arm

Arm® CoreLink™ MMU-700 System Memory Management Unit

Revision: r1p2

Technical Reference Manual

Non-Confidential

Issue 08

101542_0102_08_en

Arm® CoreLink™ MMU-700 System Memory Management Unit

Technical Reference Manual

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
0102-07	9 June 2023	Non-Confidential	First issue for r1p2 REL release
0102-08	5 July 2023	Non-Confidential	Second issue for r1p2 REL release

Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com.

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

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

1.1 Product revision status

The r*x*p*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.

Convention	Use
italic	Citations.
bold	Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	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: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`

Convention	Use
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Caution

Recommendations. Not following these recommendations might lead to system failure or damage.

Warning

Requirements for the system. Not following these requirements might result in system failure or damage.

Danger

Requirements for the system. Not following these requirements will result in system failure or damage.

Note

An important piece of information that needs your attention.

Tip

A useful tip that might make it easier, better or faster to perform a task.

Remember

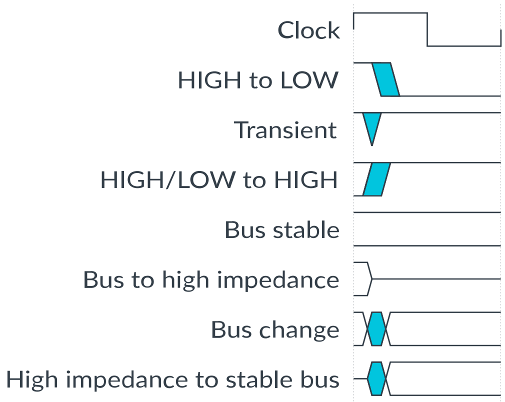
A reminder of something important that relates to the information you are reading.

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 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual	101543	Confidential
Arm® CoreLink™ MMU-700 System Memory Management Unit Release Note	107913	Confidential
Arm® CoreLink™ LPD-500 Low Power Distributor Technical Reference Manual	100361	Non-Confidential
Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual	100806	Non-Confidential

Arm product resources	Document ID	Confidentiality
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101088	Non-Confidential
Arm® CoreLink™ CMN-600AE Event Interface Connections Application Note	ARM051-799564642-325	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
Arm® System Memory Management Unit Architecture Specification, SMMU architecture version 3	IHI 0070C.a	Non-Confidential
AMBA® APB Protocol Specification	IHI 0024C	Non-Confidential
AMBA® AXI and ACE Protocol Specification	IHI 0022H	Non-Confidential
AMBA® AXI-Stream Protocol Specification	IHI 0051B	Non-Confidential
AMBA® DTI Protocol Specification	IHI 0088E.b	Non-Confidential
AMBA® Low Power Interface Specification	IHI 0068C	Non-Confidential
AMBA® LTI Protocol Specification	IHI 0089A	Non-Confidential
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487E.a	Non-Confidential
Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture	DDI 0598B.a	Non-Confidential
Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability (RAS), for Armv8-A	DDI 0587C.b	Non-Confidential
Arm® Server Base System Architecture 7.0 Platform Design Document	DEN 0029F	Non-Confidential
Arm® GIC MSI Delivery Interface	ARM AES 0019A	Confidential

Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at http://www.adobe.com.

2. Overview of MMU-700

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

2.1 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.

2.1.1 Arm architecture

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

2.1.2 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 60

2.1.3 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® AXI-Stream Protocol Specification* defines.

Related information

DTI overview on page 40

2.1.4 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 64

2.1.5 AMBA APB protocol

The MMU-700 complies with the AMBA APB4 protocol, as the *AMBA® APB Protocol Specification* defines.

2.1.6 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 33

2.1.7 LPI Q-Channel protocol

The MMU-700 complies with the LPI Q-Channel, as the AMBA® Low-Power Interface Specification defines.

Related information

TCU LPI_PD interface signals on page 197
TCU LPI_CG interface signals on page 198
TBU LPI_PD interface signals on page 220
TBU LPI_CG interface signals on page 221

2.2 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 the TBU and the TCU over the AXI5-Stream protocol (with Wakeup_Signal enabled and Check_Type not enabled) is supported using the supplied DTI interconnect components, or any other AXI5-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

2.3 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 26

2.4 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

---

Note

Depths are specified as a discrete number of entries.

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

See 3.4 Configuration parameters and methodology on page 80.

Related information

Configuration parameters and methodology on page 80

2.5 Product documentation and design flow

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

2.5.1 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.

2.5.2 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 20
Configuration parameters and methodology on page 80
Compliance on page 15

2.6 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 4.8 TCU system discovery registers on page 130 and 4.15 TBU system discovery registers on page 173.

- New parameters. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81 and 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.
- New stitching flow
- New generate executable

r0p1-r1p0

The following changes apply to this release:

- New multiple LTI interface TBU. See 3.1.2.3 LTI TBU LTI interface on page 33.
- New integration TBU. See 3.1.2.10 Integration TBU on page 35.
- PCIe CXL.IO support
- Performance improvement by changing the configuration of some RAMs
- Changes to registers. See 4. Programmers model for MMU-700 on page 94.
- Changes to parameters. See:
  - 3.4.2 Translation Control Unit buffer configuration parameters on page 81
  - 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85

r1p0-r1p1

The following changes apply to this release:

- Multiple errata fixes. See the following documents:
  - *MMU-700 System Memory Management Unit Release Note*
  - *MMU-700 System Memory Management Unit Product Errata Notice (PEN)*
  - *MMU-700 System Memory Management Unit Software Developer Errata Notice (SDEN)*
  - *MMU-700 System Memory Management Unit Product Advice Notice (PAN)*

r1p1-r1p2

The following changes apply to this release:

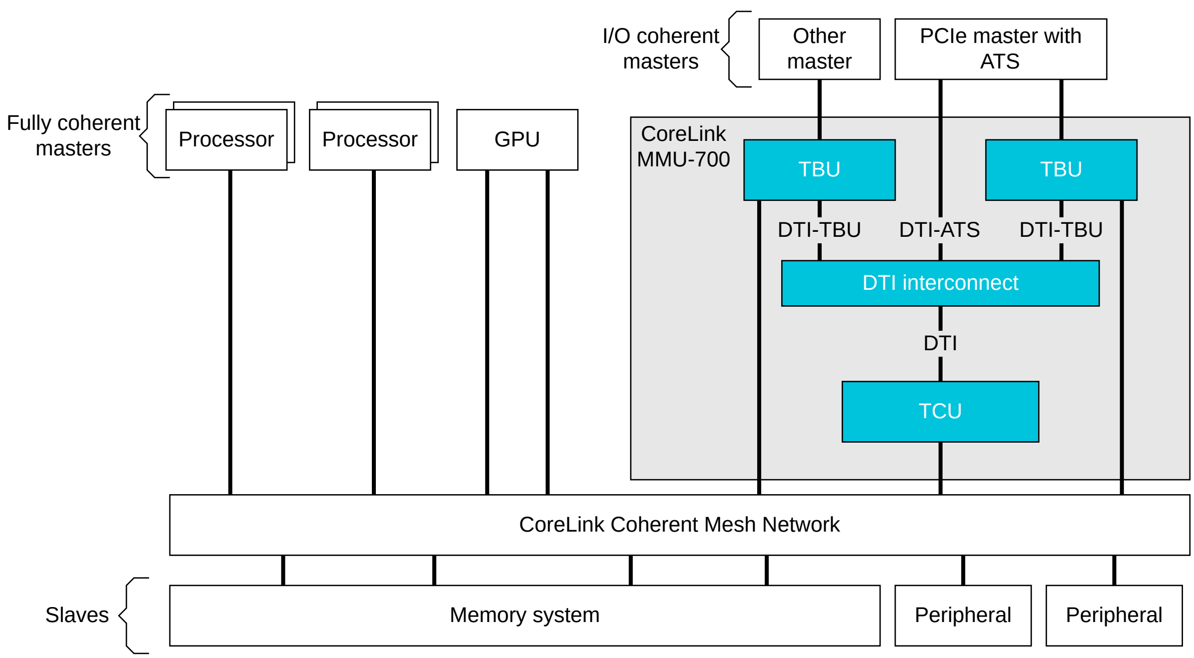
- Multiple errata fixes. See the following documents:
  - *MMU-700 System Memory Management Unit Release Note*
  - *MMU-700 System Memory Management Unit Product Errata Notice (PEN)*
  - *MMU-700 System Memory Management Unit Software Developer Errata Notice (SDEN)*
  - *MMU-700 System Memory Management Unit Product Advice Notice (PAN)*

3. Functional description of MMU-700

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

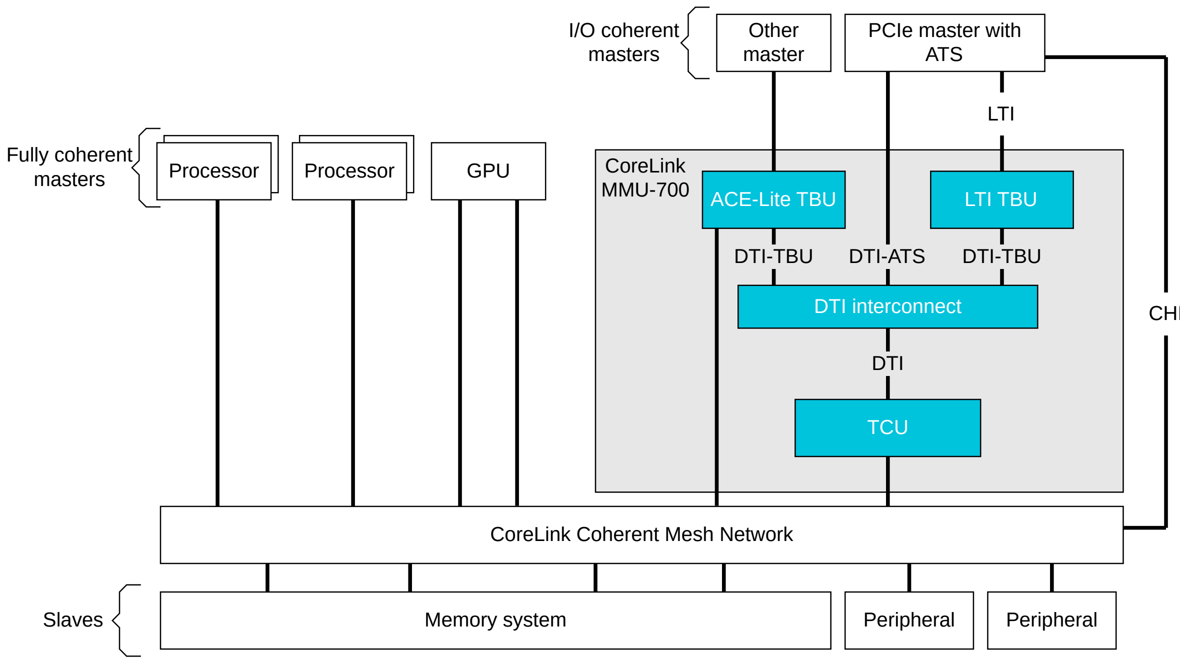
The following figure shows an example system that uses the MMU-700.

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 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.

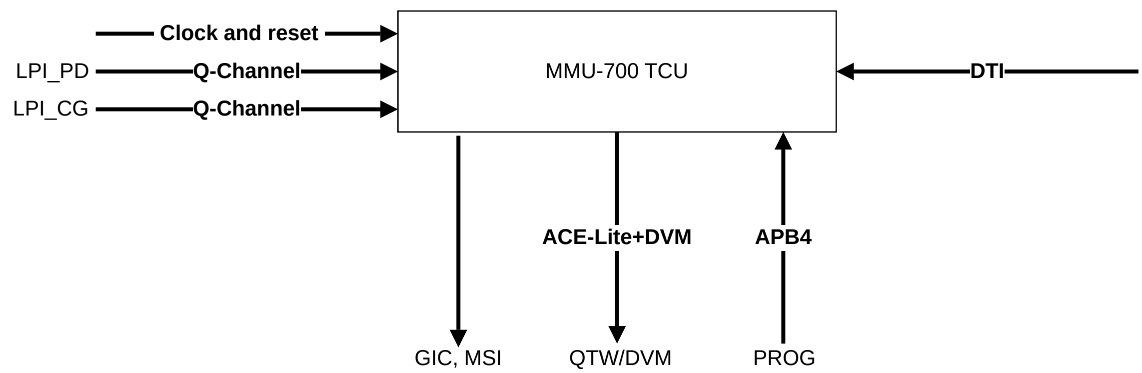
The PMU snapshot interface is common to both TCU and TBU.

3.1.1 TCU interfaces

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

The following figure shows the TCU interfaces.

Figure 3-3: TCU interfaces



Related information

Distributed Virtual Memory messages on page 52
Error responses on page 56
AMBA implementation on page 64
TCU QTW/DVM interface signals on page 193

3.1.1.1 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` = MAX(4, ceil($\log_2$(`TCUCFG_PTW_SLOTS`)) + 2).

The smallest possible `TCU_ID_WIDTH` value is 4.

See 3.4 Configuration parameters and methodology on page 80.

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 `TCU_ID_WIDTH` > 4, bits {`TCU_ID_WIDTH` - 1 :4} are 0.	`2'b00`
DVM Complete	Bits [3:2] = `2'b01`. If `TCU_ID_WIDTH` > 4, bits {`TCU_ID_WIDTH` - 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 `TCU_ID_WIDTH` > 4, bits {`TCU_ID_WIDTH` - 1:4} are 0.	`2'b00`
Event Queue write	Bits [3:2] = `2'b10`. If `TCU_ID_WIDTH` > 4, bits {`TCU_ID_WIDTH` - 1:4} are 0.	`2'b00`
MSI write	Bits [3:2] = `2'b11`. If `TCU_ID_WIDTH` > 4, bits {`TCU_ID_WIDTH` - 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.

3.1.1.2 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 196

3.1.1.3 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 197

3.1.1.4 TCU LPI_CG interface

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

Related information

TCU LPI_CG interface signals on page 198

3.1.1.5 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 AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled).

The TCU includes a slave DTI interface and each TBU includes a master DTI interface. To permit bidirectional communication, each DTI interface includes:

Master interface

One AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled) master interface

Slave interface

One AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled) slave interface

For more information, see the *AMBA® DTI Protocol Specification* and the *AMBA® AXI-Stream Protocol Specification*.

Related information

DTI overview on page 40
TCU DTI interface signals on page 198

3.1.1.6 TCU Message Signaled Interrupt interface

The *Message Signaled Interrupt* (MSI) interface provides global, per-context, and performance interrupts. A direct MSI connection to a *Generic Interrupt Controller* (GIC) is supported, to avoid complex dependencies in the system. The MSI interface is implemented using AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled).

Related information

TCU Message Signaled Interrupt interface signals on page 201

3.1.1.7 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 205

3.1.1.8 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 204

3.1.1.9 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 3.4.3 Translation Control Unit debug configuration parameters on page 83.

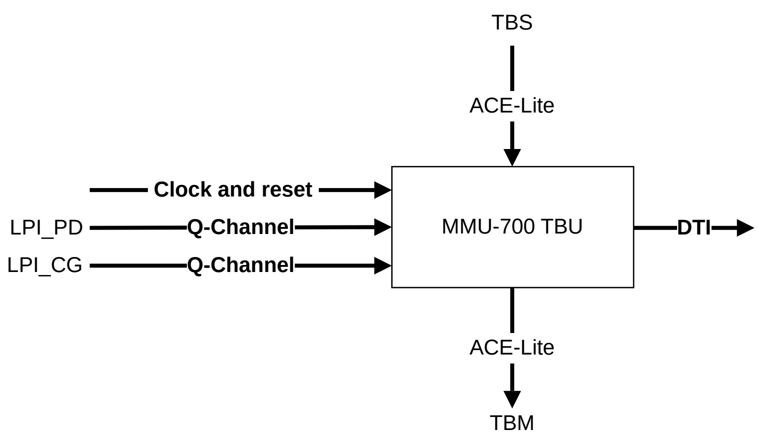
For more information about the interface signals, see B.1 TCU observation interfaces on page 239.

3.1.2 TBU interfaces

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

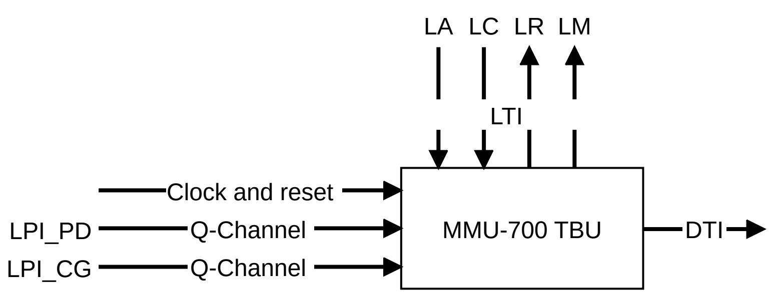
The following figure shows the ACE-Lite TBU interfaces.

Figure 3-4: ACE-Lite TBU interfaces



The following figure shows the LTI TBU interfaces.

Figure 3-5: LTI TBU interfaces



Note

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

3.1.2.1 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 3.3.2 AMBA implementation on page 64.

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 56
TBU TBS interface signals on page 206

3.1.2.2 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:

- 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88
- 3.4.9 Local Translation Interface Translation Buffer Unit configuration parameters on page 91

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 v2[1]
- Cache_Stash_Transactions
- DeAllocation_Transactions
- Atomic_Transactions
- Loopback_Signals
- Ordered Write Observation
- Poison
- Unique_ID_Support
- Read_Data_Chunking
- CMO_On_Read, Persist_CMO
- MPAM_Support

For more information, see 3.3.2 AMBA implementation on page 64.

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 56
TBU TBM interface signals on page 213
AMBA implementation on page 64

3.1.2.3 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.

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

LR
Response channel. Provides the translated address and attributes to the LTI device.

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:

- 3.3.4 Local Translation Interface implementation on page 79
- 3.4 Configuration parameters and methodology on page 80
- *AMBA® LTI Protocol Specification*

3.1.2.4 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 220

3.1.2.5 TBU LPI_CG interface

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

Related information

TBU LPI_CG interface signals on page 221

3.1.2.6 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:

Master interface

One AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled) master interface

Slave interface

One AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled) slave interface

For more information, see the *AMBA® DTI Protocol Specification* and the *AMBA® AXI-Stream Protocol Specification*.

Related information

DTI overview on page 40
TBU DTI interface signals on page 221

3.1.2.7 TBU interrupt interfaces

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

Related information

TBU tie-off signals on page 35
TBU interrupt signals on page 228

3.1.2.8 TBU tie-off signals

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

3.1.2.9 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 3.4.10 Common Translation Buffer Unit debug configuration parameters on page 92.

For more information about the interface signals, see:

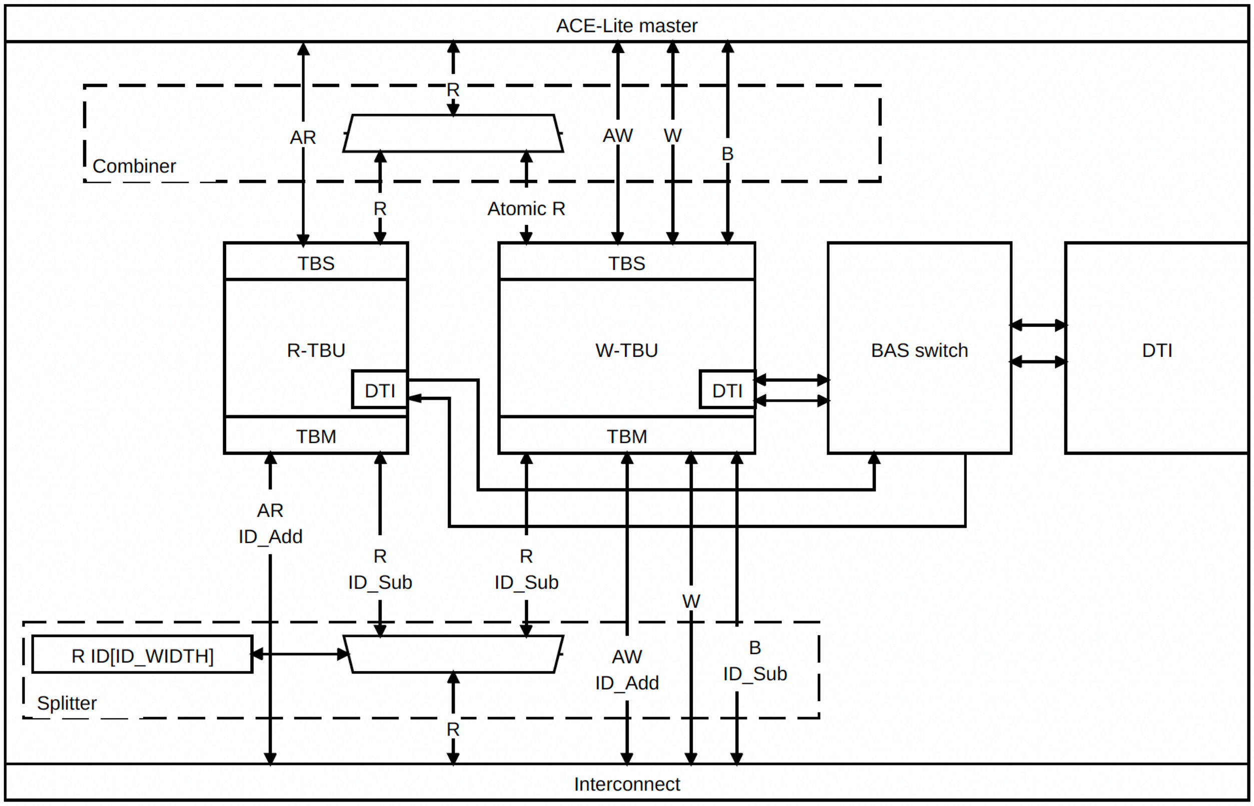
- B.2 ACE-Lite TBU observation interfaces on page 242
- B.3 LTI TBU observation interfaces on page 244

3.1.2.10 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.



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.

3.1.2.10.1 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.

3.1.2.10.2 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.

3.1.2.10.3 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.

3.1.3 DTI interconnect interfaces

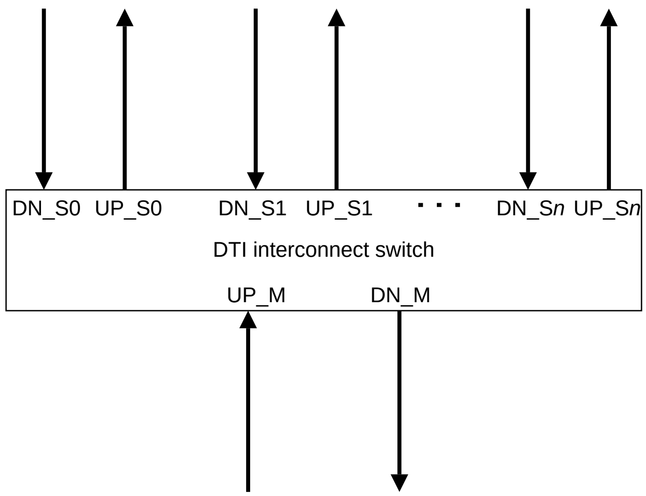
The DTI interconnect includes interfaces for each of the switch, sizer, and register slice components.

3.1.3.1 DTI interconnect switch interfaces

The DTI interconnect switch component includes dedicated interfaces.

The following figure shows the DTI interconnect switch interfaces.

Figure 3-6: DTI interconnect switch interfaces



The following table provides more information about the switch interfaces.

Table 3-3: DTI interconnect switch interfaces

Interface	Interface type	Protocol	Description
DN_Sn	Slave	AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled)	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

Note

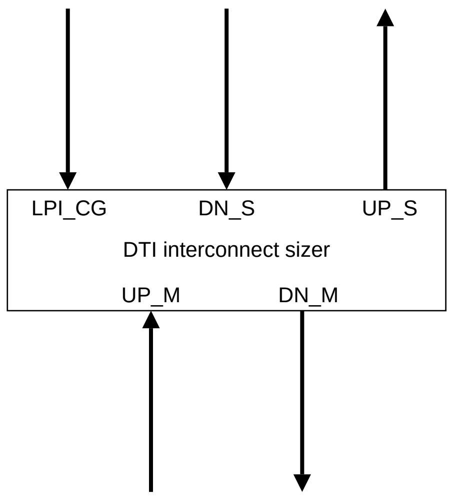
The interconnect switch does not store any data, and therefore does not require a Q-Channel clock-gating interface.

3.1.3.2 DTI interconnect sizer interfaces

The DTI interconnect sizer component includes dedicated interfaces.

The following figure shows the DTI interconnect sizer interfaces.

Figure 3-7: DTI interconnect sizer interfaces



The following table provides more information about the sizer interfaces.

Table 3-4: DTI interconnect sizer interfaces

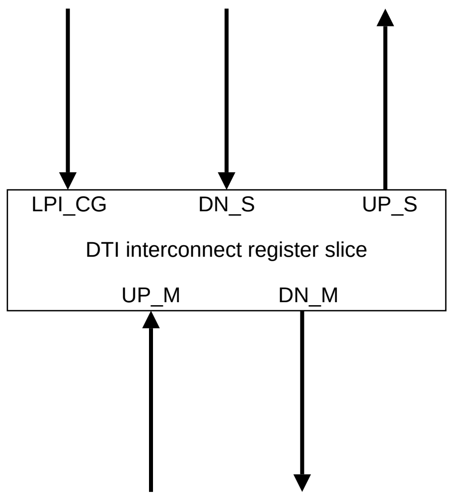
Interface	Interface type	Protocol	Description
LPI_CG	Slave	Q-Channel	Clock gating interface
DN_S	Slave	AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled)	Slave downstream interface
UP_S	Master		Slave upstream interface
DN_M	Master		Master downstream interface
UP_M	Slave		Master upstream interface

3.1.3.3 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 3-8: DTI interconnect register slice interfaces



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

Table 3-5: DTI interconnect register slice interfaces

Interface	Interface type	Protocol	Description
LPI_CG	Slave	Q-Channel	Clock gating interface
DN_S	Slave	AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled)	Slave downstream interface
UP_S	Master		Slave upstream interface
DN_M	Master		Master downstream interface
UP_M	Slave		Master upstream interface

3.2 Operation

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

3.2.1 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

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.

The TBU uses the TOK_INV_GNT field to grant invalidation tokens. The TBU grants only one invalidation token, and the TCU can only issue one invalidate message at a time.

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 TOK_TRANS_REQ field of this message specifies the number of requested translation tokens. For the TBU, the max_tok_trans signal defines the number of translation tokens that the TBU requests. The TBU must request a minimum of two translation tokens per LTI port.

A translation request to the TCU where StreamID $\geq 2^{32}$ results in a fault and an SMMUv3 C_BAD_STREAMID event. If the TBU receives an invalidation request where StreamID $\geq 2^{32}$, 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.

Note

- The TBU never generates translation requests with StreamID $\geq 2^{32}$
- The TCU never generates invalidation requests with StreamID $\geq 2^{32}$

For more information, see the *AMBA® DTI Protocol Specification*.

3.2.2 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.

3.2.2.1 SMMUv3 architectural performance events

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

The SMMU_PMCG_SMR0 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 3-6: SMMUv3 performance events for the TCU

Event	Event ID	SMMU_PMCG_SMR0 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 3-7: SMMUv3 performance events for the TBU

Event	Event ID	SMMU_PMCG_SMR0 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 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*.

3.2.2.2 MMU-700 TCU events

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

The SMMU_PMCG_SMR0 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 3-8: MMU-700 TCU performance events

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
S1L0WC lookup	`0x80`	Yes	Counts translation requests that access the S1L0WC walk cache
S1L0WC miss	`0x81`	Yes	Counts translation requests that access the S1L0WC 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
S2L0WC lookup	`0x88`	Yes	Counts translation requests that access the S2L0WC walk cache
S2L0WC miss	`0x89`	Yes	Counts translation requests that access the S2L0WC 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_PMCG_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

3.2.2.3 MMU-700 TBU events

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

The SMMU_PMCG_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 3-9: 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
Translation 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: • 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	For the ACE-Lite TBU, counts when either: • 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 For the LTI TBU, counts once whenever either an: • 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: ◦ Memory attributes or DCP, R, W, or X permission check failure in the *Translation Lookaside Buffer Unit* (TLBU) ◦ DTI fault response with Non-Abort The transaction that is responded to with FaultAbort because of DTI StreamDisable or GlobalDisable is not counted **Note:** A StashOnceShared or StashOnceUnique transaction that is terminated because of a StreamDisable or GlobalDisable translation response does not cause this event to count
Fixed Burst Termination	`0x8D`	No	For the ACE-Lite TBU, counts when the TBU issues an abort response for a fixed burst, the domain of which is determined to be shareable, post translation. For the LTI TBU, this event does not apply.
LTI port slots full	`0xD0` - `0xD7`	No	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.
LTI port out of translation tokens	`0xE0` - `0xE7`	No	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.

