b000 32 bits 0b001 36 bits 0b010 40 bits 0b011 42 bits 0b100 44 bits 0b101 48 bits 0b110 52 bits </div> You must not use other encodings. Other encodings are treated as 0b110.
sec_override	1	Input	When HIGH, certain registers are accessible to Non-secure accesses from reset, as the 4.7.7 TCU_SCR register on page 131 settings describe
ecorevnum[3:0]	4	Input	Tie this signal to 0 unless directed otherwise by Arm
msi_addr[51:0]	52	Input	If the programmed <i>Message Signaled Interrupt</i> (MSI) address in SMMU_(S)_*_IRQ_CFG0.ADDR matches msi_addr , then an MSI is generated on the TCU MSI interface
tcu_sid[31:0]	32	Input	Used as the DeviceID for TCU-generated MSIs. Note: This is only for MSIs that are issued from the dedicated AXI4-Stream MSI delivery interface.

Signal	Width, in bits	Direction	Description
sup_httu	1	Input	<p>0 When set to 0, sup_httu indicates that the ACE-Lite interface that is connected to a system that cannot support atomics. The TCU cannot perform <i>Hardware Translation Table Update</i> (HTTU) transactions.</p> <p>1 When set to 1, sup_httu indicates that the ACE-Lite interface that is connected to a system that can support atomics. The TCU uses atomic transactions to perform HTTU.</p> <p>The impact of sup_httu on SMMU_IDR0.HTTU is as follows:</p> <p>sup_httu is 1 'b0 SMMU_IDR0.HTTU is 2 'b00</p> <p>sup_httu is 1 'b1 SMMU_IDR0.HTTU is 2 'b10</p> <p>See 3.4.1 SMMUV3 implementation on page 77.</p>

For more information about the SMMUV3 ID signals, see the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#).

A.1.13 TCU ELA debug signals

The MMU-700 TCU includes *Embedded Logic Analyzer* (ELA) debug signals.

The following table shows the ELA enable signal.

Table A-13: ELA enable signal

Signal	Width, in bits	Direction	Description
ela_enable	1	Input	ela_enable is an asynchronous input port. When <i>TCUCFG_USE_ELA_DEBUG</i> is 0, the SMMU ignores the value of the signal. When <i>TCUCFG_USE_ELA_DEBUG</i> is 1, ela_enable acts as a clock enable for the TCU ELA observation interface. If ELA debug is required, drive ela_enable HIGH. If ELA debug is not required, drive ela_enable LOW to reduce the dynamic power consumption of the SMMU.

The following table shows the TCU ELA debug signals.

Table A-14: TCU ELA debug signals

Signal	Width, in bits	Direction	Description
Signal group 0 signals			
signalgrp0	128	Output	See B.1 TCU observation interfaces on page 228
sigqual0	4		
sigclken0	1		
Signal group 1 signals			
signalgrp1	128	Output	See B.1 TCU observation interfaces on page 228
sigqual1	4		
sigclken1	1		

Signal	Width, in bits	Direction	Description
Signal group 2 signals			
signalgrp2	128	Output	See B.1 TCU observation interfaces on page 228
sigqual2	4		
sigclken2	1		
Signal group 3 signals			
signalgrp3	128	Output	See B.1 TCU observation interfaces on page 228
sigqual3	4		
sigclken3	1		
Signal group 4 signals			
signalgrp4	128	Output	See B.1 TCU observation interfaces on page 228
sigqual4	4		
sigclken4	1		
Signal group 5 signals			
signalgrp5	128	Output	See B.1 TCU observation interfaces on page 228
sigqual5	4		
sigclken5	1		
Signal group 6 signals			
signalgrp6	128	Output	See B.1 TCU observation interfaces on page 228
sigqual6	4		
sigclken6	1		
Signal group 7 signals			
signalgrp7	128	Output	See B.1 TCU observation interfaces on page 228
sigqual7	4		
sigclken7	1		
Signal group 8 signals			
signalgrp8	128	Output	See B.1 TCU observation interfaces on page 228
sigqual8	4		
sigclken8	1		
Signal group 9 signals			
signalgrp9	128	Output	See B.1 TCU observation interfaces on page 228
sigqual9	4		
sigclken9	1		
Signal group 10 signals			
signalgrp10	128	Output	See B.1 TCU observation interfaces on page 228
sigqual10	4		
sigclken10	1		
Signal group 11 signals			
signalgrp11	128	Output	See B.1 TCU observation interfaces on page 228
sigqual11	4		

Signal	Width, in bits	Direction	Description
sigclken11	1		

A.2 TBU signals

This section describes the MMU-700 TBU signals.

A.2.1 TBU clock and reset signals

The TBU uses a single set of standard clock and reset signals.

The following table shows the clock and reset signals.

Table A-15: Clock and reset signals

Signal	Width, in bits	Direction	Description
clk	1	Input	Global clock
resetrn	1	Input	Global reset

A.2.2 TBU TBS interface signals

The TBU TBS interface signals are based on the AMBA ACE5-Lite signals. This interface applies to the ACE-Lite TBU and Integration TBU components.

The following table shows the TBU TBS interface signals.

Table A-16: TBU TBS interface signals

Signal	Width, in bits	Direction	Description
araddr_s	64	Input	Read address
arburst_s	2	Input	Burst type
arcache_s	4	Input	Memory type
ardomain_s	2	Input	Shareability domain
arid_s	<i>TBUCFG_ID_WIDTH</i> . See ²¹	Input	Read address ID
arlen_s	8	Input	Burst length
arlock_s	1	Input	Lock type
arloop_s	<i>TBUCFG_LOOP_WIDTH</i> . See ²²	Input	Loopback value for a read transaction. Reflected back on RLOOP .

²¹ *TBUCFG_ID_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

²² *TBUCFG_LOOP_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

Signal	Width, in bits	Direction	Description
armmussid_s	<i>TBUCFG_SSID_WIDTH</i> . See ²³	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. See 3.2.2.2 ACE-Lite TBU TBM interface on page 40 and 3.4.2 AMBA implementation on page 80.
armmusid_s	<i>TBUCFG_SID_WIDTH</i> . See ²⁴	Input	
armmussidv_s	1	Input	
armmusecsid_s	1	Input	
arprot_s	3	Input	Protection type
arqos_s	4	Input	<i>Quality of Service</i> (QoS)
aready_s	1	Output	Read address ready
arregion_s	4	Input	Region identifier
arsize_s	3	Input	Burst size
aridunq_s	1	Input	Read address channel unique ID indicator, active-HIGH
arsnoop_s	4	Input	Transaction type of read transaction
aruser_s	See ²⁵	Input	Read address (AR) channel User signal
arvalid_s	1	Input	Read address valid
rchunknum_s	<i>CHUNKNUM_WIDTH</i>	Input	Read data chunk number
rchunkstrb_s	<i>CHUNKSTRB_WIDTH</i>	Input	Read data chunk strobe
rchunkv_s	1	Input	Valid signal of RCHUNKNUM and RCHUNKSTRB
rdata_s	<i>TBUCFG_DATA_WIDTH</i> . See ²⁶	Output	Read data
rid_s	<i>TBUCFG_ID_WIDTH</i> . See ²⁷	Output	Read ID
ridunq_s	1	Input	Read data channel unique ID indicator, active-HIGH
rlast_s	1	Output	Read last

²³ *TBUCFG_SSID_WIDTH*-bit. See [3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters](#) on page 97.

²⁴ *TBUCFG_SID_WIDTH*-bit. See [3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters](#) on page 97.

²⁵ (*TBUCFG_ARUSER_WIDTH* + *LTI_TLBLOC_WIDTH_RAW* -)-bit.

Calculate the *LTI_TLBLOC_WIDTH_RAW* internal parameter as follows:

$$LTI_TLBLOC_WIDTH_RAW = (TBUCFG_DIRECT_IDX = 1) ? (TBUCFG_MTLB_DEPTH > 0) ? \log_2(TBUCFG_MTL \setminus B_DEPTH) : \log_2(4) : \log_2(TBUCFG_MTLB_PARTS)$$

When you add TBU_LTI interface signals, you must size the **latlbloc** signal. Calculate the width of **latlbloc** as follows:

$$LTI_TLBLOC_WIDTH = (LTI_TLBLOC_WIDTH_RAW < 1) ? 1 : LTI_TLBLOC_WIDTH_RAW$$

See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

²⁶ *TBUCFG_DATA_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

²⁷ *TBUCFG_ID_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

Signal	Width, in bits	Direction	Description
rloop_s	<i>TBUCFG_LOOP_WIDTH</i> . See ²⁸	Input	Loopback value for a read response
rpoison_s	<i>TBUCFG_DATA_WIDTH</i> . See ²⁹	Input	Indicates that the read data in this transfer has been corrupted
rready_s	1	Input	Read ready
rresp_s	3	Output	Read response
ruser_s	<i>TBUCFG_RUSER_WIDTH</i> . See ³⁰	Output	Read data (R) channel User signal
rvalid_s	1	Output	Read valid
awaddr_s	64	Input	Write address
awatop_s	6	Input	Atomic operation
awburst_s	2	Input	Burst type
awcache_s	4	Input	Memory type
awdomain_s	4	Input	Shareability domain
awid_s	<i>TBUCFG_ID_WIDTH</i> . See ³¹	Input	Write address ID
awlen_s	8	Input	Burst length
awlock_s	1	Input	Lock type
awmmussid_s	<i>TBUCFG_SSID_WIDTH</i> . See ³²	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. See 3.2.2.2 ACE-Lite TBU TBM interface on page 40 and 3.4.2 AMBA implementation on page 80.
awmmusid_s	<i>TBUCFG_SID_WIDTH</i> . See ³³		
awmmussidv_s	1		
awmmusecsid_s	1		
awprot_s	3	Input	Protection type
awqos_s	4	Input	QoS
awready_s	1	Output	Write address ready
awregion_s	4	Input	Region identifier
awsize_s	3	Input	Burst size
awvalid_s	1	Input	Write address valid

²⁸ *TBUCFG_LOOP_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

²⁹ (*TBUCFG_DATA_WIDTH* / 64)-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

³⁰ *TBUCFG_RUSER_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

³¹ *TBUCFG_ID_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

³² *TBUCFG_SSID_WIDTH*-bit. See [3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters](#) on page 97.

³³ *TBUCFG_SID_WIDTH*-bit. See [3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters](#) on page 97.

Signal	Width, in bits	Direction	Description
awuser_s	See ³⁴	Input	Write address (AW) channel User signal
awakeup_s	1	Input	Wakeup signal
awsnoop_s[3]	4	Input	Transaction type of write transaction
awstashnid_s[10:0]	11	Input	These signals are defined by the AXI5 Cache_Stash_Transactions extension. If <i>TBUCFG_STASH</i> = 0, these signals are ignored.
awstashniden_s	1	Input	
awstashlpid_s[4:0]	5	Input	
awstashlpiden_s	1	Input	
archunken_s	1	Input	Read data chunking enable
awidunq_s	1	Input	Write address channel unique ID indicator, active-HIGH
awloop_s	<i>TBUCFG_LOOP_WIDTH</i> . See ³⁵	Input	Loopback value for a write transaction
wdata_s	<i>TBUCFG_DATA_WIDTH</i> . See ³⁶	Input	Write data
wlast_s	1	Input	Write last
wpoison_s	<i>TBUCFG_DA \</i> <i>TA_WIDTH / 64</i> . See ³⁷	Input	Indicates that the write data in this transfer has been corrupted
wready_s	1	Output	Write ready
wstrb_s	<i>TBUCFG_DA \</i> <i>TA_WIDTH / 8</i> . See ³⁸	Input	Write strobes
wuser_s	<i>TBUCFG_WUSER_WIDTH</i> . See ³⁹	Input	Write data (W) channel User signal
wvalid_s	1	Input	Write valid
bid_s	<i>TBUCFG_ID_WIDTH</i> . See ⁴⁰	Output	Response ID

³⁴ $(TBUCFG_AWUSER_WIDTH + LTI_TLBLOC_WIDTH_RAW -)$ -bit.

Calculate the *LTI_TLBLOC_WIDTH_RAW* internal parameter as follows:

$$LTI_TLBLOC_WIDTH_RAW = (TBUCFG_DIRECT_IDX == 1) ? (TBUCFG_MTLB_DEPTH > 0) ? \log_2(TBUCFG_MTL \setminus B_DEPTH) : \log_2(4) : \log_2(TBUCFG_MTLB_PARTS)$$

When you add TBU_LTI interface signals, you must size the **latlbloc** signal. Calculate the width of **latlbloc** as follows:

$$LTI_TLBLOC_WIDTH = (LTI_TLBLOC_WIDTH_RAW < 1) ? 1 : LTI_TLBLOC_WIDTH_RAW$$

See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

³⁵ *TBUCFG_LOOP_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

³⁶ *TBUCFG_DATA_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

³⁷ $[(TBUCFG_DATA_WIDTH / 64)$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

³⁸ $(TBUCFG_DATA_WIDTH / 8)$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

³⁹ *TBUCFG_WUSER_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁴⁰ *TBUCFG_ID_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

Signal	Width, in bits	Direction	Description
bidunq_s	1	Input	Write response channel unique ID indicator, active-HIGH
bloop_s	<i>TBUCFG_LOOP_WIDTH</i> . See ⁴¹	Input	Loopback value for a write response
bready_s	1	Input	Response ready
bresp_s	3	Output	Write response
buser_s	<i>TBUCFG_BUSER_WIDTH</i> . See ⁴²	Output	Write response (B) channel User signal
bvalid_s	1	Output	Write response valid

A.2.3 TBU TBM interface signals

The TBU TBM interface signals are based on the AMBA ACE5-Lite signals. This interface applies to the ACE-Lite TBU and Integration TBU components.

The following table shows the TBU TBM interface signals.

Table A-17: TBU TBM interface signals

Signal	Width, in bits	Direction	Description
araddr_m	52	Output	Read address
arburst_m	2	Output	Burst type
arcache_m	4	Output	Memory type
archunken_m	1	Output	Read data chunking enable
ardomain_m	2	Output	Shareability domain
arid_m	<i>TBUCFG_ID_WIDTH</i> . See ⁴³	Output	Read address ID
arlen_m	8	Output	Burst length
arlock_m	1	Output	Lock type
arprot_m	3	Output	Protection type
arqos_m	4	Output	Quality of Service (QoS)
arready_m	1	Input	Read address ready
arregion_m	4	Output	Region identifier
arsize_m	3	Output	Burst size
armmusid_m	<i>TBUCFG_SID_WIDTH</i> . See ⁴⁴	Output	These signals indicate the StreamID of the originating transaction
armmusecsid_m	1	Output	

⁴¹ *TBUCFG_LOOP_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁴² *TBUCFG_BUSER_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁴³ *TBUCFG_ID_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁴⁴ *TBUCFG_SID_WIDTH*-bit. See [3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters](#) on page 97.

Signal	Width, in bits	Direction	Description
aruser_m	$TBUCFG_ARUSER_WIDTH$ See ⁴⁵	Output	Read address (AR) channel User signal
arvalid_m	1	Output	Read address valid
aridunq_m	1	Output	Read address channel unique ID indicator, active-HIGH
arloop_m	See ⁴⁶	Output	Loopback value for a read transaction. Reflected back on RLOOP
armmuflow_s	2	Input	Indicates the SMMU flow for managing translation faults
armpam_m	11	Output	Read address channel MPAM information
arsnoop_m	4	Output	Transaction type of read transaction
rchunknum_m	1, 5, 6, 7, or 8. See ⁴⁷	Output	Read data chunk number
rchunkstrb_m	1 or $TBUCFG_DA\backslash TA_WIDTH / 128$. See ⁴⁸	Output	Read data chunk strobe
rchunkv_m	1	Output	Valid signal of RCHUNKNUM and RCHUNKSTRB
rdata_m	$TBUCFG_DATA_WIDTH$. See ⁴⁹	Input	Read data
rid_m	$TBUCFG_ID_WIDTH$. See ⁵⁰	Input	Read ID
ridunq_m	1	Output	Read data channel unique ID indicator, active-HIGH
rlast_m	1	Input	Read last
rloop_m	See ⁵¹	Output	Loopback value for a read response
rpoison_m	$TBUCFG_DA\backslash TA_WIDTH / 64$. See ⁵²	Output	Indicates that the read data in this transfer has been corrupted

⁴⁵ ($TBUCFG_ARUSER_WIDTH + 5$)-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁴⁶ If $TBUCFG_OT_TRACKER_TYPE$ is 1, the width of **arloop_m** is ($TBUCFG_LOOP_WIDTH + 2$)-bit. Otherwise, the width of **arloop_m** is $TBUCFG_LOOP_WIDTH$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁴⁷ If $TBUCFG_DATA_WIDTH$ is 128, then the width of **rchunknum_m** is 8-bit.

If $TBUCFG_DATA_WIDTH$ is 256, then the width of **rchunknum_m** is 7-bit.

If $TBUCFG_DATA_WIDTH$ is 512, then the width of **rchunknum_m** is 6-bit.

If $TBUCFG_DATA_WIDTH$ is 1024, then the width of **rchunknum_m** is 5-bit.