3.2.2.4 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 3-10: 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 *Message Signaled Interrupts* (MSIs)
RELOC_CTRS	1	The PMCG registers are relocated to page 1 of the PMU address map

Field		Default value	Description for default value
SIZE		0x31	The counter group implements 32-bit counters
MPAM		0	*Memory System Resource Partitioning and Monitoring* (MPAM)
NCTR	NCTR for the TCU	TCUCFG_PMU_ COUNTERS - 1	The counter group includes TCUCFG_PMU_COUNTERS counters. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.
	TCTR for the TBU	TBUCFG_PMU_ COUNTERS - 1	The counter group includes TBUCFG_PMU_COUNTERS counters. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Related information

MMU-700 memory map on page 99

3.2.2.5 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 49
TCU PMU snapshot interface signals on page 197
TBU PMU snapshot interface signals on page 220

3.2.3 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 for MMU-700 on page 94.

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.

3.2.4 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.

Note

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 244.

Note

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

Table 3-11: 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`) - $\log_2$(`TBUCFG_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 4.6.1 TCU_CTRL register on page 111.

3.2.5 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 RAMs support RAS error detection and correction.

MMU-700 implements a combination of *Single-Error-Correct-Double-Error-Detect* (SECDED) 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 3-12: 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

3.2.6 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 4.13.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 160. 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 116
TCU_QOS register on page 113

3.2.7 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

Note

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

Note

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 95

3.2.8 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 3-13: 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`

Transaction type	Transaction width	ARID[n:2]	ARID[1:0]
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 outstanding read transactions that the TCU can issue at any time is (`TCUCFG_PTW_SLOTS` + `TCUCFG_CTW_SLOTS` + 2).

The following table shows the TCU support for write transactions.

Table 3-14: 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.

3.2.9 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. 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.

Note

The walk cache uses a round-robin *Re-Reference Interval Prediction* (RRIP) 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

Note

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

3.2.10 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.

3.2.11 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 for A-profile architecture. 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.

3.2.11.1 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 3-15: MMU-700 ACE-Lite to Armv8 memory attribute conversions

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

Note

- 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

3.2.11.2 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 3-16: MMU-700 Armv8 to ACE-Lite memory attribute conversions

Armv8 memory type	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-nGRE Device-nGnRE	Device Bufferable	System	As TBS AxLOCK value	0
Normal Inner Non-cacheable Outer Non-cacheable Normal Inner Write-Through Outer Non-cacheable Normal Inner Write-Back Outer Non-cacheable	Normal Non-cacheable Bufferable	System	As TBS AxLOCK value	0

Armv8 memory type	AxCACHE attribute	AxDOMAIN attribute	AxLOCK attribute	AxUSER Outer Cacheable
Normal Inner Non-cacheable Outer Write-Through 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	Normal Non-cacheable Bufferable	System	As TBS AxLOCK value	1
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: • Non-shareable • Outer Shareable An Armv8 shareability attribute of Inner Shareable is always output with an AxDOMAIN value of Outer Shareable because Inner Shareable is deprecated in ACE and ACE-Lite.	0	1

3.2.12 AXI USER bits that MMU-700 TBU TBM and TCU QTW/DVM define

The TBU TBM interface AxUSER signals have more bits than the `TBUCFG_AxUSER_WIDTH` parameters define. The TCU QTW/DVM interface AxUSER signals are 4 bits wide.

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

For more information, see the *Calculating Page Based Hardware Attribute (PBHA) in SMMUs* section in the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

The following table shows the MMU-700-defined TBU aruser_m and awuser_m bits, where `w` represents the AXI USER bus width that `TBUCFG_AxUSER_WIDTH` defines. See 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.

Table 3-17: MMU-700-defined TBU aruser_m and awuser_m bits

Bit position	Value
[*w*+4]	Outer Cacheable
[*w*+3:*w*]	The **IMPLEMENTATION DEFINED** PBHA bits

The TCU QTW/DVM interface AxUSER signals, aruser_qtw and awuser_qtw, are 4 bits wide. These bits provide extra attributes for SMMU-originated accesses.

The following table shows the MMU-700-defined TCU aruser_qtw and awuser_qtw bits.

Table 3-18: MMU-700-defined TCU aruser_qtw and awuser_qtw bits

Bit position	Value
AxUSER[3:0]	**IMPLEMENTATION DEFINED** PBHA bits

3.2.12.1 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 for A-profile architecture*
- *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*.

3.3 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

3.3.1 SMMUv3 implementation

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

Related information

SMMU architectural registers on page 95

3.3.1.1 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 3-19: 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	*Message Signaled Interrupts* (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
	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_CR0.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

Register	Field	Value	Description
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	EL0/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	0h43B	Arm® implementation
	Revision	MAX(*p_level*, ecorevnum)	Where *p_level* is 2 for p2
	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
SMMU_S_IDR0	MSI	1	Secure MSIs are supported
	STALL_MODEL	0'b00	Stall and terminate model is supported
SMMU_S_IDR1	S_SIDSIZE	32	32-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

Register	Field	Value	Description
SMMU_MPAMIDR	PMG_MAX	8'h01	Maximum PMG value that is permitted to be used in Non-secure state
	PARTID_MAX	2^ TCUCFG_PARTID_WIDTH - 1	Maximum PARTID value that is permitted to be used in Non-secure state
SMMU_S_MPAMIDR	HAS_MPAM_NS	0, not implemented	**0b0** The MPAM_NS mechanism for Secure state is not implemented **0b1** The MPAM_NS mechanism for Secure state is implemented
	PMG_MAX	8'h01	Maximum PMG value that is permitted to be used in Non-secure state
	PARTID_MAX	2^ TCUCFG_PARTID_WIDTH - 1	Maximum PARTID value that is permitted to be used in Non-secure state.

3.3.1.2 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`.

3.3.1.3 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

3.3.1.4 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

3.3.2 AMBA implementation

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

3.3.2.1 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 3-20: 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
	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	v2	N[2]	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	N
	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
	Write_Plus_CMO	N	N	N
	WriteZero_Transaction	N	N	N

[2] The TBM interface does not support the *Untranslated_Transactions* property. The TBM contains the axmmusecsid and axmmusid signals for backward-compatibility with other SMMU products. Normal operation of the MMU-700 does not require these signals and you can ignore them.

3.3.2.2 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.

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.

3.3.2.3 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*.

Note

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 229 describes.

3.3.2.3.1 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 3-21: 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.

3.3.2.3.2 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 3-22: 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

Armv8 Memory Type	AxCACHE (TBM, QTW)	AxDOMAIN (TBM, QTW)	AxLOCK (TBM)	AxUSER outer cacheable bit (TBM)
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 3-23: 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`

3.3.2.4 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

3.3.2.5 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

3.3.2.6 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.

3.3.2.7 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

3.3.2.8 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.

3.3.2.9 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

Note

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

Note

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.

Note

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.

3.3.3 MPAM implementation

CoreLink™ MMU-700 System Memory Management Unit implements *Memory System Resource Partitioning and Monitoring* (MPAM) with some constraints and limitations. Certain MPAM registers are implemented. Registers that are not implemented are not described.

MMU-700 uses some MPAM architectural options from the *Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A*.

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.

3.3.3.1 TCU MPAM

The CoreLink™ MMU-700 System Memory Management Unit enables you to implement TCU *Memory System Resource Partitioning and Monitoring* (MPAM). Internally, *Resource Instance Selection* (RIS) is truncated to 1 bit, but externally it is the normal 4 bits.

RIS 0 *Walk Cache Block* (WCB)
RIS 1 *Configuration Cache Block* (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 3-24: TCU MPAM registers implemented

Register	Field	Value when resource is present	Value when resource is not present	Description
MPAMF_IDR_LO (0x0000, Shared)	PARTID_MAX	1, 63, 511	1, 63, 511	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
	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	1	0	Maximum RIS value used in the MSC: Set to 1
MPAMF_SIDR (0x0008, S-Only)	S_PARTID_MAX	1, 63, 511	1, 63, 511	Set to ($2^{TCUCFG\_PARTID\_WIDTH}$ - 1)
	S_PMG_MAX	1	1	Two Secure Performance Monitoring groups supported per PARTID
MPAMF_MSMON_IDR (0x0080, Shared)	MSMON_CSU	1	RAZ/WI	Performance monitor supported for Cache Storage Usage by PARTID and PMG
	MSMON_MBWU	0	RAZ/WI	No performance monitor for Memory Bandwidth Usage by PARTID and PMG
	HAS_LOCAL_CAPT_EVNT	1	RAZ/WI	Has the local capture event generator and the MSMON_CAPT_EVNT register

Register	Field	Value when resource is present	Value when resource is not present	Description
MPAMF_CCAP_IDR (`0x0038`, Shared)	CMAX_WD	`8`	RAZ/WI	256 fractions are supported
MPAMF_CSUMON_IDR (`0x0088`, Shared)	NUM_MON	`4`	RAZ/WI	Four monitoring counters are implemented
	HAS_CAPTURE	`1`	RAZ/WI	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	All fields valid	Implementation ID Register
MPAMF_AIDR (`0x0020`, Shared)	ArchMajorRev ArchMinorRev	`0x1` `0x1`	`0x1` `0x1`	MPAM architecture v1.1
MPAMCFG_PART_SEL (`0x0100`, Banked)	PARTID_SEL	Bits [(`TCUCFG_PARTID_WIDTH` - 1):0] valid	RAZ/WI	Can select up to 512 partitions to configure, based on `TCUCFG_PARTID_WIDTH`.
	RIS	Bits [27:24] valid	RAZ/WI	Resource Instance Selector.
MPAMCFG_CMAX (`0x0108`, Banked)	CMAX	Bits [15:8] valid	RAZ/WI	Can choose up to 256 fractions.
MSMON_CFG_MON_SEL (`0x0800`, Banked)	MON_SEL	Bits [1:0] valid	RAZ/WI	Selects the monitor to configure.
	RIS	Bits [27:24] valid	RAZ/WI	Resource Instance Selector.
MSMON_CFG_CSU_FLT (`0x0810`, Banked)	PARTID	Bits [(`TCUCFG_PARTID_WIDTH` - 1):0] valid	RAZ/WI	Can select up to 512 partitions to configure, based on `TCUCFG_PARTID_WIDTH`.
	PMG	`0`	RAZ/WI	Can select up to PMG number 1.
MSMON_CFG_CSU_CTL (`0x0818`, Banked)	EN	Valid field	RAZ/WI	The monitor instance is enabled or disabled to collect information.
	CAPT_EVNT	`3'b111`	RAZ/WI	Capture occurs when a MSMON_CAPT_EVNT register is written. All other values are not supported.
	CAPT_RESET	RES0	RAZ/WI	There is no reason to ever reset a CSU monitor.
	OFLOW_STATUS	RES0	RAZ/WI	Overflow is not possible for a CSU monitor.
	OFLOW_INTR	RES0	RAZ/WI	This MPAM implementation does not support OFLOW_INTR.
	OFLOW_FRZ	RES0	RAZ/WI	Overflow is not possible for a CSU monitor.
	SUBTYPE	RES0	RAZ/WI	This field is reserved for future use.
MSMON_CSU (`0x0840`, Banked)	All fields	All fields valid	RAZ/WI	Cache storage usage value.
MSMON_CSU_CAPTURE (`0x0848`, Banked)	All fields	All fields valid	RAZ/WI	Capture cache storage usage.

Register	Field	Value when resource is present	Value when resource is not present	Description
MSMON_CAPT_EVNT (`0x0808`, Banked)	All fields	All fields valid	RAZ/WI	Capture event.

3.3.3.2 TBU MPAM

The CoreLink™ MMU-700 System Memory Management Unit enables you to implement TBU *Memory System Resource Partitioning and Monitoring* (MPAM).

- When `TBUCFG_MTLB_DEPTH == 0`, the resource is not present
- When `TBUCFG_DIRECT_IDX == 1`, the resource is present but does not have 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.

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

Table 3-25: TBU MPAM registers implemented

Register	Field	Value when resource is present	Value when resource is not present	Description
MPAMF_IDR_LO (`0x0000`, Shared)	PARTID_MAX	`1, 63, 511`	`1, 63, 511`	Set to: $(2^{\texttt{TBUCFG\_PARTID\_WIDTH}} - 1)$
	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
	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	0	0	Has Resource Instance Selector
	NO_IMPL_PART	1	0	There are no **IMPLEMENTATION DEFINED** resource controls that MPAMF_IMPL_IDR defines

Register	Field	Value when resource is present	Value when resource is not present	Description
	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	0	Maximum RIS value used in the MSC.
MPAMF_SIDR (0x0008, S-Only)	S_PARTID_MAX	1, 63, 511	1, 63, 511	Set to $(2^{\texttt{TBUCFG\_PARTID\_WIDTH}} - 1)$
	S_PMG_MAX	1	1	Two Secure Performance Monitoring groups supported per PARTID
MPAMF_MSMON_IDR (0x0080, Shared)	MSMON_CSU	1	RAZ/WI	Performance monitor supported for Cache Storage Usage by PARTID and PMG
	MSMON_MBWU	0	RAZ/WI	No performance monitor for Memory Bandwidth Usage by PARTID and PMG
	HAS_LOCAL_CAPT_EVNT	1	RAZ/WI	Has the local capture event generator and the MSMON_CAPT_EVNT register
MPAMF_CCAP_IDR (0x0038, Shared)	CMAX_WD	8	RAZ/WI	256 fractions are supported
MPAMF_CSUMON_IDR (0x0088, Shared)	NUM_MON	4	RAZ/WI	Four monitoring counters are implemented
	HAS_CAPTURE	1	RAZ/WI	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	All fields valid	Implementation ID Register
MPAMF_AIDR (0x0020, Shared)	ArchMajorRev ArchMinorRev	0x1 0x10	0x1 0x1	MPAM architecture v1.1
MPAMCFG_PART_SEL (0x0100, Banked)	PARTID_SEL	Bits [(TBUCFG_ PARTID_WIDTH - 1):0] valid	RAZ/WI	Can select up to 512 partitions to configure, based on TBUCFG_PARTID_WIDTH. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
	RIS	0	0	Resource Instance Selector. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.

Register	Field	Value when resource is present	Value when resource is not present	Description
MPAMCFG_CMAX (0x0108, Banked)	CMAX	Bits [15:8] valid	RAZ/WI	Can choose up to 256 fractions. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_MON_SEL (0x0800, Banked)	MON_SEL	Bits [1:0] valid	RAZ/WI	Selects the monitor to configure. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
	RIS	0	0	Resource Instance Selector. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_CSU_FLT (0x0810, Banked)	PARTID	Bits [(TBUCFG_PARTID_WIDTH - 1):0] valid	RAZ/WI	Can select up to 512 partitions to configure, based on TBUCFG_PARTID_WIDTH. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
	PMG	0	RAZ/WI	Can select up to PMG number 1. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (0x0818, Banked)	EN	Valid field	RAZ/WI	The monitor instance is enabled or disabled to collect information. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
	CAPT_EVNT	3'b111	RAZ/WI	Capture occurs when a MSMON_CAPT_EVNT register is written. All other values are not supported. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
	CAPT_RESET	RES0	RAZ/WI	There is no reason to ever reset a CSU monitor. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.

Register	Field	Value when resource is present	Value when resource is not present	Description
	OFLOW_STATUS	RES0	RAZ/WI	Overflow is not possible for a CSU monitor. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
	OFLOW_INTR	RES0	RAZ/WI	This MPAM implementation does not support OFLOW_INTR. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
	OFLOW_FRZ	RES0	RAZ/WI	Overflow is not possible for a CSU monitor. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
	SUBTYPE	RES0	RAZ/WI	This field is reserved for future use. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CSU (`0x0840`, Banked)	All fields	All fields valid	RAZ/WI	Cache storage usage value. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CSU_CAPTURE (`0x0848`, Banked)	All fields	All fields valid	RAZ/WI	Capture cache storage usage. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CAPT_EVNT (`0x0808`, Banked)	All fields	All fields valid	RAZ/WI	Capture event. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.

3.3.4 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 3-26: LTI properties

Name	Value	Description
`LTI_VC_COUNT`	2	Two LTI Virtual channels are chosen, one for read and one for write

Name	Value	Description
LTI_ID_WIDTH	TBUCFG_ID_WIDTH	Equal to ID width of incoming transaction
LTI_SID_WIDTH	TBUCFG_SID_WIDTH	Equal to width of incoming SID
LTI_OG_WIDTH	TBUCFG_LTI_OG_WIDTH	Equal to width of incoming ordering groups
LTI_TLBLOC_WIDTH	-[3]	Width of TLB location, in bits
LTI_LOOP_WIDTH	-[4]	Width of the LTI loopback signals, in bits
LTI_LAUSER_WIDTH	0	LTI user signals are not used
LTI_LRUSER_WIDTH	0	
LTI_LCUSER_WIDTH	0	

3.4 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*.

3.4.1 Translation Control Unit I/O configuration parameters

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

Tip

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 3-27: TCU I/O configuration parameter

Interface and module name	Parameter name	Values	Description
QTW	TCUCFG_ QTW_DATA_WIDTH	64, 128, 256, 512	ACE-Lite_DVM interface data width.

[3] The value of the LTI_TLBLOC_WIDTH parameter, which is a local parameter, is as follows:

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

[4] For the LTI TBU, the value is equal to TBUCFG_LTI_LOOP_WIDTH.

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

Interface and module name	Parameter name	Values	Description
DVM	`TCUCFG_DVM_VAS`	`49, 53`	Virtual Address Size that the system uses. The SMMU uses `TCUCFG_DVM_VAS` to perform address-based invalidations correctly.

3.4.2 Translation Control Unit buffer configuration parameters

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

Tip

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 3-28: TCU buffer configuration parameters

Parameter name	Values	Description
`TCUCFG_CC_DEPTH`	`4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096`	Configuration cache depth, in entries
`TCUCFG_WC_DEPTH`	`8, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536`	Walk cache depth, in entries **Note:** (`TCUCFG_WC_DEPTH/TCUCFG_WC_BANKS`)/`TCUCFG_WC_WAYS`) must be > 1
`TCUCFG_WC_BANKS`	`1, 2, 4`	Number of banks in Walk Cache **Note:** (`TCUCFG_WC_DEPTH/TCUCFG_WC_BANKS`)/`TCUCFG_WC_WAYS`) must be > 1
`TCUCFG_WC_WAYS`	`4, 8, 16`	Number of ways in walk cache **Note:** (`TCUCFG_WC_DEPTH/TCUCFG_WC_BANKS`)/`TCUCFG_WC_WAYS`) must be > 1
`TCUCFG_NUM_TBU`	`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.

Parameter name	Values	Description
TCUCFG_XLATE_SLOTS	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 TCUCFG_PTW_SLOTS.
TCUCFG_PTW_SLOTS	2, 4, 8, 16, 32, 64, 128, 256, 512	Number of parallel translation table walks
TCUCFG_CTW_SLOTS	1, 2, 4. **Note:** This value must not be greater than TCUCFG_PTW_SLOTS.	Number of parallel configuration table walks
TCUCFG_WC_LKP_SLOTS	2-28	Walk Cache Lookup slots. The number of lookup slots that the walk cache uses. 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. 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. Make sure that the value is not greater than TCUCFG_PTW_SLOTS. You can: • Increase the value of TCUCFG_WC_LKP_SLOTS to improve performance, but with greater area • Decrease the value of TCUCFG_WC_LKP_SLOTS to save area but with reduced performance
TCUCFG_CC_IDXGEN_MODE	0, 1	Index generation mode for the configuration cache: **0** Polynomial. Polynomial is the recommended setting for most systems **1** Simple
TCUCFG_DTI_ATS	0, 1, 2, 3, 4, 5, 6, 7, 8	Number of DTI-ATS masters. **Note:** TCUCFG_NUM_TBU is the total number of DTI-TBU and DTI-ATS masters. TCUCFG_DTI_ATS is the total number of DTI-ATS masters.
TCUCFG_PMU_COUNTERS	4, 16, 32	Number of PMU counters
TCUCFG_PARTID_WIDTH	1, 6, 9	Width of PARTID that is supported: **1** When set to 1, PARTID[8:1] sent on the DTI interface is set to 0 **6** When set to 6, PARTID[8:6] sent on the DTI interface is set to 0
TCUCFG_HZU_DEPTH	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.
TCUCFG_PREFETCH_SUPPORTED	0, 1	Specifies whether prefetch is supported

Parameter name	Values	Description
`TCUCFG_DATARAM_TYPE`	`0, 1, 2`	RAM type for data group of RAMs: **0** Two ports, that is, one port is for reads and one port is 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 you set `TCUCFG_DATARAM_TYPE` 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.
`TCUCFG_SLOTRAM_TYPE`	`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*.
`TCUCFG_CACHERAM_TYPE`	`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

3.4.3 Translation Control Unit debug configuration parameters

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

Tip

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 3-29: TCU debug configuration parameters

Parameter name	Values	Description
`TCUCFG_USE_ELA_DEBUG`	`0, 1`	Set the `TCUCFG_USE_ELA_DEBUG` 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 239.

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

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

Tip

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 ACE-Lite TBU and LTI TBU configuration parameters.

Table 3-30: Common ACE-Lite TBU and LTI TBU configuration parameters

Parameter name	Values	Description
`TBUCFG_SID_WIDTH`	`8, 16, 20, 24`	Stream ID width
`TBUCFG_SSID_WIDTH`	`1, 8, 20`	SubstreamID width
`TBUCFG_DIRECT_IDX`	`0, 1`	Direct indexing. **Note:** Must be 0 if `TBUCFG_MTLB_DEPTH` = `0`.
`TBUCFG_MTLB_PARTS`	`1, 2, 4, 8, 16`	Number of main TLB partitions. **Note:** Must be `1` if `TBUCFG_MTLB_DEPTH` = `0`. Must be `1` if `TBUCFG_DIRECT_IDX` = `1`. `TBUCFG_MTLB_PARTS` × `TBUCFG_MTLB_DEPTH` must not exceed `65536`.
`TBUCFG_LTI_OG_WIDTH`	`1-5`	LTI ordering groups width. Number of ordering groups = `2^ TBUCFG_LTI_OG_WIDTH`. **Note:** The top-level block being used implicitly sets the number of LTI ports. For information about the legal combinations of the Number of LTI Ports, and the values of the TBUCFG_LTI_OG_WIDTH and TBUCFG_SLOTRAM_TYPE parameters, see the next table, named *Legal combinations of Number of LTI Ports,* `TBUCFG_LTI_OG_WIDTH` *and* `TBUCFG_SLOTRAM_TYPE` *parameters*.

Parameter name	Values	Description
`TBUCFG_LA_HNDSHK_MODE`	0-3	Handshake mode on address channel before TLB lookup. Supported values are as follows: **0 or 1** FWD. Registered on the forward path only, that is, the direction that LAVALID on the corresponding interface indicates. **2 or 3** BP. Bypass register slice, not registered.
`TBUCFG_LR_HNDSHK_MODE`	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 `TBUCFG_LR_HNDSHK_MODE` 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

The following table shows the legal combinations of the values for the Number of LTI Ports, and the `TBUCFG_LTI_OG_WIDTH` and `TBUCFG_SLOTRAM_TYPE` parameters.

Table 3-31: Legal combinations of Number of LTI Ports, `TBUCFG_LTI_OG_WIDTH` and `TBUCFG_SLOTRAM_TYPE` parameters

Number of LTI Ports	TBUCFG_SLOTRAM_TYPE	TBUCFG_LTI_OG_WIDTH
1	0	1-5
1	1 or 2	1-4
2	0	1-4
2	1 or 2	1-3
4	0	1-3
4	1 or 2	1-2
8	0	1-2
8	1 or 2	1

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

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

Tip

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 ACE-Lite TBU and LTI TBU buffer configuration parameters.

Table 3-32: Common ACE-Lite TBU and LTI TBU buffer configuration parameters

Parameter name	Values	Description
`TBUCFG_XLATE_SLOTS`	`2, 4, 8, 16, 32, 64, 128, 256, 512`	Number of translation slots, controlling the Hit-Under-Miss capability of the TBU
`TBUCFG_MTLB_LKP_SLOTS`	`2-28`	Number of MTLB lookup slots. Use the default value to provide the best performance when one page size is active in the MTLB. You can: • 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 Ensure that the value of `TBUCFG_MTLB_LKP_SLOTS` is not greater than `TBUCFG_XLATE_SLOTS`
`TBUCFG_UTLB_DEPTH`	`4, 8, 12, 16, 32, 64`	MicroTLB depth, in entries
`TBUCFG_MTLB_DEPTH`	`0, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536`	Main TLB depth, in entries
`TBUCFG_MTLB_WAYS`	`4, 8, 16`	Number of ways in the MTLB
`TBUCFG_MTLB_BANKS`	`1, 2, 4`	Number of banks in the MTLB
`TBUCFG_PMU_COUNTERS`	`4, 16, 32`	Number of PMU counters
`TBUCFG_PARTID_WIDTH`	`1, 6, 9`	Width of PARTID supported. When set to 1, PARTID[8:1] that the DTI interface receives is ignored. When set to `6`, PARTID[8:6] received on the DTI interface is ignored.
`TBUCFG_HZRD_ENTRIES`	`0, 4, 8, 16, 32, 64`	Number of hazard entries

Parameter name	Values	Description
`TBUCFG_SLOTRAM_TYPE`	`0, 1`	RAM type for slot 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 See the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.
`TBUCFG_CACHERAM_TYPE`	`0, 1`	RAM type for cache 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
`BUCFG_DATARAM_TYPE`	`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 you set `TBUCFG_DATARAM_TYPE` 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.

3.4.6 ACE-Lite Translation Buffer Unit register slice configuration parameters

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

Tip

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 register slice configuration parameters.

Table 3-33: ACE-Lite TBU register slice configuration parameters

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

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

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

Tip

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 3-34: ACE-Lite TBU I/O configuration parameters

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

Interface and module name	Parameter name	Values	Description
TBS, TBM	TBUCFG_WBUF_DEPTH	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	TBUCFG_LFIFO_DEPTH	0, 4	Latency FIFO depth. Supported values are as follows: 0, 4.
TBS, TBM	TBUCFG_OT_TRACKER_TYPE	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	TBUCFG_ROT_DEPTH	4, 8, 16, 32, 64, 128, 256, 512	Number of outstanding read transactions. This configuration is valid only when TBUCFG_OT_TRACKER_TYPE is 0.
TBS, TBM	TBUCFG_WOT_DEPTH	4, 8, 16, 32, 64, 128, 256, 512	Number of outstanding write transactions. This configuration is valid only when TBUCFG_OT_TRACKER_TYPE is 0.
TBS, TBM	TBUCFG_DATARAM_TYPE	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.

3.4.8 Integration TBU configuration parameters

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 for the Integration TBU.