Otherwise, the width of **rchunknum_m** is 1-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁴⁸ If $TBUCFG_DATA_WIDTH$ is 64, then the width of **rchunkstrb_m** is 1-bit. Otherwise, the width of **rchunkstrb_m** is ($TBUCFG_DATA_WIDTH / 128$)-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁴⁹ $TBUCFG_DATA_WIDTH$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁵⁰ $TBUCFG_ID_WIDTH$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁵¹ If $TBUCFG_OT_TRACKER_TYPE$ is 1, the width of **rloop_m** is ($TBUCFG_LOOP_WIDTH + 2$)-bit. Otherwise, the width of **rloop_m** is $TBUCFG_LOOP_WIDTH$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁵² ($TBUCFG_DATA_WIDTH / 64$)-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

Signal	Width, in bits	Direction	Description
rready_m	1	Output	Read ready
rresp_m	2	Input	Read response
ruser_m	<i>TBUCFG_RUSER_WIDTH</i> . See ⁵³	Input	Read data (R) channel User signal
rvalid_m	1	Input	Read valid
awaddr_m	52	Output	Write address
awatop_m	6	Output	Atomic operation
awburst_m	2	Output	Burst type
awcache_m	4	Output	Memory type
awdomain_m	2	Output	Shareability domain
awid_m	<i>TBUCFG_ID_WIDTH</i> . See ⁵⁴	Output	Write address ID
awlen_m	8	Output	Burst length
awlock_m	1	Output	Lock type
awmmusid_m	<i>TBUCFG_SID_WIDTH</i> . See ⁵⁵	Output	These signals indicate the StreamID of the originating transaction.
awmmusecsid_m	1		The <i>Generic Interrupt Controller</i> (GIC) uses these signals to determine the DeviceID of MSIs that originate from upstream masters.
awprot_m	3	Output	Protection type
awqos_m	4	Output	QoS
awready_m	1	Input	Write address ready
awregion_m	4	Output	Region identifier
awsize_m	3	Output	Burst size
awstashnid_m	11	Output	The AXI5 Cache_Stash_Transactions extension defines these signals. See AMBA® AXI and ACE Protocol Specification .
awstashniden_m	1	Output	
awstashlpid_m	5	Output	
awstashlpiden_m	1	Output	If <i>TBUCFG_STASH</i> = 0, these signals are ignored.
awakeup_m	1	Output	Wakeup signal
awidunq_m	1	Output	Write address channel unique ID indicator, active-HIGH
awloop_m	See ⁵⁶	Output	Loopback value for a write transaction
awmmuflow_s	2	Input	Indicates the SMMU flow for managing translation faults
awmpam_m	11	Output	Write address channel MPAM information
awsnoop_m[3:0]	4	Output	Transaction type of write transaction

⁵³ *TBUCFG_RUSER_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁵⁴ *TBUCFG_ID_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁵⁵ *TBUCFG_SID_WIDTH*-bit. See [3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters](#) on page 97.

⁵⁶ If *TBUCFG_OT_TRACKER_TYPE* is 1, the width of **awloop_m** is (*TBUCFG_LOOP_WIDTH* + 2)-bit. Otherwise, the width of **awloop_m** is *TBUCFG_LOOP_WIDTH*-bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

Signal	Width, in bits	Direction	Description
awuser_m	$TBUCFG_AWUSER_WIDTH$ See ⁵⁷	Output	Write address (AW) channel User signal
awvalid_m	1	Output	Write address valid
wdata_m	$TBUCFG_DATA_WIDTH$ See ⁵⁸	Output	Write data
wlast_m	1	Output	Write last
wpoison_m	$TBUCFG_DATA_WIDTH / 64$. See ⁵⁹	Output	Indicates that the write data in this transfer has been corrupted
wready_m	1	Input	Write ready
wstrb_m	$TBUCFG_DATA_WIDTH / 8$. See ⁶⁰	Output	Write strobes
wvalid_m	1	Output	Write valid
wuser_m	$TBUCFG_WUSER_WIDTH$ See ⁶¹	Output	Write data (W) channel User signal
bidunq_m	1	Input	Write response channel unique ID indicator, active-HIGH
bid_m	$TBUCFG_ID_WIDTH$. See ⁶²	Input	Response ID
bloop_m	See ⁶³	Output	Loopback value for a write response
bready_m	1	Output	Response ready
bresp_m	2	Input	Write response
buser_m	$TBUCFG_BUSER_WIDTH$ See ⁶⁴	Input	Write response (B) channel User signal
bvalid_m	1	Input	Write response valid

⁵⁷ $(TBUCFG_AWUSER_WIDTH + 5)$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁵⁸ $TBUCFG_DATA_WIDTH$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁵⁹ $(TBUCFG_DATA_WIDTH / 64)$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁶⁰ $(TBUCFG_DATA_WIDTH / 8)$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁶¹ $TBUCFG_WUSER_WIDTH$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁶² $TBUCFG_ID_WIDTH$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁶³ If $TBUCFG_OT_TRACKER_TYPE$ is 1, the width of **awloop_m** is $(TBUCFG_LOOP_WIDTH + 2)$ -bit. Otherwise, the width of **awloop_m** is $TBUCFG_LOOP_WIDTH$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

⁶⁴ $TBUCFG_BUSER_WIDTH$ -bit. See [3.5.7 ACE-Lite Translation Buffer Unit I/O configuration parameters](#) on page 101.

A.2.4 TBU PMU snapshot interface signals

The following table shows the TBU PMU snapshot interface signals.

Table A-18: TBU PMU snapshot interface signals

Signal	Width, in bits	Direction	Description
pmusnapshot_req	1	Input	PMU snapshot request. The PMU snapshot occurs on the rising edge of pmusnapshot_req . Note: Connect to the debug infrastructure of your SoC.
pmusnapshot_ack	1	Output	PMU snapshot acknowledge. The TBU uses this signal to acknowledge that the PMU snapshot has occurred. This signal is LOW after reset. Note: Connect to the debug infrastructure of your SoC.

A.2.5 TBU LPI_PD interface signals

The following table shows the TBU LPI_PD interface signals.

Table A-19: TBU LPI_PD interface signals

Signal	Width, in bits	Direction	Description
qactive_pd	1	Output	Component active
qreqn_pd	1	Input	Quiescence request
qacceptn_pd	1	Output	Quiescence accept
qdeny_pd	1	Output	Quiescence deny

For more information about these signals, see the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

A.2.6 TBU LPI_CG interface signals

The following table shows the TBU LPI_CG interface signals.

Table A-20: TBU LPI_CG interface signals

Signal	Width, in bits	Direction	Description
qactive_cg	1	Output	Component active
qreqn_cg	1	Input	Quiescence request
qacceptn_cg	1	Output	Quiescence accept
qdeny_cg	1	Output	Quiescence deny

For more information about these signals, see the [AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces](#).

A.2.7 TBU DTI interface signals

The following table shows the TBU DTI interface signals.

Table A-21: TBU DTI interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_dn	1	(Master to slave), Output	Flow control signal
tready_dti_dn	1	(Slave to master), Input	Flow control signal
tdata_dti_dn	160	(Master to slave), Output	Message data signal
tlast_dti_dn	1	(Master to slave), Output	Indicates the last cycle of a message
tkeep_dti_dn	20	(Master to slave), Output	Indicates valid bytes
tvalid_dti_up	1	(Slave to master), Input	Flow control signal
tready_dti_up	1	(Master to slave), Output	Flow control signal
tdata_dti_up	160	(Slave to master), Input	Message data signal
tlast_dti_up	1	(Slave to master), Input	Indicates the last cycle of a message
tkeep_dti_up	20	(Slave to master), Input	Indicates valid bytes
twakeup_dti_up	1	(Slave to master), Input	Wakeup signal
twakeup_dti_dn	1	(Master to slave), Output	Wakeup signal

For more information about the DTI signals, see the [AMBA® 4 AXI4-Stream Protocol Specification](#).

For more information about DTI protocol messages, see the [AMBA® DTI Protocol Specification](#).

A.2.8 TBU LTI interface signals

This interface applies to the TBU LTI components.

This interface implements an LTI interface as the [AMBA® LTI Protocol Specification](#) defines. The [AMBA LTI Specification](#) uses several properties to define signal widths. [3.4.4 Local Translation Interface implementation](#) on page 93 describes how the MMU-700 defines the LTI interface properties. For TBU LTI components with multiple interfaces, these have 2, 4, or 8 instances of an LTI interface. In these components, each LTI signal has the port number appended, starting from 0.

A.2.9 TBU interrupt signals

The TBU interrupt signals are edge-triggered. The interrupt controller must detect the rising edge of these signals.

The MMU-700 TBU cannot output these interrupts as *Message Signaled Interrupts* (MSIs). These signals must be connected to an interrupt controller.

The following table shows the TBU interrupt signals.

Table A-22: TBU interrupt signals

Signal	Width, in bits	Direction	Description
ras_fhi	1	Output	Fault handling RAS interrupt for a contained error
ras_eri	1	Output	Error recovery RAS interrupt for an uncontained error
ras_cri	1	Output	Critical error interrupt, for an uncontrollable uncorrected error
pmu_irpt	1	Output	PMU interrupt

A.2.10 TBU tie-off signals

The TBU tie-off signals are sampled between exiting reset and the LPI_PD interface first entering the Q_RUN state. Ensure that the value of these signals does not change when the LPI_PD interface is in the Q_STOPPED or Q_EXIT state for the first time after exiting reset.

The following table shows the TBU tie-off signals.

Table A-23: TBU tie-off signals

Signal	Width, in bits	Direction	Description
ns_sid_high	32 - <i>TBUCFG_SID_WIDTH</i> See ⁶⁵	Input	Provides the high-order StreamID bits for all transactions with a Non-secure StreamID that pass through the TBU
s_sid_high	32 - <i>TBUCFG_SID_WIDTH</i> See ⁶⁶	Input	Provides the high-order StreamID bits for all transactions with a Secure StreamID that pass through the TBU
max_tok_trans	$\log_2 T \setminus$ <i>BUCFG_XLATE_SLOTS</i> . See ⁶⁷	Input	Indicates the number of DTI translation tokens to request when connecting to the TCU, minus 1

⁶⁵ (32 - *TBUCFG_SID_WIDTH*)-bit. See [3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters](#) on page 97.

⁶⁶ (32 - *TBUCFG_SID_WIDTH*)-bit. See [3.5.4 Common Local Translation Interface and ACE-Lite Translation Buffer Unit configuration parameters](#) on page 97.

⁶⁷ ($\log_2 TBUCFG_XLATE_SLOTS$)-bit. See [3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters](#) on page 98.

Signal	Width, in bits	Direction	Description
pcie_mode	1	Input	<p>You must tie this signal HIGH when the TBU is connected to a PCIe interface.</p> <p>When this signal is HIGH, the TBU interprets the input AXI memory types as encoding PCI <i>No Snoop</i> information.</p> <p>For the TBU to provide correct operation, transactions from the PCIe interface must be delivered to the TBU with the following AXI memory types:</p> <p>Normal Non-Cacheable Bufferable When <i>No Snoop</i> is set for the transaction</p> <p>Write-Back When <i>No Snoop</i> is not set for the transaction</p> <p>If this signal is HIGH, the attributes of TBS interface transactions are always combined with the translation attributes, even if stage 1 translation is enabled. That is, the transaction attributes are always calculated as if the DTI_TBU_TRANS_RESP.STRW field is EL1-S2, regardless of the actual STRW value.</p> <p>If this signal is HIGH, the input attribute and shareability override information in the ATTR_OVR field of the DTI_TBU_TRANS_RESP message is ignored. For SMMUv3, PCIe masters do not support this feature.</p>
sec_override	1	Input	When HIGH, some registers are accessible to Non-secure accesses from reset, as the 4.14.3 TBU_SCR register on page 171 settings describe.
ecorevnum[3:0]	4	Input	Tie this signal to 0 unless we recommend otherwise
utlb_roundrobin	1	Input	<p>Defines the MicroTLB entry replacement policy.</p> <p>When LOW, the MicroTLB uses a <i>Pseudo Least Recently Used</i> (PLRU) replacement policy. This policy typically provides the best average performance.</p> <p>When HIGH, the MicroTLB uses a round-robin replacement policy. With this policy, the oldest entry is evicted when the MicroTLB is full.</p> <p>Tie this signal HIGH if you want to prevent newer translations from being evicted, even if older translations have been used more recently. Otherwise, tie this signal LOW.</p>
poison_support	1	Input	<p>Note: poison_support applies only to the ACE-Lite TBU. Determines how the ACE-Lite TBU handles RAS errors in the write data buffer.</p> <p>When LOW, the ACE-Lite TBU does not drive the wpoison signal HIGH after detecting an uncorrectable error in the write data buffer and reports an uncontainable uncorrected RAS error.</p> <p>When HIGH, the ACE-Lite TBU does drive the wpoison signal HIGH after detecting an uncorrectable error in the write data buffer and reports a deferred RAS error. wpoison is driven HIGH for all write data beats of the corrupted transaction. This does not affect the pass-through of the wpoison signal from the TBS interface to the TBM interface, or rpoison from the TBM interface to the TBS interface. That is, if poison_support is LOW, and wpoison is driven HIGH on the TBS interface, then the ACE-Lite TBU drives the TBM wpoison HIGH for the related transaction. However, it is expected that poison_support is a system-wide setting, and that no components in the system can generate poison if poison_support is LOW.</p>

A.2.11 TBU ELA debug signals

The MMU-700 TBU includes *Embedded Logic Analyzer* (ELA) debug signals.

The following table shows the ELA enable signal.

Table A-24: ELA enable signal

Signal	Width, in bits	Direction	Description
ela_enable	1	Input	ela_enable is an asynchronous input port. When <i>TBUCFG_USE_ELA_DEBUG</i> is 0, the SMMU ignores the value of the signal. When <i>TBUCFG_USE_ELA_DEBUG</i> is 1, ela_enable acts as a clock enable for the TBU ELA observation interface. If ELA debug is required, drive ela_enable HIGH. If ELA debug is not required, drive ela_enable LOW to reduce the dynamic power consumption of the SMMU.

ACE-Lite TBU ELA debug signals

The following table shows the ACE-Lite TBU ELA debug signals.

Table A-25: ACE-Lite TBU ELA debug signals

Signal	Width, in bits	Direction	Description
Signal group 0 signals			
signalgrp0	128	Output	See B.2 ACE-Lite TBU observation interfaces on page 231.
sigqual0	4		
sigclken0	1		
Signal group 1 signals			
signalgrp1	128	Output	See B.2 ACE-Lite TBU observation interfaces on page 231.
sigqual1	4		
sigclken1	1		
Signal group 2 signals			
signalgrp2	128	Output	See B.2 ACE-Lite TBU observation interfaces on page 231.
sigqual2	4		
sigclken2	1		
Signal group 3 signals			
signalgrp3	128	Output	See B.2 ACE-Lite TBU observation interfaces on page 231.
sigqual3	4		
sigclken3	1		
Signal group 4 signals			
signalgrp4	128	Output	See B.2 ACE-Lite TBU observation interfaces on page 231.
sigqual4	4		
sigclken4	1		

LTI TBU ELA debug signals

The following table shows the LTI TBU ELA debug signals.

Table A-26: LTI TBU ELA debug signals

Signal	Width, in bits	Direction	Description
Signal group 0 signals			
signalgrp0	128	Output	See B.3 LTI TBU observation interfaces on page 234.
sigqual0	4		
sigclken0	1		
Signal group 1 signals			
signalgrp1	128	Output	See B.3 LTI TBU observation interfaces on page 234.
sigqual1	4		
sigclken1	1		
Signal group 2 signals			
signalgrp2	128	Output	See B.3 LTI TBU observation interfaces on page 234.
sigqual2	4		
sigclken2	1		
Signal group 3 signals			
signalgrp3	128	Output	See B.3 LTI TBU observation interfaces on page 234.
sigqual3	4		
sigclken3	1		
Signal group 4 signals			
signalgrp4	128	Output	See B.3 LTI TBU observation interfaces on page 234.
sigqual4	4		
sigclken4	1		
Signal group 5 signals			
signalgrp5	128	Output	See B.3 LTI TBU observation interfaces on page 234.
sigqual5	4		
sigclken5	1		
Signal group 6 signals			
signalgrp6	128	Output	See B.3 LTI TBU observation interfaces on page 234.
sigqual6	4		
sigclken6	1		

A.3 TCU and TBU shared signals

This section describes the MMU-700 shared TCU and TBU signals.

A.3.1 TCU and TBU test and debug signals

The test and debug signals are common to the TCU and TBU.

The following table shows the test and debug signals.

Table A-27: Test and debug signals

Signal	Width, in bits	Direction	Description
dftcgen	1	Input	Clock gate enable. To enable architectural clock gates for the clock, clk , set this signal HIGH during scan shift.
dftrstdisable	1	Input	Reset disable. To disable reset, set this signal HIGH during scan shift.
dftramhold	1	Input	Preserve RAM state. To preserve the state of the RAMs and their connected registers, set this signal HIGH during scan shift.
MBISTRESETN	1	Input	MBIST mode reset. This active-LOW signal is encoded as follows: <div> 0 Reset MBIST functional logic. 1 Normal operation. </div> To prevent unintended reset of the functional logic, keep the MBISTRESETN signal in the inactive state, HIGH, during scan testing
MBISTREQ	1	Input	MBIST test request. This signal is encoded as follows: <div> 0 Normal operation. 1 Enable MBIST testing. </div>

A.4 DTI signals

This section describes the MMU-700 DTI signals.

A.4.1 DTI interconnect switch signals

The DTI interconnect switch provides signals for each of its interfaces.

The switch provides one DN_*Sn* slave downstream interface per slave interface. The following table shows the DN_*Sn* signals.

Table A-28: DTI interconnect switch DN_*Sn* interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_dn_<i>sn</i>	1	Input, slave to master	Flow control signal
tready_dti_dn_<i>sn</i>	1	Output, master to slave	Flow control signal
tdata_dti_dn_<i>sn</i>	<i>DATA_WIDTH</i> . Can be 32, 80, or 160. See ⁶⁸	Slave to master	Message data signal

⁶⁸ *DATA_WIDTH* (width of the payload). Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.

Signal	Width, in bits	Direction	Description
tid_dti_dn_sn	ID_WIDTH. See ⁶⁹	Input, slave to master	Indicates the master that initiated the message
tlast_dti_dn_sn	1	Input, slave to master	Indicates the last cycle of a message
tkeep_dti_dn_sn	See ⁷⁰	Input, slave to master	Indicates valid bytes
twakeup_dti_dn_sn	1	Input, slave to master	Wakeup signal

The switch provides one UP_Sn slave upstream interface per slave interface. The following table shows the UP_Sn signals.

Table A-29: DTI interconnect switch UP_Sn interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_up_sn	1	Output, master to slave	Flow control signal
tready_dti_up_sn	1	Input, slave to master	Flow control signal
tdata_dti_up_sn	DATA_WIDTH. Can be 32, 80, or 160. See ⁷¹	Output, master to slave	Message data signal
tdest_dti_up_sn	ID_WIDTH. See ⁷²	Output, master to slave	Indicates the master that initiated the message
tlast_dti_up_sn	1	Output, master to slave	Indicates the last cycle of a message
tkeep_dti_up_sn	See ⁷³	Output, master to slave	Indicates valid bytes
twakeup_dti_up_sn	1	Output, master to slave	Wakeup signal

The switch provides a DN_M master downstream interface. The following table shows the DN_M signals.