Table 3-35: R-TBU and the W-TBU Integration TBU configuration parameters

Integration TBU parameter	TBUs that use parameter		R-TBU/W-TBU parameter
	R-TBU	W-TBU	
TBUCFG_SID_WIDTH	Yes	Yes	TBUCFG_SID_WIDTH
TBUCFG_SSID_WIDTH	Yes	Yes	TBUCFG_SSID_WIDTH

Integration TBU parameter	TBUs that use parameter		R-TBU/W-TBU parameter
	R-TBU	W-TBU	
TBUCFG_ID_WIDTH	Yes	Yes	TBUCFG_ID_WIDTH
TBUCFG_LOOP_WIDTH	Yes	Yes	TBUCFG_LOOP_WIDTH
TBUCFG_CACHERAM_TYPE	Yes	Yes	TBUCFG_CACHERAM_TYPE
TBUCFG_SLOTRAM_TYPE	Yes	Yes	TBUCFG_SLOTRAM_TYPE
TBUCFG_DATARAM_TYPE	Yes	Yes	TBUCFG_DATARAM_TYPE
TBUCFG_USE_ELA_DEBUG	Yes	Yes	TBUCFG_USE_ELA_DEBUG
TBUCFG_LTI_OG_WIDTH_R	Yes	-[5]	TBUCFG_LTI_OG_WIDTH
TBUCFG_LTI_OG_WIDTH_W	-[6]	Yes	TBUCFG_LTI_OG_WIDTH
TBUCFG_XLATE_SLOTS_R	Yes	-[5]	TBUCFG_XLATE_SLOTS
TBUCFG_XLATE_SLOTS_W	-[6]	Yes	TBUCFG_XLATE_SLOTS
TBUCFG_DIRECT_IDX	Yes	Yes	TBUCFG_DIRECT_IDX
TBUCFG_MTLB_PARTS	Yes	Yes	TBUCFG_MTLB_PARTS
TBUCFG_UTLB_DEPTH_R	Yes	-[5]	TBUCFG_UTLB_DEPTH
TBUCFG_UTLB_DEPTH_W	-[6]	Yes	TBUCFG_UTLB_DEPTH
TBUCFG_MTLB_DEPTH	Yes	Yes	TBUCFG_MTLB_DEPTH
TBUCFG_MTLB_WAYS	Yes	Yes	TBUCFG_MTLB_WAYS
TBUCFG_MTLB_BANKS	Yes	Yes	TBUCFG_MTLB_BANKS
TBUCFG_HZRD_ENTRIES_R	Yes	-[5]	TBUCFG_HZRD_ENTRIES
TBUCFG_HZRD_ENTRIES_W	-[6]	Yes	TBUCFG_HZRD_ENTRIES
TBUCFG_PARTID_WIDTH	Yes	Yes	TBUCFG_PARTID_WIDTH
TBUCFG_MTLB_LKP_SLOTS_R	Yes	-[5]	TBUCFG_MTLB_LKP_SLOTS
TBUCFG_MTLB_LKP_SLOTS_W	-[6]	Yes	TBUCFG_MTLB_LKP_SLOTS
TBUCFG_PMU_COUNTERS_R	Yes	-[5]	TBUCFG_PMU_COUNTERS
TBUCFG_PMU_COUNTERS_W	-[6]	Yes	TBUCFG_PMU_COUNTERS
TBUCFG_LA_HNDSHK_MODE	Yes	Yes	TBUCFG_LA_HNDSHK_MODE
TBUCFG_LR_HNDSHK_MODE	Yes	Yes	TBUCFG_LR_HNDSHK_MODE
TBUCFG_DATA_WIDTH	Yes	Yes	TBUCFG_DATA_WIDTH
TBUCFG_AWUSER_WIDTH	Yes	Yes	TBUCFG_AWUSER_WIDTH
TBUCFG_WUSER_WIDTH	Yes	Yes	TBUCFG_WUSER_WIDTH
TBUCFG_BUSER_WIDTH	Yes	Yes	TBUCFG_BUSER_WIDTH
TBUCFG_ARUSER_WIDTH	Yes	Yes	TBUCFG_ARUSER_WIDTH
TBUCFG_RUSER_WIDTH	Yes	Yes	TBUCFG_RUSER_WIDTH
TBUCFG_STASH_SUPPORT	-[5]	Yes	TBUCFG_STASH_SUPPORT
TBUCFG_WBUF_DEPTH	-[6]	Yes	TBUCFG_WBUF_DEPTH

[5] W-TBU is not configurable.
[6] R-TBU is not configurable.

Integration TBU parameter	TBUs that use parameter		R-TBU/W-TBU parameter
	R-TBU	W-TBU	
TBUCFG_LFIFO_DEPTH_R[7]	Yes	-[5]	TBUCFG_LFIFO_DEPTH
TBUCFG_LFIFO_DEPTH_W	-[6]	Yes	TBUCFG_LFIFO_DEPTH
TBUCFG_WOT_DEPTH_W	-[6]	Yes	TBUCFG_WOT_DEPTH
TBUCFG_ROT_DEPTH_R	Yes	-[5]	TBUCFG_ROT_DEPTH
TBUCFG_ROT_DEPTH_W	-[6]	Yes	TBUCFG_ROT_DEPTH
TBUCFG_OT_TRACKER_TYPE	Yes	Yes	TBUCFG_OT_TRACKER_TYPE
TBUCFG_SI_AW_HNDSHK_MODE	-[6]	Yes	TBUCFG_SI_AW_HNDSHK_MODE
TBUCFG_SI_W_HNDSHK_MODE	-[6]	Yes	TBUCFG_SI_W_HNDSHK_MODE
TBUCFG_SI_B_HNDSHK_MODE	-[6]	Yes	TBUCFG_SI_B_HNDSHK_MODE
TBUCFG_SI_AR_HNDSHK_MODE	Yes	-[5]	TBUCFG_SI_AR_HNDSHK_MODE
TBUCFG_SI_R_HNDSHK_MODE_R	Yes	-[5]	TBUCFG_SI_R_HNDSHK_MODE
TBUCFG_SI_R_HNDSHK_MODE_W	-[6]	Yes	TBUCFG_SI_R_HNDSHK_MODE
TBUCFG_MI_AW_HNDSHK_MODE	-[6]	Yes	TBUCFG_MI_AW_HNDSHK_MODE
TBUCFG_MI_W_HNDSHK_MODE	-[6]	Yes	TBUCFG_MI_W_HNDSHK_MODE
TBUCFG_MI_B_HNDSHK_MODE	-[6]	Yes	TBUCFG_MI_B_HNDSHK_MODE
TBUCFG_MI_AR_HNDSK_MODE	Yes	-[5]	TBUCFG_MI_AR_HNDSHK_MODE
TBUCFG_MI_R_HNDSHK_MODE_R	Yes	-[5]	TBUCFG_MI_R_HNDSHK_MODE
TBUCFG_MI_R_HNDSHK_MODE_W	-[6]	Yes	TBUCFG_MI_R_HNDSHK_MODE

3.4.9 Local Translation Interface Translation Buffer Unit configuration parameters

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

Tip

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.

[7] We recommend that you always set this parameter to 0

Table 3-36: LTI TBU configuration parameters

Parameter name	Values	Description
TBUCFG_LTI_ID_WIDTH	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 TBUCFG_LTI_ID_WIDTH >= TBUCFG_LTI_OG_WIDTH. See 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.
TBUCFG_LTI_LOOP_WIDTH	1-256	LTI loop width

3.4.10 Common Translation Buffer Unit debug configuration parameters

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

Tip

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 3-37: Common TBU debug configuration parameters

Parameter name	Values	Description
TBUCFG_USE_ELA_DEBUG	0, 1	Set the TBUCFG_USE_ELA_DEBUG 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: • B.2 ACE-Lite TBU observation interfaces on page 242 • B.3 LTI TBU observation interfaces on page 244

3.5 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*.

Note

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 3.4.3 Translation Control Unit debug configuration parameters on page 83.

For TBU configuration options, see 3.4.10 Common Translation Buffer Unit debug configuration parameters on page 92.

Signals

For TCU observation signals, see B.1 TCU observation interfaces on page 239.

For ACE-Lite TBU observation signals, see B.2 ACE-Lite TBU observation interfaces on page 242.

For LTI TBU observation signals, see B.3 LTI TBU observation interfaces on page 244.

4. Programmers model for MMU-700

The programmers model describes the MMU-700 registers.

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 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.

Note

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 SMMU_S_CR2	Secure global Control Registers	Provides the controls for Secure table and queue access attributes
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

Register	Name	Description
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
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

Register	Name	Description
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
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_SMR0	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
SMMU_PMCG_CNTENCLR0	SMMU PMCG Counter Enable Clear register	Provides the clear mechanism for the counter enables
SMMU_PMCG_INTENSET0	SMMU PMCG Interrupt contribution Enable Set register	Provides the set mechanism for the counter interrupt contribution enables
SMMU_PMCG_INTENCLR0	SMMU PMCG Interrupt contribution Enable Clear register	Provides the clear mechanism for the counter interrupt enables
SMMU_PMCG_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_PMCG_OVSSET0	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_PMCG_CAPR	SMMU PMCG Counter shadow value Capture Register	Controls the counter shadow value capture mechanism
SMMU_PMCG_SCR	SMMU PMCG Secure Control Register	Secure Control Register
SMMU_PMCG_CFGR	SMMU PMCG Configuration information Register	Provides information about the PMCG implementation
SMMU_PMCG_CR	SMMU PMCG Control Register	Contains the Performance Monitor control flags
SMMU_PMCG_CEID0 - SMMU_PMCG_CEID1	SMMU PMCG Common Event ID registers	Contains the lower and upper 64 bits of the Common Event identification bitmap
SMMU_PMCG_IRQ_CTRL	SMMU PMCG IRQ enable register	Contains the Performance Monitors IRQ enable
SMMU_PMCG_IRQ_CTRLACK	SMMU PMCG IRQ enable Acknowledge register	Provides acknowledgment of the completion of updates to SMMU_PMCG_IRQ_CTRL
SMMU_PMCG_AIDR	SMMU PMCG Architecture Identification Register	Provides the Performance Monitor Architecture Identification
SMMU_PMCG_ID_REGS	ID registers	IMPLEMENTATION DEFINED
SMMU_PMCG_PMAUTHSTATUS	PMU Authentication Status register	Performance Monitor authentication status
SMMU_PMCG_PMDEVARCH	PMU Device Architecture register	Performance Monitor architecture identifier
SMMU_PMCG_PMDEVTYPE	PMU Device Type register	Performance Monitor device type

Related information

SMMUv3 implementation on page 60

4.2 MMU-700 memory map

The MMU-700 memory map contains all registers.

4.2.1 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 - 0x7FFFFC`	TBU61 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.

Note

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

Note

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.

4.2.2 TCU memory map

The TCU memory map contains various categories of registers.

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

- 4.6 TCU microarchitectural registers on page 111 for controlling microarchitectural features
- 4.8 TCU system discovery registers on page 130
- 4.7 TCU RAS registers on page 121
- 4.5 TCU PMU registers on page 108

The following registers are also included:

- 4.9 TCU PIU integration registers on page 149.
- 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: • SMMUv3 registers, page 0 • TCU *Performance Monitor Counter Group* (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

Address range	Description
0x08E00-0x08E78	Microarchitectural registers System discovery registers Integration registers
0x08E80-0x08EFC	*Reliability, Availability, and Serviceability* (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

4.2.3 TBU memory map

The TBU memory map contains various categories of registers.

The TBU registers contain the following:

- IMPLEMENTATION DEFINED 4.13 TBU microarchitectural registers on page 158 for controlling microarchitectural features
- 4.15 TBU system discovery registers on page 173
- 4.14 TBU RAS registers on page 164
- Direct access to cache state
- 4.12 TBU PMU registers on page 155
- 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

Note

Any regions that the table does not show are reserved.

4.3 MMU-700 registers summary

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

4.3.1 TCU identification register 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 register summary

Offset	Name	Type	Description
0x00FFC	SMMU_CIDR3	RO	4.4 TCU component and peripheral ID registers on page 107. All registers are RO.
0x00FF8	SMMU_CIDR2	RO	
0x00FF4	SMMU_CIDR1	RO	
0x00FF0	SMMU_CIDR0	RO	
0x00FEC	SMMU_PIDR3	RO	
0x00FE8	SMMU_PIDR2	RO	
0x00FE4	SMMU_PIDR1	RO	
0x00FE0	SMMU_PIDR0	RO	
0x00FDC	SMMU_PIDR7	RO	
0x00FD8	SMMU_PIDR6	RO	
0x00FD4	SMMU_PIDR5	RO	
0x00FD0	SMMU_PIDR4	RO	

4.3.2 TCU and TBU PMU identification register summary

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

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

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

Offset	Name	Type	Description
0x02FB8	SMMU_PMCG_PMAUTHSTATUS	RO	4.5 TCU PMU registers on page 108 4.12 TBU PMU registers on page 155
0x02FD0	SMMU_PMCG_PIDR4	RO	
0x02FD4	SMMU_PMCG_PIDR5	RO	
0x02FD8	SMMU_PMCG_PIDR6	RO	
0x02FDC	SMMU_PMCG_PIDR7	RO	
0x02FE0	SMMU_PMCG_PIDR0	RO	
0x02FE4	SMMU_PMCG_PIDR1	RO	
0x02FE8	SMMU_PMCG_PIDR2	RO	
0x02FEC	SMMU_PMCG_PIDR3	RO	
0x02FF0	SMMU_PMCG_CIDR0	RO	
0x02FF4	SMMU_PMCG_CIDR1	RO	
0x02FF8	SMMU_PMCG_CIDR2	RO	
0x02FFC	SMMU_PMCG_CIDR3	RO	

4.3.3 TCU Reliability, Availability, and Service register 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 register summary

Offset	Name	Type	Width	Description
0x08E80	TCU_ERRFR	RO, Secure	64-bit	4.7.1 TCU_ERRFR register on page 121
0x08E88	TCU_ERRCTLR	RW, Secure	64-bit	4.7.2 TCU_ERRCTLR register on page 123
0x08E90	TCU_ERRSTATUS	RW, Secure	64-bit	4.7.3 TCU_ERRSTATUS register on page 123
0x08EC0	TCU_ERRGEN	RW, Secure	64-bit	4.7.4 TCU_ERRGEN register on page 127

4.3.4 TCU microarchitectural register 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 register summary

Offset	Name	Type	Width	Description
0x08E00	TCU_CTRL	RW	32-bit	4.6.1 TCU_CTRL register on page 111
0x08E04	TCU_QOS	RW	32-bit	4.6.2 TCU_QOS register on page 113
0x08E08	TCU_CFG	RO	32-bit	4.6.3 TCU_CFG register on page 114

Offset	Name	Type	Width	Description
0x08E10	TCU_STATUS	RO	32-bit	4.6.4 TCU_STATUS register on page 115
0x08E18	TCU_SCR	RW, Secure	32-bit	4.6.7 TCU_SCR register on page 119
0x09000-0x093FC	TCU_NODE_CTRL*n*	RW	32-bit	4.6.5 TCU_NODE_CTRLn register on page 116
0x09400-0x097FC	TCU_NODE_STATUS*n*	RO	32-bit	4.6.6 TCU_NODE_STATUSn register on page 118
0x09800-0x0981C	TCU_WC_SxLy_CMAX	RW	32-bit	4.6.8 TCU_WC_SxLy_CMAX registers on page 120

4.3.5 TCU system discovery register 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 register summary

Offset	Name	Type	Width	Description
0x08E34	TCU_SYSDISC0	RO	32-bit	4.8.1 TCU_SYSDISC0 system discovery register on page 131
0x08E38	TCU_SYSDISC1	RO	32-bit	4.8.2 TCU_SYSDISC1 system discovery register on page 132
0x08E3C	TCU_SYSDISC2	RO	32-bit	4.8.3 TCU_SYSDISC2 system discovery register on page 133
0x08E40	TCU_SYSDISC3	RO	32-bit	4.8.4 TCU_SYSDISC3 system discovery register on page 134
0x08E44	TCU_SYSDISC4	RO	32-bit	4.8.5 TCU_SYSDISC4 system discovery register on page 135
0x08E48	TCU_SYSDISC5	RO	32-bit	4.8.6 TCU_SYSDISC5 system discovery register on page 136
0x08E4C	TCU_SYSDISC6	RO	32-bit	4.8.7 TCU_SYSDISC6 system discovery register on page 137
0x08E50	TCU_SYSDISC7	RO	32-bit	4.8.8 TCU_SYSDISC7 system discovery register on page 138
0x08E54	TCU_SYSDISC8	RO	32-bit	4.8.9 TCU_SYSDISC8 system discovery register on page 139
0x08E58	TCU_SYSDISC9	RO	32-bit	4.8.10 TCU_SYSDISC9 system discovery register on page 140
0x08E5C	TCU_SYSDISC10	RO	32-bit	4.8.11 TCU_SYSDISC10 system discovery register on page 141
0x08E60	TCU_SYSDISC11	RO	32-bit	4.8.12 TCU_SYSDISC11 system discovery register on page 142
0x08E64	TCU_SYSDISC12	RO	32-bit	4.8.13 TCU_SYSDISC12 system discovery register on page 143
0x08E68	TCU_SYSDISC13	RO	32-bit	4.8.14 TCU_SYSDISC13 system discovery register on page 144
0x08E6C	TCU_SYSDISC14	RO	32-bit	4.8.15 TCU_SYSDISC14 system discovery register on page 145
0x08E70	TCU_SYSDISC15	RO	32-bit	4.8.16 TCU_SYSDISC15 system discovery register on page 146
0x08E74	TCU_SYSDISC16	RO	32-bit	4.8.17 TCU_SYSDISC16 system discovery register on page 147
0x08E78	TCU_SYSDISC17	RO	32-bit	4.8.18 TCU_SYSDISC17 system discovery register on page 148

4.3.6 TCU integration register 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 register summary

Offset	Name	Type	Width	Description
0x08E20	ITEN	RW	32-bit	4.9.1 ITEN register for the TCU on page 149
0x08E24	ITOP_PIU	RW	32-bit	4.9.2 ITOP register for the TCU Programmer Interface Unit on page 150
0x08E2C	ITOP_TMU	RW	32-bit	4.10.1 ITOP register for the TCU Translation Management Unit on page 152
0x08E30	ITIN_TMU	RO	32-bit	4.10.2 ITIN register for the TCU Translation Management Unit on page 153

4.3.7 TBU identification register 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 register summary

Offset	Name	Type	Description
0x00FFC	SMMU_CIDR3	RO	4.11 TBU component and peripheral ID registers on page 154
0x00FF8	SMMU_CIDR2	RO	
0x00FF4	SMMU_CIDR1	RO	
0x00FF0	SMMU_CIDR0	RO	
0x00FEC	SMMU_PIDR3	RO	
0x00FE8	SMMU_PIDR2	RO	
0x00FE4	SMMU_PIDR1	RO	
0x00FE0	SMMU_PIDR0	RO	
0x00FDC	SMMU_PIDR7	RO	
0x00FD8	SMMU_PIDR6	RO	
0x00FD4	SMMU_PIDR5	RO	
0x00FD0	SMMU_PIDR4	RO	

4.3.8 TBU Reliability, Availability, and Serviceability register 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 register summary

Offset	Name	Width	Type	Description
0x08E80	TBU_ERRFR	64-bit	RO, Secure	4.14.1 TBU_ERRFR register on page 165
0x08E88	TBU_ERRCTLR	64-bit	RW, Secure	4.14.2 TBU_ERRCTLR register on page 166
0x08E90	TBU_ERRSTATUS	64-bit	RW, Secure	4.14.3 TBU_ERRSTATUS register on page 167
0x08EC0	TBU_ERRGEN	64-bit	RW, Secure	4.14.4 TBU_ERRGEN register on page 170

4.3.9 TBU microarchitectural register 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 register summary

Offset	Name	Type	Width	Description
0x08E00	TBU_CTRL	RW	32-bit	4.13.1 TBU_CTRL register on page 158
0x08E04	TBU_LTI_PORT_RESOURCE_LIMIT	RW	32-bit	4.13.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 160
0x08E18	TBU_SCR	RW, Secure	32-bit	4.13.3 TBU_SCR register on page 163

4.3.10 TBU system discovery register 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 register summary

Offset	Name	Type	Width	Description
0x08E30	TBU_SYSDISC0	RO	32-bit	4.15.1 TBU_SYSDISC0 system discovery register on page 173
0x08E34	TBU_SYSDISC1	RO	32-bit	4.15.2 TBU_SYSDISC1 system discovery register on page 174
0x08E38	TBU_SYSDISC2	RO	32-bit	4.15.3 TBU_SYSDISC2 system discovery register on page 175
0x08E3C	TBU_SYSDISC3	RO	32-bit	4.15.4 TBU_SYSDISC3 system discovery register on page 176
0x08E40	TBU_SYSDISC4	RO	32-bit	4.15.5 TBU_SYSDISC4 system discovery register on page 177
0x08E44	TBU_SYSDISC5	RO	32-bit	4.15.6 TBU_SYSDISC5 system discovery register on page 178
0x08E48	TBU_SYSDISC6	RO	32-bit	4.15.7 TBU_SYSDISC6 system discovery register on page 179
0x08E4C	TBU_SYSDISC7	RO	32-bit	4.15.8 TBU_SYSDISC7 system discovery register on page 180
0x08E50	TBU_SYSDISC8	RO	32-bit	4.15.9 TBU_SYSDISC8 system discovery register on page 181
0x08E54	TBU_SYSDISC9	RO	32-bit	4.15.10 TBU_SYSDISC9 system discovery register on page 182
0x08E58	TBU_SYSDISC10	RO	32-bit	4.15.11 TBU_SYSDISC10 system discovery register on page 183
0x08E5C	TBU_SYSDISC11	RO	32-bit	4.15.12 TBU_SYSDISC11 system discovery register on page 184
0x08E60	TBU_SYSDISC12	RO	32-bit	4.15.13 TBU_SYSDISC12 system discovery register on page 185
0x08E64	TBU_SYSDISC13	RO	32-bit	4.15.14 TBU_SYSDISC13 system discovery register on page 186
0x08E68	TBU_SYSDISC14	RO	32-bit	4.15.15 TBU_SYSDISC14 system discovery register on page 187

4.3.11 TBU integration register 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 register summary

Offset	Name	Type	Width	Description
`0x08E20`	ITEN	RW	32-bit	4.16.1 ITEN register for the TBU on page 189
`0x08E24`	ITOP_TBU	RW	32-bit	4.16.2 ITOP_TBU register on page 190
`0x08E28`	ITIN_TBU	RW	32-bit	4.16.3 ITIN_TBU register on page 191

4.4 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(*p_level*, ecorevnum)	REVAND, minor revision, where *p_level* is `2` for p2.
		[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

Name	Offset	Field	Value	Description
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

4.5 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.

4.5.1 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_SMR0
  - All counters share this mask register
  - The mask is 32 bits because the TCU uses 32-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 4.5.3 SMMU_PMCG_CFGR fields on page 110.
- SMMU_PMCG_CR
- SMMU_PMCG_CEID{0-1}. See 4.5.4 SMMU_PMCG_CEID{0-1} registers on page 110.
- 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

4.5.2 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 3.2.2.1 SMMUv3 architectural performance events on page 42.

The following events are also counted for prefetch accesses:

0x80-0x90
: Walk cache events.

0x92-0x94
: Configuration cache events.

0xC0-0xC8
: RAS events.

4.5.3 SMMU_PMCG_CFGR fields

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 3.2.2.4 SMMUv3 PMU register architectural options on page 46.

4.5.4 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

4.5.5 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(*p_level*, ecorevnum)	REVAND, minor revision, where *p_level* is 2 for p2

Address	Name	Field	Value	Description
		[3:0]	0x00	CMOD
0x02FE8	SMMU_PMCG_PIDR2, Peripheral ID2	[7:4]	0x01 for r1	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
0x02FE0	SMMU_PMCG_PIDR0, Peripheral ID0	[7:0]	0x87	PART_0: bits [7:0] of the Part number
0x02FDC	SMMU_PMCG_PIDR7, Peripheral ID7	-	RES0	Reserved
0x02FD8	SMMU_PMCG_PIDR6, Peripheral ID6	-	RES0	Reserved
0x02FD4	SMMU_PMCG_PIDR5, Peripheral ID5	-	RES0	Reserved
0x02FD0	SMMU_PMCG_PIDR4, Peripheral ID4	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code
0x02FB8	SMMU_PMCG_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.

4.6 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 4.6.7 TCU_SCR register on page 119 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.

You can only write to the 4.6.1 TCU_CTRL register on page 111, 4.6.2 TCU_QOS register on page 113, 4.6.5 TCU_NODE_CTRLn register on page 116, and 4.6.8 TCU_WC_SxLy_CMAX registers on page 120 registers 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.

4.6.1 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 we direct you to do so.

Configurations

The TCU_CTRL register is available in all configurations.

Attributes

The TCU_CTRL register attributes are as follows:

Width

32-bit

Functional group

4.3.4 TCU microarchitectural register summary on page 103

Address offset

`0x08E00`

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-1: TCU_CTRL register bit assignments

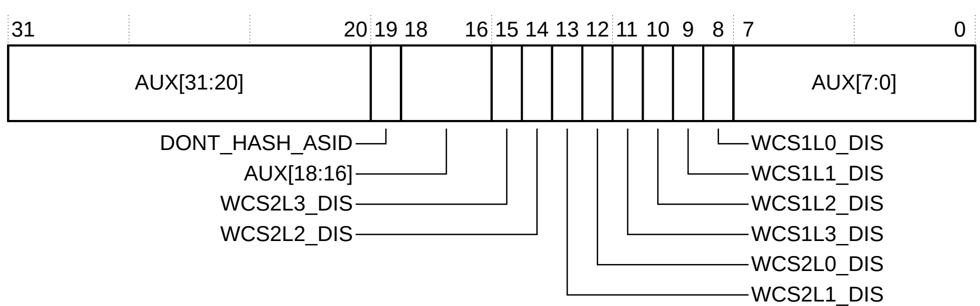


Table 4-21: TCU_CTRL register bit descriptions

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

4.6.2 TCU_QOS register

The TCU_QOS register selects the QoS value to attach to transactions 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

4.3.4 TCU microarchitectural register summary on page 103

Address offset

`0x08E04`

Type

RW

Reset value

0

Usage constraints

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-2: TCU_QOS register bit assignments

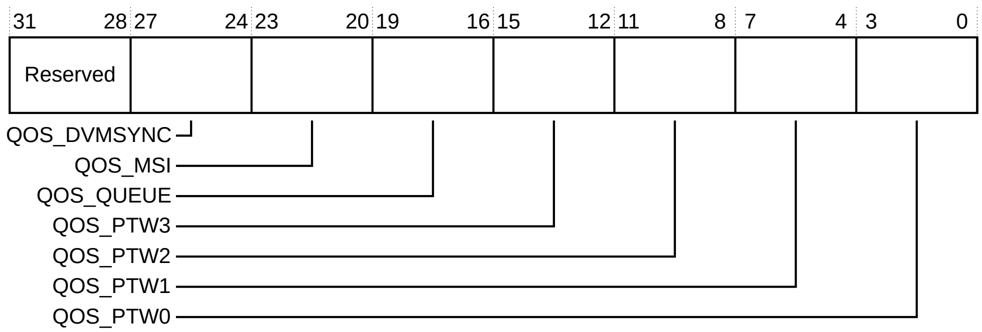


Table 4-22: TCU_QOS register bit descriptions

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_CTRL*n*.PRIORITY = 3
[11:8]	QOS_PTW2	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRL*n*.PRIORITY = 2
[7:4]	QOS_PTW1	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRL*n*.PRIORITY = 1
[3:0]	QOS_PTW0	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRL*n*.PRIORITY = 0

4.6.3 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

4.3.4 TCU microarchitectural register summary on page 103

Address offset

`0x08E08`

Type

RO

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-3: TCU_CFG register bit assignments

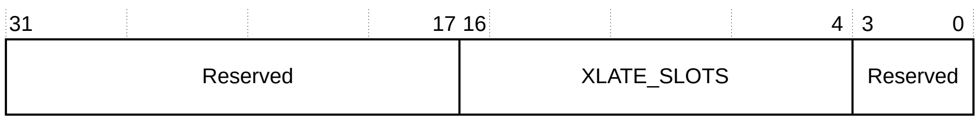


Table 4-23: TCU_CFG register bit descriptions

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 `TCUCFG_XLATE_SLOTS`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.
[3:0]	-	Reserved

4.6.4 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

4.3.4 TCU microarchitectural register summary on page 103

Address offset

`0x08E10`

Type

RO

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-4: TCU_STATUS register bit assignments

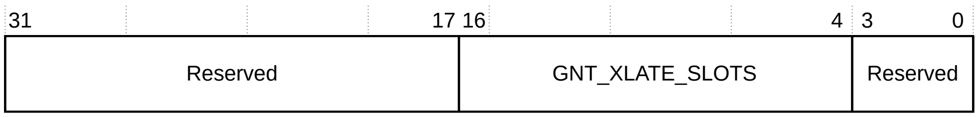


Table 4-24: TCU_STATUS register bit descriptions

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

4.6.5 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 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

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

The TCU_NODE_CTRL*n* register is available in all configurations.