Table A-30: DTI interconnect switch DN_M interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_dn_m	1	Output, slave to master	Flow control signal
tready_dti_dn_m	1	Input, master to slave	Flow control signal
tdata_dti_dn_m	DATA_WIDTH. Can be 32, 80, or 160. See ⁷⁴	Output, slave to master	Message data signal
tid_dti_dn_m	ID_WIDTH. See ⁷⁵	Output, slave to master	Indicates the master that initiated the message

⁶⁹ ID_WIDTH = log₂(total number of masters being switched)-bit.

⁷⁰ (DATA_WIDTH / 8)-bit.

⁷¹ DATA_WIDTH (width of the payload). Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.

⁷² ID_WIDTH = log₂(total number of masters being switched)-bit.

⁷³ (DATA_WIDTH / 8)-bit.

⁷⁴ DATA_WIDTH (width of the payload). Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.

⁷⁵ ID_WIDTH = log₂(total number of masters being switched)-bit.

Signal	Width, in bits	Direction	Description
tlast_dti_dn_m	1	Output, slave to master	Indicates the last cycle of a message
tkeep_dti_dn_m	$DATA_WIDTH / 8$	Output, slave to master	Indicates valid bytes
twakeup_dti_dn_m	1	Output, slave to master	Wakeup signal

The switch provides an UP_M master upstream interface. The following table shows the UP_M signals.

Table A-31: DTI interconnect switch UP_M interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_up_m	1	Input, master to slave	Flow control signal
tready_dti_up_m	1	Output, slave to master	Flow control signal
tdata_dti_up_m	$DATA_WIDTH$. Can be 32, 80, or 160. See ⁷⁶	Input, master to slave	Message data signal
tdest_dti_up_m	ID_WIDTH . See ⁷⁷	Input, master to slave	Indicates the master that initiated the message
tlast_dti_up_m	1	Input, master to slave	Indicates the last cycle of a message
tkeep_dti_up_m	$DATA_WIDTH / 8$	Input, master to slave	Indicates valid bytes
twakeup_dti_up_m	1	Input, slave to master	Wakeup signal

A.4.2 DTI interconnect sizer signals

The DTI interconnect sizer provides signals for each of its interfaces.

The sizer provides an LPI_CG clock gating interface. The following table shows the LPI_CG signals.

Table A-32: DTI interconnect sizer LPI_CG interface signals

Signal	Width, in bits	Direction	Description
qactive_cg	1	Output	Component active
qreqn_cg	1	Input	Quiescence request
qacceptn_cg	1	Output	Quiescence accept
qdeny_cg	1	Output	Quiescence deny

The sizer provides a DN_S slave downstream interface. The following table shows the DN_S signals.

⁷⁶ $DATA_WIDTH$ (width of the payload). Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.

⁷⁷ $ID_WIDTH = \log_2(\text{total number of masters being switched})$ -bit.

Table A-33: DTI interconnect sizer DN_S interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_dn_s	1	Input	Flow control signal
tready_dti_dn_s	1	Output	Flow control signal
tdata_dti_dn_s	<i>INPUT_DATA_WIDTH</i> . Can be 8, 32, 80, or 160. See ⁷⁸	Input	Message data signal
tid_dti_dn_s	<i>ID_WIDTH</i> . See ⁷⁹	Input	Indicates the master that initiated the message
tlast_dti_dn_s	1	Input	Indicates the last cycle of a message
tkeep_dti_dn_s	<i>INPUT_DATA_WIDTH</i> / 8	Input	Indicates valid bytes
twakeup_dti_dn_s	1	Input	Wakeup signal

The sizer provides an UP_S slave upstream interface. The following table shows the UP_S signals.

Table A-34: DTI interconnect sizer UP_S interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_up_s	1	Output	Flow control signal
tready_dti_up_s	1	Input	Flow control signal
tdata_dti_up_s	<i>INPUT_DATA_WIDTH</i> . Can be 8, 32, 80, or 160. See ⁸⁰	Output	Message data signal
tdest_dti_up_s	<i>ID_WIDTH</i> . See ⁸¹	Output	Indicates the master that initiated the message
tlast_dti_up_s	1	Output	Indicates the last cycle of a message
tkeep_dti_up_s	<i>INPUT_DATA_WIDTH</i> / 8	Output	Indicates valid bytes
twakeup_dti_up_s	1	Output	Wakeup signal

The sizer provides a DN_M master downstream interface. The following table shows the DN_M signals.

Table A-35: DTI interconnect sizer DN_M interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_dn_m	1	Input	Flow control signal
tready_dti_dn_m	1	Input	Flow control signal
tdata_dti_dn_m	<i>OUTPUT_DATA_WIDTH</i> . Can be 8, 32, 80, or 160. See ⁸²	Output	Message data signal
tid_dti_dn_m	<i>ID_WIDTH</i> . See ⁸³	Output	Indicates the master that initiated the message
tlast_dti_dn_m	1	Output	Indicates the last cycle of a message
tkeep_dti_dn_m	<i>OUTPUT_DATA_WIDTH</i> / 8	Output	Indicates valid bytes
twakeup_dti_dn_m	1	Input	Wakeup signal

The sizer provides an UP_M master upstream interface. The following table shows the UP_M signals.

⁷⁸ *INPUT_DATA_WIDTH*. Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.

⁷⁹ *ID_WIDTH* = \log_2 (total number of masters that are connected to the sizer)-bit.

⁸⁰ *INPUT_DATA_WIDTH*. Can be 160-bit, 80-bit, 32-bit, or 8-bit.

⁸¹ *ID_WIDTH* = \log_2 (total number of masters that are connected to the sizer)-bit.

⁸² *OUTPUT_DATA_WIDTH*-bit. *OUTPUT_DATA_WIDTH* can be 160-bit, 80-bit, 32-bit, or 8-bit.

⁸³ *ID_WIDTH* = \log_2 (total number of masters that are connected to the sizer)-bit.

Table A-36: DTI interconnect sizer UP_M interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_up_m	1	Input	Flow control signal
tready_dti_up_m	1	Output	Flow control signal
tdata_dti_up_m	<i>OUTPUT_DATA_WIDTH</i> . Can be 8, 32, 80, or 160. See ⁸⁴	Input	Message data signal
tdest_dti_up_m	<i>ID_WIDTH</i> . See ⁸⁵	Input	Indicates the master that initiated the message
tlast_dti_up_m	1	Input	Indicates the last cycle of a message
tkeep_dti_up_m	<i>OUTPUT_DATA_WIDTH</i> / 8	Input	Indicates valid bytes
twakeup_dti_up_m	1	Input	Wakeup signal

A.4.3 DTI interconnect register slice signals

The DTI interconnect register slice provides signals for each of its interfaces.

The register slice provides an LPI_CG clock gating interface. The following table shows the LPI_CG signals.

Table A-37: DTI interconnect register slice LPI_CG interface signals

Signal	Width, in bits	Direction	Description
qactive_cg	1	Output	Component active
qreqn_cg	1	Input	Quiescence request
qacceptn_cg	1	Output	Quiescence accept
qdeny_cg	1	Output	Quiescence deny

The register slice provides a DN_S slave downstream interface. The following table shows the DN_S signals.

Table A-38: DTI interconnect register slice DN_S interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_dn_s	1	Slave to master	Flow control signal
tready_dti_dn_s	1	Master to slave	Flow control signal
tdata_dti_dn_s	<i>DATA_WIDTH</i> . Can be 8, 32, 80, or 160. See ⁸⁶	Slave to master	Message data signal
tid_dti_dn_s	<i>ID_WIDTH</i> . See ⁸⁷	Slave to master	Indicates the master that initiated the message
tlast_dti_dn_s	1	Slave to master	Indicates the last cycle of a message
tkeep_dti_dn_s	<i>DATA_WIDTH</i> / 8	Slave to master	Indicates valid bytes

The register slice provides an UP_S slave upstream interface. The following table shows the UP_S signals.

⁸⁴ *OUTPUT_DATA_WIDTH*. Can be 160-bit, 80-bit, 32-bit, or 8-bit.

⁸⁵ *ID_WIDTH* = $\log_2(\text{total number of masters that are connected to the sizer})$ -bit.

⁸⁶ *DATA_WIDTH* of the register slice. Can be 160-bit, 80-bit, 32-bit, or 8-bit.

⁸⁷ *ID_WIDTH* = $\log_2(\text{total number of masters that are connected to the register slice})$ -bit.

Table A-39: DTI interconnect register slice UP_S interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_up_s	1	Master to slave	Flow control signal
tready_dti_up_s	1	Slave to master	Flow control signal
tdata_dti_up_s	<i>DATA_WIDTH</i> . Can be 8, 32, 80, or 160. See ⁸⁸	Master to slave	Message data signal
tdest_dti_up_s	<i>ID_WIDTH</i> . See ⁸⁹	Master to slave	Indicates the master that initiated the message
tlast_dti_up_s	1	Master to slave	Indicates the last cycle of a message
tkeep_dti_up_s	<i>DATA_WIDTH</i> / 8	Master to slave	Indicates valid bytes

The register slice provides a DN_M master downstream interface. The following table shows the DN_M signals.

Table A-40: DTI interconnect register slice DN_M interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_dn_m	1	Slave to master	Flow control signal
tready_dti_dn_m	1	Master to slave	Flow control signal
tdata_dti_dn_m	<i>DATA_WIDTH</i> . Can be 8, 32, 80, or 160. See ⁹⁰	Slave to master	Message data signal
tid_dti_dn_m	<i>ID_WIDTH</i> . See ⁹¹	Slave to master	Indicates the master that initiated the message
tlast_dti_dn_m	1	Slave to master	Indicates the last cycle of a message
tkeep_dti_dn_m	<i>DATA_WIDTH</i> / 8	Slave to master	Indicates valid bytes

The register slice provides an UP_M master upstream interface. The following table shows the UP_M signals.

Table A-41: DTI interconnect register slice UP_M interface signals

Signal	Width, in bits	Direction	Description
tvalid_dti_up_m	1	Master to slave	Flow control signal
tready_dti_up_m	1	Slave to master	Flow control signal
tdata_dti_up_m	<i>DATA_WIDTH</i> . Can be 8, 32, 80, or 160. See ⁹²	Master to slave	Message data signal
tdest_dti_up_m	<i>ID_WIDTH</i> . See ⁹³	Master to slave	Indicates the master that initiated the message
tlast_dti_up_m	1	Master to slave	Indicates the last cycle of a message
tkeep_dti_up_m	<i>DATA_WIDTH</i> / 8 ⁹⁴	Master to slave	Indicates valid bytes

⁸⁸ *DATA_WIDTH* of the register slice. Can be 160-bit, 80-bit, 32-bit, or 8-bit.

⁸⁹ *ID_WIDTH* = \log_2 (total number of masters that are connected to the register slice)-bit.

⁹⁰ *DATA_WIDTH* of the register slice. Can be 160-bit, 80-bit, 32-bit, or 8-bit.

⁹¹ *ID_WIDTH* = \log_2 (total number of masters that are connected to the register slice)-bit.

⁹² *DATA_WIDTH* of the register slice. Can be 160-bit, 80-bit, 32-bit, or 8-bit.

⁹³ *ID_WIDTH* = \log_2 (total number of masters that are connected to the register slice)-bit.

⁹⁴

Appendix B ELA signal descriptions

This section describes the **SIGNALGRP<n>**, **SIGQUAL<n>**, and **SIGCLKEN<n>** signals of the TCU and TBU components that are used to interface with external ELA.

B.1 TCU observation interfaces

This section describes the TCU observation interfaces, **SIGNALGRP<n>**, **SIGQUAL<n>**, and **SIGCLKEN<n>** signals that are used to interface to an external CoreSight™ ELA-600 Embedded Logic Analyzer. **<n>** represents the number in the signal name.

Signal group output ports are present on each component. However, only a subset is used.

The **SIGCLKEN<n>** signal is set to 1 for the signal groups in the 'Enabled signal groups' column in the following table. Groups that are not enabled have their **SIGCLKEN<n>** signals set to 0. If **ela_enable** is driven LOW, all **SIGCLKEN<n>** signals are set to 0.

The following table shows the signal group output ports that are valid for the TCU.

Table B-1: Number of signal groups per module for the TCU

Component	Parameter	Enabled signal groups	Total
TCU	<i>TCUCFG_QTW_DATA_WIDTH</i> ≤ 128	0, 1, 2, 3, 4, 5, 6, 10	8
	<i>TCUCFG_QTW_DATA_WIDTH</i> == 256	0, 1, 2, 3, 4, 5, 6, 7, 10, 11	10
	<i>TCUCFG_QTW_DATA_WIDTH</i> == 512	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	12

The following table shows the **SIGNALGRP<n>** bits for the signal groups of the TCU.

Some buses, if configured to be larger than the 128-bit signal group width, are spread across multiple groups. The MMU-700 delays sections of the signal by a cycle so that the ELA can sample 128-bit chunks of the data one cycle after another. The *Number of cycles of delay* column in the table indicates the number of cycles, from when the signal is observable on a MMU-700 interface, to when the signal is observable on the ELA observation interface.

Table B-2: TCU observation interface signals

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b[MSB..LSB]	Number of cycles of delay
0	[127:0]	tdata_dti_dn [127:0]	1'b0, (tready_dti_dn AND tvalid_dti_dn), tready_dti_dn , tvalid_dti_dn	1
1	[127:0]	tdata_dti_up [127:0]	1'b0, (tready_dti_up AND tvalid_dti_up), tready_dti_up , tvalid_dti_up	1
2	[127:124]	Unused	-	-
	[123:118]	tid_dti_dn	tvalid_dti_up , (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn , (tready_dti_dn AND tvalid_dti_dn)	0
	[117:86]	tdata_dti_dn [159:128]		

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b[MSB..LSB]	Number of cycles of delay
	[85:66]	tkeep_dti_dn		
	[65]	tlast_dti_dn		
	[64]	twakeup_dti_dn		
	[63]	tready_dti_dn		
	[62]	tvalid_dti_dn		
	[61:56]	tdest_dti_up		
	[55:24]	tdata_dti_up[159:128]		
	[23:4]	tkeep_dti_up		
	[3]	tlast_dti_up		
	[2]	twakeup_dti_up		
	[1]	tready_dti_up		
	[0]	tvalid_dti_up		
3	[127]	Unused	-	-
	[126]	ridunq_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
	[125:118]	rpoison_qtw		
	[117]	rlast_qtw		
	[116:106]	rid_qtw		
	[105]	rready_qtw		
	[104]	rvalid_qtw		
	[103:93]	arid_qtw		
	[92:92]	aridunq_qtw		
	[91:81]	armpam_qtw		
	[80:79]	ardomain_qtw		
	[78:75]	aruser_qtw		
	[74:71]	arqos_qtw		
	[70:67]	arcache_qtw		
	[66:65]	arburst_qtw		
	[64:62]	arsize_qtw		
	[61:54]	arlen_qtw		
	[53:2]	araddr_qtw		
	[1]	arready_qtw		
	[0]	arvalid_qtw		
4	[127]	Unused	-	-
	[126]	crready_qtw	1'b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
	[125]	crvalid_qtw		
	[124:114]	bid_qtw		
	[113]	bidunq_qtw		
	[112]	bready_qtw		

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b[MSB..LSB]	Number of cycles of delay
	[111]	bvalid_qtw		
	[110]	awakeup_qtw		
	[109:99]	awid_qtw		
	[98:93]	awatop_qtw		
	[92]	awidunq_qtw		
	[91:81]	awmpam_qtw		
	[80:79]	awdomain_qtw		
	[78:75]	awuser_qtw		
	[74:71]	awqos_qtw		
	[70:67]	awcache_qtw		
	[66:65]	awburst_qtw		
	[64:62]	awsize_qtw		
	[61:54]	awlen_qtw		
	[53:2]	awaddr_qtw		
	[1]	awready_qtw		
	[0]	awvalid_qtw		
5	[127]	syscoack_qtw	2'b00, (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
	[126]	syscoreq_qtw		
	[125:62]	wstrb_qtw		
	[61]	wlast_qtw		
	[60]	wready_qtw		
	[59]	wvalid_qtw		
	[58]	acwakeup_qtw		
	[57:6]	acaddr_qtw		
	[5:2]	acvmidext_qtw		
	[1]	acready_qtw		
	[0]	acvalid_qtw		
6	[127:0]	rdata_qtw[127:0]	1'b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw	1
7	[127:0]	rdata_qtw[255:128]	1'b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw	2
8	[127:0]	rdata_qtw[383:256]	1'b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw	3
9	[127:0]	rdata_qtw[511:384]	1'b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw	4

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
10	[127:0]	wdata_qtw_demuxed ⁹⁵ [127:0]	1'b0, (wready_qtw AND wvalid_qtw), wready_qtw, wvalid_qtw	1
11	[127:0]	wdata_qtw_demuxed[255:128]	1'b0, (wready_qtw AND wvalid_qtw), wready_qtw, wvalid_qtw	2

B.2 ACE-Lite TBU observation interfaces

This section describes the ACE-Lite TBU observation interfaces, **SIGNALGRP<n>**, **SIGQUAL<n>**, and **SIGCLKEN<n>** signals that are used to interface to an external CoreSight™ ELA-600 Embedded Logic Analyzer. <n> represents the number in the signal name.

Signal group output ports are present on each component. However, only a subset is used.

The **SIGCLKEN<n>** signal is set to 1 for the signal groups in the 'Enabled signal groups' column in the following table. Groups that are not enabled have their **SIGCLKEN<n>** signals set to 0. If **ela_enable** is driven LOW, all **SIGCLKEN<n>** signals are set to 0.

The following table shows the signal group output ports that are valid for the ACE-Lite TBU.