Attributes

The TCU_NODE_CTRL*n* register attributes are as follows:

Width

32-bit

Functional group

4.3.4 TCU microarchitectural register summary on page 103

Address offset

`0x09000-0x093FC`

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-5: TCU_NODE_CTRLn register bit assignments

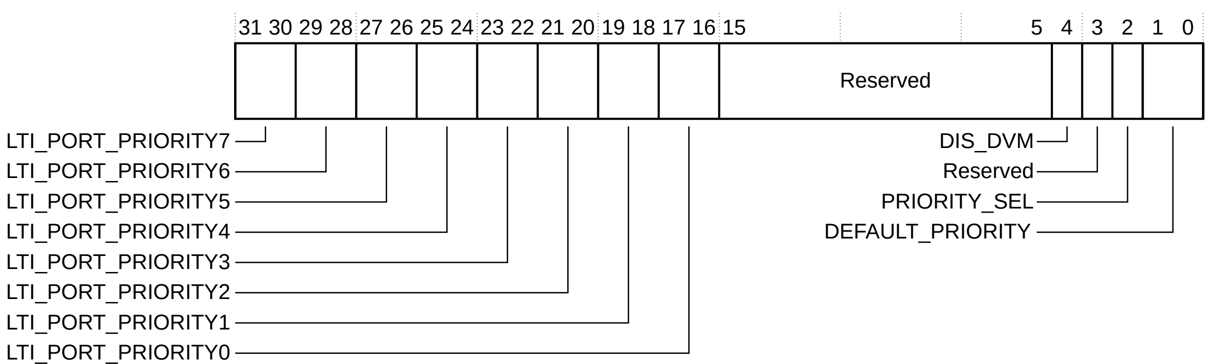


Table 4-25: TCU_NODE_CTRL*n* register bit descriptions

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
[23:22]	LTI_PORT_PRIORITY3	Priority for LTI Port 3 for this node, if the port exists

Bits	Name	Description
[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 *m*.
[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.

4.6.6 TCU_NODE_STATUSn register

The TCU_NODE_STATUSn register provides status for each node, similarly to TCU_NODE_CTRL*n*. Each node has a single status register.

Configurations

The TCU_NODE_STATUS*n* register is available in all configurations.

Attributes

The TCU_NODE_STATUS*n* register attributes are as follows:

Width

32-bit

Functional group

4.3.4 TCU microarchitectural register summary on page 103

Address offset

`0x09400-0x097FC`

Type

RO

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-6: TCU_NODE_STATUS register bit assignments

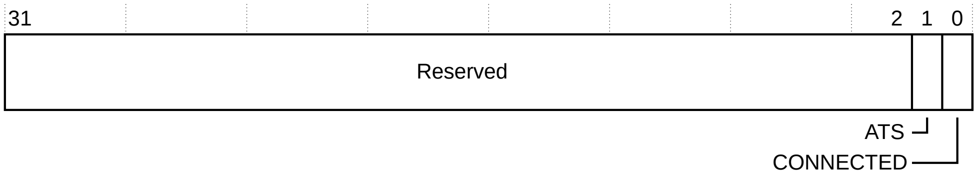


Table 4-26: TCU_NODE_STATUS*n* register bit descriptions

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.

4.6.7 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_PMCG_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

The TCU_SCR register is available in all configurations.

Attributes

The TCU_SCR register attributes are as follows:

Width

32-bit

Functional group

4.3.4 TCU microarchitectural register summary on page 103

Address offset

`0x08E18`

Type

Secure, RW

Reset value

sec_override. See A.1.12 TCU tie-off signals on page 204.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-7: TCU_SCR register bit assignments

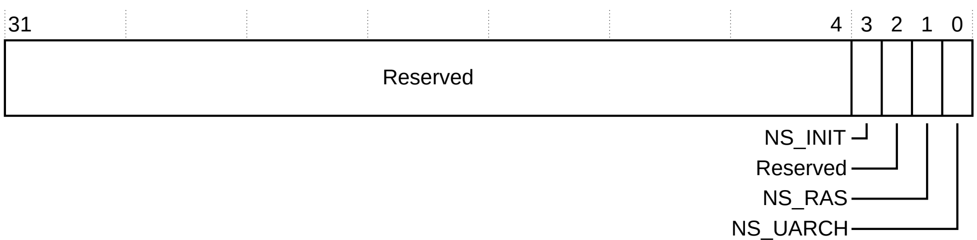


Table 4-27: TCU_SCR register bit descriptions

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 204.

[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 204.
		
		If Secure translation might be used, we recommend that software does not set this bit.

4.6.8 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

Note

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

4.7 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 4.6.7 TCU_SCR register on page 119.

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

4.7.1 TCU_ERRFR register

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

Configurations

The TCU_ERRFR register is available in all configurations.

Attributes

The TCU_ERRFR register attributes are as follows:

Width

32-bit

Functional group

4.3.3 TCU Reliability, Availability, and Service register summary on page 103

Address offset

`0x08E80`

Type

S, RO

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-8: TCU_ERRFR register bit assignments

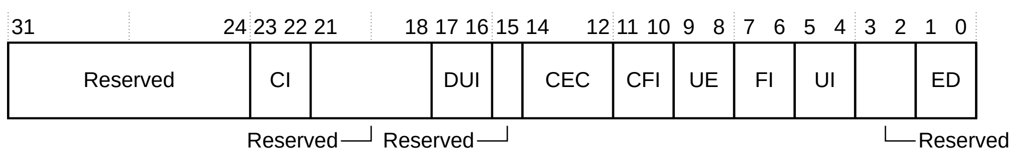


Table 4-29: TCU_ERRFR register bit descriptions

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	-

Bits	Name	Description	Value
[1:0]	ED	Error detection is always enabled	01b

4.7.2 TCU_ERRCTLR register

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

Configurations

The TCU_ERRCTLR register is available in all configurations.

Attributes

The TCU_ERRCTLR register attributes are as follows:

Width

32-bit

Functional group

4.3.3 TCU Reliability, Availability, and Service register summary on page 103

Address offset

`0x08E88`

Type

S, RW

Reset value

8

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-9: TCU_ERRCTLR register bit assignments

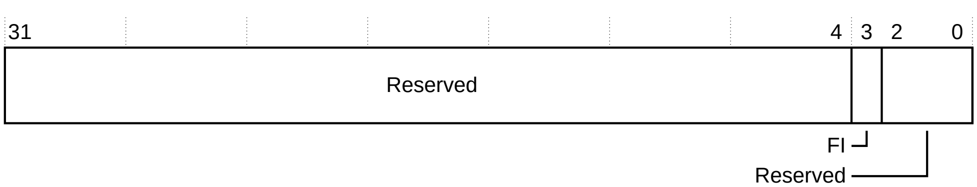


Table 4-30: TCU_ERRCTLR register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3]	FI	Fault handling interrupt enable. See the ras_fhi signal in A.1.9 TCU interrupt signals on page 200.
[2:0]	-	Reserved

4.7.3 TCU_ERRSTATUS register

Use the TCU error status register to find out whether different types of error have occurred. 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

The TCU_ERRSTATUS register is available in all configurations.

Attributes

The TCU_ERRSTATUS register attributes are as follows:

Width

32-bit

Functional group

4.3.3 TCU Reliability, Availability, and Service register summary on page 103

Address offset

`0x08E90`

Type

Secure, RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-10: TCU_ERRSTATUS register bit assignments

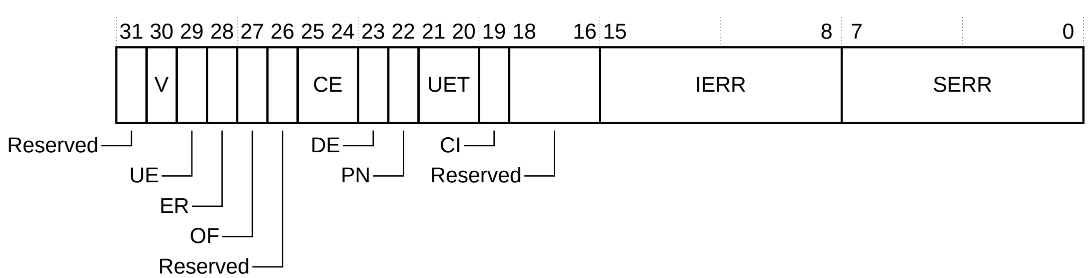


Table 4-31: TCU_ERRSTATUS register bit descriptions

Bits	Name	Description
[31]	-	Reserved

Bits	Name	Description
[30]	V	Status Register valid. The possible values of this bit are: **0** ERRSTATUS is not valid **1** ERRSTATUS is valid. At least one error has been recorded 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. This bit resets to zero on a reset.
[29]	UE	Uncorrected error, or errors. The possible values of this bit are: **0** No errors that could not be corrected or deferred **1** At least one error detected that has not been corrected or deferred If the OF bit is set to 1 and is not being cleared to zero in the same write, direct writes to this bit are ignored. This bit is read/write-one-to-clear.
[28]	ER	Error Reported. The possible values of this bit are: **0** No in-band error (External abort) is reported **1** The node to the master making the access or other transaction signaled an External abort This bit is read/write-one-to-clear.
[27]	OF	Overflow. Multiple errors are detected. This bit is set to 1 when: • Multiple errors are detected on the same cycle • A new error occurs when there is already a valid record in the register This bit is read/write-one-to-clear.
[26]	-	Reserved
[25:24]	CE	Correctable Error, or errors. **00b** No correctable errors recorded **10b** At least one Corrected error recorded Other values are Reserved. 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.
[23]	DE	Deferred error, or errors. The possible values of this bit are: **0** No errors were deferred **1** At least one error was not corrected and deferred 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. This bit is read/write-one-to-clear.
[22]	PN	Poison. The possible values of this bit are: **0** Uncorrected error or deferred error is recorded because a corrupt value was detected, for example, by an *Error Detection Code* (EDC) **1** Uncorrected error or deferred error is recorded because a poison value was detected This bit is read/write-one-to-clear.

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) Accessing this field has the following behavior. This field is not valid and reads UNKNOWN if any of the following are true: • TCU_ERRSTATUS.V == 0b0 • TCU_ERRSTATUS.UE == 0b0 Otherwise, this field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an UNKNOWN value.
[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 This bit is read/write-one-to-clear.
[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.

Bits	Name	Description
[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.

4.7.4 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 4.7.3 TCU_ERRSTATUS register on page 123.

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.

Tip

This register has identical fields to 4.7.3 TCU_ERRSTATUS register on page 123.

Configurations

The TCU_ERRGEN register is available in all configurations.

Attributes

The TCU_ERRGEN register attributes are as follows:

Width

64-bit

Functional group

4.3.3 TCU Reliability, Availability, and Service register summary on page 103

Address offset

`0x08EC0`

Type

Secure, RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-11: TCU_ERRGEN register bit assignments

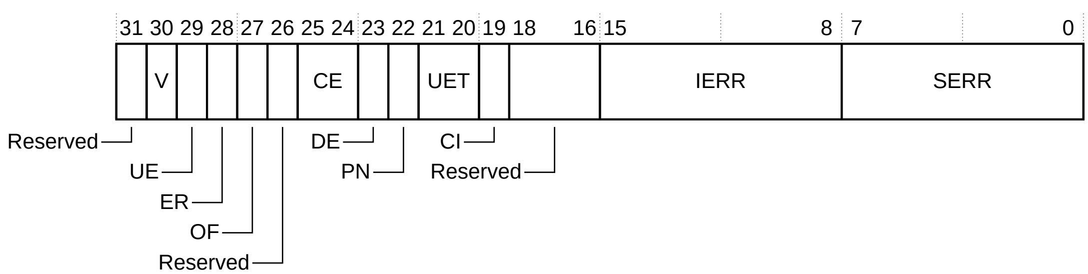


Table 4-32: TCU_ERRGEN register bit descriptions

Bits	Name	Description
[31]	-	Reserved
[30]	V	Status Register valid. The possible values of this bit are: **0** ERRSTATUS is not valid **1** ERRSTATUS is valid. At least one error has been recorded 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. This bit resets to zero on a reset.

Bits	Name	Description
[29]	UE	Uncorrected error, or errors. The possible values of this bit are: **0** No errors that could not be corrected or deferred **1** At least one error detected that has not been corrected or deferred 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.
[28]	ER	Error Reported. The possible values of this bit are: **0** No in-band error (External abort) is reported **1** The node to the master making the access or other transaction signaled an External abort Writes to this field are ignored.
[27]	OF	Overflow. Multiple errors are detected. This bit is set to 1 when: • 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 This bit is cleared by writing a 1 to it. A write of 0 is ignored.
[26]	-	Reserved
[25:24]	CE	Correctable Error, or errors. **00b** No correctable errors recorded **10b** At least one Corrected error recorded Other values are Reserved. 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.
[23]	DE	Deferred error, or errors. The possible values of this bit are: **0** No errors were deferred **1** At least one error was not corrected and deferred This error is raised when wpoison is set in BIU. 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. This bit is read/write-one-to-clear.
[22]	PN	Poison. The possible values of this bit are: **0** Uncorrected error or deferred error is recorded because a corrupt value was detected, for example, by an *Error Detection Code* (EDC) **1** Uncorrected error or deferred error is recorded because a poison value was detected Writes to this field are ignored.

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 TCU system discovery registers

This section describes the TCU system discovery registers.

4.8.1 TCU_SYSDISC0 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC0 register is available in all configurations.

Attributes

The TCU_SYSDISC0 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E34`

Type

RO

Reset value

`TCUCFG_WC_DEPTH`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-12: TCU_SYSDISC0 register bit assignments

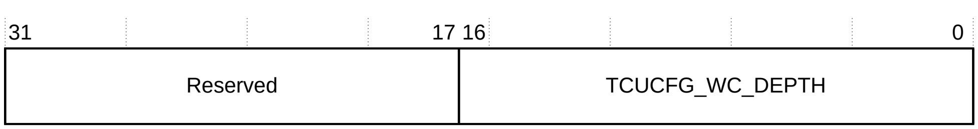


Table 4-33: TCU_SYSDISC0 register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved

Bits	Name	Description

[16:0]	TCUCFG_WC_DEPTH	The read data reflects the chosen parameter value, for example:
		`17'h0_0008 : 8`
		....
		`17'h1_0000 : 65536`

4.8.2 TCU_SYSDISC1 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC1 register is available in all configurations.

Attributes

The TCU_SYSDISC1 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E38`

Type

RO

Reset value

`TCUCFG_CC_DEPTH`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-13: TCU_SYSDISC1 register bit assignments

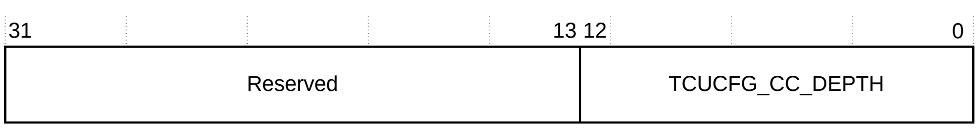


Table 4-34: TCU_SYSDISC1 register bit descriptions

Bits	Name	Description
[31:13]	-	Reserved

Bits	Name	Description

[12:0]	TCUCFG_CC_DEPTH	The read data reflects the chosen parameter value, for example:
		`13'h0004 : 4`
		....
		`13'h1000 : 4096`

4.8.3 TCU_SYSDISC2 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC2 register is available in all configurations.

Attributes

The TCU_SYSDISC2 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E3C`

Type

RO

Reset value

`TCUCFG_WC_WAYS`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-14: TCU_SYSDISC2 register bit assignments

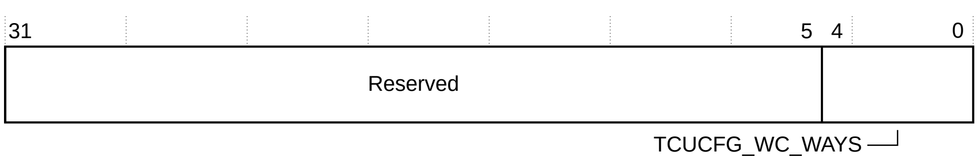


Table 4-35: TCU_SYSDISC2 register bit descriptions

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`

4.8.4 TCU_SYSDISC3 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC3 register is available in all configurations.

Attributes

The TCU_SYSDISC3 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E40`

Type

RO

Reset value

`TCUCFG_WC_BANKS`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-15: TCU_SYSDISC3 register bit assignments

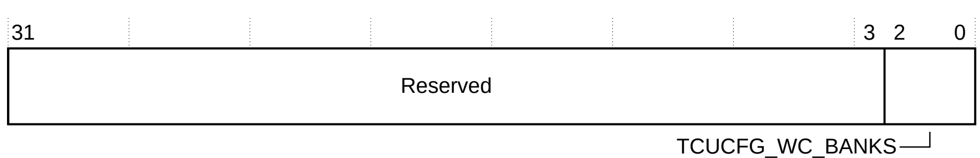


Table 4-36: TCU_SYSDISC3 register bit descriptions

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`

4.8.5 TCU_SYSDISC4 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC4 register is available in all configurations.

Attributes

The TCU_SYSDISC4 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E44`

Type

RO

Reset value

`TCUCFG_XLATE_SLOTS`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-16: TCU_SYSDISC4 register bit assignments

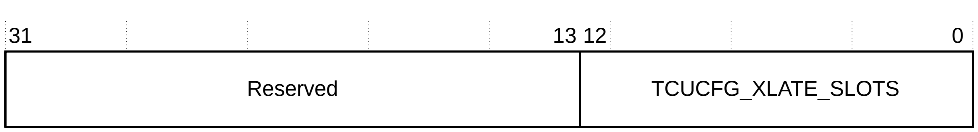


Table 4-37: TCU_SYSDISC4 register bit descriptions

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`

4.8.6 TCU_SYSDISC5 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC5 register is available in all configurations.

Attributes

The TCU_SYSDISC5 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E48`

Type

RO

Reset value

`TCUCFG_PTW_SLOTS`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-17: TCU_SYSDISC5 register bit assignments

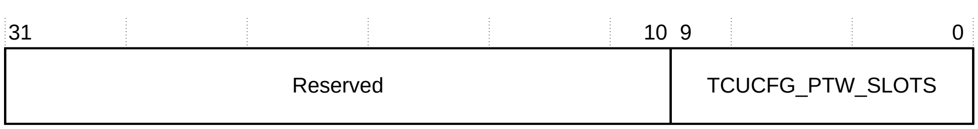


Table 4-38: TCU_SYSDISC5 register bit descriptions

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`

4.8.7 TCU_SYSDISC6 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC6 register is available in all configurations.

Attributes

The TCU_SYSDISC6 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E4C`

Type

RO

Reset value

`TCUCFG_CTW_SLOTS`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-18: TCU_SYSDISC6 register bit assignments

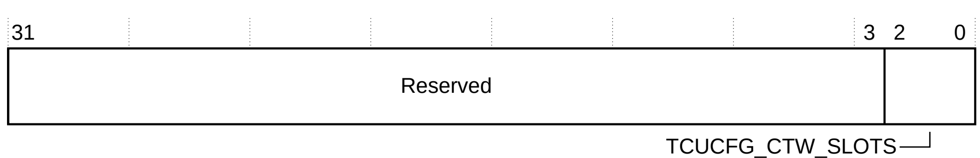


Table 4-39: TCU_SYSDISC6 register bit descriptions

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`

4.8.8 TCU_SYSDISC7 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC7 register is available in all configurations.

Attributes

The TCU_SYSDISC7 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E50`

Type

RO

Reset value

`TCUCFG_CC_IDXGEN_MODE`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-19: TCU_SYSDISC7 register bit assignments

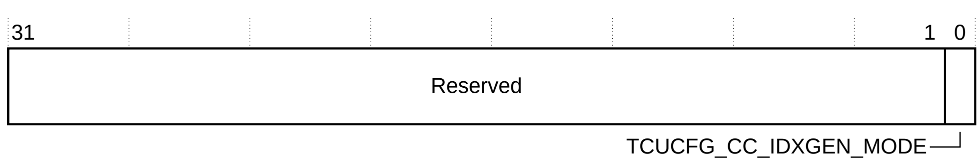


Table 4-40: TCU_SYSDISC7 register bit descriptions

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`

4.8.9 TCU_SYSDISC8 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC8 register is available in all configurations.

Attributes

The TCU_SYSDISC8 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E54`

Type

RO

Reset value

`TCUCFG_DTI_ATS`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-20: TCU_SYSDISC8 register bit assignments

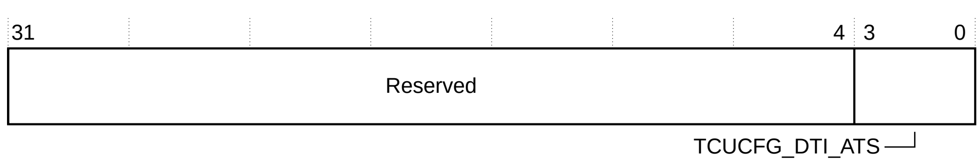


Table 4-41: TCU_SYSDISC8 register bit descriptions

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`

4.8.10 TCU_SYSDISC9 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC9 register is available in all configurations.

Attributes

The TCU_SYSDISC9 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E58`

Type

RO

Reset value

`TCUCFG_NUM_TBU`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-21: TCU_SYSDISC9 register bit assignments

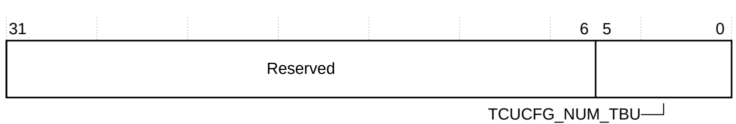


Table 4-42: TCU_SYSDISC9 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved

[5:0]	TCUCFG_NUM_TBU	The read data reflects the chosen parameter value, for example:
		
		`6'h01 : 1`
		
		....
		
		`6'h3E : 62`
		
		Note:
		Legal values are 14 and 62.

4.8.11 TCU_SYSDISC10 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC10 register is available in all configurations.

Attributes

The TCU_SYSDISC10 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E5C`

Type

RO

Reset value

`TCUCFG_NUM_PMU_COUNTERS`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-22: TCU_SYSDISC10 register bit assignments

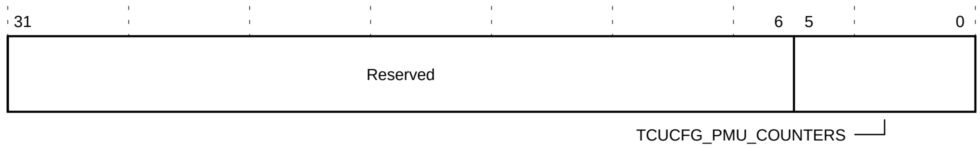


Table 4-43: TCU_SYSDISC10 register bit descriptions

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`

4.8.12 TCU_SYSDISC11 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC11 register is available in all configurations.

Attributes

The TCU_SYSDISC11 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E60`

Type

RO

Reset value

`TCUCFG_PARTID_WIDTH`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-23: TCU_SYSDISC11 register bit assignments

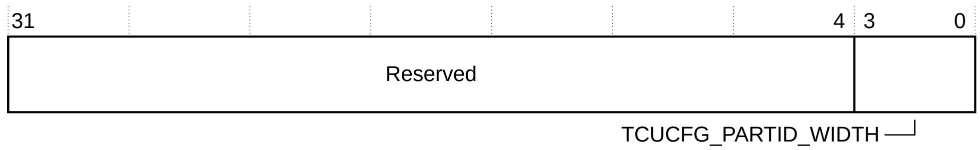


Table 4-44: TCU_SYSDISC11 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved

[3:0]	TCUCFG_PARTID_WIDTH	The read data reflects the chosen parameter value, for example:
		`4'b0001 : 1`
		....
		`4'b1001 : 9`

4.8.13 TCU_SYSDISC12 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC12 register is available in all configurations.

Attributes

The TCU_SYSDISC12 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E64`

Type

RO

Reset value

`TCUCFG_HZU_DEPTH`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-24: TCU_SYSDISC12 register bit assignments

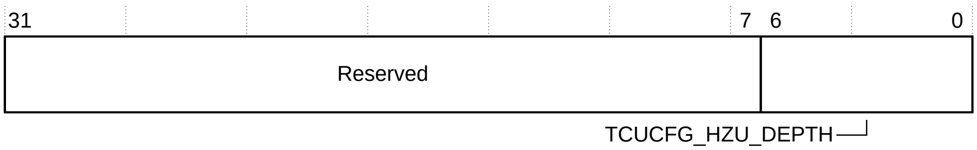


Table 4-45: TCU_SYSDISC12 register bit descriptions

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`

4.8.14 TCU_SYSDISC13 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC13 register is available in all configurations.

Attributes

The TCU_SYSDISC13 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E68`

Type

RO

Reset value

`TCUCFG_PREFETCH_SUPPORTED`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-25: TCU_SYSDISC13 register bit assignments

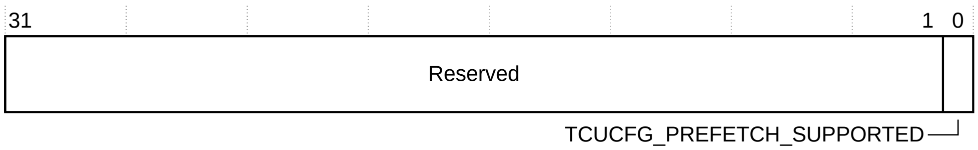


Table 4-46: TCU_SYSDISC13 register bit descriptions

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`

4.8.15 TCU_SYSDISC14 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC14 register is available in all configurations.

Attributes

The TCU_SYSDISC14 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E6C`

Type

RO

Reset value

`TCUCFG_DATARAM_TYPE`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-26: TCU_SYSDISC14 register bit assignments

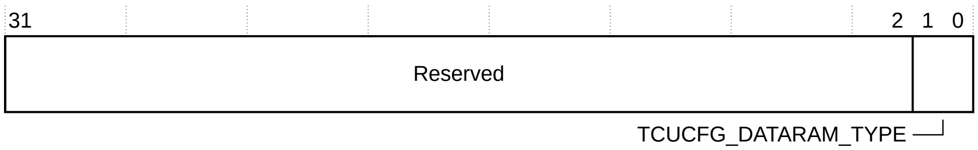


Table 4-47: TCU_SYSDISC14 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved

[1:0]	TCUCFG_DATARAM_TYPE	The read data reflects the chosen parameter value, for example:
		`2’b00 : 0`
		....
		`2’b10 : 2`

4.8.16 TCU_SYSDISC15 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC15 register is available in all configurations.

Attributes

The TCU_SYSDISC15 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E70`

Type

RO

Reset value

`TCUCFG_SLOTRAM_TYPE`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-27: TCU_SYSDISC15 register bit assignments

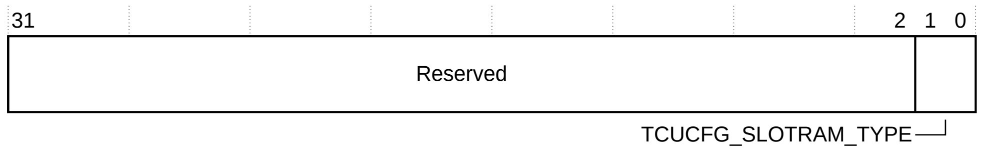


Table 4-48: TCU_SYSDISC15 register bit descriptions

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`

4.8.17 TCU_SYSDISC16 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC16 register is available in all configurations.

Attributes

The TCU_SYSDISC16 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E74`

Type

RO

Reset value

`TCUCFG_CACHERAM_TYPE`. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-28: TCU_SYSDISC16 register bit assignments

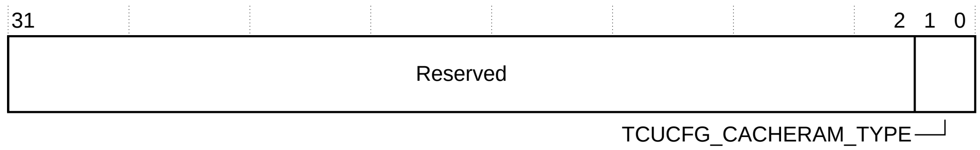


Table 4-49: TCU_SYSDISC16 register bit descriptions

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`

4.8.18 TCU_SYSDISC17 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

The TCU_SYSDISC17 register is available in all configurations.

Attributes

The TCU_SYSDISC17 register attributes are as follows:

Width

32-bit

Functional group

4.3.5 TCU system discovery register summary on page 104

Address offset

`0x08E78`

Type

RO

Reset value

`TCUCFG_QTW_DATA_WIDTH`. See 3.4.1 Translation Control Unit I/O configuration parameters on page 80.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-29: TCU_SYSDISC17 register bit assignments

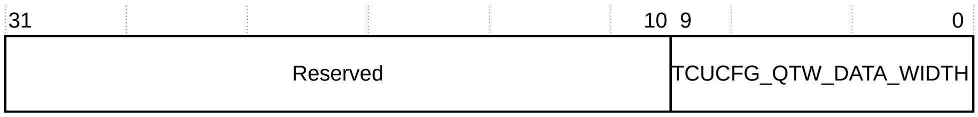


Table 4-50: TCU_SYSDISC17 register bit descriptions

Bits	Name	Description
[31:10]	-	Reserved

[9:0]	TCUCFG_QTW_DATA_WIDTH	The read data reflects the chosen parameter value, for example:
		`10’h040 : 64`
		....
		`10’h200 : 512`

4.9 TCU PIU integration registers

This section describes the *Programmer Interface Unit* (PIU) integration registers.

4.9.1 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.

Configurations

The ITEN register is available in all configurations.

Attributes

The ITEN register attributes are as follows:

Width

32-bit

Functional group

4.3.6 TCU integration register summary on page 104

Address offset

`0x08E20`

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-30: ITEN register bit assignments

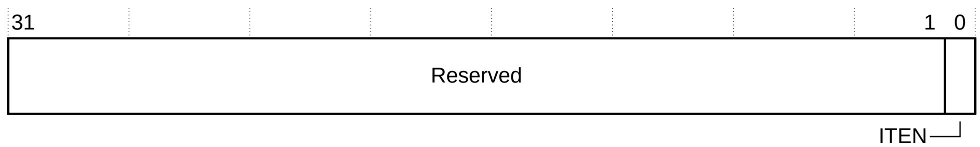


Table 4-51: ITEN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	ITEN	**0** Integration mode is disabled **1** Integration mode is enabled

4.9.2 ITOP register for the TCU Programmer Interface Unit

This section describes the TCU ITOP register for the *Programmer Interface Unit* (PIU).

Configurations

The ITOP register is available in all configurations.

Attributes

The ITOP register attributes are as follows:

Width

32-bit

Functional group

4.3.6 TCU integration register summary on page 104

Address offset

`0x08E24`

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-31: ITOP register bit assignments

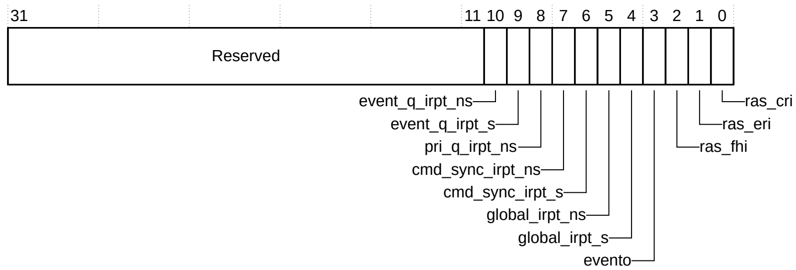


Table 4-52: ITOP register bit descriptions

Bits	Name	Description
[31:11]	-	Reserved, SBZ
[10]	event_q_irpt_ns	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[9]	event_q_irpt_s	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[8]	pri_q_irpt_ns	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[7]	cmd_sync_irpt_ns	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[6]	cmd_sync_irpt_s	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[5]	global_irpt_ns	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[4]	global_irpt_s	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified

Bits	Name	Description
[3]	evento	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[2]	ras_fhi	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[1]	ras_eri	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[0]	ras_cri	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified

4.10 TCU TMU integration registers

This section describes the TCU *Translation Management Unit* (TMU) integration registers.

4.10.1 ITOP register for the TCU Translation Management Unit

This section describes the ITOP register for the TCU *Translation Management Unit* (TMU).

Configurations

The ITOP register is available in all configurations.

Attributes

The ITOP register attributes are as follows:

Width

32-bit

Functional group

4.3.6 TCU integration register summary on page 104

Address offset

`0x08E2C`

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-32: ITOP register bit assignments

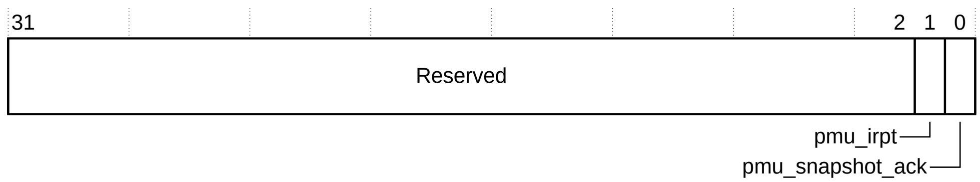


Table 4-53: ITOP register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved, SBZ
[1]	pmu_irpt	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified.
[0]	pmu_snapshot_ack	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified.

4.10.2 ITIN register for the TCU Translation Management Unit

This section describes the ITIN register for the TCU *Translation Management Unit* (TMU).

Configurations

The ITIN register is available in all configurations.

Attributes

The ITIN register attributes are as follows:

Width

32-bit

Functional group

4.3.6 TCU integration register summary on page 104

Address offset

`0x08E30`

Type

RO

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-33: ITIN register bit assignments

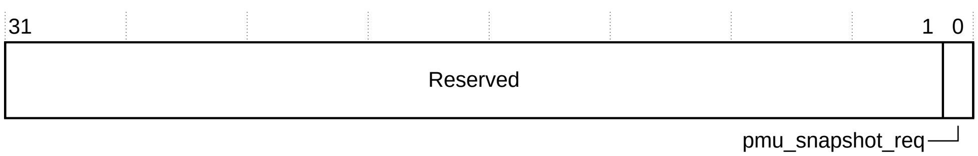


Table 4-54: ITIN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved, SBZ
[0]	pmu_snapshot_req	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1, depending on the value of the input signal that is specified

4.11 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 4-55: 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(*p_level*, ecorevnum)	REVAND, minor revision. Where *p_level* is 2 for p2.
		[3:0]	`0x00`	CMOD
SMMU_PIDR2, Peripheral ID2	`0x00FE8`	[7:4]	`0x01`	REVISION, major revision
		[3]	1	JEDEC-assigned value for DES always used

Name	Offset	Field	Value	Description
		[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

4.12 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.

4.12.1 Registers

The TBU and TCU support the same PMCG registers.

See 4.5 TCU PMU registers on page 108.

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.

4.12.2 Events

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

For more information, see 3.2.2 Performance Monitoring Unit on page 41.

The following table shows the architectural events that are implemented.

Table 4-56: 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 4-57: 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`	Translation 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
`0x8B`	DCMO downgrade	Yes	-	Counts once whenever either: • 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: • 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-Abort. The transaction responded with FaultAbort because of DTI StreamDisable, GlobalDisable is not counted.

Event ID	Description	Filterable	Secure only	Description
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 - 0xE7	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.

4.12.3 SMMU_PMCG_CFGR fields

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_CFCR fields values, see 3.2.2.4 SMMUv3 PMU register architectural options on page 46.

See also 3.4 Configuration parameters and methodology on page 80.

4.12.4 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 4-58: SMMU_PMCG_CEID{0-1} registers

Name	Offset	Value
SMMU_PMCG_CEID0	0x02e20	0x00000087
SMMU_PMCG_CEID1	0x02e28	0x00000000

4.12.5 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 4-59: PMU ID registers

Name	Offset	Field	Value	Description
SMMU_PMCG_CIDR3, Component ID3	0x02FFC	[7:0]	0xB1	Preamble
SMMU_PMCG_CIDR2, Component ID2	0x02FF8	[7:0]	0x05	Preamble

Name	Offset	Field	Value	Description
SMMU_PMCG_CIDR1, Component ID1	0x02FF4	[7:0]	0x90	Preamble
SMMU_PMCG_CIDR0, Component ID0	0x02FF0	[7:0]	0x0D	Preamble
SMMU_PMCG_PIDR3, Peripheral ID3	0x02FEC	[7:4]	MAX(*p_level*, ecorevnum)	REVAND, minor revision, where *p_level* is 0 for p0.
		[3:0]	0x00	CMOD
SMMU_PMCG_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_PMCG_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_PMCG_PIDR0, Peripheral ID0	0x02FE0	[7:0]	0x88	PART_0: bits [7:0] of the Part number
SMMU_PMCG_PIDR7, Peripheral ID7	0x02FDC	-	RES0	Reserved
SMMU_PMCG_PIDR6, Peripheral ID6	0x02FD8			Reserved
SMMU_PMCG_PIDR5, Peripheral ID5	0x02FD4			Reserved
SMMU_PMCG_PIDR4, Peripheral ID4	0x02FD0	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code
SMMU_PMCG_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.

4.13 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 4.13.3 TBU_SCR register on page 163 is Secure-only.

Non-secure access to the 4.13.3 TBU_SCR register on page 163 is *Read-As-Zero* (RAZ)/*Writes-Ignored* (WI).

Non-secure access to the following registers, when TBU_SCR.NS_UARCH = 0, is RAZ/WI:

- 4.13.1 TBU_CTRL register on page 158
- 4.13.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 160
- 4.16 TBU integration registers on page 188:
  - 4.16.1 ITEN register for the TBU on page 189
  - 4.16.2 ITOP_TBU register on page 190
  - 4.16.3 ITIN_TBU register on page 191

4.13.1 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

The TBU_CTRL register is available in all configurations.

Attributes

The TBU_CTRL register attributes are as follows:

Width

32-bit

Functional group

4.3.9 TBU microarchitectural register summary on page 106

Address offset

`0x08E00`

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-34: TBU_CTRL register bit assignments

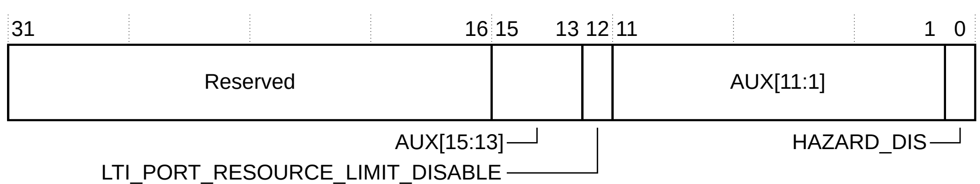


Table 4-60: TBU_CTRL register bit descriptions

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	This bit is used only when there is more than one LTI port configured. An ACE-Lite TBU only has one LTI port internally. **0** When 0, the 4.13.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 160 is used to control the usage of translation slots and DTI credits by LTI ports. **1** When 1, all LTI ports are permitted to use the maximum resource. For reasons of avoiding deadlock and starvation, a few slots and DTI credits are reserved for each LTI port regardless of the value of this register field and the 4.13.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 160. Therefore, if multiple LTI ports are present, it is not possible for any individual port to use all the slots or DTI credits.
[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	**0** 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. **1** 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.

4.13.2 TBU_LTI_PORT_RESOURCE_LIMIT register

The TBU_LTI_PORT_RESOURCE_LIMIT register limits the resource usage for each LTI port.

Non-secure access to TBU_LTI_PORT_RESOURCE_LIMIT when TBU_SCR.NS_UARCH-=-0 is RAZ/WI. See 4.13.3 TBU_SCR register on page 163. 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 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.
- DTI translation tokens that the TCU returns in the CONDIS_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.

Note

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 4-61: 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
`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 × `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,(TBUCFG_XLATE_SLOTS*<lti_port_resource_limit>)), (TBUCFG_XLATE_SLOTS-(2*(NUM_LTI_PORTS-1))))

Note

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.

Configurations

The TBU_LTI_PORT_RESOURCE_LIMIT register is available in all configurations.

Attributes

The TBU_LTI_PORT_RESOURCE_LIMIT register attributes are as follows:

Width

32-bit

Functional group

4.3.9 TBU microarchitectural register summary on page 106

Address offset

`0x08E04`

Type

RW

Reset value

`4'hF`

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-35: TBU_LTI_PORT_RESOURCE_LIMIT register bit assignments

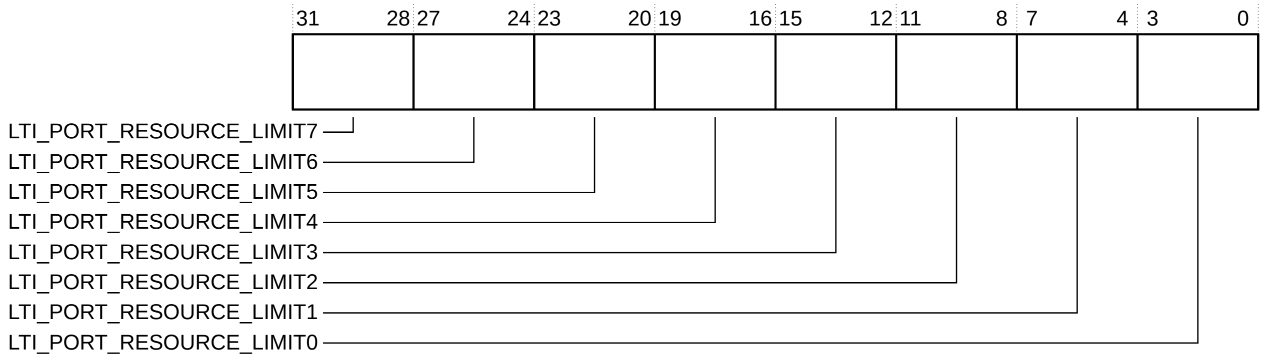


Table 4-62: TBU_LTI_PORT_RESOURCE_LIMIT register bit descriptions

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

4.13.3 TBU_SCR register

This section describes the TBU Secure Control register.

Configurations

The TBU_SCR register is available in all configurations.

Attributes

The TBU_SCR register attributes are as follows:

Width

32-bit

Functional group

4.3.9 TBU microarchitectural register summary on page 106

Address offset

`0x08E08`

Type

RW

Reset value

sec_override. See A.2.10 TBU tie-off signals on page 229.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-36: TBU_SCR register bit assignments

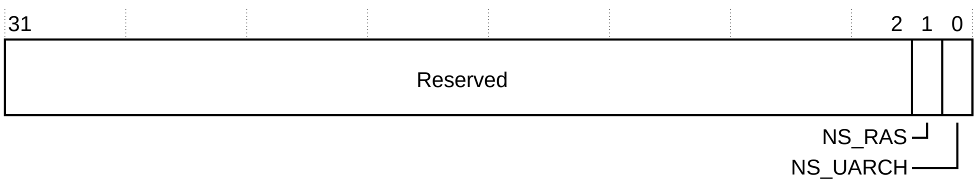


Table 4-63: TBU_SCR register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1]	NS_RAS	Non-secure register access that is permitted for *Reliability, Availability, and Serviceability* (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.2.10 TBU tie-off signals on page 229.

[0]	NS_UARCH	Non-secure register access that is permitted for the following microarchitecture registers:
		• 4.13.1 TBU_CTRL register on page 158
		• 4.13.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 160
		• 4.16 TBU integration registers on page 188:
		  ◦ 4.16.1 ITEN register for the TBU on page 189
		  ◦ 4.16.2 ITOP_TBU register on page 190
		  ◦ 4.16.3 ITIN_TBU register on page 191
		
		When this bit is 0, Non-secure writes to the following registers are ignored, and Non-secure reads from these registers return zero:
		• 4.13.1 TBU_CTRL register on page 158
		• 4.13.2 TBU_LTI_PORT_RESOURCE_LIMIT register on page 160
		• 4.16 TBU integration registers on page 188:
		  ◦ 4.16.1 ITEN register for the TBU on page 189
		  ◦ 4.16.2 ITOP_TBU register on page 190
		  ◦ 4.16.3 ITIN_TBU register on page 191
		
		The sec_override input sets reset value of this signal. See A.2.10 TBU tie-off signals on page 229.
		
		If Secure translation might be used, we recommend that software does not set this bit.

4.14 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 4.14.3 TBU_ERRSTATUS register on page 167.

If TBU_ERRCTLR.FI is set, an interrupt is raised on ras_fhi. See 3.1.2.7 TBU interrupt interfaces on page 35.

4.14.1 TBU_ERRFR register

The TBU_CTRL register disables TBU features. Do not modify the bits in this register unless Arm® instructs you to do so.

Configurations

The TBU_ERRFR register is available in all configurations.

Attributes

The TBU_ERRFR register attributes are as follows:

Width

32-bit

Functional group

4.3.8 TBU Reliability, Availability, and Serviceability register summary on page 105

Address offset

`0x08E80`

Type

Secure, RO

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-37: TBU_ERRFR register bit assignments

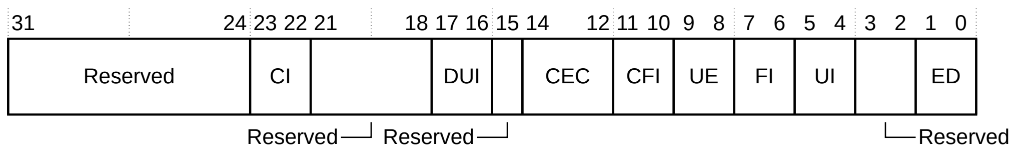


Table 4-64: TBU_ERRFR register bit descriptions

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
[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

4.14.2 TBU_ERRCTLR register

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

Configurations

The TBU_ERRCTLR register is available in all configurations.

Attributes

The TBU_ERRCTLR register attributes are as follows:

Width

32-bit

Functional group

4.3.8 TBU Reliability, Availability, and Serviceability register summary on page 105

Address offset

`0x08E88`

Type

S, RW

Reset value

8

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-38: TBU_ERRCTLR register bit assignments

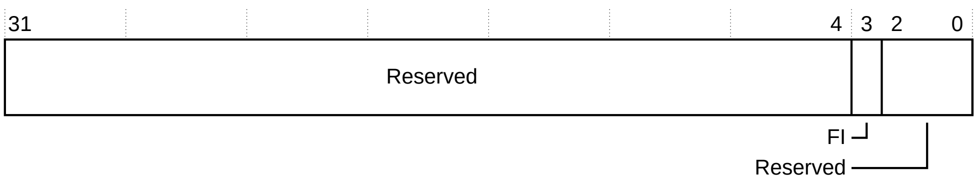


Table 4-65: TBU_ERRCTLR register bit descriptions

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

4.14.3 TBU_ERRSTATUS register

Use the TBU error status register to find out whether different types of error have occurred. 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

The TBU_ERRSTATUS register is available in all configurations.

Attributes

The TBU_ERRSTATUS register attributes are as follows:

Width

32-bit

Functional group

4.3.8 TBU Reliability, Availability, and Serviceability register summary on page 105

Address offset

`0x08E90`

Type

Secure, RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-39: TBU_ERRSTATUS register bit assignments

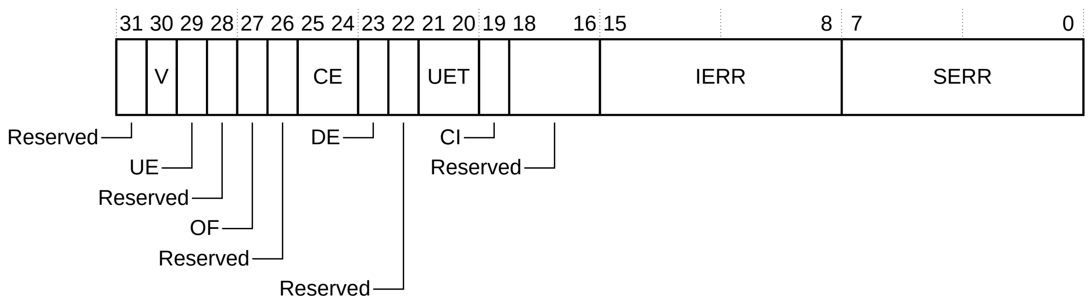


Table 4-66: TBU_ERRSTATUS register bit descriptions

Bits	Name	Description
[31]	-	Reserved
[30]	V	Status Register valid. The possible values of this bit are as follows: **0** ERRSTATUS is not valid **1** ERRSTATUS is valid, meaning that at least one error has been recorded This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See . This bit resets to zero on a reset.
[29]	UE	Uncorrected errors. The possible values of this bit are: **0** No errors that could not be corrected or deferred **1** At least one error is detected that has not been corrected or deferred This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See .
[28]	-	Reserved
[27]	OF	Overflow. Multiple errors detected. This bit is set to 1 when: • 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 This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See .
[26]	-	Reserved
[25:24]	CE	Correctable Errors: **00b** No correctable errors recorded **10b** At least one corrected error recorded Other values are Reserved. Clearing depends on other ERRSTATUS fields. See .

Bits	Name	Description
[23]	DE	Deferred errors. The possible values of this bit are as follows: **0** No errors were deferred **1** At least one error was not corrected and deferred This error is raised when the *Bus Interface Unit* (BIU) sets wpoison. This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See .
[22]	-	Reserved
[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 as follows: **0b00** Uncorrected error, UnContainable error (UC) Writes to this field are ignored.
[19]	CI	Indicates whether a critical error condition has been recorded. The possible values of this bit are as follows: **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. This field indicates the source of the error as follows: **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. Writes to this field are ignored.

Bits	Name	Description
[7:0]	SERR	Error code. This provides information about the earliest unacknowledged Error. It can contain the following values: **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 All other values are reserved. Writes to this field are ignored.

4.14.4 TBU_ERRGEN register

The TBU_CTRL register disables TBU features. Do not modify the bits in this register unless Arm® instructs you to do so.

The field locations are same as for the 4.14.3 TBU_ERRSTATUS register on page 167.

When this register is updated, the 4.14.3 TBU_ERRSTATUS register on page 167 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 4.14.3 TBU_ERRSTATUS register on page 167.

Configurations

The TBU_ERRGEN register is available in all configurations.

Attributes

The TBU_ERRGEN register attributes are as follows:

Width

32-bit

Functional group

4.3.8 TBU Reliability, Availability, and Serviceability register summary on page 105

Address offset

`0x08EC0`

Type

S, RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-40: TBU_ERRSTATUS register bit assignments

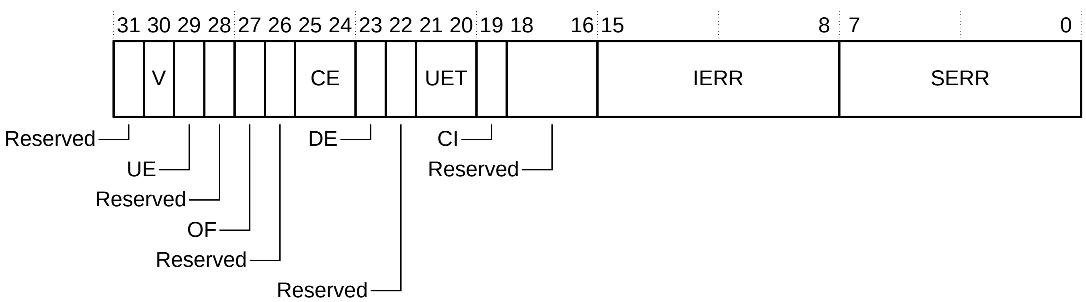


Table 4-67: TBU_ERRSTATUS register bit descriptions

Bits	Name	Description
[31]	-	Reserved
[30]	V	Status Register valid. The possible values of this bit are as follows: **0** ERRSTATUS is not valid **1** ERRSTATUS is valid, meaning that at least one error has been recorded This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See . This bit resets to zero on a reset.
[29]	UE	Uncorrected errors. The possible values of this bit are: **0** No errors that could not be corrected or deferred **1** At least one error is detected that has not been corrected or deferred This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See .
[28]	-	Reserved

Bits	Name	Description
[27]	OF	Overflow. Multiple errors detected. This bit is set to 1 when: • 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 This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See .
[26]	-	Reserved
[25:24]	CE	Correctable Errors: **00b** No correctable errors recorded **10b** At least one corrected error recorded Other values are Reserved. Clearing depends on other ERRSTATUS fields. See .
[23]	DE	Deferred errors. The possible values of this bit are as follows: **0** No errors were deferred **1** At least one error was not corrected and deferred This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See .
[22]	-	Reserved
[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 as follows: **0b00** Uncorrected error, UnContainable error (UC) Writes to this field are ignored.
[19]	CI	Indicates whether a critical error condition has been recorded. The possible values of this bit are as follows: **0** No critical error condition **1** Critical error condition Writes to this field are ignored.
[18:16]	-	Reserved

Bits	Name	Description
[15:8]	IERR	**IMPLEMENTATION DEFINED** error code. This field indicates the source of the error as follows: **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. Writes to this field are ignored.
[7:0]	SERR	Error code. This provides information about the earliest unacknowledged Error. It can contain the following values: **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 All other values are reserved. Writes to this field are ignored.

4.15 TBU system discovery registers

This section describes the TBU system discovery registers.