Table B-3: Number of signal groups per module for the ACE-Lite TBU

Component	Parameter	Enabled signal groups	Total
ACE-Lite TBU		0, 1, 2, 3, 4	5

The following table shows the **SIGNALGRP<n>** bits for the signal groups of the ACE-Lite TBU.

Some buses, if configured to be larger than the 128-bit signal group width, are spread across multiple groups. The MMU-700 delays sections of the signal by a cycle so that the ELA can sample 128-bit chunks of the data one cycle after another. The *Number of cycles of delay* column in the table indicates the number of cycles, from when the signal is observable on a MMU-700 interface, to when the signal is observable on the ELA observation interface.

Table B-4: ACE-Lite TBU observation interface signals

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
0	[127:123]	awuser_TLBLOC	1'b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
	[122]	awidunq_m		
	[121:116]	awatop_m		
	[115:114]	awdomain_m		
	[113:110]	awqos_m		

⁹⁵ When the *TCUQTWDATAWIDTH* parameter is set to 512, the **wdate_qtw_demuxed** signal contains the active 256 bits of the 512-bit bus. The **wstrb_qtw** signal remains as 64 bits and is unmodified. See [3.5 Configuration parameters and methodology](#) on page 94.

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
	[109:107]	awprot_m		
	[106:103]	awcache_m		
	[102:101]	awburst_m		
	[100:98]	awsize_m		
	[97:90]	awlen_m		
	[89:86]	awregion_m		
	[85:54]	awid_m		
	[53:2]	awaddr_m		
	[1]	awready_m		
	[0]	awvalid_m		
1	[127:114]	Unused	-	-
	[113:103]	awmpam_m	1'b0, (awready_m AND awvalid_m), awready_m, awvalid_m	2
	[102:83]	awmmusid_m		
	[82]	awmmusecsid_m		
	[81:72]	awloop_m		
	[71]	awstashlpiden_m		
	[70:66]	awstashlpid_m		
	[65]	awstashniden_m		
	[64:54]	awstashnid_m		
	[53:2]	awaddr_m		
	[1]	awready_m		
	[0]	awvalid_m		
2	[127:117]	Unused	-	-
	[116:65]	araddr_m	1'b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
	[64:63]	ardomain_m		
	[62:59]	arqos_m		
	[58:56]	arprot_m		
	[55:52]	arcache_m		
	[51:50]	arburst_m		
	[49:47]	arsize_m		
	[46:39]	arlen_m		
	[38:34]	aruser_TLBLOC		
	[33:2]	arid_m		
	[1]	arready_m		
	[0]	arvalid_m		
3	[127:113]	Unused	-	-
	[112:]	archunken_m	1'b0, (arready_m AND arvalid_m), arready_m, arvalid_m	2
	[111:101]	armpam_m		
	[100:91]	arloop_m		

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
	[90]	aridunq_m		
	[89:70]	armmusid_m		
	[69]	armmusecsid_m		
	[68:65]	arregion_m		
	[64:63]	ardomain_m		
	[62:59]	arqos_m		
	[58:56]	arprot_m		
	[55:52]	arcache_m		
	[51:50]	arburst_m		
	[49:47]	arsize_m		
	[46:39]	arlen_m		
	[38:34]	aruser_TLBLOC		
	[33:2]	arid_m		
	[1]	arready_m		
	[0]	arvalid_m		
4	[127:98]	Unused	-	-
	[97]	wlast_m	1'b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
	[96]	wready_m		
	[95]	wvalid_m		
	[94:85]	bloop_m		
	[84:84]	bidunq_m		
	[83:82]	bresp_m		
	[81:50]	bid_m		
	[49]	bready_m		
	[48]	bvalid_m		
	[47]	ridunq_m		
	[46:37]	rloop_m		
	[36]	rlast_m		
	[35:34]	rresp_m		
	[33:2]	rid_m		
	[1]	rready_m		
	[0]	rvalid_m		
5	[127:0]	Unused	-	-
6				
7				
8				
9				
10				
11				

B.3 LTI TBU observation interfaces

This section describes the LTI TBU observation interfaces, **SIGNALGRP<n>**, **SIGQUAL<n>**, and **SIGCLKEN<n>** signals that are used to interface to an external CoreSight™ ELA-600 Embedded Logic Analyzer. **<n>** represents the number in the signal name.

Signal group output ports are present on each component. However, only a subset is used.



The signals that this interface reports are after multiple LTI interfaces are multiplexed together, so shows traffic on all LTI interfaces.

The **SIGCLKEN<n>** signal is set to 1 for the signal groups in the 'Enabled signal groups' column in the following table. Groups that are not enabled have their **SIGCLKEN<n>** signals set to 0. If **ela_enable** is driven LOW, all **SIGCLKEN<n>** signals are set to 0.

The following table shows the signal group output ports that are valid for the LTI TBU.

Table B-5: Number of SignalGroups per module for the LTI TBU

Component	Parameter	Enabled signal groups	Total
LTI TBU	When <i>TBUCFG_LTI_LOOP_WIDTH</i> ≤ 128 bits	0, 1, 2, 3, 5	5
	When <i>TBUCFG_LTI_LOOP_WIDTH</i> > 128 bit	0, 1, 2, 3, 4, 5, 6	7

The following table shows the **SIGNALGPR<n>** bits for the signal groups of the LTI TBU.

Some buses, if configured to be larger than the 128-bit signal group width, are spread across multiple groups. The MMU-700 delays sections of the signal by a cycle so that the ELA can sample 128-bit chunks of the data one cycle after another. The *Number of cycles of delay* column in the table indicates the number of cycles, from when the signal is observable on a MMU-700 interface, to when the signal is observable on the ELA observation interface.

Table B-6: LTI TBU observation interface signals

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
0	[127:112]	latlbloc	3'b000, lavalid	0
	[111:80]	laid		
	[79:78]	laflow		
	[77:74]	laattr		
	[73:70]	latrans		
	[69:67]	laprot		
	[66:65]	lacredit		
	[64:1]	laaddr		
	[0]	lavalid		

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
1	[127]	Unused	-	-
	[126:122]	laog	3'b000, lavalid	1
	[121]	laogv		
	[120:101]	lassid		
	[100]	lassidv		
	[99:80]	lasid		
	[79]	lasecsid		
	[78]	lavc		
	[77:74]	laattr		
	[73:70]	latrans		
	[69:67]	laprot		
	[66:65]	lacredit		
	[64:1]	laaddr		
	[0]	lavalid		
2	[127:121]	Unused	-	-
	[120]	lmaskclose	2'b00, lcvalid, linvalid	0
	[119]	lmactive		
	[118]	lmopenack		
	[117]	lmopenreq		
	[116]	lcctag		
	[115]	lccredit		
	[114]	lcvalid		
	[113:103]	lrmpam		
	[102:99]	lrhwattr		
	[98:95]	lrattr		
	[94:43]	lraddr		
	[42:40]	lrprot		
	[39:37]	lrresp		
	[36]	lrctag		
	[35:4]	lrld		
	[3]	lrv		
	[2:1]	lrcredit		
	[0]	linvalid		
3	[127:0]	laloop[127:0]	3'b000, lavalid	1
4	[127:0]	laloop[255:128]	3'b000, lavalid	2
5	[127:0]	lrloop[127:0]	3'b000, linvalid	1
6	[127:0]	lrloop[255:128]	3'b000, linvalid	2
7	[127:0]	Unused	-	-
8				

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
9				
10				
11				

Appendix C Software initialization examples

This appendix provides examples of how software can initialize and enable the MMU-700.

C.1 Initializing the SMMU

Software must initialize the MMU-700 before you can use it.

The MMU-700 supports Secure and Non-secure translation worlds. This section defines how to initialize Non-secure translation. The procedures for initializing Secure translation are similar, and require you to access the corresponding MMU-700 Secure registers.



This section does not describe how to create translation tables. For more information, see the [Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile](#).

For more information about MMU-700 initialization, see the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#).

C.1.1 Allocating the Command queue

The MMU-700 uses the Command queue to receive commands. Software must allocate memory for the Command queue and configure the appropriate registers in the SMMU.

About this task

To allocate the Command queue, ensure that your software performs the following steps:

Procedure

1. Allocate memory for the Command queue.
2. Configure the Command queue size and base address by writing to the SMMU_CMDQ_BASE register.



The queue size can affect how many bits of the SMMU_CMDQ_CONS and SMMU_CMDQ_PROD indices are writeable. It is therefore important that you perform this step before writing to SMMU_CMDQ_CONS and SMMU_CMDQ_PROD.

3. Set the queue read index in `SMMU_CMDQ_CONS` and the queue write index in `SMMU_CMDQ_PROD` to 0.



Setting the queue read index and the queue write index to the same value indicates that the queue is empty.

C.1.2 Allocating the Event queue

The MMU-700 uses the Event queue to signal events. Software must allocate memory for the Event queue and configure the appropriate registers in the MMU.

About this task

To allocate the Event queue, ensure that your software performs the following steps:

Procedure

1. Allocate memory for the Event queue.
2. Configure the Event queue size and base address by writing to the `SMMU_EVENTQ_BASE` register.



The queue size can affect how many bits of the `SMMU_EVENTQ_CONS` and `SMMU_EVENTQ_PROD` indices are writeable. It is therefore important that you perform this step before writing to `SMMU_EVENTQ_CONS` and `SMMU_EVENTQ_PROD`.