4.15.1 TBU_SYSDISC0 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC0 register is available in all configurations.

Attributes

The TBU_SYSDISC0 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E30`

Type

RO

Reset value

`TBUCFG_MTLB_DEPTH`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-41: TBU_SYSDISC0 register bit assignments

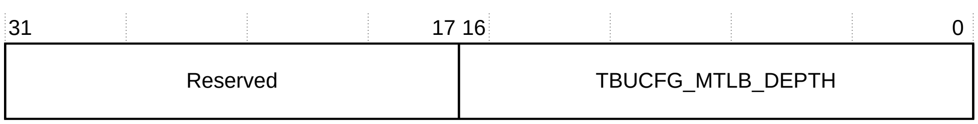


Table 4-68: TBU_SYSDISC0 register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved

[16:0]	TBUCFG_MTLB_DEPTH	The read data reflects the chosen parameter value, for example:
		`17'h0_0008 : 8`
		....
		`17'h1_0000 : 65536`

4.15.2 TBU_SYSDISC1 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC1 register is available in all configurations.

Attributes

The TBU_SYSDISC1 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E34`

Type

RO

Reset value

`TBUCFG_UTLB_DEPTH`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-42: TBU_SYSDISC1 register bit assignments

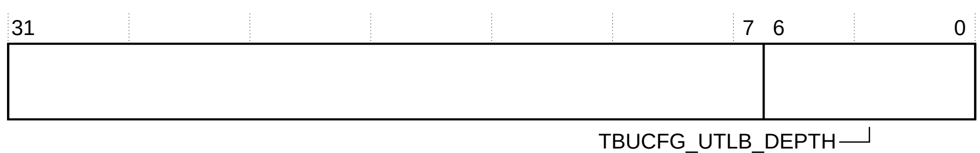


Table 4-69: TBU_SYSDISC1 register bit descriptions

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`

4.15.3 TBU_SYSDISC2 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC2 register is available in all configurations.

Attributes

The TBU_SYSDISC2 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E38`

Type

RO

Reset value

`TBUCFG_MTLB_WAYS`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-43: TBU_SYSDISC2 register bit assignments

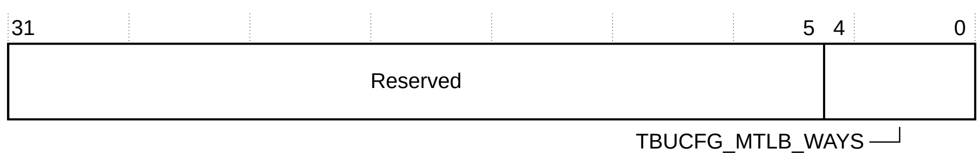


Table 4-70: TBU_SYSDISC2 register bit descriptions

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`

4.15.4 TBU_SYSDISC3 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC3 register is available in all configurations.

Attributes

The TBU_SYSDISC3 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E3C`

Type

RO

Reset value

`TBUCFG_MTLB_BANKS`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-44: TBU_SYSDISC3 register bit assignments

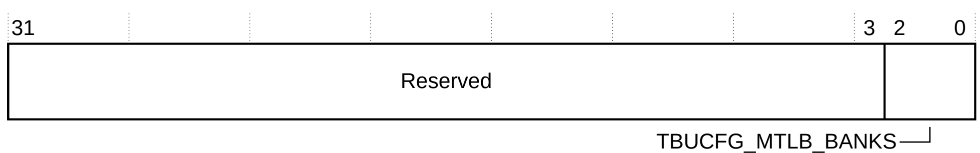


Table 4-71: TBU_SYSDISC3 register bit descriptions

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`

4.15.5 TBU_SYSDISC4 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC4 register is available in all configurations.

Attributes

The TBU_SYSDISC4 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E40`

Type

RO

Reset value

`TBUCFG_XLATE_SLOTS`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-45: TBU_SYSDISC4 register bit assignments

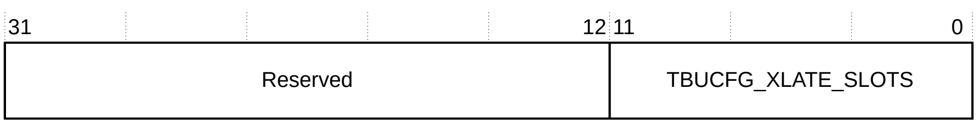


Table 4-72: TBU_SYSDISC4 register bit descriptions

Bits	Name	Description
[31:12]	-	Reserved

[11:0]	TBUCFG_XLATE_SLOTS	The read data reflects the chosen parameter value, for example:
		`11'h0002 : 2`
		....
		`11'h200 : 512`

4.15.6 TBU_SYSDISC5 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC5 register is available in all configurations.

Attributes

The TBU_SYSDISC5 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E44`

Type

RO

Reset value

`TBUCFG_PMU_COUNTERS`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-46: TBU_SYSDISC5 register bit assignments

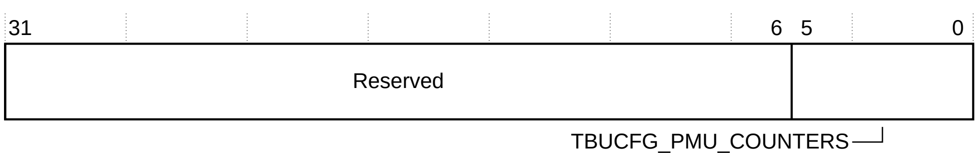


Table 4-73: TBU_SYSDISC5 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved

[5:0]	TBUCFG_PMU_COUNTERS	The read data reflects the chosen parameter value, for example:
		`6'h04 : 4`
		....
		`6'h20 : 32`

4.15.7 TBU_SYSDISC6 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC6 register is available in all configurations.

Attributes

The TBU_SYSDISC6 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E48`

Type

RO

Reset value

`TBUCFG_SID_WIDTH`. See 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-47: TBU_SYSDISC6 register bit assignments

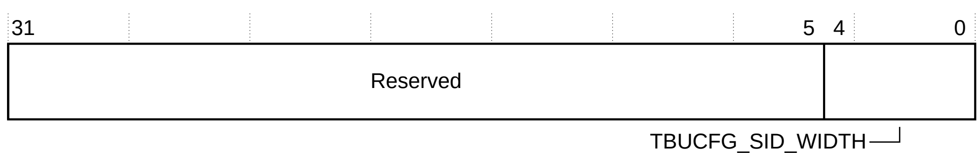


Table 4-74: TBU_SYSDISC6 register bit descriptions

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`

4.15.8 TBU_SYSDISC7 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC7 register is available in all configurations.

Attributes

The TBU_SYSDISC7 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E4C`

Type

RO

Reset value

`TBUCFG_SSID_WIDTH`. See 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-48: TBU_SYSDISC7 register bit assignments

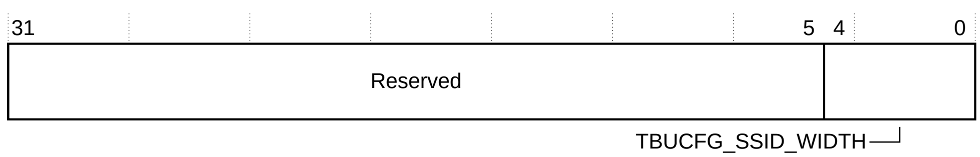


Table 4-75: TBU_SYSDISC7 register bit descriptions

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`

4.15.9 TBU_SYSDISC8 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC8 register is available in all configurations.

Attributes

The TBU_SYSDISC8 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E50`

Type

RO

Reset value

`TBUCFG_DIRECT_IDX`. See 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-49: TBU_SYSDISC8 register bit assignments

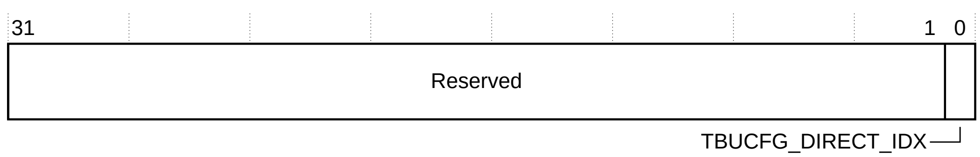


Table 4-76: TBU_SYSDISC8 register bit descriptions

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`

4.15.10 TBU_SYSDISC9 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC9 register is available in all configurations.

Attributes

The TBU_SYSDISC9 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E54`

Type

RO

Reset value

`TBUCFG_MTLB_PARTS`. See 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-50: TBU_SYSDISC9 register bit assignments

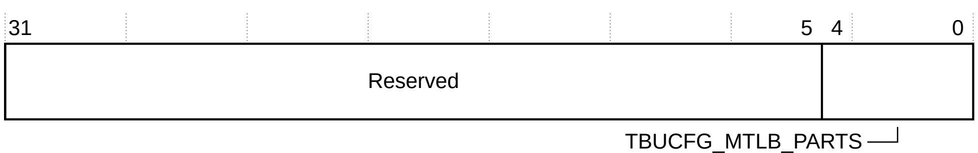


Table 4-77: TBU_SYSDISC9 register bit descriptions

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`

4.15.11 TBU_SYSDISC10 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC10 register is available in all configurations.

Attributes

The TBU_SYSDISC10 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E58`

Type

RO

Reset value

`TBUCFG_LTI_OG_WIDTH`. See 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-51: TBU_SYSDISC10 register bit assignments

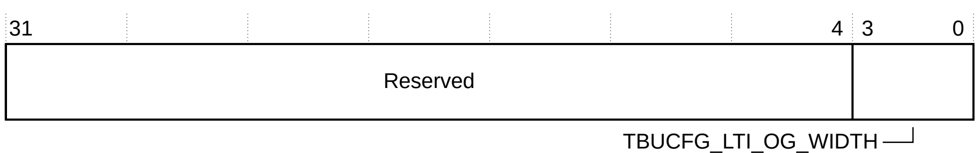


Table 4-78: TBU_SYSDISC10 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved

[3:0]	TBUCFG_LTI_OG_WIDTH	The read data reflects the chosen parameter value, for example:
		`3'h00 : 0`
		....
		`3'h20 : 05`

4.15.12 TBU_SYSDISC11 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC11 register is available in all configurations.

Attributes

The TBU_SYSDISC11 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E5C`

Type

RO

Reset value

`TBUCFG_PARTID_WIDTH`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-52: TBU_SYSDISC11 register bit assignments

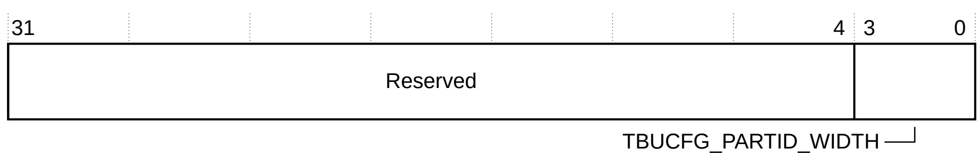


Table 4-79: TBU_SYSDISC11 register bit descriptions

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`

4.15.13 TBU_SYSDISC12 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC12 register is available in all configurations.

Attributes

The TBU_SYSDISC12 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E60`

Type

RO

Reset value

`TBUCFG_HZRD_ENTRIES`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-53: TBU_SYSDISC12 register bit assignments

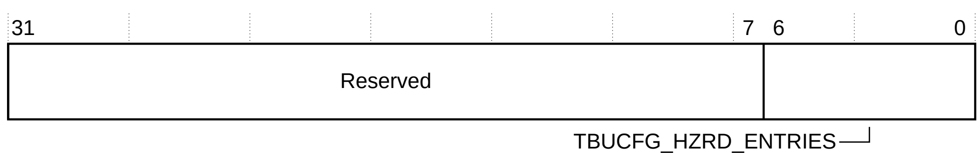


Table 4-80: TBU_SYSDISC12 register bit descriptions

Bits	Name	Description
[31:7]	-	Reserved

[6:0]	TBUCFG_HZRD_ENTRIES	The read data reflects the chosen parameter value, for example:
		`7'h00 : 0`
		....
		`7'h40 : 64`

4.15.14 TBU_SYSDISC13 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC13 register is available in all configurations.

Attributes

The TBU_SYSDISC13 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E64`

Type

RO

Reset value

`TBUCFG_SLOTRAM_TYPE`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-54: TBU_SYSDISC13 register bit assignments

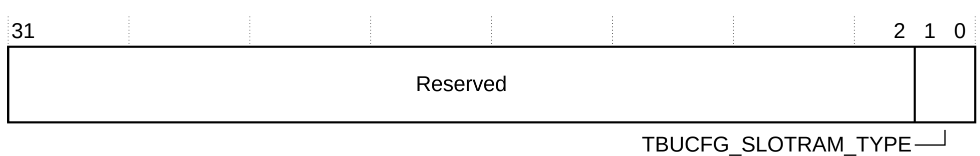


Table 4-81: TBU_SYSDISC13 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved

[1:0]	TBUCFG_SLOTRAM_TYPE	The read data reflects the chosen parameter value, for example:
		`2'b00 : 0`
		....
		`2'b01 : 1`

4.15.15 TBU_SYSDISC14 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

The TBU_SYSDISC14 register is available in all configurations.

Attributes

The TBU_SYSDISC14 register attributes are as follows:

Width

32-bit

Functional group

4.3.10 TBU system discovery register summary on page 106

Address offset

`0x08E68`

Type

RO

Reset value

`TBUCFG_CACHERAM_TYPE`. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-55: TBU_SYSDISC14 register bit assignments

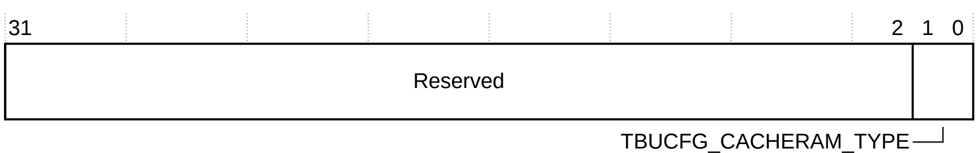


Table 4-82: TBU_SYSDISC14 register bit descriptions

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`

4.16 TBU integration registers

This section describes the TBU integration registers.

4.16.1 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. Controlling these output pins helps system integrators to integrate the SMMU into the system and perform basic connectivity checks.

Configurations

The ITEN register is available in all configurations.

Attributes

The ITEN register attributes are as follows:

Width

32-bit

Functional group

4.3.11 TBU integration register summary on page 106

Address offset

`0x08E20`

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-56: ITEN register bit assignments

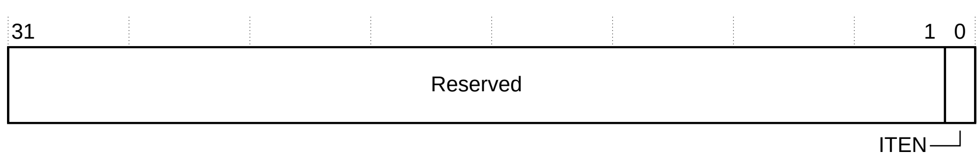


Table 4-83: ITEN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	ITEN	**0** Integration mode is disabled **1** Integration mode is enabled

4.16.2 ITOP_TBU register

This section describes the ITOP register for the TBU.

Configurations

The ITOP_TBU register is available in all configurations.

Attributes

The ITOP_TBU register attributes are as follows:

Width

32-bit

Functional group

4.3.11 TBU integration register summary on page 106

Address offset

`0x08E24`

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-57: ITOP_TBU register bit assignments

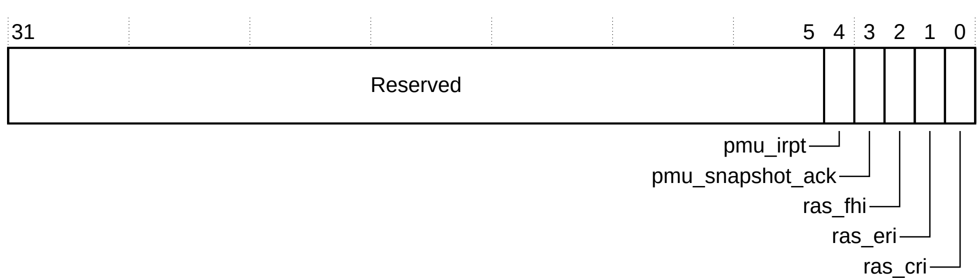


Table 4-84: ITOP_TBU register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, SBZ
[4]	pmu_irpt	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[3]	pmu_snapshot_ack	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[2]	ras_fhi	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[1]	ras_eri	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified
[0]	ras_cri	• When ITEN.ITEN == 0, the register value Should-Be-Zero • When ITEN.ITEN == 1, the register value can be 0 or 1 and this value is also driven to the output signal that is specified

4.16.3 ITIN_TBU register

This section describes the ITIN register for the TBU.

Configurations

The ITIN_TBU register is available in all configurations.

Attributes

The ITIN_TBU register attributes are as follows:

Width

32-bit

Functional group

4.3.11 TBU integration register summary on page 106

Address offset

`0x08E28`

Type

RO

Reset value

0

Bit descriptions

The following figure and table show the bit assignments and bit descriptions.

Figure 4-58: ITIN_TBU register bit assignments

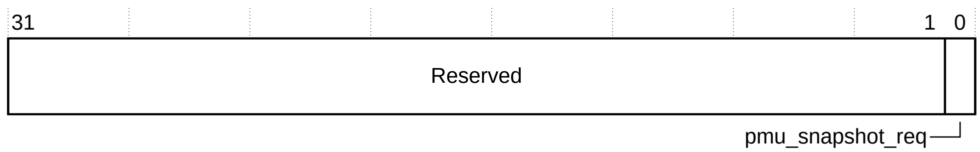


Table 4-85: ITIN_TBU register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved, SBZ
[0]	pmu_snapshot_req	• When ITEN.ITEN == 0, the register value Should-Be-Zero. • When ITEN.ITEN == 1, the register value can be 0 or 1, depending on the value of the input signal that is specified

Appendix A Signal descriptions for MMU-700

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.

Signal definitions

Table A-1: Clock and reset signals

Signal	Direction	Description
clk	Input	Global clock. Width is 1-bit.
resetn	Input	Global reset. Width is 1-bit.

A.1.2 TCU QTW/DVM interface signals

The TCU QTW/DVM interface signals are based on the AMBA ACE5-Lite signals.

Signal definitions

Table A-2: TCU QTW/DVM interface signals

Signal	Direction	Description
acaddr_qtw	Input	Snoop address. Width is 52-bit.
acprot_qtw	Input	Snoop protection type. Width is 3-bit.
acready_qtw	Output	Snoop address ready. Width is 1-bit.
acsnoop_qtw	Input	Snoop transaction type. Width is 4-bit.
acvalid_qtw	Input	Snoop address valid. Width is 1-bit.
arid_qtw	Output	Read address ID The width of arid_qtw is equal to `QTW_ID_WIDTH`. `QTW_ID_WIDTH` is calculated as follows: (($\log_2$(`TCUCFG_PTW_SLOTS`) + 2) > 4) ? ($\log_2$(`TCUCFG_PTW_SLOTS`) + 2) : 4; See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.
araddr_qtw	Output	Read address. Width is 52-bit.

Signal	Direction	Description
arburst_qtw	Output	Burst type. Width is 2-bit.
arcache_qtw	Output	Memory type. Width is 4-bit.
ardomain_qtw	Output	Shareability domain. Width is 2-bit.
aridunq_qtw	Output	Width is 1-bit.
arlen_qtw	Output	Burst length. Width is 8-bit.
arlock_qtw	Output	Lock type. Width is 1-bit.
armpam_qtw	Output	Width is 11-bit.
arprot_qtw	Output	Protection type. Width is 3-bit.
arqos_qtw	Output	QoS identifier. Width is 4-bit.
arready_qtw	Input	Read address ready. Width is 1-bit.
arsize_qtw	Output	Burst size. Width is 3-bit.
arsnoop_qtw	Output	Transaction type. Width is 4-bit.
aruser_qtw	Output	Hardware attribute information. Width is 4-bit.
arvalid_qtw	Output	Read address valid. Width is 1-bit.
awid_qtw	Output	Write address ID Width is `QTW_ID_WIDTH`-bit. `QTW_ID_WIDTH` is calculated as follows: (($\log_2$(`TCUCFG_PTW_SLOTS`) + 2) > 4) ? ($\log_2$(`TCUCFG_PTW_SLOTS`) + 2) : 4; See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.
awaddr_qtw	Output	Write address. Width is 52-bit.
awatop_qtw	Output	Width is 6-bit.
awburst_qtw	Output	Burst type. Width is 2-bit.
awcache_qtw	Output	Memory type. Width is 4-bit.
awdomain_qtw	Output	Shareability domain. Width is 2-bit.
awidunq_qtw	Output	Width is 1-bit.
awlen_qtw	Output	Burst length. Width is 8-bit.
awlock_qtw	Output	Lock type. Width is 1-bit.
awmpam_qtw	Output	Width is 11-bit.
awprot_qtw	Output	Protection type. Width is 3-bit.
awqos_qtw	Output	QoS identifier. Width is 4-bit.
awready_qtw	Input	Write address ready. Width is 1-bit.
awsize_qtw	Output	Burst size. Width is 3-bit.
awsnoop_qtw	Output	Transaction type. Width is 4-bit.
awuser_qtw	Output	Hardware attribute information. Width is 4-bit.
awvalid_qtw	Output	Write address valid. Width is 1-bit.
crready_qtw	Input	Snoop response ready. Width is 1-bit.
crresp_qtw	Output	Snoop response. Width is 5-bit.
crvalid_qtw	Output	Snoop response valid. Width is 1-bit.

Signal	Direction	Description
rdata_qtw	Input	Read data. The width of rdata_qtw is equal to TCUCFG_QTW_DATA_WIDTH See 3.4.1 Translation Control Unit I/O configuration parameters on page 80.
rid_qtw	Input	Read data ID Width is QTW_ID_WIDTH-bit. QTW_ID_WIDTH is calculated as follows: (($\log_2$(TCUCFG_PTW_SLOTS) + 2) > 4) ? ($\log_2$(TCUCFG_PTW_SLOTS) + 2) : 4; See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.
ridunq_qtw	Input	Width is 1-bit.
rlast_qtw	Input	Read last. Width is 1-bit.
rpoison_qtw	Input	Read poison input to the TCU. Width is 8-bit.
rready_qtw	Output	Read ready. Width is 1-bit.
rresp_qtw	Input	Read response. Width is 2-bit.
rvalid_qtw	Input	Read valid. Width is 1-bit.
wdata_qtw	Output	Write data The width of wdata_qtw is equal to TCUCFG_QTW_DATA_WIDTH-bit. See 3.4.1 Translation Control Unit I/O configuration parameters on page 80.
wlast_qtw	Output	Write last. Width is 1-bit.
wpoison_qtw	Output	Width is (TCUCFG_QTW_DATA_WIDTH/64)-bit.
wready_qtw	Input	Write ready. Width is 1-bit.
wstrb_qtw	Output	Write strobe The width of wstrb_qtw is calculated as (TCUCFG_QTW_DATA_WIDTH/8)-bit. See 3.4.1 Translation Control Unit I/O configuration parameters on page 80.
wvalid_qtw	Output	Write valid. Width is 1-bit.
bid_qtw	Input	Response ID Width is QTW_ID_WIDTH-bit. QTW_ID_WIDTH is calculated as follows: (($\log_2$(TCUCFG_PTW_SLOTS) + 2) > 4) ? ($\log_2$(TCUCFG_PTW_SLOTS) + 2) : 4; See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.
bidunq_qtw	Input	Width is 1-bit.
bready_qtw	Output	Response ready. Width is 1-bit.
bresp_qtw	Input	Write response. Width is 2-bit.
bvalid_qtw	Input	Write response valid. Width is 1-bit.
awakeup_qtw	Output	Wakeup. Width is 1-bit.
acwakeup_qtw	Input	Snoop wakeup. Width is 1-bit.
acvmidext_qtw	Input	Snoop Extended *Virtual Machine IDentifier* (VMID). Width is 4-bit.
syscoreq_qtw	Output	Width is 1-bit.

Signal	Direction	Description
syscoack_qtw	Input	Width is 1-bit.

A.1.3 TCU programming interface signals

The TCU programming interface signals are based on the AMBA APB4 signals.

Signal definitions

Table A-3: TCU programming interface signals

Signal	Direction	Description
paddr_prog	Input	Peripheral address. The width of paddr_prog is either 21-bit or 23-bit. If `TCUCFG_NUM_TBU` is 62, the width of paddr_prog is 23-bit. Otherwise, the width of paddr_prog is 21-bit. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.
psel_prog	Input	Peripheral select Width is 1-bit.
penable_prog	Input	Enable for transfer Width is 1-bit.
pwrite_prog	Input	Write transaction indicator Width is 1-bit.
pprot_prog	Input	Protection type Width is 3-bit.
pwdata_prog	Input	Write data Width is 32-bit.
pstrb_prog	Input	Write data strobe Width is 4-bit.
pslverr_prog	Output	Error response Width is 1-bit.
prdata_prog	Output	Read data Width is 32-bit.
pready_prog	Output	Transfer ready Width is 1-bit.
pwakeup_prog	Input	Interface wakeup Width is 1-bit.

A.1.4 TCU SYSCO interface signals

The following table shows the TCU SYSCO interface signals.

Signal definitions

Table A-4: TCU SYSCO interface signals

Signal	Direction	Description
syscoreq_qtw	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. Width is 1-bit.
syscoack_qtw	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. Width is 1-bit.

A.1.5 TCU PMU snapshot interface signals

The following table shows the TCU PMU snapshot interface signals.

Signal definitions

Table A-5: TCU PMU snapshot interface signals

Signal	Direction	Description
pmusnapshot_req	Input	PMU snapshot request. The PMU snapshot occurs on the rising edge of pmusnapshot_req. **Note:** Connect to the debug infrastructure of your SoC. Width is 1-bit.
pmusnapshot_ack	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. Width is 1-bit.

A.1.6 TCU LPI_PD interface signals

The following table shows the TCU LPI_PD interface signals.

Signal definitions

Table A-6: TCU LPI_PD interface signals

Signal	Direction	Description
qactive_pd	Output	Component active. Width is 1-bit.
qreqn_pd	Input	Quiescence request. Width is 1-bit.
qacceptn_pd	Output	Quiescence accept. Width is 1-bit.
qdeny_pd	Output	Quiescence deny. Width is 1-bit.

A.1.7 TCU LPI_CG interface signals

The following table shows the TCU LPI_CG interface signals.

Signal definitions

Table A-7: TCU LPI_CG interface signals

Signal	Direction	Description
qactive_cg	Output	Component active. Width is 1-bit.
qreqn_cg	Input	Quiescence request. Width is 1-bit.
qacceptn_cg	Output	Quiescence accept. Width is 1-bit.
qdeny_cg	Output	Quiescence deny. Width is 1-bit.

A.1.8 TCU DTI interface signals

The following table shows the TCU DTI interface signals.

In the following table, the 'Direction' has the following meaning:

Input

Slave to master

Output

Master to slave

Signal definitions

Table A-8: TCU DTI interface signals

Signal	Direction	Description
tvalid_dti_dn	Output	Flow control signal. Width is 1-bit.
tready_dti_dn	Input	Flow control signal. Width is 1-bit.
tdata_dti_dn	Output	Message data signal. Width is 160-bit.
tid_dti_dn	Output	Identifies the master that initiated the message. Width is 4-bit or 6-bit. The width of tid_dti_dn is calculated as follows: If `TCUCFG_NUM_TBU` is 62, the width of tid_dti_dn is 6-bit. Otherise, the width of tid_dti_dn is 4-bit. See 3.4.2 Translation Control Unit buffer configuration parameters on page 81.
tlast_dti_dn	Output	Indicates the last cycle of a message. Width is 1-bit
tkeep_dti_dn	Output	This signal indicates valid bytes. Width is 20-bit
tvalid_dti_up	Input	Flow control signal. Width is 1-bit
tready_dti_up	Output	Flow control signal. Width is 1-bit
tdata_dti_up	Input	Message data signal. Width is 160-bit
tdest_dti_up	Input	Identifies the master that is receiving the message. Width is 4-bit or 6-bit. 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 3.4.2 Translation Control Unit buffer configuration parameters on page 81.
tlast_dti_up	Input	Indicates the last cycle of a message. Width is 1-bit.

Signal	Direction	Description
tkeep_dti_up	Input	Indicates valid bytes. Width is 20-bit.
twakeup_dti_up	Input	Wakeup signal. Width is 1-bit.
twakeup_dti_dn	Output	Wakeup signal. Width is 1-bit.

For more information about the DTI signals, see the *AMBA® AXI-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 following as *Message Signaled Interrupts* (MSIs) on the QTW/DVM interface and the dedicated MSI delivery interface:

- Secure and Non-secure Event queue
- SYNC complete commands
- Global interrupts

If the system supports capturing MSIs from the TCU, there is no requirement to connect the corresponding interrupt signals in this interface.

Signal definitions

Table A-9: TCU interrupt interface signals

Signal	Direction	Description
event_q_irpt_s	Output	Event queue, Secure interrupt. The event_q_irpt_s signal asserts a Secure interrupt to indicate that the Event queue is not empty. Width is 1-bit.
event_q_irpt_ns	Output	Event queue, Non-secure interrupt. The event_q_irpt_ns signal asserts a Non-secure interrupt to indicate that the Event queue is not empty. Width is 1-bit.
cmd_sync_irpt_ns	Output	SYNC complete, Non-secure interrupt. The cmd_sync_irpt_ns signal asserts a Non-secure interrupt to indicate that the `CMD_SYNC` command is complete. Width is 1-bit.

Signal	Direction	Description
cmd_sync_irpt_s	Output	SYNC complete, Secure interrupt. The cmd_sync_irpt_s signal asserts a Secure interrupt to indicate that the `CMD_SYNC` command is complete. Width is 1-bit.
global_irpt_ns	Output	The global_irpt_ns signal asserts a global Non-secure interrupt. Width is 1-bit.
global_irpt_s	Output	The global_irpt_s signal asserts a global Secure interrupt. Width is 1-bit.
ras_fhi	Output	Fault handling RAS interrupt for a contained or an uncontained error. **Note:** TCU_ERRCTLR.FI can also enable or disable ras_fhi. See 4.7.2 TCU_ERRCTLR register on page 123. Width is 1-bit. **Note:** The MMU-700 cannot output this interrupt as an MSI.
ras_eri	Output	Error recovery RAS interrupt for an uncontained error. Width is 1-bit. **Note:** The MMU-700 cannot output this interrupt as an MSI.
ras_cri	Output	Critical error interrupt for an uncontainable uncorrected error. Width is 1-bit. **Note:** The MMU-700 cannot output this interrupt as an MSI.
pmu_irpt	Output	The pmu_irpt signal asserts a PMU interrupt. Width is 1-bit. **Note:** The MMU-700 cannot output PMU interrupts as MSIs.
pri_q_irpt_ns	Output	Asserts a *Page Request Interface* (PRI) queue interrupt. Width is 1-bit.

A.1.10 TCU Message Signaled Interrupt interface signals

This section describes the TCU *Message Signaled Interrupt* (MSI) interface.

The interface follows the Arm® AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled) protocol and uses the signals in the following table to send MSIs.

Signal definitions

Table A-10: TCU MSI interface signals

Signal	Direction	Description	Connection information
msitvalid	Output	Indicates valid data to the GIC. Width is 1-bit.	AXI-Stream signal is TVALID
msitready	Input	Indicates acceptance by the GIC. Width is 1-bit.	AXI-Stream signal is TREADY
msitdata	Output	Data being passed to the GIC. Width is 64-bit.	AXI-Stream signal is TDATA
msitwakeup	Output	Indicates that a transaction is ongoing. Width is 1-bit.	AXI-Stream signal is TWAKEUP, AMBA extension
msirtvalid	Input	Indicates that the GIC has accepted an MSI. Width is 1-bit.	AXI-Stream signal is TVALID
msirtready	Output	Indicates that the device has accepted the response packet. Width is 1-bit.	AXI-Stream signal is TREADY
msirtwakeup	Input	Indicates that a transaction is ongoing. Width is 1-bit.	AXI-Stream signal is TWAKEUP, AMBA extension

For more information about these signals, see the *Arm® GIC MSI Delivery Interface* document.

A.1.11 TCU event interface signals

The TCU event interface signal is an event output for connection to processors.

Signal definitions

Table A-11: TCU event interface signals

Signal	Direction	Description
evento	Output	Width is 1-bit. The evento signal is asserted for one cycle to indicate an event that enables processors to wake up from the *Wait For Event* (WFE) low-power state. Connect the evento signal of the TCU to the event interface of Arm® processors. Processors that use the *DynamIQ Shared Unit* (DSU) have a different event handshake mechanism. The mechanism that the DSU uses is the successor to the mechanism that some MMUs use. Arm® processors can use the following event mechanisms: • 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 **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. In both mechanisms, in the signal names: **i** Represents events that are inputs to a particular component **o** Represents events that are outputs from a particular component **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. The required conversion is from the older mechanism, eventi and evento signals, to the newer mechanism, eventiack, eventireq, eventoack, and eventoreq signals. When connecting the MMU-700 to a DSU, the only signals to consider are the following: • evento signal of the MMU-700 • eventiack and eventireq signals of the DSU

Signal	Direction	Description
evento continued	Output	Some processors have an eventi input instead. You can use the *Event Pulse to Event adapter* 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*. **Note:** To use the *Event Pulse to Event adapter* 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 *Arm® CoreLink™ CMN-600AE Event Interface Connections Application Note*.

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.

Signal definitions

Table A-12: TCU tie-off signals

Signal	Direction	Description
sup_cohacc	Input	This signal indicates whether the QTW interface is I/O-coherent. Tie HIGH when the TCU is connected to a coherent interconnect. Width is 1-bit.
sup_btm	Input	This signal indicates whether the Broadcast TLB Maintenance is supported. Tie HIGH when the TCU is connected to an interconnect that supports DVM. Width is 1-bit.
sup_sev	Input	This signal indicates whether the Send Event mechanism is supported. Tie HIGH when evento is connected. Width is 1-bit.
sup_oas[2:0]	Input	Output address size supported. The encodings for this input are as follows: `0b000` 32 bits `0b001` 36 bits `0b010` 40 bits `0b011` 42 bits `0b100` 44 bits `0b101` 48 bits `0b110` 52 bits You must not use other encodings. Width is 3-bit.

<table>
<tr><th>Signal</th><th>Direction</th><th>Description</th></tr>
<tr><td>sec_override</td><td>Input</td><td>When HIGH, certain registers are accessible to Non-secure accesses from reset, as the 4.6.7 TCU_SCR register on page 119 settings describe

Width is 1-bit.</td></tr>
<tr><td>ecorevnum[3:0]</td><td>Input</td><td>Tie this signal to 0 unless directed otherwise by Arm

Width is 4-bit.</td></tr>
<tr><td>msi_addr[51:0]</td><td>Input</td><td>If the programmed Message Signaled Interrupt (MSI) address in SMMU_(S_)_*_IRQ_CFG0.ADDR matches msi_addr, then an MSI is generated on the TCU MSI interface

Width is 52-bit.</td></tr>
<tr><td>tcu_sid[31:0]</td><td>Input</td><td>Used as the DeviceID for TCU-generated MSIs.

Note:
This is only for MSIs that are issued from the dedicated AXI5-Stream (with Wakeup_Signal enabled and Check_Type not enabled) MSI delivery interface.

Width is 32-bit.</td></tr>
<tr><td>sup_httu</td><td>Input</td><td>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 Hardware Translation Table Update (HTTU) transactions.
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.

The impact of sup_httu on SMMU_IDR0.HTTU is as follows:
sup_httu is 1'b0
SMMU_IDR0.HTTU is 2'b00
sup_httu is 1'b1
SMMU_IDR0.HTTU is 2'b10

See 3.3.1 SMMUv3 implementation on page 60.

Width is 1-bit.</td></tr>
</table>

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.

Signal definitions

Table A-13: ELA enable signals

Signal	Direction	Description
ela_enable	Input	ela_enable is an asynchronous input port. When `TCUCFG_USE_ELA_DEBUG` is 0, the SMMU ignores the value of the signal. When `TCUCFG_USE_ELA_DEBUG` 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. Width is 1-bit.

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.

Signal definitions

Table A-14: Clock and reset signals

Signal	Direction	Description
clk	Input	Global clock Width is 1-bit.
resetn	Input	Global reset Width is 1-bit.

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.

Signal definitions

Table A-15: TBU TBS interface signals

Signal	Direction	Description
araddr_s	Input	Read address. Width is 64-bit.

Signal	Direction	Description
arburst_s	Input	Burst type. Width is 2-bit.
arcache_s	Input	Memory type. Width is 4-bit.
ardomain_s	Input	Shareability domain. Width is 2-bit.
arid_s	Input	Read address ID The width of arid_s is `TBUCFG_ID_WIDTH`-bit. For information about how to set the `TBUCFG_ID_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
aridunq_s	Input	Read address channel unique ID indicator, active-HIGH. Width is 1-bit.
arlen_s	Input	Burst length. Width is 8-bit.
arlock_s	Input	Lock type. Width is 1-bit.
arloop_s	Input	Loopback value for a read transaction. Reflected back on RLOOP. The width of arloop_s is `TBUCFG_LOOP_WIDTH`-bit. For information about how to set the `TBUCFG_LOOP_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
armmuflow_s	Input	Indicates the SMMU flow for managing translation faults. Width is 2-bit.
armmussid_s	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. For information about how to set these parameters, see 3.1.2.2 ACE-Lite TBU TBM interface on page 32 and 3.3.2 AMBA implementation on page 64. The width of armmusid_s is `TBUCFG_SSID_WIDTH`-bit. For information about how to set the `TBUCFG_SSID_WIDTH` parameter, see 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.
armmusid_s	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. For information about how to set these parameters, see 3.1.2.2 ACE-Lite TBU TBM interface on page 32 and 3.3.2 AMBA implementation on page 64. The width of armmusid_s is `TBUCFG_SSID_WIDTH`-bit. For information about how to set the `TBUCFG_SSID_WIDTH` parameter, see 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.

Signal	Direction	Description
armmussidv_s	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. For information about how to set these parameters, see 3.1.2.2 ACE-Lite TBU TBM interface on page 32 and 3.3.2 AMBA implementation on page 64. Width is 1-bit.
armmusecsid_s	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. For information about how to set these parameters, see 3.1.2.2 ACE-Lite TBU TBM interface on page 32 and 3.3.2 AMBA implementation on page 64. Width is 1-bit.
arprot_s	Input	Protection type. Width is 3-bit.
arqos_s	Input	*Quality of Service* (QoS). Width is 4-bit.
arready_s	Output	Read address ready. Width is 1-bit.
arregion_s	Input	Region identifier. Width is 4-bit.
arsize_s	Input	Burst size. Width is 3-bit.
arsnoop_s	Input	Transaction type of read transaction. Width is 4-bit.
aruser_s	Input	Read address (AR) channel User signal Calculate the width of aruser_s as follows: (`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_MTLB_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` For information about how to set these parameters, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.

Signal	Direction	Description
arvalid_s	Input	Read address valid. Width is 1-bit.
rchunknum_s	Output	Read data chunk number. Width is `CHUNKNUM_WIDTH`-bit.
rchunkstrb_s	Output	Read data chunk strobe. Width is `CHUNKSTRB_WIDTH`-bit.
rchunkv_s	Output	Valid signal of RCHUNKNUM and RCHUNKSTRB Width is 1-bit.
rdata_s	Output	Read data The width of rdata_s is `TBUCFG_DATA_WIDTH`-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
rid_s	Output	Read ID The width of rid_s is `TBUCFG_ID_WIDTH`-bit. For information about how to set the `TBUCFG_ID_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
ridunq_s	Output	Read data channel unique ID indicator, active-HIGH. Width is 1-bit.
rlast_s	Output	Read last. Width is 1-bit.
rloop_s	Output	Loopback value for a read response The width of rloop_s is `TBUCFG_LOOP_WIDTH`-bit. For information about how to set the `TBUCFG_LOOP_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
rpoison_s	Output	Indicates that the read data in this transfer has been corrupted The width of rpoison_s is (`TBUCFG_DATA_WIDTH` / 64)-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
rready_s	Input	Read ready. Width is 1-bit.
rresp_s	Output	Read response. Width is 3-bit.
ruser_s	Output	Read data (R) channel User signal The width if ruser_s is `TBUCFG_RUSER_WIDTH`-bit. For information about how to set the `TBUCFG_RUSER_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.

Signal	Direction	Description
rvalid_s	Output	Read valid. Width is 1-bit.
awaddr_s	Input	Write address. Width is 64-bit.
awatop_s	Input	Atomic operation. Width is 6-bit.
awburst_s	Input	Burst type. Width is 2-bit.
awcache_s	Input	Memory type. Width is 4-bit.
awdomain_s	Input	Shareability domain. Width is 2-bit.
awid_s	Input	Write address ID The width of awid_s is `TBUCFG_ID_WIDTH`-bit. For information about how to set the `TBUCFG_ID_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
awlen_s	Input	Burst length. Width is 8-bit.
awlock_s	Input	Lock type. Width is 1-bit.
awmmuflow_s	Input	Indicates the SMMU flow for managing translation faults. Width is 2-bit.
awmmussid_s	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. For information about how to set these parameters, see 3.1.2.2 ACE-Lite TBU TBM interface on page 32 and 3.3.2 AMBA implementation on page 64. The width of awmmussid_s is `TBUCFG_SSID_WIDTH`-bit. For information about how to set the `TBUCFG_SSID_WIDTH` parameter, see 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.
awmmusid_s	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. For information about how to set these parameters, see 3.1.2.2 ACE-Lite TBU TBM interface on page 32 and 3.3.2 AMBA implementation on page 64. The width of awmmusid_s is `TBUCFG_SID_WIDTH`-bit. For information about how to set the `TBUCFG_SID_WIDTH` parameter, see 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.

Signal	Direction	Description
awmmussidv_s	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. For information about how to set these parameters, see 3.1.2.2 ACE-Lite TBU TBM interface on page 32 and 3.3.2 AMBA implementation on page 64. Width is 1-bit.
awmmusecsid_s	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. The AXI5 Untranslated_Transactions extension defines these signals. For information about how to set these parameters, see 3.1.2.2 ACE-Lite TBU TBM interface on page 32 and 3.3.2 AMBA implementation on page 64. Width is 1-bit.
awprot_s	Input	Protection type. Width is 3-bit.
awqos_s	Input	QoS. Width is 4-bit.
awready_s	Output	Write address ready. Width is 1-bit.
awregion_s	Input	Region identifier. Width is 4-bit.
awsize_s	Input	Burst size. Width is 3-bit.
awvalid_s	Input	Write address valid. Width is 1-bit.
awuser_s	Input	Write address (AW) channel User signal The width of awuser_s is (`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_MTLB_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` For information about how to set these parameters, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
awakeup_s	Input	Wakeup signal. Width is 1-bit.

Signal	Direction	Description
awsnoop_s[3]	Input	Transaction type of write transaction. Width is 4-bit.
awstashnid_s[10:0]	Input	The AXI5 Cache_Stash_Transactions extension defines the awstashnid_s[10:0] signal. If `TBUCFG_STASH_SUPPORT` = 0, the awstashnid_s[10:0] signal is ignored. Width is 11-bit.
awstashniden_s	Input	The AXI5 Cache_Stash_Transactions extension defines the awstashniden_s signal. If `TBUCFG_STASH_SUPPORT` = 0, the awstashniden_s signal is ignored. Width is 1-bit.
awstashlpid_s[4:0]	Input	The AXI5 Cache_Stash_Transactions extension defines the awstashlpid_s[4:0] signal. If `TBUCFG_STASH_SUPPORT` = 0, the awstashlpid_s[4:0] signal is ignored. Width is 5-bit.
awstashlpiden_s	Input	The AXI5 Cache_Stash_Transactions extension defines the awstashlpiden_s signal. If `TBUCFG_STASH_SUPPORT` = 0, the awstashlpiden_s signal is ignored. Width is 1-bit.
archunken_s	Input	Read data chunking enable. Width is 1-bit.
awidunq_s	Input	Write address channel unique ID indicator, active-HIGH. Width is 1-bit.
awloop_s	Input	Loopback value for a write transaction The width of awloop_s is `TBUCFG_LOOP_WIDTH`-bit. For information about how to set the `TBUCFG_LOOP_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
wdata_s	Input	Write data The width of wdata_s is `TBUCFG_DATA_WIDTH`-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
wlast_s	Input	Write last. Width is 1-bit.
wpoison_s	Input	Indicates that the write data in this transfer has been corrupted The width of wpoison_s is [(`TBUCFG_DATA_WIDTH` / 64)-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
wready_s	Output	Write ready. Width is 1-bit.

Signal	Direction	Description
wstrb_s	Input	Write strobes The width of wstrb_s is (TBUCFG_DATA_WIDTH / 8)-bit. For information about how to set the TBUCFG_DATA_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
wuser_s	Input	Write data (W) channel User signal The width of wuser_s is TBUCFG_WUSER_WIDTH-bit. For information about how to set the TBUCFG_WUSER_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
wvalid_s	Input	Write valid. Width is 1-bit.
bid_s	Output	Response ID The width of bid_s is TBUCFG_ID_WIDTH-bit. For information about how to set the TBUCFG_ID_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
bidunq_s	Output	Write response channel unique ID indicator, active-HIGH. Width is 1-bit.
bloop_s	Output	Loopback value for a write response The width of bloop_s is TBUCFG_LOOP_WIDTH-bit. For information about how to set the TBUCFG_LOOP_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
bready_s	Input	Response ready. Width is 1-bit.
bresp_s	Output	Write response. Width is 3-bit.
buser_s	Output	Write response (B) channel User signal The width of buser_s is TBUCFG_BUSER_WIDTH-bit. For information about how to set the TBUCFG_BUSER_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
bvalid_s	Output	Write response valid. Width is 1-bit.

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.

Signal definitions

Table A-16: TBU TBM interface signals

Signal	Direction	Description
araddr_m	Output	Read address. Width is 52-bit.
arburst_m	Output	Burst type. Width is 2-bit.
arcache_m	Output	Memory type. Width is 4-bit.
archunken_m	Output	Read data chunking enable. Width is 1-bit.
ardomain_m	Output	Shareability domain. Width is 2-bit.
arid_m	Output	Read address ID The width of arid_m is `TBUCFG_ID_WIDTH`-bit. For information about how to set the `TBUCFG_ID_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
aridunq_m	Output	Read address channel unique ID indicator, active-HIGH. Width is 1-bit.
arlen_m	Output	Burst length. Width is 8-bit.
arlock_m	Output	Lock type. Width is 1-bit.
arloop_m	Output	Loopback value for a read transaction. Reflected back on RLOOP Calculate the width of arloop_m as follows: 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. For information about how to set these parameters, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
armmusid_m	Output	Indicates the StreamID of the originating transaction. The width of armmusid_m is `TBUCFG_SID_WIDTH`-bit. See 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.

Signal	Direction	Description
armmusecsid_m	Output	Indicates the StreamID of the originating transaction. Width is 1-bit.
armpam_m	Output	Read address channel MPAM information. Width is 11-bit.
arprot_m	Output	Protection type. Width is 3-bit.
arqos_m	Output	*Quality of Service* (QoS). Width is 4-bit.
arready_m	Input	Read address ready. Width is 1-bit.
arregion_m	Output	Region identifier. Width is 4-bit.
arsize_m	Output	Burst size. Width is 3-bit.
arsnoop_m	Output	Transaction type of read transaction. Width is 4-bit.
aruser_m	Output	Read address (AR) channel User signal The width of aruser_m is (`TBUCFG_ARUSER_WIDTH` + 5)-bit. For information about how to set the `TBUCFG_ARUSER_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
arvalid_m	Output	Read address valid. Width is 1-bit.
rchunknum_m	Input	Read data chunk number. Width can be 1-bit, 5-bit, 6-bit, 7-bit, or 8-bit. Calculate the width of rchunknum_m as follows: 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. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.

Signal	Direction	Description
rchunkstrb_m	Input	Read data chunk strobe Calculate the width of rchunkstrb_m as follows: 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. For information about how to set the TBUCFG_DATA_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
rchunkv_m	Input	Valid signal of RCHUNKNUM and RCHUNKSTRB. Width is 1-bit.
rdata_m	Input	Read data The width of rdata_m is TBUCFG_DATA_WIDTH-bit. For information about how to set the TBUCFG_DATA_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
rid_m	Input	Read ID The width of rid_m is TBUCFG_ID_WIDTH-bit. For information about how to set the TBUCFG_ID_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
ridunq_m	Input	Read data channel unique ID indicator, active-HIGH. Width is 1-bit.
rlast_m	Input	Read last. Width is 1-bit.
rloop_m	Input	Loopback value for a read response The width of rloop_m is calculated as follows: 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. For information about how to set these parameters, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
rpoison_m	Input	Indicates that the read data in this transfer has been corrupted The width of rpoison_m is (TBUCFG_DATA_WIDTH / 64)-bit. For information about how to set the TBUCFG_DATA_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
rready_m	Output	Read ready. Width is 1-bit.
rresp_m	Input	Read response. Width is 2-bit.
ruser_m	Input	Read data (R) channel User signal The width of ruser_m is TBUCFG_RUSER_WIDTH-bit. For information about how to set the TBUCFG_RUSER_WIDTH parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.

Signal	Direction	Description
rvalid_m	Input	Read valid. Width is 1-bit.
awaddr_m	Output	Write address. Width is 52-bit.
awatop_m	Output	Atomic operation. Width is 6-bit.
awburst_m	Output	Burst type. Width is 2-bit.
awcache_m	Output	Memory type. Width is 4-bit.
awdomain_m	Output	Shareability domain. Width is 2-bit.
awid_m	Output	Write address ID The width of awid_m is `TBUCFG_ID_WIDTH`-bit. For information about how to set the `TBUCFG_ID_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
awlen_m	Output	Burst length. Width is 8-bit.
awlock_m	Output	Lock type. Width is 1-bit.
awmmusid_m	Output	Indicates the StreamID of the originating transaction. The *Generic Interrupt Controller* (GIC) uses these signals to determine the DeviceID of MSIs that originate from upstream masters. The width of awmmusid_m is `TBUCFG_SID_WIDTH`-bit. For information about how to set the `TBUCFG_SID_WIDTH` parameter, see 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.
awmmusecsid_m	Output	Indicates the StreamID of the originating transaction. The *Generic Interrupt Controller* (GIC) uses these signals to determine the DeviceID of MSIs that originate from upstream masters. Width is 1-bit.
awprot_m	Output	Protection type. Width is 3-bit.
awqos_m	Output	QoS. Width is 4-bit.

Signal	Direction	Description
awready_m	Input	Write address ready. Width is 1-bit.
awregion_m	Output	Region identifier. Width is 4-bit.
awsize_m	Output	Burst size. Width is 3-bit.
awstashnid_m	Output	The AXI5 Cache_Stash_Transactions extension defines this signal. See *AMBA® AXI and ACE Protocol Specification*. If `TBUCFG_STASH_SUPPORT` = 0, these signals are ignored. Width is 11-bit.
awstashniden_m	Output	The AXI5 Cache_Stash_Transactions extension defines this signal. See *AMBA® AXI and ACE Protocol Specification*. If `TBUCFG_STASH_SUPPORT` = 0, these signals are ignored. Width is 1-bit.
awstashlpid_m	Output	The AXI5 Cache_Stash_Transactions extension defines this signal. See *AMBA® AXI and ACE Protocol Specification*. If `TBUCFG_STASH_SUPPORT` = 0, these signals are ignored. Width is 5-bit.
awstashlpiden_m	Output	The AXI5 Cache_Stash_Transactions extension defines this signal. See *AMBA® AXI and ACE Protocol Specification*. If `TBUCFG_STASH_SUPPORT` = 0, these signals are ignored. Width is 1-bit.
awakeup_m	Output	Wakeup signal. Width is 1-bit.
awidunq_m	Output	Write address channel unique ID indicator, active-HIGH. Width is 1-bit.
awloop_m	Output	Loopback value for a write transaction The width of awloop_m is calculated as follows: 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. For information about how to set these parameters, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
awmmuflow_s	Input	Indicates the SMMU flow for managing translation faults. Width is 2-bit.

Signal	Direction	Description
awmpam_m	Output	Write address channel MPAM information. Width is 11-bit.
awsnoop_m[3:0]	Output	Transaction type of write transaction. Width is 4-bit.
awuser_m	Output	Write address (AW) channel User signal The width of awuser_m is (`TBUCFG_AWUSER_WIDTH` + 5)-bit. For information about how to set the `TBUCFG_AWUSER_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
awvalid_m	Output	Write address valid. Width is 1-bit.
wdata_m	Output	Write data The width of wdata_m is `TBUCFG_DATA_WIDTH`-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
wlast_m	Output	Write last. Width is 1-bit.
wpoison_m	Output	Indicates that the write data in this transfer has been corrupted. The width of wpoison_m is (`TBUCFG_DATA_WIDTH` / 64)-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
wready_m	Input	Write ready. Width is 1-bit.
wstrb_m	Output	Write strobes. The width of wstrb_m is (`TBUCFG_DATA_WIDTH` / 8)-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
wvalid_m	Output	Write valid. Width is 1-bit.
wuser_m	Output	Write data (W) channel User signal. The width of wuser_m is `TBUCFG_WUSER_WIDTH`-bit. For information about how to set the `TBUCFG_WUSER_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
bidunq_m	Input	Write response channel unique ID indicator, active-HIGH. Width is 1-bit.
bid_m	Input	Response ID. `TBUCFG_ID_WIDTH`-bit. For information about how to set the `TBUCFG_ID_WIDTH` parameter, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.

Signal	Direction	Description
bloop_m	Input	Loopback value for a write response. 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. For information about how to set these parameters, see 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
bready_m	Output	Response ready. Width is 1-bit.
bresp_m	Input	Write response. Width is 2-bit.
buser_m	Input	Write response (B) channel User signal. The width of buser_m is `TBUCFG_BUSER_WIDTH`-bit. See 3.4.7 ACE-Lite Translation Buffer Unit I/O configuration parameters on page 88.
bvalid_m	Input	Write response valid. Width is 1-bit.

A.2.4 TBU PMU snapshot interface signals

The following table shows the TBU PMU snapshot interface signals.

Signal definitions

Table A-17: TBU PMU snapshot interface signals

Signal	Direction	Description
pmusnapshot_req	Input	PMU snapshot request. The PMU snapshot occurs on the rising edge of pmusnapshot_req. **Note:** Connect to the debug infrastructure of your SoC. Width is 1-bit.
pmusnapshot_ack	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. Width is 1-bit.

A.2.5 TBU LPI_PD interface signals

The following table shows the TBU LPI_PD interface signals.

Signal definitions

Table A-18: TBU LPI_PD interface signals

Signal	Direction	Description
qactive_pd	Output	Component active. Width is 1-bit.
qreqn_pd	Input	Quiescence request. Width is 1-bit.
qacceptn_pd	Output	Quiescence accept. Width is 1-bit.
qdeny_pd	Output	Quiescence deny. Width is 1-bit.

A.2.6 TBU LPI_CG interface signals

The following table shows the TBU LPI_CG interface signals.

Signal definitions

Table A-19: TBU LPI_CG interface signals

Signal	Direction	Description
qactive_cg	Output	Component active. Width is 1-bit.
qreqn_cg	Input	Quiescence request. Width is 1-bit.
qacceptn_cg	Output	Quiescence accept. Width is 1-bit.
qdeny_cg	Output	Quiescence deny. Width is 1-bit.

A.2.7 TBU DTI interface signals

The following table shows the TBU DTI interface signals.

In the following table, the 'Direction' has the following meaning:

Input

Slave to master

Output

Master to slave

Signal definitions

Table A-20: TBU DTI interface signals

Signal	Direction	Description
tvalid_dti_dn	Output	Flow control signal. Width is 1-bit.
tready_dti_dn	Input	Flow control signal. Width is 1-bit.
tdata_dti_dn	Output	Message data signal. Width is 160-bit.
tlast_dti_dn	Output	Indicates the last cycle of a message. Width is 1-bit.
tkeep_dti_dn	Output	Indicates valid bytes. Width is 20-bit.
tvalid_dti_up	Input	Flow control signal. Width is 1-bit.
tready_dti_up	Output	Flow control signal. Width is 1-bit.
tdata_dti_up	Input	Message data signal. Width is 160-bit.
tlast_dti_up	Input	Indicates the last cycle of a message. Width is 1-bit.
tkeep_dti_up	Input	Indicates valid bytes. Width is 20-bit.
twakeup_dti_up	Input	Wakeup signal. Width is 1-bit.
twakeup_dti_dn	Output	Wakeup signal. Width is 1-bit.

For more information about the DTI signals, see the *AMBA® AXI-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. 3.3.4 Local Translation Interface implementation on page 79 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.

The following table shows the LTI request channel signals.

Table A-21: LTI request channel signals

Signal	Category	Description
LAVALID	Transport	Channel valid. When this signal is LOW, other TX signals on the LA channel are not valid. Width is 1-bit.
LAVC	Transport	Virtual Channel number. Width is ceil($\log_2$(`LTI_VC_COUNT`))-bit.
LACREDIT	Transport	Channel credit grant. LACREDIT is an RX signal that flows in the other direction from other LA channel signals. It is not affected by LAVALID. Width is `LTI_VC_COUNT`-bit.
LAID	Flow	Translation request ID. The ID must not match the ID of a previous request on the same VC that has not yet returned its LR channel response, unless LAOGV=1 in both requests and the value of LAOG is the same for both requests. Width is `LTI_ID_WIDTH`.
LAOGV	Flow	LAOGV Order group valid. **0** No ordering required **1** The responses of all requests with this LAOG value must be returned in order The LAOGV signal is present even if LTI_OG_WIDTH=0. In this case, there is a single order group, and LAOGV indicates whether each LTI request is ordered or unordered. Width is 1-bit
LAOG	Flow	Order group. When LAOGV is LOW, this signal is not valid. Width is `LTI_OG_WIDTH`-bit.

Signal	Category	Description
LAFLOW	Flow	Indicates the translation flow required. **0** Stall The SMMU stall fault flow can be used for this request, when it is enabled. **1** ATST The transaction has already been translated by ATS. **2** NoStall If a translation fault occurs, then even if the SMMU has enabled stall faulting for this translation context, a fault response is returned without dependence on software activity. **3** PRI If a translation fault occurs, a fault response is returned indicating that a PRI request might resolve the fault. Architecturally the request is treated as an ATS request, and translation faults do not result in an event record. This option is for use by PCIe enumerated endpoints. If this field is not Stall, then the slave must return the response in reasonable time, without dependence on software activity. It must not be Stall for PCIe masters. Width is 2-bit.
LASECSID	Context	StreamID security level **0** Non-secure StreamID **1** Secure StreamID When LAFLOW is ATST, the LASECSID signal must be 0. Width is 1-bit.
LASECSID	Context	StreamID security level **0** on-secure StreamID **1** Secure StreamID When LAFLOW is ATST, the LASECSID signal must be 0. Width is 1-bit.
LASID	Context	StreamID. Width is `LTI_SID_WIDTH`-bit
LASSIDV	Context	SubstreamID valid. When LAFLOW is ATST, the LASSIDV signal must be 0. Width is (`LTI_SSID_WIDTH` > 0 ? 1 : 0)-bit
LASSID	Context	SubstreamID. When LASSIDV is LOW, the LASSID signal must be 0. Width is `LTI_SSID_WIDTH`-bit.