3. Set the queue read index in `SMMU_EVENTQ_CONS` and the queue write index in `SMMU_EVENTQ_PROD` to 0.



Setting the queue read index and the queue write index to the same value indicates that the queue is empty.

C.1.3 Configuring the Stream table

The Stream table is a configuration structure in memory that uses a *Context Descriptor* (CD) to locate translation data for a transaction. Software must allocate memory for the Stream table, configure the table format, and populate the table with *Stream Table Entries* (STEs).

About this task

To configure the Stream table, ensure that your software performs the following steps:

Procedure

1. Allocate memory for the Stream table.

2. Configure the format and size of the Stream table by writing to `SMMU_STRTAB_BASE_CFG`.
3. Configure the base address for the Stream table by writing to `SMMU_STRTAB_BASE`.
4. Prevent uninitialized memory being interpreted as a valid configuration by setting `STE.V = 0` for each STE to mark it as invalid.
5. Ensure that written data is observable to the SMMU by performing a *Data Synchronization Barrier* (DSB) operation.
If `SMMU_IDR0.COHAAC = 0`, the system does not support coherent access to memory for the TCU. In such cases, you might require extra steps to ensure that the SMMU can observe the written data.

C.1.4 Initializing the Command queue

Software must initialize the Command queue by enabling it and checking that the enable operation is complete.

About this task

To initialize the Command queue, ensure that your software performs the following steps:

Procedure

1. Enable the Command queue by setting the `SMMU_S_CR0.CMDQEN` bit to 1.
2. Check that the enable operation is complete by polling `SMMU_S_CR0ACK` until `CMDQEN` reads as 1.

C.1.5 Initializing the Event queue

Software must initialize the Event queue by enabling it and checking that the enable operation is complete.

About this task

To initialize the Event queue, ensure that your software performs the following steps:

Procedure

1. Enable the Event queue by setting the `SMMU_S_CR0.EVENTQEN` bit to 1.
2. Check that the enable operation is complete by polling `SMMU_S_CR0ACK` until `EVENTQEN` reads as 1.

C.1.6 Invalidating TLBs and configuration caches

Before use, the MMU-700 TLBs and configuration cache structures must be invalidated by issuing commands to the Command queue. Alternatively, Secure software can invalidate all TLBs and caches with a single write.

To invalidate TLB entries, ensure that your software issues the appropriate command for the translation context. To invalidate:

- TLB entries for Non-secure EL1 contexts, issue `CMD_TLBI_NSNH_ALL`

- TLB entries for EL2 contexts, issue CMD_TLBI_EL2_ALL
- TLB entries for EL3 contexts, issue CMD_TLBI_EL3_ALL
- TLB entries for Secure EL1 contexts, issue CMD_TLBI_NH_ALL



Commands to invalidate Secure TLB entries can only be issued through the Secure Command queue. For a system that implements two Security states, Secure software must issue the appropriate command to the Secure Command queue for the first TLB invalidation. If your system does not use Secure software, you can permit Non-secure software to access SMMU_S_INIT by using either **sec_override** or the [4.7.7 TCU_SCR register](#) on page 131.

To invalidate both the TCU configuration cache and the TBU combined configuration cache and TLB, issue the CMD_CFGI_ALL command.

To force all previous commands to complete, issue CMD_SYNC.

To invalidate all configuration caches and TLB entries for all translation regimes and Security states, ensure that Secure software:

1. Sets SMMU_S_INIT.INV_ALL to 1. The SMMU sets SMMU_S_INIT.INV_ALL to 0 after the invalidation completes.
2. Polls SMMU_S_INIT.INV_ALL to check it is set to 0 before continuing the SMMU configuration.

For more information about issuing commands to the Command queue, see the [Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2](#).

C.1.7 Creating a basic Context Descriptor

A *Context Descriptor* (CD) is a data structure in system memory. A CD defines how Stage 1 translation is performed. The SubstreamID is used to select the CD.

To create a CD, ensure that your software performs the following steps:

1. Allocate 64 bytes of memory for the CD.
2. Configure the CD fields according to the information in the following table.

Table C-1: Configuring the CD

Field	Description
AA64	Translation table format: 0 AArch32. 1 AArch64.
EPD0	Enable translations for TTBO by setting EPD0 to 0.
TTB0	Base address of translation table 0.
TG0	Translation granule size for TTB0 when CD.AA64 = 1.

Field	Description
IRO	Cacheability attribute to use for translation table walks to TTBO:
ORO	00 Non-cacheable. 01 Write-Back Cacheable, Read-Allocate Write-Allocate. 10 Write-through Cacheable, Read-Allocate.
SHO	Shareability of translation table walks to TTBO:
	00 Non-shareable. 01 Outer Shareable. 10 Inner Shareable.
EPD1	If the StreamWorld supports split address spaces, enable table walks for TTB1.
ENDI	The endianness for the translation tables.
IPS	The IPA size when CD.AA64 = 1.
ASET	Defines whether the ASID values are shared with the ASID values of an Arm processor.
	Note: If you expect this context to receive broadcast TLB invalidation commands from a PE, set ASET to 0.
V	Valid CD. This field must be set to 1.

C.1.8 Creating a Stream Table Entry

Each *Stream Table Entry* (STE) configures how Stage 2 translation is performed, and how the *Context Descriptor* (CD) table can be found. The StreamID is used to select an STE.

To create an STE, ensure that your software performs the following steps:

- Allocate 64 bytes of memory for the STE.
- Set the STE.Config field as required for Stage 1 translation, Stage 2 translation, or translation bypass:

0b0b000	No traffic can pass through the MMU. An abort is returned.
0b0b100	Stage 1 and Stage 2 bypass.
0b0b101	Stage 1 translation Stage 2 bypass.
0b0b110	Stage 1 bypass Stage 2 translation.
0b0b111	Stage 1 and Stage 2 translation.
- If Stage 1 translation is enabled, you can set the following fields:

STE.S1CDMax	Controls whether STE.S1ContextPtr points to a single CD or a CD table.
STE.S1Fmt	If STE.S1CDMax > 0, configures the format of the CD table.
STE.S1ContextPtr	Contains a pointer to either a CD or a CD table. If Stage 2 translation is enabled, this pointer is an <i>intermediate physical address</i> (IPA), otherwise it is an untranslated <i>physical address</i> PA.

4. If Stage 2 translation is enabled, you can set the following fields:
- | | |
|-------------------|---|
| STE.S2TTB | Points to the Stage 2 translation table base address. |
| STE.S2PS | Contains the PA size of the stage 2 PA range. |
| STE.S2AA64 | Indicates whether the Stage 2 tables are AArch32 or AArch64 format. |
| STE.S3ENDI | Set this field to the required endianness for the stage 2 translation tables. |
| STE.S2AFFD | Disable Access Flag faults for Stage 2 translation. |
| STE.S2TG | 0b00: 4KB. |
| STE.S2IRO | 0b00: Non-cacheable. |
| and | |
| STE.S2OR0 | |
| STE.S2SH0 | |
| STE.S2VMID | Contains the VMID associated with these translations. |

C.2 Enabling the SMMU

Software can enable the SMMU by writing to SMMU_CR0 after the Stream table is populated.

About this task

To enable the SMMU, carry out the following procedure.

Procedure

1. Ensure that all Stream table entries are populated in memory.
2. Set the SMMU_CR0.SMMUEN bit to 1.
3. Check that the enable operation is complete by polling SMMU_CROACK until SMMUEN reads as 1.

Appendix D Revisions

This appendix describes the technical changes between released issues of this book.

D.1 Revisions

This appendix describes the technical changes between released issues of this book.

Table D-1: Issue 0000-01

Change	Location
First release	-

Table D-2: Differences between issue 0000-01 and issue 0000-02

Change	Location
Improvements to descriptions	Throughout the document

Table D-3: Differences between issue 0000-02 and issue 0001-03

Change	Location
Improvements to descriptions	Throughout the document
Added new parameters	<ul style="list-style-type: none"> 3.5.2 Translation Control Unit buffer configuration parameters on page 94 3.5.5 Common Local Translation Interface and ACE-Lite Translation Buffer Unit buffer configuration parameters on page 98
Added system discovery registers	<ul style="list-style-type: none"> 4.9 TCU system discovery registers on page 139 4.16 TBU system discovery registers on page 178

Table D-4: Differences between issue 0001-03 and issue 0001-04

Change	Location
Improvements to descriptions	Throughout the document
New <i>Width</i> column added to all signal description tables	A Signal descriptions on page 197

Table D-5: Differences between issue 0001-04 and issue 0100-05

Change	Location
Improvements to descriptions	Throughout the document
Updates to LTI TBU description	3.2.2.3 LTI TBU LTI interface on page 41
New <i>Integration TBU</i> section	3.2.3 Integration TBU on page 43
Updates to parameters descriptions	3.5 Configuration parameters and methodology on page 94
Updates to register descriptions	4 Programmers model on page 105

Table D-6: Differences between issue 0100-05 and issue 0100-06

Change	Location
Updates to some configuration options sections	3.5 Configuration parameters and methodology on page 94
Improvements to descriptions	Throughout the document