Signal	Category	Description
LAPROT	Transaction	Protection information. LAPROT uses the same encoding as AXI AxPROT. LAPROT[0]: PnU **0** Unprivileged access **1** Privileged access LAPROT[1]: NS **0** Secure access **1** Non-secure access LAPROT[2]: InD **0** Data access **1** Instruction access If LATRANS is SPEC, LAPROT[0] must be 0. If LASECSID=0, LAPROT[1] must be 1. If LATRANS is W, RW, SPEC, W-CMO, DCP, or W-DCP, LAPROT[2] must be 0. If LAFLOW is ATST, LAPROT[0] must be 0. If LAFLOW is ATST, LAPROT[2] must be 0. Width is 3-bit.
LAADDR	Transaction	Address. This signal is always 64 bits because virtual addresses are always 64 bits, even if the system address bus is narrower. LAADDR[11:0] does not affect the translation result, but is used to provide information to software when a translation fault occurs. Width is 64-bit.
LATRANS	Transaction	Type of the transaction that the LTI request is translating. Width is 4-bit.
LAATTR	Transaction	Attributes for the untranslated transaction. Width is 4-bit.
LALOOP	Impdef	**IMPLEMENTATION DEFINED** loopback signaling. Width is `LTI_LOOP_WIDTH`-bit.

Signal	Category	Description
LATLBLOC	Impdef	Location to access in the TLB. The meaning of the LATLBLOC signal is **IMPLEMENTATION DEFINED**. The intended use of the LATLBLOC signal is to control allocation and lookup in a TLB. For example, an implementation might guarantee that if a request is made with a particular value of LATLBLOC, and this request is followed by another request with the same value of LATLBLOC, then any translation that is cached from the first request is available to be used by the second request. Alternatively, LATLBLOC might be used to indicate a portion of a TLB to use. The LTI specification does not provide any guarantees about any such functionality. Width is `LTI_TLBLOC_WIDTH`-bit.
LAUSER	Impdef	**IMPLEMENTATION DEFINED** additional signaling. Width is `LTI_LAUSER_WIDTH` -bit.

The following table shows the LTI response channel signals.

Table A-22: LTI response channel signals

Signal	Category	Description
LRVALID	Transport	Channel valid. When this signal is LOW, other TX signals on the LR channel are not valid. Width is 1-bit.
LRVC	Transport	Virtual Channel number. Width is $clog_2$(`LTI_VC_COUNT`)-bit.
LRCREDIT	Transport	Channel credit grant. LRCREDIT is an RX signal that flows in the other direction from other LR channel signals. It is not affected by LRVALID. Width is `LTI_VC_COUNT`-bit.
LRID	Flow	Translation request ID. The value of LRID must match a translation request that has not yet had a response. Width is `LTI_ID_WIDTH`-bit.
LRCTAG	Flow	Translation completion tag. LRCTAG can be any value and must be returned by the LTI master with the completion. Width is 1-bit.

Signal	Category	Description
LRRESP	Translation	Translation response: **0: Success** The translation was successful. **1: Downgrade1** The translation was successful but the transaction type must be downgraded. **2: Downgrade2** The translation was successful but the transaction type must be downgraded. **4: FaultAbort** The translation was not successful and the transaction must be terminated. The master should indicate to the upstream device that the transaction was not successful. **5: FaultRAZWI** The translation was not successful and the transaction must be terminated. If possible, the LTI master should indicate to the requestor that the transaction was successful, by returning zero if the data was a read, and ignoring the transaction if it was a write. Cache maintenance and prefetch effects of the transaction are ignored. **6: FaultPRI** The translation was not successful but it might be resolved by issuing a PRI request. The master should issue a PRI request, and if the response from that indicates success, retry the LTI request. Width is 3-bit.
LRPROT	Translation	Translated protection information. LRPROT uses the same encoding as LAPROT. If LATRANS is SPEC, LRPROT[0] must be 0. If LASECSID=0, LAPROT[1] must be 1. If LATRANS is W, RW, SPEC, W-CMO, DCP, or W-DCP, LRPROT[2] must be 0. When LRRESP is FaultAbort, FaultRAZWI or FaultPRI, this signal is not valid. Width is 3-bit.
LRADDR	Translation	Translated address. The least significant 12 bits must equal the least significant 12 bits of LAADDR. These bits are included in the response to avoid them needing to be included in loopback and provided in the request. When LRRESP is FaultAbort, FaultRAZWI or FaultPRI, the LRADDR signal is not valid. Width is `LTI_LRADDR_WIDTH`-bit
LRATTR	Translation	Translated transaction attributes. LRATTR uses the same encoding as LAATTR. When LRRESP is FaultAbort, FaultRAZWI or FaultPRI, the LRATTR signal is not valid. Width is 4-bit.
LRHWATTR	Translation	**IMPLEMENTATION DEFINED** hardware attributes. When LRRESP is FaultAbort, FaultRAZWI or FaultPRI, this signal is not valid. Width is 4-bit.

Signal	Category	Description
LRMPAM	Translation	MPAM information. **LRMPAM[0]** MPAMNS **LRMPAM[9:1]** PARTID **LRMPAM[10]** PMG When LASECSID=0, the MPAMNS field must be 1. When LRRESP is FaultAbort, FaultRAZWI or FaultPRI, this signal is not valid. Width is 11-bit.
LRLOOP	Impdef	**IMPLEMENTATION DEFINED** loopback signaling. Must match the value of LALOOP in the request. Width is `LTI_LOOP_WIDTH`-bit.
LRUSER	Impdef	**IMPLEMENTATION DEFINED** additional signaling. Width is `LTI_LRUSER_WIDTH`-bit.

The following table shows the LTI completion channel signals.

Table A-23: LTI completion channel signals

Signal	Category	Description
LCVALID	Transport	Channel valid. When this signal is LOW, other TX signals on the LC channel are not valid. Width is 1-bit.
LCCREDIT	Transport	Channel credit grant. LCCREDIT is an RX signal which flows in the other direction from other LC channel signals. It is not affected by LCVALID. Width is 1-bit.
LCCTAG	Flow	Translation completion tag. LCCTAG must match the value that is given in LRCTAG. Width is 1-bit.
LCUSER	Impdef	**IMPLEMENTATION DEFINED** additional signaling. Width is `LTI_LCUSER_WIDTH`-bit.

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.

Signal definitions

Table A-24: TBU interrupt signals

Signal	Direction	Description
ras_fhi	Output	Fault handling RAS interrupt for a contained error. Width is 1-bit.
ras_eri	Output	Error recovery RAS interrupt for an uncontained error. Width is 1-bit.
ras_cri	Output	Critical error interrupt, for an uncontainable uncorrected error. Width is 1-bit.
pmu_irpt	Output	PMU interrupt. Width is 1-bit.

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.

Signal definitions

Table A-25: TBU tie-off signals

Signal	Direction	Description
ns_sid_high	Input	Provides the high-order StreamID bits for all transactions with a Non-secure StreamID that pass through the TBU The width of ns_sid_high is: (32 - `TBUCFG_SID_WIDTH`)-bit. See 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.
s_sid_high	Input	Provides the high-order StreamID bits for all transactions with a Secure StreamID that pass through the TBU The width of s_sid_high is (32 - `TBUCFG_SID_WIDTH`)-bit. See 3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83.
max_tok_trans	Input	Indicates the number of DTI translation tokens to request when connecting to the TCU, minus 1. **Note:** The TBU must request a minimum of two translation tokens per LTI port. However, we recommend one translation token per translation slot for most scenarios. The ACE-Lite TBU has a single internal LTI port, and the LTI TBU can have multiple LTI ports. The width of max_tok_trans is ($\log_2$`TBUCFG_XLATE_SLOTS`)-bit. See 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85.

Signal	Direction	Description
pcie_mode	Input	You must tie this signal HIGH when the TBU is connected to a PCIe interface. When this signal is HIGH, the TBU interprets the input AXI memory types as encoding PCI *No Snoop* information. For the TBU to provide correct operation, transactions from the PCIe interface must be delivered to the TBU with the following AXI memory types: **Normal Non-cacheable Bufferable** When *No Snoop* is set for the transaction **Write-Back** When *No Snoop* is not set for the transaction 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. 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. Width is 1-bit.
sec_override	Input	When HIGH, some registers are accessible to Non-secure accesses from reset, as the 4.13.3 TBU_SCR register on page 163 settings describe. Width is 1-bit.
ecorevnum[3:0]	Input	Tie this signal to 0 unless we recommend otherwise. Width is 4-bit.
utlb_roundrobin	Input	Defines the MicroTLB entry replacement policy. When LOW, the MicroTLB uses a *Pseudo Least Recently Used* (PLRU) replacement policy. This policy typically provides the best average performance. When HIGH, the MicroTLB uses a round-robin replacement policy. With this policy, the oldest entry is evicted when the MicroTLB is full. 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. Width is 1-bit.

Signal	Direction	Description
poison_support	Input	**Note:** poison_support applies only to the ACE-Lite TBU. Determines how the ACE-Lite TBU handles RAS errors in the write data buffer. 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. 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. Width is 1-bit.

A.2.11 TBU ELA debug signals

The MMU-700 TBU includes *Embedded Logic Analyzer* (ELA) debug signals.

Signal definitions

Table A-26: ELA enable signals

Signal	Direction	Description
ela_enable	Input	ela_enable is an asynchronous input port. When `TBUCFG_USE_ELA_DEBUG` is 0, the SMMU ignores the value of the signal. When `TBUCFG_USE_ELA_DEBUG` 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. Width is 1-bit.

A.2.12 Integration TBU signals

The Integration TBU signals are the same as the signals for the ACE-Lite TBU except for the differences that the following subsections describe.

The signal groups that differ for the Integration TBU when compared to the ACE-Lite TBU are as follows:

A.2.12.1 Integration TBU TBM interface signals on page 232
Differences compared to the ACE-Lite TBU signals

A.2.12.2 Integration TBU tie-off signals on page 232
Differences compared to the ACE-Lite TBU signals

A.2.12.3 ELA interface on page 233
Differences compared to the ACE-Lite TBU signals

A.2.8 TBU LTI interface signals on page 223

Not applicable because the Integration TBU is based on the ACE-lite TBU

For the ACE-Lite TBU signal descriptions, see A.2 TBU signals on page 206.

A.2.12.1 Integration TBU TBM interface signals

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.

The following table shows the interface directions and the agents.

Table A-27: Interface directions and agents

Interface direction	Agent
Producer	Integration TBU Splitter
Consumer	Downstream memory component

Several ACE-Lite properties are supported as 3.3.2 AMBA implementation on page 64 describes. The following are minor differences to the port listing when compared to the standard TBU TMB interface signals:

- AWID is (`TBUCFG_ID_WIDTH` + 1) bits wide
- ARID is (`TBUCFG_ID_WIDTH` + 1) bits wide
- BID is (`TBUCFG_ID_WIDTH` + 1) bits wide
- RID is (`TBUCFG_ID_WIDTH` + 1) bits wide

For the standard TBU TBM interface signals, see A.2.3 TBU TBM interface signals on page 213.

See the *AMBA® AXI and ACE Protocol Specification*.

A.2.12.2 Integration TBU tie-off signals

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 provide the configuration state.

The following table shows the interface directions and the agents.

Table A-28: 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 Integration TBU tie-off signals.

Table A-29: Integration TBU tie-off signals

Signal	Direction	Corresponding R-TBU/W-TBU signal name
ns_sid_high_r	Input	ns_sid_high Width is 31 - `TBUCFG_SID_WIDTH`
ns_sid_high_w	Input	ns_sid_high Width is 31 - `TBUCFG_SID_WIDTH`
s_sid_high_r	Input	s_sid_high Width is 31 - `TBUCFG_SID_WIDTH`
s_sid_high_w	Input	s_sid_high Width is 31 - `TBUCFG_SID_WIDTH`
max_tok_trans_r	Input	max_tok_trans Width is $\log_2$(`TBUCFG_XLATE_SLOTS_R`)
max_tok_trans_w	Input	max_tok_trans Width is $\log_2$(`TBUCFG_XLATE_SLOTS_W`)
sec_override	Input	sec_override Width is 1-bit
ecorevnum	Input	ecorevnum Width is 4-bit
utlb_roundrobin_r	Input	utlb_roundrobin Width is 1-bit
utlb_roundrobin_w	Input	utlb_roundrobin Width is 1-bit
pcie_mode	Input	pcie_mode Width is 1-bit
poison_support	Input	poison_support Width is 1-bit

See A.2.10 TBU tie-off signals on page 229.

A.2.12.3 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 A-30: Interface directions and agents

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

The following table shows the ELA interface signals.

Table A-31: ELA interface signals

Name	Direction	Width
ela_enable_wtbu	Input	1
signalgrp0_wtbu	Output	128
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

Name	Direction	Width
sigqual3_rtbu	Output	4
sigclken3_rtbu	Output	1
signalgrp4_rtbu	Output	128
sigqual4_rtbu	Output	4
sigclken4_rtbu	Output	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.

Signal definitions

Table A-32: Test and debug signals

Signal	Direction	Description
dftcgen	Input	Clock gate enable. To enable architectural clock gates for the clock, clk, set this signal HIGH during scan shift. Width is 1-bit.
dftrstdisable	Input	Reset disable. To disable reset, set this signal HIGH during scan shift. Width is 1-bit.
dftramhold	Input	Preserve RAM state. To preserve the state of the RAMs and their connected registers, set this signal HIGH during scan shift. Width is 1-bit.
MBISTRESETN	Input	MBIST mode reset. This active-LOW signal is encoded as follows: **0** Reset MBIST functional logic. **1** Normal operation. To prevent unintended reset of the functional logic, keep the MBISTRESETN signal in the inactive state, HIGH, during scan testing. Width is 1-bit.

Signal	Direction	Description
MBISTREQ	Input	MBIST test request. This signal is encoded as follows: **0** Normal operation. **1** Enable MBIST testing. Width is 1-bit.

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_S*n* slave downstream interface per slave interface. The following table shows the DN_S*n* signals.

In the following table, the 'Direction' has the following meaning:

Input

Slave to master

Output

Master to slave

Signal definitions

Table A-33: DTI interconnect switch DN_S*n* interface signals

Signal	Direction	Description
tvalid_dti_dn_s*n*	Input	Flow control signal. Width is 1-bit.
tready_dti_dn_s*n*	Output	Flow control signal. Width is 1-bit.
tdata_dti_dn_s*n*	Input	Message data signal The width of tdata_dti_dn_s is `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.
tid_dti_dn_s*n*	Input	Indicates the master that initiated the message The width of tid_dti_dn_s is `ID_WIDTH` = $\log_2$(total number of masters being switched)-bit.
tlast_dti_dn_s*n*	Input	Indicates the last cycle of a message. Width is 1-bit.

Signal	Direction	Description
tkeep_dti_dn_*sn*	Input	Indicates valid bytes The width of tkeep_dti_dn_s is (`DATA_WIDTH` / 8)-bit.
twakeup_dti_dn_*sn*	Input	Wakeup signal. Width is 1-bit.

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.

Signal definitions

Table A-34: DTI interconnect sizer LPI_CG interface signals

Signal	Direction	Description
qactive_cg	Output	Component active. Width is 1-bit.
qreqn_cg	Input	Quiescence request. Width is 1-bit.
qacceptn_cg	Output	Quiescence accept. Width is 1-bit.
qdeny_cg	Output	Quiescence deny. Width is 1-bit.

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.

Signal definitions

Table A-35: DTI interconnect register slice LPI_CG interface signals

Signal	Direction	Description
qactive_cg	Output	Component active. Width is 1-bit.
qreqn_cg	Input	Quiescence request. Width is 1-bit.

Signal	Direction	Description
qacceptn_cg	Output	Quiescence accept. Width is 1-bit.
qdeny_cg	Output	Quiescence deny. Width is 1-bit.

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	TCUCFG_QTW_DATA_WIDTH <= 128	0, 1, 2, 3, 4, 5, 6, 10	8
	TCUCFG_QTW_DATA_WIDTH == 256	0, 1, 2, 3, 4, 5, 6, 7, 10, 11	10
	TCUCFG_QTW_DATA_WIDTH == 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>	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>	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]	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		
	[111]	bvalid_qtw		

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n>	Number of cycles of delay
	[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
10	[127:0]	wdata_qtw_demuxed[8][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

[8] When the `TCUCFG_QTW_DATA_WIDTH` parameter is set to 512, the wdata_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 3.4 Configuration parameters and methodology on page 80.

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		
	[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		

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
1	[127:118]	Unused	-	-
	[117:107]	awmpam_m	`1'b0`, (awready_m AND awvalid_m), awready_m, awvalid_m	2
	[106:83]	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	[127:117]	Unused	-	-
	[116]	archunken_m	`1'b0`, (arready_m AND arvalid_m), arready_m, arvalid_m	2
	[115:105]	armpam_m		
	[104:95]	arloop_m		
	[94]	aridunq_m		
	[93:70]	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		

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
	[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.

Note

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 `TBUCFG_LTI_LOOP_WIDTH` <= 128 bits	`0, 1, 2, 3, 5, 7, 8`	7
	When `TBUCFG_LTI_LOOP_WIDTH` > 128 bit	`0, 1, 2, 3, 4, 5, 6, 7, 8`	9

The following table shows the SIGNALGRP<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	[127:126]	Unused	-	-
	[125:110]	latlbloc	`3'b000`, lavalid	1
	[109:78]	laid		
	[77:76]	laflow		
	[75:72]	laattr		
	[71:68]	latrans		
	[67:65]	laprot		
	[64:1]	laaddr		
	[0]	lavalid		
1	[127:123]	laog	`3'b000`, lavalid	1
	[122]	laogv		
	[121:102]	lassid		
	[101]	lassidv		
	[100:77]	lasid		
	[76]	lasecsid		
	[75:72]	lavc		
	[71:68]	latrans		

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
	[67:65]	laprot		
	[64:1]	laaddr		
	[0]	lavalid		
2	[127:115]	Unused	-	-
	[114:104]	lrmpam	`3'b000`, lrvalid	1
	[103:100]	lrhwattr		
	[99:96]	lrattr		
	[95:44]	lraddr		
	[43:41]	lrprot		
	[40:38]	lrresp		
	[37]	lrctag		
	[36:5]	lrid		
	[4:1]	lrvc		
	[0]	lrvalid		
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`, lrvalid	1
6	[127:0]	lrloop[255:128]	`3'b000`, lrvalid	2
7	[127:16]	Unused	-	-
	[15]	lcctag_7	`3'b000`, OR(lcvalid[7:0])	0
	[14]	lcvalid_7		
	[13]	lcctag_6		
	[12]	lcvalid_6		
	[11]	lcctag_5		
	[10]	lcvalid_5		
	[9]	lcctag_4		
	[8]	lcvalid_4		
	[7]	lcctag_3		
	[6]	lcvalid_3		
	[5]	lcctag_2		
	[4]	lcvalid_2		
	[3]	lcctag_1		
	[2]	lcvalid_1		
	[1]	lcctag_0		
	[0]	lcvalid_0		
8	[127:32]	Unused	-	-
	[31]	lmaskclose_7	`1'b0`, OR(lmaskclose[7:0]),OR(lmopenack[7:0]), OR(lmopenreq[7:0])	0
	[30]	lmactive_7		
	[29]	lmopenack_7		
	[28]	lmopenreq_7		

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
	[27]	lmaskclose_6		
	[26]	lmactive_6		
	[25]	lmopenack_6		
	[24]	lmopenreq_6		
	[23]	lmaskclose_5		
	[22]	lmactive_5		
	[21]	lmopenack_5		
	[20]	lmopenreq_5		
	[19]	lmaskclose_4		
	[18]	lmactive_4		
	[17]	lmopenack_4		
	[16]	lmopenreq_4		
	[15]	lmaskclose_3		
	[14]	lmactive_3		
	[13]	lmopenack_3		
	[12]	lmopenreq_3		
	[11]	lmaskclose_2		
	[10]	lmactive_2		
	[9]	lmopenack_2		
	[8]	lmopenreq_2		
	[7]	lmaskclose_1		
	[6]	lmactive_1		
	[5]	lmopenack_1		
	[4]	lmopenreq_1		
	[3]	lmaskclose_0		
	[2]	lmactive_0		
	[1]	lmopenack_0		
	[0]	lmopenreq_0		
9	[127:0]	Unused	-	-
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.

Note

This section does not describe how to create translation tables. For more information, see the *Arm® Architecture Reference Manual for A-profile architecture*.

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.

   Note

   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.

Note

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.

Note

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.

Note

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.COHACC = 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. When powered on, the MMU-700 invalidates TLBs and configuration cache structures automatically.

It might be necessary to invalidate TLBs and configuration caches manually. Secure software can also 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`

Note

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 4.6.7 TCU_SCR register on page 119.

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 TTB0 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 ORO	Cacheability attribute to use for translation table walks to TTB0: **00** Non-cacheable. **01** Write-Back Cacheable, Read-Allocate Write-Allocate. **10** Write-through Cacheable, Read-Allocate.
SH0	Shareability of translation table walks to TTB0: **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:

1. Allocate 64 bytes of memory for the STE.
2. 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.

3. 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 *intermediate physical address* (IPA), otherwise it is an untranslated *physical address* 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.S2IR0 and STE.S2OR0	`0b00`: Non-cacheable.
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_CR0ACK until SMMUEN reads as 1.

Appendix D 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
Improved descriptions	Throughout the document

Table D-3: Differences between issue 0000-02 and issue 0001-03

Change	Location
Improved descriptions	Throughout the document
Added new parameters	• 3.4.2 Translation Control Unit buffer configuration parameters on page 81 • 3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85
Added system discovery registers	• 4.8 TCU system discovery registers on page 130 • 4.15 TBU system discovery registers on page 173

Table D-4: Differences between issue 0001-03 and issue 0001-04

Change	Location
Improved descriptions	Throughout the document
Added new *Width* column added to all signal description tables	A. Signal descriptions for MMU-700 on page 193

Table D-5: Differences between issue 0001-04 and issue 0100-05

Change	Location
Improved descriptions	Throughout the document
Updated LTI TBU description	3.1.2.3 LTI TBU LTI interface on page 33
Added new *Integration TBU* section	3.1.2.10 Integration TBU on page 35
Updated parameters descriptions	3.4 Configuration parameters and methodology on page 80
Updated register descriptions	4. Programmers model for MMU-700 on page 94

Table D-6: Differences between issue 0100-05 and issue 0100-06

Change	Location
Improved descriptions	Throughout the document
Updated parameter descriptions	3.4 Configuration parameters and methodology on page 80

Table D-7: Differences between issue 0100-06 and issue 0102-07

Change	Location
Improved descriptions	Throughout the document
Updated the exact titles and version of some referenced documents	In 'Useful resources' and throughout the document where referenced
Updated the 'Support for flexible integration' section	2.2 Features on page 17
Updated the 'DTI interface' description	3.2 Operation on page 40
Updated the 'Walk cache' description	3.2 Operation on page 40
Updated the 'DTI interface' description	3.2 Operation on page 40
Improved the description	3.2 Operation on page 40
Updated the 'Sizer' description	3.2 Operation on page 40
Improved the description	3.1.1.5 TCU DTI interface on page 29
For the 'Integration TBU' section, separated out the configuration parameters and signals into the relevant sections instead of being within the 'Integration TBU' section	• 3.1.2.10 Integration TBU on page 35 • 3.4.8 Integration TBU configuration parameters on page 89 • A.2.12 Integration TBU signals on page 231
Improved the description, including changing 2^24 to 2^32 in multiple places	3.2.1 DTI overview on page 40
Improved the description	3.2.2.1 SMMUv3 architectural performance events on page 42
Updated the description of 'Buffered translation'	3.2.2.2 MMU-700 TCU events on page 43
Updated some of the descriptions	• 3.2.5 RAS implementation on page 49 • 3.2.8 TCU transaction handling on page 53 • 3.2.9 TCU prefetch on page 54
Replaced 'Armv8 memory attribute' with 'Armv8 memory type' in the column titles of the tables	3.2.11 Conversion between ACE-Lite and Armv8 attributes on page 56
Updated the description of 'Normal Inner Write-Back Outer Write-Back' in the 'Master interface memory type attribute handling' section	3.2.11 Conversion between ACE-Lite and Armv8 attributes on page 56
Added a new section for the AXI USER bits of the TCU QTW/DVM interface	3.2.12 AXI USER bits that MMU-700 TBU TBM and TCU QTW/DVM define on page 59
Changed the value of SMMU_IDR1.SIDSIZE and SMMU_S_IDR1.S_SIDSIZE from 24 to 32	3.3.1.1 ID register architectural options on page 61
Updated and reordered the list of features in the table	3.3.2.1 ACE-Lite feature support on page 64
Corrected the merging of table cells	3.3.3.1 TCU MPAM on page 73
Added a new table specifically for TBU MPAM instead of referring to the similar TCU MPAM table	3.3.3.2 TBU MPAM on page 76
Updated the descriptions for the `LTI_LAUSER_WIDTH`, `LTI_LRUSER_WIDTH`, and `LTI_LCUSER_WIDTH` parameters	3.3.4 Local Translation Interface implementation on page 79
Added a new `TCUCFG_DVM_VAS` parameter	3.4.1 Translation Control Unit I/O configuration parameters on page 80
Changed the value range of the `TBUCFG_LTI_OG_WIDTH` parameter to 1-4	3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83
Removed the value of 1024 for the `TBUCFG_XLATE_SLOTS` parameter	3.4.5 Common ACE-Lite and Local Translation Interface Translation Buffer Unit buffer configuration parameters on page 85

Change	Location
Merged cells in the 'TCU PMCG, RAS, and MPAM register allocation to regions of TCU address space' table	4.2.2 TCU memory map on page 100
Corrected the address offset of SMMU_PMCG_PMAUTHSTATUS	4.3.2 TCU and TBU PMU identification register summary on page 102
Changed the mask from 24 bits to 32 bits because the TCU uses 32-bit StreamIDs	4.5 TCU PMU registers on page 108
Corrected the address offset of SMMU_PMCG_PMAUTHSTATUS	4.5.5 PMU ID registers on page 110
Corrected the address offset of TCU_WC_S2L3_CMAX	4.6.8 TCU_WC_SxLy_CMAX registers on page 120
Corrected the reset value	4.7.2 TCU_ERRCTLR register on page 123
Updated the description of some bit fields	• 4.7.3 TCU_ERRSTATUS register on page 123 • 4.10.1 ITOP register for the TCU Translation Management Unit on page 152
Updated the description of the 'pmu_snapshot_req' bit	4.10.2 ITIN register for the TCU Translation Management Unit on page 153
Added more registers to the Non-secure access category	4.13 TBU microarchitectural registers on page 158
Updated the description	4.13.3 TBU_SCR register on page 163
Updated the description of some bit fields	4.16.2 ITOP_TBU register on page 190
Updated the description of the 'pmu_snapshot_req' bit	4.16.3 ITIN_TBU register on page 191
Added some signals to the signal list table	A.1.2 TCU QTW/DVM interface signals on page 193
Corrected the width of the tdata_dti_dn signal	A.2.7 TBU DTI interface signals on page 221
Updated the descriptions of the ras_fhi, ras_eri, ras_cri, and pmu_irpt signals	A.1.9 TCU interrupt signals on page 200
Updated the descriptions of some signals	• A.2.2 TBU TBS interface signals on page 206 • A.2.3 TBU TBM interface signals on page 213
Added the list of LTI signals to the section	A.2.8 TBU LTI interface signals on page 223
Updated the description of the max_tok_trans signal	A.2.10 TBU tie-off signals on page 229
Clarified the descriptive text	C.1.6 Invalidating TLBs and configuration caches on page 250

Table D-8: Differences between issue 0102-07 and issue 0102-08

Change	Location
Changed the permitted range of the `TBUCFG_LTI_OG_WIDTH` parameter from 1-4 to 1-5 for certain conditions	3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83
Added a new table named *Legal combinations of Number of LTI Ports, TBUCFG_LTI_OG_WIDTH and TBUCFG_SLOTRAM_TYPE parameters*	3.4.4 Common ACE-Lite and Local Translation Interface Translation Buffer Unit configuration parameters on page 83